University of Arkansas – CSCE Department
Capstone I – Final Proposal – Fall 2020

FooDecisive: A Choice for You

Aneesh Komanduri, Adam Greenfield, Huy Mai, Tay Pham, Zhi Han Weng

Abstract

A significant flaw with people is that we tend to be incredibly indecisive. Food is one example. It may seem trivial, but choosing what to eat stumps more people than you may think. For instance, you are going out with a friend for dinner or on a date, and you are too nervous about what makes the other happy, so you can't think of a good option for both of you. Not knowing what to choose can lead you to an awkward scenario where you may be unhappy with your meal. As computer science students, we recognize this problem. We will create FooDecisive to help save your time and make better decisions by effectively recommending a list of restaurants based on your preferences. One key feature in this app that will allow users to make better and quicker decisions is implementing a convincing display of recommended restaurants in a list of organized preferences and specify directions to restaurants on a Google Map. FooDecisive will save and learn users' search history and preferences to make a better recommender. To build FooDecisve, we will use technologies such as ReactJS for frontend, PostgreSQL for database management system, and Python Flask for the backend. To construct the recommender system, we will use distributed computing APIs such as PySpark on a Databricks notebook to perform machine learning and collaborative filtering for providing personalized food recommendations. The Yelp API will be used to download information regarding restaurant information such as locations, reviews, ratings, genres, etc.

1.0 Problem

People tend to have difficulties when facing what to eat, where it could be deciding between different cuisines or deciding between a long list of items on the menu. This problem that people experience is called choice overload, where people have trouble deciding when faced with a large number of options [10]. Our proposed idea, FooDecisive, would try to help the users reduce the impact of this problem.

According to the US Bureau of Labor Statistics, the time people spend on eating and drinking, and food preparation and cleanups are around an average of two hours per day [11]. With this in mind, we should not spend more time than necessary when deciding what to eat, other than actually eating or preparing. All that time could be spent on other activities, such as doing homework if you are a student or caring for others in the household if you are an adult.

As mentioned above, people's time would be wasted just on the problem of deciding what to eat. If the problem is not solved, people will spend an additional thirty minutes to an hour to decide what to eat, whereas on the other hand if the decision were already made, they could finish eating and all of the other activities related in around two hours. Even though that additional thirty minutes to an hour does not sound like much, the time would accumulate. According to a news article from the New York Post, the "average American couple spends 132 hours a year deciding what to eat [12]."

2.0 Objective

This project aims to create an application that will effectively provide a list of recommended restaurants to users based on their preferences. The application will be centered around a recommending system that will build on existing work to consider additional user preferences and display improved performance. The list of recommendations will be complemented with a map containing pinned restaurants to help users consider distance. In addition, to remedy the muddling of factors when considering a restaurant to dine at, the application will give users the option to view and save multiple lists as a way of organizing groups of preferences. This feature will require users to log into accounts to save the lists under their accounts.

3.0 Background

3.1 Key Concepts

Frontend

ReactJS [1]: React is a framework of JavaScript that was designed by Facebook. It is a declarative interface for building interactive UI's. Similar to other frameworks of JavaScript, React has a component-based architecture, where each feature can be modularized into a component. These components can then be used and reused wherever appropriate. React is a stateful framework, which means that it can keep track of the state of specific attributes. So, whenever an attribute changes, the render() method in React will automatically change the state of the attribute in the Document Object Model (DOM). Another benefit of React is that it follows a concept known as 'Virtual DOM'. This means that the DOM of a webpage does not need to refresh to update changes. React only detects the changes that have been made, compares these changes to the previous state that is stored and only changes those elements on the DOM that are different. This allows React components to run efficiently on the webpage without redundant changes. We want a user-friendly UI for our FooDecisive application, so React makes for a great choice. There will be features in the app that are similar, so modularity is key, which React provides.

Database

PostgreSQL [2]: PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. Postgres is optimal for scalable applications where tables in the database may contain millions of rows. As such, we decided to use postgres, since realistically if we wanted to scale our application to millions of users, we can do so with minimal cost and optimal performance.

