SmartSnacks

Aidan Clark [U87467963], Amy Seedhom [U38474590], Gordon Ng [U82744816], Youngjin Shin [U99506849]

I. Problem Definition

In order to reduce consumer food waste, we need to organize dynamic lists of food products that will inform the user of upcoming expiration dates, affording them the convenience of complete awareness of their pantry's inventory. We need to keep track of the user's recipes in a quick and easy-to-access place. Additionally, we need to create a user-friendly interface in order to ensure maximum satisfaction for our customers.

II. Project Objective

The app will have several easy-to-use features that contain all food-related needs in one place. Users will self-maintain an inventory of their pantry items, by adding entries for each item they currently have. These entries can be further specified by including the amount and expiration date, if applicable. They will be able to delete their entries as well, typically after the item has been completely consumed or discarded. The user will be able to view their full inventory list in three or fewer clicks. They can also indicate certain items as "Necessities" so that they will be able to more adequately keep track of their favorite food products. Additionally, there will be a page for recipes, if the user chooses to input any. Similar to their pantry list, they can add and delete as they wish.

III. Stakeholders

Our sole internal stakeholder is our development company "Smart Snackerz". As a startup, our survival as a company is directly dependent on the success of the project. Our most significant external stakeholders are our investors, who are the backbone of the funding Smart Snackerz is receiving. Given their financial stake in the company, they are looking for the project to succeed for their own monetary gain. Collaborators in the nutrition or health industry may want to implement elements of SmartSnacks within their own applications, and therefore the success of the app will directly improve their own projects. Finally, consumers of SmartSnacks will want the best product possible, and their satisfaction is important to them, but also to the rest of the stakeholders, as satisfied customers will drastically improve the reach of the app.

IV. Acceptance Criteria

Both our industry collaborators and our customers have very similar acceptance criteria. The inclusion of all relevant features, efficient load times, and a clean and clear user interface will benefit all parties involved as customers and industry collaborators are most concerned about the functionality of the product. Our investors are mostly concerned with completion under our $50 budget, as the application will be more likely to turn a profit as cost is lowered. Additionally, for the project to be considered a success, the net profits of the application must meet or exceed our investors expectations. All of the above criteria apply to our company, as we depend on customer satisfaction as well as adequate profits to keep Smart Snackerz afloat. Through our work with unaffiliated beta testers, we were able to more accurately determine the most valuable success criteria.

V. Use Case Diagram

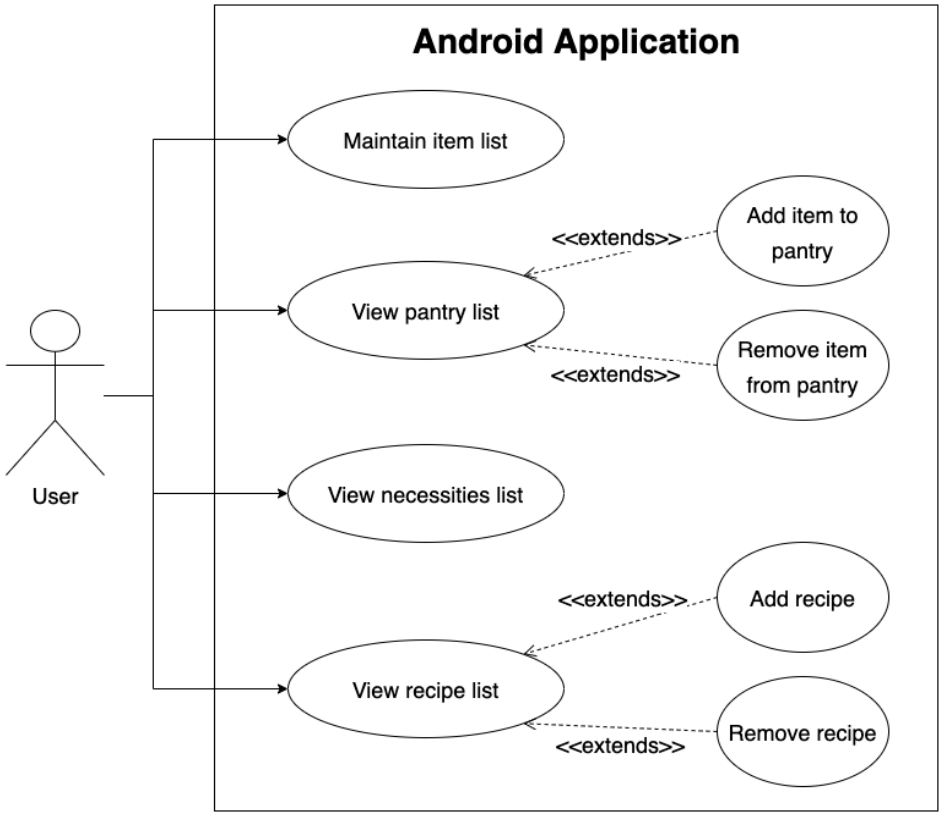

Figure 1: A use case diagram that depicts the user's possible activities and interactions with SmartSnacks.

Our use case diagram depicts our actor, the user, interacting with SmartSnacks through an Android application. Our basic features are depicted here, a self-maintained item list, an editable pantry and recipe list, as well as a necessity list for the user's favorite items. There is only one actor, as the user is able to maintain and frequently update all of the required information by themself.

VI. Selected Use Case Descriptions

Use case name: View pantry list
Actors: User
Description: Users will be able to view their pantry list. The pantry list will contain all of the items that the user has previously added to the list. Users can add items to this page through the item page, which will ask the user to input the amount and expiration date of the item. The pantry list page will then include the new item as well as the old items. Users can also delete each item on the pantry list if they choose to.
Stakeholder: User's success depends on their ability to view and maintain their pantry list successfully. The user provides all the information for the list.

