1. Delegation of Tasks

Dan Nguyen : Github, Software Requirements, Architectural Design, Unit Testing

Cade Edwards : Case Diagram, Uploading Files, Concept Execution, Slideshow

Luke Buchanan : Sequence Diagram, Cost Estimations

Matthew Singsing : Use Case Diagram, Project Scheduling, Cost, Effort, Project Timeline

Rayven Gsell : Final Project Draft Description. , Software requirements, Research

Shoham Sanyal : Sequence Diagram, Software Process Model, Slideshow, Document Creation

2. Project Deliverable 1 Content

Project Scope

1. Tastey - A Recipe Library Software
    - 1.1. Recipe Management
        - 1.1.1. Load saved recipes
        - 1.1.2. Delete saved recipes
        - 1.1.3. Add your own recipes
        - 1.1.4. Search recipes by category
    - 1.2. Recipe Reading
        - 1.2.1. Ability to choose between tabs; ingredients, steps, stories, and photos
        - 1.2.2. Scroll to go between steps
        - 1.2.3. Adjust portion sizes
        - 1.2.4. Offer recipes/items that pair well with this one (Optional)
    - 1.3. Aesthetics/Accessibility
        - 1.3.1. Day and Night Mode (Optional)
        - 1.3.2. Change font and size of text (Optional)
        - 1.3.3. Adjust theme/color of the app (Optional)
        - 1.3.4. Choose allergies/dietary restrictions

Software Process Model Choice

The software process model that is employed in our project is the waterfall model. This is because we felt that a more standard approach would be adequate for the project that we had in mind. From the beginning, we laid clear and well-thought-out ideas for what we wanted to achieve with our project. As a result, no iterations were necessary. Additionally, we created diagrams and documents that clearly state the specifications of our software.

Other software models that we ended up not using were the incremental process model, the prototyping model, and the spiral model. We didn't like the incremental process model because we felt that there were not enough parallel process flows that we could work on. Additionally, our time frame was rather short so only a few "increments" could be completed. The prototyping model wasn't an option due to the fact that we were not at the stage of implementing our project. Therefore, there was no need to create multiple prototypes. Additionally, in order to build a working prototype sacrifices must be made and we as a team decided that we would not take those sacrifices to ensure that we delivered a premium product. The final software process model which we ruled out was the spiral model. The main reason for this is that we found it difficult to identify the framework activities that would represent the spiral path. Additionally, since no implementation was being done yet the spiral model seemed unnecessary for what we needed.

