Final Project - Recipe Finder

Group 3 -

Aaron Liauw, U26704173

Jane Libby, U12953604

Kayla Terzioğlu, U41854837

Afnan Alobaid, U31148210

Esther Attar, U58153143

Boston University

Department of Computer Science

CAS CS 411 - Software Engineering

Professor Ahmed Ibrahim

June 27th, 2022

**Problem Definition:**

The Recipe Finder is a Java Application that allows its users to search through a database of recipes by ingredients. Recipe Finder is designed to make the lives of home cooks easier by optimizing the process of finding a recipe. Whether the home cook is a college student looking to make an easy meal, or a full time chef who doesn't want to waste any remaining food in their fridge, Recipe Finder allows them to use up the ingredients they have in their home without needing to go buy more. This will save the user time, money, and will help reduce their daily food waste.

**Project Objective:**

* Offer a database of recipes that users can access.
* Allow users to input the ingredients they have.
* Output recipes with the users’ ingredient specifications.

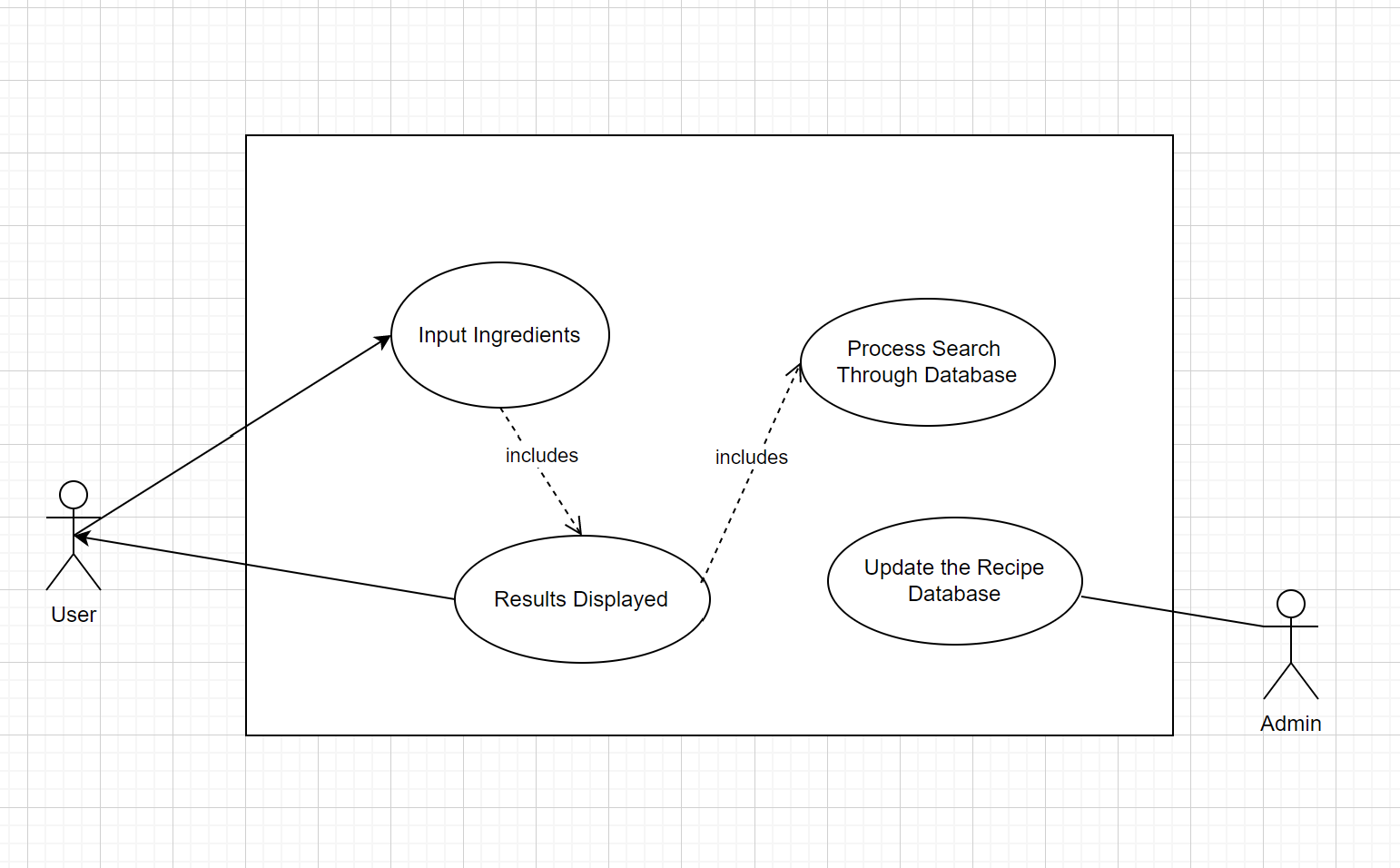
**Stakeholders List:**

* External Stakeholders: Home cooks
* Internal Stakeholders: Development Team (Software Engineers, Designers, Writers)

**Success / Acceptance Criteria for each Stakeholder:**

* Home cooks:
  + Acceptance Criteria: Recipes can be displayed, and recipes can be retrieved from the database.
* Development Team:
  + Success Criteria: The MVP is completed by the deadline of June 27th, and the project implementation works.
  + Acceptance Criteria: The project gets a good grade.

**Use Case Diagram:**



**Fig. 1. Use Case Diagram**

This is the Use Case Diagram for the Recipe Finder System. The user can input up to 3 ingredients, and have the resulting recipes displayed. The displayed results will either be the recipes, or an error message if there is no recipe with the specified ingredients. In order for the results to be displayed, they must be processed by searching through the recipe database. An admin is able to update the recipe database with new recipes.