Triggering Event: Users open the SmartSnacks application and navigate to the pantry list page.

Steps Performed

- User opens the app SmartSnacks on their Android device. The item list page is opened on the app.
- User clicks the navigation bar to view the page options, and clicks “Pantry” to view the pantry list page.
- User can scroll through the list to view all of their current pantry items in one location, as well as their amounts and expiration dates.
- User clicks the “+” button. A page pops up that asks the user to input an item name that is already in their item list, the amount, and the expiration date.
- User inputs the information and clicks the “Add” button.
- User is viewing the pantry list page again, which now includes the newly added item.
- User can click the “Delete” button below each item to remove the item from the list.

Preconditions: Have the SmartSnacks app downloaded and have some items added to their pantry.

Postconditions: View all pantry items, be able to add or remove them.

Use case name: View recipe list

Actors: User

Description: Users will be able to view their recipe list. The recipe list will contain all of the recipes that the user has previously added to the page. On this page, the user can indicate that add and remove recipes from the list.

Stakeholder: User’s success depends on their ability to view and maintain their recipe list successfully. The user provides all the information for the list.

Triggering Event: Users open the SmartSnacks application and navigate to the recipe page.

Steps Performed

- User opens the app SmartSnacks on their Android device. The item list page is opened on the app.
- User clicks the navigation bar to view the page options, and clicks “Recipes” to view the recipe list page.
- User can scroll through the list to view all of their recipe names in one location.
- User can click on a recipe name in the list, which will bring up another page that includes the instructions and details of the recipe. The user can exit this page and return back to the main recipe list.
- User clicks the “+” button. A page pops up that asks the user to input a recipe name and the instructions for the recipe.
- User inputs the information and clicks the “Add” button.
- User is viewing the recipe list page again, which now includes the newly added recipe.
- User can click the “Delete” button below each recipe name to remove the recipe from the list.

Preconditions: Have the SmartSnacks app downloaded and have some recipes added.
Postconditions: View all recipes and their instructions, and be able to remove them from the list.

VII. Sequence Diagrams

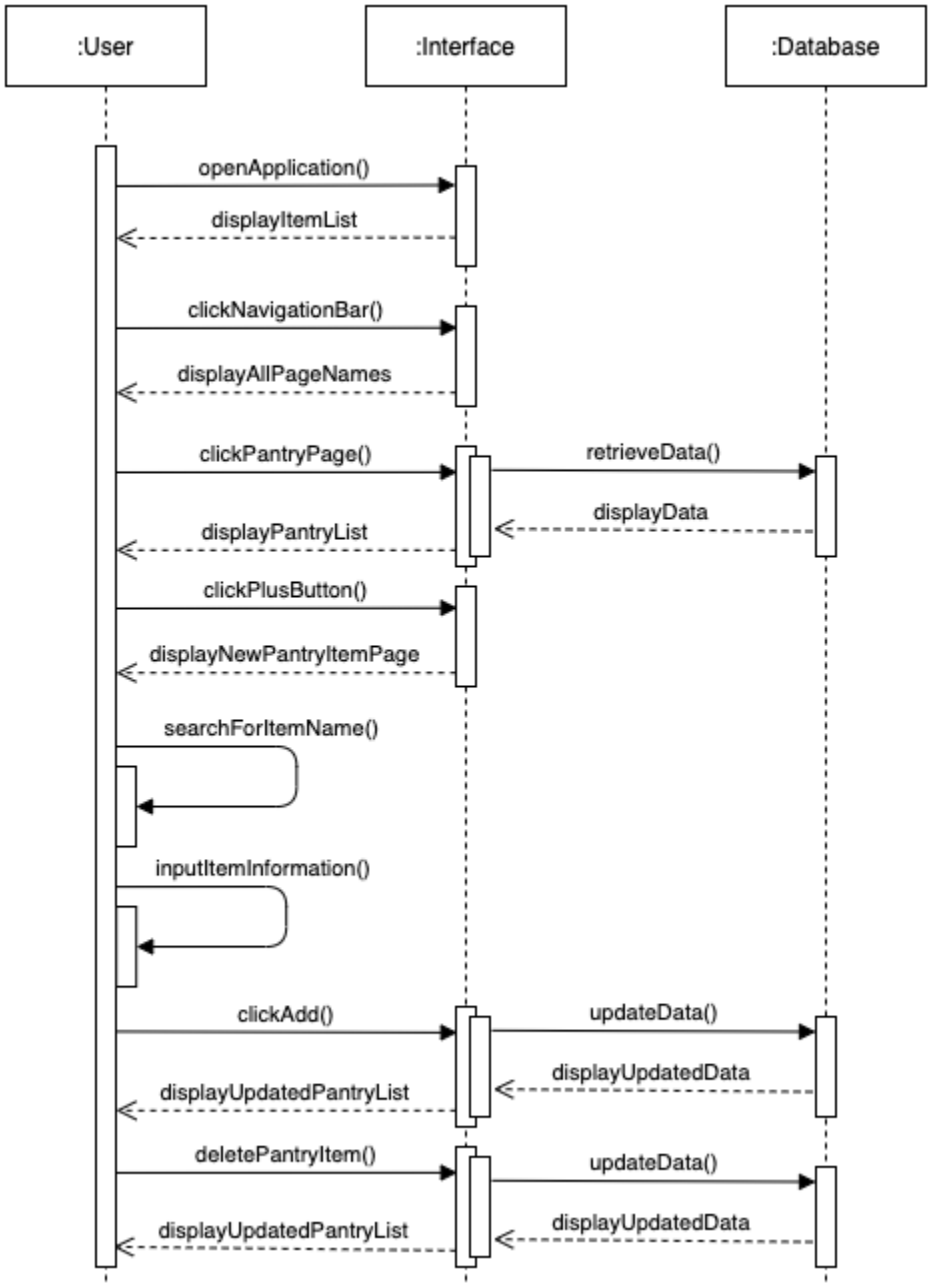

Figure 2: A sequence diagram for the use case of viewing and editing of the pantry list.

