Don’t Trip!

A Quicker Way to Plan Your Trip

Authors: Jose Cruz Arinaga, Ryan Wong
Computer Engineering Department
California Polytechnic State University, San Luis Obispo

June 2019

Table of Contents

Introduction

There are many people today who like to travel to experience all of the best foods, sightseeing, and historical landmarks in every part of the world. People can use Google Maps to navigate their way from destination to destination and also have all of the information on the internet needed to plan an amazing trip on where to go and when. However, planning a trip in a new city can be very difficult and takes up a great amount of time out of one's day. For those that wait to plan until they actually arrive at their travel destination end up wasting precious time looking for places to go, when really they should already be there. What if there was a way travelers could type up a list of things they want to eat and see, and instantly receive an itinerary and route that is organized so that they can make the most out of their time they have in a foreign city?

Currently, Google Maps does not provide users with a way to find the optimal path for a user to travel given a list of points. Given a list of destinations you may want to visit, our web application will do all of the difficult planning for you and ultimately find the most optimal path for you to take to visit all of your destinations in a timely manner. The people who will benefit most from this application are tourists or people who travel often and want to explore an unfamiliar city. Even locals running their weekly errands could save time driving around back and forth between destinations by using the application to efficiently organize the order in which they should visit their desired stores.

Background

A lot of research went into the different back-end services of the web application. To implement the trip planner, similar web apps were found online to influence different UI and backend features of the project. The main app which we used as a guide for designing the overall look of our website was Google Maps. Given that our app simply adds features on top of a regular Google Maps box, referencing the Google's "Maps" application was helpful in shaping our app into something similar to start. Once the Google Maps application was somewhat mirrored in our app, we wanted to start on the extra features that would incorporate our functionality into the application. A lot of tutorials and examples online were referenced to create our final product. This includes tutorials by Google to set up a Google Map using Google's API's and using Google OR-Tools to determine the optimal path, examples by W3Schools for setting up a side navigation panel and hamburger menu, tutorials by Microsoft for using ASP.NET and SignalR to set-up a "hub" connection between server and clients, and many other open-source projects online which related to different features of our web app. A more interesting topic which we visited while looking at ways to solve the Traveling Salesman Problem was that of genetic algorithms vs. boolean satisfiability.

