Software Requirements Specification for Cooks Companion

A recipe management platform encouraging ongoing development of recipes

Jarrad Phillips
10-6-2023

Contents

Introduction

Purpose

In today's world, cooking has become a universal skill, overcoming the boundaries of age, gender, and social strata. The wealth of resources available for recipes and culinary techniques has never been more abundant, whether it's a well-stocked bookshelf of cookbooks or bookmarked cooking webpages. However, many individuals aspire to do more than merely follow instructions. They want to refine their recipes with personal touches, innovation, and refinement. Consequently, the demand has evolved beyond the mere need for recipe storage.

This document serves as the foundations for 'Cooks Companion.' While, on the surface, Cooks Companion may appear yet another recipe management platform, it empowers you to not only organize your recipes, but also to personalise and develop them in a safe and encouraging manner. With Cooks Companion, you can effectively branch your recipes into numerous iterations, track changes, and collaborate with other users.

Stakeholders

This software is intended for any member of the public with a desire to cook and personalize their recipes. Other beneficiaries may include chefs and test kitchens for continuous recipe development.

Project Scope

Cooks Companion replicates all standard features of a recipe management system. This includes recipe creation, sorting, searching, and modification. The project will incorporate features from project management systems such as version control, rollback, and collaboration control.

Representing Cooks Companion on other operating systems like mobile phones and tablets is out of scope, but a consideration for future updates. A complete redesign of the UI would be needed for mobile support due to several features potentially taking up large areas of screen space.

User Stories

High Priority		
Completed	**ID**	**Story**
Yes	H1	As a user I can create and save a new recipe from scratch.
No	H2	As a user, I can search for recipes within my database by name, category, cuisine, or tags.
No	H3	As a user I can rollback to previous versions of my recipes so I can revert any changes I don't like.
No	H4	As a user I can branch my recipe into variants that I can develop independently.
No	H5	As a user I can bring other users in to collaborate on a recipe. I can control their read and write access.
Medium Priority		
Completed	**ID**	**Story**
Yes	M1	As a user I can use an AI assistant to suggest changes to accommodate a variety of dietary requirements
Yes	M2	As a user I can use an AI assistant to suggest potential improvements to a recipe.
Low Priority		
Completed	**ID**	**Story**
Yes	L1	As a user I can view a nutritional breakdown of my recipe.
No	L2	As a user I can import recipe from top recipe hosting sites.
No	L3	As a user I can export a single recipe, or multiple recipes to a pdf file.

Tabe 1: Table of user stories. All stories are categorized by priority, and are assigned an ID for reference.

User Flow

The current layout of the webpage is simple. Currently it offers:

- A home page for viewing favourites and recent additions.
- A Search page that allows the user to search their collection.
- A recipe viewing page where the user can view, modify, their recipes.

While the layout indicates that the search page and the recipe page are endpoints, any page can be accessed from anywhere using the shared navbar on all pages.