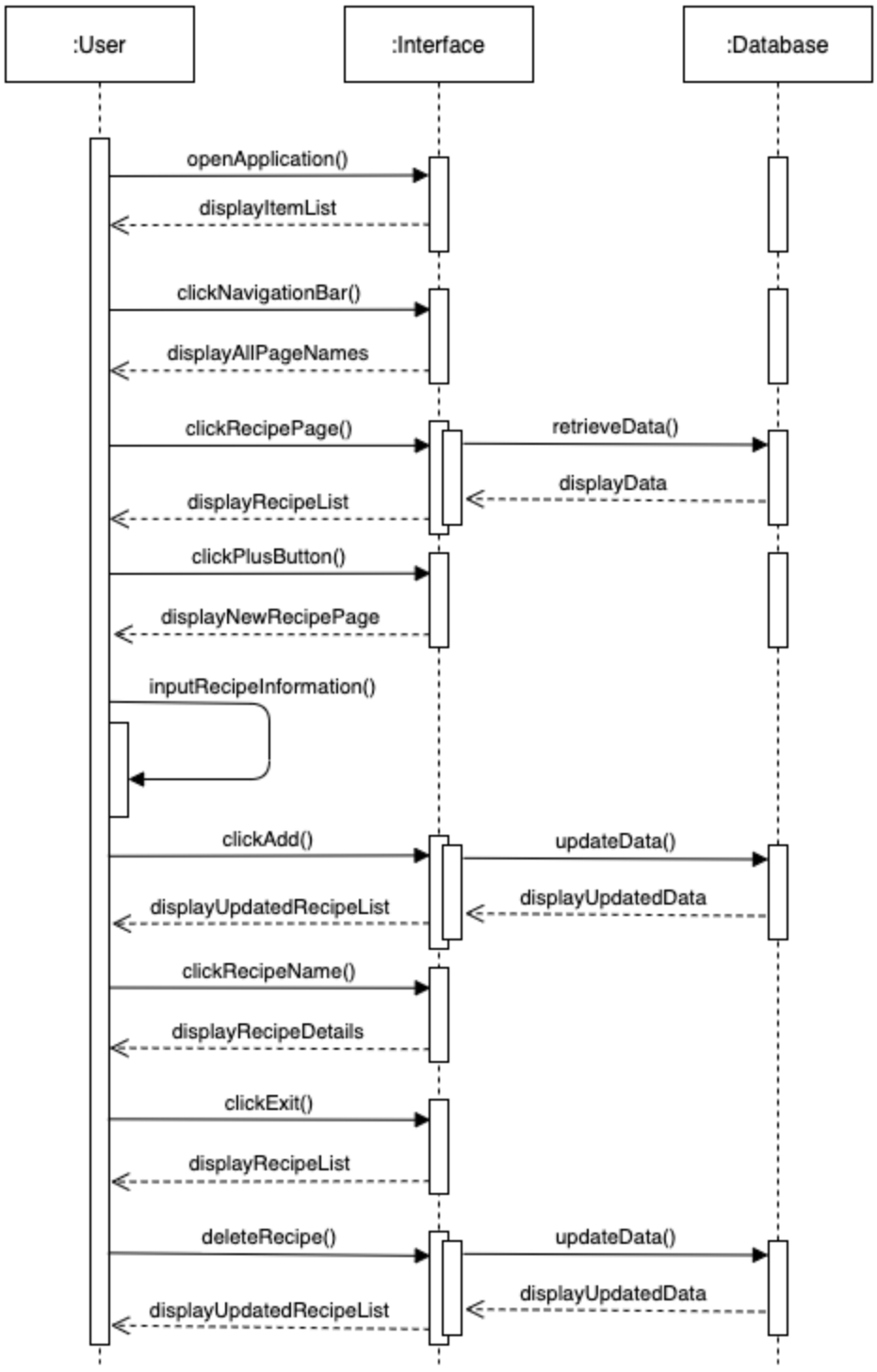

Figure 3: A sequence diagram for the use case of viewing and editing the recipe list.

The two sequence diagrams show the sequence of interactions between the user, interface, and database for the use cases of viewing the pantry list and viewing the recipe list, respectively. The messages between the user and the objects within the system are shown in chronological order from top to bottom.

