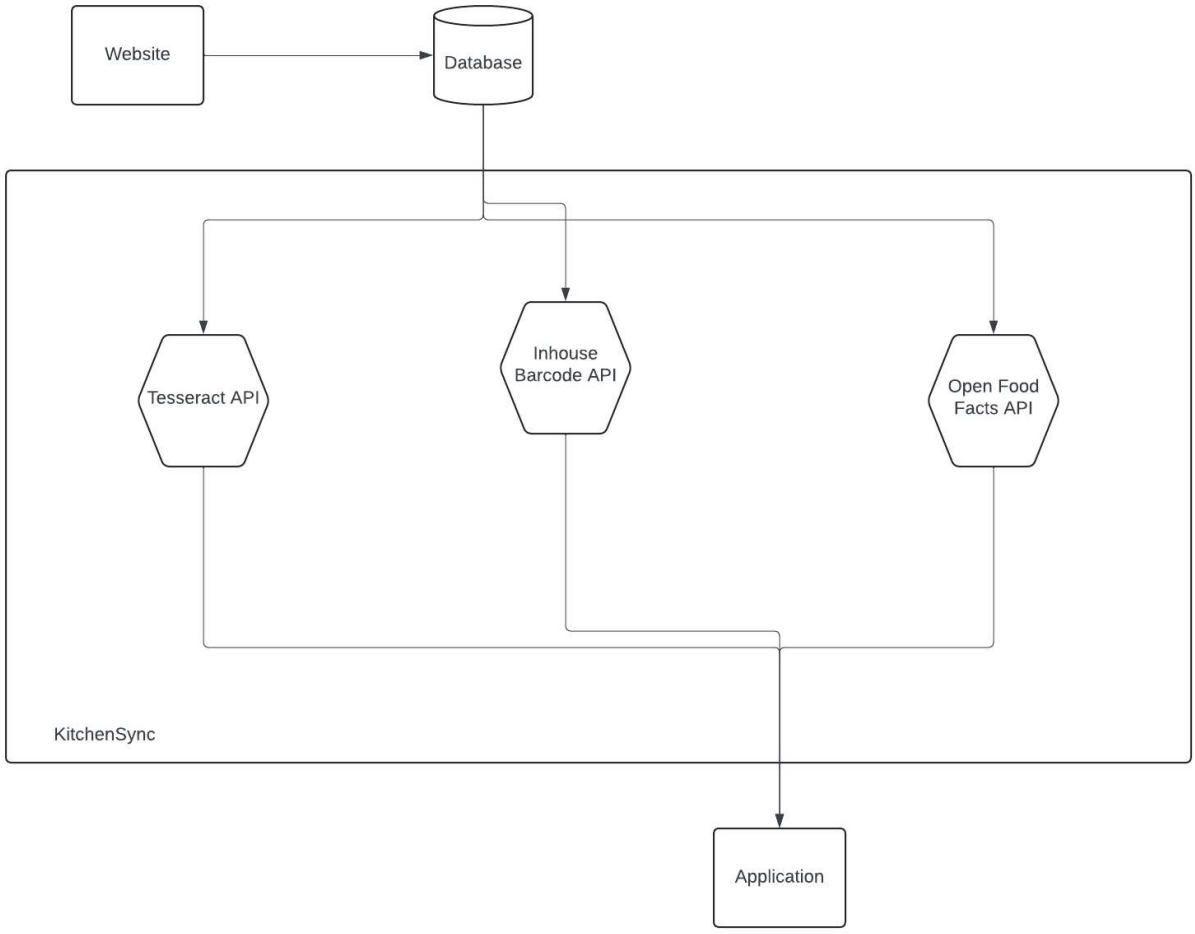
1. UML Diagram



2. Functionality and Interface

2.1 Ingredient

The ingredient class is the class that will store the data that is present in an ingredient. Attributes such as name, specific labels and nutritional information can be obtained from the Open Food Facts tool, while information such as quantity and expiration date come from the user input. The user is able to update the ingredients, whether it is through adding new ingredients and their respective details, or upgrading an ingredient's details based on certain criteria. The user is also able to move the location of an ingredient if it has been moved in real life, for better organization.

2.2 Nutritional Info

The nutritional info class stores all the nutritional information tied to the ingredient it belongs to. The user can update the ingredients nutritional info by grabbing the ingredients ID, and the user can also retrieve its nutritional info for other methods to use. Populating the nutritional information class will be handled through data

retrieved from the Open Food Facts tool. Since each ingredient will have its own nutritional values, it's nutritional information will be tied to its respective ingredient.

