

# QUICK RECIPES ANDROID APPLICATION

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of  
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Ruchi Gupta

SPRING  
2015

# QUICK RECIPES ANDROID APPLICATION

A Project

by

Ruchi Gupta

Approved by:

\_\_\_\_\_, Committee Chair  
Dr. Jinsong Ouyang

\_\_\_\_\_, Second Reader  
Dr. Scott Gordon

\_\_\_\_\_  
Date

Student: Ruchi Gupta

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

\_\_\_\_\_, Graduate Coordinator  
Dr. Jinsong Ouyang

\_\_\_\_\_  
Date

Department of Computer Science

Abstract  
of  
QUICK RECIPES ANDROID APPLICATION  
by  
Ruchi Gupta

There are a number of applications in Android store for Recipes Search but none of them support interface for searching, saving, and sharing recipes all at once. Quick Recipes is an Android application with image based UI for searching, sharing, creating and saving recipes. This app provides flexibility to user to search top rated and variety of recipes from available recipes in cloud. This is very handy application, which every user can search cloud for recipes, save recipe as favorite, share recipe with friends on social media Facebook, Twitter, Google+ and Gmail and create personal cookbook. This app is time saver providing recipes in few clicks. By combining ingredients and title search, Quick Recipes app makes finding recipes easy. With recipes being added daily there will always be something new for user to crave. The project has been implemented using Java, Android and SQLite. Food2Fork API are consumed for getting recipes from cloud.

\_\_\_\_\_, Committee Chair  
Dr. Jinsong Ouyang

\_\_\_\_\_  
Date

## ACKNOWLEDGEMENTS

I would like to thank Dr. Jinsong Ouyang, my advisor for providing me an opportunity to work on this project, which significantly broadened my knowledge on Android Development. I thank him for continuously providing the feedback, help and support to complete the project successfully.

In addition, I would like to thank Dr. Scott Gordon for his willingness to serve on the committee.

Lastly, I would like to thank the entire faculty and staff of the Department of Computer Science Engineering at California State University, Sacramento.

## TABLE OF CONTENTS

	Page
Acknowledgements.....	v
List of Figures .....	x
Chapter	
1. INTRODUCTION .....	1
2. PROJECT REQUIREMENTS.....	2
2.1 User Options.....	2
2.2 Recipe Search.....	3
2.2.1 Search by Title and Ingredients .....	3
2.2.2 Search Top Rated Recipes .....	4
2.2.3 Search Trending Recipes .....	5
2.2.4 Category Based Recipe Search.....	5
2.2.5 Ingredients Based Recipe Search .....	6
2.3 Saving Recipes.....	7
2.3.1 Add to Favorites .....	7
2.4 Sharing Recipes.....	7
2.4.1 Facebook.....	8
2.4.2 Twitter.....	9

2.4.3 Google +.....	9
2.4.4 Gmail.....	9
2.5 CookBook .....	10
2.5.1 Create Recipe.....	11
2.5.2 View Recipe .....	11
2.5.3 Delete Recipe.....	12
3. ANDROID DEVELOPMENT BASICS.....	13
3.1 Four Types of Application Components .....	13
3.1.1 Activities .....	14
3.1.2 Services.....	16
3.1.3 Content Providers .....	18
3.1.4 Broadcast Receivers .....	19
3.2 Setup Android Environment .....	19
3.3 Creating Sample Android Application .....	20
3.4 Running the Application .....	24
3.4.1. On Device .....	24
3.4.2. On Emulator .....	25
4. FEATURES IMPLEMENTATION.....	28

4.1 Application Home Page .....	28
4.2 Search Recipe .....	31
4.2.1 Search by Title or Ingredients .....	32
4.2.2 Search by Trend .....	36
4.2.3 Search by Rating .....	38
4.3 Search by Selection .....	40
4.3.1 Select Category .....	41
4.3.2 Select Ingredients .....	44
4.4 View Recipes .....	46
4.4.1 View List of Recipes .....	46
4.4.2 View Single Recipe .....	51
4.5 Save Recipes as Favorite .....	52
4.6 Share Recipes .....	55
4.6.1 Share on Facebook .....	58
4.6.2 Share on Twitter .....	59
4.6.3 Share on Google + .....	60
4.6.4 Share on Gmail .....	61
4.7 Cookbook Implementation .....	62
4.7.1 Create Recipe .....	64



4.7.2 View Recipe .....	69
4.7.3 Delete Recipe.....	71
4.8 Future Work .....	74
5. CONCLUSION.....	75
References.....	76

## LIST OF FIGURES

Figures	Page
Figure 1 Use Case Diagram.....	2
Figure 2 Search by Title and Ingredients Flow Diagram.....	3
Figure 3 Top Rated Recipe Search Flow Diagram .....	4
Figure 4 Trending Recipe Search Flow Diagram .....	5
Figure 5 Category Based Recipe Search Flow Diagram .....	5
Figure 6 Ingredients Based Recipe Search Flow Diagram .....	6
Figure 7 Adding to Favorites Flow Diagram.....	7
Figure 8 Sharing Recipes Flow Diagram.....	8
Figure 9 Cookbook Flow Diagram .....	10
Figure 10 Activity Lifecycle .....	15
Figure 11 Service Lifecycle .....	17
Figure 12 Creating New Android App in Eclipse.....	20
Figure 13 Configure Project in Eclipse.....	21
Figure 14 Configure Attributes in Eclipse .....	22
Figure 15 Android Device Chooser .....	25
Figure 16 Android Virtual Device Manager .....	26
Figure 17 Android Virtual Device Edit .....	27
Figure 18 QuickRecipes Application Home Page .....	28
Figure 19 Home Icon .....	30

Figure 20 Search Image Button .....	33
Figure 21 Search by Title Page .....	34
Figure 22 Search by Ingredients List Page .....	35
Figure 23 Trending Recipes Search Button .....	37
Figure 24 Top Rated Recipes Search Button .....	39
Figure 25 Categories Search Image Button .....	40
Figure 26 Categories List Page .....	41
Figure 27 Ingredients Search Image Button .....	44
Figure 28 Ingredients List Page .....	45
Figure 29 View Recipes List.....	47
Figure 30 View Recipe Details .....	51
Figure 31 Adding Recipe to Favorite.....	52
Figure 32 Favorite Image Button on Application Home Page .....	53
Figure 33 View Favorite Recipes List .....	54
Figure 34 Sharing with Share Button.....	55
Figure 35 Sharing Options .....	57
Figure 36 Sharing Recipe on Facebook .....	58
Figure 37 Sharing Recipe on Twitter.....	59
Figure 38 Sharing Recipe on Google+.....	60
Figure 39 Sharing Recipe on Gmail.....	61
Figure 40 Cookbook Options.....	62

Figure 41 Create Recipe Screen.....	64
Figure 42 Take Recipe Image Option .....	66
Figure 43 View Recipe List .....	69
Figure 44 Delete Recipe Screen .....	71

## **Chapter 1**

### **INTRODUCTION**

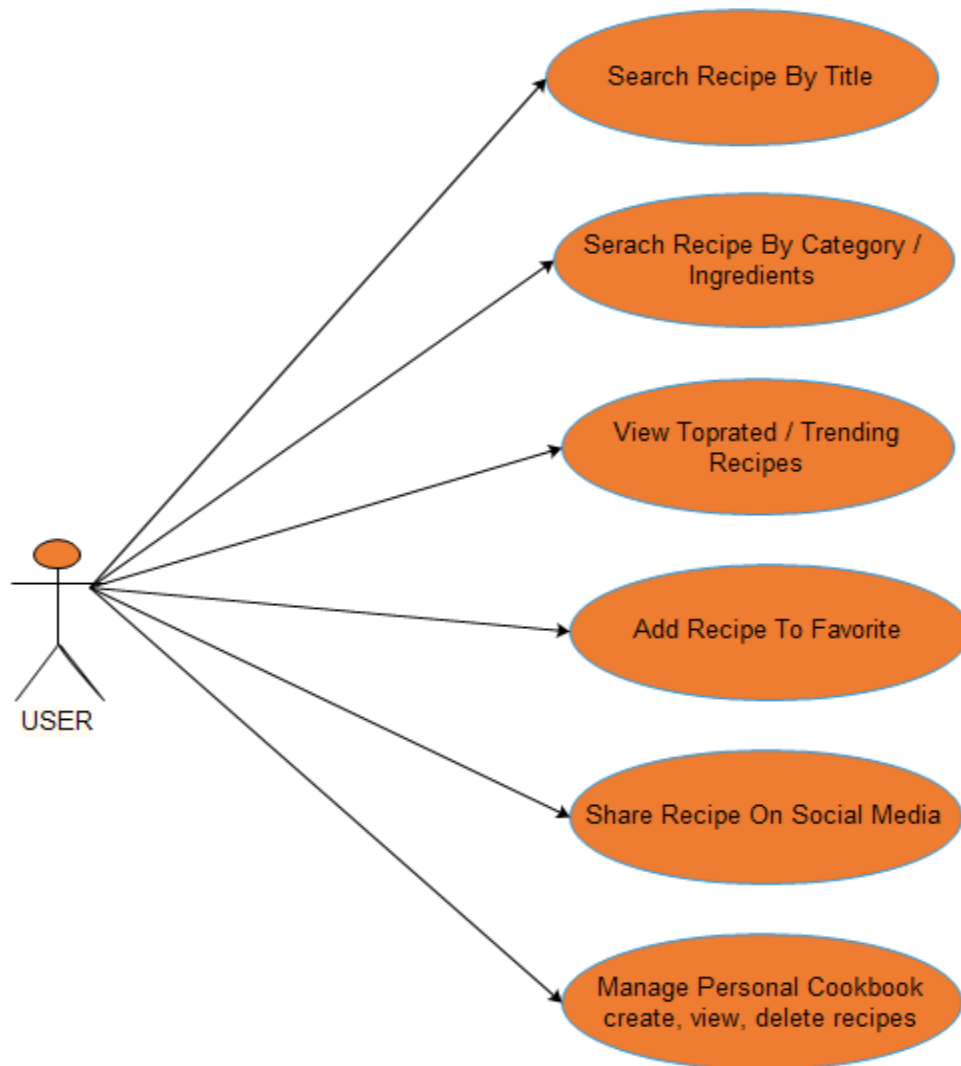
QuickRecipes application is a very useful app for people who love to cook and try out new recipes. QuickRecipes application provides user flexibility to search, share, save recipes from cloud with an additional capability to maintain personal cookbook for creating new recipe, deleting recipe that are no longer required.

My application “QuickRecipes” makes finding recipes easy. Recipes are from everyday cooks and chefs that have perfected the recipe over time. Food2Fork API are used for searching recipes from cloud. Recipes range from the decadent to the simple for whatever mood user are in. With recipes being added daily in Food2Fork database there will always be something new for user to try. Recipes are from well-known publishers All Recipes, 101 Cookbooks, Closet Cooking and many more. QuickRecipes Application provides capability to user to search top rated, trending recipes. Recipes are displayed with recipe image, title, ingredients list and cooking directions. Category based and ingredients based search is also provided to user. QuickRecipes App provide sharing capability, giving user a choice to share recipe through Facebook, Twitter, Google+ and Gmail. This application is a time saver providing recipes in few clicks. The user is given choice to create personal cookbook, where user can create recipe, view recipe and delete recipe. The interface is clean and simple. It makes use of Android image button capability to display options on home screen with image icons. The user can search recipes, view added favorite recipe list and access personal cookbook all from home screen.

## Chapter 2

### PROJECT REQUIREMENTS

#### 2.1 User Options



**Figure 1- Use Case Diagram**

Users of QuickRecipes application have following options as shown in Figure 1:

- User can search recipes from cloud. Search can be performed based on rating, trend, categories, ingredient and title.

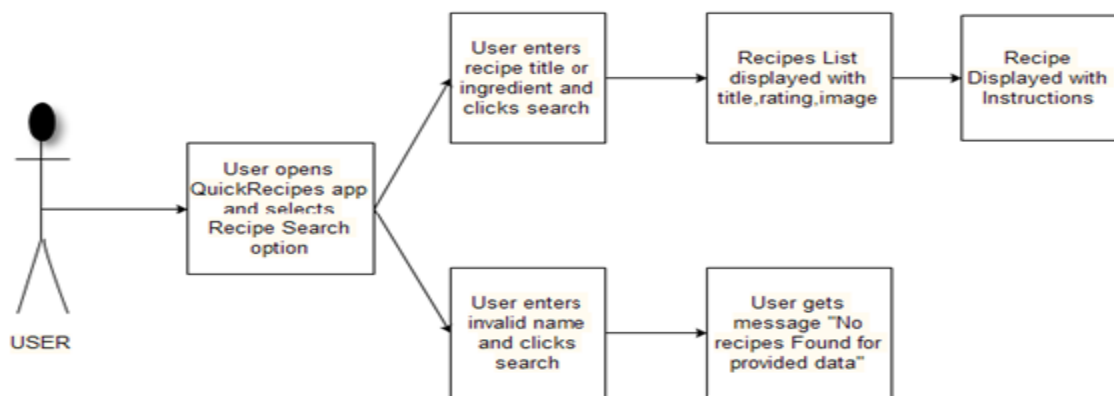
- User can view selected recipe from list of recipes.
- User has option to save recipe as favorite.
- User can share recipes on Facebook, Twitter, Gmail and Google+.
- User can manage cookbook by creating recipes, viewing recipe and deleting recipe.

QuickRecipes application provides home icon on each screen for easy navigation. User can navigate to home screen by clicking on home icon located at top right corner from other screens. The following section describes each part in detail.

## 2.2 Recipe Search

User can perform search for recipes based on title, rating, trending, categories and ingredients.

### 2.2.1 Search by Title and Ingredients

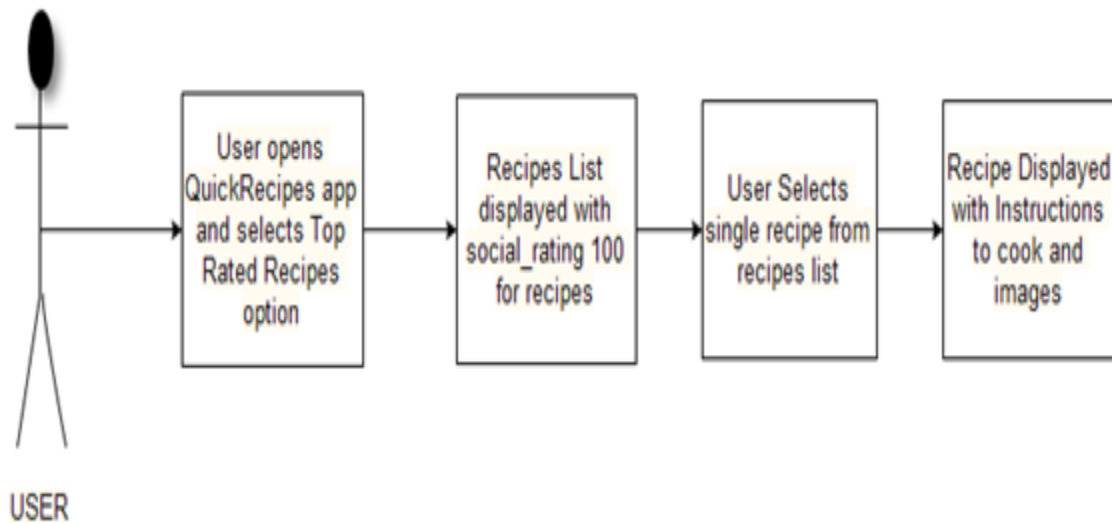


**Figure 2- Search by Title And Ingredients Flow Diagram**

User is provided with search image icon on home screen to search recipe. Clicking the search image icon on home page user is navigated to new page where user can either type

title or ingredient. Figure 2 depicts flow diagram for title, ingredients search option. On search button click, list of recipes are displayed with image, recipe title. When user enters invalid data, user gets message no recipes found for provided data.

### 2.2.2 Search Top Rated Recipes

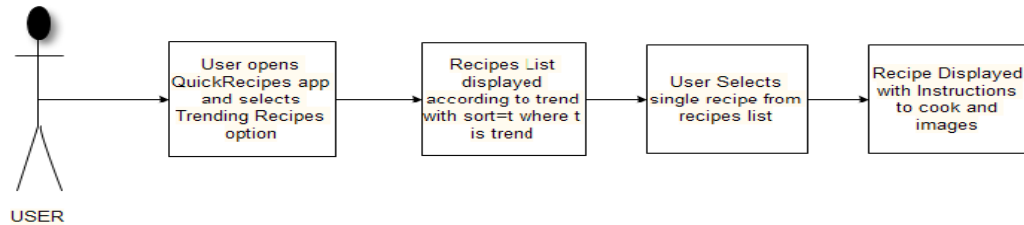


**Figure 3- Top Rated Recipe Search Flow Diagram**

Searching recipes based on rating. This rating depends on social media scores which determines best recipes. User is provided with top rated image button on home screen as illustrated in Figure 3. Clicking top rated image button user is navigated to new page where list of recipes are displayed with social rating of 100. User can select single recipe from list and can view recipe.



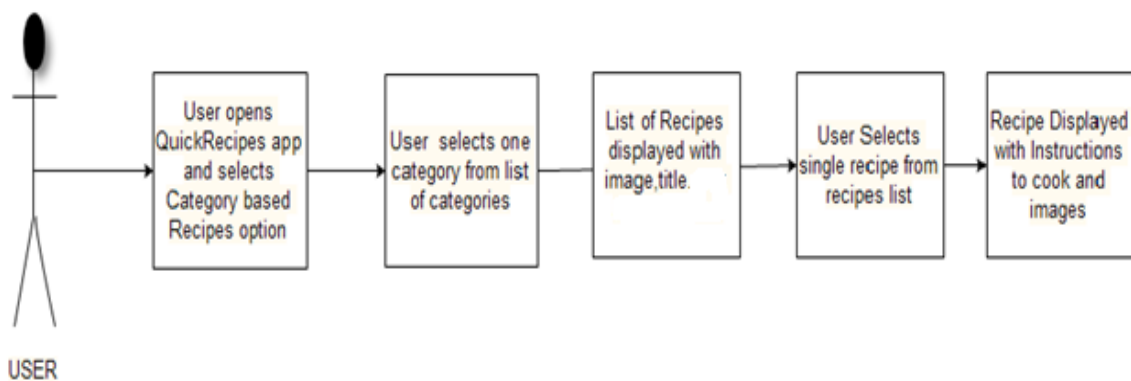
### 2.2.3 Search Trending Recipes



**Figure 4- Trending Recipe Search Flow Diagram**

Searching recipes based on trend. The most recent recipes from publishers have trend score depending on recipes gaining popularity. User is provided with trending recipes image button on home screen. Figure 4 shows searching trending recipes flow diagram. Clicking on trending image button user is navigated to new page where popular recipes list are displayed. User can select single recipe from list and can view recipe.

### 2.2.4 Category Based Recipe Search

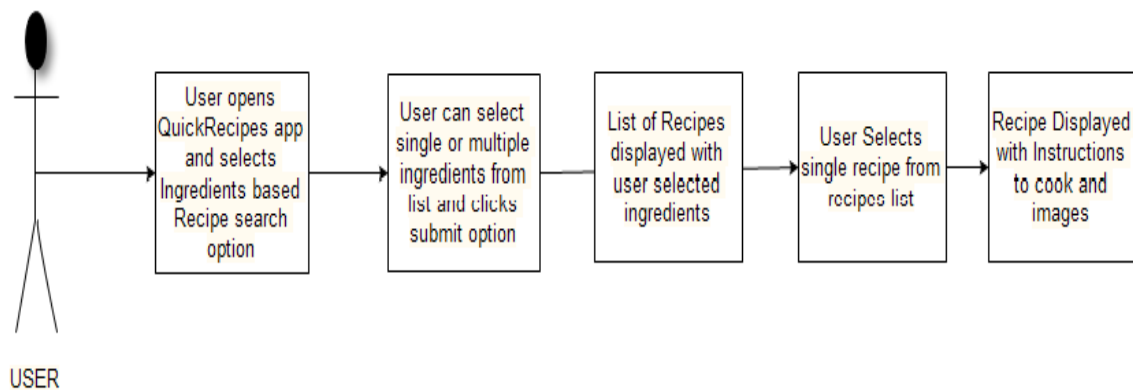


**Figure 5- Category Based Recipe Search Flow Diagram**

User is provided with categories image icon on home screen to search recipe based on category. Figure 5 illustrates flow for user to perform category based search. Recipes are

grouped under various categories. Categories include cakes, pasta, drinks, starters, pie, noodles, family meal and salad. Clicking on categories image button on home page user is navigated to new page where user can select category from list of categories. Categories are listed with image and category title. On selection of category user has option to view recipe details. For example, if user selects cakes category, user has option to view red velvet cake, chocolate cake and many other cakes cooking details.

### 2.2.5 Ingredients Based Recipe Search



**Figure 6- Ingredients Based Recipe Search Flow Diagram**

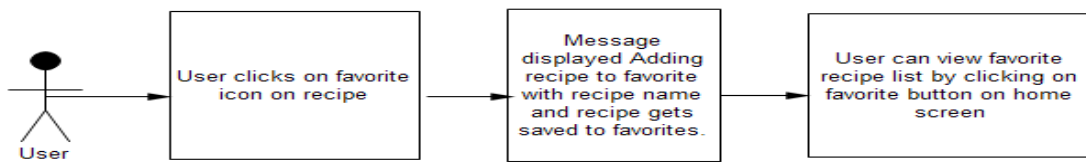
User is provided with ingredients image icon on home screen to search recipe based on ingredients. Ingredients include potato, tomato, mushroom, broccoli and eggplant. Figure 6 illustrates flow diagram for user to perform ingredients based search. Clicking on ingredients image button on home page user is navigated to new page where user can select ingredient from list of categories. Ingredients are listed with image and name. On selection of ingredient, list of recipes image and title displayed containing selected ingredient. User

can select any recipe from list and can view recipe image, cooking directions and other details.

