CS683 Project Assignment   
Grocery Shopper  
Alaap Bharadwaj

[**Overview**](#_g6igqliy7rm) **2**

[**Requirement Analysis and Test Result**](#_9dheewbiht5g) **2**

[**Design and Implementation**](#_312k3b3li0xh) **2**

[**Project Structure**](#_hkcglxnjhrt2) **2**

[**Timeline**](#_qws9eve21c07) **2**

[**Future Work (Optional)**](#_92xr55qh0x78) **3**

# Overview

(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed application. This can be the same as in your previous document. If you change it from the last iteration, please make a note.)

Through quarantine people have become more accustomed to cooking at home because it is both healthier and cost efficient. However, with so many recipes out there in the world it is hard to pinpoint exactly which recipe works and provides the required portions and nutrients needed. Users will be able to pick from a variety of recipes from various cuisines that have been tested, and will also provide detailed instructions on how to go about the recipe. After picking recipes, the user is then given a grocery list with the required portions for a week or even two if the user selects that option. This app will help users save food during the week as the portions will be as close to exact as they can be, and will also help save time before going grocery shopping.

# Related Work

(Please describe any similar applications that you have found through the online research, and the differences between your application and those applications. This is based on your project proposal. You can make any adjustment as needed.)

Jupiter: Jupiter is an app and website that allows users choose ingredients and dietary restrictions and pick from various recipes and is delivered to your door. Jupiter allows the users to pick from various recipes and discard any ingredients they do not want in the recipe. The difference between Jupiter and the app being proposed is that Jupiter expects all users to place on online order for their groceries, however, many people prefer to go to the grocery store and buy. The app being proposed will display users the closest stores they can purchase from and if the items they are looking for is available in the store or not.

Previous Student example: In the previous student’s project, the program is taking the ingredients currently in the kitchen as input and various recipes with those ingredients is then presented. In the app that is proposed in this document, it is helping the user create a grocery list for which they can create multiple dishes with.

# Requirement Analysis and Testing

(Please consolidate all your previous iteration documents, and describe all requirements implemented in your application. Each requirement includes a detailed description, and the testing results such as screenshot(s) of the application screens or log info. Highlight the requirements you have worked in this last iteration.)

I have decided to put the log in/sign up implementation towards the end as having user information and storing will be more useful alongside the optional and desirable features. For example, one way this would be helpful is if it is a returning user then with the optional feature of ordering online we can save the users checkout information so that they do not have to input every time they create an order. Some progress has been made towards the sign in page, however, this was put on halt as I want to prioritize on other features since they are more important.

On this iteration, I have solely focused on the UI and the navigation. I drew out the various UI pages I want for this app, by doing so I was then able to number the different UI pages in the order they would be shown to the user. Having done this, this made it easier to navigate through my code to make sure I was viewing the correct fragment from the correct xml files.

After halting progress on the sign in page, I started working on my home page which would display all recipes for the user to pick from. I have started doing research on the API I want to use and would be the most helpful. One API I found that I think is very helpful is the Spoonacular API which provides over thousands of recipes with detailed documentation.

Shape, square

Description automatically generatedText

Description automatically generated with medium confidence

Page on the left is the home page that gives the user the option of signing in and continuing as a guest. Page on the right is the home page that allows the user to go back to the welcome page and continue to the cart where the user will be presented with the recipes that were picked.

The sign in button currently is unavailable, however, this will be implemented later as a desirable feature. I have created this navigation so that there is room for updates in future.

Shape, square

Description automatically generated

This page is the cart page with two buttons, one that allows the user to go back to the home page so that they may pick more recipes before checking out. The second button allows the user to continue to the ingredients list after just viewing all the recipes they have chosen.

**Iteration 2:**

For this iteration, I worked on the recipe page mostly, and getting it to work with some default recipes that were created so that I may test the functionality of the recipe page well. I have created a recipe list recycler view in my home page, however, when a recipe is clicked it does not navigate to its respective detail page, I have been able to find the problem in my onClick implementation. One of the fixes I was thinking of doing was changing my home fragment to a home activity and create fragment container views to handle all the fragments that occur after the user has signed in or continued as a guest.

