



RecipeBook

Project Report

Kian Elyasi, Adrian Perez

Team 9

## Table of Contents

<b>Chapter 1. Introduction:</b>	<b>2</b>
<b>Chapter 2. Functional Requirements:</b>	<b>3</b>
<b>Chapter 3. ERD Diagram:</b>	<b>5</b>
<b>Chapter 4. Entity Set and Relationships:</b>	<b>6</b>
<b>Chapter 5. ERD Schemas:</b>	<b>10</b>
<b>Chapter 6. MySQL Workbench Content:</b>	<b>11</b>
<b>Implementation:</b>	<b>17</b>
<b>How to Run</b>	<b>22</b>
<b>Lessons Learned</b>	<b>23</b>

## Chapter 1. Introduction:

We are creating a recipe social application to manage recipes. We will create a three-tier architecture application that will target users interested in sharing and caring for recipes. The users will be able to add and remove recipes on their profiles. The users will be able to see their favorite recipes and display their favorite meals to their friends. Additionally, the recipes can also be edited to allow users to be able to remove a specific ingredient if they have a specific diet or they are cooking for a group of people that has a specific diet. The application will be very user-friendly and have a strong database that is able to store and query very specific information related to the users and recipes. Therefore, our motivation to do this project is all the forgetting recipes due throughout the past because nobody stored those recipes in a database. This problem can be fixed by developing software that efficiently stores recipes but more importantly saves all the details that are part of the recipe. Additionally, we created a feature that adds recipe ingredients to a user's grocery list. The user can easily access these ingredients if they want to go buy the ingredients at the supermarket. There are many good recipes out on the web and we want to create an application that stores all of these recipes with one click for all users.

## Chapter 2. Functional Requirements:

### Search Recipes:

- Users will be able to search recipes by complexity (quick, complex, beginner, etc.), name, main ingredients, user, suggested
- The application will return the recipes based on these fields
- The user will be able to see suggested recipes

### Display Information About Recipe:

- Users will be able to see the various recipes and their details by clicking on the desired recipe
- The system will show the user things like instructions, ingredients, quantities, pictures

### User Profile:

- Users will be able to create a profile to be saved by the application
- The database will contain encrypted passwords for specific usernames

### User Settings:

- Users will have the ability to create custom settings
- These settings will be stored in the database and used to alter the appearance of the application

### Add/Save Recipe:

- Each user will have their own list of recipes which they can add to
- Users may add someone's public recipe or add their own by creating a new recipe

### Edit Recipe:

- Users can edit recipes (e.g. remove an ingredient, add an ingredient, change the name, change the picture, make it private, etc.,)

### Remove Recipe:

- Users should also be able to remove a recipe from their list of recipes

### Share Recipe:

- Users will be able to share their recipes with other users
- Users should also be able to hide recipes from others
- The system will pick random top-rated recipes to add to a homepage of suggested recipes

