RECIPEBOOK

RecipeBook

Project Report

Kian Elyasi, Adrian Perez

Team 9

Table of Contents

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

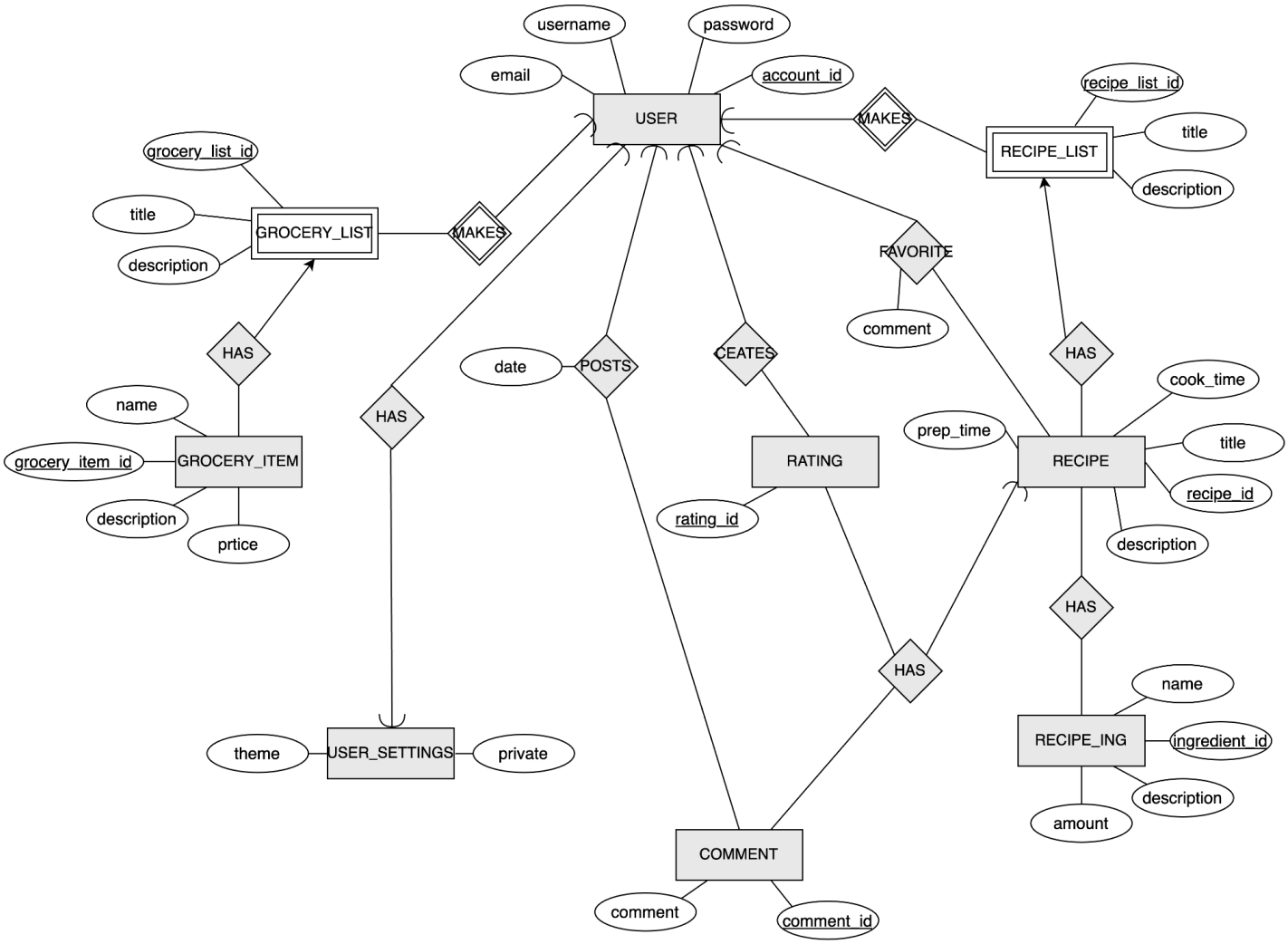
Recipe Ratings:

- 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:

