QR Based Fuel Bill Payment System

Submitted By
Priyanka Sikder
ID: 2014-1-60-054
Syeda Faria Sultana
2014-1-60-058
Suma Rani Dey
ID: 2014-1-60-106

Supervised By
MD. Nawab Yousuf Ali, PhD
Associate Professor
Department of Computer Science and Engineering
East West University

A Project Submitted in Partial Fulfillment of the Requirements for the
Degree of Bachelors of Science in Computer Science and Engineering
To the

Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Declaration

We hereby declare that, this project was done under CSE497 and under the supervisor MD. Nawab Yousuf Ali and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication.

_____

Priyanka Sikder
ID: 2014-1-60-054
Department of Computer Science and Engineering
East West University

_____

Syeda Faria Sultana
ID: 2014-1-60-058
Department of Computer Science and Engineering
East West University

_____

Suma Rani Dey
ID: 2014-1-60-106
Department of Computer Science and Engineering
East West University

Abstract

The aim of this project is to implement fuel bill payment application for android based mobile phones. We implemented this by using different services and sensors like- camera, SMS, telephone, Location and so on. We focus on how we can save time while pay the bill for fuel and also solve the problem of fractional money and we can see the traffic in fuel station. It can capture QR code, send money to the station account. We can add money to our own account, we can see how far we can drive with our current fuel & also we can see how much fuel we can buy from a specific amount of money or how much money we need to buy specific amount of fuel. We can also see the history of our expense. We will start with the basic theoretical framework of the mobile application that is Android Studio, PHP with CodeIgniter framework, JSON and API. Then we describe the design of each module and analysis, implementation levels of the project. Finally, we will make a conclusion of this project involving the future development. In this project every stage of learning and exploring was challenging for us where it allows us to have better image of the application production procedure.

Letter of Acceptance

We hereby declare that this project is from the student's own work and best effort of our, and all other source of information used have been acknowledged. This project has been submitted with our approval.

_____

MD. Nawab Yousuf Ali, PhD
Associate Professor
Department of Computer Science and Engineering
East West University

_____

Dr. Wasif Ahmed Reza
Chairperson & Associate Professor
Department of Computer Science and Engineering
East West University

Acknowledgement

First of all, we would like to thank almighty God for giving us the strength & proper knowledge to complete our project work. We would like to express our deep sense of gratitude and sincere thanks to our honorable supervisor MD. Nawab Yousuf Ali, Associate Professor, Department of Computer Science and Engineering, East West University, Aftabnagar, Dhaka, Bangladesh for his kind guidance, sharing knowledge and constant inspiration throughout this project work. Our sincere gratefulness for the faculty of Computer Science and Engineering whose friendly attitude and enthusiastic support that has given us for more than four years. We are very grateful for the motivation and stimulation from our friends and seniors. Finally, our most heartfelt gratitude goes to our beloved parents and brother for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to the end.

TABLE OF CONTENTS

LIST OF FIGURES .....

LIST OF FIOWCHART .....

LIST OF CODE.....

Introduction

The main purpose of this application is how we can save time while pay the bill for fuel through QR code scan and also solve the problem of fractional money.

1.1 Motivation

Now-a-days traffic jam is a big issue in our country. Because of traffic jam lots of time is wasted. Even now-a-days a long queue is seen in fuel filling station. As, people spend lots of time in traffic jam they don't want to spend time in filling station. Payment is one of the reason of long queue in fuel filling station. If there is any easy way to pay the bill then jam will be less. In foreign country QR code scanning is very popular. They scan QR code and pay the bill. This technic is used in every fields like- restaurant, hotel, hospital, supershop etc in foreign country. People use this technic to save the time. We like to introduce this technic in our country. That's why this idea come to our mind. We specially use filling station as we like to decrease the traffic jam. This application which has been developed in android platform and PHP.

1.2 Objective of the project

The main objective is to develop an android application which can be run in smartphone only
in android platform devices. For developing this application, we will able to

- Explain how the application is designed.
- Explain how the application is built.
- Explain how all of our functions work.
- Explain how the transaction is occurred.
- Explain how to find station and get direction.
- Explain how this application is able to save time.

1.3 Methodology of project

In this application we focused on two things one is QR code scanning and pay the bill and other one is solving the fractional money problem. You open an account and insert money in your account. After buying fuel you scan the QR code and it brings you the specific fuel filling station and you pay the bill. You can search filling station and can also see the direction from your current location. The history of expense of today, week, month and year.

1.4 Results of the project

After a lot of R&D (research and development) we successfully implemented those features and built our own application. We did a lot of Test to assure its quality and successfully this application is running smoothly in any version of android device.

Background Study

2.1 Android

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers, developed by Google in conjunction with the Open Handset Alliance. Initially developed by Android Inc., whom Google financially backed and later purchased in 2005, Google releases the Android code as open source, under the Apache License. Therefore, different Android-based phones often have different graphical user interfaces GUIs even though they use the same OS. The Android Open Source Project (AOSP), led by Google, is tasked with the maintenance and further development of Android. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of java[23] .

2.1.1 Android Overview

The first-generation Android phones were released in October 2008. According to Gartner, North American sales of Android-based phones increased 707% in the first quarter of 2010 over the previous year.1 By March 2011, a Nielsen study showed that Android had 37% of the U.S. smartphone market share, compared to 27% for Apple's iPhone and 22% for Blackberry.2 In August 2010, more than 200,000 Android smartphones were being activated each day, up from 100,000 per day only two months earlier.3 As of June 2011, more than 500,000 Android devices were being activated daily[24]. There are now over 300 different Android devices worldwide. The Android operating system was developed by Android, Inc., which was acquired by Google in July 2005. Android is used in numerous smartphones, e-reader devices and tablet computers.

2.1.2 Openness and Open Source

One benefit of developing Android apps is the openness of the platform. The operating system is open source and free. This allows you to view Android's source code and see how its features are implemented. You can also contribute to Android by reporting bugs

(see source.android.com/source/report-bugs.html) or by participating in the Open Source Project discussion groups (source.android.com/community/index.html). Numerous open-source Android apps from Google and others are available on the Internet.

Java Android apps are developed with Java—the world's most widely used programming language. Java was a logical choice for the Android platform, because it's powerful, free and open source. Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices (e.g., cell phones, pagers and personal digital assistants) and for many other purposes. Java enables you to develop apps that will run on a variety of devices without any plat- form-specific code. Experienced Java programmers can quickly dive into Android development, using the Android APIs (Application Programming Interfaces) and others available from third parties. The openness of the platform spurs rapid innovation. Android is available on devices from dozens of original equipment manufacturers (OEMs) in 48 countries through 59 carriers[16]. The intense competition among OEMs and carriers benefits customers. Java is object oriented and has access to powerful class libraries that help you develop apps quickly. GUI programming in Java is event driven—in this book, you'll write apps that respond to various user-initiated events such as screen touches and keystrokes. In addition to directly programming portions of your apps, you'll also use Eclipse to conveniently drag and drop predefined objects such as buttons and textboxes into place on your screen, and label and resize them. Using Eclipse with the Android Development Tools (ADT) Plug-in, you can create, run, test and debug Android apps quickly and conveniently, and you can visually design your user interfaces.

2.1.3 Storage

You can package data files with your application, for things that do not change, such as icons or help files[19]. You also can carve out a small bit of space on the device itself, for databases or files containing user-entered or retrieved data needed by your application. And, if the user supplies bulk storage, like an SD card, you can read and write files on there as needed.

2.1.4 Network

Android devices will generally be Internet-ready, through one communications medium or another. You can take advantage of the Internet access at any level you wish, from raw

Java sockets all the way up to a built-in WebKit-based Web browser widget you can embed in your application.

2.1.5 Multimedia

Android devices have the ability to play back and record audio and video. While the specifics may vary from device to device, you can query the device to learn its capabilities and then take advantage of the multimedia capabilities as you see fit, whether that is to play back music, take pictures with the camera, or use the microphone for audio note-taking.

2.1.6 GPS

Android devices will frequently have access to location providers, such as GPS, that can tell your applications where the device is on the face of the Earth. In turn, you can display maps or otherwise take advantage of the location data, such as tracking a device's movements if the device has been stolen.

2.1.7 Sensor

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device. You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks.

2.1.8 What Androids Are Made Of :

When you write a desktop application, you are "master of your own domain". You launch your main window and any child windows – like dialog boxes – that are needed. From your standpoint, you are your own world, leveraging features supported by the operating system, but largely ignorant of any other program that may be running on the computer at the same time. If you do interact with other programs, it is typically through an API to communicate with MySQL or another database. Android has similar concepts, but packaged differently, and structured to make phones more crash-resistant.[25]

2.1.9 Activities

The building block of the user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog in a desktop application, or the page in a classic Web app. Activity represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application. Android is designed to support lots of cheap activities, so you can allow users to keep clicking to bring up new activities and tapping the BACK button to back up, just like they do in a Web browser. For your app to be able to use activities, you must declare the activities, and certain of their attributes, in the manifest

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

2.1.10 Services

Activities are short-lived and can be shut down at any time. Services, on the other hand, are designed to keep running, if needed, independent of any activity. A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. You might use a service for checking for updates to an RSS feed, or to play back music even if the controlling activity is no longer operating. By default, a service runs in the same process as the main thread of the application. A commonly used pattern for a service

implementation is to create and run a new Thread in the service to perform the processing in the background and then to terminate the service once it has finished the processing.

2.1.11 Content Providers

Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications. The Android development model encourages you to make your own data available to other applications, as well as your own – building a content provider lets you do that, while maintaining complete control over how your data gets accessed

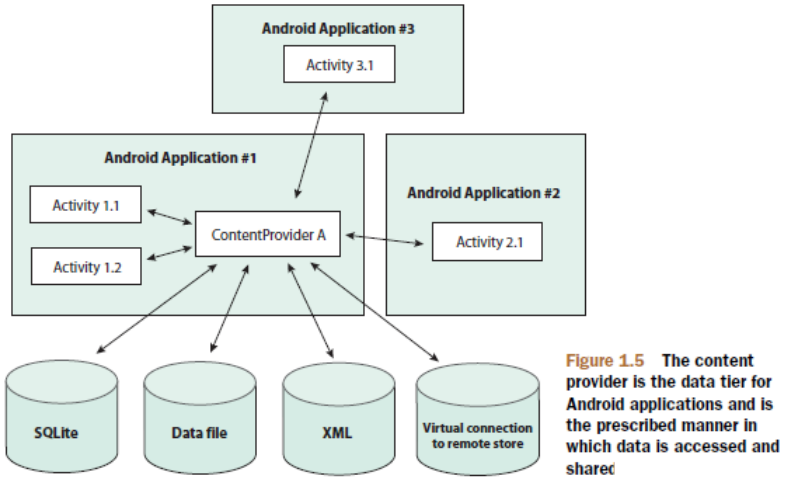


Figure 2.1: Content Provider

2.1.12 Intents

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways. It is a system messages, running around the inside of the device, notifying applications of various events, from hardware state changes, to incoming data, to application events. Not only can you respond to an Intent, but you can create your own, to launch other activities, or to let you know when specific situations arise.

```
Intent intent = new Intent(getBaseContext(), FuelCalculatorActivity.class);

startActivity(intent);
```

The Android SDK is a freely available download from the Android website. The first thing you should do before going any further in this chapter is make sure you have the Android SDK installed, along with Eclipse and the Android plug-in for Eclipse, also known as the *Android Development Tools*, or simply as the *ADT*. The Android SDK is required to build Android applications, and Eclipse is the preferred development environment for this book. The Android download page has instructions for installing the SDK, or you can refer to appendix A of this book for detailed information on installing the required development tools. As in any development environment, becoming familiar with the class structures is helpful, so having the documentation at hand as a reference is a good idea. The Android SDK includes HTML-based documentation, which primarily consists of Javadoc-formatted pages that describe the available packages and classes. The Android SDK documentation is in the /doc directory under your SDK installation. Because of the rapidly changing nature of this platform, you might want to keep an eye out for any changes to the SDK. Android's Java environment can be broken down into a handful of key sections. When you understand the contents in each of these sections, the Javadoc reference material that ships with the SDK becomes a real tool and not just a pile of seemingly unrelated material. You might recall that Android isn't a strictly Java ME software environment, but there's some commonality between the Android platforms and other Java development platforms. The next few sections review some of the Java packages (core and optional) in the Android SDK and where you can use them. The remaining chapters provide a deeper look into using many of these programming topics

2.1.13 Application Life Cycle

An application life cycle consists of the steps that the application's processes must follow from execution to termination. Every application, regardless of the language it was written in, has a specific life cycle, and Android applications are no exception. This section examines the life cycle of an ASP application and compares that to an Android application's life cycle Standard ASP Application Life Cycle.[23]

The life cycle of a standard ASP application is similar enough to that of an Android application to make this a good comparison. ASP applications step through five distinct processes from launch to disposal. These steps are required to be implemented by all ASP applications, and really define what an ASP application is. The steps, in order, are

1. Application_Start
2. Event

3. HTTP Application.Init
4. Disposal

Application Start is called when the application is requested from the server. This process in turn leads into the Event processes. When all associated application modules have loaded, HTTPApplication.Init is called. The application executes its events, and when the user attempts to close it, Dispose is called. Dispose then passes processing on to the Application_End process, which closes the application. This is a fairly standard application life cycle. Most applications follow similar life cycles: an application is created, loaded, has events, and is destroyed. The following section demonstrates how this compares to the Android application life cycle.
The Android application life cycle is unique in that the system controls much of the life cycle of the application. All Android applications, or Activities, are run within their own process. All of the running processes are watched by Android and, depending on how the activity is running (this is, a foreground activity, background activity, and so forth), Android may choose to end the activity to reclaim needed resources.

When deciding whether an activity should be shut down, Android takes into account several factors, such as user input, memory usage, and processing time.

Some of the specific methods called during the life cycle of an android activity are:

- onCreate
- onStart
- Process-specific
- onStop
- onDestroy

Flow chart of android process:

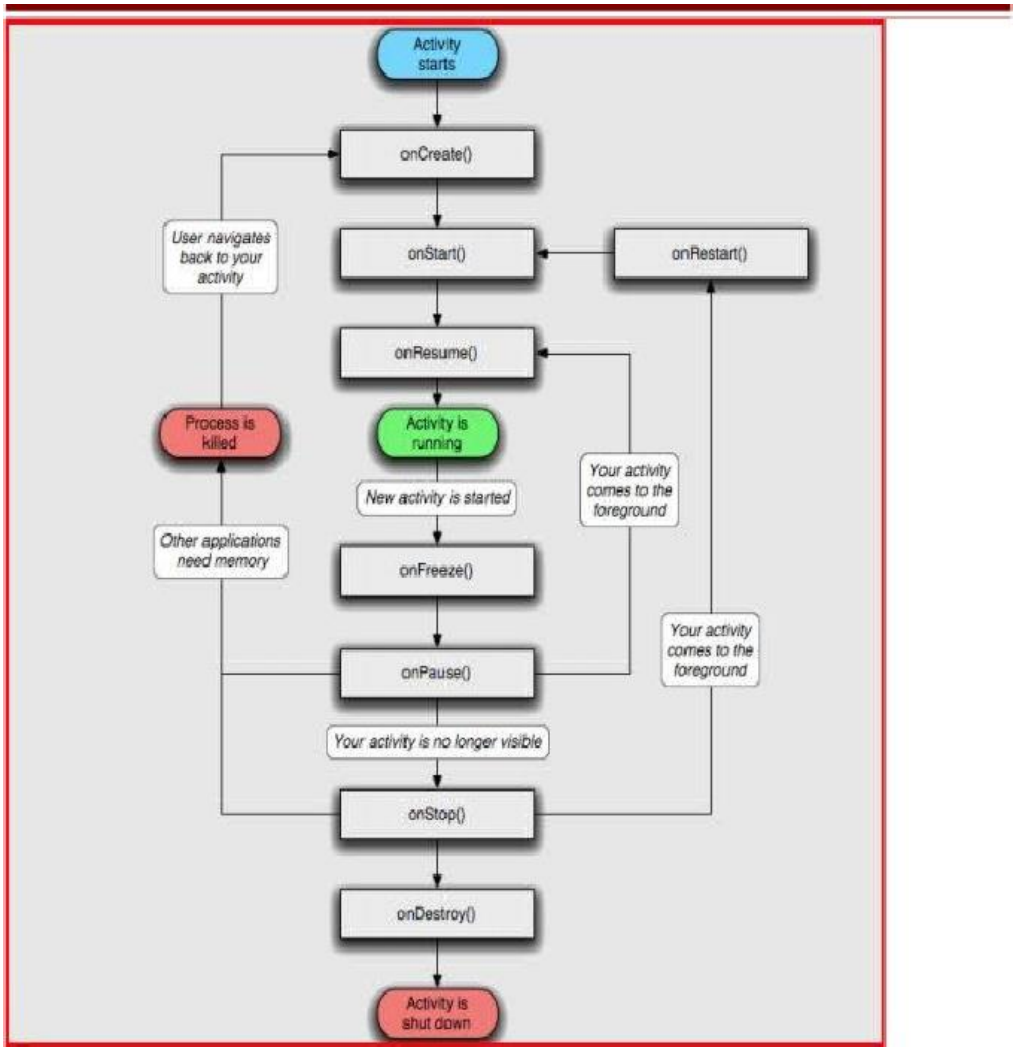


Figure 2.2: Android Lifecycle

Following the same logic as other application life cycles, an Android application is created, the processes are started, events are fired, processes are stopped, and the application is destroyed. Though there are a few differences, many application developers should be comfortable with the steps in the life cycle.

2.2 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. It was created by Rasmus Lerdorf in 1994. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Preprocessor. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks[26]. PHP is a powerful and widely-used open source server-side scripting language for generating web pages. PHP scripts are executed on the server and the result is sent to the browser as plain HTML. PHP can be integrated with the number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

2.2.1 History

PHP development began in 1995 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C, which he used to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could be used to build simple, dynamic web applications. To accelerate bug reporting and improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the Usenet discussion group comp.infosystems.www.authoring.cgi on June 8, 1995.This release already had the basic functionality that PHP has as of 2013. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax resembled that of Perl but was simpler, more limited and less consistent [26].

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it, there was never any intent to write a programming language [...] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way. A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997.

The fact that PHP was not originally designed but instead was developed organically has led to inconsistent naming of functions and inconsistent ordering of their parameters. In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping", while in some very early versions of PHP the length of the function names

was used internally as a hash function, so names were chosen to improve the distribution of hash values.

2.2.2 PHP Version

Each new version of PHP is given bellow:

- PHP 1.0- Released 0n 8 June 1995
- PHP 2.0- Released 0n 1 November 1997
- PHP 3.0- Released 0n 6 June 1998
- PHP 4.1- Released 0n 10 November 2001
- PHP 4.2- Released 0n 22 April 2002
- PHP 4.3- Released 0n 27 November 2002
- PHP 4.4- Released 0n 11 July 2005
- PHP 5.0- Released 0n 13 July 2004
- PHP 5.1- Released 0n 24 November 2005
- PHP 5.2- Released 0n 2 November 2006
- PHP 5.3- Released 0n 30 June 2009
- PHP 5.4- Released 0n 1 March 2012
- PHP 5.5- Released 0n 20 June 2013
- PHP 5.6- Released 0n 28 August 2014
- PHP 7.0- Released 0n 3 December 2015
- PHP 7.1- Released 0n 1 December 2016
- PHP 7.2- Released 0n 30 November 2017

2.2.3 PHP Framework

A PHP Framework is a basic platform that allows us to develop web applications. In other words, it provides structure. By using a Framework, you will end up saving loads of time, stopping the need to produce repetitive code, and you'll be able to build applications rapidly. Without a PHP Framework in place, it gets much more difficult to produce applications since you'll have to repeatedly code a lot of PHP. You'll also have to execute the connection between your database and whatever application you develop from scratch. Meanwhile, using a PHP Framework makes it easier for you to ensure this connection.

PHP operates on the Model View Controller (MVC) fundamentals. MVC is an architectural pattern featured in various popular programming languages which breaks apart your

domain logic from your user interface. The domain logic is the function that handles information exchange between your database and your user interface. Therefore, you're able to modify the domain logic and most importantly for designers, the user interface separately. This removes a lot of confusion and simplifies the entire developmental process.

2.2.4 List of some PHP Framework

1. Laravel- released in 2011

2. Symfony- published as free software on October 18, 2005 released under the MIT license.

3. CodeIgniter- initially released in 2006

4. CakePHP- released in 2005

2.2.5 Model Controller View(MVC)

Model–view–controller (MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user [27]. Traditionally used for desktop graphical user interfaces (GUIs), this architecture has become popular for designing web applications and even mobile, desktop and other clients. Popular programming languages like Java, C#, Ruby, PHP and others have popular MVC frameworks that are currently being used in web application development straight out of the box.

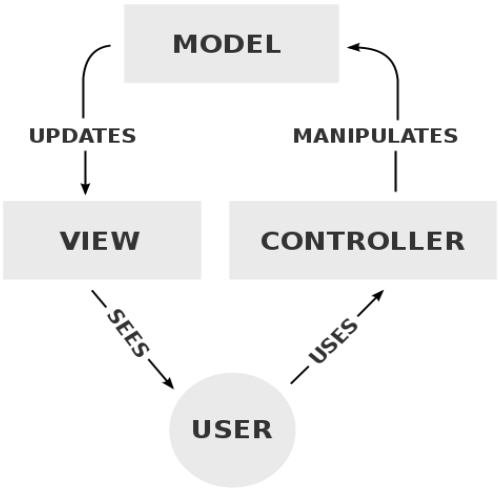


Figure 2.3: Model View Controller

2.2.6 Components

When you refer to MVC you generally perceive it as this: The **M** stands for the raw data, the **V** *(view/user interface)* represents what's actually being viewed, and **C** *(controller)* is in fact the domain logic[27]

- The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic and rules of the application.
- A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The third part or section, the controller, accepts input and converts it to commands for the model or view.

2.2.7 CodeIgniter the framework

CodeIgniter is an open-source software rapid development web framework, for use in building dynamic web sites with PHP. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. While controller classes are a necessary part of development under CodeIgniter, models and views are optional. Codeigniter can be

also modified to use Hierarchical Model View Controller (HMVC) which allows developers to maintain modular grouping of Controller, Models and View arranged in a sub-directory format.

CodeIgniter is most often noted for its speed when compared to other PHP frameworks. In a critical take on PHP frameworks in general, PHP creator Rasmus Lerdorf spoke at frOSCon in August 2008, noting that he liked CodeIgniter "because it is faster, lighter and the least like a framework. The first public version of CodeIgniter was released by EllisLab on February 28, 2006[8].

Reasons of what makes CodeIgniter a smart framework to use:

- Small footprint with exceptional performance
- MVC approach to development (although it is very loosely based which allows for flexibility)
- Generates search engine friendly clean URLs[8]
- Easily extensible
- Runs on both PHP 4 (4.3.2+) and 5 [8]
- Support for most major databases including MySQL (4.1+), MySQLi, MS SQL, Postgres, Oracle, SQLite,
  and ODBC [8].
- Application security is a focus
- Easy caching operations
- Many libraries and helpers to help you with complex operations such as email, image manipulation,
  form validation, file uploading, sessions, multilingual apps and creating apis for your app
- Most libraries are only loaded when needed which cuts back on resources needed[8].

2.3 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999[13]. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language[14].

2.3.1 JSON Structure

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.
- In JSON, they take on these forms:
  An object is an unordered set of name/value pairs. An object begins with **{** (left brace) and ends with **}** (right brace). Each name is followed by **:** (colon) and the name/value pairs are separated by **,** (comma)[14].

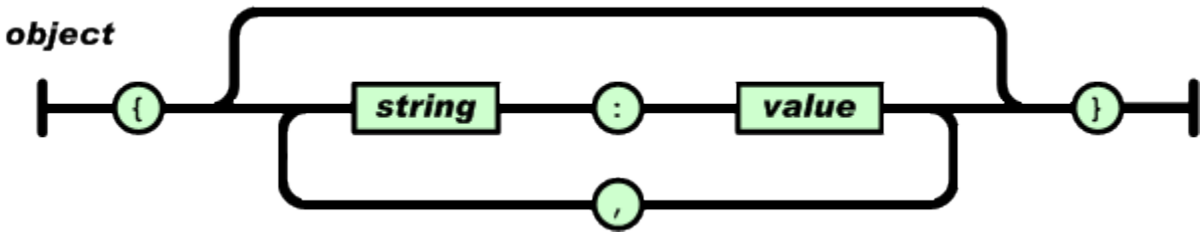
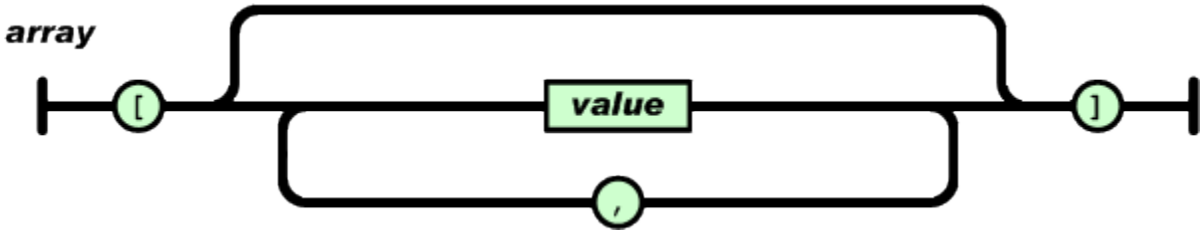


Figure 2.4: Json Key-value Object

An *array* is an ordered collection of values. An array begins with **[** (left bracket) and ends with**]** (right bracket). Values are separated by**,** (comma).



A *value* can be a *string* in double quotes, or a *number*, or **true** or **false** or **null**

Figure 2.5: Json Array

2.4 API

Application Programming Interface (API) a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service. API is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables or remote calls. POSIX, Windows API and ASPI are examples of different forms of APIs. Documentation for the API is usually provided to facilitate usage.

2.5 Purpose

Application programming interfaces make it easier for developers to use certain technologies in building applications. By abstracting the underlying implementation and only exposing objects or actions the developer needs, an API simplifies programming. While a graphical interface for an email client might provide a user with a button that performs all the steps for fetching and highlighting new emails, an API for file input/output might give the developer a function that copies a file from one location to another without requiring that the developer understand the file system operations occurring behind the scenes.

REGISTRATION

Build Your Account

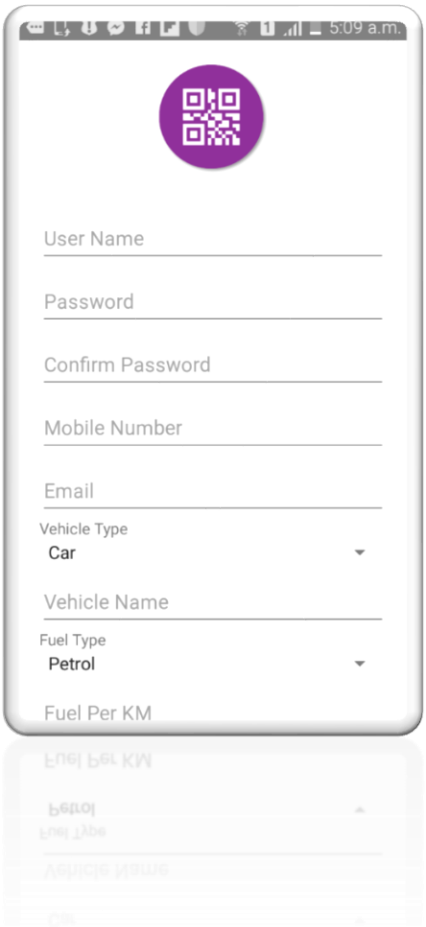


Figure 3.1: Registration Interface

3.1 Introduction:

It is a registration page where you can put your username and password and login to the app.

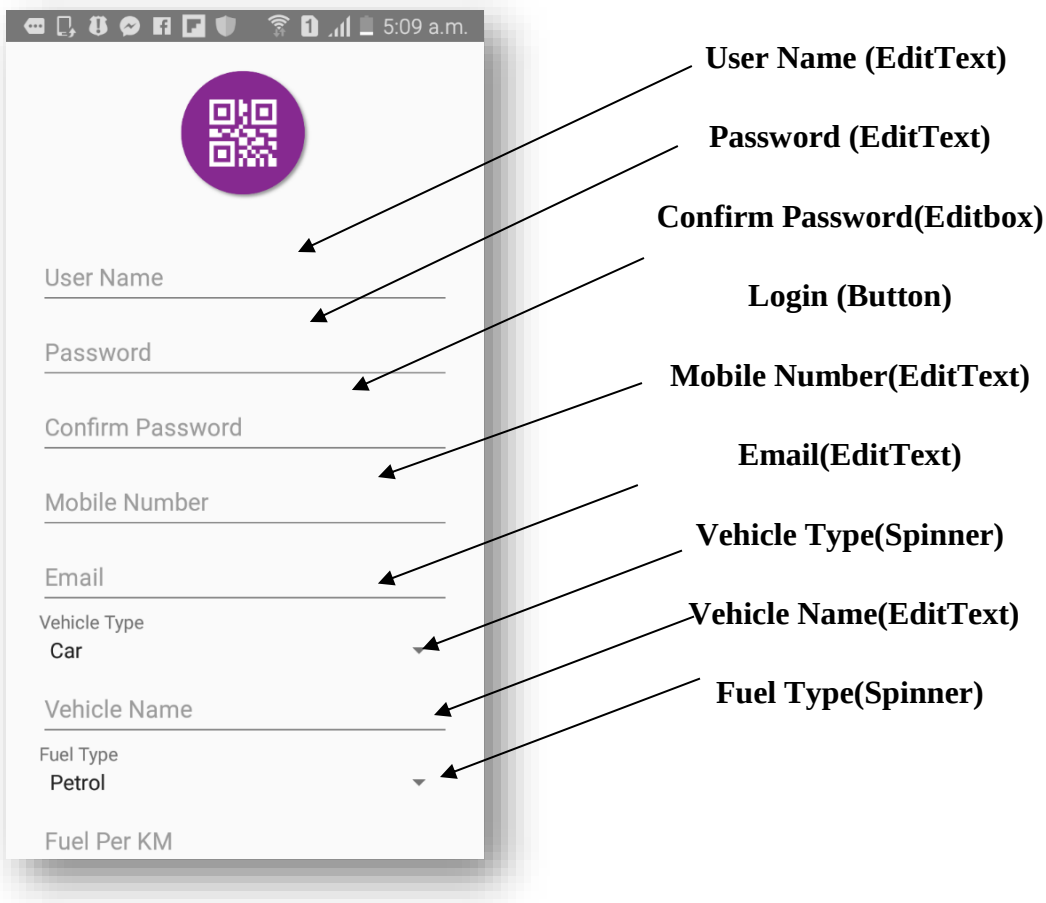


Figure 3.1: Registration Interface

3.2 Data Flow Diagram:

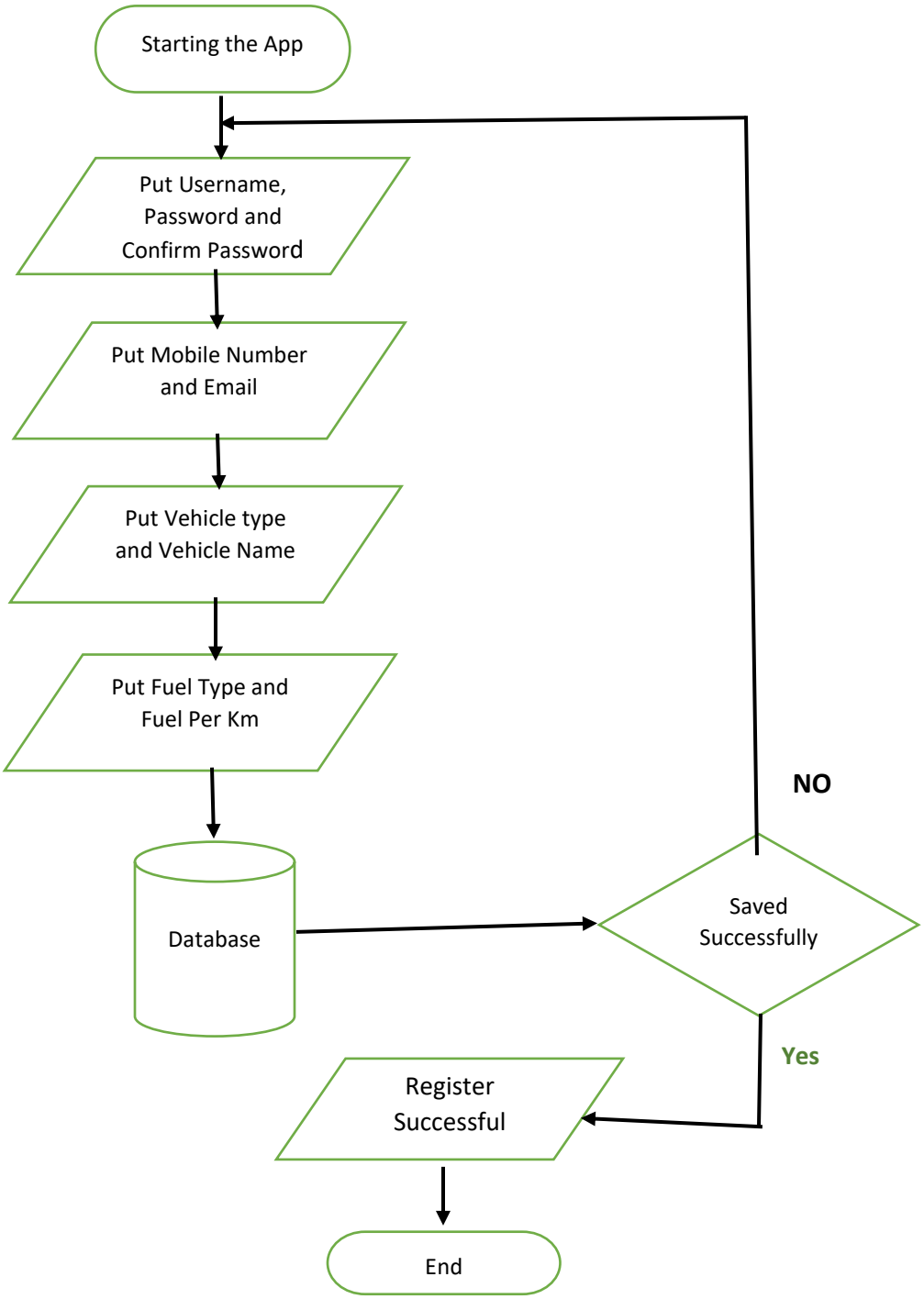


Figure 3.2: Flowchart of Registration

3.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with EditText, Spinner, TextView and Button. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

3.4 Interface of Registration:

- This is the app interface where you give username, password, mobile no, email address, your vehicle information like your vehicle name, vehicle type, what type of fuel you used in your vehicle and how much fuel you required in per km.

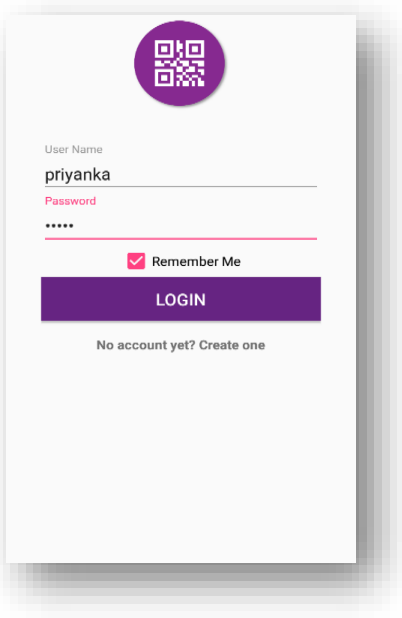
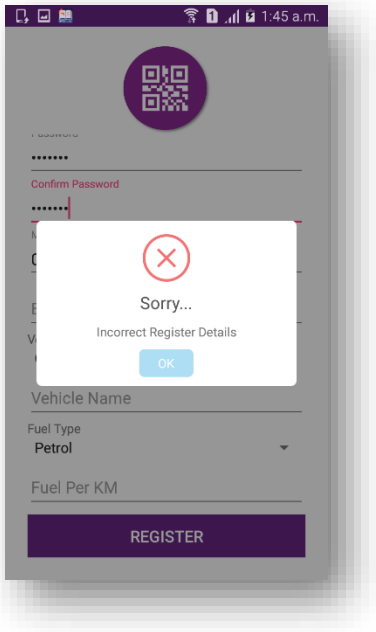
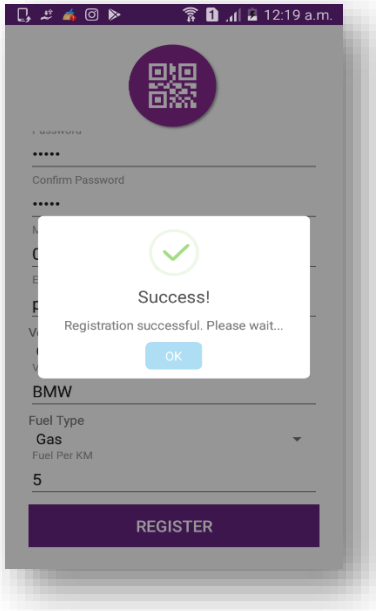


Figure 3.1: Registration Interface



- All Steps are not completely fill up

- When the request is successful then the popup screen will be confirmed.



3.5 Building the app GUI

In this section, you'll build the GUI for the **REGISTRATION System**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_register.xml file

You'll add a TextView, EditText and Button under LinearLayout and ImageView under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
    xmlns:gradient="http://schemas.android.com/apk/res-
auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:gravity="center">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:src="@drawable/logo" />


    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView2">

        <LinearLayout
            android:id="@+id/ln"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="@dimen/padding_form">


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etName"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="User Name"
                    android:inputType="text"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etPassword"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Password"
                    android:inputType="textPassword"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etConPassword"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Confirm Password"
                    android:inputType="textPassword"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etMobile"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Mobile Number"
                    android:inputType="text"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etEmail"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Email"
                    android:inputType="textEmailAddress"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Vehicle Type"
                android:textSize="14sp"

                />

            <Spinner
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:autofillHints="Vehicle Type"
                android:entries="@array/vehicleType"
                android:id="@+id/vehicleType"

                >

            </Spinner>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etVehicleName"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Vehicle Name"
                    android:inputType="text"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Fuel Type"
                android:textSize="14sp"

                />

            <Spinner
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:autofillHints="Fuel Type"
                android:entries="@array/fuelType"
                android:id="@+id/fuelType"

                >

            </Spinner>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etFuelPerkm"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Fuel Per KM"
                    android:inputType="text"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


            <Button
                android:id="@+id/btnRegister"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="6dp"
                android:text="Register"
                android:textColor="@color/colorwhite"
                android:background="@color/colorPrimary"
                android:onClick="register"
                android:textSize="@dimen/text_medium" />

        </LinearLayout>
    </ScrollView>


</RelativeLayout>
```

3.6 Java Implementation for Registration

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also

initializes the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are one threads to communicate with server to give data. This thread request data from server and get a list from server and put it on database for storing data otherwise server response false data if any error occurs. There are a few data get from server like *'user_name', 'password', 'moible_no', 'email', 'fuel_type', 'vehicle_type', 'vehicle_name' and 'fuel_per_km'*.