**Detailed Use Case Descriptions:**

|  |
| --- |
| **Use Case Name: Input Ingredients** |
| **Actors:** User |
| **Description:** The user will input their desired ingredients. |
| **Stakeholders:** The User, who wants to find a recipe. |
| **Triggering Event:** Entering ingredients when prompted |
| **Steps Performed:**   1. The User types a list of ingredients into the database. 2. The User hits enter. 3. The User’s input is processed and rendered into a format that the system can use to search through the database. |
| **Preconditions:**   1. The user opens the terminal. 2. There is a prompt for users to input ingredients. |
| **Postconditions:**  The system has a list of ingredients to search with and they have been processed by the database. |

**Fig. 2. Input Ingredients Description**

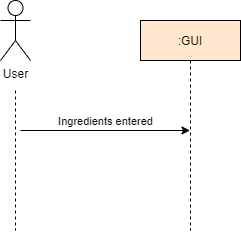
This details the process of the Use Case: Input Ingredients, in which the system receives and processes user input, ultimately rendering it into a format which can be used to search the database.

|  |
| --- |
| **Use Case Name: Results Displayed** |
| **Actors:** User |
| **Description:** If the database contains a recipe with the ingredients that the user input, the application will display the recipe. If no recipe was found, the application will output a message “No results found.” If the User inputs nothing, then the application will output nothing. |
| **Stakeholders:** The User that will see the results displayed. |
| **Triggering Event:** Searching the database for recipes with the entered ingredients. |
| **Steps Performed:**   1. The search results from the database are processed into a format that can be displayed to the user. 2. The application displays the search results. |
| **Preconditions:**   1. The system has retrieved results from the database. |
| **Postconditions:**   1. A result is displayed on the screen (either a recipe, or the message “No results found”). |

**Fig. 3. Results Displayed Description**

This details the Use Case: Results Displayed, and the process by which the system takes the results from the database search and displays them for the user.