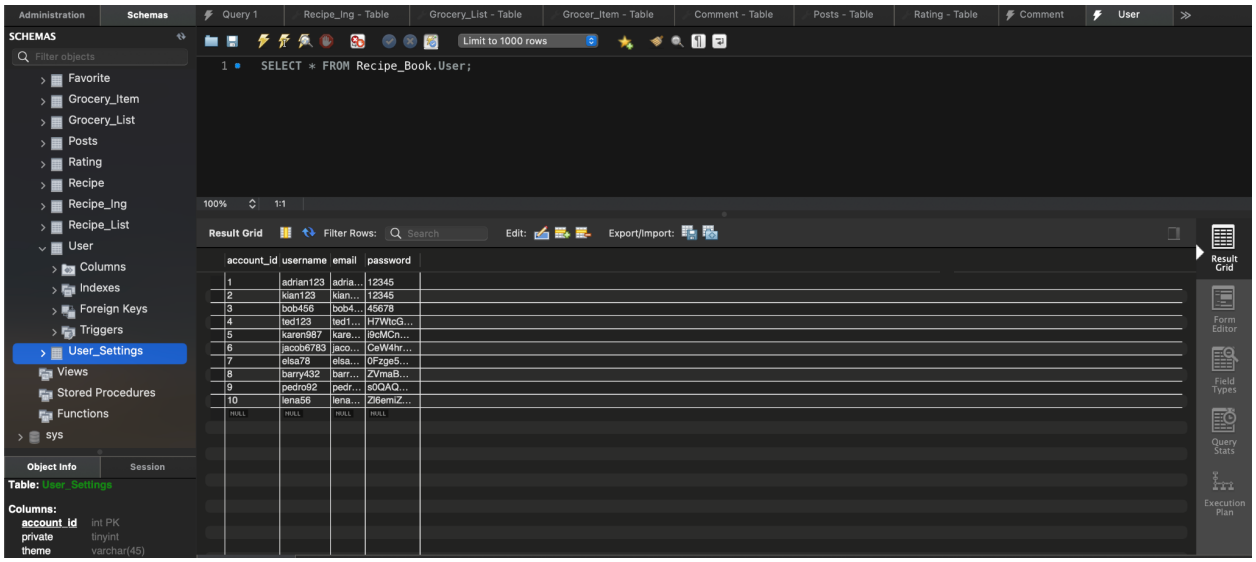
- There can be many **Ratings** created ("Creates") for exactly one **User**. Therefore we have many to at least one relationship.
- There can be many **Ratings** per single **Recipe**. Therefore we have many to at least one relationship where one Recipe can "Has" multiple Ratings.

Chapter 5. ERD Schemas:

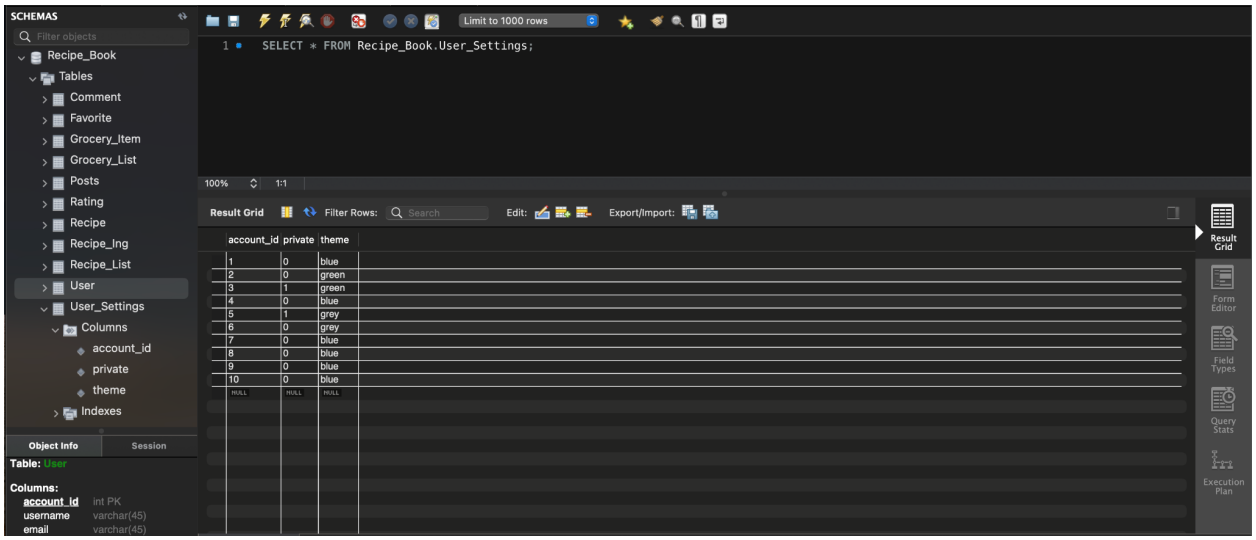
- User(account_id, username, email, password)
- User_Settings(account_id, private, theme)
- Recipe_List(recipe_list_id, account_id, title, description)
- Favorite(account_id, recipe_id, comment)
- Recipe(recipe_id, account_id, title, description, prep_time, cook_time, image_url, how_to, private)
- Recipe_Ing(ingredient_id, 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:

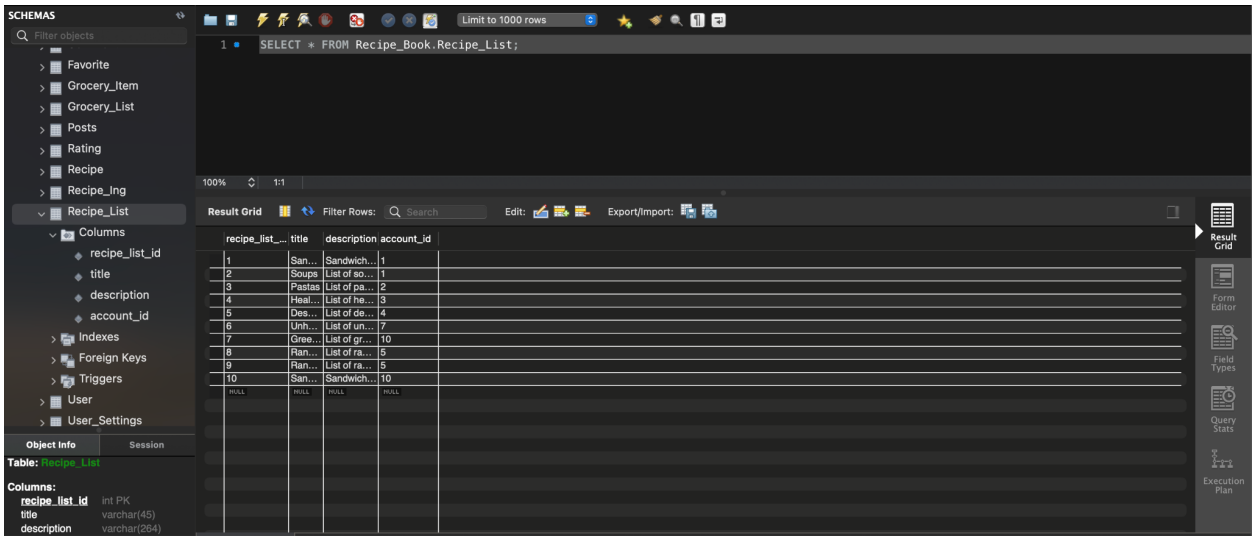
User



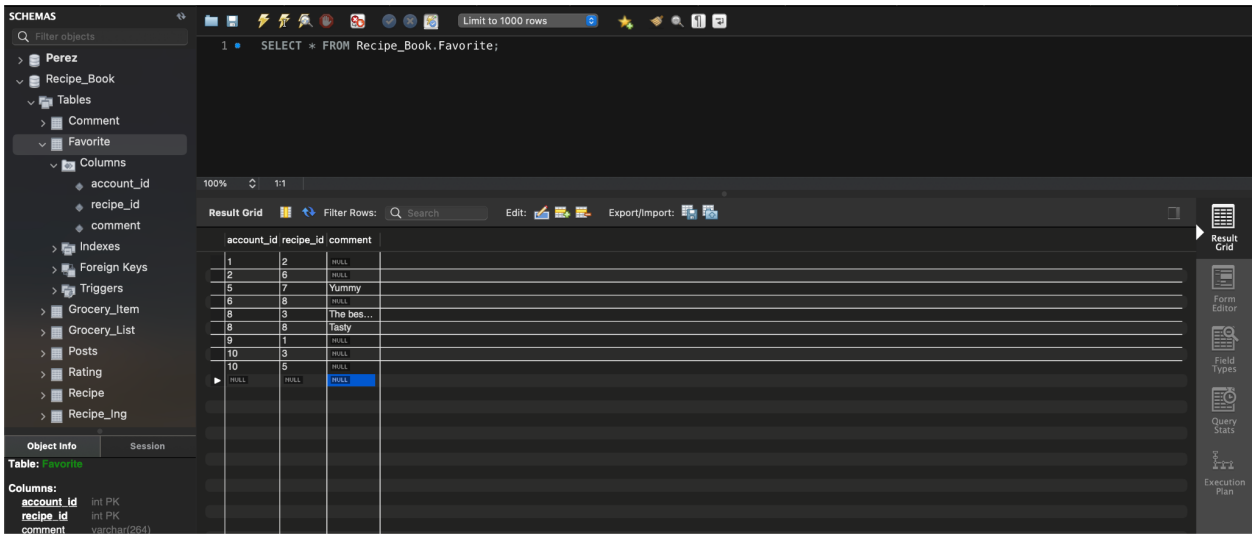
User_Settings



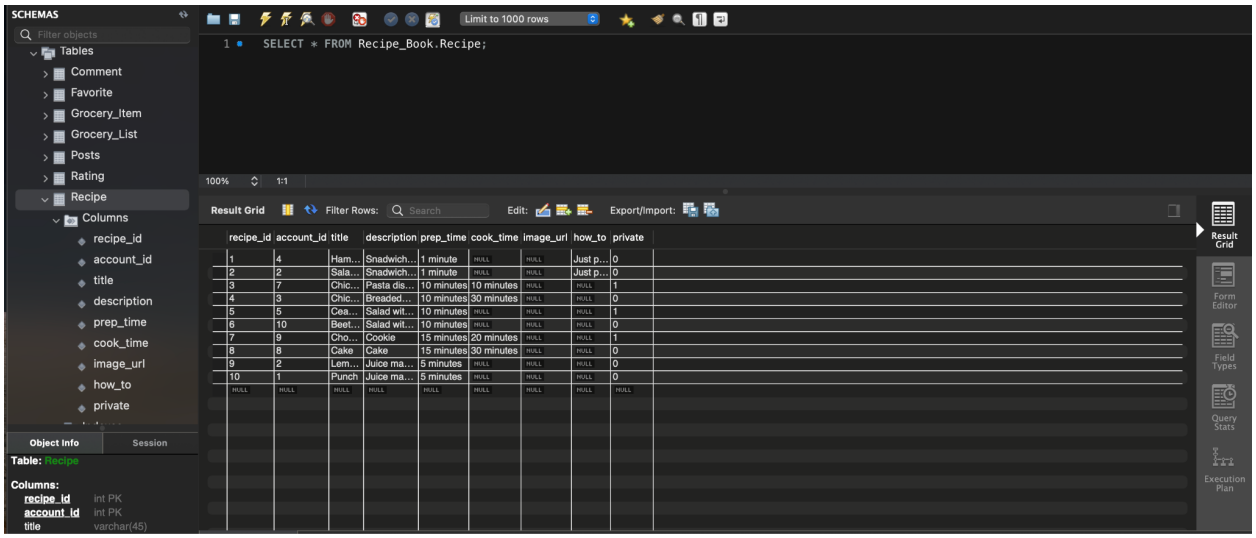
Recipe_List



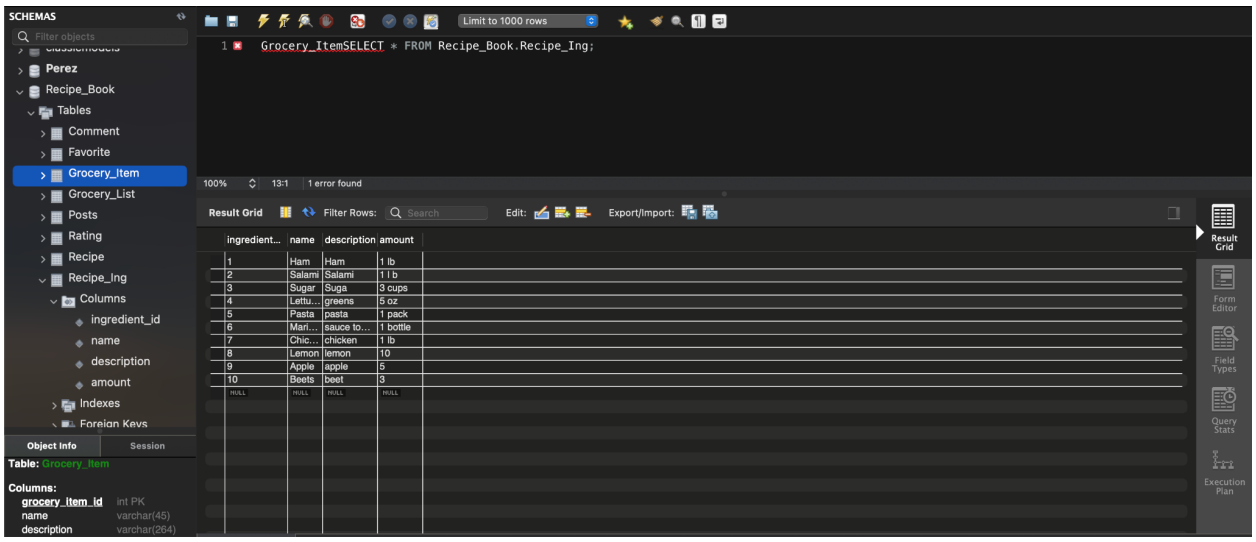
Favorite



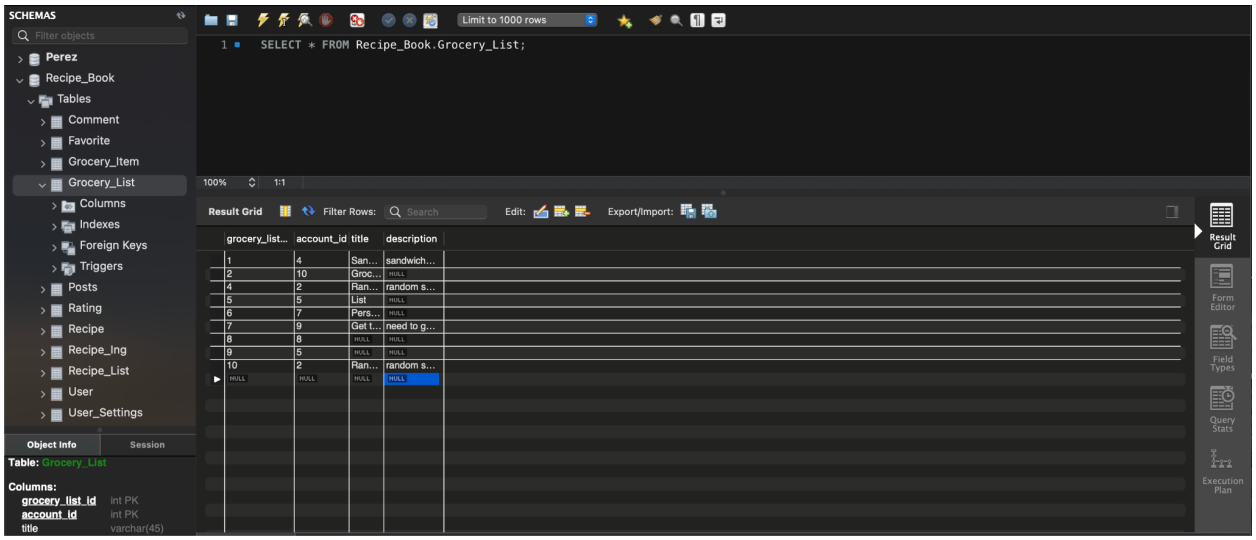
Recipe



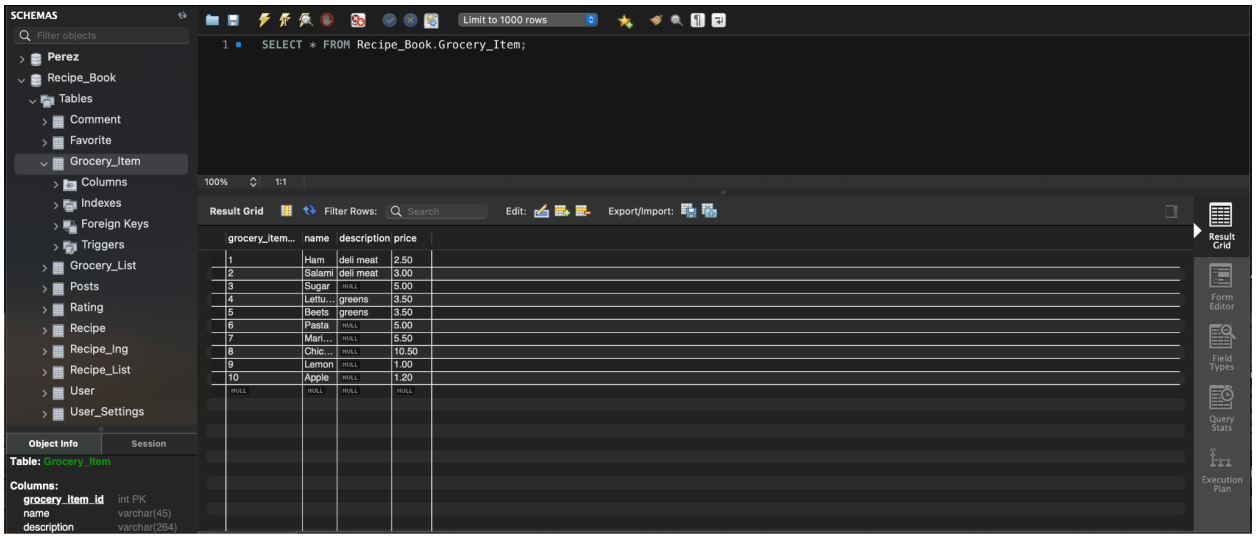
Recipe_Ing



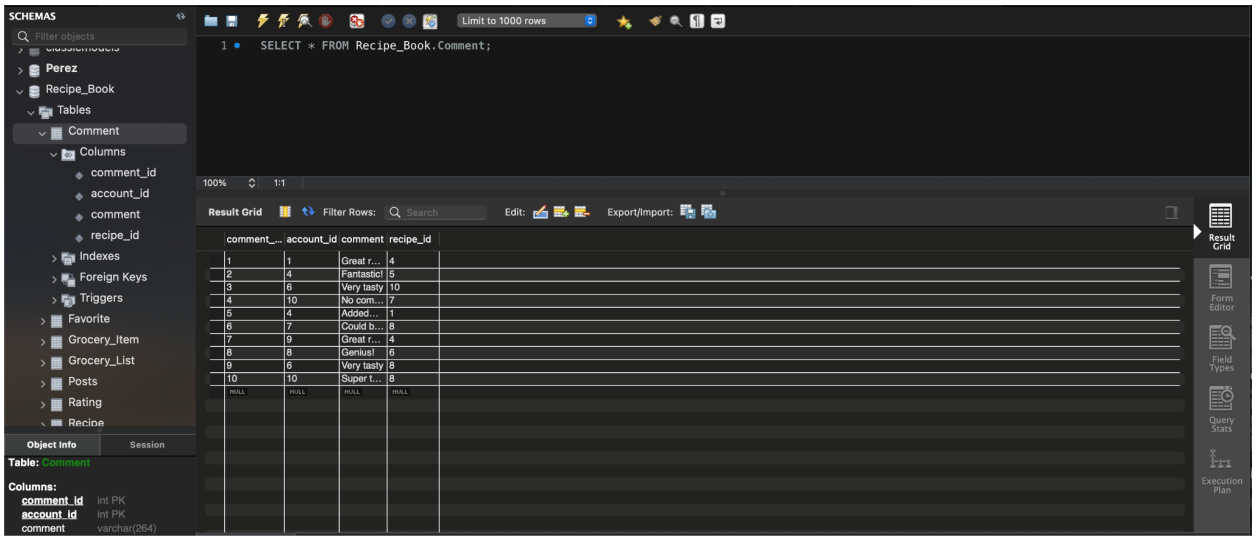
Grocery_List



