चयनित फसलों के निकटवर्ती मंडियों और मूल्य पूर्वानुमान के लिए एक मोबाइल ऐप का विकास

# Development of a Mobile App for Locating Nearby Mandis and Price Forecasts of Selected Agricultural Commodities


**By**

**BANOTH JAGDISH NAIK**


**MASTER OF SCIENCE**
**IN**
**COMPUTER APPLICATION**


**ICAR-INDIAN AGRICULTURAL STATISTICS RESEARCH INSTITUTE**

**ICAR-INDIAN AGRICULTURAL RESEARCH INSTITUTE**

**NEW DELHI – 110012**



**2019**

चयनित फसलों के निकटवर्ती मंडियों और मूल्य पूर्वानुमान के लिए एक मोबाइल ऐप का विकास

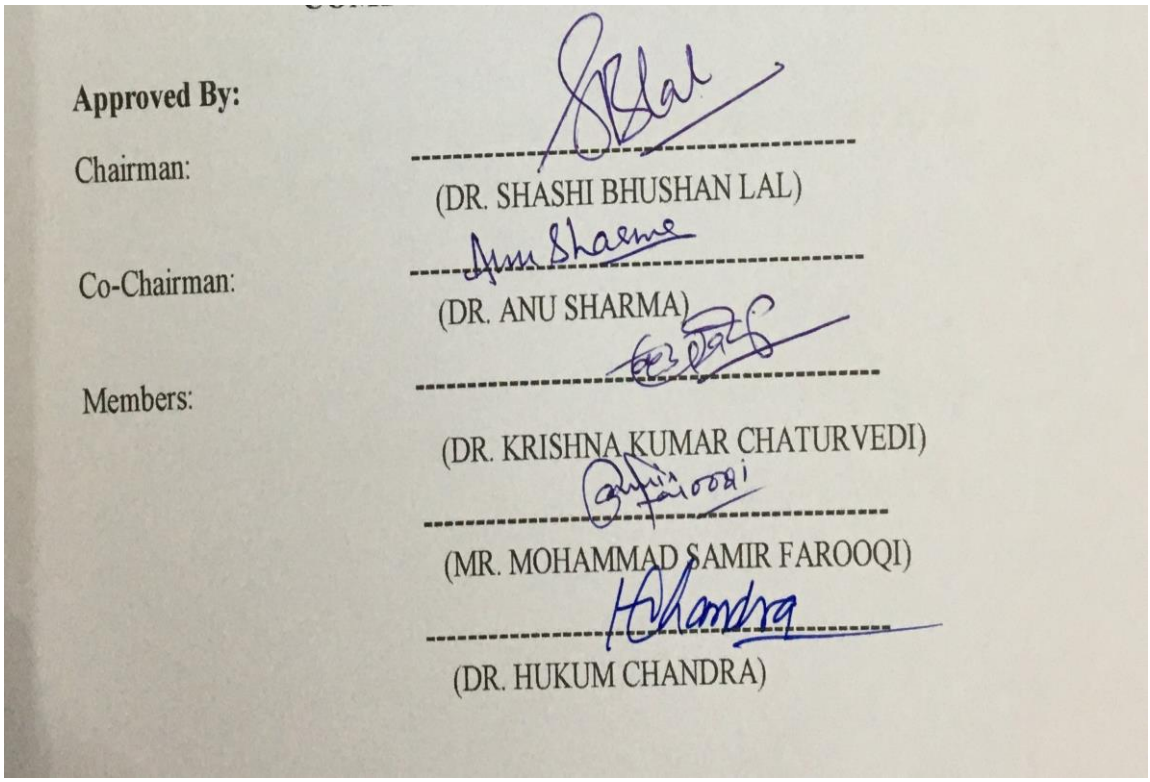
# Development of a Mobile App for Locating Nearby Mandis and Price Forecasts of Selected Agricultural Commodities

**By**

**BANOTH JAGDISH NAIK**

*A thesis submitted to the Faculty of the Post-Graduate School,*
*ICAR-Indian Agricultural Research Institute, New Delhi,*
*In partial fulfillment of the requirements for the degree of*

**MASTER OF SCIENCE**
**In**
**COMPUTER APPLICATION**



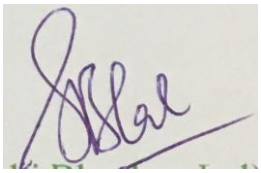
**ICAR- INDIAN AGRICULTURAL STATISTICS RESEARCH INSTITUTE, LIBRARY AVENUE, PUSA, NEW DELHI-110012**

**Dr. Shashi Bhushan Lal**
**Senior Scientist**

# CERTIFICATE

*This is to certify that the work assimilated in the thesis entitled* ***"Development of a Mobile App for Locating Nearby Mandis and Price Forecasts of Selected Agricultural Commodities"*** *submitted in partial fulfilment of the concern for the degree of* ***Master of Science in Computer Application*** *of the* ***Post-Graduate School, Indian Agricultural Research Institute, New Delhi****, is a record of bonafide research carried out by* ***Mr. Banoth Jagdish Naik*** *under my supervision and guidance. No part of this thesis has been submitted for any other degree or diploma.*

*I also certify that all collaboration and assistance received during the course of this analysis has been duly acknowledged.*



Date:12/12/2019
Place: ICAR-IASRI, New Delhi

(Shashi Bhushan Lal)
Chairperson
Advisory Committee

# *ACKNOWLEDGEMENTS*

*This thesis is the fruitful outcome of the knowledge gained over the entire period of my M.Sc. study at ICAR-Indian Agricultural Statistics Research Institute (ICAR-IASRI), New Delhi during which I have been in touch with many number of people whose contribution in varied yet myriad ways has led to the research and making of the thesis which deserves special mention. It is a pleasure to convey my gratitude to all of them by way of my humble acknowledgements.*

*First and foremost, with reverence, I want to express deepest sense of gratitude to* ***Dr. Shashi Bhushan Lal,*** *Senior Scientist, Centre for Agricultural Bioinformatics (CABin), ICAR-IASRI, New Delhi and Chairman of my Advisory Committee for his initiative, benevolence, endurance, constructive criticism and constant monitoring during the period of my study and also during preparation of this thesis. Above all and most needed, he provided me constant support and encouragement in various ways. I consider myself blessed having the privilege of being guided by him. I am really indebted to him.*

*I am also equally indebted to* ***Dr. (Mrs.) Anu Sharma****, Scientist, Centre for Agricultural Bioinformatics, ICAR-IASRI and Co-Chairman of my advisory committee for her moral support during the course of my research work. She gave me constant encouragement from time to time for completion of my research work. I am really thankful to her for providing her valuable time and helping me in understanding the basic concepts regarding my research work. Without her support and encouragement, it would have been quite difficult for me to reach the goal in time.*

*It is great privilege for me to express my esteem and profound sense of gratitude to* ***Dr. Krishna Kumar Chaturvedi****, Senior Scientist, Centre for Agricultural Bioinformatics, ICAR-IASRI, New Delhi and member of my Advisory Committee for his valuable suggestions and help.*

*It is great privilege for me to express my esteem and profound sense of gratitude to* ***Md. Samir Farooqi****, Scientist, Centre for Agricultural Bioinformatics, ICAR-IASRI, New Delhi and member of my Advisory Committee for his valuable suggestions and help.*

*I would also like to express my sincere thanks to* ***Dr. Hukum Chandra,*** *National Fellow and Principal Scientist, ICAR-IASRI, New Delhi and member of my Advisory Committee. I am really grateful to him for the support and encouragement he provided during the course of my study.*

*I would like to convey my deep sense of gratitude and appreciation to* ***Dr. Sudeep****, Professor & Head, Division of Computer Application, ICAR-IASRI, New Delhi, for his help and support during my entire course work. I am indebted to him for his valuable advice and constructive criticism during preparation of this thesis.*

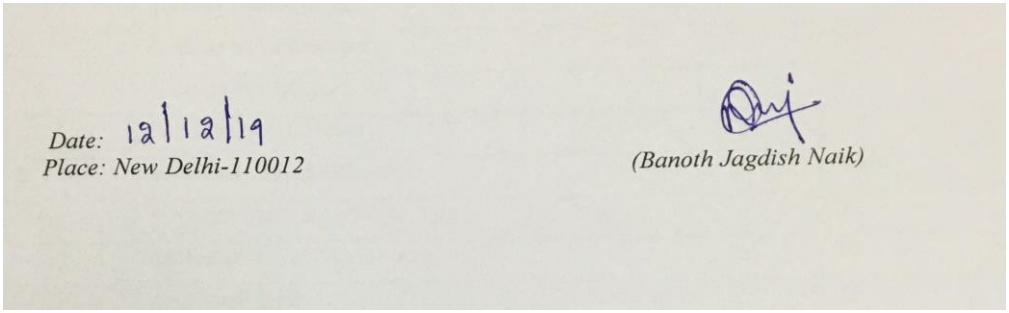
*I am deeply indebted to* ***Dr. Lalmohan Bhar****, Director, ICAR-IASRI for the help and infrastructure facilities provided by him.*

*I am indebted to* ***Dean and Joint-Director (Education)****, PG School, IARI, New Delhi for providing facilities to carry out this research work. I like to thank all the staff of PG School for their helpful attitude and cooperation, throughout the period of my study.*

*I want to express my respect and gratitude to my family, especially* ***Baba, Yaadi, Bhaiya, Bhai, Mounika*** *for their extreme love and support during this work. These were the inspirations and moral boosting which gave me sufficient energy to complete this thesis in time. I would like to convey my special thanks to my friend* ***Manoj*** *and my senior* ***Ashraful Sir*** *for their constant support and encouragement during the entire period of my study. I am also thankful to all my batchmates specially* ***Amit Saha, Abhishek, Rohit, Lishi, Vinayaka, Vinay, Rahul, Debopam, Krishna, Ankita, Tanima, Baibhav, Nitesh, Jutan, Naveen and Tanwy*** *for their friendly approach and moral support throughout the period of my study. I do not have words to express the help and support given by all my seniors specially* ***Srikanth Sir, Vaiju Sir, Tanuj Sir, Ramesh Sir, Asit Sir, Arpan Sir, Himanshu Sir, Dilip Sir, Asif Sir, Nitin Sir, Laxmi Ma'am, Shbana Ma'am, Sonica Ma'am and Kuldeep Sir.*** *I also express my special thanks to all my juniors for their affectionate support and help.*

*I take this opportunity to appreciate the help rendered by the staff of Training and Administration (TAC), and CABin, ICAR-IASRI with special thanks to* ***Sanjeev ji, Sunil ji and Gagan ji****.*

*Last, but not the least, I am thankful to ICAR-IASRI for the financial assistance provided to me in the form of fellowship during the tenure of my study. Finally, I would like to thank everybody who was imperative to the successful realization of this thesis, as well as articulate my apology that I could not mention personally one by one.*



# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER I

# INTRODUCTION

## 1.1 Background

Agriculture is one of the most important and ancient occupation of human beings. India is specially known for agriculture. It has a major role in country's financial status. Over 70% of the rural families in India have agriculture as their main source of livelihood [Arjun, 2013]. According to Food and Agriculture Organization, agriculture has proved to be backbone of Indian Economy because it adds around 23% to Gross Domestic Product (GDP) of the country. India is one of the largest producers of pulses, rice, wheat and spices. After completing various farming operations like land preparation, sowing, irrigation, fertilizer application, weeding, inter-cultural operations, pesticides application and harvesting, the main concern for the farmers is selling of their produce. Marketing of the agricultural produce has very crucial role in bringing profits to the farmer. Agricultural market or mandi is a place where the selling and buying of agricultural commodities take place. The farmers can sell their produce in these mandis but prefer to go to mandis in the nearby location. The commodities that are brought by the farmer to the mandis are divided into perishable and non-perishable types. Perishable commodities cannot stay fresh for long time such as vegetables and fruits. These commodities should be sold before they get spoiled. Non-perishable commodities are unlike perishable commodities and can be kept for longer period. To sell the commodities efficiently the farmer must know when and where to sell.

Mandi information is essential for smooth and active operation of marketing system (Nickels, 1978). Correct, adequate and timely availability of Mandi information helps to take decision about where to market the produce (Amrutha, 2009). Market information may be broadly defined as a communication or reception of knowledge or intelligence. It includes all the facts, estimates opinion and other information which affect the marketing of goods and services. It consists of all the certainties, opinions, estimates and other information which alter the marketing of goods and services. Mandi information is also beneficial for the economy as a whole. Government of India has provided many resources for spreading information about market to the farmers. Some

of the websites such as AGMARKNET (https://agmarknet.gov.in/), data.gov.in (https://data.gov.in), commodity online (https://www.commodityonline.com/) provide information about the available markets and commodity prices. The AGMARKNET website provides most of the market data that is required by the farmer. Arrivals and daily prices of the commodities in different markets all over the country is also provided and updated by this website. This important information needs to reach farmers in a convenient way for their benefit (Shakeel-Ul-Rehman *et al*., 2012). This convenient way can be an app on a mobile device that provides this information with a very user-friendly interface.

The mobile phone trade is one of the rapid growing businesses worldwide. It appears that the articulation that everybody has a cell phone isn't a long way from reality. Mobiles and telephones have hugely affected life of individuals in recent years. These handheld devices have gained popularity because of availability of inherent multiple functionalities which includes many useful apps in addition to calling facility. These devices provide flexibility to the user to access the installed utilities or apps anywhere anytime.

In recent years, there was huge technological development in terms of mobile and smart devices where human beings are shifting in huge numbers from the usage of conventional private computer systems i.e. desktop or laptop to smartphones and tablets. Less charge, clean portability, user-pleasant appearance has attracted users to these devices as they are very convenient to use in several ways throughout the day.

The smartphones not only provide effective way of communication over the network, but also provide a variety of convenient tools or app that are broadly used by the user in their day-to-day life. There are lots of mobile apps being advanced internationally imparting an awesome interface for activities that include online ticket reservation, online selling and buying of agricultural produce, mobile advisory services related to agriculture. There are apps also for different agricultural related activities such as ‘Kisan Suvidha’ for advising management practices for different pests and diseases. The emergence of the mobile apps has rewritten the definition of the mobile phone’s utilization. These mobile apps provide a convenient way for the users to access the

internet and get information related to any field. In agriculture sector also, mobile applications are becoming popular, with a huge potential for progressive agriculture. Many mobile applications are being developed for various aspects of crop production, farm management, weather forecast and other agricultural activities. Therefore, mobile apps can be a very useful tool for information and knowledge sharing as well as delivery of various useful services to the farmers for more improved and efficient agriculture.

## 1.2 Problem Definition

Farmers often face difficulties in finding the places or mandis for selling their agricultural produce. There is constrained access to the Mandi information because of farmers' proficiency level and presence of different channels of dissemination unnecessarily increases farmers' expenses (Rahman, 2003). There are a large number of vultures that destroy the advantages that the farmers should get. In spite of the fact that innovation have improved, it has not gone to the provincial dimensions as it is restricted to urban territories only.The farmers need to confront such huge numbers of hardships and need to defeat a few obstacles to get reasonable and good cost for their produce. Due to the lack of Mandi Information, farmers face the problem with the intermediates like middlemen, commission agents, brokers etc. Middlemen are people who buys the produce from the farmer at very low cost and sells them at high costs. As a result, farmers get very low price than what they are intended to get.

In this thesis, development of an android mobile app for providing the market information and price forecasts for various agricultural commodities have been described. This app provides the information about prices of commodities after fetching the current location of the farmer. It computes the forecasted prices of commodities based on the previous data and applying appropriate statistical methodology. This app is expected to help farmer in making decision about selection of mandi to sell their produce to get maximum returns.

## 1.3 Motivation

Many mobile apps have been developed for use in agriculture that help farmers for knowing the prices of commodities. An app that also provides the names of nearby

mandis with current prices would be very helpful for the farmers to get maximum returns. The forecasted prices of the commodities based on the previous data would also help farmers for taking their best decision to sell their produce.

## 1.4 Objectives

- To design and develop a mobile app for locating nearby mandis
- To develop an interface for current and forecast prices of selected commodities

## 1.5 Plan of Thesis Work

This thesis describes the development of mobile app that provides information about nearby markets for the selected commodity and also the price forecast of some selected commodities. The whole dissertation consists of five chapters as described below:

In **Chapter 1,** a brief introduction has been given which includes the description about the topic of this thesis and the detailed background of the study undertaken. The objectives of the study have also been provided.

**Chapter 2,** provides the description about the related work undertaken in the past. Relevant and related reviews on the development of different mobile apps in agriculture have been provided.

In **Chapter 3,** methodology of the study has been provided. It includes description about the app development and use of API for retrieval of prices and nearby mandis.

**Chapter 4,** describes the features and details about the developed mobile app. It also explains its functionality and generated results.

In **Chapter 5,** summary of the work undertaken has been provided.

