Recipe Management System

Name: Vignesh J

Date: 20-03-2025

Batch: Cohort - C3

Github URL: https://github.com/Vignesh025/Recipe-Management-System

Table Of Content

Problem Definition and Objectives

Problem Statement:

A web app for users to add, edit, and delete recipes with ingredients, instructions, and categories.

User Story	Description	Acceptance Criteria
Add Recipe	Add recipes with ingredients and instructions.	Store in MS SQL, link to user.
Recipe Listing	View list of recipes.	Display name, image, category, fetch by user.
Edit Recipe	Modify existing recipes.	Validate fields before updating.

Project Objectives:

1. Create a Comprehensive Recipe Management System
    - Develop a platform where users can create, view, edit, and delete their recipes
    - Implement a user-friendly interface for recipe management
2. Frontend Development
    - Create responsive UI components for recipe display and management
    - Implement state management using Redux
3. Backend Development
    - Build RESTful API endpoints for CRUD operations
    - Implement data validation and error handling
    - Set up database models and migrations
4. Security and Performance
    - Secure API endpoints with authentication
    - Implement error handling throughout the application

Frontend & Backend Architecture

Technology Stack Overview:

Frontend:

Framework: React.js

State Management: Redux with Redux Toolkit

UI Components: React Bootstrap

HTTP Client: Axios

Routing: React Router

Backend:

Framework: ASP.NET Core Web API

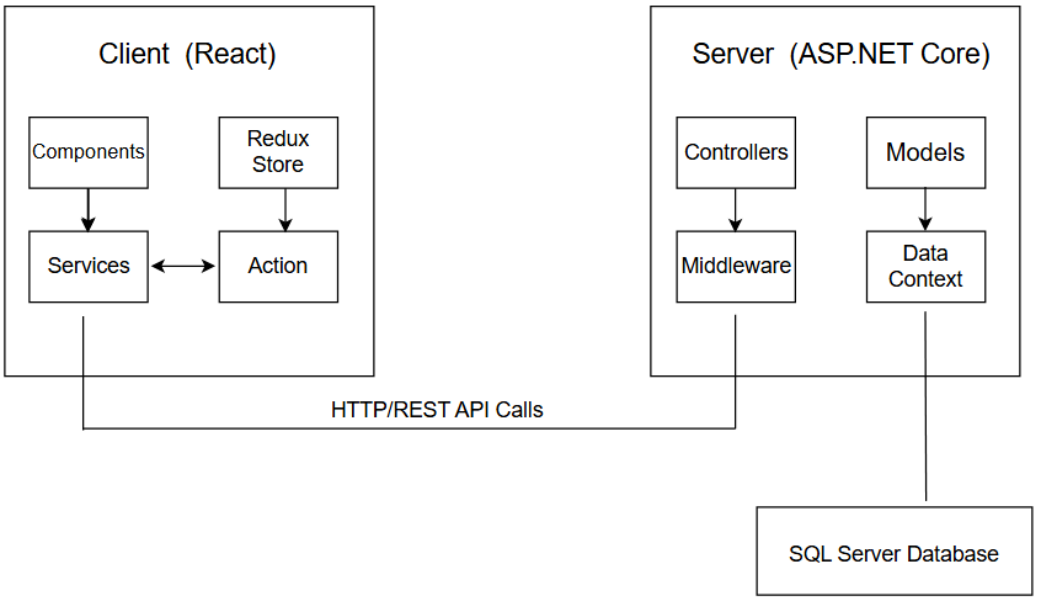
Language: C#

Database: SQL Server (configured in Program.cs)

ORM: Entity Framework Core

Authentication: Custom token-based authentication

System Design:



Frontend Architecture

1. Components Layer
    - Recipe-related components (RecipeCard, Recipe, AddRecipeModal, EditRecipeModal, DeleteConfirmModal etc.)
    - Authentication components (Login, Register)
    - Page components (HomePage, etc.)
2. State Management
    - Redux store configured in store.js
    - Authentication state managed in authSlice.js
    - Recipe data managed in recipeSlice.js
3. Services Layer
    - api.js: Base API configuration with Axios, including token handling
    - authService.js: Authentication-related API calls
    - recipeService.js: Recipe-related API calls

Backend Architecture

1. Controller Layer
    - AuthController: Handles user registration and login
    - RecipesController: Manages CRUD operations for recipes with authorization
2. Model Layer
    - User: Represents user data
    - Recipe: Represents recipe data with relationships to users
    - RecipeDTO: Data transfer object for recipes
3. Data Access Layer
    - MyAppContext: Entity Framework DbContext for database operations
4. Authentication System
    - BasicAuthenticationHandler: Custom authentication handler for token validation
    - TokenStore: In-memory storage for user tokens

Component Breakdown & API Design

Frontend Component Breakdown:

State Management:

The application uses Redux with Redux Toolkit for centralized state management:

1. Store Configuration (store.js):
    - Configures the Redux store with multiple reducers
    - Combines recipe and authentication state
2. Authentication Slice (authSlice.js):
    - Manages user authentication state
    - Handles login, registration, and logout actions
    - Stores user data and authentication token
    - Provides actions for error handling and registration success
3. Recipe Slice (recipeSlice.js):
    - Manages recipe data and CRUD operations
    - Handles loading states for different operations
    - Provides error handling for each operation
    - Implements async thunks for API interactions

Routing:

The application uses React Router for navigation:

1. Route Configuration (App.js):
    - Defines routes for different pages
    - Main routes include:
        - ➔ / - Home page
        - ➔ /recipes - All Recipes page
        - ➔ /recipes/my - My Recipes Page
        - ➔ /login - Login page
        - ➔ /register - Registration page

