PROJECT REPORT

1. Introduction

The "Kitchen Organizer" addresses the issue of managing kitchen inventory and reducing food waste by tracking expiration dates and item quantities. This application simplifies inventory management by allowing users to add, remove, and update items in their kitchen inventory. Additionally, it provides features such as search, sorting, and notifications to enhance user experience.

2. Feature Overview

Functional Requirements:

- **Expiration Date Watcher:** The app allows users to add an expiration date to each food item that can be used to alert the user. Foods change color based on their availability. Red foods are expired, and yellow foods are a low quantity. Test: One can test by ensuring the food color is correct when adding a new food, changing the expiration date, or changing the quantity of food.
- **Food Stock:** The app allows users to increase/decrease quantity, delete, and update the foods in their inventory. Test: One can test by changing the food quantity, expiration date, min quantity, and deleting the food to ensure the food updates properly.
- **Alerts:** The app displays a pop-up telling users which foods are running low or have expired. Users can choose to check individual collections or all inventories. Test: One can test by checking that the results for the "Check Current Collection's Inventory" and the "Check All Inventory" buttons display the correct results.
- **Search and Sort:** Users can search for specific foods in their inventory. Users can also sort the display order based on either name or expiration. Test: One can test by changing the value in the "Sort By" combo box and ensuring the foods are sorted properly. Search can be tested by typing something into the search bar, clicking 'Search', and ensuring all foods containing the characters entered are displayed.
- **Add New Food Feature:** Users will be able to add new types of foods into the app's database. Test: One can test by using the "Add New Item" button to ensure new foods are added properly.
- **Multiple Kitchens:** Users can create multiple kitchens in the app so they can keep track of items in multiple kitchens. Test: One can test by using the "Add Collection" and "Remove Collection" buttons and checking that collections are added and removed properly. Also, ensure that the "Select Kitchen Collection" combo box changes collections properly.

Non-functional Requirements:

- **Data Integrity:** Our app stores expiration dates of foods. Users enter the expiration date as the number of days until expiration, but in the database, it's stored as a date to ensure the expiration date is always up to date. When the user retrieves from the database, the date is turned into and displayed as the number of days until the expiration date. Additionally, the user cannot enter a negative number for the expiration date, a non-positive number for quantity, or a negative number for minimum quantity. We also ensure that the user cannot enter the incorrect data type for fields.
- **Volume Test Run:** The app displays a limited number of food objects at a time to ensure that large quantities of food objects do not significantly impact performance. We have tested the app with thousands of randomly generated foods in multiple collections to ensure that the app's performance is acceptable.
- **Usability:** All buttons, text fields, and combo boxes are clearly labeled, and all features that need explaining are explained in the help dialog after pressing the 'Help' button. Additionally, whenever an error occurs, there's an explanation for why the error occurred, and what the user can do to prevent the error from occurring again.
- **Portability:** Our app can run on multiple operating systems including Windows, MacOS, and Linux with no modifications so it is portable.