2.3 Recipe

The recipe class handles recipes (both those initially seeded and those that are created by the users) and their contents. The class handles the name of the recipe, the list of ingredients and steps, specific tags for easier handling, and a value for complexity, in addition to its rating, serving size, equipment needed to cook, times to cook and prepare, as well as images of the recipe. The user can create new recipes with an inbuilt recipe method, as well as deleting or updating recipes that they have made.

2.4 Meal Planner

The meal planner class handles the planning of meals up to two weeks in advance. It will contain a map of meals associated with dates and IDs that the user assigns to it. In addition the user can move around the dates in which meals occur, as well as add or remove meals from the map if desired. When the meals are generated it will also generate a shopping list containing the ingredients needed to make the meals.

2.5 Shopping List

The shopping list class manages the creation of shopping lists based on meal plans and user preferences. There will be a method to create shopping lists based on the meal plan as well as the list of ingredients currently in your inventory, as well as methods to optimize shopping and its cost. Methods to get ingredients from the fewest stores possible, and to compare prices between stores that fall within user preferences will optimize the cost and time for the end user.

2.6 Admin

The admin class is responsible for managing users and their content, such as recipes and reviews. It will have methods for flagging content of a user for review by admins, and another method to remove content if it violates any terms of service. In addition, the admin class is also responsible for seeding the initial database of recipes by importing multiple recipes from a file.

2.7 Meal

The meal class stores data related to a meal, containing the recipes associated with it, along with cooking times and prep times, serving sizes as well as the possibilities for leftovers after eating. In this class, the user is able to modify the

amount of time it takes to prepare the meal and the number of servings per meal through their respective methods.

2.8 Store

The store class handles data that the store would contain, such as its name, distances from the user (which can include multiple stores of the same name in a general area), and whether or not it requires the user to have a respective account. With this class the user can filter through stores based on certain preferences as well as get the price per unit of an ingredient to help find the best deals.

2.9 User

The user class stores all the information of the user for the app. It contains the username, password, and email of the user as well as their preferences, which are handled by the user preferences class. To initialize the user class, the user will create a new account including an email, username, and password. From there, the user can log in to their account with a login method, as well as being able to update their profile and their respective user preferences. A separate class will handle the user preferences.

2.10 User Preferences

The user preferences class handles preferences that the user may have, such as specific stores that they want to shop or the maximum distances from stores to the user's residence as well as specific dietary restrictions (i.e. religious diets, allergies, vegan/vegetarian, etc.). The user is able to update their user preferences through a method that gets their respective ID, while the application can retrieve the user's preferences to tailor meals and store choices to the user's benefit.

2.11 Rating

The rating class will handle feedback that the user gives to a recipe. It will store information about the user, the numerical score they give in the rating, and the feedback regarding the recipe. The class will handle users giving reviews to recipes, as well as checking if the feedback for a recipe meets certain criteria.

