HEALTHY HOME

by

SOWMYA MOTHEKANI

B.E., OSMANIA UNIVERSITY, INDIA, 2014

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Mitchell L. Neilsen

Copyright

SOWMYA MOTHEKANI

2016

Abstract

Every home has the challenge of matching its grocery to family member's demands. How well the home maker manages this challenge has a major impact on the food wastage and money spent on buying those. Any typical family with 4 people will spend at least $1500 per month for the groceries and the same family waste around $1000 worth of food in a year. These families don't know the overall environmental impact of food waste which is piling up in landfills. From the survey done by Environmental Protection Agency, this food waste accounts for 20% of landfill waste.

Healthy Home(HH) Android application helps the user to get the re-order point of the groceries based on the quantity remaining. The user will input the entire grocery list which he/she has purchased and the daily quantity consumed. The application will take this as the input and give the recommendation of what to cook and how many calories that give to your body. It can distinguish and let you know by what time the food should be consumed based on the Best by date or whether it is Perishable/Non-perishable. This application will recommend you the ROP (Reorder Point) and SS (Safety Stock) which any household has to carry based on the DDLT (Demand During the Lead Time). We are using the Supply Chain Industry standard formula to calculate the SS (Safety Stock) and ROP (Reorder Point).

ROP = DDLT+ SS

*SS = 2 Days' worth of Food (Defaulting) Instead of going with the Industry standard formula.

The user is provided with the database which consists of the standard grocery list with their calories information. The user is also provided with the standard cuisines (recipes) information in place which helps us in building the recommendation systems used to suggest recipes for users.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my major professor, Dr. Mitch Neilsen, for his guidance throughout my project development. He gave me a great level of independence while I am developing this project. His knowledge in the technical details of this project helped me to a great extent. I would like to thank my committee members, Dr. Daniel Andresen and Dr. Torben Amtoft for serving on my committee and also for their constant support throughout my Master's program by offering valuable knowledge. I am honored to have the privilege of taking courses under the professors of the CIS department who helped me dig a big deeper into the depths of the Computer Science with this Master's program at K-State. I would also like to thank the members of the CIS department staff who were helpful and supporting at every step during my Master's program at K-State.

I would also thank my friends who have been a constant source of support and my brothers, Revanth Mothekani and Yeshwanth Mothekani for being with me through thin and thick and for being the constant source of encouragement and motivation. I would finally like to thank my parents, Mr. Rajender Mothekani and Mrs. Sunitha Mothekani, for being my biggest strength at every stage of my life.

Chapter 1 - INTRODUCTION

1.1 Project Description

Healthy Home(HH) Android Application helps the user to get the re-order point of the groceries based on the quantity remaining. The user will input the entire grocery list which he/she has purchased and the daily quantity consumed. The application will take this as the input and give the recommendation of what to cook and how many calories will that give to your body. It can distinguish and let you know by what time the food should be consumed based on the Best by date or whether it is Perishable/Non-perishable. This application will recommend you the ROP (Reorder Point) and SS (Safety stock) which any household has to carry based on the DDLT (Demand during the Lead Time). We are using the Supply Chain Industry standard formula to calculate the SS (Safety Stock) and ROP (Reorder Point).

ROP = DDLT+ SS

*SS = 2 Days' worth of Food (Defaulting) Instead of going with the Industry standard formula.

The user is provided with the database which consists of the standard grocery list with their calories information. The user is also provided with the standard cuisines (recipes) information in place which helps us in building the recommendation systems used to suggest the user with recipes.

1.2 Motivation

Every home has the challenge of matching its grocery to family member's demands. How well the home maker manages this challenge has a major impact on the food wastage and money spent on buying those. Any typical family with 4 People will spend at least $1500 per month for the groceries and the same family waste around $1000 worth of food in a year. These families don't know the overall environmental impact of food waste which is piling up in landfills. As, per the survey done by Environmental Protection Agency this food waste round up for 20% of landfills. To avoid this wastage this android application provides an easily accessible and convenient way.

Chapter 2 - Android SDK

2.1 Introduction

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google [3]. Android gives you a world-class platform for creating applications. Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects. There are various Getting Started, Reference Guides, developer SDK, API documentation, and design guidelines available online that will give everything one needs to start developing apps.

2.2 Background

Android Operating System is a Linux based Operating System which can be used on smartphones and tablets. It is initially developed by Android Inc. and later acquired by Google. It is an open source development platform powered by a modified Linux 2.6 Kernel [3].

2.3 Android Architecture

Android operating system is a stack of software components which is roughly divided into six sections and four main layers as shown in the architecture diagram(Fig.2.1).

- Linux Kernel - Android is built up on the Linux Kernel. Linux is already being used extensively from so many years and its kernel had received so many security patches. Linux Kernel provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of for a mobile computing environment, the Linux kernel provides Android with several key security features, including, a user-based permissions model, Process isolation, Extensible mechanism for secure IPC and the ability to remove unnecessary and potentially insecure parts of the kernel.

- Hardware Abstraction Layer (HAL): It just gives Applications direct access to the Hardware resources.

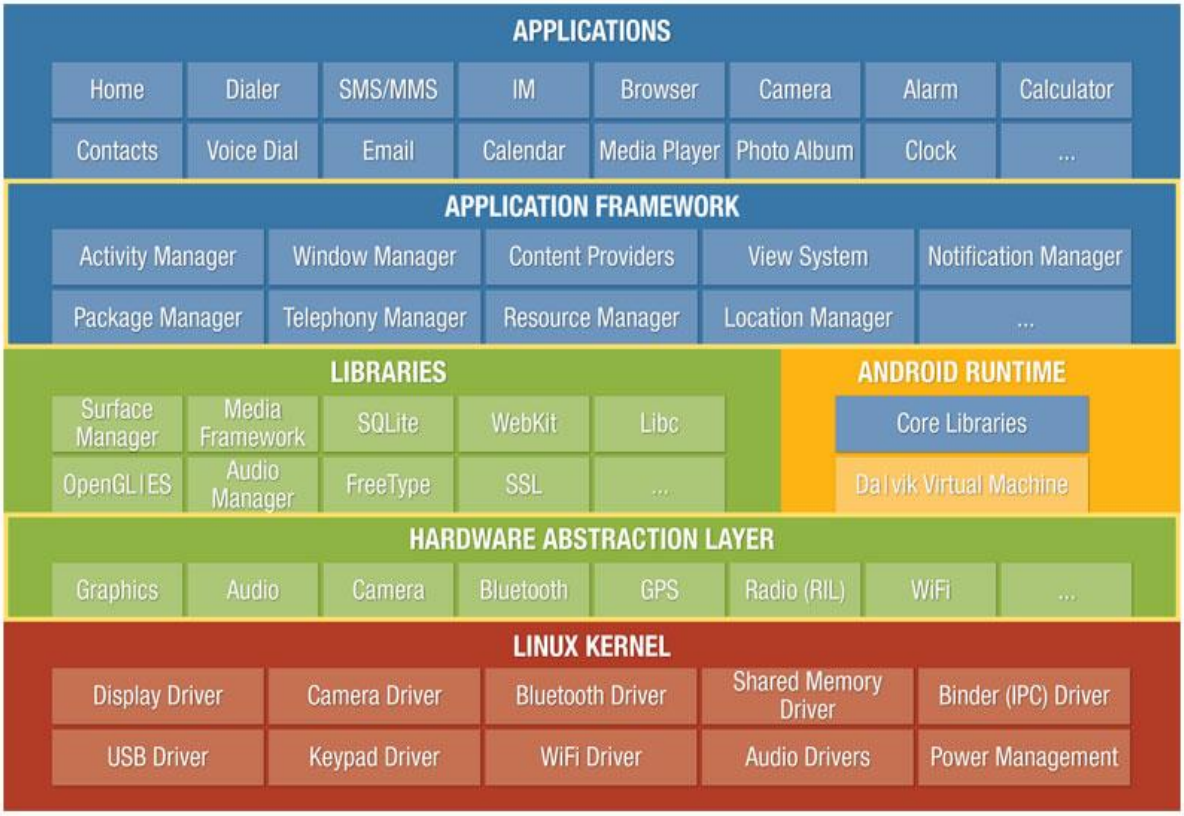

Figure 2-1 Android Architecture [1]

- Libraries: They include Android Runtime and Dalvik. The libraries shown in the image are very necessary without which application will not run like Webkit library is used for browsing the web, SQLite library is used for maintaining SQL database and so on.
- Android Runtime(ART): It is an alternative to Dalvik Virtual Machine which has been released with Android 4.4 as an experimental release, in Android Lollipop (5.0) it will completely replace Dalvik Virtual Machine. Major change in ART is because of Ahead-of-time(AOT) Compilation and Garbage Collection. In Ahead-of-time(AOT) Compilation, android apps will be compiled when user installs them on their device whereas in the Dalvik used Just-in-time(JIT) compilation in which byte code are compiled when user runs the app. Dalvik Virtual Machine which is specifically designed by Android Open Source Project to execute application written for Android. Each app running in the Android Device has its own Dalvik Virtual Machine.

- Application Framework: The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.
- Applications - One will find all the Android application at the top layer. One will write application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

2.4 Data Storage

Android provides several options for you to save persistent application data. The solution choice depends on your specific needs, such as whether the data should be private to your application or accessible to other applications and how much space your data requires.
The data storage options are the following:

- Shared Preferences - Store private primitive data in key-value pairs.
- Internal Storage - Store private data on the device memory.
- External Storage - Store public data on the shared external storage.
- SQLite Databases - Store structured data in a private database.
- Network Connection - Store data on the web with your own network server.

This application primarily uses SQLite Databases as a storage option. This application uses Shared Preferences as a storage option for login details as all the preferences can be stored as primitive data in key-value pairs.

2.5 Connectivity

Android provides rich APIs to let your app connect and interact with other devices over Bluetooth, NFC, Wi-Fi in addition to standard network connections. Most network-connected Android apps will use HTTP to send and receive data. Android includes two HTTP clients:

- HttpURLConnection
- Apache HTTP Client

Both support HTTPS, streaming uploads and downloads, configurable timeouts etc.

2.6 Media Playback

The Android Multimedia Framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw resources), from standalone files in the file system, or from a data stream arriving over a network connection, all using MediaPlayer APIs.

2.7 AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.)

This application uses this class to perform network operations as these operations cannot be done directly on main thread.

Chapter 3 - Requirement Analysis

For requirement analysis requirements were elicited through requirements gathering by analyzing the need of the user obtaining a clear set of unambiguous requirements and recording those requirements.

3.1 Requirement Gathering

As this is an application for the common user, the graphical design that is the front end is main aspect and it should be taken care that it will be easy to use for any user and navigation shouldn't be any confusing. Main requirements were gathered after brainstorming with couple of my friends who provided me an insight that I should allow the allow to enter the quantity consumed in the cups and slices rather than gallons and loafs. The main requirements of this application are

- User can view at a week data of the food consumed and the bar graph of calories received.
- View the list of groceries entered, daily usage of groceries along with the remaining list.
- Provide the user with notifications at breakfast, lunch and dinner.
- Provide the user with an ability to add ratings to the recipe and also allowing the user to add recipe.
- Provide the user with notifications about the expiry date.
- Provide the user with a notification of about the re-order point of the category.

3.1.1 Functional Requirements

- *Grocery List Update:* This module allows the user to add or edit the groceries that are brought and populate the list based on same on which a search operation scan be performed.
- *Remaining List check:* This module allows the user to see the remaining list of recipe after the consumption.
- *Daily-Usage List Update:* This module allows the user to enter the daily usage of groceries and populate the list with the same.
- *Calorie-Graph check:* This module provides the user with the amount of calories consumed on that particular day for that week.

- *Re-order point check:* This module allows the user to check for the re-order points of the individual groceries.
- *Recipe check:* This module allows the user to Check for the recipe based on the ingredients and also allows the user to add the recipe.
- *Settings:* This module allows the user to set alarms for the Breakfast, Lunch and Dinner.

3.1.2 Non-Functional Requirements

Non-functional requirements describe aspects that are not directly related to the functional behavior of the system which mainly include user interface and human factors such as:

- The user interface is designed such that the user requires very little, if not no knowledge of Android to access this application.
- The functionalities of each screen and navigation through the screens of the application is described in the help module.
- This application is user friendly and very interactive.
- This application prompts error messages if the user gives incorrect input or tries to access the functionalities of the application incorrectly.
- User is required to have basic knowledge of English to access the application.