# CHAPTER II

# REVIEW OF LITERATURE

## 2.1 General Perspective

Now-a-days mobile device has become an important part of human life. Almost every individual is having a smart phone because it helps in the day to day activities of every individual. It has become the most convenient and easy mean of communication because of advancement in the mobile technology. Earlier, people were comfortable with the web-based systems after Internet facility was made available. But, nowadays, with the advent of mobile phones, development of a number of apps for performing various actions on it, people are using mobile devices for getting most of the activities. Even rural part of India has also started using mobile devices for quite some time for carrying out their various activities including financial transactions.

## 2.2 Previous work done

A number of studies have been conducted for development of many useful mobile applications for use in agricultural and rural sector. These developments have been very useful for progress in the farm activities and rural livelihood. Some of them have been listed and described below:

**KSUSoy YieldCalc:** This android app has been developed to estimate soybean yield before harvesting using conventional approach (Bandyopadhyay, 2015). In this investigation a versatile application "KSUSoy YieldCalc" was built that provides yield estimation of soybean crop before reaping. It can be installed on any mobile platform including iOS across 190 countries for the users of different economic status. It utilizes the customary methodology for estimation of yield. "KSUSoy YieldCalc" has been developed for Android as base to serve farmers, agronomists and experts. They also deliver good performance to save time and enhance confidence of the farmers. The application was created utilizing Android Software Development Kit (SDK) improvement stage with Java and Extensible Markup Language (XML) coding. The

Department of Agronomy at Kansas State University (KSU), Kansas, has tried the application with promising outcomes.

**Krishi Ville:** This is also an android based solution for Indian agriculture (Singhal *et al.,* 2011). This is a mobile based application developed for the farmers to provide guidance in their farming activities. It has been designed taking Indian farming in consideration. This application provides the following:

- Update of the different agricultural commodities,
- Weather forecast update, and
- Agricultural news update

**Kisan Suvidha:** It is a versatile application created to help farmers by giving significant data very fast. With snap of a catch, they can get the data on climate, advertise costs, agro warnings, plant security, Integrated Pest Management (IPM) practices and so on. Some of these highlights are extraordinary climate cautions, market cost of products and the maximum prices in all the states of India. Kisan Suvidha is available at: http://www.kisansuvidha.com/.

**Kisan Network:** It is very good app for Indian farmers (Kisan) which displays videos and content on organic farming, machinery for agriculture, weather information and farming news. It also provides information about business, government subsidies, minimum support price (MSP) and daily mandi rates including update of maximum and minimum prices of crops available in mandis of India. It is available at https://kisannetwork.com/.

**Agro Connect:** It is a digital market that links farmers and buyers. Members of this also include sellers, distributors, manufacturers, academicians and scientists, consultants and many more. It is available at http://agroconnect.com.ng/. Using this app following activities can be performed.

- ✓ Get data on the best way to shield crops from bug assaults. Get data on pesticide dose and time of use.

- ✓ Discuss bug related issues, crop development issues with different farmers and specialists of the field. Individuals experience may also be added for benefit of other farmers.
- ✓ Attach photographs to make the inquiry clearer.
- ✓ Check most recent Market Prices of any product in this agribusiness application.
- ✓ Farmers can check Daily Weather - For Kisan Suvidha alongside day by day climate.

**Digital Mandi India:** This App helps in checking the most recent Indian farming items and Mandi costs from various states. It is simple to utilize and the application empowers farmers, dealers and all others to know the refreshed Mandi cost of mandis of India. It's primary highlights are Sync information from the Indian government entryway agmarknet. It is available at: https://digital-mandi-india.en.aptoide.com/.

**Agricultural Marketing Information network (AGMARKNET):** It is website that helps various stakeholders such as farmers, industry, policy makers and academic institutions by providing information about markets. This website helps those farmers who do not have sufficient resources to get adequate market information. It facilitates web-based information flow of the daily arrivals and prices of commodities in the agricultural produce markets spread across the country. It is available at: http://agmarknet.gov.in/.

Price foresting is the prediction of the prices that may prevail in the future. This can be useful for farmers to find best suitable mandi to get good returns. Some of the studies related to price forecasting made by the researchers have been mentioned below.

Box and Pierce (1970) were the pioneer of a new area in time-series forecasting and modeling. They developed ARIMA and its component models that are important and widely used techniques in time-series analysis. The peculiarities of ARIMA model are its well adaptability, flexibility and wide applicability.

Box *et al.* (2007) provided a detail description of ARIMA methodology and its component model along with several applications on real data sets.

Farhath *et al*. (2016) gave an excellent review on ARIMA model. They show that the ARIMA is a superior to exponential smoothing techniques when the data is reasonably long and the correlation between past observations is stable.

# CHAPTER III

# MATERIAL AND METHODS

A mobile app named "MandiInfo" has been developed for finding the nearby mandis and provides price forecasts of the commodities based on the previous years' data. In this chapter, the details about the developed mobile application, its techniques, utilized virtual devices and systems, settings and configurations as well as backend database have been discussed. Various kinds of software and tools have been used for carrying out the development work. They are Android Studio Integrated Development Environment (IDE), java programming language and libraries, SQLite database and related Android Virtual Device (AVD) tools.

## 3.1 Platform used for Mobile App

Currently, mobile devices from different manufacturing companies use different operating systems, for example, Apple uses iOS, Samsung-Bada, Nokia-Symbian operating systems and so on. However, android is most popularly used all over the world. The development of a mobile app for android platform uses open source-based software tools with many important features and functionalities.

Android operating system (OS) is Linux Kernel based and has java programming interface with several essential components as shown in Fig 3.1. The main requirements for the development of mobile apps are: Android Software Development Kit (SDK), a complete stack of OS, middleware and applications that provide all the required tools.

Knowledge of java programming and architecture of android is required to develop an android mobile app. Programs written in java for android app is compiled to bytecode format within the built environment of Android Studio. When this app is installed on any android device, the bytecode is compiled by Android Runtime (ART) to the format used by the central processing unit (CPU). Brahler (2010) summarizes efficiency and performance as main goals of the android architecture, both in the execution of the application and in the implementation of the old application. Android software stack structure comprises of applications, middleware, an OS, services, libraries and run-time environment. Each layer of the stack and their corresponding elements are strongly

integrated and clearly tuned to contribute optimum application build up and execution environment for the device.

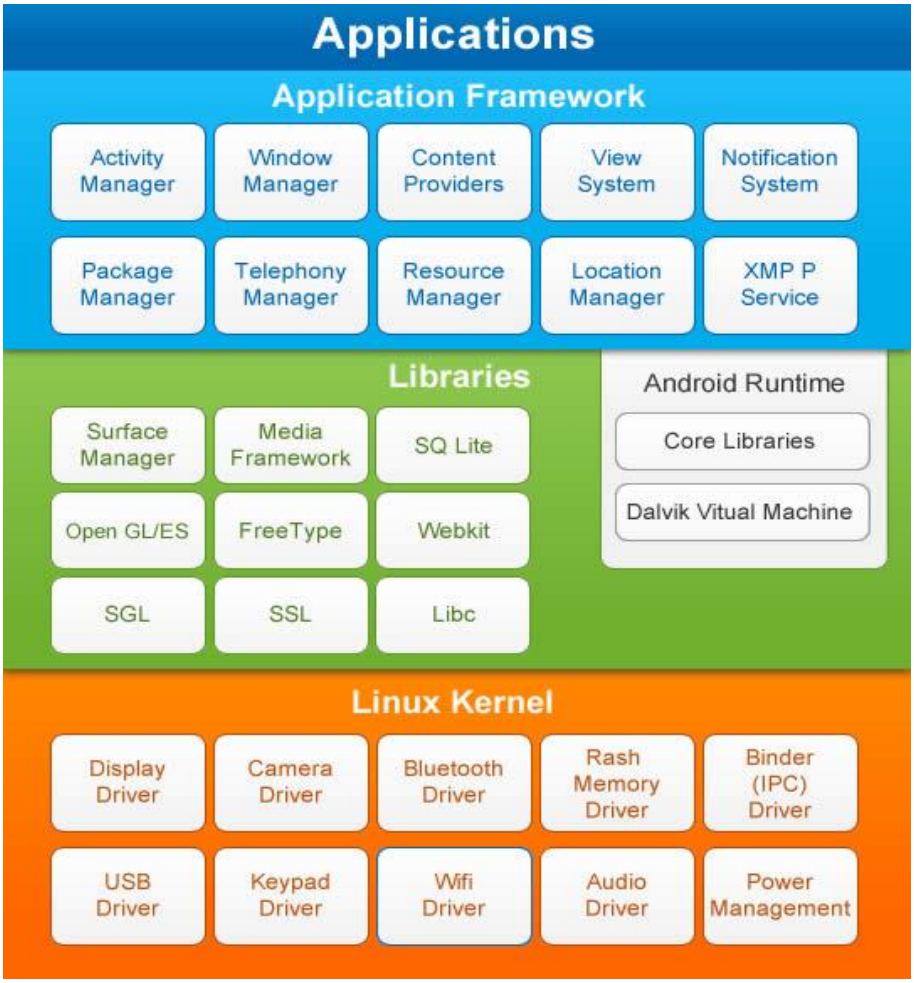



Fig 3.1: Android stack

**(Source:** https://www.linuxinsider.com/story/61961.html?welcome=1210907223)

## 3.2 Architecture of MandiInfo Mobile App

The architecture of the developed app has mainly two layers - Client-Side Interface Layer (CSIL) and Database Layer. The details about these layers have been described in the following section:

**Client-Side Interface Layer (CSIL):** The mobile app itself is the client-side interface layer that runs on platform like android. XML, Styles of android software development kit (SDK) and Java programming language implement the interface. The design or layout creation of the mobile app is carried out using XML files. The view format of the

mobile app is specified by the Style files of android. The logic and the actual activities of the application are implemented by Java programming language.

**Database Layer:** It is the layer where the data from web resources is retrieved, stored and displayed to the user. If the source data is not updated or available, this app uses the previously stored data from the database to display on the app.

The architecture of the developed app has been shown in Fig 3.2 below.

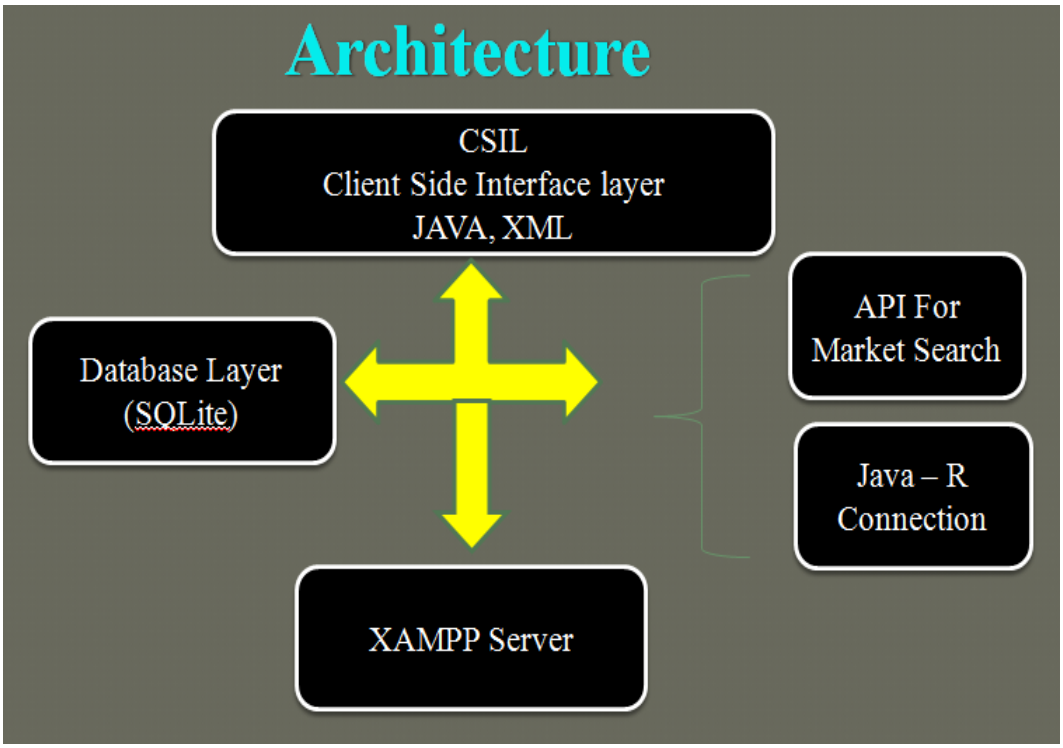



Fig 3.2: Android app architecture

**XAMPP Server:** This server is used for deployment of web environments used in the developed mobile app. XAMPP is freely available server.

## 3.3 App workflow

The app workflow has been described in Fig 3.3. It gives the complete description of the flow of information in the developed mobile app. It starts from the dashboard and moves to different features provided by the app. First step in the workflow is finding the nearby markets that gives the option to select the commodity list. After selecting commodity, it presents the data such as nearby markets with the selected commodity along with

variety, grade, maximum, minimum and modal prices. Next is the selection of state and district which gives list of all markets in the selected district along with variety, maximum, minimum and modal price.

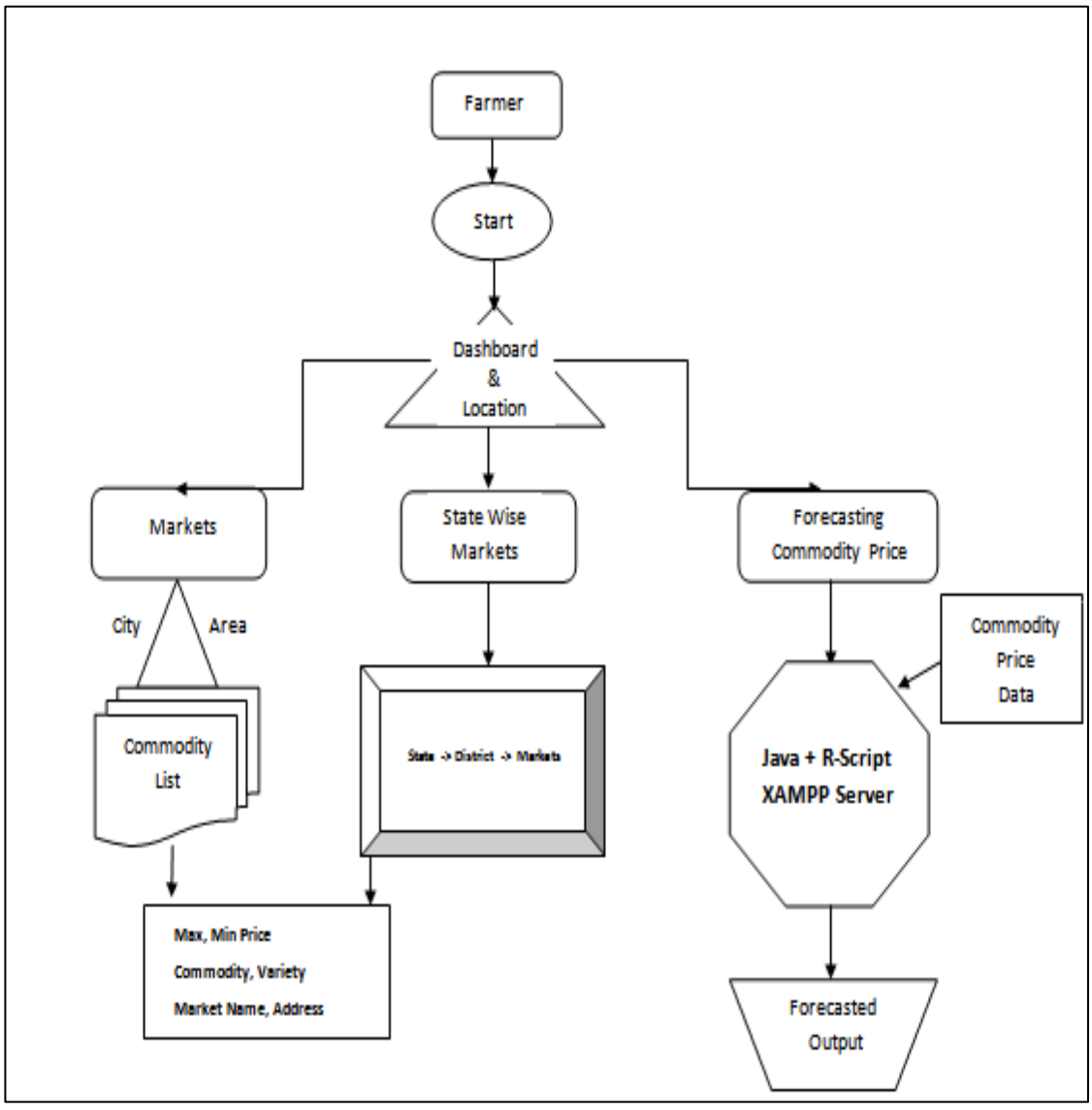



Fig 3.3: Flow Chart of app activities

## 3.4 Methodology for App Development

MandiInfo has been developed for the android environment. This app also connects with the database to provide results at various stages. It also connects to the server to deploy the web environment variables. R script has also been written for adding the functionality of price forecasting to the app. This script uses inbuilt statistical library of R. Integration of R script with java program has also been incorporated for development of this app.

