



ASSAM HOUSE

East Asian Food Recipes App

Submitted by: Yeong De Jong (T00185309)

Computing in Software Development - Stage 2
Native Mobile Apps Design & Development

Date Submitted: 03/05/2017

Table of Contents

| | | |
|------|---|----|
| 1. | Application Description | 3 |
| 1.1. | Implementation Schedule | 4 |
| 2. | Application Components Description..... | 5 |
| 2.1. | Login Activity | 5 |
| 2.2. | Signup Activity | 7 |
| 2.3. | Reset Password Activity..... | 9 |
| 2.4. | Home Activity (Cuisine Category)..... | 11 |
| 2.5. | Cuisine List Activity | 13 |
| 2.6. | Cuisine Recipe Activity..... | 15 |
| 3. | Testing Strategy | 19 |
| 4. | Future Updates..... | 20 |
| 5. | References | 21 |

1. Application Description

IDE Version: 2.3.1

Android SDK Build Tools Version: 25.0.2

“Assam House” is a simple East Asian food recipe application that aim to introduce variety of cuisines that represents the characteristic of that country. Malaysia, Thailand, Vietnam, Indonesia, and South Korea (Korea) were displayed in this application for simplicity. Appropriate cuisine’s descriptions, cuisine’s ingredients, and cuisine’s direction is displayed back to user according to user’s selection.

“Assam House” application provides step by step guide for preparing a healthy home-made meal and the information of the cuisines that goes with user’s diet. Information about what the cuisine is about, preparation time, cooking time, and total number of servings is displayed to user. Users can follow the step by step guide provided while preparing ingredients and while cooking.

Lastly, when the application launched, user is prompted to enter his/her email and a password as a representation of identity for logging in to the home activity whereas for new users, users must register their account before logging in to the home activity. User can also change his/her password if he/she has forgotten the password during account registration.

1.1. Implementation Schedule

| | April | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | May | | | |
|--------------------------------|-------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| Login Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Signup Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset Password Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Home Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cuisine Category | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Logout Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cuisine List Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Search Cuisine Images | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Search Cuisine Recipes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Insert Cuisine into Database | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Retrieve Cuisine from Database | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Logout Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cuisine Recipe Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Logout Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tab Function and Layout | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Documentation & Changes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 1: Implementation Schedule


 2-3 hours per day.

Table 1 above shows the application implementation schedule where all requirements are divided into sprints. All activities are implemented in both landscape view and portrait view. Components of each activity is completed and will be explained in the following.