VIII. System Architecture

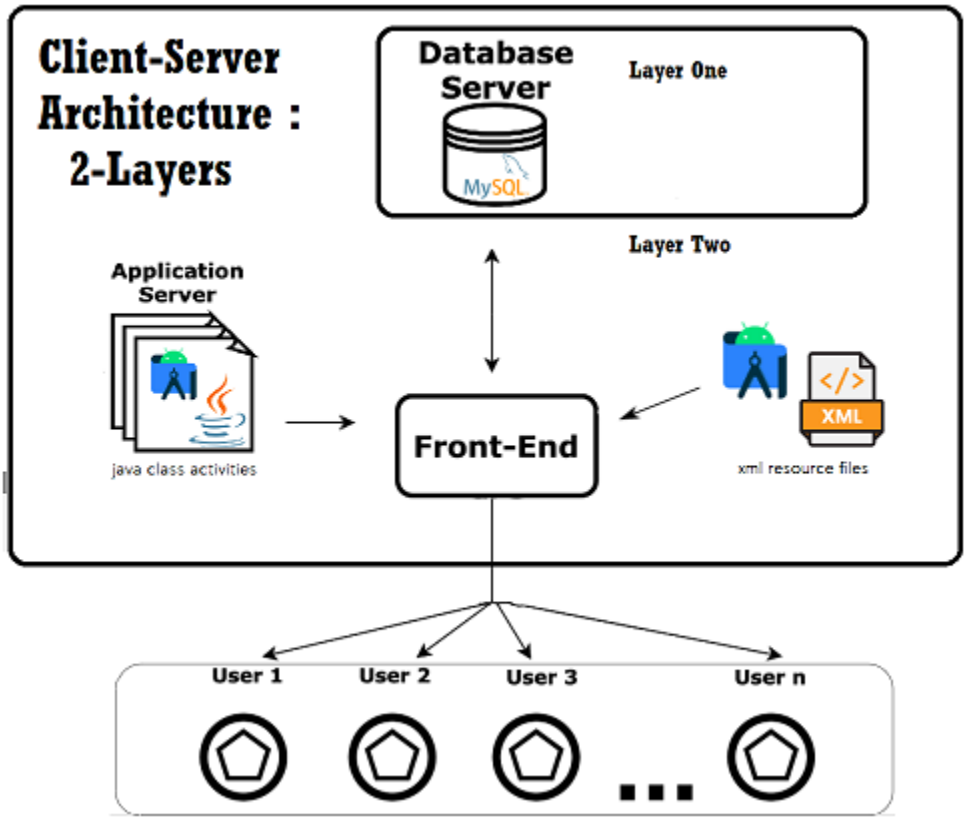

Figure 4: A diagram showing the connections between our components.

This diagram depicts the user interacting with Smart Snacks, which is functioning through a two-layered Client-Server architecture. The front-end primarily consists of Android Studio, implementing XML files for the layouts (made up of fragments and views), which are then organized with Java Class Activities by referencing java identification tags in the XML files. The other layer consists of the MySQL database.

IX. Detailed Class Diagrams

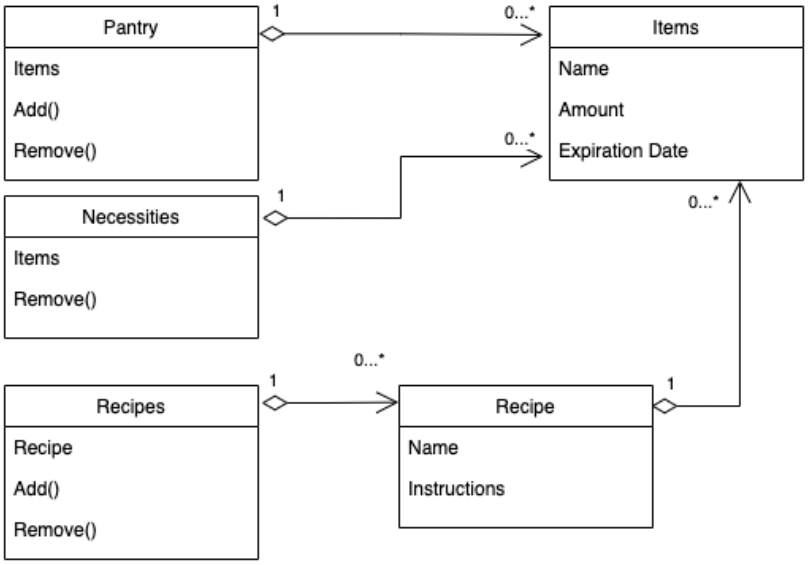

Figure 5: A class diagram showing the relationship between the classes, as well as their attributes, methods, multiplicities, and associations.

