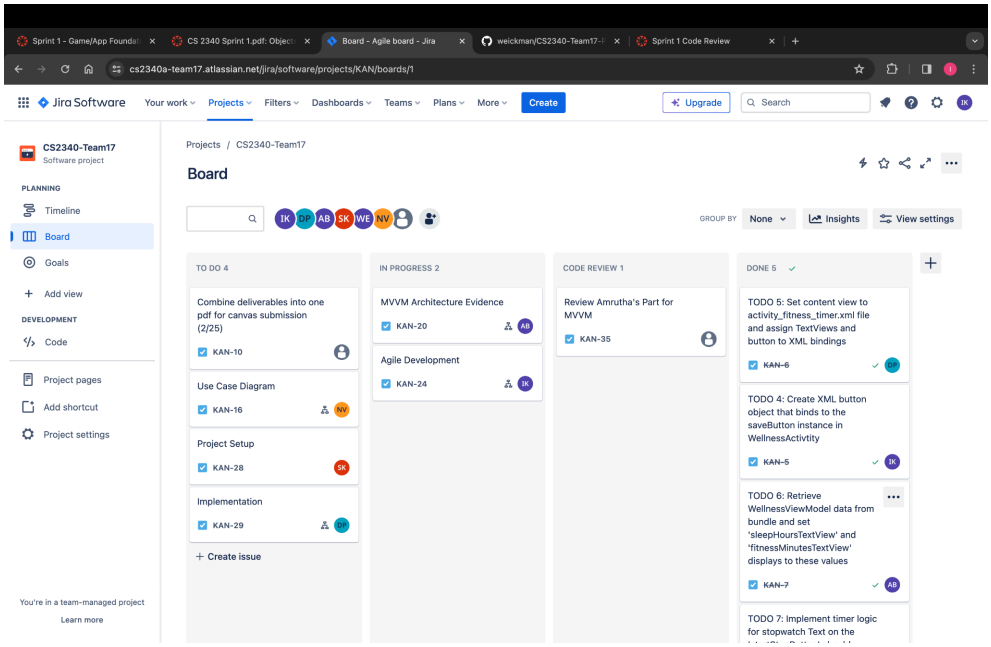
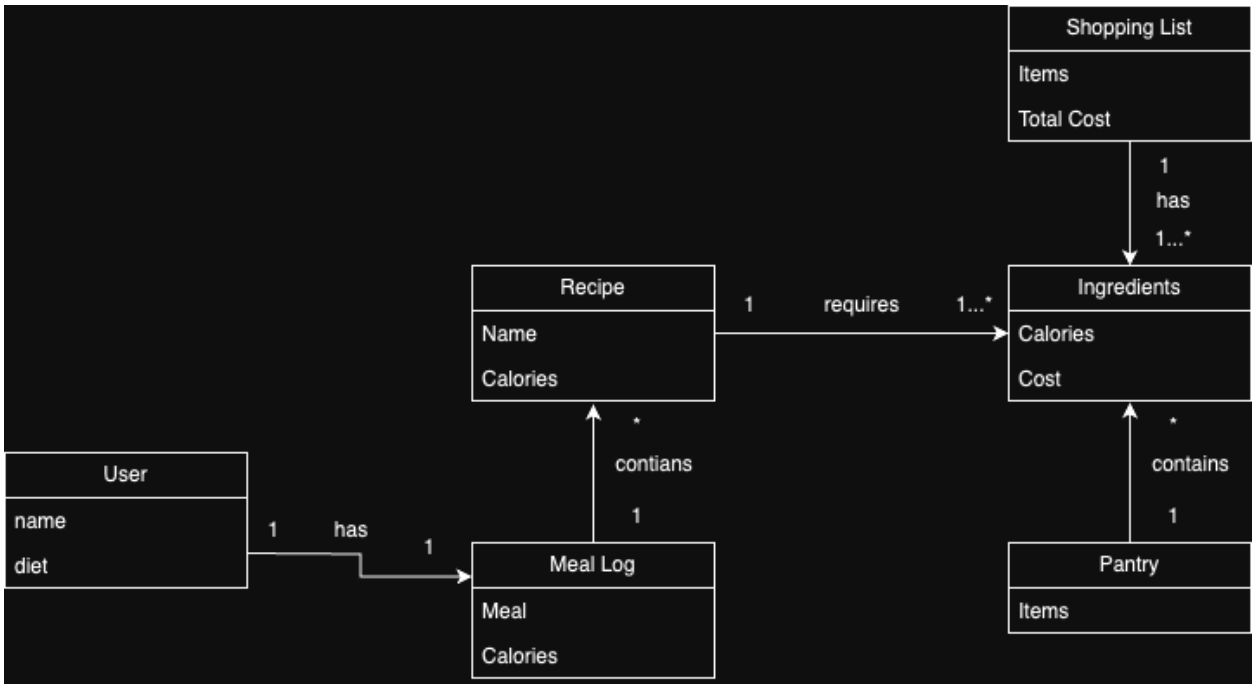
Agile Development



1. The team mainly utilized the project management tool, Jira, to keep track of todos and ensure that each member of the team was comple/ng their fair share of work. Whereas we did have a group chat in which we discussed the division of labor, placing the todos into Jira helped us consolidate the tasks that each person was responsible for. Addi/onally, team members were able to report their poten/al progress on the project management tool without crowding the group chat with messages. For more description on the layout of Jira itself, our group used four columns: To Do, In Progress, Code Review, and Done. We used this same layout for sprint 0.5, but without the code review column and decided to add that sec/on in for sprint 1. The To Do column holds the tasks that have not yet been started. Currently, they are all sprint 1 tasks. The In Progress and Done columns are self explanatory in that they mark the tasks that are currently in progress or completed. The code review column indicates that one of our members has submiLed code that requires reviewing in github. Each task is assigned to a member of the team and some contain subtasks that need comple/ng as well.

2. The team conducted scrum meetings every Friday, mee/ng for thirty minutes from 2- 2:30. During these thirty minute meetings, instead of working on the project, we would bring up any concerns with the sprints, such as trouble with github or jira, or any difficul/es with the workload. If there were any issues with project implementation, or a need for increased understanding on certain ideas necessary to their part of the sprint, we discussed it at the scrum mee/ngs. It was in such mee/ngs that we finalized due dates and planned the progress of the project. Addi/onally, we used about five minutes to provide each other with feedback on our work and communica/on within the group, and used this feedback to effectively improve our respec/ve interpersonal skills.

Domain Model



Classes:

- User
- Recipe
- Meal Log
- Shopping List
- Ingredient
- Pantry

Attributes:

- Name
- Diet
- Calories
- Meal
- Items
- Cost/Total cost

MVVM Architecture

During Sprint 1, our team utilized the Model-View-ViewModel (MVVM) architecture to structure our Android app in a flexible and accessible manner. We carefully separated our codebase into the three distinct layers: Model, View, and ViewModel. The "Model" layer consisted of our data logic, including classes representing entities such as users, recipes, ingredients, pantry items, and shopping lists. These model classes contained data structures and logic to manipulate and manage our application's data. The "View" layer consisted of XML layout files that defined the visual components and UI structure for each screen of our app. Meanwhile, the "ViewModel" layer contained Java classes responsible for interacting with the model layer and preparing data for display in the UI. By sticking to MVVM principles, we were able to address separation of concerns, improve code maintainability, and make testing of our application components easier. Additionally, the MVVM architecture allowed for smoother collaboration among our team by providing clear guidelines on code organization and responsibilities. Overall, the usage of MVVM in Sprint 1 laid a solid foundation for the succeeding development phases of our project.

Use Case Diagram