**1. How I set up the database**

I used **phpMyAdmin** with **XAMPP** to set up the database. The database is called recipe_hub. I created multiple tables to handle different parts of the app. This includes:

- users – to store logins
- recipes – for all the user-submitted recipes
- ingredients – and a connection table recipe_ingredients
- steps – for step-by-step instructions
- categories – to sort recipes
- tags and recipe_tags – for tagging like “Vegan” or “Spicy”
- ratings, comments, favorites – for user interaction

All tables are linked using foreign keys, and I made sure to follow normalization rules to avoid duplicated data. I used proper data types like VARCHAR, INT, and TEXT depending on the content.

**2. How I set up a virtual server**

To run my site locally like it was online, I installed **XAMPP**, which includes Apache and MySQL. I placed all my PHP files inside:

C:\xampp\htdocs\recipe-hub\recipe-hub\

I accessed the site through the browser using:

http://localhost/recipe-hub/recipe-hub/

MySQL runs through phpMyAdmin, where I created the database and managed the tables. This setup helped me test everything before it goes live.

**3. Techniques used to build a dynamic web app**

I used **PHP** to handle logic and **MySQL** to pull data from the database. The site changes depending on what the user does. For example:

- Logged-in users see more options in the navbar (Add Recipe, Logout).
- Recipes are shown based on what’s in the database.
- Tags and categories are loaded from the DB automatically.
- Logged-in users can rate, favorite, comment, and edit their own posts.

I used if statements to check if users are logged in ($_SESSION) and foreach loops to display ingredients, tags, and steps dynamically.

**4. Techniques used to manipulate data**

I used **prepared statements** to safely insert, update, and delete data. Here's how I handled it:

- INSERT used for new users, recipes, steps, ratings, etc.
- UPDATE used for editing recipes.
- DELETE used for removing recipes or ingredients.
- SELECT used everywhere to show lists of recipes, tags, ingredients, and more.

All SQL was run through **PDO** using the $pdo->prepare() and $pdo->execute() methods. This makes it more secure and avoids SQL injection.

**5. Testing (based on IPO logic)**

I tested the whole project by going through every user action possible. Each test was done at least once with both valid and invalid inputs.

**Test Case 1: Register User**

- Input: username, email, password
- Result: User created and redirected > Passed

**Test Case 2: Login**

- Input: correct and wrong credentials
- Result: Session starts, or error shows > Passed

**Test Case 3: Add Recipe**

- Input: title, description, category, ingredients, steps, tags
- Result: New recipe saved in DB and shown on homepage > Passed

**Test Case 4: Edit Recipe**

- Input: change text and save
- Result: Recipe updates > Passed

**Test Case 5: Rate a Recipe**

- Input: 1–5 stars
- Result: Rating saved and averaged > Passed

**Test Case 6: Add Favorite**

- Action: Click Favorite
- Result: Recipe appears in favorites section > Passed

**Test Case 7: Add Tags**

- Input: tick checkboxes

- Result: Tags saved and shown in recipe view > Passed

**Test Case 8: Commenting**

- Input: comment box
- Result: Comment saved and shown > Passed

**Test Case 9: Form Validation**

- Input: empty forms
- Result: Error messages show, nothing submitted > Passed

Every failed test was fixed and tested again until it passed.

**Final Thoughts**

The project works well, all data is dynamic and secure. Users can do everything from register to managing their own recipes. The database handles relationships between all tables smoothly. I've tested every feature and it's all functional

GITHUB LINK
https://github.com/HipsterLlama1356/recipe-hub