ChatGPTravel

By Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

Project Overview

Our project aims to bridge the gap between knowing where you want to travel to, but not knowing what to do at your destination once you arrive. We plan to develop a web service, called ChatGPTravel, that creates a travel itinerary for a user-specified location based on their intended travel preferences and activities in the area. We plan to leverage the wealth of knowledge available ChatGPT through its generative AI APIs to generate suggested activities and combine it with our algorithmic knowledge to create itineraries optimized on objective measures, such as cost, rating, travel time and a measure that combines all three measures.

Value Proposition

Currently, many different travel services are trying to integrate generative AI into travel planning services. For example, Tripnotes is a service that provides AI-powered smart itinerary scheduling and planning[1], but it does not account for user preferences. Moreover, Expedia, an online travel agency, has begun utilizing generative AI to allow users to converse with an AI-powered chatbot about aspects of trip planning and save recommendations it makes for later[2]. However, neither of these services generates unique, personalized, optimized itineraries.

Technical Innovation

The purpose of our project is to create optimized itineraries for users based on their indicated preferences of objective measures such as cost, rating or travel time. Our project is a novel combination of traditional optimization algorithms that will be used to minimize and maximize objective measures and an itinerary that consists of activities suggested by generative AI. There are product, such as Wonderplan.ai [3]that create itineraries based on preferences, such as a budget, the number of travelers, and activities of interest. However, these products do not optimize the itinerary to be the cheapest, best rated, or least travel time. Therefore, there are no products on the market that combine algorithmic optimization techniques and AI-generated travel itineraries to create a unique user experience.

---

[1] https://aicenter.ai/products/tripnotes#:~:text=TripNotes%20is%20an%20AI%2Dpowered,travel%20plans%20efficiently%20and%20effectively.

[2] https://www.cio.com/article/481289/expedia-poised-to-take-flight-with-generative-ai.html

[3] https://wonderplan.ai/

Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

Customer Description:

Our ideal customers are middle-class individuals with disposable income who want to travel to new places but do not know much about those places. These users might have some preferences, such as minimizing cost, experiencing the best-rated activities, or traveling as time efficiently as possible. However, they do not have strong opinions about what they should do and see once they arrive at their destination. This project intends to address a pain point many people experience of wanting to travel to a location, but not knowing what they should do and see there. Without this web service, customers would have to spend a considerable amount of time reading articles and reviewing potential activities before creating a complete itinerary for themselves. Our service would decrease this time investment for users and hopefully enable them to travel more.

User Stories

As a traveler, I would like to specify a destination, arrival, and departure date to get an itinerary for the specified destination that has the least transit time, so I can be as efficient as possible on my trip.

As a traveler, I would like to specify a destination, arrival, and departure date to get an itinerary for the specified destination that is the cheapest possible itinerary, so I can spend as little as possible on my trip.

As a traveler, I would like to specify a destination, arrival, and departure date to get an itinerary for the specified destination that is the best-rated itinerary, so I can maximize the quality of the activities I perform on my trip.

As a traveler, I would like to view my itinerary so I can know what I will be doing on my trip.

As a traveler, I would like to be able to view detailed information about each activity in my itinerary, so I can know the location, description, and rating of the activities in my itinerary that I plan to do on my trip.

As a traveler, I would like to be able to view my itinerary on a map, so I can have a visual representation of my itinerary that is a more digestible view.

As an experienced traveler, I would like to view my past or saved itineraries, so I can plan multiple trips or view different types of trips in the same location.

