**REQUIRMENTS CAPTURED**

**1. Detailed User Stories**

User stories describe the features of your application from the perspective of the end-user. They follow the format:

"As a [user role], I want to [goal], so that [benefit]."

Here's a breakdown of user stories for the core features of the recipe sharing platform:

**1.1 User Authentication**

* **Registration:**
  + As a new user, I want to create an account using my email address and password, so that I can save recipes, share my own recipes, and interact with other users.
  + As a new user, I want to have the option to register using a social media account (e.g., Google, Facebook), so that I can quickly create an account.
  + As a user, I want to receive an email verification link after registration, so that I can confirm my email address and activate my account.
  + As a user, I want to be notified if my chosen username or email is already taken, so that I can choose a different one.
* **Login:**
  + As a registered user, I want to log in to my account using my email address and password, so that I can access my saved recipes and profile.
  + As a registered user, I want to have the option to log in using a social media account, so that I can quickly access my account.
  + As a user, I want to have a "Forgot Password" option, so that I can reset my password if I forget it.
* **Profile Management:**
  + As a registered user, I want to be able to edit my profile information (e.g., username, profile picture, bio), so that I can personalize my profile.
  + As a registered user, I want to be able to change my password, so that I can keep my account secure.
  + As a registered user, I want to be able to delete my account (with a confirmation), so that I can remove my data from the platform.

**1.2 Recipe Management**

* **Create Recipes:**
  + As a registered user, I want to be able to create a new recipe by providing a title, ingredients list, step-by-step instructions, cooking time, serving size, difficulty level, and cuisine type, so that I can share my culinary creations with others.
  + As a registered user, I want to be able to upload one or more images of my recipe, so that I can make my recipe more visually appealing.
  + As a user, I want the system to automatically save my recipe as I'm creating it (draft feature), so that I don't lose my work if I accidentally close the browser.
* **Browse Recipes:**
  + As a user (registered or unregistered), I want to be able to search for recipes using keywords (e.g., "chicken", "pasta", "dessert"), so that I can quickly find recipes that match my interests.
  + As a user, I want to be able to filter recipes by categories (e.g., "vegetarian", "vegan", "Italian", "breakfast"), so that I can narrow down my search.
  + As a user, I want to be able to sort recipes by popularity (e.g., most viewed, highest rated), rating, or date added, so that I can find the best or newest recipes.
* **View Recipe Details:**
  + As a user, I want to be able to view all the details of a recipe, including the title, ingredients, instructions, cooking time, serving size, difficulty level, cuisine type, images, author, ratings, and comments, so that I can get all the information I need to cook the dish.
* **Edit/Delete Own Recipes:**
  + As a registered user, I want to be able to edit my own recipes, so that I can correct mistakes or update them.
  + As a registered user, I want to be able to delete my own recipes, so that I can remove them from the platform.

**1.3 User Interaction**

* **Rate Recipes:**
  + As a registered user, I want to be able to rate recipes (e.g., using a star rating system), so that I can express my opinion about the recipe.
* **Comment on Recipes:**
  + As a registered user, I want to be able to leave comments on recipes, so that I can ask questions, provide feedback, or share my own experiences with the recipe.
* **Save Recipes to Favorites:**
  + As a registered user, I want to be able to save recipes to my favorites list, so that I can easily find them later.
* **Follow Other Users:**
  + As a registered user, I want to be able to follow other users, so that I can see their newly created recipes and activity in my feed.

**1.4 Homepage**

* As a user, I want to see featured recipes on the homepage, so that I can discover popular or recommended dishes.
* As a user, I want to see the newest recipes on the homepage, so that I can find fresh ideas.
* As a user, I want to see popular recipes on the homepage, so that I can see what other users are enjoying.