2. Application Components Description

2.1. Login Activity

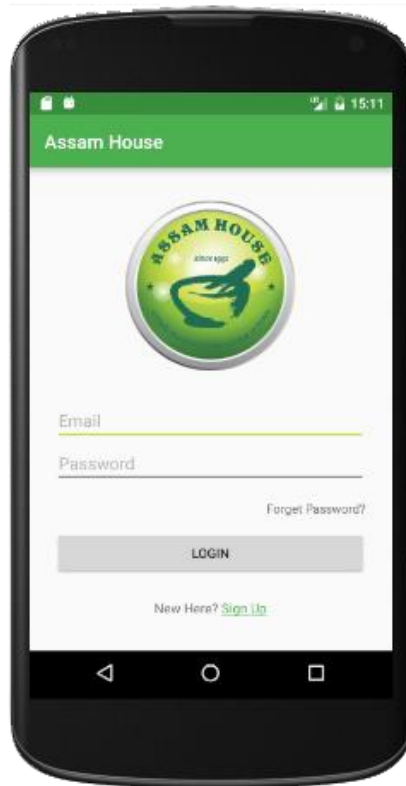


Image 1: Login Activity Portrait View

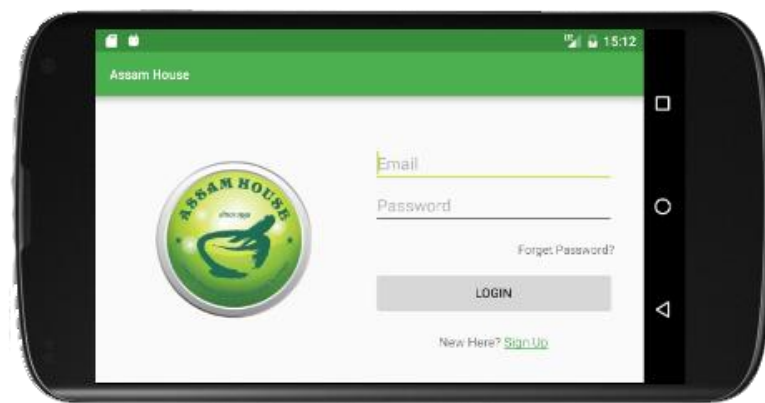


Image 2: Login Activity Landscape View

Images above shows the Login Activity in both portrait and landscape view. Login Activity is displayed to user when the application is launched for the first time after installing the application. For new user, user is requiring to create an account for logging in by clicking the “Sign Up” text view in the activity. According to writing style in Material Design, text should be understandable by anyone, anywhere, regardless of their culture or language (Design, 2017). The writing style for “Sign Up” complies the android design principles that the “Sign Up” text view is referring users to the labels on UI elements instead of the type of the element. User can also change the password of the account by clicking the “Forget Password?” text view in the activity.

“Shared Preferences” is used to store the details of the user’s account. Shared Preferences allow programmer to store and retrieve data in the form of key-value pair. When the login button is clicked, the application checks the email and password entered with the value stored in SharedPreferences.Editor class. If the email or password entered does not match the value stored in SharedPreferences.Editor class, a simple feedback is displayed to user with an error message which states that “Invalid Email/Password” using Toast class.

Login Activity is displayed when the application launched for the first time and at the same time, this activity calls to the method in DB_Recipes class to insert data into the database. SQLite Database is the database used to store cuisine recipes data. DB_Recipes is a subclass that extends SQLiteOpenHelper that helps to manage database creation and version controlling. The reason of using SQLiteOpenHelper class is that SQLiteOpenHelper class takes care of opening the database if database exists, creating a database if database does not exist, and upgrading database if necessary (developer.android.com, 2017).

When DB_Recipes is invoked, it executes an SQL statement to create a table to store cuisine recipes data in the onCreate() method. User-defined method removeAllData() is called to remove all data before inserting data into the database. The reason of calling removeAllData() method is to prevent duplicates data while inserting data into database when the application is relaunched. After calling removeAllMethod(), it calls to user-defined insert method in DB_Recipes class to insert recipes into the database so that data can be accessed in other activities in the application.

After doing some research, the author found that almost all application supports one time login and the account is logged out only when the user selects the logout option in the application. The method used to implement one time login in this application is to store a key-value pair with type Boolean to check if user has logged in or not using SharedPreferences class. A value of “true” will be stored if user has logged in successfully and finish() method will be called to indicate that Login Activity is done and should be closed. The system will check the key-value pair when the application is relaunched. If the value is “true”, Home Activity will be displayed to user, else Login Activity will be displayed to user.

The reason to implement Login Activity as shown in Image 1 and 2 is to keep the UI as brief as possible and is easy to understand and used by user.