### Recipe Ratings:

- Users may have the option to give a recipe a rating

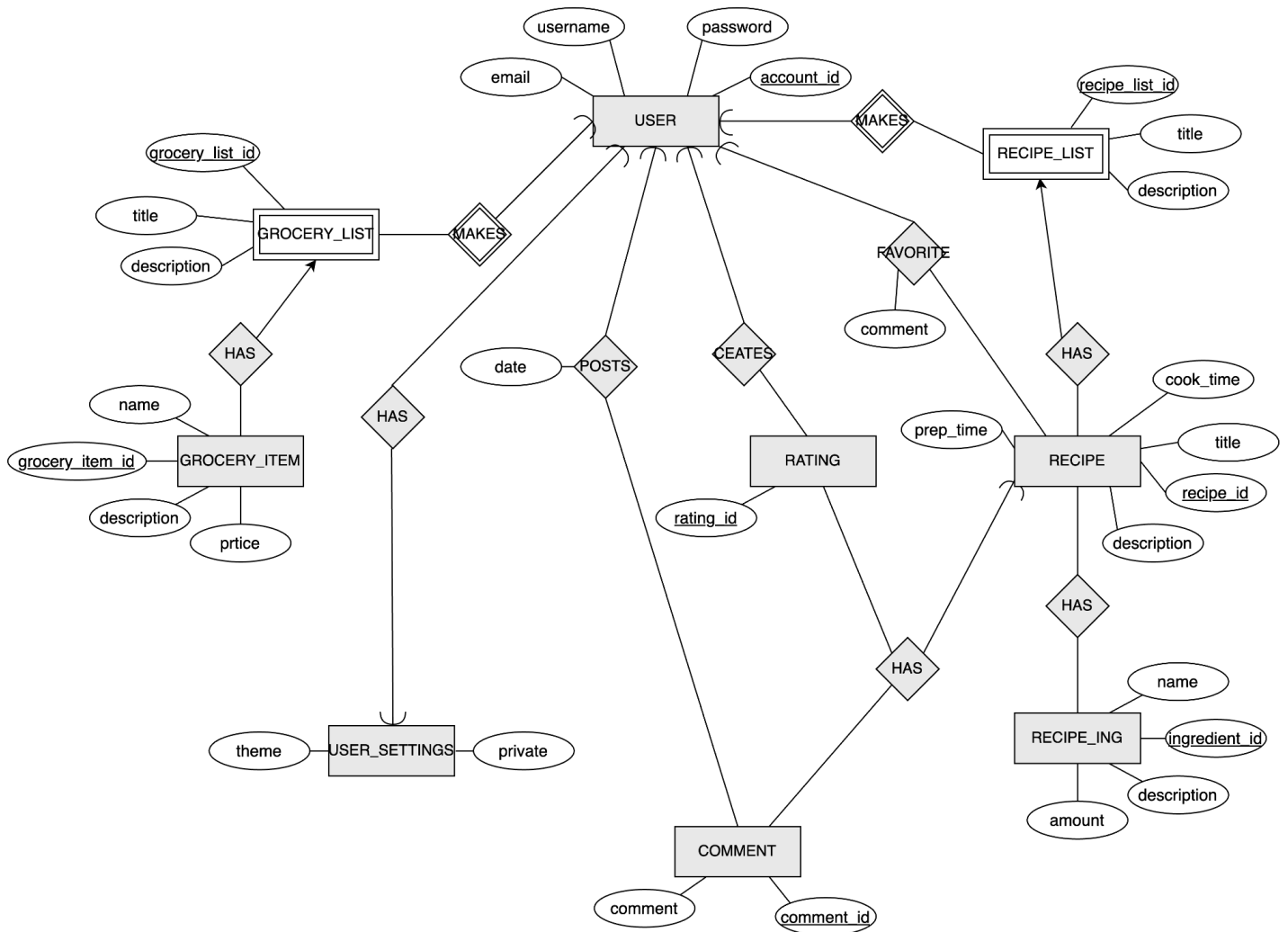
Comments:

- Users may have the option to leave a comment on a recipe

Grocery List:

- Users should be able to add all recipe ingredients to a grocery/shopping list with prices

## Chapter 3. ERD Diagram:



## Chapter 4. Entity Set and Relationships:

Primary Key - PK

Foreign Key - FK

1. **User Entity Set:** The User entity set is responsible for all the users in our database. The PK is the **account\_id** of the user.

2.

### User Entity Relationships:

- A **User** “makes” one or more **Recipe\_List** and its PK is **recipe\_id** along with **account\_id** due to **Recipe\_List** being a weak entity set. We have a many-to-one relationship with User and Recipe List where exactly one user can have many recipe lists.
- A **User** “makes” one or more **Grocery\_List** and its PK is **recipe\_list\_id** along with **account\_id** due to **Grocery\_List** being a weak entity set. We have a many-to-one relationship with User and Grocery\_List where exactly one user can have many grocery lists.
- A **User** “favorites” one or more **Recipes** and its PK is **recipe\_id** along with **account\_id** due to **Recipes** being a weak entity set. We have a many-to-one relationship where a user can favorite one or more recipes.
- A **User** “has” exactly one **User\_Settings** and its PK is **account\_id** along with **account\_id** due to **User\_Settings** being a weak entity set. We have a one-to-one relationship where there is exactly one user with exactly one user setting.
- A **User** “posts” one or more **Comments** and its PK is **comment\_id** along with **account\_id** due to **Comment** being a weak entity set. We have a many-to-one relationship where one user can post multiple comments.
- A **User** “creates” one or more **Ratings** and its PK is **rating\_id** along with **account\_id** due to **Ratings** being a weak entity set. We have a many-to-one relationship where one user can post multiple ratings.