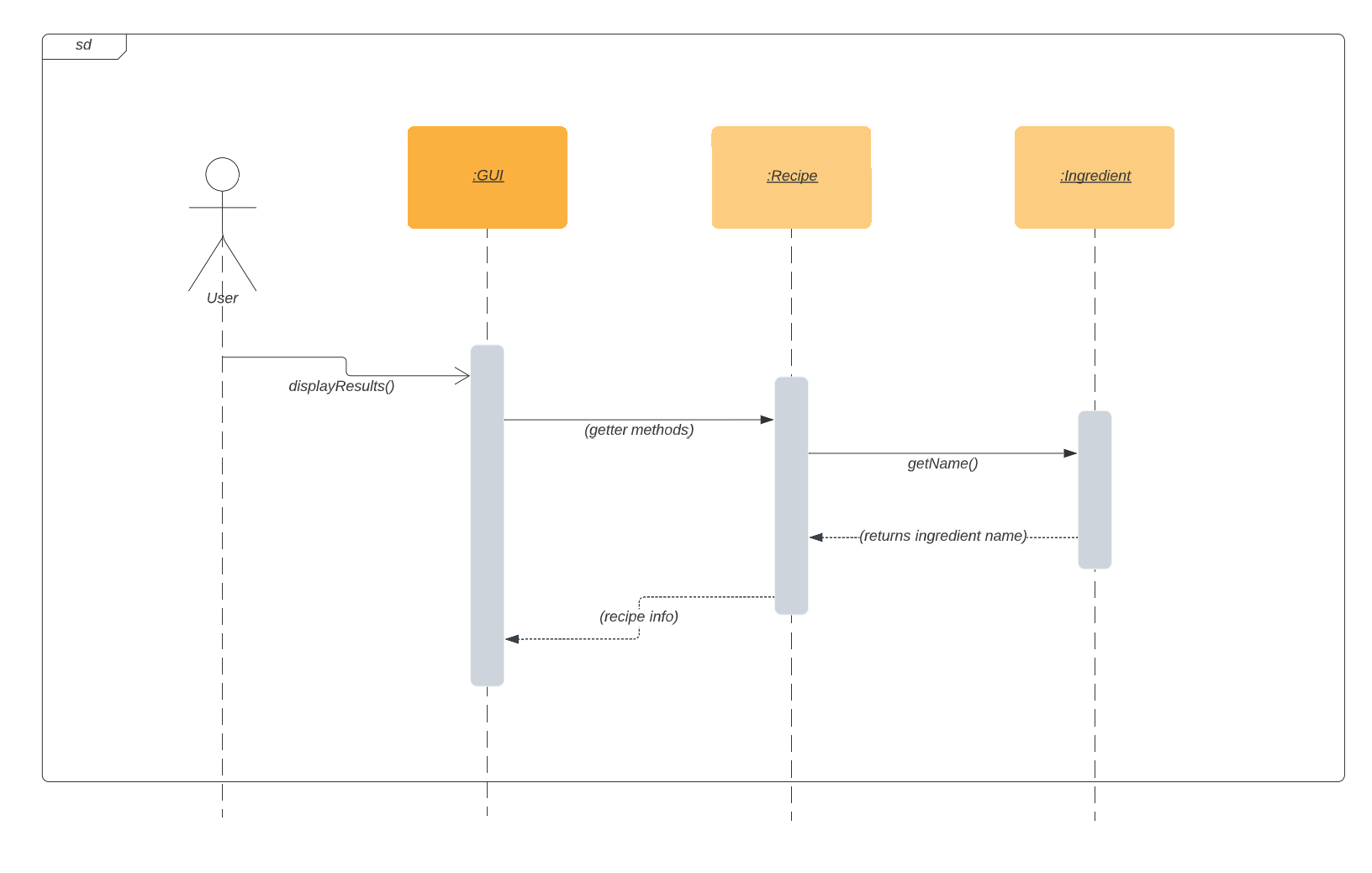
**Input Ingredients Use Case Sequence Diagram:**

****

**Fig. 4. Input Ingredients Sequence Diagram**

This is the Input Ingredients Sequence Diagram, which details the interactions between the classes of the system in the “Input Ingredients” use case. The simplicity of the diagram is a result of the fact that the bulk of the work is done internally by the GUI class.

