Product Name: FoodScoop

Team Members:
- Surya Subbarao (surya04@ucla.edu), Discussion 1C
- Arya Gharib (argharib@ucla.edu), Discussion 1C
- Weiqi Yu (weiqiyu@ucla.edu), Discussion 1B
- Aminah Khan (aminahk@ucla.edu), Discussion 1A
- Omar Elamri (omarelamri@ucla.edu), Discussion 1A

Summary: Our project aims to strengthen the UCLA dining experience by providing a better interface for dining menus, nutritional information, and meal planning. We seek to make it easier for users to filter foods by dietary restrictions, save diet or meal plans, and aggregate nutritional information. Users will be able to create accounts and get notifications of when their "favorite meals" are available.

Introduction

The University of California, Los Angeles (UCLA) is a public university located in Los Angeles, California. With an undergraduate population of over 31,600 students and 5,000 living on campus, UCLA manages to provide a highly acclaimed residential dining experience. Even with a dedicated dining website, it is often difficult for students to navigate the plethora of options available to find a meal plan which works best for them. For example, the website has no mobile application (which most users access it from), labels for vegetarian, Halal, etc. have both false positives and negatives, and generally, it looks very unappealing. Not to mention that the website is riddled with performance issues and bugs that require a complete reload.

The app itself is also fairly incomplete. There is a Nutritive Analysis page, but it fails to give recommendations as to which meals a user should choose based on their preferences. It also doesn't let the user track which meals they've chosen; nor can people rate them.

Hence, it is evident that the UCLA dining website needs a complete overhaul. A new application built from the ground up is necessary to fix certain problems and help students be more proactive and informed about their nutritional lifestyle.

Feature Overview

Note: Any starred (*) item is part of the minimum viable product. Non-starred items are subject to change.
This application will:

- (*) store user accounts in order for them to save their preferences. This will also allow users to receive push notifications and sync between multiple devices. Preferences include:
    - Preferred entrées/cuisines
    - Preferred dining halls
    - Dietary restrictions vegetarian, Halal, gluten-free
    - Comprehensive meal plans
- (*) scrape data from the current UCLA dining website.
- (*) filter entrées by:
    - Nutritional content such as caloric content, sodium, protein
    - Dietary restrictions such as vegetarian, Halal, gluten-free
- (*) let users create comprehensive meal plans.
    - (*) A comprehensive meal plan takes into account a user's preferences and creates a meal plan for the entire week based on a user's dining meal plan (19 Premier, 14 Regular, etc.)
    - (*) It will also take into account any weekly nutritional guidelines by content such as caloric content, sodium, protein.

    - Also supports intermittent fasting guidelines.
- send push notifications when activity levels are low.
- send push notifications that remind users to go to specific dining halls at the start of each meal period based on their preferences and comprehensive meal plan guidelines.

Technology Stack

This application will be using the standard React Native/Node.js framework for our frontend and backend respectively. It is expected to be deployed to both iOS/iPadOS/macOS devices and Android. Deployment to the web is not part of the minimum viable product; however, it is a fairly simple process.

On the frontend, React Native allows the application to run near native code with the convenience of writing declarative UI in HTML-like syntax and logic implementation in JavaScript. In addition, Capacitor will be used, which automates deployment to the Apple and Google App Stores. The frontend will integrate with the backend using the Axios REST API library. User login will be facilitated using Google's Sign In API. This allows the application to delegate authentication to an external service (no passwords involved).

On the backend, the Node.js application will organize its functionality using the Express REST API library. Each endpoint and its logic will be dictated by Express functions. To store user data, MongoDB and its NoSQL (no schema) paradigm will be essential. Having a flexible schema is imperative to quickly prototyping applications that are production ready. A Python script (using BeautifulSoup) will scrape the data from UCLA dining website every hour.

From the developer operations perspective, the backend will be containerized with multiple Docker instances.

1) The first Docker instance will be the Node.js application itself. The PM2 monitoring library will invoke the Node process to start the application while monitoring any potential issues.
2) The second Docker instance will run the Python web scraping script.
3) The third Docker instance will run MongoDB's database.
4) The fourth Docker instance will host the static files (HTML, CSS, JS) that the frontend generates.
5) The fifth Docker instance will run Caddy, a reverse proxy that interfaces with the wider internet. This will run between the user, Node.js REST API and the static files.

During development, these Docker instances will run on a local machine. For production, the instances will run on a Google Cloud Platform virtual machine running Debian.