# High-Level Design (HLD) for Tasty Recipes

### 

### **1. Architecture Overview**

* **Frontend**: React (with Tailwind CSS) for styling, enabling dynamic and responsive user interfaces for:
  + Users: Browsing, sharing, liking, commenting, and saving recipes.
  + Super Admin: Managing platform maintenance and user activities.
* **Backend**: Node.js with Express for handling API requests and server-side logic.
* **Database**: MongoDB to store user data, recipes, comments, likes, and saved recipes.
* **Cloud Services**:
  + **AWS S3**: For storing recipe images.
  + **Authentication**: JSON Web Tokens (JWT) for user authentication and role-based access control.
  + **Background Tasks**: AWS Lambda (or similar services) for asynchronous tasks, such as image optimization.

### **2. Core Components**

#### **User Interfaces**

1. **User Dashboard**:
   * Browse and search recipes.
   * Share, create, edit, delete, like, comment, and save recipes.
2. **Super Admin Panel**:
   * Monitor user activity.
   * Manage users (e.g., deactivate accounts) and ensure platform maintenance.

#### **Recipe Management**

* CRUD operations for recipes.
* Upload and store recipe images.

#### **Like and Comment System**

* Users can like recipes and add comments under them.
* Display the number of likes and comments on each recipe.

#### **Saved Recipes**

* Enable users to save favorite recipes and view them in a dedicated section.

### **3. High-Level API Design**

#### **Authentication and User Management**

* **POST /api/auth/signup**: User registration.
* **POST /api/auth/login**: User login and token generation.
* **GET /api/auth/profile**: Retrieve user profile details.

#### **Recipe Management**

* **POST /api/recipes**: Add a new recipe.
* **GET /api/recipes**: Get a list of all recipes (with search and filters).
* **GET /api/recipes/:id**: Get details of a single recipe.
* **PUT /api/recipes/:id**: Update a recipe.
* **DELETE /api/recipes/:id**: Delete a recipe.

#### **Like and Comment**

* **POST /api/recipes/:id/like**: Like a recipe.
* **POST /api/recipes/:id/comment**: Add a comment to a recipe.
* **GET /api/recipes/:id/comments**: Get all comments for a recipe.

#### **Saved Recipes**

* **POST /api/saved/:id**: Save a recipe to the user’s favorites.
* **GET /api/saved**: Retrieve all saved recipes.

#### **Super Admin**

* **GET /api/admin/users**: Manage user accounts and activities.

### **4. Data Models**

#### **User**

* userId: String (primary key)
* username, email, password: String
* role: Enum (User, Super Admin)
* savedRecipes: Array of recipe references

#### **Recipe**

* recipeId: String (primary key)
* title: String
* description: String
* ingredients: Array of strings
* instructions: String
* imageUrl: String (stored in AWS S3)
* owner: Reference to User
* likes: Array of User references
* comments: Array of Comment references

#### **Comment**

* commentId: String (primary key)
* content: String
* user: Reference to User
* recipe: Reference to Recipe

### **5. Workflow & Use Cases**

#### **Recipe Creation and Interaction**

1. A user logs in and shares a recipe with an image.
2. The backend saves recipe data to MongoDB and uploads the image to AWS S3.
3. Other users can browse recipes, like, comment, or save them.

#### **Super Admin Activity**

1. Super Admin monitors user activity and reviews reported recipes.
2. Admin manages user accounts, e.g., deactivate accounts or delete inappropriate content.

### **6. Deployment and DevOps**

#### **CI/CD Pipeline**

* Use GitHub Actions to automate build and deployment processes.
* Deploy backend and frontend on **Render** or **Vercel**.
* Manage environment variables for secrets like database URIs and JWT keys.

#### **Monitoring and Maintenance**

* Use **AWS CloudWatch** or **New Relic** for application monitoring.
* Set up alerts for high API latency or errors.

### **7. Security and Scalability**

* **Authentication**: Secure routes with JWT-based middleware.
* **Data Protection**: Encrypt sensitive user data (passwords and tokens).
* **Scaling**: Use horizontal scaling for the backend and CDN for static content delivery.

### **8. Metrics**

* Track the following:
  + Number of recipes uploaded.
  + Number of saved recipes.
  + Likes and comments per recipe.
  + User engagement over time (API usage, active users).