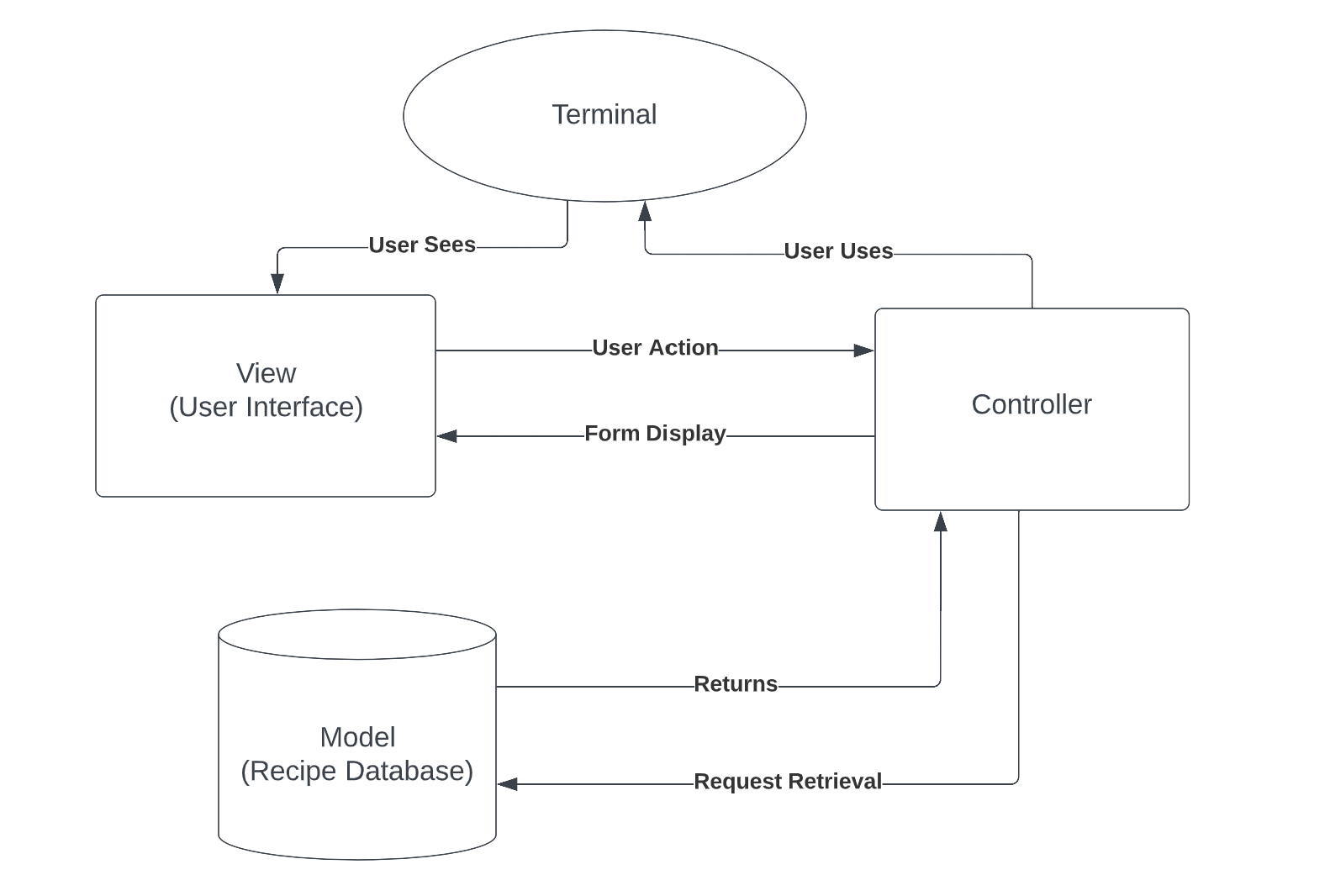
**Results Displayed Use Case Sequence Diagram:**

****

**Fig. 5. Results Displayed Sequence Diagram**

This diagram details the interaction between the classes of the system in the Results Displayed Use Case. The GUI class receives the information from the Recipe and Ingredient classes, formats that information, and then displays it for the user.