3.2 Requirement Specification

3.2.1 Software Requirements

Software Requirements for developing the application:

Operator System: Windows 8 or Mac OS
Language: Android Studio, Java
Database: MySQL
Tools: Genymotion
Technologies: Java, MySQL, Android, XML
Debugger: Genymotion Simulator, Android mobile device (Android version 4.0.4)

Software Requirements for running the application:

Framework: Android SDK Version 4.0.4 or higher

Network Required: Mobile network and Internet (cellular or Wi-Fi)

3.2.2 Hardware Requirement

Hardware Requirements for developing the application:

Processor: i3 or higher

RAM: 512 MB

Space on disk: 250 MB or higher

Hardware Requirements for running the application:

1 – Android phone with version 4.0.4 or higher

3.3 Feasibility Analysis

3.3.1 Economic Feasibility

HH application is economically feasible because it requires an android device with Android Studio 4.0.4 or higher which can be downloaded for free. However, in order to download the application, the users must have a Wi-Fi or a cellular network. Hence, it is economically feasible as a Wi-Fi or a cellular connection are available easily at economic prices.

3.3.2 Technical Feasibility

Developing this application needed a system to install Android SDK and develop the application and a device to test it. Also the testing can also be done on an emulator. This application has been tested on Nexus 5 and Genymotion emulator. Hence it is technically feasible.

Chapter 4 - System Design

System design [4] is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The system design depicts the overall product architecture, the subsystems that compose the product, and the manner in which subsystems are allocated to processors, the allocation of classes to subsystems, and the design of the user interface. The system design is reviewed by examining the object-behavior model developed during object-oriented analysis and mapping required system behavior against the subsystems designed to accomplish this behavior.

UML [5] is a standard visual modeling language widely used in system design for modeling business and similar processes, analysis, design, and implementation of software-based systems. UML diagrams can be broadly classified into Structural and Behavioral diagrams. The static aspects of the system which forms the main structure are represented by classes, interfaces, objects, components, and nodes. The following are the structural diagrams-

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioral diagrams capture the dynamic aspect like changing/ moving parts of the system. The following are the behavioral diagrams-

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

4.1 Use Case Diagram

A Use case diagram is a set of scenarios in which our system or application interacts with people, organizations, or external systems. The use case is an external view of the system that represents some action the user might perform in order to complete a task. The main components of a

use case diagram are use cases, actors and their relationships. In its simplest form, a use case can be described as a specific way of using the system from a user's (actor's) perspective. It is used to describe a function provided by the system that yields a visible result for an actor. An actor describes any entity that interacts with the system.

The following Unified Modeling Language (UML) diagram offers a way to visualize a app's architectural blueprint, including elements such as: different functionalities, individual activities, how they can interact with other activities, how the app will run, how user interact with app, external user interface etc.

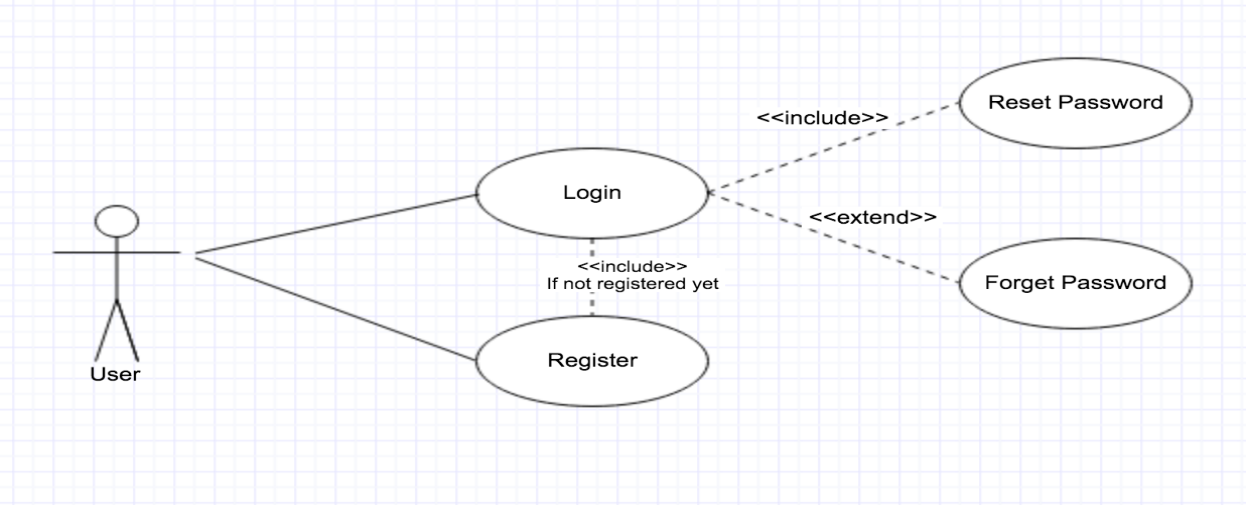

Figure 4-1 Use-Case Diagram(1)

Actors:

In our model, the user is the only actor who interacts with the system.

Use cases:

- Login- This use case denotes the sequence of actions required for the user to get access into system.
- Register- This use case denotes the sequence of actions required for the user enroll into the system.
- Forgot Password- This use case denotes the sequence of actions required for the user to know the password if forgotten.

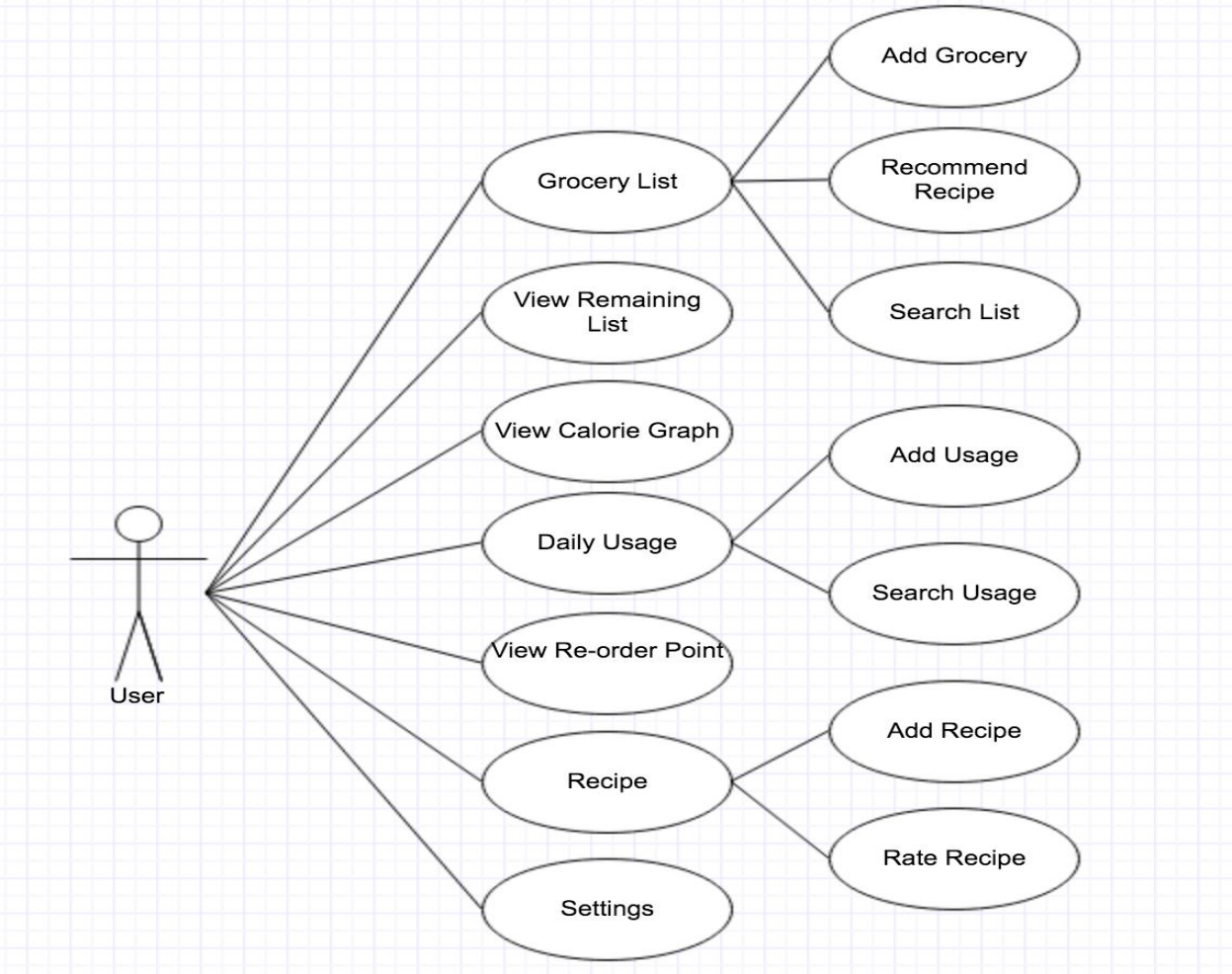

Figure 4-2 Use-Case Diagram(2)

Use cases:

- Grocery List- This use case denotes the sequence of actions required for the user to add a grocery or to get recommendation of recipes from groceries or search for them.
- Remaining List- This use case denotes the sequence of actions required for the user to check the remaining grocery list.
- Calorie graph- This use case denotes the sequence of actions required by the user to check the calories earned by user for that particular week.
- Daily Usage- This use case denotes the sequence of actions required by the user to add the daily consumed groceries.
- Re-order Point- This use case denotes the sequence of steps required by the user to know the re-order groceries.

- Recipe- This use case denotes the sequence of steps required by the user to add the recipe into system.
- Settings- This use case denotes the sequence of steps required by the user to set alarm and to get notifications based on them.

Associations: Association exists whenever an actor is involved in an interaction described by the use case.

- **Associations of User:**

  The user is associated with Login, Register, Grocery List, Calorie Graph, Re-order point, Recipe and Settings.

4.2 Class Diagram

The class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of the application by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. The following figures represents the class diagram of this system.

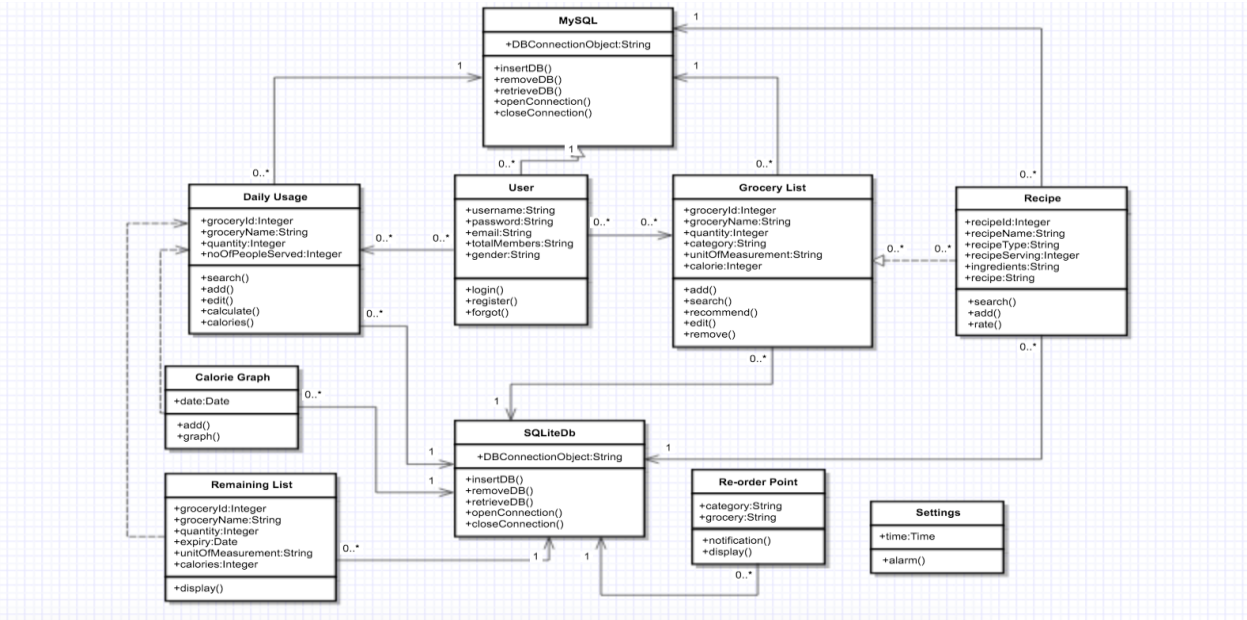

Figure 4-3 Class Diagram

4.3 Activity Diagram

Activity diagrams describe the workflow behavior of the system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by show activities that are conditional or parallel.

A fork is used when multiple activities are occurring at the same time. The branch describes what activities will take place based on a set of conditions. All branches at the same point are followed by a merge to indicate the end of the conditional behavior started by that branch.