3. **Recipe\_List Entity Set:** The **Recipe\_List** entity set is a weak entity set that is responsible for all the lists of recipes in our database. The PK is the **recipe\_list\_id** for that specific recipe. The **Recipe\_List** has FK from User: **account\_id**.

### Recipe\_List Entity Relationships:

- Each **Recipe\_List** is related (“Makes”) to exactly one **User** and its PK is **recipe\_list\_id**. This is a one-to-many relationship where there can be many **Recipe\_List** for one user.

- A **Recipe\_List** “has” one or more **Recipes** and its PK is `recipe_id`. This is a many-to-one relationship where one `recipe_list` has one or more recipes.
4. **Recipe Entity Set:** The Recipe entity set is a weak entity set responsible for all the recipes in our database. The PK is the **`recipe_id`** for that specific recipe. The recipe has an FK from Users entity set **`account_id`**.

#### Recipe Entity Relationships:

- Each **Recipe** “has” one **Recipe\_List** and its PK is `recipe_list_id`. This is a one-to-many relationship where each recipe corresponds to one `recipe_list`.
  - A **Recipe** “has” one or more **Recipe\_Ing** and its PK is `ingredient_id`. This is a many-to-one relationship where one recipe has one or more recipe ingredients.
  - Each **Recipe** can be “favorited” by one **User** and its PK is `account_id`. This is a one-to-many relationship where each recipe can be favored by one user.
5. **Recipe\_Ing Entity Set:** The Recipe\_ING entity set is a weak entity set that is responsible for containing all the ingredients for the recipes in our database. The PK is the **`ingredient_id`** for the specific ingredient. The FK for the Recipe\_Ing entity set is the **`recipe_id`** and from Recipe\_List being **`recipe_list_id`**.

#### Recipe\_Ing Entity Relationships:

- Each **Recipe\_Ing** “has” one **Recipe** and its PK is `recipe_id`. We have a one-to-many relationship where multiple recipe ingredients correspond to one recipe.
6. **User\_Setting Entity Set:** The User\_Setting entity set is a weak entity set that is responsible for containing the user settings for the user. The PK/FK is the **`account_id`** for the user setting.

#### User\_Setting Entity Relationships:

- A **User\_Setting** “has” exactly one **User** and its PK is `account_id` along with `account_id` due to User\_Settings being a weak entity set. We have a one-to-one relationship where there is exactly one user setting with exactly one user.
7. **Grocery\_Item Entity Set:** The Grocery\_Item Entity set is responsible for containing all the grocery items that we use in our applications. The PK is **`grocery_item_id`** which is the ID of the specific grocery item. The Grocery\_Item has FK from Grocery\_List: **`grocery_list_id`**.



**Grocery\_Item Entity Relationships:**

- Each **Grocery\_Item** “has” one **Grocery\_List** and its PK is **recipe\_list\_id**. This is a one-to-many relationship where there can be multiple grocery items in one grocery list.
8. **Grocery\_List Entity Set:** The **Grocery\_List** Entity Set is a weak entity set responsible for all the grocery lists that we use to make recipes. The PK is **recipe\_list\_id** for the specific grocery list in that set. **Grocery\_List** has an FK from the Users entity set **account\_id**.

**Grocery\_List Entity Relationships:**

- A **Grocery\_List** “has” one or more **Grocery\_Item** and its PK is **grocery\_item\_id**. This is a many-to-one relationship which means there are one or more grocery items on a grocery list.
  - Each **Grocery\_List** is related (“Makes”) to exactly one **User** and its PK is **account\_id**. This is a one-to-many relationship because there are multiple grocery lists for one user.
9. **Comment Entity Set:** The **Comment** entity set is a weak entity responsible for all comments made on a recipe. The Pk is **comment\_id** for the specific comment. The comment entity set has multiple Fk’s which come from User and Recipe entity sets: **account\_id** and **recipe\_id**.

**Comment Entity Set:**

- There can be many **Comments** posted (“Posts”) for exactly one **User**. Therefore we have many to at least one relationship.
  - There can be many **Comments** per single **Recipe**. Therefore we have many to at least one relationship where Recipes can “Has” multiple comments.
10. **Rating Entity Set:** The **Rating** entity set is a weak entity responsible for all ratings associated with a Recipe. The PK is **rating\_id** for specific rating instances. This entity set has multiple FKs which come from User and Recipe entity sets: **account\_id** and **recipe\_id**.

