2017

A Mobile App Development Framework For Database Connectivity And High-Level Services

Venkatadri Raparla
University of Alabama at Birmingham

A MOBILE APP DEVELOPMENT FRAMEWORK FOR DATABASE CONNECTIVITY AND HIGH-LEVEL SERVICES

by

VENKATADRI RAPARLA

LEON JOLOLIAN, COMMITTEE CHAIR
MURAT M. TANIK
KARTHIKEYAN LINGASUBRAMANIAN

A THESIS

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

BIRMINGHAM, ALABAMA

2017

A MOBILE APP DEVELOPMENT FRAMEWORK FOR DATABASE CONNECTIVITY AND HIGH-LEVEL SERVICES

VENKATADRI RAPARLA

ELECTRICAL AND COMPUTER ENGINEERING

ABSTRACT

Frameworks have been successfully implemented in order to simplify, in part, the software development process. There are many frameworks developed for android platforms, normally targeted towards specific tasks such as login, database connectivity etc.

While various frameworks provide developers with the ability to include services into the app during development, the difficulty of doing so increases as the number of frameworks grows larger. Each framework provides one or more services, however integrating those services into the app requires intimate knowledge of all the details present in the framework. When developers include services from more than one framework, the inherent difficulty in dealing with each of the services and the possible interaction between services creates a difficult barrier for the developer to overcome.

In the proposed framework, a number of services are integrated within a service layer that abstracts the underlined frameworks preventing these services from being visible to the developer. Instead, the developer sees a single interface from which services can be provisioned in a systematic way. All the details regarding the access of a service from within a framework is hidden from the developer.

While this approach of hiding the frameworks may limit the overall configurability of the services, it does provide a limited version of the services that is easy for the developer to integrate into the app. It is obvious that the developer is presented with two choices when integrating services into an app. On one hand, the developer can opt to use the services directly from the frameworks, however this will require knowledge of the framework and the willingness to deal with the corresponding complexities. In return, the developer is able to have the full range in configuring the service as it is offered by the framework. On the other hand, the developer can use the abstract layer that encapsulates the various frameworks without having much knowledge about the individual frameworks. This simplifies the task of the developer in integrating these services, however some portion of the configuration for each of the services will be outside the reach of the developer.

The two alternatives can be viable in certain situations. For instance, if the standard service configuration is adequate for the problem attained then the approach that uses the abstraction layer may be preferable since it will be easier to use. However, if the developer wants to configure the services in a very specific way, they will have to use the frameworks directly to achieve their goals.

Keywords: Framework, Mobile App, Database Connectivity, Services

DEDICATION

I dedicate this thesis to my family, professors and friends who has provided support throughout my life and always had faith in my success.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

APPENDIX

LIST OF TABLES

LIST OF FIGURES

Figure *Page*

LIST OF ABBREVIATIONS

DBMS	Database Management Systems
ODBC	Open Database Connectivity
JDBC	Java Database Connectivity
API	Application Programming Interface
SDK	Software Development Kit
NPM	Node Package Manager
HTML	Hyper Text Markup Language
IP	Internet Protocol
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
OS	Operating System
ADT	Android Development Tools
AVD	Android Virtual Device
XML	eXtensible Markup Language
PDO	PHP Data Objects
AWS	Amazon Web Services

CHAPTER 1

INTRODUCTION

This chapter introduces the problem that is the subject of this research and discusses the viability of the proposed solution within the context of existing frameworks. We will provide the motivation for this research and the general outline of the solution.

The thesis document is divided into six chapters as follows: Chapter 1 provides the introduction and the context of the research carried out in this thesis. Chapter 2 provides a comprehensive survey of the available frameworks in the market that have direct impact on my work. Chapter 3 describes the overall system architecture for the proposed solution. Chapter 4 discusses the various platforms and technologies used in developing the proposed framework. In chapter 5 we present the results of the case studies that where examined in this research. In chapter 6 we provide an evaluation of the work and results generated by this thesis. We also provide some suggestions for future work that can further built on the research carried out in this thesis.

Section 1.1 gives a schema of database importance while developing mobile apps. It also provides information on division of frameworks based on requirements. The following section 1.2 explains which databases are in built for most using frameworks today. And, it describes the limitations towards frameworks over connectivity with different databases. Moreover, it illustrates how the proposing framework rectifies the complication facing by

many frameworks related to database connectivity. In section 1.3 i.e. in document structure, the different topics that are going to be discussed in further chapters are depicted.

1.1 Problem Definition

The problem addressed in this research is related to the development of a framework to facilitate the process of creating user services within an app that require database connectivity. The framework allows developers to create mobile apps with database connectivity in an automated way using high-level abstractions. By encapsulating the database details regarding database connectivity and database access, developers can use the proposed framework with limited technical knowledge about databases and the tasks related to connecting the app to the server to include database related services.

In this research, we introduced a novel approach for implementing a framework for mobile app development based on three-tier architecture. The first tier represents the mobile app where access to user services is initiated. The second tier represents the server which receives requests from the client app and acts on executing the services based on database operations. The third tier is where the data is stored within the remote databases.

The proposed framework aims to provide a template in which the connection pattern remains the same for all interactions with databases using ODBC connectivity. This approach relieves the developer from requiring having knowledge about various databases and all the intricacies involved in managing the databases in run time.

1.2 Motivation

The motivation for this research is based on the desire to reduce the level of complexity for using frameworks that involve database connectivity. In general, app development frameworks have been successfully implemented to simplify, in part, the software development process for developers. Most mobile frameworks have their in-built database usage, however, each has its own interacting pattern. Many mobile apps in use today make use of databases to implement the user level services as part of the services provided by the app. Some apps requiring access to large databases will have the database management systems installed on a remote sever. Existing frameworks such as Xamarin, Sencha Touch, and Ionic provide a plug-in for managing ODBC connection pools for accessing the data on external databases.

1.3 Importance of DB Connectivity Within Apps

The stipulation of mobile app development in the present rising market arouse multitudinous pristine frameworks. These frameworks favoring the development by reducing loads of repetitive code and providing support for building of something by expanding the structure into usefulness. There are two discrete types of frameworks that will normally serve for accomplishment of the task. The first is theoretical framework, is a study based on an existing theory or theories. The second is conceptual framework, on the other hand, is something you can develop yourself based on this theory. Numerous frameworks provided pavement for developing related apps in a flexible mode. While providing frameworks for app development it is much important of providing feasibility with working databases. As workability with database is crucial in most of apps that are

delivering today, the sight of proposing framework is for database connectivity and intended services through it.

1.4 Limitations of Current Database Connectivity for Frameworks

There are few selective frameworks provided contribution in performing database connectivity. Basic4Android is one those selective frameworks which holds providence to connect with different database apart from SQLITE[1] [1]. Connectivity can be achieved by creating the helper classes for the respective databases. But the vicinity of this framework is limited because with this cardinal level of app development can be performed. Framework7 provides the connectivity with SQL SERVER[2] [2] by default. It is an entity framework suitably for developing the apps using C#[3] i.e. on Visual Studio. Making a connection with other databases hardly possible using this framework. Spring for android is a part of services offered by Spring Framework that is solely for android mobile app development. Although possessing SQLITE as in built one it needs external driver to connect with other databases. Play and Fat free frameworks are more flexible to work on compared to the above frameworks.

Play framework[4] as a part of services providing plugin for managing ODBC[5] [3] connection pools. So, it makes a flexible way for interacting with any database easily. Fat

[1] RDBMS in a C library with no client-server database engine

[2] RDBMS database server developed by Microsoft with storing and retrieving data functionality

[3] Multi-paradigm programming language developed by Microsoft with distinct programming disciplines

[4] Open source web application framework developed in Scala which follows MVC architectural pattern

[5] Application programming interface (API) for Php to access database by the client

free framework[6] seems to be same as play framework coming to the functionality. But in general, its handling is so easy compared to play framework. As its name, it handles very light code and so very easy to manage. Even all these frameworks offering connection with different databases through ODBC or in any other way flexibility of connection with databases comes into the matter over here.

The proposed framework aims to fulfill few things that lack in the frameworks mentioned above. Even though few frameworks provide portability to connect with different databases, the methodology made them complicated. To make the process simply, an android package which provides accessibility to connect any database can be implemented with this framework. Generally, the connection to database can be done in two ways in which the first way is through REST services and the second way through ODBC. Within the android package there will be helper classes and their implementation methods of different databases which is extended by DB Helper class. This will make the developers to perform their jobs easy.

The developers need accessibility to databases while providing some services to the user. Developer will use the framework if he finds feasible with working it. Login is the most required service in most of the apps. So, checking workability with these types of services will keep a good impact rather than other type of services. Creating a package with the help of ODBC will be in the first level and services with the help of package will be in the second level of two tier architecture.

[6] Open source web framework developed by F3 factory with light weight code with readability and extensibility

CHAPTER 2

A SURVEY OF DIFFERENT FRAMEWORKS

The overall idea of this chapter is to provide information about the research of different frameworks in their domain and inside details of constituents required for achieving the proposing thesis. Section 2.1 gives the information on different essential elements of a framework. Section 2.2 discusses widely used present frameworks in the industry with the frameworks Ionic Framework, Basic4Android, Framework7, Spring for Android, RoboGuice, and Robolectric in sub-sections 2.2.1 through 2.2.6 respectively. By considering the advantages and limitations of the above frameworks, a direct comparison with the proposed framework is given in section 2.3.

2.1 Essential Elements of a Framework

A framework is a real or conceptual structure intended to serve as a support or guide for building of something that expands the structure into something useful. Frameworks are categorized into distinct types even in computer terminology based on the purpose. Some of them are enterprise architecture framework, software framework, web application framework, framework oriented design, etc. Most often, a framework is a layered architecture in which it indicates what kind of programs can be developed.

Android is one of the popular platforms for mobile app development. Consequently, the development of android mobile apps eventually got prominence. So, we are getting many android development IDE's[1] for developing mobile apps. Meanwhile we are getting mobile frameworks for making the development of apps easily. Some of the frameworks which are sophisticating and supporting the development of android mobile apps are Basic4Android[2], Framework7, and spring for Android etc. The background and services offered by the above-mentioned frameworks are explained in the following paragraphs.

2.2 Widely Used Frameworks

2.2.1 Ionic Framework

Ionic Framework [4] was developed by Drifty Co. in 2013 offers a complete open-source SDK. It is developed to provide hybrid mobile app development/cross platform development. It is constructed on the top of Angular JS and Apache Cordova equip tools and services for app development with technologies like HTML5, and CSS. Ionic framework is written in JavaScript, and the developers used JavaScript (now using TypeScript) for developing the apps require NodeJS, Node Package Manager (NPM) and Cordova for deployment.

The app developed with Ionic framework need to build a RESTful API [5] for talking to backend service. But the Ionic framework is more sophisticated for front end and the developer should take care of everything on back end. Even though this framework

[1] Software application providing wide range of facilities to the programmers for software development

[2] Application development tool developed by anywhere software ltd. with programming language like Visual Basic

provided varied customizable ways for connecting to database, it is strenuous to connect through server which happens mostly in business purpose apps.

2.2.2 Basic4Android

Basic4Android (B4A) was developed by Anywhere Software Ltd. It is a rapid application tool for native android applications which is an alternative to programming Java and Android SDK [6]. It provides a visual designer which simplifies the process of building user interfaces targeting phones and tablets with varied sizes. Generally, Java is a flexible language to make on android apps but this provides a platform with supporting language near to visual basic which makes the app development even simple. In addition to the elimination of the usage of Eclipse, there are no extra runtimes and dependencies. Testing can be done normally as in AVD manager. It supports all types of applications such as games, databases, connectivity, sensors and hardware.

2.2.3 Framework7

Framework7[3] is a HTML framework for building IOS and android apps. It is one of a set of open source libraries and solutions from iDangero.us. It does not rely on any particular framework rather than DOM manipulation framework (DOM), which is like a mobile optimized version of jQuery. The main motto of Framework7 is to give an opportunity to create android and IOS apps with HTML, CSS and JAVASCRIPT easily. With Framework7 it is easy to customize styles to your app as all the styles are divided by parts into small. less files. It also provides the feature of native scrolling with quick reactions without scrolling issues. Furthermore, there is no need to learn anything new to

[3] HTML framework developed by iDangero.us for building android and ios apps

code using framework 7 because it uses JavaScript API methods to make changes within the app.

2.2.4 Spring for Android

Android related extension for Spring Framework[4] [7] is Spring for Android. Its primitive purpose is to simplify the development of native android applications. The main feature offered by it is we can build an Android client that consumes a spring based RESTful web service. It offers Auth support for accessing secure APIs [8]. The Spring for Android provides RestTemplate module to work with android applications. To access data from external web services we need to authorize and authenticate mobile application through OAuth protocol. Moreover, it had provided helper classes to connect with SQLite database along with third party authentication with Facebook.

2.2.5 RoboGuice

RoboGuice is a Google Guice on Android. It is a dependency injection framework which makes android development more intuitive and convenient. With this framework, you can reduce copious amounts of code that you write for performing common tasks such as resource allocation, accessing android system services and event handling. RoboActivity is the class to be extended for dependency injection while handling this framework. The principal aim of dependency injection is to separate dependency resolution behavior. This makes it possible to work away from hard coded dependencies, so that we can make changes at any possible time of execution. It makes it easier to work with code as well as

[4] Application framework developed by Pivotal Software which provides extensions for building web applications on top of java EE platform

testing which in turn promote more business logic. It is also flexible in advocating our own base classes along with required methods with their functionalities.

2.2.6 Robolectric

Robolectric is a unit test framework that defangs the Android SDK jar so you can test drive the development of your Android app. It allows to run Android tests inside the JVM on your workstation. It assists layout views, fragments and lot of other stuff that are implemented in Native C. So, Android applications can be tested on the JVM in seconds without an emulator or device. There is an alternative for Robolectric, which is nothing but advanced Mock frameworks, which works similar to this framework.

The above discussed Frameworks may not have solved the complete problem issue. They had resolved some issues depending on their problem statements. Whenever there is a complexity and loads of code needed in varied activities we can make it generic and it can be used anywhere with a single line of code.

2.3 Typical Services Provided by Frameworks

Service is functionality or set of functionalities offered for different purposes to the clients and can be reused any number of times. Frameworks intended to offer a service or number of services to implement them easily when a developer is working to implement such type of service in the project. The frameworks discussed in the above frameworks offer their own intended services which are based on database connectivity and some general services as well.

Ionic framework provides different components for working with numerous services. It provides server component for building the applications or services related to database connectivity. Along with server component, it gives Restful API to work on top of data model. This framework offers a great service for geolocation, if the developer need to use that service in development. Xamarin [9] also provides a platform to work with different sets of services related to database connectivity. It provides database connectivity using a SqlClient package and the required services is extended using an open connection string.

Sencha Touch also provides an API to make database connectivity. While sending connectivity request, the request is send in AJAX followed by the server-side code. PhoneGap is another framework which provides database API for managing database connectivity. This framework provides a rating service in which the developer can implement this service easily by using the framework in the project. The Spring framework is intended to work on android apps only. It also provides database connectivity through JDBC where the access to database is maintained through native-API driver. Spring provides login-service, this functionality is easily implemented by inhibiting this framework into the project.

2.4 Advantages of Current Frameworks Models

Current existing frameworks providing an impressive set of services for developing sophisticated mobile apps. There would be some specific advantages to use those frameworks for development. The advantages can be service oriented or the easy utilization of platform.

Ionic framework provides the advantage that one source supports two different platforms (Android and iOS). The development toolkit of Ionic includes HTML, CSS and JavaScript which makes easy for most of the developers. Moreover, it is embedded with another framework angular makes it easy to develop browser based mobile apps. Xamarin is intended to offer feature of developing cross platform apps. It is a native app development framework with shared app logic for varying operating systems. It provides location oriented service which can be promoted easily into the mobile app. Sencha Touch is mobile web based framework where we can develop mobile apps for all platforms. In Sencha Touch, user interface is developed with the help of HTML, CSS and JavaScript components. The apps developed using this framework are speed with customizable user interface [10], easy data management. PhoneGap is an open source framework intended to provide lane for cross platform apps. It provides an advantage of using graphical data services into app with easy implementation. The existing frameworks providing several advantages such as cross platform apps, open source, unique service oriented, and flexibility.

2.5 Limitations of Current Frameworks Services

Limitations exist for most of the systems because all the frameworks are not perfect for implementing all the features needed. Current existing frameworks have some limitations in facilitating the services to be utilized by the developers. Each framework suffers some limitations while offering the platform for app development.

