USF SOFTWARE ENGINEERING

CAPSTONE PROPOSAL

PROPOSED BY
Kelsey Breen

whisk it

TABLE OF CONTENTS

Tech Stack 3

Project Overview 4

Features & Functionality 5

Development Approach

i. Database schema 7

ii. API Access 8

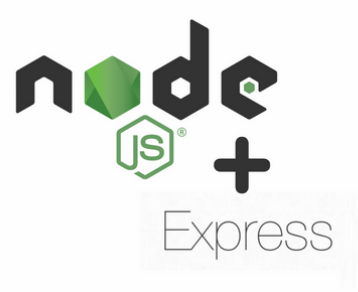
iii. Security Focus 9

iv. User Flow 10

THE TECH STACK

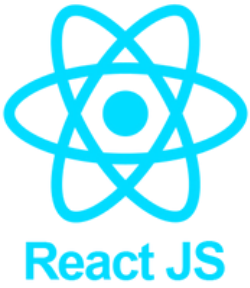
PostgreSQL
Relational Database Management System

PostgreSQL will be used to store user data, including usernames, passwords, recipe collections, notes, reviews and more.



Node.js
Open Source Server Environment

Node.js will be used to develop the server-side RESTful API. Express, a Node.js web application framework, will be used to handle routing and middleware.



React JS
JavaScript Framework for Front-End Development

ReactJS will be used to develop the front-end of the application. The build tool Create React App will be used to build the application.

Redux
State Management

Redux is the Javascript library that will be used for state management on the client side. Redux Toolkit will be used to configure the store, and Redux will handle both synchronous and asynchronous data management.

PROJECT OVERVIEW

THE CONCEPT

Introducing the ultimate recipe app, where storing and finding your favorite recipes has never been easier. With Whisk It, you can keep all of your favorite recipes in one place, making it easier than ever to find that perfect dish for any occasion.

But that's not all! You can also make notes about recipes, leave reviews so that you can remember any changes that you made or share your thoughts with others. Whether you're an experienced chef or just starting out, Whisk It is the perfect tool to help you create amazing meals that you and your loved ones will enjoy.

Grilled nectarine & burrata salad

★★★★★

Chiu Chow smacked cucumber

★★★★☆

THE APP

Whisk It is a single page web application built for browsing on a computer or mobile device. The focus of this application is the front-end user functionality and user interface. However, it will also include a backend, where user data will be stored and accessed via React.

The goal is to provide an easy to use storage system for recipes and notes for users interested in cooking. Based on demographics of similar applications, it is anticipated this application would be used by primarily females (80:20) with approximately 60% of users between the age of 18 and 34, with the remaining users in older age brackets.

The app name is "Whisk It" with a single letter icon and full logo. The app colors are purple (#e798ff), beige (#ffeeeb), green (#86e66d), white (#ffffff) and orange (#f5694d). The font is Albertan Pro.

