MyConcierge: A Chatbot for Group Restaurant Recommendations

Krish Chhabra, Arpan Kumar, Daniela Martinez Banda, Aubree Rodriguez, Ravish Shardha, and Will Thompson

Department of Computer Science and Engineering, Texas A&M University

CSCE 482: Senior Capstone Design

Dr. Paul Taele

February 18, 2024

Abstract

Feeling like you are a part of a social group is an important aspect of an individual's mental health, with a key contributing factor being group gatherings such as dining together. Especially in our post-pandemic society, the need to facilitate in-person social interactions is paramount to the well-being of individuals and communities. MyConcierge seeks to facilitate these gatherings by minimizing the difficulty of selecting a restaurant suited to the preferences of a group. This chatbot-driven restaurant recommendation system leverages the intuitiveness of the increasingly popular chat interface and the extensive data provided by Yelp and Google Maps in order to create a unique and informative user experience. Evaluation of the application was performed through user studies with 20 participants (including 2 focus groups of 3-4 people), ultimately finding that the application was intuitive, easy to use, understood user preferences, and provided quality recommendations. The greatest areas of improvement uncovered from these studies stemmed from shortcomings in the current state of large language models such as difficulty parsing negative queries and slow processing times. Overall, the project demonstrated success in employing a low barrier-to-entry and free-form interface in a domain that generally requires a rigid profile creation process with exhaustive questionnaires.

Introduction

When traveling to a new city, it is always a challenge to find good restaurants outside of the large chains. You likely want to get a feel for where locals in the area like to eat for a unique experience. The task becomes even more challenging when traveling in a group as pleasing the entire group can be time-consuming, overwhelming, and difficult. Our virtual concierge service removes the headache and streamlines this decision process; it is now quick and easy to find restaurants in a new area that fit the group's preferences while still providing quality restaurant recommendations. In addition, this functionality is packaged into an easy-to-use interface that mimics traditional interactions with a hotel concierge. Users simply have a quick chat with the virtual concierge and instantly get a map of the city showing which restaurants are best for you and your group. Quickly look at some photos, pick the perfect spot for you, and let Google Maps lead the way!

Domain Context

The current implementations of digital concierge services focus on providing hotel room services for a single room. These services are commonly provided through a touchscreen device that is located in the hotel room and provides services like making reservations, room service orders, and itinerary planning (Hollander, 2023). Customers are expected to already know what they want to order as these interfaces lack the ability to explore food options in the area. Our solution specifically aims to help a group of individuals find restaurants in a new city that meet some criteria. The group interacts with a concierge chatbot by either electing a single person to "speak" for the group or each person providing their input. The group would then receive tailored recommendations making it easy for the group to find good restaurants in a new location. With only 1.95% of luxury hotels providing a concierge service in the US, and 46% of

users on Hotels.com reporting that they would use a digital concierge, there is a large desire for such an application as ours (Kathy H, 2017).

Case Scenarios

Imagine a group of friends traveling to Austin, Texas for the first time. They arrive in the city and want to get something to eat, but they do not know what is in the area and only recognize the names of chain restaurants. One of the members of the group pulls up the MyConcierge website and starts chatting with the virtual concierge. After providing the city they are in, they type up a few sentences describing the group's preferences such as what cuisines they like, any dietary restrictions they may have, whether they are feeling adventurous or want something reliably good, etc. Once the concierge has enough information, it will direct them to a map view of the city with various recommended restaurants marked. They can easily view some photos and general information about the restaurants to find the best fit. Once they have found one, they can click a link that provides directions to the restaurant through Google Maps.

Goals and Constraints

The main goal of this project is to enhance a group's decision-making ability in a fraction of the time. We aim to facilitate decision-making for groups in unfamiliar locations by providing tailored restaurant recommendations based on their preferences and dietary restrictions. Other goals this project focuses on include creating a user-friendly web interaction, diversity in recommendations, and scalability for large groups of people.

In order to achieve these goals, we must consider the data privacy of our users. Because of this, we have decided to avoid a user-login system which would also serve as a barrier to entry for new users. Another constraint is the need for recommendations to be not only accurate but also quickly delivered to users. This constraint will need to be a major consideration when

constructing our recommender system and chatbot. Finally, the acquisition and storage of restaurant information/data for the project must be done in a manner that is efficient, inexpensive, and abides by data access regulations and policies.

Solution Summary

Our virtual concierge service addresses the challenge of group dining decisions in unfamiliar cities or large groups with different preferences. By leveraging a chatbot interface, users can easily express their preferences inherently, allowing the system to generate tailored restaurant recommendations. The system is designed to mimic asking a traditional concierge for recommendations to facilitate user-friendly and intuitive interactions. Once the virtual concierge has a good understanding of the group's preferences, the group will be directed to a map with pinned restaurant recommendations, analogous to the concierge pulling out a map to circle restaurants of interest. The key to our solution lies in the careful balance between the traditional and virtual worlds, providing more information than a traditional concierge could in an interface that is more intuitive and approachable than modern virtual solutions.

Evaluation Summary

In terms of evaluation, our comprehensive plan encompasses both functionality and usability aspects. On the functionality front, we will rigorously assess the accuracy of the recommendation algorithm, ensuring it aligns with all the users' preferences. This evaluation will be done through a combination of "Ranking Metrics" such as "Hit Rate" and empirically derived "User Satisfaction Metrics" (Kumar & Sharma, 2016). To evaluate the chatbot functionality, we must conduct user satisfaction studies in addition to objectively evaluating the ability of the underlying algorithms to extract only the important information and sentiments from user queries. Along with the accuracy of these models, response time is a critical factor in user

satisfaction and will be evaluated to guarantee quick and efficient interactions. From a usability perspective, our evaluation plan includes a detailed survey of the user interface, seeking feedback on clarity, simplicity, and intuitiveness. User satisfaction will be gauged to understand overall impressions and ease of use, while specific attention will be given to how well the system accommodates group dynamics and the decision-making process. When conducting these user satisfaction surveys, users from a broad range of ages, knowledge, and abilities must be included in order to obtain a good measure of the accessibility of the final product.

Project Management

Roles and Responsibilities

Our team is organized into 4 unique responsibility groups with 6 unique technical roles. The technical responsibility groups consist of algorithms specialists, front-end, back-end, and full-stack developers. Beyond these technical responsibilities, there will be non-technical responsibilities such as documentation review, schedule monitoring, and quality control. In terms of project management, we have planned to take the group/democratic approach to leadership, with area experts guiding discussions and decisions that fall under their purview. Our team believes that this management model will help foster success and reduce conflict within the team. Because there will not be a hierarchy in place, each teammate can feel more comfortable when introducing an issue or concern throughout the development and implementation process.

The algorithms specialists consist of Will and Krish, with each focusing on the development of a particular algorithm. Will will be responsible for researching and developing the natural language processing (NLP) model which will serve as the backbone for the concierge chatbot. With this responsibility, Will must design a model to extract important requests and sentiments from user chats, prompting users as necessary to retrieve all data needed for quality

recommendations. Krish will be responsible for the algorithms behind the recommendation system. The model he develops will transform Will's extracted requests and preferences into restaurant recommendations that can be presented to the user. Additionally, Krish will need to aid Arpan and Ravish in the acquisition and storage of restaurant data for the model.

Daniela and Aubree will lead the front-end development of the project. As such, they will be responsible for designing and implementing the user interface for the application. Two main interfaces will be divided between the two developers: the chatbot and the presentation of the results. Daniela will be responsible for the chatbot interface, which is meant to mimic modern messengers and flow as though the user is talking with a real person. Aubree will be responsible for the result presentation, which consists of the map of restaurant recommendations as well as the accompanying textual display of available restaurants that best match the user's queries. Since these forward-facing interfaces will require careful design and implementation, we envision most team members will aid in front-end development towards the end of the project.

The back-end development will be headed by Arpan. This aspect of the project will handle the restaurant data collection and storage so that the recommender system can be used to filter accurate results and present them to the user. As of right now, only the restaurant data needs to be stored in a database since there is no collection of user information; however, this may broaden as the project progresses and the scope potentially grows.

The full-stack development will be handled by Ravish. He will be the main facilitator of communication between the two sides of the website, the front, and back end. Ravish will aid any developers who need advice or recommendations and complete some tasks that qualify as front-end development and back-end development. He will be in charge of having a broad view of the system's algorithms and tools in order to narrow down the list of results presented on the

front-end side of things, with an interactive map and dynamic restaurant display. On the front end, his main responsibility is determining and implementing the best way to display restaurant information so the user can more easily choose from the curated selection given by the concierge.

In addition to technical tasks, non-technical responsibilities include reviewing documentation, monitoring schedules, ensuring quality control, conducting accessibility analysis, managing stakeholder relations, and overseeing scope management. Krish will be charged with reviewing documentation and ensuring all information is accurate and presented in a manner that meets the necessary requirements for delivery. Will will be responsible for monitoring the team's ability to create a schedule and adhere to it. Aubree will handle quality control and ensure that all elements of the project meet predetermined standards and requirement specifications. Daniela is charged with analyzing the site's accessibility level and ensuring that the site can be easily used by people of all technological, cognitive, and physical abilities. Arpan is handling the stakeholder relations and facilitating communication with Professor Taele and any potential users of the site, ensuring our product properly satisfies the necessary requirements. Finally, Ravish will be managing the scope of the project. He needs to have an understanding of the progress of each member to determine whether the scope of the project needs to be adjusted.

Software Development Methodology

With a team of six team members, there will be quite a bit of development done in parallel so constant communication and collaboration between each member is vital. Our team decided to tackle this issue by utilizing a modified version of Agile development. Rather than having a dedicated scrum master, people in each role will guide discussions and decisions with subject matter that falls under their area of expertise. We feel that this organization is ideal as

members with the deepest understanding of a certain aspect of the project will be able to best decipher what must be discussed and which directions to pursue.

One principle of the Agile development methodology that will be greatly beneficial to the project is its focus on adaptability and the fact that it results in consistent minimum viable products (MVPs) at the conclusion of every sprint. Currently, our team has a solid high-level understanding of what's required for the project; however, our assumptions are still very far off from what is required. Agile development allows the flexibility to change our roadmap as we undergo this development journey and receive user feedback while also maintaining that we consistently are producing a tangible product. For sprint 1, our team is planning to solely focus on a single-user experience. In future sprints, we may expand this to include other functionalities such as multi-user access if we feel that it is an achievable goal. This approach will help balance what is possible to develop in the allotted time for the project with user needs as specified by feedback on MVPs at the end of each sprint.

Requirements

The primary objective of our virtual concierge service is to simplify the process of finding suitable restaurants for groups in new cities. Our solutions focus on providing a quick and efficient way for users to discover local dining options that match their preferences while streamlining the decision-making process. This section gives an outline of the problems we are trying to solve with our project based on user stories and defines our key features and goals based on how we want the user experience to feel.

The first feature that will be implemented in our design is a chatbox interface where users will be able to interact with a virtual concierge, mimicking traditional interactions with a hotel concierge. Our next feature is a personalized recommendation algorithm; based on user

communications with the virtual concierge regarding restaurant requirements and preferences, our algorithm will quickly and accurately provide a list of recommended restaurants that fit our user's needs and criteria. Similar to the restaurant recommendation application described by McCarthy (2002), our application will be focused on recommendations for groups. Because of this focus on group recommendations, additional emphasis must be placed on individual user privacy as members of the group may not be comfortable sharing their preferences with other members. The final key feature of the application is an intuitive yet comprehensive interface to retrieve restaurant information. Users should be able to easily view a representative picture of recommended dining options in their location of interest through some form of map interface. Detailed information about each recommended restaurant including photos, hours of operation, etc. should be available upon user interaction.

User Stories and Usage Scenarios

Based on the expected interactions with the application, we have developed a list of user stories and usage scenarios for a wide range of potential users in our application's target audience.

1. John (Business Traveler) - Discover Nearby Gems - Enhance Experience

   John, a business traveler, arrives in a new city. He engages with the virtual concierge, expressing his interest in local cuisine. The system suggests unique nearby restaurants, enhancing John's unique travel experience.

2. Emily and Friends - Streamlined Decision-Making - Time Efficiency

   Emily and her friends, unfamiliar with Austin, Texas, use the virtual concierge to pick a place to eat. The system quickly provides tailored restaurant recommendations, saving time and ensuring a cohesive group dining experience.

3. Alex (Food Enthusiast) - Diverse Recommendations - Culinary Exploration

   Alex specifies the desire for unique and diverse dining options throughout the city. The concierge recommends a mix of restaurants, considering various cuisines and atmospheres.

4. Lisa (Health Conscious) - Dietary Restrictions Considered - Stress-free Dining

   Lisa is Muslim and as such has specific dietary restrictions. She uses the virtual concierge to help her decide where to eat in a simple stress-free manner. The concierge considers her dietary needs and recommends restaurants aligned with her restrictions.

5. Family with Kids - Family-Friendly Suggestions - Hassle-free Outing

   A family with kids uses the concierge to find family-friendly restaurants. The system considers preferences suitable for children, ensuring a hassle-free and enjoyable outing.

6. Maria (Solo Explorer) - Solo Dining Adventure - Discover Local Favorites

   Maria likes to see new places as a solo traveler and discover local favorites, enriching her travel experience.

7. Chris (Event planner) - Group Event Planning - Efficient Decision-Making

   Chris is planning a group event and uses our concierge to efficiently make dining decisions that cater to diverse preferences within the group.

8. Tourist Couple - Romantic Evening - Simple Memorable Date

   Sarah and James use the visual concierge to plan a romantic evening, ensuing a memorable date with restaurant recommendations tailored to their preferences.

9. Vegetarian Group - Vegetarian-Friendly Options - Inclusive Dining

   Samantha and her vegetarian friend utilize the concierge to find restaurants with diverse vegetarian-friendly options, ensuring inclusive and satisfying dining experiences for all.

10. Mike (Adventure Seeker) - Adventurous Dining - Culinary Exploration

    Mike asked the concierge to discover adventurous dining options known for certain dishes or weird tastes, enhancing his travel experience through culinary explorations.

11. Business Group - Corporate Dining - Impress Clients

    A corporate team uses the virtual concierge to find upscale and impressive dining options for a business meeting, ensuring a successful and memorable client experience.

12. Raj (bored college student) - Diverse Cuisine Exploration - Enrich college Experience

    Raj is tired of eating the same takeout food during finals and asks the virtual concierge for other fun places that he can try.

13. HR Manager - Team Building Event - Inclusive Dining

    An HR manager uses the virtual concierge to plan a team-building event with inclusive dining options, ensuring a cohesive and enjoyable experience for the entire team.

14. Kim (Fitness Enthusiast) - Healthy Dining Choices - Easy Fitness Live

    Kim, a fitness enthusiast, engages with the virtual concierge to find restaurants with healthy and fitness-friendly dining choices, that align with her wellness goals.

15. Elderly Couple - Simple Dining - Easy Decision-Making

    Evelyn and Walter, an elderly couple, engage with the virtual concierge to find simple and uncomplicated dining options, ensuring an easy and stress-free decision-making process.

16. Daniel (Sole Business Traveler) - Solo Business Dining - Efficient Choices

    Daniel, a solo business traveler, uses the virtual concierge to efficiently find dining options suited for solo business meals, optimizing his time during work-related travel.

17. Senior Book Club Gathering - Casual Fathering - Comfortable Atmosphere

A senior book club uses the virtual concierge to find a comfortable and casual dining venue for their gathering, ensuring a relaxed and enjoyable atmosphere for their members.

18. Emma (College Student) - Budget-Friendly Options - Affordable Dining

    Emma, a college student, engages with the virtual concierge to find budget-friendly dining options, ensuring affordability and financial convenience.

19. Student Group Session - Study Session Fuel - Quick Bites

    A group of college students uses the virtual concierge to find quick and convenient dining options suitable for a study session, ensuring efficient fueling during their academic activities.

20. Graduate Celebration Dinner - Family inclusive place - Special Occasion

    Graduating college students use the virtual concierge to find upscale and celebratory dining options for a special graduation dinner, ensuring a memorable and fitting end to their academic journey.

While this list of user stories is not exhaustive, it provides examples of how our application is intended to be used to better define the requirements and necessary features of the project.

Definition of Success

Defining what constitutes a successful project requires breaking it down into two categories: project management success and product success (Bannerman, 2008). Both aspects are vitally important to the success of our group throughout the semester. It is important to have a great product by the project's conclusion, but how we get there is just as important.

The first part relates to how well we did as a team at managing the project throughout development, including how well we managed our budget, scope, and time. We realize that completing this course is more than just completing a software project; it is also focused on learning how to work effectively in a group using methodologies such as Agile development. We will have been successful in this regard if we are able to keep a steady pace each week, with each person completing multiple stories per week to ensure that deadlines are met. We will know if we have done this correctly if we are not cramming and rushing to complete the project in the final week. We also strive to stick to our initial project scope, making sure that we do not implement unnecessary features before ensuring that the base product is achieved. Feature creep is common in software development, so we will work hard to stay focused on the task at hand while being sure to stay within budget. At the moment, we do not require any budget and we hope to keep it as low as possible moving forward. As for evaluating if we succeed in this area, Bannerman (2008) provides the following: "The conventional approach is that an assessment of performance is made in a post-project review based on whether the project was completed 'on time, within budget and to specification.'"

