Ronney Sanchez

COMP4620 GUI Programming II

5/1/19

Final Report

Project Overview

Throughout the semester, we have created an app called "Quickster" that would take a current address and prompt the user for multiple numbers of addresses that they want to visit. The app would calculate the most efficient route to visit each address. We have done some research about other applications that exist similar to the application that we wanted to implement. However, in those applications, the user only needs to use one destination at a time to visit with some stops features. If the user really wanted to visit multiple addresses, the they have to first enter their first address that they want to visit, use the stop feature, enter the second address to visit, use the stop feature again, enter the third address, and so on. Even though there is a stop feature, the user needs to type the addresses again which results in tedious typing on the app and the route is not efficient. Therefore, the user still needs to figure out which route is efficient to go. With our app called "Quickster" we have the ability for the user to enter all the address at the start. When the user enters the current address and all the addresses that they want to visit, the web app will calculate and permute the most efficient route. In other apps that exist out there, if you have a large list of addresses to visit and once you visit those addresses, you lose those data. In our app, we planned to have a tracker that lists each address that must be visited. The user would then check those addresses off and continue down the address list as they travel along. The disadvantage of our app is that the user must type their current address and type all the addresses that they must visit. It is a lot of typing, but the advantage is that it is only need to be done once in the beginning rather than typing continuously after each address visited down the list. To make the app more convenient to the user we would like to eliminate the time needed to enter any data by pulling the address from a file saved from the user, but since we ran out of time, we had to do the implementation based on the user's address typing.

Related Work

Google Maps:

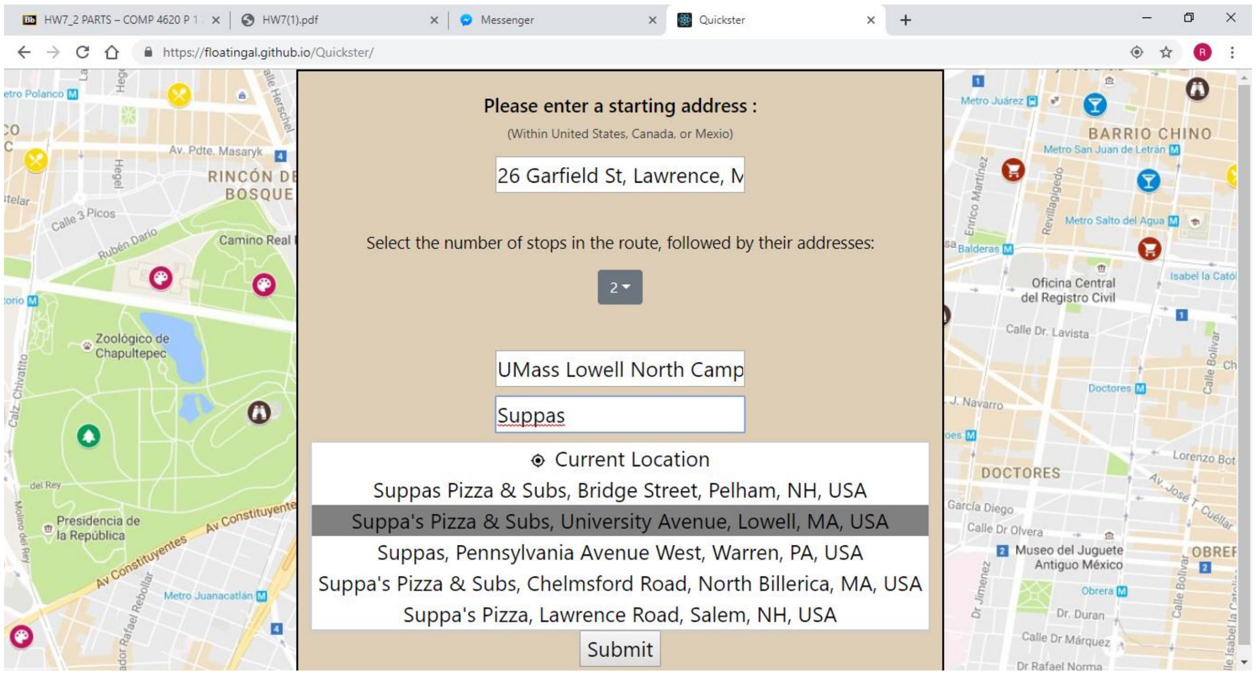
Google Maps give the direction to users from one point to another and it could add as many locations as possible. The problem is that Google Maps picks its own route to the destination and does not account for the fastest route regardless of traffic areas.