Ionic framework offers only geolocation service related to database connectivity. If needed to use any other service, the developer should use that specific service providing framework. And its performance is less because of semi native feature. When the app loads, lot of resources must load for proper implementation, in-turn takes more loading time. Xamarin provides a unique service of rating depending on database connectivity. And integrating of other framework into this for implementing any other service is difficult. Spring framework is providing login service for the developers to make it easy to implement in an application. But this framework provides handling of this service only for only android apps. Sencha Touch is offering graphical data service to implement by developers in an application. If the developer wants to implement any other services in the same application which is not provided by this framework, the implementation of that service is difficult. In an application if a developer wants to implement different services, he needs to use different service oriented frameworks. While doing this, integrating all the frameworks into single app becomes very difficult. The app will not be efficient to use and the app speed may decrease. By considering some of these limitations regarding services, implemented a new model of providing multiple services in a single framework explained in the next chapter.

Table 1. Advantages and limitations of different frameworks

FRAMEWORK	ADVANTAGE	LIMITATION
Ionic Framework	Toolkit includes HTML, CSS and JavaScript. It also embeds with angular to develop browser based apps easily	Semi native framework with enormous size ng modules. Only offers Geolocation services related to connectivity
Xamarin Studio	Provides platform for cross platform apps with native in nature. Same shared logic different app platforms	App loading time is more. Offers only rating service related to database connectivity
Sencha Touch	Apps runs fast with customizable user interface and easy data management	Offers only graphical data service and difficult to integrate different framework
PhoneGap	Open source framework portable to all platforms. Flexible to use	No provision to interact with server environment to establish connection with database
Spring Framework	Provides platform for android app development. Easy to implement and provides both front end and backend service	Provides connection to database through JDBC. Provides sign in service related to connectivity

2.6 Summary

Frameworks simplifies the hardness needed in implementing the software development process. There will be some advantages and disadvantages for most of the frameworks to use. In the above sections, we discussed few advantages and limitations of current existing mobile frameworks. The main advantages and limitations of those discussed frameworks is explained in the above tabular form.

CHAPTER 3

PROPOSED SYSTEM AND ARCHITECTURE

While various frameworks provide developers with the ability to include services into the app during development, the difficulty of doing so increases as the number of frameworks grows larger. Each framework provides one or more services, however integrating those services into the app requires intimate knowledge of all the details present in the framework. When developers include services from more than one framework, the inherent difficulty in dealing with each of the services and the possible interaction between services creates a difficult barrier for the developer to overcome.

In the proposed framework, a number of services are integrated within a service layer that abstracts the underlined frameworks preventing these services from being visible to the developer. Instead, the developer sees a single interface from which services can be provisioned in a systematic way. All the details regarding the access of a service from within a framework is hidden from the developer.

While this approach of hiding the frameworks may limit the overall configurability of the services, it does provide a limited version of the services that is easy for the developer to integrate into the app. It is obvious that the developer is presented with two choices when integrating services into an app. On one hand, the developer can opt to use the services directly from the frameworks, however this will require knowledge of the framework and the willingness to deal with the corresponding complexities. In return, the developer is able

to have the full range in configuring the service as it is offered by the framework. On the other hand, the developer can use the abstract layer that encapsulates the various frameworks without having much knowledge about the individual frameworks. This simplifies the task of the developer in integrating these services, however some portion of the configuration for each of the services will be outside the reach of the developer.

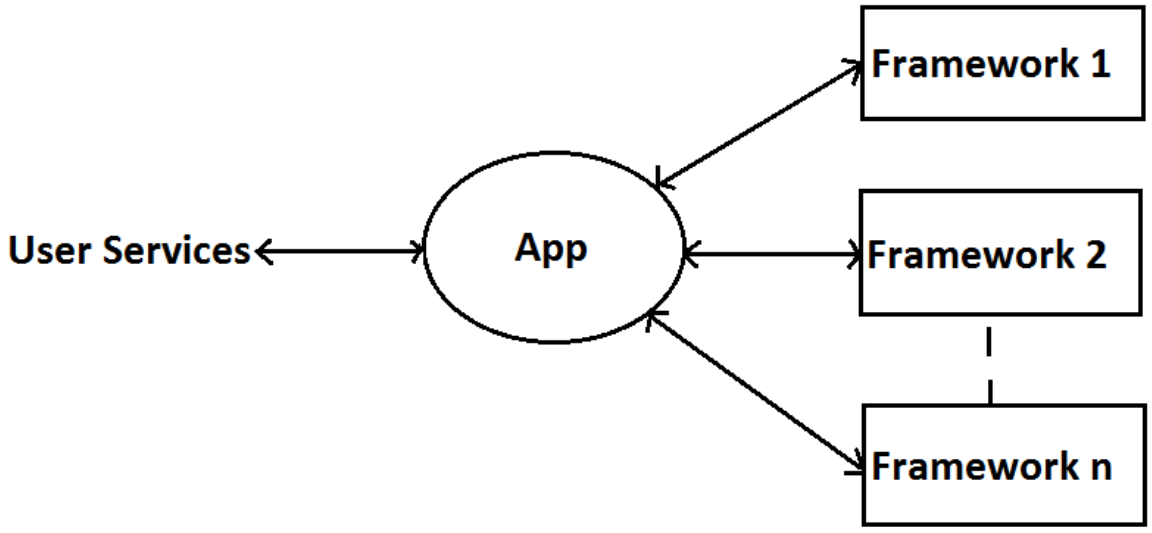

Figure 1. Existing approach

The two alternatives can be viable in certain situations. For instance, if the standard service configuration is adequate for the problem attained then the approach that uses the abstraction layer may be preferable since it will be easier to use. However, if the developer wants to configure the services in a very specific way, they will have to use the frameworks directly to achieve their goals.

Chapter 3 provides information on the proposed system with their central details extended by inmost explanation of architecture for proposed system. Section 3.1 illustrates the proposed system architecture i.e. three-tier architecture. Within this architecture there are different components which contribute to the achievement of successful research. Next

section 3.2 explains the components related to first tier i.e. app tier of three-tier architecture. The components include mobile service API, set of some services, and android SDK[1]. Sub sections of 3.2 explains the components of first layer in detail. The section 3.3 explains all

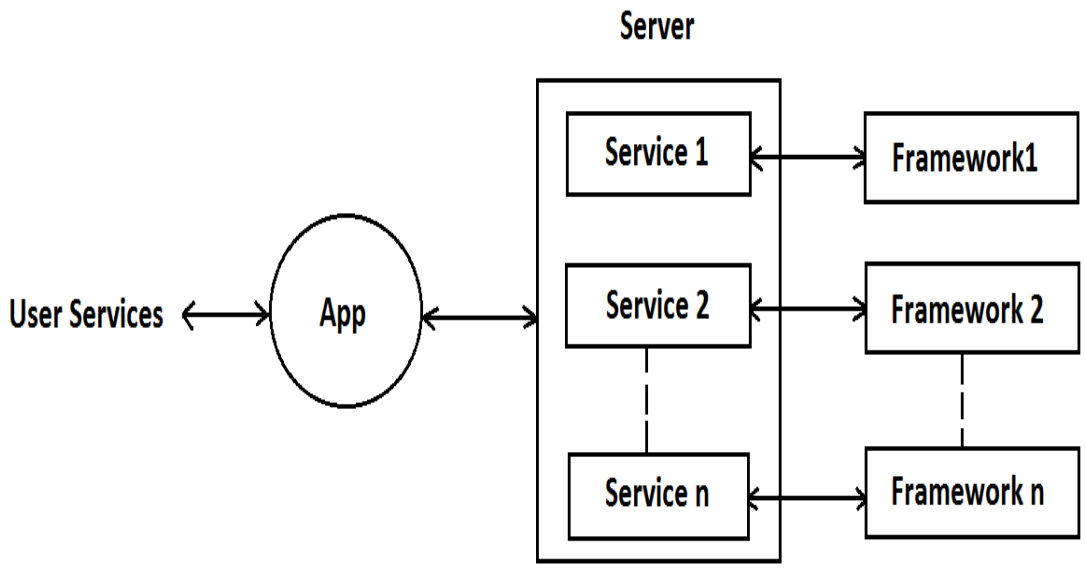

Figure 2. Proposed approach

services offered by second tier i.e. server tier with subsequent implementation of sign in, logging, analytics, and user rating services. In the sub sections of 3.3, functionalities of every component of server layer are well explained. In the next section 3.4, the details of data tier are explained with their internal component details in sub section. Furthermore, this chapter depicts the presumed operation of the overall system that should be achieved as the result.

[1] set of android development tools which includes debugger, emulator, tutorials etc.

3.1 Three-Tier Architecture

The thesis statement is offering an Android Framework which provides workability in three tiers. Figure 3 shows the architecture of mobile apps operating in run time configuration. The overall system architecture with individual details are explained in the following paragraphs.

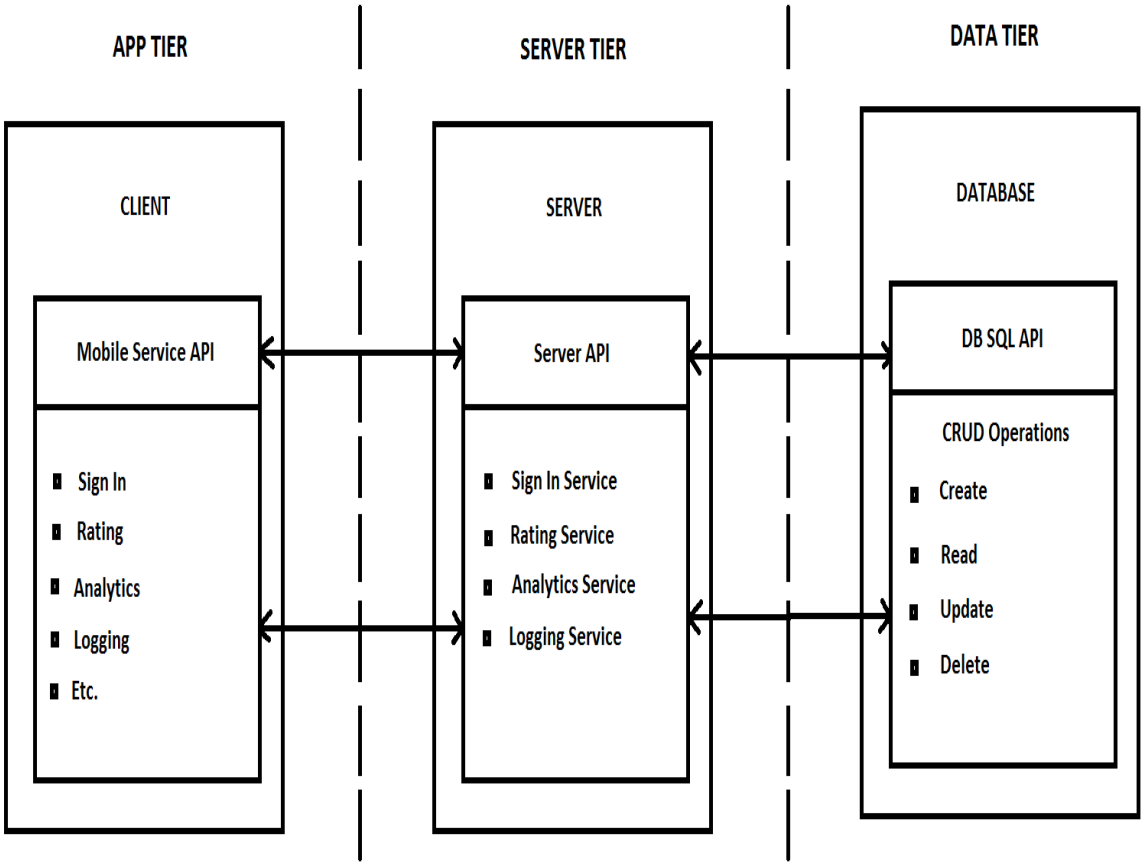

Figure 3. Three-tier architecture

The first tier is called app tier because it deals with all the functionalities that should happen on client side. App tier is using the functionality from mobile service API with some set of useful services like sign in, rating, analytics and logging. This tier transparently uses the second tier to implement those services. The second tier is called server tier which

basically uses the functionality from server API for implementing the services such as sign in service, rating service, analytics service, and logging service as requested from client side. This server tier makes use of next tier for performing CRUD operations such as create, read, update and delete. The third tier is called data tier which involves in CRUD operations for different databases. The CRUD operations include create, read, update and delete queries to perform on databases. The server tier of the framework interacts with data tier for encapsulating the database access and interaction with database.

The services that are implemented in the server tier are provided by different frameworks in general. Moreover, if we want to implement those services all the providing frameworks are integrated to perform the task. But if the developer uses this framework, the set of those services can be implemented at once. And the general difficulty faced in app tier is shifted to server tier and the framework takes care of that complexity in implementation.

3.2 Components of App Tier

The app tier of three-tier architecture provides the platform for mobile app development where the user can send request for data, basically called client. There are different components helpful in making up the client process. Mobile service API is one of those provides interface for requesting of different services from the client. For the development of mobile app, the major component required in app tier is Android SDK which is explained in the sub section of app components.

3.2.1 Android SDK

Android Software Development is for developing new mobile applications for Android OS. There are several development environments available for mobile app development but this helps in developing with Java language by using Android Software Development Kit(SDK).

Android SDK encompasses different development tools such as emulator, debugger, libraries etc. Even though Android Studio is an official IDE for android mobile app development, developers can also use Eclipse with Android development tools. Moreover, you can use any sort of text editors for coding or making changes in Java and Xml files. And, more advanced Android SDK's are available nowadays by enhancing the properties of older Android SDK. The major tools available in android SDK are Android debug bridge and Fastboot.

3.3 Components of Server Tier

The server tier of three-tier architecture deals with different tasks at same time. First the server will receive several requests from client whereas in existing frameworks different service requests are obtained from different frameworks which is providing a pure set of that service. After getting the requests from the client, the server implementation of those services will take place. At the same time, it provides interaction with database for CRUD operations through database connectivity. The different components of this tier are helpful in making up the complete functionality. The components are ODBC, server, services which include sign in service, user rating service, analytics service, and logging service explained in the next sub sections.

3.3.1 ODBC

Open Database Connectivity is a standard application programming interface (API) for accessing database management systems (DBMS) which aimed to make it independent of operating systems and database systems. It is the generic API provided by Microsoft and Simba technologies for interacting with any database to perform various database operations. It provides methods to access any kind of tabular data, especially data stored in a relational database. It works on a variety of platforms such as Windows, Mac OS and UNIX. Using ODBC statements in a program, we can access files from different databases which includes dBase, Access, Excel, DB2, and Text. The most important advantage is an application written using ODBC both in server side and client side can be used in different platforms with some modified data access code.

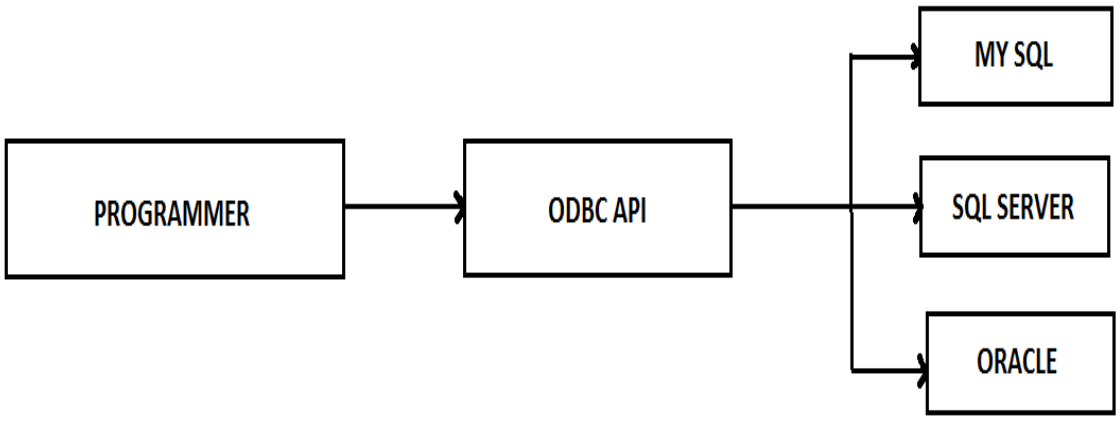

Figure 4. ODBC

3.3.1.1 Drivers

Drivers are always used to establish a bridge between the application and database. ODBC accomplishes access to DBMS using an ODBC driver which acts as a translation bridge between the application and the DBMS. By writing the ODBC statements through

an ODBC driver manager in the application, it helps in querying the database. It is the specification given by Microsoft whose implementations are done by various database vendors and they developed driver implementation classes. It also abstracts the details associated with working with driver objects. Moreover, a separate module/driver is needed for every database to be accessed. Generally, all major databases consist their own drivers including other sources like MS-Excel, address book systems and CSV files.