2. Navigation Logic:
    - Programmatic navigation using useNavigate hook
    - Authentication-based redirects (e.g., redirecting authenticated users from home to recipes)

UI Components:

Page Components:

1. HomePage:
    - Landing page with rotating background images
    - Call-to-action buttons for login/register
    - Redirects authenticated users to the recipes page
2. RecipePage:
    - Main dashboard for authenticated users
    - Displays all users recipe collection
    - Contains navigation header and footer
3. MyRecipePage:
    - Displays user's recipe collection
    - Contains navigation header and footer
4. LoginPage & RegisterPage:
    - Container pages for authentication forms
    - Simple layout with navbar and footer

Core Components:

1. Authentication Components:
    - Login.jsx: Form for user login with validation
    - Register.jsx: Form for user registration with validation
2. Recipe Management Components:
    - RecipeList.jsx: Container for displaying recipe cards
    - RecipeCard.jsx: Card display for individual recipes
    - Recipe.jsx (RecipeDetailModal): Modal for viewing recipe details
    - MyRecipeList.jsx: Container for displaying recipe cards of the current user

- MyRecipeCard.jsx: Card display for individual recipes of the current user
- MyRecipe.jsx (RecipeDetailModal): Modal for viewing recipe details of the current user
- AddRecipeModal.jsx: Form for creating new recipes
- EditRecipeModal.jsx: Form for updating existing recipes
- DeleteConfirmModal.jsx: Caution popup for deleting recipes

3. Shared UI Elements:
    - Bootstrap components (Button, Card, Modal, Form, etc.)
    - React Bootstrap Icons for visual elements

Service Layer:

1. API Service (api.js):
    - Configures Axios with base URL and headers
    - Implements request interceptor for token authentication
    - Implements response interceptor for error handling
2. Authentication Service (authService.js):
    - Handles user registration, login, and logout
    - Manages token storage in localStorage
3. Recipe Service (recipeService.js):
    - Implements CRUD operations for recipes
    - Handles image URL validation and processing

API Design:

Endpoints:

Authentication Endpoints:

1. Register User:
    - POST /api/Auth/register
    - Payload: { username, password }
    - Response: Success message

2. Login User:
    - POST /api/Auth/login
    - Payload: { username, password }
    - Response: { userId, username, token, message }

Recipe Endpoints:

1. Get All Recipes:
    - GET /api/Recipes
    - Authentication: Required
    - Response: Array of recipe objects
2. Get My Recipes:
    - GET /api/Recipes/my
    - Authentication: Required
    - Response: Array of recipe objects
3. Get Single Recipe:
    - GET /api/Recipes/{id}
    - Authentication: Required
    - Response: Recipe object
4. Create Recipe:
    - POST /api/Recipes
    - Authentication: Required
    - Payload: Recipe object
    - Response: Created recipe
5. Update Recipe:
    - PUT /api/Recipes/{id}
    - Authentication: Required
    - Payload: Updated recipe object
    - Response: No content (204)
6. Delete Recipe:
    - DELETE /api/Recipes/{id}
    - Authentication: Required
    - Response: No content (204)

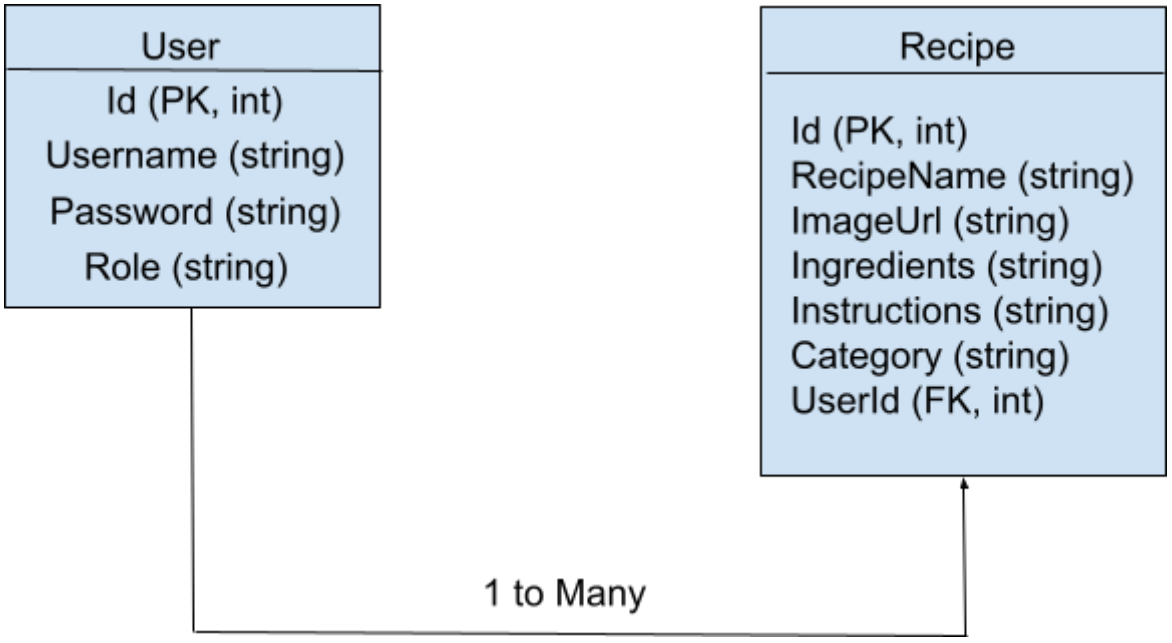
Authentication Mechanism:

The application uses a custom token-based authentication system:

1. Token Generation:
    - When a user logs in, the server generates a Base64-encoded GUID as a token
    - The token is associated with the user's ID in an in-memory TokenStore
    - The token is returned to the client
2. Token Storage:
    - Frontend stores the token in localStorage
    - Token is included in the Authorization header of subsequent requests
    - Format: Bearer {token}
3. Authentication Handling:
    - BasicAuthenticationHandler intercepts requests with Authorization headers
    - Extracts the token and looks up the associated user ID in TokenStore
    - Creates a ClaimsPrincipal with user claims if token is valid
    - Controllers use the [Authorize] attribute to require authentication
4. Authorization Logic:
    - Controllers extract the authenticated user's ID from claims
    - Recipe endpoints enforce ownership
    - Users can only manage their own recipes but can view all the recipes
    - Admins can manage all the recipes.
    - Appropriate status codes (403 Forbidden) are returned for unauthorized access
5. Token Invalidation:
    - Frontend removes the token from localStorage on logout
    - Backend has no explicit token invalidation (tokens remain in TokenStore)

Database Design

Entity-Relationship Diagram (ERD):



Relationship Details:

1. **One-to-Many Relationship:** A User can have multiple Recipes, but each Recipe belongs to exactly one User.
2. **Foreign Key:** The UserId field in the Recipe table references the Id field in the User table.
3. **Cascade Delete:** When a User is deleted, all associated Recipes are automatically deleted (configured with onDelete: ReferentialAction.Cascade).

Current Database Schema:

Based on the migration files, the database schema is implemented as follows:

Users Table

- **Id**: Integer, Primary Key, Auto-increment
- **Username**: nvarchar(max), Required
- **Password**: nvarchar(max), Required (stores SHA-256 hashed passwords)

- **Role**: nvarchar(max), Required (By default: “user”)

Recipes Table

- **Id**: Integer, Primary Key, Auto-increment
- **RecipeName**: nvarchar(max), Required
- **ImageUrl**: nvarchar(max), Nullable
- **Ingredients**: nvarchar(max), Required
- **Instructions**: nvarchar(max), Required
- **Category**: nvarchar(max), Required
- **UserId**: Integer, Foreign Key to Users.Id, Required

Project Flow

Github URL: https://github.com/Vignesh025/Recipe-Management-System

➔ Open two terminals: one for the frontend and one for the backend.

➔ In the first terminal, navigate to the **client** folder and run:

```
npm start
```

- This will start the frontend.

In the second terminal, navigate to the **server** folder and run:

```
dotnet run
```

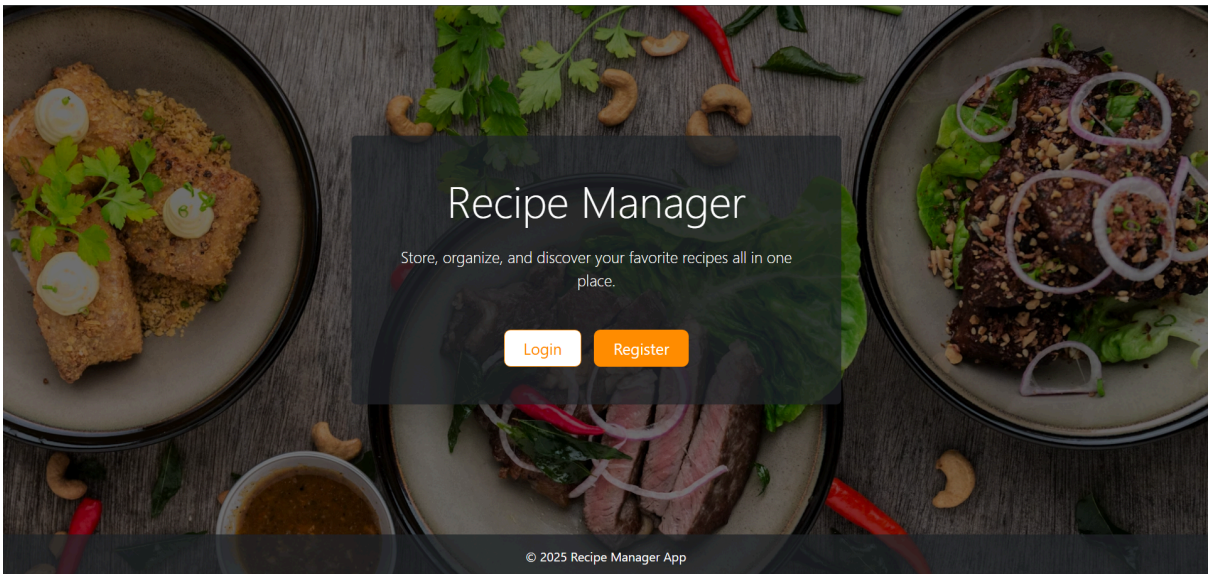
- This will start the backend.

Home Page:

➔ At first, you'll be on the homepage, where you'll find options for login and registration.

➔ If you're already a registered user, click the **Login** button to go to the login page.

➔ If you're a new user, click the **Register** button to go to the registration page.



Register Page:

➔ On the registration page, you need to enter a username and password.

➔ The username must be unique.