Waze:

Waze will give directions to users. However, it only allows just one address at a time therefore finding the best possible route would not be applicable. Waze is only useful on mobile phones but not on computers.

System Specification

Our app is a navigation suggestion application that will work with addresses within Canada, United States of America and Mexico including features address suggestion, a drop down menu that will dynamically render input boxes and uses a permutation function that will give the best possible outcome of the route to the user.


HW7_2 PARTS – COMP 4620 P 1
HW7(1).pdf
Messenger
Quickster
https://floatingal.github.io/Quickster/

Please enter a starting address :

(Within United States, Canada, or Mexio)

26 Garfield St, Lawrence, M

Select the number of stops in the route, followed by their addresses:

2

UMass Lowell North Camp

Suppas

Current Location

Suppas Pizza & Subs, Bridge Street, Pelham, NH, USA

Suppa's Pizza & Subs, University Avenue, Lowell, MA, USA

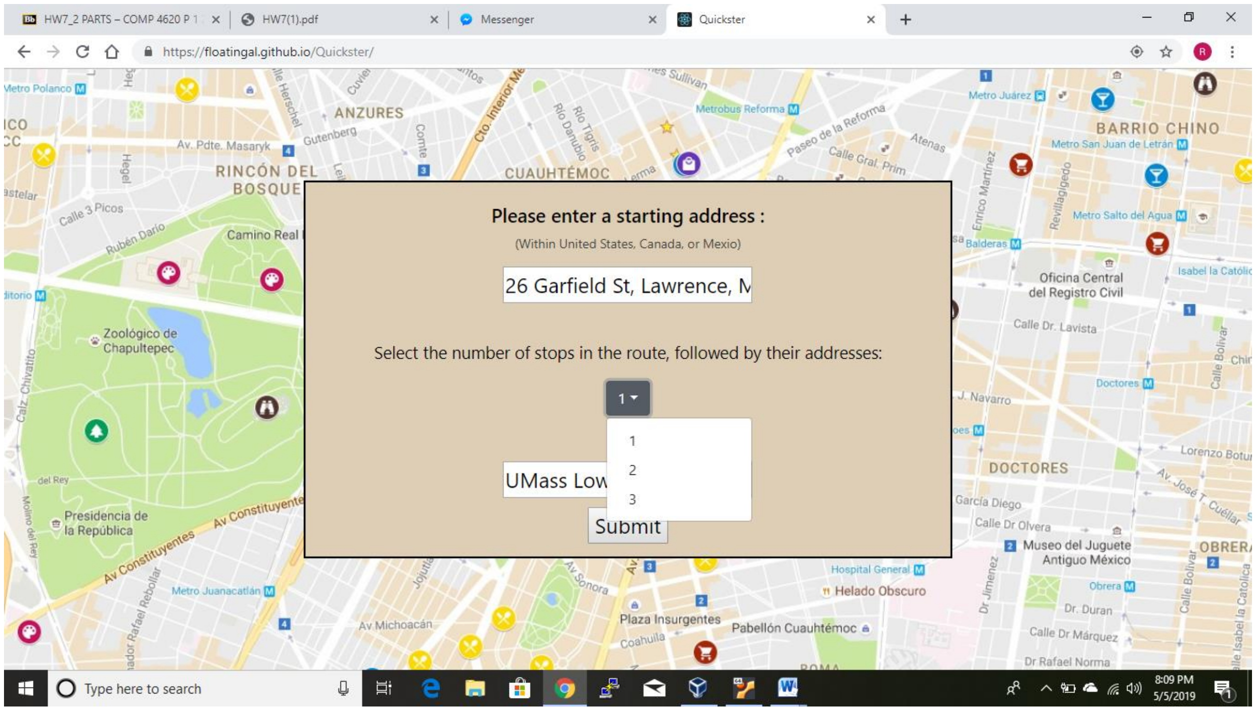
Suppas, Pennsylvania Avenue West, Warren, PA, USA