```
public class RegisterActivity extends Activity {

    String registerUrl, serverResponse, success,msg,
Result;
    EditText name, password, conpassword, mobile,
vehicleName, fuelPerkm, email;
    Spinner vehicleType, fuelType;
    private ProgressDialog pDialog;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    HttpConnectionClass httpClass;
    JSONObject userRegisterInfoJSON = null;
    private Handler handler = new Handler();


    @Override
    protected void onCreate(@Nullable Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        initialize();
    }

    public void initialize() {
        registerUrl = getString(R.string.server_address)
                + "api/api_register/register";

        name = findViewById(R.id.etName);
        password = findViewById(R.id.etPassword);
        conpassword = findViewById(R.id.etConPassword);
        mobile = findViewById(R.id.etMobile);
        email = findViewById(R.id.etEmail);
        vehicleType = findViewById(R.id.vehicleType);
        vehicleName = findViewById(R.id.etVehicleName);
        fuelType = findViewById(R.id.fuelType);
        fuelPerkm = findViewById(R.id.etFuelPerkm);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);

        httpClass = new
HttpConnectionClass(RegisterActivity.this);
    }

    public void register(View v) {
        if (internetValidation() &&
passwordValidation()) {
            new Thread(new LoadRegisterTask()).start();
            showpDialog();
        }
    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(RegisterActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    public boolean passwordValidation() {
        if
(password.getText().toString().equals(conpassword.getTex
t().toString())) {
            return true;
        } else {
            new SweetAlertDialog(RegisterActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.passwordHeader))

.setContentText(getString(R.string.passwordMessage))
                    .show();
            return false;
        }
    }

    private class LoadRegisterTask implements Runnable {


        LoadRegisterTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(registerUrl); // here
is your URL path

                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("user_name",
name.getText().toString());
                postDataParams.put("password",
password.getText().toString());
                postDataParams.put("monbile_no",
mobile.getText().toString());
```

```
                postDataParams.put("email",
email.getText().toString());
                postDataParams.put("fuel_type",
fuelType.getSelectedItem().toString());
                postDataParams.put("vehicle_type",
vehicleType.getSelectedItem().toString());
                postDataParams.put("vehicle_name",
vehicleName.getText().toString());
                postDataParams.put("fuel_per_km",
fuelPerkm.getText().toString());

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                userRegisterInfoJSON = new
JSONObject(serverResponse);

                    /*JSONObject eachObjFromJSONArray =
userLoginInfoJSON
                            .getJSONObject("otp");*/


                success = userRegisterInfoJSON
                        .getString("success");
                msg = userRegisterInfoJSON
                        .getString("msg");

                Result = "";

            }catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(RegisterActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {
                            successAlert();
                        } else {
                            new
SweetAlertDialog(RegisterActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))
                                        .setContentText(msg)
                                        .show();
                            }
                        }
                    }
                });
            }

        }



        private void showpDialog() {
            if (!pDialog.isShowing())
                pDialog.show();
        }

        private void hidepDialog() {
            if (pDialog.isShowing())
                pDialog.dismiss();
        }

        public void successAlert() {
            new SweetAlertDialog(RegisterActivity.this,
SweetAlertDialog.SUCCESS_TYPE)
                    .setTitleText("Success!")
                    .setContentText("Registration
successful. Please wait...")
                    .show();
            Thread delay = new Thread() {

                public void run() {

                    try {
                        sleep(2000);
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    } finally {
                        startActivity(new
Intent(getBaseContext(),
                                LoginActivity.class));
                        finish();
                    }
                }
            };
            delay.start();


        }
}
```

Code 3.3 Java Code of Registration

3.7 PHP Implementation for Registration

The request data From PHP end it can get data from Android through API, store data into database and response a message to Android.

- Register API

```
function register() {
        $params['user_name'] = $this->input-
>post('user_name', TRUE);
        $params['password'] = md5($this->input-
>post('password', TRUE));
        $params['monbile_no'] = $this->input-
>post('monbile_no', TRUE);
        $params['email'] = $this->input->post('email',
TRUE);
        $params['fuel_type'] = $this->input-
>post('fuel_type', TRUE);
        $params['vehicle_type'] = $this->input-
>post('vehicle_type', TRUE);
        $params['vehicle_name'] = $this->input-
>post('vehicle_name', TRUE);
        $params['fuel_per_km'] = $this->input-
>post('fuel_per_km', TRUE);
        $params['accountNumber'] = $this->input-
>post('monbile_no', TRUE);
        $params['amount'] = 0;
        $params['active'] = 1;

        $date = date("Y-m-d H:i:s");
        $params['create_dt_tm'] = $date;
        $params['update_dt_tm'] = $date;

      /*  $params['imei_number'] = $this->input-
>post('imei_number', TRUE);
        $params['os_code'] = $this->input->post('os_code',
TRUE);
        $params['device_info'] = $this->input-
>post('device_info', TRUE);
*/

        $this->load->library('form_validation');
       // $this->form_validation->CI = & $this;
        $this->form_validation->set_rules('user_name',
'User Name', 'required|trim|callback_checkUser');
        $this->form_validation->set_rules('email',
'Email', 'trim|required|callback_checkEmail');
        $this->form_validation->set_rules('monbile_no',
'Monbile No', 'trim|required|callback_checkMobile');


        if ($this->form_validation->run() == FALSE) {

            $info = "Incorrect Register Details";
            $json = array(
                "success" => false,
                "msg"=>$info
            );

            echo json_encode($json);
            die();
        }

        $res = array();

        $result = $this->Api_register_model-
>insertRegisterInfo($params);

        $info = "Inserted Successfully.";
        $success = "true";

        $json = array(
            "success" => $success,
            "msg" => $info
        );

        echo json_encode($json);
    }
```

Code 3.4 PHP Code of Registration

- Registration database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	register_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	user_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	password	varchar(32)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	monbile_no	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
7	vehicle_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
8	vehicle_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
9	fuel_per_km	int(11)			No	None			Change Drop Primary Unique Index More
10	accountNumber	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
11	amount	int(20)			No	None			Change Drop Primary Unique Index More
12	active	int(11)			Yes	1			Change Drop Primary Unique Index More
13	create_dt_tm	date			No	None			Change Drop Primary Unique Index More
14	update_dt_tm	date			No	None			Change Drop Primary Unique Index More

LOGIN

Enter Your Fuel Payment System Account

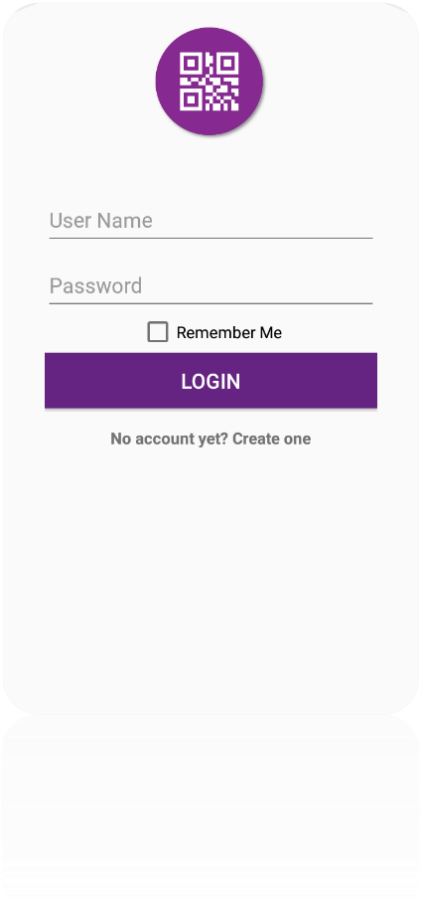


Figure 4.1: Login Interface

4.1 Introduction:

It is a login page where you can put your username and password and login to the app.

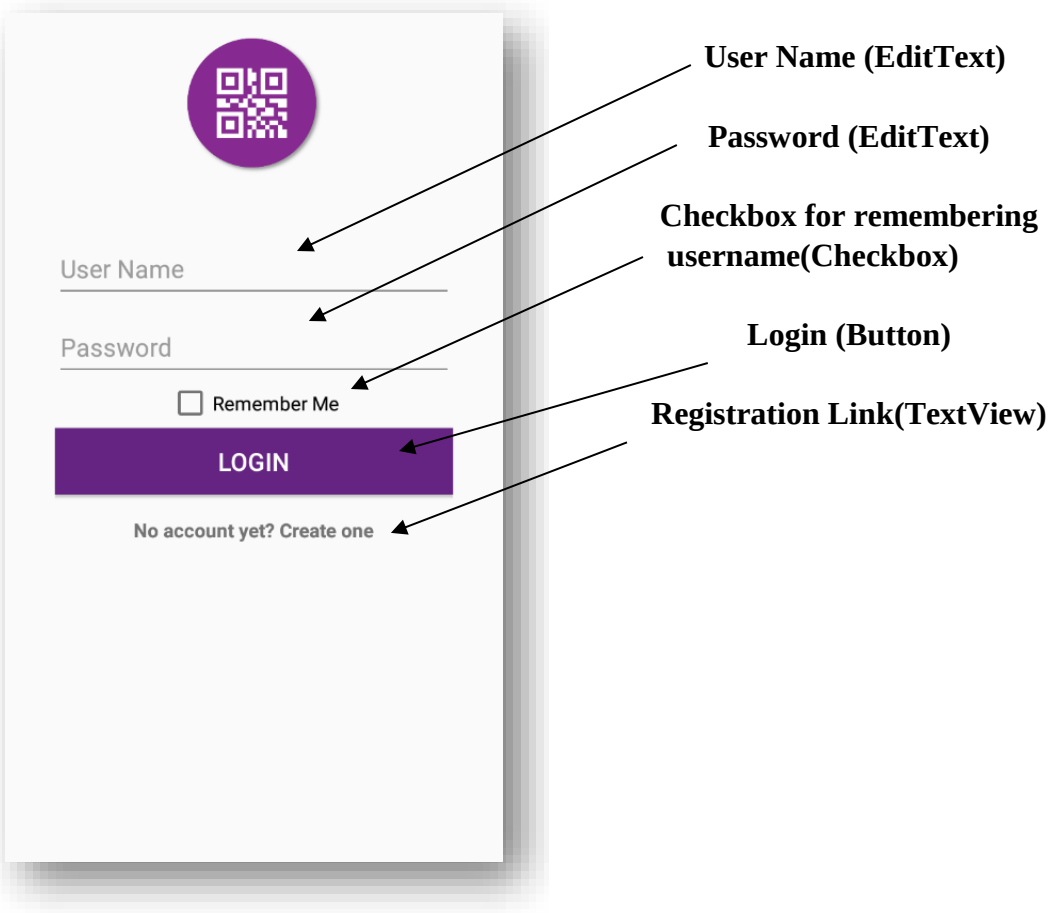


Figure 4.1: Login Interface

4.2 Data Flow Diagram:

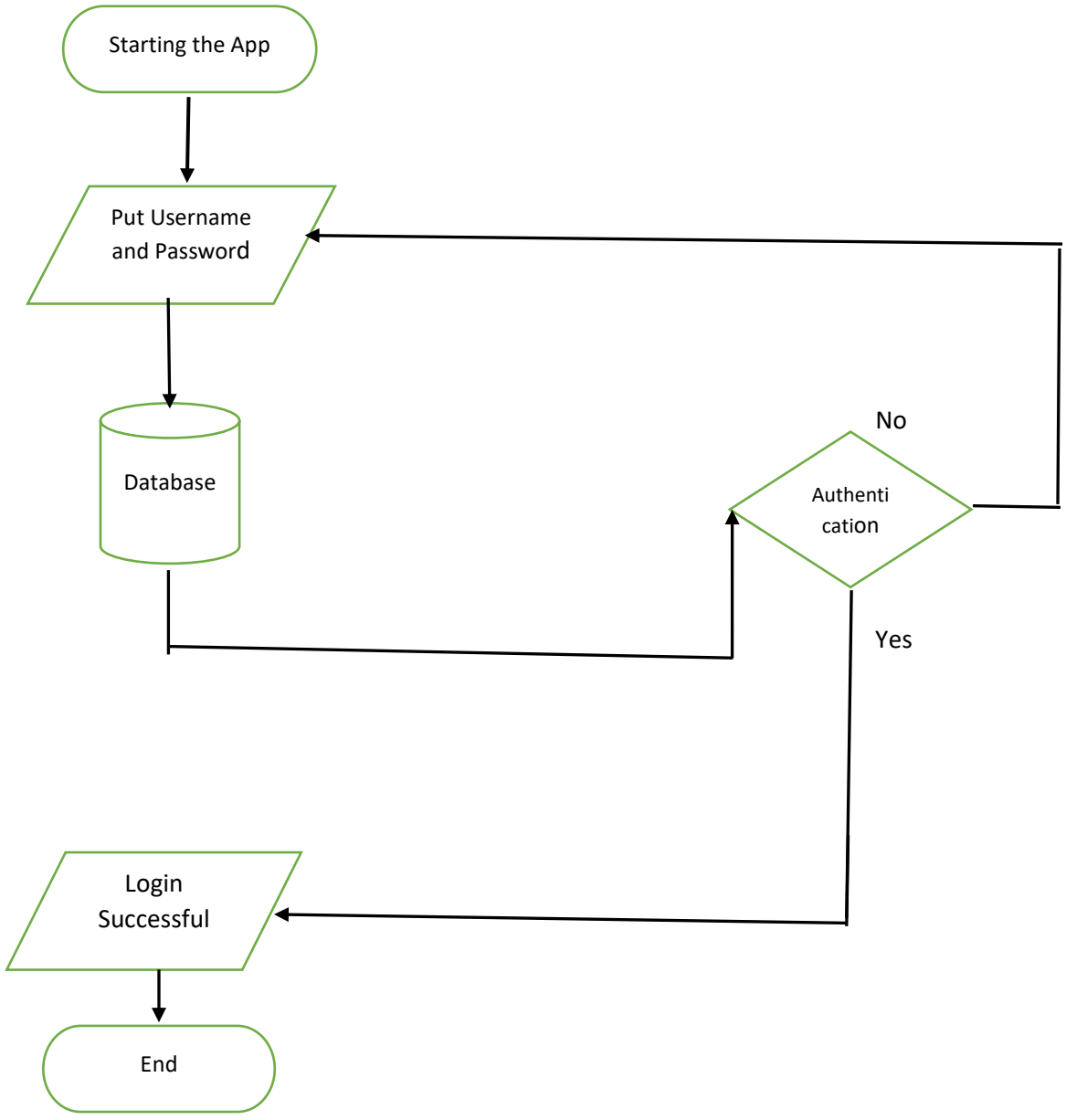


Figure 4.2: Flowchart of Login

4.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, Json is used and also database is used to store the value. In android, you'll programmatically interact with two EditTexts, a Checkbox, a TextView, a ImageView and Button. You'll create these components by direct manipulation of the GUI layout's XML. An *EditText*—often called a text box or text field in other GUI technologies—is a subclass of *TextView* that can display text and accept text input from the user. A *Button*—often called a slider in other GUI technologies. A Checkbox- user can choose a value or not. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of Json. Json encode it. Json take the value in the form of json array with a key value.

4.4 Interface of LOGIN:

- This is the app interface with "priyanka" set as username and "" is provided as password and checkbox is chosen. It can be changed anytime later.

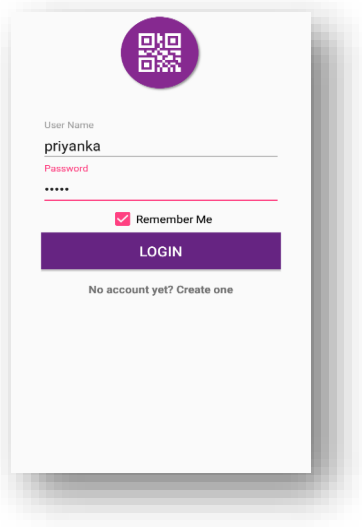
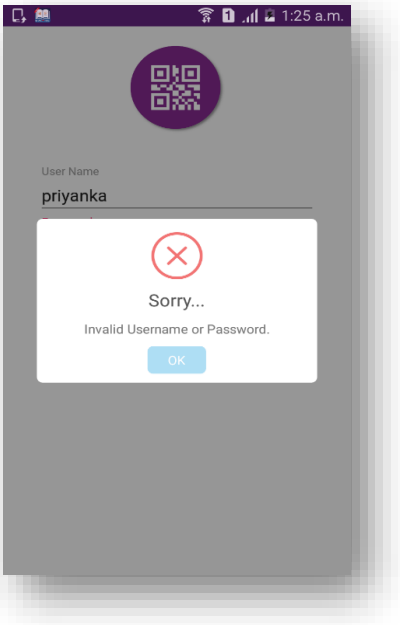
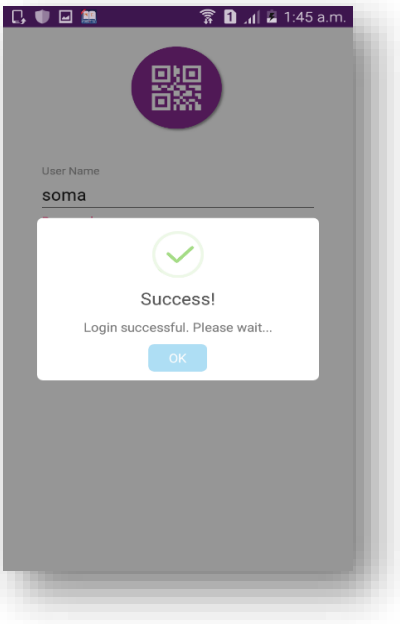


Figure 4.1: Login Interface

- Username or password is incorrect



- When the request is successful then the popup screen will be confirmed.



4.5 Building the app GUI

In this section, you'll build the GUI for the **LOGIN System**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_login.xml file

You'll add a TextView, EditText, CheckBox, Button under LinearLayout and ImageView under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:gradient="http://schemas.android.com/apk/res-
auto"
    android:focusable="true"
    android:focusableInTouchMode="true">

    <LinearLayout
        android:id="@+id/main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView2"
        android:padding="@dimen/padding_sides"
        android:orientation="vertical">

        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <EditText
                android:id="@+id/etUserName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="User Name"
                android:inputType="textEmailAddress"
                android:textSize="@dimen/text_medium" />

        </android.support.design.widget.TextInputLayout>

        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <EditText
                android:id="@+id/etPassword"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Password"
                android:inputType="textPassword"
                android:textSize="@dimen/text_medium" />

        </android.support.design.widget.TextInputLayout>
```

```
        <CheckBox
            android:id="@+id/checkboxRemember"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Remember Me" />

        <Button
            android:id="@+id/btnLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="6dp"
            android:textColor="@color/colorwhite"
            android:background="@color/colorPrimary"
            android:layout_marginTop="2dp"
            android:onClick="signIn"
            android:text="Login"
            android:textSize="@dimen/text_medium" />


    <TextView
        android:id="@+id/tvRegister"
        android:layout_below="@+id/main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:clickable="true"
        android:onClick="register"
        android:text="No account yet? Create one"
        android:textSize="@dimen/text_small"
        android:textStyle="bold"
        android:layout_gravity="center" />

    </LinearLayout>

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginTop="20dp"
        android:layout_centerHorizontal="true"
        android:src="@drawable/logo" />

        </RelativeLayout>
```

4.6 Java Implementation for LOGIN

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initializes the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are one threads to communicate with server to get data. This thread request data from server and get a list from server and put it on arrayList for showing a list otherwise server response false data if any error occurs. There are a few data get from server like *'registerId', 'accountNumber', 'amount', 'moible_no', 'email', 'user_name', 'fuel_type', 'vehicle_type', 'vehicle_name' and 'fuel_per_km'*.

```
public class LoginActivity extends Activity {

    String loginUrl = "", serverResponse = "", Result =
"", success = "",registerId="",accountNumber="",

amount,monbile_no,email,user_name,fuel_type,vehicle_type,v
ehicle_name,fuel_per_km;
    CheckBox saveUser;
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    JSONObject userLoginInfoJSON = null;
    HttpConnectionClass httpClass;
    SharedPreferencesClass storePreference;
    EditText etUserName, etPassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        initialize();
    }

    public void initialize() {
        loginUrl = getString(R.string.server_address)
                + "api/api_login/login";
        httpClass = new HttpConnectionClass(this);
        storePreference = new
SharedPreferencesClass(getApplicationContext());
        etUserName = findViewById(R.id.etUserName);
        etPassword = findViewById(R.id.etPassword);
        saveUser= (CheckBox)
findViewById(R.id.checkboxRemember);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);


etUserName.setText(storePreference.getString("userName"));
        checkbox();
    }

    public void checkbox(){

if(!storePreference.getString("userName").equals("")) {
            saveUser.setChecked(true);
        }
        saveUser.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (((CheckBox) v).isChecked()) {
                    storePreference.putString("userName",
etUserName.getText().toString());

                } else {
                    storePreference.putString("userName",
"");
                }
            }
        });
    }

    public void register(View v) {
        Intent intent = new Intent(LoginActivity.this,
                RegisterActivity.class);
        startActivity(intent);
    }

    public void signIn(View v) {
        if (internetValidation()) {
            new Thread(new LoadLoginTask()).start();
            showpDialog();
        }
    }


    private class LoadLoginTask implements Runnable {


        LoadLoginTask() {
        }


        @Override
        public void run() {

            try {
                URL url = new URL(loginUrl); // here is
your URL path
```

```
                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("user_name",
etUserName.getText().toString());
                postDataParams.put("password",
etPassword.getText().toString());

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                userLoginInfoJSON = new
JSONObject(serverResponse);

                success = userLoginInfoJSON
                        .getString("success");
                if(success.equals("true")) {
                    JSONArray loginArray =
userLoginInfoJSON
                            .getJSONArray("info");

                    JSONObject eachObjFromJSONArray =
loginArray
                            .getJSONObject(0);
                    registerId = eachObjFromJSONArray
                            .getString("register_id");
                    accountNumber = eachObjFromJSONArray
                            .getString("accountNumber");
                    amount = eachObjFromJSONArray
                            .getString("amount");
                    monbile_no = eachObjFromJSONArray
                            .getString("monbile_no");
                    email = eachObjFromJSONArray
                            .getString("email");
                    user_name = eachObjFromJSONArray
                            .getString("user_name");


                    fuel_type = eachObjFromJSONArray
                            .getString("fuel_type");
                    vehicle_type = eachObjFromJSONArray
                            .getString("vehicle_type");
                    vehicle_name = eachObjFromJSONArray
                            .getString("vehicle_name");
                    fuel_per_km = eachObjFromJSONArray
                            .getString("fuel_per_km");


storePreference.putString("registerId", registerId);

storePreference.putString("accountNumber", accountNumber);

                    storePreference.putString("amount",
amount);

storePreference.putString("monbile_no", monbile_no);
                    storePreference.putString("email",
email);
                    storePreference.putString("user_name",
user_name);

                    storePreference.putString("fuel_type",
fuel_type);

storePreference.putString("vehicle_type", vehicle_type);

storePreference.putString("vehicle_name", vehicle_name);

storePreference.putString("fuel_per_km", fuel_per_km);
                }

                Result = "";

            }catch (Exception ex) {
                Result = "Exception";
            }
```

```
            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(LoginActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {
                            successAlert();
                        } else {
                            new
SweetAlertDialog(LoginActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))

.setContentText(getString(R.string.lofinFailMessage))
                                    .show();
                        }
                    }
                }
            });
        }

    }


    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(LoginActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    public void successAlert() {
        new SweetAlertDialog(LoginActivity.this,
SweetAlertDialog.SUCCESS_TYPE)
                .setTitleText("Success!")
                .setContentText("Login successful. Please
wait...")
                .show();
        Thread delay = new Thread() {

            public void run() {

                try {
                    sleep(2000);
                } catch (Exception ex) {
                    ex.printStackTrace();
                } finally {
                    startActivity(new
Intent(getBaseContext(),
                            HomeMainActivity.class));
```

```
                finish();
            }
        }
    };
```

```
        delay.start();

    }
}
```

Code 4.3 Java Code of Login

4.7 PHP Implementation for Login

The request data From PHP end it can get data from Android through API, store data into database and response a message to Android.

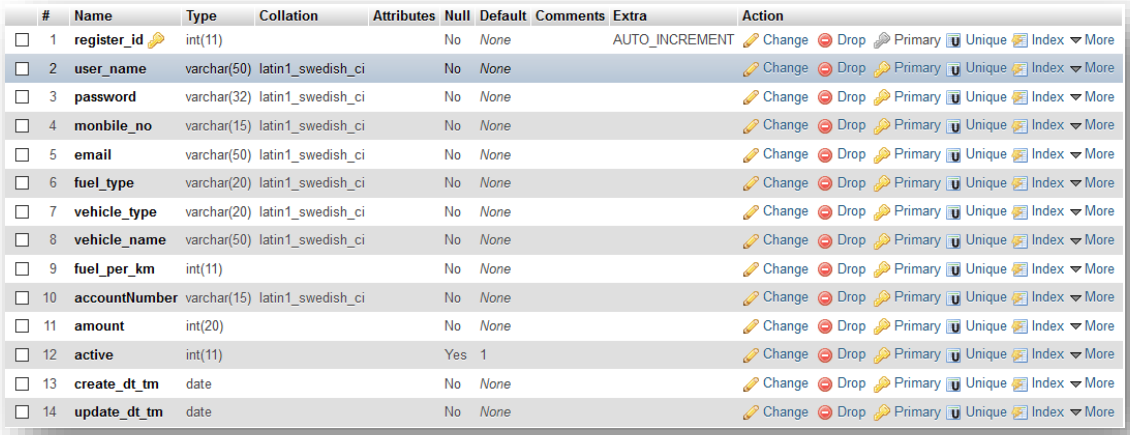
- Login API

```
function login() {

        $params['user_name'] = $this->input-
>post('user_name', TRUE);

        $params['password'] = md5($this->input-
>post('password', TRUE));



        $this->load->library('form_validation');



        $this->form_validation->set_rules('user_name', 'User
Name', 'required|trim|callback_checkUser');

        $this->form_validation->set_rules('password',
'Password', 'trim|required|callback_checkPassword');



        if ($this->form_validation->run() == FALSE) {

            $info = "Invalid User Name or Password.";

            $json = array(

                "success" => false,

                "msg" => $info

            );



            echo json_encode($json);

            die();
```

```
        }

        $res = array();

        $result = $this->Api_login_model-
>checkLoginInfo($params);



        if (!$result):


            $info = "Incorrect Login Details";

            $success = "false";

        else:

            $res = $result->row();

            $info[] = $res;

            $success = "true";

        endif;



        $json = array(

            "success" => $success,

            "info" => $info

        );

        echo json_encode($json);

    }
```

Code 4.3 Php Code of Login

- Login database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	register_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	user_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	password	varchar(32)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	monbile_no	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
7	vehicle_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
8	vehicle_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
9	fuel_per_km	int(11)			No	None			Change Drop Primary Unique Index More
10	accountNumber	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
11	amount	int(20)			No	None			Change Drop Primary Unique Index More
12	active	int(11)			Yes	1			Change Drop Primary Unique Index More
13	create_dt_tm	date			No	None			Change Drop Primary Unique Index More
14	update_dt_tm	date			No	None			Change Drop Primary Unique Index More

Home Page

App Interface

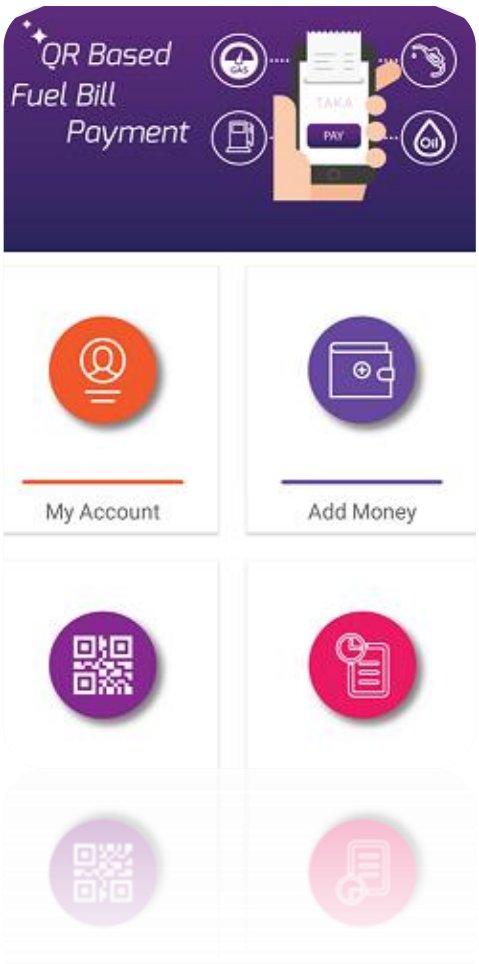
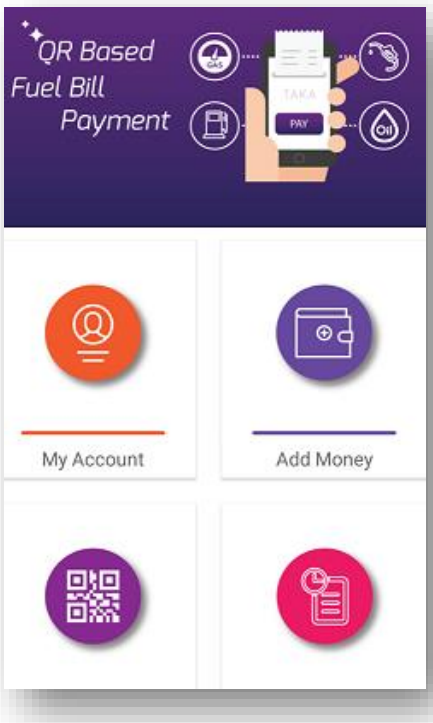


Figure 5.1: Dashboard Interface

5.1 Introduction:

This chapter is based on our project's Home Page. By giving user name and password user can log in to this home page. By using intent user can go to one page to another.

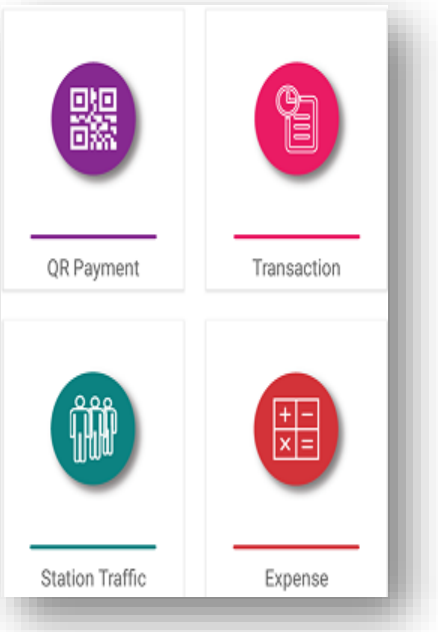


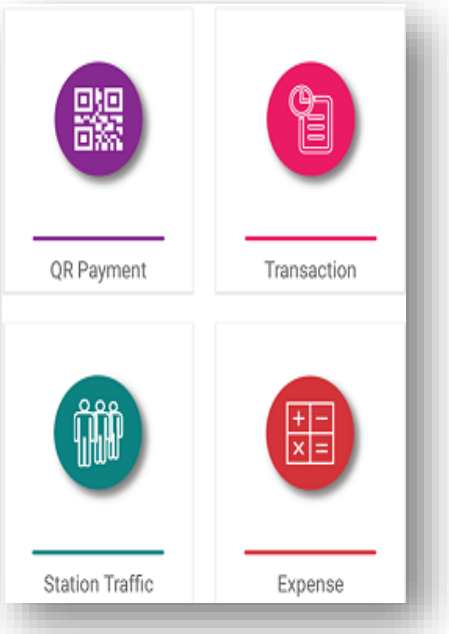
- **My Account:** By clicking in this a user can see own's name, account name, mobile number, picture and his amount by using intent.

- **Add Money**: User can add money to the account. By this amount user can payment for the fuel taken.

- **QR Payment:** In this user can do the payment of fuel. When a user would take fuel ,user have to scan the corresponding QR code of that particular station for payment.

- **Transaction**: In this module user can see the amount of money user have spend to a particular station in a list.





- **Station Traffic**: In this content user can see the traffic of a station. Every user's entry makes an increasing number of vehicles, which is here we named it station traffic.

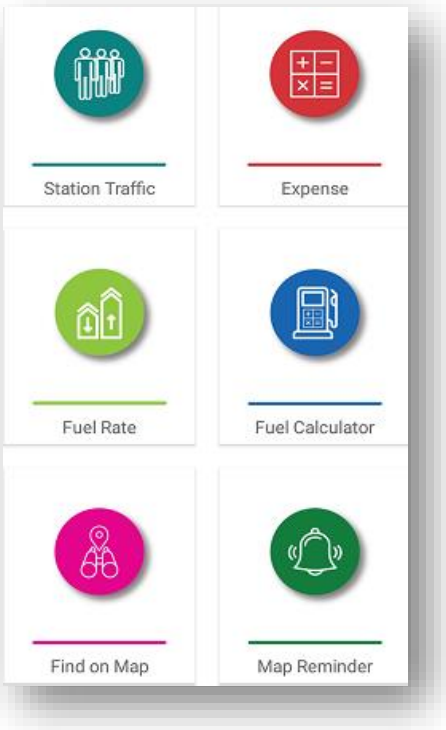
- **Expense:** User's expense history can be seen in this content. How much a user has done payment in a day, week, month and as well as year.

.

.

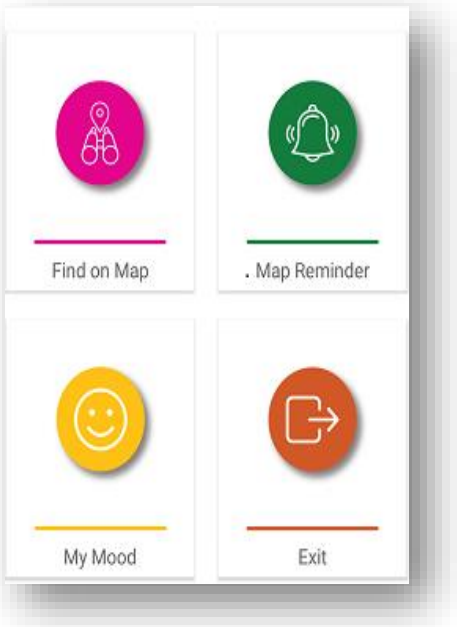
- **Fuel Rate:** In this user can see the fuel rate like if giving input of fuel type, fuel weight user can know the current rate of that much fuel.

- **Fuel Calculator:** If a user gives a input of fuel user can see the price of that much of fuel amount or give a input of a price a user can see the fuel quantity corresponding to that price.



- **Find on Map:** User can find the station location in this module. In the search bar user can search the station name and user will see that particular station's location on google map.

- **Map Reminder:** In map reminder user can see that how far user can go by the current weight of fuel he/she have. Like user have 2 litres of fuel, so by this weight user can go 10 km.



- **My Mood:** In this there are two parts:
  1. **Normal Mood**: In normal mood user is not driving car so user can receive the incoming call.
  2. **Driving Mood:** In this mood user is driving the car and if they have any incoming call then from user's phone incoming caller will receive a busy tone and will get a message that user will call that incoming caller later.

- **Exit:** User can logout from account by clicking the exit button. Exit leads a user to the login page again.

My Account

User Personal and Vehicle Information

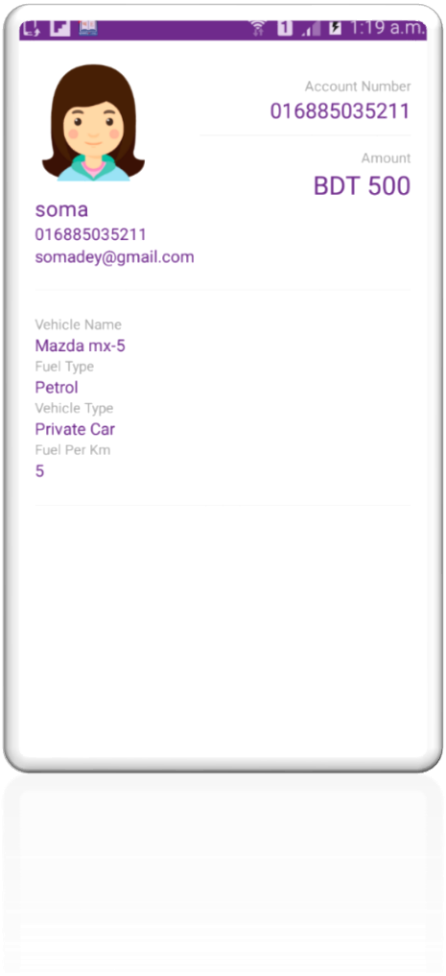


Figure 6.1: My Account Interface

6.1 Introduction:

It is a page where you can see your profile and vehicle information that you give in registration time and also can see the amount that you have in your own account.

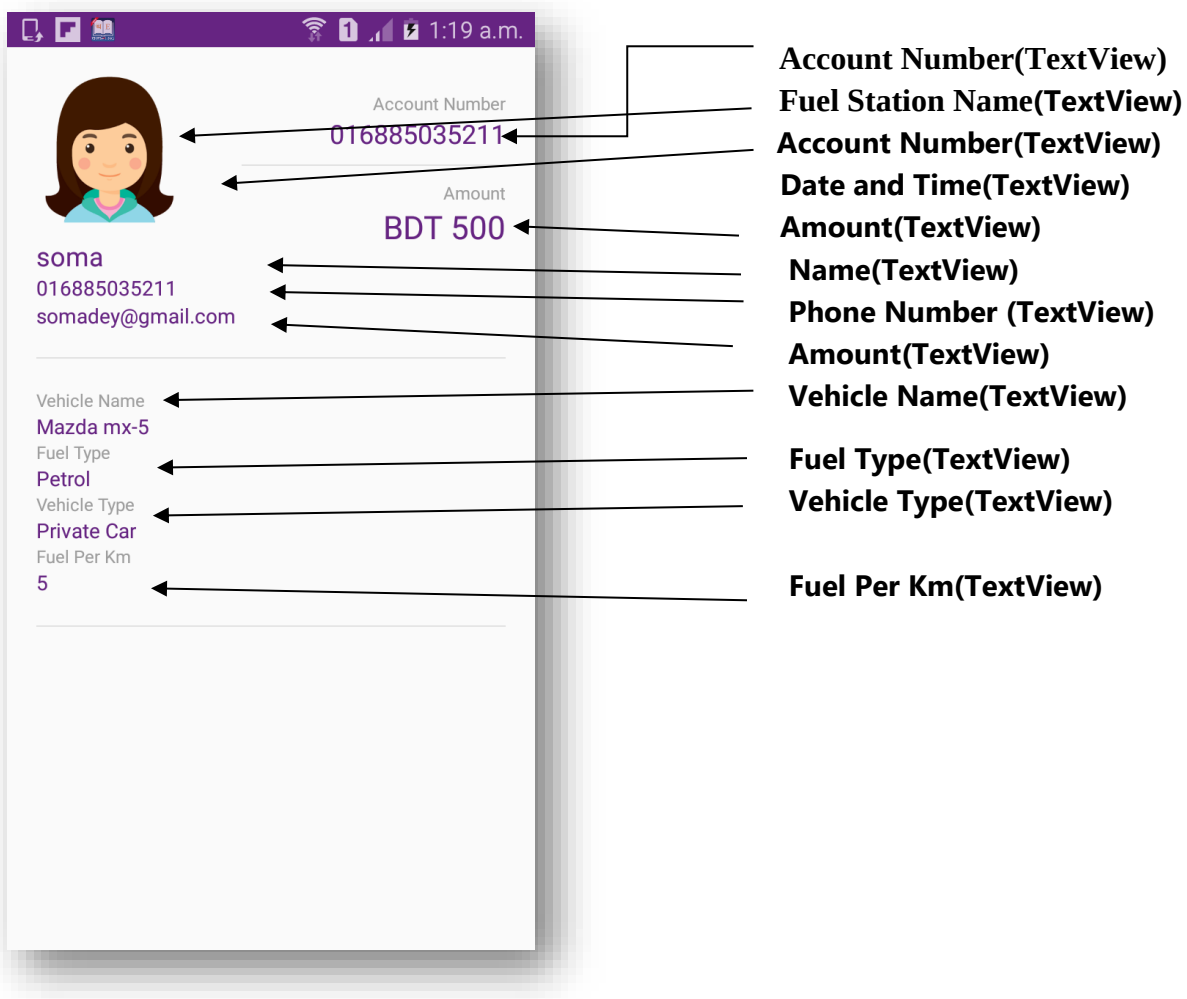


Figure 6.1: My Account Interface

6.2 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with TextView, View and ImageView. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

6.3 Interface of My Account:

- This is the app interface with TextView, View and ImageView for watching the profile history.