Graphical user interface, application, email, Teams

Description automatically generated

**Iteration 3:**

For this iteration, I have worked on the recipe list feature, to be able to view the recipes and add them to the cart. I have switched my structure of files to work with a view model so that I may be able to add all the recipes to the room database and then when recipes are added the recipe is added to another entity that holds all the recipes that the user has added to their cart. The two features that are being worked on currently are the home page recipe list and the cart page that both displays a list of the recipes while on displays all the recipes, the cart page displays only recipes chosen by the user. (partially completed -> viewmodel initializing problem in HomeActivity so unable to view home activity – but from previous changes list view was working)

For the implementation of the cart recipes list, I was debating on whether to use a database to hold the cart or just using a list variable to hold the recipes, making it slightly easier. I am still debating between the two (need professors input) as putting it in the database makes it more open for expanding in the future, for example if put into a database we are then able to access old grocery lists, so if the user wants to have the same thing then they do not have to find it through the recipe list but can just go to their old grocery lists. (partially completed)

Another implementation used for this iteration was changing the way the buttons were on each fragment, so I created two fragment container one for the recipe list, cart, recipe detail and one to hold the menu options, so that no matter what page after the welcome will have the menu options below so that it is separate from the fragments. (partially completed)

The goal for this iteration for me was to separate everything like the way we were taught in class, I focused on structuring my project so that I may be able to view my project in a better fashion instead of a mess. For some structure rearrangement here are some diagrams I drew out to visualize the new structure and how it would work alongside the other fragments and activities.

A white board with writing on it

Description automatically generated with medium confidence

Text, whiteboard

Description automatically generated  
A white board with writing on it

Description automatically generated with medium confidenceText, whiteboard

Description automatically generated

**Iteration 4:**

For this iteration I have completed the API get requests and able to display them on the recipe list recycler list. I am able to pick various recipes and look at their information, for example, I can look at the ingredients and other stuff as well. Currently the page is just the basic page where it shows only the recipe name and ingredients used. I am in the process of making the UI look nicer for the detail page and as well the recipe list fragment. The main goal of this iteration was to complete the higher level functionality such as talking to the API and room database and then focusing on the details involved with each page.

Graphical user interface, application

Description automatically generatedChart

Description automatically generated with low confidence

The screenshots above show the recipe list page and the detail page after having applied the API functionality.

**Iteration 5:**

After having completed the API functionality. I started working on making the UI more friendly with the user. The first thing I focused on was to display the recipe in a neat manner, and give the user the ability to the view all the information available for the user. The screenshot below shows how much the detail page has changed, from just showing the recipe name and the recipe ingredient. Now the ingredients are nicely formatted along with the instructions as well, the user is also shown the quantities of each ingredient along with the units associated with it.

Text

Description automatically generatedGraphical user interface, application

Description automatically generated

The two screenshots above, as you can see there has been major changes to the previous iteration. The change in the menu button, as well as adding images to the card view and to the detail page, therefore, giving users more information to set their cart.

A lot of additions were made, one of them is the cart page. The screenshots below show the two possibilities with the cart page. One where there could be recipes in the cart or one where there is nothing in the cart. When there is nothing the user is displayed with a toast message that says that they need to add recipes to be able to view their cart recipes.

Graphical user interface, application

Description automatically generatedBackground pattern

Description automatically generated with low confidence

Some features within the cart feature itself, is now when viewing the detail of the cart recipe you can delete the recipe from the cart. The moment this is done the user is navigated back to the cart page displaying the updated the cart list without the recipe that they had deleted. The screenshot below shows the cart recipe detail page. This still displays the same information as shown in the recipe detail page, however, gives an option to delete.