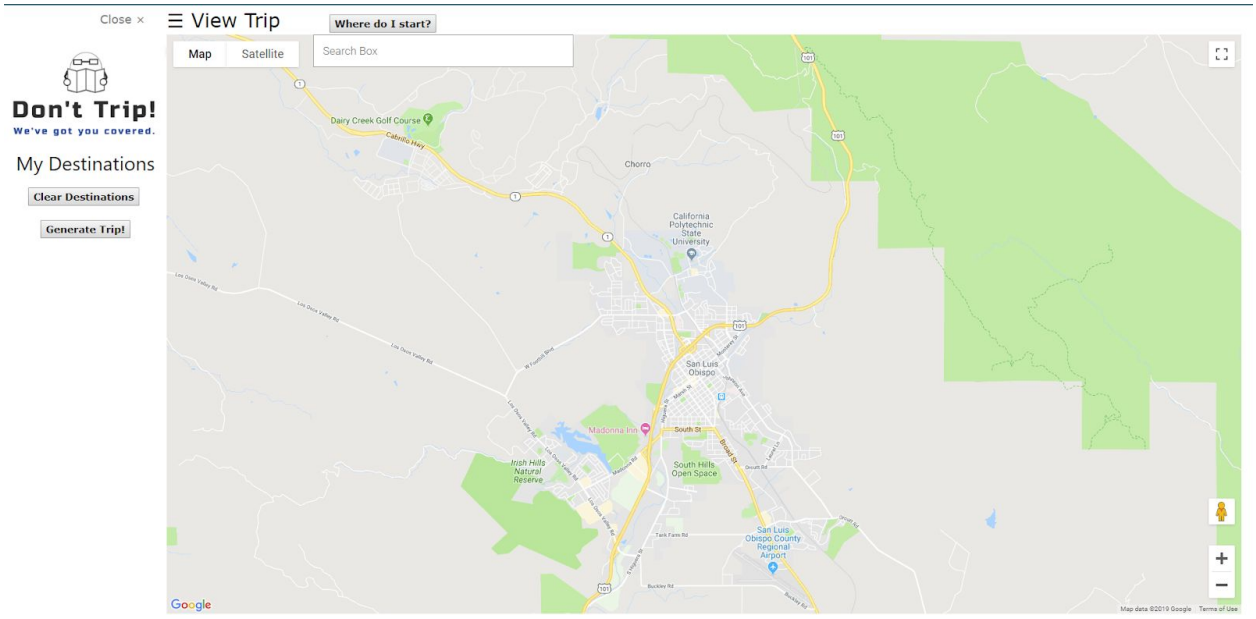
Tools

The following APIs were used to provide the main functionality for this project:

- Maps Javascript API: The foundation on which the project is built. Used to draw the map and enable interaction from the user. Provides functionality such as being able to drop markers on the map, create Info Windows for each marker, and switch between a satellite and traditional map view.
- Distance Matrix API: Used to calculate the distances between a set of locations. Takes an array of locations in the form of a latitude and longitude pair and returns a matrix with distances between each pair of locations. Can take up to ten locations.
- Directions API: Once the distance matrix is returned from the Distance Matrix API, this API is used to draw the optimal route between each location.
- Geocoding API: Used to translate latitude and longitude pairs to human readable addresses.
- Places API: Used to display information about a location that is already known by Google. Mainly used when someone searches for a location (such as a restaurant or museum) to display information about that location.

In addition to the APIs listed above, the Google OR-Tools Travelling Salesman Problem (TSP) Solver was used to calculate the optimal path from the given distance matrix boolean satisfiability. Because a JavaScript version of the solver was not available, the C# version was used. Because the C# version for the TSP Solver was used, the project needed to involve a web server to receive and push data to and from the C# backend and Javascript frontend. To achieve this communication between the C# server and Javascript client, Microsoft's .NET framework and signalR library were used to pass the necessary data for the TSP Solver to work. The .NET framework was also used to build the components into an ASP.NET application. This application was then hosted on a virtual machine running Ubuntu 16.04. In order to allow the application to be accessed from the outside world, an Nginx web server was used as a reverse proxy server.