**3.4.1 Reading Location using "Google Places" API:** One or more components of the android app are written as java classes. A Google Application Programming Interface (API) has been used that automatically detects the user's current location details. Google provides an API with an API key for this purpose. The API from Google is shown in Fig 3.4.

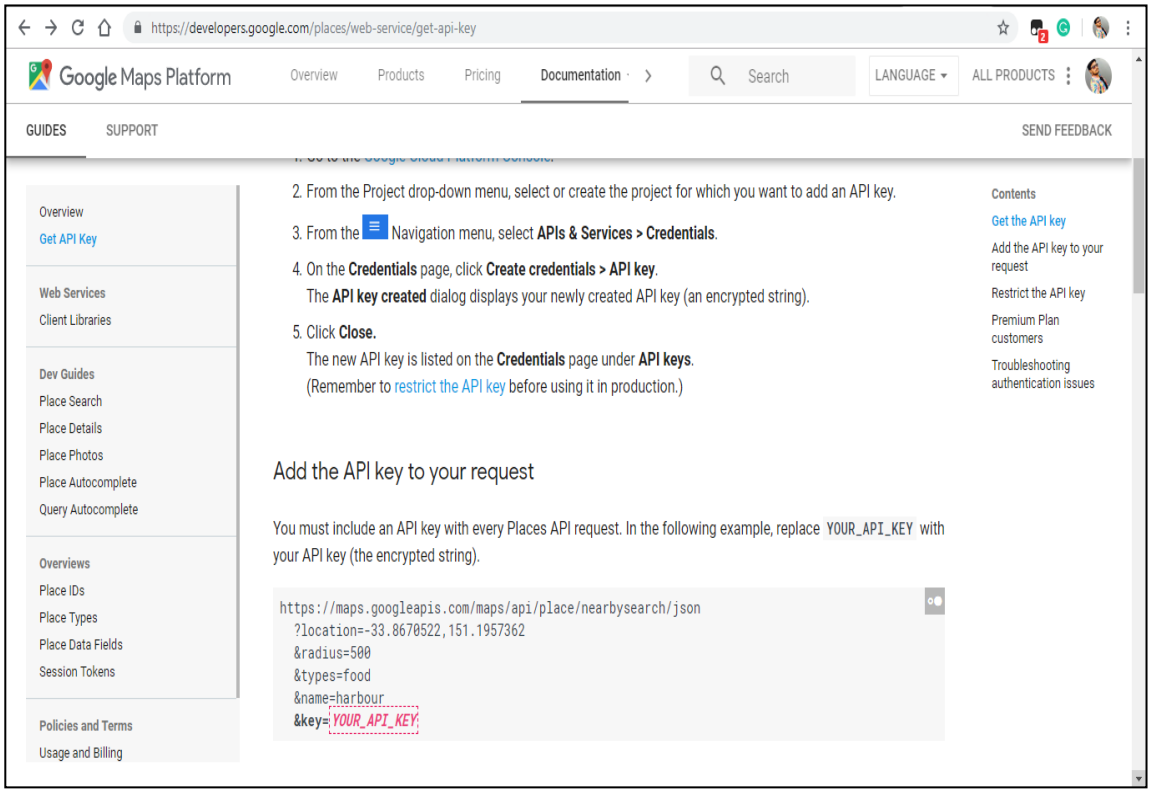



Fig 3.4: Google places API

(Source: https://developers.google.com/places/web-service/intro)

**3.4.2 Mandi and Price details for Commodities:** The web portal called AGMARKNET contains data about different agricultural commodities and their markets available in India. The API of this portal is provided by Government of India (GOI) website https://data.gov.in. A program has been written in Android studio for parsing the data by using an API available on this website. A screenshot of the webpage of this website where API can be downloaded is shown in Fig 3.5. AGMARKNET also provides script which uses http://data.gov.in for retrieval of statewise district markets of India. This script has been used in MandiInfo app for providing details about latest market prices available for various commodities. MandiInfo always displays the

retrieved data of current date. It was noticed that sometimes the current data is unavailable on this website. In this case this API becomes non-functional. Therefore, a database has been developed to save the data retrieved. This data is used to display on the app whenever the data of current date is not uploaded by them. Fig 3.6 shows the type of data stored in database.


Fig 3.5: AGMARKNET API from data.gov.in

(Source: https://data.gov.in/resources/current-daily-price-various-commodities-various-markets-mandi/api)

**3.4.3 Price Forecasting for Selected Commodity:** Price forecasting or prediction has been a challenging area in research. Generally, price forecasting is done by the use of time series approach. Lot of efforts have been made for the development and improvement of many time series forecasting models. Box Jenkins ARIMA methodology (Box and Pierce, 1970 and Box *et al.*, 2007) is one of the prominent linear

time series models which is broadly used for univariate time series data analysis. ARIMA model is very popular these days because of its statistical properties.

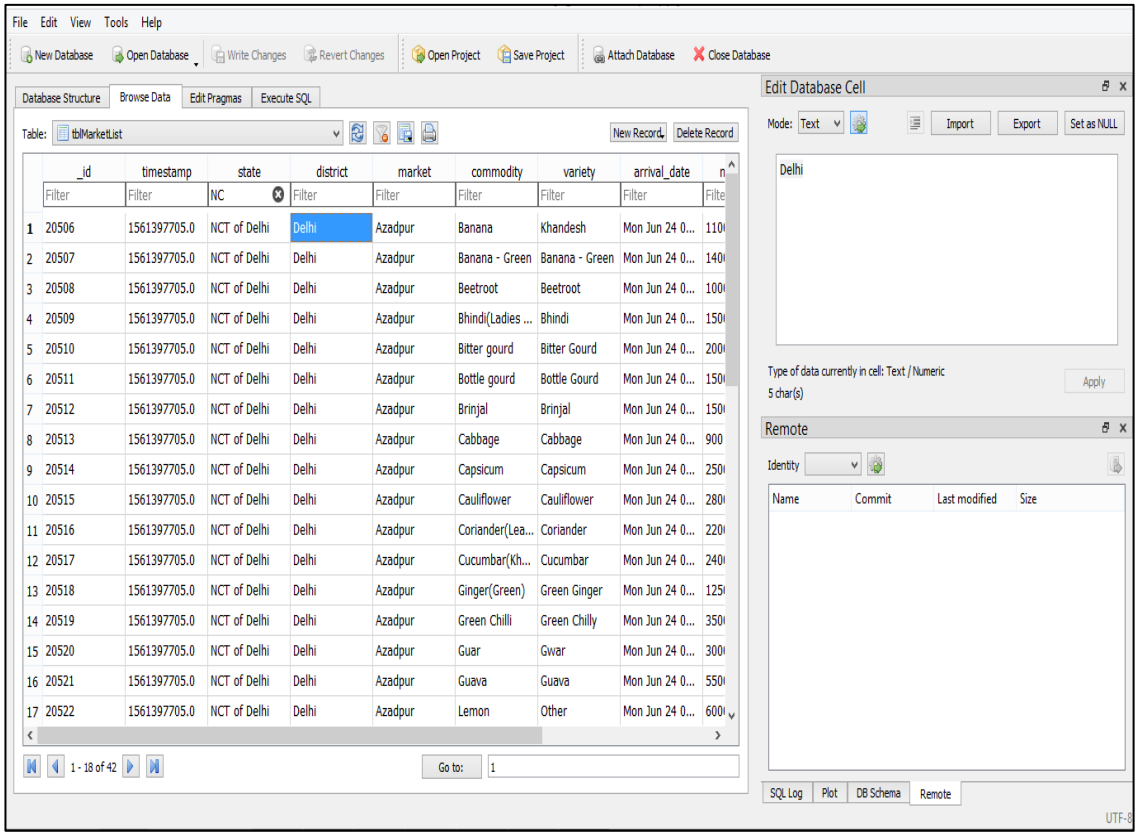



Fig 3.6: DB browser with stored data

ARIMA is a pliable class of models having moving average (MA), Autoregressive models (AR) models and a merging of both MA and AR models.

The future value of a variable is sequentially related to the past values of the variable itself and random error in ARIMA model. The model expresses a time series process and is linear univariate time series model say, $\{y_t\}, t = 1,2, \dots, n$ in three sets of parameters as

$$\varphi(L)\Delta^d y_t = \theta(L)e_t$$

where $y_t$ is real observation and $e_t$ is error term observed at time $t$; $\varphi(L)$ and $\theta(L)$ is polynomial of lag operator $L$ of order p and q respectively in root outside the unit circle; $e_t$ supposed to be independent and similarly distributed with mean zero and variance $\sigma^2$. A schematic representation of ARIMA methodology by Box-Jenkins has been shown in Fig 3.7.

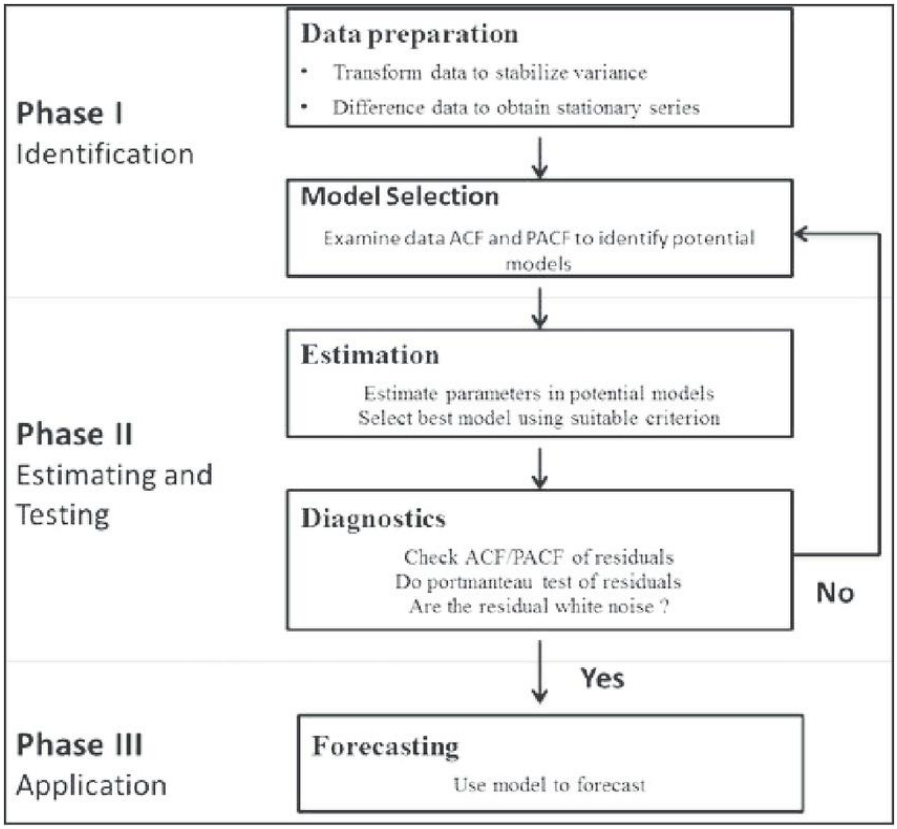



Fig 3.7: Schematic representation of ARIMA methodology by Box-Jenkins

Price forecasting facility has been added to the app by writing an R script to carry out the statistical computation. This script is executed from java program on Android studio. Fig 3.8 shows R package window where forecast package is being installed for price forecasting. Result of the R script is presented on web browser within the developed app.

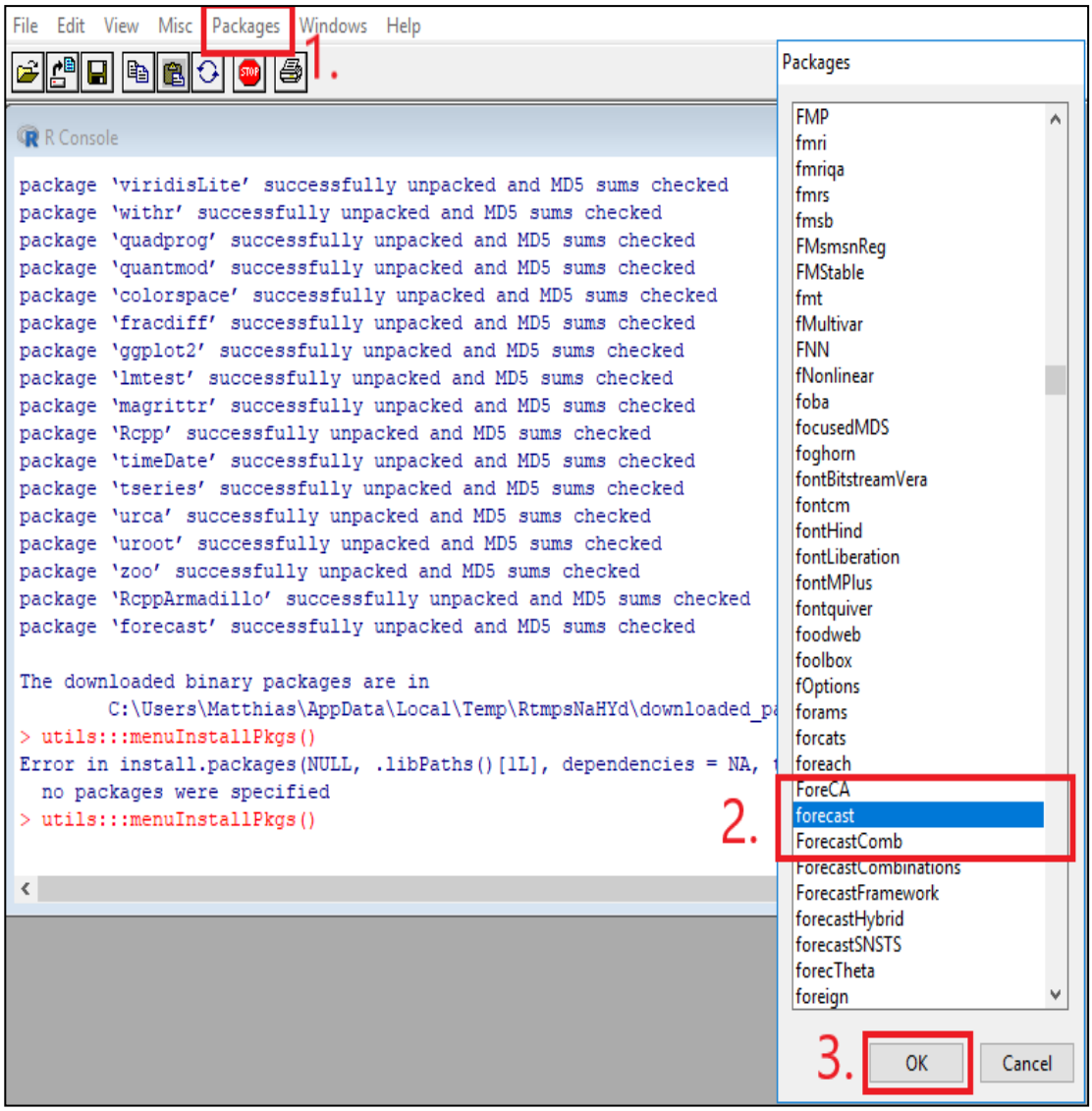



Fig 3.8: Forecast package installation in R

(Source: https://cran.r-project.org/bin/windows/base/)

XAMPP server has been used as web server for deployment of web app. Fig 3.9 shows the control panel of XAMPP server.

