Detailed Project Proposal

SOUS-CHEF : at your service

By Utkarsh Pratap Singh - 200050146, Aditya Kadoo - 200050055, Prerak Meshram - 200050110 & Mundre Sthavir Dharmpal - 200050080

Overview of Application

Sous chef is a recipe-tracking web app that helps users manage and share their personal recipes and also browse new recipes made by other users. It also comes with an inventory feature to track available ingredients for the user. Users can search for a recipe based on keywords, ingredients, tags, and other filters. The app can help users to plan meals based on dietary restrictions and ingredient availability. Users can also generate grocery lists by referencing their inventory and meal plans.

Users

There is only one class of users in our app. All users, once signed in, can add their recipes or view recipes of other users and rate those recipes and filter recipes based on tags, specific user, minimum rating, highest rating, maximum time for making, hot recipes in the past hour, newest recipes and controversial recipes .

Data to be stored

1. **Chef's Table**: Contains
    a. **Chef_ID**
    b. **Name**
    c. **Password**

2. **Recipes Table**: Contains
    a. **Recipe_ID**
    b. **Title**
    c. **Serves**
    d. **Last Modified Timestamp**
    e. **Duration Interval**
    f. **Visibility**
    g. **Foreign key Chefs: AuthorId**

3. **Ingredients Table**: Contains
    a. **Ingredient_ID**
    b. **Name**
    c. **Kind**
4. **Tags Table**: Contains
    a. **Tag_ID**
    b. **Name**
5. **Steps Table**: Contains
    a. **Recipe_ID**
    b. **Description**
    c. **Step Number**
    d. **Foreign Key Recipes: Owner**
6. **Shopping List Relation**: Contains
    a. **Chef_ID**
    b. **Ingredient_ID**
    c. **Quantity**
7. **Requirements Relation**: Contains
    a. **Recipe_ID**
    b. **Ingredient_ID**
    c. **Quantity**

8. **Tagged Relation**: Contains
    a. **Recipe_ID**
    b. **Tag_ID**

9. **Bookmarks Relation**: Contains
    a. **Recipe_ID**
    b. **Chef_ID**
10. **Ratings Relation**
    a. **Recipe_ID**
    b. **Chef_ID**
    c. **Rating**
    d. **Last Modified Timestamp**

Application Structure

Backend

- POST /login [Login User]
- POST /signup [Create User]
- GET /auth [Check Authorisation]
- GET /logout [Logout User]
- GET /chef/{id} [Get Chef by id]
- GET /chef [Get Chefs list by query]
- POST /recipe [Create recipe]
- GET /recipe/{id} [Get recipe by id]
- POST /recipe/{id} [Update recipe by id]
- GET/step /{id}/{step} [Get step of a recipe]
- DELETE /recipe/{id} [Delete recipe by id]
- POST /recipe/shop/{id} [Add all ingredients of recipe to shopping list by id]
- GET /recipe [Get recipes list]
- GET /recipes [Get recipe names by query]
- POST /ingredient [Create ingredient]
- GET /ingredient [Get ingredients list by query]
- POST /tag [Create Tag]
- GET /tag [Get tags list by query]
- DELETE /bookmark/{id} [Delete bookmark]
- POST /bookmark/{id} [Add bookmark]
- DELETE /rating/{id} [Delete rating]
- POST /rating/{id} [Add rating]
- GET /shoppinglist [Get user shopping list by id]
- POST /shoppinglist/{id} [Create ingredient in shopping list by id]
- DELETE /shoppinglist/{id} [Delete ingredient in shopping list by id]

UI

We built an android application for our project . Our tech stack comprises of:

1) **Flutter** for building a charming UI
2) **NodeJS** for backend api calls
3) **ExpressJS**
4) **PostgreSQL**(Relational Database) for the Database
5) **Render** for hosting web service and postgres database.

Login Page

Initial page where :

1.Users will enter their credentials to log into the application.

2.Unauthorized users cannot manage their private recipes, inventory and grocery shopping list.

Signup Page

1.Create user with userid, password and name.

Home Page

1.Default page for any user where they can access their profile.

2.User's public and private recipes and bookmarked recipes are also shown.

3.Options for adding and editing the user's recipes.

Grocery Shopping List Page

This page shows the :

1.Ingredients to be bought by the user and the corresponding quantities.

2.User can add new ingredients and edit existing ingredients.

Discover Page

Here the users can,

1. Browse through recipes posted by all the users.

2. Filter it by using tags for cuisines , ratings or authors.

3. Filter/sort recipes based on tags, specific user, minimum rating, highest rating, maximum time for making, hot recipes in the past hour, newest recipes and controversial recipes .

Add/Edit Recipe Page

Here a user can :

1. add/edit their own recipe.
2. add/edit steps and ingredients present in the recipe.
3. Simple interface where the user needs to add simple instructions describing the recipe and the data is stored systematically in the database.

Recipe Page

This page shows:

1. all the relevant information of a recipe.
2. It lists all the ingredients and their quantities required for this recipe.
3. Users can also update their shopping list according to the requirements of the recipe directly.

Follow recipe

1. Will show the steps sequentially for the user to follow along with the option to go to the next or previous step.
2. Each step will show a timer for the duration required to execute the step.
3. At the end the user will have an option to rate the recipe.

Other aspects

For session management, we'll be following some protocols such as:

- We'll be using server side cookies for login sessions using **Express-sessions**
- Reasonable session Timeouts to maintain privacies as well as ample time to avoid frequent session expirations and authorization of user's credentials at each request
- Implementing access controls for sensitive information or preventing unauthorized access.
- Using **hash functions**, **salts** to encrypt the session IDs and passwords to make it attack resistant.

Future Work

If we had more time we would have liked to implement these also :

- We'll be using server side cookies for login sessions using **Express-sessions**
- Reasonable session Timeouts to maintain privacies as well as ample time to avoid frequent session expirations and authorization of user's credentials at each request
- Implementing access controls for sensitive information or preventing unauthorized access.
- Using **hash functions**, **salts** to encrypt the session IDs and passwords to make it attack resistant.

Our Learning