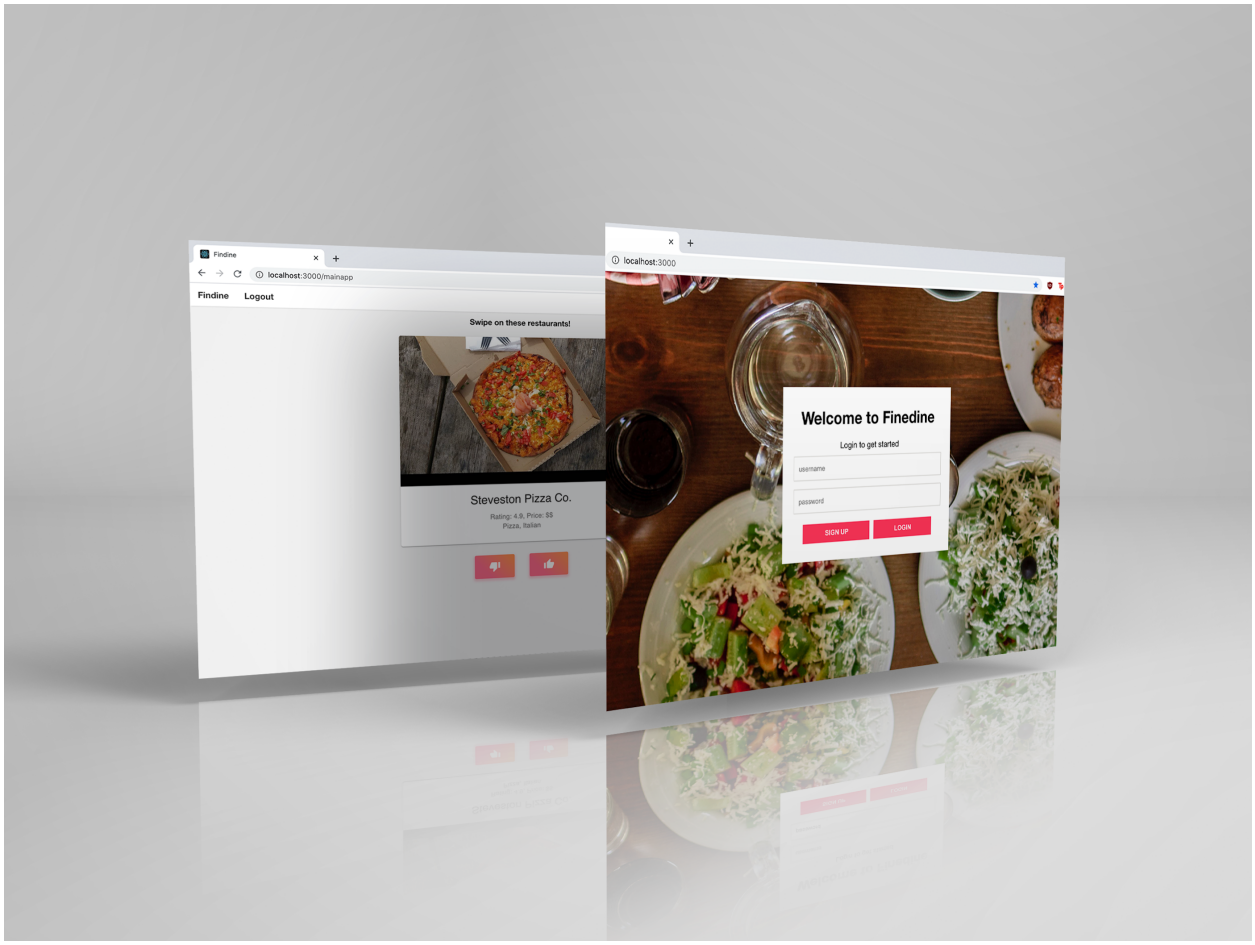
CMPT470 Final Project Report

Findine



Naomi Chao	nlchao@sfu.ca
Yu Xi He	yuxih@sfu.ca
Howard Chang	hhc27@sfu.ca
Hsuan-Yi (Eric) Lin	hyl29@sfu.ca
Henry Yip	hya59@sfu.ca

OVERVIEW

Findine is a restaurant suggestion web application with Tinder-like mechanics. Users can provide constraints such as cost, maximum travel distance, dietary options (vegetarian, gluten free, allergies, etc), or personal preferences, and the web application will then present users with suggestions. Users can anonymously swipe to like or dislike the restaurant suggestions of their choice. In the couple example, user A can pair with user B in order to find the “matching” restaurant. Once both users have swiped their restaurants, the application will display a suggested restaurant based on their preferences.

GENERAL ARCHITECTURE

Findine is a MERN application. A React framework is used as findine’s front-end to display our web application. Nodejs and Express.js are used as our back-end server framework to process client requests and responses. Lastly, Findine stores most of its data inside MongoDB, and a SQL database is used as an internal service maintained by Findine.

Users will first either sign up an account with us or log in to the application. Findine’s authentication feature is responsible for the sign up and log in features. This feature communicates with our AccountDB (SQL) when users log in and ProfileDB (MongoDB) when users sign up.

After access has been granted, users will be redirected to our main feature page where this page communicates with SessionDB to ensure the current user is connected. Then the server will communicate with RestaurantDB to obtain the restaurant objects for the page to parse. Users can then swipe on the restaurants they like or dislike and the information will be transferred to ProfileDB.