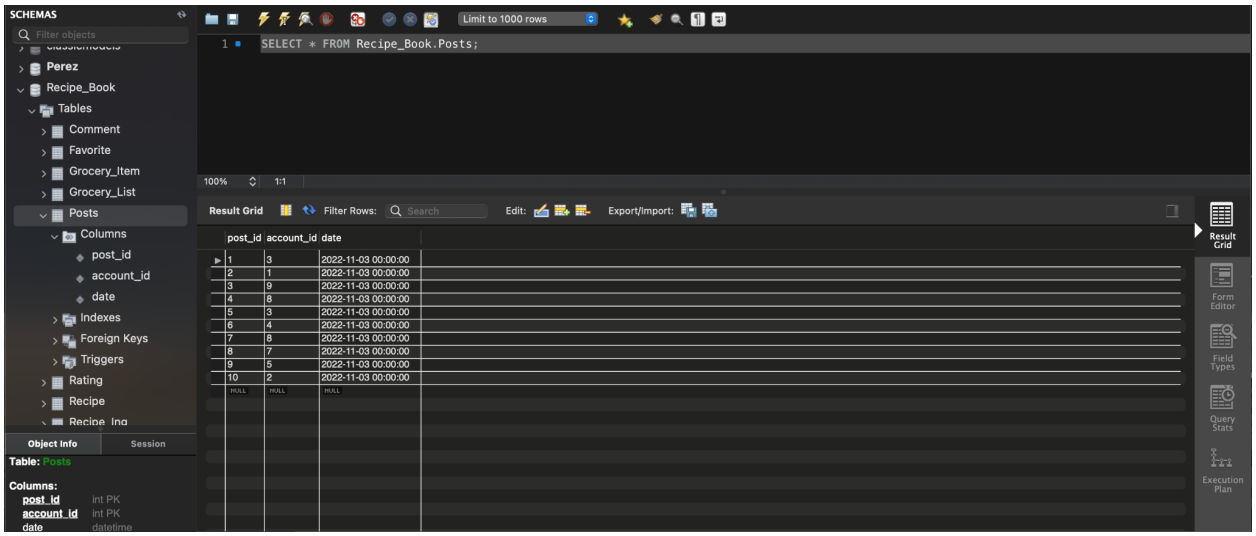
Grocery_Item



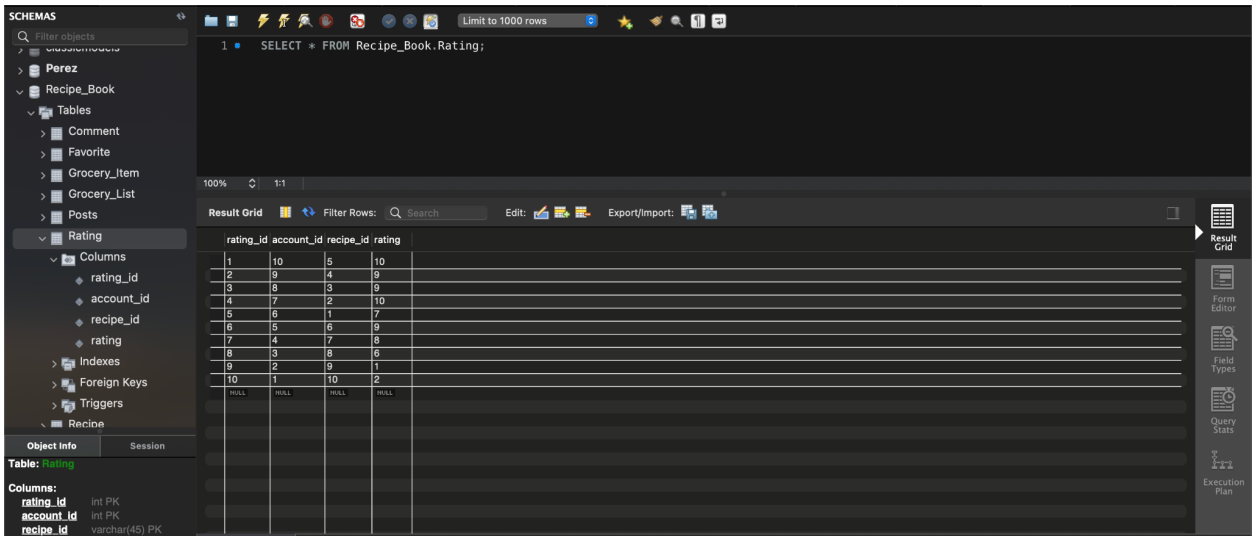
Comment



Posts



Rating


SCHEMAS
Filter objects
Perez
Recipe_Book
Tables
Comment
Favorite
Grocery_Item
Grocery_List
Posts
Rating
Columns
rating_id
account_id
recipe_id
rating
Indexes
Foreign Keys
Triggers
Recipe
Object Info
Session
Table: Rating
Columns:
rating_id int PK
account_id int PK
recipe_id varchar(45) PK