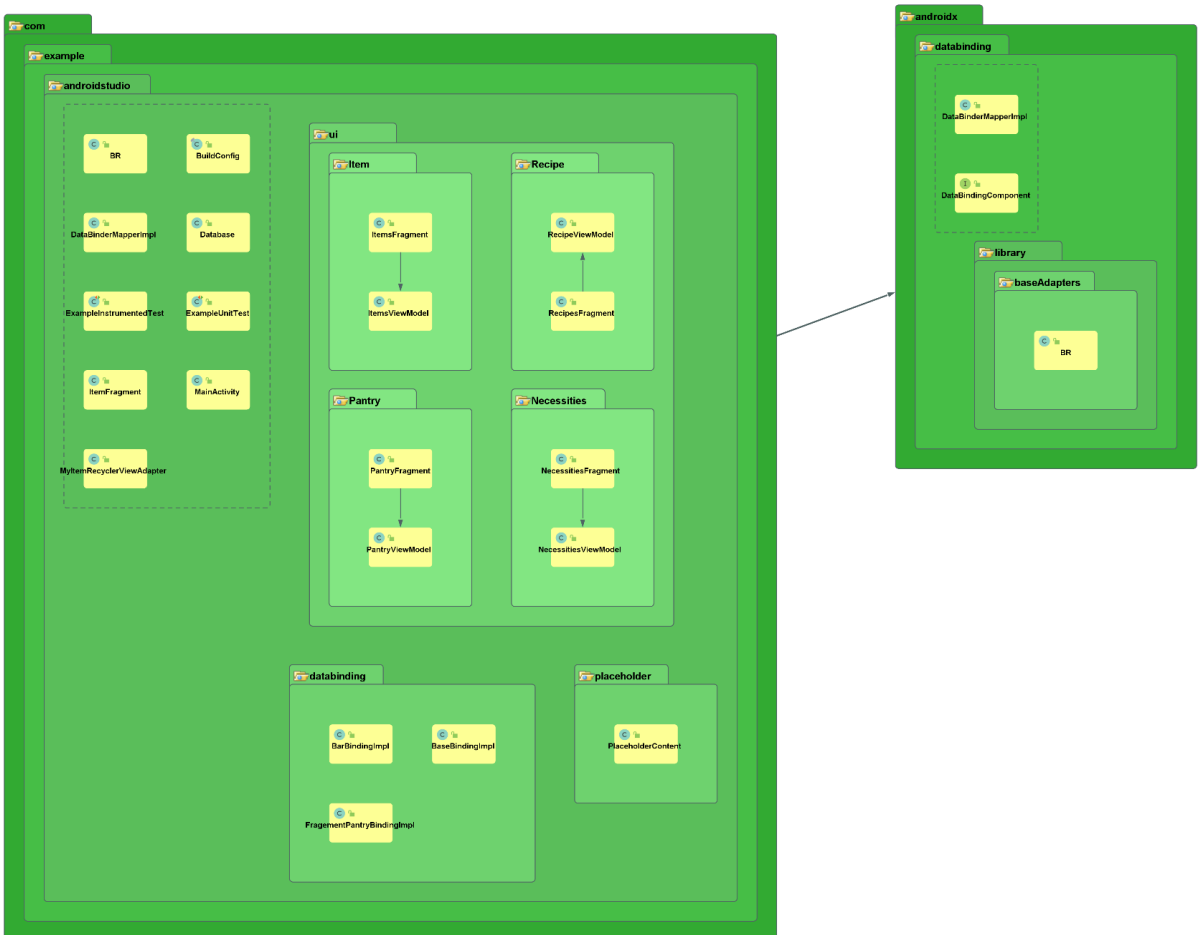

Figure 6: A detailed class diagram showing all of the classes, fragments, and pages within our application on Android Studio.

These two class diagrams depict the pre and post developmental stages of our application. The earlier diagram depicts the raw stages of design when we were not using Android Studio. However, once we moved to the coding stage, the complexity of our classes drastically increased, resulting in multiple class diagrams being required to display the project's progression.

X. State-machine Diagrams

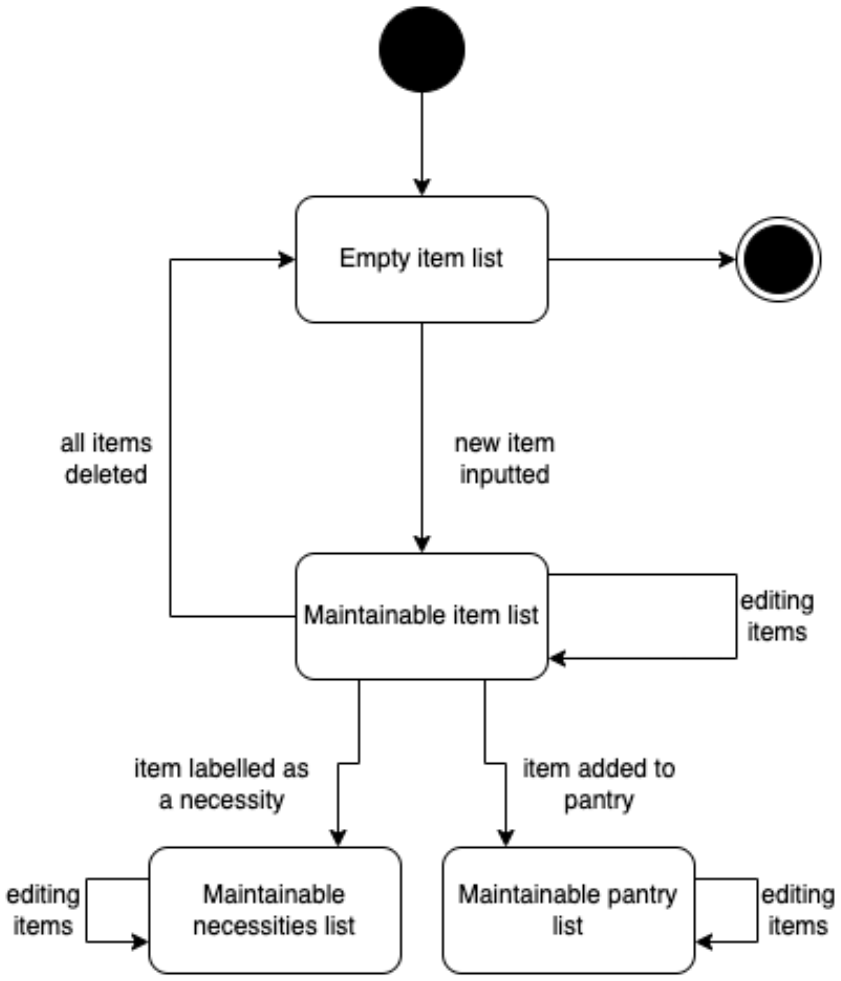

Figure 7: A state machine diagram showing the states of the Items, Pantry, and Necessities classes.