Suppa's Pizza & Subs, Chelmsford Road, North Billerica, MA, USA

Suppa's Pizza, Lawrence Road, Salem, NH, USA

Submit

As you can see, as I start typing addresses down, it starts rendering for addresses related to what I currently typed using the autocomplete function from the Google Maps API.


HW7_2 PARTS – COMP 4620 P 1
HW7(1).pdf
Messenger
Quickster
https://floatingal.github.io/Quickster/
Type here to search
8:09 PM
5/5/2019

Please enter a starting address :

(Within United States, Canada, or Mexio)

26 Garfield St, Lawrence, M

Select the number of stops in the route, followed by their addresses:

1

1

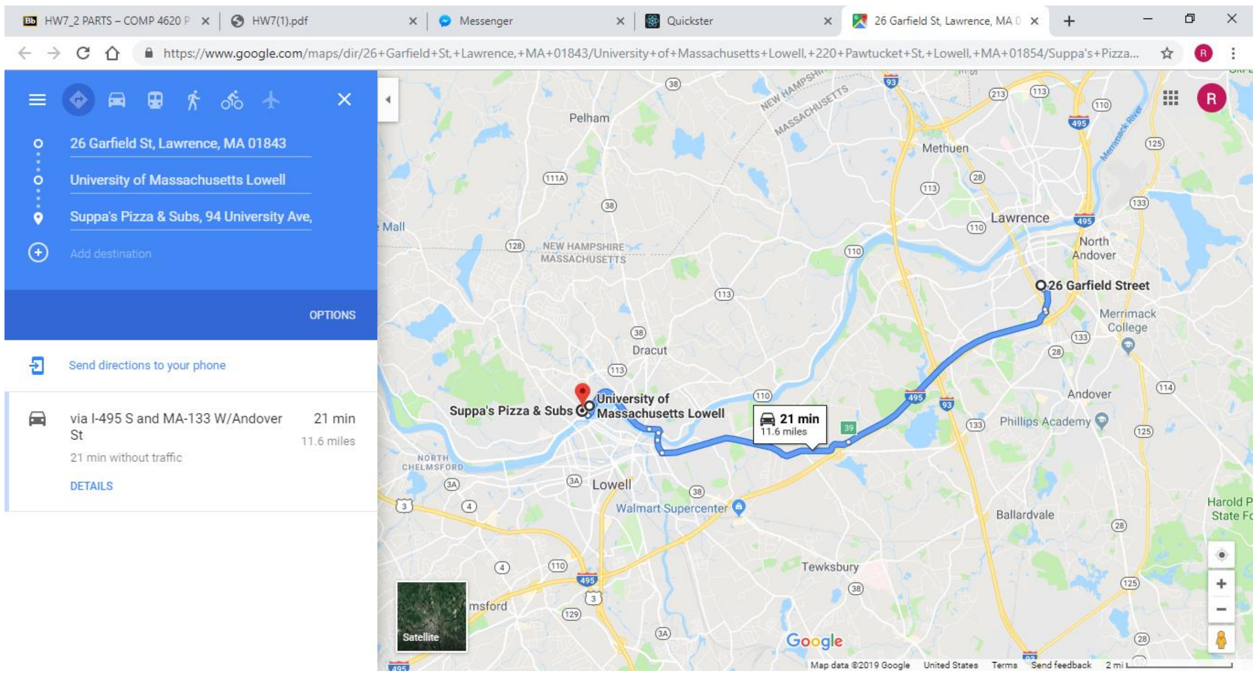
2

3

UMass Low

Submit

We used a dropdown menu to request number of addresses planned to visit. We limited our requests to three because as number of requests increases, the longer the computer takes to do the permutations to calculate the fastest route in which it could end up taking forever which the computer could get hung up.


