IngrediEAT

CS 591 E1 Final Project Report

Winner Winner Chicken Dinner

Juntao Wang U18801732
Yanshan Song U93089867
Zixiang Wei U97992068
Jingxin Dai U69300525
Changhao Liang U16843909

Boston University

March 5, 2021

Contents

1 Introduction

1.1 Background

Considering that the virus makes us unable to eat outside, we can only cook at home. So Learning how to cook delicious food is very necessary for us.

1.2 Aim

Most applications in the market can only recommend recipes for users. We want to focus on recommending recipes by considering user's available ingredients. Also keep a favourite recipe list for our user.

2 Detailed App Walkthrough

Welcome Page Welcome page is the first page you will see when you run the app.

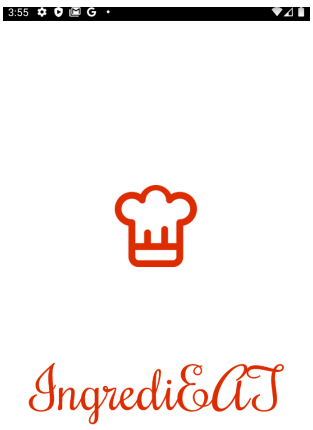

Figure 1: Welcome Page

Login Page If the user have not login, they need to login through their google account first.

Figure 2: Login Page

Pantry Page In our app the pantry page, user could see all the category of ingredients. There are two way for user to select ingredients and add them to user's cart.

The user could search and select ingredients from all category, by typing words or speech recognition in pantry page, shows in Figure 3.

Also user could select one category from the list. And view all ingredient, search ingredient from the category. If user want to add one ingredient, just click the chip and the add button, shows in Figure 4

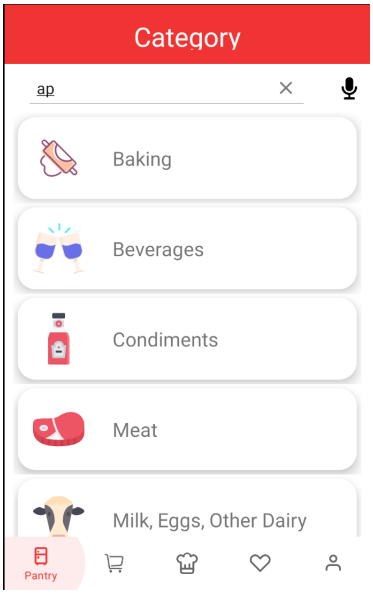

Figure 3: Pantry Page

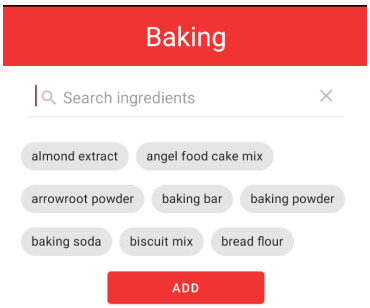

Figure 4: search ingredients from selected category

Cart Page All the ingredients selected from pantry, will be added to user cart. User could manage their ingredients here, unselect some of the ingredients and click update button to remove the ingredients from cart. Or click the clear button clear all the ingredients from cart.

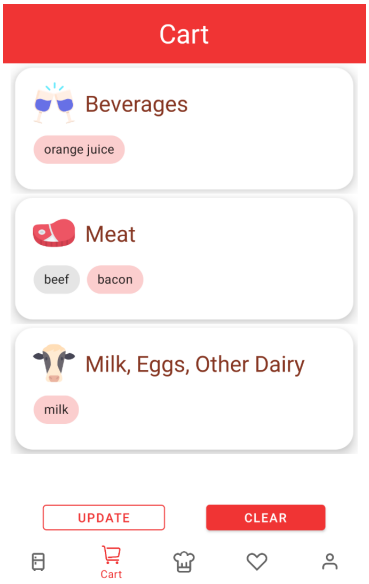

Figure 5: Cart Page

Recipe Page Based on ingredients in user cart, the app will recommend recipes for users. User could select one interested recipe and see detailed information, i.e., the rating of the recipe, the ingredients needed for the recipe, the cooking steps.

We classify the ingredients to user available and missing ingredients. The available ingredients will be shown in red chip, missing ingredients shows in grey chip, as shows in Figure 7.

Also user could give their own rating for the recipe here, as shown in Figure 8. If user like this recipe, he can set this recipe as favourite. This recipe will add to user favourite list and show in the favourite page.