## 3.5 Activity Life Cycle of Android

Android application has a distinct and easy workflow as shown in Fig 3.10. When the Activity in java file is extended, *android.app.activity* class is imported. The *main()* method is not present in android application. The different methods used in activity life cycle are

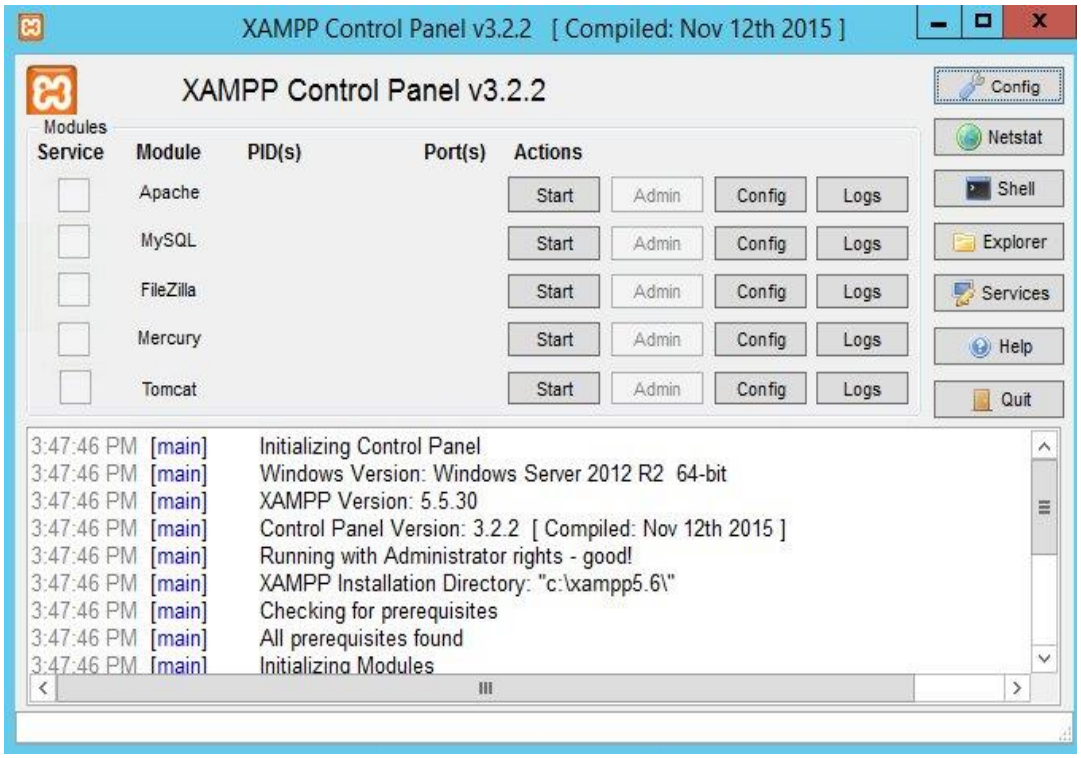



Fig 3.9: The XAMPP server control panel

(Source: https://www.apachefriends.org/index.html)

- ***onCreate():*** When an activity is created, *onCreate()* method is called. This is the first method being called.
- ***onStart():*** When an activity becomes visible to the user *onStart()* method is called. *onStart()* is immediately called by *onCreate()*.
- ***onResume():*** After *onStart()* method, *onResume()* method is called. This is called when activity interacts with the users continuously.
- ***onPause():*** Whenever an activity pauses, *onPause()* method is called.
- ***onStop():*** Whenever an activity loses connection with the user, *onStop()* method is called.
- ***onRestart():*** Whenever an activity is stopped and priority is to restart it, *onRestart()* is called.
- ***onDestroy():*** Whenever an activity is destroyed, *onDestroy()* method is called.

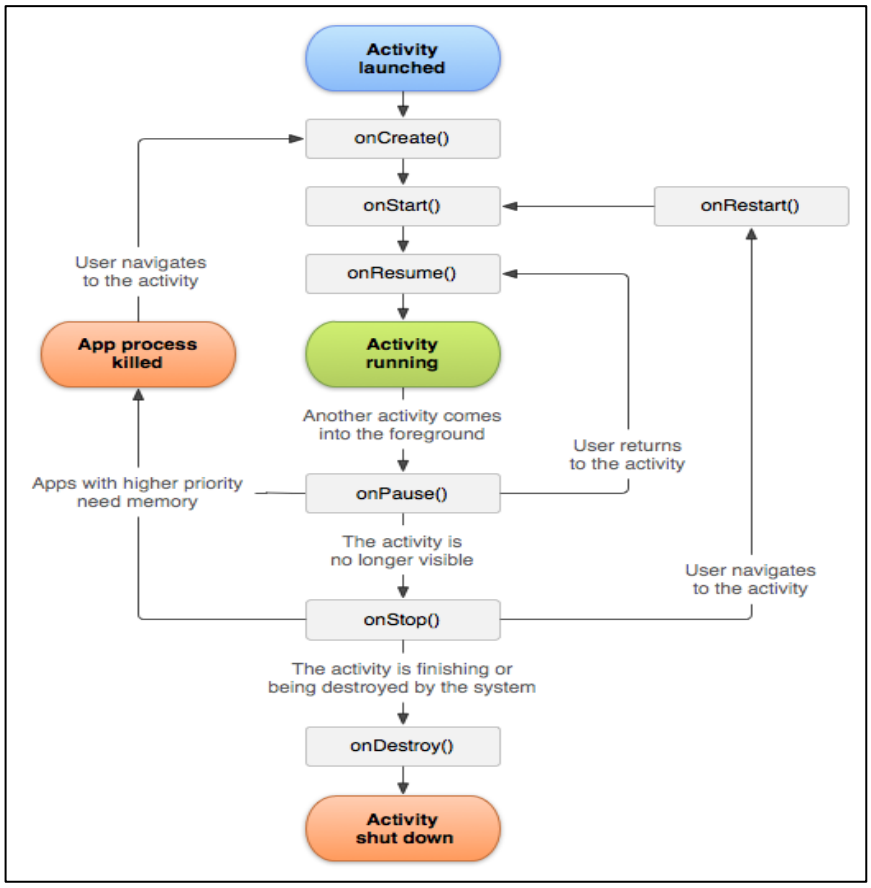



Fig 3.10: Android activity life cycle
(Source: https://developer.android.com/guide/components/activities/activity-lifecycle)

## 3.6 Programming Environment

### 3.6.1 eXtensible Markup Language (XML)

XML has been created and developed by World Wide Web Consortium (W3C) XML Working Group. This group of the W3C in the year of 1988 developed and released it. XML is a computer language which uses markup and has extended its capability. Between different systems, it helps to share data in a structured manner. It can be defined as a Meta markup language, which means, it provides information about the information. The core of all web development is XML. It is used in every type of website whether it is a news site, a social media site or any other website.

XML is the basic version of the Standard Generalized Markup Language (SGML) and easily understandable, useable and readable. As it is a platform independent component, it is supported by a large no. of platforms and provides the flexibility to use

unstructured, semi structured, and structured data for transfer of information between different types of application. Based upon the need of the user, it allows defining their own tags. Usually, XML documents create an expandable hierarchical tree structure that consists of several attribute lists within it. The XML has the following components as described below.

- **Declaration:** The declaration consists of details that arrange an XML processor to parse the XML document. It has three syntax - version, encoding and standalone.
- **Tags:** The code written inside the angle brackets is known as tags similar to HTML. Tags and data are stored together in XML.
- **Elements:** The fundamental building blocks of the XML document are called Elements. One root element must be present in XML document. Element is the text between the start and end tags.
- **Attributes:** Extra information about the elements is given by attributes. Names of attributes must be unique. To assign the values they need to be declared with single or double quotes.
- **Comments:** Extra knowledge about the document is given by Comments. It can be given anywhere inside the document. Compiler does not parse the comments.

### 3.6.2 JAVA Programming Language

JAVA is a programming language which is broadly spread and most commonly used for app development. Team of Sun Microsystem Company under the supervision of James Gosling in the year of 1991 developed Java. First it was named as ‘Green Talk’, ’Oak’ and changed to ‘JAVA' in 1995. It is a high-level programming language that is broadly used for the desktop as well as web applications development. Java is platform free programming language. It is easy, secure, movable interface to the users. It is based on the principles of Object-Oriented Programming (OOP). There are four main characteristics of OOP followed by java i.e. Encapsulation, Polymorphism, Abstraction and Inheritance. Binding up of data and methods together is known as encapsulation. Unauthorized access of the information is restricted from the outside world by Encapsulation. By defining the data and the methods within the packages and classes, and by assigning many types of access specifiers to them Encapsulation can be achieved. In java, polymorphism can be achieved by the method overloading and overriding with

the using same method for various functions. The Inheritance principle of the OOP creates a parent child relationship between different classes and permits to accumulate properties of one class to the other.

Java permits to write programs for the android application. Many packages, interfaces, abstract classes are available in android that are written in JAVA programming language. Each activity in the android app is a java class that is sub class of the Activity class of the application.

## 3.7 Application Development Environments

### 3.7.1 Android Studio IDE

Android studio is very popular for development of a mobile application. Android studio is the Google's official integrated development environment (IDE). The Google's conference held on May 16, 2013 released this IDE officially. The different classes used in android studio are described in table 3.1.

Table 3.1: Description of imported classes provided by android

| **Class** | **Description** |
|---|---|
| android.app.Activity | An activity is a single, focused thing that the user can do. |
| android.content.Intent | An intent is an abstract description of an operation to be performed. |
| android.graphics.BitmapFactory | Creates Bitmap objects from various sources, including files, streams, and byte-arrays. |
| android.os.Bundle | A mapping from String keys to various Parcelable values. |
| android.util.Log | API for sending log output. |
| android.view.Menu | Interface for managing the items in a menu. |
| android.view.MenuInflater | This class is used to instantiate menu XML files into Menu objects. |
| android.view.MenuItem | Interface for direct access to a previously created menu item. |
| android.view.View | This class represents the basic building block for user interface components. |
| android.view.View.OnClickListener | Interface definition for a callback to be invoked when a view is clicked. |
| android.widget.AdapterView | An AdapterView is a view whose children are determined by an Adapter. |

| | |
|---|---|
| android.widget.ArrayAdapter | This adapter can be used to provide views for an AdapterView, Returns a view for each object in a collection of data objects provided and can be used with list-based user interface widgets such as ListView or Spinner. |
| android.widget.Button | A user interface element, the user can tap or click to perform an action. |
| android.widget.Spinner | A view that displays one child at a time and lets the user pick among them. |
| android.widget.Toast | A toast is a view containing a quick little message for the user. The toast class helps you create and show those. |
| android.widget.CheckBox | A checkbox is a specific type of two-state button that can be either checked or unchecked. |
| android.widget.ListView | Displays a vertically-scrollable collection of views, where each view is positioned immediatelybelow the previous view in the list. |
| android.content.ContentValues | This class is used to store a set of values that the ContentResolver can process. |
| android.content.Context | Interface to global information about an application environment. |
| android.database.Cursor | This interface provides random read-write access to the result set returned by a database query |

Android Studio is freely offered under the Apache License 2.0. Android studio IDE is developed by combining the robust code editor with a completely User Interface (UI) build system dependent on the gradle. Android studio version 3.2 (released in April, 2018) with JDK 8, has been used for this app development. An android studio consists of the code editor, tool windows and the layout editor tool (Smyth, 2017).

### 3.7.2 AVD and API level

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet that is used for simulation in the emulator. AVD in combination with android emulator provides an environment of a virtual device in which Android app is installed. It is useful for the purpose of right functionality and testing of the app in the android studio. It is a software that mimics the real device, sharing the hardware resources of the hosting device. Usually, its pattern is used with its corresponding APIs.

API Level is the integer value which uniquely identifies the framework. API revision is described by means of the Android platform. The Android platform gives a framework API for the applications to use and interact with the underlying Android device.

Different versions of API levels are released by android. Table 3.11 gives the details of android code names, versions and also API levels.

Table 3.2: Android versions with API levels

| Code name | Version | API level |
|---|---|---|
| Pie | 9.0 | API level 28 |
| Oreo | 8.1 | API level 27 |
| Oreo | 8.0 | API level 26 |
| Nougat | 7.1 | API level 25 |
| Nougat | 7.0 | API level 24 |
| Marshmallow | 6.0 | API level 23 |
| Lollipop | 5.1 | API level 22 |
| Lollipop | 5.0 | API level 21 |
| KitKat | 4.4 - 4.4.4 | API level 19 |
| Jelly Bean | 4.3.x | API level 18 |
| Jelly Bean | 4.2.x | API level 17 |
| Jelly Bean | 4.1.x | API level 16 |
| Ice Cream Sandwich | 4.0.3 - 4.0.4 | API level 15, NDK 8 |
| Ice Cream Sandwich | 4.0.1 - 4.0.2 | API level 14, NDK 7 |
| Honeycomb | 3.2.x | API level 13 |
| Honeycomb | 3.1 | API level 12, NDK 6 |
| Honeycomb | 3.0 | API level 11 |
| Gingerbread | 2.3.3 - 2.3.7 | API level 10 |
| Gingerbread | 2.3 - 2.3.2 | API level 9, NDK 5 |
| Froyo | 2.2.x | API level 8, NDK 4 |
| Eclair | 2.1 | API level 7, NDK 3 |
| Eclair | 2.0.1 | API level 6 |
| Eclair | 2.0 | API level 5 |
| Donut | 1.6 | API level 4, NDK 2 |
| Cupcake | 1.5 | API level 3, NDK 1 |
| (no code name) | 1.1 | API level 2 |
| (no code name) | 1.0 | API level 1 |

(Source: https://source.android.com/setup/start/build-numbers)

## 3.8. Database Design

### 3.8.1 SQLite

In any Android device, there are numerous ways to store information. SQLite is one of the available databases for storing information. It is a straightforward database that

accompanies Android OS. In Android, coordinating SQL is a repetitive assignment as it needs composing parcel of code to store straightforward information. SQLite is an open source database that stores information in content record on a gadget. Android comes with inbuilt SQLite database. All the social database highlights are upheld by SQLite.

SQLite is installed in Android Studio. SQLite is an in-process library which executes an independent, zero-configuration, server-less, value-based SQL database motor. Zero-configuration means we don't need to configure it in our framework like other databases. It is server-less because it doesn't require a different framework or server to work. A total SQLite database can be put in only a solitary cross-stage circle document. It is lightweight and exceptionally little in size. Its completely configured record is only 800 KB and with overlooked highlights, it is only 450 KB. It is independent, which means there is no need of outside conditions (Vogel, 2010). Fig 3.11 shows classes identified with SQLite.

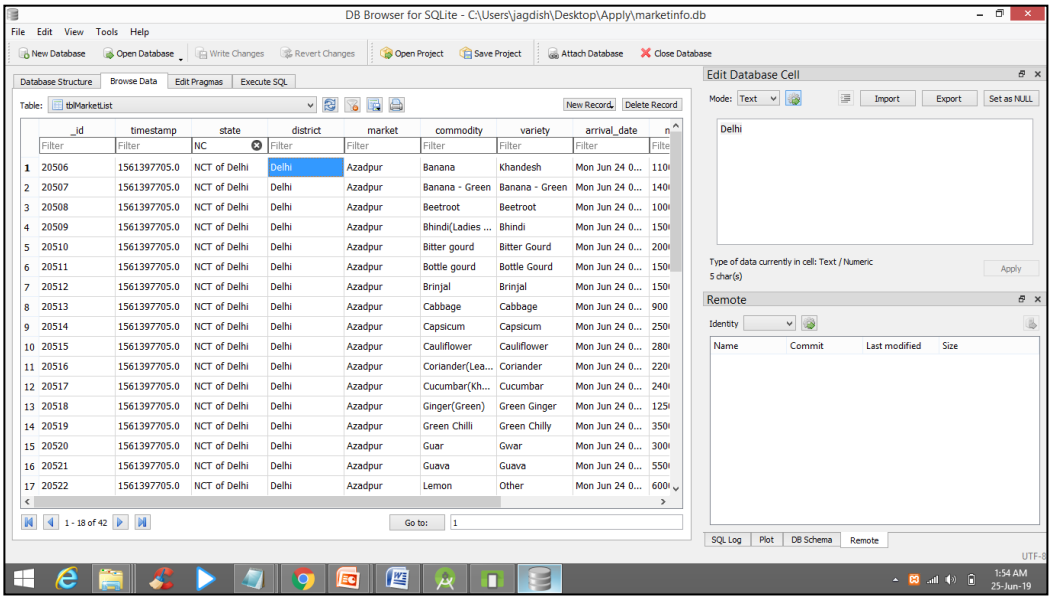


Fig 3.11: Description of SQLite database

# CHAPTER IV

# RESULTS AND DISCUSSION

An android based mobile app named as 'MandiInfo' has been developed using the methodologies and technologies discussed in previous chapter. The app is developed for finding the nearby mandis and information about market prices in different states throughout the country. It also forecasts the prices of different commodities in different markets based on the previous years' data. After carrying out the requirement analysis and following the app development programming discussed in earlier chapters this mobile App has been developed.

MandiInfo is an android based application which allows users to access the nearby markets with commodity selection.

## 4.1 Features of the Application

Broadly, the features of 'MandiInfo' app has been mentioned below

- Detection of user's location
- Information retrieval interface for markets and commodity prices
- Commodity selection from the available list
- Selection of state and district
- Price forecasting for selected commodities
- Viewing results: Output can be viewed in pdf format and also can be shared through email, whatsapp, google drive, bluetooth etc.

## 4.2 App Icon and Splash Screen Activity

Fig 4.1 shows the icon of 'MandiInfo' app and its splash screen. The splash screen shows the title of the app and a picture related to agricultural transport activity.