Software Requirements

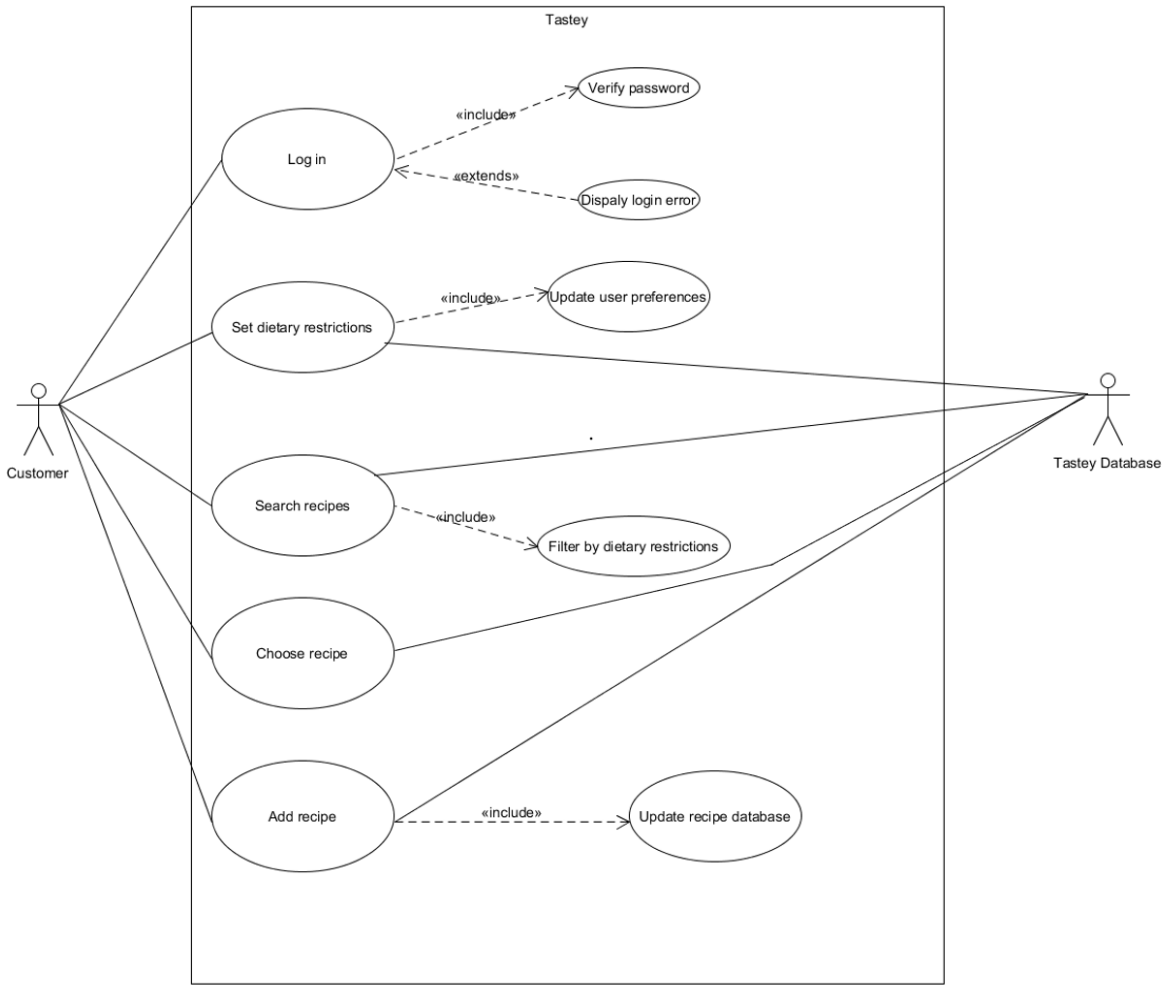
Functional Requirements

1. User should be able to save recipes for future view
2. User should be able to search for recipes based on keywords
3. User should be able to edit the recipe and save it to the database
4. The database should be able to accept imported recipes from other databases
5. User should be able to request the database to gather more recipes from the internet