2.2. Signup Activity

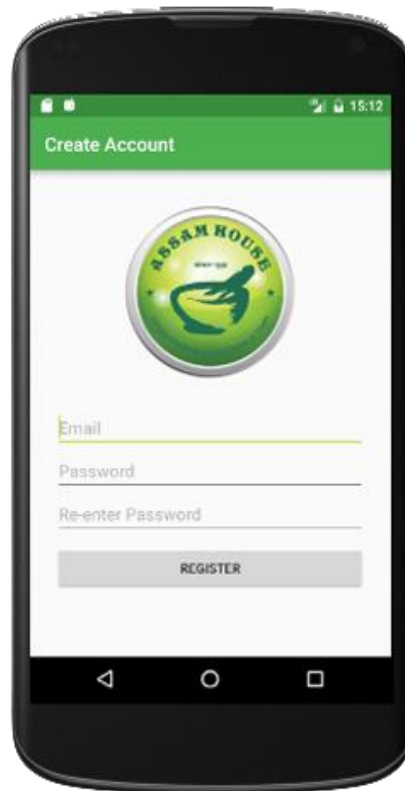


Image 3: Signup Activity Portrait View

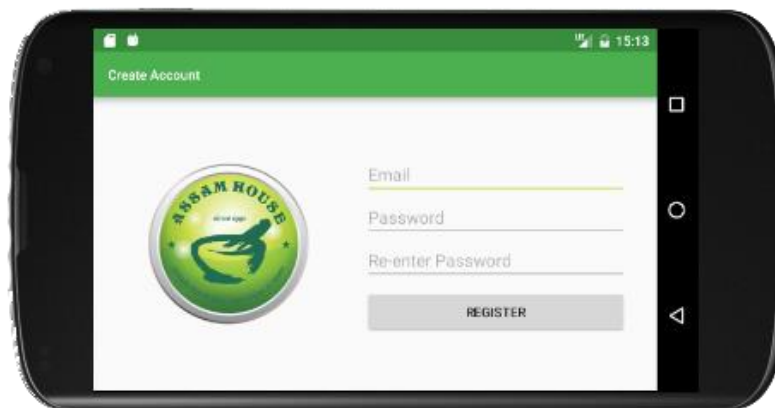


Image 4: Signup Activity Landscape View

Images above shows the Signup Activity in both portrait and landscape view. In this activity, user is required to provide an email address and password for registration. When user clicks on the register button after providing the appropriate information, the system validates the details provided. The system first validates the email address provided to check if the email address matches the pattern in Patterns class. If the email address provided is not valid, an appropriate error message will be displayed to indicate that email provided is invalid and focus the cursor to email TextView.

If the email address provided is a valid email address, the system then validates the password provided by checking if the text entered in the Password TextView is equals to the Re-enter Password TextView and is either one of the TextView is empty. If password provided is invalid, the system clears the content of both Password and Re-enter Password TextView and display an error message to indicate that password does not match. User is then required to re-enter the password again.

Once the details provided has been successfully validated, the system stores the email address and password provided into the system using SharedPreferences class. The data stored as a key-value pair in the system. The system will then display a message to indicate that account registered successfully and brings the user back to Login Activity using Intent class and Signup Activity is closed by calling finish().

The reason to implement Signup Activity as shown in Image 3 and 4 is to keep the UI as brief as possible and is easy to understand and use by user.

2.3. Reset Password Activity



Image 5: Reset Password Activity Portrait View



Image 6: Reset Password Activity Landscape View

Images above shows Reset Password Activity in both portrait and landscape view. User can reset the password provided when registering the account. When user clicks on reset password button, system retrieves the "Email" value stored in the preferences and checks if the email entered matches the value stored in the preferences. If the email entered matches the value stored in the preferences, the system checks the if both password matches and is not empty. Once the details successfully validated, the system overwrites the "Password" value stored in the preferences with a new value and display a message to indicate that the password changed successfully. The system then brings the user back to Login Activity and close this activity by calling finish() method.

The reason to implement Reset Password Activity as shown in Image 5 and 6 is to keep the UI as brief as possible and is easy to understand and use by user. Same layout is used in Login Activity, Signup Activity, and Reset Password Activity because this activity is related to each other where the function of Signup Activity is to create an account, Reset Password Activity is to reset or change to password of the account and the function of Login Activity is to login using the account created. According to Android Design Principles, if it looks the same, it should act the same (Android, 2017).