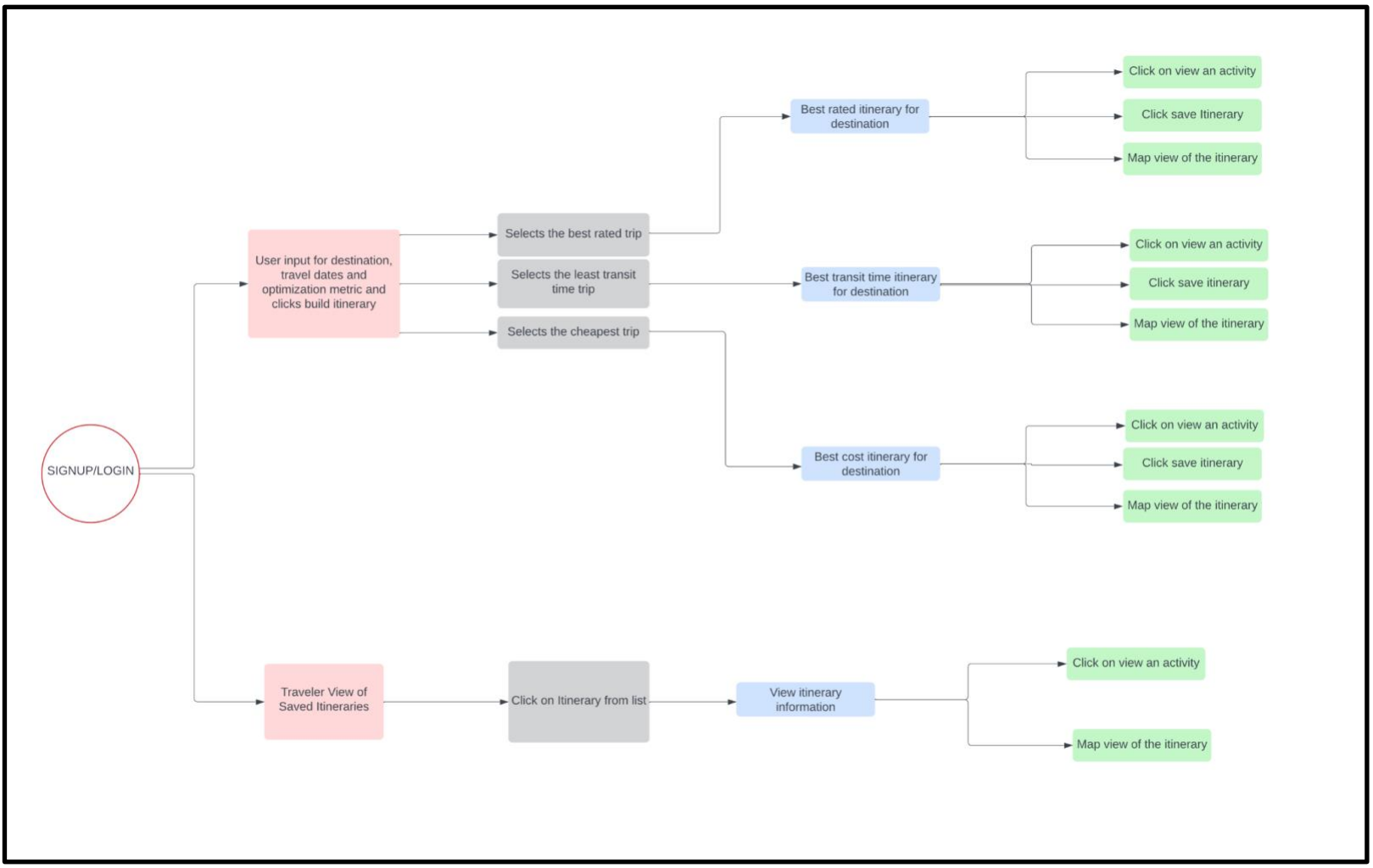
Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

As an experienced traveler, I would like to rate the different activities I visit, so in the future the application can recommend similar or better activities during a trip to the same or new destinations.

