Creating a **Recipe Sharing Platform** using **React.js**, **Laravel/CodeIgniter**, **MySQL**, and **JWT Authentication** involves both frontend and backend development, with various phases and steps. Here's a **roadmap** to guide you through the entire process:

**1. Project Planning & Design**

* **Define the Core Features**: Identify the main features your app will have:
  + User registration, login, and authentication using JWT
  + CRUD operations for recipes (Create, Read, Update, Delete)
  + Recipe categories (vegetarian, non-vegetarian, vegan, etc.)
  + Search, filter, and sort recipes
  + Rating and commenting system for recipes
  + User profile (view/edit own profile, upload images)
  + Recipe sharing (social sharing options)
* **UI/UX Design**: Create wireframes or mockups for:
  + Home page
  + Recipe details page
  + User authentication (login/signup)
  + User profile page
  + Recipe creation page

Tools: Figma, Sketch, or Adobe XD for mockups.

**2. Set Up the Development Environment**

* **Frontend**:
  + Install Node.js and npm (Node Package Manager).
  + Initialize a new React project using create-react-app or Vite.
  + Set up required libraries like React Router, Axios for API calls, and styling libraries (e.g., Bootstrap, Material UI).
* **Backend**:
  + Install PHP and Composer (for Laravel or Codeigniter).
  + Set up a new Laravel or CodeIgniter project.
  + Set up MySQL database and configure it in your backend.

**3. Backend Development: API Creation**

**Step 1: Database Design**

* Design the MySQL database schema to handle users, recipes, ratings, and comments.
  + **Users Table**: Stores user information (name, email, password).
  + **Recipes Table**: Stores recipe details (title, ingredients, instructions, image URL, user\_id).
  + **Categories Table**: Stores recipe categories (e.g., vegetarian, vegan).
  + **Ratings Table**: Stores ratings and reviews for recipes.
  + **Comments Table**: Stores user comments for each recipe.

**Step 2: Set Up Authentication (JWT)**

* Implement JWT authentication for user login and registration.
  + Use Laravel's built-in authentication or implement JWT manually using **Laravel Passport** or **JWT-Auth** package.
  + Create routes for **signup**, **login**, and **logout** endpoints that return JWT tokens.
  + Protect routes (e.g., recipe creation, user profile) with middleware to ensure only authenticated users can access them.

**Step 3: Create API Endpoints**

* **User Authentication Endpoints**:
  + POST /api/register (for user registration)
  + POST /api/login (for user login)
  + GET /api/logout (for user logout)
* **Recipe CRUD Endpoints**:
  + POST /api/recipes (create a new recipe)
  + GET /api/recipes (fetch all recipes)
  + GET /api/recipes/{id} (get a specific recipe)
  + PUT /api/recipes/{id} (update a recipe)
  + DELETE /api/recipes/{id} (delete a recipe)
* **Recipe Rating and Comment Endpoints**:
  + POST /api/recipes/{id}/rate (submit a rating)
  + POST /api/recipes/{id}/comment (submit a comment)

**Step 4: Connect Backend to MySQL**

* Set up **Eloquent ORM** in Laravel or use **CodeIgniter’s Active Record** for database interactions.
* Create models and migrations for all necessary tables (Users, Recipes, Ratings, Comments).

**4. Frontend Development (React.js)**

**Step 1: Set Up React Project**

* Install and set up libraries:
  + react-router-dom for routing.
  + axios for making HTTP requests to your API.
  + redux or React Context API for state management.

**Step 2: Create Components**

* **Authentication**:
  + Create login, signup, and logout forms.
  + Use **JWT** to manage tokens in local storage or HTTP-only cookies for security.
* **Recipe List Page**: Display a list of recipes fetched from the backend, with options to filter by category and sort.
* **Recipe Detail Page**: Display the full details of a recipe, including instructions, ingredients, and comments.
* **Recipe Creation Page**: Form for users to create and upload recipes, including the title, ingredients, and instructions.
* **Profile Page**: A user profile page to display user details and their posted recipes.
* **Comments and Ratings**: Implement the functionality to post comments and rate recipes.

**Step 3: Connect Frontend to Backend**

* Use **Axios** to connect your frontend React app to the backend API endpoints.
  + For example, fetch recipes with axios.get('/api/recipes'), and submit new recipes with axios.post('/api/recipes', data).

**Step 4: Handle Authentication**

* Store JWT tokens in local storage or HTTP-only cookies upon login.
* Add a middleware or hook (e.g., PrivateRoute) to protect routes that require authentication.
* Ensure users can only view/edit their recipes or profile after logging in.