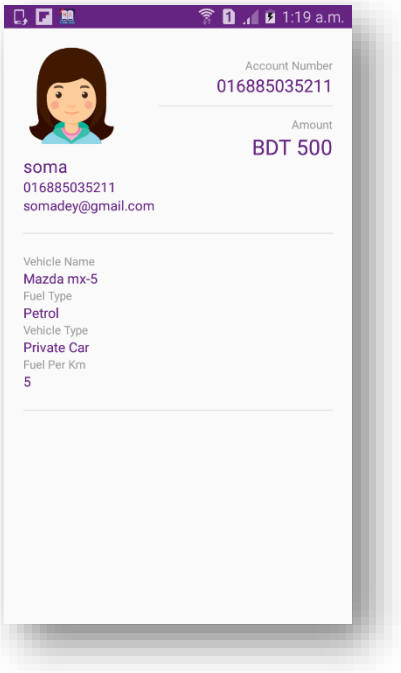


Figure 6.1: My Account Interface

6.4 Building the app GUI

In this section, you'll build the GUI for the **Transaction**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_profile.xml file

You'll add a TextView, View and ImageView under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/padding_list">
    <RelativeLayout
        android:id="@+id/layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/profilepicture"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/profile_picture"/>
        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/profilepicture"
            android:text="Name"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_medium"
            android:layout_marginTop="10dp"/>
        <TextView
            android:id="@+id/mobile"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/name"
            android:text="Mobile"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>
        <TextView
            android:id="@+id/email"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/mobile"
            android:text="Email"
            android:layout_marginBottom="20dp"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginTop="20dp"
            android:layout_above="@+id/vehicle_name_text"
            android:background="@color/colorAsh"/>

        <TextView
            android:id="@+id/vehicle_name_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/email"
            android:text="Vehicle Name"
            android:layout_marginTop="20dp"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>
        <TextView
            android:id="@+id/vehicle_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/vehicle_name_text"
            android:text="Vehicle Name"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>

        <TextView
            android:id="@+id/fuel_type_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/vehicle_name"
            android:text="Fuel Type"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>
        <TextView
            android:id="@+id/fuel_type"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/fuel_type_text"
            android:text="Vehicle Name"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>

        <TextView
            android:id="@+id/vehicle_type_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/fuel_type"
            android:text="Vehicle Type"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>
        <TextView
            android:id="@+id/vehicle_type"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/vehicle_type_text"
            android:text="Vehicle Name"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>

        <TextView
            android:id="@+id/fuel_per_km_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/vehicle_type"
            android:text="Fuel Per Km"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>
```

```
        <TextView
            android:id="@+id/fuel_per_km"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/fuel_per_km_text"
            android:text="Vehicle Name"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_small"/>
        <TextView
            android:id="@+id/accnotext"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Account Number"
            android:layout_marginTop="10dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>

        <TextView
            android:id="@+id/accno"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Account Number"
            android:layout_marginBottom="10dp"
            android:layout_below="@+id/accnotext"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_medium"/>

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginTop="20dp"
            android:layout_marginLeft="40dp"
```

```
android:layout_toRightOf="@+id/profilepicture"
            android:layout_above="@+id/amountText"
            android:background="@color/colorAsh"/>


        <TextView
            android:id="@+id/amountText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Amount"
            android:layout_marginTop="10dp"
            android:layout_below="@+id/accno"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:textColor="@color/colorgrey"
            android:textSize="@dimen/text_micro"/>

        <TextView
            android:id="@+id/amount"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_below="@+id/amountText"
            android:layout_centerVertical="true"
            android:text="BDT 0"
            android:textColor="@color/colorPrimary"
            android:textSize="@dimen/text_large" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginTop="20dp"
            android:layout_below="@+id/fuel_per_km"
            android:background="@color/colorAsh"/>
    </RelativeLayout>
</RelativeLayout>
```

6.5 Java Implementation for My Account

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initializes the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are one threads to communicate with server to get data. This thread request data from server and get a list from server and put it on arrayList for showing a list otherwise server response false data if any error occurs. There are a few data get from server like *'name', 'amount', 'mobile', 'email', 'vehicle_name, 'fuel_type' etc.*

```
public class ProfileActivity extends Activity {

    TextView accno, amount, name, mobile, email,
vehicle_name, fuel_type, fuel_per_km, vehicle_type;
    SharedPreferencesClass storePreference;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    String amountUrl,serverResponse = "", Result =
"",newAmount="",success="";
    JSONObject amountJSON = null;
    HttpConnectionClass httpClass;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);
        initialize();
    }

    public void initialize() {
        storePreference = new
SharedPreferencesClass(getApplicationContext());
        amountUrl = getString(R.string.server_address)
                + "api/api_login/amount";
        httpClass = new HttpConnectionClass(this);

        accno = findViewById(R.id.accno);

accno.setText(storePreference.getString("accountNumber"))
;

        amount = findViewById(R.id.amount);
       // amount.setText("BDT
"+storePreference.getString("amount"));

        name = findViewById(R.id.name);

name.setText(storePreference.getString("user_name"));

        mobile = findViewById(R.id.mobile);

mobile.setText(storePreference.getString("monbile_no"));

        email = findViewById(R.id.email);

email.setText(storePreference.getString("email"));

        vehicle_name = findViewById(R.id.vehicle_name);

vehicle_name.setText(storePreference.getString("vehicle_n
ame"));

        fuel_type = findViewById(R.id.fuel_type);

fuel_type.setText(storePreference.getString("fuel_type"))
;

        vehicle_type = findViewById(R.id.vehicle_type);

vehicle_type.setText(storePreference.getString("vehicle_t
ype"));

        fuel_per_km = findViewById(R.id.fuel_per_km);

fuel_per_km.setText(storePreference.getString("fuel_per_k
m"));

        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        if (internetValidation()) {
            new Thread(new LoadAmountTask()).start();
            showpDialog();
        }

    }

    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(ProfileActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    private class LoadAmountTask implements Runnable {


        LoadAmountTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(amountUrl); // here is
your URL path

                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("accountNumber",
storePreference.getString("accountNumber"));

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                amountJSON = new
JSONObject(serverResponse); /*
{"success":"true","info":[{"amount":"465"}]}*/

                success = amountJSON
                        .getString("success");
                if (success.equals("true")) {
                    JSONArray loginArray = amountJSON
                            .getJSONArray("info");

                    JSONObject eachObjFromJSONArray =
loginArray
                            .getJSONObject(0);
                    newAmount = eachObjFromJSONArray
                            .getString("amount");

                }
```

```
                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(ProfileActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        amount.setText("BDT "+newAmount);
                    }
                }
            });
        }

    }

}
```

Code 6.2 Java Code of My Account

6.6 PHP Implementation for My Account

The request data From PHP end it can get data from Android through API, store data into database and response a message to Android.

- My Account API

```
function amount() {
        $params['accountNumber'] = $this->input-
>post('accountNumber', TRUE);


        $res = array();

        $result = $this->Api_login_model->getAmount($params);

        /*var_dump($result);
        die();*/

        if (!$result):
            $info = "Incorrect Login Details";
            $success = "false";            }
                else:
            $res = $result->row();
            $info[] = $res;
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );

        echo json_encode($json);
    }
```

Code 6.3 PHP Code of My Account

- My Account database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	register_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	user_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	password	varchar(32)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	monbile_no	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
7	vehicle_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
8	vehicle_name	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
9	fuel_per_km	int(11)			No	None			Change Drop Primary Unique Index More
10	accountNumber	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
11	amount	int(20)			No	None			Change Drop Primary Unique Index More
12	active	int(11)			Yes	1			Change Drop Primary Unique Index More
13	create_dt_tm	date			No	None			Change Drop Primary Unique Index More
14	update_dt_tm	date			No	None			Change Drop Primary Unique Index More

ADD MONEY

Add money request and previous history

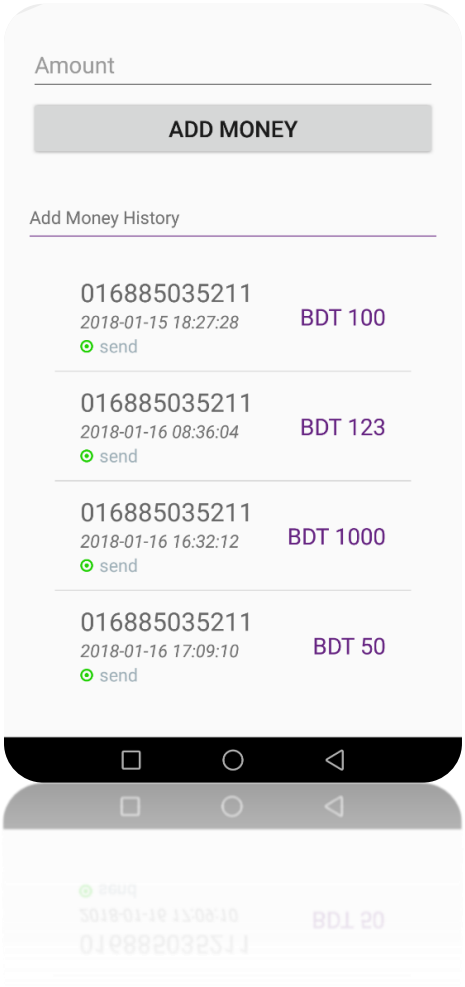


Figure 7.1: Add Money Interface

7.1 Introduction:

It is a page where you can put the amount that you want to enter in your account and see the previous history of your add money request.

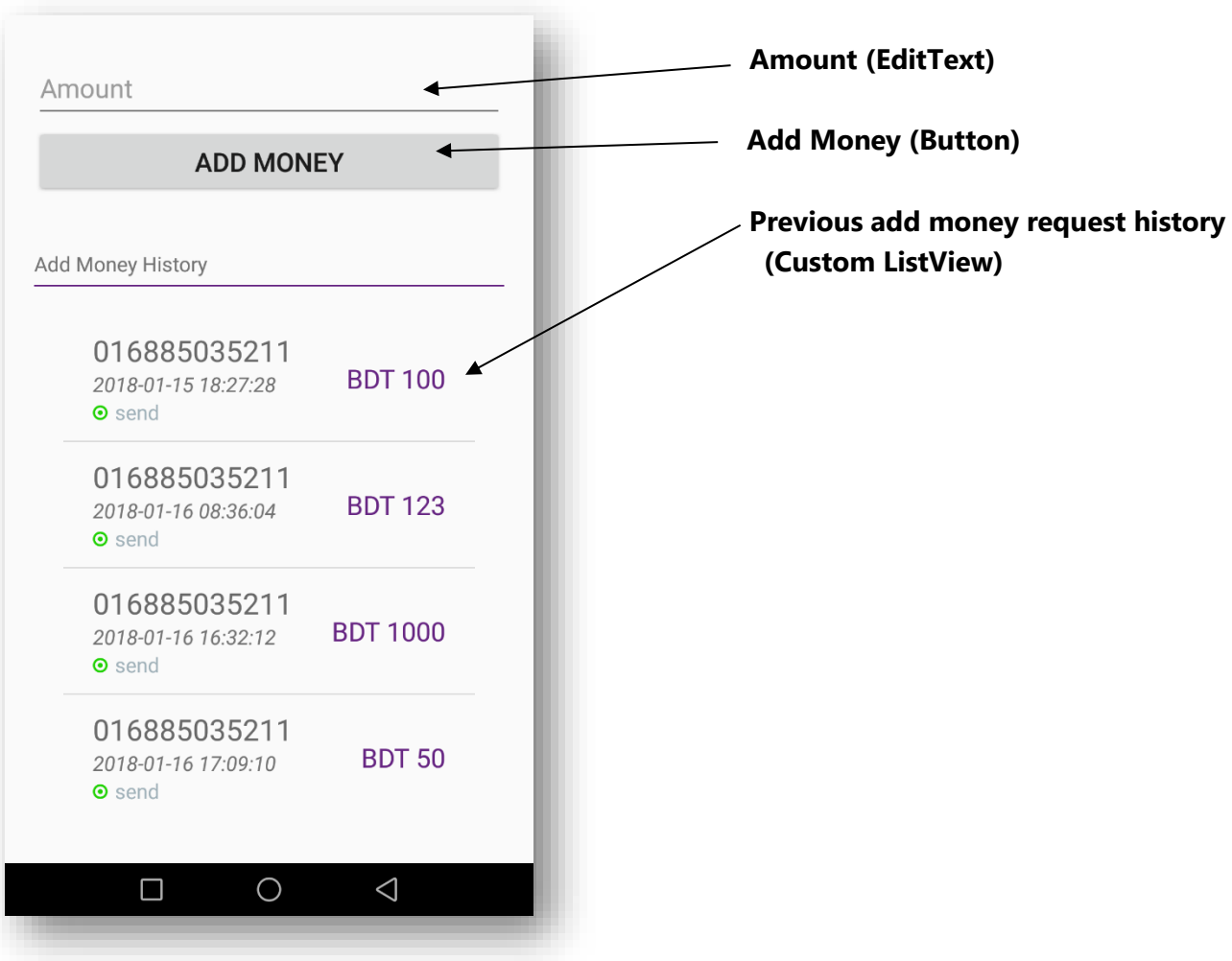


Figure 7.1: Add Money Interface

7.2 Data Flow Diagram for Add Money

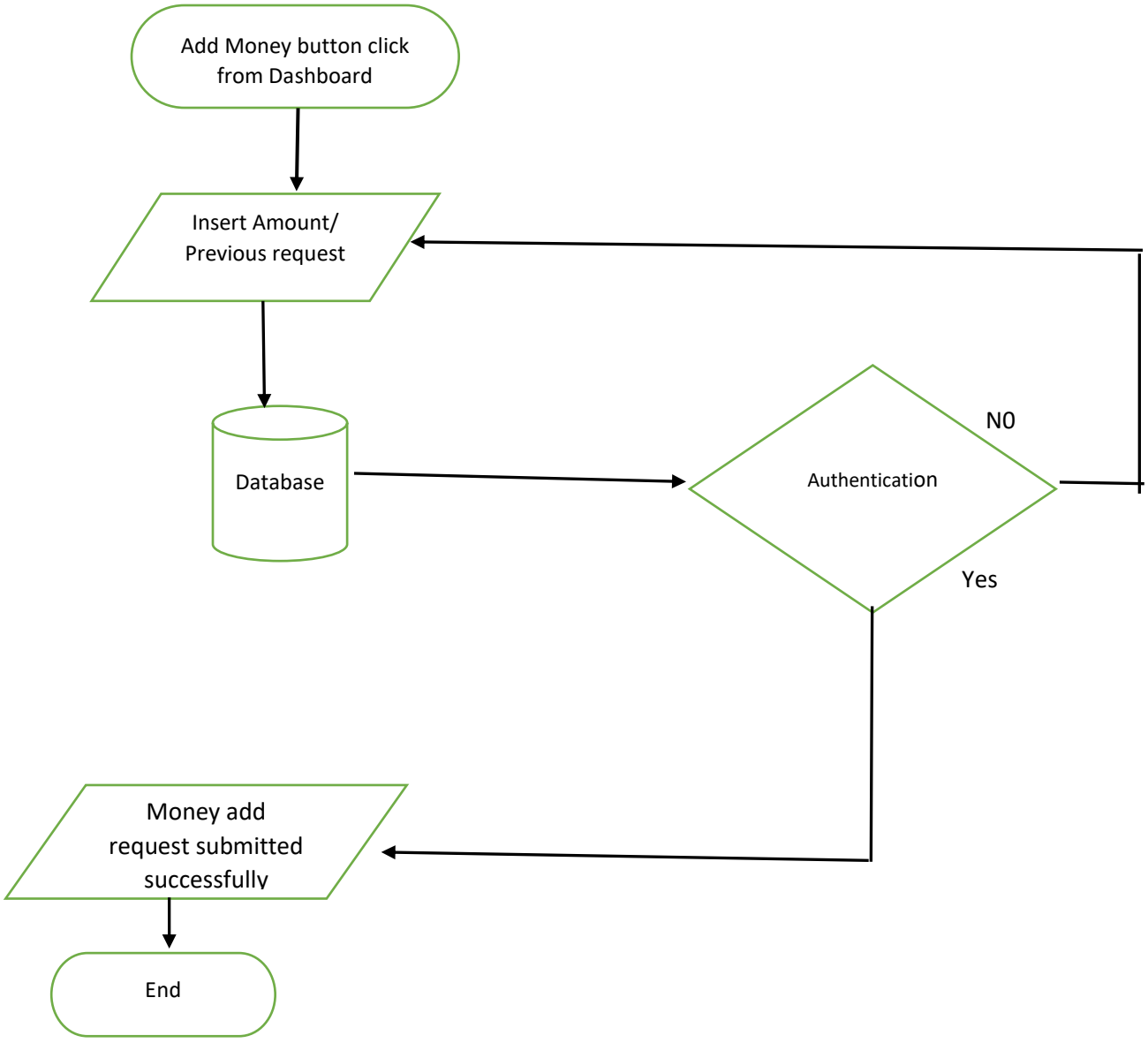


Figure 7.2: Flowchart of Add Money

7.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with EditText, Custom ListView, and Button. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

7.4 Add Money Request:

- This is the app interface with EditText for enter the amount of money and a ListView for Previous add money request.

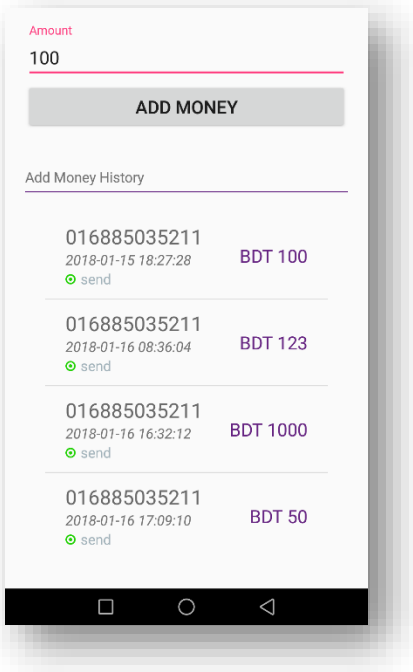
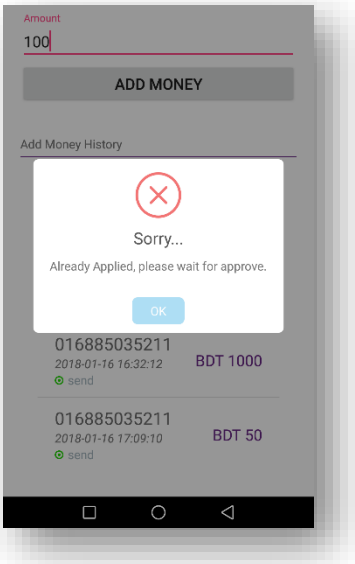
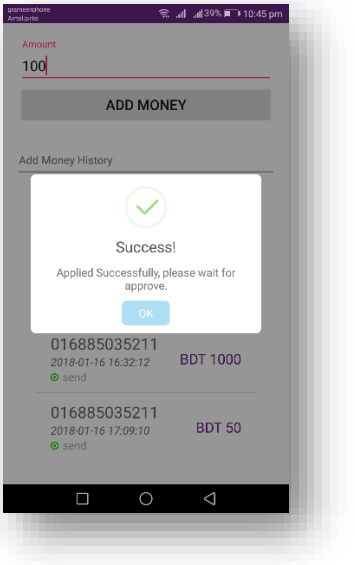


Figure 7.1: Add Money Interface

- For incorrect request



- When the request is successful then the popup screen will be confirmed.



7.5 Building the app GUI

In this section, you'll build the GUI for the **Add Money**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_add_money.xml file

You'll add a EditText, Button and ListView under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:padding="@dimen/padding_list">

    <LinearLayout
        android:id="@+id/main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView2"
        android:orientation="vertical">

        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <EditText
                android:id="@+id/etAmount"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Amount"
                android:inputType="textEmailAddress"
                android:textSize="@dimen/text_medium" />

        </android.support.design.widget.TextInputLayout>

        <Button
            android:id="@+id/btnAddMoney"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="6dp"
            android:layout_marginTop="2dp"
            android:onClick="addMoney"
            android:text="Add Money"
            android:textSize="@dimen/text_medium" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginTop="30dp"
            android:text="Add Money History" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginTop="5dp"
            android:background="@color/colorPrimary" />

        <ListView
            android:id="@+id/listView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="20dp"
            android:scrollbars="none" />
    </LinearLayout>

</RelativeLayout>
```

Adding the Components in activity_add_money.xml file for custom listView item

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:paddingLeft="20dp"
    android:paddingBottom="10dp"
    android:paddingRight="20dp"
    android:paddingTop="10dp"
    android:background="@drawable/ripple"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/accountNumber"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textColor="#94000000"
        android:layout_weight="1"
        android:text="TextView" />

    <TextView
        android:id="@+id/date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/accountNumber"
        android:textStyle="italic"
        android:text="TextView" />

    <TextView
        android:id="@+id/status"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/date"
        android:layout_toRightOf="@+id/imageView"
        android:layout_marginLeft="5dp"
        android:textColor="#ae78909c"
        android:text="Mobile" />
```

```
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="10dp"
        android:layout_height="10dp"
        android:layout_marginTop="5dp"
        android:layout_below="@+id/date"
        app:srcCompat="@drawable/error_circle" />

    <TextView
        android:id="@+id/amount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:textColor="@color/colorPrimary"
        android:layout_centerVertical="true"
        android:textSize="@dimen/text_medium"
        android:text="BDT 0"/>
</RelativeLayout>
```

7.6 Java Implementation for Add Money

Among the variable EditText into which input the amount of money, one is Button for adding money, ListView for previous add amount history with status.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are two threads to communicate with server to get data, one for add money thread which can communicate to server and send add money request with response. There are a few data get from server like *'accountNumber', 'amount', 'dateTime' and 'status'*.

On other thread request data from server and get a list from server and put it on arrayList for showing a list otherwise server response false data if any error occur.

```
public class AddMoneyActivity extends Activity {

    String addMoneyUrl = "", addMoneyHistoryUrl = "",
serverResponse = "", Result = "", success = "", msg = "",
            accountNumber = "", amount = "", status = "",
dateTime = "";

    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    JSONObject addMoneyInfoJSON = null;
    JSONObject addMoneyHistoryInfoJSON = null;
    HttpConnectionClass httpClass;
    SharedPreferencesClass storePreference;
    EditText etAmount;
    ArrayList<AddMoneyHistoryHelper> listAddMoneyHistoryDetail
= new ArrayList<AddMoneyHistoryHelper>();
    ListView lv;
    private AddMoneyHistoryAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_money);
        initialize();
    }

    public void initialize() {
        addMoneyUrl = getString(R.string.server_address)
                + "api/api_add_money/addmoney";
        addMoneyHistoryUrl = getString(R.string.server_address)
                + "api/api_add_money/add_money_history";
        httpClass = new HttpConnectionClass(this);
        etAmount = findViewById(R.id.etAmount);
        lv = (ListView) findViewById(R.id.listView);

        storePreference = new
SharedPreferencesClass(getApplicationContext());
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);

        addMoneyHistory();
    }

    public void addMoneyHistory() {
        if (internetValidation()) {
            new Thread(new LoadMoneyHistoryTask()).start();
            showpDialog();
        }
```

```
    }


    public void addMoney(View v) {
        if (internetValidation()) {
            new Thread(new LoadAddMoneyTask()).start();
            showpDialog();
        }
    }

    private class LoadMoneyHistoryTask implements Runnable {


        LoadMoneyHistoryTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(addMoneyHistoryUrl); // here
is your URL path


                JSONObject postDataParams = new JSONObject();
                postDataParams.put("accountNumber",
storePreference.getString("accountNumber"));

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                addMoneyHistoryInfoJSON = new
JSONObject(serverResponse);
                success = addMoneyHistoryInfoJSON
                        .getString("success");
                if (success.equals("true")) {
                    JSONArray eachObjFromJSONArray =
addMoneyHistoryInfoJSON
                            .getJSONArray("info");

                    listAddMoneyHistoryDetail.clear();

                    for (int i = 0; i <
eachObjFromJSONArray.length(); i++) {
                        JSONObject eachObjFromJSONOb =
eachObjFromJSONArray.getJSONObject(i);

                        accountNumber = eachObjFromJSONOb
                                .getString("accountNumber");
                        amount = eachObjFromJSONOb
                                .getString("amount");
                        status = eachObjFromJSONOb
                                .getString("status");
                        dateTime = eachObjFromJSONOb
                                .getString("dateTime");

                        listAddMoneyHistoryDetail.add(new
AddMoneyHistoryHelper(
                                accountNumber, amount, status,
dateTime));


                    }
                }
                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else if
(listAddMoneyHistoryDetail.isEmpty()) {
                        new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))
                                .setContentText("No add money
request found yet.")
                                .show();
                    } else {
                        if (success.equals("true")) {
                            mAdapter = new
AddMoneyHistoryAdapter(getBaseContext(),
                                    listAddMoneyHistoryDetail);
                            lv.setAdapter(mAdapter);
                        } else {
                            new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))
                                    .setContentText("No data
found")
                                    .show();
                        }
                    }
                }
            });
        }

    }


    private class LoadAddMoneyTask implements Runnable {


        LoadAddMoneyTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(addMoneyUrl); // here is your
URL path

                JSONObject postDataParams = new JSONObject();
                postDataParams.put("amount",
etAmount.getText().toString());
                postDataParams.put("accountNumber",
storePreference.getString("accountNumber"));
                // Log.e("params", postDataParams.toString());

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                addMoneyInfoJSON = new
JSONObject(serverResponse);


                success = addMoneyInfoJSON
                        .getString("success");
                msg = addMoneyInfoJSON
                        .getString("msg");
                msg = msg.replaceAll("\\<.*?\\>", "");
```

```
                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    }  else {
                        if (success.equals("true")) {
                            addMoneyHistory();
                            new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.SUCCESS_TYPE)
                                    .setTitleText("Success!")
                                    .setContentText(msg)
                                    .show();
                        } else {
                            new
SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))
                                    .setContentText(msg)
                                    .show();
                        }
                    }
                }
            });
        }

    }


    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(AddMoneyActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

}
```

Code 7.3 Java Code of Add Money

7.7 PHP Implementation for Add Money

The request data From PHP end it can get data from Android through API , store data into database and response a message to Android. Admin user can view a requested data who tries for add money request. They can confirm request after that data will be updated in database and user can view the request data with current balance.

- add money API

```
function addmoney() {
        $params['accountNumber'] = $this->input-
>post('accountNumber', TRUE);
        $params['amount'] = $this->input->post('amount', TRUE);
        $params['status'] = "pending";

        $date = date("Y-m-d H:i:s");
        $params['dateTime'] = $date;


        $this->load->library('form_validation');

        $this->form_validation->set_rules('accountNumber',
'Account Number', 'required|trim|callback_checkAccountNumber');


        if ($this->form_validation->run() == FALSE) {
            $info = "Already Applied, please wait for
approve.";
            $json = array(
                "success" => false,
                "msg" => validation_errors('<p>', '</p>'),
            );

            echo json_encode($json);
            die();
        }

    $res = array();

    $result = $this->Api_add_money_model->addMoneyInfo($params);

    $info = "Applied Successfully, please wait for approve.";
    $success = "true";
```

```
    $json = array(
       "success" => $success,
       "msg" => $info
    );
```

```
    echo json_encode($json);

}
```

- previous history of add money add money API

```
function add_money_history() {
        $params['accountNumber'] = $this->input-
>post('accountNumber', TRUE);


        $result = $this->Api_add_money_model-
>addMoneyHistoryInfo($params);

        $res=array();
        if($result){
            foreach($result->result() as $addmoney){

                $res["accountNumber"]=$addmoney->accountNumber;
                $res["amount"]=$addmoney->amount;
                $res["status"]=$addmoney->status;
                $res["dateTime"]=$addmoney->dateTime;
                $json[]=$res;
            }
```

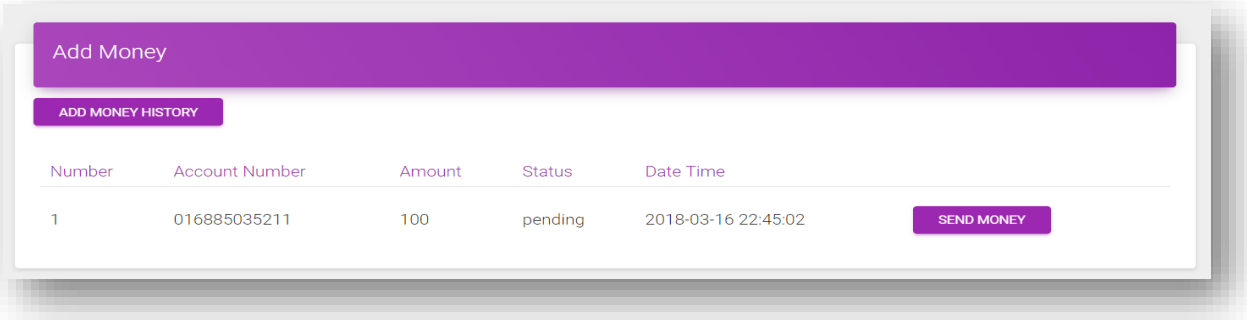
```
            echo json_encode(
                array(
                    "success"=>"true",
                    "info"=>$json
                )
            );
            die();
        }

        echo json_encode(
            array(
                "success"=>"false",
                "info"=>"There are no data found."
                )
            );
            die();

    }
```

Code 7.4 PHP Code of Add Money

- Add money database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary More
2	accountNumber	varchar(15)	latin1_swedish_ci		No	None			Change Drop Primary More
3	amount	int(20)			No	None			Change Drop Primary More
4	status	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary More
5	dateTime	datetime			No	None			Change Drop Primary More

- Add money request accept by admin screen



QR PAYMENT

Scan QR code and give payment

Figure 8.1: QR payment Interface

8.1 Introduction:

It is a QR code scanning page where you can scan QR code of the fuel station and through scanning you can pay your bill.

8.2 Data Flow Diagram for QR Payment

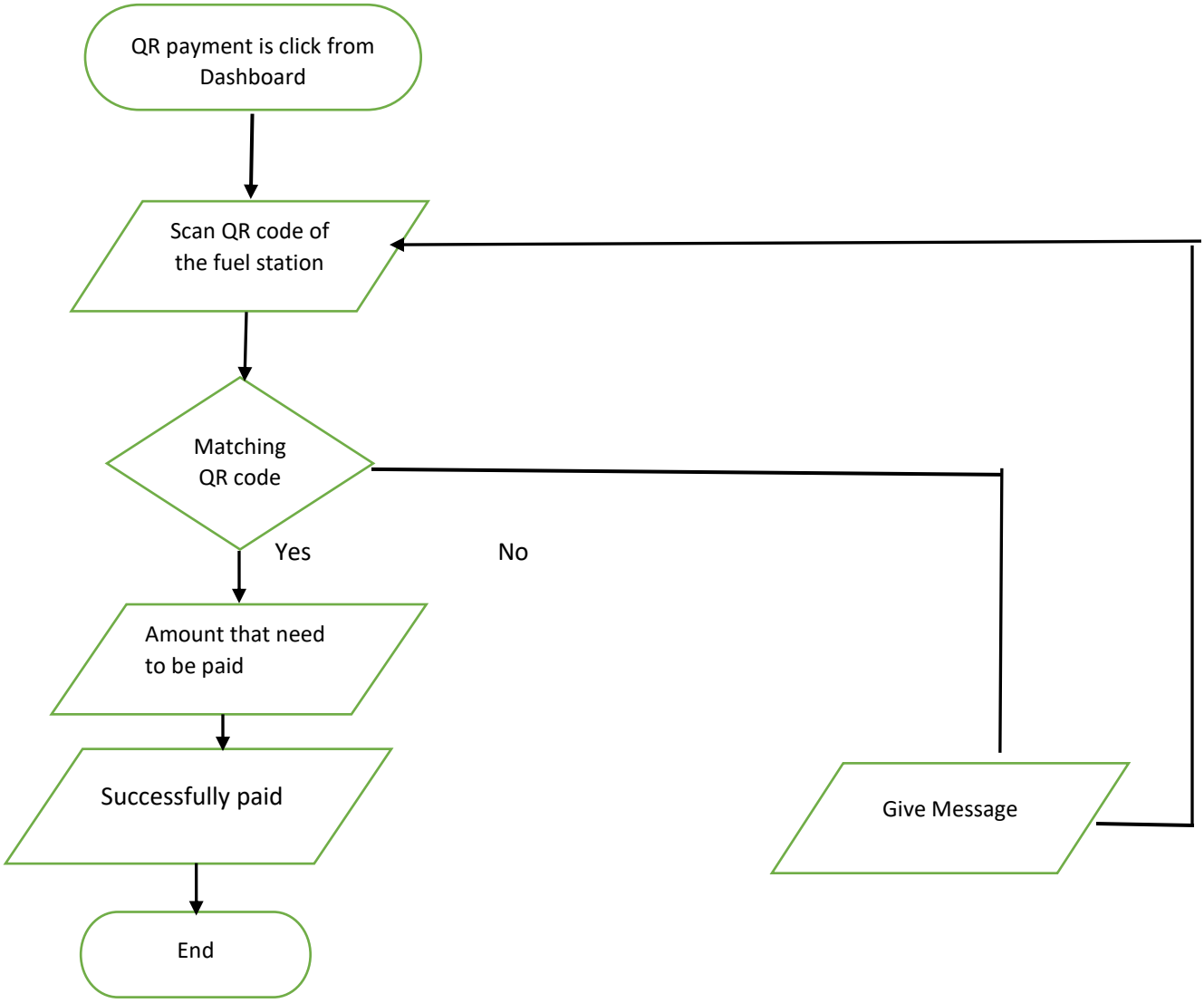


Figure 8.2: Flowchart of QR Payment

8.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses PHP and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with SurfaceView, ImageView and TextView. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

8.4 QR Code Scanning:

- This is the app interface with SurfaceView and ImageView for scanning QR code and TextView for showing value of the QR code.

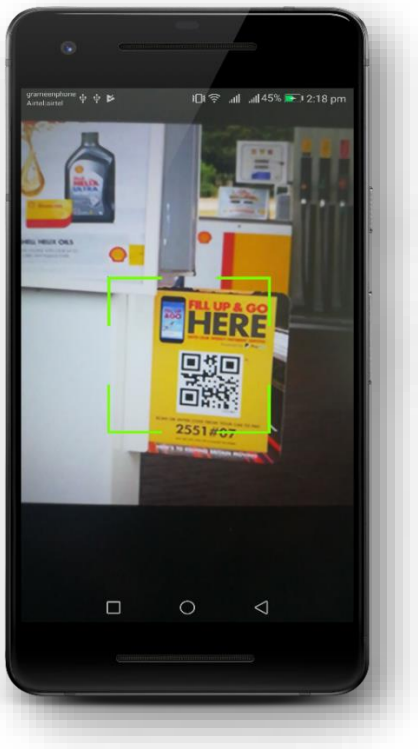


Figure 8.1: QR payment Interface

8.5 Building the app GUI

In this section, you'll build the GUI for the **QR payment**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_qrscan.xml file

You'll add a SurfaceView, ImageView and TextView under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

      <SurfaceView
        android:id="@+id/surfaceViewQR"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true" />
```

```
  <ImageView
        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:src="@drawable/cam"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />


  <TextView
        android:id="@+id/textViewQR"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="12dp"
        android:visibility="invisible"
        android:text="TextView" />


</RelativeLayout>
```

8.6 Java Implementation for QR Payment

Among the variable SurfaceView provides a dedicated drawing surface embedded inside of a view hierarchy, ImageView for capturing Qr code and TextView for showing value of QR code.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components.

```
public class QRScanActivity extends Activity{

    private static final int READ_REQUEST_CODE = 42;

    public String QRcodeText;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;
    private SurfaceView cameraView;
```

```
    private TextView barcodeInfo;

    public Uri QRImgURI;
    private static final String TAG = "MainActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_qrscan);
```

```
  /*      if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.M) {
            if
(checkSelfPermission(Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
new String[]{Manifest.permission.CAMERA}, 1);
                Log.d("QR","Cam Not Permission ");
            } else if
(checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED){
                ReadQR();
                Log.d("QR","Cam Permission ");
            }
        }

        if (Build.VERSION.SDK_INT <
Build.VERSION_CODES.M) {
            ReadQR();
        }
*/
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.KITKAT) {
            Window w = getWindow(); // in Activity's
onCreate() for instance

w.setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMI
TS, WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        }
        ReadQR();


    }


    public void ReadQR() {
        cameraView = (SurfaceView)
findViewById(R.id.surfaceViewQR);
        barcodeInfo = (TextView)
findViewById(R.id.textViewQR);

        barcodeDetector =
                new BarcodeDetector.Builder(this)

.setBarcodeFormats(Barcode.QR_CODE)   //QR code
//ALL_FORMATS
                        .build();


        cameraSource = new CameraSource
                .Builder(this, barcodeDetector)
                .setRequestedPreviewSize(1920, 1080)
                .setAutoFocusEnabled(true)
                .setRequestedFps(15.0f)
                .build();


        cameraView.getHolder().addCallback(new
SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder
holder) {
                try {
                    if
(ActivityCompat.checkSelfPermission(getApplicationContext
(), Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
                        // TODO: Consider calling
                        //
ActivityCompat#requestPermissions
                        // here to request the missing
permissions, and then overriding
                        //   public void
onRequestPermissionsResult(int requestCode, String[]
permissions,
                        //
int[] grantResults)
                        // to handle the case where the
user grants the permission. See the documentation
                        // for
ActivityCompat#requestPermissions for more details.
                        return;
                    }

cameraSource.start(cameraView.getHolder());

                } catch (IOException ie) {
                    Log.e("CAMERA SOURCE",
ie.getMessage());
                }
            }

            @Override
            public void surfaceChanged(SurfaceHolder
holder, int format, int width, int height) {

            }

            @Override
            public void surfaceDestroyed(SurfaceHolder
holder) {
            }
        });

        barcodeDetector.setProcessor(new
Detector.Processor<Barcode>() {
            @Override
            public void release() {

            }

            @Override
            public void
receiveDetections(Detector.Detections<Barcode>
detections) {
                final SparseArray<Barcode> barcodes =
detections.getDetectedItems();
                if (barcodes.size() != 0) {

                    barcodeInfo.post(new Runnable() {
// Use the post method of the TextView
                        public void run() {
                            QRcodeText =
barcodes.valueAt(0).displayValue;//20$$
                            Boolean QRDetector =
QRcodeText.contains("$$");
                            String qrstring =
QRcodeText.replace("$$","");//20

                            if (QRDetector) {
                                beepSound();
                                Intent rowntent = new
Intent(getBaseContext(),

PaymentActivity.class);

//rowntent.putExtra("qr_text_code", qrstring);

rowntent.putExtra("qr_text_code", qrstring);
                                startActivity(rowntent);

Toast.makeText(getBaseContext(),
                                        "QR Code:::::
```

```
"+QRcodeText, Toast.LENGTH_SHORT)
                                        .show();
                                cameraSource.release();

barcodeDetector.release();
                                finish();
                            }
                            else {

Toast.makeText(getBaseContext(),
                                        "QR Code::::: 
wrong  "+QRcodeText, Toast.LENGTH_SHORT)
                                        .show();
                            }

                        }
                    });
                }

            }
        });


    }

    protected void beepSound() {
        try {
            Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFI
CATION);
            Ringtone r =
RingtoneManager.getRingtone(getApplicationContext(),
notification);
            r.play();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

       /* cameraSource.release();
        barcodeDetector.release();*/


    }


    @Override
    public void onRequestPermissionsResult(int
requestCode,
                                           String
permissions[], int[] grantResults) {
        switch (requestCode) {
            case 1: {

                // If request is cancelled, the result
arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    finish();
                    Intent intent = new Intent(this,
QRScanActivity.class);
                    startActivity(intent);

                    ReadQR();
                    // permission was granted, yay! Do
the
                    // contacts-related task you need to
do.
                } else {

                    // permission denied, boo! Disable
the
                    // functionality that depends on this
permission.
                    Toast.makeText(QRScanActivity.this,
"Permission denied to read your External storage",
Toast.LENGTH_SHORT).show();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
{
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            cameraSource.release();
            barcodeDetector.release();
            finish();
            return true;
        }

        return super.onKeyDown(keyCode, event);
    }
}
```

Code 8.3 Java Code of QR Code Scanning

8.7 Building the app GUI for Payment

In this section, you'll build the GUI for the **Payment**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_payment.xml file

You'll add a TextView, EditText and Button under RelativeLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coordinatorLayout"
    android:layout_width="match_parent"
    android:padding="@dimen/padding_list"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:weightSum="1">



        <!--Start main body-->
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1">


            <ScrollView
                android:id="@+id/scrollView"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:scrollbars="none">

                <!--Start add account-->

                <RelativeLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:orientation="horizontal">

                    <TextView
                        android:id="@+id/textStationName"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                        android:text="Station Name"
                        android:gravity="center"

android:textColor="@color/colorPrimary"

android:textSize="@dimen/text_large"/>
                    <TextView
                        android:id="@+id/textaccount"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                        android:text="Account"

android:layout_below="@+id/textStationName"
                        android:gravity="center"
                        android:textColor="@color/bgTint"

android:textSize="@dimen/text_small"/>
                    <TextView
                        android:id="@+id/textlocation"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                        android:text="Location"

android:layout_below="@+id/textaccount"
                        android:gravity="center"
                        android:textColor="@color/bgTint"

android:textSize="@dimen/text_small"/>


<android.support.design.widget.TextInputLayout
                        android:id="@+id/input_acc_no"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@+id/textlocation"
                        android:layout_marginTop="10dp"
                        android:visibility="visible">

                        <EditText
                            android:id="@+id/acc_no"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                            android:hint="Account Number"
                            android:inputType="number"
                            android:singleLine="true"

android:textColor="@color/colorPrimary" />

</android.support.design.widget.TextInputLayout>



<android.support.design.widget.TextInputLayout
                        android:id="@+id/input_amount"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@+id/input_acc_no"
                        android:visibility="visible">

                        <EditText
                            android:id="@+id/amount"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                            android:hint="Amount"
                            android:inputType="number"
                            android:singleLine="true"

android:textColor="@color/colorPrimary"/>

</android.support.design.widget.TextInputLayout>
```

```
                    <Button
                        android:id="@+id/requestPayment"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:layout_below="@id/input_amount"
                        android:layout_marginTop="40dp"
                        android:text="Make Payment"
                        android:clickable="true"
                        android:onClick="requestPayment"
```

```
android:background="@color/colorPrimary"

android:textColor="@color/colorwhite"
                            />

                </RelativeLayout>

                <!--End main-->

            </ScrollView>
        </RelativeLayout>
        <!--End main body-->


    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

8.8 Java Implementation for Payment

Among the variable EditText into which input the amount that you have to pay, one is Button for payment, TextView for history.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are two threads to communicate with server to get data, one for payment thread which can communicate to server and send money to fuel station's account and subtract that money from your account. There are a few data get from server like *'stationName', 'location' and 'toAcc'*.