Backend

Flask [3]: Flask is a lightweight microframework of Python that is often used for web development. It has a Jinja template engine that creates HTML templates that are returned to the webpage. Flask keeps things simple and in that it is very flexible. For FooDecisive, we want to use Flask to develop the backend of the application. We will use Flask to build REST API's in the backend. Security is important when creating a web application that stores data. We want to keep user data secure. Fortunately, Flask protects against one of the most common security problems: cross-site scripting (XSS). So, authentication will be made easy with Flask. Additionally, since Flask is a framework of Python and most machine learning development is done with Python due to the plethora of libraries, developing machine learning models and returning data to the webpage will be very convenient.

Distributive Computing

Databricks [15]: The Databricks platform is a simple platform to store and manage all of your data for analytics workloads. For the purposes of this project, a community version of Databricks will be used to train our recommender system. Databricks requires users to create clusters to run a Jupyter Notebook on. This allows a distributive computing approach that can handle large amounts of data.

Apache Spark [4]: We will be utilizing this popular distributed computing framework to train our recommender system model. Apache Spark is a unified analytics engine for large-scale data processing. To process the user ratings data, we will utilize an API in python called PySpark. We will establish a JDBC connection to databricks and feed in our ratings table to be processed using

PySpark. Once the model is trained and the predictions are calculated, the results (in a pyspark dataframe) are committed back to the postgres database via JDBC.

Concepts

Collaborative Filtering [5]: Collaborative filtering uses similarities between users AND items simultaneously to provide recommendations. It is a much better alternative to Content-based filtering, which strictly uses historical item preference of the given user to recommend new items. This doesn’t take into account how users may be alike and could have similar taste. Collaborative Filtering models can recommend an item to user A based on the interests of a similar user B. Each user and item get embeddings in a lower dimensional space for comparison. This is done by methods such as Matrix Factorizations to generate embeddings. The matrix for the user and item, respectively, embed the features of the user or item. The two embedding matrices are initially randomly initialized and updated after every iteration of optimization based on the loss so that the elements of the matrices represent accurate embeddings for users and items. This allows us to get a compact representation of users and their preferences and can allow us to predict that a given user would like a given item and ultimately recommend the top preferred items to a given user.

External APIs

Yelp API [6]: Yelp provides API’s that developers can use to build their applications while integrating Yelp features and real-time data. For FooDecisive, we use the Yelp Fusion REST API and the Yelp Open Dataset for user reviews and restaurants. The Dataset will be used to develop the aforementioned recommender system and the REST API will be used to retrieve information regarding restaurants such as location, reviews, ratings, genre, etc.

Google Maps API [16]: The Google Maps API is a service offered by Google to developers who want to embed the Google Maps Platform into their applications. For our application, we would like to show users a Google Map to show them directions to the specific restaurant that a user is viewing. The API allows embedding the maps into a UI and even allows location services via the Geocoding API so that users can view restaurants in their region.

WitAI API [17]: To increase interactivity with users, we will be creating a conversational chatbot. The chatbot will be responsible for understanding user intents and based on queries, extract information that can be fed into the Yelp Fusion API to search for restaurants that the user would like to view. Wit AI can be used to build natural language experiences and chatbots seamlessly. Wit AI also provides an HTTP API to send queries where a response is generated and sent back to the client. We will leverage this to send user queries to the API and use natural language generation to allow the chatbot to respond to the user.

Cloud Services Used

Heroku: Frontend (React) and Backend (Flask)

Amazon Relational Database Service (RDS): PostgreSQL

3.2 Related Work

Many people struggle to decide where to eat. As a result, there are several restaurant/food recommendation systems that aim to solve this problem. One basic implementation is known as the Wheel of Dinner [7]. This system does not recommend specific restaurants or take into account the user's location. It is merely a wheel that contains different types of food (e.g., Mexican, pizza, etc.) and allows the user to spin the wheel, landing on a random spot. This system is intended to be a quick and easy method to help people decide what to eat. To that end, the Wheel of Dinner is relatively effective. However, we can improve on this by taking into account the user's preferences and location, which will allow us to recommend actual restaurants near the user that they will have a higher probability of enjoying.

Perhaps the most popular recommendation system is Yelp [8]. Yelp allows users to review not only restaurants but also tradesmen (e.g. plumbers, mechanics) and a variety of other services. The website then recommends these restaurants/tradespeople/services based on users' reviews. Yelp is an effective service and is widely used for a reason, but the filtering can be improved. The user's preferences can be used to predict the rating a user would give to a restaurant, even if they have never visited the restaurant before. A more robust recommendation system can be built based on these predicted ratings, combined with community reviews.