Flow Diagram

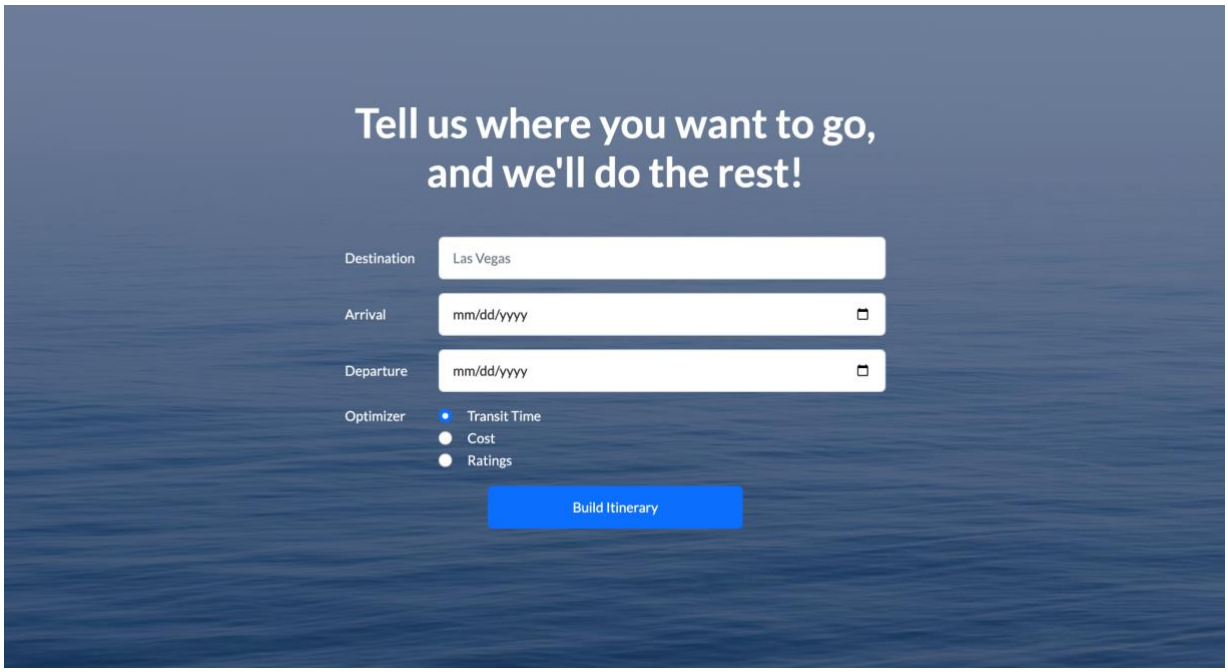
First, the user must log in or sign up if needed and either navigate to the screen where they can give input for an itinerary or check their previously saved itineraries. If the user chooses to enter input for a new itinerary they will be prompted to give a destination, travel dates, and choose an optimization metric. The application will then produce the itineraries based on the chosen optimization. The user will then have the option to save the itinerary, view specific activities in the itinerary, or view the itinerary on a map. If the user clicks on the traveler view of saved itineraries they will be able to click on any itinerary from their list of saved itineraries and their associated information. When viewing a specific activity's information the user will be able to see the activity, address, description, and rating.

Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

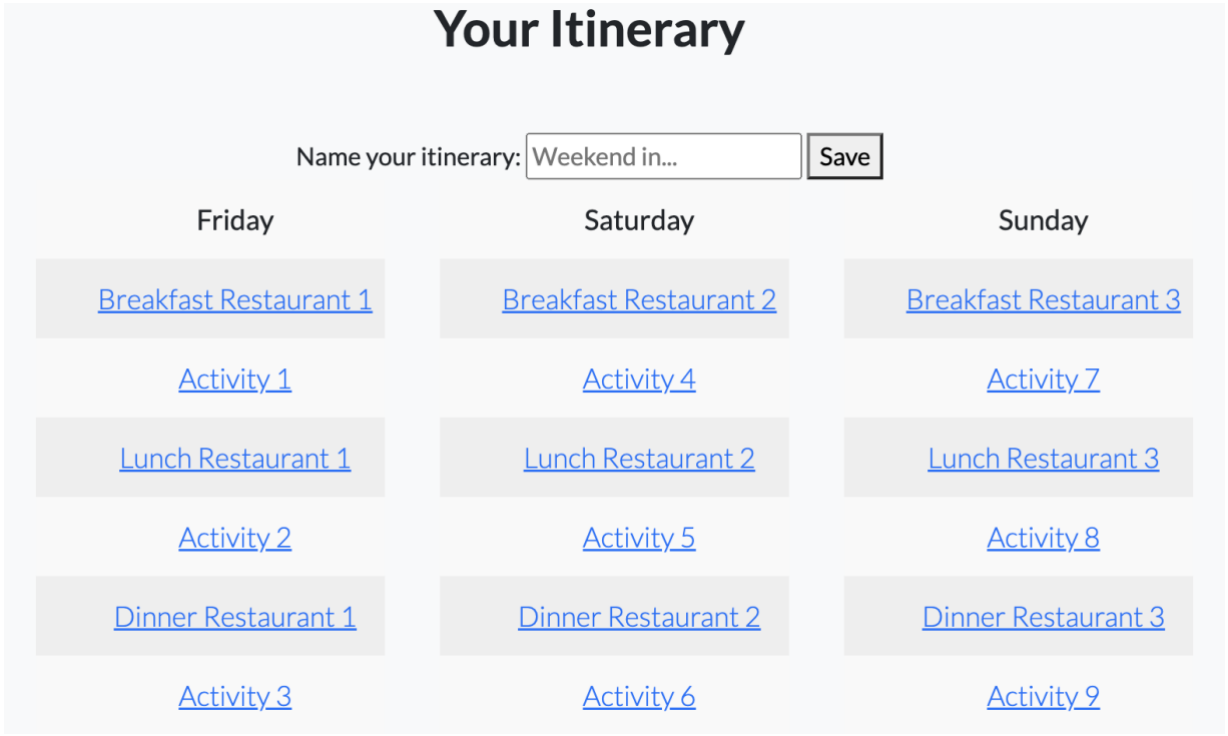


Mockups/Wireframes

Traveler - Input Travel Details



Traveler - View and Save Generated Itinerary



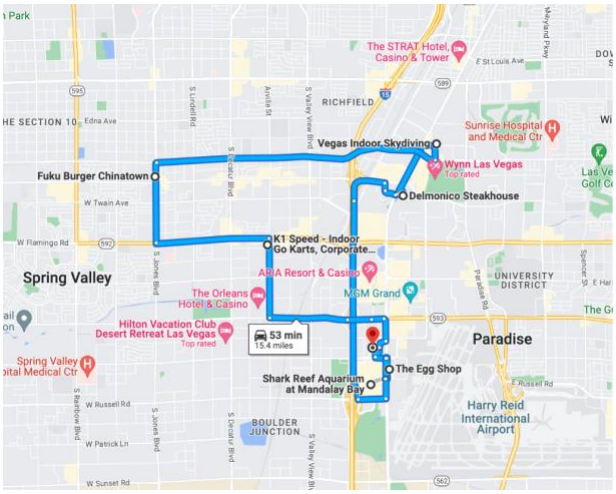
Traveler - View Restaurant/Activity Information

Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

Breakfast Restaurant 1

Address
Description
Ratings

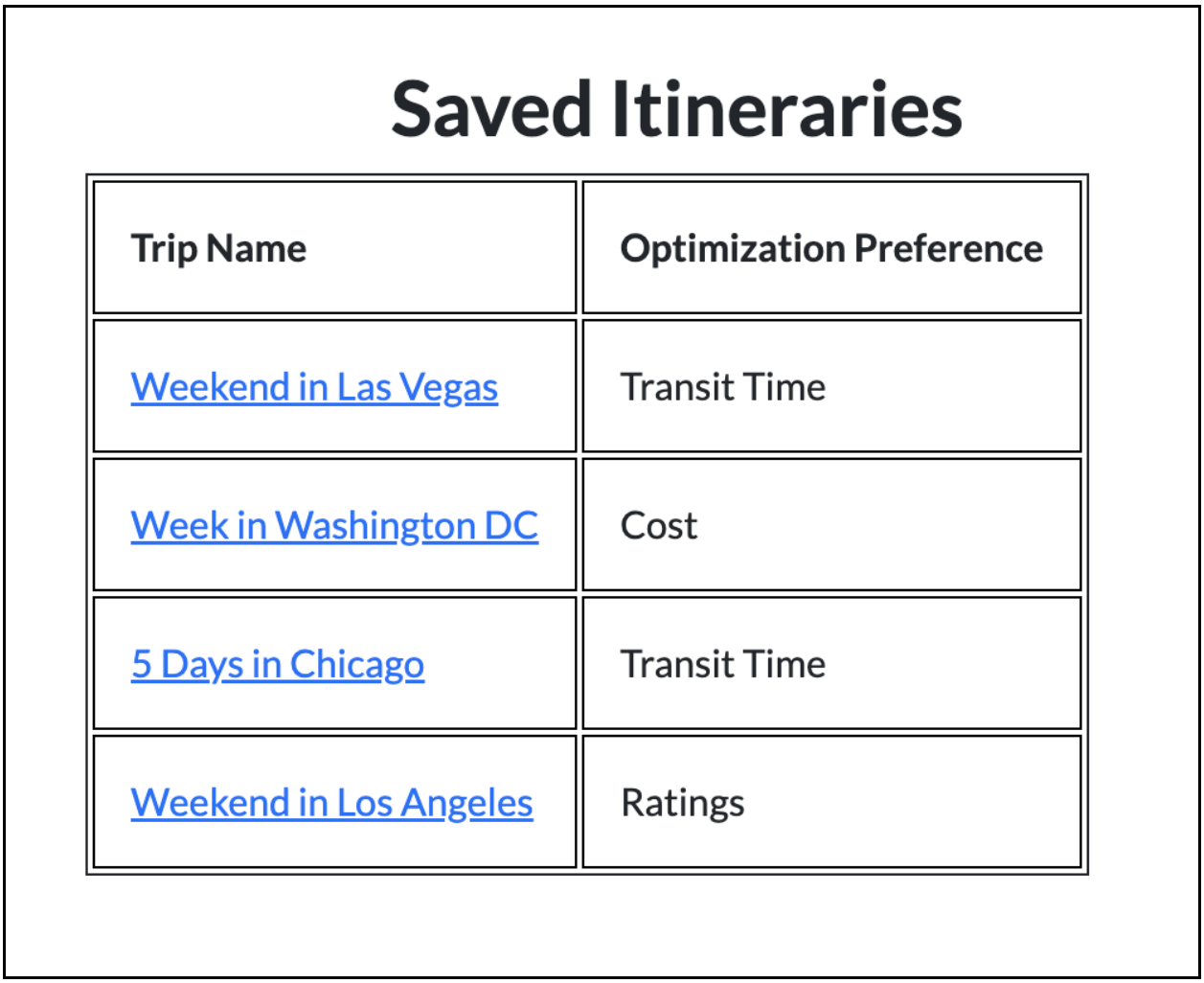
Traveler - View Map of Itinerary Activities/Restaurants