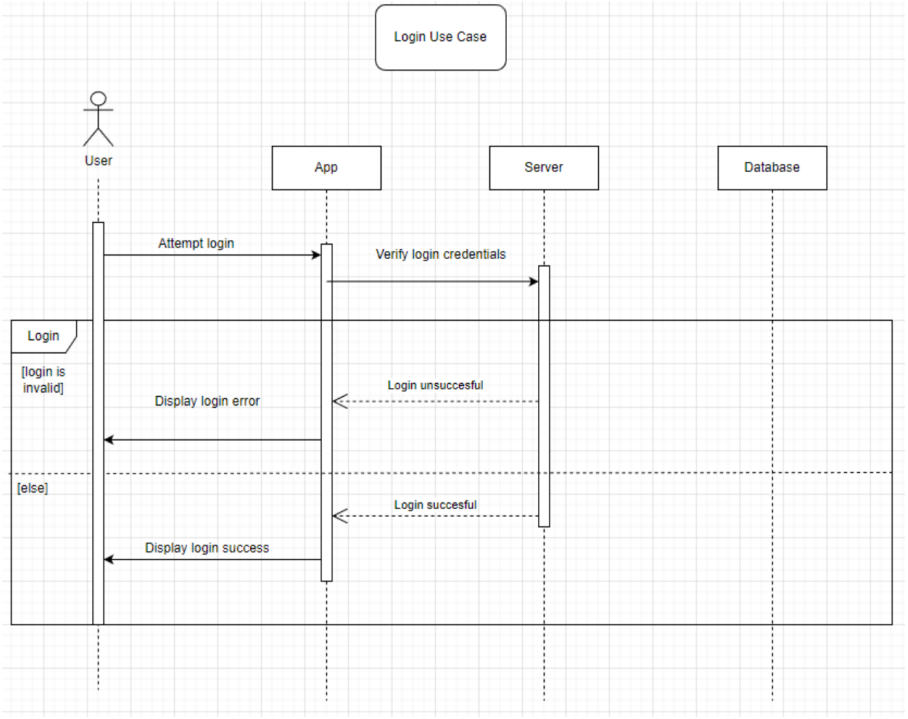
Non-Functional Requirements

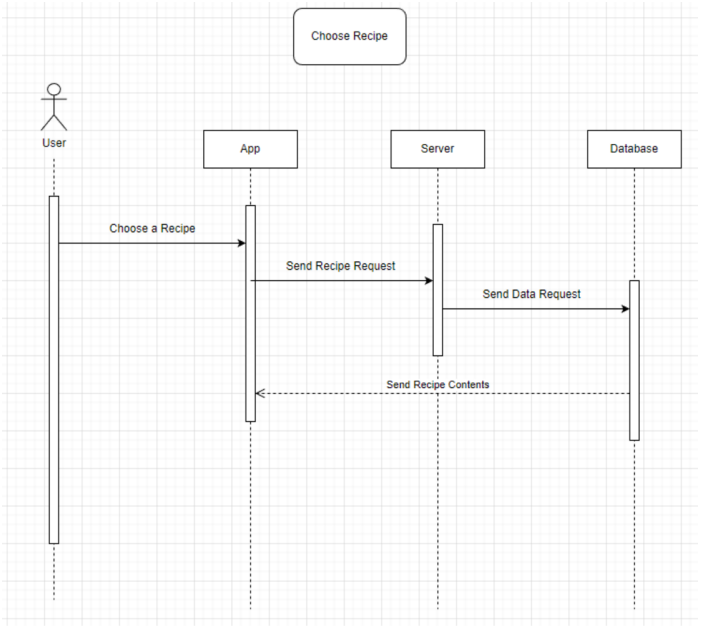
```
Usability:       A system to show recipes which include ingredients list, steps to cook, and final product picture.
Performance:     The database should be able to store at least 100 recipes.
Space:           The software will take at most 50 MB.
Environmental:   Recipe should not contain exotic ingredients and should ask user to not waste any ingredients.
Operational:     The software needs to work offline and be available 24/7.
                 User should be able to login to their account to access their recipes.
Development:     The software should allow system admins to add, remove, and manage user accounts.
Accounting:      The software will be free and open source.
Safety/Security: The software must have a authenication process to prevent unauthorized access and encryption to protect user data.
                 The software must have user access control to prevent any unauthorized function use. The software must comply with laws and regulations.
```