Another notably similar system is TripAdvisor [9]. Like Yelp, restaurant recommendations are just one part of what TripAdvisor does. The website allows the user to enter a variety of preferences. However, there are some potentially useful preferences that are missing (e.g., distance). Again, we can use the user's preferences to predict their unvisited restaurants' ratings, which will result in more accurate recommendations.

The overall goal of FooDecisive is to build a recommendation system that is more accurate than similar implementations. We can take collaborative filtering a step further by asking the user the right questions and processing their answers and community reviews to recommend restaurants that they have the highest probability of enjoying. This will result in higher customer satisfaction and retention.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

Requirements

- Register/Login Page and authenticate credentials
- Provide a search bar to search restaurants
- User should be able to enter their preferences
- Provide a list of top-k recommended restaurants personalized to user
- Should be able to view location, ratings, genre, contact info, covid-19 protocols
- Integrate Google Maps to provide directions to a given restaurant (shortest path)
- Integrate Chatbot feature to request for restaurant information

4.2 High-Level Architecture

FooDecisive is designed to be a convenient food recommender system-based application that will be able to provide personalized recommendations to users. The initial page that a user lands on will be the Login/Register page. This page will allow a user to either login with their credentials if they already have an account or register for an account. User credential information will be stored in the PostgreSQL database.

Once logged in, the user is taken to the Home page. This page includes a search bar to search for restaurants in the local area. We plan on allowing location services to help with finding restaurants nearby. This page will also include a list of popular restaurants that a user might be interested in. This page will also include a conversational chatbot feature that allows a user to ask it for restaurants in the local area.

For a user to enter in their ratings for restaurants, there will be a ratings page that the user can navigate to and fill out the form to submit user ratings to the database. These ratings will then be processed in the backend by the recommender system and will return restaurants that are in line with the preferences provided. The top 15 most similar restaurants will be displayed on this page. These restaurants are then stored as Past Recommendations in case the user wants to know what restaurants they had visited in the past. A user will also be able to add restaurants they prefer to a Favorite Restaurants list. These restaurants can be deleted by the user as well.

Once these restaurants are displayed, the user can click on one of the restaurant options to navigate to a page that details the information of the restaurant including name, ratings, genre, COVID-19 protocols/resources, and a Google Map option to set directions to the selected restaurant. We plan to allow a QR code reader so users can conveniently pull up the directions on their phone by reading the QR code provided on the website.

The following sketches detail the functionality and UI of the components in the application:

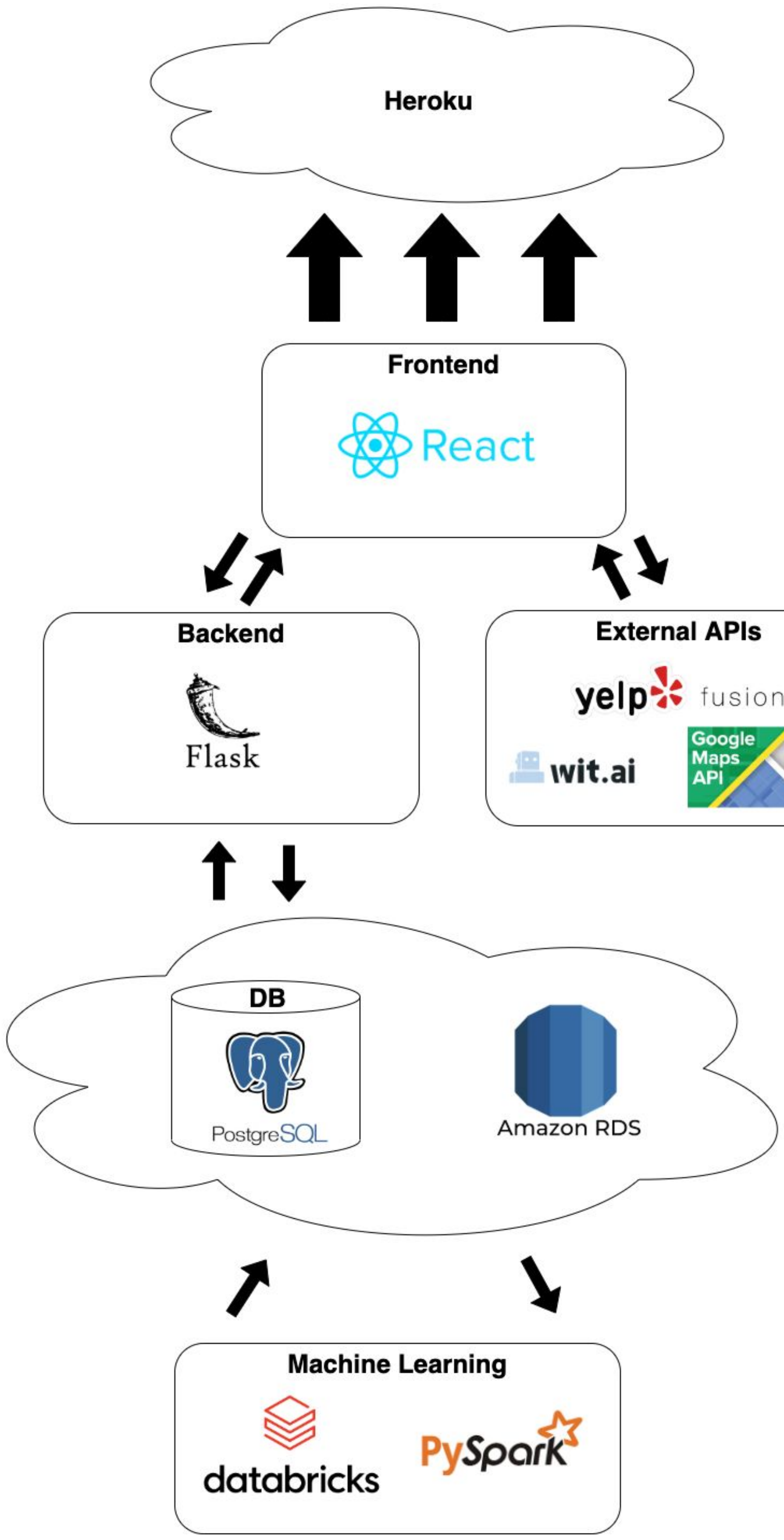
