ECE 49595 Senior Design Proposal

https://github.com/jtimmerb/Eatr

Group

Tony Ni
ni86@purdue.edu

Javon Timmerberg
jtimmerb@purdue.edu

Varun Dengi
vdengi@purdue.edu

Goals

Our goal is to design an application that will help users learn more recipes that they can cook at home and simultaneously make either healthier or dietary choices that are more inline with their lifestyle goals. One of the main appeals of being able to order takeout is the wide variety of options available that a customer can scroll through, and we hope that giving customers the option to visualize more options that they can make at home, they will be more motivated to cook at home and spend less by ordering takeout less.

Idea

We are proposing a web/phone application that prompts a user for desired ingredients and returns back a list of recipes that match the highest amount of ingredients. We store the recipes in a database and match users with potential recipes for a given list of desired ingredients. The user can then swipe for desired recipes that are then dropped into a list of recipes that they can view at any point. A user can then choose a specific recipe they swiped on previously and begin tuning the amount of ingredients needed to cook the recipe for a specific number of meals. After choosing the meal size, a user can change the inclusion of specific ingredients and compare the calorie and macromolecule amounts in real time if they desire. Finally, the program will give the user the full recipe with instructions scraped from the internet.

Proposed Frameworks and Components (OSS)

We will use TypeScript to build the Scrapper, the Backend Service, and the Frontend. Typescript is a typed language based on Javascript, giving us the ability to have precise definition of any structure we create. Having strict types will allow us to build and refactor our code easier because the shape of all objects are defined.

Frontend

For the frontend we will use Figma to blueprint our UI/UX design and iterate through different designs to choose one that will represent the product the best. We will use the React Framework to build the frontend and we will use TypeScript as our language. We will use React because it is the most popular frontend framework in the industry and the most beginner friendly and we will use TypeScript because it is JavaScript but with typing which can help us build better coding semantics.

OSS Components

- ReactJS
- TypeScript
- NodeJS
- ExpressJS
- JWT
- Vite

Backend

For the Backend Service we plan to build it using ExpressJS and NodeJS and with TypeScript. ExpressJS and NodeJS are the most popular frameworks for setting up API's because it is fast and easy for beginners to learn and set up. We are using TypeScript again because it gives us the option of typing on top of JavaScript's set of features.

Database

For the Database we are planning to use PostgreSQL to store our information. PostgreSQL is a relational database that fits our needs because recipes and ingredients are related in nature. Furthermore, we plan to give users the ability to search recipes based on ingredients which would be a query-intensive task that a SQL database will be a better fit for handling. We are also planning to use Sequelize which is an ORM for NodeJS that can help abstract the database interactions to a higher level.

OSS Components

- ExpressJS
- NodeJS
- TypeScript
- PostgresSQL
- Sequelize

Data types

User

1. User id (PK): integer
2. Name: string

Recipe

1. Recipe ID (PK): int
2. Name: string
3. Recipe steps: string []

Ingredient

1. Ingredient ID (PK): integer
2. Name: text
3. Serving size: text
4. Calories: integer
5. Protein: integer
6. Carbs: integer
7. Fat: integer

Recipe-ingredient-membership

1. Membership ID (PK): integer
2. Recipe ID (FK): integer
3. Ingredient ID (FK): integer
4. Ingredient amount: string

User-recipe-membership

1. Membership ID (PK): integer
2. User ID (FK): integer
3. Recipe ID (FK): integer

User-pantry-membership

1. Membership ID (PK): integer
2. User ID (FK): integer
3. Ingredient ID (FK): integer
4. Ingredient amount: string

Deliverables

- Database schema for storing recipes and user information
    - Initial set of recipes, ingredients, and users for development
- Set of basic APIs to be able to reliably communicate with backend database

- Frontend user interface

- Users can reliably create a local session to access their unique list of pantry ingredients and stored recipes.
    S - Action relates directly to user interactions between frontend, and backend database querying. Also incorporates user interactions with the authentication page.
    M - Can measure success through tracking database interactions.
    A - Can be split up between frontend authentication and backend database interactions.
    R - There are packages and tutorials explaining these interactions online.
    T - Dependent on specific actions:
        - Frontend authentication
        - Backend database querying
- Application can query specific recipes given prioritized ingredients and returns a list of recipes ordered based on the largest amount of ingredients matched.
    S - Action relates directly to user interactions between frontend, and backend database querying.
    M - Can measure success through tracking database interactions.
    A - Can be split up between frontend user interactions, business logic implementation, and backend database interactions.
    R - There are packages and tutorials explaining these interactions online.
    T - Dependent on specific actions:
        - Frontend design
        - Business logic prioritizing recipes based on a predetermined list of ingredients
        - Backend database querying
- Application gives users a selection of queryable ingredients to filter their recipe search.
    S - Action relates directly to user interactions between frontend, and backend database querying.
    M - Can measure success through tracking database interactions.
    A - Can be split up between frontend user interactions, and backend database interactions.
    R - There are packages and tutorials explaining these interactions online.
    T - Dependent on specific actions:
        - Frontend implementation of ingredient list
        - Backend database querying
- Users can create new recipes to be stored in the database and accessed by other users through the app.
    S - Action directly relates to having an implemented frontend design to input new recipes and backend actions of storing new recipe.
    M - Successful operation will allow for the full process of creating a new recipe and verifying its storage along with other recipes in the backend database.
    A - Can be split up between frontend UI to input new recipes and backend operations to store recipes along with others.