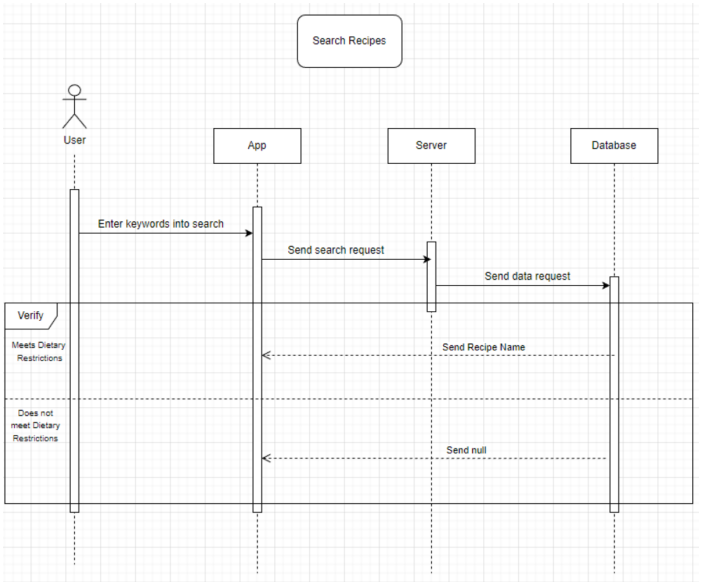
Use Case Diagram

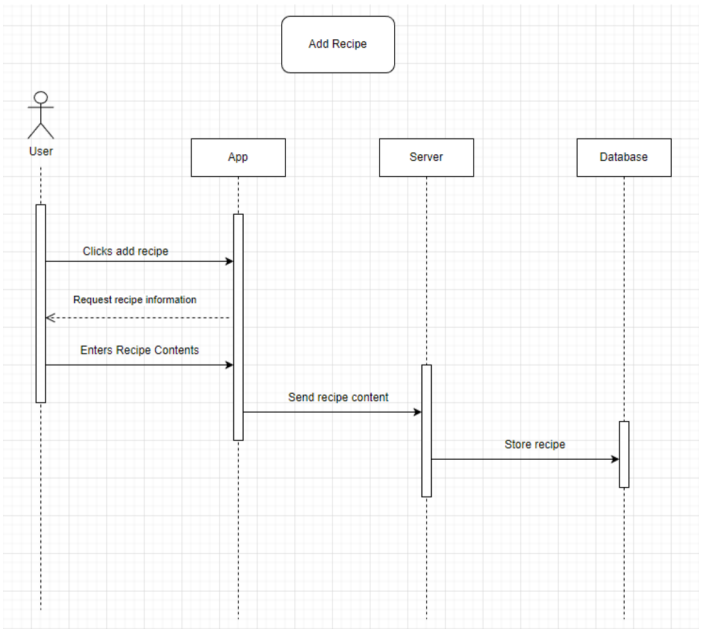


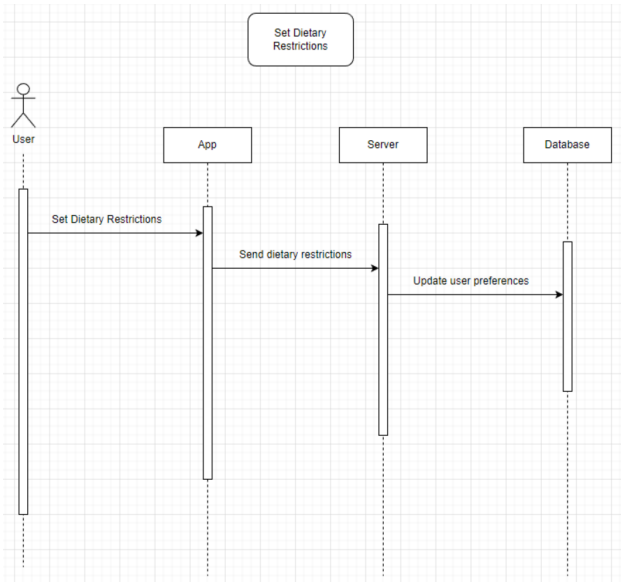
Sequence Diagrams



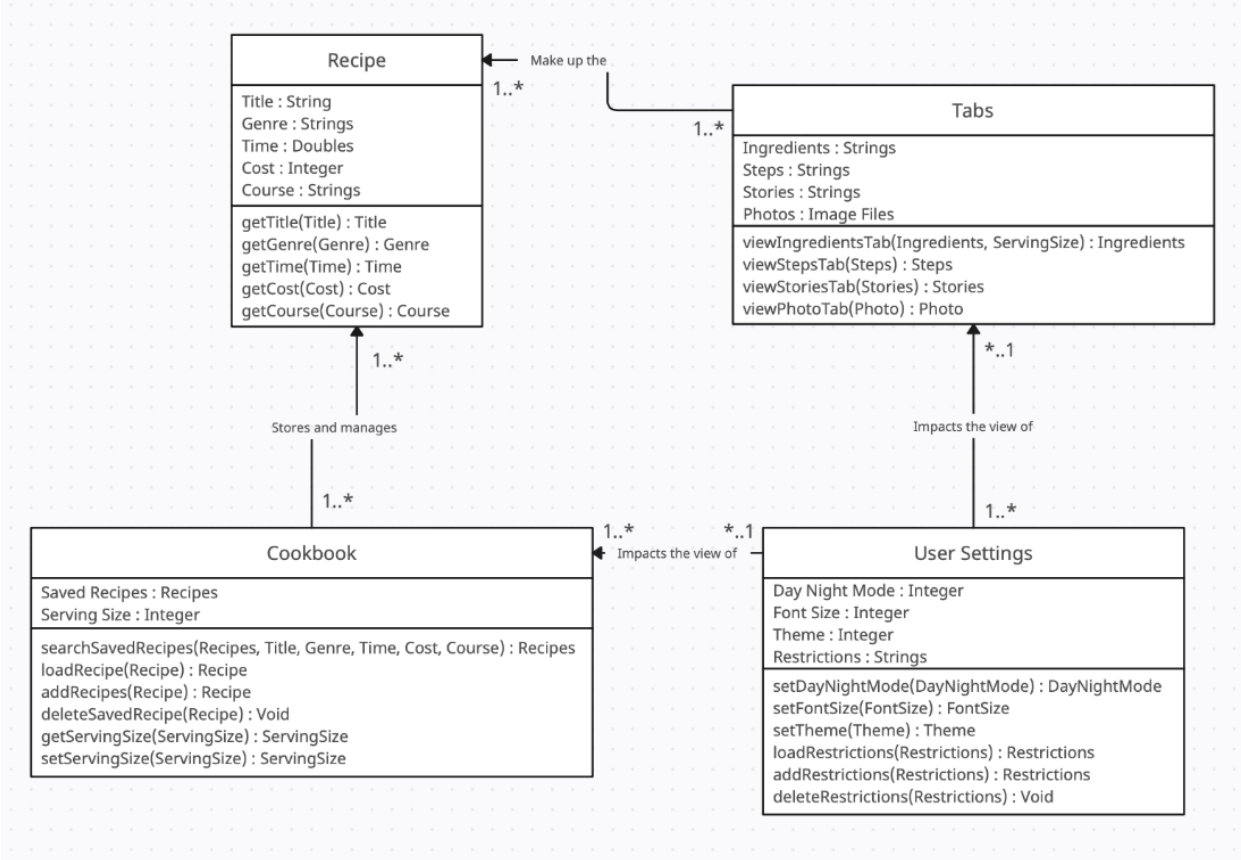



Search Recipes
User
App
Server
Database
Enter keywords into search
Send search request