whisk it

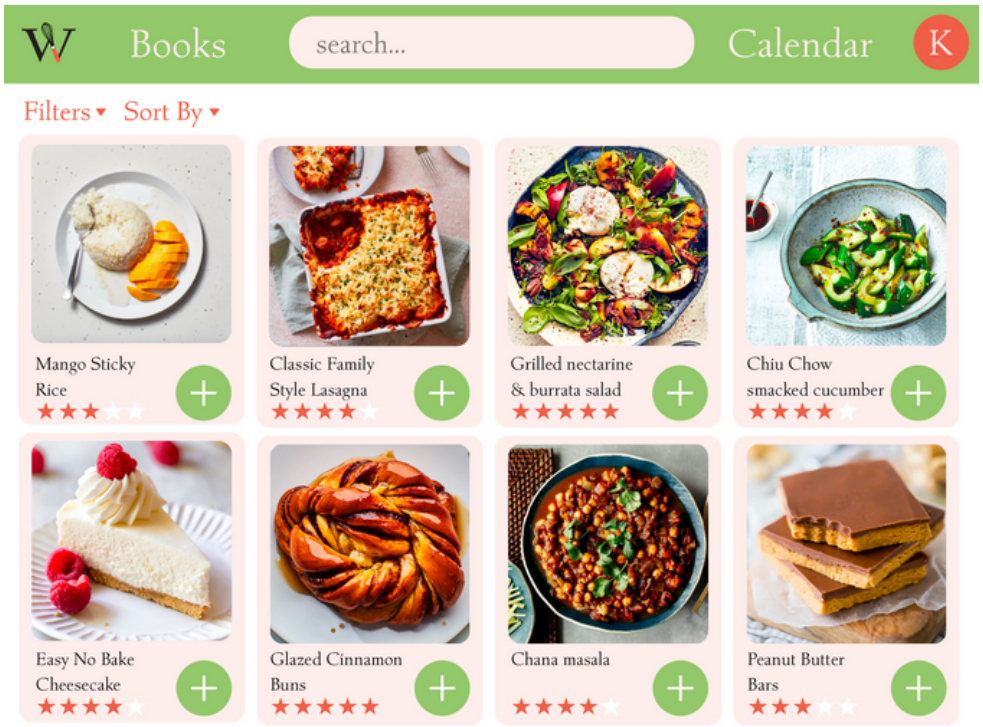
FEATURES / FUNCTIONALITY

THE HOMEPAGE

Upon entering the application, all users (authenticated and non-authenticated) can scroll through hundreds of recipe cards with pictures, names and ratings.

Users will see the green "add" button to add these recipes to their "books", which are collections of the user recipes. Users must be logged in to add to their books. Non-authenticated users whom click on the add button will be directed to the login/sign up page.

Users can filter recipes by keywords, or sort by rating.



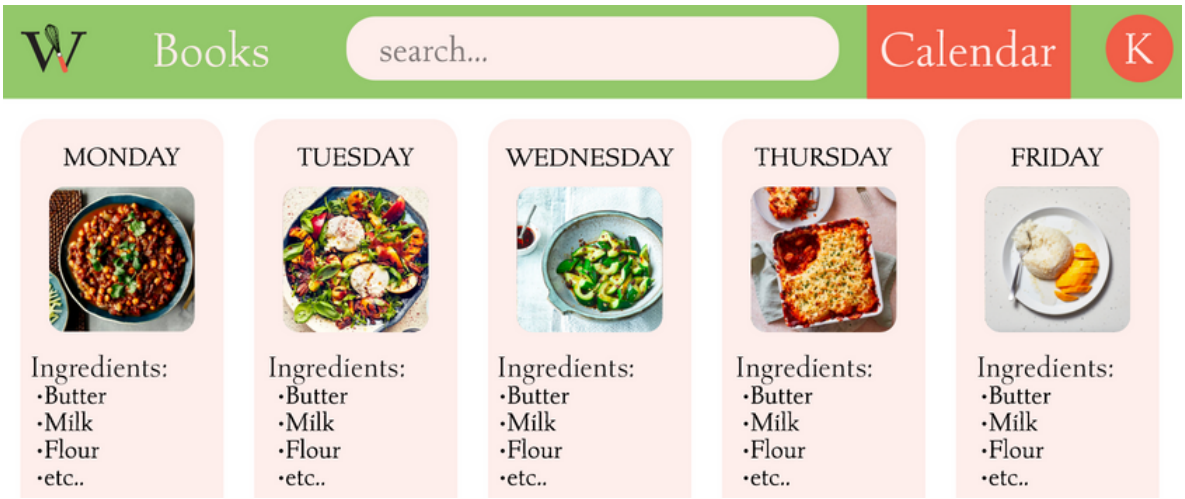
THE BOOKS

The books are the users own collections of recipes saved in a simple folder on their dashboard.

FEATURES CONT.

THE CALENDAR

Users can create a weekly calendar by dragging and dropping recipes into the day of the week. Dropping a recipe in a day also provides a shopping list of ingredients needed for the recipe so that users can create a weekly grocery list.