The following tools are used on the activity diagram toolbox to model activity diagrams:

- Decisions: A decision represents a specific location in activity diagram where the work flow may branch based upon guard conditions.
- States: A state represents a condition or a situation during the life of an object during which it satisfies some condition or waits for some event.
- Transactions: A state transition indicates that an object in the source state will perform certain specified event occurs or when certain conditions are satisfied.
- Start states: A start state (also called initial state) explicitly shows the beginning of a work flow.
- End states: An end state represents a final state or terminal state.

Activity Diagram for Healthy Home

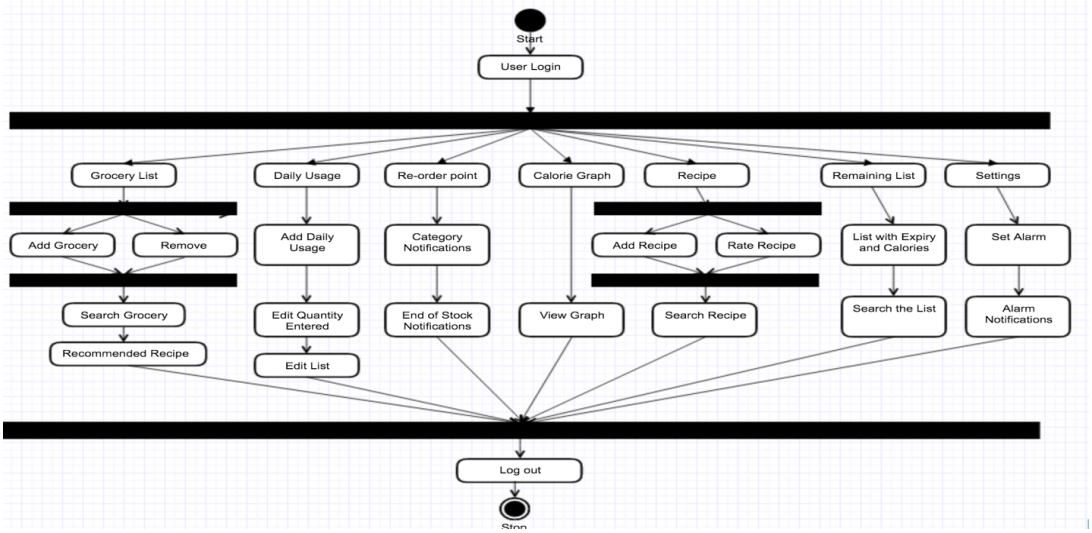

Figure 4-4 Activity Diagram

Chapter 5 - Android Framework Components

Android apps are written in the Java programming language. Android SDK provides us all the required API libraries and developer tools to build, test and debug apps for Android. The Android SDK tools compile your code—along with any data and resource files—into an APK: An Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app. Healthy Home application has been developed in Android studio Integrated Development Environment (IDE) with the Android Developer Tool(ADT) plugin. HH application can be run on an emulator or generated .apk file can be installed on an android smartphone or tablet.

5.1 App Manifest

Every application must have an AndroidManifest.xml [6] file in its root directory. The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code. Among other things, the manifest does the following:

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of. It names the classes that implement each of the components and publishes their capabilities. These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

The following is the AndroidManifest.xml file present in this application:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.healthy.home"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/_AppTheme" >
        <activity
```

```
    android:name="com.healthy.home.MainActivity"
    android:label="@string/app_name" >
</activity>
<activity android:name="SplashScreenActivity" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name="LoginActivity" >
</activity>
<activity android:name="RegisterActivity" >
</activity>

<receiver android:name="com.healthy.alaram.BreakFastAlaramReceiver" >
</receiver>
<receiver
    android:name="com.healthy.alaram.SampleBootReceiver"
    android:enabled="false" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" >
        </action>
    </intent-filter>
</receiver>

<service android:name="com.healthy.alaram.SampleSchedulingService" />
<receiver android:name="com.healthy.alaram.DinnerAlaramReceiver"></receiver>
<receiver android:name="com.healthy.alaram.LunchAlaramReceiver"></receiver>
<activity android:name="AddReceipe"></activity>
<activity android:name="RecommandedReciepe"></activity>
```

```
  </application>
</manifest>
```

According to the manifest file, the minimum sdk version for this application is 11. Therefore, the devices below sdk version 11 that is android Honeycomb will not be able to run this application. Different activities and their intents are mentioned in this file for HH.

5.2 Activity

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen. An application usually consists of multiple activities that are loosely bound to each other.

Managing the lifecycle of your activities by implementing callback methods is crucial to developing a strong and flexible application. When an activity transitions into and out of the different states, it is notified through various callback methods. All of the callback methods are hooks that you can override to do appropriate work when the state of your activity changes. The following skeleton activity [7] includes each of the fundamental lifecycle methods:

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
```

```
protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be
"paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
                    // The activity is about to be destroyed.
      }
      }
```

The following figure illustrates these loops and the paths an activity might take between states. The rectangles represent the callback methods you can implement to perform operations when the activity transitions between states.

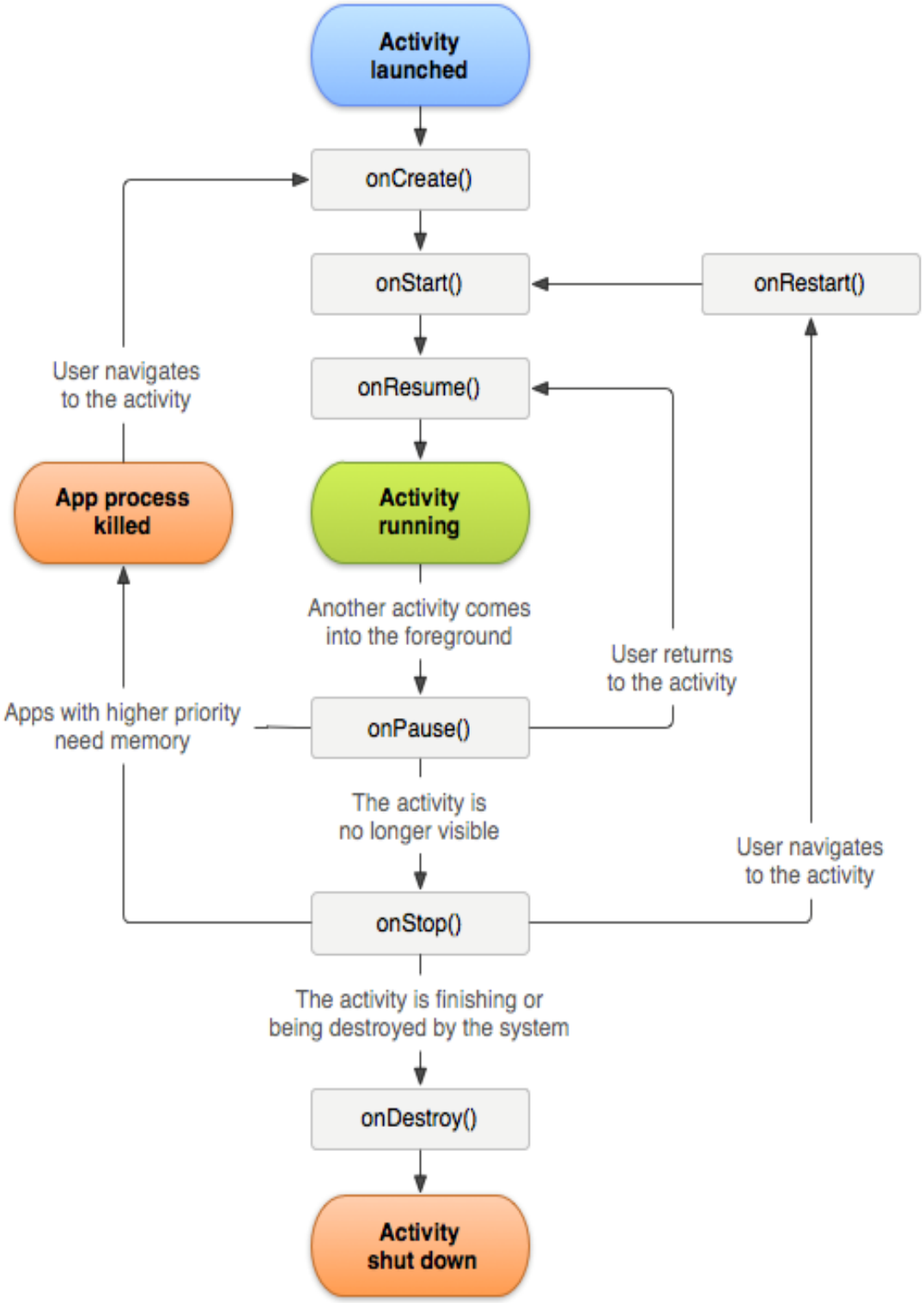

Figure 5-1 Activity Lifecycle [7]

5.3 Main Activities Involved in Healthy Home

5.3.1 LoginActivity

This activity mainly deals with the login page. The onClick() function mainly deals with forget password and register buttons functionality. The onValidiationSuccess() function determines what must be done once the correct username and password credentials are entered by the user. The onSuccess() function verifies the username and password entered to match with those stored in database using user service. The onException() function sets an error dialog to be displayed when the credentials entered by the user does not match those in the database.

5.3.2 RegisterActivity

This activity mainly deals with the registration of the user. We use methods like onCreate(), onValidationSuccess() are used to take the details entered by the users and then to verify that the details are unique and does not exists in the database. We use the onException() method to raise errors if any of the details entered by the user are not matching the format set. This displays the error message as a toast.

5.3.3 AddRecipeActivity

This activity mainly deals with the user adding recipe to the application. The onClick() is used to know whether the values are entered or not if not entered error is given. Then onPostExecute() is used to know the output or result i.e. if the recipe is successfully added or not. Then all the values are added to the database by using dbHelper().

5.3.4 RecommendedRecipeActivity

This activity mainly deals with the user getting the recommended Recipes based on the grocery list. Initially for which a connection is established with the database and then the groceries list is being evaluated and stored as an array and then the recipe is searched for the array elements then the list is given.

5.3.5 AlarmActitvity

This activity mainly deals with the user getting notified for setting an alarm. This is done by using the enabling alert signals to users. The triggering events play a major role in this.

5.4 Intent

An Intent is a messaging object you can use to request an action from another app component. There are following two types of intents supported by Android:

- *Explicit Intents:* These intents are going to be connected internal world of application, suppose if you want to connect one activity to another activity, we can do this quote by explicit intent, below image is connecting first activity to second activity by clicking button.
- *Implicit Intents:* These intents do not name a target and the field for the component name is left blank. Implicit intents are often used to activate components in other applications.

Although intents facilitate communication between components in several ways, there are three fundamental use-cases:

- To start an activity
- To start a service
- To deliver a broadcast

5.5 Services

A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it. A service is implemented as a subclass of Service.

5.6 Layout Inflator

Layout Inflator android component helps loading the layout XML file into its view objects such as ProgressBar, TextView etc. It is used in conjunction getLayoutInflater() or

getSystemService(String) to retrieve a standard LayoutInflater instance which is already hooked up to the current context.

"LayoutInflater li = (LayoutInflater) context.getSystemService

(Context.LAYOUT_INFLATER_SERVICE);

5.7 Content Providers

A content provider manages a shared set of app data. One can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access. As such, any app with the proper permissions can query part of the content provider to read and write information about a particular person. Content providers are also useful for reading and writing data that is private to your app and not shared. A content provider is implemented as a subclass of ContentProvider and must implement a standard set of APIs that enable other apps to perform transactions. For more information, see the Content Providers developer guide.

5.8 Broadcast Receiver

A broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Apps can also initiate broadcasts—for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work. For instance, it might initiate a service to perform some work based on the event. A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object.

Chapter 6 - Databases

SQLite [8] is an open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. To manage our own databases, we are provided with the classes from Android.database.sqlite package. The MySQL [9] Database powers the most demanding Web, E-commerce and Online Transaction Processing (OLTP) applications so that is why it is used.

Healthy Home application uses the MySQL as the backend and SQLite provided by the Android as front-end. MySQL is used as a database at the webserver and PHP is used to fetch data from the database. Our application will communicate with the PHP page with necessary parameters and PHP will contact MySQL database and will fetch the result and return the results to us.

Main tables that are needed by the HH application include the login table wherein the user details are stored when he/she registers. The second table required is for storing the rating of recipe given by the user and next one is for storing the daily usage and the other one for storing the grocery list given by user. Finally, a table to store the recipes. All of these tables are populated by the user and then some of them can be edited by user such as grocery list.

Chapter 7 - Graphical User Interface

The front end of the HH application was developed using XML. Each and every component is briefly described in the following sections.