## 2.3 Saving Recipes

By using QuickRecipes application user has option to save searched recipes from cloud in favorites list.

### 2.3.1 Add to Favorites



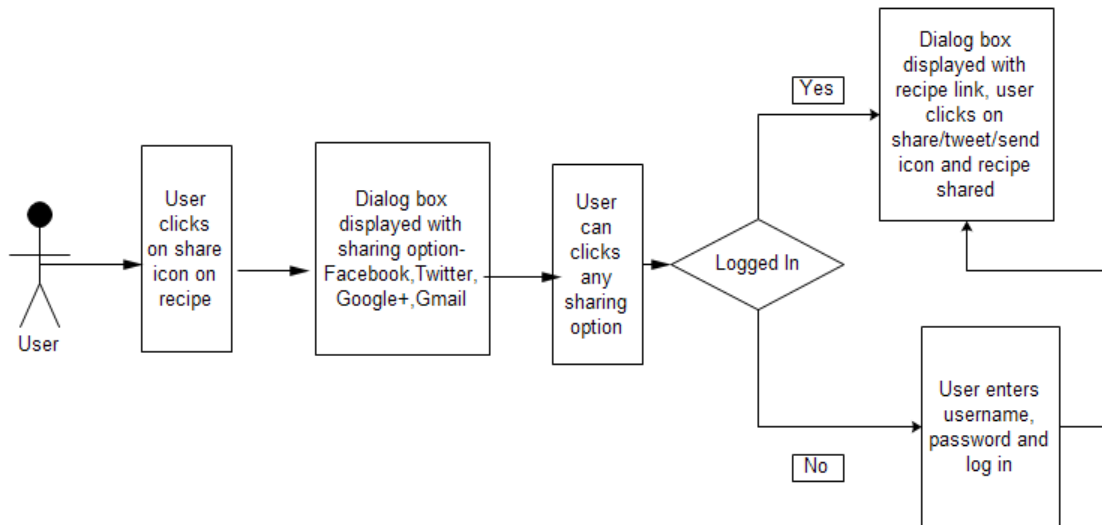
**Figure 7- Adding to Favorites Flow Diagram**

The application provides user an option to perform recipe search. Result of search operation is list of recipes. This list consist of recipe image, title, favorite icon and share icon for each recipe in list. Upon clicking the favorite image icon, user can save recipe as favorite. When recipe is added toast message “adding recipe as favorite” with recipe name is displayed and screen refreshes. Recipe added can be seen under Favorite Image Button on home screen. Figure 7 shows flow diagram to add recipes to favorite.

## 2.4 Sharing Recipes

QuickRecipes application provides share button for social media sharing. User has option to share recipes link with friends and family. QuickRecipes application provide sharing on Facebook, Twitter, Gmail and Google+. Upon clicking on share button toast message “sharing recipe” with recipe name is displayed and a dialog box pops up with Facebook,

Twitter, Google+ and Gmail icons. User can select any option for sharing. Figure 8 shows flow diagram for sharing recipes.



**Figure 8- Sharing Recipes Flow Diagram**

### **2.4.1 Facebook**

User can share recipe link on Facebook by clicking on Facebook icon from dialog box. If user is already logged in on Facebook, selected recipe link is sent on user timeline with image and recipe name. User has option to write something and click on share or just click share button. User has choice to share on his timeline, on friend's timeline, in a group or in his own page. If user is not logged in, click on Facebook icon navigates user to Facebook log in page before user can share.

### **2.4.2 Twitter**

User can share recipe link on Twitter by clicking on Twitter icon from dialog box. Upon clicking on Twitter icon, if user is logged in, recipe link is sent and user has option to tweet by clicking on tweet button. User can also add location during sharing by clicking on enable location. If user is not logged in, click on Twitter icon navigates user to Twitter log in page. After entering valid user name and password user can tweet recipe link.

### **2.4.3 Google+**

User can share recipes on Google+ by clicking on Google+ icon from dialog box. If user is logged in, upon clicking on Google+ icon, recipe link is sent on share page with recipe image and recipe publisher website information. User has option to add location, disable reshares, disable comments and select people to share. User can share by clicking on arrow sign on right side of page.

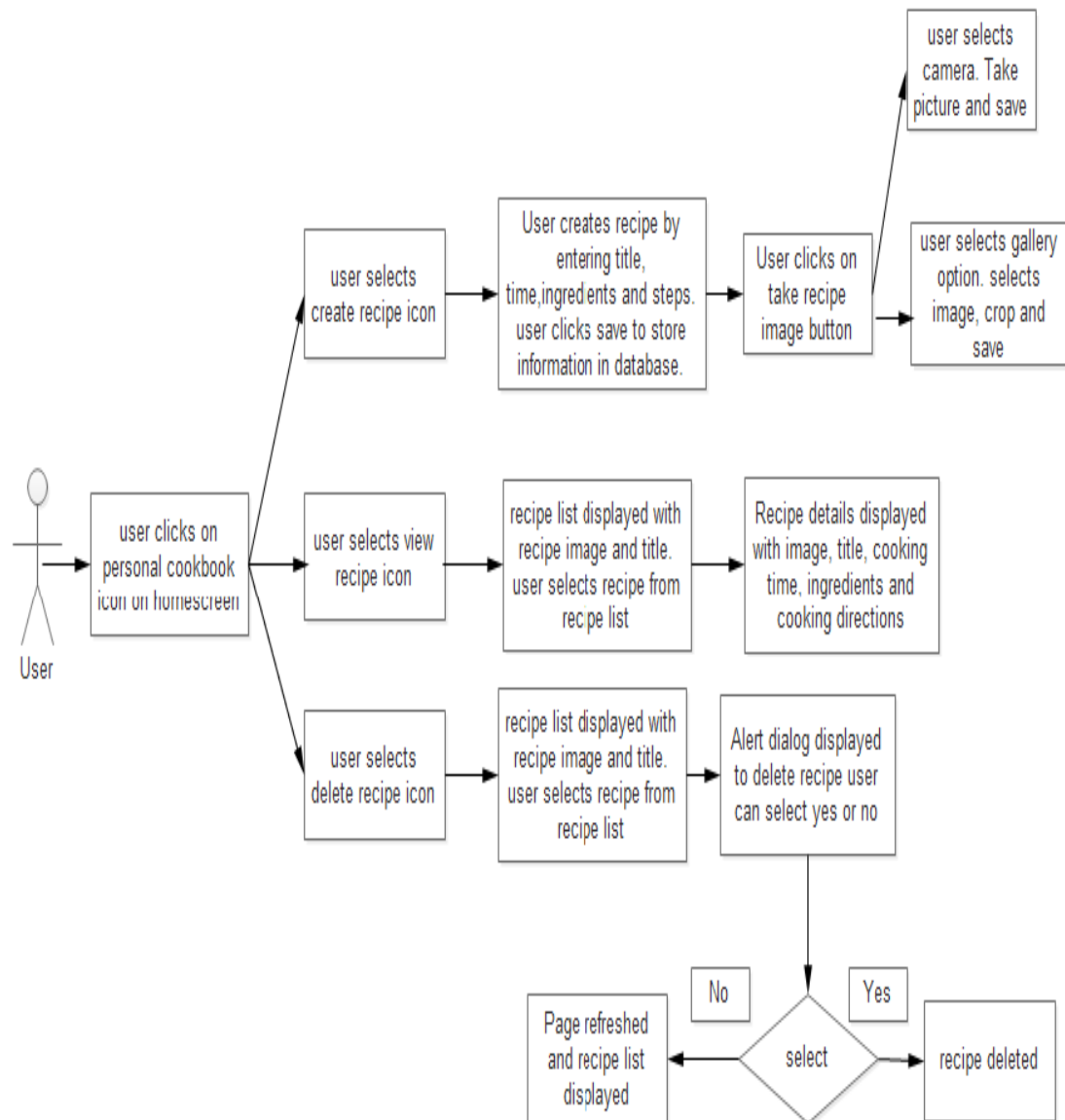
### **2.4.4 Gmail**

User can share recipe link on Gmail by clicking on Gmail icon from dialog box. If user is already logged in on Gmail, selected recipe link is copied on new email dialog. User has option to write receiver email address, edit subject, add more comments on body of email and click send button. If user is not logged in, click on Gmail icon navigates user to Gmail log in page before user can send email.

## 2.5 Cookbook

User has option to manage cookbook by creating recipe, viewing recipe and deleting recipe.

Figure 9 shows flow diagram for managing cookbook in application.



**Figure 9- Cookbook Flow Diagram**

### **2.5.1 Create Recipe**

User can create and manage his personal cookbook on phone, tablet. When user clicks personal cookbook image button on home page, user gets navigated to new screen. This new screen provides three image button for creating, viewing and deleting recipe respectively. On clicking, click image to add recipes button on personal cookbook screen, new activity screen template is displayed. Template provide fields for recipe title, cooking time, ingredients list for selection and cooking steps. User can create recipe by filling details in template fields and clicking on save button. When user clicks save button, new row gets added in database for created recipe. Take recipe image button allows user to save recipe image. User has option to capture image from camera or select image from gallery. Additional features like crop image, discard selected image are also provided.

### **2.5.2 View Recipe**

QuickRecipes application save recipes added by user in database providing capability to view saved recipes when required. When user clicks personal cookbook image button on home page, user gets image button for viewing recipe. On clicking, view recipes image button on personal cookbook screen, list of user created recipes are displayed with recipe image and recipe title. If recipe image is not provided by user during creation of recipe default image gets displayed with message no image available. User can select recipe from list by clicking on name. Upon click recipe title, time required for cooking, ingredients needed, cooking directions and recipe image are displayed

### **2.5.3 Delete Recipe**

User has delete option in personal cookbook screen. User can click image button with cross sign for performing deletion. Upon click, user saved recipe list gets displayed with image and title. User can select recipe from list, on selection alert dialog box is displayed with message deleting recipe. User can select yes button for deletion, and no to cancel delete operation. When user clicks yes, recipe gets deleted and page is refreshed. Upon clicking no, user is navigated to recipe list screen.



## **Chapter 3**

### **ANDROID DEVELOPMENT BASICS**

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smartphones and tablet computers. To make things easier for developers, Google had made Android as open source and releases the code under the Apache License. Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language [1].

QuickRecipes application is written in java programming language using SQLite database. The code of QuickRecipes App is compiled using Android SDK tools into an android package, an archive file with an .apk suffix. This .apk file is considered as one application and is used to install the application in all android devices like tablets and phones.

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact [1][2]. There are four different types of application components: activities, services, broadcast receivers and content providers.

#### **3.1 Four Types of Application Components**

Android provides four application components: activities, services, content providers and broadcast receivers. Below section describes each component in detail.

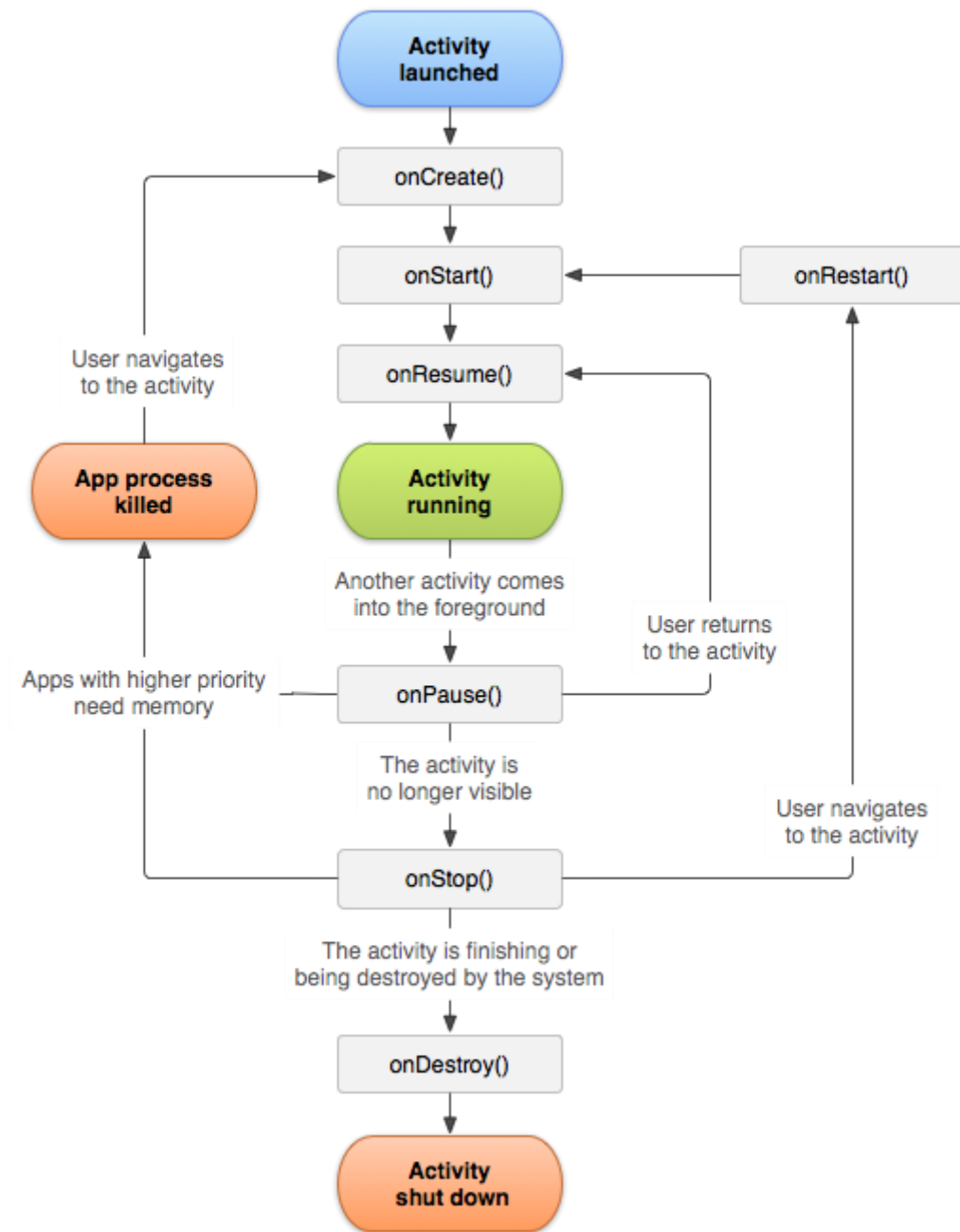
### 3.1.1 Activities

Activity represents a single screen with a user interface. For example, in this QuickRecipes application there are many activities such as search by title and ingredient activity, category activity, cookbook create recipe activity and so on. One activity in the application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Whenever a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. This back stack follows LIFO(last in first out) stack mechanism, so, when the user is done with the current activity and presses the back button, it is popped from the stack (and destroyed) and the previous activity resumes.

Activity must be declared in the manifest file to be accessible by the system. Intent filters can also be specified in the `<activity>` element to declare how other application components may activate it.

The activity automatically includes an intent filter that declares the activity responds to the "main" action and should be placed in the "launcher" category. The intent filter looks like this:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```



**Figure 10- Activity Lifecycle[3]**

The callback methods for activity are as follows which are shown in Figure 10.

**onCreate():** This is called when the activity is first created. This is always followed by **onStart()**.

**onStart():** This is called just before the activity becomes visible to the user. This is followed by **onResume()** if the activity comes to the foreground, or **onStop()** if it becomes hidden.

**OnResume():** This is called just before the activity starts interacting with the user. This is always followed by **onPause()**.

**onPause():** This is called when the system is about to start resuming another activity. And this is followed either by **onResume()** if the activity returns back to the front, or by **onStop()** if it becomes invisible to the user.

**onStop():** This is called when the activity is no longer visible to the user. This is followed either by **onRestart()** if the activity is coming back to interact with the user, or by **onDestroy()** if this activity is going away

**onRestart():** This is called after the activity has been stopped, just prior to it being started again. This is always followed by **onStart()**.

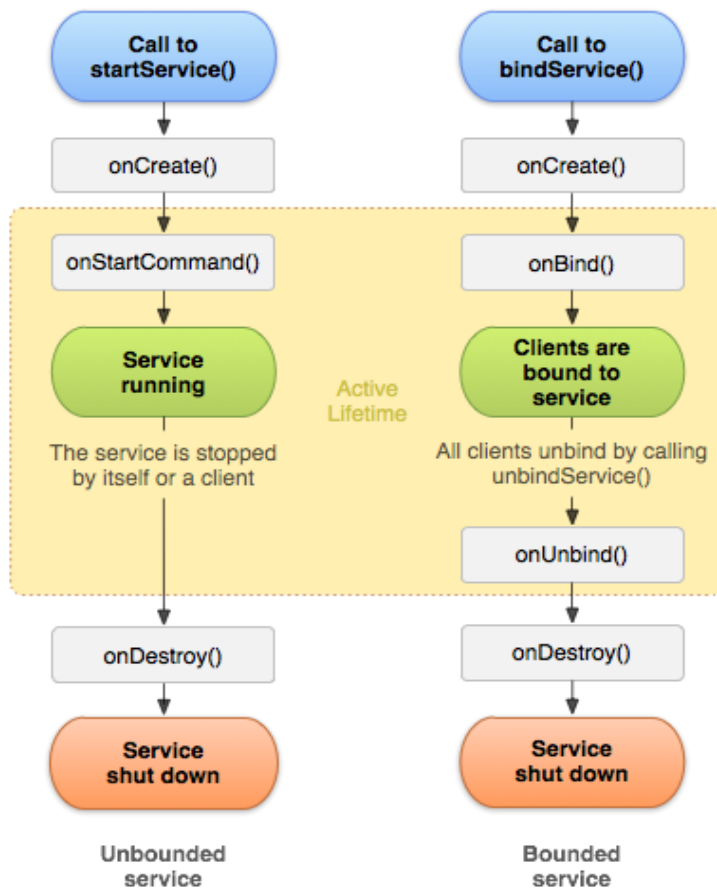
**OnDestroy():** This is called before the activity is destroyed. This is followed by nothing[1].

### **3.1.2 Services**

A service is a component which would be running in background without direct interaction with the user. As the service has no user interface it is not bound to the lifecycle of an activity. Services are used mostly when there is a time consuming and long task, like loading an Image, or a File, or download something for the Internet and asynchronous tasks in general.

Service has two forms: Started and Bound. Below section describes in detail:

1. **Started:** When an activity starts `startService()`, then a service is "started". A service can run in the background indefinitely once started, even if the component that started it is destroyed. For example, it might download a image over the network. When the download is completed, the service should stop itself.
2. **Bound:** When an application component calls `bindService()`, a service is "bound". A bound service provides a client-server interface that allows components to interact with the service, send requests and get results. Figure 11 shows lifecycle of services in android.



**Figure 11- Service Lifecycle[4]**

The callback methods are:

`onStartCommand()`: This method is called when an activity requests for a service to be started by calling `startService()`.

`onBind()`: This method is called when an activity wants to bind with the service by calling `bindService()`.

`onCreate()`: This method is called when the service is first created to perform setup of the service before `onStartCommand` or `onBind` is called.

`onDestroy()`: This method is called when service is no longer required and is to be destroyed[1].

### **3.1.3 Content Providers**

Content providers are used to provide access to other applications to a shared set of application data i.e content providers provide a mechanism through which the data stored by one Android application can be made accessible to other applications. The data is usually stored in file system or SQLite database which the application can access and through content provider, other applications can query or modify the data depending on the settings of the content provider.

A content provider is implemented as a subclass of `ContentProvider` and an application accesses the data from a content provider with a `ContentResolver` client object. In order to insert data into provider, `ContentResolver.insert()` method is used. This method inserts a new row into the provider and returns a content URI for that row. In order to update a row

or delete a row `ContentResolver.update()` and `ContentResolver.delete()` methods can be used respectively[1].

#### **3.1.4 Broadcast Receivers**

A broadcast receiver allows registering for system or application events. When any registered event occurs, receivers for an event will be notified by the Android runtime. System broadcast include screen turning off, the battery is low, or a picture was captured. Applications broadcast would include letting other applications know that some data has been downloaded to the device and is available for them to use. A receiver can be registered via `AndroidManifest.xml`.

The implementing class for a receiver extends the `BroadcastReceiver` class. If the event for which the broadcast receiver has registered happens the `onReceive()` method of the receiver is called by the Android system[5].

### **3.2 Setup Android Environment**

The first step in starting a project would be to download the Android ADT bundle. The bundle provides all features needed to develop the app and also includes a version of the Eclipse IDE with built in ADT(Android Developer Tools) .

