Building a New Yale Dining Application for Android

1. Abstract

This project involves developing a new Yale Dining application for the Android operating system. The existing Yale Dining app on Android is not very pleasant to use, due to dated visuals, usability issues, and several recurring bugs. Much like the existing app, the app created for this project displays the current menu items and nutritional information for each dining hall on campus, using information retrieved from the Yale Dining API. It also includes the ability to set a dining hall as a favorite, and have the app automatically navigate to that hall’s menu when launched. The app contains several useful features not included in the original app. The app can highlight menu items that contain certain dietary traits, allowing people with dietary restrictions to easily find out which foods they should avoid. Users are also able to mark their favorite dishes in the app, and receive notifications when those dishes are being served again. In addition, the app prioritizes the information it displays based on the time of day and location. For example, a dining hall’s lunch menu will be shown first if it is close to lunchtime, and the dining halls in the main screen are organized based on their distance from the user’s location.
The app was written primarily in Java, and created and tested using Android Studio. The app gathers the menu information from the Yale Dining API daily, and stores the information locally in a SQLite database. When displaying information, the app queries the database to retrieve the relevant information.

2. Motivation

In addition to improving on the original app, I wanted to use my project as an opportunity to learn more about areas of software development that I was unfamiliar with. I had made one Android application before in my spare time, but it was relatively simple, and I wanted to create an application with more complex features. Also, I had very little experience creating and working with databases, and felt like this project would be a great way to gain experience in this area.

3. The Database

The app uses a SQLite relational database to store data about the dining halls and their menus. The database contains four tables: dining_hall, menu_item, nutrition_item, and ingredient. The dining hall table contains the location, name, contact info, and managers of each dining hall. The menu item table contains information about the names of the food items being served, and the meals that they are part of. The nutrition item table contains nutritional information for each food item, and the ingredient table holds information on the food items’ ingredients. Due to how the Yale Dining API was organized, I had to make several compromises in the database’s design to make information retrieval easier. For example, the API combined information for meals (breakfast, lunch, dinner) with information for food items (apple, waffles) in the same table. I decided to keep this organization in my application’s table, because it would have taken too
much effort to separate information into different tables.
4. Functionality

The main page of the app has a list of all the dining halls, their distance from the user’s location, and their percentage occupied. The list is separated by whether the halls are open or closed, and these lists are then sorted by distance.

Fig. 1: Table creation statements for the tables in the database.
Fig. 2: The app’s main page.

The menu page for each dining hall displays the menus for each meal being served in the dining hall for that day. If the hall has already stopped serving a meal, it is excluded from the list. The page also contains an option to favorite the dining hall. If the hall is favorited, the app will navigate directly to the hall’s menu page the next time it is launched.
The item detail page displays all of the nutritional information for a menu item. In the original Yale Dining app, the dietary traits, ingredients, and nutritional facts were separated into tabs. My app combined the three sections into one, and made them more distinctive visually. The page also contains a button that allows the user to follow a menu item. If a menu item is followed, the user will receive a notification the next time it is being served in a dining hall. I used Android’s JobService API to implement the feature. When the user first starts the app, a process is started in the background that runs even after the user closes the app. Every 6 to 12 hours, the process
retrieves the menus for all the dining halls and checks to see if the item is being served. If the item is being served, a notification is sent to the user.

Figs. 4 and 5: The item detail page with notifications enabled. The user receives a notification for a followed item.

When the user first launches the app, they are prompted to indicate dietary traits that they want to avoid. The app then uses this information to highlight menu items on the menu page that contain
these traits, making it easier for the user to see which dishes they should avoid.

Figs. 6 and 7: The trait selection screen. The menu screen, with dishes containing marked traits highlighted in red.

The app contains a settings page for users to manage the personalized information they have stored in the app. The page contains two options, one for dietary traits and another for followed items. The dietary traits preference screen allows users to select and remove traits for the app to use in highlighting dishes. The followed items screen allows users to remove items from the follow list.
Figs. 8 and 9: The settings and followed items screens.

5. Further Work

Additional functionality I could add to the app includes an info page for each dining hall that displays its address and contact information, and a recommendation engine that suggests dishes for the user to try, based on their favorited items and dietary preferences.