System Design

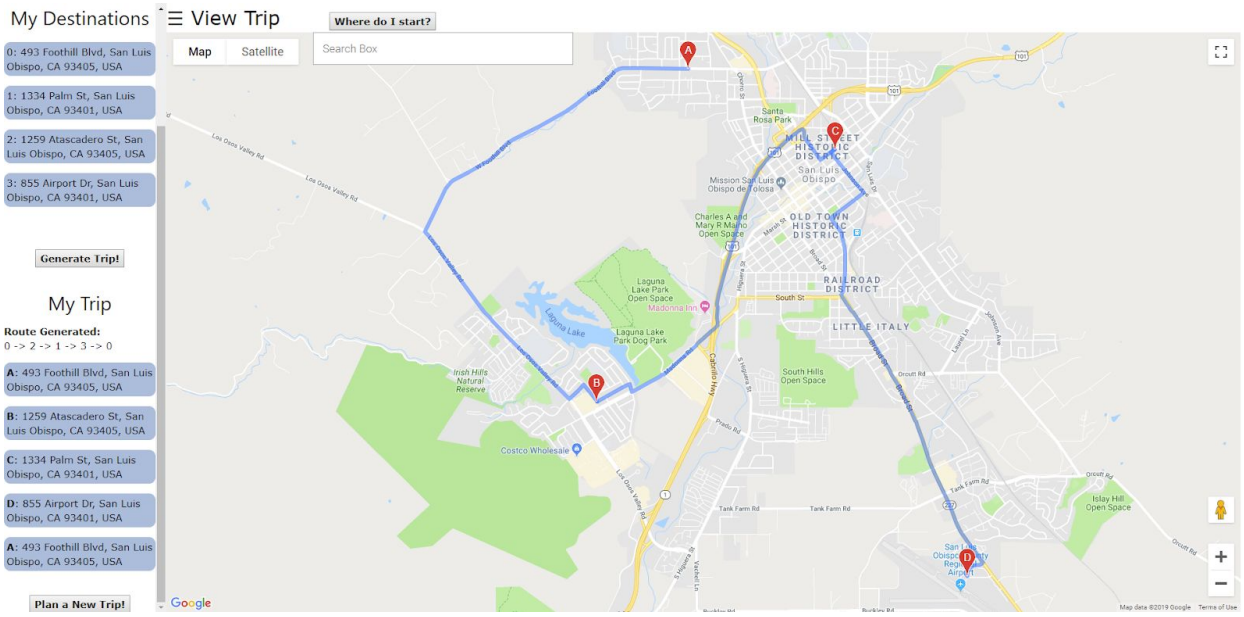


First view upon visiting the site.

Front End

As of the writing of this report, DNS has not been set up for the website, so users must visit 35.233.151.12 to visit the page. Upon accessing the page, users are greeted with a Google Map centered on San Luis Obispo. The "Where do I start?" button can be clicked to view information on how to use the website. Users can click on the map to add a single marker at a time or use the search box to find a specific address or set of locations based on a keyword. Once a marker has been placed, clicking on it will display an Info Window with the location's latitude and longitude and its geocoded address retrieved from the Geocoding API. If a user clicks on a marker that was placed as the result of a search, the Info Window displays information retrieved from the Places API. All Info Windows also contain a button that allows a user to add a location to the My Destinations list on the left-hand panel. Up to ten locations can be added to this list. At any time, the user can click on the "Clear Destinations" button to remove all markers from the map and return it to its initial state.

Once the user is satisfied with their list of destinations, they can click on the "Generate Trip" button to have the map display the optimal route between the points. In addition to the drawn route, a new section will appear on the left-hand panel titled "My Trip". This list is identical to the destinations list, but the locations are listed in the order that is shown by the route on the map. The first location is also treated as the last location, so it is listed twice. Below this list is also a new button titled "Plan a New Trip!" which will clear all markers and routes and return the map to its initial state. A sample search result can be seen below.



Results of a sample trip using four locations.

Back End

The bulk of this project is written in JavaScript. When a user adds a marker to the list of destinations, the marker's latitude and longitude are added into an array of latLng objects. This array is passed into Google Maps' distance matrix service which then returns a matrix containing the distances between each pair of points. This matrix is represented as an array or arrays in JavaScript. To actually obtain the optimal path from this matrix, the matrix must be sent to the C# TSP solver via .NET and the signalR library. The solver calculates the path and returns it as a string of indices separated by arrows. Each index corresponds to a spot in the array of user locations. Once the path string is received on the JavaScript end, the string is parsed and each index is converted into an integer which is then placed into an array of indices. This array is used to access the unsorted array of user locations and list them in the correct order on the side panel. Since the array of locations holds them in the form of a latLng object, the object must be converted into a human-readable address string using the Geocoding API.

The project is built using Microsoft's .NET framework to link all the necessary files into a single csproj file which can be run as an ASP.NET application. In this manner, we are able to host the project locally so that we can open it in browser. During testing, the project was hosted on http://localhost:5000. In order to make the project available to the public, it was moved over to a Google VM instance running Ubuntu 16.04 with 6.75GB of memory. Once on this VM, the project was configured to run on http://localhost:5321. In order to access it from a browser that is not on the machine, an Nginx web server was set up to function as a reverse proxy server. The server will listen for any request made to port 80 (the HTTP port) and forward that request to an internal location. In this case, that internal location was http://localhost:5321. This means that

once the VM instance is running, any user can access the application by navigating to the VM's external IP address at http://35.233.151.12.

Results

Ultimately, our project met the core functionality requirements which we had set at the beginning of our senior project. However, where our project does give the user the correct output and is not easily breakable (throws any uncaught errors), there are definitely still some UI elements missing from the web page to help guide users who have never used our web application before. To test our use-case theory, we had a few people from Cal Poly test our project to see if they also had similar thoughts. We had a set of people from varying technological backgrounds test our website with minimal instruction. They were only told what the website does, the limitations, and very basic instructions such as how to add a marker to the map. Anything else was left up to the user to discover. After interacting with the website in various ways, we had each user answer the questions below. They were also told to leave any additional comments they might have. The results from the survey can be seen further below.