Perform the following steps after downloading:

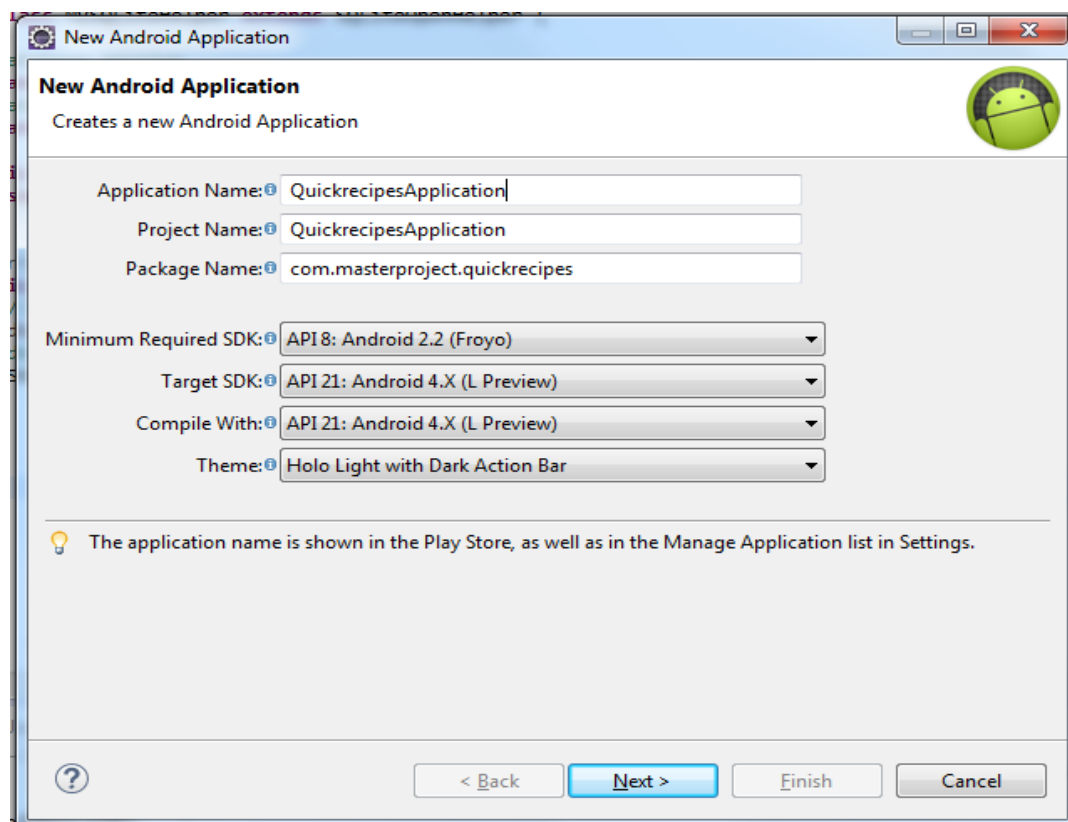
1. Unpack the ZIP file (named `adt-bundle-<os_platform>.zip`) and save it to an appropriate location, such as a "Development" directory in our home directory.
2. Then open the `adt-bundle-<os_platform>/eclipse/` directory and launch **eclipse**.

3. In Eclipse, click on the SDK manager and download the latest SDK tools and platforms .

### 3.3 Creating Sample Android Application

The android project consists of all the code required for the android application. The SDK tools which we downloaded provide basic functionality for us to start coding our application[6]. Figure 12 shows creating new android application screenshot.

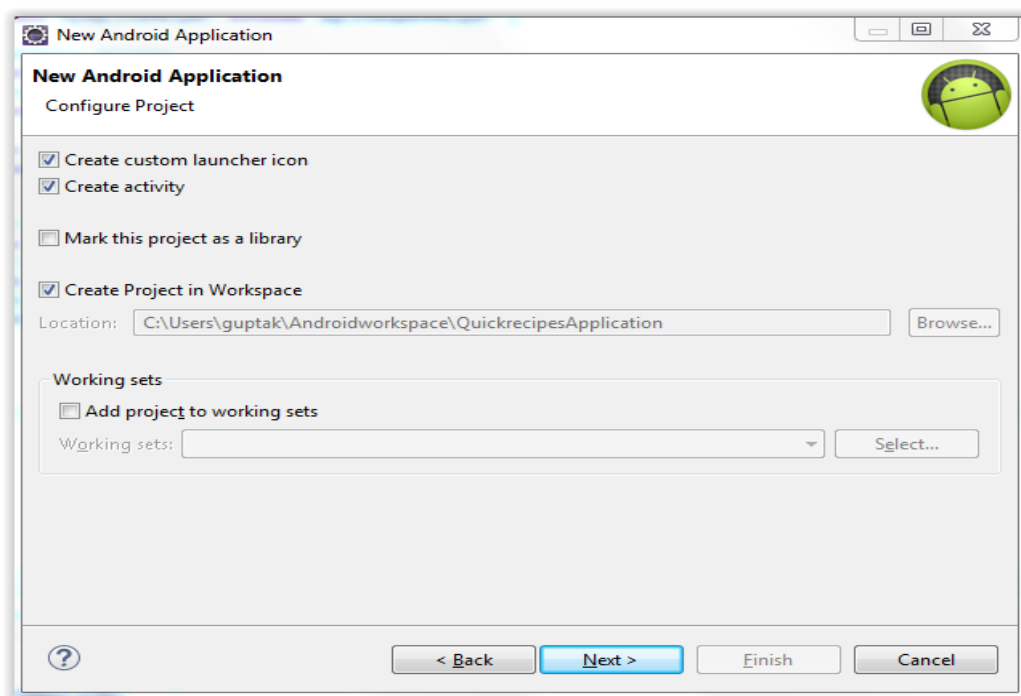
1. First need to click 'New' in the toolbar.
2. When the window appear appears, we need to open the Android folder, select Android Application Project, and then click on Next.



**Figure 12- Creating New Android App in Eclipse**



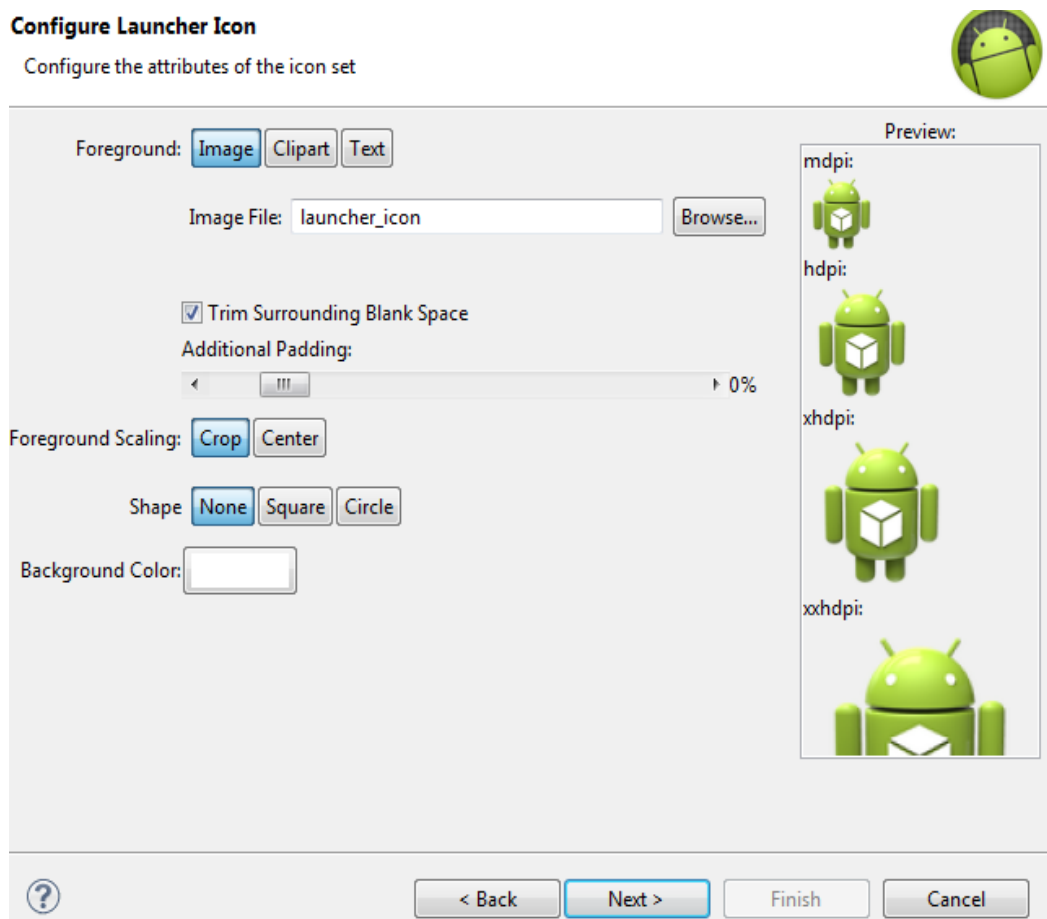
In the android application box, we need to fill appropriate values like the application name is the app name that appears to users which is "QuickrecipesApplication" for this sample project. The project name is the name of our project directory and the name visible in Eclipse. The Package Name is the package namespace for our app which is "com.masterproject.quickrecipes" for this sample project. The lowest version that this app supports is the minimum required SDK. If we want to support multiple devices, this needs to be set to lowest version available such that the application can perform its basic operations. The target SDK is the highest version of Android for this application. The platform version with which we compile this sample app is specified in the Compile With. The theme specifies the Android UI style to apply for your app. On clicking next, the screen as shown in Figure 13 appears:



**Figure 13- Configure Project in Eclipse**

This screen is to configure the project and is left at default selections

On clicking next, the next screen Figure 14 will create a launcher icon for this app.



**Figure 14- Configure Attributes in Eclipse**

On clicking Next, for this sample project we select BlankActivity and click Next. The other details are left at default and click Finish.

The app by default stores hello world code and we need to just run it. The code looks as follows:

```
package com.masterproject.quickrecipes;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

The Manifest file will provides the characteristics of the app. The code looks like this:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />

<TextView
    android:layout_width="wrap_content"
```

```

    android:layout_height="wrap_content"
    android:text="@string/hello_world" />

```

```

</RelativeLayout>

```

The src directory contains all source files required for our application.

The res directory contains sub directories like layout, drawable-hdpi, menu, values etc.

The manifest file which describes the fundamental characteristics of the application and defines each of its components [7].

The app is run by clicking on the QuickrecipesApplication package-> RunAs -> Android application. On running this, the main activity class starts and loads a layout file that says "Hello World".

### 3.4 Running the Application

The application can be run either on device or on emulator.


#### 3.4.1. On Device

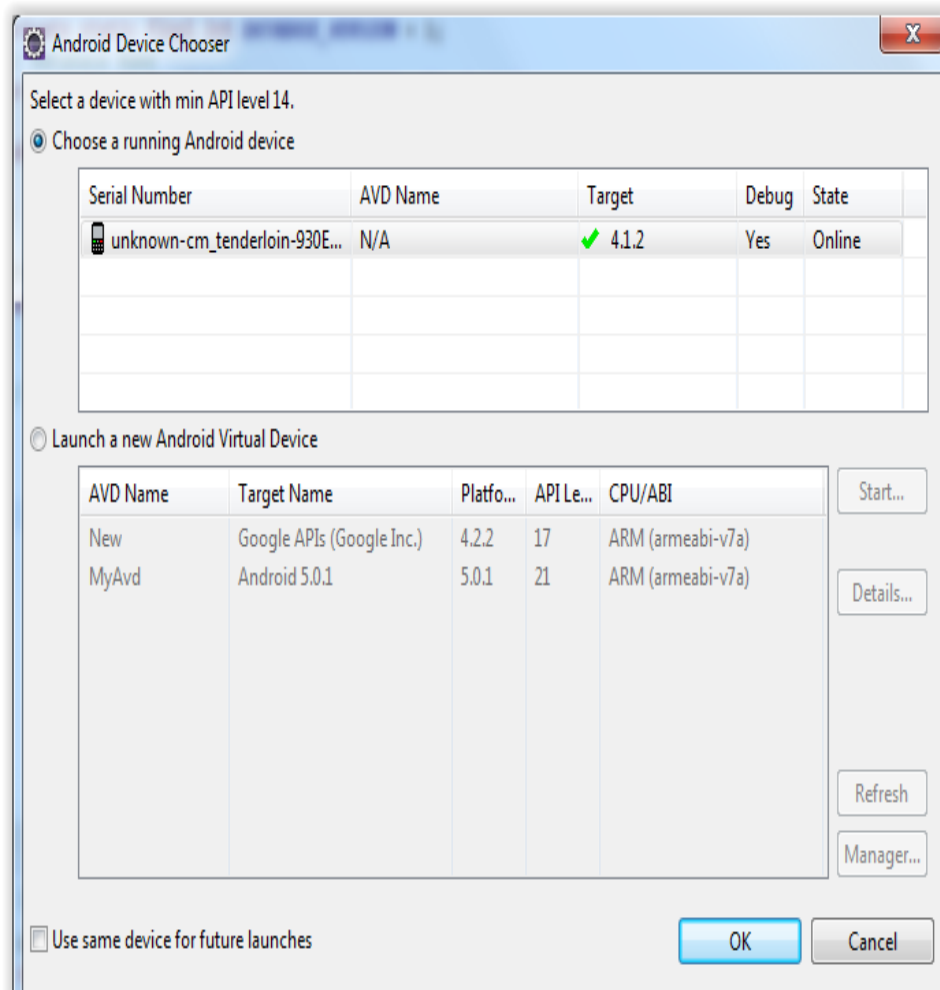
There are two ways the application can be run on device:

a. The android device needs to be connected to the laptop with USB cable. On connecting the device, option of downloading USB driver would appear in case the driver is not already installed. Then enable **USB debugging** on the device.

This can be found under Settings->Developer options

b. The second way is to run the app from Eclipse:

- On Eclipse, click on **Run**  from the toolbar.
- In the **Run as** window that appears, need to select **Android Application** and click **OK**.

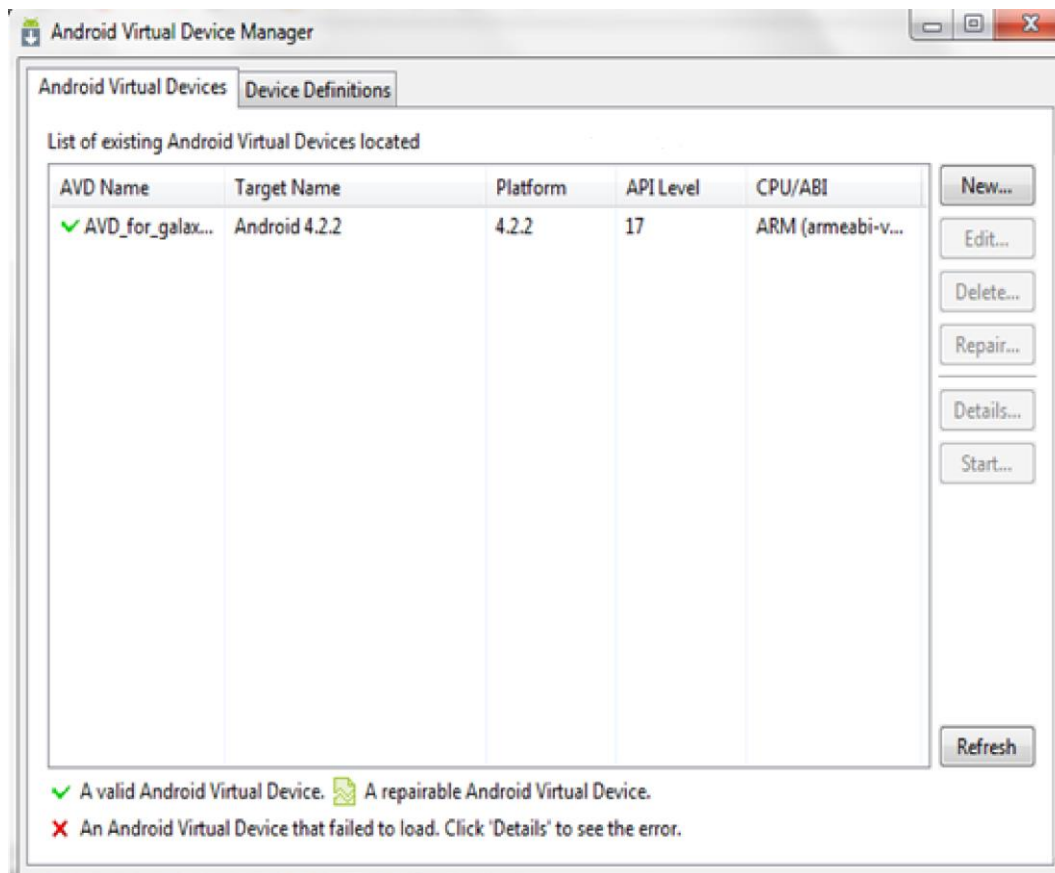


**Figure 15- Android Device Chooser**

This will cause eclipse to install the app on the connected device and start it as shown in Figure 15.

### **3.4.2. On Emulator**

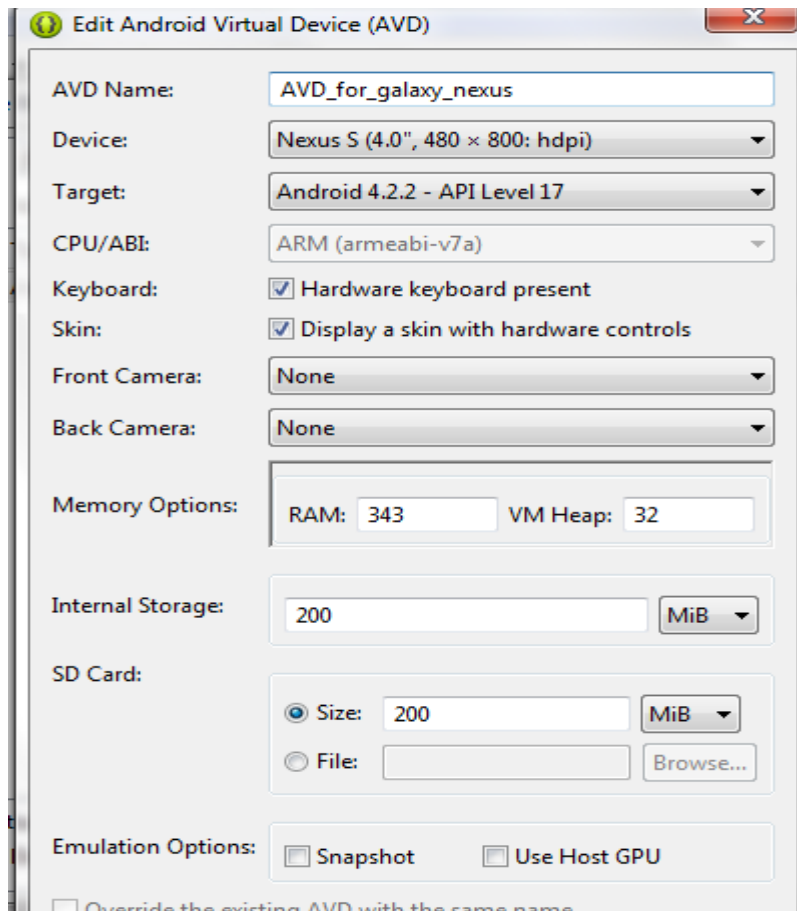
When the application needs to be run on Eclipse, then first step is to set the AVD i.e Android Virtual Device. The AVD is a device configuration for the Android emulator that allows to model different devices.



**Figure 16- Android Virtual Device Manager**


In order to create AVD, first need to launch the Android Virtual Device Manager by clicking on the toolbar. The next step is to click on 'New' in the Android Virtual Device Manager panel shown in Figure 16.

Next step is to fill in the details for the AVD like name, a platform target, an SD card size as shown in Figure 17



**Figure 17- Android Virtual Device Edit**

Click Okay. The new AVD will appear in the Android Virtual Device Manager screen. Select the new AVD and click **Start**. Wait for the emulator to boot up and then unlock the emulator screen.

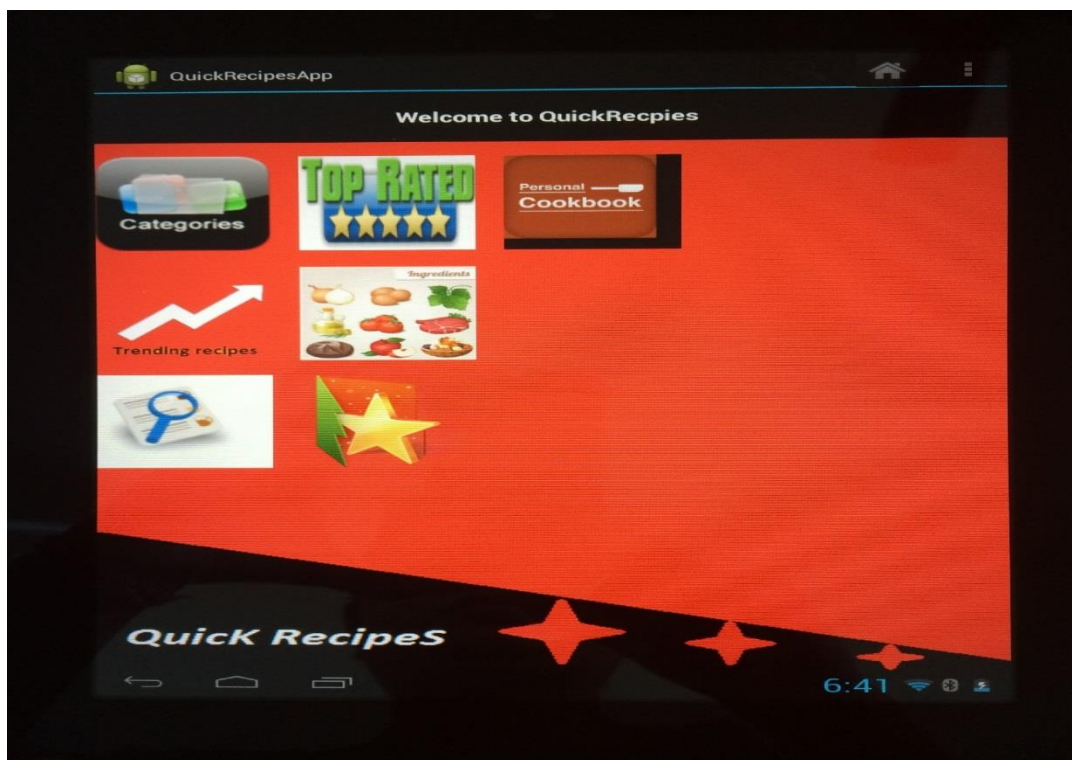
In order to run the app from Eclipse, click **Run**  from the toolbar. And then select **Android Application** and click **OK**. Eclipse will install the app on the AVD and starts it

## Chapter 4

### FEATURES IMPLEMENTATION

#### 4.1 Application Home Page

When user opens QuickRecipes application, application home page as shown in Figure 18 is displayed. User can perform any available option from application home page. The user can search recipes by clicking on categories, top rated, trending, ingredients and search image button. User can view recipes added as favorite by clicking on image button with star on home screen. User can also manage cookbook by creating new recipe, deleting recipe from personal cookbook icon from home screen.