26 Garfield St, Lawrence, MA 01843

University of Massachusetts Lowell

Suppa's Pizza & Subs, 94 University Ave,

Add destination

OPTIONS

Send directions to your phone

via I-495 S and MA-133 W/Andover St — 21 min — 11.6 miles

21 min without traffic

DETAILS

Once I hit the submit button, the app starts doing the permutation to find the fastest route to the requested addresses. This opens a new tab to Google Maps.

Project Design

Based on our project design, we have used Google Maps to track the most efficient route to our destinations. We used the Google Maps API to calculate the time between two places in real time. We also used JSON to do the permutations of sorting the list of address in order from closest to farthest because it did not matter for the user whether to put the list of addresses in order, we assumed that our application will do it for the user for which we have accomplished. We have limited of number of addresses to three because as our number of addresses increase, the processing time of our application becomes exponential. We have used the Autocomplete from the Google Maps API to allow our users for efficient input of addresses and cuts time off from typing the full string of addresses. Also for our application, we used React because this Javascript library allows us to create reusable UI components. React is also used for handling view layer for web and mobile apps in which we tested our application on the computer and in each of our phones. Our applications still remains unaffected in which based on any screen size of any device, our web remains in place with the inputs and buttons. We basically used React components to render responsive interfaces. For our CSS design, we used a Google Map image as the background image of our application as a demonstration for our use of Google Maps. A lot of students asked us why we chose a map of Mexico as our background image and our answer is because our app is not only useful for our county, but also Canada and Mexico as well. Basically the regional countries in our border are also useful for this app. Since we are using an image for our background, we decided to cover our text inputs with a text box so that it does not correlate or overlap with any text from the background image. We decide to use a light tan background color for our text box because the map view of Google Maps provides a good sense of matching to that color and also we are using a black font color. Therefore, our text must be visible to the user and it should give contrast to the background color of the text box. We also have our input box centered in the screen and provided a solid line border around the box for a touch of abstraction for our app.

Team Work

Throughout the semester, the project was divided into four labor parts. Albara Mehene’s part was to do some research about React and how it is use. He was also in charge of installing the Javascript libraries for React and JSON files for the project. Each of these libraries was stored in a node modules folder to be used for the semester project. Albara did the skeleton part of the web page which is doing the form input layout, the submission button, and the dropdown menu. Chhoung Tiv’s part was to do the error handling of the user inputs for each address request box. It is a user validator where if the user doesn’t type anything, an error message will show up telling the user to type it out. He was also implementing the submit function that does the permutation of all the addresses and then open Google Maps in a new tab to find the best route to the destination. We used Javascript to do the math calculation along with the Google Map’s API for the calculation. We had to render the routes to Google Maps. Liam Reynold’s job was to do the permutation calculation of sorting the addresses in order to increasing distance, which is from closest to farthest. Given the list of addresses that the user inputs, we must sort them in order from increasing distance to give the fastest route to the user after each address visited. He wrote the duration function that takes two addresses as input and returns the time it would take to travel in seconds as an integer. He worked on doing the permutation to work asynchronously with the application. Last but not least, my job was to do the CSS styling of the web page. Based on our project design we decided to use a Google Map image as the background for our demonstration of the use of Google Maps. However, we are using input forms for the user to input list of addresses in each form rows so I decided to close our inputs including our text with a tan text box divider so that our text does not overlap with any text from the background image. I also centered the text box in the middle of the web page and placed a solid border line around the box for a touch of abstraction. However, I noticed that when more number of addresses is being requested, the box changes size, therefore I tried to give a fixed size of the text box. It did not work so then I left it alone. These were our four labors to our semester project.