Graphical user interface, text

Description automatically generated

Another feature that was added and completed was the checkout page. This page focused on displaying the completed grocery list along with the units to the user so that the user may take this list and use it to go shopping. This feature involved adding all the ingredients together and adding any duplicates if necessary while making sure the units are correct as duplicates do occur. The screenshot below shows how the user will be able to view the grocery list.

Text

Description automatically generated with low confidence

The last feature that was implemented was the nearby stores feature. Where the user can view the closest grocery stores near them. This allows the user to minimize the travel time to stores. The screenshot below shows the feature, the blue dot shows the users current location while the red markers are locations of grocery stores near them. When a marker is tapped on the title of the store is displayed so that the user knows which store is which.

Map

Description automatically generated

# Design and Implementation

(Please consolidate all your previous iteration document, and describe Android components and features you have used to implement your application. For each feature you used, provide a brief description and supporting evidences, such as sample code, log info, or screenshots, as well as specify mapped requirements and files/classes in your project. Highlight the features you have used in this last iteration.)

**Iteration 1:**

In this iteration, android components and features were not maximized. I have been researching on the various components that will be used in the pages that I have designed. The biggest component/feature I will be using will be API. For this iteration I focused on the setting up my project to be able to complete feature by feature in the later iterations. Going through the documentation and examples of using the API help a lot in learning how to integrate into my app. By focusing on the setting up the app to more updates over time I feel made me understand the structure of my classes and xml files better.

While implementing the various pages, for my home page I wanted to create a Fragment Container View so that I may keep the bottom buttons the same while the rest of the page keeps changing depending on the button they have clicked, however, when I tried for some reason the app was crashing, I want to try and re-implement this as this feature is very useful in creating some good code and generalizing most of it. By doing this I will also be able to stop copying buttons from page to page and keep it all on the xml file, therefore, if changes need to be done they can be done easily.

**Iteration 2:**

For this iteration, I mainly focused again on the UI and put to practice the advanced UI techniques that were taught in class and throughout the lab. Working on this has been a lot harder than the previous iteration as there are so many more different things that are going on. I used a fragment container view for the recycler view which was hard to implement into my project. One of the problems I had was that I was unable to view the recipes detail. Since the structure is different to what I have been working on, I have been working on changing the structure and changing the home fragment to an activity.

**Iteration 3:**

One of the big features used in this project iteration was the room database. I used this for the Recipe table, so that I may hold all the recipes that will received by the API (worked on next for the recipe table). I have not implemented the API yet because I want to get all the functionality to work before adding the API functionality. To do this I am just using the room database and adding default recipe values into the database so that I can conduct tests on the app. As you can in the screenshot below I created the viewmodel and then added recipes to it so that I will be able to work with some default values.Text

Description automatically generated

Graphical user interface, application, email

Description automatically generatedShape, square

Description automatically generated

The screenshots above on the right is diaplaying two different fragment containers, one fragment container that is currently displaying the recipe list, while the second fragment is the menu options at the bottom of the page. The menu options container is constantly there as navigating through out the app. While the fragment container holding the recipe list will keep changing as the different menu options button are clicked. The screenshot to the left and above is diaplying the cart page, which hasn’t been implemented yet but will be using a recycler list to view the recipes chosen by the user.

The screenshot below shows what happens when the recipe is clicked from the recipe list. This page has not yet been implemented to show the recipe and give the option of adding the recipe to the cart. However, I am now able to click through the different recipes, with a few lines of code and debugging I should be able to view and add the recipe to my cart.

Shape

Description automatically generated

**Iteration 4:**

For this iteration, I have used room databases and the retrofit functionality to be able to perform a get request from an API.

Text

Description automatically generated

The code above shows the work I did to perform the API calls. This code is in the repo which enqueues the request and then completes certain tasks depending on if the request is successful or not. Some problems I ran into while implementing this was the mistake in the spelling of the api key tag for the query which ended up retrieving nothing and would create an error for the recycler list as it cannot display null values.

Now that the API is working, I have started working on storing the cart recipes. I have created a room database for the cart and have started adding the cart view model functionality into the project. After doing this I will be then calculating a grocery list for the user to be able to view.