Saturday in Las Vegas

The Egg Shop
Shark Reef Aquarium at Mandalay Bay
Delmonico Steakhouse
Vegas Indoor Skydiving
Fuku Burger
K1 Speed Indoor Go Karts

Traveler - View Saved Itineraries



Saved Itineraries

Trip Name	Optimization Preference
Weekend in Las Vegas	Transit Time
Week in Washington DC	Cost
5 Days in Chicago	Transit Time
Weekend in Los Angeles	Ratings

Technical Specifications:

First, we will use generative AI services, such as the chatGPT to yield a set of possible destinations and activities that can be visited during a trip at a specified location. This will be done by creating a list of queries that will return elements of an itinerary, such as "the best restaurants in X" or "things to do in X." The process is supposed to define a finite search space, which we can then search using optimization algorithms.

Before we begin the optimization process, we must confirm that the destinations and activities suggested by chatGPT are possible. Often, generative AI APIs do not always return completely accurate information. For example, chatGPT [4] was only trained on data up until 2022. There have also been multiple instances where chatGPT[5] has returned factually inaccurate information. Therefore, we

[4] https://chat.openai.com/

[5] https://deepchecks.com/risks-of-large-language-models/#:~:text=Consequently%2C%20LLM%2Dgenerated%20answers%20can,of%20biased%20or%20discriminatory%20answers.

must confirm the existence, availability, category and other pertinent information of each destination or activity using the YelpAPI[6] and Foursquare API[7] to programmatically access the information about business available on Yelp and enrich the information we already have about the destination. As such, we can pass in the information we currently know about a destination or activity, such as the name and city it is located in and receive confirmation it exists and others receive useful information about the business, such as its hours, address and phone number. However, the YelpAPI and Foursquare API cannot yield all of the information we might need about a given destination or activity. Therefore, we plan to augment the results with other APIs, such as the Google Maps API for the amount of time it will take to travel from point A to point B and travel APIs to estimate cost accurately.

Once we have a list of viable destinations we can develop the algorithmic portion of the project. Our project is considered a combinatorial optimization problem[8], which is a class of optimization problems with discrete decision variables and a finite search space, but the space is still too large for an exhaustive search to be a realistic option. Our search space is the list of destinations yielded through the generative AI and confirmation process, while the decision variables are the objective measures, such as cost, rating, travel time and a composite score that accounts for all of these three measures. Therefore, we can create itineraries using combinatorial optimization techniques and algorithms.