**Rating Entity Set:**

- There can be many **Ratings** created (“Creates”) for exactly one **User**. Therefore we have many to at least one relationship.
- There can be many **Ratings** per single **Recipe**. Therefore we have many to at least one relationship where one Recipe can “Has” multiple Ratings.

## Chapter 5. ERD Schemas:

- User(account\_id, username, email, password)
- User\_Settings(account\_id, private, theme)
- Recipe\_List(recipe\_list\_id, account\_id, title, description)
- Favorite(account\_id, recipe\_id, comment)
- Recipe(recipe\_id, account\_id, title, description, prep\_time, cook\_time, image\_url, how\_to, private)
- Recipe\_Ing(ingredient\_id, name, description, amount)
- Grocery\_List(grocery\_list\_id, account\_id, title, description)
- Grocery\_Item(grocery\_item\_id, name, description, price)
- Comment(comment\_id, account\_id, recipe\_id, comment)
- Posts(comment\_id, account\_id, date)
- Rating(rating\_id, account\_id, recipe\_id, rating)

# Chapter 6. MySQL Workbench Content:

## User

1 • SELECT \* FROM Recipe\_Book.User;

account_id	username	email	password
1	adrian123	adria...	12345
2	kem123	kem...	12345
3	bob456	bob4...	45678
4	ted123	ted1...	H7WtG...
5	karen987	kare...	@cMcN...
6	jacob6783	jaco...	CeW4hr...
7	elisa78	elisa...	0fZge5...
8	berry432	ber...	2VmaB...
9	pedro92	pedr...	sQOAO...
10	lena56	lena...	Z6emiZ...
NULL	NULL	NULL	NULL

Table: **User**  
Columns: account\_id (int PK), private (tinyint), theme (varchar(45))

## User\_Settings

1 • SELECT \* FROM Recipe\_Book.User\_Settings;

account_id	private	theme
1	0	blue
2	0	green
3	1	green
4	0	blue
5	1	grey
6	0	grey
7	0	blue
8	0	blue
9	0	blue
10	0	blue
NULL	NULL	NULL

Table: **User**  
Columns: account\_id (int PK), username (varchar(45)), email (varchar(45))

## Recipe\_List

[illegible]

## Favorite

SCHEMAS
Limit to 1000 rows

Filter objects

- Perez
  - Recipe\_Book
    - Tables
      - Comment
      - Favorite**
        - Columns
          - account\_id
          - recipe\_id
          - comment
        - Indexes
        - Foreign Keys
        - Triggers
        - Grocery\_Item
        - Grocery\_List
        - Posts
        - Rating
        - Recipe
        - Recipe\_Ing

```
1 • SELECT * FROM Recipe_Book.Favorite;
```

100%
1:1

Result Grid
Filter Rows: 
Edit: Export/Import:

account_id	recipe_id	comment
1	2	NULL
2	6	NULL
5	7	Yummy
6	8	NULL
8	3	The bes...
9	8	Yasy
9	1	NULL
10	3	NULL
10	6	NULL
NULL	NULL	NULL

Object Info

Table: **Favorite**

Columns:

- account\_id int PK
- recipe\_id int PK
- comment varchar(255)

Session

## Recipe

Limit to 1000 rows

1 SELECT \* FROM Recipe\_Book.Recipe;

100% 1:1

recipe_id	account_id	title	description	prep_time	cook_time	image_url	how_to	private
1	4	Ham...	Shadwich...	1 minute	HULL	HULL	Just p...	0
2	2	Sala...	Shadwich...	1 minute	HULL	HULL	Just p...	0
3	7	Chic...	Pasta dis...	10 minutes	10 minutes	HULL	HULL	1
4	3	Chic...	Breaded...	10 minutes	30 minutes	HULL	HULL	0
5	5	Cea...	Salad wit...	10 minutes	HULL	HULL	HULL	1
6	10	Beet...	Salad wit...	10 minutes	HULL	HULL	HULL	0
7	9	Cho...	Cooke	15 minutes	20 minutes	HULL	HULL	1
8	8	Cake	Cake	15 minutes	30 minutes	HULL	HULL	0
9	2	Lem...	Juice ma...	5 minutes	HULL	HULL	HULL	0
10	1	Punch	Juice ma...	5 minutes	HULL	HULL	HULL	0
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Object Info Session