Send data request
Verify
Meets Dietary Restrictions
Send Recipe Name
Does not meet Dietary Restrictions
Send null

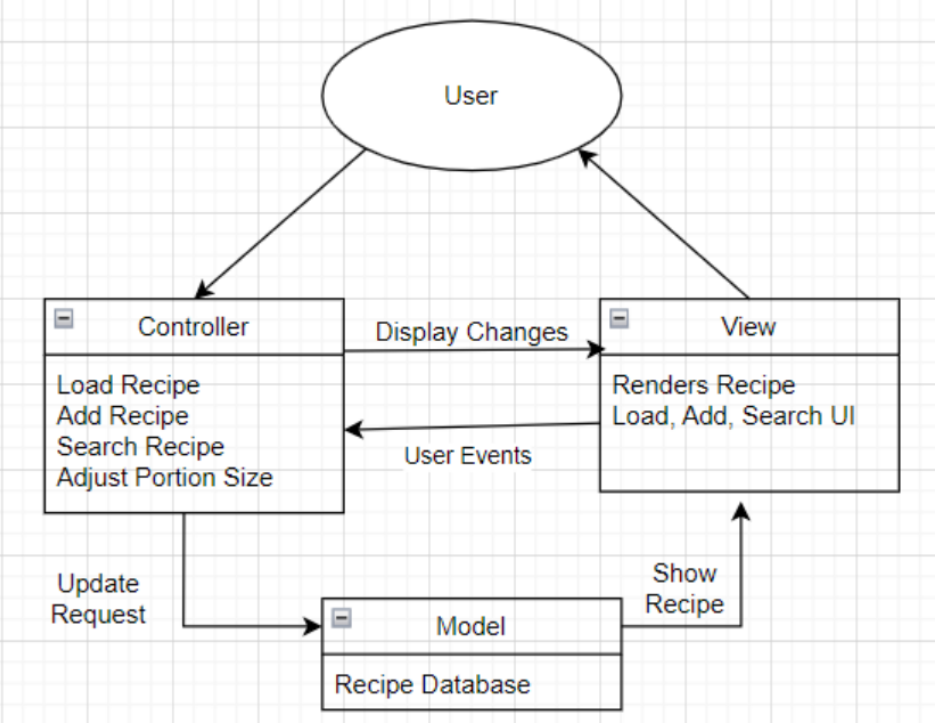

Add Recipe
User
App
Server
Database
Clicks add recipe
Request recipe information
Enters Recipe Contents
Send recipe content
Store recipe