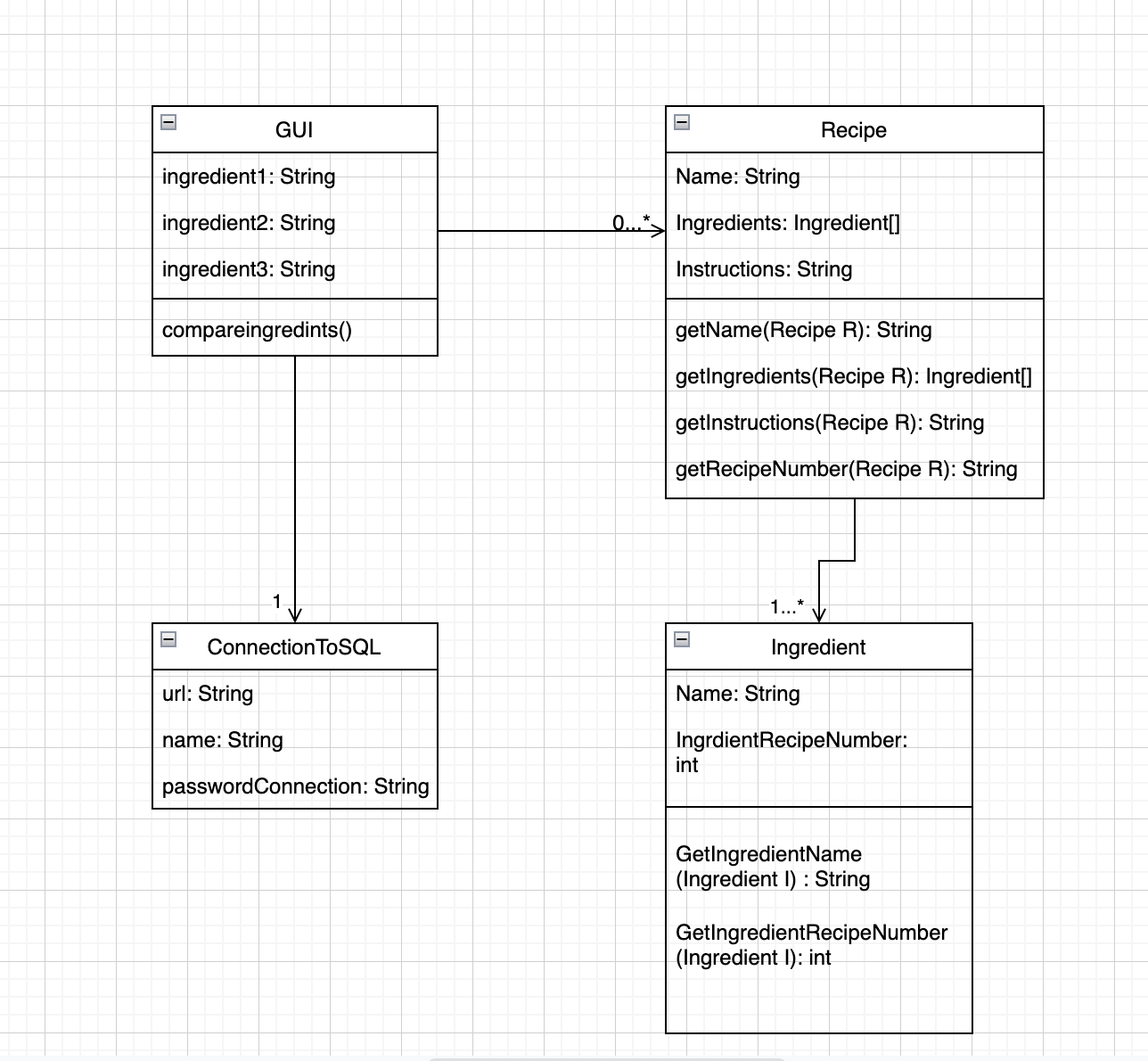
**System Architecture:**



**Fig. 6. Software Architecture**

This is the system architecture of Recipe Finder. The Model-View-Controller pattern is utilized. As a Java application, users can interact with the interface, while a controller mediates between view (user interface) and mode (database).

