**COSC 3011: Software Design Course Project (Spring 2025)**

Recipe Book

By: Noah Canen, Olivia Brown, Ashlyn Erickson, Megan Rose, Nathanael Connell, Adam Pannell

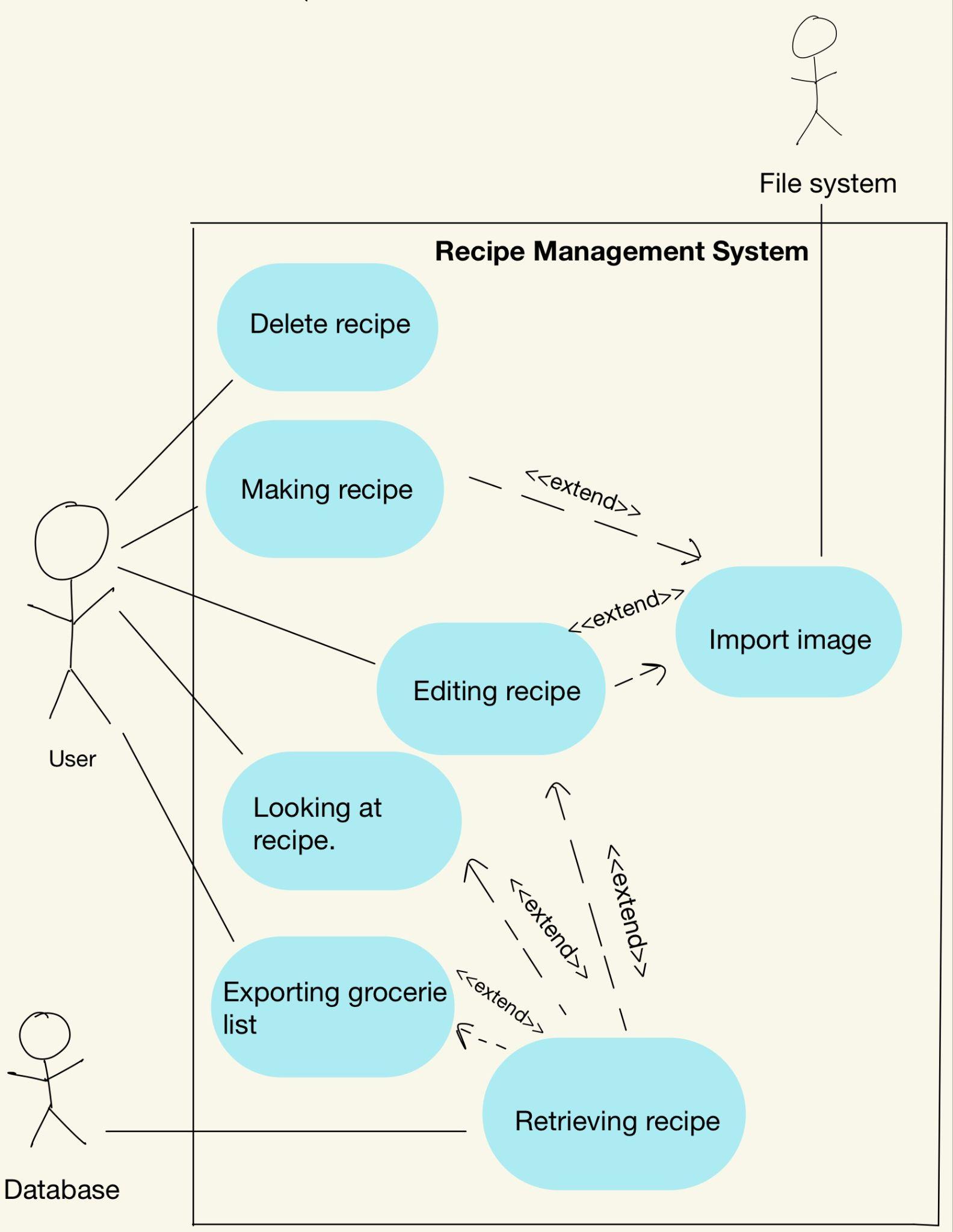
1. **Introduction**

Many people have experienced the frustration of losing a cherished recipe, whether it's a family tradition passed down through generations or a personal creation perfected over time. Recipe Book solves this problem by providing a simple way to save and view your recipes right from your home computer. With a user-friendly application, you can easily add a recipe title, description, image, ingredients, and instructions–keeping everything you need in one place. Whether you're cooking today or years from now, your recipes will be accessible. Recipe Book also includes another feature, exporting ingredients directly into a grocery list. This allows you to not worry about forgetting a key ingredient. Made for ease and convenience, Recipe Book is the perfect digital place to hold your recipes.