Once all this done, for the UI I want to change up a few things to make it more pretty and more interactive. I want to put my menu options at the top of the page and create a slider which allows the user to slide between their cart and checkout page (grocery list) making it more interactive.

**Iteration 5:**

For this iteration, I was able to complete various features, the cart page, checkout page, and the nearby store page. To begin with for the cart page, I used a room database to hold the cart recipes. The database was added to when the users are prompted to add to the cart in the recipe detail page. The users are given a text box to input a quantity, as this could be different from the serving size specified by the recipe. A viewmodel is used to be able to access the room database. This feature was a bit simpler to add after being able to view the list of recipes. After this all that was required was to be able to parse the objects received from the API call to creating structured strings so that they may be saved into the room database. The screenshots below shows the code that was used to be able to parse the instructions and ingredients into something that is can be parsed again to display to the user.

Text

Description automatically generated

Text

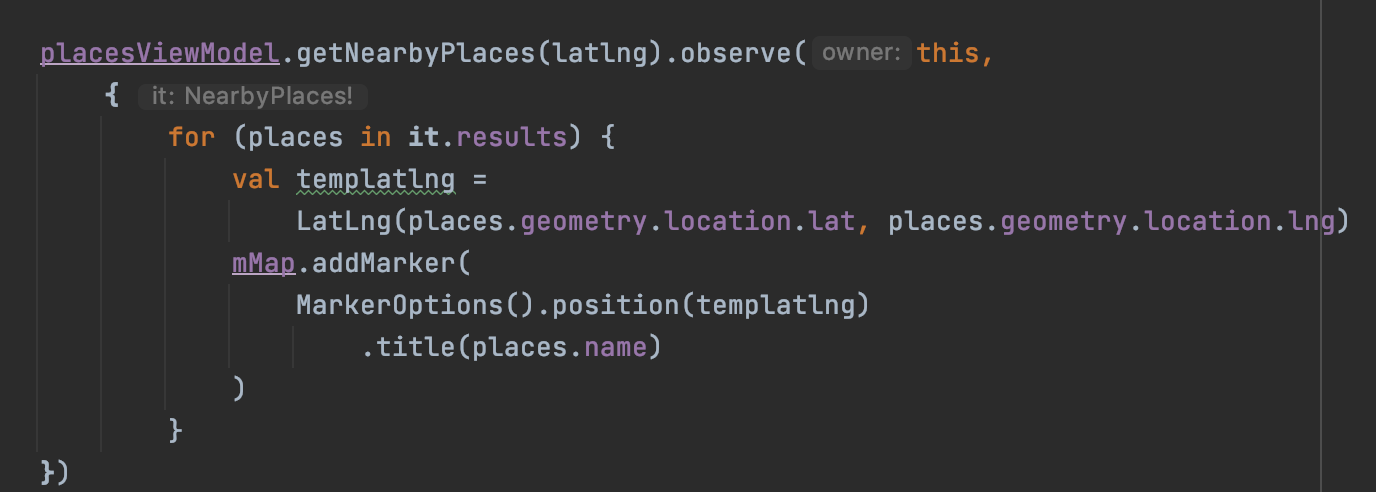
Description automatically generated

For the second feature implemented which was the checkout page, this also used the room database to access the cart recipes so that calculations can be made to be able to display the measurements and units for all the recipes in the users’ grocery list. To achieve a uniform unit between the similar ingredients was difficult but was achieved using multiple functions to have unit conversions. This was tedious but eventually was working to be able to switch the unit of duplicate ingredients so that the user does not see incorrect results for the measurements of the ingredients. The screenshot below just shows a couple of the conversions, while there were quite a lot more.

Text

Description automatically generated

For the last feature that was implemented which was the nearby stores page, this page displays the current location and the grocery stores that are closest to the user. To implement this feature the google places and google maps API was used. The google maps API was used to display a map interface to the user along with a blue dot for the current location and red markers to highlight the closest grocery stores. The google places API was used to retrieve nearby places based on the location to search around, the radius, and some keywords to describe what kind of places we want to be retrieved.