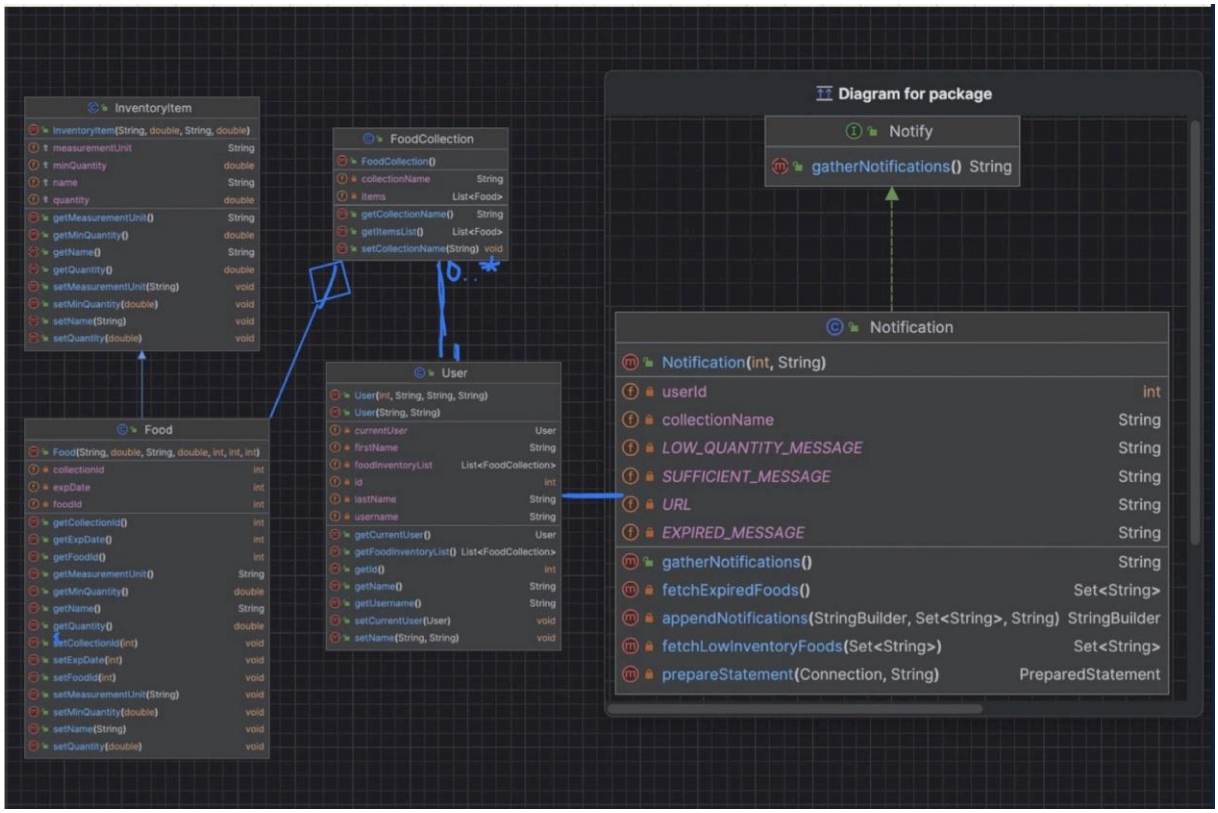
3. Software Architecture and Design

The application is implemented using Java programming language with JavaFX for the user interface. It follows a Model-View-Controller (MVC) architecture to separate the presentation layer from the business logic and data access layer.

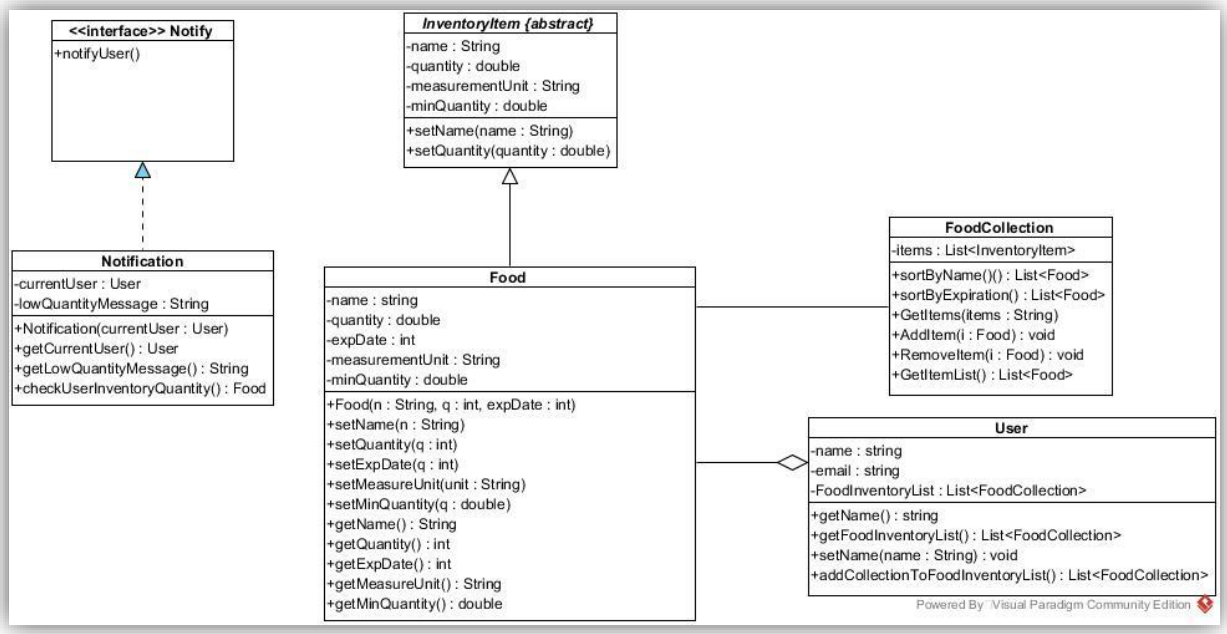
UML Diagrams:

Revised UML diagrams for classes, abstract classes, and interfaces are provided below:

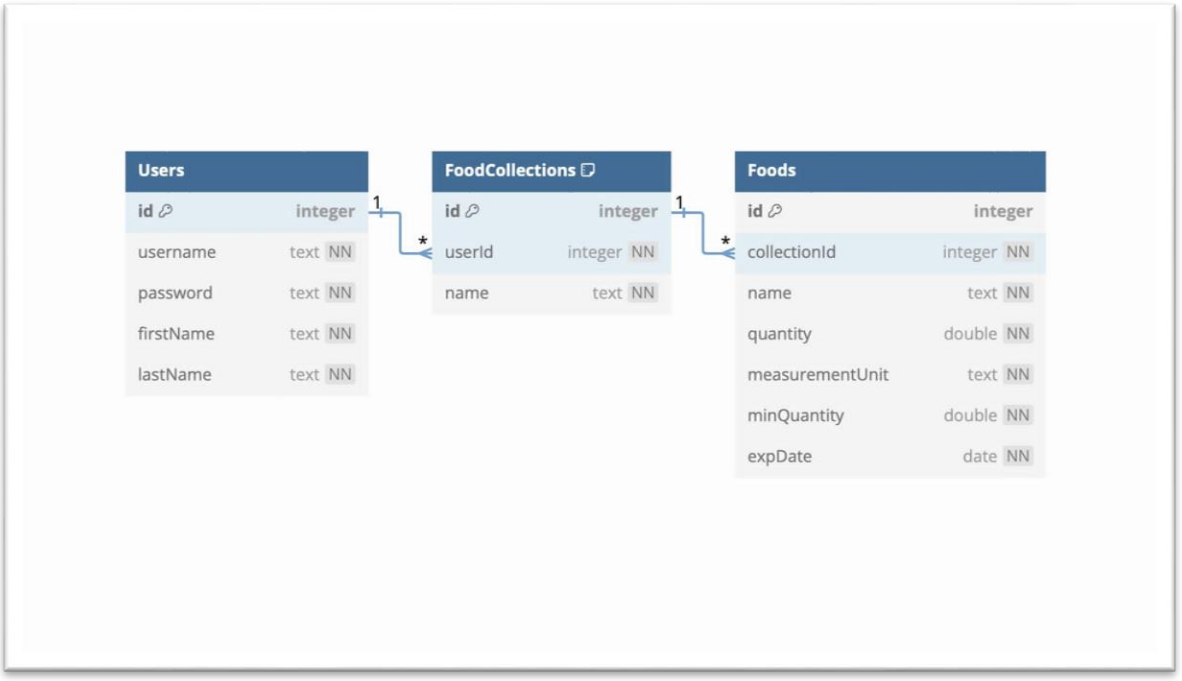
UPDATED UML



Old UML



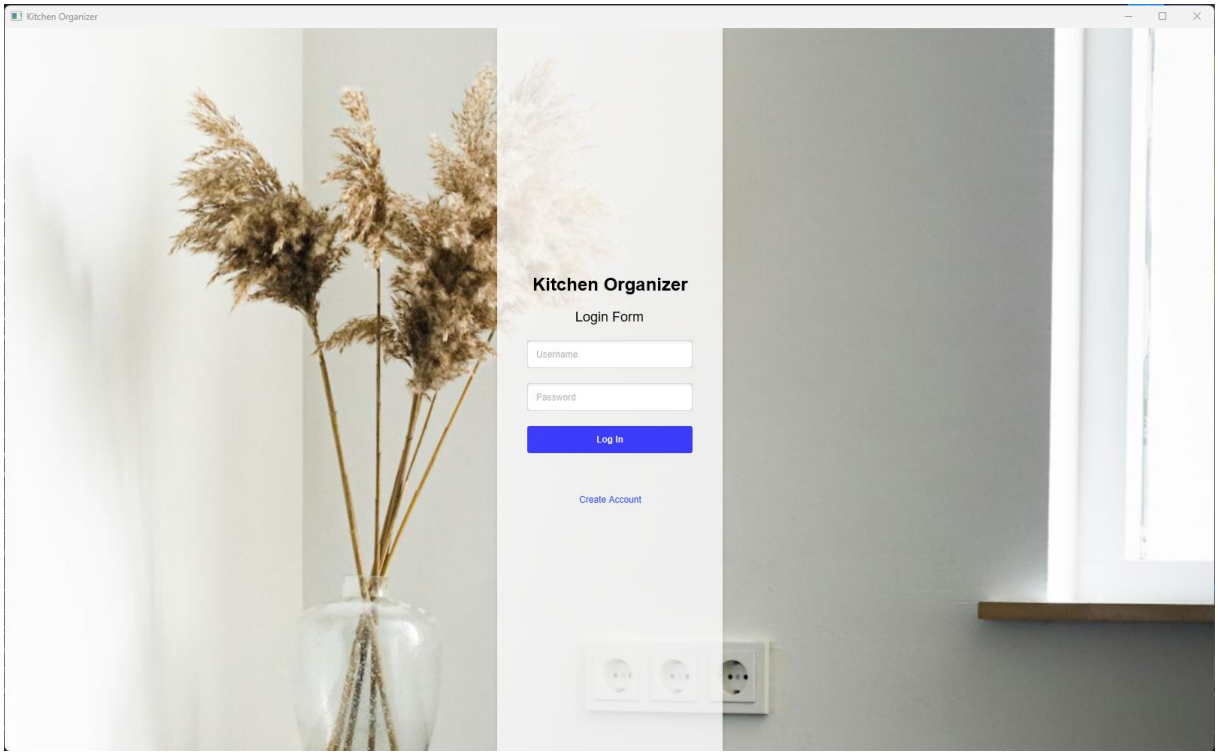
Database Diagram:



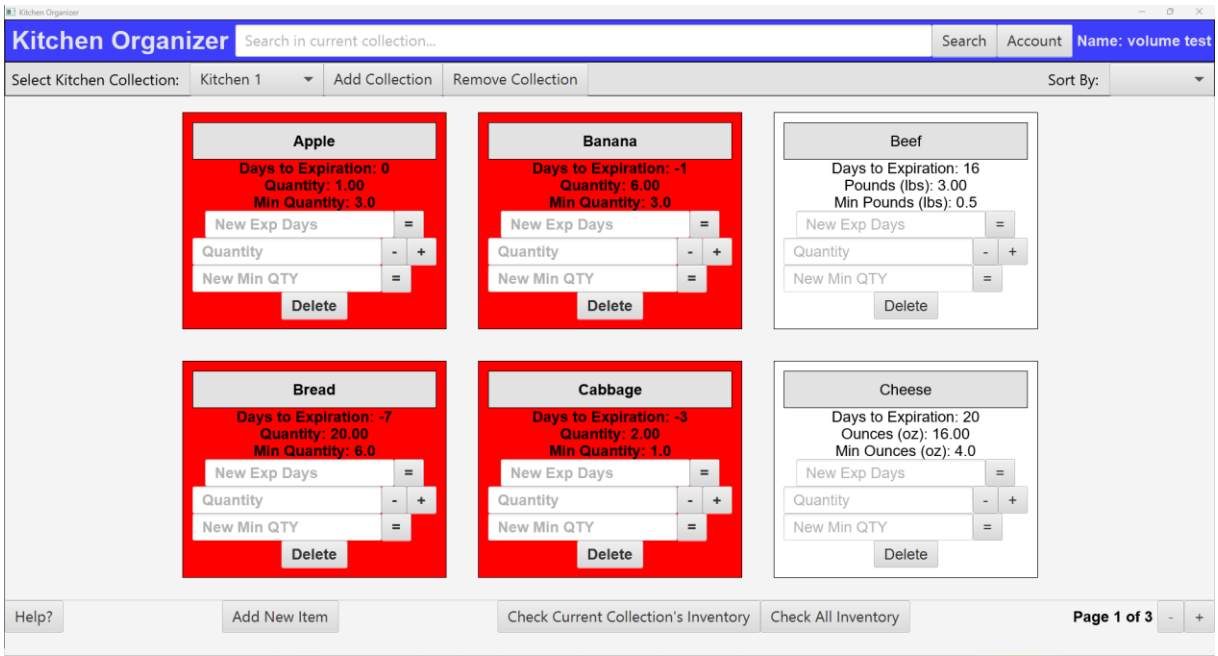
4. User Interface Design

The user interface is designed using JavaFX to provide a modern and intuitive experience for users. Wireframes and design flats are included to showcase the final UI design.