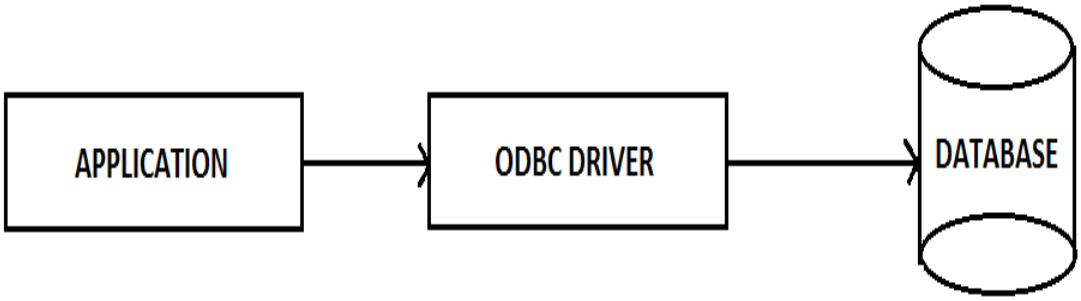

Figure 5. ODBC driver

The ODBC library includes APIs for the tasks such as making a connection to a database, creating SQL or MYSQL query statements, executing SQL or MYSQL query statements in the database, and viewing and modifying results.

3.3.2 Server

A server is nothing but a computer program provides its services to other computer programs which is in the same computer or different computer/system. It is designed to process the requests and safely deliver the data to clients through a network or internet. In abstract, the computer running server program is technically referred as a server. A server is nothing but a computer program provides functionality to another computer program

called “client”. It formulates a client-server model architecture in which a single computation is distributed across multiple processes.

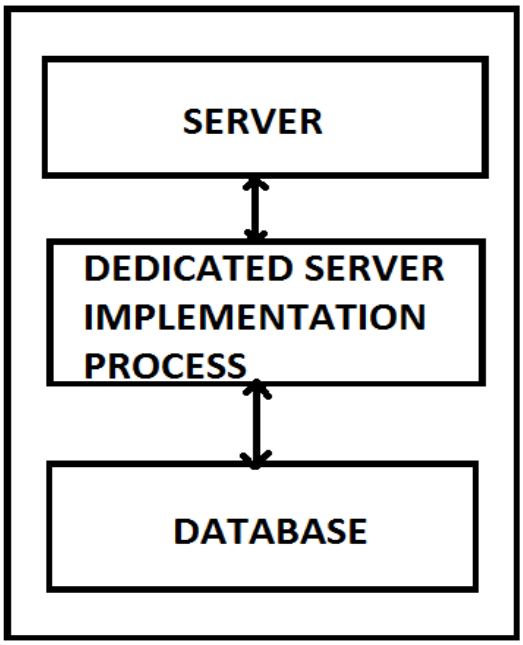

Figure 6. Server

Functionalities of servers is often referred as services offers performing computations for different clients, resource/data sharing among mutinous clients. Generally, servers are categorized based on their purpose namely web server, proxy server, virtual server, and file server etc. A server can communicate with multiple clients and a client can send request to multiple servers. And, this process of communication can occur on same device or various devices connected over a network.

3.3.3 Services

Frameworks simplifies the hardness needed for implementing dedicated services offered by them. While offering services, they should deal with database connectivity for

interacting with server databases. The proposed framework offers services such as sign in, rating, analytics, and logging with easy implementation.

3.3.3.1 Sign In

Sign In is the process of identifying and authenticating their individual gain of access to a computer system. While creating an account, users provides personal information and intermittently provides secured information as well. For signing into an app user should provide credentials such as username, email, phone number, and password. Hence the security layer within these credentials must be cutting-edge and complex to understand for others.

Developers should make a validation of credentials while developing the app that should be very difficult to hack by others for the users setting their account. All the computers use cookies for the websites to track the sessions. The developer and user must be attentive whether use data, user information will usually be deleted from user's computer through cookies. It provides secure entrusted access to the working server providing the user details. And, it provides access to the required server by enclosing the user credentials. As this module is using in most of the mobile apps developing today, providing this service as a part of framework will make the developer job easy.

3.3.3.2 User Rating

User rating gives valuable evaluation of an object or a thing. It can be calculated by finding the weighted average of all the ratings. Getting user reviews is a key requirement in most of the online shopping applications which allows user to post feedback relating to any shopping item or restaurant. The ratings for an app can be trusted in the case that rating is done through market app.

Rating the app through the app page in play store/app store is the best way to trust on the review. If the rating handled within the app there is a possibility that developer can manipulate user rating at any time. The best way to do this is by getting installed packages from package manager class and should check market class is installed. The best way we provide the user this feature, the more it is feasible and robust for the user.

3.3.3.3 Logging

Logging is the process of writing the different information into the console or text document or into iPlanet. Apache log4j is a java based logging utility from Apache software foundation. There are many java logging frameworks and apache log4j is famous of those. Logging is the process of storing events in a file with certain scope to understand the activity of the certain system and to diagnose problems. Generally, developers log the information related to exceptions to resolve the issue easily.

Logger accepts throwable objects with overloaded log methods and Java util logging package is the standard for Java standard edition. Logging can be a very good feature when we are developing huge application because we may come across lot of useful and useless information. Using this app data, we can make something beneficent if the app is developing for business purpose.

3.3.3.4 Analytics

Analytics is the process of discovering or interpreting understandable patterns in data in familiar known as analysis of data or data analytics. It is the process of inspecting the data sets to obtain the evaluation on specialized systems. The randomly generated data is unordered and analyzing those data is difficult. For example, the data generated through social networking sites such as Facebook, Twitter is huge. So, performing analytics on

these types of data is arduous. First, we should covert this into a useful format in an orderly manner depending on the type then performing analytics will be easy. It finds so much prominence by providing evaluation through twitter polls and an immense tool in implementing data analytics.

There are different areas where analytics is of prominent are business analytics, predictive analytics, risk analytics, market analytics, and software analytics. Developers should use different algorithms and software that harness analytics to perform these types of analytics. To perform these analytics there are several services offered by big companies as google analytics, Microsoft azure analytics, IBM Watson analytics, Apple analytics etc.

3.4 Components of Data Tier

The data tier layer of three-tier architecture provides the platform to deal with databases. Upon establishment of connection with it, this tier offers to perform CRUD operations. The main component of this tier is databases such as MySQL, SQL SERVER, ORACLE etc. depending on developer's flexibility.

3.4.1 Database

Database is the structured collection of schemas, tables, queries etc. Database is generally for storing different forms of structured data. Within the database you can store substantial amounts of data in a systematic manner. To query the data i.e. select, modify, create and delete data from database we have relational database management

systems(RDBMS[2]) like MYSQL, SQL SERVER, ORACLE, SQLITE etc. Not all the developers are proficient in above provided databases.

Flexibility of working with database depends on proficiency of the developer to deal with it. Although the way of querying the data is different, all these will do quite similar job. In this architecture, the database depends on which one we are using while developing the app. All these different databases are made into a package by using the concept of inheritance and can be used based on requirement.

The approach for making the thesis statement is justified in following paragraphs. The developer who wants to use MYSQL as a database for developing an app need to establish a connection between Android Studio and MYSQL database. Also for establishing connections between other databases and Android Studio, it is needed to perform a connection with ODBC. Instead of making an establishment of connection every-time the proposed framework intends to offer a package for connection establishment with any prescribed database.

[2] database management system based on relational model used for maintaining financial records, personal data etc.

CHAPTER 4

USED PLATFORMS AND TECHNOLOGIES

This chapter discusses different platforms and technologies used in the project to support the proposed thesis. This includes platforms used for both front-end and backend and replaceable suggestions that can brace this thesis. Present chapter explains all the technologies used in a systematic manner. Section 4.1 explains the platforms used for front end and subsection 4.1.1 explains about Android Studio. Section 4.2 explains the platforms and technologies used for backend in the project and the subsequent subsections 4.2.1, 4.2.2 and 4.2.3 explains about PHP Storm, Amazon Cloud and MySQL database. The next section 4.3 explains the other suggested platforms and technologies that can be used to proceed on the thesis.

4.1 Platforms Used in Front End

The front-end platform used in developing mobile app is Android Studio. Front end mostly helps the user for entering the data, user validations, design, and viewable objects. There are various tools that will reinforce front-end are HTML, CSS, JavaScript etc. But Android Studio provides a bridge for both front-end and backend, more priority for front-end. The main objective of front-end is to ensure the user whether it can able to read the relevant data easily and enter the information required.

4.1.1 Android Studio

Android Studio is an Integrated Development Environment (IDE) for developing android mobile apps. It was developed by Google and it is written in Java. It is compatible with different operating systems such as Windows, macOS, and Linux. Initially, Google provided Android Development Tools (ADT) for Eclipse to develop native android mobile apps but after that in 2013 Google released stable version of Android Studio.

The features offered in the initial release of Android Studio helpful in developing medium scale native mobile app with SQLite as mobile inbuilt database. They are providing new features in every update version of their release to make their IDE a stable one. The features include Gradle based build support, bug fix, special refactoring, and template based wizards. It is also providing layout editor to drag and drop the UI components and preview the layout at the same time. It provides a super feature called Android Virtual Device (AVD) offers an emulator to build and emulate the app using several types of android devices. The recent version of Android Studio is providing a built-in feature for supporting Google cloud platform and support for android wear apps.

The layout in Android Studio is described in XML than in Java as it is easy to understand and Java parsing is strenuous. The newest update version Android 3.0 is supporting a new programming language called Kotlin. In my thesis Android Studio is used for developing an app for front-end.

4.2 Platforms and Technologies Used in Backend

The platform used in backend for the android app is PHP storm. The name backend itself says it is the back bone for any mobile app. The technology used for connecting to database through server is Amazon cloud. Mostly it deals with the issues related to the data requests obtained from front end for talking to the services.

The apps that are developing now accessing the database indirectly through an external application. The mobile app developed for this thesis followed the same methodology to provide a template that brings flexibility for the developers. The subsections 4.2.1, 4.2.2,4.2.3 explains PHP Storm, Amazon cloud and MYSQL respectively.

4.2.1 PHP Storm

PHP Storm is a cross-platform IDE for PHP developed by JetBrains supports Windows, Mac OS X and Linux operating systems written in Java. This platform provides editor for PHP, HTML and JavaScript and it is widely used for commercial purpose. It is also included with a SQL editor for querying the data prominent for querying database in mobile apps.

PHP is a backend programming language on server-side used to query databases like MySQL. The other databases like SQL Server, Oracle can also be queried if we use PHP Data Objects (PDO). PHP can also be used to communicate with front-end languages like HTML and JavaScript. So, it is a powerful language which can be used most for developing mobile apps. PHP Storm is the best platform that supports app development using PHP language.

The key features PHP editor of PHP storm offers rich code editor, code completion, bug check, refactoring, and syntax highlighting. Debugging tools are powerful and PHP Unit tests can run instantly with code coverage. It supports HTML5 and provides code completion, error highlighting for JavaScript, CSS and HTML.

4.2.2 Amazon Cloud

Amazon Elastic Compute Cloud (EC2) is the cloud computing platform provided by Amazon. It is offering Amazon Web Services (AWS) which we can rent virtual computer for running our own applications. It is also delivering an instance through which a user can deploy their applications with the help of web services. Its cloud platform is called elastic because a user can create, launch and terminate server instances at any point of time and these can be utilized on hourly basis based on the usage of active server.

Amazon web Services (AWS) is cloud computing platforms provided by Amazon to individual users, public sectors, and financial companies through a paid subscription. Initially it offers a free trial version which the user has only limited access to the services offered by the company. The scope of AWS is increasing everyday by the usage of these services for storage, computing, deployment, mobile development, analytics and Internet of Things (IoT). It is having several service centers all over the world majorly in United States.

Amazon Web Services for analytics now became very prominent as it is offering features including Amazon Athena for ETL, server-less querying SQL, Amazon Elastic MapReduce (EMR) for running MapReduce queries and Amazon QuickSight for business intelligence. Amazon's cloud is now bigger than Azure, Google and IBM clouds combined.

These features braced me to use Amazon Cloud for running the server to send queries to the database in my mobile app.

4.2.3 MySQL

MySQL is an open source Relational Database Management System (RDBMS) developed by Oracle corporation. It is written in C, C++ and it is compatible with Windows, Linux, Solaris, macOS, and FreeBSD. In the name MySQL, "My" is the author's daughter name and SQL means Structured Query Language. The applications developed in PHP, Joomla, WordPress, Drupal uses MySQL database.

MySQL is offered as open source MySQL Community Server and Enterprise Server. There are several good features provided by MySQL including cross-platform support, directing cursors, stored procedures, updatable views, DDL commands, SSL support, Unicode support, query catching, and built-in replication support. MySQL is also having a very good feature that it can run on cloud computing platform especially Amazon EC2. In my project, Amazon EC2 is used for maintaining the running server which can send queries to the database. In addition, PHP is the language used for writing the server code to access the Amazon server. So, MySQL is the default database any sort of development using PHP which is also used as database for my mobile app.

4.3 Other Suggested Platforms and Technologies

This section of the chapter explains the other suggested platforms and technologies that can be used in both client side and server side. Along with these topics this section explains the differentiation between the usage of ODBC and JDBC.

Generally, in any sort of work there can be other suggestible possibilities to perform. In those possibilities, we follow the methodology that is more useful and gives satisfactory results. Sometimes there will be two options which results in similar output, then we perform the task using the method in which most people do for obtaining the required output. In the same way, there will be other suggestible technologies and platforms can be used to support the thesis.

Android Studio is used for client-side programming in the mobile app which is the best platform to develop android apps. So, there will be no other suggested platform for front-end ahead of Android Studio. But there are some subsidiary platforms that can be used instead of platforms and technologies that are used for the development on server side for mobile app. The subsections 4.3.1, 4.3.2, 4.3.3 and 4.3.4 explains NetBeans, ODBC vs JDBC, other cloud platforms, and databases.

4.3.1 *NetBeans*

NetBeans is an Integrated Development Environment (IDE) for software development. It is developed by Sun Microsystems written in Java. It supports windows, Mac OS X, Linux and Solaris operating systems. This IDE is mainly intended for Java but we can also develop apps which is based on C++ and PHP. It is one of the subsidiary for PHP Storm because it offers most of the features that PHP Storm does. And it is free to use for the enterprise edition for which we should pay more for PHP Storm.

In the app, the backend server code is written in PHP but we can use Java language code for connecting to the server. In that case we JDBC connectivity for the server to talk with the database. There are few advantages of using Java code for server and JDBC for

talking to database. But in the present market developers PHP and ODBC in the app because it simplifies the several complexity tasks that developers face using Java. Eclipse is also one of the best IDE for developing Java based applications.

4.3.2 *ODBC vs JDBC*

ODBC is a standard Microsoft Windows interface helps in the conveyance between Relational Database Management Systems (RDBMS) and the applications (Web or Mobile) written in C, C++ and PHP whereas JDBC establishes communication between database management systems and the application written in Java. Database toolbox containing C++ library natively connects to an ODBC driver likewise database toolbox containing Java library natively connects to JDBC driver.

The significance of using ODBC driver is that the performance is very fast while importing and exporting data. In addition, it is the best one for importing and exporting memory-intensive data. Similarly, JDBC offers the advantage of using any type of operating system while you are working i.e. platform independence. It also offers advantage of using complex and long data types.

4.3.3 *Other Cloud Platforms*

Amazon server is used in the application for connecting to MySQL database provided by Amazon cloud. There are several other cloud platforms providing the platform of running the server on their clouds. Google cloud is one the cloud platform providing platform and tools for developing apps on the top of it. Microsoft Azure is also a cloud platform where we can run our own server on the top of it. IBM also introduced IBM cloud for running virtual computers and for developing applications with the database is

maintained on the cloud. Along with these features, they are offering their own tools for handling analytics where we can maintain Hadoop clusters.

There are also some other cloud computing platforms where we maintain our server. DigitalOcean is one among those which provides infrastructure and tools for development of an app and maintaining the server. Along with these cloud computing platforms, there are some companies which sells virtual private servers and dedicated web hosting. Linode is also liability company providing virtual private server on monthly paid basis.

4.3.4 *Other Databases*

MySQL is the database used in the development of mobile app as MySQL is the default database for PHP language. There are other databases which we can use while developing the mobile application. Generally, mobile app developing platforms provide databases for internal storage of data namely SQLite. But these databases can store very less amounts of data mostly it stores app initialization methods and intermediate variables.