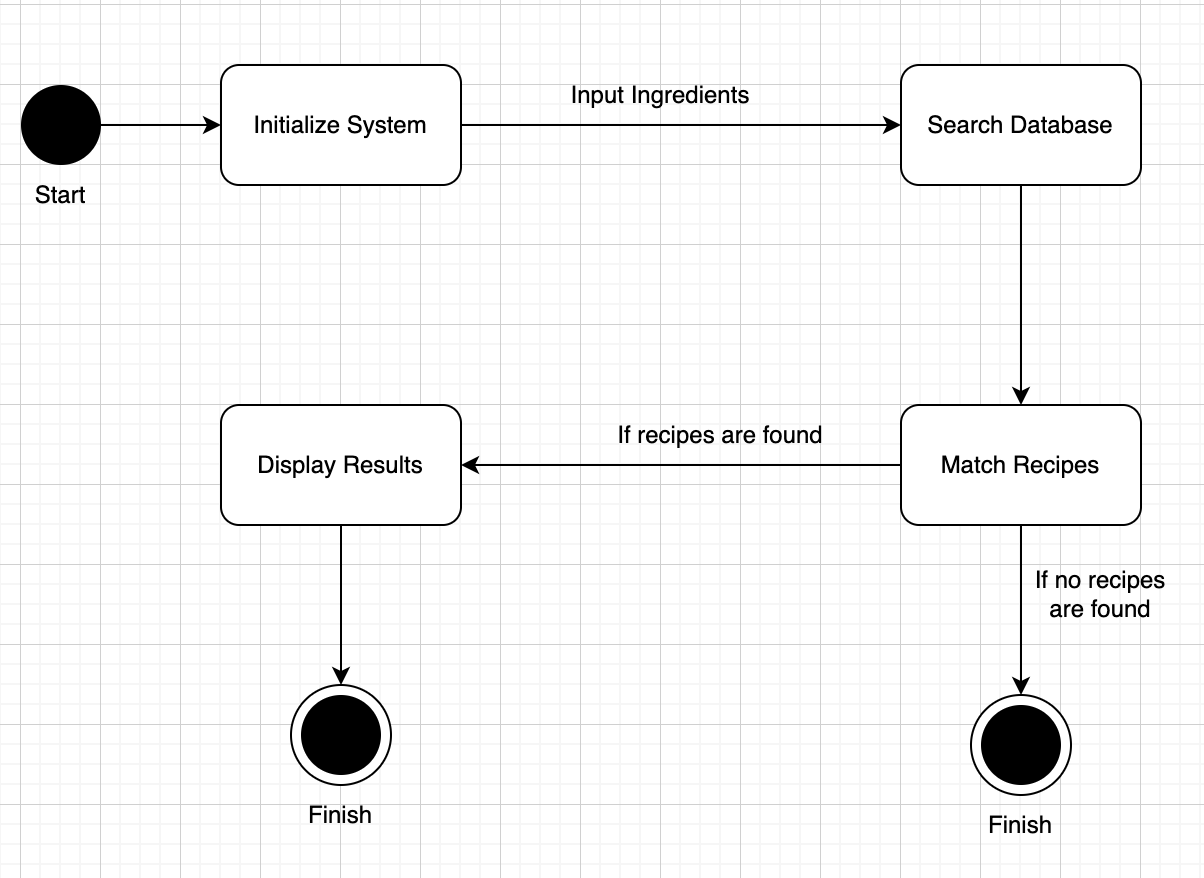
**Class Diagram:**

****

**Fig. 7. Class Diagram**

This Class Diagram demonstrates the classes utilized by the system and their relationships. The core functions of the system are contained within the GUI class. The ConnectionToSQL class serves to let the GUI access the database, and the Recipe and Ingredient classes primarily serve to wrap the information retrieved from the database.