We now move on to how we would evaluate the success of our product. For one, we aim to have a quick and easy-to-use virtual concierge that will help people find good restaurants in a new area. To evaluate how easy our application is to use, we will employ techniques such as first-click testing to see where users click first when trying to complete a task (usability.gov, 2013a). This will show us how a new user naturally navigates our application without instructions. Our goal is that they are able to get restaurant recommendations that are meaningful to them without feeling confused or frustrated throughout the process. We will also use surveys to gain better feedback on what people liked and disliked after using the application. We will

consider these recommendations as we evaluate our site, make changes, and retest to see if it has improved (usability.gov. 2013b). The other aspect of measuring success is whether "the deliverable … does not provide sufficient benefit to [the users]" (Bannerman, 2008). Ultimately, this project is supposed to make the process of exploring restaurants in a new area easier, so we must ensure this goal is met. One way that we will evaluate our recommendation system is by using a metric called hit rate; the hit rate is a simple metric where "if one of the recommendations in a user's top-end recommendations is something they actually [liked], you consider that a hit" (Chokhra, 2021). We will see how many of the top recommendations the user finds interesting and use that to evaluate the effectiveness of the system. Another metric that we will find useful is diversity, which is "how broad a variety of items [our] recommender systems is showing to users" (Chokhra, 2021). We want our system to provide reliable results while also giving the user some more creative recommendations as well. Evaluating how well our natural language model is able to figure out what the user wants will be easily evaluated by looking at the recommendation results. If they do not contain anything that the user mentioned, we will know that the model is not working properly. We will again employ the hit rate metric to determine how many keywords from the original text are parsed by the model. Our model should be able to determine what type of food users do and do not want while ignoring unimportant information.

Design Exploration

With the requirements of the project in mind, we can begin discussing how our team plans to fulfill the product requirements outlined above. These discussions involve careful considerations of possible solution designs, ultimately culminating in an in-depth presentation of the design we believe will best suit the project's requirements. In the creation of these designs,

we acknowledge that inclusivity to a wide range of users is essential for our product to have a meaningful impact on our target audience. As such, we will additionally outline key inclusion and accessibility considerations that we must be cognizant of throughout the design, implementation, and testing of our product.

Comparison of Potential Solutions

The problem we seek to solve, considered in its simplest form, is that it is difficult for groups of individuals to decide where they would like to eat when in an unfamiliar location. When considering potential solutions to this problem, we found the key decision to be made dealing with the manner in which user information would be collected. On one side of the spectrum lies a rigidly structured survey while on the other is a completely freeform interface. In order to balance the challenges and benefits of both of these options, our solution falls somewhere towards the center of said spectrum.

Our initial consideration for the project involved a rigid questionnaire that needed to be filled out by each user of the group. From a computing standpoint, this solution offered the easiest implementation. Assuming users accurately answer the questions in the survey and that the survey is comprehensive of any preferences the user could have, all that needs to be done is to aggregate the results of individual users into a group consensus and filter a restaurant database to find the most appropriate recommendations. From a user's perspective, however, the process of filling out the survey is very slow and off-putting and serves as a major barrier to entry for using the application. Additionally, this interface is not much of an improvement over traditional ways a group would decide on a restaurant, instead merely reformatting the same experience.

The second potential solution takes a completely opposite approach, opting for a barebones interface with merely a text box to enter preferences into. After users enter the

guidelines for the restaurant they would like in a free-form manner, the group would be provided with the recommended restaurants based on their aggregated preferences. The main problem with this interface is the fact that users have no guidance or feedback as to what information the application needs in order to provide accurate recommendations. In the worst case scenarios, the user could provide too little information, information that is not relevant to the recommendation of restaurants, or too much information as they struggle to collect their thoughts into a cohesive query. In the best case, the application would serve as little more than a search bar, with users filtering their thoughts into keywords as they would with traditional search engines they are likely accustomed to. The user experience that this solution would provide would range from negative to neutral, without providing much benefit to users over readily available services in either case.

The final approach, which we have decided to pursue, involves the collection of information through a chatbot interface. In this solution, the user communicates with a chatbot in order to describe the preferences of the group. The chatbot will prompt the user with requests for information that has not yet been provided but is needed for the system to make informed recommendations. This solution allows for an interaction that is likely familiar to the user as it mimics interactions with a traditional concierge that would provide recommendations. Additionally, the user is not bogged down with the tedious questions of a survey while also not having to guess what information the application needs due to the real-time feedback. Overall, this solution maximizes user satisfaction with recommendations while minimizing user frustration and barrier-to-entry during the data collection phase.

Lo-fi Prototyping

Home/Landing Page

The landing page (Figure 1) will be the first impression and initial attraction point of the website. The visuals of this page should be inviting and invoke the feeling of approaching a traditional concierge in a hotel lobby. In addition to being visually appealing, this page should establish the ringable bell icon that will be used throughout the rest of the application. This bell reflects a bell that would appear on a traditional concierge's desk which can be rung to get their attention. Analogously, the bell icon in our application will be employed for the user to receive the "attention" of the virtual concierge, initiate a new conversation with the chatbot, or return to an ongoing conversation to provide additional requests.

Figure 1

Home/Landing Page: Initial Sketch Vs. Finialized Wireframe

Chatbot Page

The chatbot page (Figure 2) will be the hub of interaction between the user and the concierge chatbot. The user will select the chat box at the bottom of the screen and enter any information relevant to their restaurant search such as how many people are in their party, what kind of cuisine they want to eat, where they are located, etc. Once the user feels like they have given the concierge a sufficient amount of information and the chatbot has no further follow-up

questions, the user will be directed to the map page (Figure 3) to view the restaurants that best fit their preferences.

Figure 2

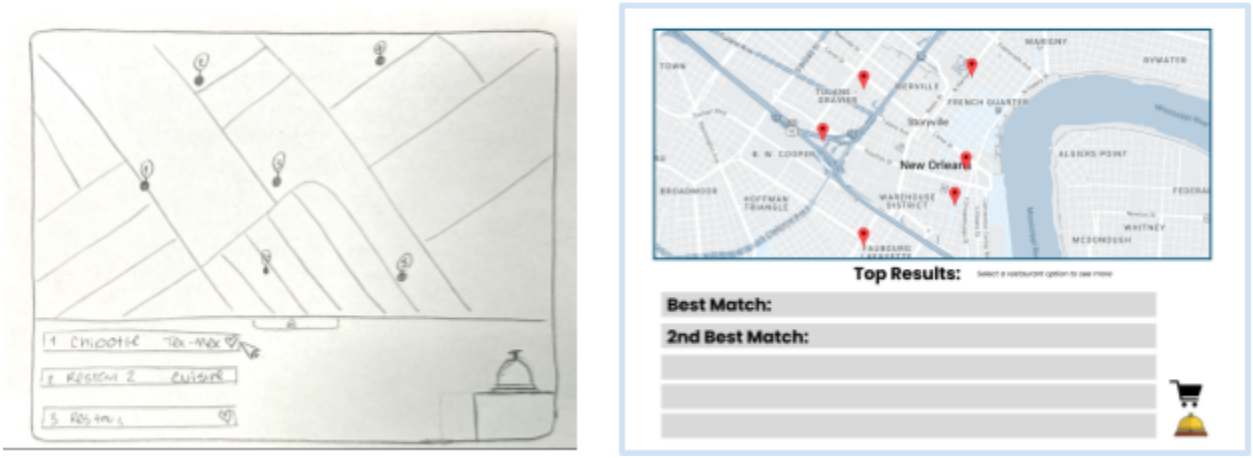
Chatbot Page: Initial Sketch Vs. Finialized Wireframe

Map Page

The map page (Figure 3) will display the top 3-5 restaurant candidates that the concierge believes best fit the user's requirements. There will be an interactive map that will have pins indicating where each restaurant is located, with the top choice being gold and the remaining being orange. Below the map, there will be a list of the top restaurant candidates that can be selected by the user to see more information. At any time during the process, the user can select the bell icon and return to the chatbot page (Figure 2) to clarify any requirements or to modify their preferences. One possible addition to this page could be a slider of sorts that would allow the user to customize how many restaurant recommendations are displayed on the interactive map (i.e. the user can set some value *n*, and the map and table below will be adjusted to display the top *n* restaurants generated by the recommender system). This feature would allow for greater customizability for the user, but we will need to conduct more user studies to determine if this would be a desirable feature.

Figure 3

Map Page: Initial Sketch Vs. Finialized Wireframe



Results Page

The results page (Figure 4) will be displayed when the user selects one of the restaurants from the list below the map in Figure 3. This page will display the selected restaurant candidate in focus first, with the other recommended restaurants being displayed alongside the first in a carousel format. Each restaurant card will showcase essential details such as the restaurant's name, cuisine type, average rating, average price point, and operating hours, as well as one or more captivating photos highlighting the venue. Additionally, a user-friendly interactive feature labeled "Send to my phone" will be positioned at the bottom of the restaurant card. This feature enables users to conveniently transfer the restaurant's information directly to their personal device via text or email, ensuring they remember the establishment's name and facilitating easy sharing with other members of their party.

Figure 4.

Carousel/Results Page: Initial Sketch Vs. Finialized Wireframe

System Design

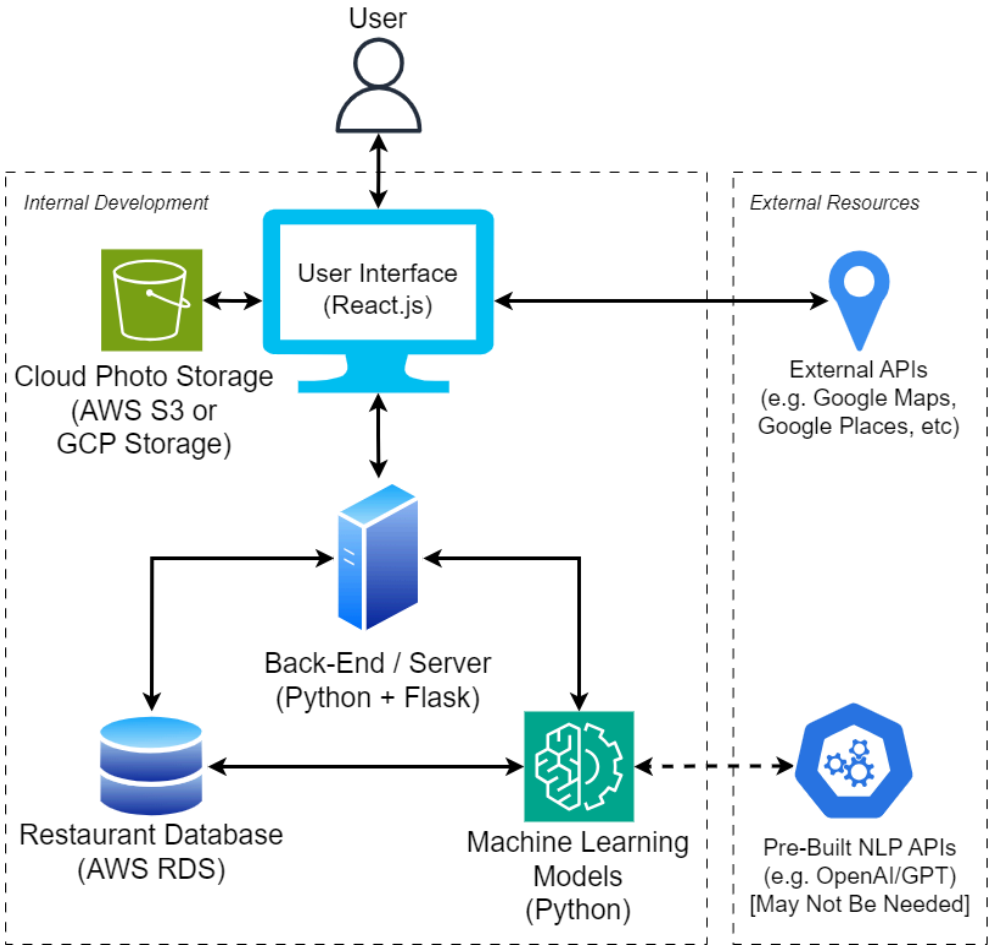
For our full-stack application (Figure 5), we have decided to utilize a versatile front-end built with React.js and a lightweight yet powerful back-end built with Flask. React.js will allow us to manage the complex interactions between the visual components of our application. Additionally, the prevalence of this framework in web development will allow us to leverage various component libraries such as MaterialUI in order to design a modern-looking application. The front-end of our application will communicate with external APIs such as the Google Maps and Google Places APIs, a cloud storage solution for restaurant photos, and the internal REST API we will develop with Flask.

Our back-end will be developed using Flask, a lightweight Python framework that will enable the front-facing components of our application to receive data important to the user's experience. This information includes restaurant data stored in the application's database (hosted through AWS RDS), chatbot responses based on the current state of user prompting, and restaurant recommendations generated through our recommender system. The natural language processing (NLP) solution as well as our recommender system will be developed in Python due

to the many available libraries and seamless integration with our Flask back-end. We are still evaluating whether our approach to the NLP solution should employ the usage of third-party, pre-trained models such as OpenAI's GPT.

Figure 5

Initial System Diagram of MyConcierge Application



Pilot Studies

In the initial wireframe design, we included a "heart" or favorites feature that enabled users to tag multiple restaurants as favorites. At the end of their browsing session, they could choose to email the selected restaurant cards to themselves, which we felt would be particularly useful for users operating on a kiosk system. However, following peer evaluation of these

designs, user feedback suggested that the favorites feature was redundant. Users expressed trust in the concierge's recommendations and were unlikely to send themselves a list of restaurants. Instead, they preferred to make selections directly from the curated list provided by the concierge, ensuring a more tailored and efficient decision-making process at the kiosk. By removing the unnecessary favorites feature, we also opted to discard the "Favorites" page in which users would select restaurants from their favorites to send to themselves. Instead, we streamlined the process by integrating the "Send to my phone" button directly into the restaurant card results page. Users expressed that this process was less convoluted and that they were more likely to take advantage of the sharing feature when in this form.

Inclusion, Diversity, Equity, and Accessibility Considerations

Given the fact that the problem we seek to solve falls in the food domain, there are many considerations in inclusivity beyond what is necessary from a technical/software standpoint that we must take into account. Diversity in food recommendations is incredibly important, especially in an age of inclusivity and culinary appreciation. We believe in ensuring a broad array of options for individuals with diverse dietary needs, whether by choice or due to religious beliefs. We aim to offer inclusive, accommodating, and delicious restaurant options, regardless of one's dietary restrictions. "Food is … one of the most widespread material expressions of social relationships in human society," so by ignoring a large portion of the population that has dietary restrictions we not only limit their ability to fully participate in social gatherings but also risk alienating them from the communal experience that food often provides (Cohen & Stefon, 2023). Others who choose to follow a vegan, vegetarian, or any other diet that excludes specific ingredients like gluten or dairy, do so for a variety of reasons. Some individuals choose to exclude or minimize their consumption of meat for various reasons, such as medical

considerations, reducing their environmental impact, or abstaining from what they perceive as contributing to harm to animals. (McCarthy, 2020). Regardless of the reason for these dietary decisions, we must make sure that our application is sufficiently accommodating.

From a software-related standpoint, our decision to pursue a kiosk interface comes with additional accessibility considerations. One consideration is the fact that interactions with the kiosk for each user will likely be short and infrequent, meaning that we must design our application to be sleek and intuitive (Lazar et al., 2021). Another consideration is that kiosks are usually positioned in public spaces that can be very crowded and noisy. As such, it is important to limit the amount of audio queues that are unaccompanied by visual cues (Lazar et al., 2021). Additionally, audio produced by the kiosk must not contain user information as this would project the private information to the public parties around the kiosk (Lazar et al., 2021). These concerns along with many others are outlined by Lazar et al. (2021), and we must keep all of these concerns in mind when developing our user interface. It is important to note that the control we have over some of the accessibility concerns is diminished due to the fact that we are not developing the hardware of the kiosk itself; however, proper design practices in our software can minimize the effects that potential hardware shortcomings could have on users. One example of this is the fact that touchscreen interfaces are fundamentally harder for older adults to use, regardless of their experience level with touch-screen devices (Elboim-Gabyzon et al., 2021). We must aim to minimize these difficulties by providing large buttons and texts in addition to minimizing the amount of user interaction needed to operate the application.

One final inclusivity feature we must consider is the potential for supporting multiple different languages. According to the U.S. Census Bureau (2022), 21.7% of people in the U.S. “speak a language other than English at home.” Given this high percentage, our application will

become much more inclusive to these individuals if we are able to support multiple languages in our interface in order to help these individuals feel more comfortable throughout their interaction with the application. Because our project employs NLP techniques, this form of support becomes much more complicated than other applications that do not require text analysis and generation. The added complexity means that, while this feature is quite desirable, it is likely a stretch goal that will not be completed by the delivery of the final product. As an alternative, we must seek to simplify the language presented throughout the application in order to support the 8.2% of "people who speak English less than very well" in the U.S. (U.S. Census Bureau, 2022).