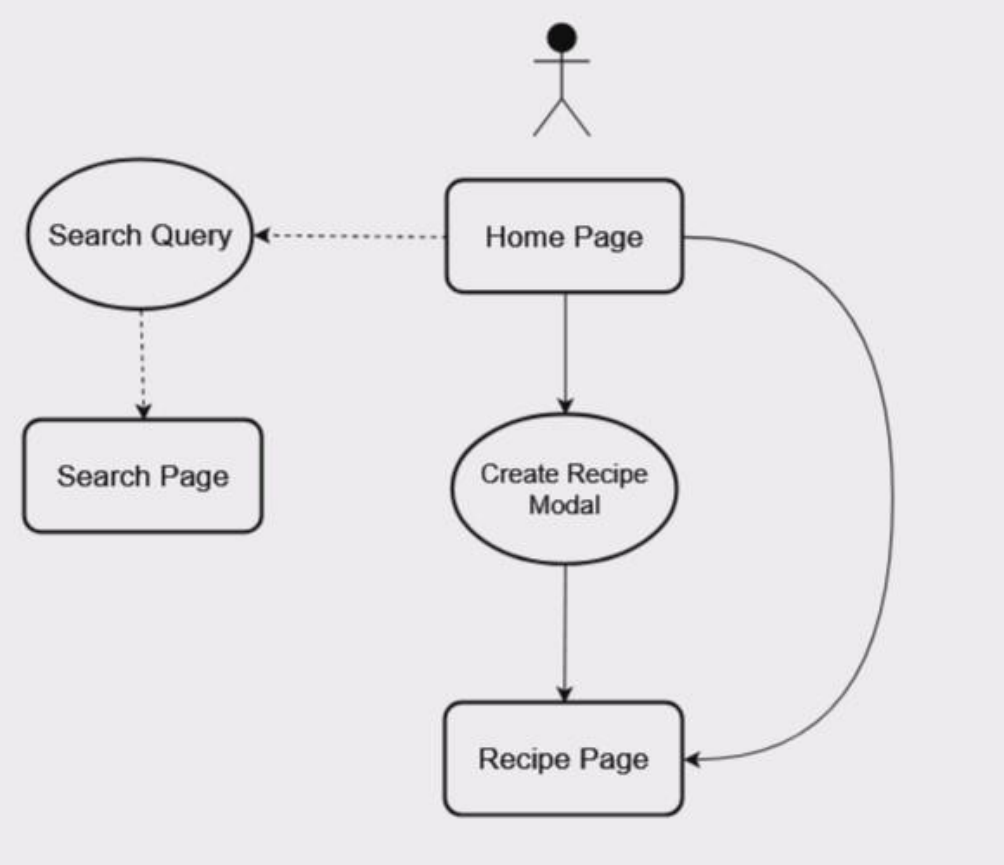

Figure 1: Screenshot of a user flow diagram made using 'Draw.io'.

UI Design

LoFi UI designs can be found on the Figma link provided below. The designs provided by Figma are initial designs made in the early stages of the project. While the changes were significant however, the core layout and style are preserved in the final project.

https://www.figma.com/file/mWxds041Ib3HP06MjWsEu8/Untitled?type=design&node-id=0%3A1&mode=design&t=uruJp6O3I5Zd2Dcg-1

A HiFi design was never properly implemented due to time constraints.

Database Design

The physical model is defined using JavaScript's data types. Current the database is designed and implemented using the non-relational database 'MongoDB'.