7.1 Login Page

Figure 7.1 shows the login screen of the HH application. A login page of the application is the screen asking your credentials to login to this particular application. The login page also contains “Forgot password” option through which a user can easily retrieve his/her password using the or respective username.

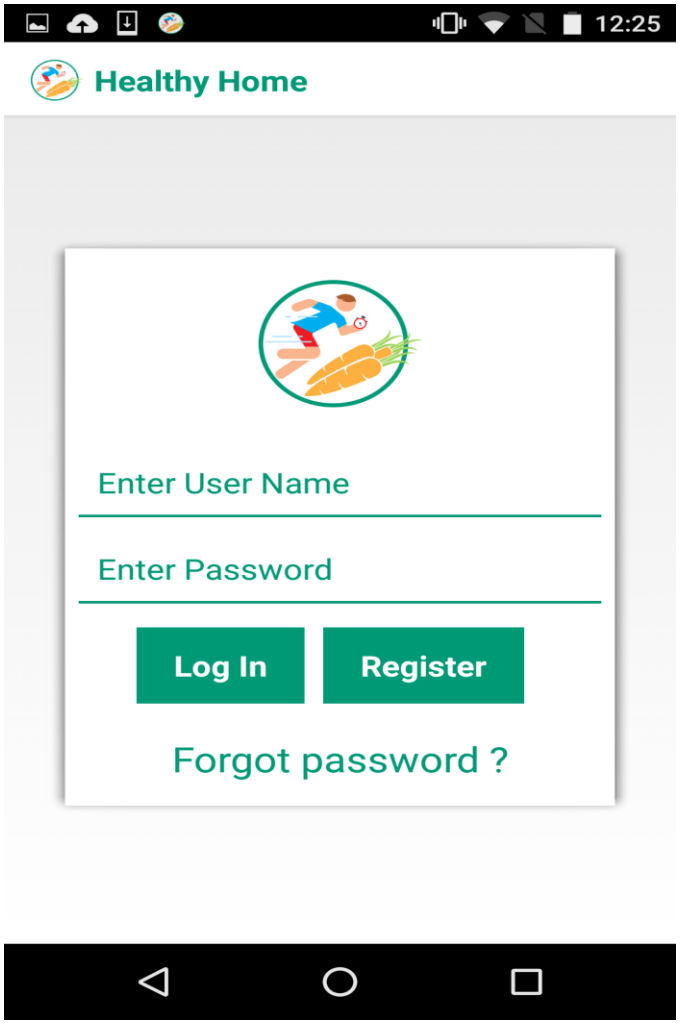

Figure 7-1 Login Page

7.2 Registration Page

Healthy Home application contains a login and registration process in order to authenticate a user. The user can enter his/her details which are saved in the database, and thereby authenticating the user into the respective application and use it on accordingly. Figure 7.2 shows the Registration page of HH application including username which is selected by the user, the user's e-mail address, a password containing 8 alpha numeric characters, a confirm password field match the password and total members in a family.

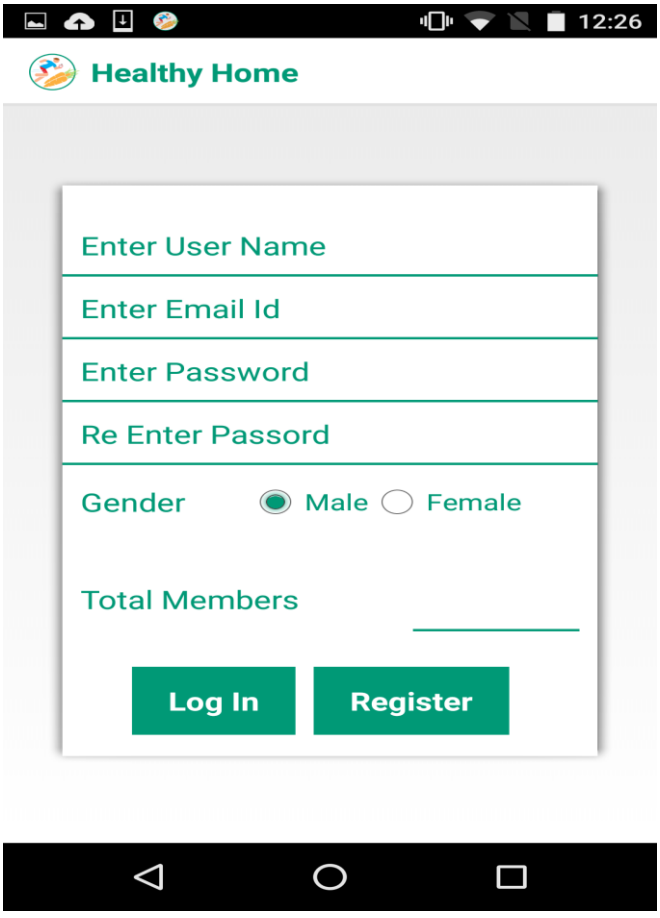

Figure 7-2 Register Page

7.3 Home Screen

Figure 7.3 shows the Home Screen for the HH application. Under this segment there are seven different options: Grocery List, Daily Usage, Remaining List, Calorie graph, Re-order point, Recipe, Settings. The logout option is located at the bottom of this fragment.

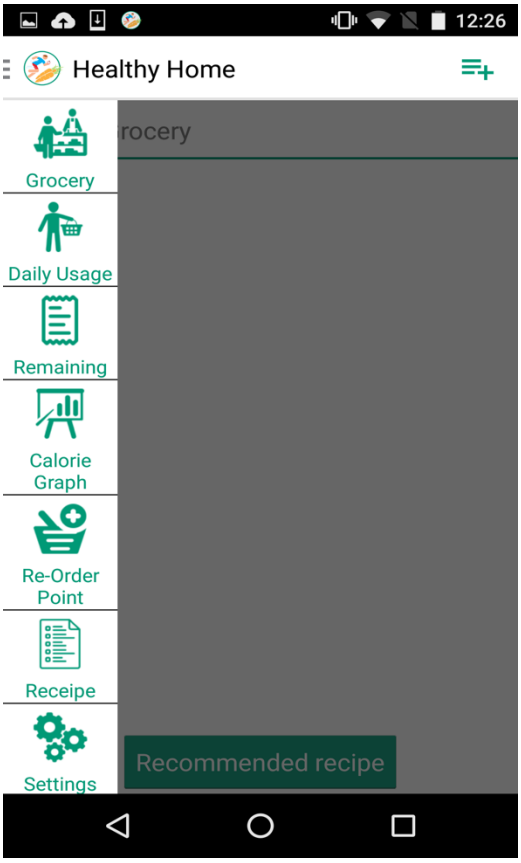

Figure 7-3 Home Screen

7.4 Grocery List

When the user selects the Grocery List option a screen appears as shown in Figure 7.5 which contains the grocery list entered by the user. On the top right corner of the screen has an add symbol which when selected opens a dialogue box as shown in Figure 7.4 which is used to add a grocery item to list by providing name, quantity and category. Once the add grocery is completed the user is redirected to the updated grocery list. The user is given recommendations of recipes based on the grocery list entered in the Recommended Recipe button.

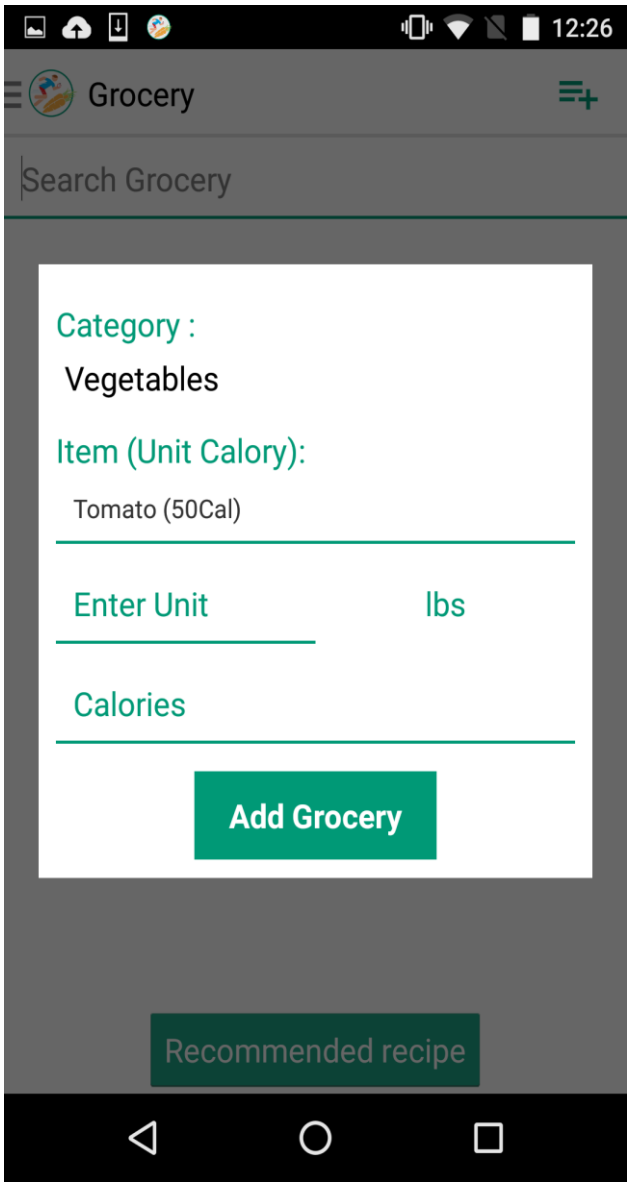

Figure 7-4 Add Grocery

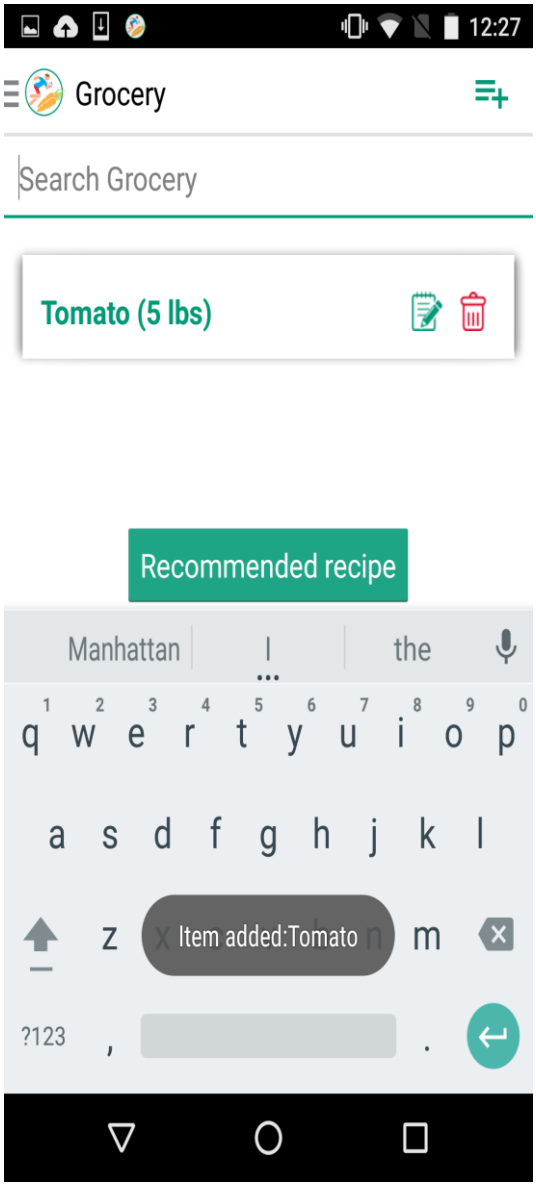

Figure 7-5 Grocery List

7.5 Daily Usage

When the user selects the daily usage option a screen appears as shown in Figure 7.6 which contains the list of groceries already consumed by the user in that particular week. On the top right corner of the screen has an add button which when selected opens a dialogue box as shown in Figure. 7.7 which is used to add the daily quantities consumed by each individual person and the total number of people served. After entering this the respected quantities are subtracted from the list and the Remaining List is updated. Once the grocery usage is added the user is redirected to the updated Daily Usage List.