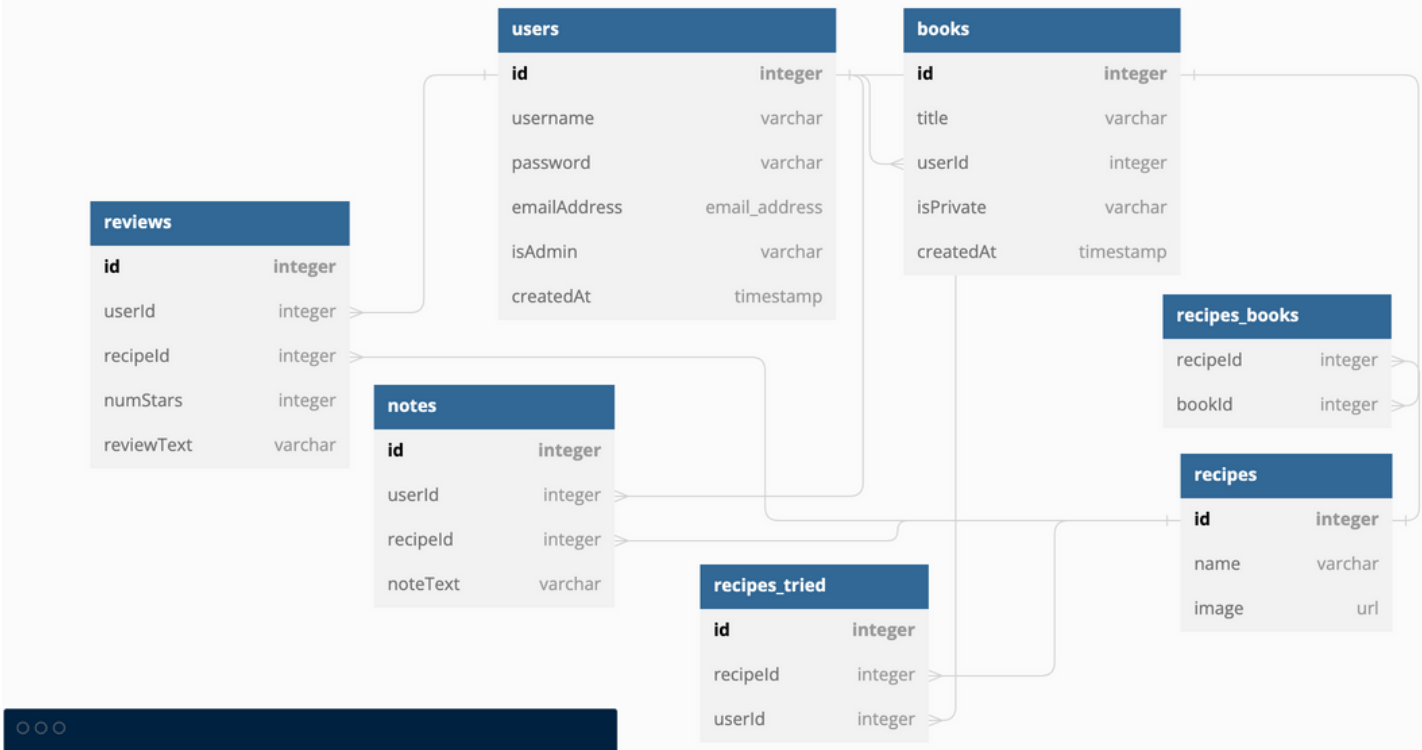


THE RECIPE CARD

The recipe card is the feature that sets this application apart from the rest. Users are able to complete most of their functionality here, creating notes about the recipe, provide a rating and review. Users can also view reviews from other users, as well as mark the recipe as tried.

DEVELOPMENT APPROACH

I. Database Schema




```
Table reviews {
  id integer [primary key]
  userId integer
  recipeId integer
  numStars integer
  reviewText varchar
}

Table users {
  id integer [primary key]
  username varchar
  password varchar
  emailAddress email_address
  isAdmin varchar
  createdAt timestamp
}

Table books {
  id integer [primary key]
  title varchar
  userId integer
  isPrivate varchar
  createdAt timestamp
}

Table recipes {
  id integer [primary key]
  name varchar

}

Table recipes_books {
recipeId integer
bookId integer
}

Table notes {
  id integer [primary key]
  userId integer
  recipeId integer
  noteText varchar
}

```

DEVELOPMENT APPROACH

II. API Access

THE RECIPE API

The API that I plan to use for this project is the Edamam API to gather recipe data.
Link | Docs for Recipe API

EXAMPLE

BASE URL: https://api.edamam.com/

> To get recipes:
Endpoint: api/recipes/v2

This is a sample of the database model and response. Authentication is required to access the API. Both application key and application id are required to be sent as parameters. Various filters can be added, including search terms, cuisine types, and more. Including these filters via query parameters limits the amount of information that is returned.