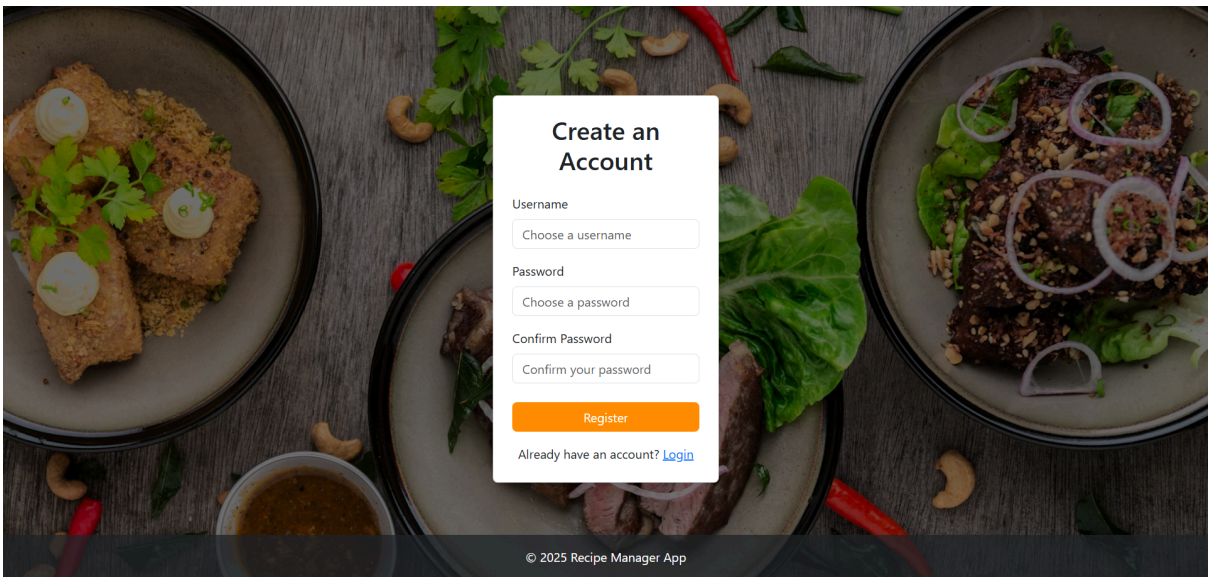
➔ The password and confirm password fields must match.

➔ By default, all registered users will have the role of "user."

➔ An admin role can be assigned to a user through the database.

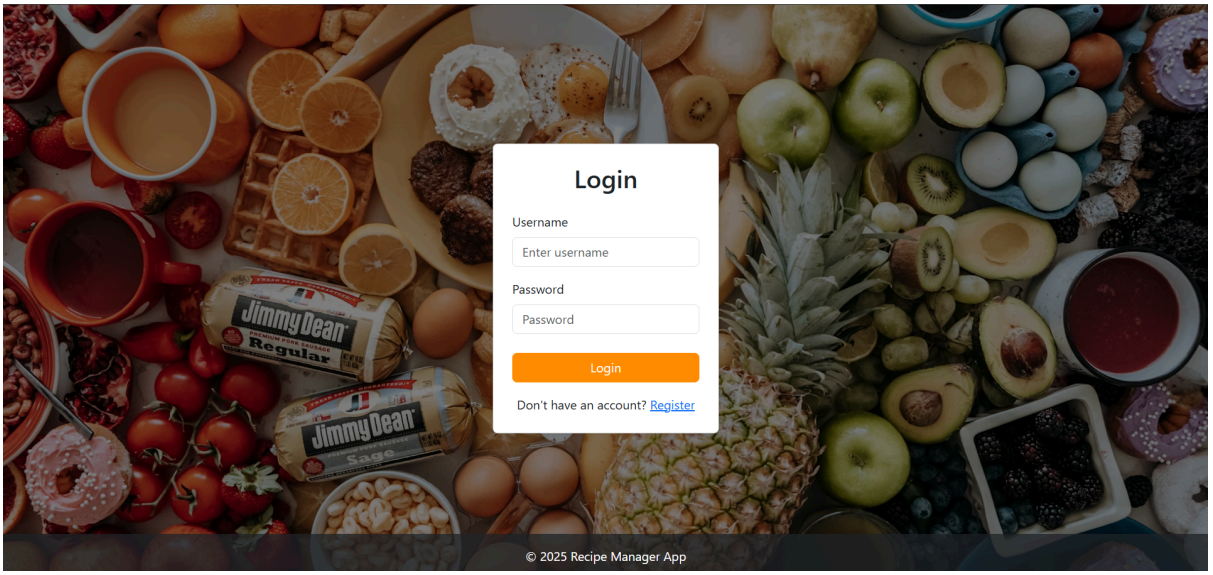
(Update users set role='admin' where id=*user_id)*

➔ After registering, you'll be redirected to the login page.



Login Page:

- ➔ On the login page, enter your username and password, then click the **Login** button.
- ➔ If the credentials are correct, you'll be redirected to the **All Recipes** page.