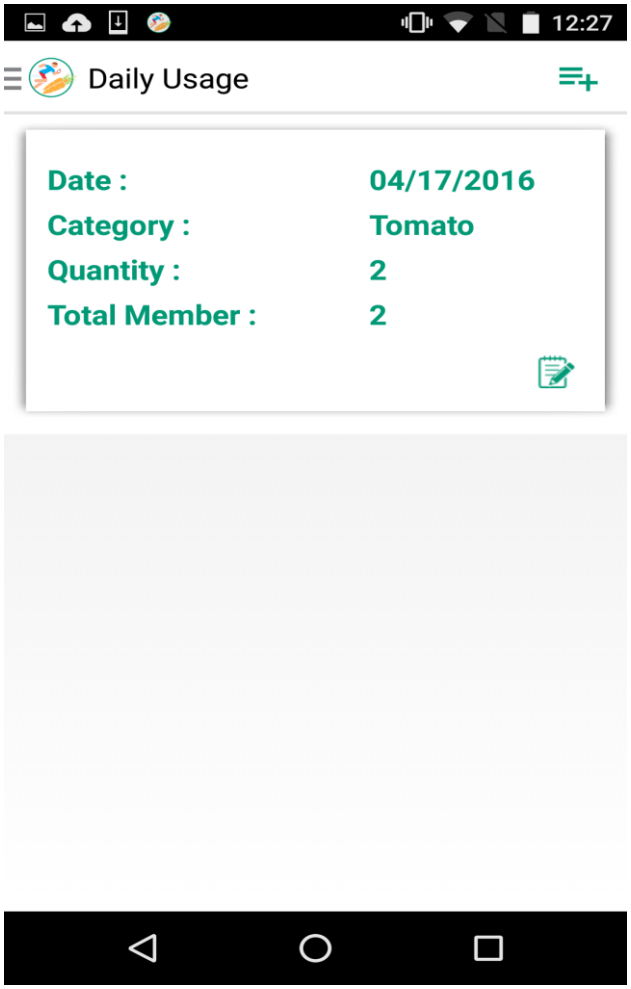

Figure 7-6 Daily Usage List

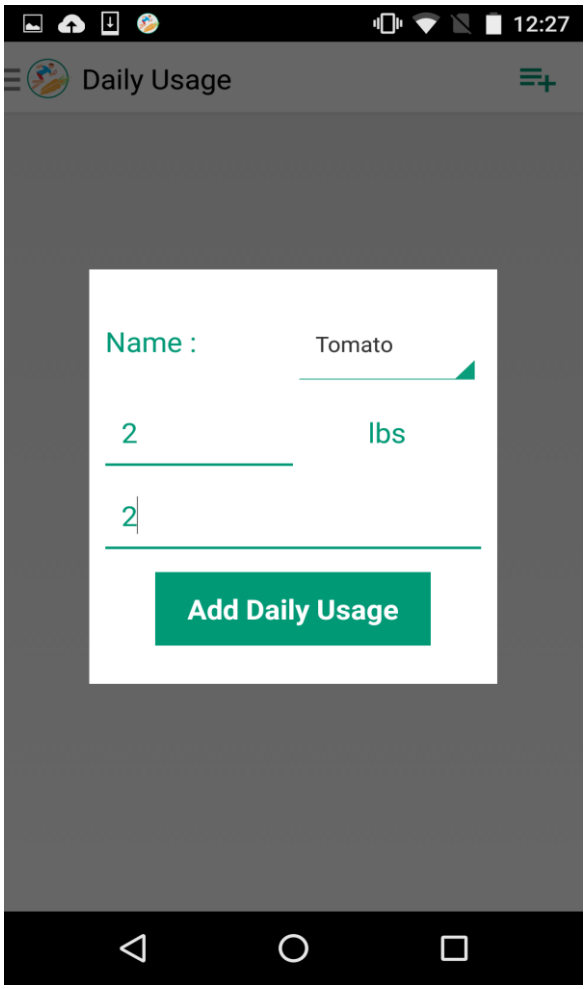

Figure 7-7 Add Daily Usage

7.6 Remaining List

When the user selects the Remaining List option a screen appears as shown in Figure 7.8 which contains the Remaining List of groceries along with the respective expiry dates and the calories that can be earned by consuming each unit.

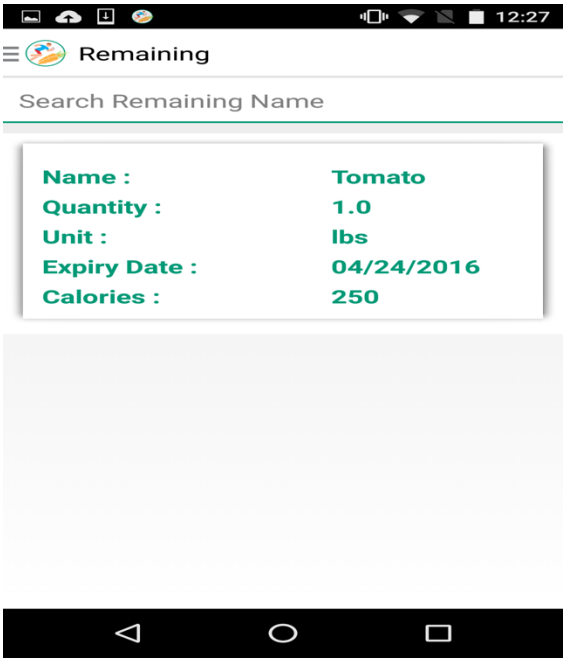

Figure 7-8 Remaining List

7.7 Calorie graph

When the user selects the Calorie graph option a screen appears as shown in Figure 7.9 which consist of the bar graph with the calories earned by each individual in that particular week categorized by dates.

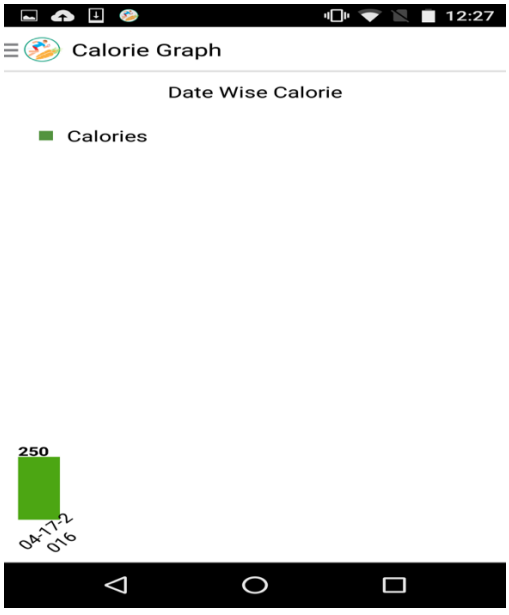

Figure 7-9 Calorie Graph

7.8 Re-order point

When the user selects the Re-order point option a screen with the list of re-order groceries and the notifications based on the category that are almost finished and needed to be refilled appears as shown in figure based on the user given list of the groceries.

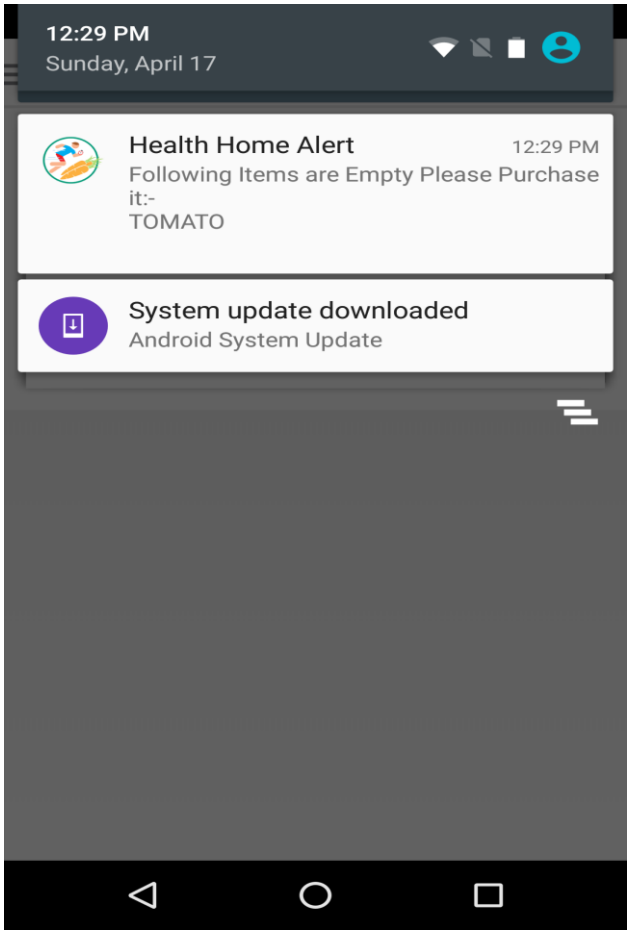

Figure 7-10 Empty Notification

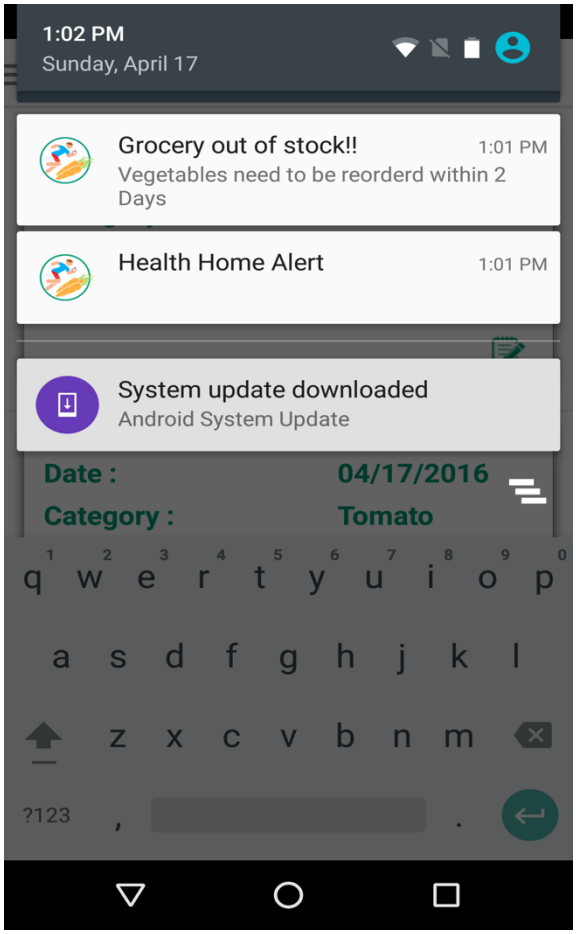

Figure 7-11 Category Notification

7.9 Recipe

When the user selects the recipe option then a screen appears as shown in Figure 7.12 in which the user is given a list of recommendations based on the items present in the Grocery List. The user can search for the recipes based on the ingredients along with that on the top right of the screen an add symbol is provided so the user can add his/her recipe. Each of the recipe comes with the Rating button which when chosen is opens up a dialogue box where the user can rate the recipe.

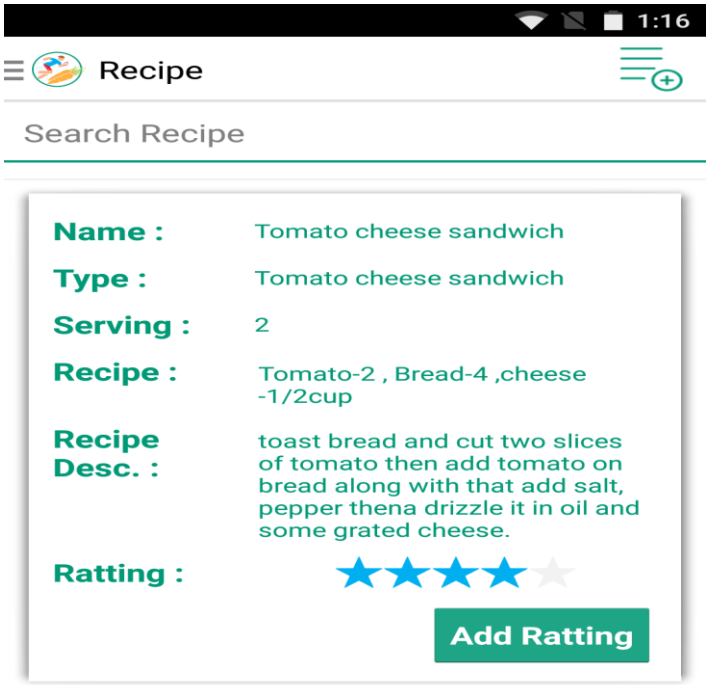

Figure 7-12 Recipe Page

7.10 Settings

When the user selects the Settings option then a screen appears as shown in Figure 7.13 in which the user can set the notifications for breakfast, lunch and dinner timings.