3. GUI

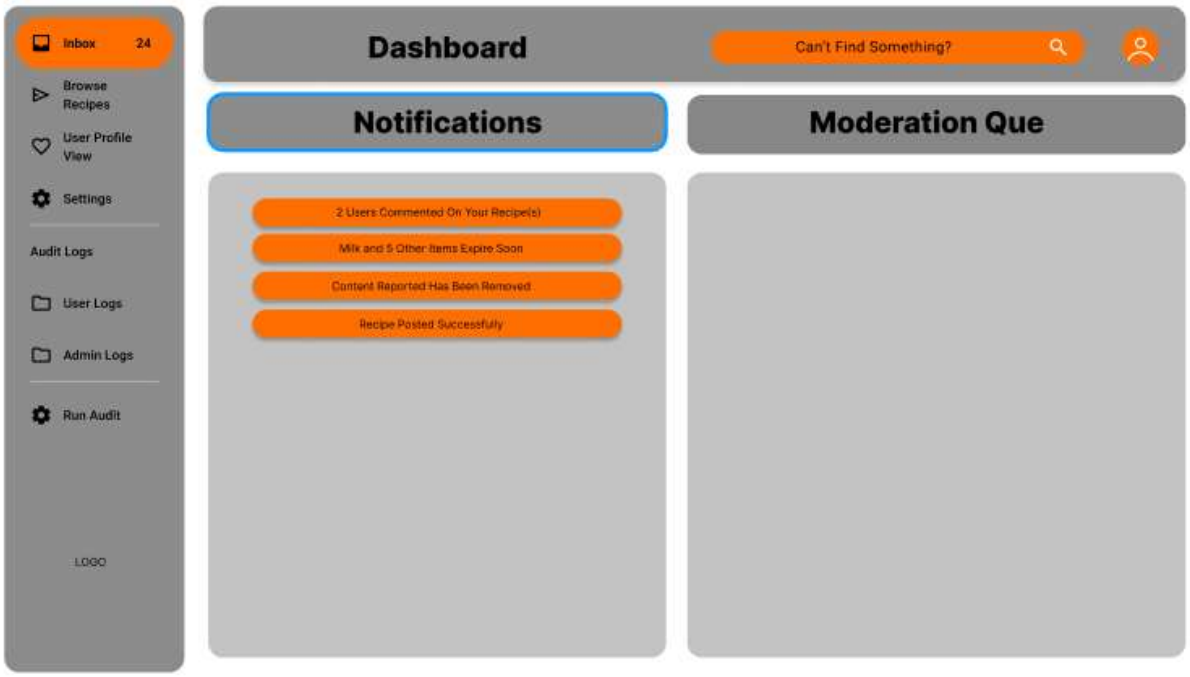
3.1 User Interface Light and Dark Modes

User Dashboard

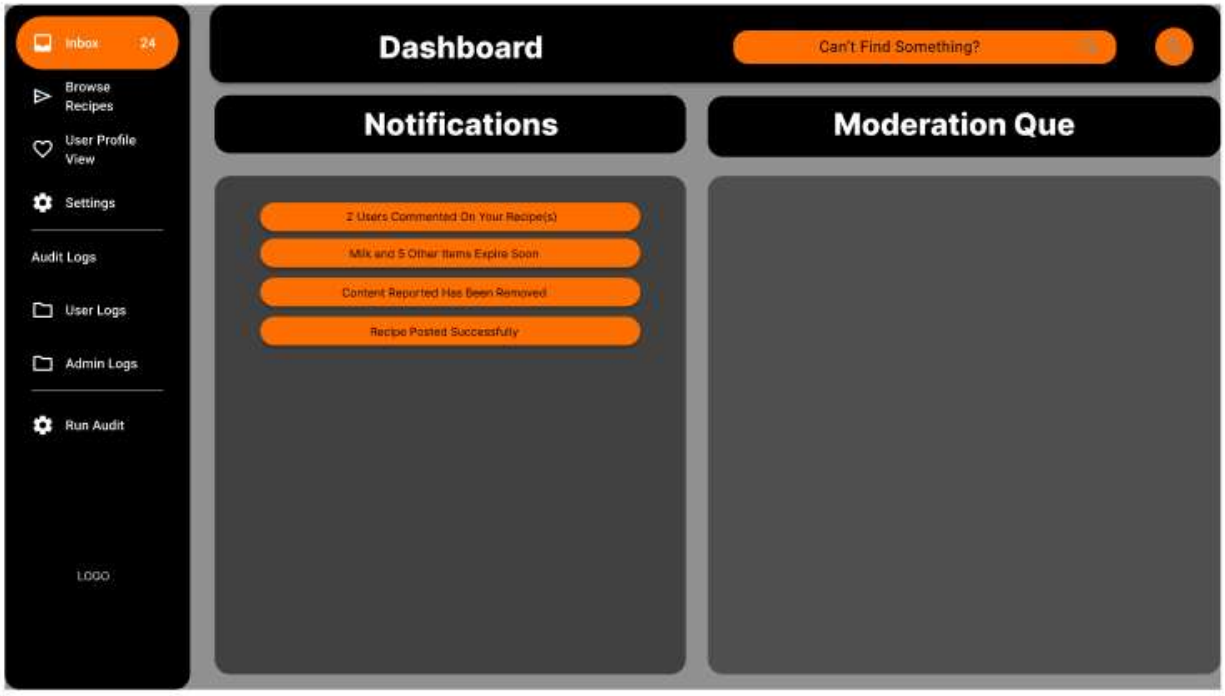
User Dashboard Dark Mode

3.2 Admin Dashboard Light and Dark Mode

Admin Dashboard

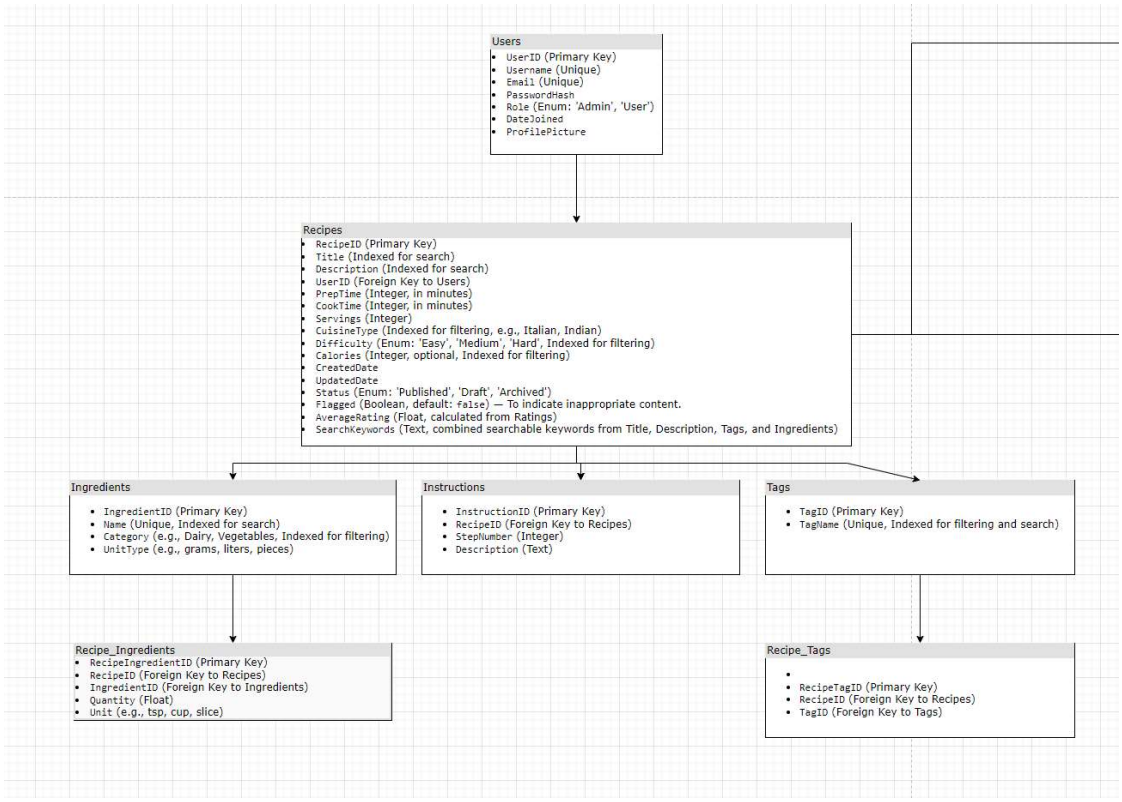


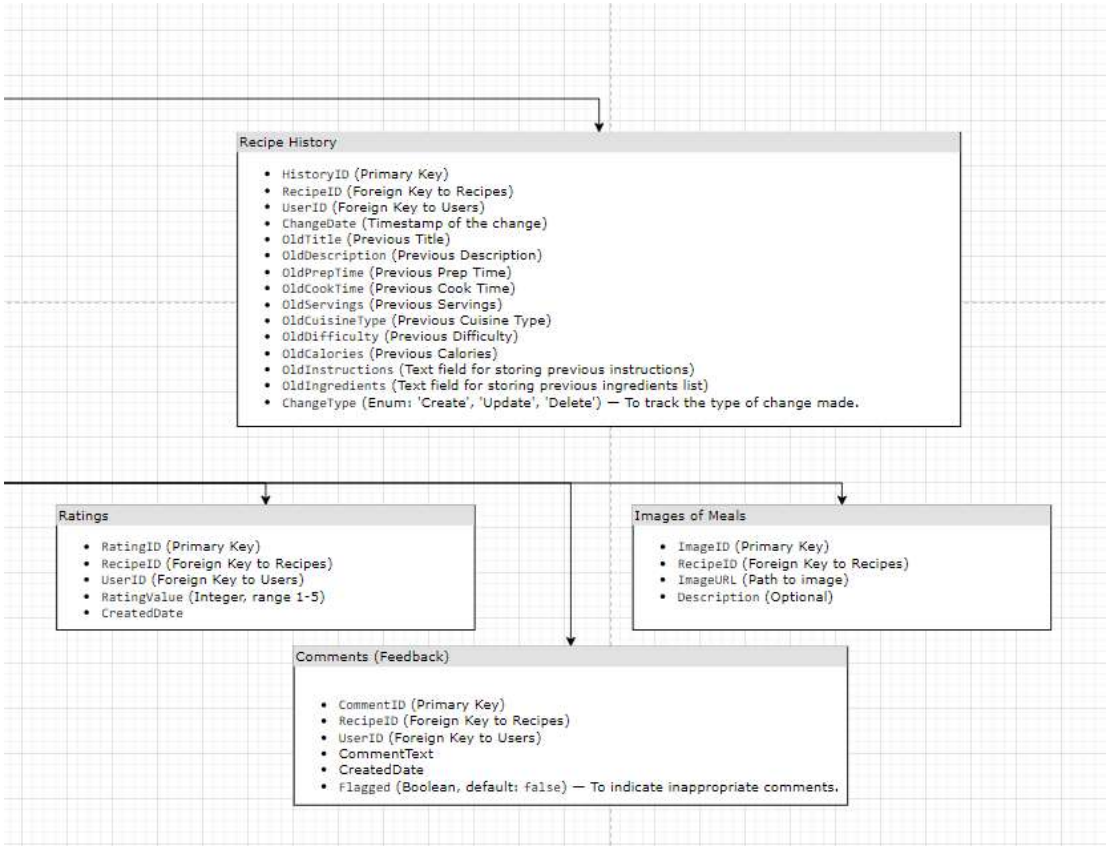
Admin Dashboard Dark Mode



4. Database Designs

4.1. ER Diagram (Recipes)





4.1.1. Users

- UserID (Primary Key)

- Username (Unique)
- Email (Unique)
- PasswordHash
- Role (Enum: 'Admin', 'User')
- DateJoined
- ProfilePicture
- Bio

4.1.2. Recipes

- RecipeID (Primary Key)
- Title (Indexed for search)
- Description (Indexed for search)
- UserID (Foreign Key to Users)
- PrepTime (Integer, in minutes)
- CookTime (Integer, in minutes)
- Servings (Integer)
- CuisineType (Indexed for filtering, e.g., Italian, Indian)
- Difficulty (Enum: 'Easy', 'Medium', 'Hard', Indexed for filtering)
- Calories (Integer, optional, Indexed for filtering)
- CreatedDate
- UpdatedDate
- Status (Enum: 'Published', 'Draft', 'Archived')