**Figure 18- QuickRecipes Application Home Page**



For each screen on QuickRecipes application there is an associated activity. The main screen of QuickRecipes application has associated activity called “SearchOptionActivity”. Each activity has an associated XML document specifying screen layout. For application homepage activity\_serachoption.xml defines categories, ingredients, favorite, search, cookbook, trending, toprated image button. The following section provides code snippets and explanation.

When user clicks image button on application home page, “onClick” method is called by android. Switch-case construct is used for handling click for image buttons. For example, if button clicked is personal cookbook, code for cookbook button will be executed inside “onClick” method. The following code executes for search recipe and search top rated recipe image button clicked.

```
public void onClick(View v) {
    final Context context = this;
    int id=v.getId();
    switch(id) {

case R.id.searchrecipeButton:
    Intent intent1 =
    new Intent(context, SearchByTitleIngredientActivity.class);
    startActivity(intent1);
    break;

case R.id.searchrateButton:
    String SearchRecipe_By_Rate_url =
    "http://food2fork.com/api/search?key=${API_KEY}&sort=r";
    Intent intent3 = new Intent(context, RecipesInfoGridActivity.class);
    Bundle bundle3 = new Bundle();
    bundle3.putString("urlString", SearchRecipe_By_Rate_url);
    intent3.putExtras(bundle3);
    startActivity(intent3);
    break;
    }}
```

Intent are asynchronous messaging object which allows application components to interact with components from the same application as well as with components contributed by other applications. Intent can contain data via bundle. In above code, intent is created with its action set to open RecipesInfoGridActivity view. When user clicks top rated image button on home page, intent is passed as an argument in startActivity() method and new screen is displayed defining layout for RecipesInfoGridActivity class. Bundle is used to pass data from one activity to another via intent.



**Figure 19- Home Icon**

QuickRecipes application provides home icon on top right corner of application home page for easy navigation. Figure 19 illustrates home icon. User can navigate from any screen to application home page by clicking on this home icon. User can also use back button for navigation. The following code snippet is executed when home icon is clicked:

```
@Override
public boolean onOptionsItemSelected(MenuItem menuItem) {
    switch (menuItem.getItemId()) {
        case R.id.action_home:
            Intent homeIntent = new Intent(this, SearchOptionActivity.class);
            homeIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(homeIntent);
        }
    return (super.onOptionsItemSelected(menuItem));
}
```

## 4.2 Search Recipe

QuickRecipes application provides searching capability to user from cloud. Food2ForK API are consumed for searching recipes from internet[8]. These API provides recipe title, recipe image, recipe social rating, publisher's name and publisher webpage link in JSON format. In this application a class is created with name **ServiceHandler.java**. This class has a method which will make http request to get JSON data from Food2Fork API and returns a JSONObject. The code snippet for ServiceHandler class is as follows:

```
public class ServiceHandler {

    static String response = null;
    public final static int GET = 1;
    public final static int POST = 2;

    public ServiceHandler() {

    }
    public String makeServiceCall(String url, int method) {
        return this.makeServiceCall(url, method, null);
    }
    /**
     * Making service call
     * @url - url to make request
     */
    public String makeServiceCall(String url, int method,
        List<NameValuePair> params) {
        try {
            // http client
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpEntity httpEntity = null;
            HttpResponse httpResponse = null;
            // Checking http request method type
            if (method == POST) {
                HttpPost httpPost = new HttpPost(url);
                // adding post params
                if (params != null) {
                    httpPost.setEntity(new UrlEncodedFormEntity(params));
                }
            }
        }
    }
}
```

```

        httpResponse = httpClient.execute(httpPost);

    } else if (method == GET) {
        // appending params to url
        if (params != null) {
            String paramString = URLEncodedUtils.format(params, "UTF-8");
            url += "?" + paramString;
        }
        HttpGet httpGet = new HttpGet(url);
        httpResponse = httpClient.execute(httpGet);
    }
    httpEntity = httpResponse.getEntity();
    response = EntityUtils.toString(httpEntity);
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return response;
}
}

```

#### 4.2.1 Search by Title or Ingredients

QuickRecipes application provides an option to user to search recipe by typing recipe title or multiple ingredients separated by comma. Figure 20 shows Search image button on home page.



**Figure 20- Search Image Button**

Upon click on search recipe image button on home page `OnClick()` method is executed in class `SearchOptionActivity.java` for search button id defined in `activity_searchoption.xml` layout file. New activity `SearchByTitleIngredientActivity` started using `startActivity()` method. Code Snippet for `OnClick()` method is as follows:

```
public void onClick(View v) {
    final Context context = this;
    int id=v.getId();
    switch(id) {

case R.id.searchrecipeButton:

    Intent intent1 =
    new Intent(context, SearchByTitleIngredientActivity.class);
    startActivity(intent1);
    break;

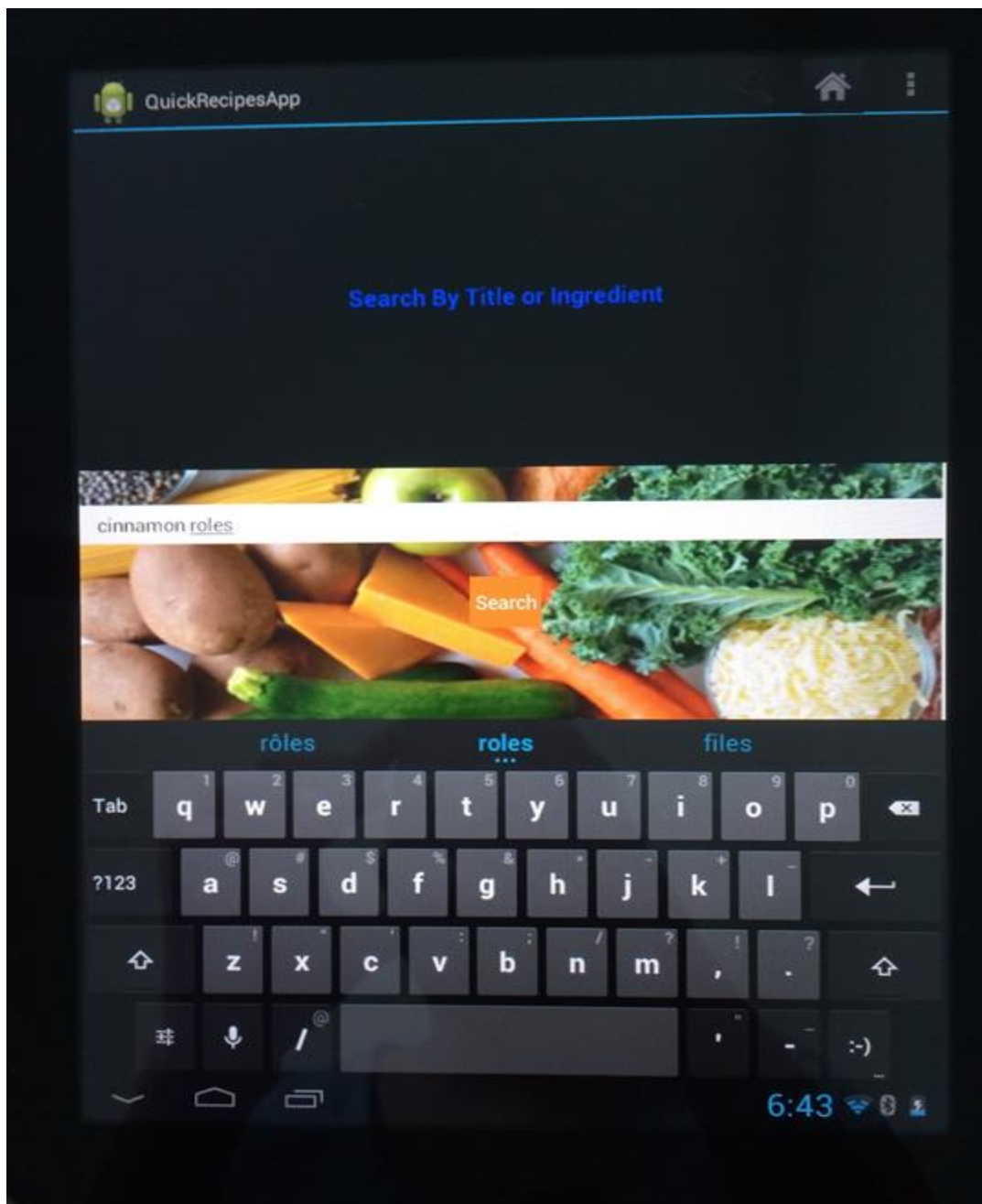
    }}

```

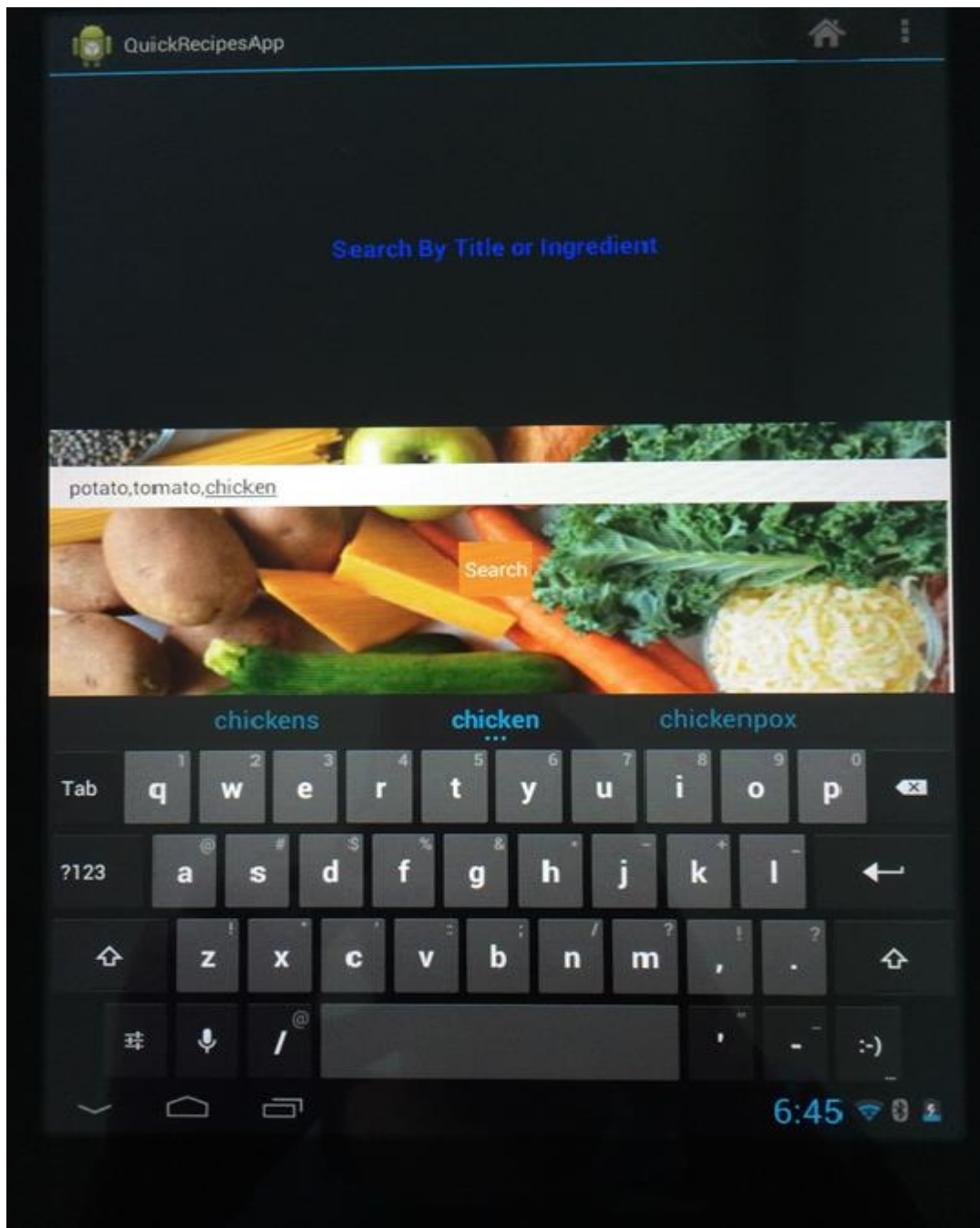
`SearchByTitleIngredientActivity` class is defined for performing search for user entered data. Layout xml for this class consist of an edittext field with request focus attribute set.

This attribute sends focus to edittext and allows keyboard to appear for typing. Data entered gets recorded and upon click on search button, search operation is performed.

Figure 21 shows user options to enter data, enter recipe title.



**Figure 21- Search by Title Page**



**Figure 22- Search by Ingredients List Page**

Figure 22 illustrates user can perform search with multiple ingredients by typing comma separated ingredient items.

The following code snippet executes when search button is clicked:

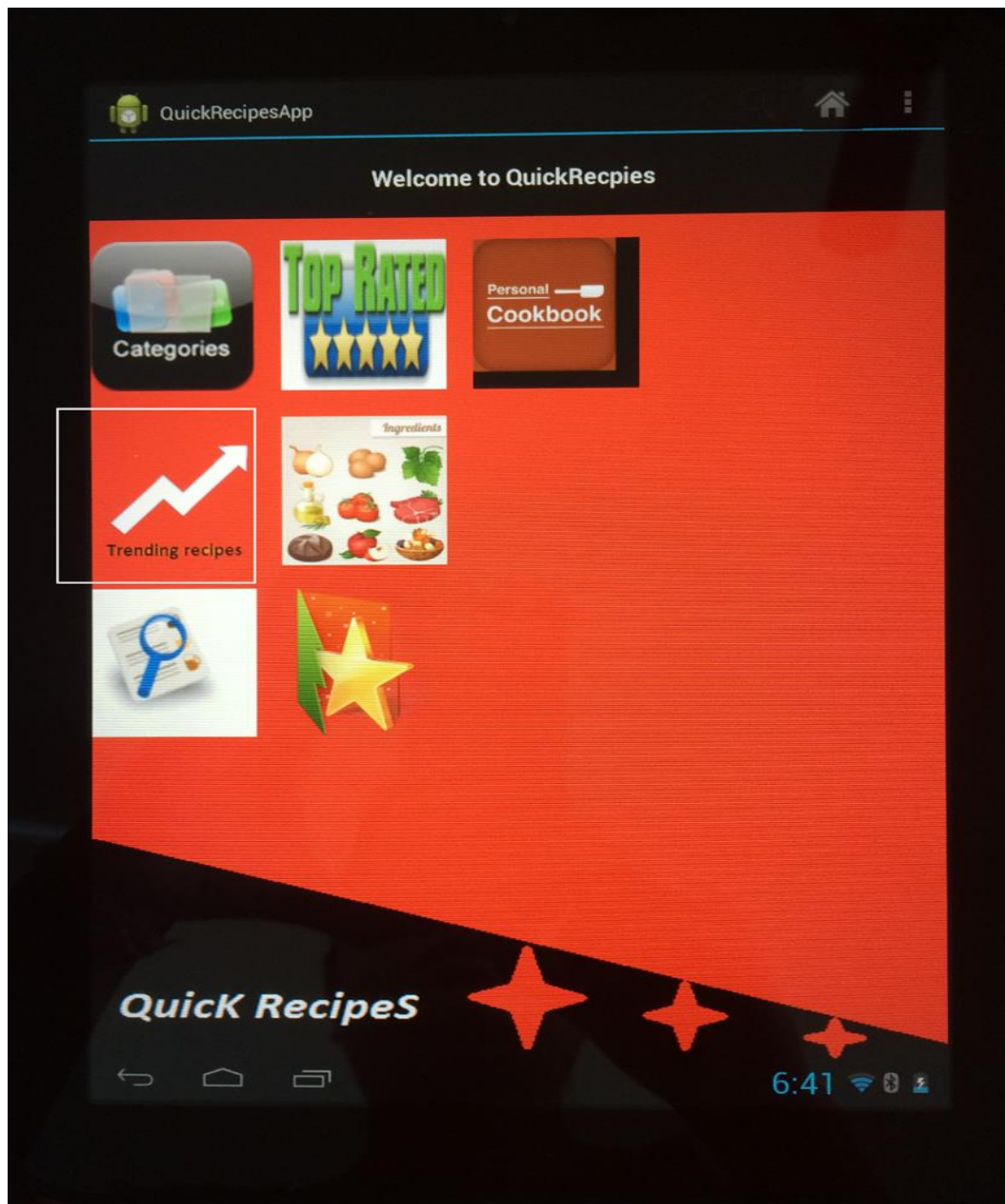
```
@Override
public void onClick(View arg0) {
    EditText et = (EditText) findViewById(R.id.editText);
    title_or_ingredient = et.getText().toString();
    System.out.print("Entered Text for Search ===== "+
        title_or_ingredient);
    String SearchRecipe__url = "http://food2fork.com/api/search?key=${API-KEY}
        &q="+title_or_ingredient;
    Intent intent = new Intent(this, RecipesInfoGridActivity.class);
    Bundle bundle = new Bundle();
    System.out.println("SearchRecipe__url=====>" +
        SearchRecipe__url);
    bundle.putString("urlString", SearchRecipe__url);
    intent.putExtras(bundle);
    startActivity(intent);
}
```

In above code snippet, user entered data is appended to search url and url is passed using intent to RecipesInfoGridActivity class where http calls are made to server using ServiceHandler class. If user entered recipe match is found on server, best match recipes are displayed.

#### 4.2.2 Search by Trend

QuickRecipes application provides an option to user to search trending recipes in just one click. Ability to perform search in few clicks makes application interface simple ease to use for user.





**Figure 23- Trending Recipes Search Button**

Figure 23 shows trending image button on application home page. User can search newly added recipes in cloud by clicking on trending recipes image button.

The following code snippet is executed when user clicks trending image button on home activity.

```

public void onClick(View v) {
    final Context context = this;
    int id=v.getId();

    switch(id) {

        case R.id.searchtrendButton:
            String SearchRecipe_By_Trend_url =
                "http://food2fork.com/api/search?key={ API-KEY } &sort=t";
            Intent intent2 = new Intent(context, RecipesInfoGridActivity.class);
            Bundle bundle2 = new Bundle();
            System.out.println("SearchRecipe_By_Trend_url=====>" +
                SearchRecipe_By_Trend_url);
            bundle2.putString("urlString", SearchRecipe_By_Trend_url);
            intent2.putExtras(bundle2);
            startActivity(intent2);
            break;

        }

    }

```

In above code snippet, searching is performed based on trendiness. The most recent recipes from publishers with a high trend score and recipes gaining popularity are displayed. Recipes are displayed in sorted order which is implemented by appending sort=t in formed search url. This search url is passed using intent to RecipesInfoGridActivity class where http calls are made to server using ServiceHandler class.

#### 4.2.3 Search by Rating

QuickRecipes application provides an option to user to search top rated recipes with social rating of 100 in just one click. User can perform search by clicking top rated image button on application home page.