**5. Testing**

* **Unit Testing**: Write unit tests for backend routes and frontend components.
  + Use **PHPUnit** for Laravel or **CodeIgniter Unit Testing**.
  + Use **Jest** and **React Testing Library** for testing React components.
* **API Testing**: Test the functionality of your API endpoints using **Postman** or **Insomnia**.
* **Manual Testing**: Ensure all features work as expected, including CRUD operations, ratings, comments, authentication, and profile management.

**6. Deployment**

**Step 1: Frontend Deployment**

* Build the production-ready React app using npm run build or yarn build.
* Deploy the app to a hosting provider like **Netlify**, **Vercel**, or **AWS S3**.

**Step 2: Backend Deployment**

* Deploy the backend to a server such as **Heroku**, **DigitalOcean**, or **AWS EC2**.
* Ensure the backend is connected to a live **MySQL database** (using **RDS** on AWS or **PlanetScale** for scaling).

**Step 3: Domain & SSL**

* Set up a domain for your application and configure **SSL** for secure HTTPS connections.

**7. Post-Deployment & Maintenance**

* **Monitor performance** and fix any bugs or issues that arise.
* Add features like **user profile picture uploads**, **recipe sharing on social media**, **notifications**, etc.
* Implement **SEO** to make the recipe platform more discoverable.
* Collect user feedback and continue to improve the platform.

**/frontend**

│

├── /public # Public assets

│ ├── index.html # HTML entry point

│ └── assets # Static files (images, fonts, etc.)

│

├── /src # Main source code

│ ├── /assets # Images, icons, styles

│ │ └── logo.png

│ ├── /components # Reusable UI components

│ │ ├── Navbar.js # Navigation bar

│ │ ├── RecipeCard.js # Display individual recipe

│ │ └── Footer.js # Footer component

│ ├── /context # Global state management (if using Context API)

│ │ └── AuthContext.js # Auth context for user login state

│ ├── /hooks # Custom hooks

│ │ └── useAuth.js # Hook for authentication

│ ├── /pages # Different views or pages

│ │ ├── HomePage.js # Home page with list of recipes

│ │ ├── RecipeDetailPage.js # Recipe detail page

│ │ └── ProfilePage.js # User profile page

│ ├── /services # API call functions (to interact with backend)

│ │ ├── recipeService.js # Functions to call recipe-related APIs

│ │ └── authService.js # Functions for login, signup, etc.

│ ├── /styles # Global styles and themes

│ │ ├── global.css # Global CSS for the app

│ │ └── theme.css # Theme styles (colors, fonts, etc.)

│ ├── App.js # Main app component

│ ├── index.js # React entry point

│ └── router.js # Route definitions using React Router

│

├── .env # Environment variables (e.g., API URL)

├── package.json # Frontend dependencies and scripts

└── README.md # Frontend project documentation│

└── .gitignore # Git ignore settings (for both frontend and backend)

**/backend**

│

├── /app # Main application logic

│ ├── /Http

│ │ ├── /Controllers # Controllers for API logic

│ │ │ ├── RecipeController.php

│ │ │ └── AuthController.php

│ │ ├── /Middleware # Middleware for JWT authentication

│ │ │ └── Authenticate.php

│ │ └── /Requests # Validation for incoming requests

│ │ └── RecipeRequest.php

│ ├── /Models # Eloquent models for DB interaction

│ │ ├── User.php

│ │ ├── Recipe.php

│ │ └── Rating.php

│ ├── /Providers # Service providers for app configuration

│ │ └── AuthServiceProvider.php

│ ├── /Services # Business logic (optional)

│ └── /Database

│ ├── /Migrations # Database migrations

│ │ ├── create\_recipes\_table.php

│ │ └── create\_users\_table.php

│ └── /Factories # Data factories for testing

│

├── /config # Configuration files (JWT, DB, etc.)

│ ├── app.php

│ ├── database.php

│ └── auth.php

│

├── /public # Public assets and entry point

│ ├── index.php # Main entry point for Laravel

│ ├── /uploads # Recipe images or files

│ └── /css, /js # Static CSS/JS files (for frontend)

│

├── /routes # Define routes

│ ├── web.php # Web routes (for frontend)

│ └── api.php # API routes (for the Recipe app)

│

├── /storage # Logs, cached files, and uploads

│ ├── /app # Application storage

│ ├── /framework # Framework-generated files

│ └── /logs # Application logs

│

├── .env # Environment variables (e.g., DB settings, JWT secret)

├── composer.json # Backend dependencies

└── README.md # Backend project documentation