Architecture/System Diagrams

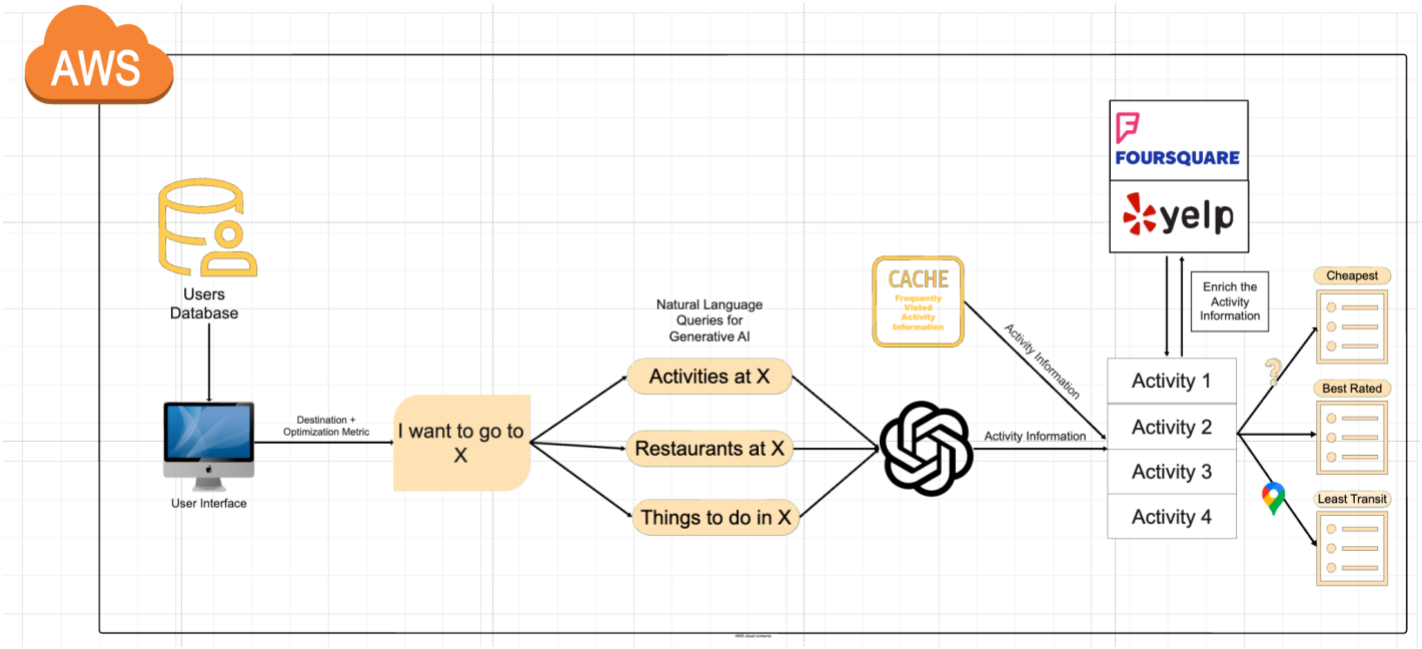
We plan on hosting our application through an AWS server once we have designed and built on the algorithmic and website portion appropriately. Therefore, the architecture design is all contained within AWS. The purpose of chatGPTravel is to take a destination and optimization metric from the user and return an itinerary for said destination using generative AI and the corresponding optimization algorithm. First, the user must log in to the application using the system UI. We plan on maintaining and updating a database of registered users with their associated user information, such as username, password, and saved itineraries. Once the user is logged into the application, they can view their saved itineraries and generate new itineraries through the user interface. The most interesting and important section of our application is the itinerary generation functionality. When the user enters a destination, dates and optimization metric, we use a set list of natural language queries that are

[6] https://docs.developer.yelp.com/docs/fusion-intro#:~:text=The%20Yelp%20Fusion%20API%20allows,simple%20scenario%20using%20the%20API.

[7] https://location.foursquare.com/products-places/?utm_source=google&utm_medium=cpc&utm_campaign=RL_Foursquare_Search_Branded&utm_term=foursquare%20api&gclid=Cj0KCQiA7OqrBhD9ARIsAK3UXh27DwgPk7acAWEOpIS1hjdgTWbhtJxp8iS52a6cKohn5-zsiGCu9zEaApRAEALw_wcB

[8] https://www.sciencedirect.com/topics/computer-science/combinatorial-optimization-problem#:~:text=Combinatorial%20optimization%20problems%20(COPs)%20are,%2C%20%26%20Vygen%2C%202012).

designed to yield the best possible destination suggestions from chatGPT. We then use these queries to get a list of destination suggestions from chatGPT. However, often the information returned from chatGPT on these destinations can be quite limited as well as there have been instances where important details about the destination, such as the address and phone number, are incorrect. Therefore, once we have generated possible destinations from chatGPT, we then use the YelpAPI to enrich the information we have about each destination by pinpointing the activity using the YelpAPI and the destination provided by the user to yield specific information about the activity, such as location, phone number, operational hours, rating, etc. and confirm the activity is available. Moreover, our application will maintain a cache of frequently visited destinations' associated activities to improve the run time of our application, given that chatGPT can be quite slow. Finally, the system needs to decide which optimization algorithm to apply: fastest, cheapest, or best rated. This decision is based on what the user selected when they specified the location and optimization metric they would like their itinerary to use. The appropriate algorithm is applied and then the itinerary is returned to the user.