**Figure 24- Top Rated Recipes Search Button**

Figure 24 shows top rated image button on application home page. The following code snippet

is executed when user clicks trending image button on home activity.

```
public void onClick(View v) {
    final Context context = this;
    int id=v.getId();

    switch(id) {
        case R.id.searchrateButton:
            String SearchRecipe_By_Rate_url =
                "http://food2fork.com/api/search?key={API_KEY} &sort=r";
            Intent intent3 = new Intent(context, RecipesInfoGridActivity.class);
            Bundle bundle3 = new Bundle();
            System.out.println("SearchRecipe_By_Rate_url=====>" +
                SearchRecipe_By_Rate_url);
```

```

        bundle3.putString("urlString", SearchRecipe_By_Rate_url);

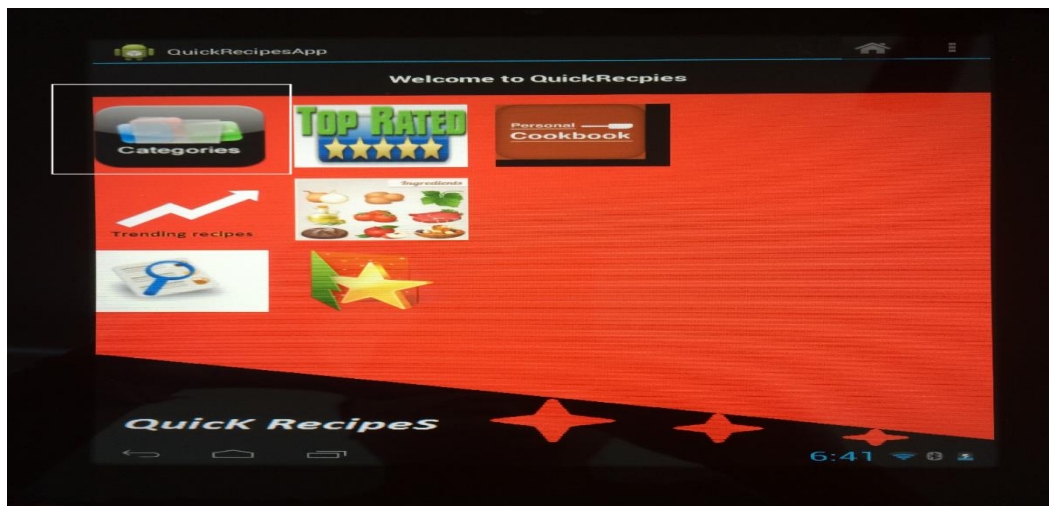
        intent3.putExtras(bundle3);
        startActivity(intent3);
        break;
    }}

```

In above code snippet, searching is performed based on rating. The rating is based on social media scores to determine best recipes. Recipes are displayed in sorted order which is implemented by appending sort=r in formed search url. This search url is passed using intent to RecipesInfoGridActivity class where http calls are made to server using ServiceHandler class and list of recipes are displayed with social rating of 100.

### 4.3 Search by Selection

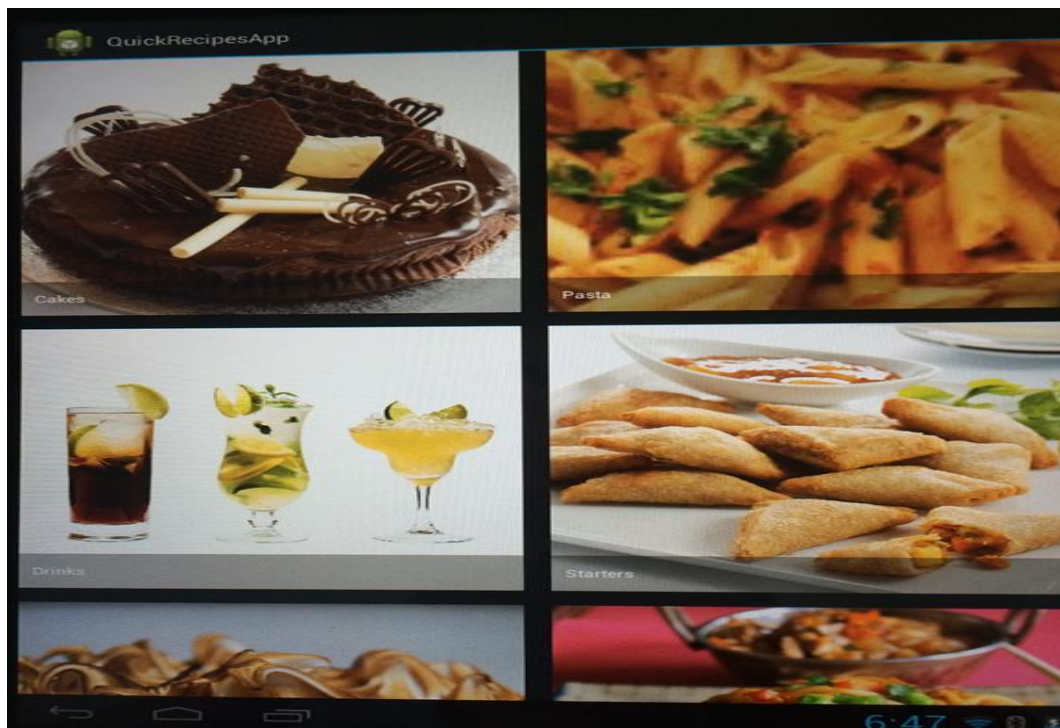
QuickRecipes application allows user to perform search based on selection. Recipes are grouped based on categories and ingredients required for cooking. Search by selection can be performed either by selecting category or ingredient. Figure 25 shows category image button.



**Figure 25- Categories Search Image Button**

### 4.3.1 Select Category

Recipes are grouped in categories. User has option to select recipe based on category. Figure 26 lists different categories available in QuickRecipes application. Categories include cakes, pie, pasta, drinks, noodles, family meal and salad. When user clicks categories image button on home page, CategoryActivity class is called and screen is displayed with recipe title and recipe image. Recipe image are loaded from drawable folder.



**Figure 26- Categories List Page**

The code snippet for CategoryActivity class is as follows:

```
public class CategoryActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gridcategorymain);
    }
}
```

```

final GridView gridView = (GridView)findViewById(R.id.gridviewcategory);
gridView.setAdapter(new GridViewImageAdapter(this));
gridView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView parent, View v, int position,
    long id) {

        Toast.makeText(
            getApplicationContext(),
            ((TextView) v.findViewById(R.id.text)).getText(),
            Toast.LENGTH_SHORT).show();
        String SearchCategory__url =
            "http://food2fork.com/api/search?key={ API_KEY }&q=" + ((TextView)
            v.findViewById(R.id.text)).getText();
        Intent intent = new Intent(CategoryActivity.this, RecipesInfoGridActivity.class);
        Bundle bundle = new Bundle();
        System.out.println("Search__url=====>" +
            SearchCategory__url);
        bundle.putString("urlString", SearchCategory__url);

        intent.putExtras(bundle);
        startActivity(intent);

    }
});
}
}

```

For displaying category list and tracking category selected android grid view is used. To connect data and grid view, GridViewImage adapter is used. Code snippet for declaring gridview view object is as:

```

final GridView gridView = (GridView)findViewById(R.id.gridviewcategory);

```

Code snippet for declaring the object holding data is as:

```

private final List<Item> mItems = new ArrayList<Item>();

```

mItems holds entry for category name and category image as shown below:

```

public GridViewImageAdapter(Context context) {
    mInflater = LayoutInflater.from(context);
}

```



```

mItems.add(new Item("Cakes", R.drawable.cakes));
mItems.add(new Item("Pasta", R.drawable.pasta));
mItems.add(new Item("Drinks", R.drawable.drinks));
mItems.add(new Item("Starters", R.drawable.starters));
mItems.add(new Item("Pie", R.drawable.pie));
mItems.add(new Item("Noodles", R.drawable.noodles));
mItems.add(new Item("FamilyMeal", R.drawable.familymeal));
mItems.add(new Item("Salad", R.drawable.salad));

}

```

GridViewImageAdapter is connected to grid view object using setAdapter() method in CategoryActivity class. Adapter uses getView() method to display view on screen. The code snippet for getView() looks as:

```

@Override
public View getView(int i, View view, ViewGroup viewGroup) {
    View v = view;
    ImageView picture;
    TextView name;

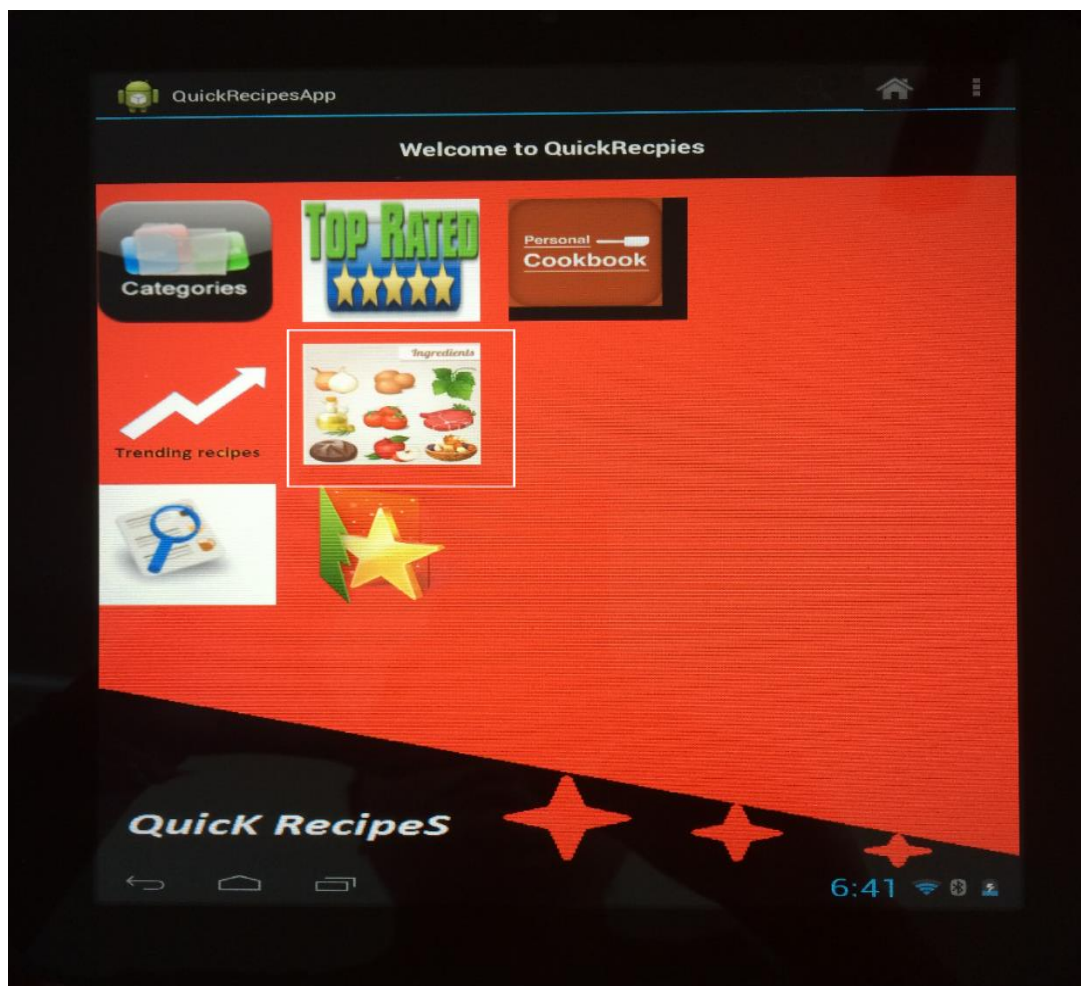
    if (v == null) {
        v = mInflater.inflate(R.layout.activity_gridcategoryitem, viewGroup, false);
        v.setTag(R.id.picture, v.findViewById(R.id.picture));
        v.setTag(R.id.text, v.findViewById(R.id.text));
    }
    picture = (ImageView) v.getTag(R.id.picture);
    name = (TextView) v.getTag(R.id.text);
    Item item = getItem(i);
    picture.setImageResource(item.drawableId);
    name.setText(item.name);
    return v;
}

```

To handle category selected, onItemClick() is implemented. Upon clicking category toast message is displayed with category name on screen and url is formed with appending category name. Search is performed by making server call from RecipesInfoGridActivity class and result containing recipe list gets displayed.

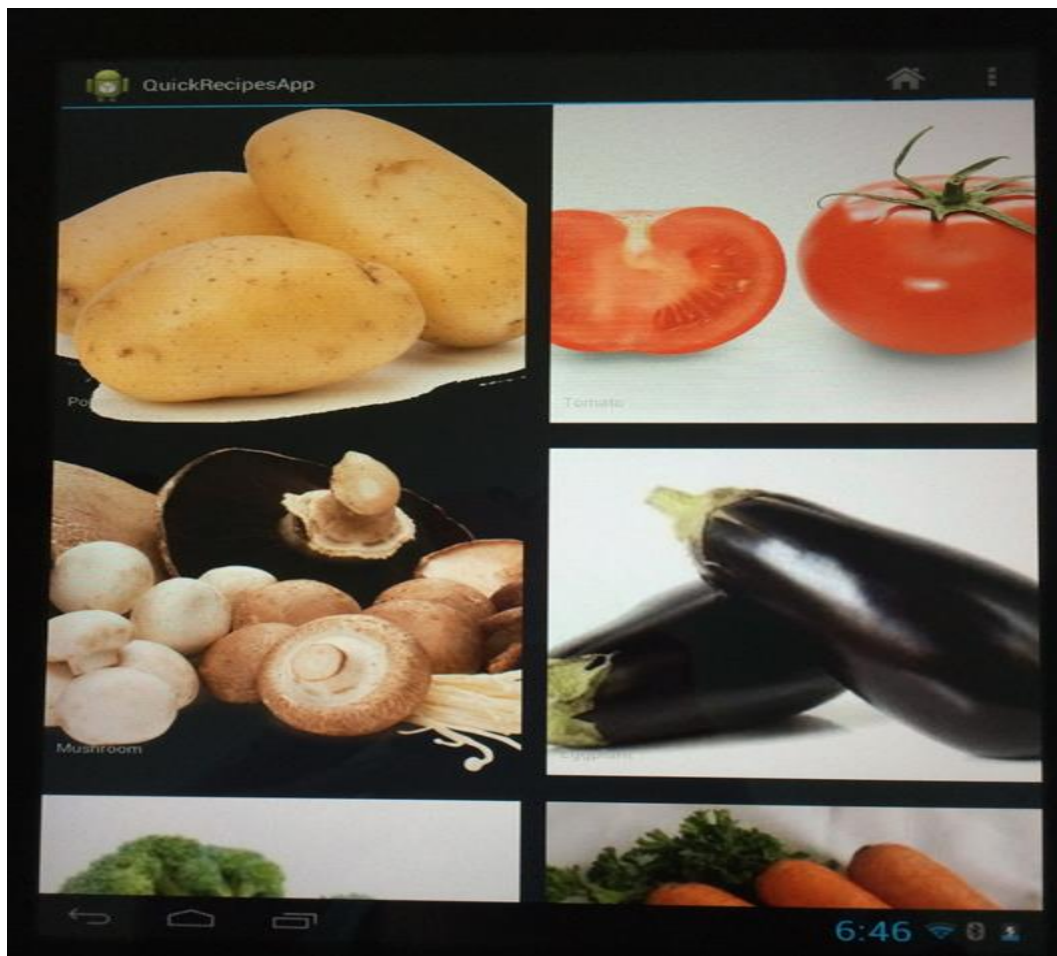
### 4.3.2 Select Ingredients

QuickRecipes application allows user to search recipe based on ingredient. Figure 27 shows ingredients image button on application home page. When user clicks ingredients image button on home screen, VegetablesGridViewActivity is called and ingredient list with image and title are displayed for user to select and search recipe based on selected ingredient. Gridview layout is used for display ingredient options. VegetableImageAdapter class is implemented to connect gridview object for display.



**Figure 27- Ingredients Search Image Button**





**Figure 28- Ingredients List Page**

Figure 28 lists ingredients available for user to perform ingredients based search by  
 Selecting ingredient from available list.Code snippet for VegetableGridViewActivity class  
 is as:

```
@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gridvegetablesmain);

    final GridView gridView = (GridView) findViewById(R.id.vegetablegridView);
```

```

gridView.setAdapter(new VegetableImageAdapter(this));

gridView.setChoiceMode(GridVew.CHOICE_MODE_SINGLE);

gridView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView parent, View v, int position,
        long id) {

        Toast.makeText(
            getApplicationContext(),
            ((TextView) v.findViewById(R.id.grid_item_vegetablelabel)).getText(),
            Toast.LENGTH_SHORT).show();
        String SearchCategory__url =
            "http://food2fork.com/api/search?key={ API_KEY } &q=" + ((TextView)
                v.findViewById(R.id.grid_item_vegetablelabel)).getText().toString().toLowerCase();
        Intent intent = new Intent(VegetablesGridViewActivity.this,
            RecipesInfoGridActivity.class);
        Bundle bundle = new Bundle();
        System.out.println("Search__url=====>" +
            SearchCategory__url);
        bundle.putString("urlString", SearchCategory__url);

        intent.putExtras(bundle);
        startActivity(intent);

    }
});

}

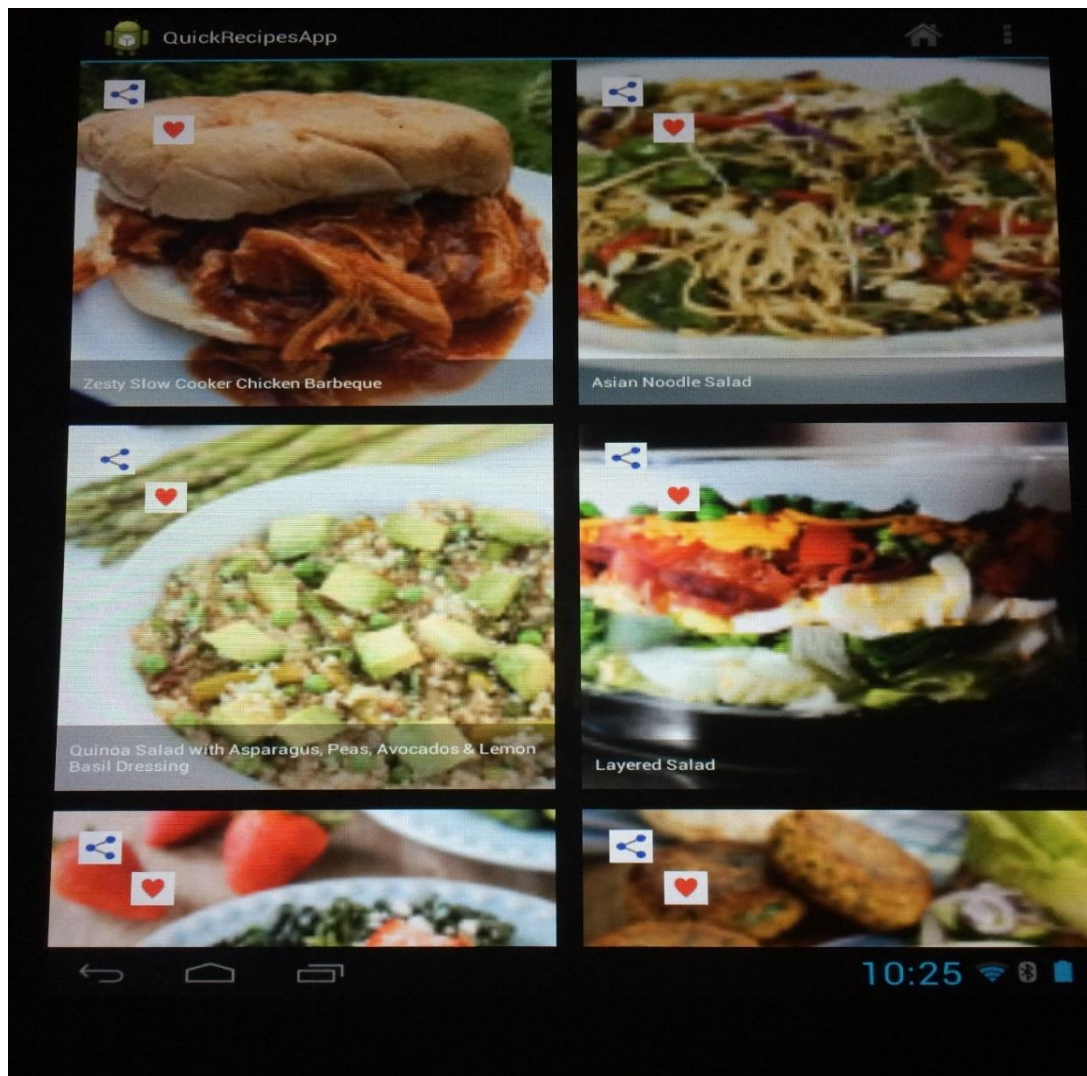
```

## 4.4 View Recipes

QuickRecipes application allows user to view recipe list with images and title. Single recipe can be viewed with images, recipe title, and details cooking directions.

### 4.4.1 View List of Recipes

QuickRecipes application allows user to view recipe list with images and title.