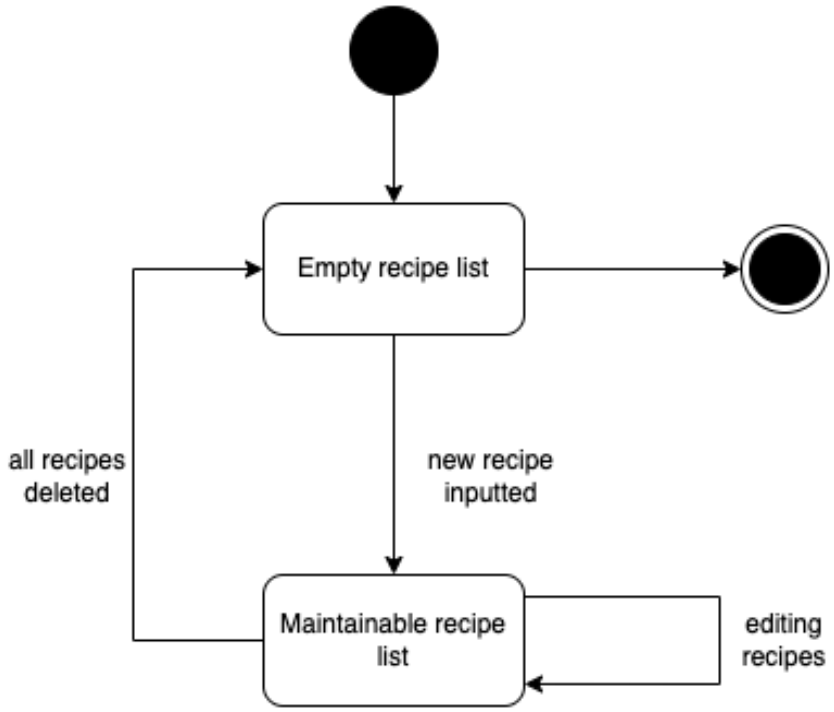

Figure 8: A state machine diagram showing the states of the Recipe class.

These two state machine diagrams show the behavior and several use cases of all the classes within the system. The diagrams are separated into two because the recipe class, which is represented by the second diagram, is completely independent of the other classes. The states and transitions of the recipe class are not related to the states and transitions of the items, pantry, and necessities class. In both cases, however, the starting state is an empty list, which can also lead to the final state, because very limited actions can be taken when a class is at this state.

XI. Entity-Relationship Diagram

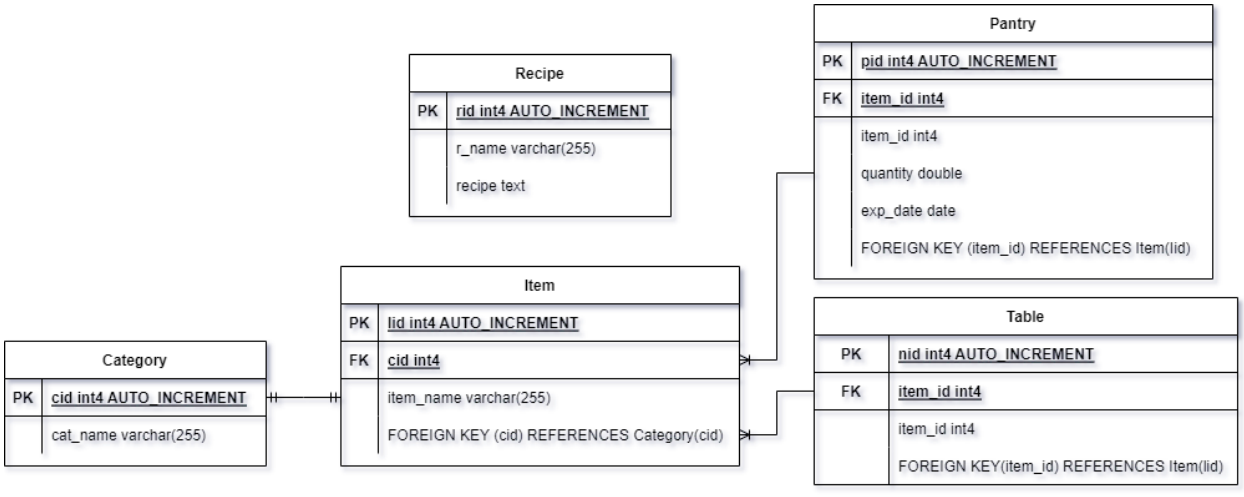

Figure 9: An ER diagram showing the relationship between the database entities.

There are multiple entities within our ER diagram, all of which have relationships besides the Recipe entity. The Recipe entity is independent, and cannot be associated with the other entities. The remaining entities all have an association with the Item entity, as it has become the base for most of our application's functionality.

XII. Github

The link to the GitHub repository is https://github.com/SmartSnackerz/SmartSnacks. All relevant information, including the code for the entire project, is provided.

XIII. Lessons Learned

In our development process, we had to familiarize ourselves with several programs, tools and websites in order to succeed with this project. While some of our members were already acquainted with GitHub, others members had to learn the process of pushing, pulling, and fetching code to collaborate with one another more effectively. All of our members understood Java, but the use of Android Studio to develop the application for Android was foreign to all members involved, and thus led to a massive learning curve that had to be overcome before efficient development could take place. Linking task organization sites like Jira to our GitHub repository allowed us to keep in line with our schedule and designate the work in an appropriate manner. Furthermore, the use of Figma[1] aided in the design of the app, which was paramount in order to keep the clean and clear user interface we dedicated ourselves to in the proposal. Overall, though the workload of developing a functional app was far beyond anything we have undertaken as students, the entire process gave us invaluable insight into what it really means to be a software engineer.

Firstly, the pressure of the deadline was unsettling to many of our members that have not participated in software development projects. With the hindsight of how stressful it is to maintain pace for an upcoming deadline, it is unimaginable what the pressure would be like when the stakes are one's entire reputation as a software engineer.

Secondly, the different angles required to complete a project like SmartSnacks displays the heightened collaborative factor of software engineering. In a project where significant programming, graphic design, diagramming, and write-ups are present, the vitality of designating tasks to individuals is evident. Playing to each group members' strengths allowed us to finish the project in higher quality and in a timely manner.

Additionally, the knowledge required to be a software developer is extensive. Using textbooks for Android Studio and online forum pages like Stack Overflow[2] were essential in getting key tasks done as we have not received any formal training in the Android Development Kit. Outside help made this project possible, and fully cemented the fact that software engineers in the field are very knowledgeable. However, it also became clear to us that no one knows everything. Even the most brilliant programmers were asking for help online, showing that there is no end to learning more about software development.