2.4. Home Activity (Cuisine Category)



Image 7: Home Activity (Cuisine Category) Portrait View

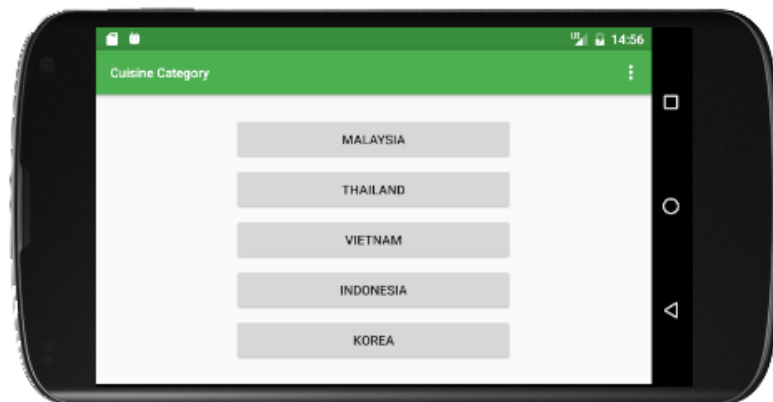


Image 8: Home Activity (Cuisine Category) Landscape View

Images above shows Home Activity (Cuisine Category) in both portrait and landscape view. For simplicity, this activity contains five buttons that represents five different cuisine category which are Malaysia, Thailand, Vietnam, Indonesia, and Korea. This activity also contains an option menu on the top-right of the application. The option menu contains “Logout” menu items that allows user to logout from the account.

The reason of using options menu to store “Logout” is that according to Android Design Principles, only shows what user need and when user need it. People get overwhelmed when seeing too much at once in an application (Android, 2017). Therefore, using options menu to hide “Logout” function that are not essential at the moment complies the Android Design Principles.

When user clicks on either one of the five buttons in Home Activity, the name of the button is stored using putExtra() method in Intent class as a key-value pair so that the name of button can be used as a keyword to search through the database in Cuisine List Activity. The system then launched Cuisine List Activity and display to user.