**Figure 29- View Recipes List**

Figure 29 shows recipes list. RecipeInfoGridActivity class is implemented for displaying recipe list. ImageLoader class is implemented for displaying images efficiently from drawable folder and from internet by providing url. If image is not available default image is displayed. Code Snippet is as follows:

```
public ImageLoader(Context context) {
    fileCache = new FileCache(context);
    executorService = Executors.newFixedThreadPool(5);
}
```

```

final int stub_id = R.drawable.temp_img;

public void DisplayImage(String url, ImageView imageView) {
    imageViews.put(imageView, url);
    Bitmap bitmap = memoryCache.get(url);
    if (bitmap != null)
        imageView.setImageBitmap(bitmap);
    else {
        queuePhoto(url, imageView);
        imageView.setImageResource(stub_id);
    }
}

```

RecipeInfoGridActivity uses background async task to execute the queries specific for the individual functionalities. The main purpose of the AsyncTask is to allow background operations and publish results on the UI thread without having to manipulate threads and/or handlers. When an asynchronous task is executed, the task goes through following steps: The onPreExecute() is invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface. During pre execute step in the application, progress dialog is shown before starting background thread.

```

@Override
protected void onPreExecute() {
    super.onPreExecute();
    // Create a progressdialog
    mProgressDialog = new ProgressDialog(RecipeInfoGridActivity.this);
    // Set progressdialog title
    mProgressDialog.setTitle("Quick Recipes");
    // Set progressdialog message
    mProgressDialog.setMessage("Loading...");
    mProgressDialog.setIndeterminate(false);
    // Show progressdialog
    mProgressDialog.show();
}

```

The `doInBackground(Params...)` is invoked on the background thread immediately after `onPreExecute()` finishes executing. This step is used to perform background computation that can take a long time. Here the parameters of the asynchronous task are passed to this step. The results returned by this step will be passed to the last step.

```
protected Void doInBackground(Void... params) {
    // Create an array
    try {
        ArrayList<HashMap<String, String>>();

        Intent intent = getIntent();

        String LOGINURL = intent.getExtras().getString("urlString");
        System.out.println("source_url =====>"
            + LOGINURL);

        ServiceHandler sh = new ServiceHandler();

        // Making a request to url and getting response
        String jsonstr = sh.makeServiceCall(LOGINURL,
            ServiceHandler.GET);

        Log.d("Response: is here-----", "> " + jsonstr);

        JSONObject jsonObj = new JSONObject(jsonstr);

        System.out.println("jsonobject" + jsonObj);

        String count = jsonObj.getString("count");
        // Locate the array name in JSON
        JSONArray jsonarray = jsonObj.getJSONArray("recipes");

        System.out.println("jsonarray" + jsonarray);
        // for (int i = 0; i < 10; i++) {
        if (count != "0") {
            for (int i = 0; i < jsonarray.length(); i++) {
                HashMap<String, String> map = new HashMap<String, String>();
                JSONObject jsonobject = jsonarray.getJSONObject(i);

                // Retrieve JSON Objects - newlyadded
```

```

        map.put("title", jsonobject.getString("title"));

        map.put("image_url", jsonobject.getString("image_url"));

        map.put("recipe", jsonobject.getString("source_url"));

        // Set the JSON Objects into the array
        arraylist.add(map);
    }
    } else {
        System.out.print("No Matching recipe found with title");
        Intent intent1 = new Intent(RecipesInfoGridActivity.this,
        NoMatchActivity.class);
        startActivity(intent1);
    }

    System.out.println("arraylist =====" + arraylist);
    } catch (JSONException e) {
        Log.e("Error", e.getMessage());
        e.printStackTrace();
    }
    return null;
}

```

The `onPostExecute(Result)` is invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter. Once the background task is completed, this step is used to dismiss the progress dialog from `onPreExecute()` method.

```

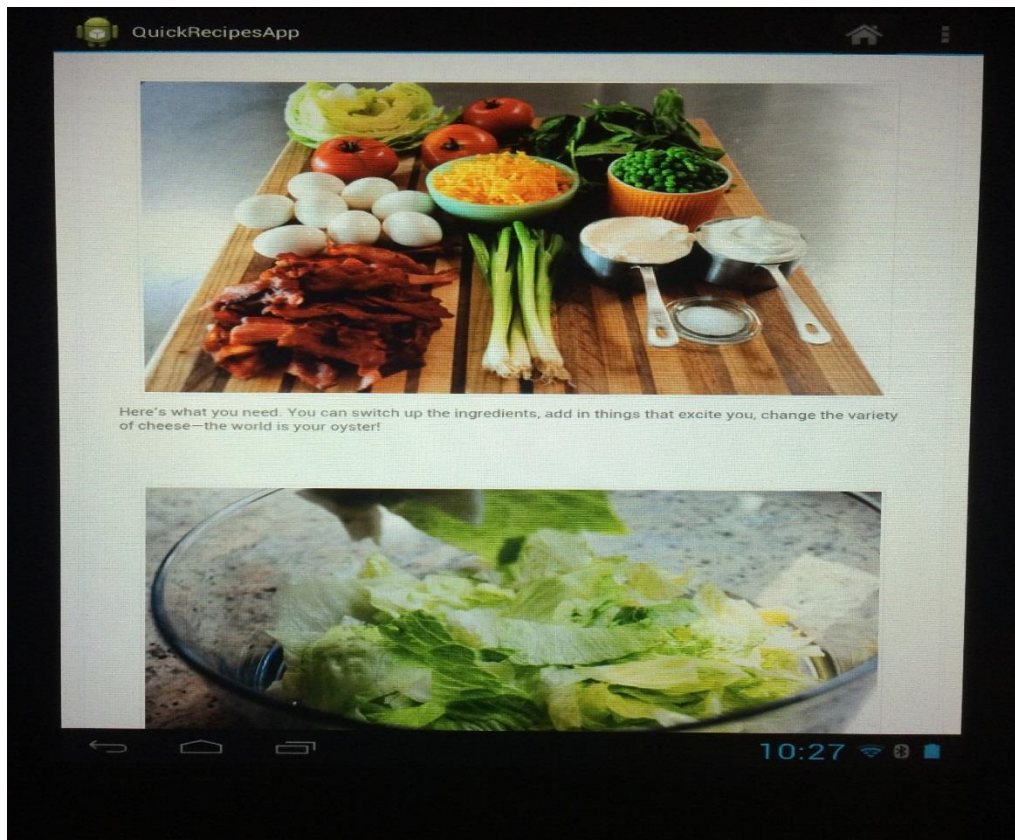
protected void onPostExecute(Void args) {
    // Locate the gridview in gridview_main.xml
    gridview = (GridView) findViewById(R.id.gridview);
    // Pass the results into GridViewAdapter.java
    adapter = new GridViewAdapter(RecipesInfoGridActivity.this,
    arraylist);
    // Set the adapter to the GridView
    gridview.setAdapter(adapter);
    // Close the progressdialog
    mProgressDialog.dismiss();
}

```



#### 4.4.2 View Single Recipe

QuickRecipes application uses android webview to display recipe details from publisher website. Application supports scrolling of page. Figure 30 illustrated displayed recipe.



**Figure 30- View Recipe Details**

SingleItemView class is implemented for displaying recipe details. onCreate() intent is parsed to receive recipe url, browser is initiated with settings and url is loaded. Code Snippet is as follows:

```
public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    // Get the view from singleitemview.xml
    setContentView(R.layout.singleitemview);
```

```

Intent i = getIntent();
source_url = i.getStringExtra("source_url");
source_url = super.getIntent().getExtras().getString("urlString");
System.out.println("source_url
=====>" + source_url);

// initialize the browser object
WebView browser = (WebView) findViewById(R.id.webView1);

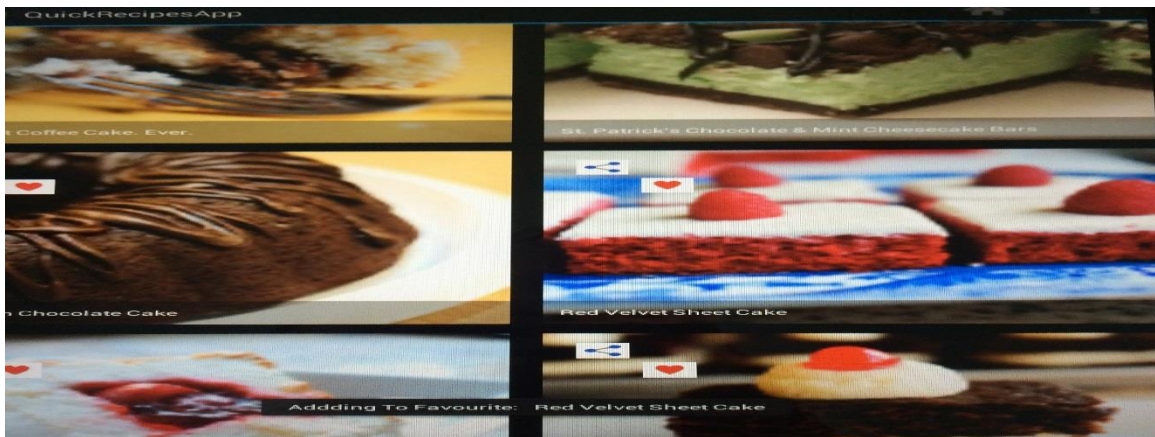
browser.getSettings().setLoadWithOverviewMode(true);
browser.getSettings().setUseWideViewPort(true);

try {
    // load the url
    browser.loadUrl(source_url);
} catch (Exception e) {
    e.printStackTrace();
}
}

```

#### 4.5 Save Recipe as Favorite

QuickRecipes application allows user to save recipe as favorite. When user performs search operation, result is list of recipes. Each recipe in list has favorite button on it. User can add recipe by clicking on favorite button. Heart in Figure 31 is add to favorite button.



**Figure 31- Adding Recipe to Favorite**

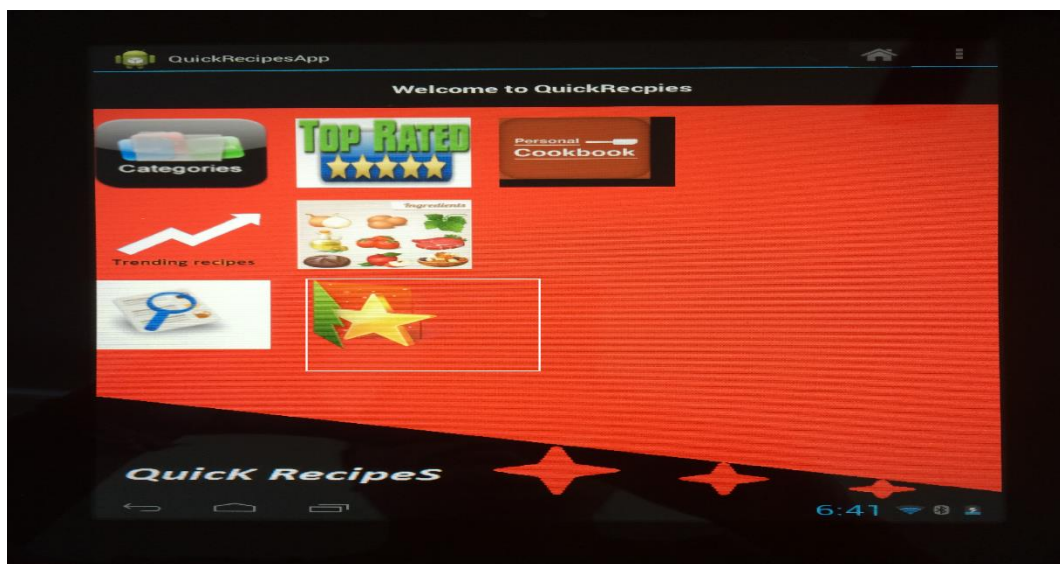


Code Snippet for add to favorite button implementation:

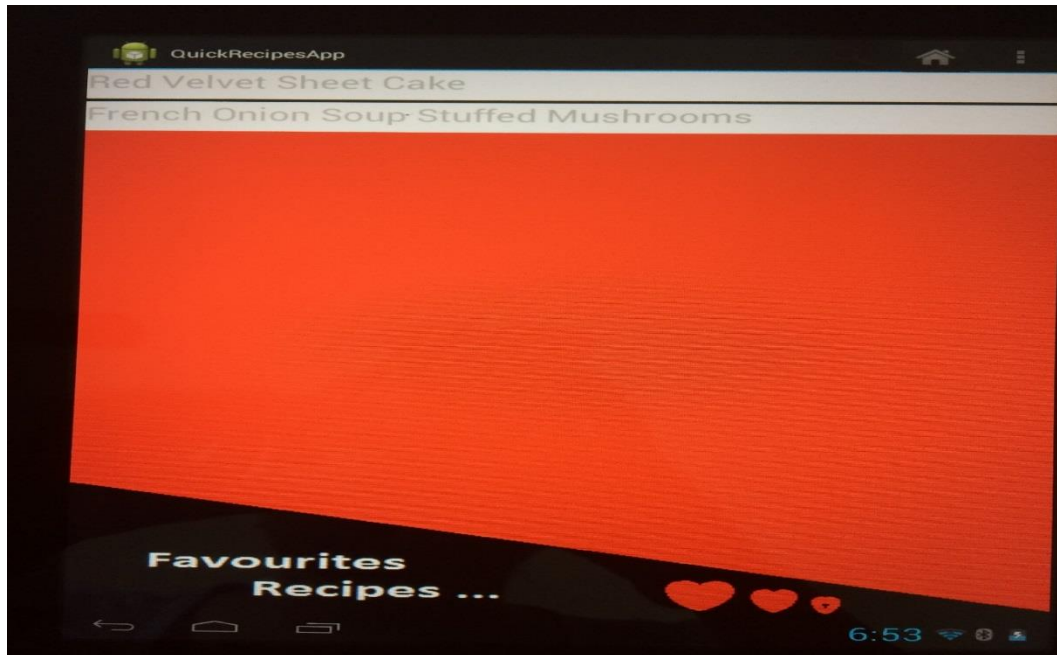
```
addtofavourite.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        resultp = data.get(position);
        System.out.println("Position Clicked =" + resultp);
        Log.i("Add to Favourite Button Clicked", "*****");
        Toast.makeText(context, "Adding To Favourite: "+resultp.get("title"),
            Toast.LENGTH_LONG).show();

        addtofavouritelist.add(resultp.get("title"));
        SharedPreferences.Editor editor = prefs.edit();
        set.addAll(addtofavouritelist);
        editor.putStringSet("title", set);
        editor.commit();
    }
});
```

QuickRecipes application uses shared preferences provided by android to store selected favorite recipe title. Recipe title are stored in key value pair. User can view list of favorite recipes by clicking on favorite icon on home page as shown in Figure 32.



**Figure 32- Favorite Image Button on Application Home Page**



**Figure 33- View Favorite Recipes List**

Figure 33 lists favorite recipes. For displaying favorite list android list view is used. To connect data and list view, android in built array adapter object is used.

Code snippet for declaring listview view object is as:

```
ListView addtofavourite_List = (ListView) findViewById(R.id.list_addtofavourite);
```

Code snippet for declaring the object holding data is as:

```
ArrayList<String> arraylist = new ArrayList<String>();
```

For retrieving values from shared preferences following code snippet is used:

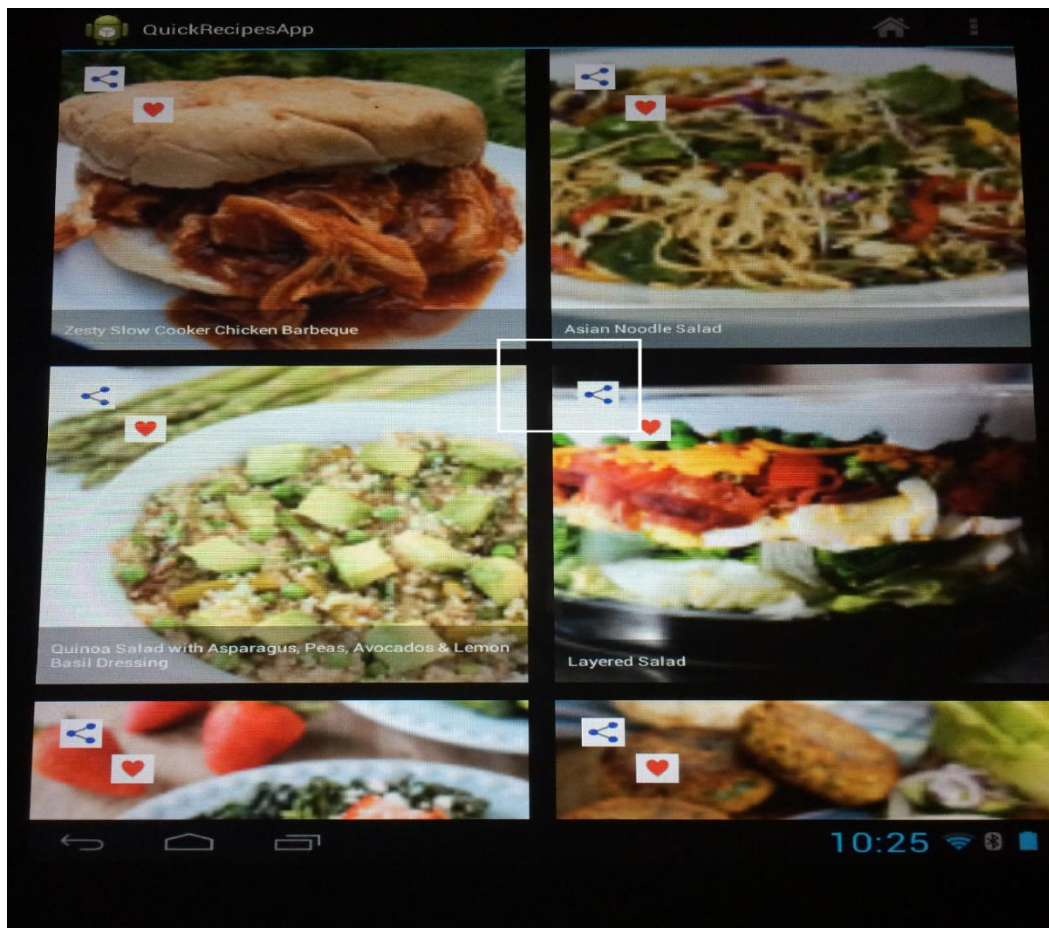
```
//Retrieve the values
Set<String> set = prefs.getStringSet("title", null);
set.toArray();
System.out.println("set.toArray().toString().toString();" + set);
arraylist.addAll(set);
```

ArrayAdapter is connected to list view object using setAdapter() method in AddToFavoriteActivity class.

```
ArrayAdapter<String> adapter = new ArrayAdapter(this,  
R.layout.activity_addtofavouriteitem,arraylist ) ;
```

#### 4.6 Share Recipes

QuickRecipes application allows user to share recipe on social media. When user performs search operation, result is list of recipes. Each recipe in list has share button on it. User can share recipe by clicking on share button. Figure 34 shows share button.



**Figure 34- Sharing with Share Button**

Code snippet implementation for share button:

```
share.setOnClickListener(new OnClickListener()
    @Override
    public void onClick(View v) {
        resultp = data.get(position);
        Log.i("Share Button Clicked", "*****");
        Toast.makeText(context, "Sharing recipe: " + resultp.get("title"),
            Toast.LENGTH_LONG).show();
        System.out.println("Sharing recipe: " + resultp.get("title"));
        String title = resultp.get("title");
        Bundle bundle = new Bundle();
        String url = resultp.get("recipe");
        System.out.println("URL=====>" + url);
        bundle.putString("urlString", url);

        List<Intent> targetedShareIntents = new ArrayList<Intent>();

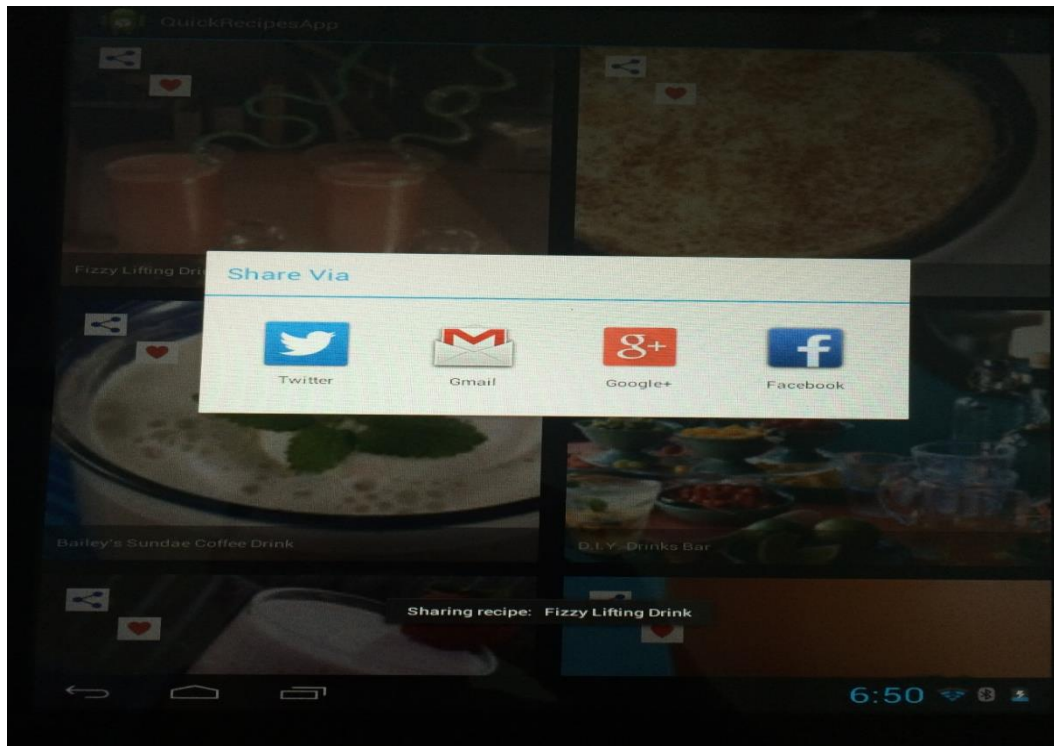
        Intent facebookIntent = getShareIntent("facebook", "subject", url);
        if(facebookIntent != null)
            targetedShareIntents.add(facebookIntent);

        Intent twitterIntent = getShareIntent("twitter", "subject", url);
        if(twitterIntent != null)
            targetedShareIntents.add(twitterIntent);

        Intent gmailIntent = getShareIntent("gmail", "subject", url);
        if(gmailIntent != null)
            targetedShareIntents.add(gmailIntent);

        Intent googlePlusIntent = getShareIntent("plus", "subject", url);
        if(googlePlusIntent != null)
            targetedShareIntents.add(googlePlusIntent);

        Intent chooser = Intent.createChooser(targetedShareIntents.remove(0), "Share
        Via");
        chooser.putExtra(Intent.EXTRA_INITIAL_INTENTS,
            targetedShareIntents.toArray(new Parcelable[]{}));
        context.startActivity(chooser);
    }
});
```



**Figure 35- Sharing Options**

Upon click on share button toast message is displayed sharing recipe with recipe name.