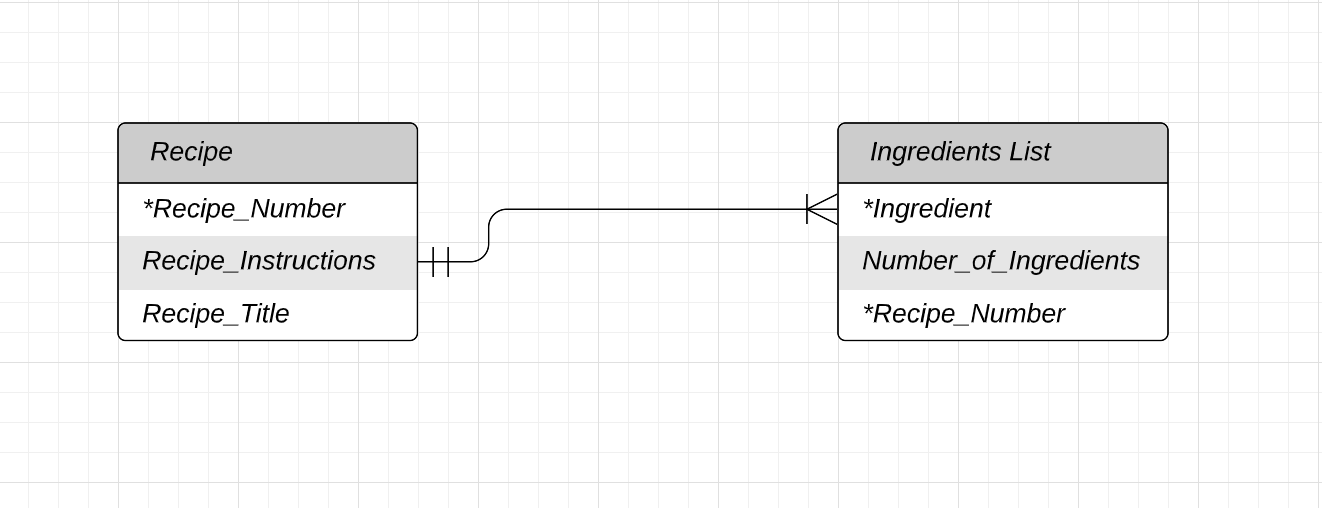
**State Machine Diagram:**



**Fig. 8. State-Machine Diagram**

This State Machine Diagram shows how the system works as a whole. It demonstrates what happens every step of the way, since initiating the system at the start. In this diagram, we can see after the system is initialized, the action of inputting ingredients triggers the search through the database. After the search, there are two possible outcomes, either no recipes are found, or the resulting recipes are displayed.

**ER Diagram:**

****

**Fig 9. ER Diagram**

This is an ER Diagram depicting the many to one relationship between Ingredients List and Recipe, where each instance of Ingredients List has a relationship with one Recipe, and each instance of Recipe has a relationship with at least one Ingredient List.

**Project Source Code GitHub Link:** https://github.com/kaylaterzioglu/Recipe-Finder

**Conclusion:**

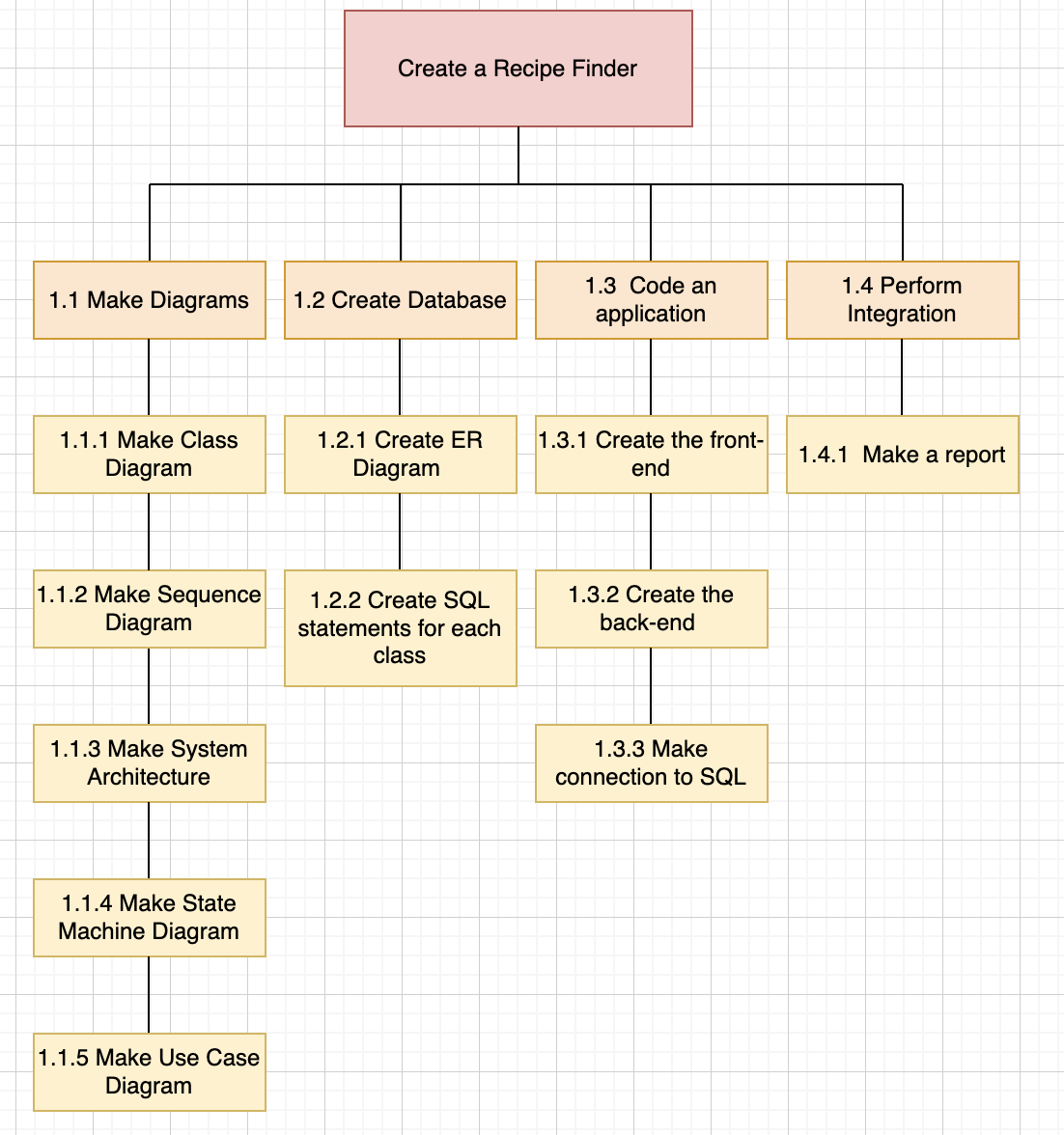
This was a valuable experience in learning how to properly plan a software project. Most of our team had little experience approaching software development in such an organized and structured fashion, which meant that the planning process was rather chaotic, yet fluid, as we were all still learning how to approach the project in an orderly way. However, the effort we put into the planning phase certainly paid off when we finally moved into the coding phase, as it gave us a much clearer understanding of what we were trying to build.