Upon entering the results page, users will be presented the one final suggestion. This is done by communicating with SessionDB to check sessions, ProfileDB to get liked restaurants from both users, and MatchDB to save the matched restaurants after comparing the two lists from two paired users. Users can also choose to see other matched restaurants by retrieving all matched restaurants from MatchDB.

Users can also navigate to see other users' comments for a particular restaurant, the forum page is populated through system retrieving objects from ForumDB.

TECHNICAL REQUIREMENTS

Internal and External Services

Internal Service: Our team has decided to maintain a SQL Database (accountDB) that stores lists of usernames and passwords. This SQL database is hosted locally inside our Google Cloud Platform Our authentication can then compare the user inputs with records from our database before directing users to our main feature

External Service 1: Our team has decided to host other databases on MongoDB Atlas so each team member can see data interaction in real time.

External Service 2: Our team has decided to populate our restaurant data with zomato api. We first call the api to parse the response into our restaurant model and store in RestaurantDB.

Authentication: Findine uses an internal MySQL database server to store account and login information for users. A user's username is the primary key for their account and they will be prompted for their username and password upon login.

Database:
We have implemented both SQL and mongoDB databases. The SQL database is used as an internal service on Google Cloud Platform. We have chosen to use MongoDB Atlas to host our other databases. Currently this includes AccountDB, ProfileDB, RestaurantDB, MatchDB, SessionDB, and ForumDB

1. AccountDB is hosted inside a SQL server on GCP, which consists of metadata such as username and password. The server then uses this to authenticate users during login. The username is the key that links to our profileDB.

2. ProfileDB consists of other user metadata including user first, last name, partner username, and other necessary information for the app to function. We also store the restaurants each user has liked within the main feature. ProfileID is the primary key that uniquely identifies each user. This is the most important database as it is used throughout the application.

3. RestaurantDB consists of restaurant information, images, tags (such as takeout availability, payment methods, etc), cuisine(s), price range, rating, and restaurantId being the primary key. Findine populates the RestaurantDB using

zomato's api to scrape for restaurants based on user location. Currently, it is set to Robson St. RestaurantDB is used in the main feature, results page, and also the forum.

4. MatchDB consists of usernames of current user and their partner. This is the database designed to store the coinciding liked restaurants between the two paired users, which the server will use to determine the best recommendation. The MatchDB is mainly used in our results page.

5. SessionDB inside MongoDB consists of users currently online using findine. This is how Findine persists login and determines the current user.

6. ForumDB inside MongoDB consists of feedback and comments for each restaurant where users can view and comment. The primary key being a restaurant as one restaurant should only have one forum.

Server-side Language:
Findine uses Nodejs and Express to process back-end calculation, request/response routings.

FEATURES

Registration

New users can sign up to use Findine with their specified username and password. Upon registration, they can login to use the application's features and will be assigned their own profile.

Login/Logout

Findine has its own authentication system so that users can access their account data and application features. The account will save information that pertains to a user for application functionality, including their first and last name and their partner for restaurant matching. To login, users will be prompted for their username and password.

Partner Matching

Findine features a matching system that allows a user to match and partner up with another user so that the two can decide on a restaurant to visit together. Currently the

prototype Findine v1 makes the users pair up with each other at sign up page. The Findine team will be updating this in the next iterations.

Restaurant Selection

The restaurant selection feature allows users to pick and choose which restaurants they would consider going to.

Upon entering the restaurant selection process of the application, a user will be presented twenty restaurants one by one. They will be presented restaurant cards that display a featured image, its price rating (based on Zomato's price range), and the type of cuisine(s). Users can then individually react to the restaurants with a thumbs up or thumbs down to express whether they would consider the restaurant or not. After the twenty restaurant options are swiped, the user will await their result.

The user's partner will also be doing the same on their end. Once the users have both finished matching, a final suggested restaurant result will be presented to them. This result is customized to their preferences from the matching feature of the app, as well as the restaurant swiping feature.

Restaurant Result/Suggestion

The results page will feature a suggested restaurant based on a user and their partner's preferences from the previous match and restaurant swipe feature. Information about the restaurant including its featured image, the type of cuisine(s), and the expense level will be presented. The user and their partner will now have the option of visiting this restaurant outside of the application, or if they are both unhappy with the suggested restaurant result, they may choose to try swiping again for a different result or click 'See More' to display all coinciding restaurants.

OVERALL COMMENTS

Findine's registration, authentication system, restaurant swiping feature all work as intended, and are scalable to include price/cuisine filtering and group restaurant selection (more than two users).

~~Currently, the lobby page allows for users to filter restaurants by price. However, please note that it only works on the admin account (below). Changing the price filters on the admin account will affect every account. To apply price filters, log in to the admin account and select price ranges (by default, all prices are selected). After pressing ready, refresh the next page once and the filters should apply.~~

Some challenges that our team faced included learning different tech stacks and libraries that we were unfamiliar with in a tight timeframe. One example would be react, our team initially underestimated how complex it might be to convert our html files over to react jsx files and the new application structure.

In addition, the lack of documentation and guides to deploying a MERN stack application on IIS significantly pushed back our schedule.

LOGIN CREDENTIALS FOR TESTING

Username: bobby470
Password: passmepls

This user will be paired up with our test user and is awaiting for Mr.Bobby to swipe.
Of course if Bobby or the TAs want to create an account of their own, please input “test” in the partner username at sign up page.

~~Admin account to access price filters:~~
~~Username: appleyapples~~
~~Password: 123123~~
~~For filter feature please read coments~~

APPLICATION URL

Findine External IPL: 34.67.154.164

Repository: https://csil-git1.cs.surrey.sfu.ca/hyl29/findine