External APIs and Frameworks:

OpenAI API:

- Goal
  - Generate recommendations for activities and restaurants in the location that the user wishes to visit.
- Description
  - This API is used in the ChatGPT_AI.py file in the code. The API receives a query and returns information in JSON format. This is then sent to a separate file, either

Activities.json or Restaurants.json depending on what is being output. The query that is sent to the API is generated based on the input from the user regarding where they would like to travel. The user inputs a destination, the destination is then fed into the query, the query is sent to the API, and the API returns the results to be sent to the JSON files. The text-davinci-003 model is being used to generate the responses.

- Endpoints
    - GET /v1/completions

Yelp API:

- Goal
    - Verify and enrich the restaurant recommendations given by the generative AI.
- Description
    - This API is used in the Yelp.py file in the code. The API uses the information in the Restaurants.json file and passes in the name of an individual restaurant, as well as the city in which it is located. A request is then sent using these two pieces to return the information that the Yelp database has on the restaurant. The information we are interested in receiving are the name, address, latitude and longitude, phone number, cost estimate, restaurant category, and ratings of the restaurant. This information is then stored in an Activity object to be used when generating optimized itineraries later on in the workflow.
- Endpoints
    - GET /v3/businesses/search/{restuarant_name}&{city}

Foursquare API:

- Goal
    - Verify and enrich the activity recommendations given by the generative AI.
- Description
    - This API is used in the Foursquare.py file in the code. It plays a role very similar to that of the Yelp API. The name of an individual activity from the Activities.json file gets passed in with the latitude and longitude. The API then returns information about the activity that is stored in its respective database. Similar to the Yelp API, the name, address, latitude and longitude, phone number, cost estimate, activity category, and ratings are stored in an Activity object to be used later in the workflow to generate optimized itineraries.
- Endpoints
    - GET /v3/places/search/{activity_name}&{latitude}&{longitude}

Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

Google Maps API

- Goal
    - Calculates the amount of time it takes to travel between two activities' addresses
- Description
    - This API is used in the GoogleMaps.py file in the code. It connects to the least transit time algorithm (fastest_itinerary) functionality and is called when the user requests an itinerary that has the least possible transit time. The parameters are an origin and destination, which are generated by the travel_time function and passed into this API to get the result. The travel_time is also utilized to implement the total_itinerary_travel_time, which calculates the travel time for the entire itinerary once generated by the fastest_itinerary function.
- Endpoints
    - GET /directions/{origin}&{destination}

Algorithms:

Best Rated Algorithm :

Goal: Create an itinerary with the best-rated activities/restaurants using rating data from the Yelp, Tripadvisor, and Foursquare API.

Description: The implementation of the best-rated algorithm returns the best-rated itinerary. Initially, the algorithm filters through a list of destinations, focusing on restaurants, and selects clusters of top-rated restaurants based on the user's specified rating range for breakfast, lunch, and dinner. The list of activities is passed as an argument to the function. Each activity will have a rating attribute, which comes from the Yelp API. For the alpha demo, the algorithm adopts a greedy strategy. It will iteratively choose the highest-rated restaurant within the specified cluster for each meal category: breakfast, lunch, and dinner. The selection process continues until the top three restaurants are chosen for the day within their appropriate category. Future iterations of this algorithm aim to incorporate advanced optimization such as dynamic programming. We plan to use dynamic programming to have the algorithm implement a decision-making process that will adapt based on more user-specific input. Moreover, we intend to implement conditional statements to adjust the algorithm's behavior to make more specific decisions, such as the types of activities, modality (indoor/outdoor), and physicality of activities to determine the best itinerary for the specific user.

Sidra Hussain, Colin Ranck, Sarah Jagerdeo, and Evan Fries

Least Travel Time Algorithm:

Goal: Create an itinerary with the least amount of time spent in transit using the GoogleMaps API to calculate the distance between two activities/restaurants and the Yelp, Tripadvisor, and Foursquare API for information about the activities/restaurants.