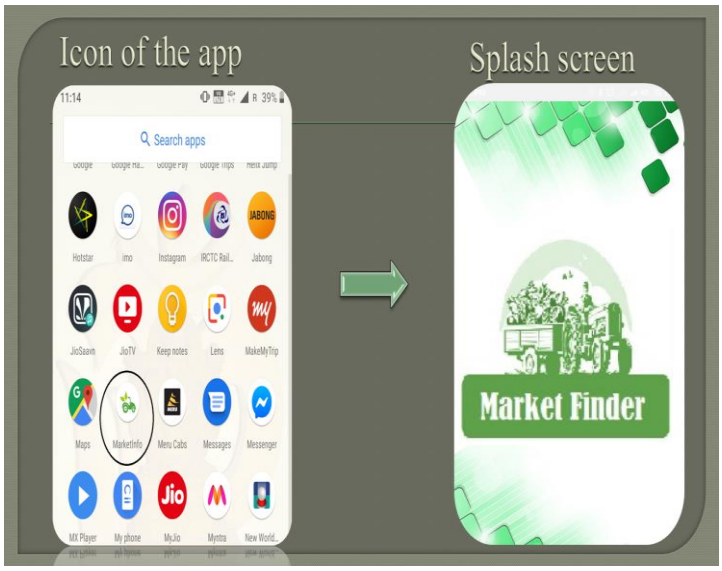



Fig 4.1: ‘MandiInfo’ app icon and its splash screen

## 4.3 Home screen of the mobile application

Home screen of the ‘MandiInfo’ app has been shown in Fig 4.2. It displays the location details and three other options for selection.

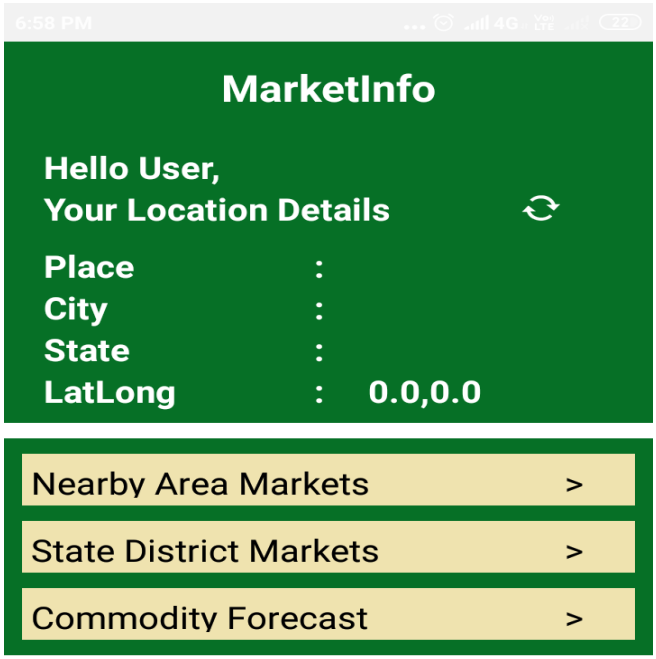



Fig 4.2: Home page of the ‘Mandi Info’ app

The three other options provided by this app are - *Nearby Area markets, State and district markets and Price forecasting.*

## 4.4 Detection of user's location

When the user clicks on the refresh button provided on the home page the app automatically detects the user's location by executing google places API in the background. The Fig 4.3 shows the screen capture of the home page giving the names of location, city, state, latitude and longitude.

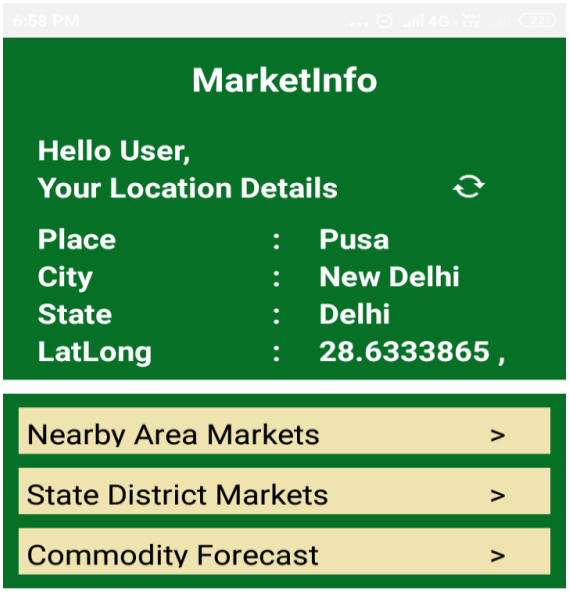



Fig 4.3: Location details

## 4.5 Finding nearby markets

When user taps on the nearby markets option, a dropdown list of commodities is displayed. After the user selects the desired commodity from the available options, it shows all the available markets nearby for the selected commodity with its maximum, minimum and modal prices for the current day.

The data about market prices of mandis is available on AGMARKNET portal. The data from this portal can be retrieved using the API provided by data.gov.in. In case the current data is not uploaded on AGMARKNET website, the app is unable to retrieve the current price. In this situation, MandiInfo app takes the mandi price of previous day which is stored in its inbuilt database. The database has been developed on SQLite database software. Fig 4.3 shows the dropdown list of commodities when user selects the option of 'nearby markets'.

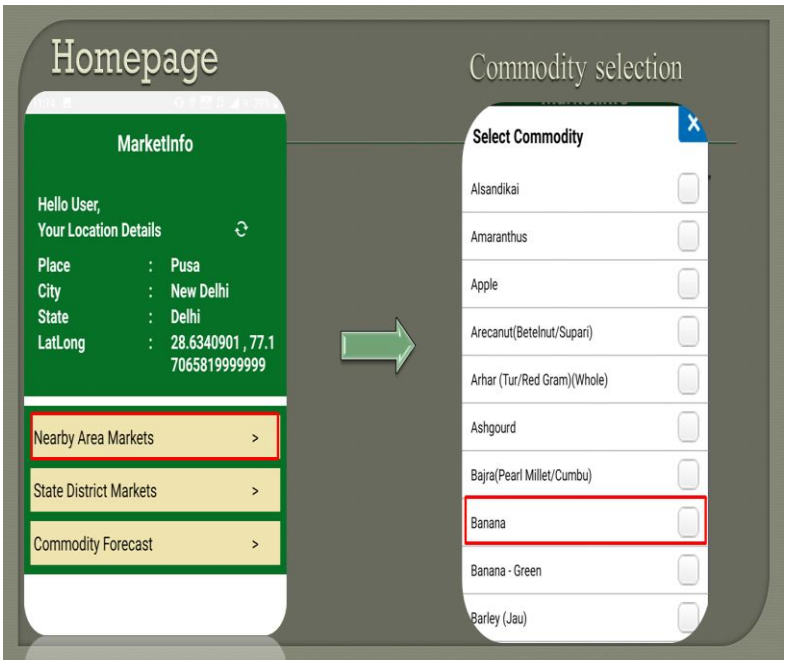



Fig 4.4: 'Mandi Info' showing the selection of commodity

***Selection of commodity:*** When the user selects a required commodity, the app displays markets for that commodity, its variety, minimum, maximum and modal prices. If the user is interested in all commodities then app displays all the markets with different commodities. The Fig 4.4 represents selection of specific and general commodities.


Fig 4.5: Selection of specific commodity (Left) and all commodities (Right)

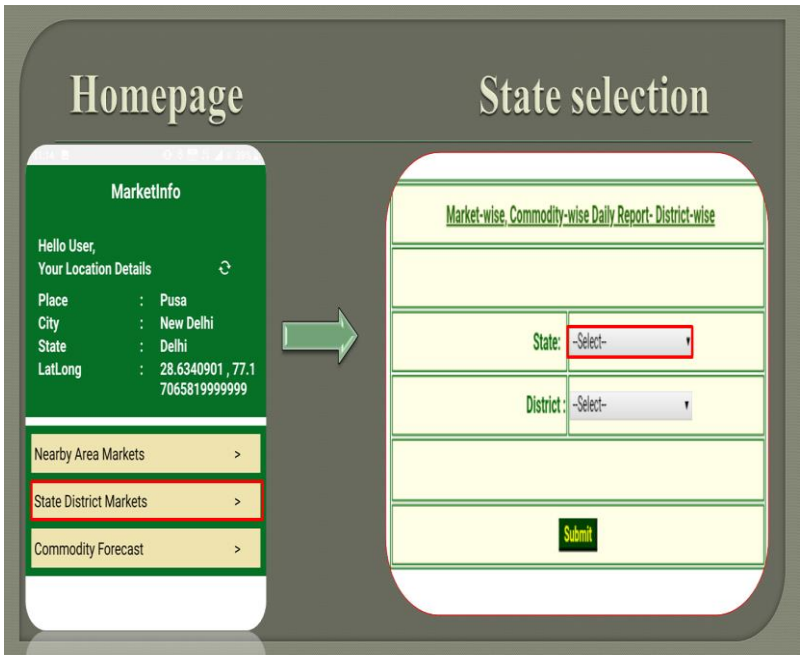



Fig 4.6: Page retrieved by webURL in 'MandiInfo'


Fig 4.7: Selection of state and district markets in 'MandiInfo'

The output is displayed in a tabular format which gives the information about markets, commodities, arrivals, unit of arrivals, variety, grade, minimum, maximum and modal prices of the commodities. The sample data about markets in the district is shown in Fig 4.8.

Latest Market Prices Available Hyderabad, Telangana (26/06/2019)

NR : Not Reported

| Market | Commodity | Arrivals | Unit of Arrivals | Variety | Grade | Minimum Price | Maximum Price | Modal Price | Unit of Price |
|---|---|---|---|---|---|---|---|---|---|
| Bowenpally | Beetroot | 9.2 | Tonnes | Beetroot | FAQ | 1000 | 2600 | 2200 | Rs/Quintal |
| | Bhindi(Ladies Finger) | 28.6 | Tonnes | Bhindi | FAQ | 1000 | 3500 | 2800 | Rs/Quintal |
| | Bitter gourd | 21.3 | Tonnes | Bitter Gourd | FAQ | 1500 | 4000 | 3200 | Rs/Quintal |
| | Bottle gourd | 54.2 | Tonnes | Bottle Gourd | FAQ | 600 | 2000 | 1800 | Rs/Quintal |
| | Brinjal | 49.6 | Tonnes | Arkasheela Mattigulla | FAQ | 1000 | 3000 | 2500 | Rs/Quintal |
| | Cabbage | 97.1 | Tonnes | Cabbage | FAQ | 800 | 2200 | 1800 | Rs/Quintal |
| | Capsicum | 72.5 | Tonnes | Capsicum | FAQ | 1200 | 3600 | 3000 | Rs/Quintal |
| | Carrot | 153.7 | Tonnes | Carrot | FAQ | 600 | 5200 | 3500 | Rs/Quintal |
| | Cauliflower | 15.6 | Tonnes | African Sarson | FAQ | 1500 | 6500 | 4500 | Rs/Quintal |
| | Cluster beans | 42 | Tonnes | Cluster Beans | FAQ | 1500 | 4000 | 3200 | Rs/Quintal |
| | Colacasia | 8.1 | Tonnes | Colacasia | FAQ | 1500 | 2600 | 2200 | Rs/Quintal |
| | Cucumbar(Kheera) | 18.1 | Tonnes | Cucumbar | FAQ | 600 | 2000 | 1800 | Rs/Quintal |
| | Drumstick | 29.4 | Tonnes | Drumstick | FAQ | 1500 | 3500 | 3000 | Rs/Quintal |
| | Field Pea | 5.2 | Tonnes | Field Pea | FAQ | 2500 | 7500 | 5500 | Rs/Quintal |
| | French Beans (Frasbean) | 18.2 | Tonnes | French Beans (Frasbean) | FAQ | 4000 | 8000 | 6500 | Rs/Quintal |
| | Green Chilli | 117.1 | Tonnes | Green Chilly | FAQ | 2500 | 6500 | 5500 | Rs/Quintal |
| | Onion | 37.9 | Tonnes | 1st Sort | FAQ | 600 | 1200 | 1000 | Rs/Quintal |
| | Onion Green | 0.8 | Tonnes | Onion Green | FAQ | 6000 | 10000 | 8500 | Rs/Quintal |
| | Potato | 386.8 | Tonnes | (Red Nanital) | FAQ | 500 | 1500 | 1200 | Rs/Quintal |
| | Pumpkin | 25.8 | Tonnes | Other | FAQ | 400 | 1600 | 1200 | Rs/Quintal |
| | Raddish | 1 | Tonnes | Other | FAQ | 500 | 1000 | 800 | Rs/Quintal |
| | Ridgeguard(Tori) | 45 | Tonnes | Other | FAQ | 1500 | 4500 | 3500 | Rs/Quintal |
| | Sweet Potato | 11.5 | Tonnes | Hosur Green | FAQ | 2000 | 2600 | 2400 | Rs/Quintal |
| | Tomato | 350.1 | Tonnes | Deshi | FAQ | 1000 | 3400 | 2800 | Rs/Quintal |
| | Yam (Ratalu) | 26 | Tonnes | Other | FAQ | 1500 | 2500 | 2200 | Rs/Quintal |
| Gaddiannaram | Grapes | 10.9 | Tonnes | Black | Medium | 2000 | 5000 | 2714 | Rs/Quintal |
| | Guava | 18.3 | Tonnes | Guava | Medium | 667 | 3667 | 1333 | Rs/Quintal |
| | Mango | 200.5 | Tonnes | Totapuri | Medium | 350 | 1050 | 500 | Rs/Quintal |
| | Mousambi(Sweet Lime) | 290 | Tonnes | Mousambi | Medium | 600 | 3100 | 1300 | Rs/Quintal |
| Gudimalkapur | Beetroot | 3.2 | Tonnes | Beetroot | FAQ | 1400 | 2000 | 1700 | Rs/Quintal |
| | Bhindi(Ladies Finger) | 15.8 | Tonnes | Bhindi | FAQ | 1000 | 3000 | 2000 | Rs/Quintal |
| | Bitter gourd | 6.8 | Tonnes | Bitter Gourd | FAQ | 1500 | 2500 | 2000 | Rs/Quintal |
| | Bottle gourd | 7.4 | Tonnes | Bottle Gourd | FAQ | 600 | 800 | 700 | Rs/Quintal |
| | Brinjal | 13.4 | Tonnes | Arkasheela Mattigulla | FAQ | 500 | 1500 | 1000 | Rs/Quintal |
| | Cabbage | 6.9 | Tonnes | Cabbage | FAQ | 800 | 1400 | 1100 | Rs/Quintal |
| | Capsicum | 3 | Tonnes | Capsicum | FAQ | 2500 | 3500 | 3000 | Rs/Quintal |
| | Carrot | 19.3 | Tonnes | Carrot | FAQ | 1000 | 4000 | 2500 | Rs/Quintal |
| | Cauliflower | 3.9 | Tonnes | African Sarson | FAQ | 1000 | 3000 | 2000 | Rs/Quintal |
| | Cluster beans | 4.7 | Tonnes | Cluster Beans | FAQ | 2000 | 3000 | 2500 | Rs/Quintal |

Fig 4.8: Sample data of all the markets of hyderabad district in Telangana

## 4.6 Price Forecasting

A user interested in getting the forecasted price of any commodity, needs to tap on the option of price forecasting. It also displays a dropdown box filled with list of commodities for user's convenience. After selection of a commodity, dropdowns for

state, district and markets have been provided. Sequence of flow for commodity price forecast has been shown in Fig 4.9 and 4.10.

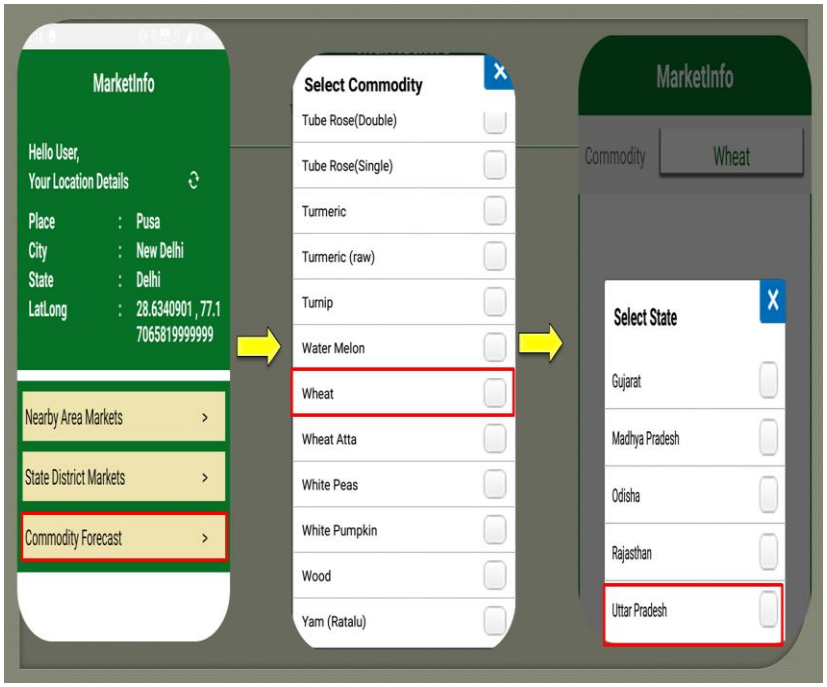



Fig 4.9: Sequence of selection of commodity and state in 'MandiInfo'


Fig 4.10: Selection of district and market in 'MandiInfo'

If a user taps on 'Price Forecasting' option, the forecasted price of wheat for next 5 days is shown. Fig 4.11 shows the forecasted price for Agra from 21/06/19 to 25/06/19.