On other thread request data from server and get values from server and put it on TextView for showing values otherwise server response false data if any error occur.

```
public class PaymentActivity extends Activity {
    String paymenturl = "", stationUrl = "",
serverResponse = "", Result = "", msg = "", success1 = "",
success = "", location, toAcc, stationName;
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    JSONObject userLoginInfoJSON = null;
    JSONObject paymentInfoJSON = null;
```

```
    HttpConnectionClass httpClass;
    SharedPreferencesClass storePreference;
    EditText acc_no, amounttext;
    TextView textlocation, textStationName, textaccount;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_payment);
        initialize();
```

```
    }

    public void initialize() {
        stationUrl = getString(R.string.server_address)
                + "api/api_payment/station_info";
        paymenturl = getString(R.string.server_address)
                + "api/api_payment/payment";
        httpClass = new HttpConnectionClass(this);
        storePreference = new
SharedPreferencesClass(getApplicationContext());
        textStationName = (TextView)
findViewById(R.id.textStationName);
        textaccount = (TextView)
findViewById(R.id.textaccount);
        textlocation = (TextView)
findViewById(R.id.textlocation);

        acc_no = (EditText) findViewById(R.id.acc_no);
        amounttext = (EditText) findViewById(R.id.amount);
        String accNumbr =
storePreference.getString("accountNumber");
        acc_no.setText(accNumbr);
        acc_no.setKeyListener(null);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);

        if (internetValidation()) {
            new Thread(new LoadStationInfoTask()).start();
            showpDialog();
        }
    }

    public void requestPayment(View v) {
        if (internetValidation()) {
            new Thread(new LoadPaymentTask()).start();
            showpDialog();
        }
    }

    private class LoadPaymentTask implements Runnable {


        LoadPaymentTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(paymenturl); // here is
your URL path

                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("toAcc",
textaccount.getText().toString());
                postDataParams.put("fromAcc",
acc_no.getText().toString());
                postDataParams.put("amount",
amounttext.getText().toString());
                postDataParams.put("stationName",
textStationName.getText().toString());

                // Log.e("params",
postDataParams.toString());

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);
                paymentInfoJSON = new
JSONObject(serverResponse);

                    /*JSONObject eachObjFromJSONArray =
userLoginInfoJSON
                            .getJSONObject("otp");*/


                success1 = paymentInfoJSON
                        .getString("success");
                msg = paymentInfoJSON
                        .getString("msg");
                msg = msg.replaceAll("\\<.*?\\>", "");

                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success1.equals("true")) {
                            new
SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.SUCCESS_TYPE)

.setTitleText("Success!")
                                .setContentText(msg)

.setConfirmText("Done")

.setConfirmClickListener(new
SweetAlertDialog.OnSweetClickListener() {
                                    @Override
                                    public void
onClick(SweetAlertDialog sDialog) {

startActivity(new Intent(PaymentActivity.this,
HomeMainActivity.class));
                                        finish();
                                    }
                                })
                                .show();
                        } else {
                            new
SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                        }
                    }
                }
            });
        }
```

```
    }

    private class LoadStationInfoTask implements Runnable
{


        LoadStationInfoTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(stationUrl); // here is
your URL path

                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("station_id",
getIntent().getStringExtra("qr_text_code"));

                // Log.e("params",
postDataParams.toString());

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                userLoginInfoJSON = new
JSONObject(serverResponse);

                    /*JSONObject eachObjFromJSONArray =
userLoginInfoJSON
                            .getJSONObject("otp");*/


                success = userLoginInfoJSON
                        .getString("success");
                if (success.equals("true")) {
                    JSONArray loginArray =
userLoginInfoJSON
                            .getJSONArray("info");

                    JSONObject eachObjFromJSONArray =
loginArray
                            .getJSONObject(0);
                    stationName = eachObjFromJSONArray
                            .getString("name");
                    toAcc = eachObjFromJSONArray
                            .getString("account_no");
                    location = eachObjFromJSONArray
                            .getString("location");

                }

                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }
```

```
            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {

textStationName.setText(stationName);
                            textaccount.setText(toAcc);

textlocation.setText(location);
                        } else {
                            new
SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                    .show();
                        }
                    }
                }
            });
        }

    }


    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(PaymentActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;

    }
}
```

Code 8.4 Java Code of Payment

8.9 Java Script Implementation for QR Code Generator

QR code is generated with the help of JAVA Script (JS). When camera read the QR code of the fuel station it shows the value of that QR code like- 'station_id' with '$$'. This '$$' sign is unique here. It differentiates these QR codes from random QR codes.

- Generating QR code

```
<script>

        new QRCode("station_qr", {

            text: "<?php echo
$stationDetails['station_id'];?>$$",

            colorDark : "#000000",

            colorLight : "#ffffff"

        });

    </scrip
```

8.10 PHP Implementation for Payment

The request data From PHP end it can get data from Android through API, store data into database and response a message to Android. Admin user can view a requested data who tries for add money request. They can confirm request after that data will be updated in database and user can view the request data with current balance.

- information of station station_info API

```
function station_info() {
        $params['station_id'] = $this->input-
>post('station_id', TRUE);


        $this->load->library('form_validation');
       // $this->form_validation->CI = & $this;

        $this->form_validation->set_rules('station_id',
'Station Id', 'required|trim|callback_checkstationid');



        if ($this->form_validation->run() == FALSE) {
            $info = "Station Not Available, Please try again";
            $json = array(
                "success" => false,
                "msg" => $info
            );

            echo json_encode($json);
            die();
        }

        $res = array();

        $result = $this->Api_payment_model-
>getStationInfo($params);

        /*var_dump($result);
        die();*/

        if (!$result):
            $info = "Station Not Available, Please try again";
            $success = "false";
        else:
            $res = $result->row();
            $info[] = $res;
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );

        echo json_encode($json);


    }
```

- bill payment of fuel payment API

```
function station_info() {
        $params['station_id'] = $this->input->post('station_id', TRUE);


        $this->load->library('form_validation');
       // $this->form_validation->CI = & $this;

        $this->form_validation->set_rules('station_id', 'Station Id',
'required|trim|callback_checkstationid');



        if ($this->form_validation->run() == FALSE) {
            $info = "Station Not Available, Please try again";
            $json = array(
                "success" => false,
                "msg" => $info
            );

            echo json_encode($json);
            die();
        }
        $res = array();

        $result = $this->Api_payment_model->getStationInfo($params);

        /*var_dump($result);
        die();*/

        if (!$result):
            $info = "Station Not Available, Please try again";
            $success = "false";
        else:
            $res = $result->row();
            $info[] = $res;
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );

        echo json_encode($json);
    }
```

Code 8.5 PHP Code of Paymeny

- station_setup database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary Unique More
5	start_time	time			No	None			Change Drop Primary Unique More
6	end_time	time			No	None			Change Drop Primary Unique More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	latitude	decimal(10,6)			No	None			Change Drop Primary Unique More
9	longitude	decimal(10,6)			No	None			Change Drop Primary Unique More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
11	amount	int(20)			No	None			Change Drop Primary Unique More
12	traffic	int(11)			No	None			Change Drop Primary Unique More
13	create_dateTime	datetime			No	None			Change Drop Primary Unique More
14	update_dateTime	datetime			No	None			Change Drop Primary Unique More

Transaction

Payment Transaction History

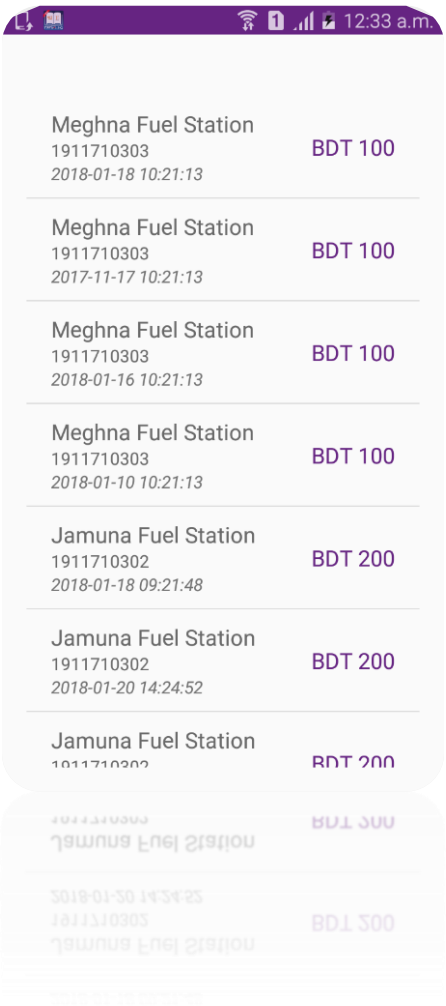


Figure 9.1: Transaction Interface

9.1 Introduction:

It is a page where you can see in which fuel station how much money you pay and also can see the time and date of your payment.

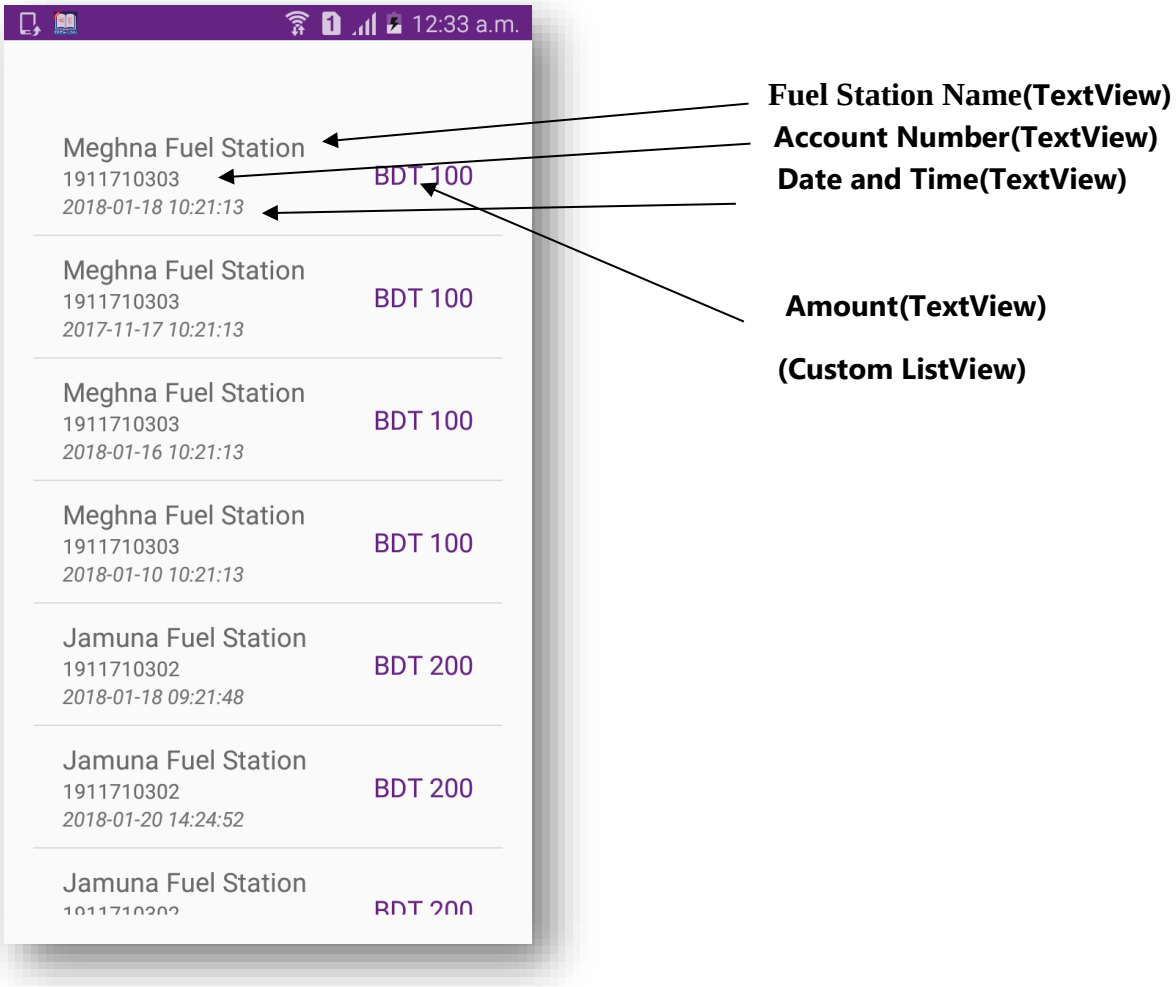


Figure 9.1: Transaction Interface

9.2 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with TextView and Custom ListView. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

9.3 Interface of Transaction:

- This is the app interface with TextView for watching the transaction history.

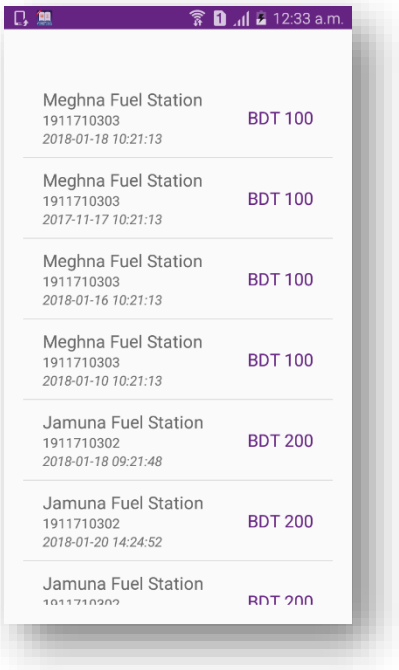


Figure 9.1: Transaction Interface

9.4 Building the app GUI

In this section, you'll build the GUI for the **Transaction**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_.xml file

You'll add a TextView, EditText and ListView under LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

tools:context="project.afinal.fuelpay.StationAreaSearchActivity
">


    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        android:text="Search your area:"/>

    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp">

        <EditText
            android:id="@+id/search"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Search here..."
            android:inputType="textEmailAddress"
            android:textSize="@dimen/text_medium" />

    </android.support.design.widget.TextInputLayout>

    <ListView
        android:id="@+id/listView"
        android:padding="20dp"
        android:layout_width="match_parent"
        android:scrollbars="none"
        android:layout_height="wrap_content" /></LinearLayout>
```

Adding the Components in activity_station_search_area.xml file for custom listView item

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:paddingLeft="20dp"
    android:paddingBottom="10dp"
    android:paddingRight="20dp"
    android:paddingTop="10dp"
    android:background="@drawable/ripple"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/stationName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="@dimen/text_medium"
        android:textColor="#94000000"
        android:layout_weight="1"
        android:text="TextView" />

    <TextView
        android:id="@+id/toAcc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="@dimen/text_small"
        android:layout_below="@+id/stationName"
        android:textColor="#94000000"
        android:layout_weight="1"
        android:text="TextView" />

    <TextView
        android:id="@+id/create_dateTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/toAcc"
        android:textSize="@dimen/text_small"
        android:textStyle="italic"
        android:text="TextView" />

    <TextView
        android:id="@+id/amount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
```

```
        android:textColor="@color/colorPrimary"
        android:layout_centerVertical="true"
        android:textSize="@dimen/text_medium"
        android:text="BDT 1000.00"/>
```

```
</RelativeLayout>
```

9.5 Java Implementation for Transaction

Among the variable ListView is for showing current rate of fuels.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are one threads to communicate with server to get data. This thread request data from server and get a list from server and put it on arrayList for showing a list otherwise server response false data if any error occurs. There are a few data get from server like *'stationName', 'amount', 'create_dateTime' and 'toAcc'*.

```
public class TransactionActivity extends Activity {

    ListView lv;
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    JSONObject transactionListInfoJSON = null;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    HttpConnectionClass httpClass;
    String transactionListUrl, serverResponse, Result,
create_dateTime,stationName, amount, toAcc, success;
    ArrayList<TransactionsHelper> listTransactionDetail =
new ArrayList<TransactionsHelper>();
    private TransactionAdapter mAdapter;
    SharedPreferencesClass storePreference;
    EditText search;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

setContentView(R.layout.activity_station_area_search);
        initialize();
        // listViwePopulate();
    }

    public void initialize() {
        transactionListUrl =
```

```
getString(R.string.server_address)
                + "api/api_transaction/transaction";
        httpClass = new HttpConnectionClass(this);
        storePreference = new
SharedPreferencesClass(getApplicationContext());
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        lv = (ListView) findViewById(R.id.listView);
        search = (EditText) findViewById(R.id.search);
        search.setVisibility(View.GONE);

        if (internetValidation()) {
            new Thread(new
LoadTransactionListUrlTask()).start();
            showpDialog();
        }
    }

    private class LoadTransactionListUrlTask implements
Runnable {


        LoadTransactionListUrlTask() {
        }


        @Override
        public void run() {
```

```
            try {
                URL url = new URL(transactionListUrl); //
here is your URL path


                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("accountNumber",
storePreference.getString("accountNumber"));

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                transactionListInfoJSON = new
JSONObject(serverResponse);
                success = transactionListInfoJSON
                        .getString("success");
                listTransactionDetail.clear();
                if(success.equals("true")) {
                    JSONArray eachObjFromJSONArray =
transactionListInfoJSON
                            .getJSONArray("info");

                    for (int i = 0; i <
eachObjFromJSONArray.length(); i++) {
                        JSONObject eachObjFromJSONOb =
eachObjFromJSONArray.getJSONObject(i);

                        toAcc = eachObjFromJSONOb
                                .getString("toAcc");
                        amount = eachObjFromJSONOb
                                .getString("amount");
                        create_dateTime =
eachObjFromJSONOb

.getString("create_dateTime");
                        stationName = eachObjFromJSONOb

.getString("stationName");

                        listTransactionDetail.add(new
TransactionsHelper(
                                toAcc, amount,
create_dateTime, stationName));

                        //Log.i("each element",
locationName.toString() + location.toString());
                    }
                }

                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(TransactionActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if
(!listTransactionDetail.isEmpty()) {
                            mAdapter = new
TransactionAdapter(getBaseContext(),

listTransactionDetail);
                            lv.setAdapter(mAdapter);
                        } else {
                            new
SweetAlertDialog(TransactionActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))
                                    .setContentText("No
transaction yet")
                                    .show();
                        }
                    }
                }
            });
        }

    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new
SweetAlertDialog(TransactionActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }


    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }
}
```

Code 9.2 Java Code of Transaction

9.6 PHP Implementation for Transaction

The request data From PHP end it can get data from Android through API, store data into database and response a message to Android.

- Transaction API

```
function transaction() {
    $params['fromAcc'] = $this->input->post('accountNumber', TRUE);

    $result = $this->Api_transaction_model->getTransactionInfo($params);

    $res=array();
    if($result){
        foreach($result->result() as $station){

                                        $res["toAcc"]=$station->toAcc;
                                        $res["amount"]=$station->amount;

        $res["create_dateTime"]=$station->create_dateTime;
         $res["stationName"]=$station->stationName;

                                        $json[]=$res;
        }
        echo json_encode(
                                                array(

        "success"=>"true",
                                                        "info"=>$json
                                                )
        );
        die();
    }

    echo json_encode(
                                array(
                                        "success"=>"false",
                                        "info"=>"There are no data found."
                                        )
            );
        die();

}
```

Code 9.3 PHP Code of Transaction

- Transaction database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	transactionId	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	fromAcc	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	toAcc	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	amount	int(20)			No	None			Change Drop Primary Unique Index More
5	stationName	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	create_dateTime	datetime			No	None			Change Drop Primary Unique Index More

Station Traffic

Search Places and Find Filling station with vehicle queue

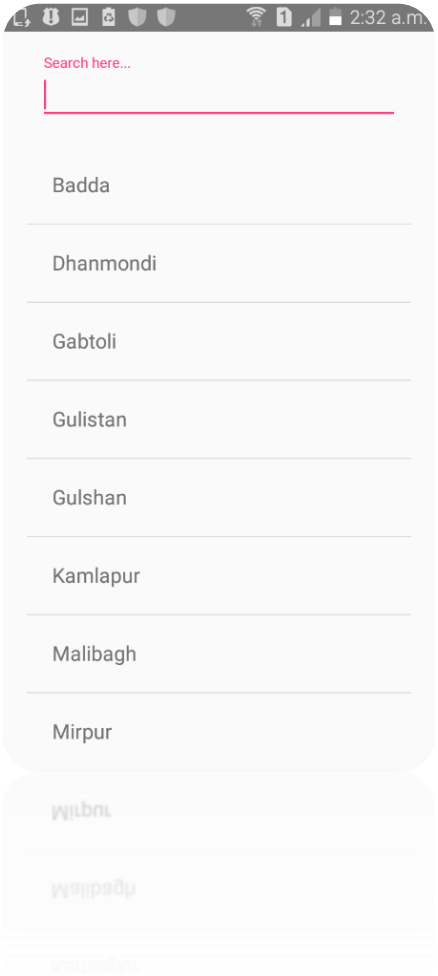


Figure 10.1: Station Trafic Interface

10.1 Introduction:

In this page you can see search fuel filling station by their place name and also how many vehicles is entered in a particular fuel station by vehicle queue number.

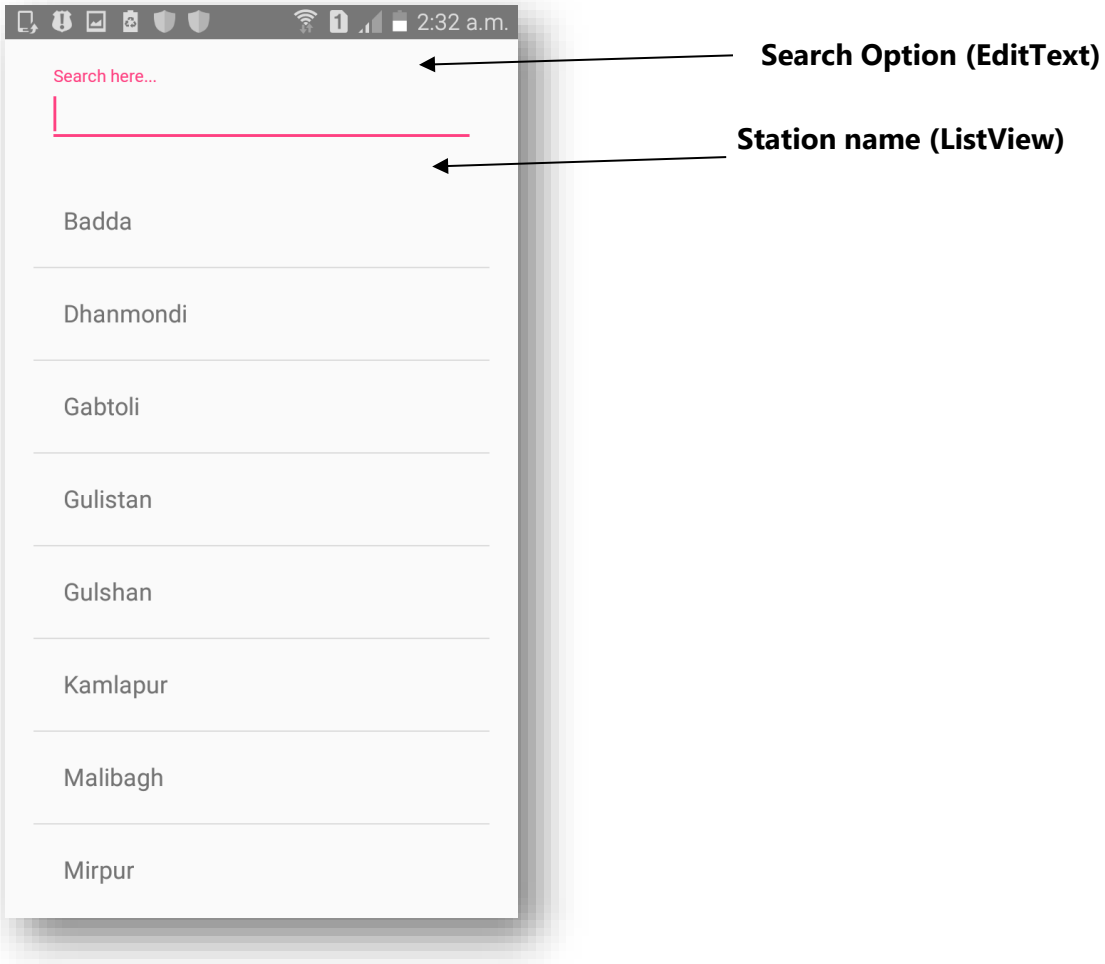


Figure 10.1: Station Trafic Interface

10.2 Data Flow Diagram for Station Traffic

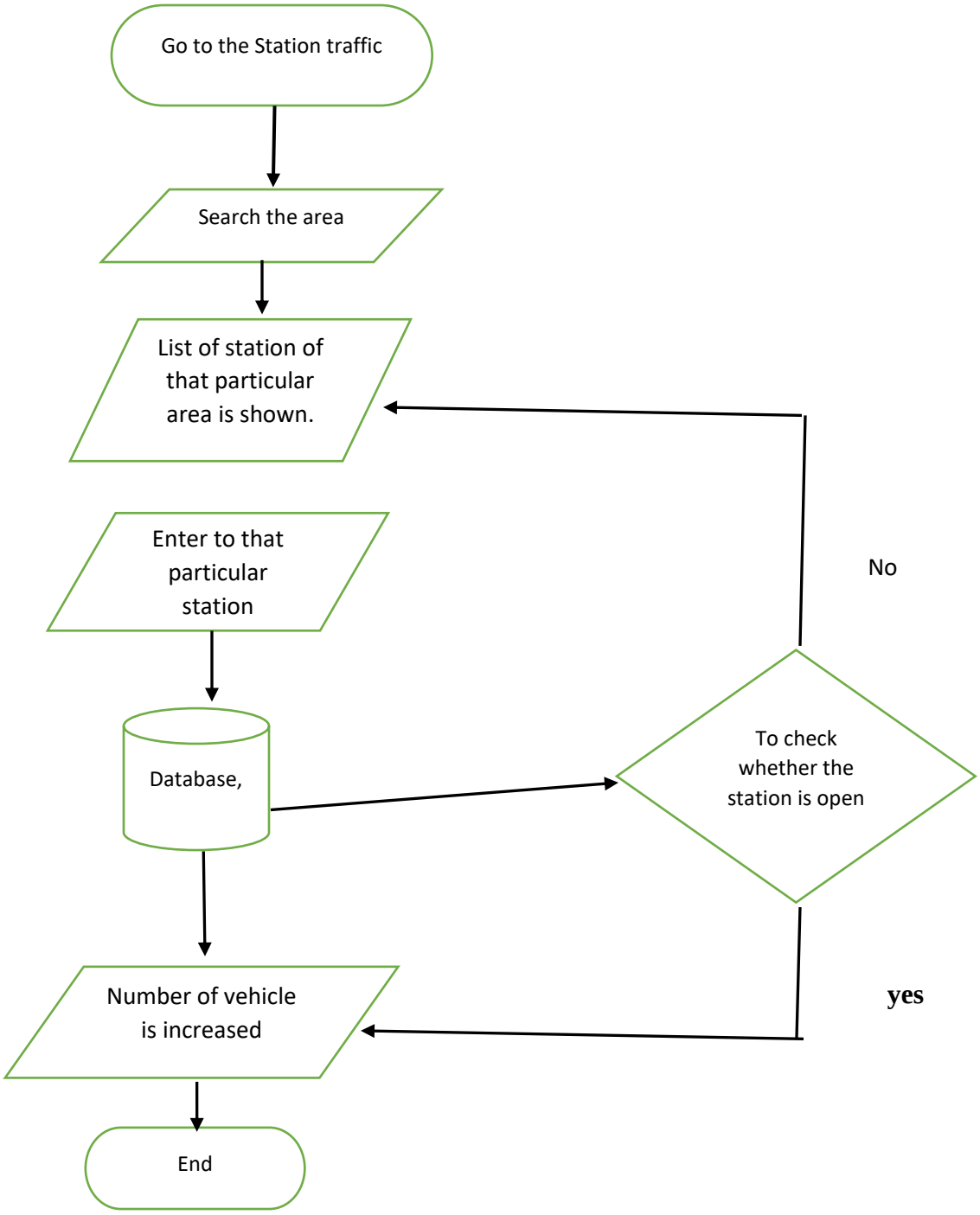


Figure 10.2: Flowchart of station traffic

10.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with EditText, Custom ListView, and Button. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

10.4 Intraface of Station Traffic:

- In this user can search the area and according that the fuel station list will be shown.

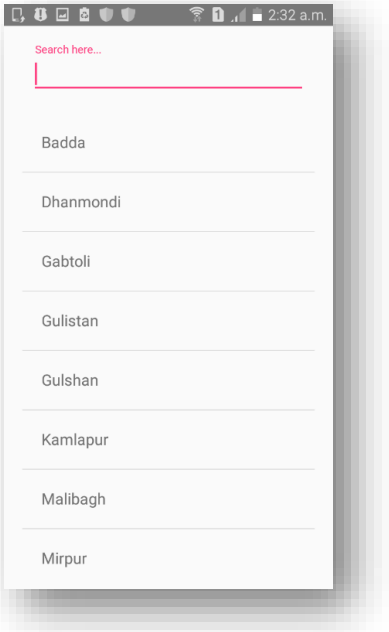
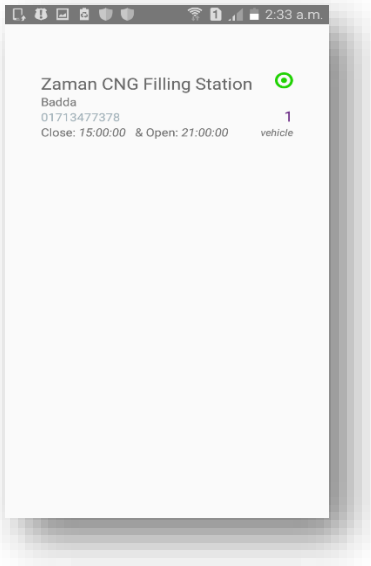
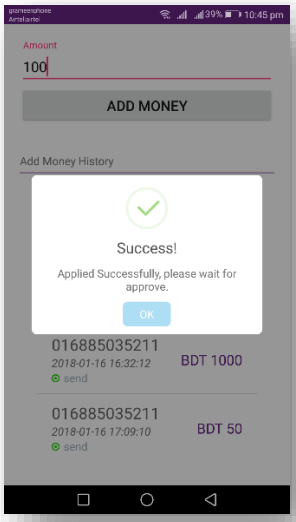


Figure 10.1: Station Trafic Interface



- List of station is shown by listview. Entry of every user in a station is increased the number of vehicle and the green light indicates that the station is open now.

- In the above one is green light and another is red light. Red light means this station is closed now.



10.5 Building the app GUI

In this section, we will build the GUI for the station traffic. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_station_area _search.xml file

We add EditText, ListView under RelativeLayout.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

tools:context="project.afinal.fuelpay.StationAreaSearchAc
tivity">


    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        android:text="Search your area:"/>

    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp">

        <EditText
            android:id="@+id/search"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Search here..."
            android:inputType="textEmailAddress"
            android:textSize="@dimen/text_medium" />

    </android.support.design.widget.TextInputLayout>

    <ListView
        android:id="@+id/listView"
        android:padding="20dp"
        android:layout_width="match_parent"
        android:scrollbars="none"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Adding the Components in activity_station_details_area_search.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="18dp"
        android:layout_marginRight="18dp"
        android:layout_marginTop="57dp"
        android:text="AutoCompleteTextView" />
</RelativeLayout>
```

10.6 Java Implementation for Station Traffic

Among the variable EditText into which search the area of station. ListView for display the station names.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There is thread which make sure that user enter to a station and traffic count increases.

```
public class TrafficAreaSearchDetailsActivity extends
Activity {

    ListView lv;
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    JSONObject detailsAreaListInfoJSON = null;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    HttpConnectionClass httpClass;
    String areaListUrl, serverResponse, Result, location,
locationName,stationId,entryUrl, open, close,
mobile,station_id,traffic,status,stationName,success;
    ArrayList<StationDetailsAreaHelper>
listStationDetailsAreaDetail = new
ArrayList<StationDetailsAreaHelper>();
    private StationDetailsAreaAdapter mAdapter;
    SharedPreferencesClass storePreference;
    EditText search;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

setContentView(R.layout.activity_station_area_search);
        initialize();
        listViewPopulate();
    }

    public void initialize() {
        areaListUrl = getString(R.string.server_address)
                +
"api/api_fuel_station/station_details_area";
        entryUrl = getString(R.string.server_address)
                + "api/api_fuel_station/station_entry";
        storePreference = new
SharedPreferencesClass(getApplicationContext());
        httpClass = new HttpConnectionClass(this);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        lv = (ListView) findViewById(R.id.listView);
        search = (EditText) findViewById(R.id.search);
        search.setVisibility(View.GONE);

        if (internetValidation()) {
            new Thread(new
LoadDetailsAreaListTask()).start();
            showpDialog();
        }
    }

    public void listViewPopulate() {
        lv.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent,
View view,
                                    int position, long id) {
                stationId =
listStationDetailsAreaDetail.get(position).getStationid();
                stationName =
listStationDetailsAreaDetail.get(position).getName();
                confirmation();

            }
        });
    }

    public void confirmation(){
        new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.SUCCESS_TYPE)
                .setTitleText("Are you sure?")
                .setContentText("Do you want to enter
"+stationName+" ?")
                .setConfirmText("Yes,enter!")
                .setConfirmClickListener(null)
                .setConfirmClickListener(new
SweetAlertDialog.OnSweetClickListener() {
                    @Override
                    public void onClick(SweetAlertDialog
sDialog) {
                        // reuse previous dialog instance
                        sDialog.setTitleText("Done!")
                                .setContentText("Your enter
the "+stationName)
                                .setConfirmText("OK")
                                .showCancelButton(false)

.setCancelClickListener(null)

.setConfirmClickListener(null)

.changeAlertType(SweetAlertDialog.SUCCESS_TYPE);
                        if (internetValidation()) {
                            new Thread(new
LoadTrafficEntryListTask()).start();
                            showpDialog();
                        }

storePreference.putString("trafficCount", "1");
                    }
                })
                .show();


    }
    private class LoadTrafficEntryListTask implements
Runnable {


        LoadTrafficEntryListTask() {
        }


        @Override
        public void run() {
```

```
            try {
                URL url = new URL(entryUrl); // here is your
URL path


                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("stationId", stationId);

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                detailsAreaListInfoJSON = new
JSONObject(serverResponse);
                success = detailsAreaListInfoJSON
                        .getString("success");

                Result = "";

            }  catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {
                            if (internetValidation()) {
                                new Thread(new
LoadDetailsAreaListTask()).start();
                                showpDialog();
                            }
                        } else {
                            new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))

.setContentText(getString(R.string.lofinFailMessage))
                                    .show();
                        }
                    }
                }
            });
        }

    }

    private class LoadDetailsAreaListTask implements
Runnable {


        LoadDetailsAreaListTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(areaListUrl); // here is
your URL path
```

```
                JSONObject postDataParams = new
JSONObject();
                postDataParams.put("location",
getIntent().getStringExtra("location"));

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                detailsAreaListInfoJSON = new
JSONObject(serverResponse);
                listStationDetailsAreaDetail.clear();
                success = detailsAreaListInfoJSON
                        .getString("success");

                JSONArray eachObjFromJSONArray =
detailsAreaListInfoJSON
                        .getJSONArray("info");

                for (int i = 0; i <
eachObjFromJSONArray.length(); i++) {
                    JSONObject eachObjFromJSONOb =
eachObjFromJSONArray.getJSONObject(i);

                    locationName = eachObjFromJSONOb
                            .getString("name");
                    location = eachObjFromJSONOb
                            .getString("location");
                    open = eachObjFromJSONOb
                            .getString("start_time");
                    close = eachObjFromJSONOb
                            .getString("end_time");
                    mobile = eachObjFromJSONOb
                            .getString("mobile_no");
                    status = eachObjFromJSONOb
                            .getString("is_active");
                    traffic = eachObjFromJSONOb
                            .getString("traffic");
                    station_id = eachObjFromJSONOb
                            .getString("station_id");
                    listStationDetailsAreaDetail.add(new
StationDetailsAreaHelper(
                            locationName, location, open,
close, mobile,status,traffic,station_id));

                    //Log.i("each element",
locationName.toString() + location.toString());
                }

                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {
                            mAdapter = new
StationDetailsAreaAdapter(getBaseContext(),

listStationDetailsAreaDetail);
                            lv.setAdapter(mAdapter);
                        } else {
```

```
                                new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))

.setContentText(getString(R.string.lofinFailMessage))
                                        .show();
                                }
                            }
                        }
                    });
            }

        }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new
SweetAlertDialog(TrafficAreaSearchDetailsActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                        .show();
                return false;
            }
            return true;
        }

        private void showpDialog() {
            if (!pDialog.isShowing())
                pDialog.show();
        }

        private void hidepDialog() {
            if (pDialog.isShowing())
                pDialog.dismiss();
        }
}
```

```
public class StationAreaSearchActivity extends Activity {

    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    JSONObject areaListInfoJSON = null;
    HttpConnectionClass httpClass;
    String areaListUrl, serverResponse, Result, location,
success, station_id;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    ArrayList<StationAreaHelper> listStationAreaDetail =
new ArrayList<StationAreaHelper>();
    private StationAreaAdapter mAdapter;
    ListView lv;
    EditText search;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

setContentView(R.layout.activity_station_area_search);
        initialize();
        listViwePopulate();
    }

    public void initialize() {
        areaListUrl = getString(R.string.server_address)
                + "api/api_fuel_station/station_area";
        httpClass = new HttpConnectionClass(this);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        lv = (ListView) findViewById(R.id.listView);
        search = (EditText) findViewById(R.id.search);

        search.addTextChangedListener(new TextWatcher() {

            @Override
            public void onTextChanged(CharSequence s, int
start, int before, int count) {
                mAdapter.getFilter().filter(s.toString());
            }
            @Override
            public void beforeTextChanged(CharSequence s,
int start, int count,
                                          int after) {
            }
            @Override
            public void afterTextChanged(Editable s) {
            }
        });
```

```
        if (internetValidation()) {
            new Thread(new LoadAreaListTask()).start();
            showpDialog();
        }
    }

    public void listViwePopulate() {
        lv.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent,
View view,
                                    int position, long id)
{

                String location =
listStationAreaDetail.get(position).getLocation();

                Bundle localBundle = new Bundle();
                localBundle.putString("location",
location);

                Intent rowntent = new
Intent(StationAreaSearchActivity.this,

StationDetailsAreaSearchActivity.class);
                rowntent.putExtras(localBundle);
                startActivity(rowntent);
            }
        });
    }

    private class LoadAreaListTask implements Runnable {


        LoadAreaListTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(areaListUrl); // here is
your URL path


                serverResponse =
httpClass.httpGetConnection(url);

                areaListInfoJSON = new
JSONObject(serverResponse);
```

```
                success = areaListInfoJSON
                        .getString("success");
                if(success.equals("true")) {
                    if(!listStationAreaDetail.isEmpty()){
                        listStationAreaDetail.clear();
                    }
                    JSONArray eachObjFromJSONArray =
areaListInfoJSON
                            .getJSONArray("info");

                    for (int i = 0; i <
eachObjFromJSONArray.length(); i++) {
                        JSONObject eachObjFromJSONOb =
eachObjFromJSONArray.getJSONObject(i);


                        station_id = eachObjFromJSONOb
                                .getString("station_id");
                        location = eachObjFromJSONOb
                                .getString("location");
                        listStationAreaDetail.add(new
StationAreaHelper(location, station_id));

                        Log.i("each element",
station_id.toString() + location.toString());
                    }
                }

                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(StationAreaSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else {
                        if (success.equals("true")) {
                            mAdapter = new
StationAreaAdapter(StationAreaSearchActivity.this,

listStationAreaDetail);
                            lv.setAdapter(mAdapter);
                        } else {
                            new
SweetAlertDialog(StationAreaSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.failHeader))

.setContentText(getString(R.string.failMessage))
                                    .show();
                        }
                    }
                }
            });
        }

    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new
SweetAlertDialog(StationAreaSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }


}
```

Code 10.3 Java Code of Station Traffic

10.7 PHP Implementation for Station traffic

- Fuel_station _ API_station_area

```
class Api_fuel_station extends CI_Controller {

function __construct(){
   parent::__construct();
   date_default_timezone_set('Asia/Dhaka');
   $this->load->model('Api_station_model');

   // if(!$this->session->userdata('validated')){
   //    redirect('Login');
   // }