1. **Product Specification**

With this digital Recipe Book you can choose to create or look at the recipes you have already made. In the recipe template you have the option to add an image or not. You can add ingredients and their quantity with the click of a button and if you make a mistake you can delete the ingredient by selecting it even after you have saved it. There is the option to export the ingredients to a grocery list so you can save it for later. There is also a button that lets you categorize the recipe for easier access later, you can choose an entree, appetizer, dessert, or drink. On the look at the recipe page you can find the list of the recipes you have already made to view them and you can click on the categories to narrow down the search.

* 1. **UML Diagram**



* 1. **UML Tables**

| **Use-case Description** | **Create Recipe** |
| --- | --- |
| Related Requirements | Importing Images |
| Goal in Context | Adding a new recipe with a possible photo, ingredients, and measurements |
| Preconditions | Click the button to add a new recipe, retrieve the database |
| Successful End Condition | A new recipe is created by the user |
| Failed End Condition | The recipe is not saved |
| Primary Actors | User |
| Secondary Actor | Recipe database |
| Trigger | The user asks the database to create a new recipe |
| Main Flow | 1)The user asks the database to create a new recipe  2) Write in the recipe manually in a basic template, there is a place for  ingredients and instructions  3) Possibly add photo  4) New recipe created  5) Recipe is saved when finished and added to the database |
| Extension | N/A |

| **Use-case Description** | **Delete Recipe** |
| --- | --- |
| Related Requirements | None |
| Goal in Context | The user might no longer want that recipe inside their recipe book |
| Preconditions | A recipe needs to be created |
| Successful End Condition | The recipe is no longer inside the storage system and is no longer  being displayed |
| Failed End Condition | A window saying “Failed to delete the recipe”. The recipe is not  deleted from the database and is still displayed |
| Primary Actors | User |
| Secondary Actor | Database |
| Trigger | Pressing the delete button and confirming that you want to delete the  recipe |
| Main Flow | 1. The user presses the delete button 2. The user confirms that the recipe should be deleted 3. The recipe is deleted from the database 4. The list of recipes is redisplayed, missing the deleted recipe |
| Extension | 3.1 The database fails to delete the recipe  3.2 Display a window saying failed to delete the recipe  3.3 The recipe is not deleted from the database and is still displayed |

| **Use-case Description** | **Retrieve Recipe** |
| --- | --- |
| Related Requirements | Edit recipe, Export Grocery List, Look at recipe |
| Goal in Context | Allows access to data to look at a recipe and gather the information |
| Preconditions | When we query for the data, there should be a recipe there that we are  looking for. So, create and edit use cases should have been completed |
| Successful End Condition | If the query brings back a recipe, this is a successful pull |
| Failed end Condition | If there is no recipe, this is a failure |
| Primary Actors | User |
| Secondary Actor | Database |
| Trigger | User queries for a recipe |
| Main Flow | 1. User queries for information (recipe) 2. Database returns recipe |
| Extension | 2.1 If the recipe is not present, return an error of the recipe not being  found |

| **Use-case Description** | **Edit Recipe** |
| --- | --- |
| Related Requirements | Import Images |
| Goal in Context | A user will be able to edit an existing recipe in the database |
| Preconditions | A user needs to retrieve a recipe before they can edit it |
| Successful End Condition | The retrieved recipe is edited to the user’s liking |
| Failed end Condition | If there is no recipe to edit, an error message will be shown stating  that the recipe cannot be found |
| Primary Actors | The user |
| Secondary Actor | The database |
| Trigger | The edit button is clicked for the recipe to be edited |
| Main Flow | 1. The user retrieves a recipe  2. The user presses the “edit” button  3. The user edits the recipe to their liking  4. The user presses the “save” button  5. The recipe has successfully been edited |
| Extension | 1.1 The user is unable to retrieve a recipe because it doesn’t exist  1.2 An error message is displayed  4.1 The user fails to press the “save” button  4.2 A message is displayed, asking the user if they want to save the  changes they made |