Alert dialog view containing facebook, twitter, google+ and gmail is displayed for user to select option from dialog box as shown in Figure 35. Intent is used for passing recipe link to shared option. Package manager object is used for sharing intent using `getShareIntent()` method.

```
private Intent getShareIntent(String type, String subject, String text ) {
    PackageManager packageManager = context.getPackageManager();
    boolean found = false;
    Intent share = new Intent(android.content.Intent.ACTION_SEND);
    share.setType("text/plain");

    // gets the list of intents that can be loaded.
    List<ResolveInfo> resInfo = packageManager.queryIntentActivities(share, 0);
    System.out.println("resinfo: " + resInfo);
    if (!resInfo.isEmpty()){
        for (ResolveInfo info : resInfo) {
```



```

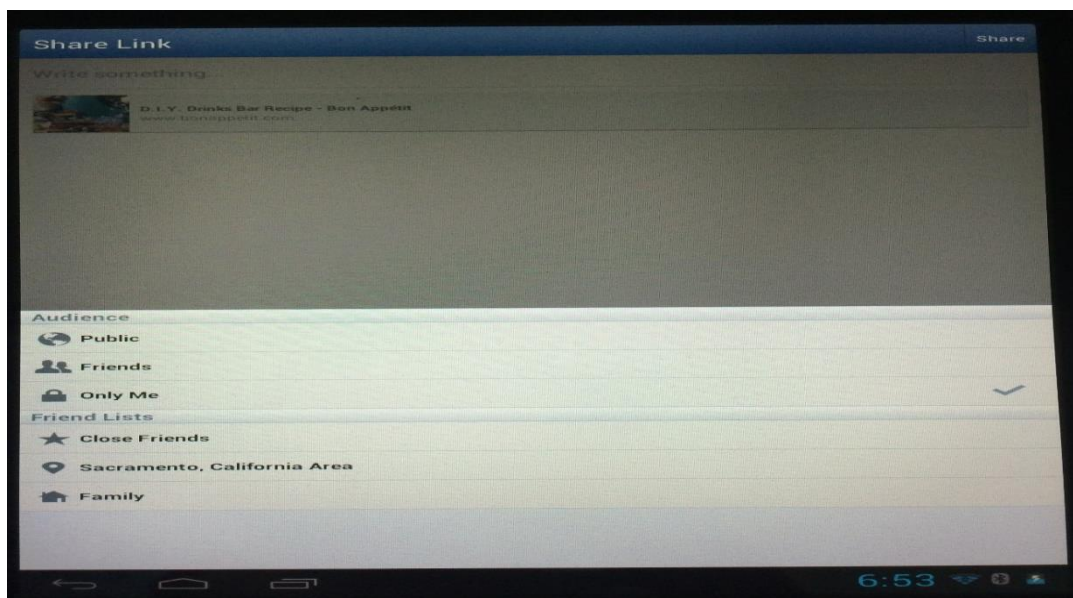
        if (info.activityInfo.packageName.toLowerCase().contains(type) ||
            info.activityInfo.name.toLowerCase().contains(type) ) {
            share.putExtra(Intent.EXTRA_SUBJECT, subject);
            share.putExtra(Intent.EXTRA_TEXT, "check out this recipe: " + text);
            share.setPackage(info.activityInfo.packageName);
            found = true;
            break;
        }
    }
    if (!found)
        return null;

    return share;
}
return null;
}

```

#### 4.6.1 Share on Facebook

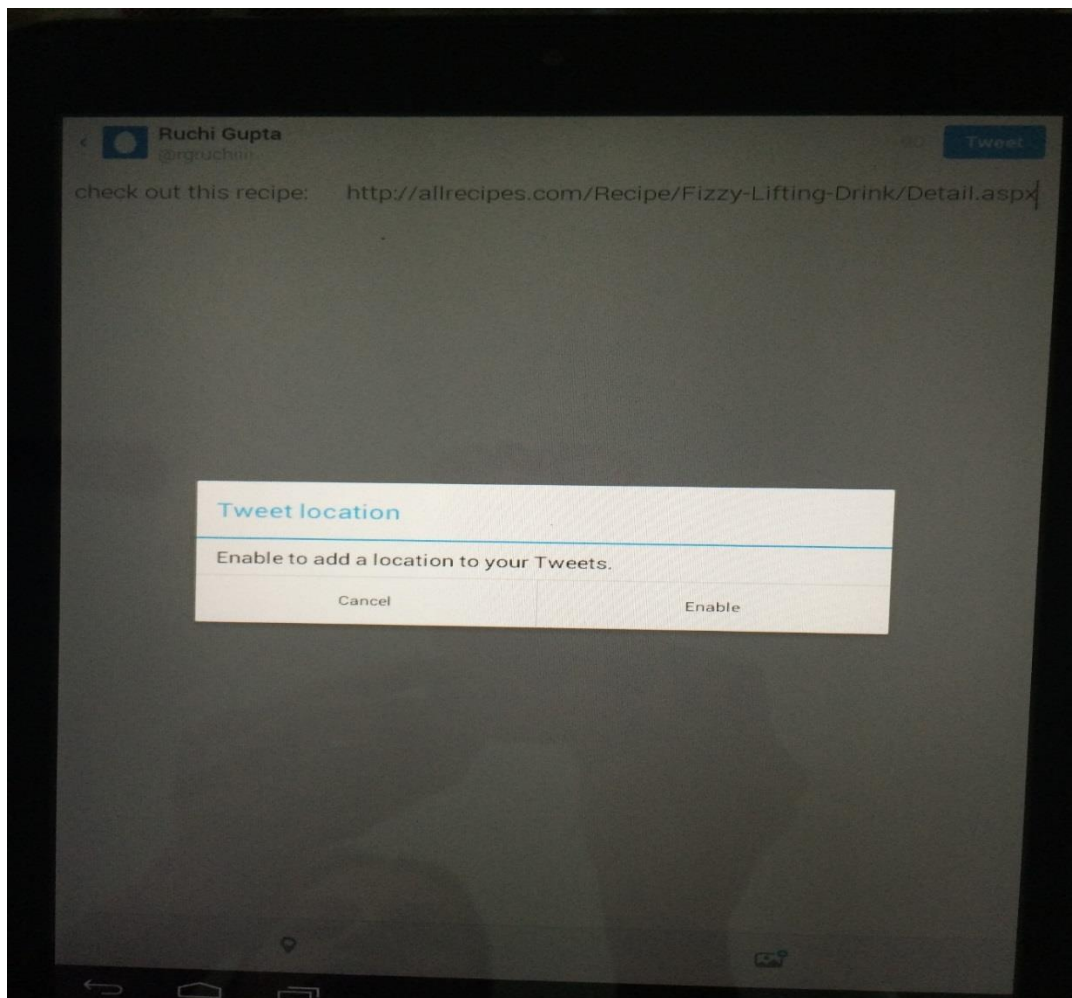
Upon Click on Facebook button, QuickRecipes application lets user to add a personalized message to links before sharing on their timeline, in groups or to friends as illustrated in Figure 36.



**Figure 36- Sharing Recipe on Facebook**

#### 4.6.2 Share on Twitter

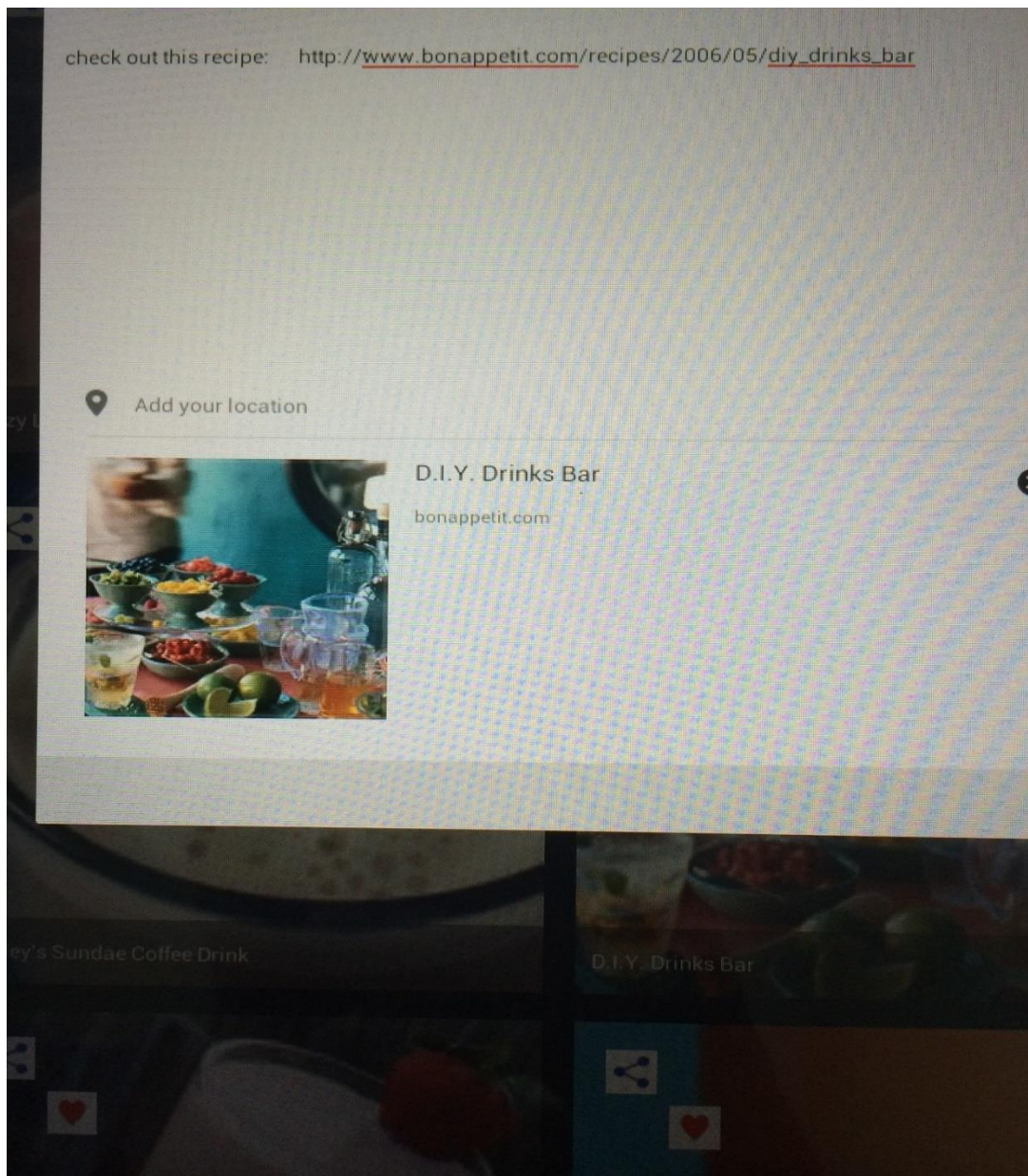
User can share recipe link on Twitter by clicking on Twitter icon from dialog box. Package Manager object, ResolveInfo object and intents are used for sharing. “com.Twitter.Android” package is used to verify if twitter is installed on application. If result is true, user enter login information and recipe link is passed through putExtra() method in app. User can enable location during tweet with enable option in dialog box as shown in Figure 37.



**Figure 37- Sharing Recipe on Twitter**

#### 4.6.3 Share on Google+

User can share recipe link on Google+ by clicking on Google+ icon from dialog box. If user not logged in, QuickRecipes application asks for credentials and then share recipe link in user circles. User can also add location during sharing as illustrated in Figure 38.

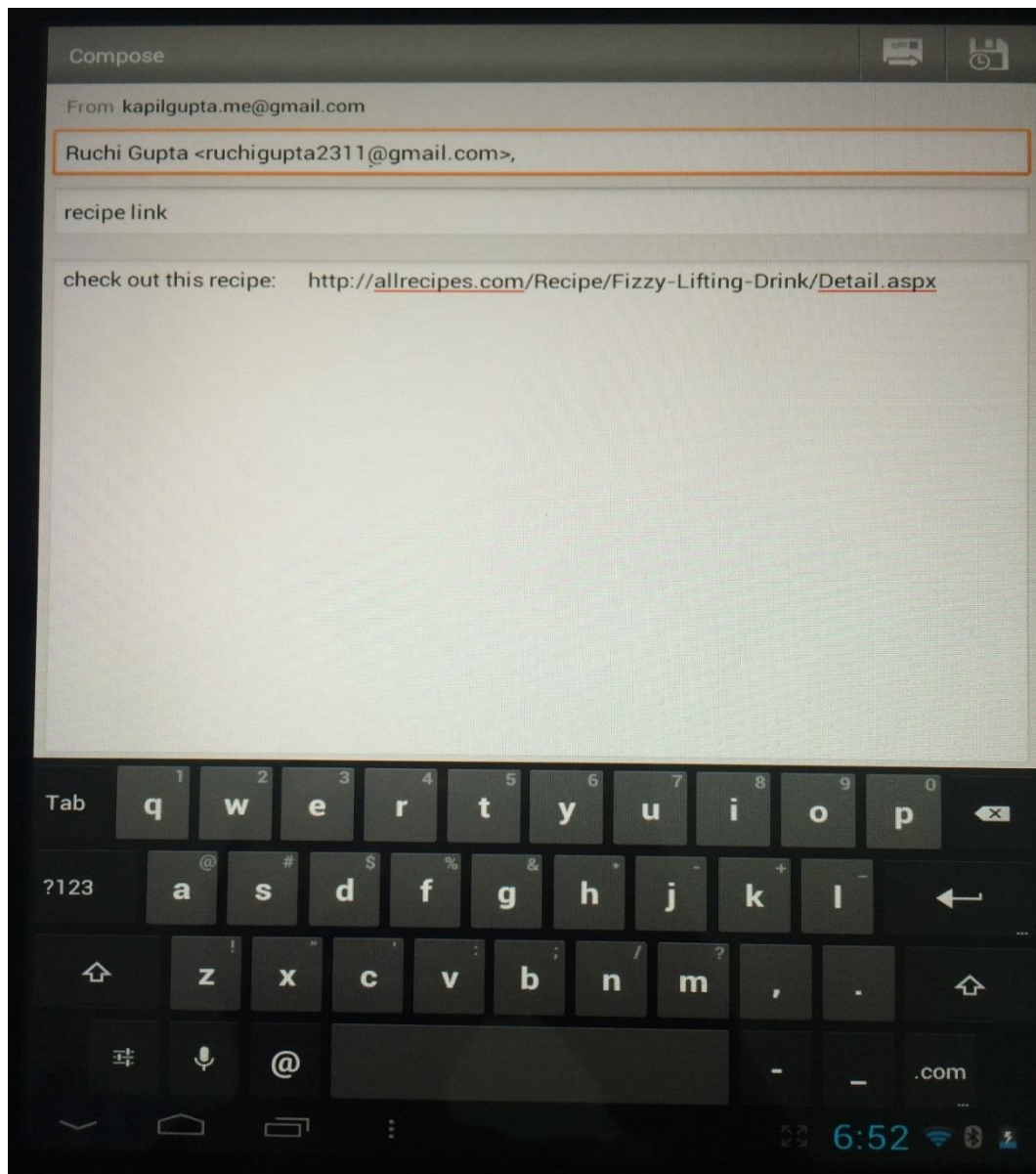


**Figure 38- Sharing Recipe on Google+**



#### 4.6.4 Share on Gmail

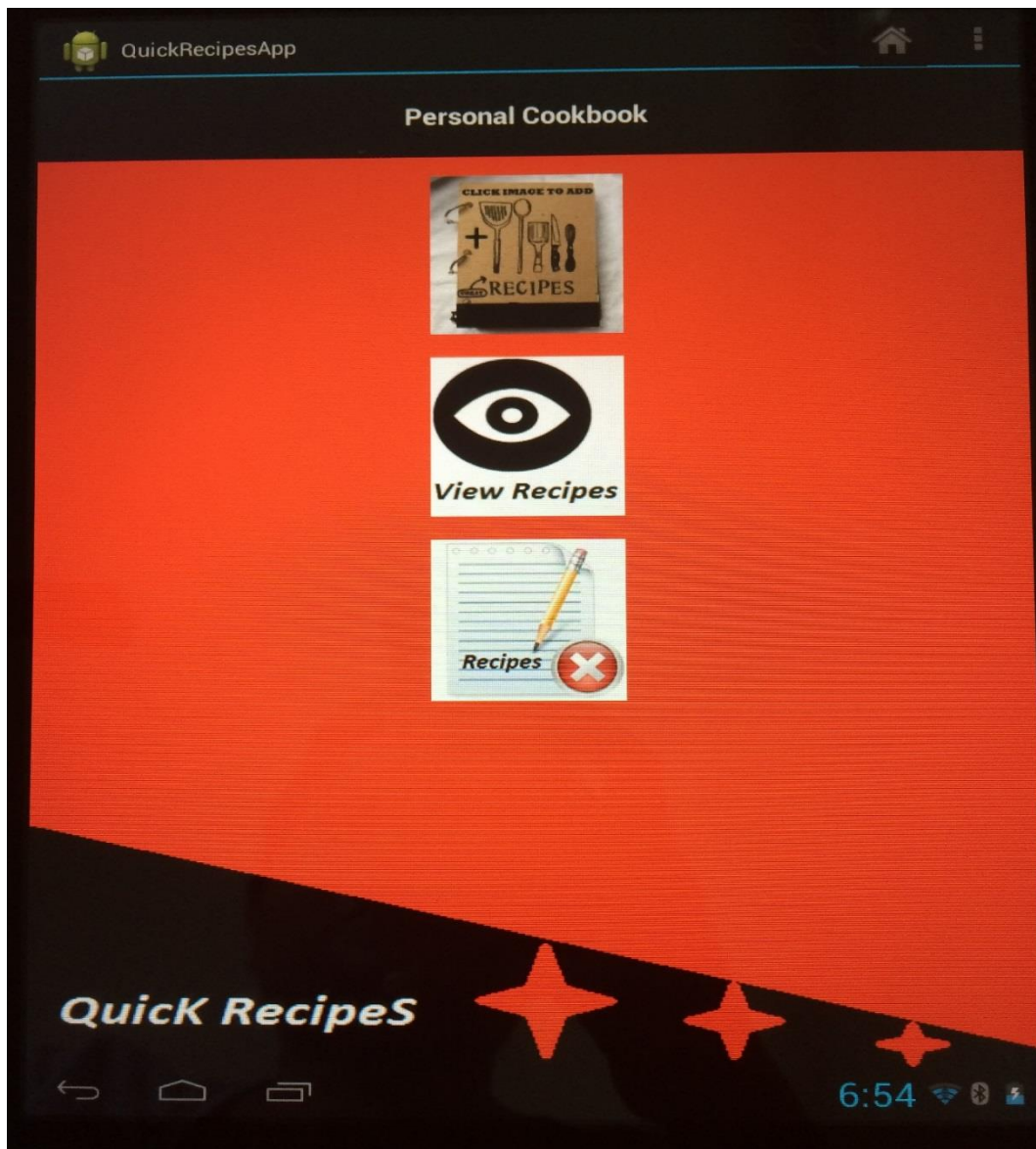
User can share recipe link on Gmail by clicking on Gmail icon from dialog box. User enters receiver email id, subject, and clicks on send button to share recipe by email as shown in Figure 39.



**Figure 39- Sharing Recipe on Gmail**

#### 4.7 Cookbook Implementation

User has option to create and manage personalized cookbook by creating new recipes. QuickRecipes application provides an image button on homepage for cookbook. On cookbook image button click, user is navigated to CookbookActivity class which provides option to create view and delete recipes. Figure 40 shows cookbook options for user.