Fig 4.11: Forecasted price of wheat in 'MandiInfo'

## 4.7 Notifications

'MandiInfo' also displays notifications at various points of time during its use. For example, when user tries to access the mobile app without turning the internet on, the notification "Please Check Your Internet Connection and Location" is displayed on the app is shown in 4.12. MandiInfo app retrieves real time data using APIs, therefore, internet connection is necessary to access the information.

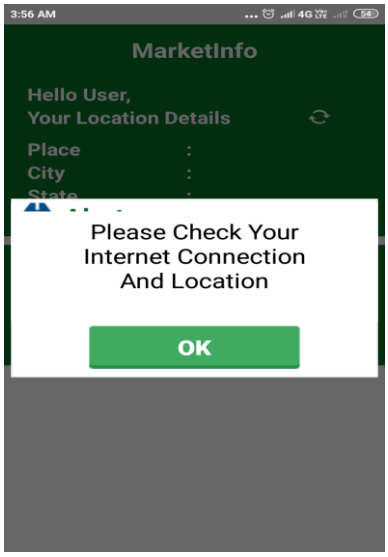



Fig 4.12: Notifications on 'MandiInfo' when internet is off

Similarly, when the internet connection is weak and the app is unsuccessful in loading the data, it gives a notification message "Please try again after some time" as shown Fig 4.13.

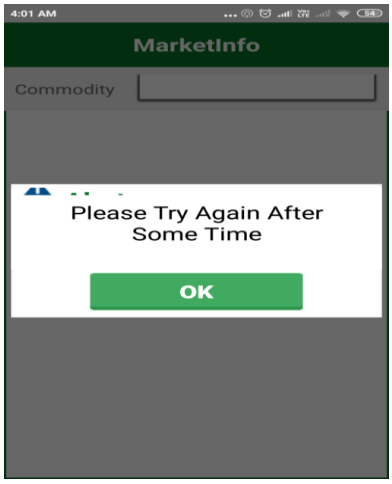



Fig 4.13. Notification when the network is weak

Another notification is displayed on the app when user clicks end button to exit from the app. The message “Do You Want to Exit” is displayed as shown in Fig 5.3.

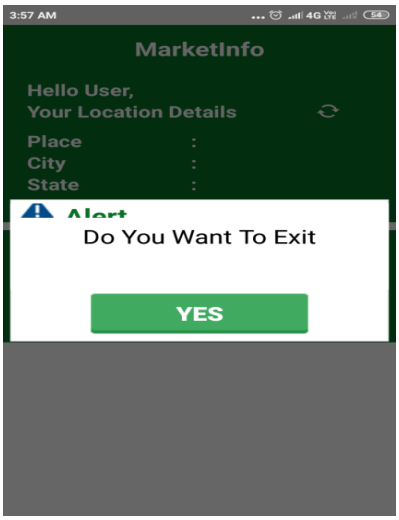



Fig 4.14: Confirmation notification to exit on ‘MandiInfo’

## 4.8 Discussion

Indian farmers are dependent on agriculture to run their families. They carry out various farming operations such as preparation of land, sowing, irrigation, weeding, spraying pesticides, harvesting and selling their produce to the market. In general, they face

problems in selling their produce to get the adequate price in return. Therefore, knowledge about the price and mandi for getting good profit is necessary. Unfortunately, most of the farmers are illiterate and lack the knowledge about mandi. At the same time, it is good to note that smart phones have reached our farmers because of faster technology advancement. These smartphones could be the best source of information transfer for the benefit of farmers. These devices are capable of running numerous applications on a single handset. Many useful apps have been developed for use in agriculture sector also. Apps have been developed for weather forecasting, management of farm, crop production and so on. The transfer of information can be efficiently and easily done even to the rural areas with the help of mobile apps. Therefore, an app called 'MandiInfo' has been developed for providing information about nearby mandis, prices of various commodities in these mandis and forecasting of price for coming days. This app has been developed for an android based smart phone and has been made available on Google Play Store.

This app is user-friendly and gives the output on screen very fast when the network connection speed is not a limitation. The result or output screens have been provided with facility of sharing it on other apps loaded on the device such as whatsapp, gmail, other email apps, pdf and so on.

Although MandiInfo is a very useful mobile based app but still there exists many areas where its functionality can be further extended. Some of the future improvement areas are listed below:

- The MandiInfo app can be made available in play store.
- ARIMA model is used in price forecasting. More advanced models can be used for efficient forecasting.
- As of now the web environments were used to run app on localhost. It can be deployed on webserver.
- This app is developed in English language and can be extended to multi-lingual app.

# CHAPTER V

# SUMMARY AND CONCLUSION

Agriculture is the backbone of Indian economy as more than 70% of the rural households depends on agriculture either directly or indirectly. Agriculture is the main source of livelihood for rural population of India. Agricultural farmers face the problem of low returns or sometimes even lose the commodity they bring to the market. It happens because the farmers do not get Mandi Information on time. There are many sources that provide the Mandi Information but all of them are web-based systems and portal. The web-based systems are not easily accessible by the farmers as most of them are illiterate and do not know how to use them. Moreover, these websites are accessible on devices like Desktops or Personal Computers (PCs) which the farmers are unable to afford and access. Mobile phones are the devices that are available at affordable prices now. Every class of people nowadays are able to keep mobile phones. Therefore, mean of communication which is easily accessible for the farmers can be through mobile phones. Keeping this in mind an effort has been made to develop an android based application for providing the Mandi Information to the farmers.

This study was undertaken with two objectives. The first objective was to design and develop a mobile app for locating nearby mandis and second objective was to develop an interface for current and forecasted prices of selected commodities. Considering these objectives, an android based mobile application named MandiInfo has been developed using XML coding and java programming language on the android platform. MandiInfo interacts with the database created using SQLite in the background.

MandiInfo has been developed by using android studio IDE with JDK version-8. It has the feature of detecting user's location automatically. The user's location, city, state, latitude and longitude are detected using the google places API provided by the Google. First objective provides the prices of commodities in nearby markets by retrieving data from AGMARKNET. Variety, grade, maximum, minimum and modal prices are provided to the user on the app. The market price data is retrieved from AGMARKNET using an API available at http://data.gov.in. The retrieved data is then parsed and made available on the app in an appropriate form. As it is dynamic data, it gets updated every

day for which the internet connection is mandatory. In case the current data is not updated, the app shows the previous day's data stored in the database of MandiInfo.

MandiInfo app can also show the prices of commodities of other states, if user wishes to see. This data can also be shared as pdf format to other places as well as on whatsapp, bluetooth, shareit etc. Price forecasting is another feature provided by MandiInfo app. It uses Rscript integrated with java and web environments. ARIMA model and forecast package have been used for price forecasting. XAMPP server has been used as local server to deploy the web environment. Forecast prices of five days for a commodity has also been provided on MandiInfo app.

The mobile application has been successfully developed for retrieval of market prices and forecasting prices of the commodities by which the farmer gets beneficial. Therefore, it is concluded that this mobile application is very convenient for easy and timely retrieval of the Mandi Information for the farmers and also for predicting the prices of the commodities that may prevail in the markets in future.

**Development of a Mobile App for Locating Nearby Mandis and Price Forecasts of Selected Agricultural Commodities**

# ABSTRACT

In recent years, tremendous improvements have been seen in the field of mobile technology. This enhancement has made the smartphones and mobile apps, an integral part of human life. These days, the smartphones and mobile apps have made remarkable changes in all sectors of the India. The mobile apps are being used everywhere from highly developed cosmopolitan cities to the rural villages. In this study, an attempt has been made to develop a mobile application named 'MandiInfo' for Indian farmers. This mobile application is aimed to provide timely and valid Mandi information to the farmers and prices of the commodities in the nearby markets. This app can also help farmers by providing the forecasted prices of the commodities. The farmers are able to check the prices of the commodities in the markets of all over India and can share data with others in the form of pdf file. In this thesis, the 'MandiInfo' app has been developed for the android platform with minimum SDK version of API 21: Android 5 (Lollipop). The application is based on the 3-tier architecture of the software development. The client-side interface is the android application which is implemented by the JAVA programming language and XML. The next layer is a database layer. The database layer contains the data of AGMARKNET portal. APIs from data.gov.in, AGMARKNET and Google places API has been used for interconnection among various layers. The application has been tested with the help of API from AGMARKNET and provided the intended results as specified. The developed android application 'MandiInfo' would be of great use to the farmers to deal with current commodities prices like maximum, minimum and modal prices in the market and forecasted prices.

**Keywords**: Mobile App, Market information access, Application Programming Interface (API), Location Details, Price Forecasts.

# चयनित फसलों के निकटवर्ती मंडियों और मूल्य पूर्वानुमान के लिए एक मोबाइल ऐप का विकास


## सार

पिछले कुछ वर्षों में, मोबाइल प्रौद्योगिकी के क्षेत्र में जबरदस्त विकास देखा गया है। इस प्रगति ने स्मार्टफोन और मोबाइल ऐप को मानव जीवन का अभिन्न अंग बना दिया है। इन दिनों, स्मार्टफोन और मोबाइल ऐप ने भारतीय अर्थव्यवस्था के सभी क्षेत्रों में उल्लेखनीय बदलाव किए हैं। अत्यधिक विकसित शहरों से लेकर छोटे-छोटे गांवों तक हर जगह मोबाइल ऐप का उपयोग किया जा रहा है। इस अध्ययन में, भारत के किसानों के लिए 'मंडी इंफो' नामक एक मोबाइल एप्लिकेशन विकसित करने का प्रयास किया गया है। इस मोबाइल एप्लिकेशन का उद्देश्य किसानों को समय पर मान्य मंडी जानकारी प्रदान करना है जो आस-पास के बाजारों में प्रचलित वस्तुओं की कीमतों के बारे में है। यह ऐप किसानों को वस्तुओं के पूर्वानुमानित मूल्य प्रदान करने में भी मदद करता है। यह ऐप पूरे भारत के बाजारों में वस्तुओं की कीमतों की जांच करने में सक्षम हैं और यह पी.डी.एफ. फाइल के रूप में दूसरों के साथ डेटा साझा कर सकता है। इस थीसिस में, (मंडी इंफो 'ऐप को एंड्रॉइड प्लेटफॉर्म पर एपीआई 21: Android 5 (लॉलीपॉप) के न्यूनतम एसडीके संस्करण के साथ विकसित किया गया है। यह ऐप सॉफ्टवेयर विकास के 3-स्तरीय वास्तुकला पर आधारित है। क्लाइंट-साइड इंटरफ़ेस एंड्रॉइड एप्लिकेशन है जो जावा (Java) प्रोग्रामिंग भाषा (Language) और एक्स.एम.एल. द्वारा कार्यान्वित किया गया है। अगली लेयर एक डेटाबेस लेयर है। डेटाबेस लेयर में AGMARKNET पोर्टल का डेटा होता है। Data.gov.in, AGMARKNET और Google स्थानों से API का उपयोग विभिन्न परतों के बीच परस्पर संबंध के लिए किया गया है। यह ऐप data.gov.in पर उपलब्ध ए.पी. आई. (API) की मदद से AGMARKNET से वस्तुओं की कीमतों को निकलता है और परिणाम में दर्शाता है | विकसित एंड्राइड मंडीइन्फो (MandiInfo) बाजार के अधिकतम, न्यूनतम, मोडल कीमतों के बारे में सूचना एवं फोरकास्टेड कीमतों को प्रदान कर किसानों को बहुत लाभान्वित करेगा|

# BIBLIOGRAPHY


AGMARKNET Available at: http://agmarknet.gov.in/

Agro Connect Available at: http://agroconnect.com.ng/

Amrutha, C.P. (2009). Market information system and its application for Agricultural commodities in Karnataka state – A case of onion. *Unpublished Ph.D. thesis*, University of Agricultural Sciences, Dharwad.

Android Studio: The Official IDE of Android (2018) Available at:https://developer.android.com/studio/index.html

Arjun, K. M. (2013). Indian agriculture-status, importance and role in Indian Economy. *International Journal of Agriculture and Food Science Technology*, **4(4)**, 343-346.

Bandyopadhyay, T. (2015). KSUSoy YieldCalc: An innovative native android app to estimate soybean yield before harvest using conventional approach. *Doctoral dissertation*, Kansas State University, Manhattan.

Box, G. E. P. and Pierce, D. A. (1970). Distribution of residual autocorrelations in Autoregressive-Integrated Moving Average time series models, *Journal of the American Statistical Association*, **65**, 1509–1526.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (2007). *Time-Series Analysis: Forecasting and Control, 3rd edition*. Pearson Education, India.

Farhath, Z. A., Arputhamary, B. and Arockiam, L. (2016). A Survey on ARIMA Forecasting using Time Series Model. *International Journal of Computer Science and Mobile Computing* (IJCSMC), **5(8)**, 04-109.

Brahler, S. (2010). *Analysis of the Android Architecture*. Karlsruhe Institute for Technology, 1-8.

Digital Mandi India Available at: https://digital-mandi-india.en.aptoide.com/

Kisan Network: Available at: https://kisannetwork.com/

Suvidha, K. (2016). Mobile application (Online). Available at: https://play.google.com/store/apps/details?id=in.cdac.bharatd.agriapp&hl=en
Life Cycle of an Activity: Available at: https://developer.android.com/guide/components/activities.html.

Nickels, W. G. (1978). *Marketing principles: a broadened concept of marketing*. Englewood Cliffs, New Jersey, Prentice-Hall.

Fazlur, R. M. (2003). Agricultural Marketing System in Bangladesh. Agricultural Marketing, **45 (4)**, 29-32.

Rehman, S. U., Selvaraj, M. and Ibrahim, M. S. (2012). Indian agricultural marketing-a review. *Asian Journal of Agriculture and Rural Development*, **2(1)**, 69-75.

Singhal, M., Verma, K. and Shukla, A. (2011). Krishi Ville-Android based Solution for Indian Agriculture, 5th *International Conference of Advanced Networks and Telecommunication Systems*, IEEE, 1-5.

Smyth, N. (2017). Android Studio 3.0 Development Essentials-Android 8 Edition. Payload Media, Inc.

SQLite Tutorial: Available at: https://www.tutorialspoint.com/sqlite/index.htm

SQLite: Available at: https://www.sqlite.org/

Tutorial, XML (2006). W3Schools. Available at: http://www.w3schools. com/xml /default.asp.

Vogel, L. (2010). Android SQLite Database and Content Provider-Tutorial. Java, Eclipse, Android and Web Programming Tutorials.

Vogel, L. (2013). Android Development Tutorial. Based on Android.