The snippet of code above shows the method used to fetch the nearby places. As you can see I am using an observe so that if the current location does change then the nearby locations may change as well when requested. Within the observe I perform a for loop so that I may mark all the locations that are being returned onto the map for the user to be able to view.

# Project Structure

(Please provide a screenshot(s) of your project structure, which should show all the packages, java files and resource files in your project. You should also highlight any files/packages you have changed, added/deleted in this iteration comparing with the previous iteration).

Text

Description automatically generated

The structure to the left has been changed a lot. The picture to the left is the newest version of the structure. The files have been split to 4 main packages, repository, sources, ui, and viewmodel. Creating these four packages helps create some separation between all the files and functions that are run in files itself. As the program gets larger and larger there will be more files, by separating it this way it becomes easier to find your way and find the files you are trying to find.

# Timeline

(Please provide an updated status to specify in which iteration each requirement is implemented using which Android features.)

|  |  |  |
| --- | --- | --- |
| Iteration | Application Requirements  (E/D/O) | Android Components and Features |
| 1 | Recipe List Displayer (Done) | Spoonacular API  RecyclerList  Room Database (Only Adding) |
| 2 | Cart List Displayer (Done) | Room Database   * Holding Cart Recipes   RecyclerList |
| 3 | Checkout Page (Done) | Simple ListView   * Using Array Adapter   Room Database   * Accessing Cart Recipes   Unit Conversions   * Ml, cups, cups, tbsps, tbsp |
| 4 | Nearby Stores (Done) | Google Maps API  Google Places API  Allowing permissions for fine location/coarse location |
| 5 | Display Availability of Items at Stores (Incomplete) | Use Various APIs   * Of various different stores   Use maps to display and then allow for detail page of the store and maybe what ingredients are missing at the store they are viewing |

# Future Work (Optional)

(Please describe possible future works. Particularly the requirements you planned but didn’t get time to implement, and possible Android components or features to implement them.)

Display Availability of Items at Stores:

This feature is a very cool one, as it displays the user if the availability of the items in various stores. However, since the information for this is very hard to come by only a few stores actually have live data of the availability of the items. This feature will use Various APIs Of various different stores and then use maps to display and then allow for detail page of the store and maybe what ingredients are missing at the store they are viewing

Signin/Login Page:

This feature will be implemented in the future, so that users may be able to view their past grocery lists and carts so that if the user liked one of the recipes from a previous order they may be able to go back and add it to their current cart.

Allowing user to choose radius and change current location:

This will help the user pick and choose where they want to search for stores, by doing this the user can search it up on the go and do not have to be home to be able to view the closest stores.

Order Orders Online:

By having this feature, it allows the users to have a choice between ordering all the groceries online or going to the grocery store physically and then shopping. Various people prefer both, so by adding this feature the app becomes more open for a larger target audience.

# Project Links

**Project Presentation Kaltura Capture Link**

Please submit a link to your video or use the Mashups tool to upload your video for review by clicking the "Write Submission" button. It is recommended that you use Kaltura Capture. Kaltura Capture can be accessed through the MyMedia link available from your Online Campus Dashboard (not directly from the course itself). If you have not used Kaltura before, please see this [Kaltura Capture information sheet](https://onlinecampus.bu.edu/bbcswebdav/pid-6970632-dt-content-rid-19162119_1/xid-19162119_1) with instructions on recording and posting videos. You may also upload your video to YouTube if you like.

If you host your project on github or bitbucket, you can share the link here. (Optional)

# References

<https://spoonacular.com/food-api/docs>

<https://www.jupiter.co/home>

<https://rapidapi.com/logicbuilder/api/target-com-store-product-reviews-locations-data/>

<https://scrapingrobot.com/blog/costco-api/>

<https://developers.google.com/maps/documentation/places/web-service/overview>