Table: **Recipe**

Columns:

- recipe\_id int PK
- account\_id int PK
- title varchar(45)

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

## Recipe\_Ing

Limit to 1000 rows

1 Grocery\_ItemSELECT \* FROM Recipe\_Book.Recipe\_Ing;

100% 13:1 1 error found

ingredient...	name	description	amount
1	Ham	Ham	1 lb
2	Salami	Salami	1 lb
3	Sugar	Suga	3 cups
4	Lettu...	greens	5 oz
5	Pasta	pasta	1 pack
6	Mari...	sauce to...	1 bottle
7	Chic...	chicken	1 lb
8	Lemon	lemon	10
9	Apple	apple	5
10	Beets	beet	3
HULL	HULL	HULL	HULL

Object Info Session

Table: **Grocery\_Item**

Columns:

- grocery\_item\_id int PK
- name varchar(45)
- description varchar(254)

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

## Grocery\_List

SCHEMAS

Filter objects

- Perez
  - Recipe\_Book
    - Tables
      - Comment
      - Favorite
      - Grocery\_Item
      - Grocery\_List**
        - Columns
        - Indexes
        - Foreign Keys
        - Triggers
      - Posts
      - Rating
      - Recipe
      - Recipe\_Ing
      - Recipe\_List
      - User
      - User\_Settings

Object Info Session

Table: **Grocery\_List**

Columns:

grocery_list_id	int PK
account_id	int PK
title	varchar(45)

1 SELECT \* FROM Recipe\_Book.Grocery\_List;

100% 1:1

Result Grid

grocery_list...	account_id	title	description
1	4	San...	sandwich...
2	10	Groc...	NULL
4	2	Ran...	random s...
5	5	List	NULL
6	7	Pers...	NULL
7	9	Get t...	need to g...
8	8	NULL	NULL
9	5	NULL	NULL
10	2	Ran...	random s...
NULL	NULL	NULL	NULL

Form Editor

Field Types

Query Stats

Execution Plan

## Grocery\_Item

SCHEMAS

Filter objects

- Perez
  - Recipe\_Book
    - Tables
      - Comment
      - Favorite
      - Grocery\_Item
      - Grocery\_List
      - Posts
      - Rating
      - Recipe
      - Recipe\_Ing
      - Recipe\_List
      - User
      - User\_Settings

Object Info Session

Table: **Grocery\_Item**

Columns:

grocery_item_id	int PK
name	varchar(45)
description	varchar(254)

1 SELECT \* FROM Recipe\_Book.Grocery\_Item;

100% 1:1

Result Grid

grocery_item...	name	description	price
1	Ham	deli meat	2.50
2	Salami	deli meat	3.00
3	Sugar	NULL	5.00
4	Lettu...	greens	3.50
5	Beets	greens	3.50
6	Pasta	NULL	5.00
7	Marl...	NULL	5.50
8	Chic...	NULL	10.50
9	Lemon	NULL	1.00
10	Apple	NULL	1.20
NULL	NULL	NULL	NULL

Form Editor

Field Types

Query Stats

Execution Plan

## Comment

SCHEMAS

1 • SELECT \* FROM Recipe\_Book.Comment;

100% 1:1

Result Grid

comment_id	account_id	comment	recipe_id
1	1	Great r...	4
2	4	Fantastic	5
3	6	Very tasty	10
4	10	No com...	7
5	4	Added...	1
6	7	Could b...	8
7	9	Great r...	4
8	8	Genius!	6
9	8	Very tasty	8
10	10	Super t...	8
NULL	NULL	NULL	NULL

Object Info Session

Table: **Comment**

Columns:

- comment\_id int PK
- account\_id int PK
- comment varchar(264)

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

## Posts

SCHEMAS

1 • SELECT \* FROM Recipe\_Book.Posts;

100% 1:1

Result Grid

post_id	account_id	date
1	3	2022-11-03 00:00:00
2	1	2022-11-03 00:00:00
3	9	2022-11-03 00:00:00
4	8	2022-11-03 00:00:00
5	3	2022-11-03 00:00:00
6	4	2022-11-03 00:00:00
7	8	2022-11-03 00:00:00
8	7	2022-11-03 00:00:00
9	5	2022-11-03 00:00:00
10	2	2022-11-03 00:00:00
NULL	NULL	NULL

Object Info Session

Table: **Posts**

Columns:

- post\_id int PK
- account\_id int PK
- date datetime

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan



## Rating

## Implementation:

### Login


The login page allows the user to log in or signup if they do not have an account. The login button redirects the user to the home page whilst the signup button takes them to the signup page. Every login requires a username or email and password.