Related Work

Novelty

The concept of a digital concierge itself is not novel; there are digital services that try to improve the experience of hotel guests through the use of tablets that can order room service and perform simple tasks (Hollander, 2023). Our project aims to fill what is currently lacking in the digital concierge space: recommending restaurants that fit a person's desires. Our project will create a chatbot experience that will try to mimic talking with a real concierge to find restaurants in a new area. This way, people can simply type out what they are currently feeling with regard to the type of food they like/dislike, how fast it will be, the price, etc., and receive tailored restaurant recommendations. Currently, people either have to google around to find a place to eat or set up some kind of account that requires multiple steps to start getting recommendations. Our service provides an experience that balances being as fast as opening Google with the ability to use preferences without the hassle of setting up an account. This will require us to build on top of pre-existing natural language processing techniques and recommendation algorithms to be able to understand what the user wants and find restaurants that fit those desires.

Domain References

Currently, digital concierge services are tied very closely to hotels. The hotels will place some type of tablet device in the room that the guests can interact with. Hollander describes the services as providing "personalized recommendations, language translation, and automatically pulling in saved information so guests don't need to enter their name or email address every time they book a restaurant reservation" (2023). The recommendations provided by these devices are mostly preselected local restaurants which do not do much to consider what the guests want. As Hollander states, the main goal of current devices is to "take some manual tasks off of [the] front desk agents' plates" (2023). This reduces wait time for customers when doing simple tasks such as ordering room service or creating a reservation. The tablet normally has a button to request a real person to talk to when they want to get more tailored recommendations and responses. Our project aims to fill this gap. We are aiming to create a service that provides the feeling of talking to an actual person to get personalized recommendations while still providing the ease of access that a digital concierge provides.

Direct References

If you look at the Play Store or App Store, you can find multiple applications that recommend restaurants based on your location and/or preference. However, these applications either have a high point of entry/engagement or do not consider every user preference without making their UI overly complicated. One such application is called Food Map. This application acts like an extension to your maps, and based on the user-provided location, it provides all the available restaurants near that location. Furthermore, Food Map also demonstrates useful information about each restaurant such as opening hours, ratings, and address. The problem with this application is that its functionality is limited because it only filters information based on one

parameter: location. It doesn't consider that the user might have cuisine preferences or dietary restrictions. Although this application partially succeeds in addressing our problem statement of providing nearby restaurants to the users, it fails to consider user preferences. Our application would expand upon the solution provided by Food Map by considering any cuisine, dietary, or other user preferences in addition to location. To limit the barrier of entry and not overwhelm the user, we will create a chatbot where the user can ask for any of their preferences, like a concierge. Based on their input, our AI model will extract all the important parameters from the prompt and filter the restaurant options based on that extraction. In this way, we will provide more accurate recommendations to the user that match their preferences. Furthermore, we will use the same approach as Food Map to display specific information about the restaurant. However, we would implement additional useful information, such as price range, parking availability, etc.

Peripheral References

Our idea of creating a system to recommend people restaurants based on their preferences is certainly feasible as it has been done by sites like Yelp and Google. The difference between our project and what these services provide is that you need to have a good understanding of what you want coming into them. You will also have to spend time looking at different options and tweaking parameters to find places that fit what you are looking for. Services with algorithms to create recommendations based on a set of parameters exist, but our project streamlines the interface to set these parameters intuitively.

In order to set these parameters through a natural conversation, our project employs natural language processing to parse what the user wants. We will need to extract enough features from their input before being able to recommend restaurants. This is also something that

is well researched and many different techniques exist to extract information from text. Specifically for our project, we plan to train a pre-trained language transformer model such as Google's BERT to classify sentences as talking about food preferences, location preferences, time preferences, price preferences, and any other categories we need. Once we know what each sentence is about, we can then dig in deeper to determine exactly what the user's request is specifying. For example, we can perform sentiment analysis on food preference-type sentences to understand whether the preference is something that they want or don't want.

Implementation Schedule

Project Development

Our team plans to have a total of three full sprints in the development period with each sprint having a focus on different tasks:

1) Back-end Setup, Front-end MVP
2) Back-end and Front-end Integration, NLP, and Recommender model standalone success
3) NLP and Recommender model integration, Front-end finalizations

This schedule would allow each of our subteams enough time to develop their portion of the work. Right now, the biggest bottleneck is the time taken for the NLP team to read up and gain insight on how to approach the natural language processing aspect of the application. Additionally, converting the Yelp dataset into a usable restaurant database has proved to be more difficult than initially envisioned. In contrast, both the back-end and front-end can work a little more independently and set up all the major components (ex. all the webpages, endpoints, storage, etc.). In sprint 1, the primary goal was to have all of these disjoint components hosted and connected together as a proof of concept. By the end of this sprint, our team will have an MVP that is primarily front-end to receive initial feedback on our design. In addition, all the

database and file storage have been set up, tested, and connected to the application. For sprint 1, the data and photo storage were filled with sample data so that the front-end and back-end could easily be connected in sprint 2.

Sprint 2 will focus more heavily on having functional NLP and recommender models. If there are issues with data cleaning, the back-end can aid in the cleanup and insertion of data into the database and photo storage. Front-end development in sprint 2 will be focused on improving the experience of the website based on user feedback as well as communicating necessary information with the back-end.

Finally, sprint 3 will focus more heavily on polishing the product into a complete state. This will mean getting the NLP model and recommendation system fine-tuned and having the website look like a finished product. This will also be the time to fully integrate the NLP and recommendations into the full-stack application.

Project Evaluation

Our team has decided to limit the scope of our project to certain cities as these were readily available in the Yelp dataset. During the evaluation period, we will intelligently test the application to determine the quality of recommended restaurants and their relevance to the user input given. Testing of the full system from user prompts to restaurant recommendations will be tested through user studies, while testing of the individual models that extract features and provide recommendations can utilize more objective metrics. It's important that the application makes sure to output varying results, and that all the presented results have some relevance to the user input. This plan will be changed over time as the NLP team learns and understands more about the potential issues they may encounter.

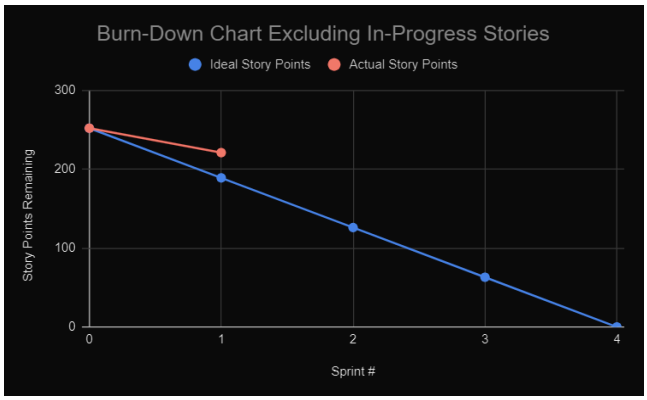
Project Finalization

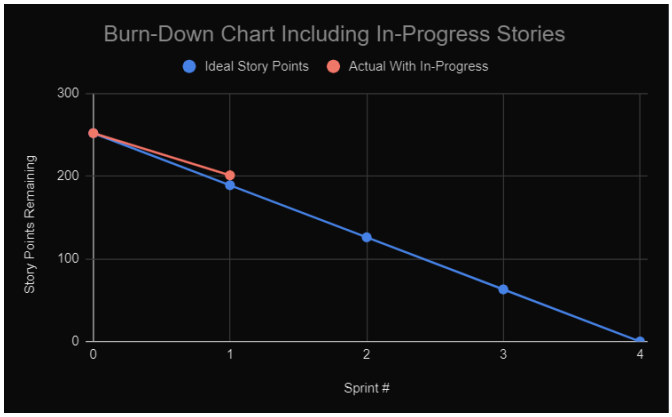
Our team will work both collaboratively and individually to make sure that all deliverables for this capstone project are properly written and submitted in a timely manner. We will ensure that good communication is practiced throughout and that will be essential to our success towards the final stage of this project.

Project Management Artifacts

Burndown Charts

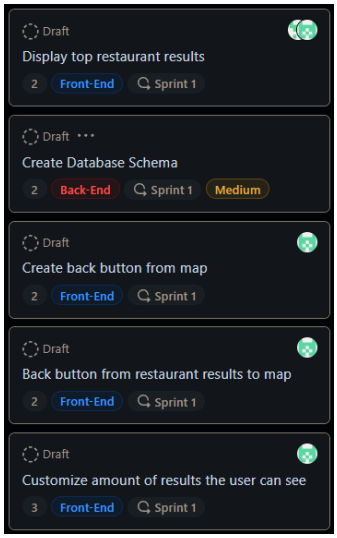
Sprint 1

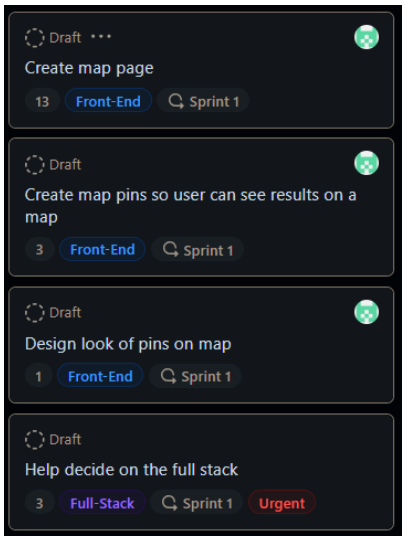


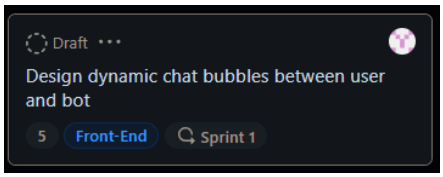


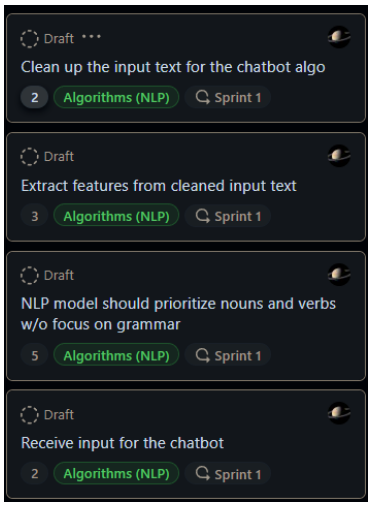
Sprint Backlogs

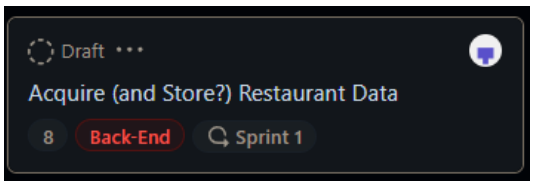
Sprint 1


Draft
Display top restaurant results
2 Front-End Sprint 1
Draft
Create Database Schema
2 Back-End Sprint 1 Medium
Draft
Create back button from map
2 Front-End Sprint 1
Draft
Back button from restaurant results to map
2 Front-End Sprint 1
Draft
Customize amount of results the user can see
3 Front-End Sprint 1

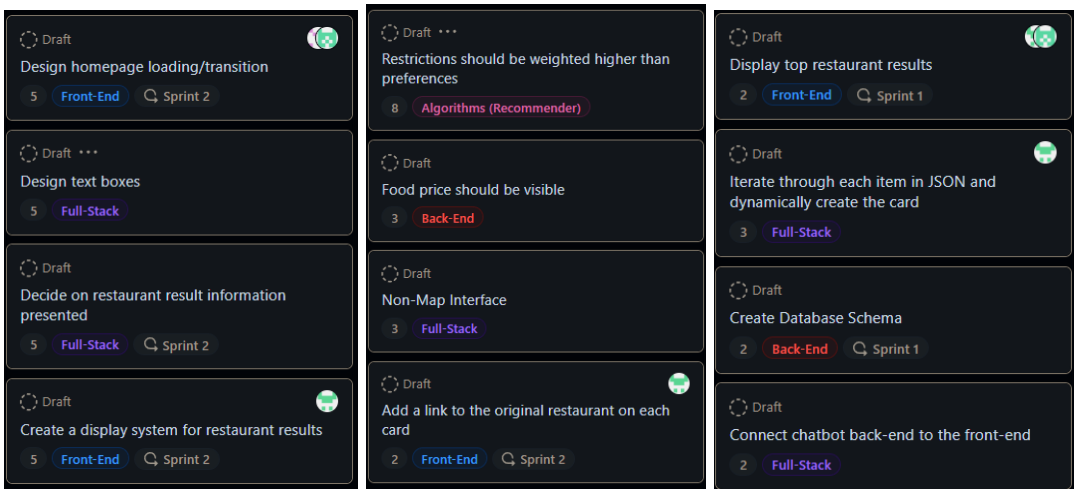

Draft
Create map page
13 Front-End Sprint 1
Draft
Create map pins so user can see results on a map
3 Front-End Sprint 1
Draft
Design look of pins on map
1 Front-End Sprint 1
Draft
Help decide on the full stack
3 Full-Stack Sprint 1 Urgent

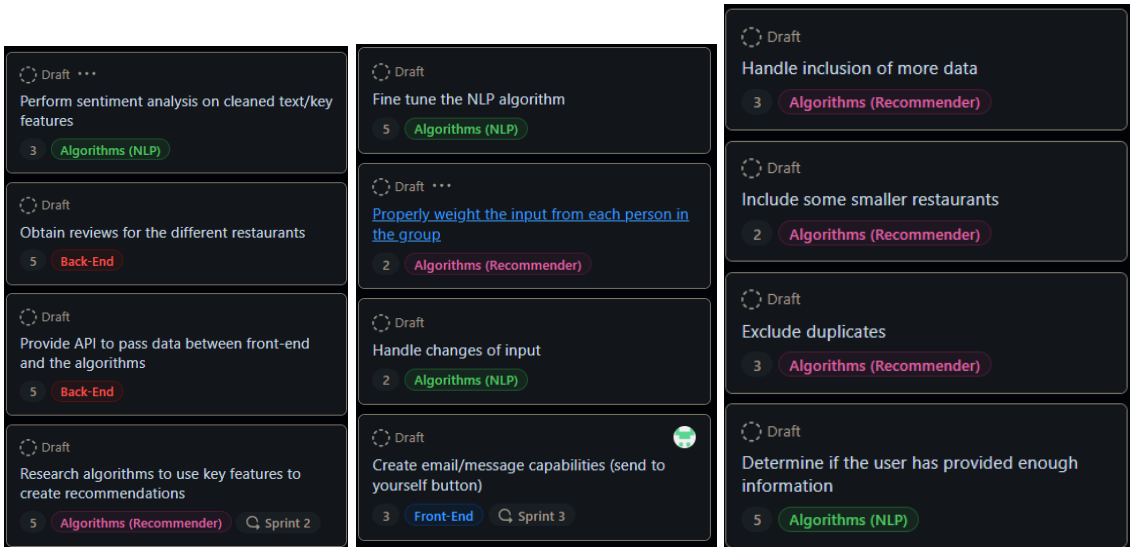

Draft
Design dynamic chat bubbles between user and bot
5 Front-End Sprint 1

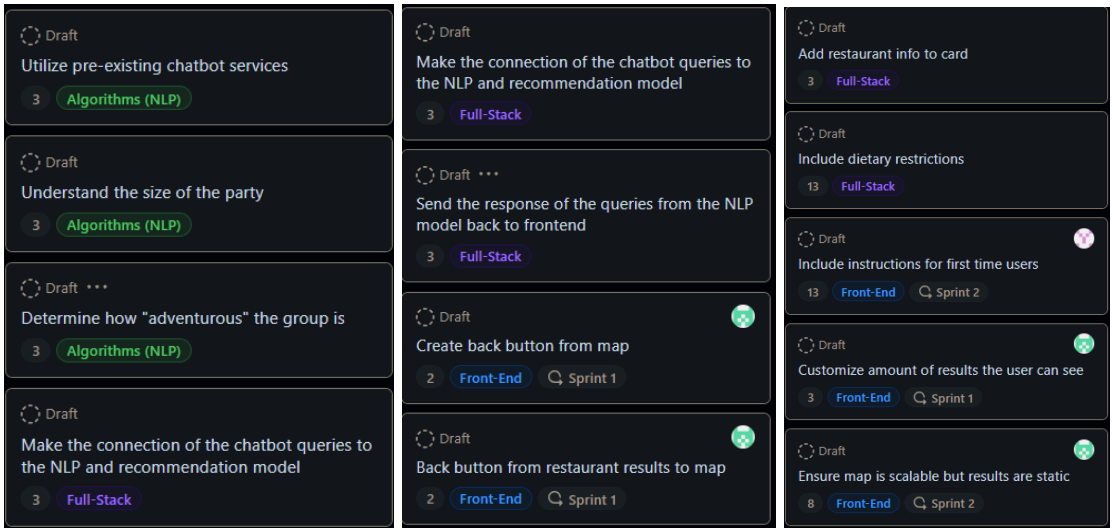

Draft
Clean up the input text for the chatbot algo
2 Algorithms (NLP) Sprint 1
Draft
Extract features from cleaned input text
3 Algorithms (NLP) Sprint 1
Draft
NLP model should prioritize nouns and verbs w/o focus on grammar
5 Algorithms (NLP) Sprint 1
Draft
Receive input for the chatbot
2 Algorithms (NLP) Sprint 1