2.5. Cuisine List Activity

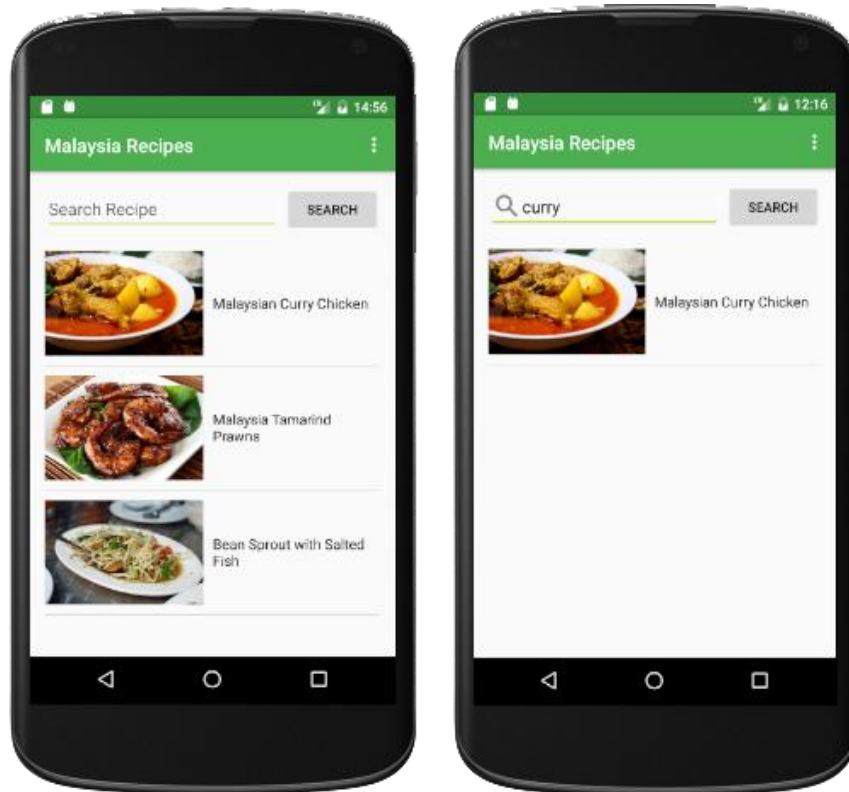


Image 9: Cuisine List Activity Portrait View

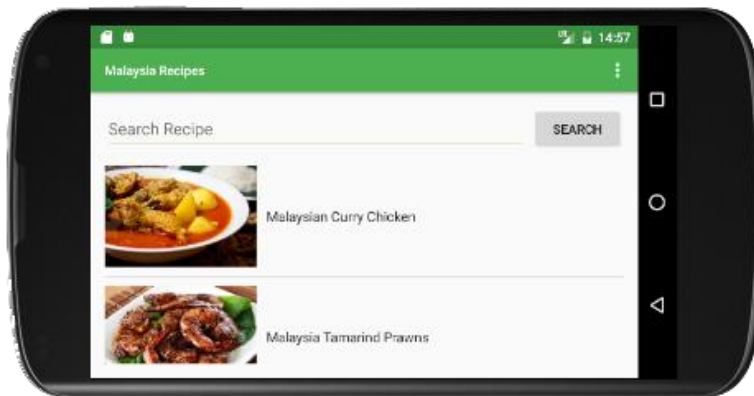


Image 10: Cuisine List Activity Landscape View

Images above shows Cuisine List Activity in both portrait and landscape view. In this activity, the system retrieves all available cuisines in the selected category from Home Activity and displays a list of cuisines of the selected category on UI using ListView. A user-defined class CustomListAdapter that extends ArrayAdapter superclass is created to implement a custom view on how the cuisines is to be listed which contains ImageView for displaying the cuisine image and TextView for display the name of the cuisine.

In DB_Recipes class, method `getSelectedCategory()` is defined takes the category as an argument to retrieve all cuisine recipes from the database and return an ArrayList back to the calling activity. Cursor interface is used which provides random read and write access to the result set returned by the database query. When cursor reads through every row of the result set, it creates a Recipes object and add each Recipes object to the ArrayList. The method returns a value of null to indicate that there is no element in the ArrayList back to Cuisine List Activity.

In Cuisine List Activity, the system checks the return value returned by the method `getSelectedCategory()` in DB_Recipes class. If the value returned is not null, the system populates the ListView and creates an on click listener to the list view, else the system will display an error message to indicate that there are no cuisines on the selected category and Home Activity is displayed back to user. User can also search a cuisine name with keyword provided as shown in Image 9. When user clicks on the search button, it invokes the method `getSearchRecipes()` method in DB_Recipes that takes the category and the keyword as an argument to retrieve cuisines with matching keyword from the database and will return an instance of ArrayList back to the calling activity.

From the activity, it will check if the array list returned is an empty list or not. If the array list returned is an empty list, the system will reset the list view and display an error message to indicate that no such keyword found from database. User can also reset the list view by deleting the text entered in the search recipe field. When user selects a cuisine to be viewed from the ListView, it creates an instance of the Intent class and add the details of the selected recipe as a key-value pair and pass the data to Cuisine Recipe Activity.

This activity contains an option menu on the top-right of the application which contains “Logout” menu items that allows user to logout from the account. The reason of using options menu to store “Logout” is that according to Android Design Principles, only shows what user need and when user need it. People get overwhelmed when seeing too much at once in an application (Android, 2017).

The idea of using both cuisine image and the name of the cuisine is that a picture worth a thousand words. This complies to Android Design Principles where it states that pictures are faster that words. Consider using images to explain ideas, it gets people’s attention and can be much more efficient than words (Android, 2017).

According to Android Design Principles, learn user’s preferences over time rather than asking user to make the same choices repeatedly (Android, 2017). The reason of implementing as shown in Image 9 and 10 is that it allows user to search a recipe by using keyword provided instead of scrolling the list view up and down when the list view contains a huge amount of cuisines which complies to Android Design Principles.