User Testing

We have decided to test our application with other groups for any issues that others might have when it came to the Beta version with user testing. During the time we had for testing, each users were asked to enter a starting address and any combination of one to three addresses, with varying resulting total route duration times. The specifics of the instructions given to users are entering initial starting location using either an address or current location indicator, selecting a fixed number of addresses from the dropdown menu bar, and entering corresponding addresses within the specified bounds manually or using Google autocomplete functionality in which addresses will be stored using their named, string representation and addresses in this beta version must be within the United States. Each user must initiate the route calculation using the “Submit” interface button. The loading time for the route will vary depending on required permutations. Google maps will be opened in a separate tab with the optimal route passed in and display on their provided UI. The users could repeat the trial process on mobile platform if it is accessible to them. During the user testing time, we noticed a lot of the users were having issues with accessing the autofill feature. We found out that the problem was too many request calls being used for the whole form. This happened because for every instance, 3 fetch calls were being called and the system of our application took a while to respond to those fetches. So users had to enter in the address manually because with many users testing our application, the response to fetch the autocomplete gets hung up. Some users found navigating the phone was a bit difficult because of the CSS cutting off. The problem was on my end that I gave the text box a fixed size for the web. Therefore, when user tested this application on a mobile phone, the input form box got thrown off and looked very clumsy. We noticed it took some time for the google maps to load to another tab and users were confused if it was processing or not because of the response traffic to the server.

Many members in different groups took actions in our testing and their suggestions in our applications were recorded. Gordan from group 3 suggested that we notify users of default transit method, we should also use U.S. address disclaimer, avoid toll route functionality, and possible international route implementation. Jon from group 4 suggested that we set the max size of input window, improve working on addresses that does not appear using autocomplete, limit the possible address input, notify users of load time after submit occurs to combat long wait times, and improve the dropdown menu response to input addresses. Matt from group 4 suggested that we add additional CSS styling and display loading screen while API is fetching.

We decided to make suggestions to our application based on the feedback we got from other group members. We decided to remove the map in the background since it confused many users and add another map image making it more reasonable. We suggested adding more CSS to make the application appealing and add a loading indicator since the processing time isn't instant. Many user assumed the address suggestions worked with anywhere in the world so we plan to add a disclaimer at the top indicating which region is supported. The disclaimer informed the user what mode of transportation is supported and we implemented the application to work with Canada, USA and Mexico accordingly with the disclaimer. We tweaked up our application and made some necessary changes. Overall our application looks more appealing.

Future Work

Our application was done at this stage but we came up with some "What If" questions about our web page. There were some implementations that we failed to accomplish but could have done better next time to our future work. One thing that we could have done better was working more on the permutations because we currently limited our address request up to three. For every address request there are factorial numbers of ways (n!) the address could be sorted in order in increasing distance therefore when our number of address request go beyond 3, the computer starts to take a long time in doing the permutations of the addresses order. The computer ends up taking forever or just gets hung up due to no response. What we could have done better is coming up with an algorithm that could do permutations in the shortest amount of time and implement that so that we can start prompting for more than three requests. Another thing that we could have done better for our future work is changing the autofill feature so that the user can render the address differently. Our input form uses and autocomplete function where when the user starts typing an address, the autocomplete lists addresses that start with those letters using a dropdown menu. Our autofill feature is kind of off a little because when many users start using our application at the same time, the autofill feature either stops working or it takes a while to render the lists of addresses with the auto complete. We could work on that feature next time if we had more time to do it. The application can be better by rendering the map on our platform rather than opening a new tab to the Google Maps website. Lastly, we could have worked more on the styling with CSS. We noticed that based on our text box input, when we request for more than one address, the box changes size unnecessarily. We could have worked better in keeping the text box at a certain fixed size so that it does not change size in that instance. We tried using the keyword "FIXED" with CSS but when the application was being access on a mobile phone, the input text box started looking very clumsy so that was the incorrect approach in fixing with that issue. With a regular computer, the application looked fine but it got thrown off with a mobile phone. We will do better next time by looking more into the Bootstrap features because we think that Bootstrap has some features that could eliminate in the change of the text box size and also give a better layout with better designing in the input box. Also, it would be great if the current background could be improved by making it the starting address that the user input. Right now the background is static and does not do anything. These are the things that we could do better in our future work.