Limit to 1000 rows
1 SELECT * FROM Recipe_Book.Rating;
100%
1:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
rating_id account_id recipe_id rating
1 10 5 10
2 9 4 9
3 8 3 9
4 7 2 10
5 6 1 7
6 5 6 9
7 4 7 8
8 3 8 6
9 2 9 1
10 1 10 2
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Implementation:

Login

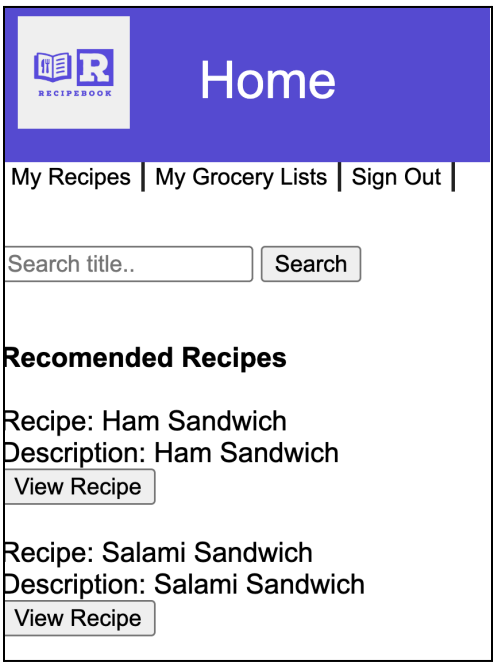