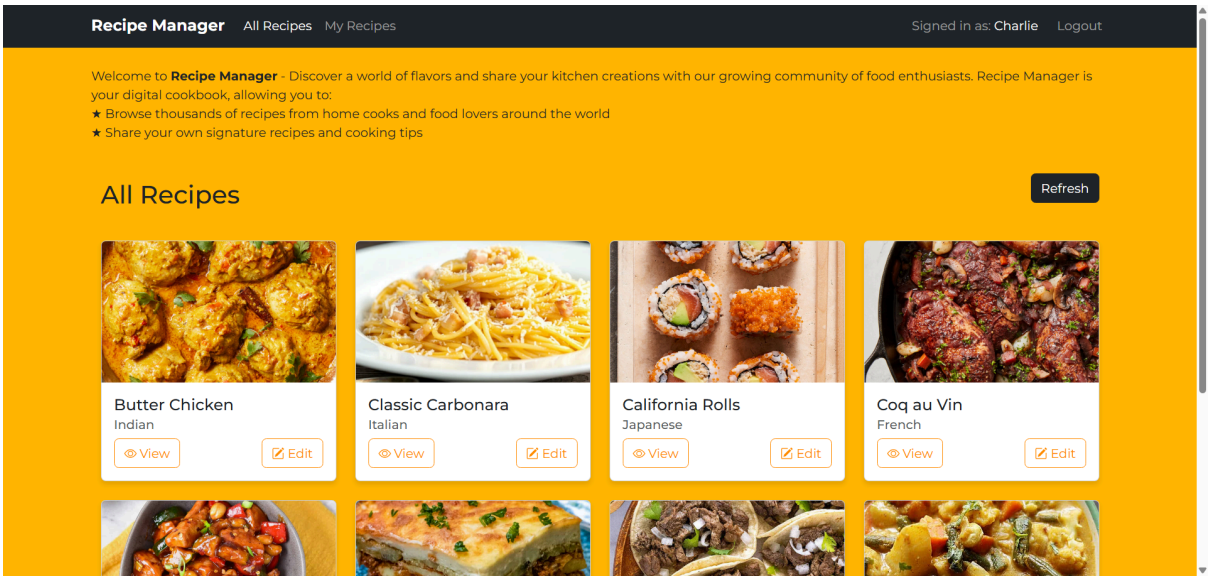
All Recipes Page:

- ➔ On the **All Recipes** page, you can view recipes added by all users.
- ➔ The header contains a navigation bar with links to the **All Recipes** page and the **My Recipes** page.
- ➔ It also displays the currently signed-in user and provides a **Logout** option.

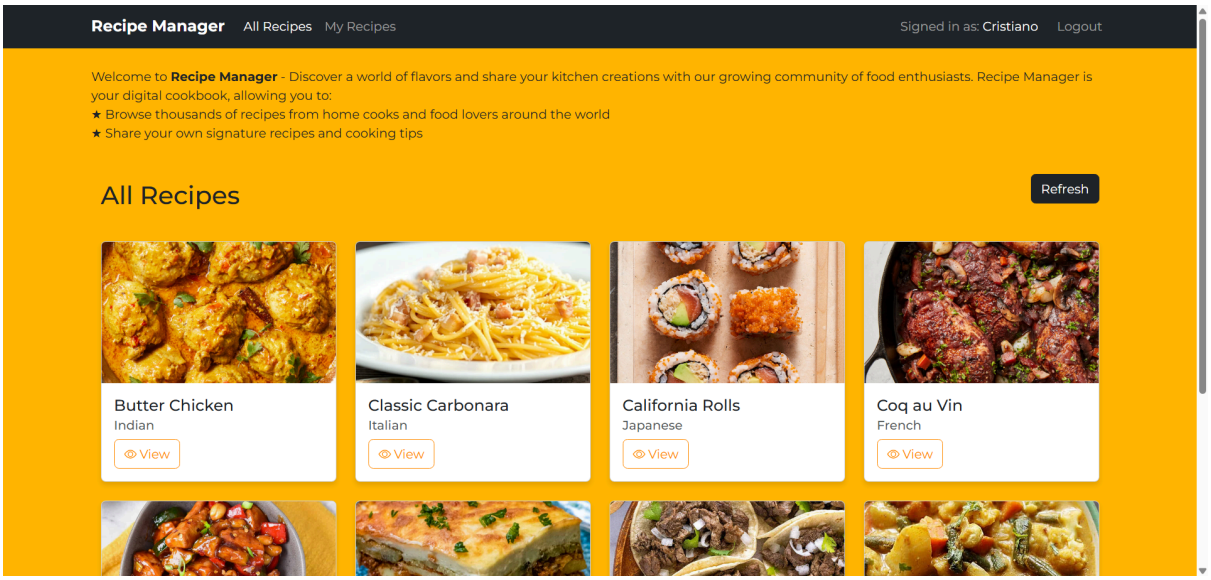
➔ You can refresh the list using the **Refresh** button.

➔ This page appears differently for users and admins:

- **Admins** have options to edit and delete any recipe.



- **Users** can only manage their own recipes on the **My Recipes** page.



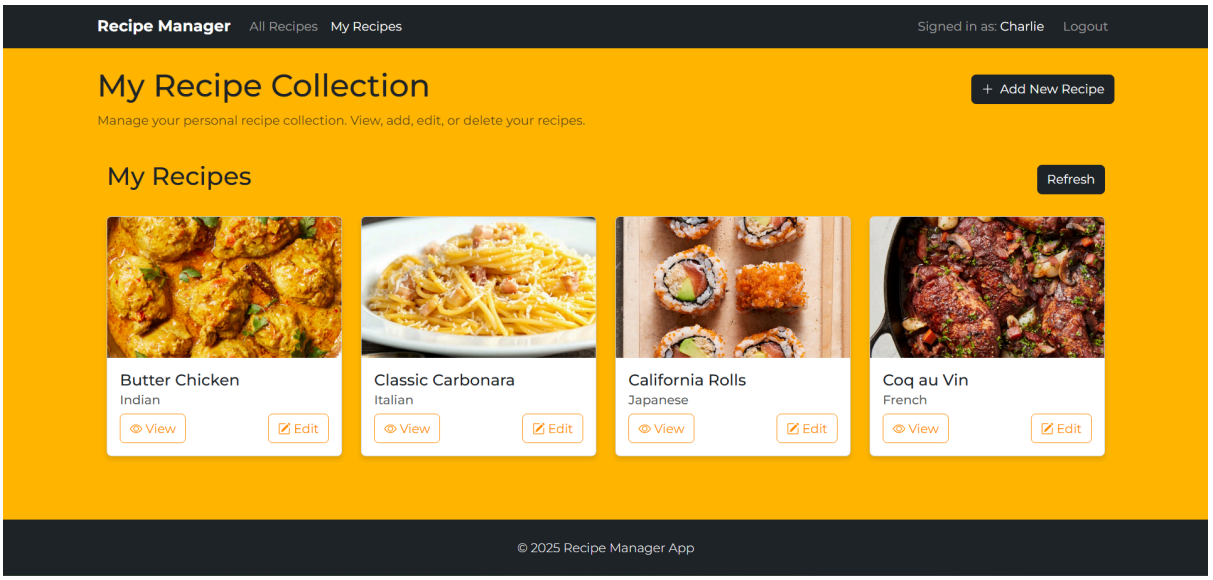
My Recipes Page:

➔ If you click on the **My Recipes** page in the navbar, you'll be redirected to the **My Recipes** page.

➔ On this page, you can view all the recipes you've added.

➔ You can **view, edit, and delete** your recipes here.

➔ To add a new recipe, click the **Add New Recipe** button.



Add New Recipe:

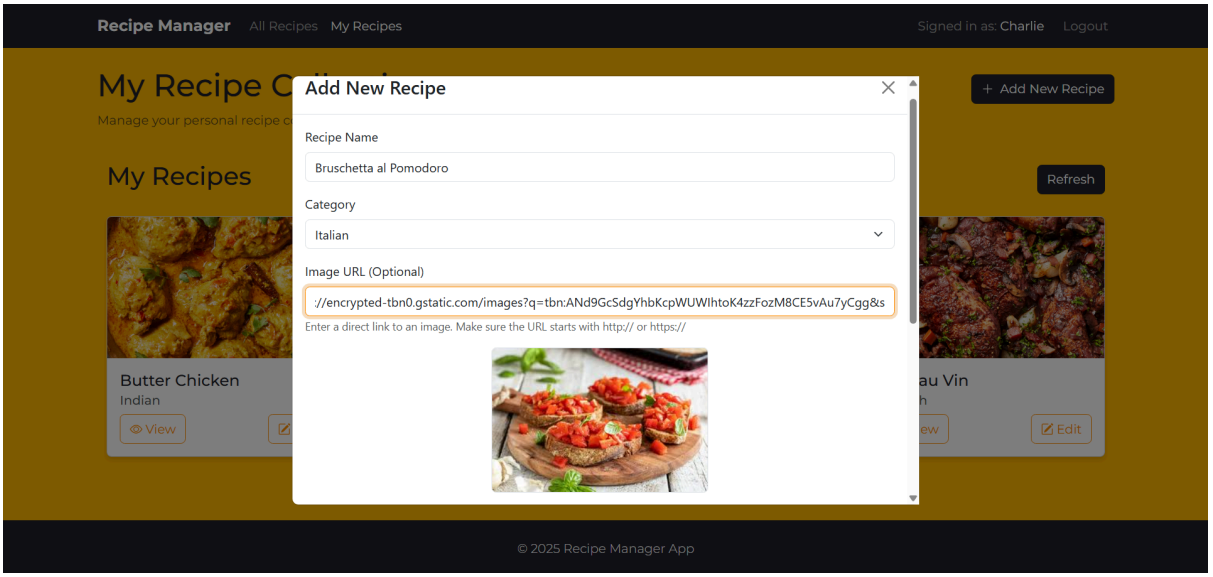
➔ When you click the **Add New Recipe** button, a **form modal** will be displayed.

➔ In the form, you need to fill in:

- Recipe name
- Category
- Image link
- Ingredients
- Instructions

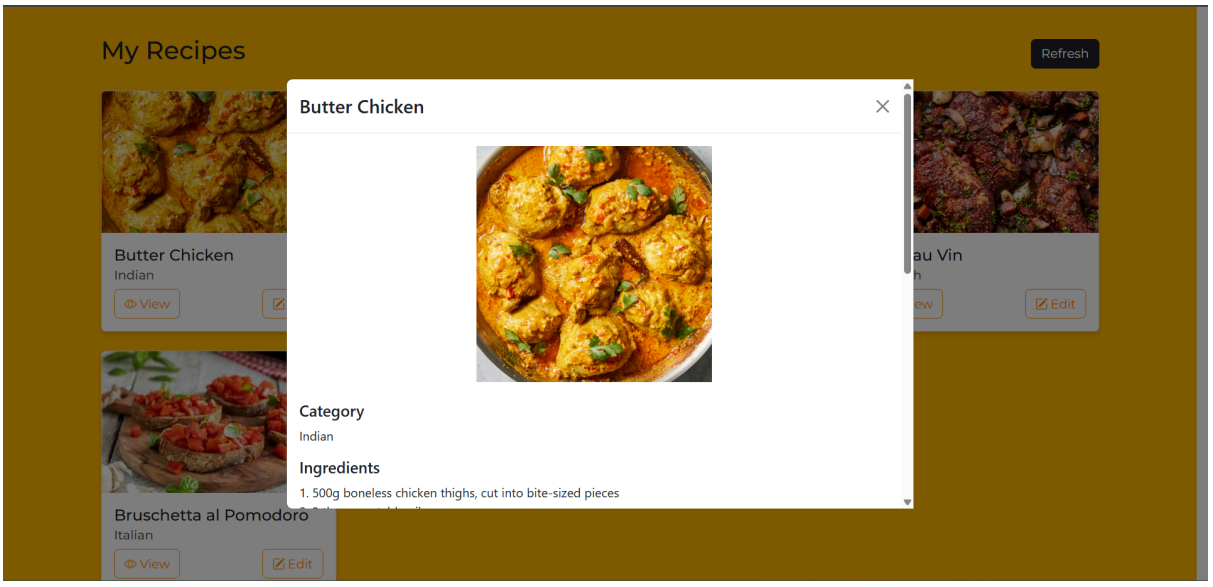
➔ After entering the image link, a **preview** of the image will be shown to help you verify if it's supported.