R - Involves basic operations to frontend and backend similar to what will be implemented in other milestones.

T - Milestone is successful after:

- Frontend implementation allowing users to create new recipes
- Backend implementation storing recipes along with the previous collection

- Application reliably stores a maximum of 10 liked recipes that can be accessed at any point by the user.

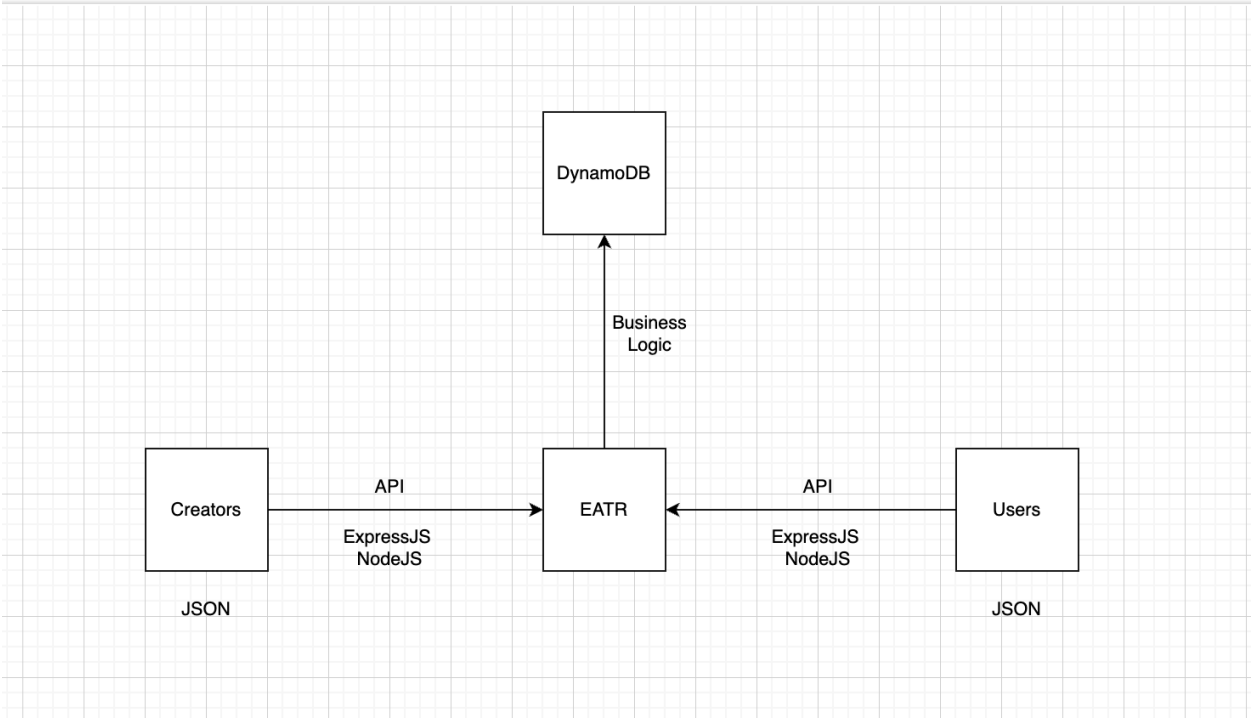
S - Specific to user operation of liking and accessing recipes through frontend and backend operations to retrieve recipes for a specific user.

M - Can be measured through successful storing and accessing of the backend database. We can also visibly see in storage when they have reached their maximum.

A - Action is split up by frontend logic of allowing the user to like a recipe and to at a later point access it again and backend logic of storing associated liked recipes to specific users.

R - Involves basic operations on the backend database and a simple frontend design.

T - Success is dependent on having the action implemented in frontend and backend operations.



User Experience

1. User launches the application.
2. User is asked to sign up or login.
3. User goes to the home page.
4. User has a view of all their recipes that they've previously liked.
    a. This is separate from recipes that they have swiped on in previous queries.
5. User has a page to enter in ingredients.
6. User has a button to start swiping on recipes.
7. After swiping on recipes, the user can view a list of all of the recipes they swiped on in this current query.
8. The user can then select a recipe and adjust the portion size of it.
9. User has the ability to see the macros of the recipes and adjust specific ingredients.