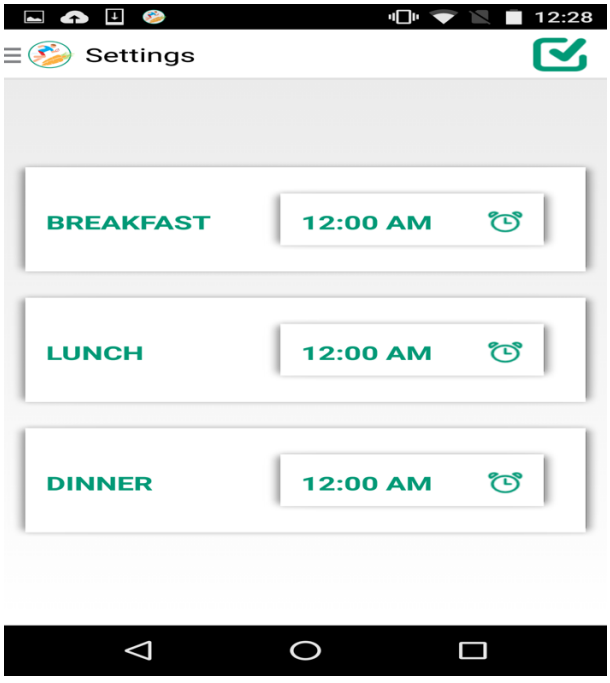

Figure 7-13 Settings

Chapter 8 - Testing

The primary goal behind testing an application is to identify the defects in that application. By verifying if the system met all the requirements such as functionality, reliability and usability. Testing also confirms, validates if the product development was in accordance with user requirements leading to product improvements. The Android SDK provides most of the tools [10] that are needed to debug applications. Dalvik Debug Monitor Server (DDMS) is a graphical program that communicates with devices. LogCat is integrated into DDMS, and outputs the messages that can be printed out using the Log class along with other system messages such as stack traces when exceptions are thrown. Various types of testing are performed on HH application and are explained in the following sections.

8.1 Unit Testing

Unit testing is a software testing method by which individual units of the application are tested separately. In this application different screens and the related functionalities are tested. Unit Testing is performed on HH and the results are tabulated below in Table 8-1.

8.1.1 Unit Test Cases

S. no	Test Case	Expected Result	Pass/Fail
1	On load of Home screen	Display Login page	Pass
2	On click of Registration	Display the screen with the registration details of the user.	Pass
3	On click of Forgot Password link	Display a dialog box which asks for your username to reset the password.	Pass
4	On click of Grocery List menu option	Displays a list of Groceries along with the add button at the right hand corner of the screen which is used to add a grocery item to list based on the category.	Pass
5	On click of Daily Usage menu option	Displays a list of Groceries consumes as entered by the user for that particular date which can be done by using the add button located at the right hand corner of the screen.	Pass

6	On click of Re-order point	Displays a list of categories on which the groceries should be re-ordered.	Pass
7	On click of Remaining List	Displays a list of the remaining groceries along with the expiry date and calories earned of each unit.	Pass
8	On click of the Calorie graph option	Displays a graph of an individual for the entire week.	Pass
9	On click of the Recipe menu option	A search tab is appeared along with the list of recipes recommended based on the grocery list.	Pass
10	On click of Rating button on Recipe page	Opens a small dialog box for the user to rate the recipe	Pass
11	On click of Settings in menu option	A screen to set the alarms for the Breakfast, Lunch and Dinner appear.	Pass

Table 8-1 Unit Test Cases

8.2 Compatibility Testing

The Healthy Home application is installed on different android devices such as Nexus 5, LG G3 and HTC One M8. The application ran with the expected resolution. In order to provide proper image resolution, images are stored with various resolutions in folders hdpi, xhdpi and xxhdpi and so on.

8.3 Integration Testing

Integration testing is performed to ensure that all the components are working as expected after integrating all the components together. In this case, the user should be able to interact with the GUI without any problem and the data should be transferred without any glitches. Following tests are performed to ensure that the system is integrated as expected.

S. no	Test Case	Expected Result	Pass/Fail
1	Adding new Recipe to database	The app should be able to search for new recipe and display in recommendation.	Pass
2	Adding new Grocery to list	The app should be able to search for the new grocery and display it in the remaining list with the expiry date.	Pass

3	Adding Daily usage	Daily usage list should be updated along with the calorie graph.	Pass
4	Grocery list modification	Grocery item quantity entered can be modified.	Pass
5	Daily usage modification	Daily usage quantity entered can be modified.	Pass

Table 8-2 Integration Test Cases

8.4 Usability Testing

Three test subjects installed the HH application on their phone and tested different functionalities of the application. They suggested few exception handling cases. All the suggested changes are implemented in the final version of the application. User should make sure that the Internet on before the application is started.

8.5 Battery Consumption

The battery consumption has ben tested using a Nexus 6 device. The application was tested for the time where battery percentage started at 100% and reduced to 10%. In the first case the phone was used to perform normal operations like audio playback, voice calls, Whatsapp texting and voice calls. In the second case along with the normal operations, the HH application was running in the background. In both the cases, the phone was using Internet with Wi-Fi OR 4G/LTE Networks.

Application Running	**Time Taken for the battery to reduce to 10%**
Normal operations like audio playback, voice calls, Whatsapp texting and voice calls	127 minutes
HH application along with other normal operations	113 minutes

Table 8-3 Battery Consumption

Chapter 9 - Performance Profiling

Android Studio and the mobile device I have used provide profiling tools [11] to record and visualize the rendering, compute memory and battery performance of the application.

9.1 Rendering Analysis Tools

9.1.1 Debug GPU Overdraw

The Debug GPU Overdraw [12] shows how to visualize overdraw on the mobile device by color-coding interface elements based on how often they are drawn underneath. This helps in recognizing where the application might be doing more rendering work than necessary and hence can help in reducing rendering overhead. I did turn on the Debug GPU Overdraw option in developer options on the mobile device in which I have installed this application. The colors on the screen hints the amount of overdraw on the screen for each pixel.

- True color: No overdraw
- Blue: Overdrawn once
- Green: Overdrawn twice
- Pink: Overdrawn three times
- Red: Overdrawn four or more times

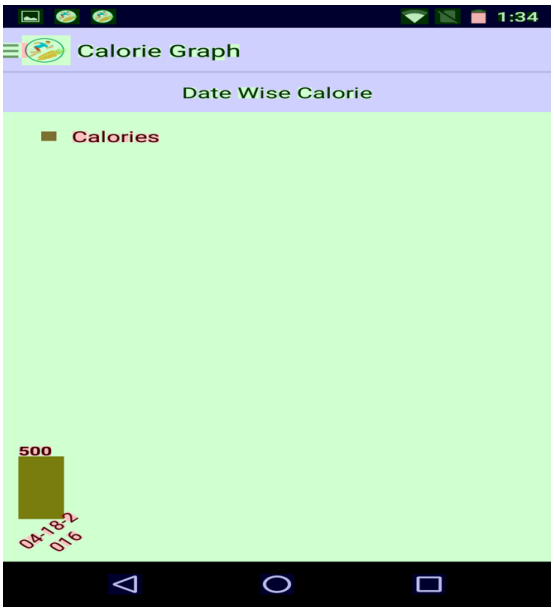

Figure 9-1 Debug GPU Overdraw for Calorie Graph

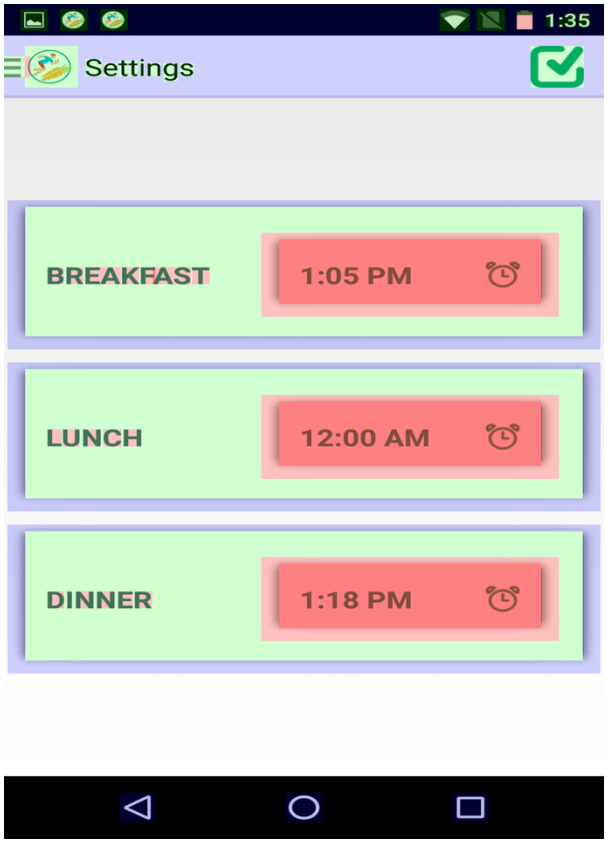

Figure 9-2 Debug GPU Overdraw for Settings

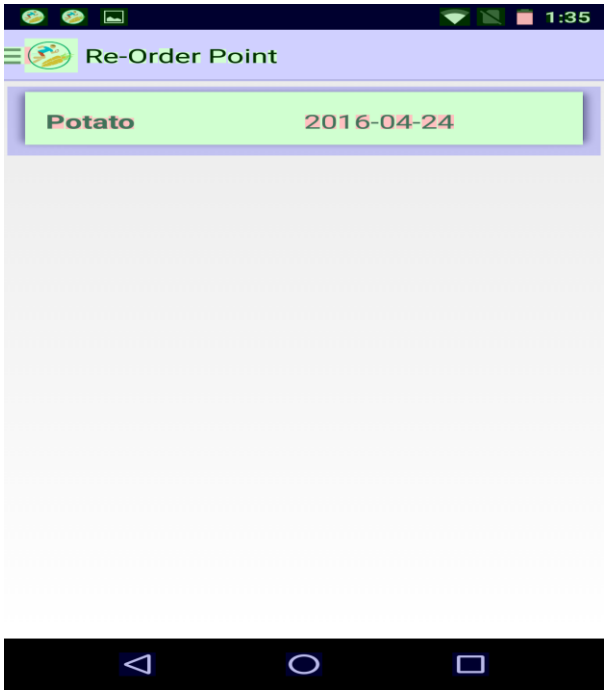

Figure 9-3 Debug GPU Overdraw for Re-Order Point

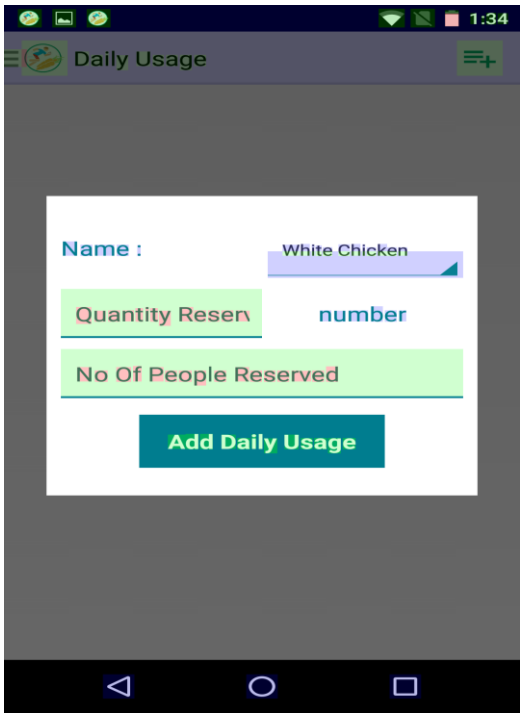

Figure 9-4 Debug GPU Overdraw for Daily-Usage

Figure 9-5 Debug GPU Overdraw for Grocery-List

The Healthy Home application has proved to have less GPU overdraw as there is no pixel with color red and a very few with color pink. There are few pixels with the green color implying the pixels are overdrawn twice which is common and cannot be avoided. The rest of the pixels on the screens are of the true color implying no overdraw.

9.1.2 Profiling GPU Rendering

Profile GPU Rendering gives a quick visual representation of how much time it takes to render the frames of UI window relative to the 16-ms-per-frame benchmark. It helps us estimate the UI performance against the 16-ms-per-frame target and helps us finding the spikes in frame rendering time associated with user or program actions. To enable this tool, I did turn on the Profile GPU Rendering and I have chosen the On Screen as bars to overlay the graphs on the screen of the mobile device.

The tool displays a graph with the horizontal axis showing time elapsing, and the vertical axis showing time per frame in milliseconds. The vertical bars shown up on the screen, appearing from left to right, graphs frame performance over time. Each vertical bar represents one frame of rendering. The green line marks the 16 millisecond target. Every time a frame crosses the green line, the application is missing a frame, and there might be a pause in animations. The graph has colored sections representing the phase of the rendering pipeline.

- The **green** line represents 16 milliseconds. Any time a bar pushes above this line, there may be pauses in the animations.
- The **blue** section of the bar represents the time used to create and update the View's display lists.
- The **purple** section of the bar represents the time spent transferring resources to the render thread.
- The **red** section of the bar represents the time spent by Android's 2D renderer issuing commands to OpenGL to draw and redraw display lists.
- The **orange** section of the bar represents the time the CPU is waiting for the GPU to finish its work.

