CS320: Software Engineering

Edibly

Group 5: Omar Osman, Justin Cheng,
Ludovic A., Haamed Syed Rahman

Manager: Aditi Bansal

Project Overview

Our project is a web-based dietary profiling app that helps users navigate local dining options based on their dietary needs/allergies, and preferences. The platform allows users to create a profile, specify allergies with severity levels, and receive personalized dining recommendations across the Five Colleges.

Technical Specification

Database: **P**ostgreSQL
Backend: **E**xpressJS
Frontend: **R**eactJS
Runtime Environment: **N**odeJS

User Creates a Profile

function register(): -> This function will be located in the frontend

Parameters: Username, email and password.

Frontend: User fills out form and clicks register.

Backend: Auth0 validates input, returns error if invalid/already exists.

Database: Check if email linked to an already existing account, if not create new account.

Backend: Returns a success response if successful, error message if account already exists.

Returns: JSON containing either success or error response

Testing

register():

We use a dummy array of userObj in the backend

Check if user email/username already exists

If valid information:

- Assert the response from backend is a success response

If invalid

- Assert the response from backend is an error response

After this works, we can try actually testing auth0 authentication with dummy info

User Logs into Profile

function authenticateUser(string, string) -> Success/Error:

-> This function will be located in the backend

Parameter(s): Username/email and password.

Frontend: User submits their credentials.

Backend(registerCrendentials()): Receives the credentials which is sent to the Auth0 authentication service.

Database(sendToAuth(string,string)): Handled by Auth0(cross-checks credentials against stored user data)

Backend: Returns an Auth0 Access Token if valid.

Frontend: Transfers the user to the home/landing page

Returns: A success message if the login was valid or else, an error.

Testing

Front -> Back -> DB -> Back -> Front
Step 1 2 3 4

authenticateUser(user1234, 12345678):

1. Test that the data retrieved by Express is valid and retrieves the inputs.
2. Mock authentication to simulate requests to the Auth0 endpoint.
   Mock return a valid token stored with these arguments: TOK123
   3. Confirm that the token is being stored properly in the database
   4. Confirm that the frontend is properly redirecting to the Landing page, then shows a success message.

After the mock authentication is working perfectly, we can then try using other features from that dummy profile.

User Receives Notifications For Favorite Meals

function setFavorite(): -> This function will be located in the backend:
Called when user clicks on the setFavorite button associated with a meal.
Parameter(s): Associated userObject who clicked the button, associated mealObject to be favorited.
Frontend: User clicks the "Favorite" button on a meal.
Backend: Sends a request to the server to store the favorite meal.
Database: Stores the user-meal relationship in a "favorites" table.
Returns: 1 on successful addition to favorites, -1 otherwise.

function notifyUser(): -> This function will be located in the frontend
Once a meal is added back to the menu of a dining location for the day, this function must be called to notify every user who favorited said meal.
Parameter(s): mealObject
Backend: Queries the database for users who favorited the meal.
Database: Retrieves list of users to notify.
Backend: Sends notifications via email or push notifications.
Returns: 1 on successful call, -1 otherwise

Testing

setFavorite()
Check if the meal is already in the user's favorites list:
- Query the favorites table to see if an entry exists for the given user_id and meal_id.
- If the query returns a result, return -1 (meal is already favorited).

Insert the favorite meal if it doesn't exist
- If no result is found, insert a new row into the favorites table.
- Return 1 to indicate success.

notifyUser()
Query the Database for Users Who Favorited the Meal
- Check the favorites table to find user_ids of users who favorited the given meal_id.

Send Notifications
- If users exist, console.log the notification for each user.
- If no users have favorited the meal, return false.

If at least one user was notified, return true.

User Searches for a Meal/Location

function getLocation() -> This function will be located in the backend:

Parameter(s): User-inputted dining location name.

Frontend: User enters a dining location and clicks search.

Backend: Calls the database API to fetch meals at the specified location.

Database: Retrieves matching meal entries.

Backend: Formats and returns the results.

Frontend: Displays the meals in the UI.

Returns: Array of Meal objects whose location matches the input.

function getMeal() -> Backend:

Parameter(s): User-inputted meal name.

Frontend: User searches for a meal by name.

Backend: Queries the database for meals matching the input.

Database: Retrieves relevant meal entries.

Backend: Formats and returns the results.

Frontend: Updates the UI with matching meals.

Returns: Array of Meal objects whose name contains the input string.

Testing

getLocation()
We use a dummy array of tuples with format [mealObj, locationObj] that simulates fetching meals based on location from the database.
Filter array by whether the second field of a tuple matches the input string.
The filtered array will be put into a JSON format.
Console.log the filtered array

getMeal()
We use a dummy array of objects that simulates fetching meals from the database.
Filter array by whether it contains input string.
The filtered array will be put into a JSON format.
Console.log the filtered array