The commercial apps contain enormous amounts of data within the app. In addition, the data sending as a payload is also huge. For storing and maintaining enormous amounts of data we need a real database like MySQL, SQL SERVER and ORACLE. Even though I am using MySQL as a database in this app, the developers can use any one of those databases in a real application.

CHAPTER 5

APPLYING THE FRAMEWORK: CASE STUDY WITH RESULTS

This chapter explains the whole project with case study and results obtained with the help of detailed explanation. The case study includes the type of methodology in dealing different services using existing frameworks and using proposed framework with the whole idea of concept that supports the thesis. The next parts of this chapter will explain every result obtained with the help of the figures that reinforce the thesis statement. The section 5.1 explains the details of case study about different services. The following section 5.2 and 5.3 explains the methodologies used in dealing rating service with existing frameworks and proposed framework with supporting results achieved as a part of the project with the subsequent explanation of each result obtained with supporting flowcharts. The section 5.4 provides the explanation of trade-offs between the two approaches. The last section 5.5 provides emphasis on applying the same methodology to some of generalized services.

5.1 Details of Case Study

The justification to thesis can be well explained by developing a mobile app that includes implementation of different services. Every framework is not proficient to provide workability on all the services. Major of the frameworks today are self service oriented with providing platform for performing some other minor tasks or services. So, if want to

implement any other service apart from providing service, the best thing is to use new flexible framework that provides efficient implementation of the service.

The developers implementing the services may have different notions such as flexibility for the user, commercial purpose, and analysis. There are different services being implemented by developers these days but user level service, rating service, analytics service, and logging service being utilized in their business process implementation. These services can be implemented in diverse ways depending upon the utilization of data obtained by various companies. Similarly, various frameworks provide implementation in diverse ways for different services. The proposed framework allows developers to create mobile apps with database connectivity in an automated way, using high-level abstractions. It also provided implementation of these services in a traditional way where any developer can use with limited technical knowledge about connectivity and the services.

Xamarin studio provides API for dealing with rating services along RESTful services to deal with databases. Spring framework offers plug-ins to interact with user authentication where the developers can use Google, Facebook or Twitter for signing into the app. Sencha Touch is providing user interactive graphical data service where developers can implement this service easily with this framework. Next section explains the implementation of rating service using current existing frameworks.

5.2 Existing Approach for Provisioning Rating Service

Frameworks provides useful services for implementation in a mobile app for developers. Rating service is for evaluating the average rating given by the users to display

their experience with the app. This section explains the implementation of rating service using current existing frameworks.

The implementation of any service in an app follows some methodology. Here the rating service is implemented in different blocks. Initial block explains the process that developer follows in client side while executing the service. The flowchart bellows explains the implementation process followed by developers to deal with rating service on client side.

Initializing the environment makes the system to initialize all the variables with parameters and methods that maintain app run time configuration. After loading the environment, the developer should create an instance for loading the layout. This will load all the individual components existed in the xml file of the layout. As we are implementing the rating functionality, we need to define methods with initialized variables for rating. Within the methods defined for rating we need to provide implementation details for each method. All this implementation can be possible by using mobile service API which provides supporting methods to implement. When the user provide rating as the input to the app, it will be evaluated through all the process happened in the initial stages to provide input data to server. All the explained process will happen on client side and once the input data is sent to server, the service configuration will be completed.

After initiating the rating service from client side, the client data will reach the server in a xml or Json format. The flowchart below explains the server-side process for initiating the rating service. After obtaining input from client, we need to initialize the service environment. Initializing the service environment includes loading server configurations.

We need to configure rating service parameters as well after the service initialization. It will load all the parameter inputs received from client data.

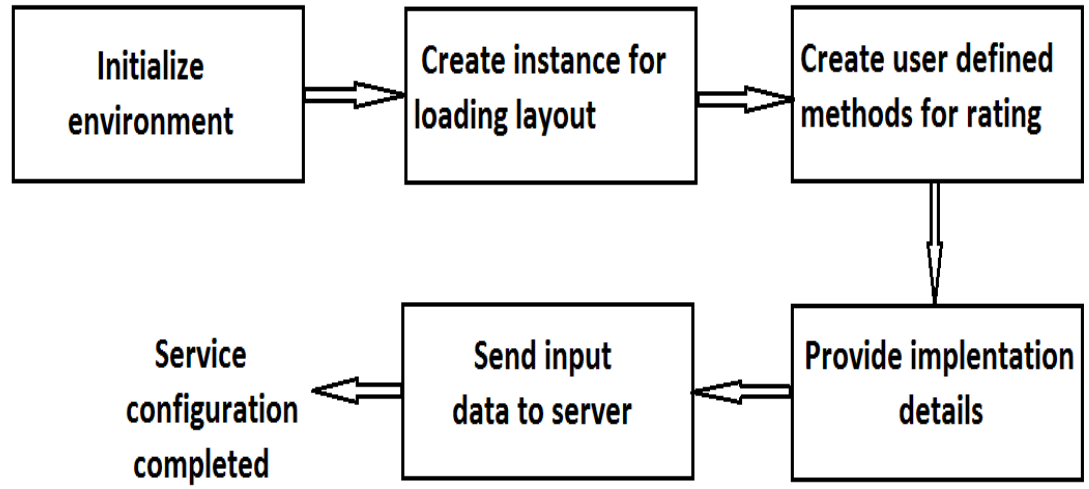

Figure 7. Process for initiating rating service using existing frameworks

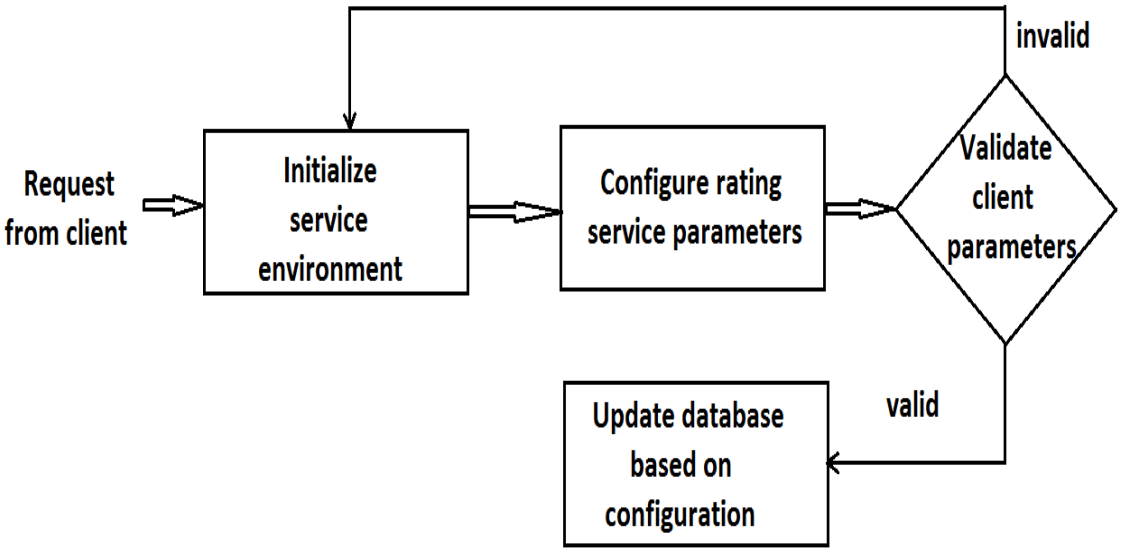

Figure 8. Server-side process explaining for initiating rating service

After configuration of service parameters, we need to validate client parameters. If the parameters obtained from the client is valid, it will proceed with database connectivity

through ODBC. If the client data obtained is invalid, then it will send call back to the client saying bad request and again it looks for new client requests. If the data is valid, then it establishes connection with external database with the help of connectivity. From here we can able to perform CRUD operations on database. It will update database results based on configuration. The implementation is done in three blocks namely client, server and database. Using current existing frameworks, most of the implementation should be done on client side. On the server side, while implementing this rating service we need to make a call back every time for finding weighted average. After this implementation, the result will be send to database for storing.

5.3 Using Proposed Framework to Provision Rating Service in an App

This section explains the provisioning of rating service in an app using proposed framework. The block diagram below explains the rating service for proposed framework on client side. For initiating the rating service on client side using proposed framework, we need to initialize the remote server first. After initialization of remote server, we need to set up server configuration parameters. It also includes server URL which will be helpful in making connection of this app with the server running on cloud. The framework will take care of initializing the running app configurations and methods with parameters for implementing rating service. After setting up server configuration parameters, we need to validate configuration. If the configuration is invalid, we need to update the service again with the call back initiating the remote server again. If the configuration is valid, provide the input given by the user to the server in Json format.

The block diagram below explains the server-side process for initiating rating service using proposed framework. The request service obtained from client is send to server as input. Once after acquiring of the input, the server environment is initialized with all the service parameters. This will configure server bases on the type of request from client. After this configuration, server will establish connectivity with database through ODBC. After the establishment of connection with database, the developer can perform CRUD operations from server side through Restful APIs.

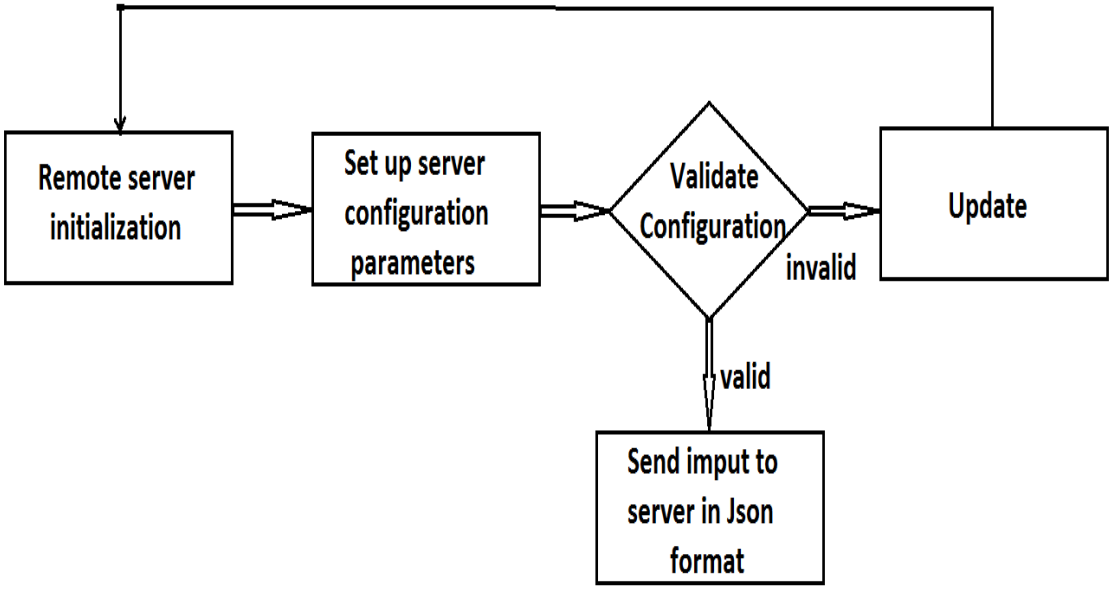

Figure 9. Flowchart explaining rating service for proposed framework

Using proposed framework, the implementation of rating service is done in tiers such as app, server and data tiers. Here most of the implementation is done on the server side which will be easily implemented using this framework. Framework will take care of initializing the service parameters and finding the weighted average. So, we can eliminate the process of call back to the input server every time.

For developing the mobile app, there are some mobile network APIs that can be used for communicating with mobile DB. As I developed android mobile app to explain it, the

inbuilt database for mobile is SQLite DB. The mobile DB is very light and its size is not more than 20 MB. So, we can store less data within the mobile database and it is not difficult for the developers to establish the connectivity between the mobile app and the database through the connecting APIs. The information that is communicated with this database is very small as it cannot store enormous amounts of data.

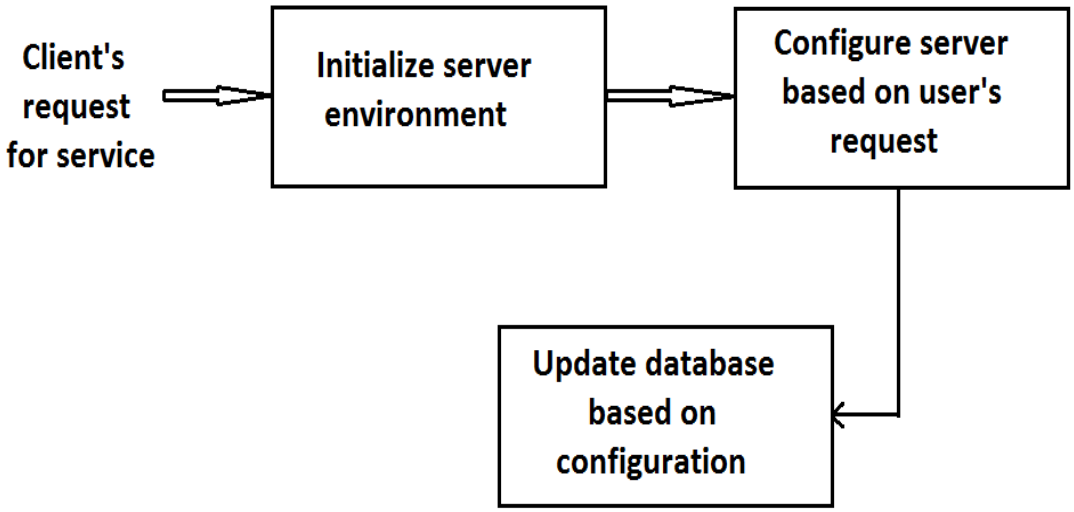

Figure 10. Process for initiating rating service in proposed framework

The second block explains the about the server and server APIs. If we consider commercial apps like amazon the storage capacity available in mobile database is not at all sufficient. So, the developer should go for large storage capable databases like MySQL, SQL SERVER and ORACLE. The mobile app is not able to talk with the huge databases easily. For that the developer should use a server to communicate with those databases. There is a process for approaching those type of connectivity.

The foremost thing is that the developer should establish the connection between mobile app and the server. After the establishment of the connection, we should write server DB access code with the help of server APIs. The server access code is different for

connecting different server databases. But it is unimportant because the app logic does not lie here. The next thing is to write a code for managing the database i.e. sending queries to the database which is same for managing all the databases. This DB managing code is important because the main app logic really matter here.

The includes server databases like MySQL, SQL SERVER, ORACLE and another DB. This block represents for storing the data which contains user's data, data related to app, secured data, commercial data etc. The next section explains the results with their explanation.

Android Studio is the IDE used in the development of project because I developed android mobile app for explaining the thesis. It is the best IDE for developing android apps and it explains both the platform and the URL used to connect with the Amazon server on the cloud. This URL provided as an input to mobile helps in finding the server that the app to connect.

The framework should provide a prefixed template that should instantiate the developer to use the same type of code with minimal changes. Here I created different server to show the fact that the developer need less change in the code with entering the new server URL to connect the server from the mobile app. It shows the main activity connecting to the current running instance of Amazon server with '/' delimiter in the ending of the URL. With the URL defining to SERVER_URL is made public can be accessed anywhere in the app without initialization. The figures explained in this section provides the example for the first block explained in the previous section. The next subsection explains the need of Amazon server with the details of server used in this project.

This subsection explains the details why we used Amazon server in the project. For the mobile app sending the data to huge databases we need a medium that will help in the communication further. There are many server providers but I used Amazon cloud because so many developers using this. I created a running instance of server on the Amazon cloud for using this server in my project. The figure 8 explains the current running instance of server on Amazon cloud.