# SAMPLE CODES

## BASEACTIVITY

```
package com.market.marketinfo.activities;

import android.app.Activity;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.market.marketinfo.FetchMarketList;
import com.market.marketinfo.R;
import com.market.marketinfo.customUI.CustomDialog;

public abstract class BaseActivity extends Activity {

    public LayoutInflater inflater;
    public LinearLayout mainToolBar;
    public LinearLayout llBase;
    private CustomDialog customDialog;
    private ProgressDialog dialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.base);
        inflater = this.getLayoutInflater();
        llBase = findViewById(R.id.llBase);
        mainToolBar = findViewById(R.id.mainToolBar);
        initializeView();
    }
```

```
    public abstract void initializeView();
    public abstract void loadData();



    @Override
    protected void onResume() {
        super.onResume();
    }

    /** Method to Show the alert dialog **/
    public void showCustomDialog(Context context, String
strTitle, String strMessage, String firstBtnName,
                                 String secondBtnName,
String from) {
        runOnUiThread(new RunshowCustomDialogs(context,
strTitle, strMessage, firstBtnName, secondBtnName, from,
true));
    }

    // For showing Dialog message.
    class RunshowCustomDialogs implements Runnable {
        private String strTitle;// Title of the dialog
        private String strMessage;// Message to be shown in
dialog
        private String firstBtnName;
        private String secondBtnName;
        private String from;
        private String params;
        private boolean isCancelable = false;
        private View.OnClickListener posClickListener;
        private View.OnClickListener negClickListener;

        public RunshowCustomDialogs(Context context, String
strTitle, String strMessage, String firstBtnName,
                                    String secondBtnName,
String from, boolean isCancelable) {
            this.strTitle = strTitle;
            this.strMessage = strMessage;
            this.firstBtnName = firstBtnName;
            this.secondBtnName = secondBtnName;
            this.isCancelable = isCancelable;
            if (from != null)
                this.from = from;
            else
                this.from = "";
```

```
        }

        public RunshowCustomDialogs(Context context, String
strTitle, String strMessage, String firstBtnName,
                                    String secondBtnName,
String from, boolean isCancelable, String params) {
            this.strTitle = strTitle;
            this.strMessage = strMessage;
            this.firstBtnName = firstBtnName;
            this.secondBtnName = secondBtnName;
            this.isCancelable = isCancelable;
            if (from != null)
                this.from = from;
            else
                this.from = "";

            if (params != null)
                this.params = params;
            else
                this.params = "";
        }

        @Override
        public void run() {
            if (customDialog != null &&
customDialog.isShowing())
                customDialog.dismiss();

            View view;

            view =
inflater.inflate(R.layout.custom_common_popup, null);

                /*customDialog = new
CustomDialog(BaseActivity.this, view,

preference.getIntFromPreference(Preference.DEVICE_DISPLAY_W
IDTH, 320) - 10,

LinearLayout.LayoutParams.WRAP_CONTENT, true);*/
            customDialog = new
CustomDialog(BaseActivity.this, view,
                    LinearLayout.LayoutParams.WRAP_CONTENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT,
true);
            customDialog.setCancelable(isCancelable);
            TextView tvTitle = (TextView)
```

```
view.findViewById(R.id.tvTitlePopup);
            TextView tvMessage = (TextView)
view.findViewById(R.id.tvMessagePopup);
            Button btnYes = (Button)
view.findViewById(R.id.btnYesPopup);
            Button btnNo = (Button)
view.findViewById(R.id.btnNoPopup);

            tvTitle.setText("" + strTitle);
            tvMessage.setText("" + strMessage);
            btnYes.setText("" + firstBtnName);

            if (secondBtnName != null &&
!secondBtnName.equalsIgnoreCase(""))
                btnNo.setText("" + secondBtnName);
            else
                btnNo.setVisibility(View.GONE);

            if (posClickListener == null)
                btnYes.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        customDialog.dismiss();

                        if (from != null &&
from.equalsIgnoreCase("cancelorder"))
                            onButtonYesClick(from, params);
                        else
                            onButtonYesClick(from);
                    }
                });
            else

btnYes.setOnClickListener(posClickListener);

            if (negClickListener == null)
                btnNo.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        customDialog.dismiss();
                        onButtonNoClick(from);
                    }
                });
            else
                btnNo.setOnClickListener(negClickListener);
```

```
            try {
                if (!customDialog.isShowing())
                    customDialog.showCustomDialog();
            } catch (Exception e) {
            }
        }
    }

    public void onButtonYesClick(String from, String
params) {

    }
    public void onButtonYesClick(String from) {

    }

    public void onButtonNoClick(String from) {

    }

    public void showLoader(String str) {
        runOnUiThread(new RunShowLoaderCustom(str));
    }

    public void showLoader(String msg, String title) {
        runOnUiThread(new RunShowLoaderCustom(msg, title));
    }

    /**
     * Name: RunShowLoader Description: This is to show the
loading progress
     * dialog when some other functionality is taking
place.
     **/
    class RunShowLoaderCustom implements Runnable {
        private String title;
        private String strMsg;

        public RunShowLoaderCustom(String strMsg) {
            this.strMsg = strMsg;
        }

        public RunShowLoaderCustom(String strMsg, String
title) {
            this.strMsg = strMsg;
            this.title = title;
        }
```

```
        @Override
        public void run() {
            try {
                if (dialog == null)
                    dialog = new
ProgressDialog(BaseActivity.this,
R.style.Theme_Dialog_Translucent);
                if (dialog == null|| (dialog != null &&
!dialog.isShowing()))
                {
                    dialog =
ProgressDialog.show(BaseActivity.this,    title, strMsg);
                    dialog.setCancelable(false);

dialog.setCanceledOnTouchOutside(false);
                }
                else if (dialog == null|| (dialog != null
&& dialog.isShowing()))
                {
                    dialog.setMessage(strMsg);
                }
            } catch (Exception e) {
            }
        }
    }

    public void hideLoader() {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                try {
                    if (dialog != null &&
dialog.isShowing())
                        dialog.dismiss();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

    }

    public void loadTotalMarketList(){
        Intent uploadTraIntent=new
Intent(this,FetchMarketList.class);
        startService(uploadTraIntent);
```

```
    }

}
```

## Dashboard activity

```
package com.market.marketinfo.activities;

import android.content.Intent;
import android.location.Address;
import android.provider.Settings;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.google.android.gms.maps.model.LatLng;
import com.market.marketinfo.R;
import com.market.marketinfo.customUI.AppConstants;
import com.market.marketinfo.location.GPSCallback;
import com.market.marketinfo.location.GPSErrorCode;
import com.market.marketinfo.location.GPSPreference;
import com.market.marketinfo.location.GPSUtills;

import java.util.Calendar;

public class DashboardActivity extends BaseActivity
implements GPSCallback
{

    private LinearLayout llMain;
    public GPSUtills gpsUtills ;
    private TextView
tvLatLong,tvState,tvCityName,tvAreaName;
    private ImageView ivRefresh;
    private LinearLayout
llNearByAreaMarkets,llNearByCityMarkets,llStateDistrictMark
ets,llCommodityForeCast;
    private Address locationDetails;
    private GPSPreference gpsPreference;

    @Override
    public void initializeView() {
        llMain =
(LinearLayout)inflater.inflate(R.layout.dashboard,null);
```

```
llBase.addView(llMain,LinearLayout.LayoutParams.MATCH_PAREN
T,LinearLayout.LayoutParams.MATCH_PARENT);
        tvLatLong =
(TextView)llMain.findViewById(R.id.tvLatLong);
        tvState =
(TextView)llMain.findViewById(R.id.tvState);
        tvCityName =
(TextView)llMain.findViewById(R.id.tvCityName);
        tvAreaName =
(TextView)llMain.findViewById(R.id.tvAreaName);
        ivRefresh = (ImageView)
llMain.findViewById(R.id.ivRefresh);
        llNearByAreaMarkets = (LinearLayout)
llMain.findViewById(R.id.llNearByAreaMarkets);
        llNearByCityMarkets = (LinearLayout)
llMain.findViewById(R.id.llNearByCityMarkets);
        llStateDistrictMarkets = (LinearLayout)
llMain.findViewById(R.id.llStateDistrictMarkets);
        llCommodityForeCast = (LinearLayout)
llMain.findViewById(R.id.llCommodityForeCast);
        gpsUtills =
GPSUtills.getInstance(DashboardActivity.this);
        gpsUtills.setLogEnable(true);
        gpsUtills.setPackegeName(getPackageName());
        gpsUtills.setListner(DashboardActivity.this);
        gpsUtills.isGpsProviderEnabled();

        ivRefresh.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(gpsUtills.isInternetConnectionAvailable()){
                    gpsUtills.getCurrentLatLng();
                }else{

showCustomDialog(DashboardActivity.this,"Alert","Please
Check Your Internet Connection","Ok","","internet");
                }
            }
        });
        llNearByCityMarkets.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(gpsUtills.isInternetConnectionAvailable()&&locationDetai
```

```
ls!=null){
                    Intent intent = new
Intent(DashboardActivity.this, MarketListActivity.class);

intent.putExtra("locationDetails",locationDetails);
                    intent.putExtra("isFromCity",true);
                    startActivity(intent);
                }else{

showCustomDialog(DashboardActivity.this,"Alert","Please
Check Your Internet Connection And
Location","Ok","","internet");
                }
            }
        });
        llNearByAreaMarkets.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(gpsUtills.isInternetConnectionAvailable()&&locationDetai
ls!=null){
                    Intent intent = new
Intent(DashboardActivity.this, MarketListActivity.class);

intent.putExtra("locationDetails",locationDetails);
                    intent.putExtra("isFromCity",true);
                    startActivity(intent);
                }else{

showCustomDialog(DashboardActivity.this,"Alert","Please
Check Your Internet Connection And
Location","Ok","","internet");
                }
            }
        });
        llStateDistrictMarkets.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(gpsUtills.isInternetConnectionAvailable()){
                    Intent intent = new
Intent(DashboardActivity.this,
CustomWebViewActivity.class);

intent.putExtra("locationDetails",locationDetails);
```

```
                    intent.putExtra("isFromCity",false);
                    startActivity(intent);
                }else{

showCustomDialog(DashboardActivity.this,"Alert","Please
Check Your Internet Connection","Ok","","internet");
                }
            }
        });
        llCommodityForeCast.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(gpsUtills.isInternetConnectionAvailable()){
                    Intent intent = new
Intent(DashboardActivity.this, CommodityForeCast.class);

intent.putExtra("locationDetails",locationDetails);
                    intent.putExtra("isFromCity",false);
                    startActivity(intent);
                }else{

showCustomDialog(DashboardActivity.this,"Alert","Please
Check Your Internet Connection","Ok","","internet");
                }
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
        gpsUtills.isGpsProviderEnabled();
        gpsPreference = new
GPSPreference(DashboardActivity.this);
        if(gpsUtills.isInternetConnectionAvailable()) {
            String LastSyncDate =
gpsPreference.getStringFromPreference(GPSPreference.CURRENT
_DATA_DATE, ""); //24/06/2019
            if (!LastSyncDate.equalsIgnoreCase("")){
                if
(!LastSyncDate.equalsIgnoreCase(AppConstants.getCurrentDate
())) {  //25/06/2019
                    loadTotalMarketList();
                }
            }else{
```

```
                loadTotalMarketList();
            }
        }
    }

    @Override
    public void onButtonYesClick(String from) {
        if(from.equalsIgnoreCase("finish")){
            finish();
        }
    }

    @Override
    public void loadData() {

    }

    @Override
    protected void onStart() {
        super.onStart();
        gpsUtills.connectGoogleApiClient();
//        gpsUtills.getCurrentLatLng();
    }

    @Override
    public void onBackPressed() {
        showCustomDialog(DashboardActivity.this,"Alert","Do
You Want To Exit","Yes","","finish");
    }

    @Override
    public void onButtonYesClick(String from, String
params) {

    }

    @Override
    public void gotGpsValidationResponse(Object response,
GPSErrorCode code) {
        if(code ==
GPSErrorCode.EC_GPS_PROVIDER_NOT_ENABLED){
        }
        if(code == GPSErrorCode.EC_GPS_PROVIDER_ENABLED){
            gpsUtills.getCurrentLatLng();
        }
        if(code == GPSErrorCode.EC_LOCATION_FOUND){
            LatLng latLng = (LatLng)response;
```

```
gpsUtills.getAddressFromLatLng(latLng.latitude,latLng.longi
tude);
        }
        else if(code == GPSErrorCode.EC_ADDRESS_FOUND){
            locationDetails = (Address)response;
            if(locationDetails!=null){

tvLatLong.setText(locationDetails.getLatitude() +" ,
"+locationDetails.getLongitude());

tvState.setText(locationDetails.getAdminArea()+"");

tvAreaName.setText(locationDetails.getSubLocality()+"");

tvCityName.setText(locationDetails.getLocality()+"");
            }
        }
    }
}
```

**Android manifest.xml**

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.market.marketinfo">

    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.SDCARD_WRITE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMM
ANDS" />
    <uses-permission
android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/app_icon"
```

```
        android:label="@string/app_name"
        android:usesCleartextTraffic="true"
        android:theme="@style/AppTheme">
        <provider

android:name="android.support.v4.content.FileProvider"

android:authorities="com.market.marketinfo.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data

android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths"/>
        </provider>
        <service
android:name="com.market.marketinfo.FetchMarketList" />
        <activity

android:name="com.market.marketinfo.activities.SplashScreen
Activity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity

android:name="com.market.marketinfo.activities.DashboardAct
ivity"
            android:screenOrientation="portrait" />
        <activity

android:name="com.market.marketinfo.activities.MarketListAc
tivity"
            android:screenOrientation="portrait" />
        <activity

android:name="com.market.marketinfo.activities.CommodityFor
eCast"
            android:screenOrientation="portrait" />
        <activity

android:name="com.market.marketinfo.activities.CustomWebVie
```

```
wActivity" />
    </application>
</manifest>
```

**ServiceURL**

```
package com.market.marketinfo.network;

public class ServiceUrl {

    public static String MAIN_URL =
"https://api.data.gov.in/resource/9ef84268-d588-465a-a308-
a864a43d0070?api-key=";

    public static String API_KEY =
"579b464db66ec23bdd000001b539da91d6e048fc5b1c8f3cd97c9d9f";

    public String sampleURL =
"https://api.data.gov.in/resource/9ef84268-d588-465a-a308-
a864a43d0070?api-
key=579b464db66ec23bdd000001cdd3946e44ce4aad7209ff7b23ac571
b&format=xml&offset=0&limit=10";

    public static String TRIPURA = "fb9c1a5c-198e-4627-
bf7b-8a2dadbc5d51";
    public static String UTTARAKHAND = "93f81135-7533-4ccf-
b3ba-5c8210633a91";
    public static String MAHARASHTRA = "92e7b0a1-1462-47ba-
b8e1-f66784a720b4";
    public static String MADHYAPRADESH = "06a6f995-6f98-
4439-b9b2-abff1861e71d";
    public static String RAJASTHAN = "c8b057d9-fd3d-4695-
a7d9-c5c16b6504c2";
    public static String KERALA = "20138aa5-e53f-4d83-a7e2-
f280688b11cc";
    public static String UTTARPRADESH = "28913708-c969-
4918-b3ac-48d8b1cba903";
    public static String MIZORAM = "99a9e627-834a-45ce-
9923-7cfa2a09513d";
    public static String NCTOFDELHI = "5efb8949-f82c-4f65-
a9f3-40112526e29b";
    public static String PUNJAB = "97878f8a-97c3-4b52-b4bd-
a40c530d0fe6";
    public static String ODISHA = "afd4d5d8-2f57-4288-9b0d-
eaffaca51ba3";
```

```
    public static String KEY_STATE_TRIPURA="Tripura";
    public static String
KEY_STATE_UTTARAKHAND="Uttarakhand";
    public static String
KEY_STATE_MAHARASHTRA="Maharashtra";
    public static String
KEY_STATE_MADHYAPRADESH="Madhya%20Pradesh";
    public static String KEY_STATE_RAJASTHAN="Rajasthan";
    public static String KEY_STATE_KERALA="Kerala";
    public static String
KEY_STATE_UTTARPRADESH="Uttar%20Pradesh";
    public static String KEY_STATE_MIZORAM="Mizoram";
    public static String
KEY_STATE_DELHI="NCT%20of%20Delhi";
    public static String KEY_STATE_PUNJAB="Punjab";
    public static String KEY_STATE_ODISHA="Odisha";

    public enum ServiceAction {
        GET_DATA,
        LOGINAUTH,
        DELETE_SESSION,
        CREATESESSION,
        MOVIE_LIST
    }
}
```

**URL Jsonpost**

```
package com.market.marketinfo.network;

import android.text.TextUtils;

import com.market.marketinfo.dataobjects.RequestDO;
import com.market.marketinfo.dataobjects.ResponceDo;
import com.market.marketinfo.parsers.BaseJsonParser;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.Reader;
import java.io.StringWriter;
import java.io.Writer;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.Set;

public class UrlJsonPost {

    HttpURLConnection httpURLConnection;
    InputStream inputStream;
    URL url;
    ResponceReceiverInterface responceReceiverInterface;

    public UrlJsonPost(ResponceReceiverInterface
responceReceiverInterface) {
        this.responceReceiverInterface =
responceReceiverInterface;
    }

    public void sendJsonRequest(RequestDO requestDO) throws
MalformedURLException {
        ResponceDo responceDo = null;
        try {
            switch (requestDO.webmethods) {
                case GET:
                    String mainURL = "" +
requestDO.request;
```

```
                    mainURL = mainURL.replace(" ", "%20");
                    url = new URL(mainURL);
                    httpURLConnection = (HttpURLConnection)
url.openConnection();

httpURLConnection.setRequestMethod("GET");

httpURLConnection.setReadTimeout(150000);

httpURLConnection.setConnectTimeout(150000);
                    if (requestDO.hmHeaders != null &&
requestDO.hmHeaders.size() > 0) {
                        HashMap<String, String> hashMap =
requestDO.hmHeaders;
                        Set<String> mapKeys =
hashMap.keySet();
                        for (String strHeaderKey : mapKeys)

httpURLConnection.setRequestProperty(strHeaderKey,
requestDO.hmHeaders.get(strHeaderKey));
                    }
                    break;
                case POST:
                case DELETE:
                    String mainURLPost = "" +
requestDO.request;
                    mainURLPost = mainURLPost.replace(" ",
"%20");
                    url = new URL(mainURLPost);
                    httpURLConnection = (HttpURLConnection)
url.openConnection();
                    if (requestDO.serviceAction ==
ServiceUrl.ServiceAction.DELETE_SESSION) {

httpURLConnection.setRequestMethod("DELETE");
                    } else {

httpURLConnection.setRequestMethod("POST");
                    }

                    httpURLConnection.setDoOutput(true);

httpURLConnection.setReadTimeout(150000);

httpURLConnection.setConnectTimeout(150000);
                    if (requestDO.hmHeaders != null &&
requestDO.hmHeaders.size() > 0) {
```