- Flagged (Boolean, default: false) — To indicate inappropriate content.
- AverageRating (Float, calculated from Ratings)
- SearchKeywords (Text, combined searchable keywords from Title, Description, Tags, and Ingredients)

4.1.3. Ingredients

- IngredientID (Primary Key)
- Name (Unique, Indexed for search)
- Category (e.g., Dairy, Vegetables, Indexed for filtering)
- UnitType (e.g., grams, liters, pieces)

4.1.4. Recipe_Ingredients

- RecipeIngredientID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- IngredientID (Foreign Key to Ingredients)
- Quantity (Float)
- Unit (e.g., tsp, cup, slice)

4.1.5. Instructions

- InstructionID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- StepNumber (Integer)
- Description (Text)

4.1.6. Tags
- TagID (Primary Key)
- TagName (Unique, Indexed for filtering and search)

4.1.7. Recipe_Tags
- RecipeTagID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- TagID (Foreign Key to Tags)

4.1.8. Comments
- CommentID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- UserID (Foreign Key to Users)
- CommentText
- CreatedDate
- Flagged (Boolean, default: false) — To indicate inappropriate comments

4.1.9. Ratings
- RatingID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- UserID (Foreign Key to Users)
- RatingValue (Integer, range 1-5)
- CreatedDate

4.1.10. Images
- ImageID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- ImageURL (Path to image)
- Description (Optional)

4.1.11. Recipe_History
- HistoryID (Primary Key)
- RecipeID (Foreign Key to Recipes)
- UserID (Foreign Key to Users)
- ChangeDate (Timestamp of the change)
- OldTitle (Previous Title)
- OldDescription (Previous Description)
- OldPrepTime (Previous Prep Time)
- OldCookTime (Previous Cook Time)
- OldServings (Previous Servings)
- OldCuisineType (Previous Cuisine Type)
- OldDifficulty (Previous Difficulty)
- OldCalories (Previous Calories)
- OldInstructions (Text field for storing previous instructions)
- OldIngredients (Text field for storing previous ingredients list)
- ChangeType (Enum: 'Create', 'Update', 'Delete') — To track the type of change made.