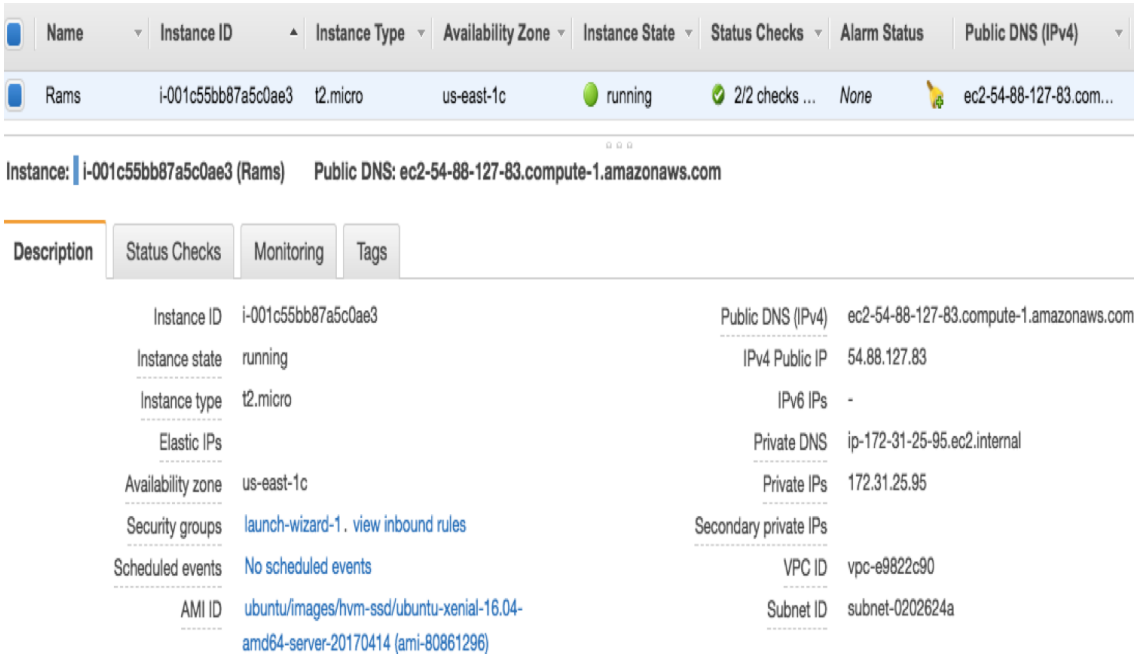

Figure 11. Running instance of current server

The Figure 8 showing all the details of running instance of the server. Within the figure, it is showing the details of name of the server, instance ID, instance type, availability zone, instance status, public DNS, and public IP. If you observe clearly, the public DNS in figure 8 is used in the URL from client side. So, this server helps in the transferring of data between mobile app and server database. The next subsection provides the information of PHP Storm IDE. It explains about the IDE used for communication between server and the

server database i.e. PHP Storm. We need to write a server code for accessing the server database. So, we need a connectivity to establish a connection between the server we are using and the server DB. This can be achieved by using Open Database Connectivity (ODBC). After getting the access of server DB, we can manage the DB with the help of some queries to the database. The managing code for every DB is same so we can use this code for all. The next sub section explains different figures related to the information in the database.

Here we explained all types of data stored in MySQL with the help of figurative explanation. MySQL is used as a database because it is a default database for the developers using PHP in the backend. It explains that the database is maintained through the server we created on the amazon cloud. This can be assured by looking at the URL in the figure 10 shows the DNS for the server running.

After connecting to the database with the base of server we can able to send data from mobile app. The mobile developed for explaining the thesis is a restaurant app primarily for ordering food from restaurant. So, the app must contain a login page for the user where the user can sign up and login. If the user signs up for an account, the data should be send to database with user details. It shows the data that the users signed up with their username, encrypted password, and email address. It is also giving the options to edit, copy or delete the user data by the administrator. On the left side of the figure you can see different tables created in the database such as users, user group, foods, reviews, and groups.

The restaurant app consists of list of restaurants where you can select one of those restaurants and the next page shows several types of items in the menu. If you select one

type among those, it will take you to items selection page with photos of the item. The next figure shows the information related to food section in the database. It shows the database having the information of different food items. Along with that, it is showing the images of the food items available in the restaurant. It is also providing the option of manipulating the items in the list.

The app is also offering an option to rate a restaurant which is important feature in the present commercial market. The next figure shows the data relevant to the rating feature. The figure 13 shows the rating given to different restaurants.

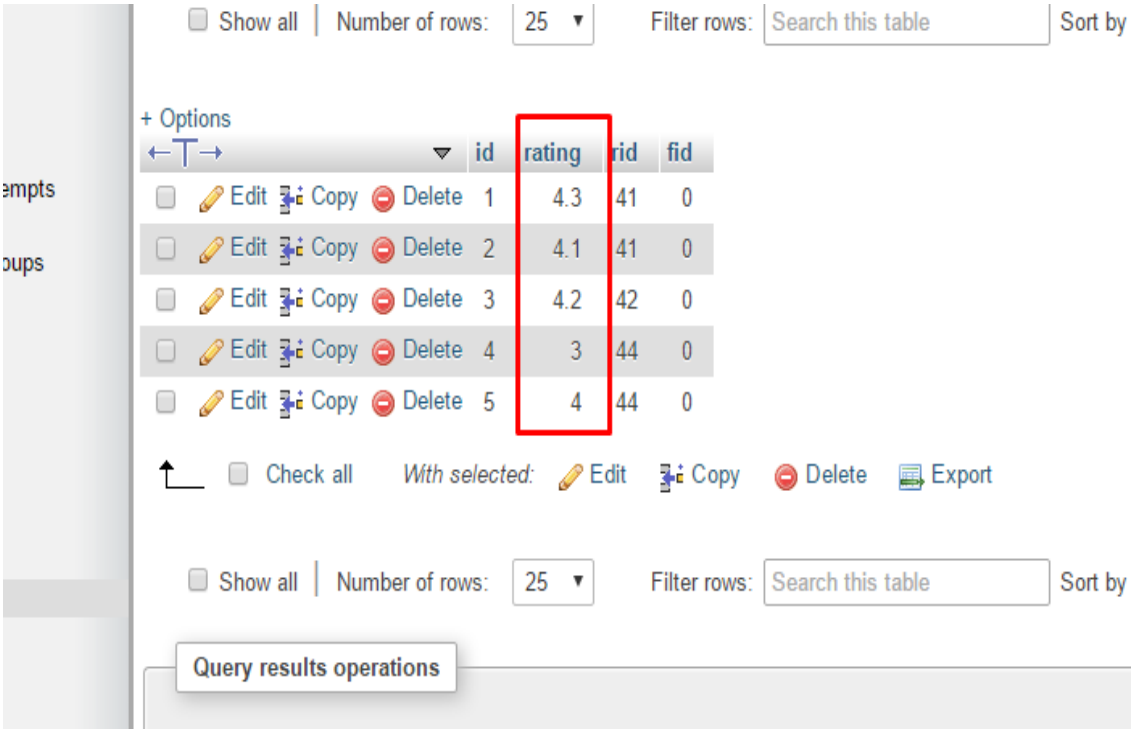

Figure 12. Database showing rating data

After selecting the list of items from the menu, you need to order it. So, the next page prompts you to order page showing the details of quantity of items and the total bill you should pay. Then it shows the option of paying the bill for the total amount.

5.4 Trade-offs Between the Two Approaches

Services can be implemented in mobile several diverse ways. Even though, the implementation using different frameworks will be different the whole sense always depends on flexibility of using frameworks. The section 5.2 explained about the implementation of rating service using current existing frameworks and 5.3 explained the implementation of same rating service using proposed framework. This section explains the trade-offs between two approaches.

While implementing rating service using current existing framework, more development process will take on client side. In the development process, the developer should take care of every criteria of initialization and configuration. If we are implementing the same rating service using proposed framework, the most part of the development process takes place on server-side. Proposed framework will hide the general implementation details and the developer should focus on operating the data.

The implementation process for other services like sign in, analytics, and logging will be different in each case, if the developer is using current existing frameworks. Different frameworks provide platform for implementing different services. So, the developer should learn every providing framework to implement those services in an app. It will be so difficult for the developer for learning and implanting those services.

The proposed framework provides different implementation mechanism for dealing with these services. If the developer knows how to implement one of the above-mentioned services, just the developer should replicate the development mechanism for rest of the

services as well. Moreover, the developer with limited technical knowledge about database connection and using these services can also implement easily with the proposed framework.

5.5 Generalizing the Approach

The above sections explain the implementation of rating service using current existing frameworks and proposed framework with tradeoffs between them. This section explains the application of implementation structure to other services such as sign in, analytics and logging using proposed framework.

Sign in service can be implemented by hiding all the details related to authorization with server providers and obtaining token ID to initialize the process. Maintenance of user data and sending request to server will be taken care by developer itself. Once after establishment of connection with server, data management will be taken place in database layer through server.

This sub section explains analytics and logging obtained from different real time commercial app. The figure 14 explains information related to several types of mobile devices using the infinity mobile with number of users for each type. It is also showing the information of different users using the mobile app from various locations. All the analytics here I am explaining about the Google analytics. It also provides the information related to real time analytics by tracking their events and screens.

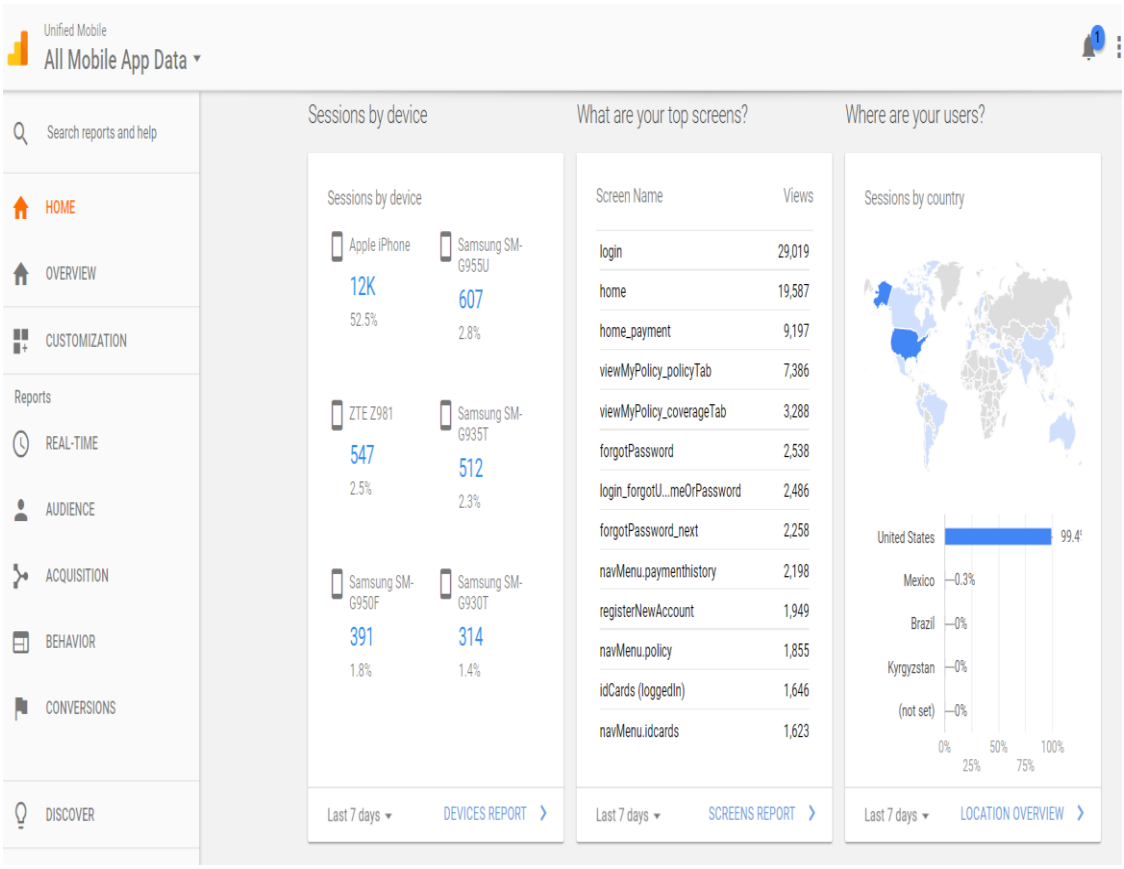

Figure 13. Infinity mobile app showing analytics

The Google analytics also shows the details of the app crashes happened over a period. Moreover, it shows the number of payments happened using mobile app.

The next figure 15 explains the analysis of various log information in terms of responsive graphs. I created logs for the different metrics, login information, feedback information and google analytics. After creating these log files, I moved it to Splunk to analyze the data. These are all the results obtained in different projects that supports my thesis.

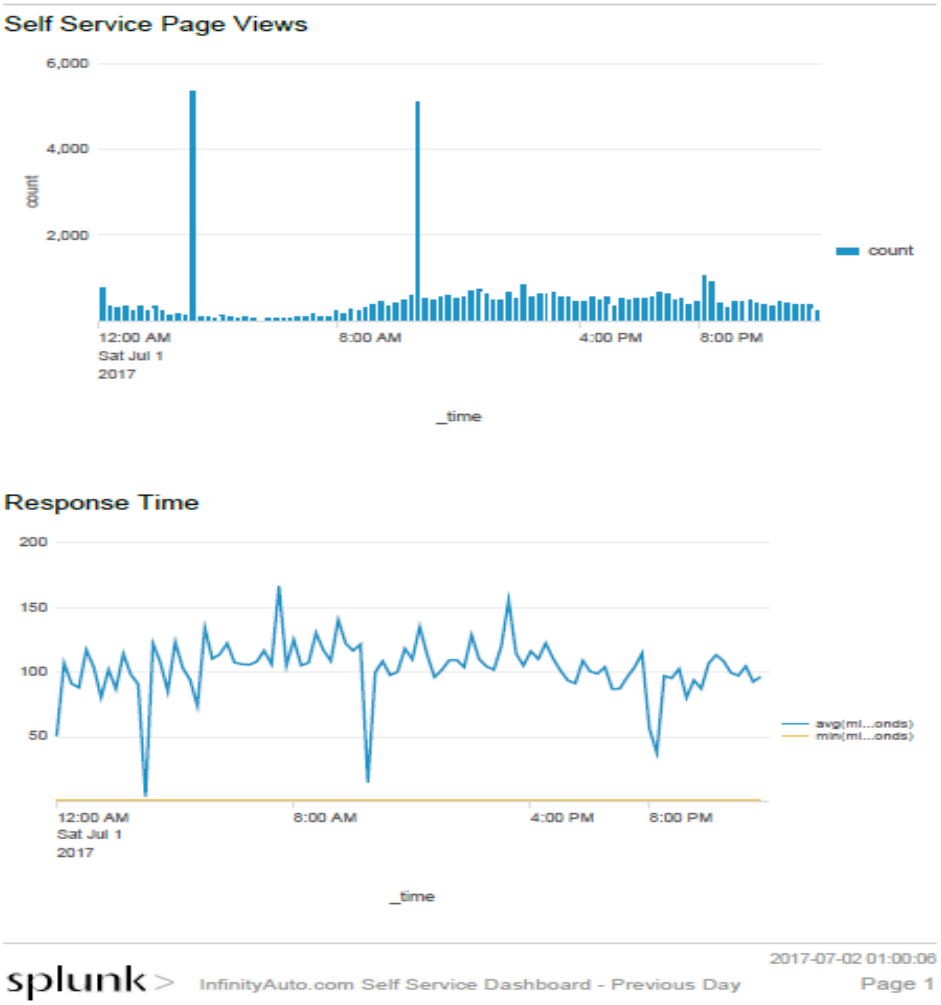

Figure 14. Splunk showing analytics for logs

CHAPTER 6

EVALUATION AND FUTURE WORK

This chapter come up with few challenges that are overlaying for mobile app development compared to different platforms supporting their app development. The section 6.1 provides the explanation of evaluation over distinctive features between the proposed framework and current existing frameworks. The next section 6.2 explains limitations faced by the proposed framework by considering several aspects related to services, management etc. The section 6.3 provides the summary of proposed framework regarding usefulness and flexibility in implementation. The last section 6.4 gives the information of how to improve the proposed system with some suggestions and how the framework can be commercialized to make it a good option for developers.

6.1 Evaluation of Proposed System

Evaluation is needed for every new system for making comparison with the existing one to see how good the system is functioning in some aspects of considerations. In this proposed framework, some of the features supported by this system made it worth to use. Existing frameworks offers a dedicated service for implementation in an app and it is very difficult to integrate new service in an app. This framework provided a platform that can easily integrate different services in apps. Few existing frameworks offer the feature of extending more services on the top of their architecture. The proposed framework is an

open framework where we can extend as many services that you can implement. Few existing systems provide services where you can implement in any environment. This framework offers services in which you can implement in any type of platform. Most of the current existing systems promotes traditional way of implementing the server rather than maintaining in the cloud. Proposed framework provides the server-side implementation in the cloud which is more resilience and accessible. The above discussed desired features with comparison is made in the below tabular form.