Q1: When you first visited the page, how would you rate (1-10) the overall look of the website?

1 (I hate it) → 5 (It's okay) → 10 (I love it)

Q2: How intuitive was the app to use without any instruction?

1 (very confusing) → 5 (had to do some thinking) → 10 (very intuitive)

Q3: How satisfied were you with the results of the generated trip?

1(very unsatisfied) → 5(could be better) -> 10 (very satisfied ;))

Q4: How would you rate the design of our logo?

1 (It needs a lot of work) → 5 (Looks good enough) → 10 (Can't be more perfect)

Q5: How easy is it to interpret the results that are displayed once a trip is generated?

1 (not easy) -> 5 (could use more explaining) -> 10 (very easy)

Q6: Would you rely on this service to generate an optimal trip in another country? **(y/n)**

Q7: Have you ever found yourself struggling to plan a day trip with multiple destinations in the past? **(y/n)**

Q8: Have you used an app similar to this one? (Besides Google Maps)

Please list the name of the app(s) below or put "n/a"

Q9: Would you see yourself using this website when planning a trip? **(y/n)**

Q10: Would you recommend this website to any friends or family? **(y/n)**

Figure 1. Survey Questions 1-10

Subject #	Age	Major	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Subject 1	23	Business, IS	7	7	8	10	9	Y	Y	No	Y	Y
Subject 2	20	Business, F/A	8	6	7	8	8	y	y	Yes, Waze	y	Y
Subject 3	22	Mechanical Engineering	7	8	8	8	7	Y	Y	N/A	Y	Y
Subject 4	21	Chemistry	7	9	1	8	10	N	Y	Yes. waze	N	N
Subject 5	34	Manufacturing Engineering	6	7	5	6	8	N	Y	No	Y	Y

Table 1. Survey Results - Answers to Questions 1-10

Subject #	Additional Comments
Subject 1	● I think there needs to be clearer instructions for the users. Maybe a step by step guide ● Make the formatting cleaner, I would maybe reduce the amount of exclamation points to be more professional ● Maybe add a favorites list, to add favorite destinations instead of having to select a location everytime and selecting the "add to list" button ● For future upgrades, maybe implement actual directions as well within the app so users don't have to exit out of your application and go to google maps to find directions
Subject 2	● Possibly change the markers to be more prominent, show up better among other locations ● Sometimes if your location is too specific it seems to not place a marker ● Maybe have it tell you how many miles the trip will be or a time estimate
Subject 3	● I thought it was easy to use in the beginning but when I tried to click on locations (rather than looking them up) to add them to my trip, it would not let me ● No specific directions to go to the specified destinations (which means I have to use another external app if I were to use this feature for directions)

	● The trip summary was nice but I would have liked it if there has been an estimated time for the overall trip (and even maybe from one destination to another if I clicked on one section of the trip) ● As mentioned above, the markers could be more prominent (maybe a different color than the dark red since that's the same color Google Maps use to pinpoint a location). I didn't have any trouble finding the marker but I could see how it might be difficult for others
Subject 4	● Great app, overall design of page is a little basic but also straight to the point and shows the important relevant information, such as my trips on the left ● Does not let me remove or change trip unless restarting which is very inconvenient. Also it gave me a trip that made me go across country back and forth twice which definitely would not be the fastest trip possible ● Very user friendly and easy to use with limited instructions. ● Would have liked if it told me the expected time or distance of my trip ● Marker was very small and hard to stop sometimes when my location was in a large area like downtown San Francisco
Subject 5	● It would be nice to include some type of tutorial/explanation of the app ● Have the balloon automatically popup on a location to add it to the list/add button close to the search bar