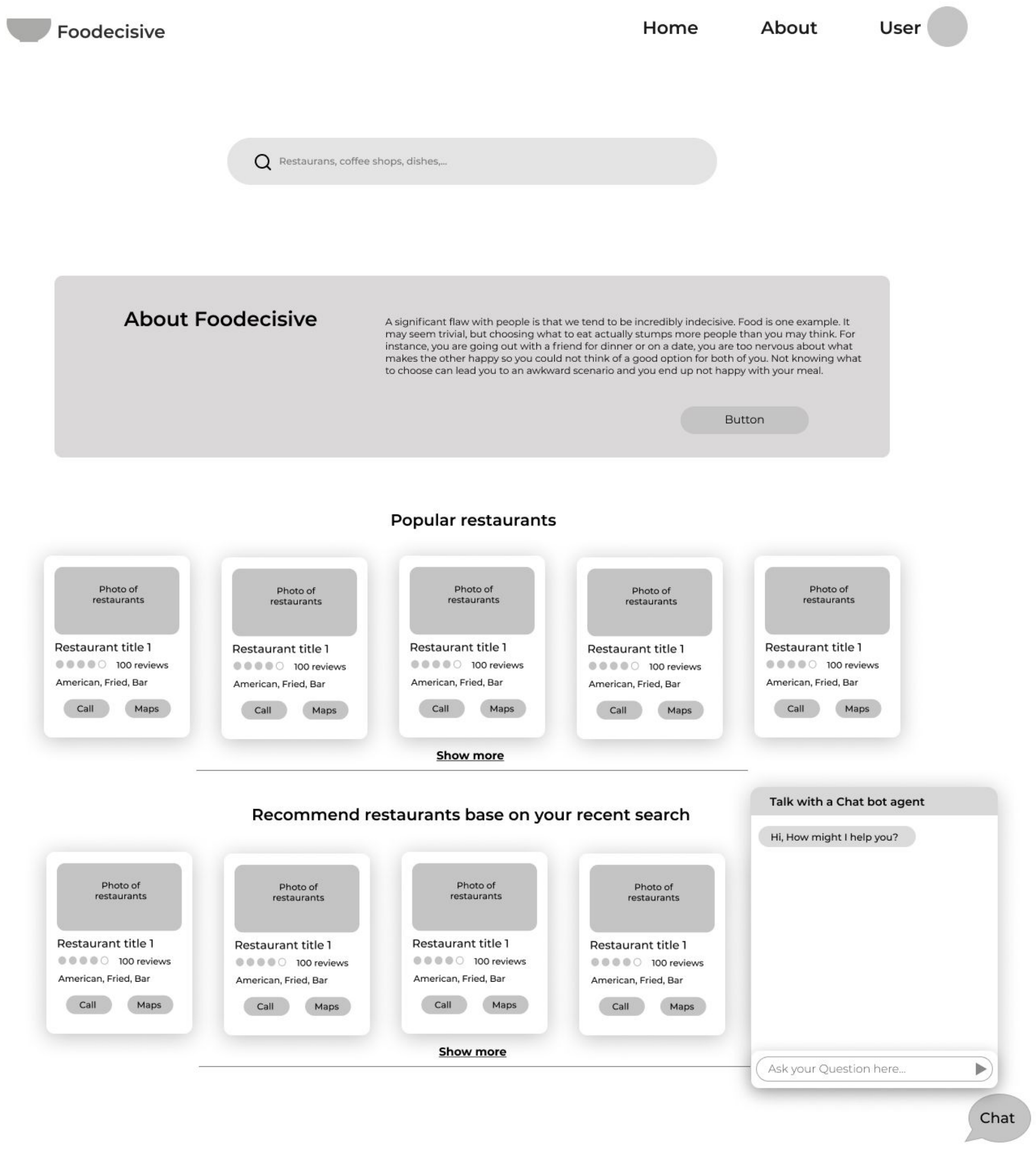