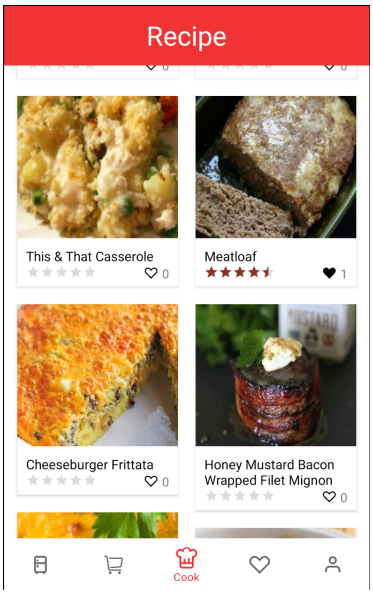

Figure 6: Recipe Page

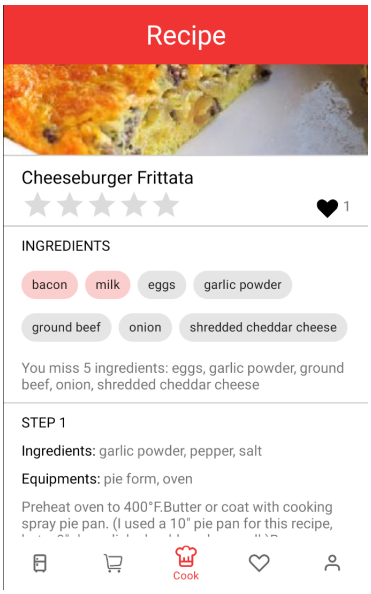

Figure 7: Recipe detail 1

Figure 8: Recipe detail 2

Favourite Page when user switch to favorite page. It will show all recipes that user set favourite.

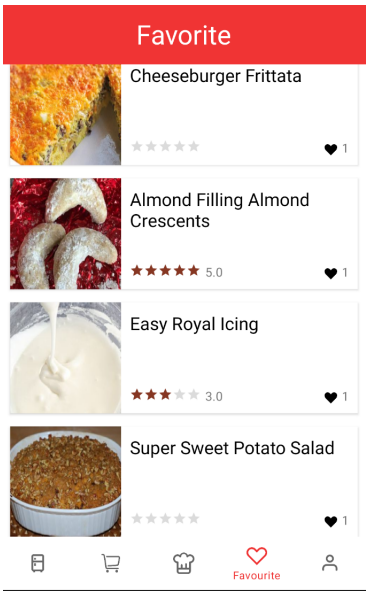

Figure 9: Favourite Page

User Page On the user page, we can see user's information. And user can sign out from this page.

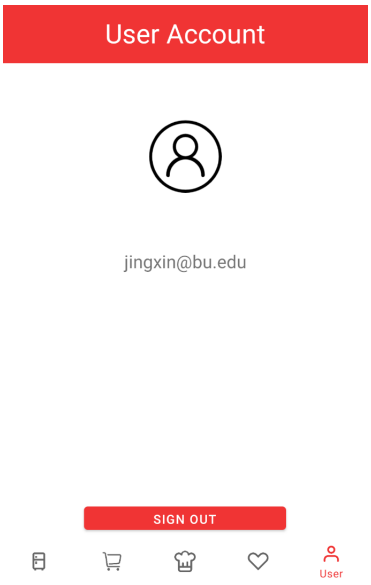

Figure 10: User Page

3 Implementation Details

3.1 App implementation

Material Design Material design is a designing language developed by google. Material means digital material that mimic physical materials. Like paper cards, they have physical surface and edges. They also have seams and shadows if piled together. That's exactly the style of material design.

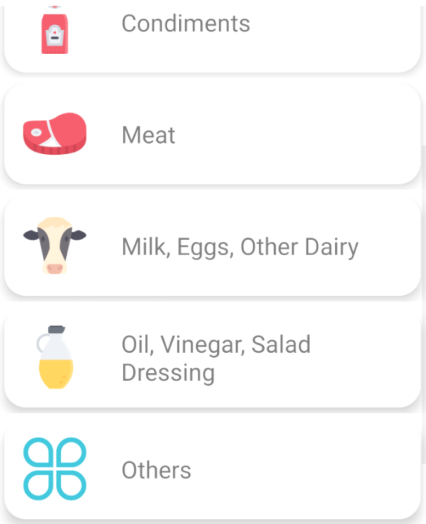

Figure 11: Card View

Google provides abundant material design widgets in their Material package. As a kind of flat design style, material design makes views look simple but beautiful.

Fragment Since too many activities would be hard to handle, IngrediEAT only instantiates 4 activities, including login and welcome activity, main activity and base activity. We use different fragments for other interfaces instead of activities. Another advantage of fragments is to keep the reusability. We do not hope

to duplicate the same work such as the bottom navigation bars in different interfaces. Also, fragments are coded with loose coupling, they would only communicate through the main activity.

Since android doesn't allow us to send message directly from the current thread to background threads, the hided background fragment would not update immediately when we click the back button. Some techniques are required in order to update these fragments synchronously. The first approach is using Handler. By Creating a handler in the current thread, it would be able to send message to the background thread.

Method runOnUiThread() is implemented with the handler.post(). It uses Handler to do the same things. Since it has created the Handler inside the method, we do not need to instantiate Handler by ourselves when we apply this method, which is more convenient for implementation.