Table 2. Survey Results - Additional Comments

Evaluation

Future Changes from feedback

A few things became apparent when we had people test our website. First, as we had guessed before the survey, it seemed that instructions on how to operate the website were not very clear and that the app wasn't entirely intuitive upon first glance. Most people did not know that they had to click on the marker in order for the button to add to the list to appear. Even though this is still a work in progress that can be expanded upon in the future, it seemed that people were expecting more features to be available. The most common complaint was that the map would display the route, but would not show any directions for the route. People found it inconvenient that they would have to visit an external site to obtain directions. This issue was made even more apparent by the fact that the path would not always appear to be the most optimal, especially for trips where locations were separated by great distances. Another requested feature was the ability to save specific locations or trips for future reference. The appearance of some of the markers was also subject to criticism, since some of them were too small to easily notice on the map. Despite these concerns, most people were satisfied with the results and expressed interest in continued development and said that they would be willing to use it if it was more refined.

Coding Challenges

One of the biggest challenges was learning a brand new framework (C#/ASP.NET) and implementing a successful data transfer between the server and client using SignalR. The reason for using a brand new framework is because Google's Traveling Salesman Problem Solver was not available in Javascript but had a C# implementation and we had known about the capabilities of using SignalR in ASP.NET to talk between server and client before working on this project. Using the ASP.NET single page application bootstrap seemed simple and intuitive at first as the .NET framework takes care of setting up the server for you with just the run of a command. Where .NET makes the project easy to run, it also encapsulates all of the difficult details of the code inside of a "black box," making it difficult to understand what's really happening between the client and server. Through a lot of research, we were able to gain a better understanding of where the server code (C#) and where the client code (Javascript and HTML) resided in the project. The next difficult task was figuring out how to successfully transfer data between the client frontend and server backend using .NET's SignalR library. With the use of one of Microsoft's examples using SignalR (the bootstrap project was called "SignalRChat"), setting up and establishing the connection was trivial. However, figuring out how to send the client's 2-D array distance matrix from the client-side to the C# TSP Solver on the server was the more difficult task. We first tried sending the distance matrix as a JSON object and then using C#'s built-in methods for parsing JSON objects. Through much trial and error, we determined that parsing the distance matrix as a JSON object was not going to work. By default, SignalR will send data over its channel as a JSON-formatted string. We finally decided to convert the distance matrix from a 2D array into a JSON string on the client, send that over to the server along with the distance matrix array's length, and then manually parse the string on the server using the string.split() method. Where this method may not have been the cleanest way to implement data passing between server and client, we were able to successfully pass parsed distance matrix into the C# TSP Solver and then send the result of the solver back to the client as a string.

Another challenge that was faced towards the end of the project was how to make the application available for anyone to access. It was not immediately apparent that an external web server was needed to host the application. At first glance, it seemed that the .NET framework was already running its own web server, so we assumed that the issue was with incorrect configuration of the settings file. After some digging around, it became clear that we would need another service to host the application. Reading through the .NET documentation revealed that apache and nginx were two services that were commonly used to host ASP.NET applications. Nginx was chosen for no particular reason and an online tutorial was followed in order to get the server running.

Conclusion

While our web application “Don’t Trip” may not be the most intuitive for a new user to figure out, given a list of destinations, it successfully outputs the optimal route for the user. The feedback we received in our survey will be used to brush up the UI of the website and also make the learning process while using our service much easier. In addition to the fixes that will be made to make the user experience more enjoyable, there are also a lot of extra features which we did not have the time to implement. One of the biggest features which we were hoping to have finished is a “save” feature for users to save their generated optimal trips to reference later. This would be very useful as users would not have to stay on our website the whole day just to view their optimal trip but instead would be able to revisit the site at their convenience and view their saved optimal trips. Another big feature that would be beneficial to the user is choosing an optimal route that takes into account wait times at restaurants or attractions, peak times of when the chosen destinations are busiest and other factors that relate to specific types of destinations. With this feature, users would have the option to generate their optimal trip based solely off of distance (default functionality) or can check off factors which they want the app to also account for. Aside from the extra features of our service, there is still a lot of work that needs to be done to the UI to give consumers the ability to use this website on their own. Future development will be made on this project to try and bring it up to a fully functional app so that eventually, travellers and even local residents across the world will no longer have to stress over deciding what to visit when.