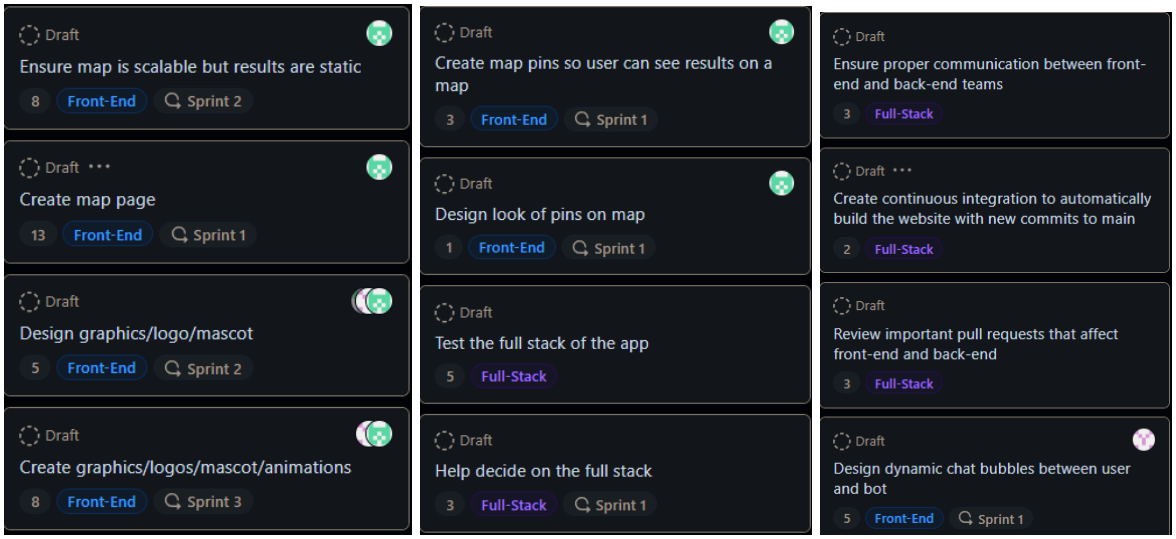

Draft
Acquire (and Store?) Restaurant Data
8 Back-End Sprint 1

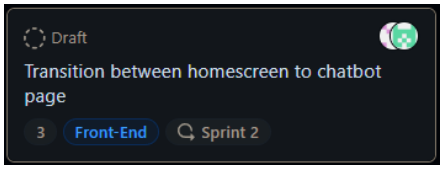
Product Backlog



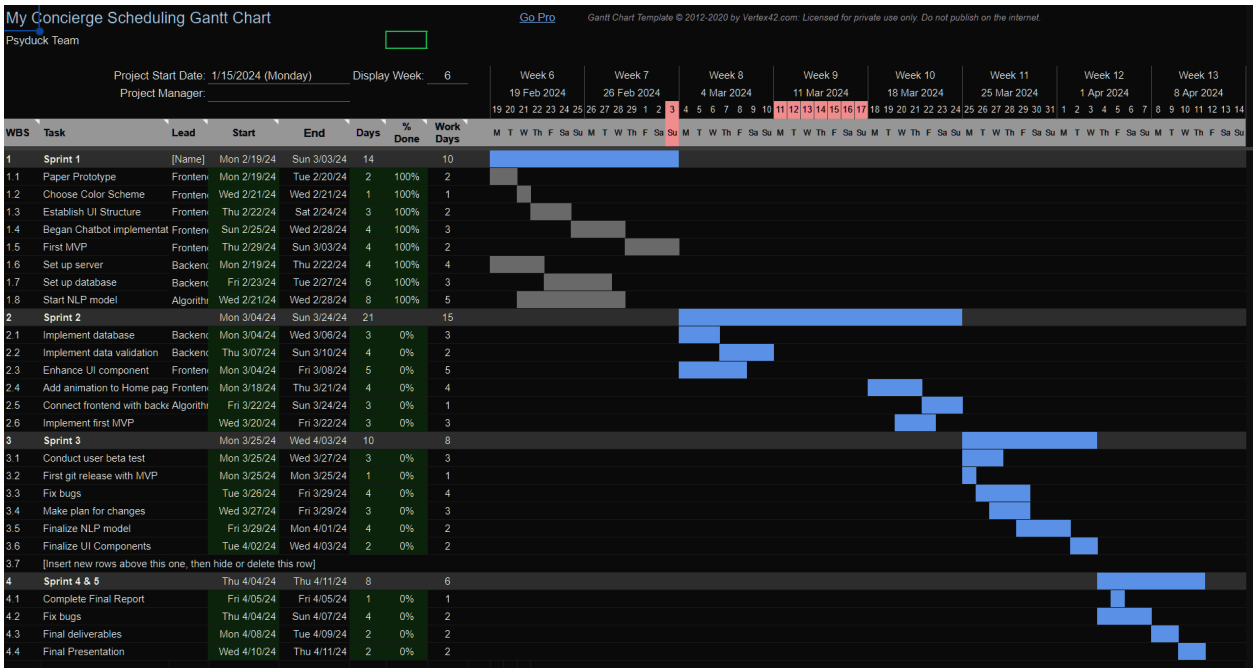



Draft
Utilize pre-existing chatbot services
3 Algorithms (NLP)
Draft
Understand the size of the party
3 Algorithms (NLP)
Draft
Determine how "adventurous" the group is
3 Algorithms (NLP)
Draft
Make the connection of the chatbot queries to the NLP and recommendation model
3 Full-Stack
Draft
Make the connection of the chatbot queries to the NLP and recommendation model
3 Full-Stack
Draft
Send the response of the queries from the NLP model back to frontend
3 Full-Stack
Draft
Create back button from map
2 Front-End Sprint 1
Draft
Back button from restaurant results to map
2 Front-End Sprint 1
Draft
Add restaurant info to card
3 Full-Stack
Draft
Include dietary restrictions
13 Full-Stack
Draft
Include instructions for first time users
13 Front-End Sprint 2
Draft
Customize amount of results the user can see
3 Front-End Sprint 1
Draft
Ensure map is scalable but results are static
8 Front-End Sprint 2


Draft
Ensure map is scalable but results are static
8 Front-End Sprint 2
Draft
Create map page
13 Front-End Sprint 1
Draft
Design graphics/logo/mascot
5 Front-End Sprint 2
Draft
Create graphics/logos/mascot/animations
8 Front-End Sprint 3
Draft
Create map pins so user can see results on a map
3 Front-End Sprint 1
Draft
Design look of pins on map
1 Front-End Sprint 1
Draft
Test the full stack of the app
5 Full-Stack
Draft
Help decide on the full stack
3 Full-Stack Sprint 1
Draft
Ensure proper communication between front-end and back-end teams
3 Full-Stack
Draft
Create continuous integration to automatically build the website with new commits to main
2 Full-Stack
Draft
Review important pull requests that affect front-end and back-end
3 Full-Stack
Draft
Design dynamic chat bubbles between user and bot
5 Front-End Sprint 1


Draft
Transition between homescreen to chatbot page
3 Front-End Sprint 2

Gantt Chart



System Design

Functional Design

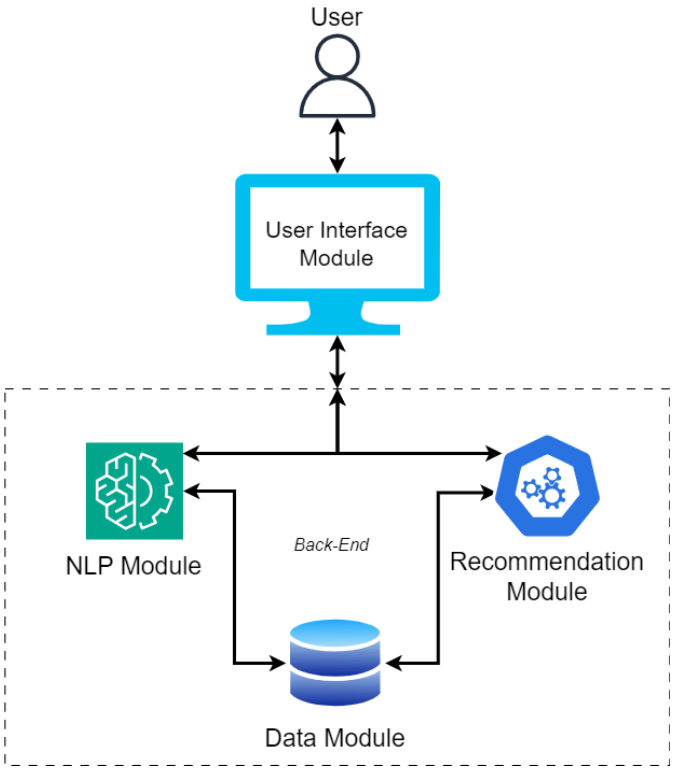
As a full-stack web application, the overall design of our system is quite complex. However, this complex design can be decomposed into several more manageable modules that together encompass the entirety of the application's functionalities. These modules consist of the User Interface Module, NLP Module, Recommendation Module, and Data Module.

Level-0 Diagram

The following diagram (Figure 6) provides a high-level overview of how the 4 modules of our application work together in order to create a comprehensive system and experience.

Figure 6

Level-0 Diagram of MyConcierge Application



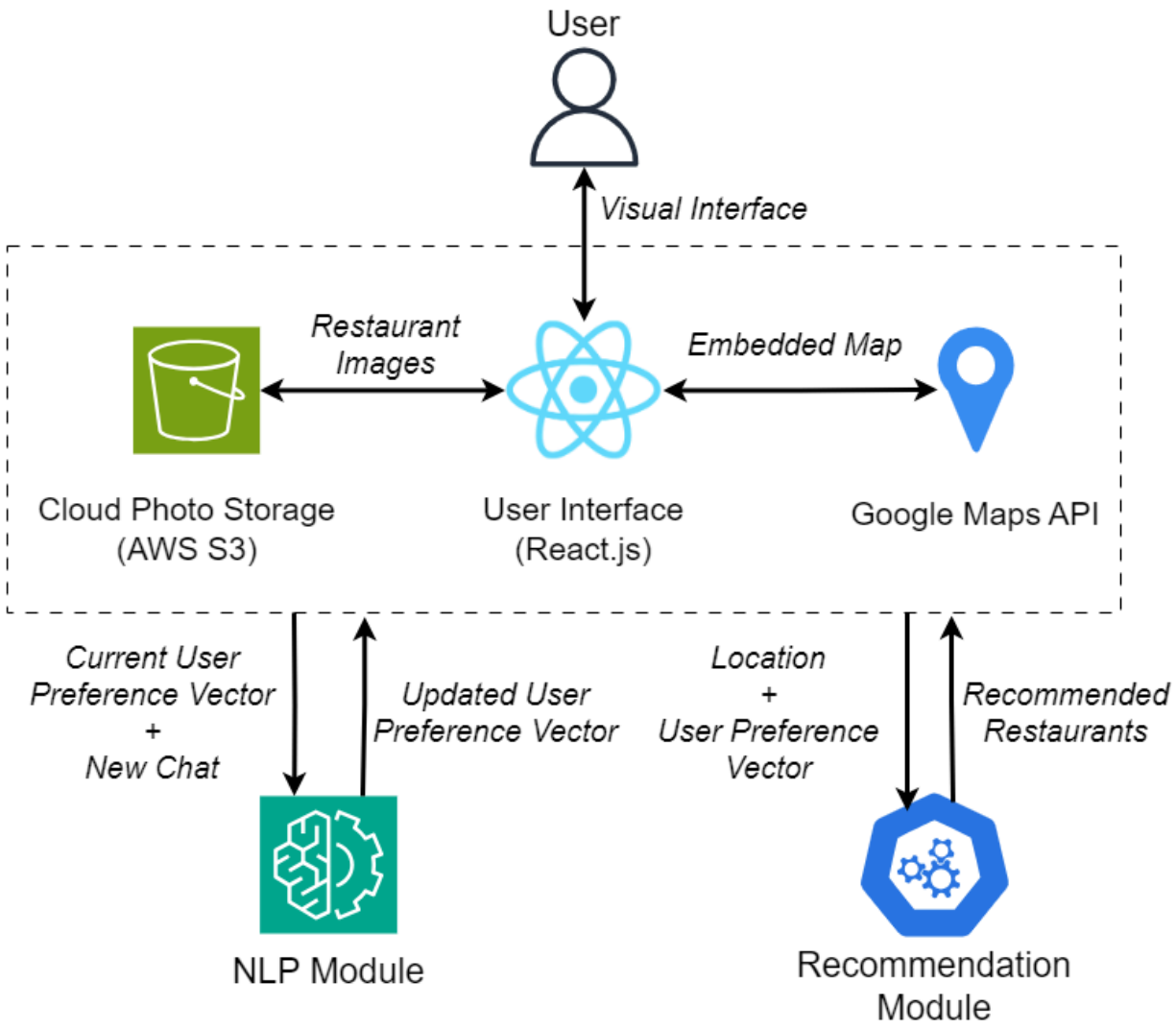
Level-1 Diagrams

The level-1 diagrams provide a more detailed view of each module of the application as well as how information is passed into, out of, and through these modules.

User Interface Module. The user interface (Figure 7) is the user-facing module of the application, capturing user input and visualizing the data of the application.

Figure 7

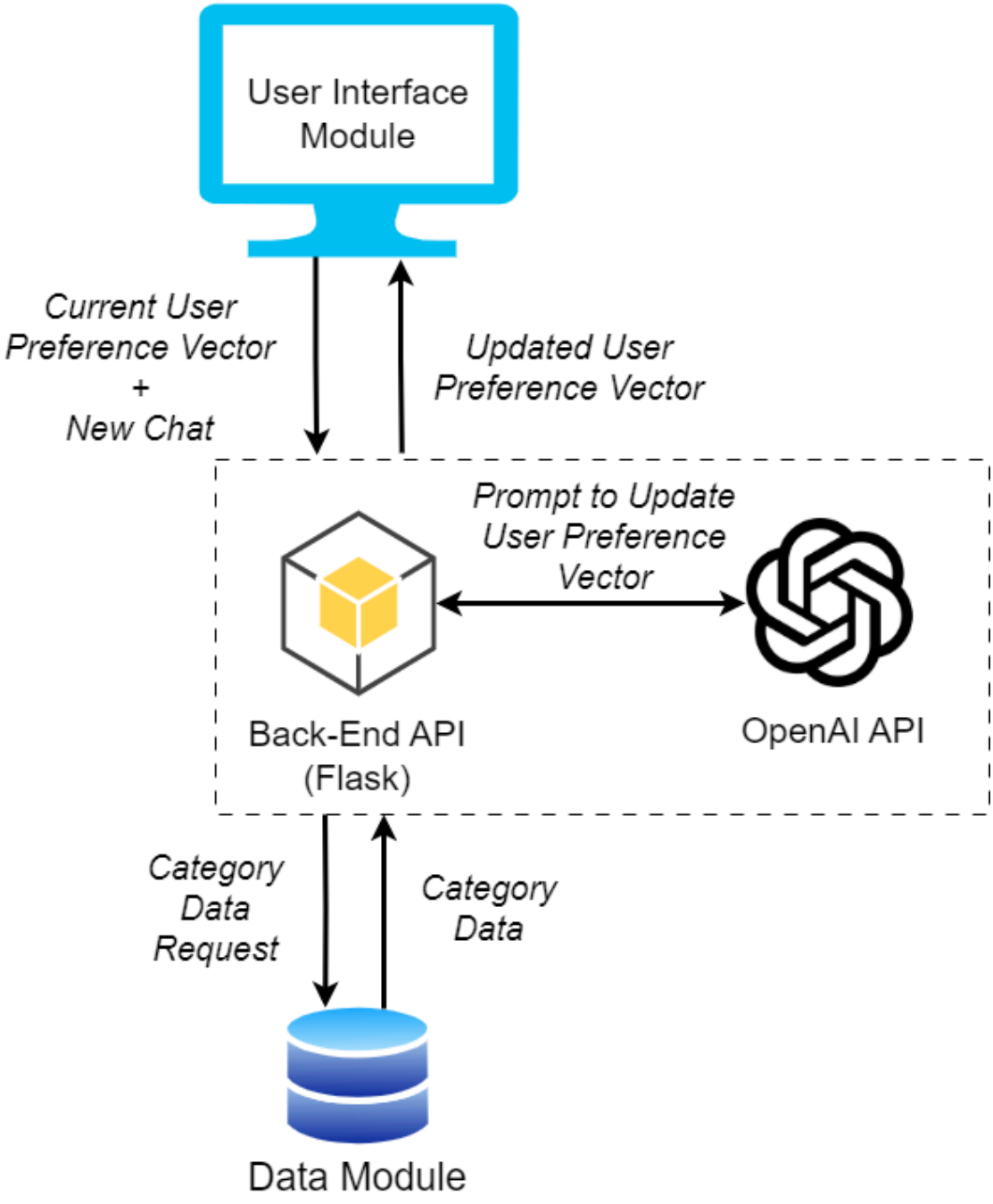
Level-1 Diagram of User Interface Module



NLP Module. The NLP module (Figure 8) is the portion of the application that is responsible for parsing user queries in order to extract key preferences.

