**Project Objective**

The objective of this project (tentatively named “Mercury” for the Roman God of speed) is to build a web-based travel route optimizer that takes a provided list of addresses and returns the optimum route of travel (along with other helpful information) back to the user. It will utilize the Google Maps API to get longitude/latitude coordinates and route distance/times as well as the Python Pulp linear optimizer to determine and generate the optimum travel route.

**Features**

* Utilizes Flask for creating a web application.
* Retrieves coordinates using Google Maps API based on user-input addresses.
* Calculates driving distance and duration between coordinates.
* Calculates estimated travel costs based on a user supplied vehicle MPG rating and a current average gas price estimate.
* Utilizes a linear programming model to determine the optimal route.
* Provides real-time optimization results, including total distance, total duration, an ordered destination path, turn-by-turn directions (I hope), and an embedded interactive Google map of the route.
* Gives users the ability to save a PDF of their travel report for future reference.

**Target Audience**

The application is designed for any travelers who need to generate optimized routes for multiple waypoints, such as delivery drivers, field service personnel, vacation planners or anyone requiring an efficient path between various locations. It is currently restricted to driving but this could potentially be expanded as Google supports various modes of transportation (walk, bike, bus, train, etc)

**Resources Used**

* Utilizes Flask for web application structure.
* Incorporates Google Maps API for geocoding, directions, and embedding maps.
* Applies linear programming using the PuLP library for optimization.
* The HTML interface will feature a user-friendly form with dynamic address input fields.
* CSS styling will enhance the visual presentation of the form and the displayed result.
* JavaScript will manage the dynamic form elements, initiate optimization, and update the UI.

**Planned Design Overview**

A diagram of a program

Description automatically generated

**Project Status**

I believe I am making reasonable progress on the project so far. The completed list of tasks includes the following:

1. I have generated a python program that accepts a user specified address and uses the Google Geocode API to provide the longitude and latitude of the location:

(https://maps.googleapis.com/maps/api/geocode/json?address={address}&key={GOOGLE\_API\_KEY}')

Menu:

1) Get Address Coordinates

2) Get Driving Distance

3) Get Driving Time

4) Exit

Enter your choice (1-4): 1

Enter an address: 610 Maryville University Dr, St. Louis, MO 63141

Coordinates for 610 Maryville University Dr, St. Louis, MO 63141: (38.6447014, -90.50582659999999)

1. Furthermore, the same program will calculate the distance in miles (or time in hours) between an origin set of coordinates and a destination set of coordinates using the Google Maps directions API. As an example, it calculates the distance from home to the Maryville school library as follows:

(https://maps.googleapis.com/maps/api/directions/json?origin={origin[0]},{origin[1]}&destination={destination[0]},{destination[1]}&key={GOOGLE\_API\_KEY}")

Menu:

1) Get Address Coordinates

2) Get Driving Distance

3) Get Driving Time

4) Exit

Enter your choice (1-4): 2

Enter origin coordinates (lat,lng): 38.6447014, -90.50582659999999

Enter destination coordinates (lat,lng): 38.5890988, -90.63621789999999

Driving distance between coordinates ['38.6447014', ' -90.50582659999999'] and ['38.5890988', ' -90.63621789999999'] is 11.479829225 miles

1. With these two pieces working, I then focused on sending a list of unordered destination pairs to the Pulp linear optimizer so it could calculate routes that minimize either total driving time or total driving distance. For example, starting from home, today I might wish to visit school, Busch Stadium, Sam’s Club in South County and then return back home. What is the route that minimizes the miles driven or drive time needed to accomplish this? The starting address is entered first but the remaining destination addresses can be entered in any order as follows:

Solve Menu:

Enter Starting Address 1 (or 'done' to finish): 16866 Hickory Crest Drive, Wildwood MO 63011

Enter Address 2 (or 'done' to finish): 700 Clark Ave, St. Louis, MO 63102

Enter Address 3 (or 'done' to finish): 610 Maryville University Dr, St. Louis, MO 63141

Enter Address 4 (or 'done' to finish): 4512 Lemay Ferry Rd, St. Louis, MO 63129

Enter Address 5 (or 'done' to finish): done

*Passing this to the Solver……….*

Welcome to the CBC MILP Solver

Version: 2.10.3

Build Date: Dec 15 2019

Problem MODEL has 14 rows, 15 columns and 42 elements

Coin0008I MODEL read with 0 errors

Option for timeMode changed from cpu to elapsed

Continuous objective value is 60.5942 - 0.00 seconds

Cgl0003I 0 fixed, 0 tightened bounds, 6 strengthened rows, 0 substitutions

Cgl0004I processed model has 14 rows, 15 columns (15 integer (12 of which binary)) and 48 elements

Result - Optimal solution found

Objective value: 69.60473668

Enumerated nodes: 0

Total iterations: 0

Time (CPU seconds): 0.02

Time (Wallclock seconds): 0.02

Option for printingOptions changed from normal to all

Total time (CPU seconds): 0.03 (Wallclock seconds): 0.03

## **Optimization Result**

Status: Optimal

Total Driving Distance: 69.6 miles

Ordered Path:

 Waypoint 1: From ***16866 Hickory Crest Dr, Ballwin, MO 63011*** to ***4512 Lemay Ferry Rd, St. Louis, MO 63129***

 Waypoint 2: From ***4512 Lemay Ferry Rd, St. Louis, MO 63129*** to ***700 Clark Ave, St. Louis, MO 63102***

 Waypoint 3: From ***700 Clark Ave, St. Louis, MO 63102*** to ***610 Maryville University Dr, St. Louis, MO 63141***

 Waypoint 4: From ***610 Maryville University Dr, St. Louis, MO 63141***to ***16866 Hickory Crest Dr, Ballwin, MO 63011***

1. The final piece I’ve worked on so far is building an interactive embedded Google Map for the generated optimum route. For the optimum travel route produced in step 3) above, the program will build a url based Google maps string that produces a route map that will look something like this:

A map with a route

Description automatically generated

The list of items I’ll be working on next to further build out the design include:

* Understand the PuLP optimizer better to ensure optimum results.
* Design and code the front end (HTML or Angular?)
* Integrate the flask server to accept and return data from the front-end.
* Design and code the trip estimated cost calculation.
* Design and code turn-by-turn directions for each leg of the route.
* Configure the ability to save a PDF of a route for future reference.