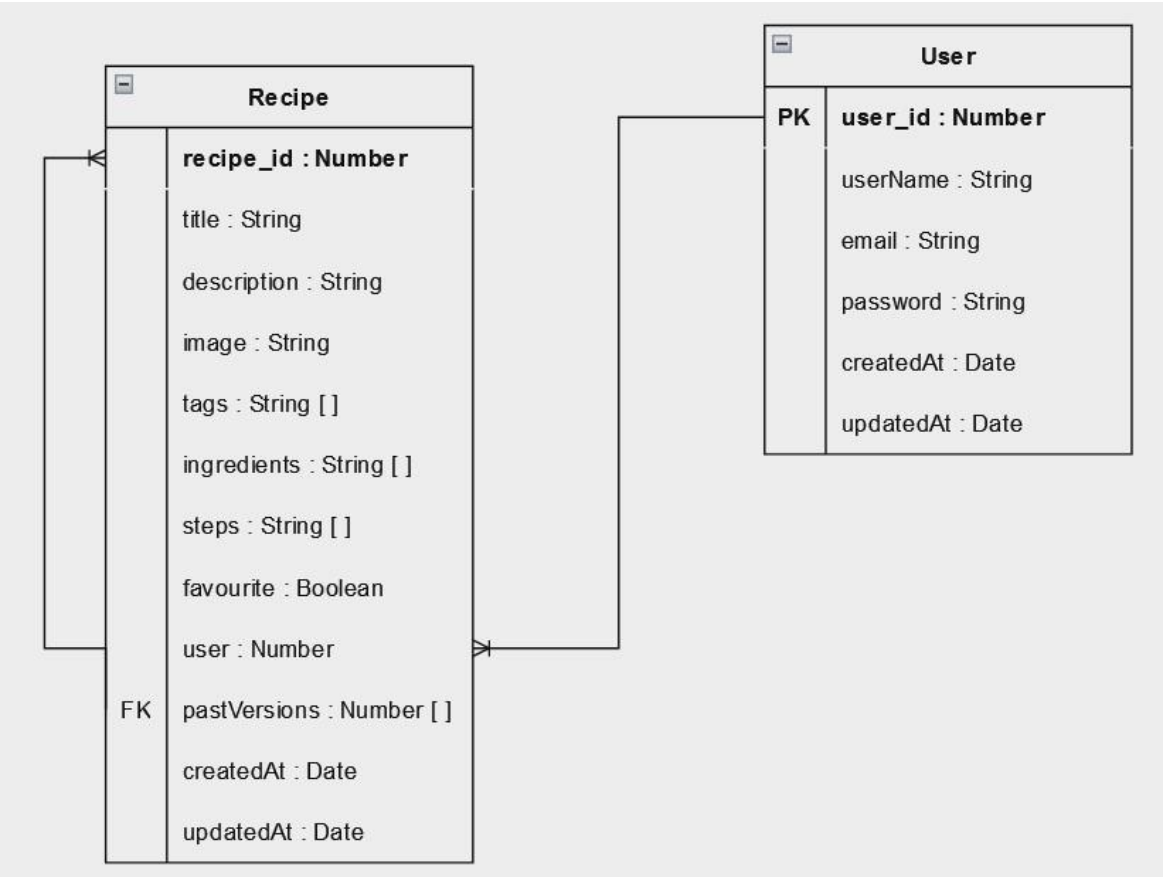

Figure 2: Screenshot of a user flow diagram made using 'Draw.io'.

Backend Design

The database design diagram was drawn using "draw.io".

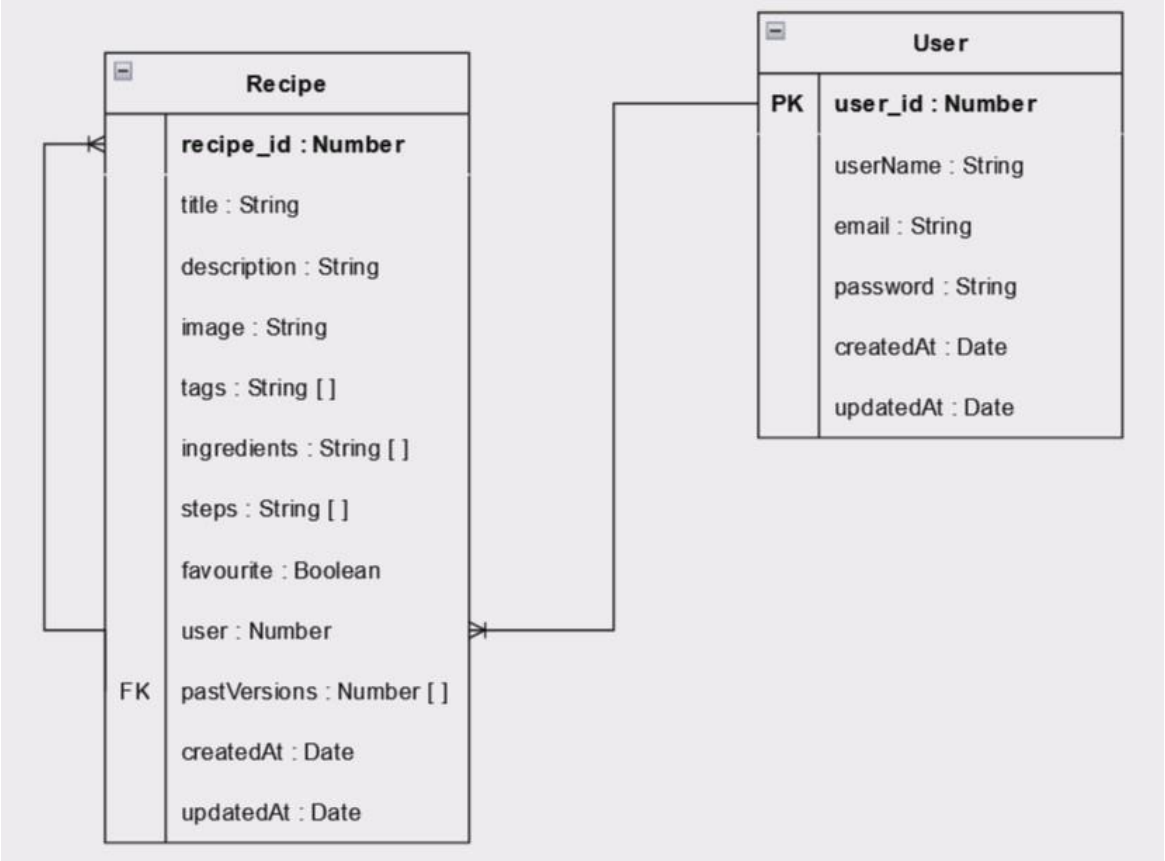

Figure 2: Screenshot of a user flow diagram made using 'Draw.io'.