The screenshot shows a login form with a blue header containing the word "Login" in white. Below the header, there are two input fields: "Email:" with the value "adrian123" and "Password:" with masked characters ".....". At the bottom, there are two buttons: "login" and "Sign Up".

### SignUp

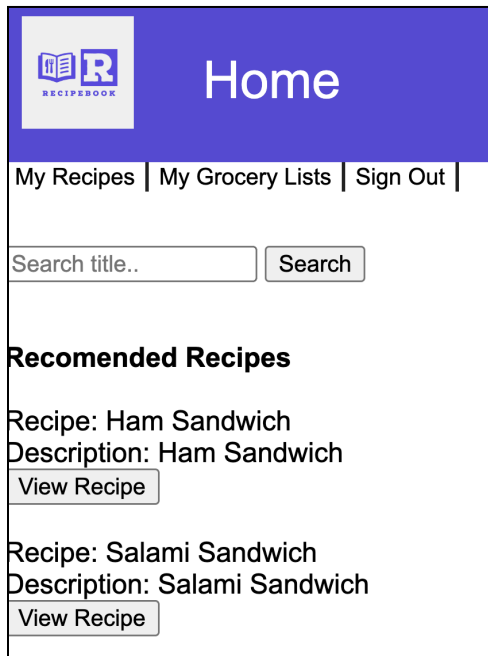
The signup page allows the user to create an account. Once the user has a successful signup (meaning the user information imputed in the form does not already exist) they are presented with a button that will redirect them to the login page.



The screenshot shows a sign up form with a blue header containing the words "Sign Up" in white. Below the header, there are three input fields: "Username:" with the value "adrian123", "Email:" which is empty, and "Password:" with masked characters ".....". Below these fields is a "Sign Up" button. At the bottom, there is a "User Added" message and a "Back to Login" button.

## Home And Recommended Recipes

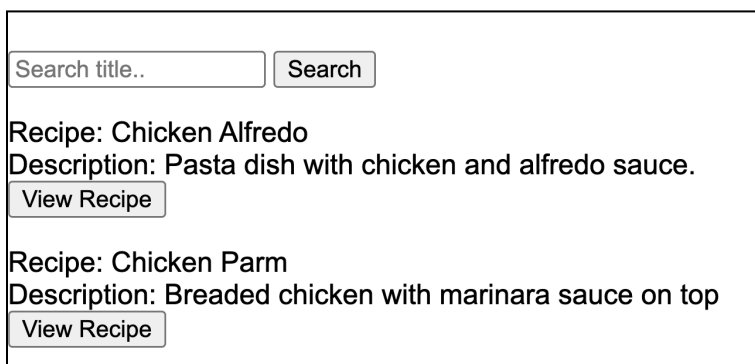
The home page contains links to other pages, searches, and recommended recipes. The recommended recipes exclude the user's recipes and any recipes that are private. From the recipes that show up, you can view them to get more information about the recipe.



The screenshot shows the 'Home' page of the RecipeBook application. It features a blue header with the 'RecipeBook' logo and the word 'Home'. Below the header is a navigation bar with links for 'My Recipes', 'My Grocery Lists', and 'Sign Out'. A search bar with the placeholder text 'Search title..' and a 'Search' button is located below the navigation bar. The main content area is titled 'Recommended Recipes' and lists two recipes: 'Ham Sandwich' and 'Salami Sandwich'. Each recipe entry includes the recipe name, a description, and a 'View Recipe' button.

## Search

The search allows you to look up recipes that the user didn't make. The search takes the imputed value and does a query to all recipes that aren't the user and aren't private. From the recipes that show up, you can view them to get more information about the recipe.