Figure 8

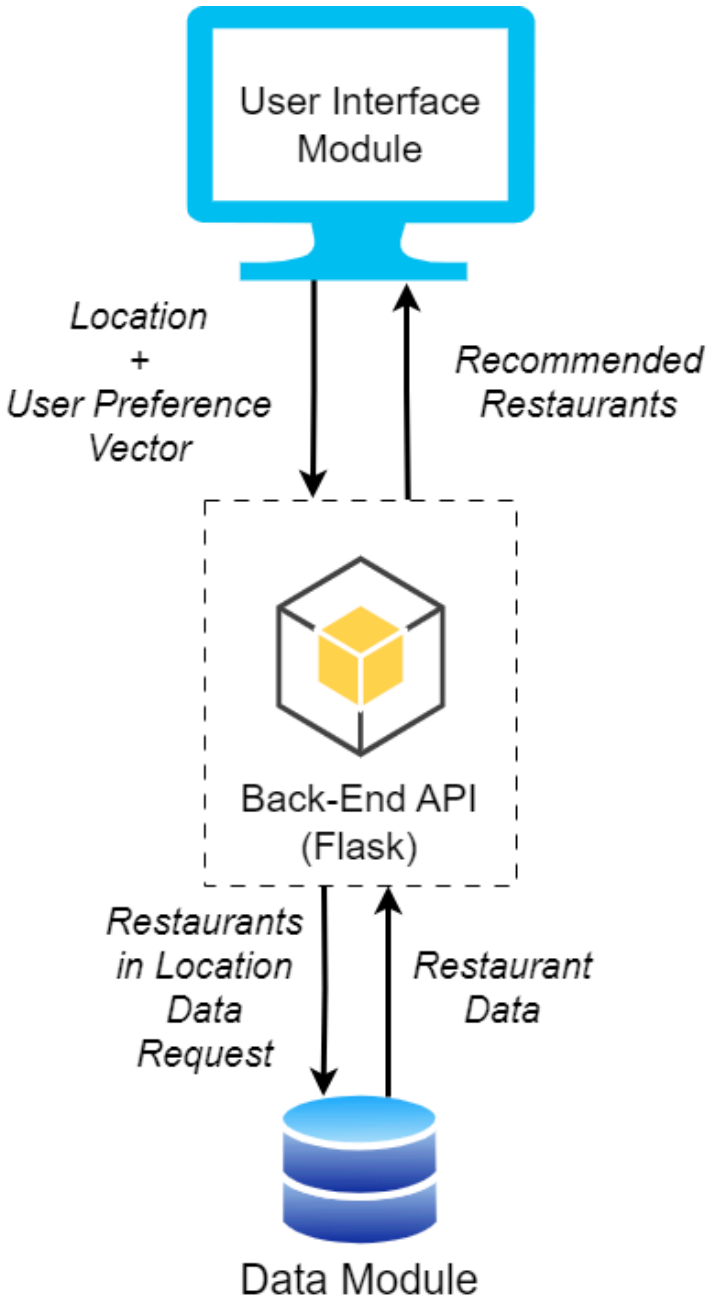
Level-1 Diagram of NLP Module



Recommendation Module. The recommendation module (Figure 9) is responsible for converting the user's preferences into a ranked list of restaurant recommendations.

Figure 9

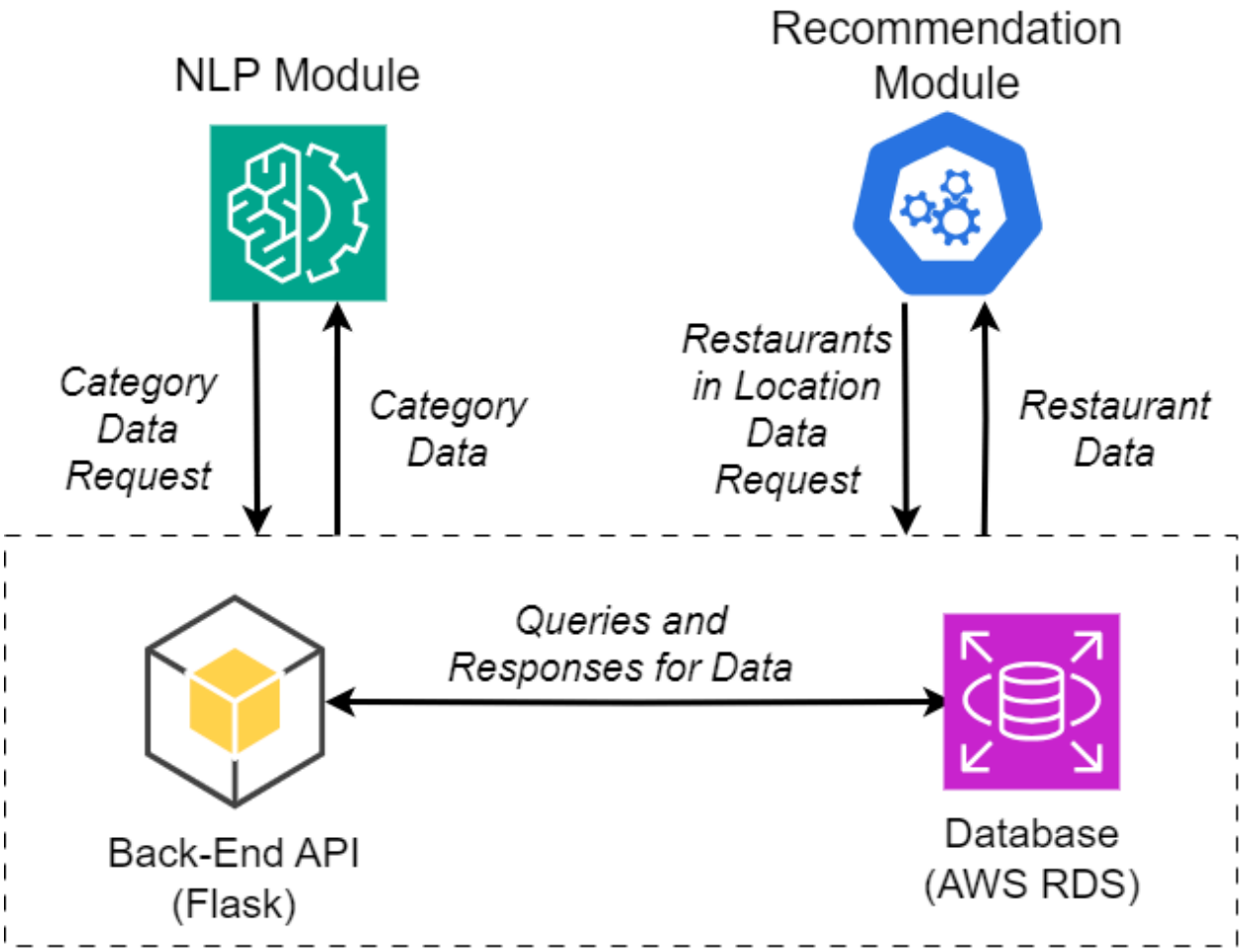
Level-1 Diagram of Recommendation Module



Data Module. Finally, the data module (Figure 10) is responsible for fetching the data from the database that the other modules require.

Figure 10

Level-1 Diagram of Data Module



Module Specifications

The following section will delve further into the details behind the modules and associated level-1 diagrams introduced above.

User Interface Module. The portion of the application that users interact with is the user interface module. This module will be largely implemented using React.js, and it will consist of 3 main pages that the user will be presented with. After a small landing page, users will interact with our chatbot. Each chat will be sent to the NLP module along with the current state of the user preference vector in order to retrieve an updated user preference vector and the appropriate

response for the chatbot to make. Once the NLP module has determined that the user has given enough information, the user interface module will retrieve the restaurant recommendations from the recommendation module and direct the user to the restaurant page in which they can view the recommendations. This page will consist of an embedded map with pins for each restaurant powered by the Google Maps API, as well as the restaurant information associated with each pin. There will also be a card carousel that has more in-depth information on the website as well as images from the Amazon S3 bucket.

NLP Module. The NLP model will be built upon Open AI's GPT 3.5 model. It will receive the input from the chatbot on the front-end to run the analysis. Once the input data is received, it will be cleaned and parsed for relevant information using a GPT model. This will be done by using a standardized template that GPT will have to format its response to match. After GPT has provided its analysis, the NLP model will create a vector that indicates how much each cuisine is wanted or not wanted along with any other features that GPT can find. Some of these features include the desired speed of the restaurant (sit-down vs fast food), cuisine, dietary restrictions, price, etc. The output vector from the NLP model will be stored on the front-end so that future calls to the model can build upon its previous work. The vector will then be used by the recommendation module to provide a match between what the user provided and what restaurants are in the area.

Recommendation Module. The recommendation module is responsible for converting user preferences into ranked restaurant recommendations. The module will take in the user preference vector and location preference from the user interface module as well as the restaurant data for the preferred location from the data module. The user preference vector is compared to each restaurant in the area (potentially after being scaled by some correlation/similarity matrix)

in order to provide a score for each restaurant. The restaurants will then be sorted based on this score in order to retrieve the top 15 recommendations. Once complete, the ranked restaurants will be passed to the user interface module so that they may be appropriately displayed to the user.

Data Module. The final major module of the application is the data module. This module is responsible for managing access to the application's database. The data module will largely be interacting with the NLP and recommendation modules, supplying them with category, attribute, and restaurant information as needed. This data will be transmitted through API endpoints of a Flask back-end, and it will originate from a MySQL database hosted on AWS RDS.

API Specifications

In this section, we will outline the specifications and functionalities of various APIs that play a pivotal role in enhancing the capabilities and user experience of our application. Below, we delve into the details of the OpenAI API and the Google Maps API, highlighting their respective contributions to our application ecosystem.

OpenAI API. The OpenAI API is a cloud-based service provided by OpenAI that allows developers to access and integrate OpenAI's NLP capabilities into their applications. It provides access to cutting-edge language models like Generative Pre-trained Transformer (GPT) and other AI models developed by OpenAI. Early in development, it became clear that any NLP model we could implement in a reasonable timeframe would not be able to compete with GPT 3.5, one of OpenAI's most advanced GPT models. As a result, we have decided to use OpenAI's API (specifically for GPT 3.5) in order to analyze user chats and populate the user preference vector. This saves not only time and resources but also provides a more accurate and organic interface that users can interact with during their session.

Google Maps API. The Google Maps API allows us to embed Google Maps functionality into our application. It provides various features such as displaying maps, adding markers, customizing map styles, and implementing directions and routes. By utilizing this API, we aim to create a more aesthetically pleasing, interactive, and inherently intuitive geolocation experience, enhancing user engagement and usability for both groups and individuals alike.

Data Design

This entire application relies on data. It has to know about the restaurants nearby, their attributes, and whether the client would enjoy eating there. To satisfy this, we've gone ahead and stored records of the restaurant data in several cities around the country and their related attributes.

Flow of Data

The user will have the ability to input data to the chatbot. This text will then be parsed through with all the relevant keywords extracted from it using the ChatGPT model and returned to the user. This will be passed to the front-end which will query the data stored in the database through an endpoint checking which restaurants are most similar to the data provided. Finally, these data points will be sent back to the front-end to be displayed in an appealing manner to the user.

Relational Database

The application will require a database to pre-cache restaurants in specific target cities alongside their attributes. The data that will be stored comes from the usage of the Yelp dataset and API and will facilitate the selection of restaurants based on the user's preferences. The team decided to host the database on AWS to allow for better integration with our services and so that all team members would be able to access and modify the data if needed without the limitations

of it being too complex to be accessed. The database contains 3 tables: Restaurants, Categories, and Attributes.

The Restaurants table will consist of the attributes listed in the table below. Some more confusing attributes include Metro_Area, Category_Sum, and Attribute_Sum. The Metro_Area of a restaurant refers to the supported metropolitan area that this restaurant falls under, whereas the City of a restaurant is the specific suburb or neighborhood that this restaurant falls under. The Category_Sum and Attribute_Sum of a restaurant are bitmask-able sums of category and attribute IDs, respectively, that are used to efficiently store a restaurant's categories and attributes as well as quickly convert this to restaurant category vectors for the recommendation algorithm.

Restaurants		
ID	Restaurant_Name	Address
City	State	Zip_Code
Metro_Area	Latitude	Longitude
Stars	Num_Reviews	Category_Sum
Attribute_Sum	Hours	Base_Image_URL
Num_Images	Yelp_URL	Phone

The Categories and Attributes tables are small tables used to store the possible categories and attributes for restaurants in the database. Categories are things that our recommendation system uses to generate recommendations such as cuisines, dietary restrictions, and price ranges. Attributes are characteristics of a restaurant that may be important to the user but are not used for generating recommendations such as if the restaurant is wheelchair accessible.

Categories	Category_ID	Category_Name

Attributes	Attribute_ID	Attribute_Name

Restaurant Image Storage

Images that are from the Yelp dataset and its associated restaurants will be stored in an AWS S3 bucket. This provides easy access for the front-end and helps ensure that our images will be quickly accessible regardless of the user's location. On the other hand, images from the Yelp API will be accessed through URLs provided by Yelp directly.

Budget Costs

For our project, we have two main categories of resources: hosting and API. All of our needs are software related as our project does not require any hardware.

Internal Course Resource Needs

For our hosting needs, we are using a relational database to store information on all of the restaurants. This stores the features that we extracted from the Yelp database. We are using AWS to host our database and we are also using Amazon S3 buckets for data storage. We want to have images to go along with each of the restaurants which will require us to store many images for quick retrieval. We are hosting our website on Vercel.

As for the other half, we are using some different APIs to make our website. For the front-end, we are using Google Maps to provide an interactive map that lets the user view the results. We are also looking into the Google Places API to be able to get more detailed information on specific locations. For the back-end, we are using Open AI's GPT API to use their latest transformer model: GPT 3.5. This will provide all of the natural language processing for the back-end to gain an understanding of what the user wants.

Overall Cost of Project to an Outside Company

For the maintenance costs, our hosting platform is free. The database costs 5 dollars per month. The database and Amazon S3 storage would be a max of 23 dollars per month which is likely an overestimate since we would have to store 1 TB of images to hit that number. Most likely the Google Maps API will not cost anything because we will not require the resources. The Open AI API will be one of our bigger costs. This will most likely be over 20 dollars per month depending on the size of our user base.

Item	Description	Quantity	Unit Price	Total Price	Purpose
Vercel	Hosting Platform	1	free	free	Hosts our website
Database	Stores restaurant data	1	$5 per month	$15 for 3 months	Allows quick access to restaurant info
S3 buckets	Stores images and any other data	1	$23 per month max	$69 for 3 months	Will provide quick access to images
GPT 3.5	Perform NLP	1	> $20 per month	> $60 for 3 months	Will break down the user input to provide us with what they want.

Personnel Costs

The salary cost of the development team:

- Project Manager: $50/hour
- Front-end Developer: $40/hour
- Back-end Developer: $45/hour
- NLP Engineer: $55/hour
- Recommender System Engineer: $55/hour
- Quality Assurance Engineer: $35/hour

The weekly development hours and cost (considering full-time employees):

Team Member	**Weekly Hours**	**Hourly Rate**	**Weekly Cost**
Project Manager	40	$50	$2000
Front-end Developer	40	$40	$1600
Back-end Developer	40	$45	$1800
NLP Engineer	40	$55	$2200
Recommender System Eng.	40	$55	$2200
Quality Assurance Eng.	40	$35	$1400
Total per week	**240**	**$280**	**$11,200**

Total salary estimated costs based on the total person-hours:

Task	Time Estimates (hours)	Total Development Cost
Project planning and requirements gathering	90	$4500
Front-end development	120	$4800
Back-end development	60	$2700
Chatbot training and integration	100	$5000
NLP model development and integration	90	$4950
Recommender system development and integration	120	$6600
Testing and debugging	100	$3500
Deployment and optimization	60	$3000
Total Development Phase	**740**	**$35,050**

The salary cost of the maintenance team:

- Technical Support Engineer: $35/hour
- System Administrator: $40/hour

Total salary estimated costs based on the total person-hours:

Task	Total Cost
Bug fixes and updates	$320 (8 hours x $40/hr)
Server maintenance	$160 (4 hours x $40/hr)
Customer support	$280 (8 hours x $35/hr)
Total Maintenance Per Week	**$760/week**

Evaluation Procedures

Our team's capstone project is an AI-powered food recommendation system that can take custom user prompts to produce a curated list of nearby eateries that the client is predicted to like based on their predispositions. In order to facilitate an application that is capable of this functionality, our team will be incorporating internal testing to ensure both that the technical aspect of the project is working as well as ensuring that the natural language processing in conjunction with the recommendation algorithm is all working properly to display the best results it can. In addition, to make sure that the application is well received by potential users, our team will go on to interview twenty participants to assess and agglomerate user feedback to result in a cohesive product that is functional and user-friendly.