The login page allows the user to log in or signup if they do not have an account. The login button redirects the user to the home page whilst the signup button takes them to the signup page. Every login requires a username or email and password.

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.

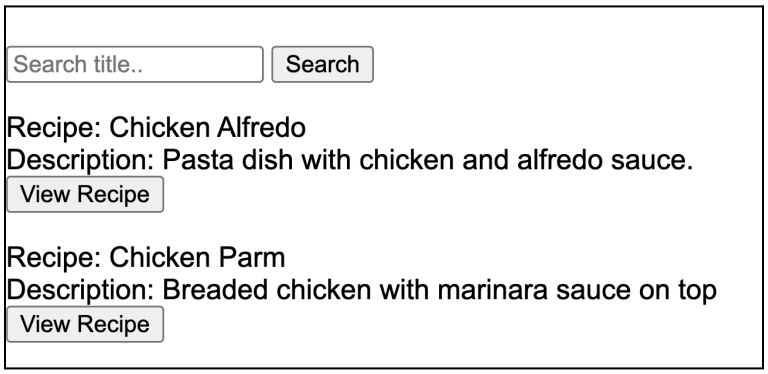
Home And Recommended Recipes

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.



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.



Comments

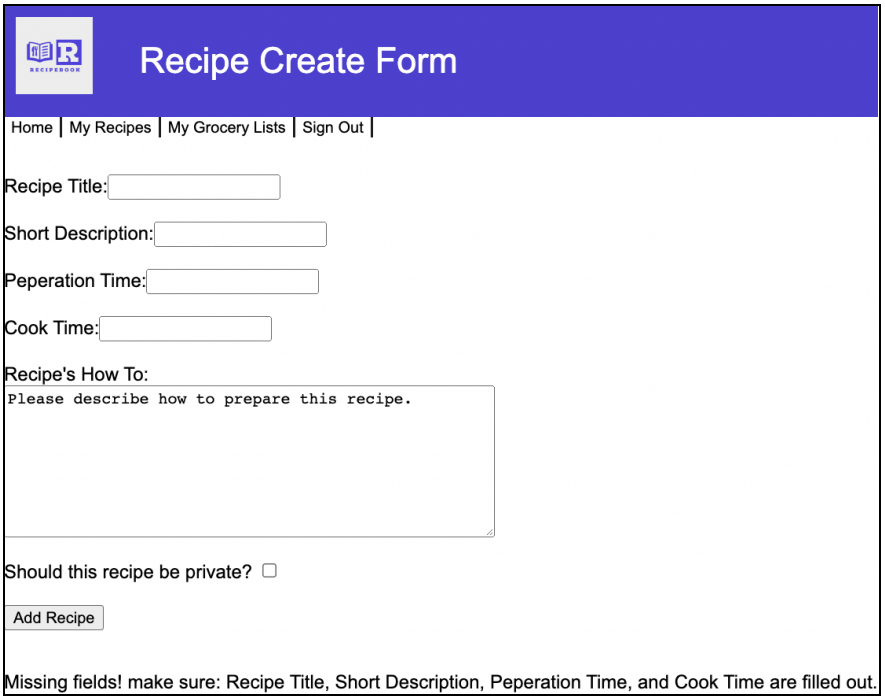
After the user has been redirected to the view page and is viewing the desired recipe they may leave a comment. Since the user is signed in all the user needs to provide is the comment in the input box then press “Add Comment”. This will add the comment to the database.