2.6. Cuisine Recipe Activity



Image 11: Cuisine Recipe Activity (Description Tab) Portrait View

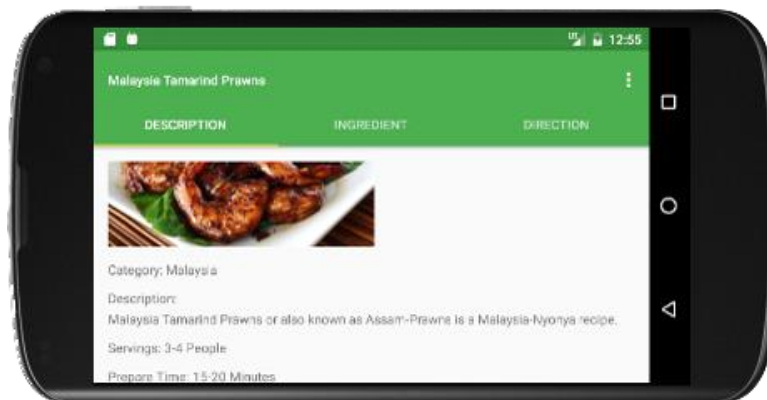


Image 12: Cuisine Recipe Activity (Description Tab) Landscape View

Images above shows the “Description” Tab in Cuisine Recipe Activity in both portrait and landscape view. “Description” tab provides information about what the cuisine is about, total servings, preparation time, and cooking time.

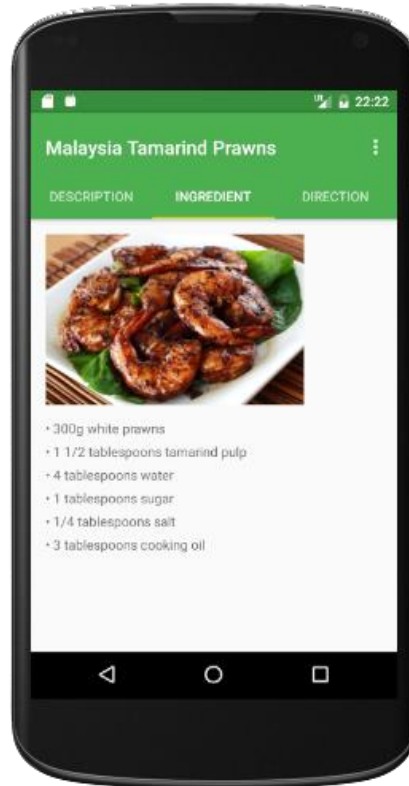


Image 13: Cuisine Recipe Activity (Ingredient Tab) Portrait View

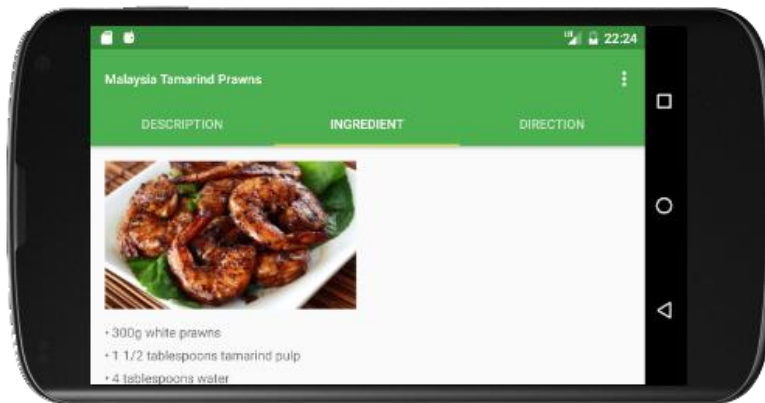


Image 14: Cuisine Recipe Activity (Ingredient Tab) Landscape View

Images above shows the “Ingredient” tab in Cuisine Recipe Activity in both portrait and landscape view. “Ingredient” tab provides a list of ingredients needed for the selected cuisine.