}

function station_area() {

   $result = $this->Api_station_model->areaListInfo();
```

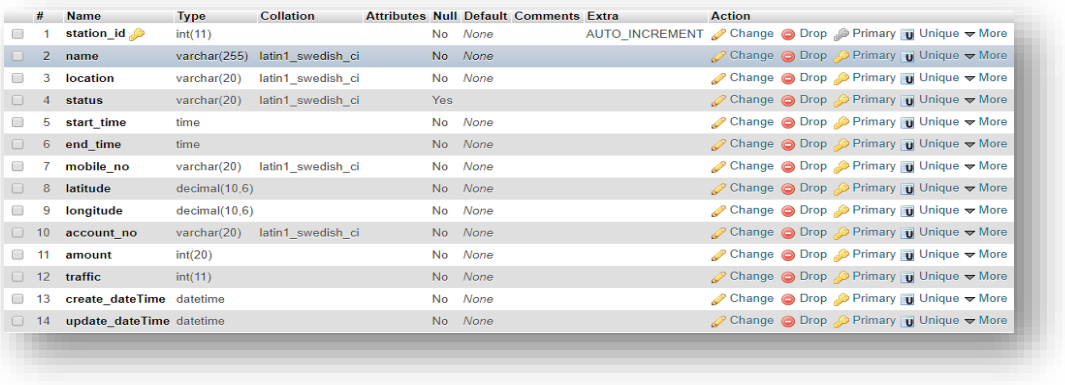
```
    if (!$result):
        $info = "No data found";
        $success = "false";
    else:
        $info = $result->result();
        $success = "true";
    endif;

    $json = array(
        "success" => $success,
        "info" => $info
    );

    echo json_encode($json);
}
```

- Fuel_station_ API_station_entry

```
function station_entry() {

    $params['stationId'] = $this->input->post('stationId', TRUE);

    $result = $this->Api_station_model->entryInfo($params);

    if (!$result):
        $info = "No data found";
        $success = "false";
    else:
        $info = $result->result();
        $success = "true";
    endif;

    $json = array(
        "success" => $success,
        "info" => $info
    );

    echo json_encode($json);
}

function station_details_area() {
    $params['location'] = $this->input->post('location', TRUE);
    //$params['active'] = $this->input->post('active', TRUE);
    $params['time'] = date('H:i:s');

    $result = $this->Api_station_model->areaDetailsListInfo($params);
    $dates=date("H:i:s");
    $res=array();
    if($result){
        foreach($result->result() as $station){
        if($station->start_time<$dates && $station->end_time>$dates){
                                                    $res["is_active"]="true";
                                                    }
                        else{
                        $res["is_active"]="false";
                                                    }
                                                    $res["start_time"]=$station-
>start_time;
                                                    $res["end_time"]=$station->end_time;
                                                    $res["name"]=$station->nam
$res["location"]=$station->location  $res["mobile_no"]=$station->mobile_no;
            $res["traffic"]=$station->traffic;
            $res["station_id"]=$station->station_id;
                                                    $json[]=$res;
            }
            echo json_encode(
                                                    array(
                                                            "success"=>"true",
                                                            "info"=>$json
                                                    )
            );

            die();
    }

    echo json_encode(
                                        array(
                                                    "success"=>"false",
                                                    "info"=>"There are no data found."
                                                    )
                );
                die();


}
```

Code 10.4 PHP Code of Station Traffic

- Station_setup database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary Unique More
5	start_time	time			No	None			Change Drop Primary Unique More
6	end_time	time			No	None			Change Drop Primary Unique More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	latitude	decimal(10,6)			No	None			Change Drop Primary Unique More
9	longitude	decimal(10,6)			No	None			Change Drop Primary Unique More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
11	amount	int(20)			No	None			Change Drop Primary Unique More
12	traffic	int(11)			No	None			Change Drop Primary Unique More
13	create_dateTime	datetime			No	None			Change Drop Primary Unique More
14	update_dateTime	datetime			No	None			Change Drop Primary Unique More

Expense

Transaction History in a day, month, week and year

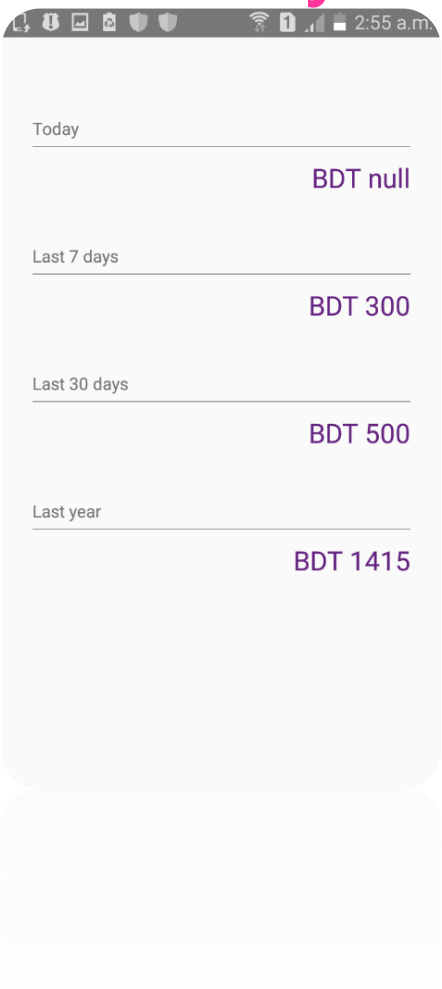


Figure 11.1 Expense Interface

11.1 Introduction:

It is an expense page where a user's transaction history is recorded.

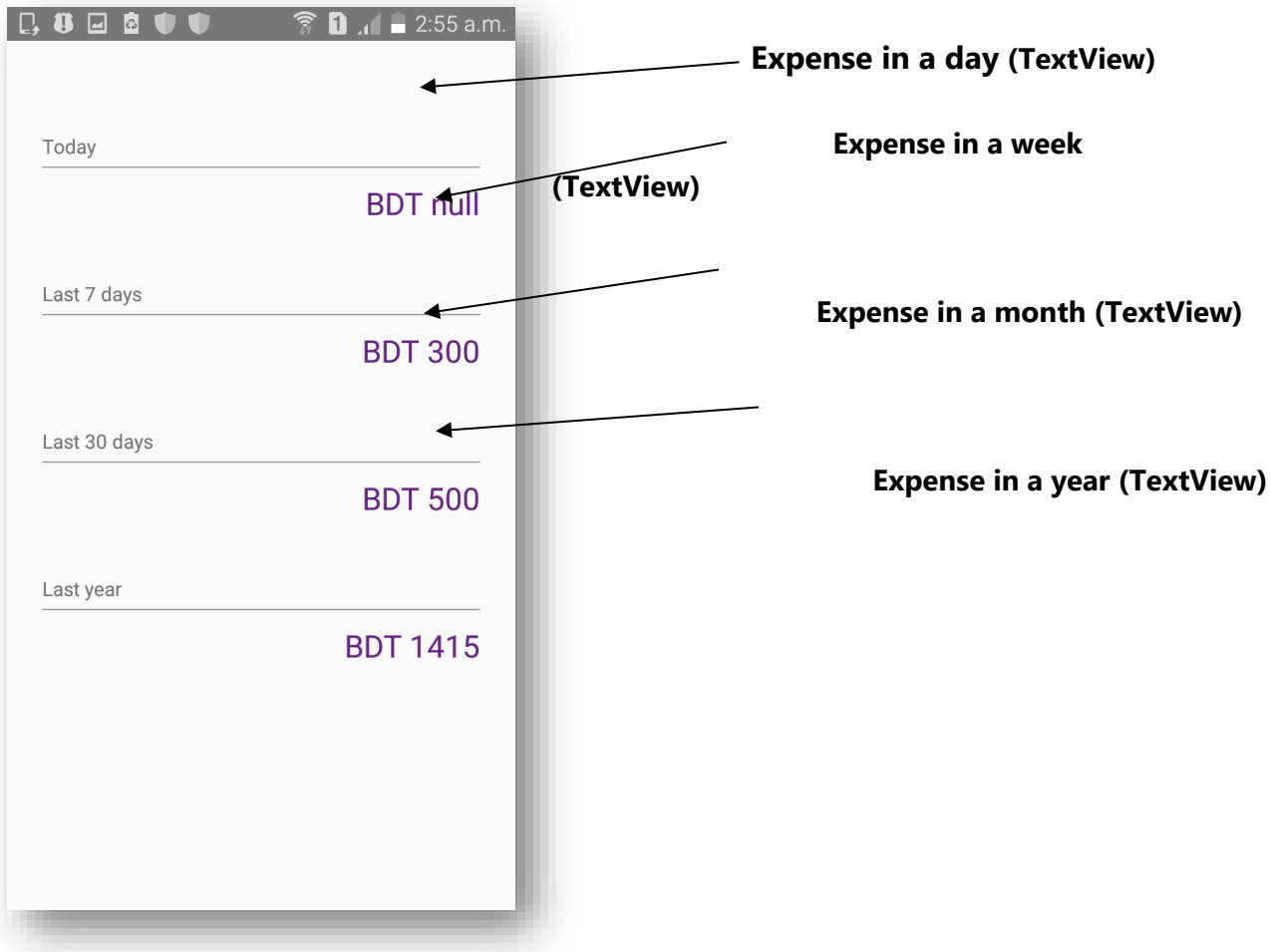


Figure 11.1 Expense Interface

11.2 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with EditText, Custom ListView, and Button. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

11.3 Interface of Expense

- This is the app interface with TextView And View for showing the history of expense of a user.

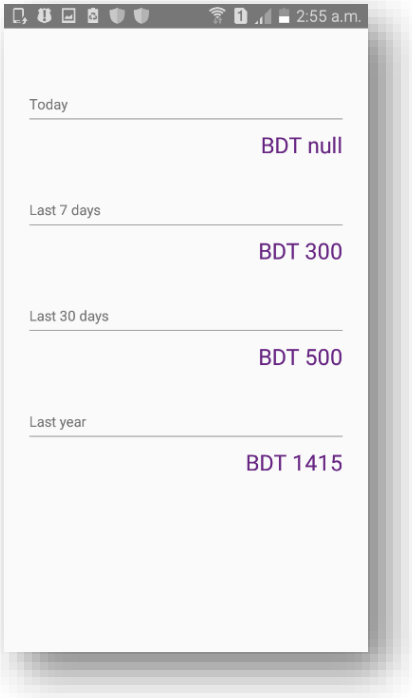


Figure 11.1 Expense Interface

11.4 Building the app GUI

In this section, you'll build the GUI for the **Expense**. For this you need cost calculator's xml. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_cost_calculator.xml file for TextView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:gravity="center">


    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView2">

        <LinearLayout
            android:id="@+id/ln"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="@dimen/padding_form">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Today"
                android:layout_marginTop="40dp"
                android:textSize="14sp"/>
            <View
                android:layout_width="wrap_content"
                android:layout_height="1dp"
                android:layout_marginTop="5dp"
                android:background="@color/colorgrey"/>
            <TextView
                android:id="@+id/today"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="BDT 0"
                android:layout_marginTop="10dp"
                android:textColor="@color/colorPrimary"
                android:gravity="end"
                android:textSize="@dimen/text_large"/>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Last 7 days"
                android:layout_marginTop="40dp"
                android:textSize="14sp"/>
            <View
                android:layout_width="wrap_content"
                android:layout_height="1dp"
                android:layout_marginTop="5dp"
                android:background="@color/colorgrey"/>
            <TextView
                android:id="@+id/week"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="BDT 00"
                android:layout_marginTop="10dp"
                android:textColor="@color/colorPrimary"
                android:gravity="end"
                android:textSize="@dimen/text_large"/>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Last 30 days"
                android:layout_marginTop="40dp"
                android:textSize="14sp"/>
            <View
                android:layout_width="wrap_content"
                android:layout_height="1dp"
                android:layout_marginTop="5dp"
                android:background="@color/colorgrey"/>
            <TextView
                android:id="@+id/month"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="BDT 000"
                android:layout_marginTop="10dp"
                android:textColor="@color/colorPrimary"
                android:gravity="end"
                android:textSize="@dimen/text_large"/>

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Last year"
                android:layout_marginTop="40dp"
                android:textSize="14sp"/>
            <View
                android:layout_width="wrap_content"
                android:layout_height="1dp"
                android:layout_marginTop="5dp"
                android:background="@color/colorgrey"/>
            <TextView
                android:id="@+id/year"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="BDT 0000"
                android:layout_marginTop="10dp"
                android:textColor="@color/colorPrimary"
                android:gravity="end"
                android:textSize="@dimen/text_large"/>
        </LinearLayout>
    </ScrollView>
</RelativeLayout>
```

11.5 Java Implementation for Expense

Among the variable Textview user see the text of today, week, month and year. In this module we need transaction's data to see the expense record.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

```
public class ExpenseActivity extends Activity {
    String calculateUrl, serverResponse, Result, success,
day,week,month,year;
    TextView todayText,weekText,monthText,yearText;
    private ProgressDialog pDialog;
    HttpConnectionClass httpClass;
    JSONObject calculateInfoJSON = null;
    private Handler handler = new Handler();
    SharedPreferencesClass storePreference;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_cost_calculator);

        initialize();

    }

    public void initialize() {
        calculateUrl = getString(R.string.server_address)
                + "api/api_transaction/cost_calculator";
        httpClass = new
HttpConnectionClass(ExpenseActivity.this);
        storePreference = new
SharedPreferencesClass(getApplicationContext());

        todayText = findViewById(R.id.today);
        weekText = findViewById(R.id.week);
        monthText = findViewById(R.id.month);
        yearText = findViewById(R.id.year);

        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        if (internetValidation()) {
            new Thread(new LoadcalCalculateTask()).start();
            showpDialog();
        }
    }


    private class LoadcalCalculateTask implements Runnable {


        LoadcalCalculateTask() {
        }


        @Override
        public void run() {
```

```
            try {
                URL url = new URL(calculateUrl); // here is
your URL path

                JSONObject postDataParams = new JSONObject();
                postDataParams.put("accountNumber",
storePreference.getString("accountNumber"));

                serverResponse =
httpClass.httpPostConnection(postDataParams, url);

                calculateInfoJSON = new
JSONObject(serverResponse);

                    /*JSONObject eachObjFromJSONArray =
userLoginInfoJSON
                            .getJSONObject("otp");*/


                success = calculateInfoJSON
                        .getString("success");
                if(success.equals("true")) {
                    JSONArray loginArray = calculateInfoJSON
                            .getJSONArray("info");

                    JSONObject eachObjFromJSONArray =
loginArray
                            .getJSONObject(0);
                    day = eachObjFromJSONArray
                            .getString("day");
                    week = eachObjFromJSONArray
                            .getString("week");
                    month = eachObjFromJSONArray
                            .getString("month");
                    year = eachObjFromJSONArray
                            .getString("year");

                }
                Result = "";

            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(ExpenseActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))
```

```
.setContentText(getString(R.string.errorMessage))
                                .show();
                    }  else {
                        if (success.equals("true")) {
                            todayText.setText("BDT "+day);
                            weekText.setText("BDT "+week);
                            monthText.setText("BDT "+month);
                            yearText.setText("BDT "+year);
                        } else {
                            new
SweetAlertDialog(ExpenseActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                    .show();
                        }
                    }
                }
            });
        }

    }
```

```
    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(ExpenseActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

    private void showpDialog() {
        if (!pDialog.isShowing())
            pDialog.show();
    }

    private void hidepDialog() {
        if (pDialog.isShowing())
            pDialog.dismiss();
    }
}
```

Code 11.2 Java Code of Expense

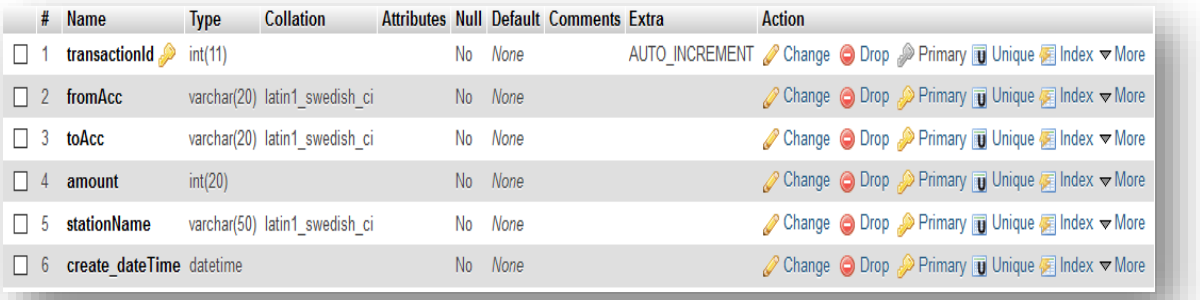
11.6 PHP Implementation for Expense

To get the expense we need transaction's cost calculator function and the it gives data to expense which makes the history.

```
function cost_calculator() {

    $params['fromAcc'] = $this->input->post('accountNumber', TRUE);
    $date = date("Y-m-d H:i:s");
    $params['toDateTime'] = $date;

    $fromdate = date("Y-m-d"). ' 00:00:00';
    $params['fromDateTime'] = $fromdate;

    /* echo $fromdate;
    die();*/


    $d365 = date('Y-m-d H:i:s', strtotime('-365 days'));
    $params['fromDateTime365'] = $d365;
    $d30 = date('Y-m-d H:i:s', strtotime('-30 days'));
    $params['fromDateTime30'] = $d30;
    $d7 = date('Y-m-d H:i:s', strtotime('-7 days'));
    $params['fromDateTime7'] = $d7;

    /*echo $date;
    echo $d2;
    die();*/

    $result = $this->Api_transaction_model->getCostCalculatorInfo($params);
    $json[]=$result;
    if ($result){
        echo json_encode(
            array(
                "success"=>true,
                "info"=>$json
            )
        );
        die();
    }

    echo json_encode(
        array(
            "success"=>false,
            "msg"=>"There are no data found."
        )
    );
```

Code 11.3 Java Code of Expense

- Transaction database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	transactionId	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	fromAcc	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	toAcc	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	amount	int(20)			No	None			Change Drop Primary Unique Index More
5	stationName	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	create_dateTime	datetime			No	None			Change Drop Primary Unique Index More

FUEL RATE

Current rate of different type of fuel

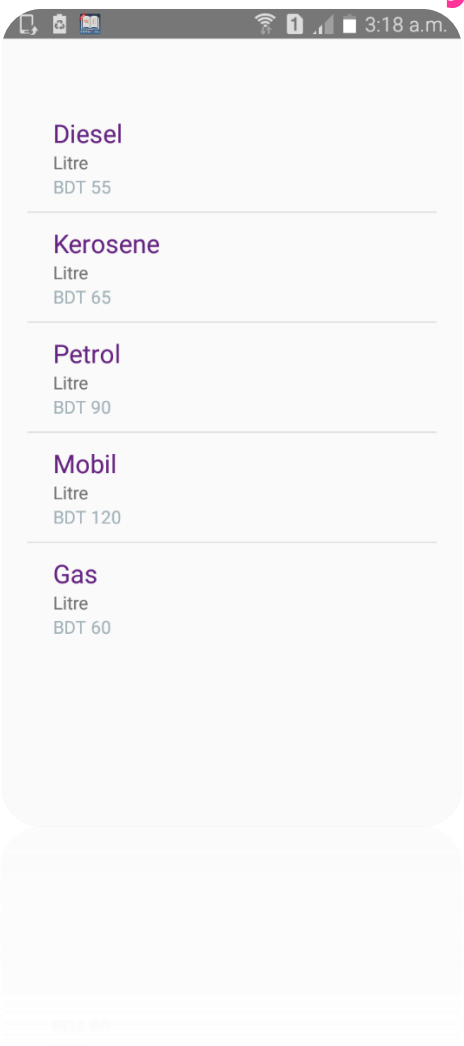


Figure 12.1 Fuel Rate Interface

12.1 Introduction:

It is a page where you can see the current rate of different types of fuel.

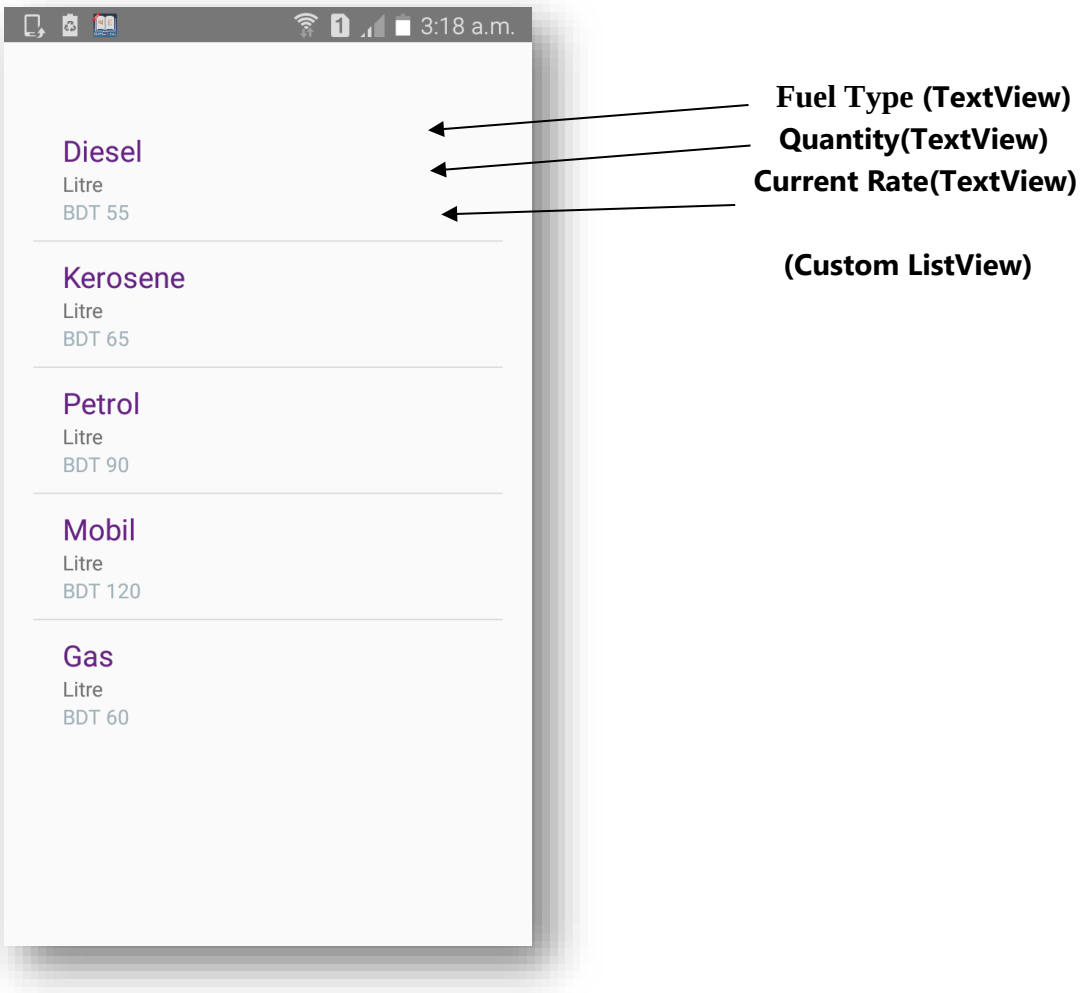


Figure 12.1 Fuel Rate Interface

12.2 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with TextView and Custom ListView. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

12.3 Interface of Fuel Rate:

- This is the app interface with TextView for watching the current rate of different types of fuel.

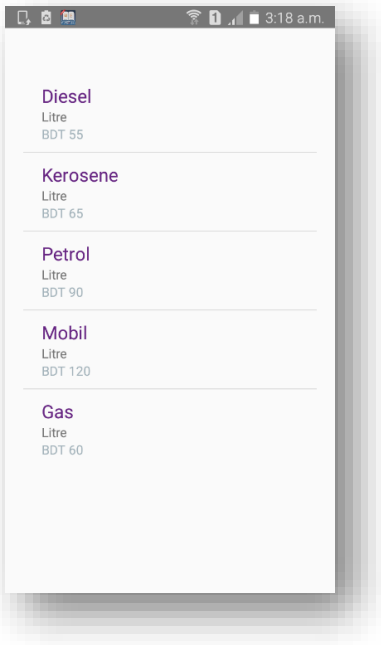


Figure 12.1 Fuel Rate Interface

12.4 Building the app GUI

In this section, you'll build the GUI for the **Fuel Rate**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_station_area_search.xml file

You'll add a TextView, EditText and ListView under LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

tools:context="project.afinal.fuelpay.StationAreaSearchActivity
">


    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        android:text="Search your area:"/>

    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp">

        <EditText
            android:id="@+id/search"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Search here..."
            android:inputType="textEmailAddress"
            android:textSize="@dimen/text_medium" />

    </android.support.design.widget.TextInputLayout>

    <ListView
        android:id="@+id/listView"
        android:padding="20dp"
        android:layout_width="match_parent"
        android:scrollbars="none"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Adding the Components in activity_station_search_area.xml file for custom listView item

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:paddingLeft="20dp"
    android:paddingBottom="10dp"
    android:paddingRight="20dp"
    android:paddingTop="10dp"
    android:background="@drawable/ripple"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/fuelType"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textColor="@color/colorPrimary"
        android:layout_weight="1"
        android:text="TextView" />

    <TextView
        android:id="@+id/weight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/fuelType"
        android:text="TextView" />

    <TextView
        android:id="@+id/amount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/weight"
        android:textColor="#ae78909c"
        android:text="Mobile" />


</RelativeLayout>
```

12.5 Java Implementation for Fuel Rate

Among the variable ListView is for showing current rate of fuels.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initializes the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

There are one threads to communicate with server to get data. This thread request data from server and get a list from server and put it on arrayList for showing a list otherwise server response false data if any error occurs. There are a few data get from server like *'fuelType', weight' and 'amount'*.

```
public class FuelRateActivity extends Activity {

    ListView lv;
    private Handler handler = new Handler();
    private ProgressDialog pDialog;
    JSONObject fuelRateInfoJSON = null;
    InternetConnectionDetector internetDetector = new
InternetConnectionDetector(this);
    HttpConnectionClass httpClass;
    String fuelRateUrl, serverResponse, Result, weight,
fuelType, amount, success;
    ArrayList<FuelRateHelper> listFuelRate = new
ArrayList<FuelRateHelper>();
    private FuelRateAdapter mAdapter;
    EditText search;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_station_area_search);
        initialize();
        // listViwePopulate();
    }

    public void initialize() {
        fuelRateUrl = getString(R.string.server_address)
                + "api/api_fuel_rate/fuel_rate";
        httpClass = new HttpConnectionClass(this);
        pDialog = new ProgressDialog(this);
        pDialog.setMessage("loading...");
        pDialog.setCancelable(false);
        lv = (ListView) findViewById(R.id.listView);
        search = (EditText) findViewById(R.id.search);
        search.setVisibility(View.GONE);

        if (internetValidation()) {
            new Thread(new LoadFuelRateTask()).start();
            showpDialog();
        }
    }

    private class LoadFuelRateTask implements Runnable {

        LoadFuelRateTask() {
        }


        @Override
        public void run() {


            try {
                URL url = new URL(fuelRateUrl); // here is
your URL pat

                serverResponse =
httpClass.httpGetConnection(url);

                fuelRateInfoJSON = new
JSONObject(serverResponse);
                success = fuelRateInfoJSON
                        .getString("success");

                JSONArray eachObjFromJSONArray =
fuelRateInfoJSON
                        .getJSONArray("info");

                for (int i = 0; i <
eachObjFromJSONArray.length(); i++) {
                    JSONObject eachObjFromJSONOb =
eachObjFromJSONArray.getJSONObject(i);

                    weight = eachObjFromJSONOb
                            .getString("weight_measurements");
                    fuelType = eachObjFromJSONOb
                            .getString("fuel_type");
                    amount = eachObjFromJSONOb
                            .getString("amount");
                    listFuelRate.add(new FuelRateHelper(
                            weight, fuelType,
amount));

                }

                Result = "";
```

```
            } catch (Exception ex) {
                Result = "Exception";
            }


            handler.post(new Runnable() {
                @Override
                public void run() {
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(FuelRateActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    }  else {
                        if (success.equals("true")) {
                            mAdapter = new
FuelRateAdapter(getBaseContext(),
                                    listFuelRate);
                            lv.setAdapter(mAdapter);
                        } else {
                            new
SweetAlertDialog(FuelRateActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.loginFailHeader))

.setContentText(getString(R.string.lofinFailMessage))
```

```
                                    .show();
                        }
                    }
                }
            });
        }

    }

    public boolean internetValidation() {
        if (!internetDetector.isConnectedToInternet()) {
            new SweetAlertDialog(FuelRateActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }
}
.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                    .show();
            return false;
        }
        return true;
    }

}
```

Code 12.2 Java Code of Fuel Rate

12.6 PHP Implementation for Fuel Rate

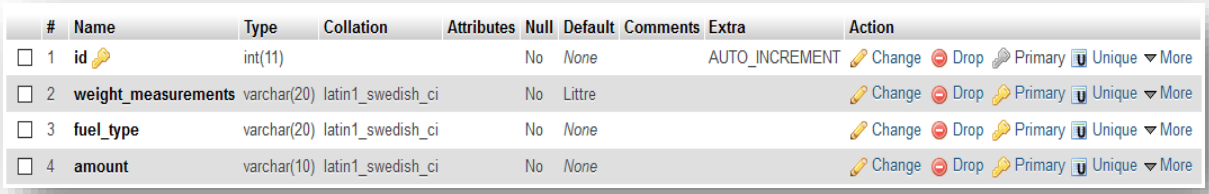
The request data From PHP end it can get data from Android through API, store data into database and response a message to Android.

- Fuel rate API

```
function fuel_rate() {

        $result = $this->Api_rate_model->fuelRateInfo();

        if (!$result):
            $info = "No data found";
            $success = "false";
        else:
            $info = $result->result();
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );


        echo json_encode($json);
    }
```

Code 12.3 PHP Code of Fuel Rate

- database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	weight_measurements	varchar(20)	latin1_swedish_ci		No	Littre			Change Drop Primary Unique More
3	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	amount	varchar(10)	latin1_swedish_ci		No	None			Change Drop Primary Unique More

Fuel Calculator

Calculate Fuel or Price

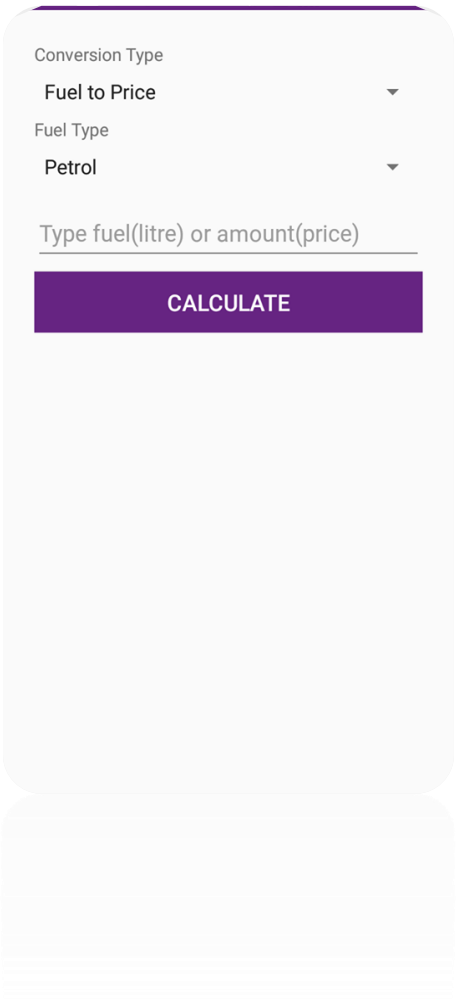


Figure 13.1: Fuel Calculator Interface

13.1 Introduction:

It is a fuel rate page where you can see fuel to price rate and vise versa and you can put the type of fuel of your vehicle.

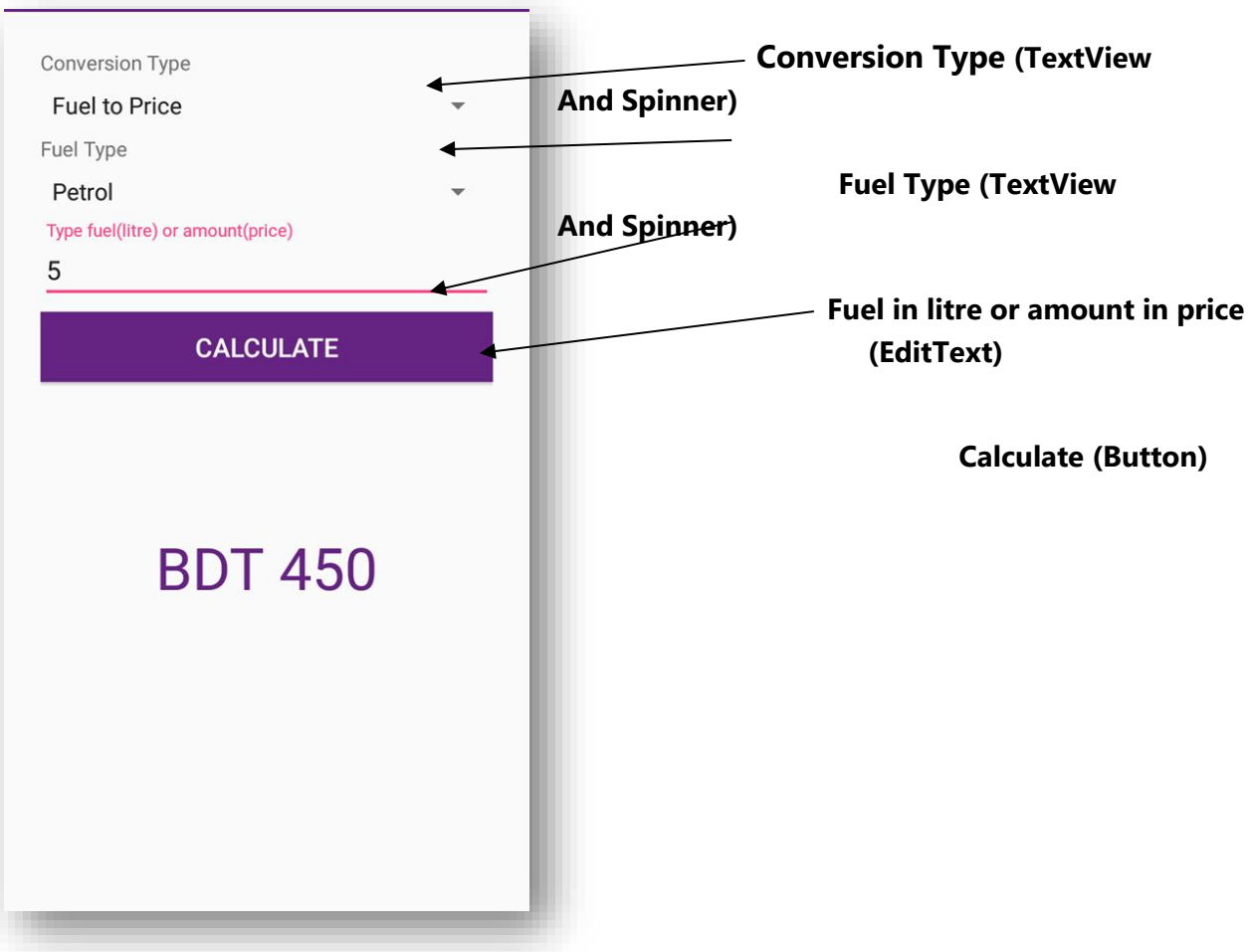


Figure 13.1: Fuel Calculator Interface

13.2 Data Flow Diagram for Fuel calculator

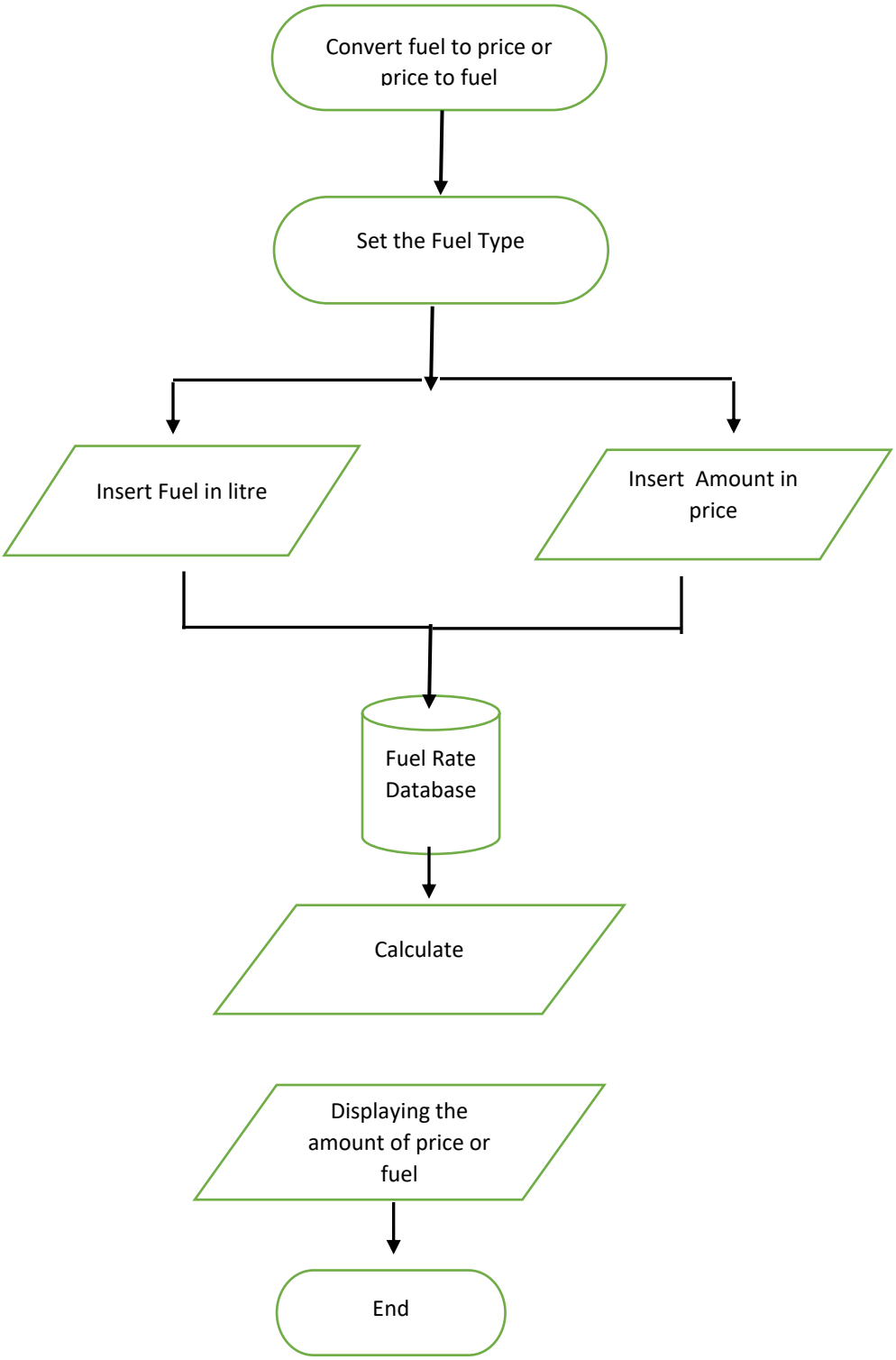


Figure 13.2: Flowchart of Fuel Calculator

13.3 Technologies Overview:

This chapter uses many Java object-oriented programming capabilities, including classes, anonymous inner classes, objects, methods, interfaces and inheritance and in the backend it uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, you'll programmatically interact with EditText, Custom ListView, and Button. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

13.4 Interface of Fuel Calculator

- This is the app interface with TextView And EditView for enter the weight of Fuel or Price and a Calculate Button to calculate.

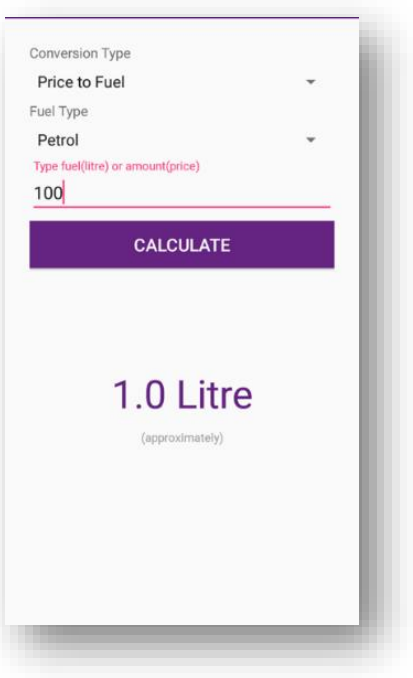


Figure 13.1: Fuel Calculator Interface

13.5 Building the app GUI

In this section, you'll build the GUI for the **Fuel Rate**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_fuel_calculator.xml file for TextView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/andro
id"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:gravity="center">


    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView2">

        <LinearLayout
            android:id="@+id/ln"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="@dimen/padding_form">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Conversion Type"
                android:textSize="14sp"/>

            <Spinner
                android:id="@+id/conversionType"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:autofillHints="Conversion Type"
                android:entries="@array/fuelCal">

            </Spinner>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Fuel Type"
                android:textSize="14sp"/>

            <Spinner
                android:id="@+id/fuelType"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:autofillHints="Fuel Type"
                android:entries="@array/fuelType">

            </Spinner>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etInput"

 android:layout_width="match_parent"

android:layout_height="wrap_content"
                    android:hint="Type fuel(litre) or
amount(price)"
                    android:inputType="number"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


            <Button
                android:id="@+id/btnRegister"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="6dp"
                android:onClick="calculate"
                android:text="Calculate"
                android:textColor="@color/colorwhite"

android:background="@color/colorPrimary"
                android:textSize="@dimen/text_medium"
/>

            <TextView
                android:id="@+id/resultText"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="100dp"
                android:text="100 Ltr"
                android:gravity="center"
                android:visibility="gone"
                android:textColor="@color/colorPrimary"

android:textSize="@dimen/text_extra_large"/>

            <TextView
                android:id="@+id/approxText"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="10dp"
                android:text="(approximately)"
                android:gravity="center"
                android:visibility="gone"
                android:textColor="@color/colorgrey"
                android:textSize="@dimen/text_micro"/>

        </LinearLayout>
    </ScrollView>


</RelativeLayout>
```

13.6 Java Implementation for Fuel calculator

Among the variable Textview user can set the text of fuel to price or price to fuel and then another Textview to set the fuel type later into the EditText iwhich input the amount of price or weight of fuel, below is Button for calculate and finally another Textview to show the result.