Set Dietary Restrictions
User
App
Server
Database
Set Dietary Restrictions
Send dietary restrictions
Update user preferences

Class Diagram



Architecture - MVC



3. Project Scheduling, Cost, Effort and Pricing Estimation, Project Duration and Staffing

1) Project Scheduling

For our project schedule first we have taken into account our personnel size. Since this is a passion project, we accounted for there only being 6 members of the team who will be working on this project during the week, not including weekends, for 3 hours a day. We have made an end date estimate of at minimum 1 month and maximum 4 months. We came to this conclusion based on the overall simplicity of this project as many of our functions and their parts were marked as simple to implement in terms of complexity and there are only a few marked average or complex. We have obtained our maximum range by scaling up our initial predictions 4x.

2) Cost, Effort and Pricing Estimation

	Function Category	Count	Simple	Avg	Complex	Count x Complexity
1	Number of User Input	**8**	**3**	4	6	**24**
2	Number of User Output	**8**	**4**	5	7	**33**
3	Number of User Queries	**1**	**3**	4	6	**3**
4	Number of data files and relational tables	**2**	**7**	10	15	**25**
5	Number of external interfaces	**0**	**5**	7	10	**0**
					GFP	**85**

(1) Does the system require reliable backup and recovery?
5
(2) Are data communications required?
3
(3) Are there distributed processing functions?
1
(4) Is performance critical?
1
(5) Will the system run in an existing, heavily utilized operational environment?
1
(6) Does the system require online data entry?
3
(7) Does the online data entry require the input transaction to be built over multiple screens or operations?
3
(8) Are the master files updated online?
4
(9) Are the inputs, outputs, files, or inquiries complex?
2
(10) Is the internal processing complex?
1
(11) Is the code designed to be reusable?
1
(12) Are conversion and installation included in the design?
1
(13) Is the system designed for multiple installations in different organizations?
1
(14) Is the application designed to facilitate change and ease of use by the user?
4

FP=GFP*PCA=85*0.96=81.6

3) **Estimated cost of hardware products**
   Considering the cost for hardware products, the only hardware components our application will need is a dedicated server to host the recipe information, in addition to any recipes users will want to upload to the application. Given the relatively simplistic nature of the application, the server will mostly be responsible for text data as well as visual data to account for recipes with a visual image of the dish.As such, we can expect server costs to be around $150 a month to store the necessary information, putting us at an annual cost of about $1800. This is subject to change, as our user population and the services of our app may evolve throughout Tastey's life cycle.
4) **Estimated cost of software product**
   As we will not be using any licensed software to write Tastey's core code, we estimate that we will have no software cost associated with development. The majority of the app's development costs will lie in hardware costs, as well as the salaries of the team and any additional maintenance required after the product has been released. However, if we wish to upload our application to certain mobile devices, like Apple or Google, then we will need to pay for a fee for them to host our app. Ranging from $25 to $100.
5) **Estimated cost of personnel**
   Our development team consists of 6 passionate members, and the maximum time working on development is estimated to be about 4 months. Excluding weekends and following this schedule, we can expect each member to work on the project for 3 hours every day of the week. Taking into consideration the workload of the project, which consists mostly of building the base code for users to search up and upload recipes, we have estimated an hourly salary of about $20/hour. Thus, over the 4-month development cycle, we can expect salary costs to be about $9600.

4. Test Plan

Below is the source code written to create Recipe Database with add and search feature. The code is written in Java. The code is written in Object-Oriented-Programming style. In which the recipe database is a class and the recipe is a class.