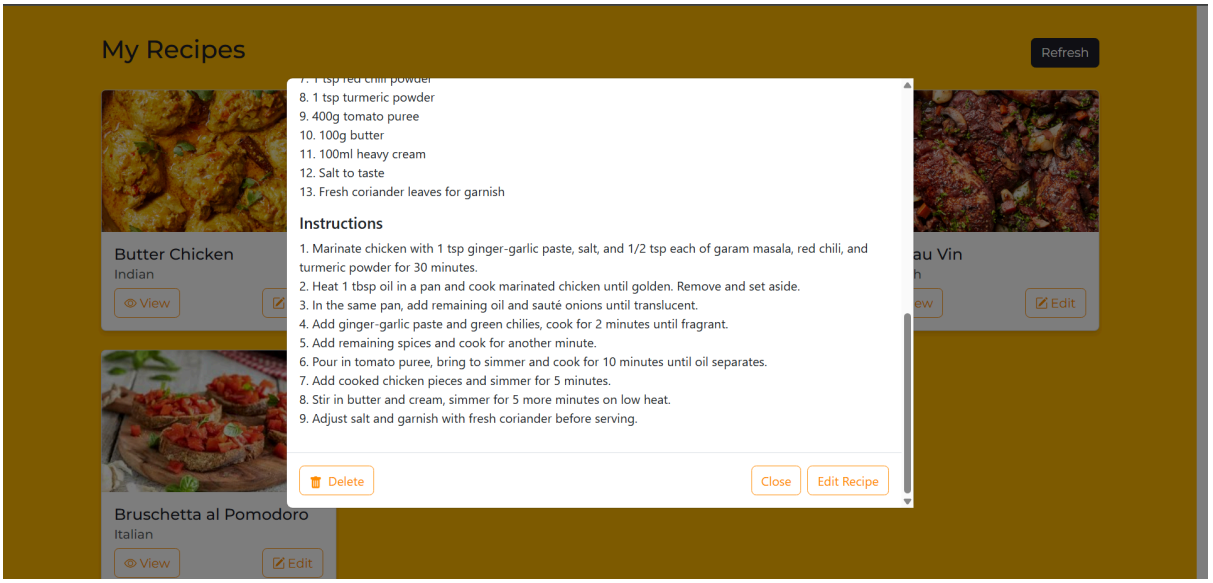
➔ Once all details are filled in, click **Add Recipe** to add the recipe.



View Recipe:

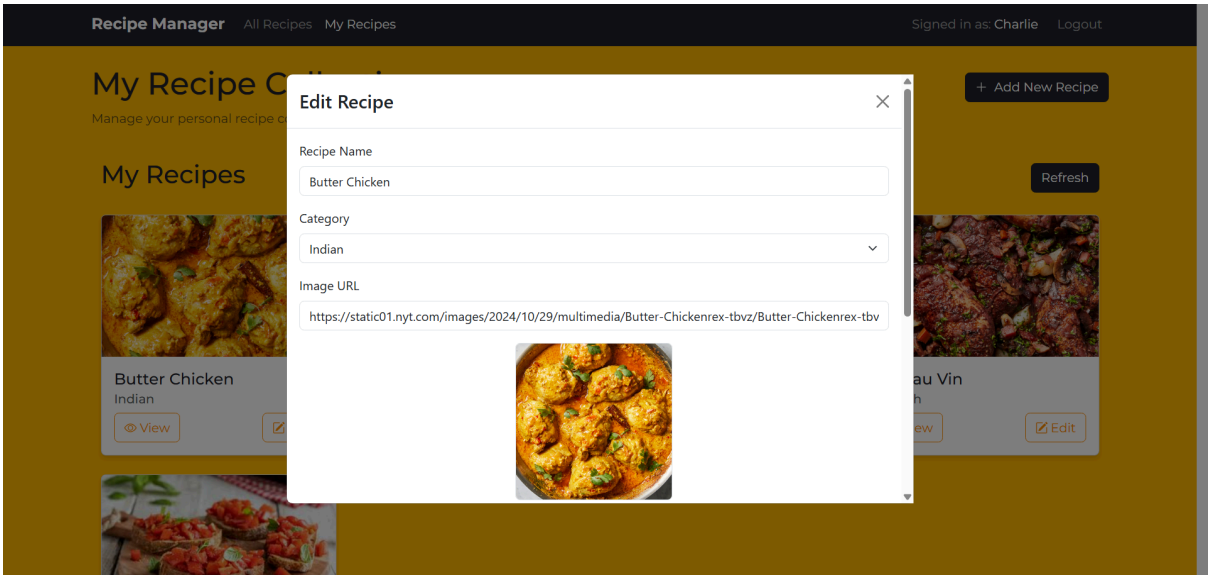
- ➔ When you click the **View** button on a recipe card, a **view modal** will be displayed.
- ➔ In the view modal, you'll see:
  - Recipe name
  - Image
  - Category
  - Ingredients
  - Instructions
- ➔ At the bottom, there will be options to **Edit, Delete,** and **Close** the modal.