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass also being set.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:gravity="center">


    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView2">

        <LinearLayout
            android:id="@+id/ln"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="@dimen/padding_form">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Conversion Type"
                android:textSize="14sp"/>

            <Spinner
                android:id="@+id/conversionType"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:autofillHints="Conversion
Type"
                android:entries="@array/fuelCal">

            </Spinner>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Fuel Type"
                android:textSize="14sp"/>
```

```
            <Spinner
                android:id="@+id/fuelType"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:autofillHints="Fuel Type"
                android:entries="@array/fuelType">

            </Spinner>


<android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/etInput"

android:layout_width="match_parent"

android:layout_height="wrap_content"
                    android:hint="Type fuel(litre) or
amount(price)"
                    android:inputType="number"

android:textSize="@dimen/text_medium" />

</android.support.design.widget.TextInputLayout>


            <Button
                android:id="@+id/btnRegister"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="6dp"
                android:onClick="calculate"
                android:text="Calculate"
                android:textColor="@color/colorwhite"

android:background="@color/colorPrimary"
                android:textSize="@dimen/text_medium"
/>

            <TextView
                android:id="@+id/resultText"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="100dp"
```

```
                android:text="100 Ltr"
                android:gravity="center"
                android:visibility="gone"

android:textColor="@color/colorPrimary"

android:textSize="@dimen/text_extra_large"/>

            <TextView
                android:id="@+id/approxText"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="10dp"
```

```
                android:text="(approximately)"
                android:gravity="center"
                android:visibility="gone"
                android:textColor="@color/colorgrey"

android:textSize="@dimen/text_micro"/>

        </LinearLayout>
    </ScrollView>


</RelativeLayout>
```

Code 13.3 Java Code of Fuel Calculator

13.7 PHP Implementation for Fuel Calculator

For calculating the fuel rate's database is needed. There is a function of fuel calculator in Fuel rate from PHP end it can get data from Android through API.

```
function fuel_calculator() {

        $params['fuelType'] = $this->input-
>post('fuelType', TRUE);


        $result = $this->Api_rate_model-
>getFueltoPriceCalculatorInfo($params);



        if (!$result):
            $info = "No data found";
            $success = "false";
```

```
        else:
            $info = $result->result();
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );

        echo json_encode($json);
    }
}
```

Code 13.4 PHP Code of Fuel Calculator

- Fuel Rate database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	weight_measurements	varchar(20)	latin1_swedish_ci		No	Littre			Change Drop Primary Unique More
3	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	amount	varchar(10)	latin1_swedish_ci		No	None			Change Drop Primary Unique More

Find on Map

Easy to search your nearest fuel station

Figure 14.1: Find On Map Interface

14.1 Introduction:

We integrate google map in our project, here easily find nearest fuel station with desired information, user can also search using search box.

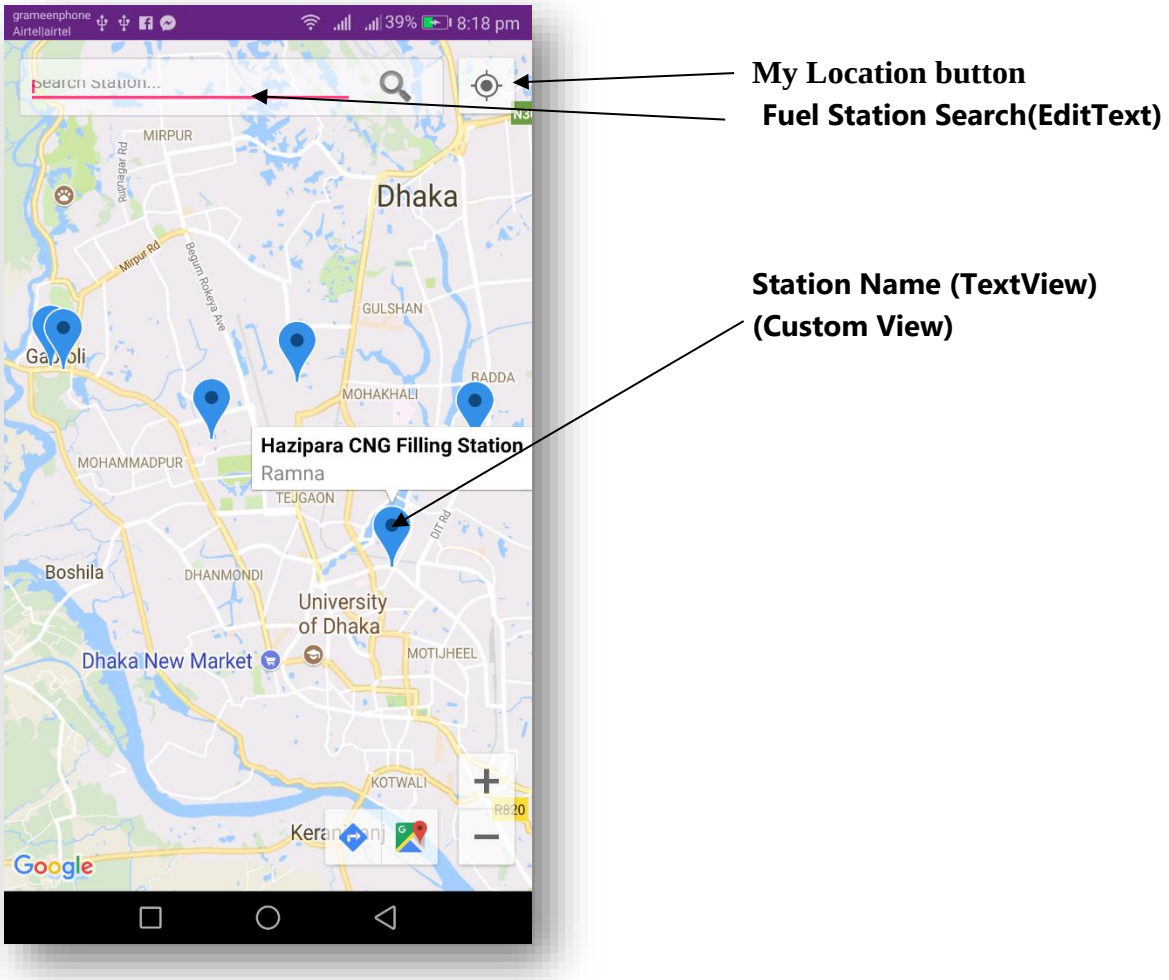


Figure 14.1: Find On Map Interface

14.2 Technologies Overview:

This chapter uses google map using API and also uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, map view, edittext for search box and location button for users position. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value. We also create google account and create project in google api console. Then enable google API for our project and also create android api key which is integrated on our project.

14.3 Building the app GUI

In this section, you'll build the GUI for the **Map**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_.xml file

You'll add a ProgressBar, Fragment for map and AutoComplete Textview under FrameLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleLargeInverse"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:visibility="invisible" >
    </ProgressBar>

    <fragment
        android:id="@+id/map"

android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />
```

```
    <LinearLayout
        android:id="@+id/layout1"
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="60dp"
        android:layout_marginTop="12dp"
        android:background="@drawable/back_curve"
        android:orientation="horizontal"
        android:padding="5dp"
        android:weightSum="1" >

        <AutoCompleteTextView
            android:id="@+id/autoCompleteTextView"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".8"
            android:hint="Search Station..."
            android:textColor="#000000"
            android:textSize="12sp" />

        <ImageView
            android:id="@+id/search"
            android:layout_width="0dp"
```

```
            android:layout_height="match_parent"
            android:layout_weight=".2"

android:src="@android:drawable/ic_search_category_default" />
```

```
    </LinearLayout>

</FrameLayout>
```

14.4 Java Implementation for Find on Map

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_map_all_search);

        mapUrl = getString(R.string.server_address)
                + "api/api_fuel_station/station_map";

        pDialog = new ProgressDialog(this);
        pDialog.setMessage("Map is loading...");
        pDialog.setCancelable(false);
        showpDialog();

        httpClass = new HttpConnectionClass(this);

        autoCompleteTextView = (AutoCompleteTextView)
findViewById(R.id.autoCompleteTextView);
        search = (ImageView) findViewById(R.id.search);
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        internetDetector = new
InternetConnectionDetector(this);
        storePreference = new SharedPreferencesClass(this);
        storePreference.putString("buttonAction", "1");


        mContext = this;

        if (lm == null)
            lm = (LocationManager) this

.getSystemService(Context.LOCATION_SERVICE);
        enableGPSNetwork();

        mProgressBar = (ProgressBar)
findViewById(R.id.progressBar);

        new Thread(new LoadLocationTask()).start();
```

```
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
                R.layout.auto_complete_text_view,
NameArrayList);
        autoCompleteTextView.setAdapter(adapter);

        autoCompleteTextView
                .setOnItemClickListener(new
AdapterView.OnItemClickListener() {

                    @Override
                    public void onItemClick(AdapterView<?>
arg0, View arg1,
                                                int arg2, long
arg3) {

                        InputMethodManager in =
(InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

in.hideSoftInputFromWindow(arg1.getWindowToken(), 0);
                        in.hideSoftInputFromWindow(

arg1.getApplicationWindowToken(), 0);

                    }
                });

        search.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String st = "";
                if
(autoCompleteTextView.getText().toString().length() > 1) {
                    st =
autoCompleteTextView.getText().toString();
                    if (NameArrayList.indexOf(st) != -1) {

                        Log.e("HI", "index: " +
NameArrayList.indexOf(st));
                        int j = NameArrayList.indexOf(st);
```

```
                        Log.e("search: " + st, "index: " + j);
                        branch = NameArrayList.get(j);
                        lat_s = LatitudeArrayList.get(j);
                        lon_s = LongitudeArrayList.get(j);

                        lat = Double.parseDouble(lat_s);
                        lon = Double.parseDouble(lon_s);

                        CameraPosition cameraPosition = new
CameraPosition.Builder()
                                .target(new LatLng(lat,
lon)).zoom(16).build();
                        mMap.animateCamera(CameraUpdateFactory

.newCameraPosition(cameraPosition));
                    } else {

Toast.makeText(getApplicationContext(), "Type the location
Name",
                                Toast.LENGTH_SHORT).show();
                    }

                } else {
                    Toast.makeText(getApplicationContext(),
"Type the location Name",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });


    }

    private void setUpMapIfNeeded() {
        // Check if we were successful in obtaining the map.
        if (mMap != null) {
            setUpMap();
        }
    }

    private void setUpMap() {
        // Hide the zoom controls as the button panel will
cover it.

        mMap.getUiSettings().setZoomControlsEnabled(true);
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and
then overriding
            //   public void onRequestPermissionsResult(int
requestCode, String[] permissions,
            //                                          int[]
grantResults)
            // to handle the case where the user grants the
permission. See the documentation
            // for ActivityCompat#requestPermissions for more
details.
            return;
        }
        mMap.setMyLocationEnabled(true);
        mMap.setOnMyLocationButtonClickListener(this);

        retrieveLocationForMap();


        mMap.setOnInfoWindowClickListener(this);

        // Pan to see all markers in view.
        // Cannot zoom to bounds until the map has a size
        final View mapView =
getSupportFragmentManager().findFragmentById(
                R.id.map).getView();
        if (mapView.getViewTreeObserver().isAlive()) {

mapView.getViewTreeObserver().addOnGlobalLayoutListener(
                    new OnGlobalLayoutListener() {
                        // We use the new method when
supported
                        @SuppressLint("NewApi")
                        // We check which build version we are
using.
                                LatLngBounds.Builder builder =
new LatLngBounds.Builder();

                        @SuppressWarnings("deprecation")
                        @SuppressLint("NewApi")
                        @Override
                        public void onGlobalLayout() {
                            for (LatLng place : allPlaces) {
                                builder.include(place);
                            }
                            LatLngBounds bounds =
builder.build();
                            if (Build.VERSION.SDK_INT <
Build.VERSION_CODES.JELLY_BEAN) {
                                mapView.getViewTreeObserver()

.removeGlobalOnLayoutListener(this);
                            } else {
                                mapView.getViewTreeObserver()

.removeOnGlobalLayoutListener(this);
                            }
                            CameraUpdate cu =
CameraUpdateFactory
                                    .newLatLngBounds(bounds,
70);
                            // mMap.moveCamera(cu);
                            hidepDialog();
                            // mMap.animateCamera(cu);
                        }
                    });
        }
    }

    static LatLng currentPosition;
    Marker markerName = null;

    private void retrieveMyLocation() {

        Location location = mMap.getMyLocation();
        currentPosition = new LatLng(location.getLatitude(),
                location.getLongitude());
        if (markerName != null) {
            markerName.remove();
        }
        markerName = mMap.addMarker(new MarkerOptions()
                .position(currentPosition)
                .title("My Location")
                .snippet(
                        "Lat:" + location.getLatitude() +
"Lng:"
                                + location.getLongitude()));
    }

    private void retrieveLocationForMap() {

        for (int i = 0; i < finalLocationList.size(); i++) {

            String placeName =
finalLocationList.get(i).getCategoryName();
            Log.e("placeName", placeName);

            String latitudeString =
finalLocationList.get(i).getLatitude();
            String longitudeString =
finalLocationList.get(i).getLongitude();
            String address =
```

```
finalLocationList.get(i).getAddress();

            Double latitude = Double.valueOf(latitudeString);
            Double longitude =
Double.valueOf(longitudeString);

            LatLng place = new LatLng(latitude, longitude);
            Log.e("place LatLng", place.toString());

            allPlaces.add(place);
            NameArrayList.add(placeName);
            LatitudeArrayList.add(String.valueOf(latitude));
            LongitudeArrayList.add(String.valueOf(longitude));

            // Add markers to the map.
            addMarkersToMap(place, placeName, address);

        }

    }

    Marker mMarker;
    ArrayList<Marker> markers = new ArrayList<Marker>();

    private void addMarkersToMap(LatLng place, String
placeName,
                                 String address) {

       /* StringBuilder addressBuilder = new
StringBuilder(200);
        addressBuilder.append(address);
        String addressFinal = addressBuilder.toString();*/

        mMarker = mMap.addMarker(new MarkerOptions()
                .position(place)
                .title(placeName)
                .snippet(address)
                .icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));
        markers.add(mMarker);

    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        LatLng hcmus = new LatLng(23.7505129, 90.3950225);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(hcmus, 12));
      /*originMarkers.add(mMap.addMarker(new MarkerOptions()
            .title("Đại học Khoa học tự nhiên")
            .position(hcmus)));*/

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

            return;
        }
        mMap.setMyLocationEnabled(true);
    }

    private class LoadLocationTask implements Runnable {

        boolean internetchecked;

        @SuppressLint("NewApi")
        @Override
        public void run() {

            handler.post(new Runnable() {
                @Override
                public void run() {
```

```
mProgressBar.setVisibility(ProgressBar.VISIBLE);
                    storePreference.putString("buttonAction",
"0");
                }
            });


            @SuppressWarnings({"unchecked", "rawtypes"})
                    RunnableFuture internetStatus = new
FutureTask(
                    new Callable<Boolean>() {

                        @Override
                        public Boolean call() throws Exception
{
                            ConnectivityManager cm =
(ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
                            NetworkInfo netInfo =
cm.getActiveNetworkInfo();
                            if (netInfo != null &&
netInfo.isConnected()) {
                                try {
                                    URL url = new
URL("http://www.google.com");
                                    HttpURLConnection urlc =
(HttpURLConnection) url
                                            .openConnection();

urlc.setConnectTimeout(3000);
                                    urlc.connect();
                                    if (urlc.getResponseCode()
== 200
                                            ||
urlc.getResponseCode() == 302) {
                                        return true;
                                    }
                                } catch (MalformedURLException
e1) {
                                    e1.printStackTrace();
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                            }
                            return false;
                        }
                    });
            // start the thread to execute it
            new Thread(internetStatus).start();
            try {
                // Get the result
                internetchecked = (Boolean)
internetStatus.get();

            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (ExecutionException e) {
                e.printStackTrace();
            }

            if (internetchecked) {
                try {

                    URL url = new URL(mapUrl); // here is your
URL pat

                    String serverResponse =
httpClass.httpGetConnection(url);

                    JSONObject JSONObjecttt = new
JSONObject(serverResponse);
                    catagoriesArray = JSONObjecttt
                            .getJSONArray("info");

                    if (!finalLocationList.isEmpty()) {
                        finalLocationList.clear();
```

```
                    }

                    for (int i = 0; i <
catagoriesArray.length(); i++) {

                        JSONObject eachObjFromJSONArray =
catagoriesArray
                                .getJSONObject(i);
                        Log.e("Number " + i + "JSONArray",

eachObjFromJSONArray.toString());

                        String branchName =
eachObjFromJSONArray
                                .getString("name");
                        String latitude = eachObjFromJSONArray
                                .getString("latitude");
                        String longitude =
eachObjFromJSONArray
                                .getString("longitude");
                        String address = eachObjFromJSONArray
                                .getString("location");

                        finalLocationList.add(new
LocationHelper(branchName,
                                latitude, longitude,
address));

                    }


                } catch (Exception ex) {
                    Result = "Exception";
                }
            } else {
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        // Internet Connection is not present
                        new
SweetAlertDialog(MapAllStationSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                                .show();

                    }
                });
            }

            handler.post(new Runnable() {
                @Override
                public void run() {

mProgressBar.setVisibility(ProgressBar.INVISIBLE);
                    storePreference.putString("buttonAction",
"1");
                    hidepDialog();
                    if (Result.equals("Exception")) {
                        new
SweetAlertDialog(MapAllStationSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)
.setTitleText(getString(R.string.errorHeader))

.setContentText(getString(R.string.errorMessage))
                                .show();
                    } else if (internetchecked) {
                        if (!finalLocationList.isEmpty()) {
                            showpDialog();
                            setUpMapIfNeeded();
                        } else {
                            // Internet Connection is not
present
                            new
SweetAlertDialog(MapAllStationSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)

.setTitleText(getString(R.string.internetHeader))

.setContentText(getString(R.string.internetMessage))
                                .show();
                        }
                    }
                }
            });
        }
    }
    public static boolean gps_enabled = false, network_enabled
= false;

    @Override
    public boolean onMyLocationButtonClick() {
        // Toast.makeText(this, "MyLocation button clicked",
Toast.LENGTH_SHORT)
        // .show();
        enableGPSNetwork();

        if (gps_enabled && network_enabled) {
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    retrieveMyLocation();
                }
            }, 3000);

        } else {
            // Internet Connection is not present
            // Internet Connection is not present
            new
SweetAlertDialog(MapAllStationSearchActivity.this,
SweetAlertDialog.ERROR_TYPE)
                    .setTitleText("Enable Location Settings")
                    .setContentText("Please Enable your GPS
Settings")
                    .show();

        }
        return false;
    }

    @Override
    protected void onResume() {
        super.onResume();

        enableGPSNetwork();

    }
```

Code 14.2 Java Code of Find on map

14.5 PHP Implementation for Map

The request data From PHP end it can get data from Android through API, get data from database and response a message to Android.

- Map API

```
function station_map() {

        $result = $this->Api_station_model-
>allAreaInfo();

        if (!$result):
            $info = "No data found";
            $success = "false";
        else:
            $info = $result->result();
            $success = "true";
        endif;

        $json = array(
            "success" => $success,
            "info" => $info
        );

        echo json_encode($json);
    }
```

Code 14.3 PHP Code of Find on map

- Station_setup database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary More
5	start_time	time			No	None			Change Drop Primary More
6	end_time	time			No	None			Change Drop Primary More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary More
8	latitude	decimal(10,6)			No	None			Change Drop Primary More
9	longitude	decimal(10,6)			No	None			Change Drop Primary More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary More
11	amount	int(20)			No	None			Change Drop Primary More
12	traffic	int(11)			No	None			Change Drop Primary More
13	create_dateTime	datetime			No	None			Change Drop Primary More
14	update_dateTime	datetime			No	None			Change Drop Primary More

Map Reminder

Get your fuel alert before each journey

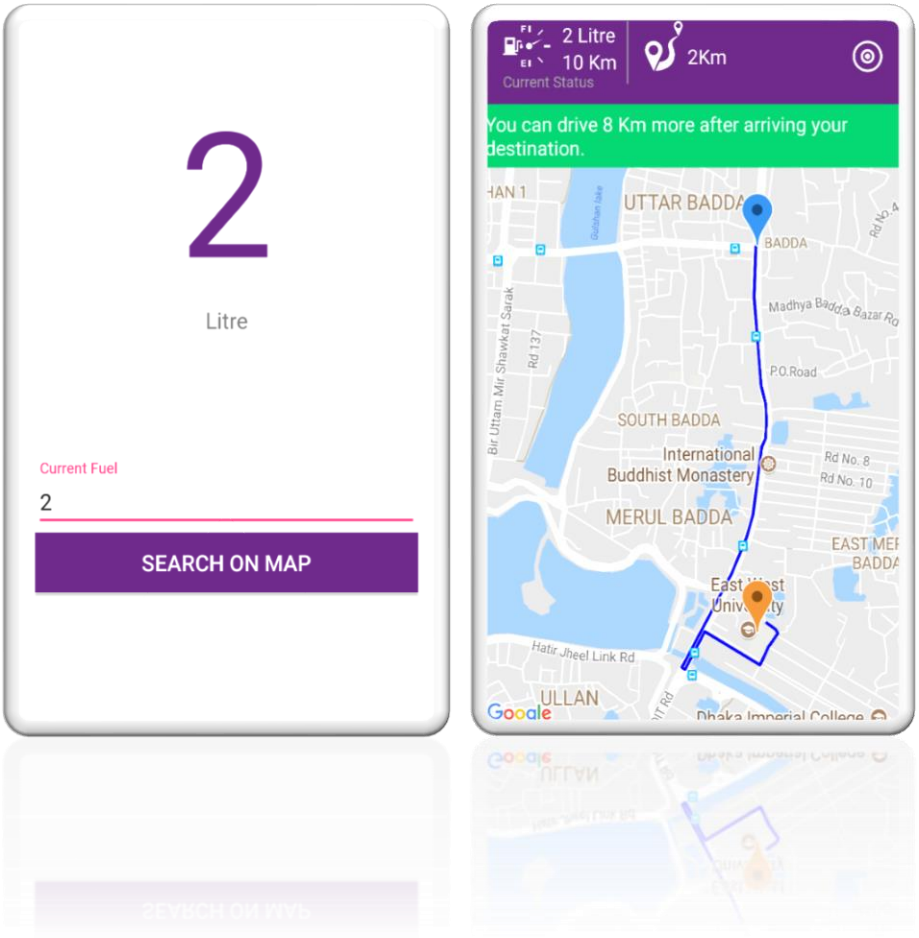


Figure 15.1: Map Reminder Interface

15.1 Introduction:

We integrate google map in our project, here easily find nearest fuel station with desired information, user can also search using search box.

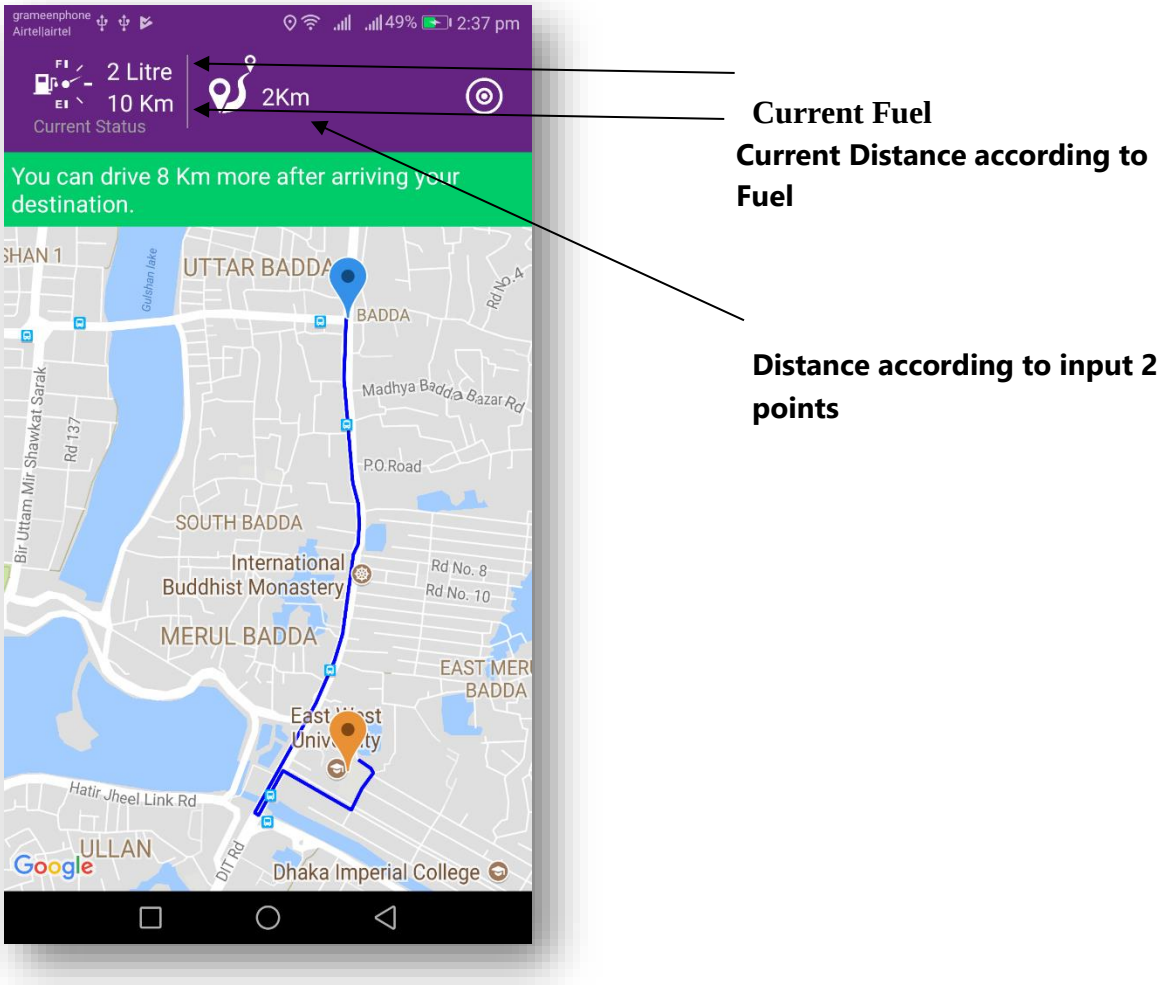


Figure 15.1: Map Reminder Interface

15.2 Technologies Overview:

This chapter uses google map using API and also uses php and to make a connection between PHP and Android, JSON is used and also database is used to store the value. In android, map view, edittext for enter fuel amount and location button for users position. You'll create these components by direct manipulation of the GUI layout's XML. You'll use event handling and anonymous inner classes to process the user's GUI interactions. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value. We also create google account and create project in google api console. Then enable google API for our project and also create android api key which is integrated on our project.

15.3 Building the app GUI

In this section, you'll build the GUI for the **Map**. At the end of this section, we'll present the XML for this module's layout.

Adding the Components in activity_map_input_distance.xml file

You'll add a ProgressBar, textview, edittext and button.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:padding="@dimen/padding_list"
    android:focusableInTouchMode="true">


    <LinearLayout
        android:id="@+id/main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView2"
        android:orientation="vertical">
        <TextView
            android:id="@+id/txtFuel"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="0"
            android:gravity="center"
            android:textSize="140sp"
            android:textColor="@color/colorPrimary"
            android:layout_marginTop="30dp"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="Litre"
            android:gravity="center"
            android:textSize="@dimen/text_medium"/>

        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="100dp">

            <EditText
                android:id="@+id/etfuel"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Current Fuel"
                android:inputType="number"
                android:textSize="@dimen/text_medium" />

        </android.support.design.widget.TextInputLayout>


        <Button
            android:id="@+id/btnSearchMap"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:layout_marginBottom="6dp"
            android:layout_marginTop="2dp"
            android:onClick="searchOnMap"
            android:text="Search on map"
            android:background="@color/colorPrimary"
            android:textColor="@color/colorwhite"
            android:textSize="@dimen/text_medium" />
```

```
    </LinearLayout>

</RelativeLayout>
```

Adding the Components in activity_map_input_distance.xml file

You'll add a fragment for map, textview, edittext and button.

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <RelativeLayout
          android:id="@+id/layout"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:background="@color/colorPrimary"
          android:orientation="horizontal"
          android:padding="10dp">
          <ImageView
              android:id="@+id/imgCurrnetFuel"
              android:layout_width="40dp"
              android:layout_height="40dp"
              android:layout_marginLeft="10dp"
              android:src="@drawable/current_fuel" />

          <TextView
  xmlns:tools="http://schemas.android.com/tools"
              android:id="@+id/currentFuel"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="0 Litre"
              android:layout_marginLeft="10dp"
              android:layout_toRightOf="@+id/imgCurrnetFuel"
              android:textColor="@color/colorwhite"
              android:textSize="16sp" />
          <TextView
  xmlns:tools="http://schemas.android.com/tools"
              android:id="@+id/currentKm"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_marginLeft="10dp"
              android:text="0 km"
              android:layout_toRightOf="@+id/imgCurrnetFuel"
              android:layout_below="@+id/currentFuel"
              android:textColor="@color/colorwhite"
              android:textSize="16sp" />
          <TextView
  xmlns:tools="http://schemas.android.com/tools"
              android:id="@+id/current"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_marginLeft="10dp"
              android:text="Current Status"
              android:layout_below="@+id/imgCurrnetFuel"
              android:textColor="@color/colorgrey"
              android:textSize="12sp" />
          <View
              android:id="@+id/view"
              android:layout_width="1dp"
              android:layout_height="50dp"
              android:layout_marginLeft="10dp"
              android:layout_toRightOf="@+id/currentFuel"
              android:background="@color/colorgrey"/>

          <ImageView
              android:id="@+id/imgDistance"
              android:layout_width="40dp"
              android:layout_height="40dp"
              android:layout_toRightOf="@+id/view"
              android:layout_marginLeft="10dp"
              android:src="@drawable/distance" />

          <TextView
  xmlns:tools="http://schemas.android.com/tools"
              android:id="@+id/distance"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_toRightOf="@+id/imgDistance"
              android:layout_centerVertical="true"
              android:text="0 km"
              android:textColor="@color/colorwhite"
              android:textSize="16sp" />


          <Button
              android:id="@+id/button"
              android:layout_width="25dp"
              android:layout_height="25dp"
              android:layout_marginRight="10dp"
              android:layout_alignParentEnd="true"
              android:layout_centerVertical="true"
              android:background="@drawable/mylocation" />
      </RelativeLayout>

      <LinearLayout
          android:id="@+id/layout1"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_below="@+id/layout"
          android:background="@color/warning"
          android:visibility="gone"
          android:orientation="horizontal">
          <TextView
              android:id="@+id/remining"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:padding="5dp"
              android:text="your remaining fuel"
              android:textColor="@color/colorwhite"
              android:textSize="16sp" />

      </LinearLayout>

      <fragment xmlns:tools="http://schemas.android.com/tools"
          android:id="@+id/map"

  android:name="com.google.android.gms.maps.SupportMapFragment"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:layout_below="@+id/layout1"

  tools:context="com.example.admin1.googlemapdemo.MapsActivity"
```

```
/>
</RelativeLayout>
```

15.4 Java Implementation for Map Reminder

The onCreate method which is auto-generated when you create the app's project—is called by the system when an Activity is *started*. The initialize method is called in onCreate method. It typically initializes the Activity's instance variables and GUI components. It also initialize the HttpConnectionClass and SharedPreferencesClass. Different property of ProgressDialog class is also being set.

Java file for *MapInputDistaceReminderActivity.java*

```
public class MapInputDistanceActivity extends Activity{
    EditText etfuel;
    TextView txtFuel;
    @Override
    protected void onCreate(@Nullable Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_map_input_distance);
        initialize();
    }

    public void initialize() {
        etfuel = findViewById(R.id.etfuel);
        txtFuel =findViewById(R.id.txtFuel);

        etfuel.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                // Do some thing now
            }

            @Override
            public void onTextChanged(CharSequence s, int
start, int before, int count) {
                // Change the TextView background color
                //tv.setBackgroundColor(Color.YELLOW);

                // Get the EditText text and display it on
TextView
                txtFuel.setText(etfuel.getText());
            }

            @Override
            public void afterTextChanged(Editable s) {
                // Do something at this time
            }
        });
    }
    public void searchOnMap(View v) {
        Bundle localBundle = new Bundle();
        localBundle.putString("currentFuel",
etfuel.getText().toString());
        Intent intent = new Intent(getBaseContext(),
MapDistanceReminderActivity.class);
        intent.putExtras(localBundle);
        startActivity(intent);
    }
}
```

Java file for *MapDistaceReminderActivity.java*

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_distance_reminder);
        // Obtain the SupportMapFragment and get notified when
the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        storePreference = new
SharedPreferencesClass(getApplicationContext());
```

```
        layout =findViewById(R.id.layout1);

        button = (Button) findViewById(R.id.button);
        txtDistance = (TextView) findViewById(R.id.distance);
        currentFuel = (TextView)
findViewById(R.id.currentFuel);
        currentKm = (TextView) findViewById(R.id.currentKm);
        remining = (TextView) findViewById(R.id.remining);

        getFuel = getIntent().getStringExtra("currentFuel");
        currentFuel.setText(getFuel+" Litre");

        getKm = Integer.parseInt(getFuel) *
Integer.parseInt(storePreference.getString("fuel_per_km"));
        currentKm.setText(getKm+" Km");

        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        googleApiClient.connect();
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getCurrentLocation();
            }
        });
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        CameraPosition cameraPosition = new
CameraPosition.Builder().target(
                new LatLng(23.768748,
90.425642)).zoom(16).build();
        MarkerOptions marker = (new
MarkerOptions().position(new LatLng(23.768748,
90.425642)).title("Marker in my location"));

marker.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescrip
torFactory.HUE_ORANGE));
        markerPoints.add(new LatLng(23.768748, 90.425642));
        mMap.addMarker(marker);

mMap.animateCamera(CameraUpdateFactory.newCameraPosition(camera
Position));


        mMap.setOnMapClickListener(new
GoogleMap.OnMapClickListener() {
            @Override
            public void onMapClick(LatLng latLng) {
                addPath(latLng);
            }
        });
    }

    private void addPath(LatLng latLng) {
        if (markerPoints.size() > 1) {
            markerPoints.clear();
            mMap.clear();
            txtDistance.setText("0 Km");
            layout.setVisibility(View.GONE);
        }

        // Adding new item to the ArrayList
        markerPoints.add(latLng);

        // Creating MarkerOptions
        MarkerOptions options = new MarkerOptions();

        // Setting the position of the marker
        options.position(latLng);
```

```
        if (markerPoints.size() == 1) {

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescri
ptorFactory.HUE_VIOLET));
        } else if (markerPoints.size() == 2) {

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescri
ptorFactory.HUE_AZURE));
        }

        // Add new marker to the Google Map Android API V2
        mMap.addMarker(options);

        // Checks, whether start and end locations are captured
        if (markerPoints.size() >= 2) {
            LatLng origin = (LatLng) markerPoints.get(0);
            LatLng dest = (LatLng) markerPoints.get(1);

            // Getting URL to the Google Directions API
            String url = getDirectionsUrl(origin, dest);
            Log.d("DistanceMapsActivity", "getDirectionsUrl: "
+ url);
            Toast.makeText(this, url,
Toast.LENGTH_SHORT).show();
            DownloadTask downloadTask = new DownloadTask();

            // Start downloading json data from Google
Directions API
            downloadTask.execute(url);
        }
    }

    public void getCurrentLocation() {

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            Location location =
LocationServices.FusedLocationApi.getLastLocation(googleApiClie
nt);
            if (location == null) {

LocationServices.FusedLocationApi.requestLocationUpdates(google
ApiClient, mLocationRequest, this);
            } else {
                moveMap(location);
            }
        }
    }


    private void moveMap(Location location) {

        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        LatLng latLng = new LatLng(latitude, longitude);


        Toast.makeText(this, latitude + "" + longitude,
Toast.LENGTH_LONG).show();
        addPath(latLng);
    }

    private class DownloadTask extends AsyncTask<String, Void,
String> {

        @Override
        protected String doInBackground(String... url) {

            String data = "";

            try {
                data = downloadUrl(url[0]);
            } catch (Exception e) {
```

```
                Log.d("Background Task", e.toString());
            }
            return data;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

            ParserTask parserTask = new ParserTask();


            parserTask.execute(result);

        }
    }

    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String, String>>>> {

        // Parsing the data in non-ui thread
        @Override
        protected List<List<HashMap<String, String>>>
doInBackground(String... jsonData) {

            JSONObject jObject;
            List<List<HashMap<String, String>>> routes = null;

            try {
                jObject = new JSONObject(jsonData[0]);
                DirectionsJSONParser parser = new
DirectionsJSONParser();

                routes = parser.parse(jObject);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return routes;
        }

        @Override
        protected void onPostExecute(List<List<HashMap<String,
String>>> result) {
            ArrayList points = null;
            PolylineOptions lineOptions = new
PolylineOptions();

            lineOptions.width(8);
            lineOptions.color(BLUE);
            MarkerOptions markerOptions = new MarkerOptions();

            points = new ArrayList();
            try {
                List<HashMap<String, String>> path =
result.get(0);
                List<HashMap<String, String>> distance =
result.get(1);
                List<HashMap<String, String>> duration =
result.get(2);

                // To draw path between two location
                drawPath(path);
                for (int j = 0; j < path.size(); j++) {
                    HashMap<String, String> point =
path.get(j);

                    double lat =
Double.parseDouble(point.get("lat"));
                    double lng =
Double.parseDouble(point.get("lng"));
                    LatLng position = new LatLng(lat, lng);

                    points.add(position);
                }

                lineOptions.addAll(points);
                lineOptions.geodesic(true);

                // To count distance between two location
                int totalDistance = 0;
                for (int j = 0; j < distance.size(); j++) {
                    HashMap<String, String> point =
distance.get(j);

                    int dist =
Integer.parseInt(point.get("distance"));
                    totalDistance = totalDistance + dist;
                }
                String totalDist =String.valueOf(totalDistance
/ 1000 + "Km");
                txtDistance.setText(totalDist);

                //for visible info layout
                layout.setVisibility(View.VISIBLE);
                int finalDist = totalDistance/1000;
                if(getKm>finalDist) {
                    int existFuel = getKm-finalDist;

layout.setBackgroundColor(getResources().getColor(R.color.succe
ss));
                    remining.setText("You can drive
"+existFuel+" Km more after arriving your destination.");
                }else{
                    int existFuel = finalDist-getKm;

layout.setBackgroundColor(getResources().getColor(R.color.warni
ng));
                    remining.setText("You want to drive
"+finalDist+" km but need fuel for "+existFuel+" Km extra");
                }
                // To count duration between two location
                /*int totalDuration = 0;
                for (int k = 0; k < duration.size(); k++) {
                    HashMap<String, String> point =
duration.get(k);

                    int dist =
Integer.parseInt(point.get("duration"));
                    totalDuration = totalDistance + dist;
                }
                convertSecondToHour(totalDuration);*/
            } catch (IndexOutOfBoundsException e) {
                e.printStackTrace();
            }

            // for covering the path drawn in display
            boolean hasPoints = false;
            Double maxLat = null, minLat = null, minLon = null,
maxLon = null;

            if (lineOptions != null && lineOptions.getPoints()
!= null) {
                List<LatLng> pts = lineOptions.getPoints();
                for (LatLng coordinate : pts) {
                    // Find out the maximum and minimum
latitudes & longitudes
                    // Latitude
                    maxLat = maxLat != null ?
Math.max(coordinate.latitude, maxLat) : coordinate.latitude;
                    minLat = minLat != null ?
Math.min(coordinate.latitude, minLat) : coordinate.latitude;

                    // Longitude
                    maxLon = maxLon != null ?
Math.max(coordinate.longitude, maxLon) : coordinate.longitude;
                    minLon = minLon != null ?
Math.min(coordinate.longitude, minLon) : coordinate.longitude;

                    hasPoints = true;
                }
            }

            if (hasPoints) {
                LatLngBounds.Builder builder = new
```

```
LatLngBounds.Builder();
                builder.include(new LatLng(maxLat, maxLon));
                builder.include(new LatLng(minLat, minLon));

mMap.moveCamera(CameraUpdateFactory.newLatLngBounds(builder.bui
ld(), 48));
                mMap.addPolyline(lineOptions);
            }
        }
    }

    private void drawPath(List<HashMap<String, String>> path) {
    }

  /*  private void convertSecondToHour(int totalDuration) {
        int hours = totalDuration / 3600;
        int minutes = (totalDuration % 3600) / 60;
        int seconds = (totalDuration % 3600) % 60;

        String totalTime = String.valueOf(hours) + ":" +
                String.valueOf(minutes) + ":" +
String.valueOf(seconds);
        txtTime.setText(totalTime);
    }*/

    private String getDirectionsUrl(LatLng origin, LatLng dest)
{

        // Origin of route
        String str_origin = "origin=" + origin.latitude + "," +
origin.longitude;

        // Destination of route
        String str_dest = "destination=" + dest.latitude + ","
+ dest.longitude;

        // Sensor enabled
        String sensor = "sensor=false";
        String mode = "mode=driving";

        // Building the parameters to the web service
        String parameters = str_origin + "&" + str_dest + "&" +
sensor + "&" + mode;

        // Output format
        String output = "json";

        // Building the url to the web service
        String url =
"https://maps.googleapis.com/maps/api/directions/" + output +
"?" + parameters;


        return url;
    }

    private String downloadUrl(String strUrl) throws
IOException {
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try {
            URL url = new URL(strUrl);

            urlConnection = (HttpURLConnection)
url.openConnection();

            urlConnection.connect();

            iStream = urlConnection.getInputStream();

            BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));

            StringBuffer sb = new StringBuffer();

            String line = "";
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }

            data = sb.toString();

            br.close();

        } catch (Exception e) {
            Log.d("Exception", e.toString());
        } finally {
            iStream.close();
            urlConnection.disconnect();
        }
        return data;

    }

    @Override
    protected void onStart() {
        if (googleApiClient != null) {
            googleApiClient.connect();
        }
        super.onStart();
    }

    @Override
    protected void onStop() {
        googleApiClient.disconnect();
        super.onStop();
    }

    @Override
    protected void onPause() {
        super.onPause();
        if (googleApiClient.isConnected()) {

LocationServices.FusedLocationApi.removeLocationUpdates(googleA
piClient, this);
            googleApiClient.disconnect();
        }
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {

LocationServices.FusedLocationApi.removeLocationUpdates(googleA
piClient, this);
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCU
RACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {

LocationServices.FusedLocationApi.requestLocationUpdates(google
ApiClient,
                    mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {
        if (connectionResult.hasResolution()) {
            try {
                // Start an Activity that tries to resolve the
error
                connectionResult.startResolutionForResult(this,
```

```
9000);
                Log.d("DistanceMapsActivity",
"onConnectionFailed: ");
            } catch (IntentSender.SendIntentException e) {
                e.printStackTrace();
                Log.e("DistanceMapsActivity",
"onConnectionFailed: ");
            }
        } else {
            Log.i("DistanceMapsActivity", "Location services
connection failed with code " +
connectionResult.getErrorCode());
        }
    }
}
```