Functionality Evaluation

For our functionality testing, we are using a combination of two techniques: unit testing and integration testing. We use unit testing to test individual components of the program in isolation to ensure that each component is working properly. This allows us to feel confident that each component works well before trying to integrate them. As for integration testing, this has allowed us to ensure that each part of the project is working together once we start putting them all together. We used integration testing to validate our full-stack application from start to finish.

Unit Testing

Our unit testing comprises both manual and automated processes. We first start by manually testing the code as one does as they program. We use this phase as a preliminary process to catch bugs and to obtain a working piece of code. This is more than simply writing code and moving on to the next thing to be written; we take more time to think deeply as to how the code works and start to find the edge cases. As these come up, we fix/prevent errors from

occurring by making changes to the code. Once this step is complete, we move to use the unit test framework in Python to test our back-end and Jest for our React front-end. These two frameworks allow us to write automated unit tests for each function so that we can better catch bugs and/or prevent bugs in the future. One of unit testing's biggest advantages is that it makes sure that future changes to the code have to consider the previous use cases already designed. This means that if a future change were to break some previous functionality, it would cause a unit test to fail. The unit test(s) would have to be fixed before the code could be merged. Due to the small scope of our project, we require that all unit tests be passed in order to merge code. We aim to have at least 80% code coverage with the remaining 20% being small chunks of code that do not require testing.

Integration Testing

As for our integration testing, we do a manual mocking of data to test multiple parts of the code working together. We do this by creating an integration testing function that generates some mock data and then calls a function that does multiple things, and then validates the output after the function returns. This allows us to easily test whether multiple layers of the project are working properly together and returning the correct output. There are some difficulties with integration testing with one being our limited mocking capabilities; for example, we are unable to have a specific function always return a value so that we can force certain outcomes. We are only able to control what we input to the function, and we are not able to control what any of the subsequent functions will return. This limits our ability to test all possible outcomes. We are also unable to deeply test our GPT functionality since it is not deterministic; it is hard to write tests to validate the output of a function where the output is changing even with the same input. We construct our integration tests in a bottom-up approach where we test at the lowest level first and

move up. This allows us to ensure that all code below a particular function has already been tested and passed, meaning that only the current function needs to be investigated. We also only perform integration tests after the relevant sections have undergone unit testing.

System/Validation Testing

For our system/validation testing, we focus on evaluating the accuracy and effectiveness of our AI-powered food recommendation system. We begin by validating the dataset used by our recommendation system. This involves checking the accuracy, completeness, and relevance of the data to ensure that our system operates on reliable information. We will test the algorithms used in our recommendation system to assess their accuracy in predicting user preferences and we will collect feedback from users who interact with our concierge in terms of accuracy and relevance of the results received. The results will be evaluated by comparing the recommendations provided by our system with the user's actual preferences and choices. This comparison will help us better understand the system's ability to understand user preferences and tailor recommendations accordingly. The comparisons of the results will be conducted by A/B testing which would allow us to identify which algorithms or approaches yield the most accurate and satisfactory recommendations.

Functionality Evaluation Results and Discussion

Unit Testing.

Details. Unit testing was performed throughout the development of our application. Due to our application being quite small, we relied on manual testing. We did this by thoroughly testing each component as we developed. We made sure that each developer tested their code until they felt that it was fully tested. We used git to blame people as bugs occurred as a method

of determining how well each developer was testing their code. This system worked quite well for us, and we feel satisfied with our level of testing.

For the front-end, we made sure that both happy and not-happy paths were tested. For example, we tested what happens when you go straight to the map page without entering anything into the chatbot. We also tested what happens when the chatbot is given garbage information, and what happens when the cards are missing information (ratings, hours of operation, images, etc).

For the back-end, each API function was tested on both proper data and improper data. We made sure to catch all exceptions and handle them accordingly. This made our back-end very robust to failure; in the worst case, the front-end might have to resend the request, instead of the entire website crashing. We also faked database errors to ensure that our code that interacted with the database handled errors if they occurred. The GPT API and recommendation systems were also heavily tested and used both real and fake input to ensure correctness.

Results. For the front-end, we found some edge cases that we were not handling properly like trying to go straight to the map page. It used to provide an empty map with no results; we fixed this by making it give random recommendations so that at least the user can see some results even without providing any information. Whenever something is missing from a card, we fill it with an appropriate piece of text informing the user that the data is unavailable. When images cannot be found we display a default image with the robot. Speaking of images, we found through our testing that many restaurants did not have images. We found that images were linked to specific locations, so we greatly improved this by merging the images of restaurants under the same chain.

For the back-end, we found many errors that were not being properly handled. With the GPT API, we were not handling the case that it returned “I don’t know”. We were always expecting GPT to give us some kind of response, however this was not the case. We also found lots of little errors that could happen at random like invalid array access; we had some loops that expected the previous code to have properly populated an array, but sometimes this did not happen. We found many bugs like that and handled them properly.

Discussion. For the size and scope of our project, we felt that our unit testing worked very well. Since there weren’t too many components and each component was developed by one person, manual testing was a great option. We didn’t have to worry about the overhead of setting up some kind of automated testing. It felt that automated testing would be overkill for our application. That being said, if this project were to be expanded and more people be brought onto the team then we think that automated testing would be necessary.

Integration Testing.

Details Integration testing was used to test how multiple components would work together. We performed this kind of testing when a new feature was added that had to interact with other components. The idea is that the preexisting components have already been unit-tested and tested on their integration so that they can be believed to be correct; this meant that only the new component needed to be tested and validated. We also did this process manually as we felt that it was the best choice for our project.

For the front-end, the biggest part was making sure the pages connected together smoothly. We tested this by going between the pages by using the bell button, the back button on the browser, and by changing the URL. We made sure that each of the cases worked as expected and that the opposite direction worked as well. We also had to test the cards/map pins to ensure

that they worked together well. We did this by clicking on different cards/pins and making sure that the matching card/pin was properly selected and displayed.

For the back-end, we did integration testing to ensure that the NLP model could properly communicate with the API endpoint that gets called from the front-end. We also made sure to test the communication between the recommendation system and its corresponding API endpoint. Lastly, we tested the endpoint responsible for retrieving data from the database. We performed these tests by using Postman to make both valid and invalid requests and verify the output.

***Results*.** For the front-end, we discovered that there were some errors when going between the pages like the chat history being lost and/or acting weird. We got this fixed so that the chat was remembered and added a reset button so that the user could start a fresh chat if desired. We also found that the carousel acted strange when selecting pins on the map and when selecting cards: it would always move the selected pin/card to become the leftmost item in the carousel. This felt very clunky and confusing. We solved this problem by changing how we detected when the user clicked on the cards/pins; we made it so that the three visible cards do not move if possible and instead the selected card/pin is highlighted.

For the back-end, we actually found that the different parts integrated together very well. Since the unit testing was done well, it was straightforward to get everything connected together. We did find a few edge cases where some unexpected return values were not being properly handled. Those were easy to fix by simply catching the invalid return values and handling them accordingly.

***Discussion*.** Again, we felt that manual integration testing worked great for the scope of our project. We felt that it would have taken more time and effort to get some form of automated

integration testing to work than it would to just test it ourselves. We are very pleased with the reliability of our application and how the different components connect to each other. As with the unit testing, it would be great to automate this process if the project were to continue and the team expanded.

System/Validation Testing.

***Details*.** Effectiveness and accuracy were two main areas of emphasis when performing our strenuous testing. We saved the system testing/validation for the end. After we had performed all the unit and integration tests, we put the entire system through tests to verify how well it worked. This helped us find the last set of bugs to be fixed. As with the other methods of testing, we did this manually. One of our most effective methods of testing the entire system was letting other people use it. By letting others use our application, we found that people sometimes used the application completely differently than we expected which led to some bugs. Other than bugs, this also helped us improve the overall interface to make it easier and more intuitive to use.

***Results*.** By conducting thorough system and validation testing, we were able to ensure that our virtual concierge website meets the needs and expectations of users, providing them with reliable and enjoyable restaurant recommendations. We were able to verify that a lot of different use cases work by testing them ourselves; we also got to test more use cases by having others use the application and push its limits. Through this, we found some bugs and undesirable behavior. It also showed us what our system was capable of and what it could not understand.

***Discussion*.** Overall, we feel that our system testing went well and we feel confident in our product. We found that watching other people use our application greatly helped us improve our interface and make the application more intuitive. Through this process, we found the limits of our system; it is unable to accurately understand every single query that is thrown at it because

it does not have categories for everything. This meant the user experience could either be really good, or quite poor. Although the website itself works very well and everything is very robust.

Usability Evaluation

The Usability Evaluation aimed to assess the effectiveness, efficiency, and user satisfaction with our AI-powered food recommendation system. The evaluation process encompassed a combination of observations, interviews, focus groups, and surveys, involving 20 participants.

Usability Evaluation Goals:

- Evaluate the system's ability to provide relevant and accurate food recommendations
- Measure the speed and ease of use in navigation and interaction with the application.
- Assess user satisfaction levels with the system's features, design, and overall performance.
- Ensure the application is accessible and intuitive for users with varying technological backgrounds.
- Incorporate user feedback to improve the system's design, functionality, and user experience.

User Study Target Group

Our target group for the user study comprises individuals who frequently dine out and are interested in exploring new restaurants. Additionally, we want to make sure our user study group has a wide age gap and we consider people from different age ranges including older people who are not well accustomed to dealing with technology. This selection adequately covers potential user groups as it targets a demographic likely to benefit from and provides valuable feedback on our food recommendation system. We also liked this group selection as it would give us an

insight into how accessible our web app is and some of the features that would better help our users get the results and recommendations they enjoy.

User Study Recruitment

Two main ways that we are going to reach out and contact the participants are through email and social media posts. Both of these ways would be used to send the user study request to our target group mentioned above.

This would be the content of the email we send:

Subject: Invitation to Participate in Our AI-Powered Food Recommendation System User Study

Dear [Recipient's Name],

We are reaching out to invite you to participate in a user study for our Capstone project, an AI-powered food recommendation system. Your feedback will play a crucial role in helping us refine our application and make it more user-friendly. This study aims to understand the user-friendliness and functionality of our program by gathering feedback from users like yourself. The eligibility criteria are that the participants must be individuals who frequently dine out and are interested in trying new eateries. You would need to interact with our application and provide your honest feedback about its features and usability. Furthermore, we might ask you to answer a few questions about your experience. This study could take around 20-30 minutes.

If you are interested in participating, please reply to this email with your availability for a user study session.

The social media post would contain the same content mentioned in the email above. However, it would be presented as a Canva digital flier or a message/post. The post will have less information presented and a link present so any interested parties can look for more information. This request would mainly be posted on Instagram, Facebook groups that conduct free user studies, GroupMe discussions, and student channels on Discord. However, this list is subject to change based on our final product.

For both methods, we will ensure that the instructions on how to participate are clear and that we are going to follow up with our interested participants to schedule a user study session.

User Study Protocol

There will be a very specific user study protocol to ensure that the gathered user feedback is useful and relevant to our project and that our team can gather the most effective information and data to decide what to change (or not change) in our capstone project. One of the most important features for us is to complete the study in an in-person setting so that we can better gauge peer reactions from our participants. In an application like the one that we are building, our team needs to make sure that the user experience is as seamless and as intuitive as possible. Even small delays or inconveniences should be noted down to be fixed. The pre-study interview will be fairly informal. We want to assess how participants currently interact with the technology around them to decide where to eat in unfamiliar environments. For example, when placed in a new city on vacation, it is meant to help determine where to go to get food. Our team wants to survey common existing methods to assess how they are built upon and potential weaknesses expressed by our competitors. We would like a two-pronged approach with one being relatively freeform and the other being a little more guided. Firstly, we will ask the users to type in any prompts into the application and rate how well they receive the responses. For example, do they

think the responses are a good or bad response to what they were initially thinking about? Continuing with this line of questioning, we will then ask the user to ask specific questions and rate how well they received the responses. They will be asked to think out loud during the process and give us their thoughts during the process. There will be a small post-interview asking if there are any changes that they would like to see in the project as we continue. Each interview will take roughly 5 minutes.

Planned Questions

Pre-Study Questions.

1. How often do you dine out per week?
   A. Never
   B. Occasionally (1-2 times)
   C. Frequently (3-5 times)
   D. Very frequently (more than 5 times)

2. What factors influence your decision when choosing a restaurant? (Select all that apply)
   A. Cuisine type
   B. Price range
   C. Location
   D. Ambiance/atmosphere
   E. Dietary restrictions/Preferences
   F. Recommendations from friends/family
   G. Online reviews
   H. Other (please specify)

3. How likely are you to try a new restaurant based solely on an online recommendation?
   A. Very unlikely
   B. Unlikely
   C. Neutral
   D. Likely
   E. Very likely

4. Have you used a virtual concierge or restaurant recommendation service before? If yes, please specify which one(s) and your experience.
   A. Yes
   B. No

5. What features or functionalities would you expect from a virtual concierge recommending restaurants?

Post-Study Questions.

1. Did the virtual concierge accurately understand and respond to your queries regarding restaurant recommendations?
   A. Strongly Disagree
   B. Disagree
   C. Neutral
   D. Agree
   E. Strongly Agree

2. How satisfied are you with the restaurant recommendations provided by the virtual concierge?
   A. Very Dissatisfied
   B. Dissatisfied
   C. Neutral
   D. Satisfied
   E. Very Satisfied

3. Were the recommended restaurants suitable for your preferences and requirements?
   A. Not at all
   B. Somewhat
   C. Moderately
   D. Mostly
   E. Completely

4. Did you find the user interface of the virtual concierge easy to navigate and interact with?
   A. Very Difficult
   B. Difficult
   C. Neutral
   D. Easy
   E. Very Easy

5. Did you encounter any difficulties or challenges while using the virtual concierge? If yes, please specify.

6. How likely are you to use this virtual concierge again in the future?
   A. Very Unlikely
   B. Unlikely
   C. Neutral
   D. Likely
   E. Very Likely

7. What suggestions do you have for improving the virtual concierge's performance and usability?

8. Would you recommend this virtual concierge to others looking for restaurant recommendations?
   A. Definitely not
   B. Probably not
   C. Neutral
   D. Probably yes
   E. Definitely yes

9. Did the virtual concierge meet your expectations in terms of recommending suitable restaurants in the area?
   A. Not at all
   B. Somewhat
   C. Moderately
   D. Mostly
   E. Completely

10. Any additional comments or feedback you would like to provide about your experience with the virtual concierge?

Usability Evaluation Results and Discussion

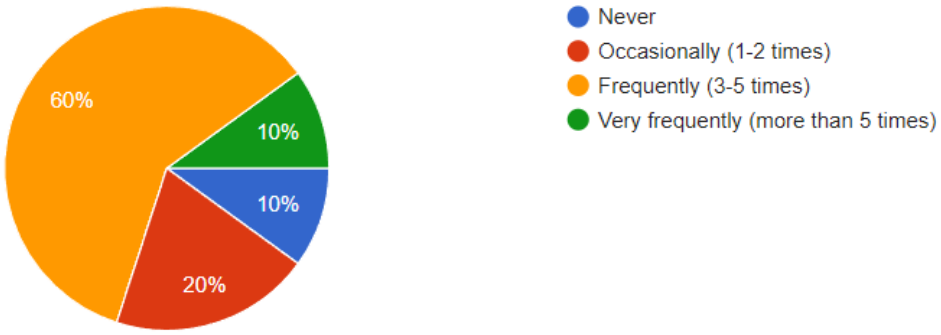
Pre-Study Survey.

Details. Before allowing a user to try the application, we wanted to gather some unbiased information relevant to the application. We accomplished this by performing a survey before

having the user use the application. This helped us gain insight into what people are looking for in a restaurant recommendation system and how useful it would be as a whole.

Figure 11

How often do you dine out per week?



Results. We started by asking the users how much they dine out during the week on average which resulted in the pie chart seen in Figure 11. As you can see, the majority of people dine out 3-5 times a week with 90% of people dining out at least once a week or more. This tells us that our application has the potential to help a lot of people find restaurants. The next question allowed the user to select any/all options that influence where they decide to eat. Through this, we found that the cuisine type and price were the two most important factors. These two were followed by recommendations from friends and online reviews. From this, we know that we must match the cuisine and price range as closely as we can and then make sure to emphasize good reviews. Our third question explored whether the user would try a restaurant based entirely on an online recommendation. All of the respondents indicated that they would be willing to trust an online recommendation which validates that our system can be trusted. None of the respondents had ever used an online restaurant recommendation system before which meant that

we had a clean slate to work with when it comes to expectations. This also means that the bar is set quite high as the users don't understand the limitations of the technology in this space.