Recycler View Recycler view is similar to List View. One most difference is that Recycler View has inbuilt viewHolder RecyclerView.ViewHolder.

Since usually we have many recipes in our recipe list. We would not like it to load all the stuff at the beginning, it is better to use a view holder, to only instantiate the views on the screen, In this way we speeds up the rendering. Also we use Glide to load images from urls, glide is an image loading library that is really smooth when we load big images. With these techniques we make the rendering and scrolling fast and smooth.

Recycler view also provides convenient approach to make the views look smart, for example, ItemDecoreation is used to set the margins between recipe cards; And staggeredGridLayoutManeger is used to create a waterfall layout effect.

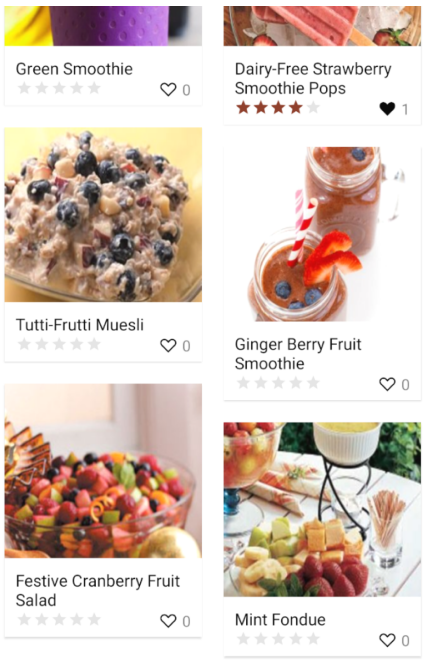

Figure 12: Staggered Grid Layout

3.2 Backend

Architecture In the project, we install *MySQL* and deploy the server side program powered by *Spring Boot* in the cloud server in AWS. We open *8080 port* of the server, through this port, the client side can communicate with the server side. This architecture can make sure each user enjoys the reliable services provided by the server side once he installs the app.

Database Tables We design 8 tables

* **user**: store the basic information of the user, including google_id, email, name. The google_id can help us uniquely identify a user in the database.
* **ingredient**: store all categories and their ingredients
* **equipment**: store equipment. step_equipment uses id to get corresponding equipment.
* **recipe:** store the basic information of one recipe, including its id, img_url, the number of people who like it and the number of people who have rated it.
* **step, step_ingredient, step_equipment:** store steps of making one recipe.
* **user_recipe:** related to the user behaviour. The table creates the connection between the user table and recipe table. This table records the data of behavior of one user performing on a specific recipe. In the future, we can analyze the user behavior with the data stored in this table and implemented the personalized recipe recommendation function.

Spring boot In the server side, We design and implement a collection of restful APIS that can receive the http requests from the client side and send back the json data. The client side uses the **OkHttp** framework developed by Square to send the Get/Post requests to the server side and retrieve the json data. Once it successfully retrieve the data, we use a tool named **fastjson** developed by *alibaba* to parse the json data. It can directly transform the json data to the objects that we will use in the Android program.

4 Technical Challenges

Update UI Threads Since Android does not allow different threads send messages to each other directly, we have to deal with the problem how to update the background thread when we go back from the current interface to the background interface. We found two solutions to this problem. The first is to use Handler. We use this approach to update the data when we go back from recipe instruction interface to the recipe list interface. Another solution is runOnUiThread() method. We use it to set the progressing bar.

API Data Results queried from Spoonacular API sometimes not very correct. For example, he recipe steps will miss something, or its content is even not the step at all. This case seldom happens.
The other API data problem is the **used and unused ingredients**, they are also not very accurate, sometimes they are different from our selected ingredients which we use to request.

Apk Keytool To integrate our app into apk and make it run on different devices, we generate a *keystore.jsk* to our app. Since we use Google SignIn and Google account, we need to register the SHA-1 in keystore on google and import a *credential.json* into *app/* folder. With the keystore, we can use the same SHA-1 to generate the credential.json. Then all user could use our app in any device and sign in with their google account.

5 Future Work

There are also some future work we have not done currently. They are in our work plan at first but we do not have enough time to implement them and they are not the core function of our app.

Fling in ScrollView For now, to refresh the recipe interface, you can only fling the whole fragment instead of fling in the ScrollView which contains recipes.

Recommendation We want to recommend recipes by user favourite or weather. For example, if it is cold like snowy, our app will recommend some hot food to warm you body.

Social Functions To make user can communicate with each other, we also want to add some social functions. For example, after cooking, user could took a photo of the meal or publish some detailed comments so that other users can see them and learn experience from them. Besides, if a clever user invents delicious or novel recipes, he could also upload them and share with others. We also want to add follow and followers functions, this will help users to follow experienced bloggers and learn cooking skills. We think social functions can make user rely more on our app and attract more users to join.