API calls can access recipes directly via URI or an id, as well as return a random set of recipes based on various criteria.

Data cannot be saved within our database, except for URI, images, and ids. These will all be saved in our recipes database.

THE USER API

We will have our own server that stores user data. This RESTful API will return JSON data.

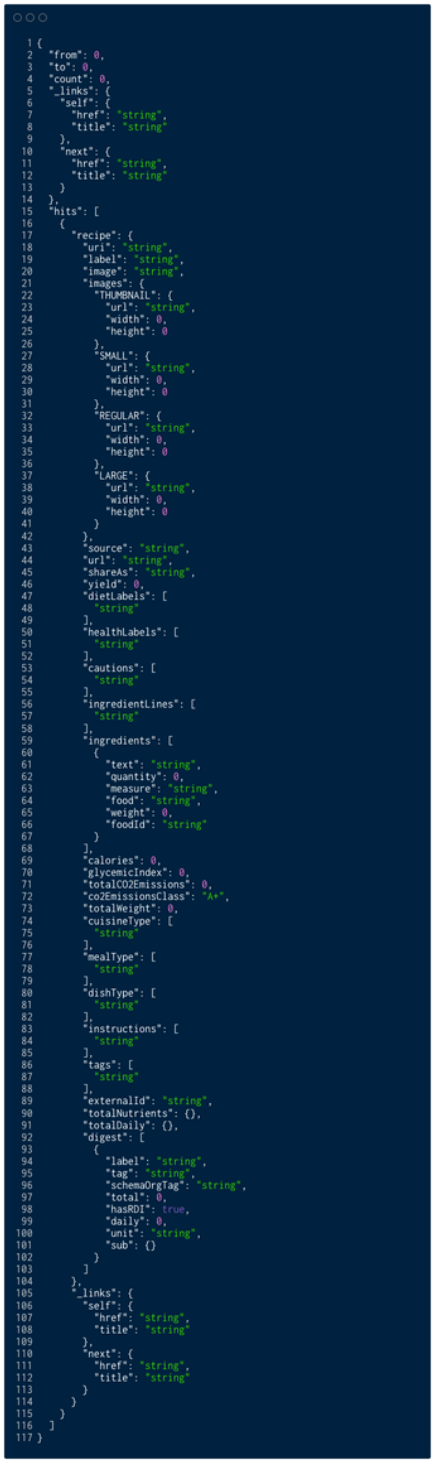
Routes:

POST /api/auth/login
POST /api/auth/signup

GET /api/users
GET /api/users/:id
PATCH /api/users/:id
DELETE /api/users/:id

GET /api/recipes
POST /api/recipes
GET /api/recipes/:id
PATCH /api/recipes/:id
DELETE /api/recipes/:id

This pattern would be repeated for:
/books
/reviews
/notes



DEVELOPMENT APPROACH

III. Security Focus

AUTHENTICATION

All features, besides recipe browsing, require authentication of the user. Each user will be required to create an account so that data, including books, reviews and ratings, notes, etc. can be stored.

User accounts will collect the following information:
> First and Last Name
> Username
> Email Address
> Password
> Optional Profile Image

When users sign up, their passwords will be encrypted using bcrypt.

When users log in, their password will be verified using JSON Web Tokens, which will be returned from our own API storing application data.

LocalStorage will be utilized to store user (non-sensitive) information so that the information can persist.

Authentication will be required for every route except /api/auth/login and /api/auth/signup for our user API. POST, PATCH and DELETE routes will require authentication to match the userId or have admin credentials.

DEVELOPMENT APPROACH

IV. User Flow