The application shows good GPU Rendering. Except for some screens, almost all the vertical bars in all other screens are below the green line. The taller orange lines in these screens might be due to the fact that it takes CPU time to connect to internet.

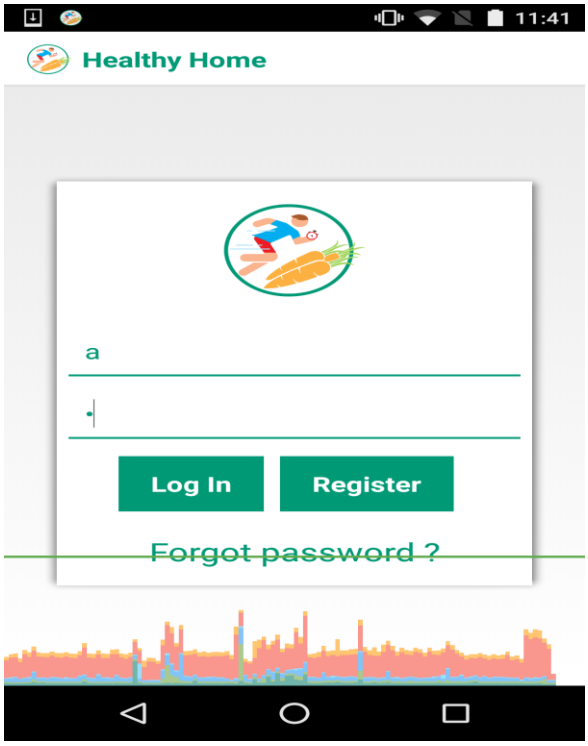

Figure 9-6 Profiling GPU rendering for Login Screen

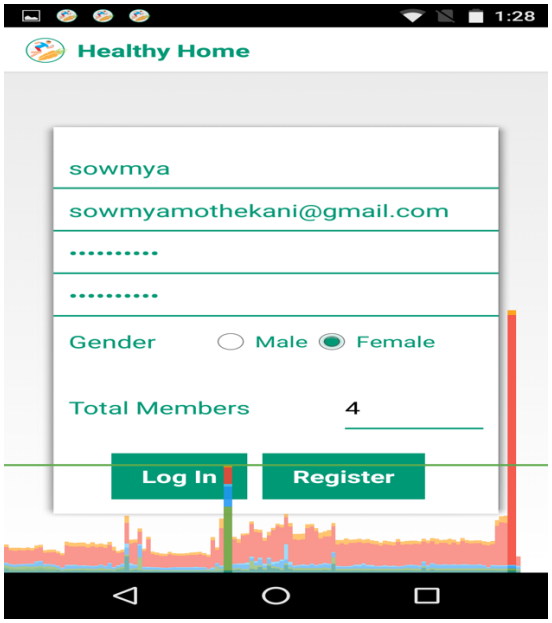

Figure 9-7 Profiling GPU rendering for Register Screen

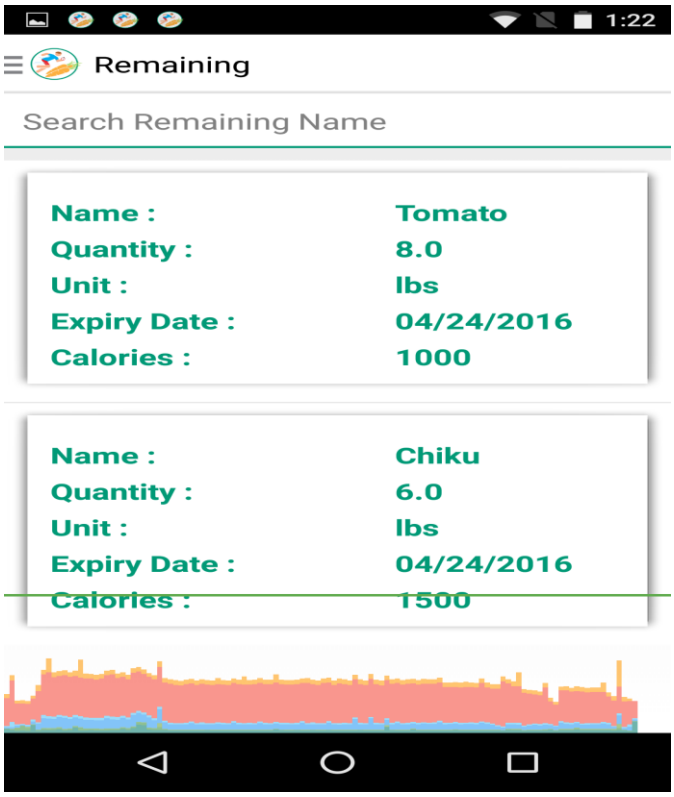

Figure 9-8 Profiling GPU rendering for Remaining List Screen

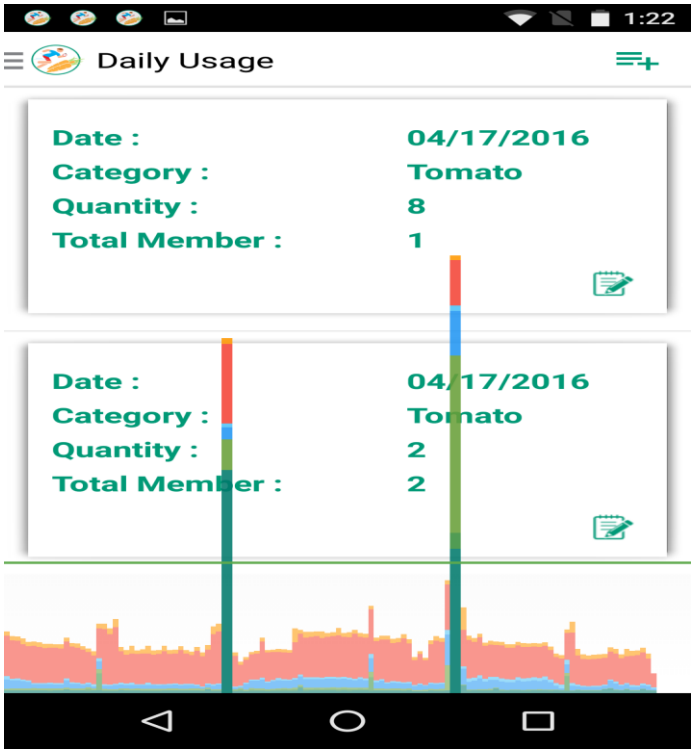

Figure 9-9 Profiling GPU rendering for Daily-Usage Screen

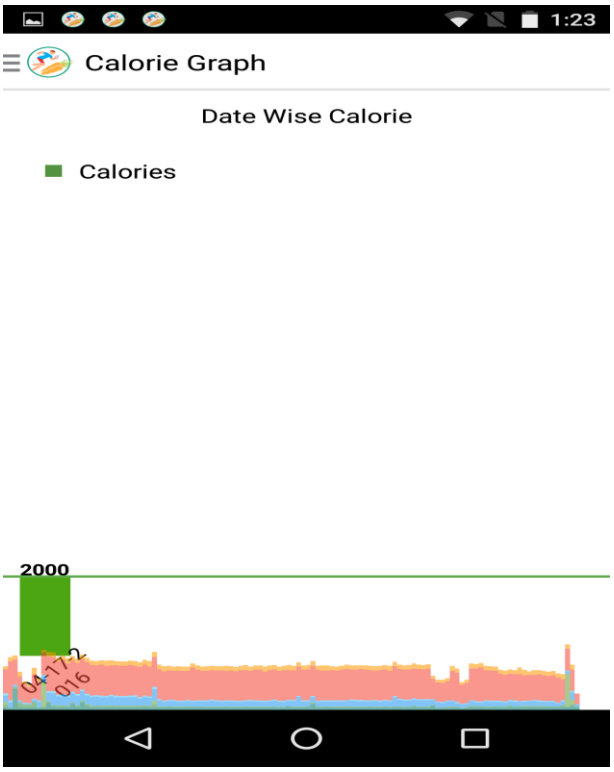

Figure 9-10 Profiling GPU rendering for Calorie Graph Screen

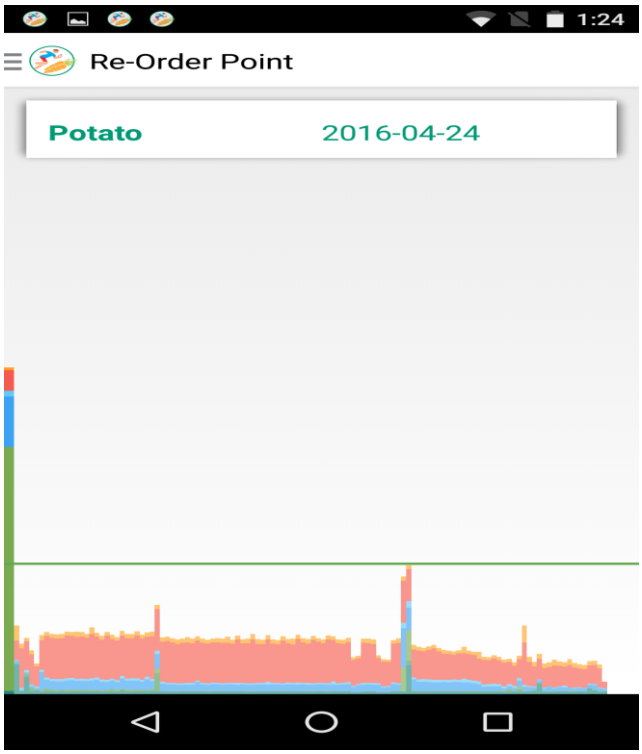

Figure 9-11 Profiling GPU rendering for Re-Order Point Screen

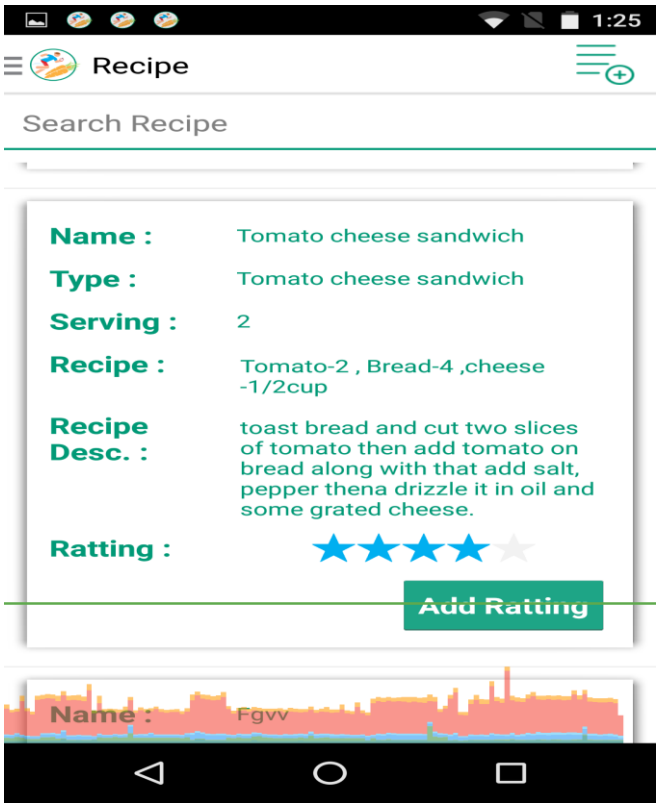

Figure 9-12 Profiling GPU rendering for Recipe Screen

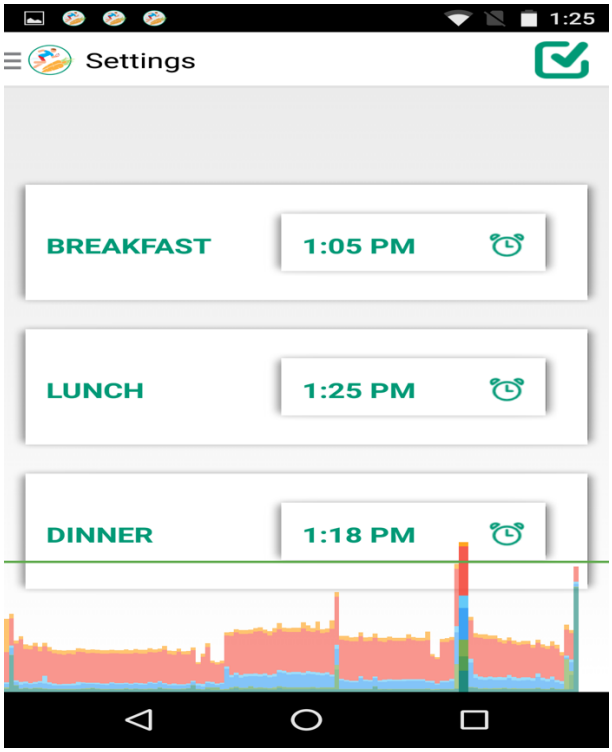

Figure 9-13 Profiling GPU rendering for Settings Screen

9.2 Profiling with Traceview

Traceview[13] is a graphical viewer for execution logs. It can help us debug the application and profile its performance. I have created a trace log file using DDMS for the add grocery process. The Traceview generated by loading the log file displays the log data in two panels:

- **A timeline panel** - describes when each thread and method started and stopped. Each thread's execution is shown in its own row, with time increasing to the right.
- **A profile panel** - provides a summary of all the time spent in a method. The inclusive time is the time spent in the method plus the time spent in any called functions. The exclusive time is the time spent in the method. The parent and children of the method is displayed in this panel.