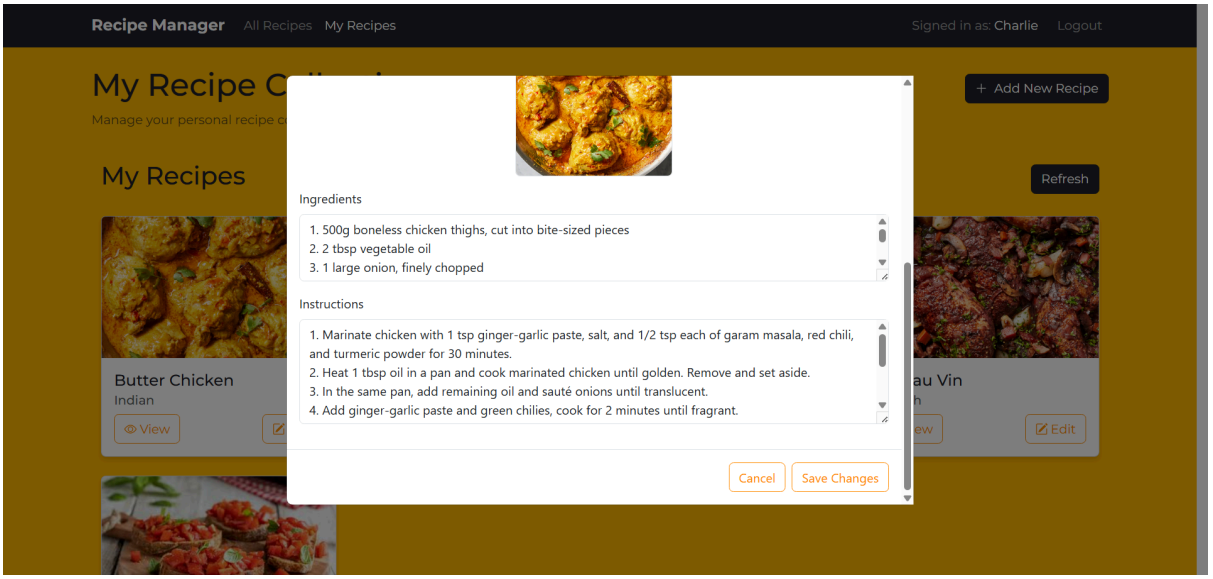




Edit Recipe:

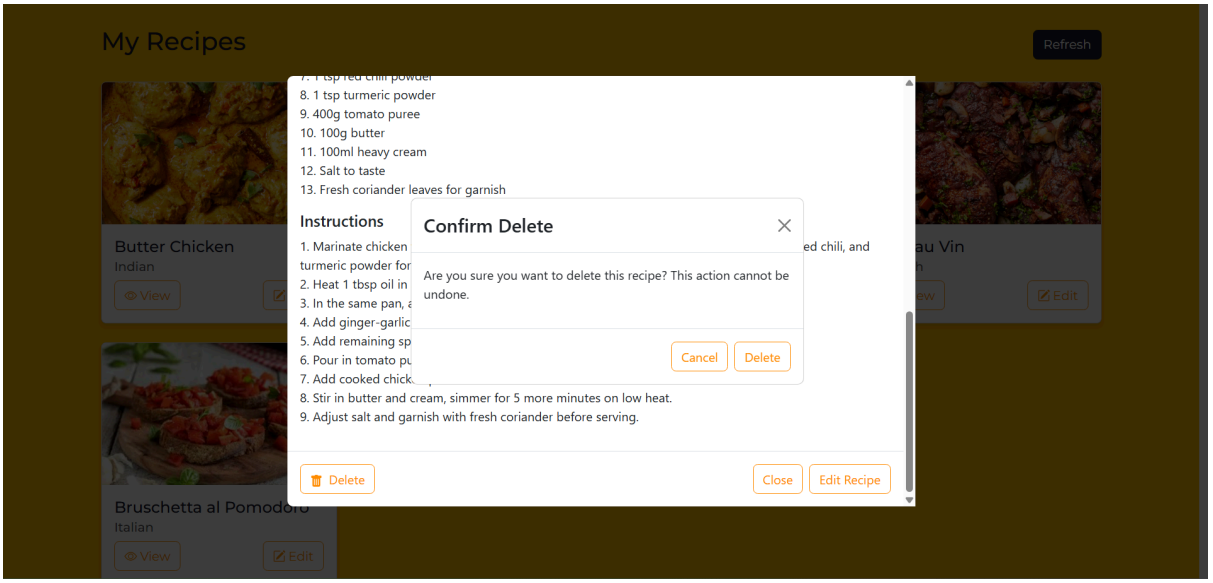
- ➔ When you click the **Edit** button, an **edit modal** will appear.
- ➔ This modal contains a form pre-filled with the recipe details.
- ➔ You can modify the details and click **Save** to update the recipe.
- ➔ The **Edit** option is available only to **admins** and the **user who created the recipe**.





Delete Recipe:

- ➔ When you click the **Delete** button, a **delete modal** will pop up.
- ➔ The modal will warn that the process is **irreversible** and ask, **"Are you sure you want to delete it?"**
  - Clicking **OK** will delete the recipe.
  - Clicking **Cancel** will keep the recipe.
- ➔ This option is available only to **admins** and the **user who created the recipe**.



Logout:

➔ There is a **Logout** option in the header.

➔ Clicking **Logout** will log you out and redirect you to the **Login** page.