Heroku
Frontend
React
Backend
Flask
External APIs
yelp fusion
wit.ai
Google Maps API
DB
PostgreSQL
Amazon RDS
Machine Learning
databricks
PySpark

Figure 1: Overall High Level Design


Foodecisive
Home
About
User
Restaurans, coffee shops, dishes,...
About Foodecisive
Button
Popular restaurants
Photo of restaurants
Restaurant title 1
100 reviews
American, Fried, Bar
Call
Maps
Show more
Recommend restaurants base on your recent search
Talk with a Chat bot agent
Hi, How might I help you?
Ask your Question here...
Chat

A significant flaw with people is that we tend to be incredibly indecisive. Food is one example. It may seem trivial, but choosing what to eat actually stumps more people than you may think. For instance, you are going out with a friend for dinner or on a date, you are too nervous about what makes the other happy so you could not think of a good option for both of you. Not knowing what to choose can lead you to an awkward scenario and you end up not happy with your meal.

Figure 2: Home Page

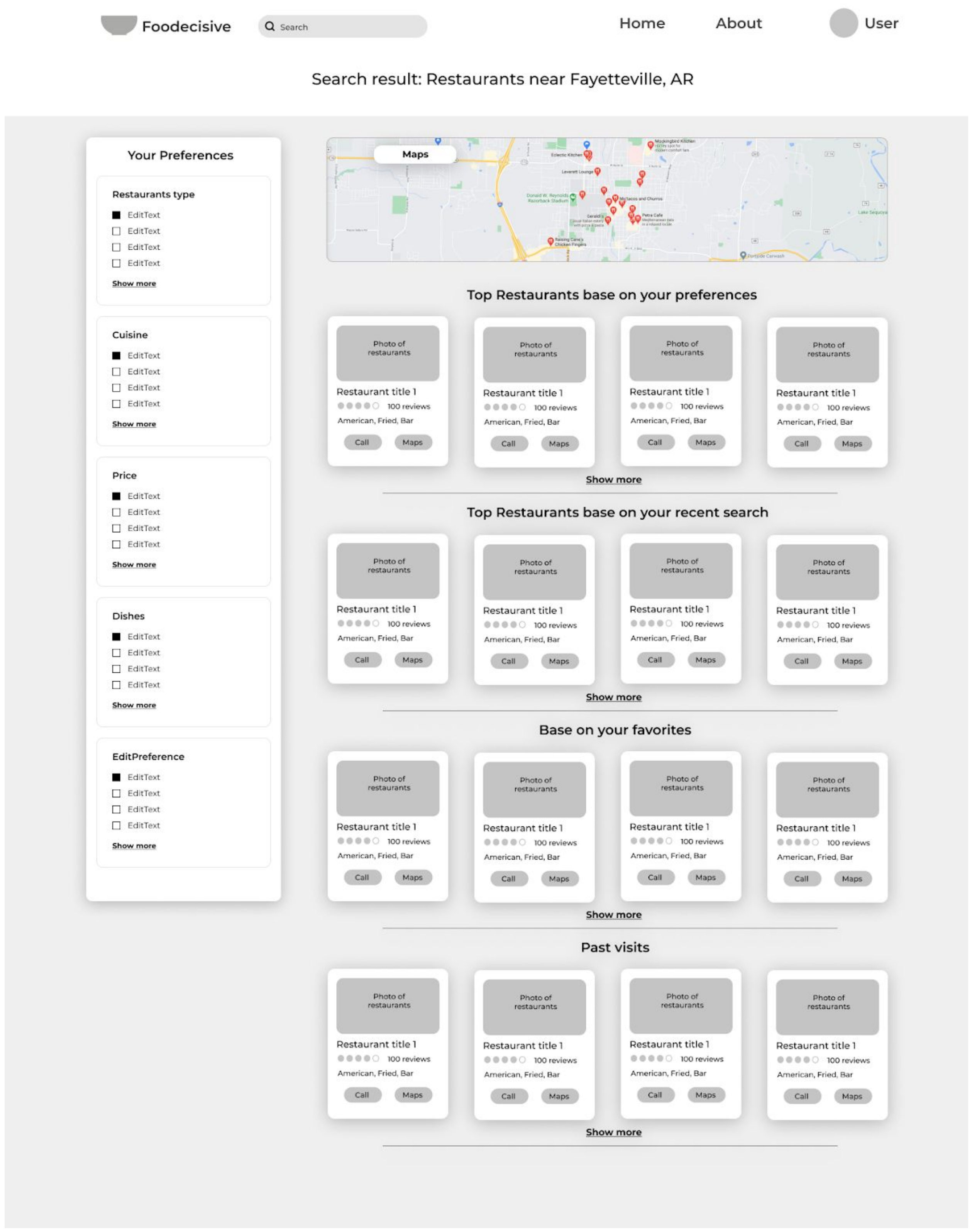


Figure 3: Preferences/Restaurant Detail Page

4.3 Risks

Risk	**Risk Reduction**
User Data Security	Flask protects against cross-site scripting (XSS)
Data Sparsity	Get multiple users to use the application to help get more accurate recommendations to users
Dimensionality & Complexity of Data	Use PostgreSQL for scalability and query optimization/performance

4.4 Tasks

1. Design of sketches/views of the application, flowchart
2. Create skeleton webpages for each view
3. Setup basic backend CRUD operations in Flask
4. Setup PostgreSQL, establishing database connectivity, and develop basic schema
5. Develop a User-Authentication system to validate user credentials
6. Research Collaborative Filtering Recommendation Systems
7. Preprocess Yelp data to include only restaurants, etc.
8. Populate Yelp ratings data into PostgreSQL database
9. Test Retrieval of restaurant data using CRUD operations
10. Use Yelp Fusion API to retrieve restaurant ratings, location, etc. (Testing) and integrate Google Maps API
11. Implement Chatbot feature for restaurant information retrieval
12. Begin development of recommender system in Python using PySpark and Databricks
13. Implement recommender system and deploy to interact with incoming data from frontend
14. Polish up frontend and add clean and aesthetically pleasing design elements
15. Connect frontend and backend through middleware for communication

4.5 Schedule

Tasks	**Dates**
1. Research recommender systems and design sketches of UI	11/14-11/28
2. Exploratory Data Analysis of Yelp Dataset in Jupyter Notebook	11/28-12/7
3. Establish database schema and connectivity to backend	12/7-12/14
4. Start to develop recommender system with PySpark using user ratings and reviews	12/14-12/28
5. Continue to tune model hyperparameters for ALS algorithm	12/28-1/11
4. Implement Home Page (Search Bar, Initial Restaurants recommended, Toolbar, Preferences page) and continue recommender system development	1/11-1/25
5. Implement User Auth and Login/Registration Page	1/25-2/8
6. Connect frontend and backend and test data retrieval from frontend	2/8-2/22
7. Implement the Chatbot and integrate Google Maps API	2/22-3/1