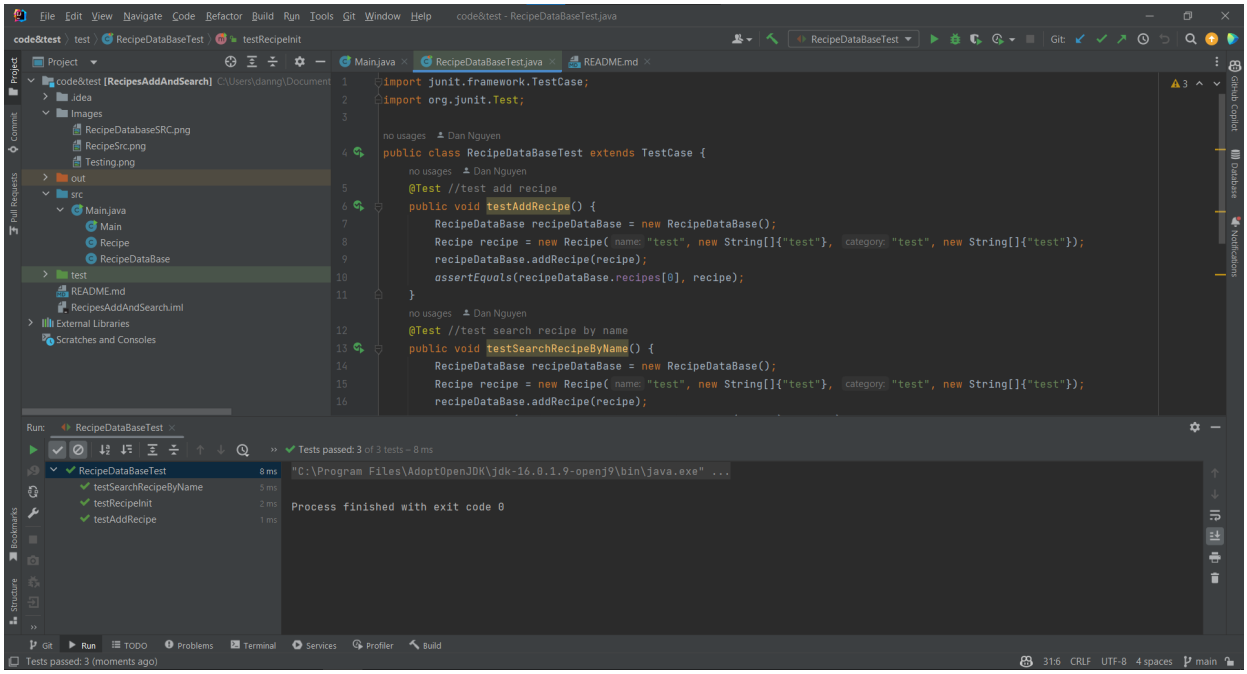
```java
class RecipeDataBase {
    8 usages
    Recipe[] recipes;
    no usages  Dan Nguyen
    public RecipeDataBase(Recipe[] recipes) { this.recipes = recipes; }
    //Init Empty DataBase
    3 usages  Dan Nguyen
    public RecipeDataBase() { this.recipes = new Recipe[0]; }
    //Search Reipe by name
    1 usage  Dan Nguyen
    public Recipe searchRecipeByName(String name) {
        for (Recipe recipe : recipes) {
            if (recipe.name.equals(name)) {
                return recipe;
            }
        }
        return null;
    }
    //Add Recipe to dataBase
    3 usages  Dan Nguyen
    public void addRecipe(Recipe recipe) {
        Recipe[] newRecipes = new Recipe[recipes.length + 1];
        for (int i = 0; i < recipes.length; i++) {
            newRecipes[i] = recipes[i];
        }
        newRecipes[newRecipes.length - 1] = recipe;
        recipes = newRecipes;
    }
}
```

```java
class Recipe {
    5 usages
    String name;
    4 usages
    String[] ingredients;
    4 usages
    String category;
    4 usages
    String[] instructions;
    2 usages
    String image;
    //init method
    no usages  Dan Nguyen
    public Recipe(String name, String[] ingredients, String category, String[] instructions, String image) {
        this.name = name;
        this.ingredients = ingredients;
        this.category = category;
        this.instructions = instructions;
        this.image = image;
    }
    4 usages  Dan Nguyen
    public Recipe(String name, String[] ingredients, String category, String[] instructions) {
        this.name = name;
        this.ingredients = ingredients;
        this.category = category;
        this.instructions = instructions;
    }
}
```