1. **Technical Specification**

Recipe Book was created using Java, Java Swing, SQLite, and Maven. The graphical user interface (GUI) was built using Java Swing, and the backend of the project was developed using a SQLite database that holds recipes created by the user and Maven, the build and dependency management tool used for managing the project dependencies.

The application can be divided into three main layers:

1. User Interface (UI)

* Built using Java Swing
* Provides various frames for user interaction, such as frames for recipe creation and recipe viewing
* Handles button-click events

1. Core Functionality

* Implemented in the Recipe and RecipeRepo classes
* The Recipe class provides functionality to individual recipes, such as recipe type, title, description, ingredients, instructions, and an optional image
* The RecipeRepo class is used for database management

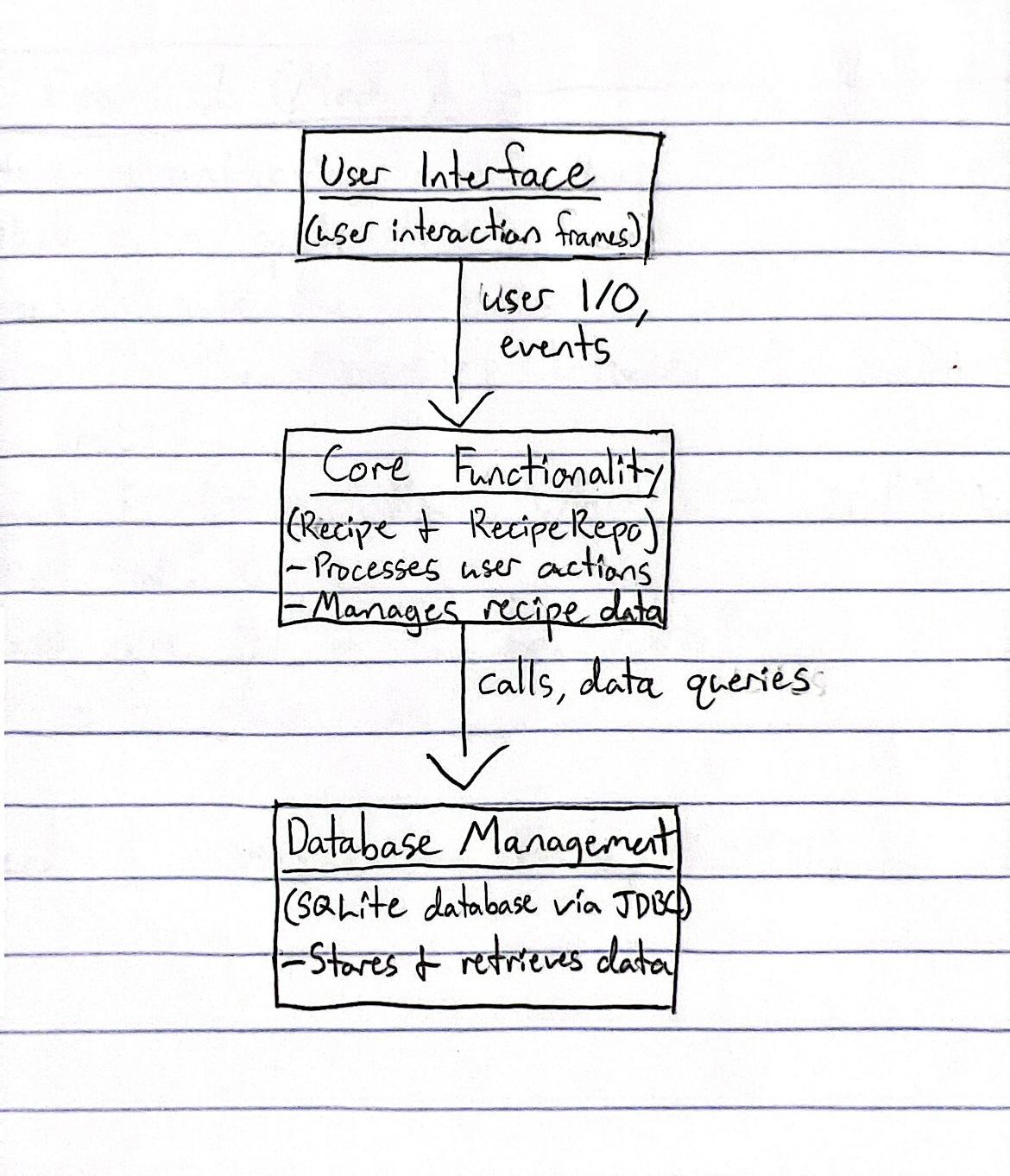
1. Database Management

* Uses SQLite and the Java Database Connectivity (JDBC) driver to store and query recipe data sent to the database

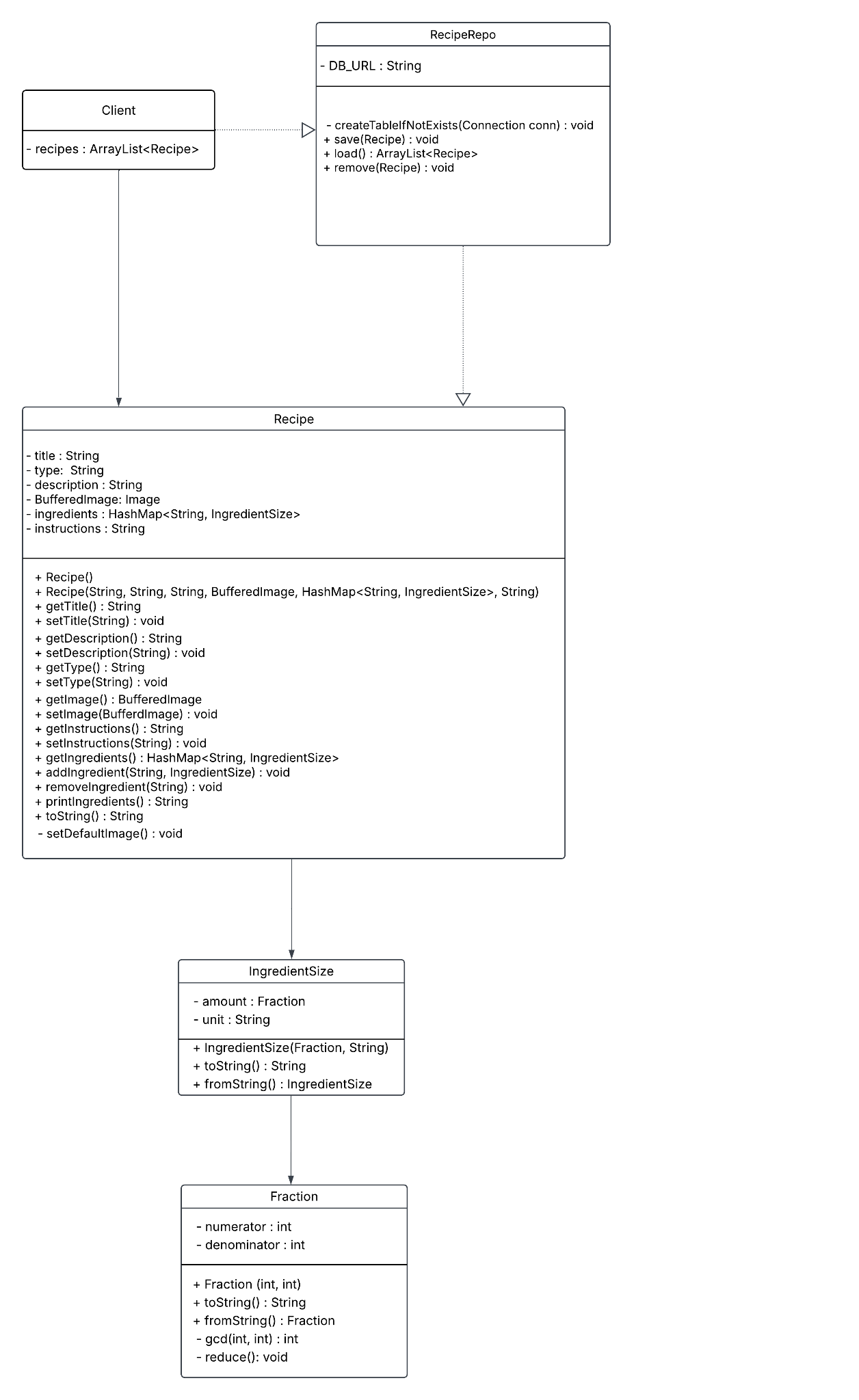
Data flow example:

When a user creates a recipe:

1. The Java Swing GUI collects input from the user
2. A Recipe object is created
3. The RecipeRepo class saves this object to the SQLite database
4. When viewing, editing, or filtering recipes, the application queries the database and sends results back to the Recipe object, which are displayed in the GUI.

High-level block diagram:

* 1. **Class Diagram**



1. **Technology and Tools**

SQLite - Our application loads from and stores Recipes in an SQLite database. This allows users to save recipes even if the application is stopped. This is handled in the RecipeRepo class

Maven - We used Maven as a dependency management tool. This is also a tool to help us access the database inside of the application with SQLite JDBC. It also helps us to standardize the program across different environments.

1. **Our Application**

A screenshot of a computer

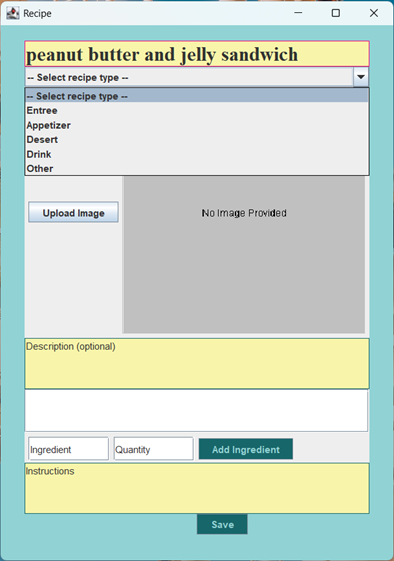
AI-generated content may be incorrect.

When you open the recipe book, the main panel will appear. With two buttons, look for recipes and create a recipe if you have not made a recipe in the past. The Look for recipes button will bring you to a page where you can usually access your previously created recipes. I haven't made a recipe yet, but I'll start by pressing 'Create Recipe'.

A screenshot of a computer

AI-generated content may be incorrect.

When you press the "Create Recipe" button, a panel called "Recipe" will open. For this example, I will create a recipe for a peanut butter and jelly sandwich. I started by entering the name of the food item into the "Title" text box.



Then I selected a recipe type. A recipe type is important because it helps you sort and find recipes later. A peanut butter jelly sandwich is an entree, so I put that as the recipe type.

A screenshot of a computer

AI-generated content may be incorrect.

Next, we add an image, which is optional but highly encouraged. We currently only accept JPEG, PNG, or GIF files. So, we'll simply select a picture with one of those file types.

A screenshot of a computer

AI-generated content may be incorrect.

Simply navigate to the File menu, select Downloads, and then select the downloaded picture.

A screenshot of a computer

AI-generated content may be incorrect.

Then the picture will appear in the gray square next to the upload image button.

A screenshot of a food recipe

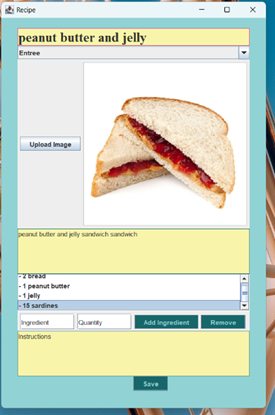
AI-generated content may be incorrect.

Next is the optional description. This is where you can add a description of the food or include some personal notes about it.