```
                        HashMap<String, String> hashMap =
requestDO.hmHeaders;
                        Set<String> mapKeys =
hashMap.keySet();
                        for (String strHeaderKey : mapKeys)

httpURLConnection.setRequestProperty(strHeaderKey,
requestDO.hmHeaders.get(strHeaderKey));
                    }

                    if
(!TextUtils.isEmpty(requestDO.payloadBody)) {

httpURLConnection.setDoOutput(true);
                        OutputStream outputStream =
httpURLConnection.getOutputStream();
                        BufferedWriter writer = new
BufferedWriter(new OutputStreamWriter(outputStream, "UTF-
8"));

writer.write(requestDO.payloadBody);
                        writer.flush();
                        writer.close();
                        outputStream.close();
                    }
                    break;
            }
            System.setProperty("http.keepAlive", "false");
//possible eof exception problem
            inputStream =
httpURLConnection.getInputStream();

            httpURLConnection.getResponseMessage();
            httpURLConnection.getResponseCode();
            //==============Convert Input Strem to String
format
            String responce = "";
            if (inputStream != null) {
                Writer writer = new StringWriter();
                char[] buffer = new char[1024];
                Reader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"));
                int n;
                while ((n = reader.read(buffer)) != -1) {
                    writer.write(buffer, 0, n);
                }
                responce = writer.toString();
```

```
                writer.close();
            }
            if (!responce.isEmpty()) {
                responceDo = new ResponceDo();
                BaseJsonParser baseJsonParser =
BaseJsonParser.getParese(requestDO.serviceAction);
                responceDo.serviceAction =
requestDO.serviceAction;
                if (baseJsonParser != null) {
                    responceDo.object =
baseJsonParser.parse(responce);
                } else {
                    responceDo.object = responce;
                }
            }

responceReceiverInterface.onResponceReceive(responceDo);
        } catch (Exception e) {
            e.printStackTrace();

responceReceiverInterface.onResponceReceive(null);
        }
    }

}
```

**Market list activity**

```
package com.market.marketinfo.activities;

import android.location.Address;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.View;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.market.marketinfo.R;
import com.market.marketinfo.adapter.MarketListAdapter;
import com.market.marketinfo.customUI.CustomBuilder;
import com.market.marketinfo.database.DatabaseHelper;
import com.market.marketinfo.dataobjects.FilterListDo;
```

```
import com.market.marketinfo.dataobjects.Record;
import com.market.marketinfo.dataobjects.RequestDO;
import com.market.marketinfo.dataobjects.ResponceDo;
import
com.market.marketinfo.network.ResponceReceiverInterface;
import com.market.marketinfo.network.ServiceUrl;
import com.market.marketinfo.network.UrlJsonPost;

import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.LinkedHashMap;
import java.util.Vector;
import java.util.function.Predicate;

import static
com.market.marketinfo.network.ServiceUrl.ServiceAction.GET_
DATA;

public class MarketListActivity extends BaseActivity
implements ResponceReceiverInterface {

    private LinearLayout llFilterLayout;
    private RecyclerView rvFilterList;
    private ArrayList<Record> alRecords = new
ArrayList<>();
    private ArrayList<Record> alTempRecords = new
ArrayList<>();
    private Address locationDetails;
    private UrlJsonPost urlJsonPost;
    private MarketListAdapter adapter;
    private boolean isFromCity = true;
    private EditText et_SearchFilter;
    private TextView tvSelectCommodity;
    private Vector<String> vecCommodities = new Vector<>();
    private String mSelectedCommodity = "";
    private DatabaseHelper mDataBase;

    @Override
    public void initializeView() {
        llFilterLayout = (LinearLayout)
inflater.inflate(R.layout.main_filter_layout, null);
        llBase.addView(llFilterLayout);
        if (getIntent().getExtras() != null) {
            locationDetails = (Address)
getIntent().getExtras().get("locationDetails");
            isFromCity = (boolean)
```

```
getIntent().getExtras().get("isFromCity");
        }
        rvFilterList = (RecyclerView)
llFilterLayout.findViewById(R.id.rvFilterList);
        et_SearchFilter = (EditText)
llFilterLayout.findViewById(R.id.et_SearchFilter);
        tvSelectCommodity = (TextView)
llFilterLayout.findViewById(R.id.tvSelectCommodity);
        rvFilterList.setLayoutManager(new
LinearLayoutManager(MarketListActivity.this));
        showLoader("LoadingData");
        mDataBase = new
DatabaseHelper(MarketListActivity.this);
        adapter = new MarketListAdapter(alRecords,
MarketListActivity.this);
        rvFilterList.setAdapter(adapter);
        loadData();
        tvSelectCommodity.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showFilter();
            }
        });
        et_SearchFilter.setVisibility(View.GONE);
        et_SearchFilter.addTextChangedListener(new
TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s,
int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int
start, int before, int count) {

            }

            @Override
            public void afterTextChanged(Editable s) {
                getSearchItems(s.toString());
            }
        });
    }

    public void getSearchItems(final String searchText) {
```

```
        synchronized ("MARKET") {
            alTempRecords.clear();
            if (searchText.equalsIgnoreCase("All")) {
                adapter.refresh(alRecords);
            } else {
                String str =
searchText.toString().toLowerCase();
                for (int i = 0; alRecords != null && i <
alRecords.size(); i++) {
                    Record recordDO = alRecords.get(i);
                    if
(recordDO.getCommodity().toLowerCase().contains(str.toLower
Case().trim()) ||
recordDO.getMarket().toLowerCase().contains(str.toLowerCase
().trim())) {
                        if
(!alTempRecords.contains(recordDO))
                            alTempRecords.add(recordDO);
                    }
                }
                if (str.length() > 0)
                    adapter.refresh(alTempRecords);
                else
                    adapter.refresh(alRecords);
            }
        }
    }


    @Override
    public void loadData() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                urlJsonPost = new
UrlJsonPost(MarketListActivity.this);
                RequestDO requestDO = new RequestDO();
                requestDO.webmethods =
RequestDO.WebMethods.GET;
                requestDO.payloadBody = "";
                requestDO.serviceAction = GET_DATA;
                requestDO.request = ServiceUrl.MAIN_URL +
"" + ServiceUrl.API_KEY + getFilterQuery() + "" +
"&format=json&limit=10000";
                try {
                    urlJsonPost.sendJsonRequest(requestDO);
                } catch (MalformedURLException e) {
```

```
                    e.printStackTrace();
                }
            }
        }).start();


    }

    private String getFilterQuery() {
        String query = "";
        if (isFromCity) {
            if
(locationDetails.getLocality().contains("Delhi") ||
locationDetails.getAdminArea().contains("Delhi")) {
                query = "&filters[state]=" +
ServiceUrl.KEY_STATE_DELHI;
            } else {
                query = "&filters[district]=" +
locationDetails.getLocality();
            }
        } else {
            query = "&filters[market]=" +
locationDetails.getSubLocality();
        }
        return query;
    }

    public void showFilter() {
        if (vecCommodities != null && vecCommodities.size()
> 0) {
            CustomBuilder customBuilder = new
CustomBuilder(MarketListActivity.this, "Select Commodity",
true);

customBuilder.setSingleChoiceItems(vecCommodities,
mSelectedCommodity, new CustomBuilder.OnClickListener() {
                @Override
                public void onClick(CustomBuilder builder,
Object selectedObject) {
                    mSelectedCommodity = (String)
selectedObject;

tvSelectCommodity.setText(mSelectedCommodity);
                    getSearchItems(mSelectedCommodity);
                    builder.dismiss();
                }
            });
```

```
            customBuilder.show();
        } else {
            showCustomDialog(MarketListActivity.this,
"Alert", "Data not updated in Agmarknet", "Ok", "",
"finish");
        }
    }



    public void getFilterCommodityData() {
        showLoader("Preparing List");
        if (alRecords != null && alRecords.size() > 0) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    for (Record record : alRecords) {
                        String commodity =
record.getCommodity();
                        if (vecCommodities != null &&
!vecCommodities.contains(commodity)) {
                            vecCommodities.add(commodity);
                        }
                    }
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            hideLoader();
                            if (vecCommodities != null &&
vecCommodities.size() > 0) {
                                mSelectedCommodity = "All";
                                vecCommodities.add(0,
"All");

tvSelectCommodity.setText(mSelectedCommodity);
                                showFilter();
                            }
                        }
                    });
                }
            }).start();
        } else {
            hideLoader();
        }
    }
```

```
    @Override
    public void onButtonYesClick(String from) {
        if (from.equalsIgnoreCase("finish")) {
            finish();
        }
    }

    @Override
    public void onResponceReceive(final Object object) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                hideLoader();
                if (object == null) {

showCustomDialog(MarketListActivity.this, "Alert", "Please
Try Again After Some Time", "Ok", "", "finish");
                } else {
                    FilterListDo obj = (FilterListDo)
((ResponceDo) object).object;
                    alRecords = obj.getRecords();
                    if (alRecords.size() <= 0) {
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                String district =
locationDetails.getLocality();
                                if
(mDataBase.openDataBase() != null)
                                    alRecords =
mDataBase.getOldData(district);
                                if(alRecords.size()<=0){
                                    alRecords =
mDataBase.getDummyData();
                                }
                                runOnUiThread(new
Runnable() {
                                    @Override
                                    public void run() {
                                        if
(alRecords.size() <= 0) {

showCustomDialog(MarketListActivity.this, "Alert", "Data
not updated in agmarknet", "Ok", "", "finish");
                                        } else {

adapter.refresh(alRecords);
```

```
getFilterCommodityData();
                                                }
                                            }
                                        });
                                    }
                                }).start();

                            } else {
                                adapter.refresh(obj.getRecords());
                                getFilterCommodityData();
                            }
                        }
                    }
                });
            }

}
```

## Commodity Forecast

```
package com.market.marketinfo.activities;

import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.market.marketinfo.R;
import com.market.marketinfo.customUI.CustomBuilder;
import com.market.marketinfo.database.DatabaseHelper;

import java.util.Collections;
import java.util.Comparator;
import java.util.Vector;

public class CommodityForeCast extends BaseActivity{
    private LinearLayout llFilterLayout;
    private Vector<String> vecCommodities = new Vector<>();
```

```
    private String commodity  = "";
    private Vector<String> vecStates = new Vector<>();
    private String state  = "";
    private Vector<String> vecDistricts = new Vector<>();
    private String district  = "";
    private Vector<String> vecMarkets = new Vector<>();
    private String market  = "";
    private DatabaseHelper mDbManager = null;
    private TextView tvSelectCommodity;
    private String URL_TOLAD=
"http://172.20.10.3:8081/jagadish/index.php";
    private RecyclerView rvFilterList;
    private WebView wvCustom;

    @Override
    public void initializeView() {
        llFilterLayout = (LinearLayout)
inflater.inflate(R.layout.main_filter_layout, null);
        llBase.addView(llFilterLayout);
        tvSelectCommodity =
(TextView)llFilterLayout.findViewById(R.id.tvSelectCommodit
y);
        rvFilterList = (RecyclerView)
llFilterLayout.findViewById(R.id.rvFilterList);
        wvCustom =
(WebView)llFilterLayout.findViewById(R.id.wvCustom);
        wvCustom.setWebViewClient(new MyBrowser());

wvCustom.getSettings().setLoadsImagesAutomatically(true);
        wvCustom.getSettings().setJavaScriptEnabled(true);

wvCustom.getSettings().setDisplayZoomControls(true);
        wvCustom.getSettings().setUseWideViewPort(true);
        wvCustom.setInitialScale(50);
        wvCustom.setVisibility(View.VISIBLE);
        rvFilterList.setVisibility(View.GONE);

        mDbManager = new
DatabaseHelper(CommodityForeCast.this);
        tvSelectCommodity.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loadFilterData(0,commodity);
            }
        });
        loadFilterData(0,commodity);
```

```
    }


    public void loadFilterData(final int type, final String
condition){
        showLoader("Loading ...");
        new Thread(new Runnable() {
            @Override
            public void run() {
                switch (type){
                    case 0:
                        vecCommodities =
mDbManager.getFilterData(0,"");
                        break;
                    case 1:
                        vecStates =
mDbManager.getFilterData(1,condition);
                        break;
                    case 2:
                        vecDistricts =
mDbManager.getFilterData(2,condition);
                        break;
                    case 3:
                        vecMarkets =
mDbManager.getFilterData(3,condition);
                        break;

                }
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        hideLoader();
                        switch (type){
                            case 0 :
                                if(vecCommodities!=null &&
vecCommodities.size()>0){

if(!vecCommodities.contains("Wheat")){

vecCommodities.add("Wheat");

Collections.sort(vecCommodities,new
IgnoreCaseComparator());
                                    }

showFilter(vecCommodities,0,commodity);
                                }else{
```

```
showCustomDialog(CommodityForeCast.this,"Alert","No Data
Available","Ok","","finish");
                                    }
                                    break;
                                case 1 :

if(commodity!=null&&commodity.equalsIgnoreCase("Wheat")){

if(vecStates!=null&&!vecStates.contains("Uttar Pradesh")){

vecStates.add("Uttar Pradesh");
                                        }

Collections.sort(vecStates,new IgnoreCaseComparator());
                                    }
                                    if(vecStates!=null &&
vecStates.size()>0){

showFilter(vecStates,1,state);
                                    }else{

showCustomDialog(CommodityForeCast.this,"Alert","No Data
Available","Ok","","finish");
                                    }
                                    break;
                                case 2 :

if(state!=null&&state.equalsIgnoreCase("Uttar Pradesh")){

if(vecDistricts!=null&&!vecDistricts.contains("Agra")){

vecDistricts.add("Agra");
                                        }

Collections.sort(vecDistricts,new IgnoreCaseComparator());
                                    }
                                    if(vecDistricts!=null &&
vecDistricts.size()>0){

showFilter(vecDistricts,2,district);
                                    }else{

showCustomDialog(CommodityForeCast.this,"Alert","No Data
Available","Ok","","finish");
                                    }
                                    break;
```

```
                                case 3 :

if(district!=null&&district.equalsIgnoreCase("Agra")){

if(vecMarkets!=null&&!vecMarkets.contains("Agra")){

vecMarkets.add("Agra");
                                        }

Collections.sort(vecMarkets,new IgnoreCaseComparator());
                                        }
                                        if(vecMarkets!=null &&
vecMarkets.size()>0){

showFilter(vecMarkets,3,market);
                                        }else{

showCustomDialog(CommodityForeCast.this,"Alert","No Data
Available","Ok","","finish");
                                        }
                                        break;
                                }
                            }
                        });
                    }
                }).start();


    }

    public void showFilter(Vector<String> list, final int
type, String filteredData){
        String title = "";
        switch (type){
            case 0:
                title = "Select Commodity";
                break;
            case 1:
                title = "Select State";
                break;
            case 2:
                title = "Select District";
                break;
            case 3:
                title = "Select Market";
                break;
        }
```

```
        if (list != null && list.size() > 0) {
            CustomBuilder customBuilder = new
CustomBuilder(CommodityForeCast.this, title, true);
            customBuilder.setSingleChoiceItems(list,
filteredData, new CustomBuilder.OnClickListener() {
                @Override
                public void onClick(CustomBuilder builder,
Object selectedObject) {
                    builder.dismiss();
                    switch (type){
                        case 0:
                            commodity = (String)
selectedObject;

tvSelectCommodity.setText(commodity);
                            loadFilterData(1,commodity);
                            break;
                        case 1:
                            state = (String)
selectedObject;
                            loadFilterData(2,state);
                            break;
                        case 2:
                            district = (String)
selectedObject;
                            loadFilterData(3,district);
                            break;
                        case 3:
                            market = (String)
selectedObject;

if(market.equalsIgnoreCase("Agra")||district.equalsIgnoreCa
se("Agra")){

wvCustom.loadUrl(URL_TOLAD);
                            }
                            break;

                    }
                }
            });
            customBuilder.show();
        } else {
            showCustomDialog(CommodityForeCast.this,
"Alert", "No Data Available.", "Ok", "", "finish");
        }
```

```
    }

    @Override
    public void loadData() {

    }

    private class MyBrowser extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView
view, String url) {
            view.loadUrl(url);
            return true;
        }

        @Override
        public void onPageFinished(WebView view, String
url) {
            super.onPageFinished(view, url);
        }
    }

    class IgnoreCaseComparator implements
Comparator<String> {
        public int compare(String strA, String strB) {
            return strA.compareToIgnoreCase(strB);
        }
    }
}
```