Description: The implementation of this algorithm will use established solutions that have been applied to the traveling salesman problem (TSP), such as greedy approaches, simulated annealing, and dynamic programming. First, the algorithm will filter the list of activities to find clusters of activities that are close together for each day in the itinerary. Ideally, a user will want to remain generally in the same area for every day of the trip to prevent unnecessary extra traveling. Then, it will pass each cluster to our least travel time algorithm to determine the order in which these activities will be visited. We intend to first implement a simple greedy algorithm for our alpha demo and improve the algorithm to return a more optimized solution by the beta demo. The greedy algorithm will work by finding the activity that is closest to the hotel in the cluster, and then continue to find the activity that is closest to the first activity, second activity, and so on until the itinerary is complete, which will then finally end at the hotel. Another approach we plan to use in the future is simulated annealing. A simulated annealing solution would perform as follows, select an initial itinerary, then randomly swap two activities in the itinerary. Next, the algorithm will use simulated annealing to compute the probability of whether we accept this new solution or not, if it has a certain probability the solution will be accepted. Finally, if time permits we will attempt to implement a dynamic programming approach. In this case, the distance value is calculated recursively. The base case will be when the number of activities in the subset is two and will return their distance. The recursive case will calculate the distance from the current city to the nearest city, and then recursively calculate the minimum distance between the remaining activities until the algorithm returns the minimum distance itinerary. If we are ultimately successful in implementing all 3 algorithms, then we plan to use information about the set of activities to determine which algorithm is best to use for a given data set. All of these algorithms have tradeoffs between time, space, and accuracy. Therefore, in any given circumstance a different algorithm can be more or less desirable for the data set.

Cheapest Itinerary:

Goal: Create an itinerary with the cheapest activities/restaurants using price data from Yelp, Tripadvisor, and Foursquare APIs

Description: The implementation of the cheapest algorithm returns the cheapest itinerary. Similar to the best-rated itinerary algorithm, this algorithm filters through a list of destinations and sorts them from cheapest to most expensive. The list of activities is passed as an argument to the function. As we do not have a way to obtain pricing data yet, this algorithm currently works only on dummy data. For the alpha demo, the algorithm adopts a greedy strategy. It iteratively chooses the cheapest restaurant for each meal category: breakfast, lunch, dinner, and non-meal. The selection process continues until the top five cheapest activities are chosen for the day within their appropriate category. This algorithm continues until all days are filled out with the cheapest activities.

Cost–Effective Itinerary:

Goal: Create an itinerary with the best rating/price ratio for activities/restaurants using price and rating data from Yelp, Tripadvisor, and Foursquare APIs

Description: This algorithm works in the same way as the cheapest itinerary algorithm, instead sorting activities by their rating/price ratio, instead of by price alone. Similar to the cheapest itinerary algorithm, it requires dummy data, as we currently do not have access to an API which can produce pricing data. This algorithm uses a greedy approach to retrieve five activities per day, ensuring that there are breakfast, lunch, dinner, and non-meal activities in their appropriate places. This algorithm continues until all days are filled out with the activities with the best rating/price ratio.

Technical Feasibility:

Many tools and technologies already exist to help us easily build our project, such as generative AI APIs, businesses and travel information APIs, and established algorithms. However, the challenge when designing and building this project will be developing methods to get the information we need from API interfaces. For example, sometimes chatGPT can be inconsistent with the results it yields on any given question asked. Therefore, we will need to design a robust list of queries that guarantees we will receive viable activities and destinations to use in our itineraries. Moreover, cost estimations are extremely difficult. There is currently no consistent way to determine how much it would cost to have dinner at a given restaurant and there are not any APIs that can return the cost of certain types of

activities. Initially, we wanted to use the travelocity API, but that is a paid service and we discovered it after the equipment request deadline, so we are currently working to find a viable alternative or submit an equipment request for it in the spring semester.

Costs, Risk and Risk Mitigation:

Generally, our project will not be particularly costly. However, we will incur a cost for hosting our website and using chatGPT. It is currently unclear how expensive using chatGPT will be because there is no publicly available information that relates the amount of use to the cost of using the API clearly. Many of these services try to obfuscate the cost to the user, so they cannot accurately control their usage. However, so far our team has not had particularly high chatGPT usage. While we cannot access the exact dollar amount that we have used in chatGPT credits, we can see how much our organization, which consists of our team along with three others, has used. During the entire fall semester our organization only used $16.37 worth of chatGPT credits for the alpha prototype development process. Our team individually has used In the future, if the cost of the chatGPT becomes too high, we plan on using a hard coded data set for algorithmic development and then integrating the AI capabilities once the algorithms are completed to minimize the amount of requests sent.