Discussion. Overall, we felt that our pre-survey went very well. We were able to gain lots of useful information about what people want in a system like ours. However, some of the results were very expected like cuisine type and price being the two most important things to people. One of the biggest questions we had was if people would trust the system; and from our pre-survey, people would certainly trust an online recommendation system which was great news for us. We probably could have created a better question to figure out which kind of systems people had used in this area since nobody had used a system like this before.

Observations.

Details. After completing the pre-survey, but before the post-survey, we had the user play with the application without instruction. We observed how they interacted with the application and what went well/not so well. This was the true test of how well our system works.

Results. We do not have any direct metrics like in the surveys to pull from, but we have notes taken during the observations. We found that everyone was able to easily navigate through our system and was able to get recommendations as well as continue chatting with the bot. We did find a few things that we did not expect; for example, some people did not read the second message that the chatbot displayed which meant they did not know how to view their recommendations. We also discovered that people use much shorter queries than we had intended. This led to our development of prompt expansion using the GPT API. The biggest flaw that was discovered through our observations was that our system simply cannot handle some very specific, although pretty standard queries. This is mostly due to our limited amount of data and short time frame for development.

Discussion. The observations were probably the most useful piece of data we acquired because they showed us how people use the system. We had built up very specific use cases that made sense to us, but it turned out that people don't use the system that way. We feel that it would have been very helpful to get more of this kind of feedback throughout the entire development process. This could have led to a much better application; with only a few weeks left, we did not have time to fix all of the issues that were found. A lot of the issues would require a rebuild from the beginning.

Post-Study Survey.

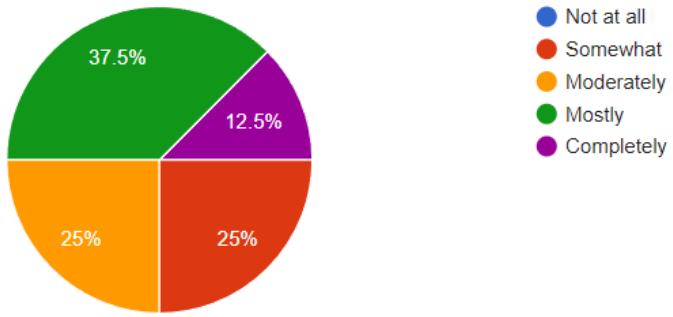
Details. After having the users try our application, we wanted to receive feedback on what they thought. We did this by providing a post-survey where they were asked questions in a similar fashion as the pre-survey. This was our point in the process of receiving vocal feedback from the user.

Results. The post-survey begins by asking the user if our chatbot accurately understood their queries. We were pleasantly surprised to find that over 60% of the participants felt that the chatbot did a good job of understanding what they were looking for. This led to our next question which asked about the quality of the restaurant recommendations themselves; 75% of participants indicated that the recommendations were good with only 12% feeling the recommendations were poor. One of the most important aspects of our application is if it provides recommendations that hit the points as discussed by the user. Figure 12 shows how the users felt our system did. Although only 12.5% felt that our system completely met their needs, 62.5% felt that our system was moderate or better. It was also great to see that we did not have any participants indicate that our system did not meet their preferences at all. It was awesome to see that 100% of the participants felt that our interface was easy to navigate. The largest problem

encountered was the chatbot not understanding specific queries. It was unable to properly understand things like “spicy cajun food”, or saying that you didn’t want something after specifying that you did want it. Whether or not the participants would use our application again was very spread out. A lot of people indicated that they would not use it again simply due to the lack of understanding. However, most people did indicate that they would recommend it to others to try.

Figure 12

Were the recommended restaurants suitable for your preferences and requirements?



Discussion. The post-survey went very well, as we were able to really understand what we did well and what we can improve on. Our interface was very highly rated as being easy to use and appealing. This was great to hear as we spent a lot of time discussing design decisions to make it as easy to use as possible. The biggest flaw in our application was the lack of understanding on the chatbot side. Unfortunately, there isn’t much we can do to improve this at the moment because of a lack of data. We are leveraging all the metrics that we have available to us in our dataset, but it isn’t enough for complex queries.

Focus Group.

Details. For focus groups, we took a random group of students and gave them a little run-through of our virtual concierge and its main goal before showing them our application. A

total of 2 focus groups with 3-4 people were sampled for our user study and were given the same rundown of our virtual concierge. We followed almost the same structure as the survey questions from our interview but instead of giving them the questions before or after the use of our app, we focused on what the focus group was saying throughout the testing of the web app.

Results. The results showed that people liked the friendliness and easy-to-navigate components of the virtual concierge. They liked the personalized messages that the chatbot provided and were satisfied with the recommendations provided. Some of the feedback we got from the focus groups was to distinguish better between the restaurants in the cards and the corresponding pin on the map since it was hard to see where it was. It was also brought up that our back button did not work as expected as it did not continue the conversation.

Discussion. From the results, we can infer that all the users found the app very easygoing and intuitive. We can also see that, for most people, this app provides semi-accurate recommendations based on their prompt. This means that our app is working as intended. Some additional feedback that the users provided was to decrease the time it takes the app to query a result based on user prompts. They also wished to see the implementation of more cities. But overall, this app had a positive view from the user.

Engineering Standard

Environmental and Health/Safety Concerns

Energy Consumption

The MyConcierge website is built using servers, databases, and AI algorithms. All of these methods require energy consumption to operate. We need to ensure that this website is hosted on energy-efficient servers and optimizing code to reduce energy consumption can mitigate this concern.

Transportation Impact

While not directly related to the technology itself, users will be influenced by the recommendations provided by our application. In order to reach the recommended restaurants, users would most likely need to travel to the specific restaurant location. Their transportation choices could impact the environment by leading to air and noise pollution. Encouraging sustainable transportation options or promoting nearby dining establishments to reduce travel distances can help mitigate this impact.

Health & Safety of Users.

Considerations should be made regarding the user interface to ensure it is intuitive and easy to use, minimizing the risk of user frustration or stress.

Social, Political, and Ethical Concerns

Social

Our project has the potential to benefit the community. It makes it easier for groups of people to find a place to eat that makes everyone happy. This will remove a lot of the discord that comes with picking restaurants amongst a group of people with different desires. It could also just make people more happy by getting to try some new restaurants in their area.

Political

We can’t see our project having any impact on the political scene. Our application focuses on getting people to a restaurant that suits their needs. We have no desire to push political views in our application.

Ethical

The only area where our app could potentially have an ethical impact is fairness. It could be possible that some people would feel that their opinions are not being weighed fair enough

within a large group. Our algorithm takes no bias and tries to be as fair as possible, but sometimes people perceive things differently than a machine does.

Manufacturability, Sustainability, and Economics

Manufacturability

The engineering practices applied to the project ensure that the software components are modular, well-documented, and follow best coding practices. This design approach makes it easier to maintain and update the software, reducing the complexity of future modifications or feature additions. Additionally, the use of industry-standard frameworks and libraries enhances compatibility and scalability, contributing to the overall manufacturability of the project.

Sustainability

Social sustainability is brought about because by facilitating group decision-making and reducing conflicts over restaurant choices, the project promotes social sustainability within communities. It fosters positive social interactions and strengthens social cohesion among users. The project contributes to economic sustainability by supporting local restaurants and businesses. By showcasing a variety of dining options, including smaller and niche establishments, it helps diversify the local economy and supports entrepreneurship in the food industry.

Economics

The MyConcierge project focuses on efficiency and resource optimization. The project optimizes resource allocation by providing personalized restaurant recommendations based on user preferences. This targeted approach minimizes wasted resources, such as time and effort spent searching for suitable dining options. From a business perspective, the project supports economic efficiency for restaurants by connecting them with potential customers who are likely to appreciate their offerings. This targeted marketing approach can lead to increased customer

satisfaction and loyalty, contributing to the long-term success and sustainability of participating restaurants.

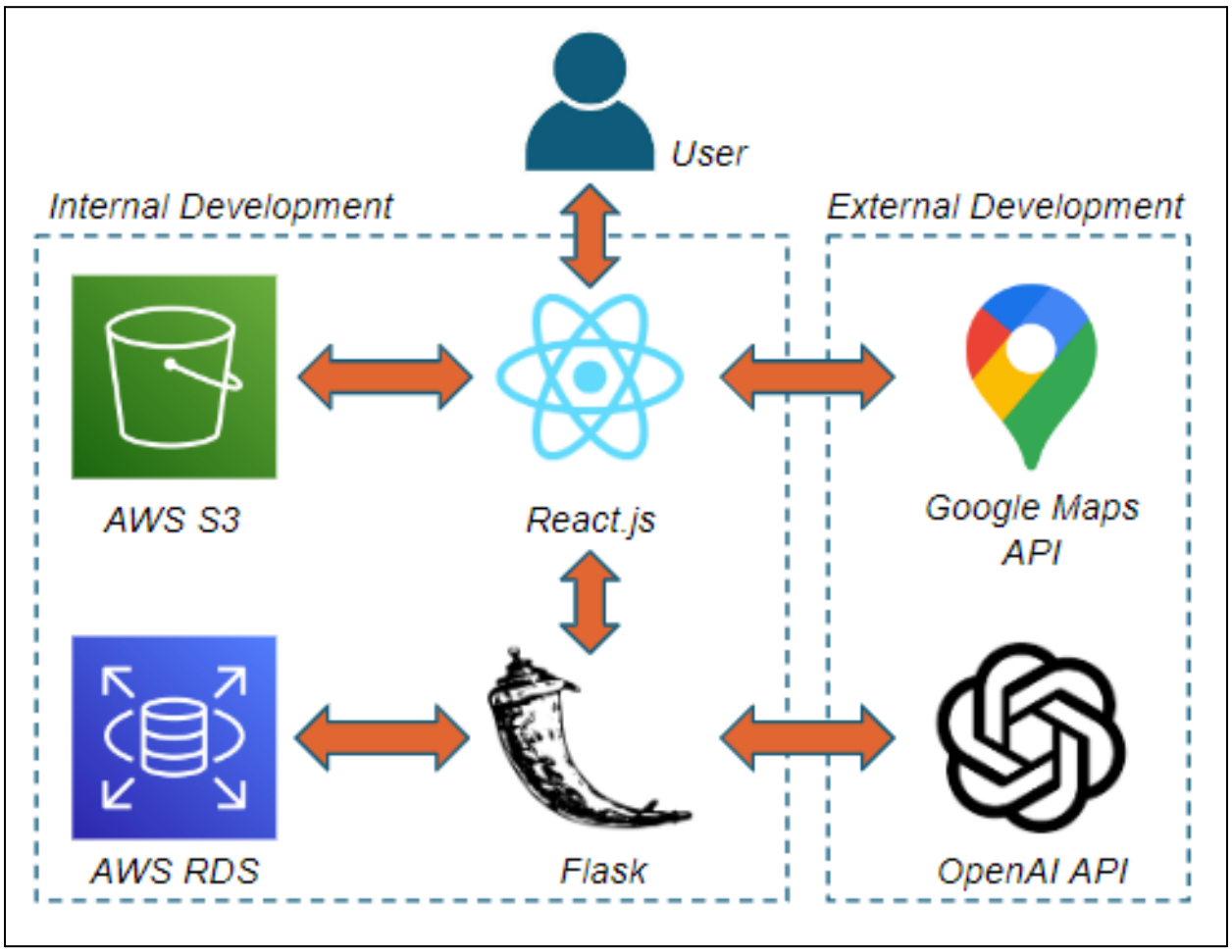
Implementation Details

System Hierarchy

The system hierarchy for this project is a combination of three major subsystems, the front-end, the back-end, and the data. The front-end system is developed using the React framework and is hosted through Vercel. The back-end system is developed using the Flask framework and is hosted through Vercel. The data is a combination of two parts, the AWS RDS database and and AWS S3 file storage. The combination of these two structures contains all the relevant information our application requires to operate. The front-end prompts the user for information regarding their location and what food they would like to eat. This raw data is sent to the back-end using GET HTTP requests. The back-end takes this information passes it to the NLP model and assesses the user's preferences. This populates the user preference vector and sends it to the recommendation system. The recommendation system takes in the user preference vector and compares that to the restaurant information in that given city. Using our recommendation algorithm, a set of restaurants is determined and sent to the front-end to display.

Figure 11

Hierarchy Diagram



Individual Components

The individual components of this system include:

- Back-end (Flask)
- Front-end (React)
- Database (AWS RDS)
- File Storage (S3 File Storage)
- Natural Language Processing (ChatGPT)
- Recommendation Algorithm

Tools Used

We employed Visual Studio Code as our IDE for developing the application and utilized Vercel for hosting our website.

Front-End

- Design Mockups: Figma served as our tool for creating mockup designs for the application's pages.
- React JS: The primary pages of our application were built using React JS with CSS. We integrated several external libraries such as Bootstrap for styling, FontAwesome for icons, react-router-dom for navigation, framer-motion for page animations, and multi-carousel for the image slideshow on the map screen.
- Custom Graphics: Procreate was utilized to create custom images for the application, including the bell icon on the start and map screens, and the robot picture on the chatbot screen.
- APIs & Widgets: We integrated the Google Maps API on the front-end to display maps and location pins for the map screen. Additionally, we incorporated a third-party widget from UserWay to enhance accessibility across our website.

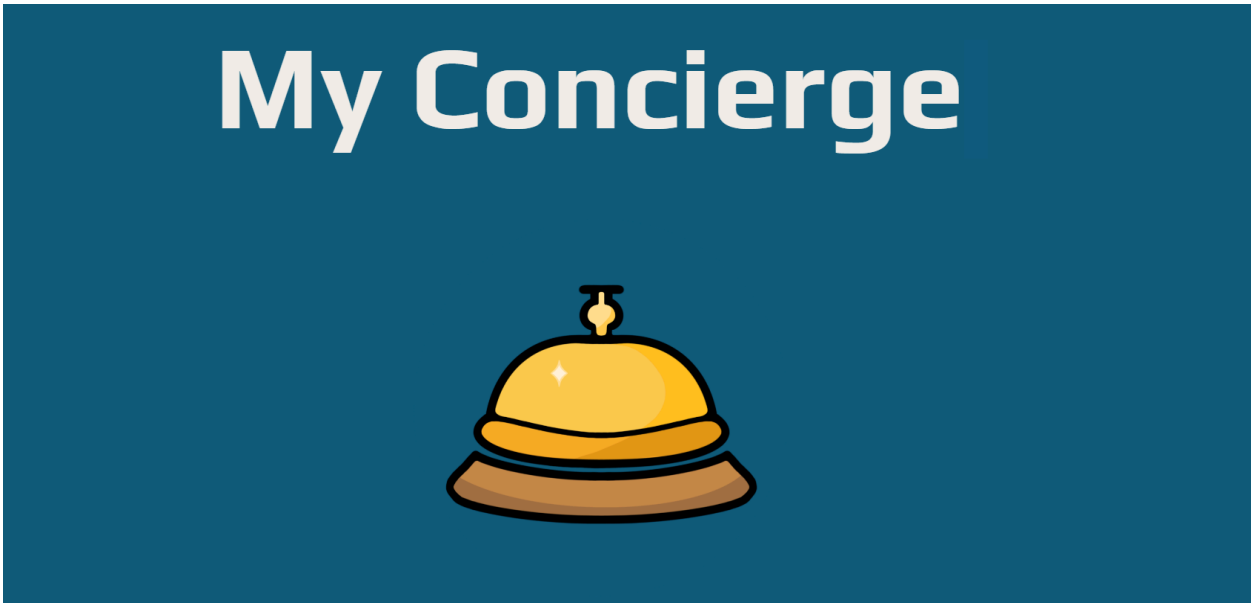
Back-End

- Python Flask: Python Flask was employed to develop queries for accessing specific data on the back-end.