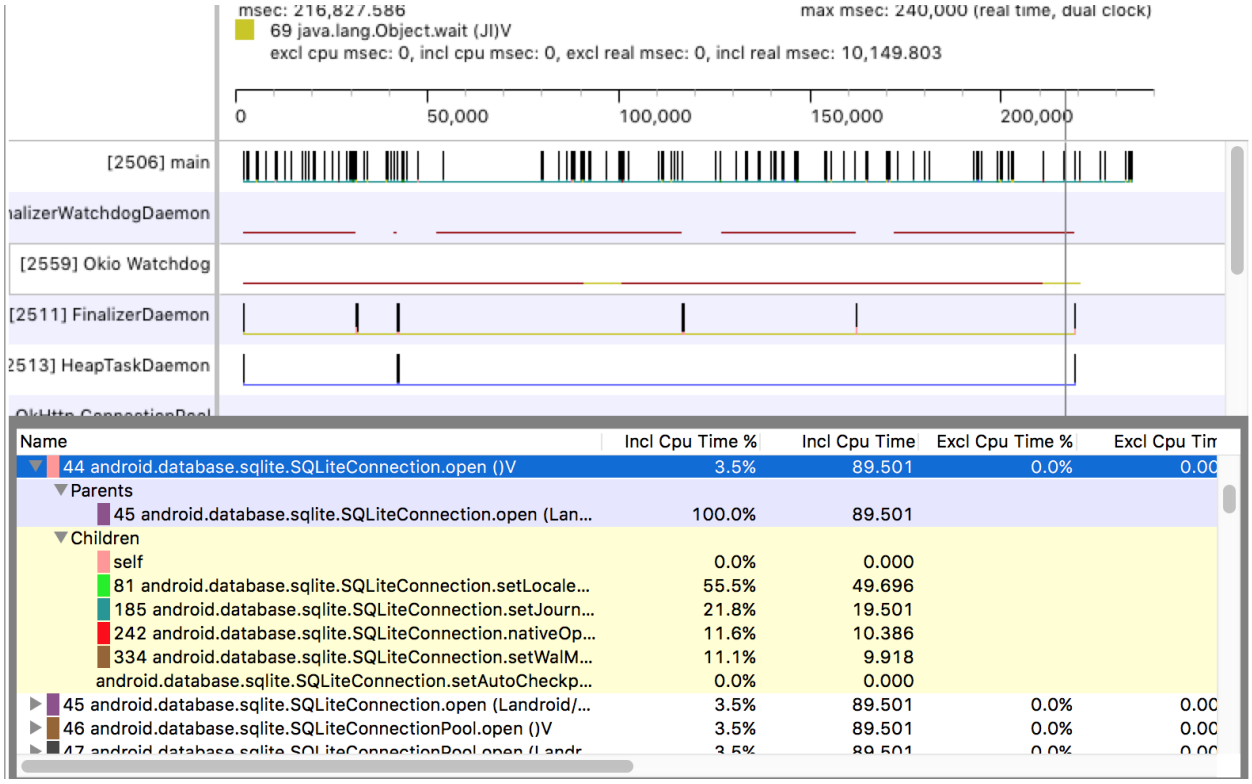

Figure 9-14 Traceview for Application

The Traceview generated for the complete running of the application looks good as the total time that is taken to initiate the features is less and the wait time is also less. CPU usage of the application is moderate which makes the application more efficient.

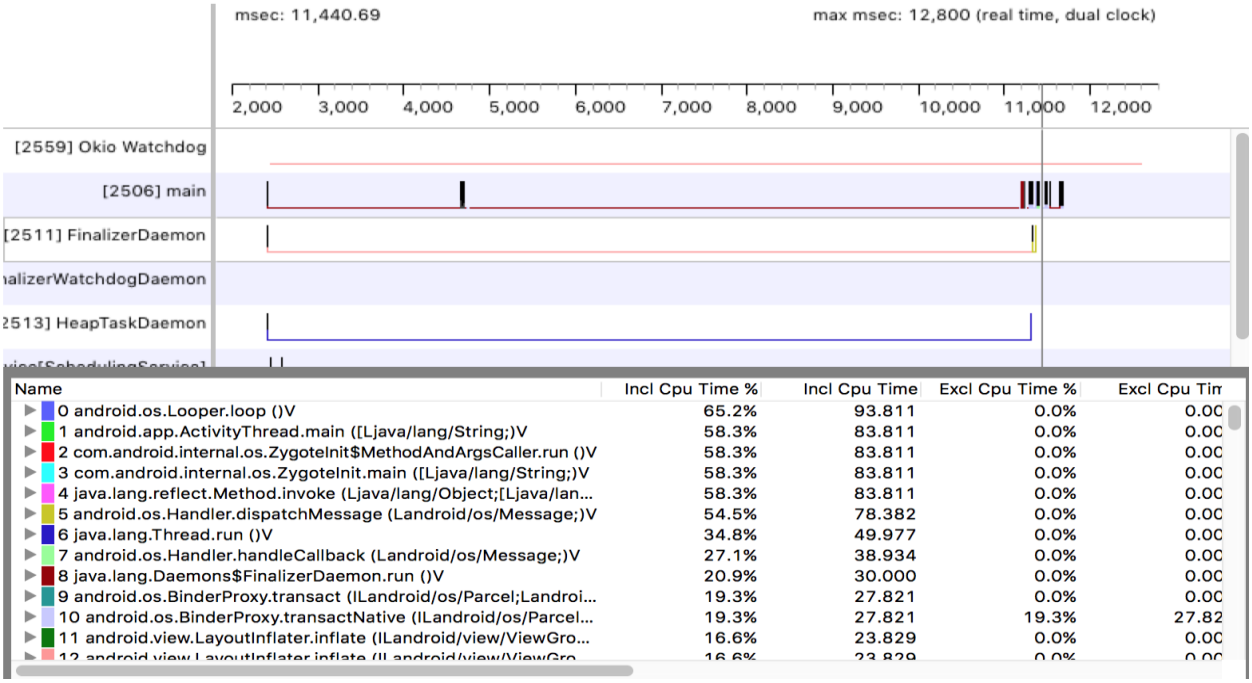

Figure 9-15 Traceview for Add Grocery functionality

9.3 CPU Load Analysis

The below pie-chart shows the CPU load for the Add Grocery method.

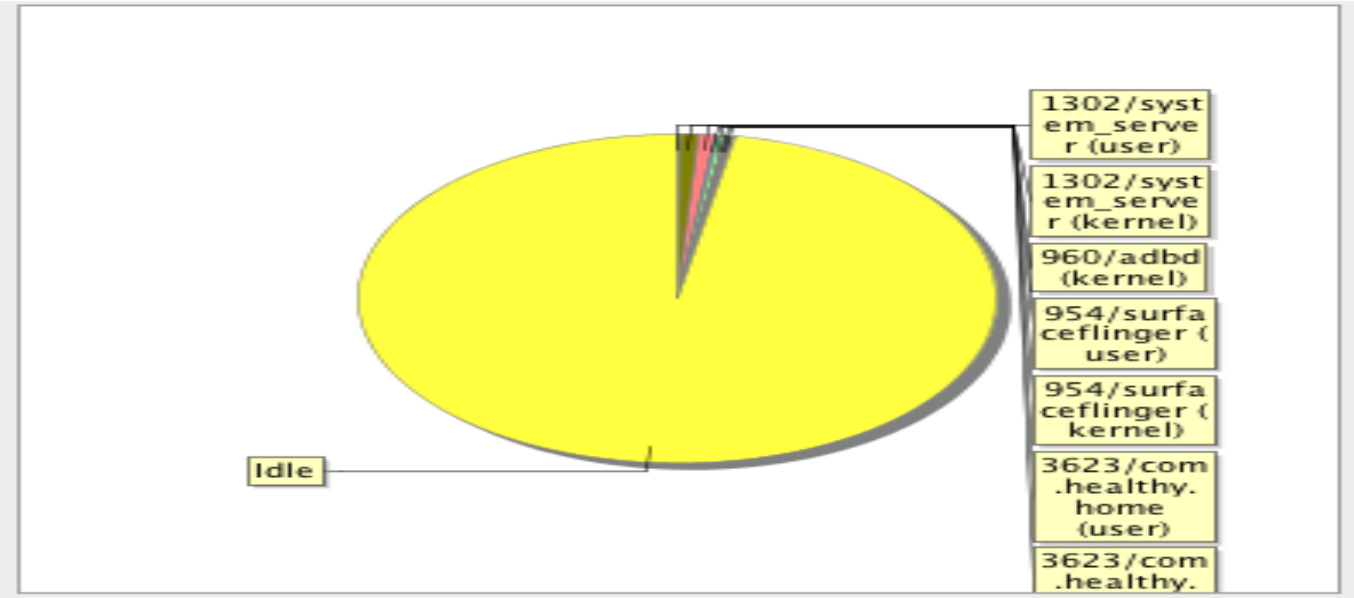

Figure 9-16 CPU Load Analysis for Add Grocery

com.healthy.home(user): (20,3%)

com.healthy.home(kernel): (10,1%)

The amount of CPU usage by the app is less so the app is proved to be efficient.

9.4 Memory Usage Analysis

This application does not use a large amount of memory on phone as all the operations are being performed on the server side. Manual testing showed that it used only 40 MB.

Chapter 10 - Conclusion and Future Work

10.1 Conclusion

The Healthy Home application helps the user to know the amount of the food consumed and the amount of food required by the user. It also provides the user with the ability to know about others recipes, also upload his recipes and also to rate the recipes. The amount of calories earned by food consumption of user are also known. Overall this application allows the user to avoid food wastage which will also positively affect the individual as well as national economy and also most importantly environment.

The HH application followed a complete software development life cycle with analysis, followed by requirements gathering, implantation which is done using Android studio with ADT plugin and testing which was done on real devices and also using Genymotion emulator. Throughout this process I have learnt android development and understood its various components and the functionality.

10.2 Future Work

The functionalities of Healthy Home application can be increased further by adding details about the carbohydrates and nutrition values and using them to synchronize with the daily routine so that one can manage their health and stay fit. Instead of the manual purchase of groceries the application can be used to place the order of groceries frequently used by user at a nearby the super market. Instead of the user entering the groceries list as input a scanner class can be used to enter grocery details.

Chapter 11 - Bibliography

[1] “An overview of the Android Architecture”

http://www.eazytutz.com/android/android-architecture/ [Feb.15, 2016]

[2] “Android-Architecture”

http://www.tutorialspoint.com/android/android_architecture.htm [Feb.15, 2016]

[3] “About Android”

http://developer.android.com/about/android.html [Feb. 15, 2016]

[4] “Systems design”

https://en.wikipedia.org/wiki/Systems_design [Feb. 20, 2016]

[5] “The Unified Modeling Language”

http://www.uml-diagrams.org/ [Feb 24, 2016]

[6] “App Manifest”

http://developer.android.com/guide/topics/manifest/manifest-intro.html. [Mar. 05, 2016]

[7] “Activity Lifecycle Callbacks”

http://developer.android.com/guide/components/activities.html [Mar. 07, 2016]

[8] “SQLite Database”

https://www.sqlite.org/features.html [Mar. 12, 2016]

[9] “MySQL Database”

https://www.mysql.com/products/enterprise/database/ [Mar. 15, 2016]

[10] “Debugging”

http://developer.android.com/tools/debugging/index.html. [Mar. 26, 2016]

[11] “Performance Profiling Tools”

http://developer.android.com/tools/performance/index.html [Apr. 12, 2016]

[12] “Profiling GPU Rendering Walkthrough”

http://developer.android.com/tools/performance/profile-gpu rendering/index.html -WhatYouNeed [Apr. 10, 2016]

[13] “Profiling with Traceview”

http://developer.android.com/tools/debugging/debugging-tracing.html [Apr. 11, 2016]