**Figure 40- Cookbook Options**

Code Snippet for cookbook image button click on application home page:

```
case R.id.cookbookButton:

    Intent intent7 = new Intent(context, CookbookActivity.class);

    startActivity(intent7);
```

Code snippet for CookbookActivity class:

```
public void onClick(View v) {

    final Context context = this;

    int id=v.getId();

    switch(id) {

        case R.id.addrecipeButton:

            Intent intent1 = new Intent(context, CBCreateRecipeActivity.class);

            startActivity(intent1);

            break;

        case R.id.viewrecipesButton:

            Intent intent2 = new Intent(context, CBViewAllRecipeActivity.class);

            startActivity(intent2);

            break;

        case R.id.deleterecipeButton:

            Intent intent3 = new Intent(context, CBDeleteListViewTitleImageActivity.class);

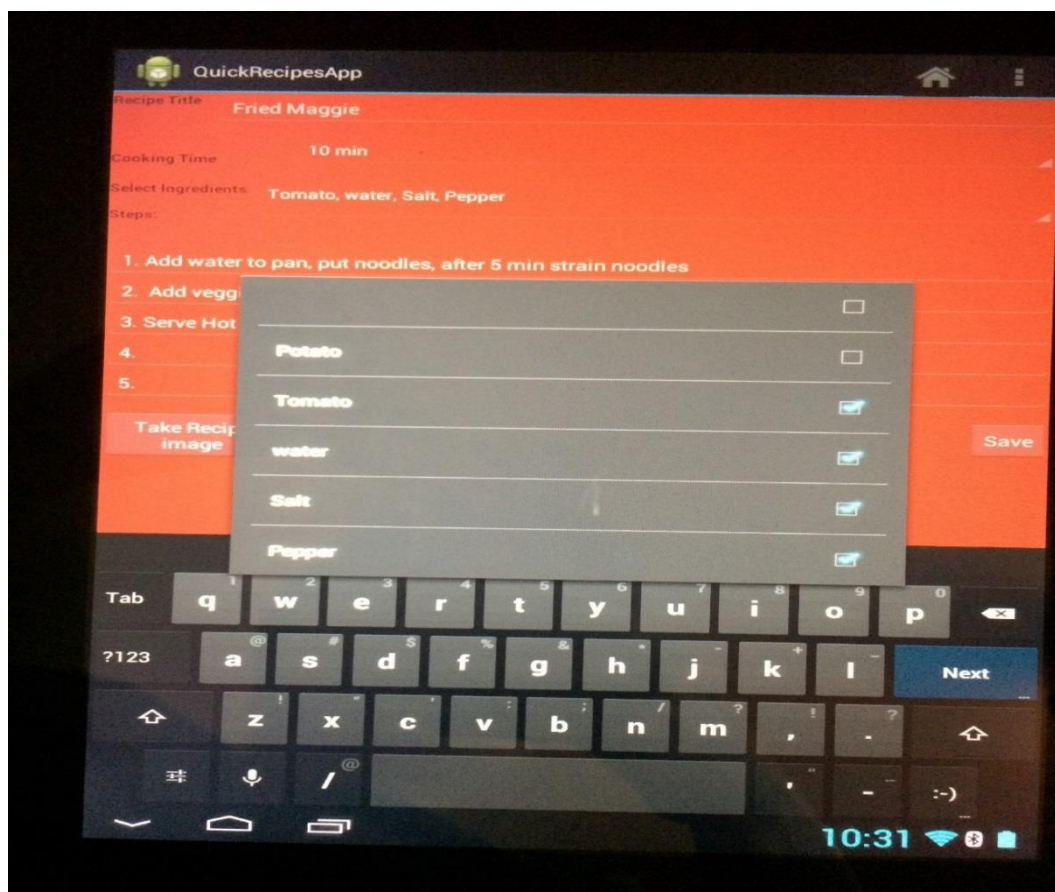
            startActivity(intent3);

            break;

    }}
}
```

### 4.7.1 Create Recipe

QuickRecipes application allows user to create recipe by providing recipe title, cooking time, ingredient, image and cooking steps. On create layout of activity\_cbcreaterecipe.xml file is displayed. User can enter recipe title and cooking steps in template provided. Android Spinner class is used for implementing cooking time. MultiSelection Spinner class is implemented for selecting ingredients. Upon click on save button information gets persisted in SQLite database. MySQLiteHelper class add method is executed for insertion. Figure 41 illustrates how user can create new recipe.



**Figure 41- Create Recipe Screen**

Code Snippet for adding recipe in database is as follows:

```
public void addRecipe(Recipe recipe){
    Log.d("addRecipe", recipe.toString());
    // 1. get reference to writable DB

    SQLiteDatabase db = this.getWritableDatabase();

    // 2. create ContentValues to add key "column"/value

    ContentValues values = new ContentValues();
    values.put(KEY_TITLE, recipe.getName()); // get title
    values.put(KEY_TIME, recipe.getTime()); // get author
    values.put(KEY_INGREDIENTS, recipe.getIngredients());
    values.put(KEY_IMAGE, recipe.getImage()); // Get Recipe Image
    values.put(KEY_STEP1, recipe.get_step1()); // Get COOKING DIRECTION
    values.put(KEY_STEP2, recipe.get_step2());
    values.put(KEY_STEP3, recipe.get_step3());
    values.put(KEY_STEP4, recipe.get_step4());
    values.put(KEY_STEP5, recipe.get_step5());

    // 3. Insert

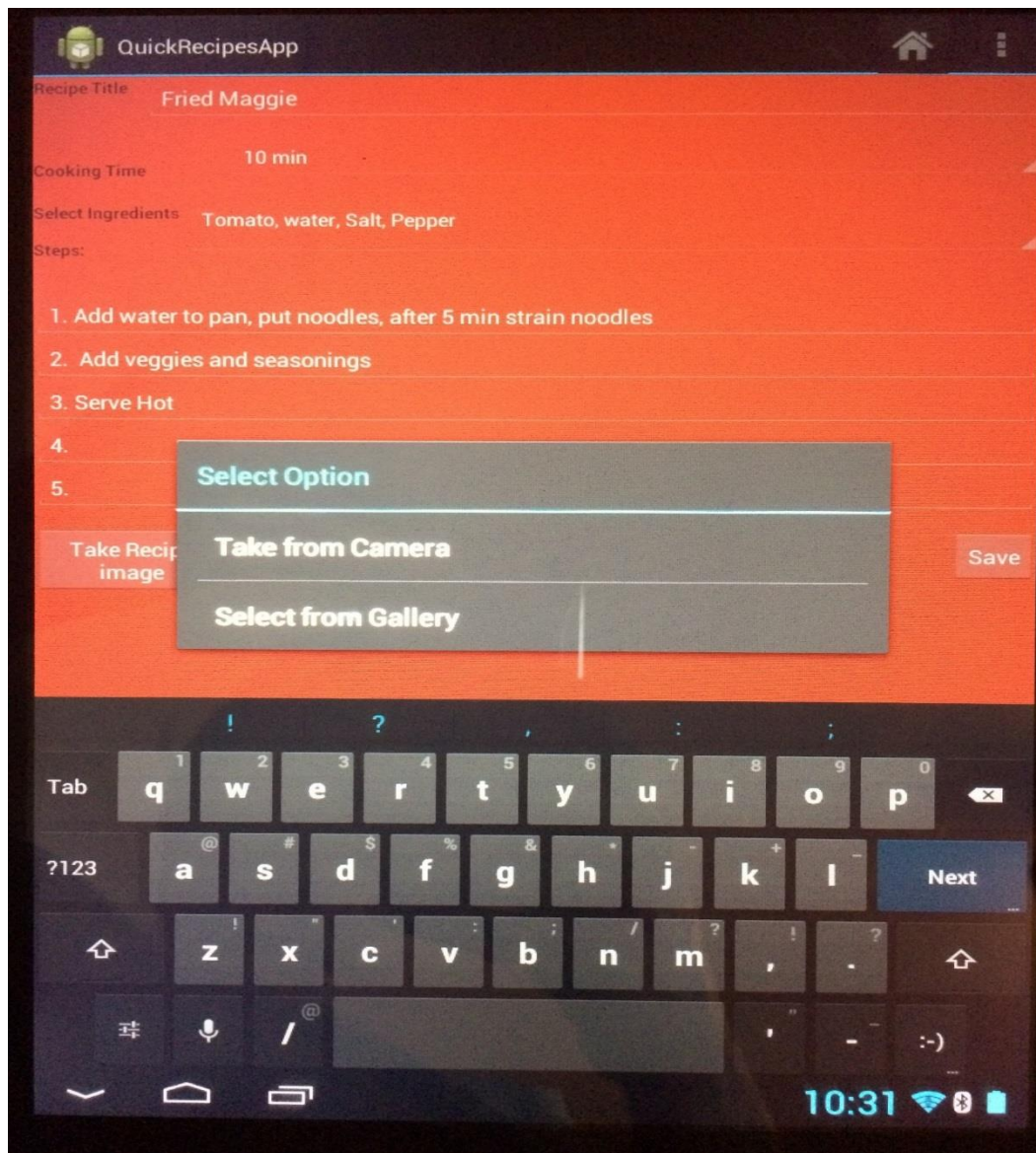
    db.insert(TABLE_RECIPES, // table
        null, //nullColumnHack
        values); // key/value -> keys = column names/ values = column values

    // 4. Close

    db.close();
}
```

In above code snippet, content values class object is used to save recipe name, time, directions, ingredients list to appropriate column in database. If data is not entered by user for a particular field, by default null values gets inserted in database. User has option to enter extra ingredients if not available in spinner list. These extra added ingredients gets persisted in database and are available to user in spinner list.





**Figure 42- Take Recipe Image Option**

User has option to add image to recipe by clicking on Take image button. Upon click `OnClick()` is executed and alert dialog is displayed to select camera or gallery option as shown in Figure 42. On selection `callCamera()` or `callGallery()` method is executed giving to option to user to take image from camera select image form gallery and crop image. Code Snippet is follows:

```

public void onClick(DialogInterface dialog, int which) {
    // TODO Auto-generated method stub
    Log.e("Selected Item", String.valueOf(which));
    if (which == 0) {
        callCamera();
    }
    if (which == 1) {
        callGallery();
    }

}
});

```

Upon selection image is compressed and stored in byte array. Database is updated with recipe image using `updateRecipe()` method. Code Snippet for `OnActivityResult()` is as follows:

```

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode != RESULT_OK)
        return;

    switch (requestCode) {
        case CAMERA_REQUEST:

            Bundle extras = data.getExtras();

            if (extras != null) {
                Bitmap yourImage = extras.getParcelable("data");
                // convert bitmap to byte
                ByteArrayOutputStream stream = new ByteArrayOutputStream();
                yourImage.compress(Bitmap.CompressFormat.PNG, 100, stream);
                byte imageInByte[] = stream.toByteArray();
                Log.e("output before conversion", imageInByte.toString());
                // Inserting Contacts
                Log.d("Insert: ", "Inserting ..");
                Log.d("temp1: ", temp1);
                Log.d("temp2: ", temp2);
                int j = db.updateRecipe(new Recipe(temp1, temp2,
                    ingre_temp, imageInByte, step_temp1, step_temp2, step_temp3, step_temp4, s
                    tep_temp5));
                Intent i = new Intent(CBCreateRecipeActivity.this,
                    CBCreateRecipeActivity.class);
            }
    }
}

```

```

        startActivity(i);
        finish();
    }
    break;
    case PICK_FROM_GALLERY:
        Bundle extras2 = data.getExtras();

        if (extras2 != null) {
            Bitmap yourImage = extras2.getParcelable("data");
            // convert bitmap to byte
            ByteArrayOutputStream stream = new ByteArrayOutputStream();
            yourImage.compress(Bitmap.CompressFormat.PNG, 100, stream);
            byte imageInByte[] = stream.toByteArray();
            Log.e("output before conversion", imageInByte.toString());
            // Inserting Contacts
            Log.d("Insert: ", "Inserting ..");
            int j = db.updateRecipe(new Recipe(temp1, temp2,
            ingre_temp,imageInByte,step_temp1,step_temp2,step_temp3,step_temp4,
            step_temp5));
            Intent i = new Intent(CBCreateRecipeActivity.this,
            CBCreateRecipeActivity.class);
            startActivity(i);
            finish();
        }

    break;
}

/**
 * open camera method
 */
public void callCamera() {
    Intent takePictureIntent = new
    Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, CAMERA_REQUEST);
    }
}

/**
 * open gallery method
 */

```



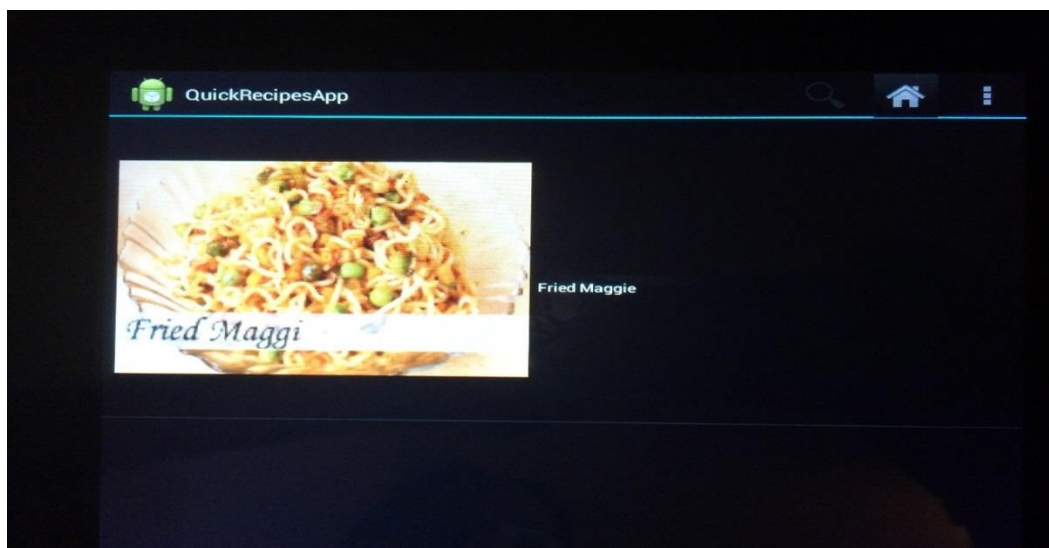
```

public void callGallery() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.putExtra("crop", "true");
    intent.putExtra("aspectX", 0);
    intent.putExtra("aspectY", 0);
    intent.putExtra("outputX", 200);
    intent.putExtra("outputY", 150);
    intent.putExtra("return-data", true);
    startActivityForResult(
        Intent.createChooser(intent, "Complete action using"),
        PICK_FROM_GALLERY);
}

```

#### 4.7.2 View Recipe

User can view recipe by clicking on view recipe button on personal cookbook page. Upon click `getAllRecipes()` is executed and list of recipes created by user are displayed with image and title. Figure 43 lists added recipe by user.



**Figure 43- View Recipe List**

Code Snippet is as follows:

```
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    db = new MySQLiteHelper(this);
    /**
     * Reading and getting all records from database
     */
    List<Recipe> recipes = db.getAllRecipes();
    for (Recipe cn : recipes) {
        String log = "ID:" + cn.getID() + " Name: " + cn.getName()
            + ",Time: " + cn.getTime() + ",Image: " + cn.getImage();

        // Writing Recipes to log
        Log.d("Result: ", log);
        // add Recipes data in arrayList
        imageArray.add(cn);
    }

    adapter = new RecipeImageAdapter(this, R.layout.screen_list, imageArray);
    final ListView dataList = (ListView) findViewById(R.id.list);
    dataList.setAdapter(adapter);
    dataList.setChoiceMode(GridView.CHOICE_MODE_SINGLE);
}
```

User can click any recipe title and view recipe details. Setting single selection from list is implemented using setChoiceMode() on listview. Upon click setOnItemClickListener() is executed and recipe title is passed in intent by using putString() method on bundle object. CBViewRecipeActivity class displays recipe details by executing getRecipeByTitle(String Title) method.

```
dataList.setOnItemClickListener(new OnItemClickListener() {

    public void onItemClick(AdapterView parent, View v, int position,
        long id) {

        Toast.makeText(
            getApplicationContext(),
```

```

((TextView) v.findViewById(R.id.txtTitle)).getText(),
Toast.LENGTH_SHORT).show();
final String title = (String) ((TextView) v.findViewById(R.id.txtTitle)).getText();

System.out.println("Item clicked" + ((TextView)
v.findViewById(R.id.txtTitle)).getText() + " " + title);

Intent intent = new Intent(CBViewAllRecipeActivity.this,
CBViewRecipeActivity.class);
Bundle bundle = new Bundle();
bundle.putString("recipetitle", title);

intent.putExtras(bundle);
startActivity(intent);

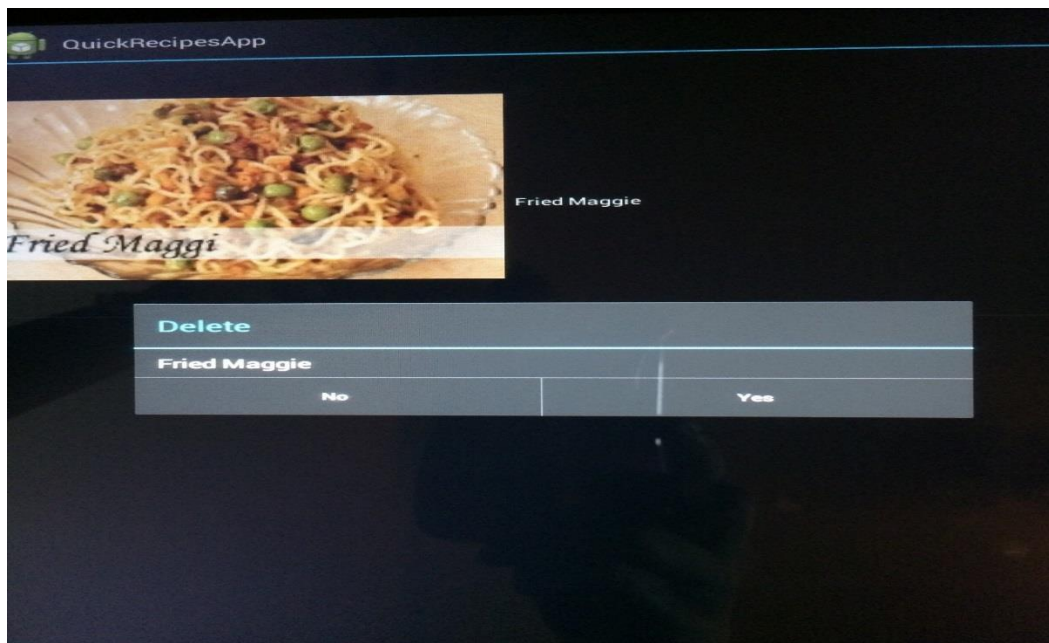
}); } }

```

### 4.7.3 Delete Recipe

User can delete recipe by clicking on delete recipe button on personal cookbook page.

Figure 44 shows deleting recipe screen:



**Figure 44- Delete Recipe Screen**

Code Snippet is as follows

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    db = new MySQLiteHelper(this);

    // Reading all Recipes from database
    List<Recipe> recipes = db.getAllRecipes();
    for (Recipe cn : recipes) {
        String log = "ID:" + cn.getID() + " Name: " + cn.getName()
        + " ,Image: " + cn.getImage();

        // Writing Recipes to log
        Log.d("Result: ", log);
        // add Recipes data in arrayList
        imageArray.add(cn);
    }
    adapter = new RecipeImageAdapter(this, R.layout.screen_list, imageArray);
    final ListView dataList = (ListView) findViewById(R.id.list);
    dataList.setAdapter(adapter);
    dataList.setChoiceMode(GridView.CHOICE_MODE_SINGLE);

    dataList.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView parent, View v, int position,
        long id) {

            Toast.makeText(
                getApplicationContext(),
                ((TextView) v.findViewById(R.id.txtTitle)).getText(),
                Toast.LENGTH_SHORT).show();
            final String title = (String) ((TextView) v.findViewById(R.id.txtTitle)).getText();

            System.out.println("Item clicked" + ((TextView)
            v.findViewById(R.id.txtTitle)).getText() + "      " + title);
            AlertDialog.Builder alertDialogBuilder = new
            AlertDialog.Builder(CBDeleteListViewTitleImageActivity.this);

            // set title
            alertDialogBuilder.setTitle("Delete");

```

```

        // set dialog message
        alertDialogBuilder
        .setMessage(((TextView) v.findViewById(R.id.txtTitle)).getText())
        .setCancelable(false)
        .setPositiveButton("Yes",new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int id) {
                // delete selected recipe
                db.deleteRecipe(title);
                CBDeleteListViewTitleImageActivity.this.finish();
            }
        })
        .setNegativeButton("No",new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int id) {
                // if this button is clicked, just close
                // the dialog box and do nothing
                dialog.cancel();
            }
        });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();

    }
});
}

```

OnCreate() method displays list of recipes. OnClick() displays alert dialog for delete operation confirmation. On yes button click, recipe gets deleted from database and page is refreshed.

#### **4.8 Future Work**

This application provides easy interface for user to search recipe from cloud, create your own recipe and share recipe. Food2Fork API is used for implementation and sources of recipes are limited. This application can be extended in many different ways. The application can be extended to provide recipe from more other sources by integration other API like BigOven. Application has add to favorite and sharing capability for cloud recipes. The application can also be extended to include sharing capability for user created recipes in cookbook.

## **Chapter 5**

### **CONCLUSION**

This application is extremely handy and useful for cooking variety of recipe with minimum search effort from internet. Internet recipe may appeal to one but not to other, some people like experimenting and trying own recipe. This app also provides ability to user to create recipe and save it for later reference. It will help people to save their time and energy in finding recipes for daily routine as well as for special occasions. And since this is a mobile application, users have the luxury to check for recipes wherever they are and save them for later. The application can be used by a broad range of users which may include parents trying to cook new recipe for kids, people who are fond of desserts, restaurants owner trying to add new item to their menu and for regular cooking. It will help to make the lives of people simpler.

## REFERENCES

- [1] Android Developer guide, from  
<http://developer.android.com/> , accessed in March 2015
- [2] Android Tutorial, from  
<http://www.tutorialspoint.com/android/>, accessed in March 2015
- [3] Android Activity Lifecycle Diagram, from  
<http://www.javatpoint.com/images/androidimages/Android-Activity-Lifecycle.png> ,  
accessed in April 2015
- [4] Android Service Lifecycle Diagram, from  
[http://www.tutorialspoint.com/android/images/android\\_service\\_lifecycle.jpg](http://www.tutorialspoint.com/android/images/android_service_lifecycle.jpg) ,  
accessed in April 2015
- [5] Android Tutorial, from  
<http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html> , accessed  
in March 2015
- [6] Marko Gargenta. *Learning Android*, O'Reilly Media, Inc, March 2011.  
[http://aiti.mit.edu/media/programs/indonesia-summer-2013/materials/gargenta -  
2011 - learning android.pdf](http://aiti.mit.edu/media/programs/indonesia-summer-2013/materials/gargenta_-_2011_-_learning_android.pdf) , accessed in March 2015
- [7] Android Developers Blog, from  
<http://android-developers.blogspot.com/> , accessed in March 2015
- [8] Food2Fork , from  
<http://food2fork.com/> , accessed in March 2015