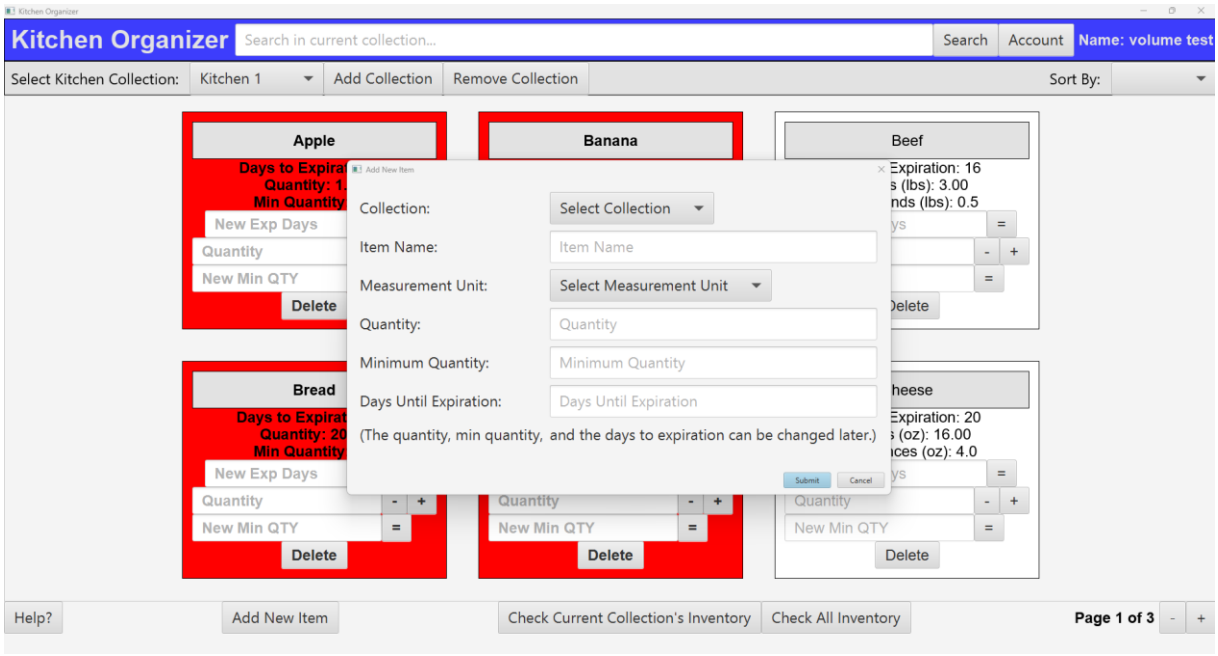
- Login screen



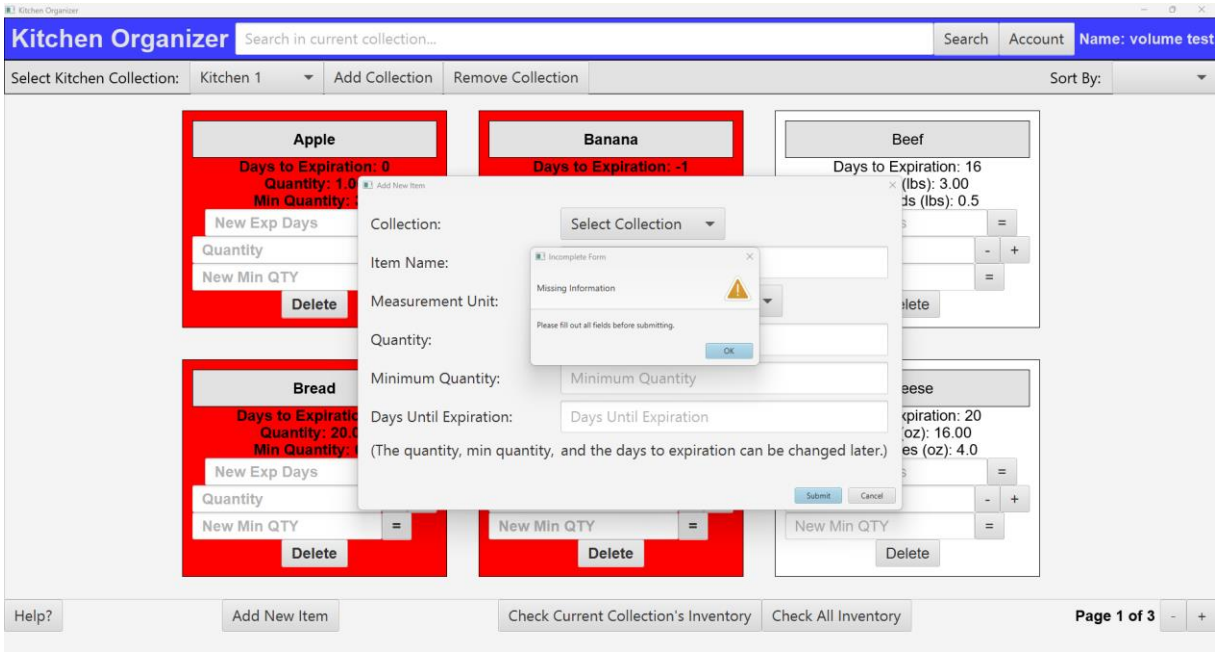
- Main screen



- Add item screen



- Error state



5. Development Process

The development process involved organized teamwork, code sharing, feature brainstorming, testing, and feature completion evaluation. We utilized version control systems like Git for

code sharing and collaboration. We divided the work among group members and gave feedback on other's work. For milestone two, we had two people figure out what classes we should use, while one person worked on the wireframes, and another worked on the UML diagram. For milestone three, our communication could have been better, and we could have had better awareness of tasks. We had issues with getting IntelliJ and GitHub set up. One group member ended up doing much of the work since others did not realize how close we were to the deadline. For milestone four, we divided the work into the main page, database, login screen, and project report/presentation. The work was more equally balanced, but one person worked on some things from another topic since their work was dependent on it. For the final project, we did a rehearsal for the presentation and estimated it would take about 18 minutes, but we [will probably make] some changes so the actual time may differ slightly.

6. Review

What Went Well:

Successful implementation of core features according to the functional requirements.

We were able to fulfil the requirements for the milestones on time.

We didn't have many disagreements or conflicts in our group.

What Didn't Go Well:

Initial challenges in understanding and implementing certain features.

Time management could have been better since we had to rush to get certain things done by the deadline.

Awareness of tasks could have been better since we did not discuss all things that needed to get done in our meetings, and this affected our productivity.

What Would We Do Differently:

Allocate more time for comprehensive testing and debugging.

Do most of the work a week or two before the deadline in case anything unexpected happens so we don't have to rush at the end.

Ensure that all tasks are discussed in meetings so we can collaborate effectively in accomplishing these tasks.

Break down complex features into smaller tasks for better management.

What We Wish We Knew:

The importance of thorough planning and requirement analysis at the beginning of the project.

The significance of regular communication and collaboration to avoid misunderstandings and conflicts.

The material this class covered in the last few weeks of the semester.

Learnings:

Gain insights into software development methodologies and best practices.

Enhance knowledge and skills in object-oriented analysis and design principles.

Understand the importance of teamwork, communication, and adaptability in project management.

7. Summary

In summary, the "Kitchen Organizer" project aimed to address the challenge of kitchen inventory management and food waste reduction. Through collaboration, planning, and iterative development, we successfully implemented a functional and user-friendly application. The project provided valuable learning experiences in software development and project management, contributing to our growth and understanding of the subject matter.