Table 2. Comparing proposed and existing frameworks over desired features

DESIRED FEATURE	PROPOSED FRAMEWORK	EXISTING FRAMEWORKS
Ease of integrating services in apps	Available and integrated easily	Not available with dedicated single service
Extensibility of available services	Open framework and can add more services	Not flexible to add more services
Platform independence	Services can be implemented any platform	Only few frameworks are platform independent
Server-side implemented in the cloud	Offers implantation in the cloud	Few frameworks offer this implementation

6.2 Limitations of Proposed Framework

Limitations does exist for every system because no system is perfect to use. While working with the system, the developers provide feedback and limitations of the system. The system provider should work to eliminate those limitations to make the system perfect. There are some limitations for my proposed framework after comparing with existing systems to resolve. Developing the framework can be challenging because the developer need to consider design considerations, flexibility of user, integrating several services etc. The next problem with the proposed system is when the developer implementing a certain feature offered by this system, the service is standalone and it is limited customizable. Framework maintenance is difficult because constant updates are needed every time depending on changes in provisioning of the providing services. More often there may be multiple frameworks for same type of service. The proposed framework may include only one such implementation which may not be suitable for few developers to implement.

The frequent challenges faced by mobile frameworks in furnishing mobile app development are databases, animation, gestures, performance etc. If we consider the problems in detail, then there is a possibility of sorting out few solutions. Some of the android mobile apps confronting performance issues in providing web view compared to others. Even though few functionalities related to web view working smooth, they lack in some important things which are handy. Most of the supported animations from JavaScript and CSS even we are having new scripting languages with substantial attributes. This makes android apps in lag position analogize to disparate mobile apps.

6.3 Summary of Proposed Framework

The proposed framework neutralizes some of the problems faced in performing database connectivity and providing several services in apps. In this research, we developed a single framework that combines multiple services offered by different frameworks. The framework abstracts from developers the details of establishing a service and its related database connectivity. By using this framework, developers can easily include multiple services into their apps with minimal effort. Most frameworks are specialized to dedicated service. However, a single app may require multiple frameworks. The proposed framework allows access to multiple services. Some of the aspects that are important in implementing this framework is explained in the below paragraphs.

Connectivity will be done by using ODBC through server which helps in providing interactive communication between Java and database. The methodology needed in performing this must be constructed in a well scripted manner. The foremost thing we needed is android package which consists of DBHelper class as a parent class. All other databases which need to be used as a part of android package such as MYSQL, SQL SERVER, ORACLE and SQLITE should make sub classes for main parent class DBHelper. All these sub classes must be extended by the parent class such that those databases can take advantage from parent class along with their own methods. So, android package consists of all the classes and methods with properties of above mentioned databases extended by super class. But to make use of this package we need some connection handling unit to control the things around. To make that connection we need ODBC maintained through server. It can be done by making a package which handles code for ODBC to make connectivity between database and server. Along with this, it consists

of code for authorization with database i.e. need to provide credentials like username and password to make connection. Whatever provided information above is for database connectivity framework. But to know the working of the framework we need to make an app to support it. It will have a demand when we make an app with furnishing services.

Demanding services will satisfy the needs for some of the commercial apps. Services such as authentication, analytics, logging and user rating will satisfy those needs and provide accountability to improve the working capability. Furthermore, we can extend the implementation of the framework by extending the features. It can be possible by making an open framework which indeed helpful in developing new features on the top.

6.4 Future Work

The eminent factors that will increase the utility of android apps always depends on the quality of features offered by the frameworks. As discussed earlier more services can be added on the top of the proposed architecture to provide more features to the developer for implementation. Standard features like performance, UI liabilities, gestures etc. are always needed for developers and it is a good implementation can be added in the future. To enhance the effectiveness of prominent features several frameworks have emerged in recent years. To inflate the properties of UI, React Native framework has been materialized to make the UI more interactive. In the same manner to supplement the underlying feature of gesture in android mobile apps jQuery Mobile and Sencha Touch has been introduced. Furthermore, to eliminate the lackluster of animation in android mobile apps Ionic

framework came with some mindboggling features to make the app animate. In the same manner, a framework is needed to eliminate the hurdles facing in databases.

The proposed framework is developed based on the considerations of android app development. Future work is needed to port this framework to be available for other platforms. The services offering by the framework are standalone, dedicated to only few features in a service for implementation. The future work is needed to import different features in same service so that more developers can be benefited with this implementation.

LIST OF REFERENCES

[1] Prithviraj, P., Santosh, B., and Nachiket, A., "A*dvanced Maryland Automated Network Disk Archiver Back-up of SQLite*" IEEE International Conference, Mumbai, 2012, doi: 10.1109/ICCICT.2012.6398203.

[2] Philip, A., Istvan, C., and Nishant, D., "A*dapting Microsoft SQL SERVER for Cloud Cmputing*" Data Engineering (ICDE), 2011 IEEE 27th International Conference on, Hannover, Germany, 2011, doi: 10.1109/ICDE.2011.5767935.

[3] D. Ye, *"Automated Testing Framework for ODBC Driver,"* Journal of Software Engineering and Applications, Vol. 4 No. 12, 2011, pp. 688-699. doi: 10.4236/jsea.2011.412081.

[4] Andrew. K, (2016, May), *"Ionic Guide,"* Retrieved from https://ionicframework.com/.

[5] R. T. Fielding, *"REST API must be hypertext driven"*, 28 October 2008, http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext- driven.

[6] OL. Google Android Developers, Android Develop Guide, http://developer.android.com/guide/topics/fundamentals.html.

[7] Pieter. H, (2016, Nov), *"Domain and Driven APIs for the Web,"* Retrieved from https://spring.io/blog?page=27.

[8] Backes, M., Bugiel, S., Gerling, S., and von StypRekowsky, *"P. Android Security Framework: Extensible multi-layered access control on Android,"* In Proc. 30th Annual Computer Security Applications Conference (ACSAC'14) (2014), ACM.

[9] David. B, (2017, Jan), *"Developing Enterprise Apps Using Xamarin,"* Retrieved from https://blog.xamarin.com/?showing=40.

[10] J. Zbick, M. Jansen, and M. Milrad, *"Towards a web-based framework to support end-user programming of mobile learning activities,"* in 2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT), 2014, pp. 204–208.

APPENDIX A

SETTING UP ANDROID STUDIO

A.1 Installing Related Softwares for Android Studio

Android Studio must be downloaded for developing android mobile apps. Here are steps that are needed for setting up development environment.

- Go to Google search engine and type Android Studio download
- Opens page showing the manual related to Android Studio with their tools
- Click on the Android Studio download with the latest version. It also shows about device i.e. Windows/Mac OS X
- After downloading Android Studio click on the downloaded package for installing
- Set the path for the software that you want it to install
- Accept all the license agreements mentioned there to proceed
- Set the RAM size that you want Android Studio to run on your machine preferably 1024 MB
- After all the installs happened on the machine, take care of downloading SDK tools

A.2 Installing SDK Development Tools

After the completion of installing Android Studio, we need to install all the SDK development tools. Here are the approaching steps for installing those tools.

- After downloading Android Studio, the platform provides distinct options to manage
- Go to tools options and navigate to the sub-option Android
- Android option provides several sub-options and click on SDK manager

- After clicking on SDK manager, it will prompt you to different window with diverse options
- Set the path for the SDK tools to install
- SDK platforms, SDK tools and SDK update sites are the tabs provided in this page
- Download all the packages available in all the tabs which primarily contains android platforms with their API levels, GPU debugging tools, support repository, Google API system images etc.
- After downloading all these packages, click ok to come back to develop environment

A.3 Setting Up AVD

After installing all the SDK development tools, we should set up an Android Virtual Device (AVD) for proceeding the app development process. Here are the steps to set up AVD.

- First open the Android Studio IDE and go to Tools menu for getting some more options in it
- After going to Tools menu select Android option in-order to get more options
- In Android option, it shows AVD Manager as one of the sub-option where we should click on it
- It shows the option of creating a virtual device after clicking on it promotes you to device selection page

- Device selection page consists of all the android devices to emulate in which we should select an image compatible to system we are using
- After clicking on finish, it will take you to your virtual devices where you can click on action of your selected device to emulate
- Once after done with setting up emulator, you can use your building app to emulate for testing

After done with installing all the things in Android Studio, create a new project with some new. It will create a Hello World project with all the dependencies for the project. Now start adding Java files along with design of the app. If you make all the components for building the app it will automatically generate xml for the layout. The xml file is generated for every page in the layout. After coding each functionality, you can test using emulator. Android Studio is used as the IDE for development of the app and client-side programming.

APPENDIX B

SETTING UP AMAZON SERVER

B.1 Subscription for Amazon Cloud Services

Amazon cloud is used for hosting the server on the cloud in which we can use any cloud server provider. In this section, we are discussing about making subscription for Amazon cloud.

- The foremost thing is go to Google Chrome or any browser and type Amazon cloud, search for it
- It shows the option of creating an account, click on it and submit all the user information including your contact number and mail ID
- The providers will call you for checking whether you are a legitimate user and you had any accounts before
- They will provide you a code for confirmation and after entering the code you can able to sign in to the page
- After signing in, you can able to see numerous services provided by Amazon cloud service and you can use the required service as you need

B.2 Creating Server on the Cloud

After creating an account, we need to create a new instance of a server that will link the app with the server database. Here are the steps for creating the server with all the details

- There are several web services provided by Amazon cloud and select a service where you can create a server

- Amazon Elastic Cloud (EC2) is the service for creating server, so click on it to click on diverse options in it
- Click on the create server option and provide the name of the server and the different attributes you need for that server
- After creating the server, the provider will provide you the Domain Name Service (DNS), IP4, subscription. It also provides all the details in the console of EC2
- Then you have select the availability of server to number of services at the same time
- This server is a paid subscription and we should pay depending on the usage of the server
- For usage of this server at various places we just need the URL i.e. domain name service of the server

After creating this server, it will act as a host for connecting the app with database, basically for various operations.

APPENDIX C

SETTING UP PHP STORM

C.1 Installing PHP Storm

PHP Storm is the Integrated Development Environment (IDE) for developing the server code. Using this server code, we can access the server database for performing the required operations from the mobile app. Here are the steps for downloading and installing PHP Storm.

- First step is to download PHP Storm. For doing that, go to any browser and search for PHP Storm software
- Select the executable file depending on the machine we are using i.e. Windows/Mac or any other operating system
- After downloading the executable file, search for the downloaded file and click on it to install on your system
- While installing it gives you several options such as directory of software to install, compatibility etc. and for each case click ok to install
- This will install PHP Storm into your running machine and when you click on desktop icon showing the software, it will download and load all the packages required for the software to run

After the installation, we need to develop the server code. For that, we need to follow few steps which are discussed in next section.

C.2 Development of Server Code

In this section, we will discuss how to develop server code and how to run it to check whether it is dealing with the database. Here are the steps we need to follow to develop and run the server code.

- First the developer should click on PHP Storm icon to run the IDE. It will pop up showing few projects if you developed before
- If you started working on the project before click on the project or else you need to start new project click on create new project
- After clicking on creating new project, give the name for the current project and set up the project directory
- After clicking ok, platform itself creates a list of folders assigned for different operations in the project
- The folders are created based on maven dependencies where it will auto build the project when you are working
- Create all the models for creating tables and managing those records by using this IDE. All these models are created in the controller's folder of the project and to perform different functionalities

APPENDIX D

JAVA AND PHP SKETCH

This chapter of appendix includes all the code related to developing mobile app. Here are different Java activities which represents the code for managing different screens in mobile app.

```
'''
created by Venkatadri Raparla
**********************
MyApplication.java
**********************
'''
public class MyApplication extends Application {
public static final String SERVER_URL = "http://ec2-54-88-127-83.compute-
1.amazonaws.com/RestDemo/";
   public static String selectedRestId = "";
   public static List<Food>selectedFoods = new ArrayList<>();
   public static void setCheckFood(Food food) {
      boolean isExist = false;
      for(Food f : selectedFoods) {
         if(f != null) {
            if (food.fid.equals(f.fid)) {
               List<Food> foods = new ArrayList<>();
               for (Food fd : selectedFoods) {
                  if(!fd.fid.equals(food.fid)) {
                     foods.add(fd);
                  }
               }
               selectedFoods = foods;
               isExist = true;
            }
         }
      }
      if(!isExist) {
         selectedFoods.add(food);
      }
   }
   public static boolean isExistFood(Food food) {
      boolean isExist = false;
      for(Food f : selectedFoods) {
         if(f != null) {
            if (food.fid.equals(f.fid)) {
               isExist = true;
            }
         }
      }
```

```
        return isExist;
    }
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
AndroidNetworking.get("https://fierce-cove-
29863.herokuapp.com/getAllUsers/{pageNumber}")
        .addPathParameter("pageNumber", "0")
        .addQueryParameter("limit", "3")
        .addHeaders("token", "1234")
        .setTag("test")
        .setPriority(Priority.LOW)
        .build()
        .getAsJSONArray(new JSONArrayRequestListener() {
@Override
public void onResponse(JSONArray response) {
        // do anything with response
        }
@Override
public void onError(ANError error) {
        // handle error
        }
        });
AndroidNetworking.post("https://fierce-cove-29863.herokuapp.com/createAnUser")
        .addBodyParameter("firstname", "Amit")
        .addBodyParameter("lastname", "Shekhar")
        .setTag("test")
        .setPriority(Priority.MEDIUM)
        .build()
        .getAsJSONObject(new JSONObjectRequestListener() {
@Override
public void onResponse(JSONObject response) {
        }
@Override
public void onError(ANError error) {
        // handle error
        }
        });
*/
```

```
'''
************************
LoginActivity.java
************************
'''
public class LoginActivity extends BaseActivity implements LoaderCallbacks<Cursor> {
private static final int REQUEST_READ_CONTACTS = 0;
private static final String[] DUMMY_CREDENTIALS = new String[]{
        "foo@example.com:hello", "bar@example.com:world"
   };
   private UserLoginTask mAuthTask = null;
   private AutoCompleteTextView mEmailView;
   private EditText mPasswordView;
   private View mProgressView;
   private View mLoginFormView;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      // Set up the login form.
      mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
      populateAutoComplete();
      mPasswordView = (EditText) findViewById(R.id.password);
      mPasswordView.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
         @Override
         public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
            if (id == R.id.login || id == EditorInfo.IME_NULL) {
               attemptLogin();
               return true;
            }
            return false;
         }
      });
      Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
      mEmailSignInButton.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View view) {
            attemptLogin();
         }
      });
      Button mEmailSignupButton = (Button)
findViewById(R.id.email_sign_up_button);
      mEmailSignupButton.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View view) {
```

```
            //attemptLogin();
            startActivity(new Intent(LoginActivity.this, SignupActivity.class));
        }
    });
    mLoginFormView = findViewById(R.id.login_form);
    mProgressView = findViewById(R.id.login_progress);
}
private void populateAutoComplete() {
    if (!mayRequestContacts()) {
        return;
    }
    getLoaderManager().initLoader(0, null, this);
}

private boolean mayRequestContacts() {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        return true;
    }
    if (checkSelfPermission(READ_CONTACTS) ==
PackageManager.PERMISSION_GRANTED) {
        return true;
    }
    if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {
        Snackbar.make(mEmailView, R.string.permission_rationale,
Snackbar.LENGTH_INDEFINITE)
                .setAction(android.R.string.ok, new View.OnClickListener() {
                    @Override
                    @TargetApi(Build.VERSION_CODES.M)
                    public void onClick(View v) {
                        requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
                    }
                });
    } else {
        requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
    }
    return false;
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_READ_CONTACTS) {
        if (grantResults.length == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
            populateAutoComplete();
        }
    }
}
private void attemptLogin() {
    if (mAuthTask != null) {
        return;
    }
    mEmailView.setError(null);
    mPasswordView.setError(null);
    String email = mEmailView.getText().toString();
    String password = mPasswordView.getText().toString();
    boolean cancel = false;
    View focusView = null;
    if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
        mPasswordView.setError(getString(R.string.error_invalid_password));
        focusView = mPasswordView;
        cancel = true;
    }
    if (TextUtils.isEmpty(email)) {
        mEmailView.setError(getString(R.string.error_field_required));
        focusView = mEmailView;
        cancel = true;
    } else if (!isEmailValid(email)) {
        mEmailView.setError(getString(R.string.error_invalid_email));
        focusView = mEmailView;
        cancel = true;
    }
    if (cancel) {
        focusView.requestFocus();
    } else {
        mAuthTask = new UserLoginTask(email, password);
        mAuthTask.execute((Void) null);*/
        login(email, password);
    }
}
public void login(String email, String password) {
    showCustomLoadingView();
    AndroidNetworking.post(MyApplication.SERVER_URL + "api/user/login")
            .addBodyParameter("email", email)
            .addBodyParameter("password", password)
            .setPriority(Priority.LOW)
            .build()
            .getAsJSONObject(new JSONObjectRequestListener() {
                @Override
                public void onResponse(JSONObject response) {
```

```
                    hideCustomLoadingView();
                    try {
                        String status = response.getString("status");
                        if(status.equals("1")) {
                            JSONObject user = response.getJSONObject("data");
                            User me = LoganSquare.parse(user.toString(), User.class);
                            startActivity(new Intent(LoginActivity.this, HomeActivity.class));
                            finish();
                        } else if (status.equals("0")) {
                            String error = response.getString("message");
                            showToast(error);
                        } else {
                            showToast("Network Error!");
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                        showToast(e.toString());
                    }
                }
                @Override
                public void onError(ANError error) {
                    hideCustomLoadingView();
                    showToast(error.getMessage());
                }
            });
    }
    private boolean isEmailValid(String email) {
        return email.contains("@");
    }
    private boolean isPasswordValid(String password) {
        return password.length() > 4;
    }
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    private void showProgress(final boolean show) {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB_MR2) {
            int shortAnimTime =
getResources().getInteger(android.R.integer.config_shortAnimTime);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
            mLoginFormView.animate().setDuration(shortAnimTime).alpha(
                    show ? 0 : 1).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
                }
            });
```

```
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mProgressView.animate().setDuration(shortAnimTime).alpha(
                    show ? 1 : 0).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
                }
            });
        } else {
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
        }
    }
    @Override
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this,
                Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
                        ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
ProfileQuery.PROJECTION,
                ContactsContract.Contacts.Data.MIMETYPE +
                        " = ?", new String[]{ContactsContract.CommonDataKinds.Email
                .CONTENT_ITEM_TYPE},
                ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");
    }
    @Override
    public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
        List<String> emails = new ArrayList<>();
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            emails.add(cursor.getString(ProfileQuery.ADDRESS));
            cursor.moveToNext();
        }
        addEmailsToAutoComplete(emails);
    }
    @Override
    public void onLoaderReset(Loader<Cursor> cursorLoader) {
    }
    private void addEmailsToAutoComplete(List<String> emailAddressCollection) {
        ArrayAdapter<String> adapter =
                new ArrayAdapter<>(LoginActivity.this,
                        android.R.layout.simple_dropdown_item_1line, emailAddressCollection);
        mEmailView.setAdapter(adapter);
    }
    private interface ProfileQuery {
        String[] PROJECTION = {
                ContactsContract.CommonDataKinds.Email.ADDRESS,
```

```
            ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
        };
        int ADDRESS = 0;
        int IS_PRIMARY = 1;
    }
    public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {
        private final String mEmail;
        private final String mPassword;
        UserLoginTask(String email, String password) {
            mEmail = email;
            mPassword = password;
        }
        @Override
        protected Boolean doInBackground(Void... params) {
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                return false;
            }
            for (String credential : DUMMY_CREDENTIALS) {
                String[] pieces = credential.split(":");
                if (pieces[0].equals(mEmail)) {
                    // Account exists, return true if the password matches.
                    return pieces[1].equals(mPassword);
                }
            }
            return true;
        }
        @Override
        protected void onPostExecute(final Boolean success) {
            mAuthTask = null;
            showProgress(false);
            if (success) {
                finish();
            } else {
                mPasswordView.setError(getString(R.string.error_incorrect_password));
                mPasswordView.requestFocus();
            }
        }
        @Override
        protected void onCancelled() {
            mAuthTask = null;
            showProgress(false);
        }
    }
}
```