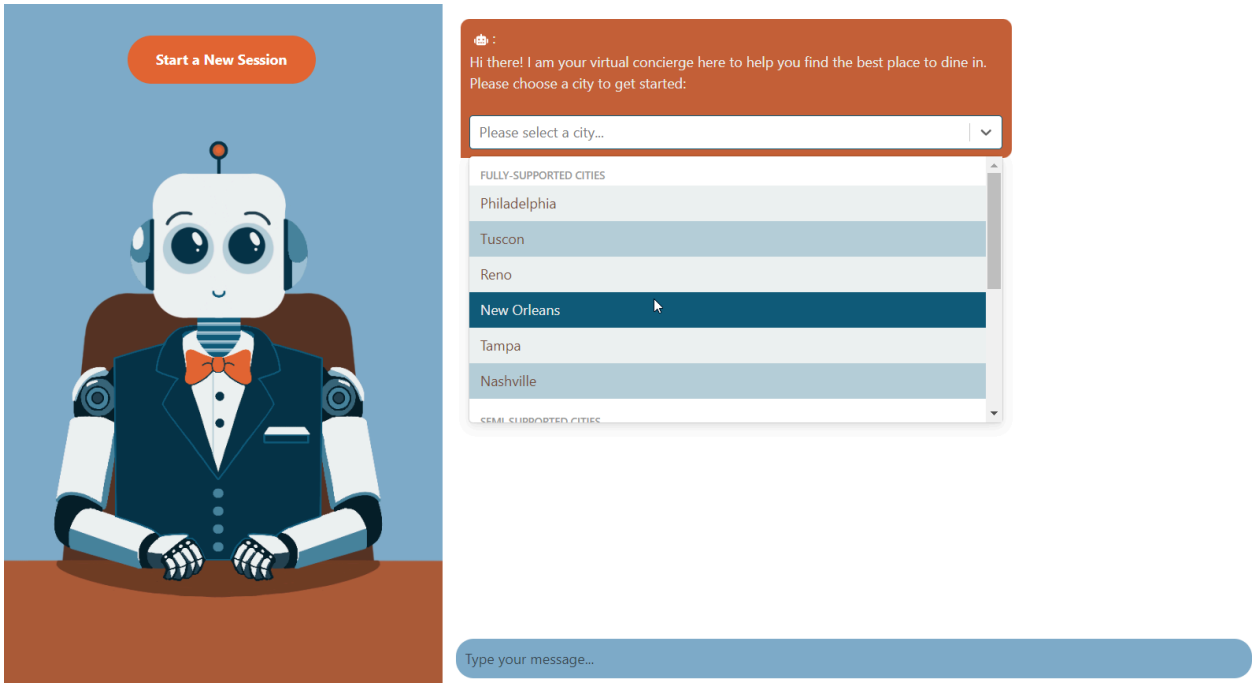
- Database: Using the Yelp dataset, we obtained information about restaurants in specific cities. We filtered and organized this data, creating an AWS database to store restaurant information. Furthermore, we utilized AWS S3 buckets to store images associated with each restaurant.
- Other APIs: We integrated the OpenAI API to train our chatbot and expand user prompts to create accurate user preference vectors for the recommender system. These user preference vectors evaluate prompts based on 35 different factors.

User Manual

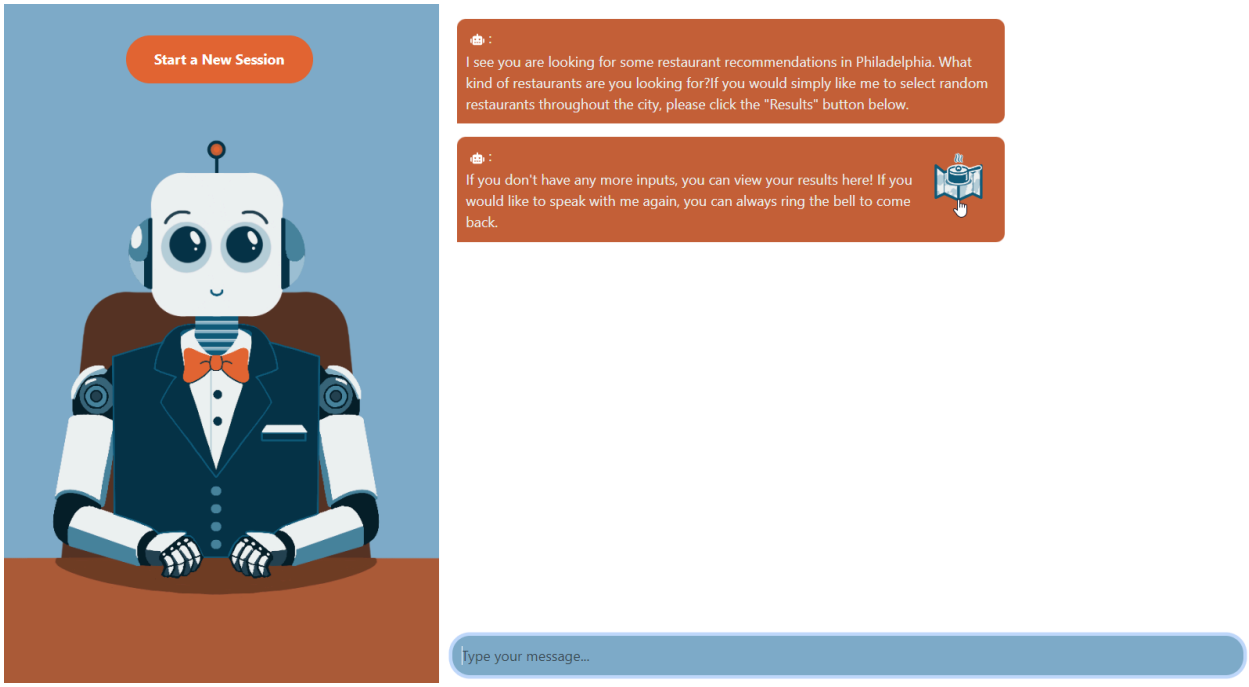
This project does not require any hardware or software. Users simply require a functioning browser like Google Chrome or Microsoft Edge. Users do not require a deep understanding of technology to operate the My Concierge application. This project aims to cater to a wide range of users, including those with varying levels of technological expertise. Our typical user is someone traveling to a new city, unfamiliar with the area, and seeking to discover new restaurants. We aimed to create an intuitive user interface with minimal barrier to entry. Users start by visiting this URL: https://my-concierge.vercel.app/. Once the site is deployed, user's will be directed to the homescreen where a bell button is present and once clicked an animation will be played.



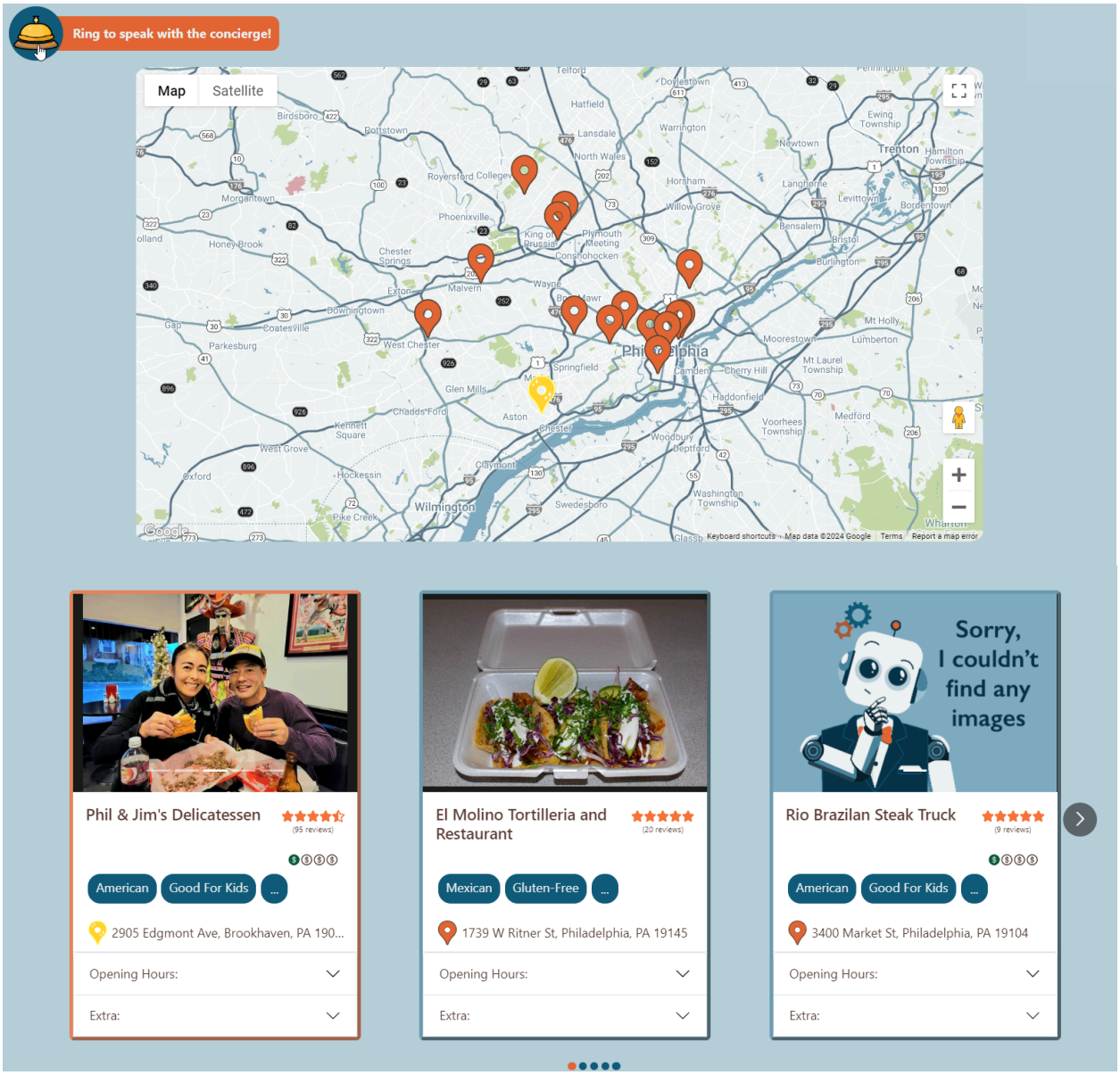
Once the button is clicked, the user will be directed to a chat screen where they can speak with the concierge chat bot. There is a list of fully supported cities and partially supported cities that users can ask the concierge about.



Users can select the icon at the end of the chat message to view results displayed on a map with respective cards below it. If a user would like to add additional details and tell the chatbot any additional information, a new list of results will appear.



Once clicking the results button, the top matches that the model generates will be displayed as pins. The top choice is indicated by a gold pin, reflecting the user's preference, while the remaining pins are marked in orange. Users have the option to select specific pins and their associated cards, allowing for a focused view.



During the session, the user can select the bell icon at the top of the screen and they will be brought back to the chatbot to add more clarifying information so more restaurant options can be generated.

Discussion

Insights

Upon thorough reflection on our team dynamics and interactions throughout this project, it is evident that communication played a pivotal role in our success. We now recognize the significance of fostering a more transparent dialogue among team members to mitigate misunderstandings and ensure alignment. Furthermore, we now understand the necessity of establishing comprehensive development plans with clear deadlines. We also think that programming in a more collaborative environment may have helped with the disconnected communication. While individual contributions to the user interface were appreciated, challenges arose during integration phases, primarily due to hesitancy among certain team members to embrace necessary changes. This highlighted the need to cultivate an environment within our team that prioritizes adaptability and collaboration. It emphasizes the significance of collective decision-making processes in facilitating smoother project advancement. Furthermore, a crucial area necessitating enhancement was our time management. Although there existed mutual trust in our team's ability to deliver a final product, last-minute rushes for reviewing and revising deliverables happened too often. The establishment of a consistent standard for the quality of deliverables by team members would diminish any inclination towards distrust and remove the necessity for comprehensive revisions. In retrospect, our future projects would benefit from an emphasis on transparent communication, clear and concise planning, and collaborative coding sessions. These refinements are sure to elevate our team's efficacy and productivity, ensuring a more streamlined and successful execution in the future.

Reflection

We think the testing performed on our product was efficient and yielded valuable insights into its functionality and performance. We would have liked to perform more in-depth tests, especially when it came to user testing to get a better understanding of user needs. Our testing approach reflects a pragmatic balance between manual and automated methods, tailored to the project's size and complexity. While manual testing proved effective for the current scope, you acknowledge the potential benefits of automated testing, especially with future project expansion and team growth. This forward-looking perspective highlights a readiness to adapt testing strategies to evolving project needs, ensuring continued efficiency and reliability throughout potential future development processes.

Future Work

Although this project satisfies the success criteria we initially planned, it has a few limitations that could be addressed in the future. One of the most challenging aspects of this project was gathering a comprehensive and precise dataset of restaurant information. We were able to get useful restaurant information through Yelp's Open Dataset, however, it only fully supported a few more obscure cities. To bridge this gap and access data from more well-known cities, we incorporated data from the Yelp Fusion API. Due to this constraint, our application could not be deployed for all cities within the United States, but we were able to cover thirteen cities with six being fully supported and seven partially supported. For future development, we would like to use more accurate data that will support all or most of the cities in the United States. Furthermore, we realized that due to the nature of the OpenAI API, our chatbot was not able to understand negative queries (e.g. I do not want Italian food). These queries are important

to take into consideration for providing representative results. Therefore, we would like to expand the functionality of our chatbot and add support for negative queries. Additionally, we noticed that though our chatbot provided suitable recommendations for users, some specific cuisines were not supported by the Yelp Open Dataset like Cajun. In the future, we aim to expand the range of supported cuisines and other categories to cater to more specific preferences, thereby enhancing the versatility of our application. Moreover, we would like to refine our recommendation algorithm to better accommodate continuous conversations. The chatbot assesses each query independently and we locally merge the resulting user-preference vectors. While this approach yields accurate results by combining the user preference vectors, it occasionally struggles to understand the variations in user desirability. Each new prompt is handled independently and even though one cuisine may be more important to the user, the chatbot does not take this adjusted weight into account. To address this, we are exploring the option of concurrently deriving preferences from the entirety of the conversation, rather than analyzing each query independently. This transition would facilitate a better understanding of user preferences, allowing the chatbot to adapt and respond more effectively to various conversational contexts with greater flexibility. Lastly, we envisioned that our application would work best as a kiosk system in hotels. Our goal is to incorporate geolocation services into our platform, enabling users to visualize their proximity to recommended restaurants. This feature will streamline the restaurant selection process, as users can easily identify nearby dining options based on their current location. Such a feature would make the application more useful on mobile devices.

Conclusion

MyConcierge represents a cutting-edge virtual concierge service designed to revolutionize group decision-making in dining experiences. Leveraging advanced technologies like OpenAI's GPT 3.5 model for natural language processing and the Google Maps API for geolocation, our platform simplifies restaurant selection for travelers and groups. Through AI-driven chatbots, users can input preferences, and receive tailored recommendations based on cuisine, dietary needs, price, and location, streamlining the decision process and offering personalized suggestions.

The modular architecture ensures scalability and adaptability, while APIs like OpenAI API and Google Maps API enhance user experiences. Our structured data design enables efficient data management and retrieval, providing comprehensive restaurant information. MyConcierge showcases the potential of AI in hospitality, promising continued advancements in recommendation algorithms, geographical coverage, and user interface enhancements for a seamless travel experience.

Sources

Bannerman, P. L. (2008). Defining project success: a multilevel framework. Paper presented at PMI® Research Conference: Defining the Future of Project Management, Warsaw, Poland. Newtown Square, PA: Project Management Institute.

Chokhra, P. (2021, April 20). Evaluating Recommender Systems. Medium. https://medium.com/nerd-for-tech/evaluating-recommender-systems-590a7b87afa5

Cohen, Y. A. and Stefon, . Matt (2023, September 5). *dietary law*. *Encyclopedia Britannica*. https://www.britannica.com/topic/dietary-law

DeepQ (2021) *Food Map* [Mobile App]. Google Play Store. https://play.google.com/store/apps/details?id=com.nerdgeeks.foodmap&hl=en&gl=US

Department of Health and Human Services. (2013a, September 6). First Click Testing. Usability.gov. https://www.usability.gov/how-to-and-tools/methods/first-click-testing.html

Department of Health and Human Services. (2013b, October 8). Usability evaluation basics. Usability.gov. https://www.usability.gov/what-and-why/usability-evaluation.html

Elboim-Gabyzon, M., Weiss, P. L., & Danial-Saad, A. (2021). Effect of Age on the Touchscreen Manipulation Ability of Community-Dwelling Adults. *International journal of environmental research and public healthand* , *18*(4), 2094. https://doi.org/10.3390/ijerph18042094

Hollander, J. (2023, November 1). A complete guide to hotel digital concierges in 2024. A Complete Guide To Hotel Digital Concierges in 2024. https://hoteltechreport.com/news/hotel-digital-concierge

Hitchens, K., & C., S. (2017, March 2). *What percentage of US hotels employ concierges? How many hotel concierges are there in the United States? How many people staying in hotels utilize hotel concierge services (when they are offered)?*. AskWonder.com. https://askwonder.com/research/percentage-us-hotels-employ-concierges-hotel-concierges-united-states-people-pcaqw6j6a

Kumar, B., & Sharma, N. (2016). Approaches, issues, and challenges in recommender systems: a systematic review. Indian J. Sci. Technol, 9(47), 1-12.

McCarthy, J. F. (2002, April). Pocket restaurant finder: A situated recommender system for groups. In Workshop on mobile ad-hoc communication at the 2002 ACM conference on human factors in computer systems (Vol. 8).

Scheibehenne, Benjamin & Greifeneder, Rainer & Todd, Peter. (2010). Can There Ever be Too Many Options? A Meta-analytic Review of Choice Overload. Journal of Consumer Research, v.37, 409-425 (2010). 37. 10.1086/651235.

U.S. Census Bureau. (2022, March 8). *Language spoken at home*. Census.gov. https://www.census.gov/acs/www/about/why-we-ask-each-question/language/