Code 15.2 Java Code of Map Reminder

ADMIN LOGIN

Enter Your System Account

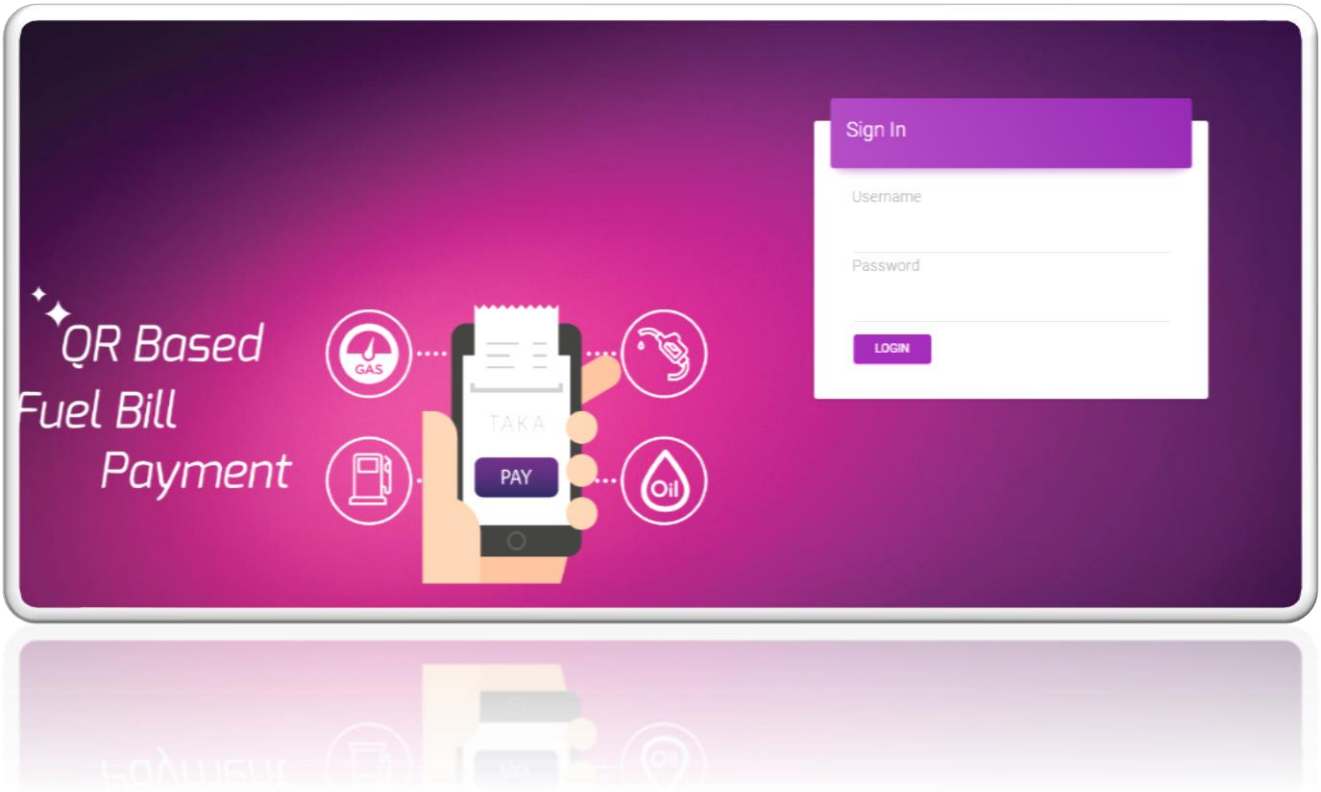


Figure 16.1: Admin Login Interface

16.1 Introduction:

It is a login page where admin can put their username and password and login to the system.

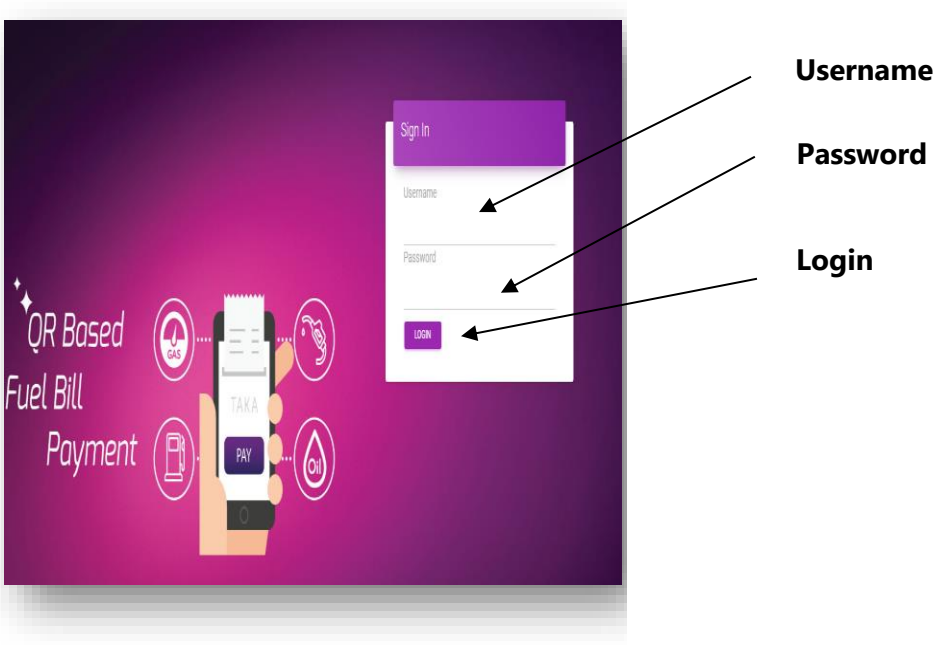


Figure 16.1: Admin Login Interface

16.2 Data Flow Diagram for Admin Login

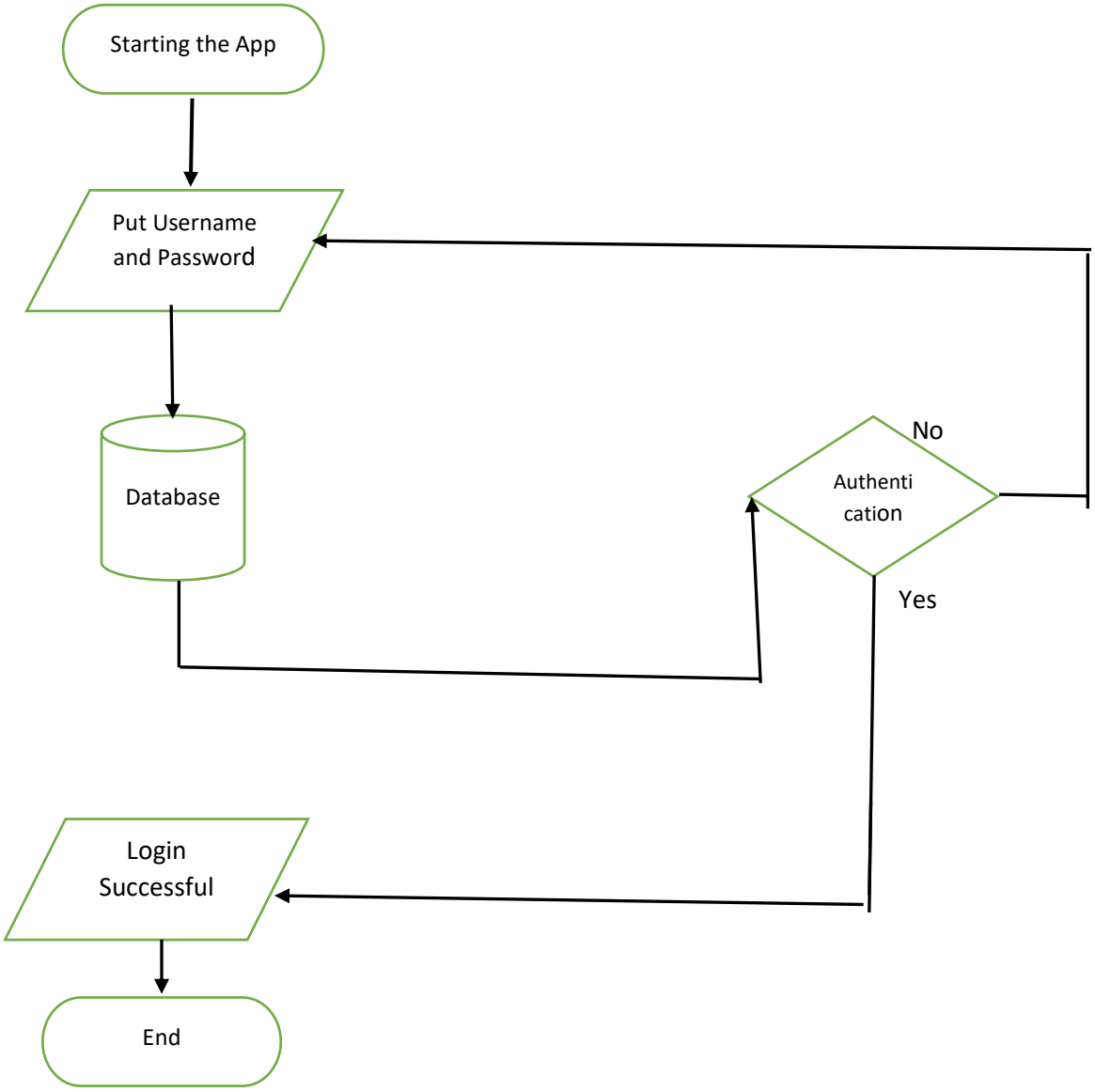


Figure 16.2: Flowchart of Admin Login

16.3 Technologies Overview:

This chapter uses many PHP object-oriented programming capabilities, build in functions, objects, methods, variables and it use CodeIgniter framework and to send values to mobile and make a connection between PHP and Android, JSON is used and also database is used to store the value. In PHP, you'll programmatically interact with Text, Button and Label. You'll create these programmatically. You'll use Bootstrap, JavaScript for designing our app. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

16.4 Interface of Admin Login:

- This is the app interface with input type="text" for enter the username of admin and giving the password of admin and also the Login button for entering in the system.

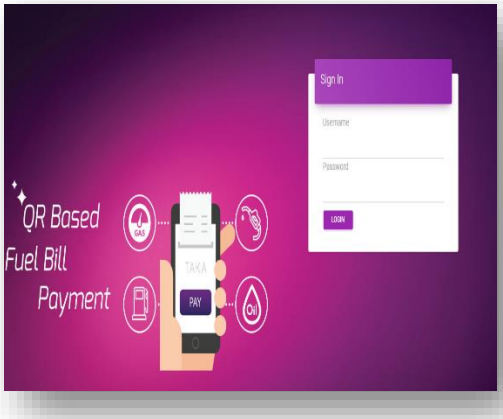
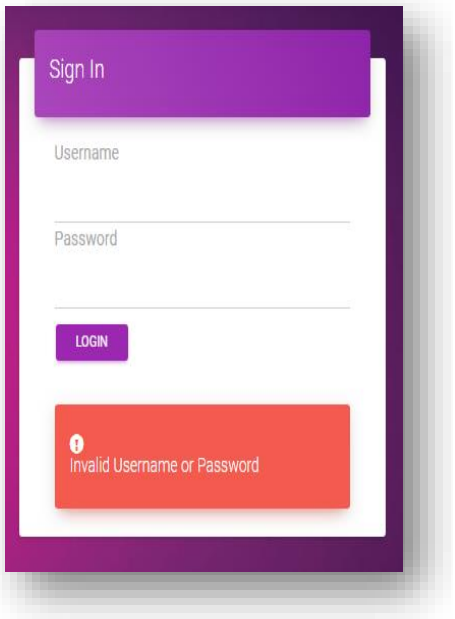


Figure 16.1: Admin Login Interface



- Username or password is incorrect

- When the request is successful then it goes to Dashboard.

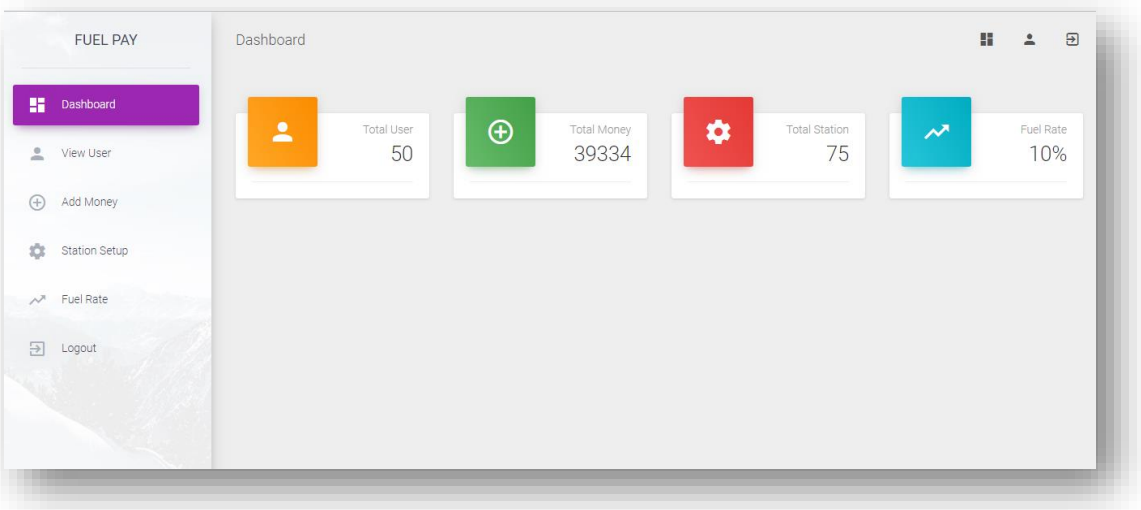


Figure 16.3: Admin Dashboard Interface

16.5 Building the app View

In this section, you'll build the View for the **Login**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in loginView.php file

You'll add a Textbox and Button.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    <link rel="apple-touch-icon" sizes="76x76" href="<?php
echo base_url();?>assets/image/apple-icon.png" />
    <link  rel="icon"  type="image/png"  href="<?php  echo
base_url();?>assets/image/favicon.png" />
    <meta                      http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1" />
    <title>Login | FuelPay</title>
    <meta  content='width=device-width,  initial-scale=1.0,
maximum-scale=1.0, user-scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <!-- Bootstrap core CSS     -->
    <link                  href="<?php                  echo
base_url();?>assets/css/bootstrap.min.css" rel="stylesheet"
/>
    <!--  Material Dashboard CSS    -->
    <link href="<?php echo base_url();?>assets/css/material-
dashboard.css" rel="stylesheet" />
    <!--   CSS for Demo Purpose, don't include it in your
project     -->
    <link href="<?php echo base_url();?>assets/css/demo.css"
rel="stylesheet" /><!--
    <link rel="stylesheet" type="text/css" href="<?php echo
base_url();?>assets/css/bootstrap.min.css"> -->

    <!--     Fonts and icons     -->
    <link  href="<?php  echo  base_url();?>assets/css/font-
awesome.min.css" rel="stylesheet">
    <link
href='https://fonts.googleapis.com/css?family=Roboto:400,70
0,300|Material+Icons' rel='stylesheet' type='text/css'>
    </head>

    <body        background=".//assets/image/loginback.jpg"
style="background-size: cover;">
    <div class="container">
                <div class="row">
                     <div class="col-md-7"></div>

<div class="col-md-4">
                                        <div    class="card"
style="margin-top: 150px">
                                            <div
class="card-header" data-background-color="purple">
                                    <h4   class="title">Sign
In</h4>
                        </div>
                                            <div
class="card-content">

                <div class="col-md-12">
                <div id="loginback">

                <form          action="<?php          echo
base_url()?>Login/login_process" method="post">

                <div class="form-group">

                                            <label
>Username</label>

                                            <input   type="text"
class="form-control" name="username" >

                                            </div>

                                            <div    class="form-
group">


                <label>Password</label>

                                            <input
type="password" class="form-control" name="password">

                                            </div>

                                                    <button
type="submit"        class="btn        btn-sm        btn-
primary">Login</button><br><br>


                <?php

                                                        if
(!$message == '')
 {


                ?>

                <div       class="alert       alert-danger"
role="alert">

                  <span     class="glyphicon     glyphicon-
exclamation-sign" aria-hidden="true"></span>

                  <span class="sr-only">Error:</span>
```

```
                    <?php        echo        $this->session-
>flashdata('message'); ?>


                </div>


                <?php


                }?>

                </form>

                </div>


</div>
                                        </div>
                                        </div>
                                    </div>

</div>

</div>
</body>
 <script         src="<?php         echo         base_url();
?>assets/js/jquery.min.js" type="text/javascript"></script>
<script         src="<?php         echo         base_url();
?>assets/js/bootstrap.js"></script>
<script                    src="<?php                    echo
base_url();?>assets/js/material.min.js"
type="text/javascript"></script>
```

```
<!--  Charts Plugin -->
<script                    src="<?php                    echo
base_url();?>assets/js/chartist.min.js"></script>
<!--  Dynamic Elements plugin -->
<script                    src="<?php                    echo
base_url();?>assets/js/arrive.min.js"></script>
<!--  PerfectScrollbar Library -->
<script  src="<?php  echo  base_url();?>assets/js/perfect-
scrollbar.jquery.min.js"></script>
<!--  Notifications Plugin    -->
<script  src="<?php  echo  base_url();?>assets/js/bootstrap-
notify.js"></script>
<!--  Google Maps Plugin    -->
<script                                type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY_H
ERE"></script>
<!-- Material Dashboard javascript methods -->
<script  src="<?php  echo  base_url();?>assets/js/material-
dashboard.js?v=1.2.0"></script>
<!-- Material Dashboard DEMO methods, don't include it in
your project! -->
<script                    src="<?php                    echo
base_url();?>assets/js/demo.js"></script>
<script type="text/javascript">
    $(document).ready(function() {

        // Javascript method's body can be found in
assets/js/demos.js
        demo.initDashboardPageCharts();



    });
</script>
</html>
```

16.6 PHP Implementation for Admin Login

Among the variable textbox into which input the username and password, one is Button for Login.

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

In View, it calls login.php controller file and in controller it calls login_process function. This function communicates with model class to get data. There are a few data get from model class like *'username' and 'password''*

- Login_process in controller class

```
public function login_process()
    {
```

```
        $userInfo['username'] = $this->input->post('username');
        $userInfo['password'] = $this->input->post('password');
```

```
        // ************* new **************
        $loginResult = $this->Login_model->doLogin($userInfo);

        if ($loginResult){

            echo "login done";
            $data = array(
                    'users_id' => $loginResult->users_id,
                    'username' => $loginResult->username,
                    'full_name' => $loginResult->full_name,
                    'contact_no' => $loginResult->contact_no,
                    'email' => $loginResult->email,
                    'validated' => true
                    );
                $this->session->set_userdata($data);
```

```
                redirect('Home/dashboard');

            }
            else{
                $response['ISSUCCESS'] = "N";
                $response['RESULT'] = "";
                $response['MESSAGE'] = "Invalid Username or
Password";

                //add flash data
             $this->session->set_flashdata('message','Invalid
Username or Password');

                redirect('Login');

            }

            echo json_encode($response);

        }                                   .
```

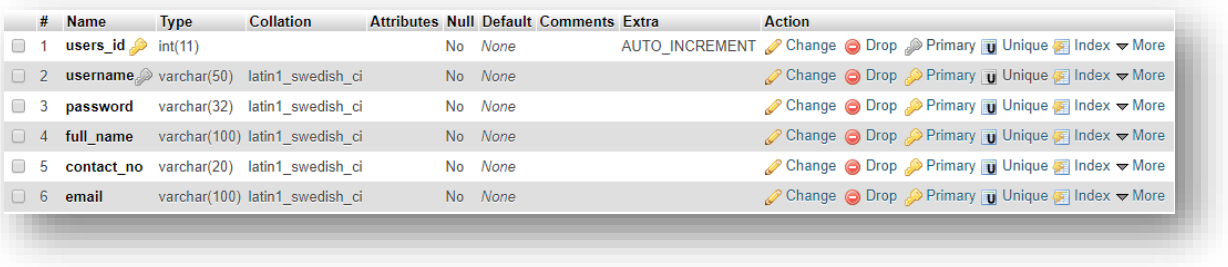
- doLogin in model class

```
function doLogin($userInfo){



        $this->db->where('username', $userInfo['username']);

        $this->db->where('password',
MD5($userInfo['password']));
```

```
$query = $this->db->get('users');

        return ($query->num_rows()> 0) ? $query->row() : false;

    }
```

Code 16.4 PHP Code Of Admin Login

- users database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	users_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	username	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	password	varchar(32)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	full_name	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	contact_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	email	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More

ADMIN STATION SETUP

Add station and Information of Station with QR Code

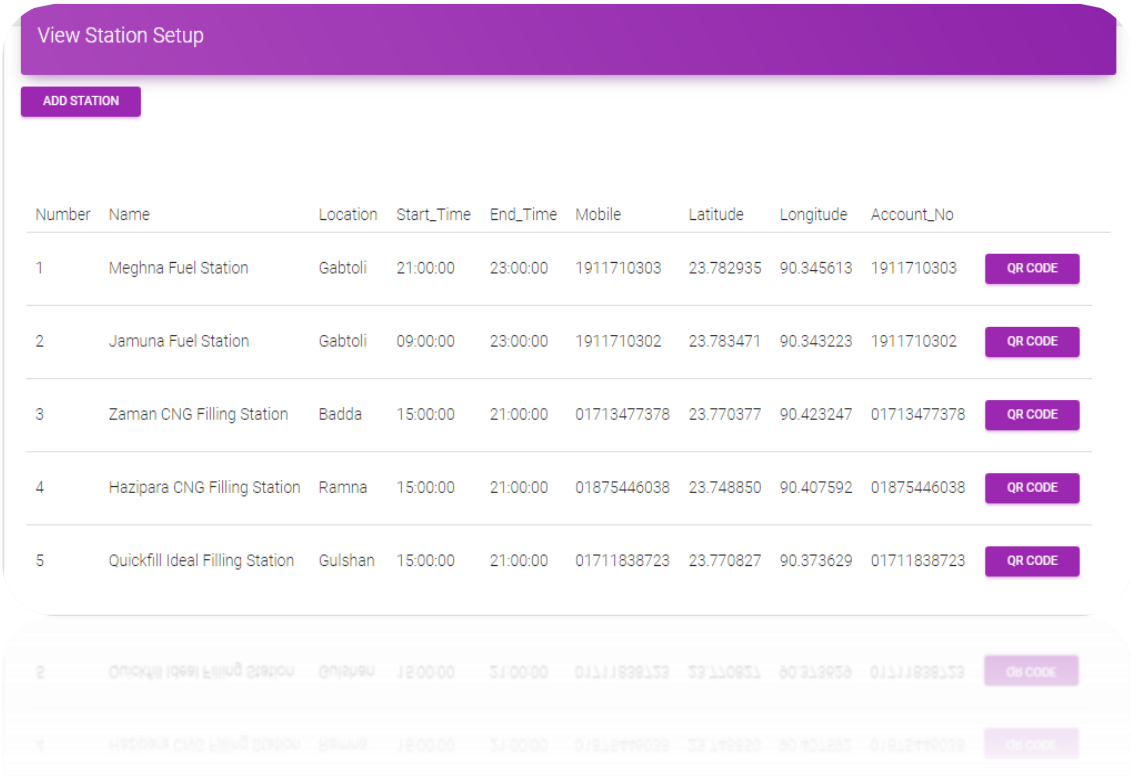


Figure 17.1: Admin Station Setup Interface

17.1 Introduction:

It is a page where you can add fuel filling station and see the information of fuel filling station and also generate QR code for each filling station.

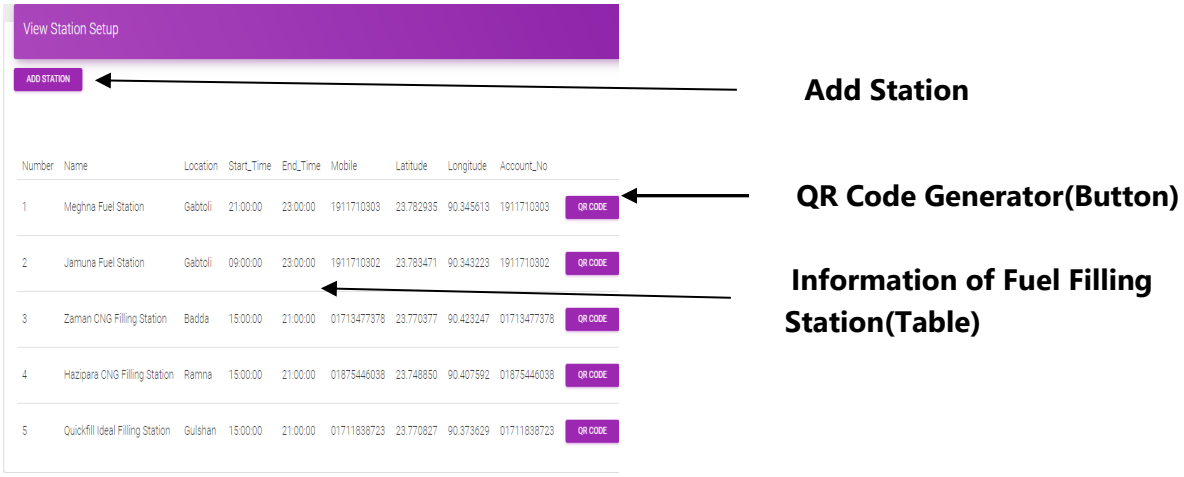


Figure 17.1: Admin Station Setup Interface

17.2 Technologies Overview:

This chapter uses many PHP object-oriented programming capabilities, build in functions, objects, methods, variables and it use CodeIgniter framework and to send values to mobile and make a connection between PHP and Android, JSON is used and also database is used to store the value. In PHP, you'll programmatically interact with Textbox, Button, Table. You'll create these programmatically. You'll use Bootstrap, JavaScript for designing our app. JavaScript is used produce QR code. This QR code is different from each other. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

17.3 Interface of Station Setup:

- This is the app interface with Button for adding new fuel filling station in the system and table for previous history of fuel filling station and also a button for generating QR code for each station.

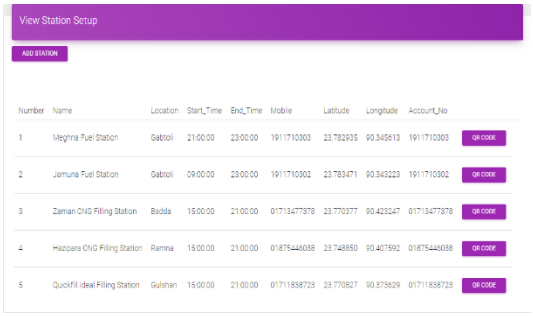


Figure 17.1: Admin Station Setup Interface

17.4 Interface of Add Station:

- This is the app interface with Button for adding new station in the system and Textboxes for writing stations information.

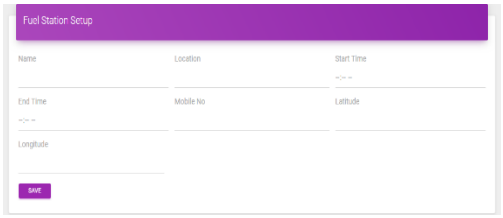
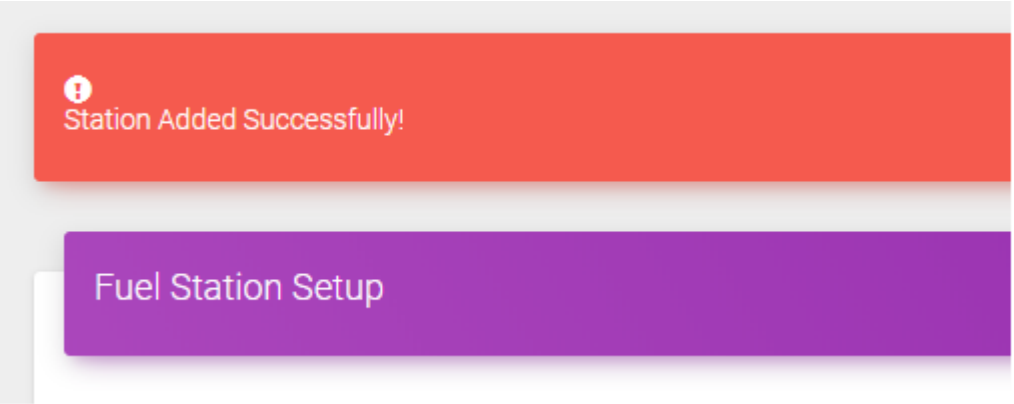


Figure 17.2: Add Station Setup Interface

- When the station's information is successfully added then it gives a message.



17.5 Interface of QR Code:

- This is the app interface where QR code is generated for each fuel filling station and Table for showing particular filling station information.

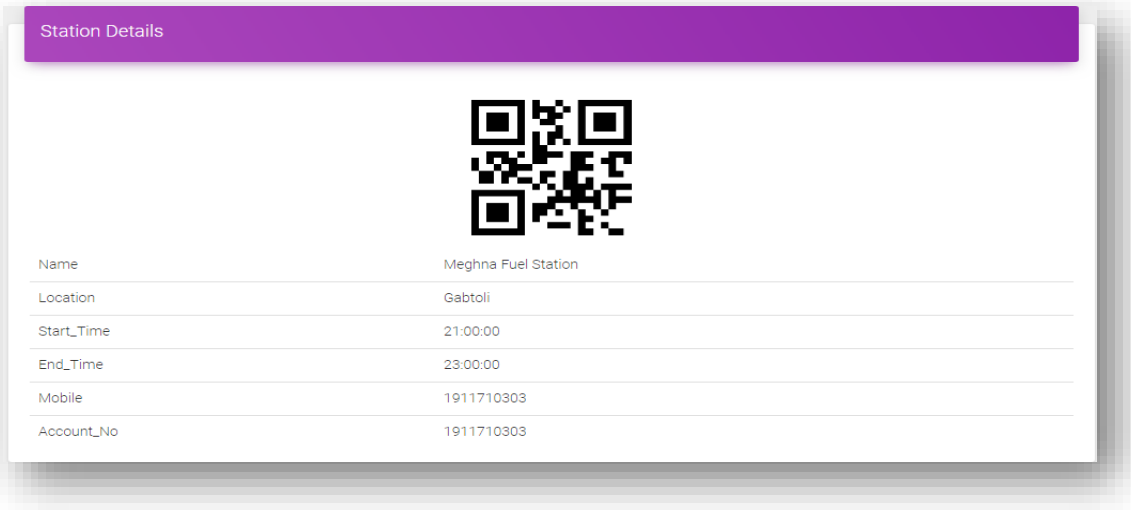


Figure 17.3: Station Details Interface

17.6 Data Flow Diagram for Add Station

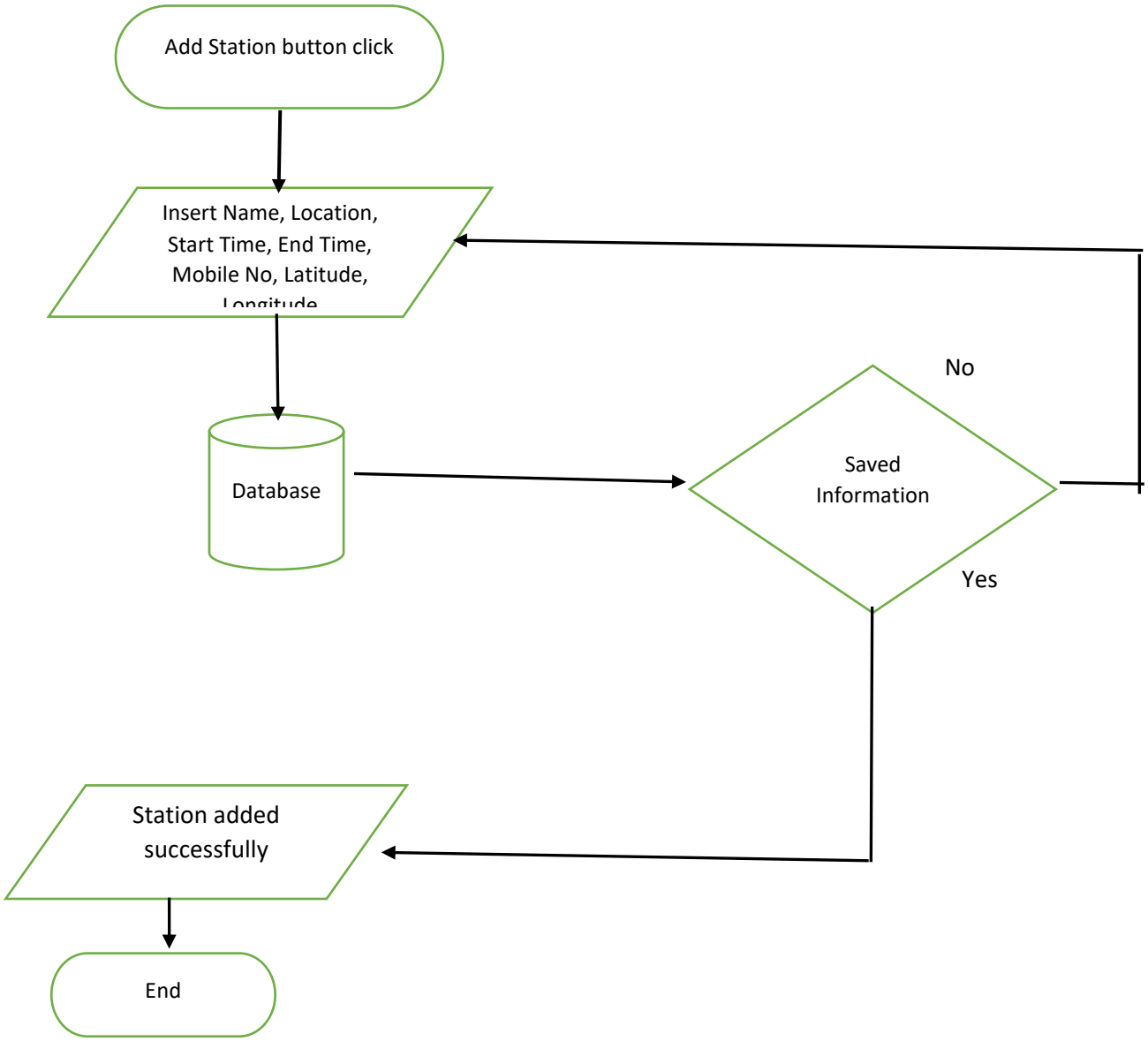


Figure 17.4: Flowchart of Add Station

17.7 Building the app View

In this section, you'll build the View for the **Station Setup**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in viewSetupView.php file

You'll add a Table and Button.

```
<div class="col-sm-12 col-md-12">
    <div class="card">
                                <div    class="card-header"
data-background-color="purple">
                                    <h4
class="title"><?php echo $panelHeading; ?></h4>
                                    <!--                <p
class="category">Here is a subtitle for this table</p> -->
                                </div>

        <form  class="form"  action="<?php  echo  base_url()
?>FuelStationSetup/stationSetup" method="POST">
            <input type="submit" style="margin-left: 15px;"
class="btn btn-primary btn-sm" value="Add Station">
        </form>
        <br><br>
        <div class="card-content table-responsive">
            <table class="table" id="dataTable">
                <thead>
                    <tr>
                        <th>Number</th>
                        <th>Name</th>
                        <th>Location</th>
                        <th>Start_Time</th>
                        <th>End_Time</th>
                        <th>Mobile</th>
                  <th>Latitude</th>
                        <th>Longitude</th>
                        <th>Account_No</th>
                         <th></th>
                        <th></th>


                    </tr>
                </thead>
                <tbody>
                <?php
                    $count=1;
                    foreach ($SetupList as $SetupList) {
                        echo "<tr>";
                            echo "<td>$count</td>";
                            echo
"<td>".$SetupList['name']."</td>";
                            echo
"<td>".$SetupList['location']."</td>";
                            echo
"<td>".$SetupList['start_time']."</td>";

                            echo
"<td>".$SetupList['end_time']."</td>";
                            echo
"<td>".$SetupList['mobile_no']."</td>";
                             echo
"<td>".$SetupList['latitude']."</td>";
                              echo
"<td>".$SetupList['longitude']."</td>";
                                echo
"<td>".$SetupList['account_no']."</td>";
                            echo "<td><a class='btn btn-sm
btn-primary'       href='"       .       base_url()       .
"FuelStationSetup/stationDetails/" .
                            $SetupList['station_id'].
"'>QR Code</a></td>";


                            $count++;
                        echo "</tr>";
                    }
                ?>
                </tbody>
            </table>
        </div>
    </div>


</div>
```

In this section, you'll build the View for the **Add Station**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in stationSetupView.php file

You'll add a Textbox, Label and Button.