**DB INFORMATION**

**1. Tables and Columns**

* **Users**
  + id (INT, Primary Key, Auto-increment)
  + username (VARCHAR, Unique, Not Null)
  + email (VARCHAR, Unique, Not Null)
  + password\_hash (VARCHAR, Not Null) \*We'll store a hash of the password, not the actual password for security.
  + profile\_picture\_url (VARCHAR, Null)
  + bio (TEXT, Null)
  + created\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)
  + email\_verified\_at (TIMESTAMP, Null)
* **Recipes**
  + id (INT, Primary Key, Auto-increment)
  + user\_id (INT, Not Null, Foreign Key referencing Users.id)
  + title (VARCHAR, Not Null)
  + description (TEXT, Null)
  + cooking\_time (INT, Null) \*In minutes
  + servings (INT, Null)
  + difficulty (VARCHAR, Null) \*e.g., "Easy", "Medium", "Hard"
  + cuisine (VARCHAR, Null) \*e.g., "Italian", "Mexican", "Chinese"
  + created\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)
  + updated\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP, ON UPDATE CURRENT\_TIMESTAMP)
* **Ingredients**
  + id (INT, Primary Key, Auto-increment)
  + name (VARCHAR, Not Null)
* **RecipeIngredients** (Junction Table to handle many-to-many relationship)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + ingredient\_id (INT, Not Null, Foreign Key referencing Ingredients.id)
  + quantity (VARCHAR, Null) \*e.g., "1 cup", "2 tbsp", "1/2 kg"
* **Instructions**
  + id (INT, Primary Key, Auto-increment)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + step\_number (INT, Not Null)
  + description (TEXT, Not Null)
* **RecipeImages**
  + id (INT, Primary Key, Auto-increment)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + image\_url (VARCHAR, Not Null)
* **Ratings**
  + id (INT, Primary Key, Auto-increment)
  + user\_id (INT, Not Null, Foreign Key referencing Users.id)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + rating (INT, Not Null) \*e.g., 1 to 5 stars
  + created\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)
* **Comments**
  + id (INT, Primary Key, Auto-increment)
  + user\_id (INT, Not Null, Foreign Key referencing Users.id)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + text (TEXT, Not Null)
  + created\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)
* **Favorites** (Junction Table for saving recipes)
  + user\_id (INT, Not Null, Foreign Key referencing Users.id)
  + recipe\_id (INT, Not Null, Foreign Key referencing Recipes.id)
  + saved\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)
* **Followers** (Junction Table for user following)
  + follower\_id (INT, Not Null, Foreign Key referencing Users.id)
  + following\_id (INT, Not Null, Foreign Key referencing Users.id)
  + followed\_at (TIMESTAMP, Default: CURRENT\_TIMESTAMP)

**2. Relationships**

* **One-to-Many:**
  + A User can create many Recipes.
  + A Recipe has many Ingredients (through RecipeIngredients).
  + A Recipe has many Instructions.
  + A Recipe has many Images.
  + A Recipe has many Ratings.
  + A Recipe has many Comments.
  + A User can create many Ratings.
  + A User can create many Comments.
* **Many-to-Many:**
  + A Recipe can have many Ingredients, and an Ingredient can be in many Recipes (handled by RecipeIngredients).
  + A User can save many Recipes, and a Recipe can be saved by many Users (handled by Favorites).
  + A User can follow many Users, and a User can be followed by many Users (handled by Followers).

**3. Data Types**

* INT: Integer (whole number)
* VARCHAR: Variable-length character string (text)
* TEXT: Large text field
* TIMESTAMP: Date and time
* BOOLEAN: True/False (not used extensively here, but could be added for things like email verification status)

**4. Key Concepts**

* **Primary Key:** Uniquely identifies each row in a table.
* **Foreign Key:** A column in one table that references the primary key in another table, establishing a relationship.
* **Unique:** Ensures that no two rows have the same value in that column (e.g., username, email).
* **Not Null:** Ensures that a column cannot have a null (empty) value.
* **Auto-increment:** Automatically generates a unique sequential number for the primary key.
* **Junction Table:** A table used to implement many-to-many relationships.

**TECH STACK**

**Front-End: React**

* **Why React?**
  + Component-Based Architecture: Makes it easy to build reusable UI elements (like recipe cards, forms, etc.).
  + Virtual DOM: Provides efficient updates to the user interface, resulting in a smooth user experience.
  + Large Community and Ecosystem: Abundant libraries and tools are available.
  + Widely Used: Great for your resume!
* **Key Libraries/Tools (within React):**
  + react-router-dom: For navigation between different pages (e.g., homepage, recipe details, user profile).
  + axios or fetch: For making API requests to the back-end to get and send data.
  + State Management (optional initially):
    - useState and useContext (built-in hooks) for simpler state management.
    - Redux or Zustand for more complex applications (can be added later if needed).
  + UI Library (optional):
    - Material UI or Chakra UI for pre-built, stylish components.

**2. Back-End: Node.js with Express**

* **Why Node.js and Express?**
  + JavaScript Everywhere: You'll be using JavaScript on both the front-end (React) and back-end (Node.js), which simplifies development.
  + Non-Blocking I/O: Node.js can handle many concurrent requests efficiently, which is important for a web application.
  + Express.js: A lightweight and flexible framework that makes it easy to build APIs (Application Programming Interfaces).
  + Large Community and Ecosystem: Lots of middleware and libraries are available.
* **Key Libraries/Tools (within Node.js/Express):**
  + express: The core framework for building the API.
  + cors: For handling Cross-Origin Resource Sharing (to allow your front-end to communicate with your back-end).
  + bcrypt: For securely hashing user passwords.
  + jsonwebtoken (JWT): For handling user authentication (creating and verifying tokens).
  + dotenv: For managing environment variables (like database credentials).
  + multer: For handling file uploads (for recipe images).
  + A database driver (see below).