The code is tested using JUnit. The test cases are written to test the add and search feature of the recipe database. Additionally, we also test the initialization of a single recipe.

```java
public class RecipeDataBaseTest extends TestCase {
    no usages  Dan Nguyen *
    @Test //test add recipe
    public void testAddRecipe() {
        RecipeDataBase recipeDataBase = new RecipeDataBase();
        Recipe recipe = new Recipe( name: "test", new String[]{"test"}, category: "test", new String[]{"test"});
        recipeDataBase.addRecipe(recipe);
        assertEquals(recipeDataBase.recipes[0], recipe);
    }
    no usages  Dan Nguyen *
    @Test //test search recipe by name
    public void testSearchRecipeByName() {
        RecipeDataBase recipeDataBase = new RecipeDataBase();
        Recipe recipe = new Recipe( name: "test", new String[]{"test"}, category: "test", new String[]{"test"});
        recipeDataBase.addRecipe(recipe);
        assertEquals(recipeDataBase.searchRecipeByName("test"), recipe);
    }
    no usages  Dan Nguyen *
    @Test //Test recipe initiation
    public void testRecipeInit() {
        Recipe recipe = new Recipe( name: "test", new String[]{"test"}, category: "test", new String[]{"test"});
        assertEquals(recipe.name,  actual: "test");
        assertEquals(recipe.ingredients[0],  actual: "test");
        assertEquals(recipe.category,  actual: "test");
        assertEquals(recipe.instructions[0],  actual: "test");
        assertNotSame(recipe.name,  actual: "test2");
        assertNotSame(recipe.ingredients[0],  actual: "test2");
        assertNotSame(recipe.category,  actual: "test2");
        assertNotSame(recipe.instructions[0],  actual: "test2");
        assertNull(recipe.image);
    }
}
```

We can run the the program. According to image below, all test cases passed.



5. Comparison

Recipe Keeper is an application that is available for mobile, tablet, and desktop devices that also collects and organizes recipes for a user, as well as an added share feature [2]. There is also a shopping tab that basically keeps a grocery list for you. This feature is particularly useful since you can add ingredients from a recipe you're currently viewing to your shopping list with one click of a button. Another feature is that there is a planner tab where you can plan your meals daily by simply adding a recipe from the application[2]. MYRecipeBook is another similar application developed by a student that has the same goal of storing recipes as well as streamlining your grocery list. This application shared some similarities as Recipe Keeper with the 'Plan a Meal' and 'View My Shopping List' features. The reason I'm discussing this application as well is because the student presents transparency in their development[1]. The student implemented a client-server architecture for this project. The application has a mobile client where the user can interact with the functionalities of the application which then interfaces with a web service that interacts with the Database Management System (DBMS)[1]. Tastey used the MVC architecture for our design because of the simplicity of our project but using client-server is definitely feasible. Tastey, Recipe Keeper, and MYRecipeBook all perform the task of managing recipes in a library-esque manner but the latter two have very fun and interesting extra functionalities. Our team definitely can see a future in implementing similar features with our own twist.

6. Conclusion

As a team we feel that our project has the potential to reach a large audience around the world due to the fact that while everyone does not like to cook, everyone loves to eat. Our project attempts to do what other apps cannot. Our goal with this project was to create something that was all in one. Other apps that we researched all had a specific niche but our goal was to take the best elements from each of these apps and combine them. One thing that was incredibly important to us was the ability to filter out certain ingredients for recipes. Whether it be for personal taste, ethical, or religious reasons, many people can not eat certain foods. Additionally people have allergies which prevent them from eating things like eggs, shellfish, and milk. By being able to filter out things like this. People can search for things like eggless cake or milk free hot chocolate. The aim of this feature was to allow for people to upload and find recipes of food which they could otherwise not enjoy and build a sense of accessibility and community around something that we all could learn to appreciate a little bit more.

7. References

[1] A. E. Khairi, rep., 2017.
Supervisor: Dr. Bouchaib Falah Ifran, Morocco
http://www.aui.ma/sse-capstone-repository/pdf/spring-2017/MYRecipeBook%20APP.pdf

[2] *The easiest way to organize your recipes* (no date) *Recipe Keeper*. Available at: https://recipekeeperonline.com/ (Accessed: April 21, 2023).

[3] “Visual tools to get things done: Connect people, docs, projects and data.,” *Creately*, 10-Jul-2019. [Online]. Available: http://creately.com/. [Accessed: 21-Apr-2023].

8. Presentation Slides

Tastey

9. GitHub

https://github.com/DanNguyenN/3354-Tastey