Additionally, we learned important lessons about how to plan a software development project, though many of those lessons came in the form of mistakes we made. These mistakes taught us that future projects would benefit more from a clearer, more ordered approach, as well as a willingness to engage with and understand the software we’d be using. We were forced to make frequent revisions to our planning documents and diagrams, in part because we had a somewhat unclear understanding of how to incorporate the database elements of our software. While this obviously presented us with some with this current project, this experience will allow us to avoid such pitfalls in any future projects that may come about, whether in school or in our careers.

**References:** There are no references to declare.

**Appendix:**

**Work Breakdown Structure:**



**Fig 10. Work Break Down Structure**

This diagram depicts the Work Break Down Structure of our Recipe Finder project. We started by making the necessary diagrams as we planned how this application would come about. Then we moved on to creating the database, and then coding the application and connecting it to the SQL database. Lastly we performed the integration of this application and created our Final Design Report.

**Task Assignment Matrix:**

|  |  |  |
| --- | --- | --- |
| **Task:** | **Task Owner:** | **Support:** |
| Problem Definition | Kayla |  |
| Problem Objective | Aaron |  |
| Stakeholders List | Kayla | Aaron |
| Success / Acceptance Criteria for each Stakeholder | Aaron | Jane |
| Use Case Diagram | Aaron | Jane |
| Detailed Use Case Description #1 | Esther | Afnan, Kayla, Jane, Aaron |
| Detailed Use Case Description #2 | Jane | Afnan, Kayla, Aaron, Esther |
| Sequence Diagram #1 | Jane |  |
| Sequence Diagram #2 | Aaron |  |
| Architecture | Aaron |  |
| Class Diagram | Jane |  |
| State Machine Diagram | Esther |  |
| ER Diagram | Kayla |  |
| Task Assignment Matrix | Kayla |  |
| Application Code | Afnan |  |
| SQL Code | Kayla | Afnan |
| Work Breakdown Structure | Esther |  |
| Conclusion | Jane |  |

**Fig 11. Task Assignment Matrix**

This table is the Task Assignment Matrix, depicting which team members worked on what parts of this project.

**Sample of commits on the selected version control system:**