'''
****
SignupActivity.java
****
'''

```
public class SignupActivity extends BaseActivity implements LoaderCallbacks<Cursor>
{
   private static final int REQUEST_READ_CONTACTS = 0;
   private static final String[] DUMMY_CREDENTIALS = new String[]{
         "foo@example.com:hello", "bar@example.com:world"
   };
   private UserLoginTask mAuthTask = null;
   private AutoCompleteTextView mEmailView;
   private EditText mPasswordView;
   private EditText mConfirmPasswordView;
   private View mProgressView;
   private View mLoginFormView;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_signup);
      mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
      populateAutoComplete();
      mPasswordView = (EditText) findViewById(R.id.password);
      mPasswordView.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
         @Override
         public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
            if (id == R.id.login || id == EditorInfo.IME_NULL) {
               return true;
            }
            return false;
         }
      });
      mConfirmPasswordView = (EditText) findViewById(R.id.confirm_password);
      mConfirmPasswordView.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
         @Override
         public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
            if (id == R.id.login || id == EditorInfo.IME_NULL) {
               attemptSignup();
               return true;
            }
            return false;
         }
      });
```

```
        Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
        mEmailSignInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(SignupActivity.this, LoginActivity.class));
            }
        });
        Button mEmailSignupButton = (Button)
findViewById(R.id.email_sign_up_button);
        mEmailSignupButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptSignup();
            }
        });
        mLoginFormView = findViewById(R.id.login_form);
        mProgressView = findViewById(R.id.login_progress);
    }
    private void populateAutoComplete() {
        if (!mayRequestContacts()) {
            return;
        }
        getLoaderManager().initLoader(0, null, this);
    }
    private boolean mayRequestContacts() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            return true;
        }
        if (checkSelfPermission(READ_CONTACTS) ==
PackageManager.PERMISSION_GRANTED) {
            return true;
        }
        if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {
            Snackbar.make(mEmailView, R.string.permission_rationale,
Snackbar.LENGTH_INDEFINITE)
                    .setAction(android.R.string.ok, new OnClickListener() {
                        @Override
                        @TargetApi(Build.VERSION_CODES.M)
                        public void onClick(View v) {
                            requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
                        }
                    });
        } else {
            requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
```

```
        }
        return false;
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode == REQUEST_READ_CONTACTS) {
            if (grantResults.length == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                populateAutoComplete();
            }
        }
    }
    private void attemptSignup() {
        if (mAuthTask != null) {
            return;
        }
        mEmailView.setError(null);
        mPasswordView.setError(null);
        mConfirmPasswordView.setError(null);
        String email = mEmailView.getText().toString();
        String password = mPasswordView.getText().toString();
        String confirmpassword = mConfirmPasswordView.getText().toString();
        boolean cancel = false;
        View focusView = null;
        if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
            mPasswordView.setError(getString(R.string.error_invalid_password));
            focusView = mPasswordView;
            cancel = true;
        }

        if (!TextUtils.isEmpty(confirmpassword) && !isPasswordValid(confirmpassword))
{
            mConfirmPasswordView.setError(getString(R.string.error_invalid_password));
            focusView = mConfirmPasswordView;
            cancel = true;
        }
        if(!password.equals(confirmpassword)) {
            mConfirmPasswordView.setError("Please input same password");
            focusView = mConfirmPasswordView;
            cancel = true;
        }
        if (TextUtils.isEmpty(email)) {
            mEmailView.setError(getString(R.string.error_field_required));
            focusView = mEmailView;
```

```
            cancel = true;
        } else if (!isEmailValid(email)) {
            mEmailView.setError(getString(R.string.error_invalid_email));
            focusView = mEmailView;
            cancel = true;
        }
        if (cancel) {
            focusView.requestFocus();
        } else {
            mAuthTask = new UserLoginTask(email, password);
            mAuthTask.execute((Void) null);*/
            signup(email, password);
        }
    }
    public void signup(String email, String password) {
        showCustomLoadingView();
        AndroidNetworking.post(MyApplication.SERVER_URL + "api/user/signup")
                .addBodyParameter("email", email)
                .addBodyParameter("password", password)
                .addBodyParameter("user_type", "customer")
                .setPriority(Priority.LOW)
                .build()
                .getAsJSONObject(new JSONObjectRequestListener() {
                    @Override
                    public void onResponse(JSONObject response) {
                        hideCustomLoadingView();
                        try {
                            String status = response.getString("status");
                            if(status.equals("1")) {
                                JSONObject user = response.getJSONObject("data");
                                User me = LoganSquare.parse(user.toString(), User.class);
                                startActivity(new Intent(SignupActivity.this, HomeActivity.class));
                                finish();
                            } else if (status.equals("0")) {
                                String error = response.getString("message");
                                showToast(error);
                            } else {
                                showToast("Network Error!");
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                            showToast(e.toString());
                        }
                    }
                    @Override
                    public void onError(ANError error) {
```

```
                hideCustomLoadingView();
                showToast(error.getMessage());
            }
        });
    }
    private boolean isEmailValid(String email) {
        return email.contains("@");
    }
    private boolean isPasswordValid(String password) {
        return password.length() > 4;
    }
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    private void showProgress(final boolean show) {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB_MR2) {
            int shortAnimTime =
getResources().getInteger(android.R.integer.config_shortAnimTime);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
            mLoginFormView.animate().setDuration(shortAnimTime).alpha(
                    show ? 0 : 1).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
                }
            });
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mProgressView.animate().setDuration(shortAnimTime).alpha(
                    show ? 1 : 0).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
                }
            });
        } else {
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
        }
    }
    @Override
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this,
                Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
                        ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
ProfileQuery.PROJECTION,
                ContactsContract.Contacts.Data.MIMETYPE +
                        " = ?", new String[]{ContactsContract.CommonDataKinds.Email
```

```
        .CONTENT_ITEM_TYPE},
        ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");
}
@Override
public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
   List<String> emails = new ArrayList<>();
   cursor.moveToFirst();
   while (!cursor.isAfterLast()) {
      emails.add(cursor.getString(ProfileQuery.ADDRESS));
      cursor.moveToNext();
   }
   addEmailsToAutoComplete(emails);
}
@Override
public void onLoaderReset(Loader<Cursor> cursorLoader) {
}
private void addEmailsToAutoComplete(List<String> emailAddressCollection) {
   ArrayAdapter<String> adapter =
         new ArrayAdapter<>(SignupActivity.this,
               android.R.layout.simple_dropdown_item_1line, emailAddressCollection);
   mEmailView.setAdapter(adapter);
}
private interface ProfileQuery {
   String[] PROJECTION = {
         ContactsContract.CommonDataKinds.Email.ADDRESS,
         ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
   };
   int ADDRESS = 0;
   int IS_PRIMARY = 1;
}
public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {
   private final String mEmail;
   private final String mPassword;
   UserLoginTask(String email, String password) {
      mEmail = email;
      mPassword = password;
   }
   @Override
   protected Boolean doInBackground(Void... params) {
      try {
         Thread.sleep(2000);
      } catch (InterruptedException e) {
         return false;
      }
      for (String credential : DUMMY_CREDENTIALS) {
         String[] pieces = credential.split(":");
```

```
            if (pieces[0].equals(mEmail)) {
                return pieces[1].equals(mPassword);
            }
        }
        return true;
    }
    @Override
    protected void onPostExecute(final Boolean success) {
        mAuthTask = null;
        showProgress(false);
        if (success) {
            finish();
        } else {
            mPasswordView.setError(getString(R.string.error_incorrect_password));
            mPasswordView.requestFocus();
        }
    }
    @Override
    protected void onCancelled() {
        mAuthTask = null;
        showProgress(false);
    }
  }
}
'''
**********
HomeActivity.java
**********
'''
public class HomeActivity extends BaseActivity {
  ListviewAdapter adapter;
  ListView listView;
  List<Rest> data = new ArrayList<>();
  private SwipeRefreshLayout mSwipeRefreshLayout;
  ListviewAdapter.OnButtonClickListener mListener;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    mContext = this;
    hideLeftIV();
    showRightIV();
    showTitle();
    setTitle("Home");
    rightIV.setOnClickListener(new View.OnClickListener() {
      @Override
```

```
            public void onClick(View view) {
                startActivity(new Intent(mContext, LoginActivity.class));
                finish();
            }
        });
        listView = (ListView) findViewById(R.id.listView);
        mListener = new ListviewAdapter.OnButtonClickListener() {
            @Override
            public void onClick(int position, boolean isUpload) {
                MyApplication.selectedRestId = data.get(position).rid;
                startActivity(new Intent(mContext, FoodsActivity.class));
            }
        };

        getData();
        adapter = new ListviewAdapter(mContext, R.layout.list_item, (ArrayList<Rest>)
data);
        adapter.setOnButtonClickListener(mListener);
        listView.setAdapter(adapter);
        mSwipeRefreshLayout = (SwipeRefreshLayout)
findViewById(R.id.swipeRefreshLayout);
        mSwipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                getData();
            }
        });
    }
    private void getData() {
        showCustomLoadingView();
        AndroidNetworking.post(MyApplication.SERVER_URL + "api/user/getAllRests")
                .setPriority(Priority.LOW)
                .build()
                .getAsJSONObject(new JSONObjectRequestListener() {
                    @Override
                    public void onResponse(JSONObject response) {
                        hideCustomLoadingView();
                        if(mSwipeRefreshLayout != null) {
                            mSwipeRefreshLayout.setRefreshing(false);
                        }
                        try {
                            String status = response.getString("status");
                            if(status.equals("1")) {
                                JSONArray array = response.getJSONArray("data");
                                data = LoganSquare.parseList(array.toString(), Rest.class);
```

```
                        if(adapter!=null) {
                            adapter = new ListviewAdapter(mContext, R.layout.list_item,
(ArrayList<Rest>) data);
                            adapter.setOnButtonClickListener(mListener);
                            listView.setAdapter(adapter);
                            adapter.notifyDataSetChanged();
                        }
                    } else if (status.equals("0")) {
                        String error = response.getString("message");
                        showToast(error);
                    } else {
                        showToast("Network Error!");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    showToast(e.toString());
                }
            }
            @Override
            public void onError(ANError error) {
                hideCustomLoadingView();
                showToast(error.getMessage());
                if(mSwipeRefreshLayout != null) {
                    mSwipeRefreshLayout.setRefreshing(false);
                }
            }
        });
    }
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        MyApplication.selectedRestId = String.valueOf(i);
        startActivity(new Intent(mContext, FoodsActivity.class));
    }
}
'''
**********
FoodsActivity.java
**********
'''
public class FoodsActivity extends BaseActivity {
    FoodListviewAdapter adapter;
    ListView listView;
    Button orderBtn;
    List<Food> data = new ArrayList<>();
    private SwipeRefreshLayout mSwipeRefreshLayout;
    FoodListviewAdapter.OnButtonClickListener mListener;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_food);
        mContext = this;
        hideLeftIV();
        showRightIV();
        showTitle();
        setTitle("Menu");
        rightIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(mContext, LoginActivity.class));
                finish();
            }
        });
        listView = (ListView) findViewById(R.id.listView);
        mListener = new FoodListviewAdapter.OnButtonClickListener() {
            @Override
            public void onClick(int position, boolean isUpload) {
                MyApplication.setCheckFood(data.get(position));
                if(adapter != null) {
                    adapter.notifyDataSetChanged();
                }
            }
        };
        adapter = new FoodListviewAdapter(mContext, R.layout.list_item_food,
(ArrayList<Food>) data);
        adapter.setOnButtonClickListener(mListener);
        listView.setAdapter(adapter);
        mSwipeRefreshLayout = (SwipeRefreshLayout)
findViewById(R.id.swipeRefreshLayout);
        mSwipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                getData();
            }
        });
        orderBtn = (Button) findViewById(R.id.btn);
        orderBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                float price = 0;
                for(Food f : MyApplication.selectedFoods) {
                    price += f.price;
                }
```

```
            showToast("Total : " + String.valueOf(price) + "$");
            startActivity(new Intent(mContext, PayActivity.class));
            finish();
        }
    });
}
@Override
public void onResume() {
    super.onResume();
    getData();
}
private void getData() {
    showCustomLoadingView();
    AndroidNetworking.post(MyApplication.SERVER_URL +
"api/food/getFoodsWithRestId")
            .addBodyParameter("rid", MyApplication.selectedRestId)
            .setPriority(Priority.LOW)
            .build()
            .getAsJSONObject(new JSONObjectRequestListener() {
                @Override
                public void onResponse(JSONObject response) {
                    hideCustomLoadingView();
                    if(mSwipeRefreshLayout != null) {
                        mSwipeRefreshLayout.setRefreshing(false);
                    }
                    try {
                        String status = response.getString("status");
                        if(status.equals("1")) {
                            JSONArray array = response.getJSONArray("data");
                            data = LoganSquare.parseList(array.toString(), Food.class);
                            MyApplication.selectedFoods = new ArrayList<Food>();
                            for (Food food : data) {
                                food.isChcek = 0;
                            }
                            if(adapter!=null) {
                                adapter = new FoodListviewAdapter(mContext,
R.layout.list_item_food, (ArrayList<Food>) data);
                                adapter.setOnButtonClickListener(mListener);
                                listView.setAdapter(adapter);
                                adapter.notifyDataSetChanged();
                            }
                        } else if (status.equals("0")) {
                            String error = response.getString("message");
                            showToast(error);
                        } else {
                            showToast("Network Error!");
```

```
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    showToast(e.toString());
                }
            }
            @Override
            public void onError(ANError error) {
                hideCustomLoadingView();
                showToast(error.getMessage());
                if(mSwipeRefreshLayout != null) {
                    mSwipeRefreshLayout.setRefreshing(false);
                }
            }
        });
    }
}
'''
**********
PayActivity.java
**********
'''
public class PayActivity extends BaseActivity {
    LinearLayout containerLL;
    Button payBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pay);
        mContext = this;
        hideLeftIV();
        showRightIV();
        showTitle();
        setTitle("Pay");
        rightIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(mContext, LoginActivity.class));
                finish();
            }
        });
        containerLL = (LinearLayout) findViewById(R.id.containerLL);
        payBtn = (Button) findViewById(R.id.payBtn);
        payBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
            showToast("Successfully paid");
            MyApplication.selectedFoods = new ArrayList<Food>();
          }
        });
        float price = 0;
        for (Food f : MyApplication.selectedFoods) {
          price += f.price;
          containerLL.addView(new PayItemView(mContext, f.name, "1", f.price+"$"));
        }
        containerLL.addView(new PayItemView(mContext, "Total", "1",
String.valueOf(price)+"$"));
    }
}
'''
**********
ListViewAdapter.java
**********
'''
public class ListviewAdapter extends BaseAdapter {
    private LayoutInflater inflater;
    private ArrayList<Rest> data;
    private Context mContext;
    private int layout;
    OnButtonClickListener mOnButtonClickListener = null;
    public ListviewAdapter(Context context, int layout, ArrayList<Rest> data){
this.inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_
SERVICE);
        this.data=data;
        this.layout=layout;
        mContext = context;
    }
    @Override
    public int getCount(){return data.size();}
    @Override
    public Rest getItem(int position){return data.get(position);}
    @Override
    public long getItemId(int position){return position;}
    @Override
    public View getView(final int position, View convertView, ViewGroup parent){
        if(convertView==null){
            convertView=inflater.inflate(layout,parent,false);
        }
        Rest listviewitem = data.get(position);
        TextView nameTV = (TextView)convertView.findViewById(R.id.nameTV);
        nameTV.setText(listviewitem.name);
        TextView ratingTV = (TextView)convertView.findViewById(R.id.ratingTV);
```

```
        ratingTV.setText(String.valueOf(listviewitem.rating));
        RatingBar ratingBar = (RatingBar) convertView.findViewById(R.id.ratingBar);
        ratingBar.setRating(listviewitem.rating);
        ImageView iv = (ImageView) convertView.findViewById(R.id.iv);
        if(listviewitem.photo != null && !listviewitem.equals("")) {
Picasso.with(mContext).load(MyApplication.SERVER_URL+listviewitem.photo).placeh
older(R.drawable.back_thumb).into(iv);
        }
        convertView.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
              if(mOnButtonClickListener != null) {
                 mOnButtonClickListener.onClick(position, true);
              }
           }
        });
        return convertView;
    }
    public void setOnButtonClickListener (OnButtonClickListener listener) {
        mOnButtonClickListener = listener;
    }
    public interface OnButtonClickListener {
        void onClick(int position, boolean isUpload);
    }
}
'''
**********
FoodListViewAdapter.java
**********
'''
public class FoodListviewAdapter extends BaseAdapter {
    private LayoutInflater inflater;
    private ArrayList<Food> data;
    private Context mContext;
    private int layout;
    OnButtonClickListener mOnButtonClickListener = null;
    public FoodListviewAdapter(Context context, int layout, ArrayList<Food> data){
this.inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_
SERVICE);
        this.data=data;
        this.layout=layout;
        mContext = context;
    }
    @Override
    public int getCount(){return data.size();}
    @Override
```

```
    public Food getItem(int position){return data.get(position);}
    @Override
    public long getItemId(int position){return position;}
    @Override
    public View getView(final int position, View convertView, ViewGroup parent){
        if(convertView==null){
            convertView=inflater.inflate(layout,parent,false);
        }
        Food listviewitem = data.get(position);
        TextView nameTV = (TextView)convertView.findViewById(R.id.nameTV);
        nameTV.setText(listviewitem.name);
        TextView priceTV = (TextView)convertView.findViewById(R.id.priceTV);
        priceTV.setText(String.valueOf(listviewitem.price) + "$");
        ImageView iv = (ImageView) convertView.findViewById(R.id.iv);
        if(listviewitem.photo != null && !listviewitem.equals("")) {

Picasso.with(mContext).load(MyApplication.SERVER_URL+listviewitem.photo).into(iv
);
        }
        ImageView checkIV = (ImageView) convertView.findViewById(R.id.checkIV);
        if(MyApplication.isExistFood(listviewitem)) {
            checkIV.setVisibility(View.VISIBLE);
        } else {
            checkIV.setVisibility(View.GONE);
        }
        convertView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mOnButtonClickListener != null) {
                    mOnButtonClickListener.onClick(position, true);
                }
            }
        });
        return convertView;
    }
    public void setOnButtonClickListener (OnButtonClickListener listener) {
        mOnButtonClickListener = listener;
    }
    public interface OnButtonClickListener {
        void onClick(int position, boolean isUpload);
    }
}
```