The screenshot shows the 'Search' page of the RecipeBook application. It features a search bar with the placeholder text 'Search title..' and a 'Search' button. Below the search bar, the results are displayed under the heading 'Recipe: Chicken Alfredo'. The description for this recipe is 'Pasta dish with chicken and alfredo sauce.' and there is a 'View Recipe' button. Another recipe, 'Chicken Parm', is also listed with the description 'Breaded chicken with marinara sauce on top' and a 'View Recipe' button.

## Comments

After the user has been redirected to the view page and is viewing the desired recipe they may leave a comment. Since the user is signed in all the user needs to provide is the comment in the input box then press “Add Comment”. This will add the comment to the database.

**Comments**  
  
Comment: This is good  
User: adrian123

### **Ratings**


After the user has been redirected to the view page and is viewing the desired recipe they may leave a rating. The user is allowed to press on a star and then submit that by pressing the “Submit” button. This rating is then stored in the database.

Thanks for your feedback !!!  

☐ ★ ☐ ★ ☐ ★ ☐ ★ ☐ ★

### **Create A Recipe**

If you click on the create recipe page within the header the user is able to create a new recipe. A form is presented, which the user fills out and submits adding this recipe to the database.



## Recipe Create Form

[Home](#) | [My Recipes](#) | [My Grocery Lists](#) | [Sign Out](#)

Recipe Title:

Short Description:

Peperation Time:

Cook Time:

Recipe's How To:  
Please describe how to prepare this recipe.


Should this recipe be private? ☐

Add Recipe

Missing fields! make sure: Recipe Title, Short Description, Peperation Time, and Cook Time are filled out.

## Edit Recipe

This page's link is located on the my recipe page. Each recipe has the option to edit whatever the user wants to change about the recipe. This will update the recipe in the database. The form is preloaded with the information that already existed in the database.



## Recipe Edit Form

[Home](#) | [My Recipes](#) | [My Grocery Lists](#) | [Sign Out](#)

Recipe Title:

Short Description:

Peperation Time:

Cook Time:

Recipes How To:

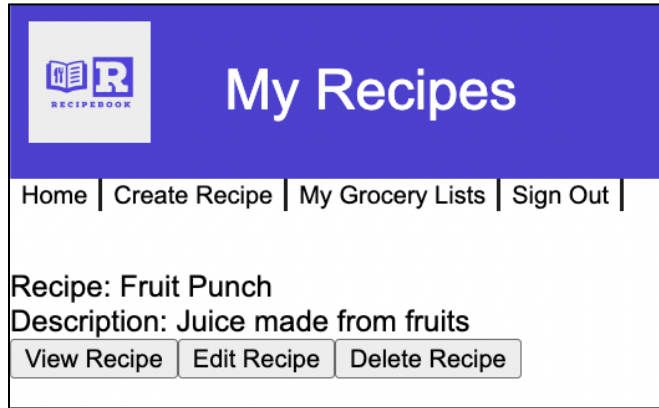
1) Stir together the cranberry juice, pineapple juice, orange juice, and lime juice in a large pitcher. Chill until you are ready to serve.  
2) Add sliced fruit and ginger ale just before serving.

Should this recipe be private? (Not Private) ☐

Save Changes

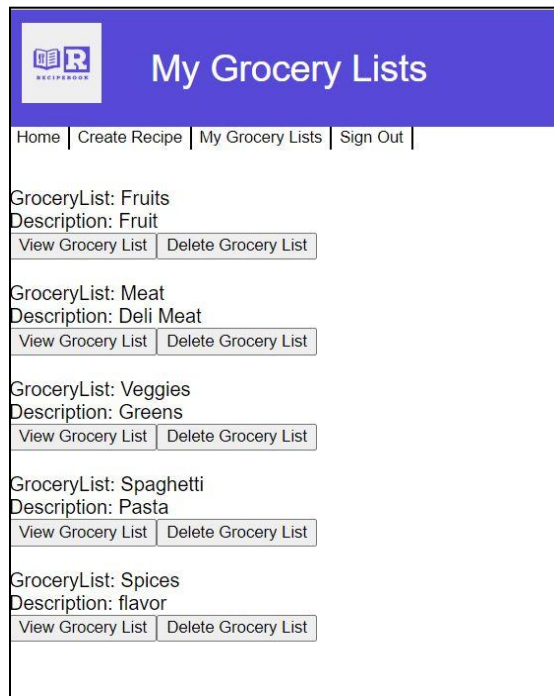
## Delete Recipe

When on my recipe page, the user has the option to delete a recipe they created. Simply by pressing the “Delete Recipe” button the user can delete the recipe from the database.