A screenshot of a recipe

AI-generated content may be incorrect.

Next are the ingredients. There are two text boxes for ingredients. Enter the name of the ingredient in the Ingredient section and the quantity in the Quantity section, then click the "Add Ingredient" button.



Then it will appear sequentially, in the order it was entered, above the ingredients and quantity text boxes. And in case you accidentally enter the wrong ingredient. Simply select the ingredient and press the delete button that will appear.

A screenshot of a food menu

AI-generated content may be incorrect.

Last, for the recipe template in the instructions, enter the instructions on how to construct your food.

Then, when you're satisfied with your recipe, press Save.

A screenshot of a recipe

AI-generated content may be incorrect.

When you press Save, the recipe will be stored in the database, allowing you to access it later. A window called 'Saved Recipe' will appear. From this page, you can do three things: export grocery list, edit recipe, and delete recipe.

Export grocery list will export the ingredients of the current recipe into an ingredient list so that you can print it out and go shopping for the ingredients needed to make that recipe.

The Edit recipe option will reopen the recipe page with all the information for the recipe already entered, so you can edit if you made a mistake.

Delete the recipe will delete the recipe from the database.

A screenshot of a computer

AI-generated content may be incorrect.

Now that we have a recipe, we can click the' Look for recipes' button on the main page. This will open the recipes page. Here, you can open and view saved recipes, and also sort by recipe type to narrow down your search.

1. **Discussion**

We wanted to add the capability to add an image of any type or at least have the

capabilities to handle a couple file types. We have the ability to include png files but that is the only type at the moment.

Another function we wanted to add was a more in depth system of ingredients to export into a grocery list and have a more interactive system but we ended up just doing an analog list with just the ingredients printed on in a column that can be manually checked off.

We wanted to add drop down menus for ingredient measurements of certain values such as oz or cups. We decided to just keep it to have the user input the value and measurements themselves.

1. **Conclusion**

I (Megan Rose) feel I learned a lot with the database. I had previously not had experience using SQLite, however, being able to learn more on SQL commands and the datatypes SQLite has to offer was definitely interesting. Using SQLite, the application was able to store information in a database saved on the user’s computer, which I feel I may need for a future project like Senior Design.

A lot was learned about UI and how Java Swing works. The group had little to no experience with Java Swing, and after working on this project we all know significantly more about it.

**Appendix - Division of Work among Team Members**

* Olivia Brown: My focus in the project were front end aspects. I created most of the UI buttons and implemented their functionality, I developed the main Export Grocery List functionality, and I helped with front end debugging whenever necessary. I also wrote up and sent the progress reports each week.
* Noah Canen: My focus on this project was the backend of the recipe template. I worked on creating the text boxes for the title, description, and ingredients. I set up the save button so that it would gather all the information from the various text boxes and hash maps and put them into the proper functions to be saved.
* Nathanael Connell: I was primarily front end through the project and I worked mostly on the main UI and the look at recipe UI. I helped create an outline of the UI in general and formatted the look at recipe UI and part of its functionality. I looked at the code periodically to make sure things made sense and will do final checks before it's fully presented.
* Ashlyn Erickson: I worked on the UI and mostly focused on the Create recipe template. I created an editable template with the title, description, ingredients/add ingredient button, and instructions and made it look aesthetically pleasing with colors. I also made the not editable view of a recipe after it was saved. I worked on incorporating it into the other UI linking the create recipe template to the create recipe button on the main page.
* Adam Pannell: I was in charge of the logic behind creating recipes. I set up the Recipe class and I made sure that the user interface would actually make recipes that could then be stored in a database. I made sure that recipes could be created, edited, and deleted all through the UI. Finally, I helped Megan wrap up her work on the database by straightening out our project dependencies for the SQLite Java Database Connectivity.
* Megan Rose: I was in charge of the database, which I worked on saving and querying recipe data to/from the database. This is written using SQLite and Java. My original solution relied on Maven for managing dependencies, and worked locally on my machine. However, when trying to transfer this to Github, I had issues. Adam helped this issue, by downloading the needed dependency into our project so I could be back on track with editing the database.