All the above files are app related java files from Android Studio which includes code for developing the app and linking URL with the server on the Cloud.

The code from here is PHP code basically the server code for accessing the MySQL database which is default for PHP development. And the code related to managing the database i.e. querying the database.

```
'''
created by Venkatadri Raparla
************************
Admin_Controller.php
************************
'''
<?php
class Admin_Controller extends MY_Controller {
	protected $mUsefulLinks = array();
	protected $mCrud;
	protected $mCrudUnsetFields;
	public function __construct()
	{
		parent::__construct();
		$this->verify_login();
		$this->mUsefulLinks = $this->mConfig['useful_links'];
	}
	protected function render($view_file, $layout = 'default')
	{
		$config = $this->mConfig['adminlte'];
		$this->mBodyClass = $config['body_class'][$this->mUserMainGroup];
		// additional view data
		$this->mViewData['useful_links'] = $this->mUsefulLinks;
		parent::render($view_file, $layout);
	}
	protected function generate_crud($table, $subject = '')
	{
		$this->load->library('Grocery_CRUD');
		$crud = new grocery_CRUD();
		$crud->set_table($table);
		if ( empty($subject) )
		{
			$crud->set_subject(humanize(singular($table)));
		}
		$this->load->config('grocery_crud');
		$this->mCrudUnsetFields = $this->config-
>item('grocery_crud_unset_fields');
		if ($this->config->item('grocery_crud_unset_jquery'))
			$crud->unset_jquery();
		if ($this->config->item('grocery_crud_unset_jquery_ui'))
			$crud->unset_jquery_ui();
		if ($this->config->item('grocery_crud_unset_print'))
```

```
                $crud->unset_print();
            if ($this->config->item('grocery_crud_unset_export'))
                $crud->unset_export();
            if ($this->config->item('grocery_crud_unset_read'))
                $crud->unset_read();
            foreach ($this->config->item('grocery_crud_display_as') as $key =>
$value)
                $crud->display_as($key, $value);
            // other custom logic to be done outside
            $this->mCrud = $crud;
            return $crud;
        }
        protected function set_crud_color_picker()
        {
            $args = func_get_args();
            if(isset($args[0]) && is_array($args[0]))
            {
                $args = $args[0];
            }
            foreach ($args as $field)
            {
                $this->mCrud->callback_field($field, array($this,
'callback_color_picker'));
            }
        }
        public function callback_color_picker($value = '', $primary_key = NULL, $field
= NULL)
        {
            $name = $field->name;
            return "<input type='color' name='$name' value='$value'
style='width:80px' />";
        }
        protected function unset_crud_fields()
        {
            $args = func_get_args();
            if(isset($args[0]) && is_array($args[0]))
            {
                $args = $args[0];
            }
            $this->mCrudUnsetFields = array_merge($this->mCrudUnsetFields,
$args);
        }
        protected function generate_image_crud($table, $url_field, $upload_path,
$order_field = 'pos', $title_field = '')
        {
            // create CRUD object
```

```
		$this->load->library('Image_crud');
		$crud = new image_CRUD();
		$crud->set_table($table);
		$crud->set_url_field($url_field);
		$crud->set_image_path($upload_path);
		if ( !empty($order_field) )
		{
			$crud->set_ordering_field($order_field);
		}
		if ( !empty($title_field) )
		{
			$crud->set_title_field($title_field);
		}
		$this->mCrud = $crud;
		return $crud;
	}
	protected function render_crud()
	{
		$crud_obj_name = strtolower(get_class($this->mCrud));
		if ($crud_obj_name==='grocery_crud')
		{
			$this->mCrud->unset_fields($this->mCrudUnsetFields);
		}
		$crud_data = $this->mCrud->render();
		$this->add_stylesheet($crud_data->css_files, FALSE);
		$this->add_script($crud_data->js_files, TRUE, 'head');
		$this->mViewData['crud_output'] = $crud_data->output;
		$this->render('crud');
	}
}
'''
**********
Admin_user_model.php
**********
'''
<?php
class Admin_user_model extends MY_Model {
}
'''
**********
Api_Controller.php
**********
'''
<?php
require_once(APPPATH.'third_party/rest_server/libraries/REST_Controller.php');
class API_Controller extends REST_Controller {
```

```
	protected $mApiKey = NULL;
	public function __construct()
	{
		parent::__construct();
		$config = $this->config->item('ci_bootstrap');
		$headers = empty($config['headers']) ? array() : $config['headers'];
		foreach ($headers as $header)
		{
			header($header);
		}
		$this->verify_token();
	}
	protected function verify_token()
	{
		$key = $this->input->get_request_header('X-API-KEY');
		$this->mApiKey = $this->api_keys->get_by('key', $key);
		if ( !empty($this->mApiKey) )
		{
			$this->mUser = $this->users->get($this->mApiKey->user_id);
			// only when the API Key represents a user
			if ( !empty($this->mUser) )
			{
				$this->mUserGroups = $this->ion_auth-
>get_users_groups($this->mUser->id)->result();
				// TODO: get group with most permissions (instead of
getting first group)
				$this->mUserMainGroup = $this->mUserGroups[0]-
>name;
			}
			else
			{
				// anonymous access via API Key
				$this->mUserMainGroup = 'anonymous';
			}
		}
	}
	protected function verify_auth($groups = 'members')
	{
		$groups = is_string($groups) ? array($groups) : $groups;

		if ( empty($this->mUser) )
		{
			if ( !in_array($this->mUserMainGroup, $groups) )
				$this->error_unauthorized();
		}
		else
```

```
                {
                        if ( !$this->ion_auth->in_group($groups, $this->mUser->id) )
                                $this->error_unauthorized();
                }
        }
        protected function success($msg = NULL)
        {
                $data = array('status' => TRUE);
                if ( !empty($msg) ) $data['message'] = $msg;
                $this->response($data, REST_Controller::HTTP_OK);
        }
        protected function created($msg = NULL)
        {
                $data = array('status' => TRUE);
                if ( !empty($msg) ) $data['message'] = $msg;
                $this->response($data, REST_Controller::HTTP_CREATED);
        }
        protected function accepted($msg = NULL)
        {
                $data = array('status' => TRUE);
                if ( !empty($msg) ) $data['message'] = $msg;
                $this->response($data, REST_Controller::HTTP_ACCEPTED);
        }
        protected function error($msg = 'An error occurs', $code =
REST_Controller::HTTP_OK, $additional_data = array())
        {
                $data = array('status' => FALSE, 'error' => $msg);
                if (!empty($additional_data))
                        $data['data'] = $additional_data;
                $this->response($data, $code);
        }
        protected function error_bad_request()
        {
                $data = array('status' => FALSE);
                $this->response($data, REST_Controller::HTTP_BAD_REQUEST);
        }
        protected function error_unauthorized()
        {
                $data = array('status' => FALSE);
                $this->response($data, REST_Controller::HTTP_UNAUTHORIZED);
        }
        protected function error_forbidden()
        {
                $data = array('status' => FALSE);
                $this->response($data, REST_Controller::HTTP_FORBIDDEN);
        }
```

```
	protected function error_not_found()
	{
		$data = array('status' => FALSE);
		$this->response($data, REST_Controller::HTTP_NOT_FOUND);
	}
	protected function error_method_not_allowed()
	{
		$data = array('status' => FALSE);
		$this->response($data,
REST_Controller::HTTP_METHOD_NOT_ALLOWED);
	}
	protected function error_not_implemented($additional_data = array())
	{
		if (ENVIRONMENT=='development')
		{
			$trace = debug_backtrace();
			$caller = $trace[1];
			$data = array(
				'url'			=> current_url(),
				'module'		=> $this->router->fetch_module(),
				'controller'	=> $this->router->fetch_class(),
				'action'		=> $this->router->fetch_method(),
			);
			if (!empty($additional_data))
				$data = array_merge($data, $additional_data);
			$this->error('Not implemented',
REST_Controller::HTTP_NOT_IMPLEMENTED, $data);
		}
		else
		{
			$this->error_not_found();
		}
	}
	protected function _generate_key()
	{
		do
		{
			// Generate a random salt
			$salt = base_convert(bin2hex($this->security-
>get_random_bytes(64)), 16, 36);
			if ($salt === FALSE)
			{
				$salt = hash('sha256', time() . mt_rand());
			}
			$new_key = substr($salt, 0, config_item('rest_key_length'));
		}
```

```
                while ($this->_key_exists($new_key));
                return $new_key;
        }
        protected function _get_key($key)
        {
                return $this->db
                        ->where(config_item('rest_key_column'), $key)
                        ->get(config_item('rest_keys_table'))
                        ->row();
        }
        protected function _key_exists($key)
        {
                return $this->db
                        ->where(config_item('rest_key_column'), $key)
                        ->count_all_results(config_item('rest_keys_table')) > 0;
        }
        protected function _insert_key($key, $data)
        {
                $data[config_item('rest_key_column')] = $key;
                $data['date_created'] = function_exists('now') ? now() : time();
                return $this->db
                        ->set($data)
                        ->insert(config_item('rest_keys_table'));
        }
        protected function _update_key($key, $data)
        {
                return $this->db
                        ->where(config_item('rest_key_column'), $key)
                        ->update(config_item('rest_keys_table'), $data);
        }
        protected function _delete_key($key)
        {
                return $this->db
                        ->where(config_item('rest_key_column'), $key)
                        ->delete(config_item('rest_keys_table'));
        }
}

'''
**********************
Food_model.php
**********************
'''
<?php
class Food_model extends MY_Model {
   public function get_where ($where) {
```

```
        $this->where = $where;
        return $this->get_all();
    }
}
'''
************************
Group_model.php
************************
'''
<?php
class Group_model extends MY_Model {
}
'''
************************
Order_model.php
************************
'''
class Order_model extends MY_Model {
    public function get_where ($where) {
        $this->where = $where;
        return $this->get_all();
    }
    public function getCustomersWithBarberId($barber_id) {
        $sql = "SELECT * FROM users WHERE id IN (SELECT customer_id FROM
orders WHERE barber_id = ".$barber_id." AND (status = 'start' OR status = 'waiting')
ORDER BY created_on DESC)";
        $result = $this->db->query($sql)->result();
        return $result;
    }
    public function getAverageCuttingTimeWithBarberId($barber_id) {
        $sql = "SELECT SUM(end_time - start_time) as sum, COUNT(customer_id) as
count FROM orders WHERE barber_id = ".$barber_id." AND status = 'end'";
        $data = $this->db->query($sql)->result();
        $result = $data[0];
        return $result;
    }
}
'''
************************
Review_model.php
************************
'''
class Review_model extends MY_Model {
    public function get_where ($where) {
        $this->where = $where;
        return $this->get_all();
```

```
    }
}
'''
************************
User_model.php
************************
'''
class User_model extends MY_Model {
    protected function callback_after_get($result = null) {
        if(isset($result->id)) {
            $sql = "SELECT SUM(rating) / COUNT(id) AS 'avg' FROM reviews WHERE
rid = $result->id";
            $rating = $this->db->query($sql)->result();
            $rating = $rating[0];
            $avg = number_format($rating->avg,1);
            if($avg == null) {
                $avg = 0;
            }
            $result->rating = $avg;
            return $result;
        }
    }
    public function get_where ($where) {
        $this->where = $where;
        return $this->get_all();
    }
}
```