Non-functional Requirements

Security

The application currently uses a register/login system to control access. This project highly values the privacy of the user and only requires the bare minimum of details to uniquely identify the user. The only details of a user required are:

- Username
- Email

- Password

Usernames are currently not intended to be unique within the database. It is simply intended to serve to make the user feel more connected with the app. On new user registering, the email must follow a regex pattern to be accepted. It must be:

- Any combination of letters, numbers, or underscores
- @ symbol
- Any combination of letters
- A dot, then 2 – 3 more letters

Another constraint for registering is password length. All passwords must be at least 5 characters long for protection against brute-force attacks. All passwords are stored as encrypted hashes on the database and verified securely on the server on login.

Once logged in to the application, a token is created for the user that is then stored in user cookies. Any route requiring a user id is protected using an auth middleware on the server.

Usability

The app is currently not published online and the userbase is small, so no system for limiting transactions have been put in place. However, the app uses the cheapest tier of MongoDB and OpenAI's ChatGPT. MongoDB is limited to a total of 512MB of storage. OpenAI allows a maximum of 3,500 requests per minute.

Ease of Use

The target audience for the application varies significantly. Therefore, the app highly prioritizes an easy-to-use user interface. Since many of the users may have never used services like ChatGPT before, the interface between ChatGPT and the user must be simple.

Testing

Unit Testing

No unit testing was performed for the project due to time constraints. At this stage the demonstrated application is considered a working prototype. Unit testing will be employed for future versions of the project.

UX/UI Testing

UX quality was ensured by testing the use of the application with friends and family without any indications of how the UI works. This testing was performed once after the first week and finally within the last stages of the project.

The colour schemes, element visibility and style, and fonts were all tested for efficiency, readability, and visual appeal at all stages of the project using friends and family as testers.

AI Testing

The ability of the user to interact with the AI is currently limited for ease-of-use and ease-of-testing. Random input was provided for each category of use and several valid use cases tested. Since the AI is controlled by OpenAI's third-part services, their management of edge cases is adequate.

Implementation

The main consideration for deployment of the application was regarding the AI. Currently the application makes use of OpenAI's ChatGPT services. These services are not free. Deploying the project to AWS would allow anyone with the link to sign up and use my AI, costing me money. Instead, the goal is to dockerize both the client and server applications and connect to MongoDB's cloud-based databases rather than a local one. This would allow me to limit who has access to the application.

At this stage in the project, a connection to a cloud-based database is available but limits certain capabilities. None of the app components have been dockerized.

End-to-end Solution

The scope of the project is big. Hence only half of the user stories could be implemented for this prototype. The project successfully implements ChatGPT to provide the user with suggestions and hints for developing their recipes. The database design and user interface are both structured to seamlessly integrate the future implementation of the remaining user stories.

Further refinement on the UI/UX is needed. Several elements are not inherently clear to the user, such as what parts of the webpage can be edited.

Out of Scope

Due to time constraints, I could only focus on either implementing the Github-like features or the AI. I chose to implement the AI. The recipe version control and branching will be implemented at a later date. Exporting the recipe to a PDF was not implemented. While an interesting feature, it did not connect to either the version control or the AI goals of the project and therefore was left out for now.

References

You can access the GitHub link to the project below:

https://github.com/Bluejay314/recipe-manager

Resources Used		
Database	**Server**	**Client**
- MongoDB - MongoDB Atlas	- Express.js - Axios - Bcrypt - Cors - Dotenv - JWT - Mongoose - Multer - OpenAI	- MUI - MUI icons - Axios - React (Vite) - React-beautiful dnd - React-cookie - React-dom - React-dropzone - React-error-boundary - React-router-dom

Tabe 2: Resources used by the project