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.

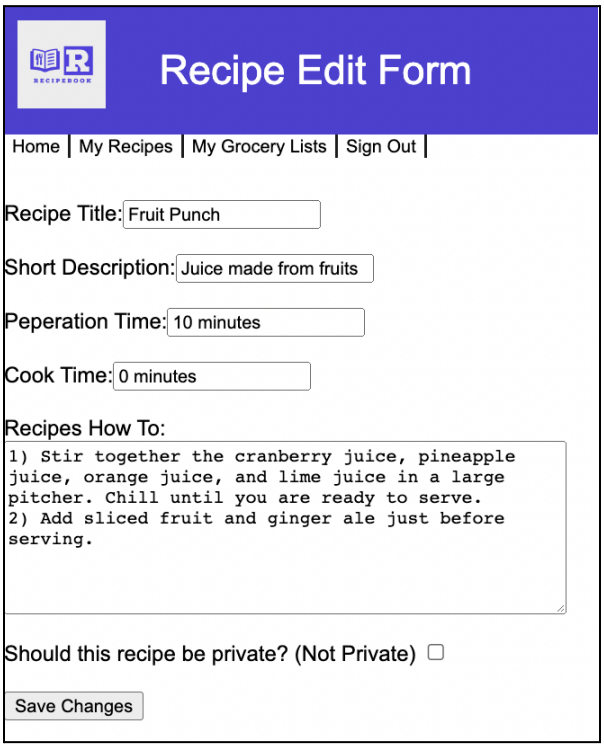
Create A Recipe

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.



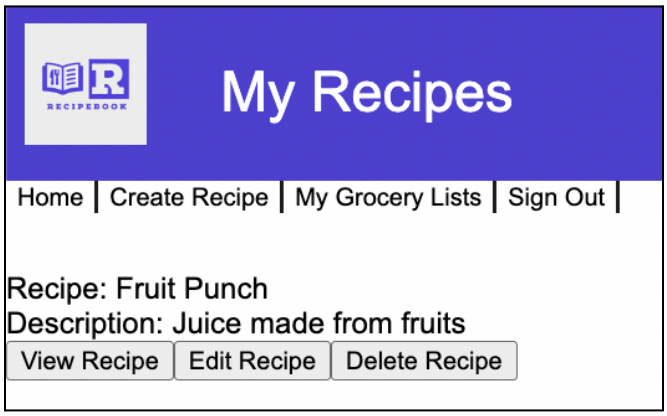
Edit Recipe

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.



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.

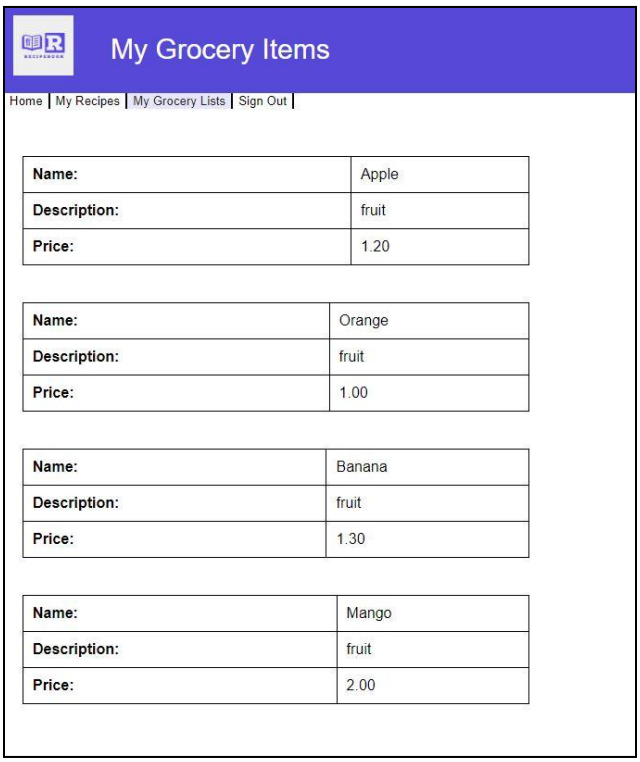


Grocery List

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.



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.