Image 15: Cuisine Recipe Activity (Direction Tab) Portrait View

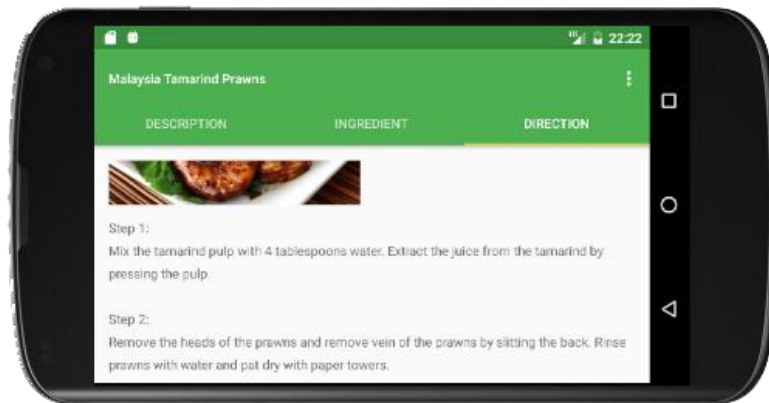


Image 16: Cuisine Recipe Activity (Direction Tab) Landscape View

Images above shows the “Direction” tab in Cuisine Recipe Activity in both portrait and landscape view. “Direction” tab provides step by step guide for preparing the selected cuisine.

The reason of using Tab Layout while implementing Cuisine Recipe Activity instead of using Scroll View in a single activity is that a single Recipe object might contains more than 10 ingredients and 10 steps for preparing the cuisine and using Tab Layout instead of scroll view is to divide three different component (description, direction, ingredient) into three different tabs so that user can have a clearer view on each tab rather than scrolling up and down the screen to view different information. Image of the selected cuisine is included in each tab so that user can view the image of the cuisine in the selected tab rather than going back and forward.

Layout used for each tab is the same as shown in images above, however, three different class (DescriptionTab, DirectionTab, IngredientTab) which extends Fragment class is defined to create the layout view for each tab. The class contains the overridden onCreateView() method which returns a View object back to the calling activity. The reason of using Fragment class is that Fragment is a piece of activity which enable more modular activity design and the layout of the Fragment can be easily changed if needed. Three different layout corresponding to the tab is created and is called in each class to set the view of the activity.

A method called getCuisineElement() is defined in Cuisine Recipe Activity to retrieve the data sent from Cuisine List Activity. The method retrieves the data sent from Cuisine List Activity via Intent class and assign the value to the related variable and create a public static Recipe object using multi-argument constructor in Recipe class so that DescriptionTab class, DirectionTab class, and IngredientTab class can access the information of the Recipe object using the accessor method in Recipe class. The reason of using public static modifier is that the information of the selected cuisine does not change.

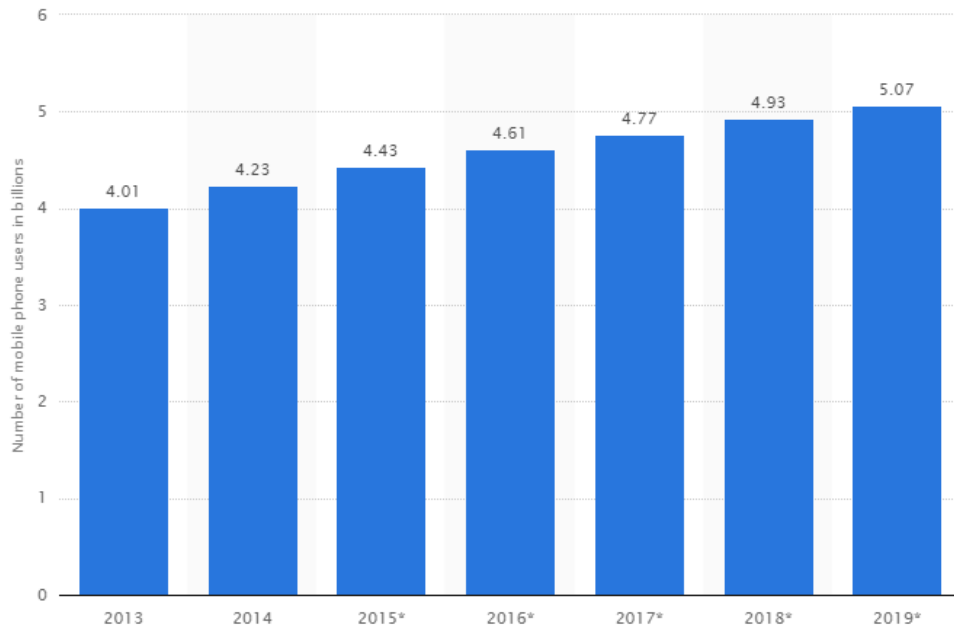
The layout used in description tab, direction tab, and ingredient tab looks the same. This is because the information displayed comes from the same cuisine selected by user. Therefore, using the same layout in description tab, direction tab, and ingredient tab to indicate that the functionality or purpose of these three tabs are the same. According to Android Design Principles, if it looks the same, it should act the same (Android, 2017).