Lastly, we realized that it was a difficult task to be able to connect our MySQL database to our application in Android Studio. As separate entities, each was able to work, but there were errors that we could not resolve when connecting them together. Our application has functional pages, a navigation bar, and pop

up windows. However, users will be unable to add items to their pantry list because of the connection errors with the database. Therefore, our application is not able to work as intended, and needs additional development to improve functionality in the future.

All in all, our work with SmartSnacks was a massive step in the right direction for our potential as software engineers. If we continue with SmartSnacks in the future, the most essential task would be to successfully connect the MySQL database to the application, so that users can use the app to its full intended functionality. It is possible we would also expand beyond MySQL, likely moving to a Google-backed development app like Firebase as it is more conducive to horizontal scaling[3]. Regardless, and despite our technical issues, this experience has vastly expanded our knowledge on the many facets of software engineering, and will forever be remembered by every one of our group members.

XIV. References

1. https://www.figma.com/community/file/1101225708697042364
2. https://stackoverflow.com
3. https://www.integrate.io/blog/firebase-vs-mysql

XV. Appendix

A. Project Work Breakdown Structure

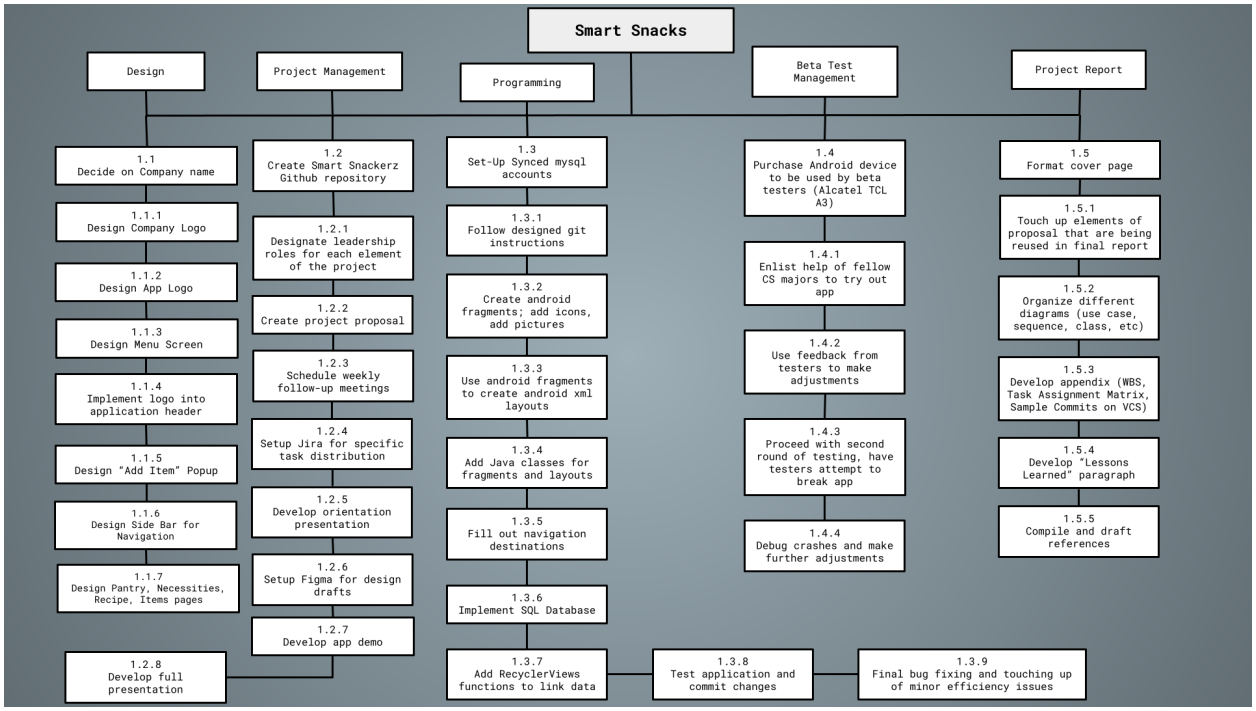

Figure 10: A work breakdown structure for the overall project, including the Android application programming and report components.

The Smart Snacks Work Breakdown Structure was organized into five different sections: Design, Project Management, Programming, Beta Test Management, and Project Report. From the early stages of brainstorming to the final testing stages, all important measures we took in the development of SmartSnacks are referenced here.

B. Task Assignment Matrix

Diagramming	AS	GN	AC	YS
Programming	YS	AC	GN	AS
Debugging	YS	GN	AC	AS
Design	YS	AC	AS	GN
Beta Test Management	GN	AS	AC	YS
Project Report Write-up	AC	AS	GN	YS

Leader
Co-leader
Help

AC - Aidan C.
AS - Amy S.
GN - Gordon N.
YS - Youngjin S.

Figure 11: Distribution of the major responsibilities for each member of Group 8. Color-code and key on the right for increased reader comprehension. “Leaders” and “Co-leaders” managed tasks with supporting input from “Help”.

Very early in our development stage, we determined that the distribution of tasks would be done in an effort to play to our members’ strengths. While our more technical-minded members were more proficient with the coding portion of the assignment, and thus spent the bulk of their time on Android Studio, many other facets of the project, such as diagramming, design, and beta testing were undertaken by the more crafty members of our group. Regardless of strengths, every member played a part in each stage of the project, but the leaders carried the burden of the responsibility in delegating tasks in their category of work. Distributing the work the way we did resulted in each member producing a relatively equal amount of value to the final project, though some contributions show up differently than others.