```
<div class="col-md-12 col-sm-12 col-xs-12">
    <?php
```

```
    if (!$message == '') {
        ?>
```

```
 <div class="<?= $message == 'Atation Data Added
Successfully!' ? "alert alert-success" : "alert alert-
danger" ?>" role="alert">
            <span class="glyphicon glyphicon-
exclamation-sign" aria-hidden="true"></span>
            <span class="sr-only">Error:</span>
            <?php echo $this->session-
>flashdata('message'); ?>
        </div>
        <?php
    }
    ?>

     <div class="card">
                                <div class="card-header"
data-background-color="purple">
                                    <h4
class="title"><?php echo $panelHeading; ?></h4>
                                    <!-- <p
class="category">Here is a subtitle for this table</p> -
->
                                </div>
        <div class="card-content">

            <form action="<?php echo
base_url()?>FuelStationSetup/saveSetup" method="post">
                <div class="row">
                    <div class="col-md-4">
                        <label>Name</label>
                        <input type="text" class="form-
control" name="StationName" id="UsernameId" required="">
                    </div>

                    <div class="col-md-4">
                        <label>Location</label>
                        <input type="text" class="form-
control" name="Location" id="LocationId" required="">
                    </div>
                    <div class="col-md-4">
                        <label>Start Time</label>
                        <input type="time" class="form-
control" name="StartTime" id="StarttimeId" required="">
                    </div>
                </div>
                <div class="row">

                    <div class="col-md-4">
                        <label>End Time</label>
                        <input type="time" class="form-
control" name="EndTime" id="EndtimeId" required="">
                    </div>
                    <div class="col-md-4">
                        <label>Mobile No</label>
                        <input type="number"
class="form-control" name="MobileNo" id="MobilenoId"
required="">
                    </div>
                    <div class="col-md-4">
                        <label>Latitude</label>
                        <input type="text" class="form-
control" name="Latitude" id="LatitudeId" required="">
                    </div>
                </div>
                <div class="row">

                     <div class="col-md-4">
                        <label>Longitude</label>
                        <input type="text" class="form-
control" name="Longitude" id="LongitudeId" required="">
                    </div>

                </div>

                <!-- <div class="form-group">
                    <input type="button" value="Save"
id="submitBtn" class="btn btn-sm btn-success">
                </div> -->
                <button type="submit" class="btn btn-sm
btn-primary">Save</button>
            </form>
        </div>

</div>

</div>
```

In this section, you'll build the View for the **QR Code**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in viewStationDetails.php file

You'll add a Table and show the values in table

```
<div class="col-sm-12 col-md-12">
    <div class="card">
                                <div class="card-header"
data-background-color="purple">
                                    <h4
class="title"><?php echo $panelHeading; ?></h4>
                                </div>


        <br><br>
```

```
            <div id="station_qr" class="qr_code"
style="width:150px; height:150px;
            margin: 0 auto;"></div>

        <div class="card-content table-responsive">
            <table class="table" id="dataTable">

 <tr>
                    <td>Name</td>
```

```
                    <td><?php echo
$stationDetails['name'];?></td>
                </tr>
                <tr>
                    <td>Location</td>
                    <td><?php echo
$stationDetails['location'];?></td>
                </tr>
                <tr>
                    <td>Start_Time</td>
                    <td><?php echo
$stationDetails['start_time'];?></td>
                </tr>
                <tr>
                    <td>End_Time</td>
                    <td><?php echo
$stationDetails['end_time'];?></td>
                </tr>
                <tr>
                    <td>Mobile</td>
                    <td><?php echo
$stationDetails['mobile_no'];?></td>
```

```
                </tr>
                <tr>
                    <td>Account_No</td>
                    <td><?php echo
$stationDetails['account_no'];?></td>
                </tr>

            </table>
        </div>
    </div>

</div>

    <script>
        new QRCode("station_qr", {
            text: "<?php echo
$stationDetails['station_id'];?>$$",
            colorDark : "#000000",
            colorLight : "#ffffff"
        });

    </script>
```

17.8 PHP Implementation for Station Setup

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

At first, it calls FuelStationSetup.php controller file and in controller it calls viewSetup function. This function communicates with model class to get data. There are a few data get from model class like *'name', 'location', 'start_time', 'end_time', 'mobile_no', 'latitude', 'longitude' and 'account_no''*

- viewSetup in controller class

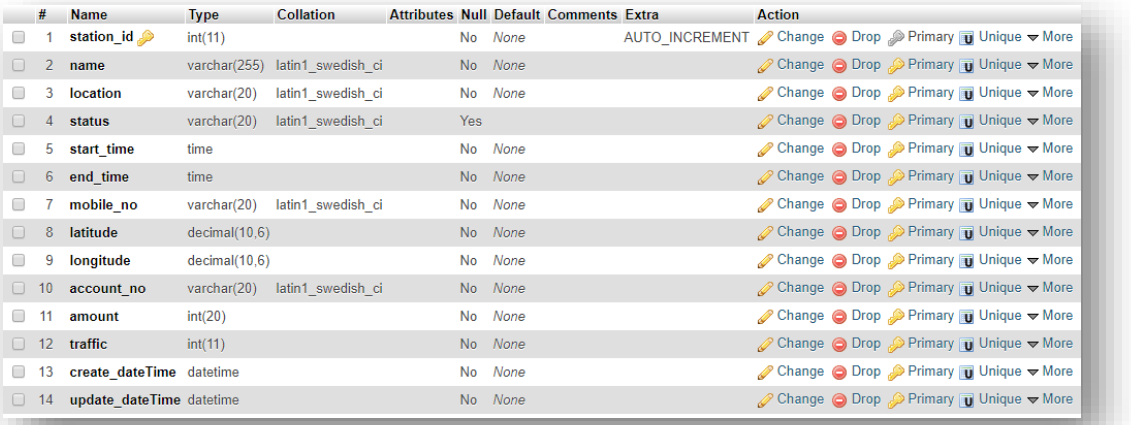
```
public function viewSetup(){
        $data['panelHeading'] = 'View Station Setup';
        $data['bodyTemplate'] = 'viewSetupView';
        $data['message'] = $this->session-
>flashdata('message');
        $data['SetupList'] = $this->Station_model-
>getSetupList();
        $this->load->view('siteTemplate', $data);
    }
```

- getSetupList in model class

```
function getSetupList(){
        $query = $this->db->get('station_setup');
        $result=$query->result_array();

        return $result;
    }
```

Code 17.5 PHP Code of Station Setup

- station_setup database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary Unique More
5	start_time	time			No	None			Change Drop Primary Unique More
6	end_time	time			No	None			Change Drop Primary Unique More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	latitude	decimal(10,6)			No	None			Change Drop Primary Unique More
9	longitude	decimal(10,6)			No	None			Change Drop Primary Unique More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
11	amount	int(20)			No	None			Change Drop Primary Unique More
12	traffic	int(11)			No	None			Change Drop Primary Unique More
13	create_dateTime	datetime			No	None			Change Drop Primary Unique More
14	update_dateTime	datetime			No	None			Change Drop Primary Unique More

17.9 PHP Implementation for Add Station

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

At first, it calls FuelStationSetup.php controller file and in controller file it calls stationSetup function. This function displays the stationSetupView.php. In stationSetupView.php, it again calls saveSetup function in FuelStationSetup.php controller file. This function communicates with model class to insert data. There are a few data that are inserted in database through model class like *'name', 'location', 'start_time', 'end_time', 'mobile_no', 'latitude', 'longitude' and 'account_no''*.

- stationSetup in controller class

```
public function stationSetup(){
        $data['panelHeading'] = 'Fuel Station Setup';
        $data['bodyTemplate'] = 'stationSetupView';
        $data['message'] = $this->session-
>flashdata('message');
```

```
$this->load->view('siteTemplate', $data);

    }
```

- saveSetup in controller class

```
function saveSetup() {

        $taskInfo = array();

        $taskInfo['name'] = $this->input-
>post('StationName',TRUE);
        $taskInfo['location'] = $this->input-
>post('Location',true);
        $taskInfo['start_time'] = $this->input-
>post('StartTime',TRUE);
        $taskInfo['end_time'] = $this->input-
>post('EndTime',TRUE);
        $taskInfo['mobile_no'] = $this->input-
>post('MobileNo',TRUE);
        $taskInfo['Latitude'] = $this->input-
>post('Latitude',TRUE);
        $taskInfo['Longitude'] = $this->input-
>post('Longitude',TRUE);
        $taskInfo['account_no'] = $this->input-
>post('MobileNo',TRUE);
        $taskInfo['status'] = "Active";
        $taskInfo['amount'] = 0;
        $taskInfo['traffic'] = 0;
        $date = date("Y-m-d H:i:s");
        $taskInfo['create_dateTime'] = $date;
        $taskInfo['update_dateTime'] = $date;


        $this->load->library('form_validation');

        $this->form_validation->set_rules('name', 'Station
Name', 'required|callback___checkStationName');
        $this->form_validation->set_rules('mobile_no', 'Mobile
No', 'required|callback___checkMobileNo');
//        $this->form_validation->set_rules('password',
'Password', 'required');
//        $this->form_validation->set_rules('contact_no',
'Contact No:', 'required|integer');
//        $this->form_validation->set_rules('email', 'Email',
'required');

        if ($this->form_validation->run() == FALSE) {

            $json = array(
                "success" => false,
                "msg" => validation_errors('<p>', '</p>')
            );

            echo json_encode($json);
           // $this->session->set_flashdata('message', 'Data
insert error');
           // redirect('FuelStationSetup/stationSetup');
           // die();
        }
//        $date = date("Y-m-d H:i:s");
//
//        $docData = array(
//            "username" => (int) $folder_id,
//            "folder_name" => $folder_name,
//            "machine_name" => $machine_name,
//            "parent_id" => (int) $parent_id,
//            "created" => $date,
//            "updated" => $date
//        );

        $rDocument = $this->Station_model-
>saveSetup($taskInfo);

//        echo json_encode($rDocument);
        $this->session->set_flashdata('message', 'Station Added
Successfully!');
            redirect('FuelStationSetup/stationSetup');
            die();

    }

    function __checkStationName($name)
    {
        $this->db->where("name",$name);
//        if((int)$this->uri->segment(4) > 0):
//            $this->db->where("user_id <>",(int)$this->uri-
>segment(4));
//        endif;
        $result = $this->db->get('station_setup');

        if($result->num_rows() > 0):
            $this->form_validation-
>set_message("__checkUserName","Please select a different
station name.");
            return false;
        endif;
        return true;
    }

    function __checkMobileNo($mobile)
    {
        $this->db->where("mobile_no",$mobile);
        $result = $this->db->get('station_setup');

        if($result->num_rows() > 0):
            $this->form_validation-
>set_message("__checkMobileNo","Please select a different
mobile number.");
            return false;
        endif;
        return true;
    }
```

- saveSetup in model class

```
function saveSetup($data = array())
{

 $this->db->insert('station_setup', $data);

    }
```

Code 17.6 PHP Code of Add Station Setup

- station_setup database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary Unique More
5	start_time	time			No	None			Change Drop Primary Unique More
6	end_time	time			No	None			Change Drop Primary Unique More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	latitude	decimal(10,6)			No	None			Change Drop Primary Unique More
9	longitude	decimal(10,6)			No	None			Change Drop Primary Unique More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
11	amount	int(20)			No	None			Change Drop Primary Unique More
12	traffic	int(11)			No	None			Change Drop Primary Unique More
13	create_dateTime	datetime			No	None			Change Drop Primary Unique More
14	update_dateTime	datetime			No	None			Change Drop Primary Unique More

17.10 PHP Implementation for QR Code

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

At first, it calls FuelStationSetup.php controller file and in controller file it calls stationDetails function. This function communicates with model class to bring data. In viewStationDetails.php file, QR code are generated by the help of Java Script and display the specific fuel station information in table. There some data that are bring from database like *'name', 'location', 'start_time', 'end_time', 'mobile_no', and 'account_no'*.

- stationDetails in controller class

```
public function stationDetails($stationId){
        $data['panelHeading'] = 'Station Details';
        $data['bodyTemplate'] = 'viewStationDetails';
        $data['message'] = $this->session-
>flashdata('message');

        $taskResult = $this->Station_model-
>getStationDetails($stationId);
        //var_dump($taskResult);
        //die();
        $data['stationDetails'] = $taskResult;

        $this->load->view('siteTemplate', $data);
    }
```

- getStationDetails in model class

```
function getStationDetails($params) {

        $this->db->select('*');
        $this->db->where("station_id", $params);
        $query = $this->db->get('station_setup');


        if ($query->num_rows() > 0) {
            return $query->row_array();
        } else {
            return 0;

 }
    }
```

Code 17.7 PHP Code of QR code

- station_setup database table of MySQL

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	station_id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	location	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	status	varchar(20)	latin1_swedish_ci		Yes				Change Drop Primary Unique More
5	start_time	time			No	None			Change Drop Primary Unique More
6	end_time	time			No	None			Change Drop Primary Unique More
7	mobile_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	latitude	decimal(10,6)			No	None			Change Drop Primary Unique More
9	longitude	decimal(10,6)			No	None			Change Drop Primary Unique More
10	account_no	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
11	amount	int(20)			No	None			Change Drop Primary Unique More
12	traffic	int(11)			No	None			Change Drop Primary Unique More
13	create_dateTime	datetime			No	None			Change Drop Primary Unique More
14	update_dateTime	datetime			No	None			Change Drop Primary Unique More

ADMIN FUEL RATE

Add Fuel Rate and Display

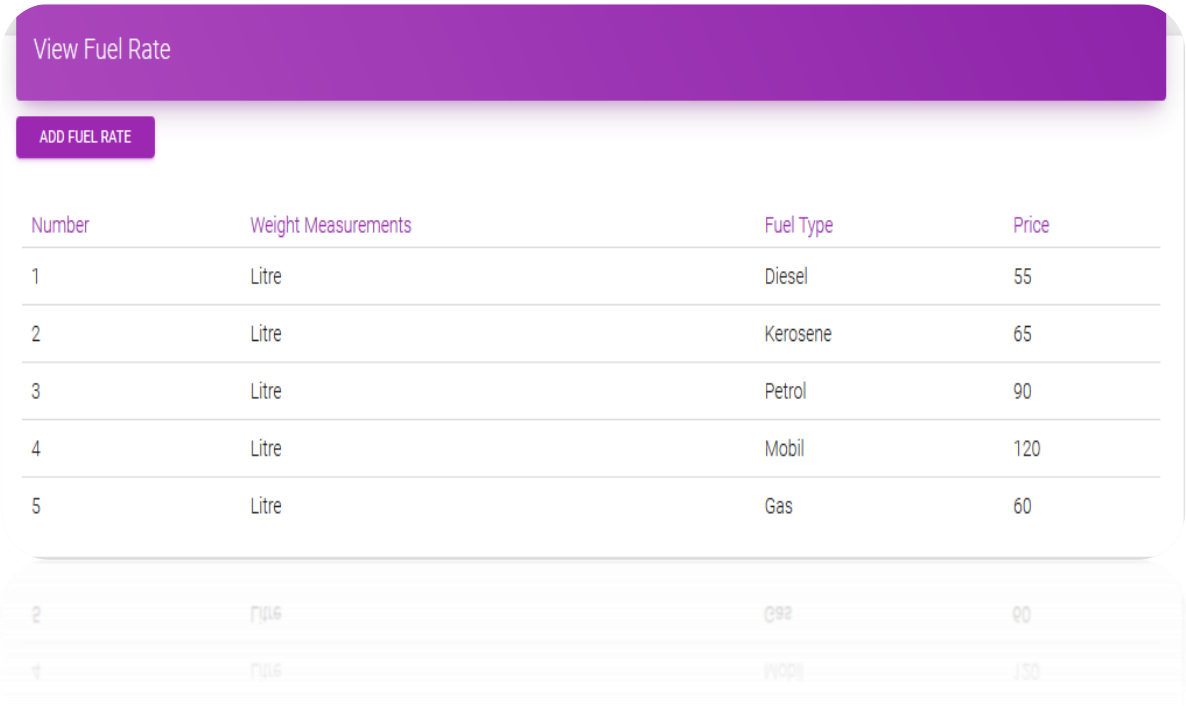


Figure 18.1: Admin Fuel Rate Setup Interface

18.1 Introduction:

It is a page where you can add or determine fuel's rate and display them.

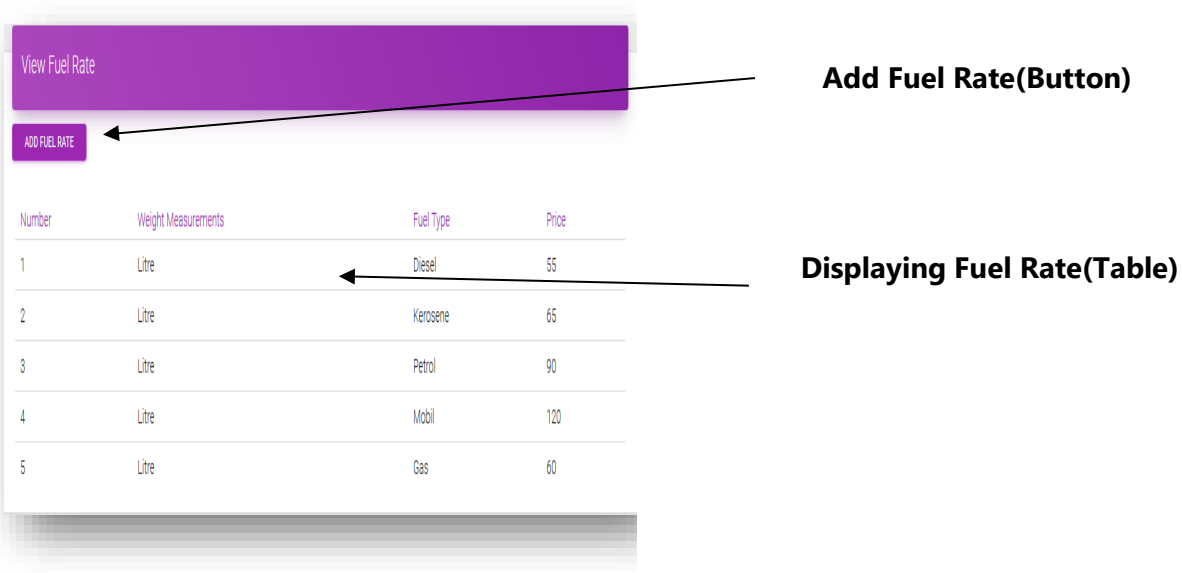


Figure 18.1: Admin Fuel Rate Setup Interface

18.2 Technologies Overview:

This chapter uses many PHP object-oriented programming capabilities, build in functions, objects, methods, variables and it use CodeIgniter framework and to send values to mobile and make a connection between PHP and Android, JSON is used and also database is used to store the value. In PHP, you'll programmatically interact with Textbox, Button, Table. You'll create these programmatically. You'll use Bootstrap, JavaScript for designing our app. JavaScript is used produce QR code. This QR code is different from each other. In PHP, as CodeIgniter framework is used it follow model view controller (MVC) concept. It first goes to controller through API. Controller catch the value and sent it to model. It validates the value from database and sent back to controller. Then controller sent it to mobile as a form of JSON. JSON take the value in the form of JSON array with a key value.

18.3 Interface of Fuel Rate:

- This is the app interface with Button for adding rate of fuel in the system and table for showing all types of fuel rate.

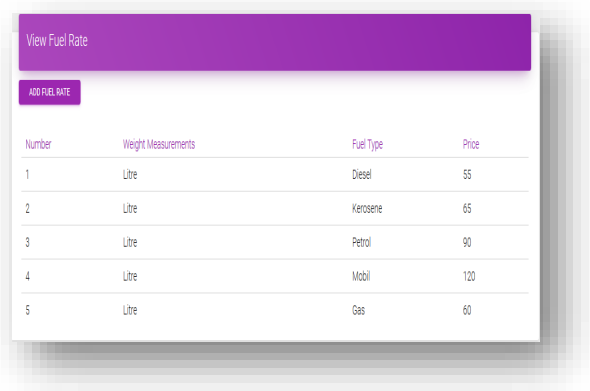


Figure 18.1: Admin Fuel Rate Setup Interface

18.4 Interface of Add Fuel Rate:

- This is the app interface with Button for adding rate of fuel in the system and Textboxes for writing information.

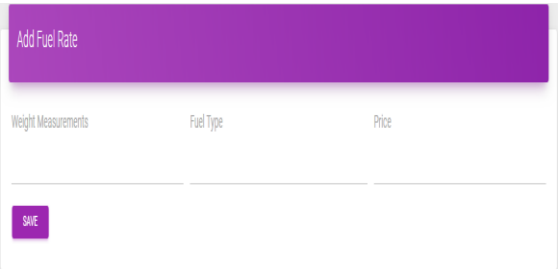
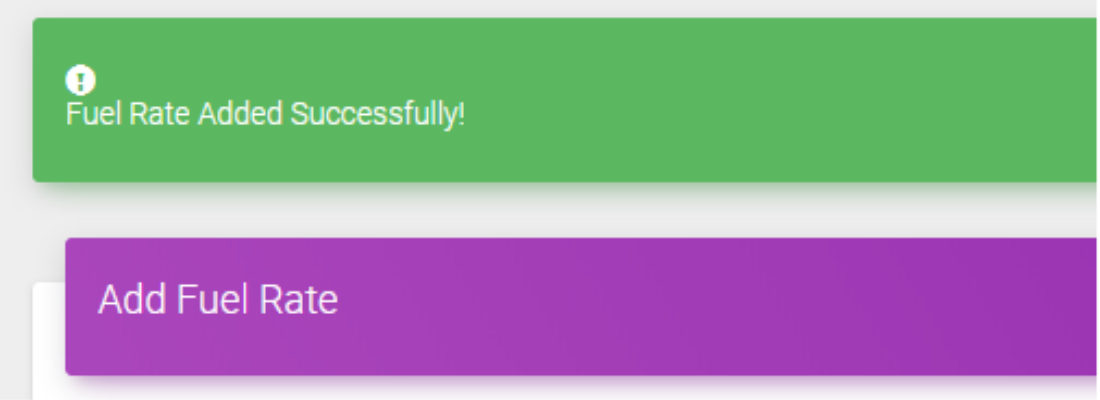


Figure 18.2: Add Fuel Rate Interface

- When the fuel rate is successfully added then it gives a message.



18.5 Data Flow Diagram for Add Fuel Rate

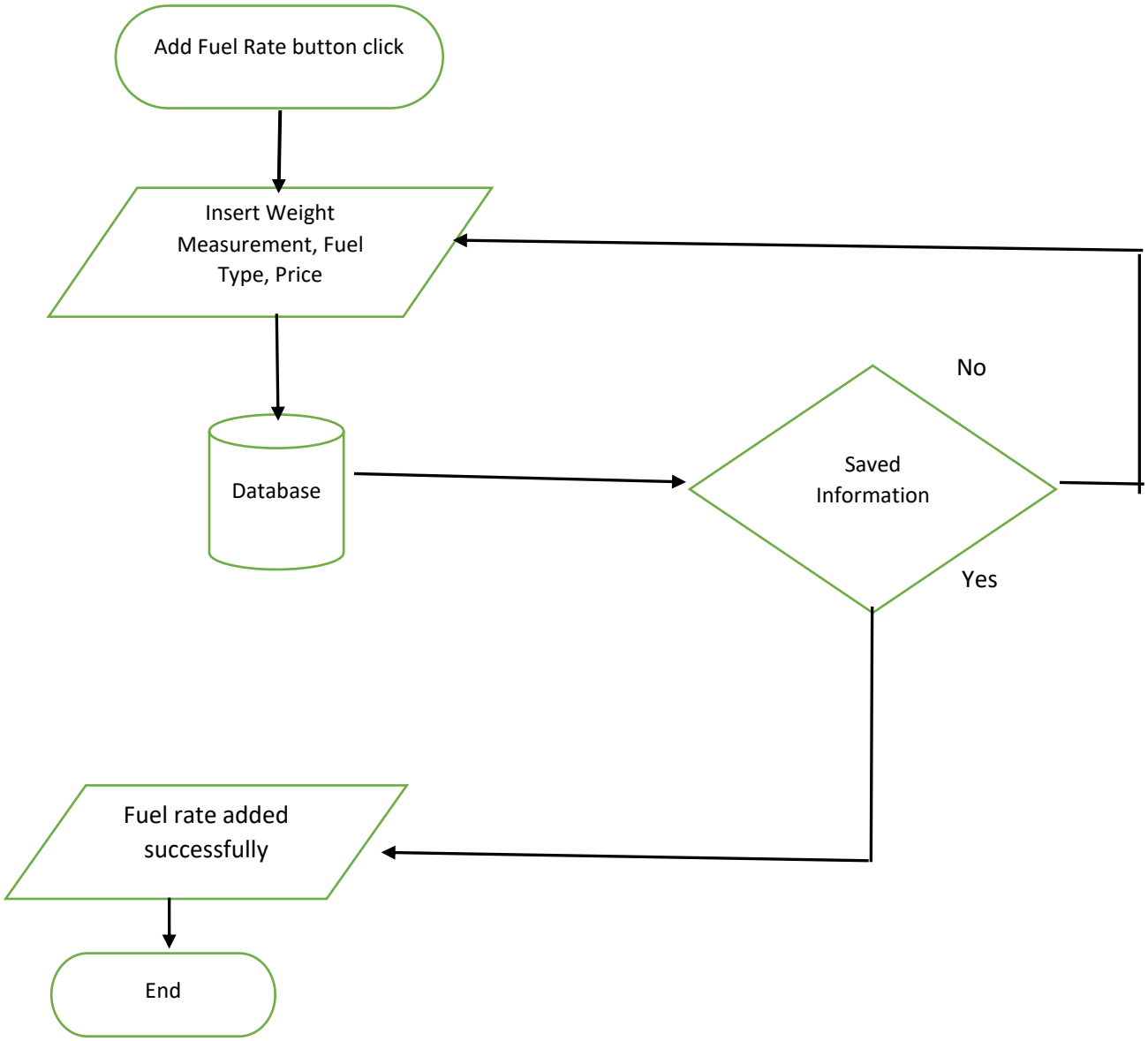


Figure 18.3: Flowchart of Add Fuel Rate

18.6 Building the app View

In this section, you'll build the View for the **Fuel Rate**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in viewFuelRateView.php file

You'll add a Table and Button.

```
<div class="col-sm-12 col-md-12">
    <div class="card">
                                <div class="card-header"
data-background-color="purple">
                                    <h4
class="title"><?php echo $panelHeading; ?></h4>
                                    <!-- <p
class="category">Here is a subtitle for this table</p> -
->
                                </div>

        <form class="form" action="<?php echo base_url()
?>FuelRate/addRate" method="POST">
            <input type="submit" style="margin-left:
15px;" class="btn btn-primary btn-sm" value="Add Fuel
Rate">
        </form>
        <div class="card-content table-responsive">
            <table class="table" id="dataTable">
                <thead class="text-primary">
                    <tr>
                        <th>Number</th>
                        <th>Weight Measurements</th>
                        <th>Fuel Type</th>
                        <th>Price</th>
                    </tr>
                </thead>
                <tbody>
                <?php
                    $count=1;
                    foreach ($FuelRates as $FuelRate) {
                        echo "<tr>";
                            echo "<td>$count</td>";
                            echo
"<td>".$FuelRate['weight_measurements']."</td>";
                            echo
"<td>".$FuelRate['fuel_type']."</td>";
                            echo
"<td>".$FuelRate['amount']."</td>";

                            $count++;
                        echo "</tr>";
                    }
                  ?>
                </tbody>
            </table>
        </div>
    </div>

</div>
```

In this section, you'll build the View for the **Add Fuel Rate**. At the end of this section, we'll present the View code for this module's layout.

Adding the Components in addFuelRateView.php file

You'll add a Textbox, Label and Button.

```
<div class="col-md-12 col-sm-12 col-xs-12">
    <?php
    if (!$message == '') {
        ?>

        <div class="<?= $message == 'Fuel Rate Added
Successfully!' ? "alert alert-success" : "alert alert-
danger" ?>" role="alert">
            <span class="glyphicon glyphicon-
exclamation-sign" aria-hidden="true"></span>
            <span class="sr-only">Error:</span>
            <?php echo $this->session-
>flashdata('message'); ?>
        </div>
        <?php
    }
    ?>
    <div class="card">
                                <div class="card-header"
data-background-color="purple">
                                    <h4
class="title"><?php echo $panelHeading; ?></h4>
```

```
                                    <!-- <p
class="category">Here is a subtitle for this table</p> -
->
                                </div>
    <div class="card-content">
    <form action="<?php echo
base_url()?>FuelRate/saveRate" method="post">
        <div class="row">
            <div class="col-md-4">
                <label>Weight Measurements</label>
                <input type="text" class="form-control"
name="weight" id="UsernameId" required="">
            </div>
            <div class="col-md-4">
                <label>Fuel Type</label>
                <input type="text" class="form-control"
name="fuel_type" id="PasswordId" required="">
            </div>
            <div class="col-md-4">
                <label>Price</label>
                <input type="text" class="form-control"
name="amount" id="fullNameId" required="">
            </div>
        </div>

        <button type="submit" class="btn btn-sm btn-
primary">Save</button>
    </form>
</div>

</div>

</div>
```

18.7 PHP Implementation for Fuel Rate

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

At first, it class FuelRate.php controller file and in controller it calls viewFuelRate function. This function communicates with model class to get data which shows in view. There are a few data get from model class like *'weight_measurement', 'fuel_type', 'amount.'*

- viewFuelRate in controller class

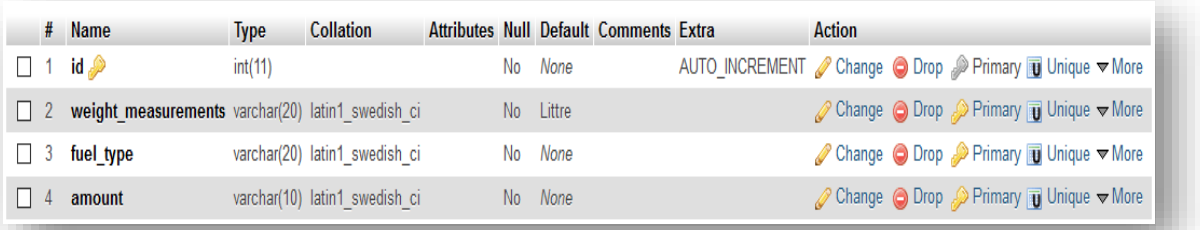
```
public function viewFuelRate(){
        $data['panelHeading'] = 'View Fuel Rate';
        $data['bodyTemplate'] = 'viewFuelRateView';
        $data['message'] = $this->session-
>flashdata('message');
        $data['FuelRates'] = $this->Rate_model-
>getRateList();

        $this->load->view('siteTemplate', $data);
    }
```

- getRateList in model class

```
function getRateList(){
        $query = $this->db->get('fuel_rate');
        $result=$query->result_array();
return $result;
    }
```

Code 18.4 PHP Code of Fuel Rate

- fuel_rate database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	weight_measurements	varchar(20)	latin1_swedish_ci		No	Litre			Change Drop Primary Unique More
3	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	amount	varchar(10)	latin1_swedish_ci		No	None			Change Drop Primary Unique More

18.8 PHP Implementation for Add Fuel Rate

PHP operates on the Model View Controller (MVC) fundamentals. CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. Here controller classes are necessary part of development under CodeIgniter. Controller class hold data which it gets from model class and sent it to app.

At first, by clicking button it calls FuelRate.php controller file and in controller file it calls addRate function. This function displays the addFuelRateView.php. In addFuelRateView.php, it again calls saveRate function in FuelRate.php controller file. This function communicates with model class to insert data. There are a few data that are inserted in database through model class like *'weight_measurement', 'fuel_type', 'amount.*

- addRate in controller class

```
public function addRate(){
        $data['panelHeading'] = 'Add Fuel Rate';
        $data['bodyTemplate'] = 'addFuelRateView';
        $data['message'] = $this->session-
>flashdata('message');
```

```
$this->load->view('siteTemplate', $data);
    }
```

- saveRate in controller class

```
function saveRate() {

        $taskInfo = array();

        $taskInfo['weight_measurements'] = $this->input-
>post('weight',TRUE);
        $taskInfo['fuel_type'] = $this->input-
>post('fuel_type',true);
```

```
        $taskInfo['amount'] = $this->input-
>post('amount',TRUE);

        $this->load->library('form_validation');

        $this->form_validation->set_rules('fuel_type', 'Fuel
Type', 'required|callback___checkFuelType');
```

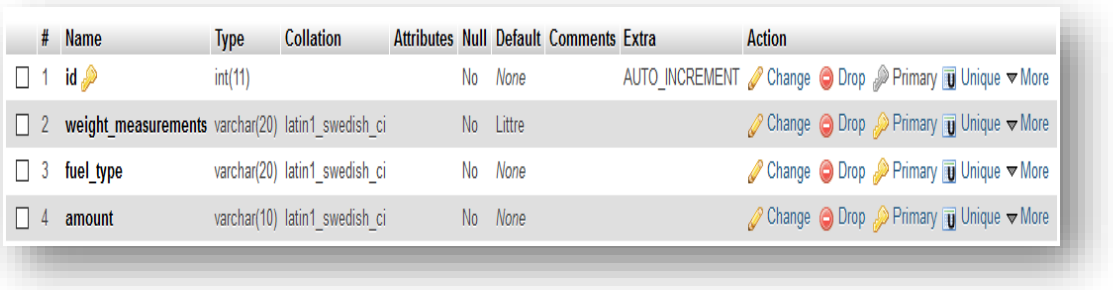
```
        if ($this->form_validation->run() == FALSE) {

            $json = array(
                "success" => false,
                "msg" => validation_errors('<p>', '</p>')
            );

            $this->session->set_flashdata('message', 'Fuel Type
is in Use');
            redirect('FuelRate/addRate');
            die();
        }


        $rDocument = $this->Rate_model-
>saveFuelRate($taskInfo);


        $this->session->set_flashdata('message', 'Fuel Rate
Added Successfully!');
            redirect('FuelRate/addRate');
            die();

    }



    function __checkFuelType($FuelType)
    {
        $this->db->where("fuel_type",$FuelType);
        $result = $this->db->get('fuel_rate');

        if($result->num_rows() > 0):
            $this->form_validation-
>set_message("__checkFuelType","Please select a different fuel
type.");
            return false;

endif;
        return true;
    }
```

Code 18.5 PHP Code of Add Fuel Rate

- saveRate in model class

```
function saveFuelRate($data = array()) {
        $this->db->insert('fuel_rate', $data);
    }
```

- fuel_rate database table of MySQL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	weight_measurements	varchar(20)	latin1_swedish_ci		No	Littre			Change Drop Primary Unique More
3	fuel_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	amount	varchar(10)	latin1_swedish_ci		No	None			Change Drop Primary Unique More

Conclusion and Future Work

After all of this description of our project, we feel satisfy to successfully conclude this project. Throughout the development of this project we've learnt a lot. After a lot of R&D we can come up some of our unique features and merge them together where now the application is working really fine. We also did a lot of testing to check our application performance and it works very well.

Throughout the time, we've developed this application, there are some ideas that came during this project was progressing. Due to lack of time and knowledge we can't developed the following things. Later we will work on this project, make it more resourceful and will include following features

1. **Payment Security:**
   Security is the primary factor of every payment system. If we integrate 2fa in our system, it will be more secure. When someone will login our system or payment through our system, before that user can get a SMS or email in their registered mobile number or email address, if anyone know others password or user name he/she can't pay fuel bill because only register user gets SMS or email.
2. **Account Upgradation:**
   In future we will integration with bank account or credit card thus user will add his personal account into our system and pay their fuel bill using those account. Now user only pay their bill in our test basis virtual account but in real life they can be use virtual account, bank account or credit card.
3. **Fuel Station App:**
   In our existing app after payment only user can get notify but fuel station user gets no confirmation. We will build an app for fuel station user that they can manage their account and get notify for each transection.
4. **Dynamic QR generation:**
   Now when someone pay their bill they must enter their amount. By mistake the amount may be mismatched thus we will build a dynamic QR in fuel station app. After taking fuel station user input desire amount or it should be get from the system then a QR code generated by the fuel station users end. After that user will scan the QR code get fuel station information with amount and press payment button, in this case no amount input can be needed from the user.
5. **Fuel Remainder:**
   Before getting reminder, user must input their rest of fuel (Littre). In future we will instigate with vehicle information system to get current fuel, after that our system will

automatically detect users can go their destination or not. However, we also integrate google location base service how much path user already drive and how much kilometer they can drive.

6. **Auto Detect Nearest Fuel Station:**
   User will set an auto reminder after finishing the fuel. when the fuel will be finish it will also show the nearest fuel station.

References

[1] " Project graphical interface design ." Available at: *Freepik.com.* www.freepik.com [Accessed in 2018].

[2] "Android DataBinding in RecyclerView – Profile Screen.” *Androidhive.* Available at : https://www.androidhive.info [Accessed in 2017].

[3] "Android PhoneStateListener/ Phone Call Broadcast Receiver Tutorial." *Study Tutorial.* Available at: https://www.studytutorial.in/android-phonestatelistener-phone-call-broadcast-receiver-tutorial[Accessed in 2018].

[4] "Android reject call." *worldbestlearningcenter.* Available at: http://www.worldbestlearningcenter.com/tips/Android-reject-call.htm[Accessed in 2018].

[5] "Android Tutorial." *Tutorials Point.* Available at: https://www.tutorialspoint.com/android/index.htm[Accessed in 2017].

[6] "Android working with Card View and Recycler View." *Androidhive*. Available at: https://www.androidhive.info/2016/05/android-working-with-card-view-and-recycler-view/[Accessed in 2017].

[7] "android-login-screen." *Github.com.* Available at: https://github.com/ymittal/android-login-screen[Accessed in 2017].

[8] "CodeIgniter Tutorial." *Tutorials Point.* Available at: https://www.tutorialspoint.com/codeigniter/index.htm[Accessed in 2017].

[9] "Device art generator." *Developer.* Available at: www.develper.android.com[Accessde in 2017].

[10] "Google Map API key Creation." *Google Cloud Platform.* Available at: https://console.cloud.google.com/api [Accessed in 2018].

[11] "Google Map Demo." *Github.com.* Available at: https://github.com/dharavp/GoogleMap-Demo [Accessed in 2018].

[12] "How to use search functionality in custom list view in Android." *stack overflow.* Available at: https://stackoverflow.com/questions/14118309/how-to-use-search-functionality-in-custom-list-view-in-android/14119383#14119383[Accessed in 2017].

[13] "Introducing JSON." Available at: https://www.json.org/[Accessed in 2017].

[14] "JSON - Introduction." *w3schools.com.* Available at: https://www.w3schools.com/js/js_json_intro.asp[Accessed in 2017].

[15] *Material Dashboard.*Available at: https://materialdesignthemes.com/themes/material-dashboard/[Accessed in 2017].

[16] Murphy, Mark L. 2011. *The Busy Coder's Guide to Android.* United States Of America. Available at: https://commonsware.com/Android/Android_3-6-CC.pdf[Accessed in 2017]

[17] *QR Code Generator.* Available at: https://www.the-qrcode-generator.com/[Accessed in 2018].

[18] "QR-Lite." *Github.com.* Available at: https://github.com/ephrine/QR-Lite[Accessed in 2018].

[19] "Sending SMS programmatically without opening message app." *Stack overflow.* Available at: https://stackoverflow.com/questions/26311243/sending-sms-programmatically-without-opening-message-app[Accessed in 2018].

[20] Simon, Jonathan. 2011. *Head first android development.* United States of America: O'Reilly. Media, Inc., 1005 Gravenstein Highway North, Sebastopol. Availabe at: ftp://ftp.micronet-rostov.ru/linux-support/books/programming/Mobile-Apps/Oreilly.Head.First.Android.Development.Jul.2012.pdf[Accessde in 2017]

[21] "Sweet Alert Dialog." *Github.com.* Available at: https://github.com/pedant/sweet-alert-dialog[Accessed in 2017].

[22] "UI Icon." *FlatIcon.* Available at : https://www.flaticon.com/[Accessed in 2017].

[23] "What is Android" Wikipidiea. Available at: https://en.wikipedia.org/wiki/Android_(operating_system)[Accessed in 2017]

[24] "Android Developer". Available at : https://developer.android.com/index.html[Accessed in 2017]

[25] "What is PHP" Available at: http://php.net/manual/en/intro-whatis.php[Accessed in 2017]

[26] "PHP" Wikipidiea. Available at: https://en.wikipedia.org/wiki/PHP[Aceessed in 2017]

[27] "Model View Controller". Available at: https://en.wikipedia.org/wiki/PHP[Accessed in 2017]

[28] "PHP Framework". Available at: https://onextrapixel.com/an-overview-of-php-framework-guides-for-developers/[Accessed in 2017]