Technical Documentation

1. Database Setup (MySQL)

For my web application, I used **MySQL** to create a database that stores all the necessary information. The database contains multiple tables, each serving a different purpose. Here's a breakdown of the tables I created:

- **User**: Stores information about each user, such as their name, email, profile picture, and any other personal data.
- **Post**: Stores posts that users create, including the title, description, images, and food type.
- **Recipe**: Stores recipes shared by users, including the title, ingredients, cooking instructions, and images.
- **Comment**: Stores comments made by users on posts or recipes.
- **Like**: Stores likes on posts or recipes, linked to the users who liked them.
- **Follow**: Tracks which users follow each other.
- **Messages**: Stores private messages exchanged between users.
- **Notifications**: Alerts users about new messages.
- **Post Rating**: Allows users to rate posts they interact with.
- **Recipe Rating**: Allows users to rate recipes they like or try.
- **User Rating**: Allows users to rate other users based on their interactions or content.
- **Save**: Tracks saved recipes by users for future reference.
- **Search**: Tracks the searches made by users for posts and recipes.
- **Share**: Records posts or recipes shared by users via links.

Each table uses **primary keys** (unique identifiers) for each entry, and **foreign keys** to connect related information across different tables. For example:

- A **comment** is connected to a specific **post** and a specific **user**.
- A **like** is associated with a **post** or **recipe** and the **user** who liked it.

2. Setting Up with Xamp (MAMP)

To work on my web application without making it public, I set up a **local server** on my computer using **MAMP**. MAMP is a software package that includes everything I need to run my website locally, including **Apache** (for the web server), **PHP** (for running the server-side scripts), and **MySQL** (for managing the database). Here's how I set it up:

- I installed MAMP on my computer and used it to create a local server environment.
- The **htdocs** folder in MAMP is where I put my project files (HTML, PHP, JavaScript, and CSS).
- I used **phpMyAdmin** (a tool within MAMP) to create and manage my database. This tool allows me to visually create tables, insert data, and run queries.
- After starting the servers in MAMP, I could open my web application by visiting `localhost` in my browser.

This setup allowed me to build and test my website without needing to put it online.

3. Building a Dynamic Web Application

A web application is one where the content updates or changes based on user actions. In my web app, the content changes when users interact with it, such as by posting a recipe, liking a post, or sending a message. Here's how I built the web app:

1. **HTML/CSS**:
   - **HTML**: I used HTML to build the basic structure of the web pages.
   - **CSS**: I applied CSS to style the pages and make them visually appealing and user-friendly.
2. **PHP**:
   - **PHP**: PHP is used to connect to the database and handle user interactions. It allows the web app to fetch real-time data from the database and display it based on the user's actions.
3. **JavaScript**:
   - **JavaScript**: I used JavaScript to add interactivity to the site. For instance, I used it to validate form inputs before submission and enhance the user experience with smoother interactions.

Key Features:

- **Create Posts**: Logged-in users can create new posts, such as recipes or updates.
- **Comments**: Users can comment on recipes or posts.
- **Likes and Shares**: Users can like and share content.
- **Instant Updates**: The content on the website updates immediately after a user performs an action, like posting a comment or liking a recipe.

This combination of HTML, CSS, PHP, and JavaScript allows for a dynamic and interactive experience, making the app responsive to user input.

4. Handling User Data and Database Interaction

Users interact with my web application through actions like adding comments or creating recipes. Here's how I manage the data they enter:

1. **Input Validation**:
   Before storing any data in the database, I make sure it's valid. For example:

- I check that a comment is not empty.
- I ensure a recipe has all the necessary fields, like the title, description, and ingredients.

2. **Database Interaction**:
   Once the input is validated, I use **PHP** to connect to the **MySQL** database and run SQL queries. Here are some of the common queries I use:

- **INSERT**: Adds new data, like a new recipe or comment, to the database.
- **SELECT**: Fetches data from the database, like getting all comments under a post.
- **UPDATE**: Changes existing data, such as updating a user's profile or editing a recipe.
- **DELETE**: Removes data, like deleting a post or comment.

3. **Security and Safety**:
   To keep the website and user data safe, I use several security measures:

- **Input sanitization**: This helps prevent malicious code (like SQL injections) from being inserted into the database.
- **Password hashing**: Instead of storing passwords in plain text, I use hashing to securely store them.

This process ensures that user data is handled correctly and securely while interacting with the database.

5. Test Cases Based on My IPO (Input-Process-Output) Chart

To make sure everything works as expected, I tested each feature of the web application. Here's a summary of the tests I ran:

Whenever a test failed, I went back and fixed the issue, then retested it until everything worked as expected.

Feature	**Input**	**Expected Output**
Create Recipe	Title, description, ingredients	Recipe saved and displayed on the recipe page
Add Comment	Comment text, recipe ID, user ID	Comment appears under the recipe
User Login	Correct username and password	Redirect to the user's dashboard
Login (Fail)	Incorrect password	Show "Incorrect password" message
Like a Post	User ID and post ID	Like count increases on the post
Follow User	User clicks "Follow" button	Button changes to "Following" and is saved
Send Message	Text message to another user	Message appears in the chat screen
Search Recipe	Keyword (e.g., "korean")	Recipes matching the keyword are shown
Rate a Recipe	Rating (1–5 stars), recipe ID	Rating saved and displayed on the recipe