8. Test functionality of Recommender System given user preferences	3/1-3/14
9. Polish up frontend for aesthetic look	3/14-3/28

4.6 Deliverables

- Design Document: Explains what our code does.
- UI drawings and flowcharts
- Database schema and initial data
- React and Flask code
- Jupyter Notebook for Recommender System (Python/PySpark)
- Final Report

5.0 Key Personnel

Adam Greenfield – Greenfield is a junior/senior Computer Science/Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has taken relevant courses such as Artificial Intelligence, Software Engineering, and Database Management Systems. He has completed relevant work in these courses. Greenfield will be responsible for backend development on the FooDecisive project.

Aneesh Komanduri – Komanduri is a senior Computer Engineering, Computer Science, and Mathematics major at the University of Arkansas. He has completed relevant coursework in Software Engineering, Database Management Systems, Numerical Linear Algebra, and Data Mining. He has internship experience as a Digital Security & Cloud Engineering Intern at Phillips 66 and research experience as a Research Assistant in the Data Science Lab. Komanduri will be responsible for development of the recommender system, assisting with backend/database development, and cloud deployment of the ML model.

Huy Mai – Mai is a senior Computer Science and Mathematics major at the University of Arkansas. He has completed relevant courses such as Software Engineering, Data Mining, Symbolic Logic I, and Abstract Algebra. He has internship experience as an Application Development Intern at J.B. Hunt Transport Services, Inc. and as a summer research intern at the

Autonomous Technology Research Center in Dayton, Ohio. Mai will be responsible for the development and deployment of the recommender system and database development.

Tay Pham – Pham is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Software Engineering, Database Management Systems and Operating Systems. He has job experience as a teaching assistant at University of Arkansas in Fort Smith. Pham will be responsible for assisting with frontend and backend development on this project.

Zhi Han Weng – Weng is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as System Synthesis & Modeling, Operating Systems, Circuits & Electronics, Low Power Digital Systems, and Software Engineering. He is currently working in a research lab for Dr. Jia Di in preparation for attending graduate school. Weng would be responsible for assisting with frontend and backend development.

6.0 Facilities and Equipment

We will be using the Pinnacle Cluster at the Arkansas High Performance Computing Center to train our recommendation system model.

7.0 References

[1] "React - A JavaScript library for building user interfaces", React, https://reactjs.org/

[2] "PostgreSQL: The World's Most Advanced Open Source Relational Database", https://www.postgresql.org/

[3] "Flask - web development, one drop at a time", https://flask.palletsprojects.com/en/1.1.x/

[4] "Apache Spark: Lightning-fast unified analytics engine", https://spark.apache.org/

[5] "Collaborative Filtering | Recommendation Systems" https://developers.google.com/machine-learning/recommendation/collaborative/basics

[6] "Yelp Developers", https://www.yelp.com/developers

[7] "Wheel of Dinner", https://wheeldecide.com/wheels/food-drink/wheel-of-dinner/

[8] "Yelp", https://www.yelp.com/

[9] "Tripadvisor: Read Reviews, Compare Prices & Book", https://www.tripadvisor.com/

[10] "Scientists Uncover Why You Can't Decide What to Order for Lunch." California Institute of Technology, https://www.caltech.edu/about/news/scientists-uncover-why-you-cant-decide-what-order-lunch-83881

[11] “Time spent in primary activities and percent of the civilian population engaging in each activity, averages per day by sex, 2019 annual averages.” U.S. BUREAU OF LABOR STATISTICS, https://www.bls.gov/news.release/atus.t01.htm

[12] “American couples spend 5.5 days a year deciding what to eat.” New York Post, https://nypost.com/2017/11/17/american-couples-spend-5-5-days-a-year-deciding-what-to-eat/

[13] “Maps Javascript API”, https://developers.google.com/maps/documentation/javascript/overview

[14] “Collaborative Filtering Recommender Systems”, http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf

[15] “Databricks”, https://databricks.com/

[16] “Google Maps Platform”, https://developers.google.com/maps

[17] “WitAI”, https://wit.ai/