C. Sample of Commits on Selected Version Control System

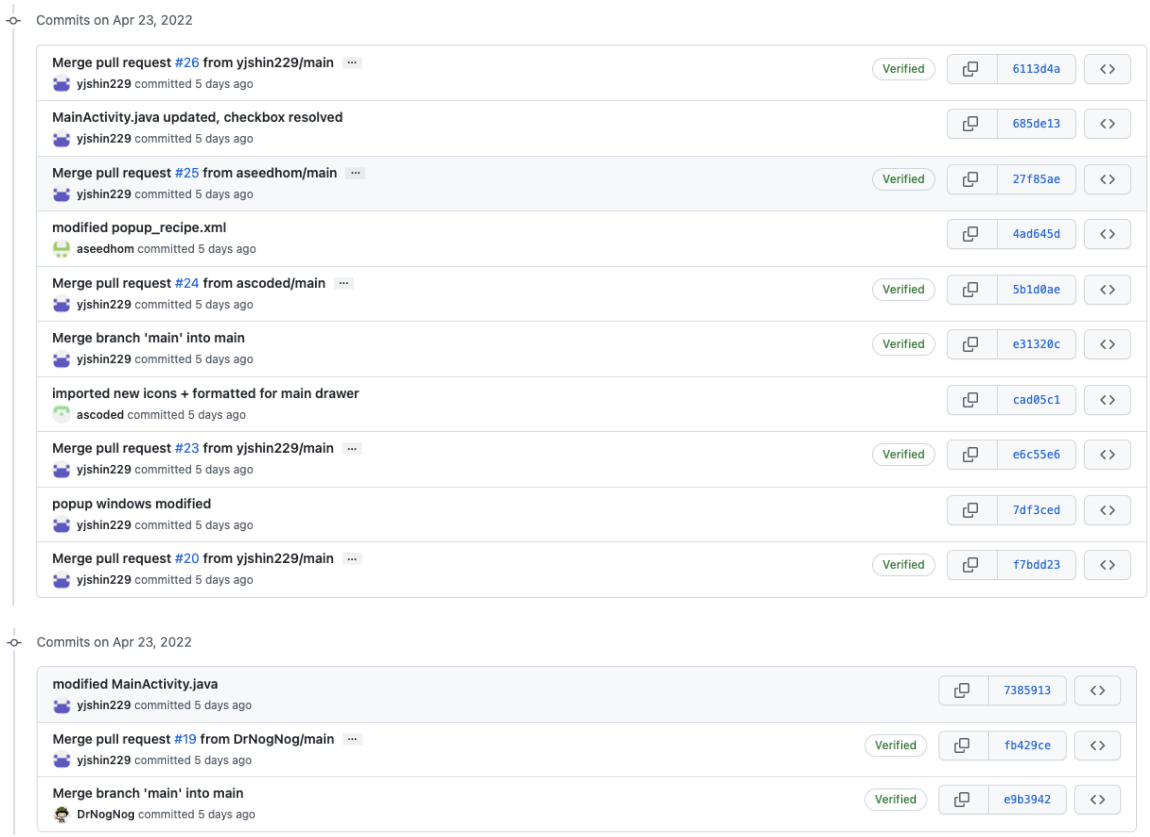

Figure 12: A screenshot of our GitHub actions. Merges were assessed as a full group, ensuring there were no conflicts with each of the commits from all members.

This sample of commits on our Version Control System, GitHub, was taken from April 23rd. This was a day that our group worked primarily on the layouts and XML files for our application on Android Studio.

D. Gantt Chart

	Name	Duration	Start	Finish	Predecessors
1	Decide project	1 day?	2/26/22 8:00 AM	2/26/22 5:00 PM	
2	Project proposal	3 days?	2/27/22 8:00 AM	3/1/22 5:00 PM	1
3	Wait for project approval	13 days?	3/2/22 8:00 AM	3/14/22 5:00 PM	2
4	Requirements documentation	15 days?	3/15/22 8:00 AM	3/29/22 5:00 PM	3
5	Diagrams	15 days?	3/15/22 8:00 AM	3/29/22 5:00 PM	4SS
6	Presentation	1 day?	3/30/22 8:00 AM	3/30/22 5:00 PM	4;5
7	Main features	10 days?	3/31/22 8:00 AM	4/9/22 5:00 PM	6
8	Sub features	7 days?	4/10/22 8:00 AM	4/16/22 5:00 PM	7
9	Working prototype + initial tests	3 days?	4/17/22 8:00 AM	4/19/22 5:00 PM	8
10	Design	4 days?	4/20/22 8:00 AM	4/23/22 5:00 PM	9
11	Final testing	2 days?	4/22/22 8:00 AM	4/23/22 5:00 PM	10FF
12	Final report	6 days?	4/24/22 8:00 AM	4/29/22 5:00 PM	11
13	Release app	1 day?	4/30/22 8:00 AM	4/30/22 5:00 PM	12

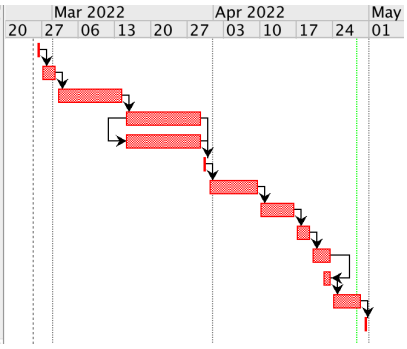

Figure 13: Our Gantt Chart from the beginning stages of our project to the final day.

We created a Gantt Chart using ProjectLibre to pace our project and create a reasonable timeline for ourselves. We included all aspects of the project in our timeline including but not limited to the proposal, report, coding, and testing.