Semi-Completed Component

The current implementation in Android Studio is slightly different from the previous documentation in CA1 that there is no “Share” function included in this activity. The reason is that in this activity, three tabs is implemented and if user selects “Share” button if is implemented, all information of that cuisine should be shared rather than just the selected tab. This is why the share function is not included in the current implementation.

3. Testing Strategy

Mobile devices have become the primary medium of interaction for users worldwide and mobile applications are driving these interactions (Kapoor, 2013). The graph below shows the number of mobile phone users worldwide from 2013 to 2019 in billions.



From the graph above, the number of mobile phone users is forecast to reach 4.77 billion as of year 2017 (Statista, 2017). A successful mobile application is that an application has to meet customer expectations and business goals.

One of the following testing strategies is to choose a device to be used for testing the application. Choosing the right device for testing is important because the device that is used for testing must represent the maximum number of target users for the application. OS Version, Screen Resolution and Form Factor have to be considered while selecting the device model. Mobile app must be tested on all major stable OS version and use different screen resolutions to test the mobile application (Kapoor, 2013).

Another testing strategy is beta testing. Beta testing is an effective method for mobile app testing where it provides access to real-world testers and real devices (Kapoor, 2013). Feedback is returned from users so that the developer can improve the performance or user interface on the mobile application.

Lastly, mobile app testing on Cloud is also an effective option to test mobile applications. Cloud computing provides a web-based testing environment where mobile applications can be deployed, tested, and managed. Using Cloud as a testing strategy helps to minimize the cost of the project and saves time (Kapoor, 2013).

4. Future Updates

The current implementation only allows user to create an account and reset the password for the account created. It also provides five cuisine categories which are Malaysia, Thailand, Vietnam, Indonesia, and Korea and the corresponding recipes in each category. For future updates, more cuisine categories can be added to the application. Adding more cuisine categories can also use as a monetization strategy where user is required to perform an in-app purchase to unlock more cuisine recipes or more cuisine categories.

Other than that, for future implementation, user can add his/her own recipe to the application. A separate category called “My Recipe” can be implement to store and display the recipe added in the application. User can also amend the details of the added recipe in the application and the system will update the details of that recipe in database. User can also remove a selected recipe in “My Recipe” category and the system will remove the selected recipe in database.

A separate category called “Favorite” can be implement to store favorited cuisines favorited by user. Another menu item or bar layout can be added in Cuisine Recipe Activity to allow user to favorite the cuisine and the favorited cuisine is stored in “Favorite” category where in this category, user can view the recipe of the cuisine without having to go through all the categories in the application.

Lastly, “Share” function can be implemented in the application so that user can share the recipe to social networks directly from the application. The current implementation does not have a “Share” option implemented where user have to screenshot the recipe and launched the social network application and share it from the social application. Implementing “Share” function in the application allows user to share the cuisines directly from the application without having to change to social network application.

5. References

Android, 2017. *Android Design Principles*. [Online]

Available at: <https://developer.android.com/design/get-started/principles.html>

[Accessed April 27 2017].

Design, M., 2017. *Writing*. [Online]

Available at: <https://material.io/guidelines/style/writing.html#writing-language>

[Accessed 25 April 2017].

developer.android.com, 2017. *SQLiteOpenHelper*. [Online]

Available at:

<https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html>

[Accessed 26 April 2017].

Kapoor, A., 2013. *5 Testing Strategies for building successful Mobile Apps*. [Online]

Available at: <https://www.netsolutionsindia.com/blog/5-testing-strategies-for-building-successful-mobile-apps/>

[Accessed 2 May 2017].

Statista, 2017. *Number of mobile phone users worldwide from 2013 to 2019 (in billions)*. [Online]

Available at: <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/>

[Accessed 2 May 2017].