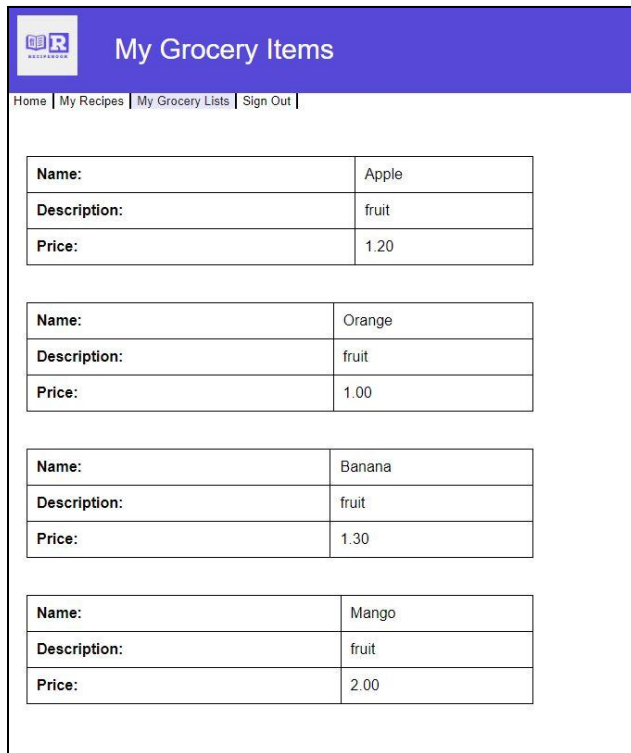
## Grocery List

When you click on the My Grocery Lists, the user gets a list of grocery lists with their name and description so the user can pick out which grocery list they want to select. Also, the user can delete the grocery lists if they do not want that list anymore by selecting delete grocery lists.



## Grocery Item

When you click on “View Grocery Lists” on the Grocery Lists pages, the user can view all the grocery items that are related to that specific grocery list. The grocery items have a description and price so the user can select the specific items they want when they go grocery shopping and the grocery items are filtered for that specific grocery list.



My Grocery Items	
Home   My Recipes   My Grocery Lists   Sign Out	
Name:	Apple
Description:	fruit
Price:	1.20
Name:	Orange
Description:	fruit
Price:	1.00
Name:	Banana
Description:	fruit
Price:	1.30
Name:	Mango
Description:	fruit
Price:	2.00

## How to Run

- 1) Make sure you installed MySQL and have the database with the correct credentials
- 2) Must have Tomcat installed
- 3) Clone or put a folder from GitHub in the Tomcat/webapps/ROOT directory
- 4) Run the database script located in the GitHub repo
- 5) In the terminal run ./startup.sh
- 6) Now that everything is running go to the login which is:  
<http://localhost:8081/CS157A-team9/LoginPage.jsp>
- 7) Use application how you would use any other online application
- 8) Use the header of our application to navigate to different parts of our implementation

## **Lessons Learned**

### **Adrian**

I learned many things during this project. One of those was the JSP language, which was very close to PHP but had some slight differences. Working with JSP I learned more about forms and even learned about making a search function. This was also my first time using more complex queries in my code which was a great learning experience. All in all this project really tested my skills in SQL, JSP, HTML, and CSS, and I can proudly say that I'm comfortable with these languages.

### **Kian**

I gained a lot of hands-on experience from this project. First, I learned about MySQL in this class and this project helped me to improve my knowledge on MySQL databases. Additionally, I learned how to use Apache Tomcat and localhost to create a web application for this course. Moreover, I learned the JSP language which is different from the Java Syntax in a variety of ways. I had to learn the JSP language syntax and how to incorporate HTML code within the JSP file. Additionally, I learned to connect the JSP language to MySQL to get the webpage to run properly. This was the first time I used more complex HTML scripts to make the buttons and other parts of my code on the webpage. In conclusion, I built on my knowledge in MySQL, HTML, JSP while learning new aspects of programming languages while creating this web application.