Assignment 4 Write-Up

Members: Yuxuan “Jason” Wang, Aryan Jain, Kathy Martinez, Wendy T. Posada

Project Overview

In this assignment, we developed a website accessing Google and MBTA APIs directly. This website allows someone to input a specific address or a landmark into an input field. The website will then return the closest MBTA stop and the distance from the given place to that stop. The application is able to account for any included symbols in the input field, e.g. ‘Prudential-Tower, Boston’. While generating the Google API location search query URL, the special characters will automatically be omitted to obtain an accurate URL. Certain landmarks, such as ‘AMC Theatre’, will return multiple results in the Google API, so the user will have to specify location of boston by appending ‘Boston’ at the end of it. We have added a layer of verification for this. If there are multiple results or there are no results, the user will be notified to correct their entry along with a suggestion. The functionality of the website is as follows:

User will input location, the Google API will verify if it can first find the location. If true, the MBTA API will find a station that is 1 mile away from user location. It will also output the distance of the stop from the entered location. The distance has been cleaned up by rounding it to only two decimal places, rather than the default twelve decimal places because such precision is not required by the user. If there are no MBTA stations within 1 mile of the user, then there is an easter egg to be discovered. Try entering ‘Babson College’ since it has no MBTA stops within 1 mile.

Project Reflection

The team process for this assignment was to evenly split the work amongst the four of us and to meet at least twice. We split the mbta_helper.py into seperate parts, so one person would work on one function, assuming the other functions were already working and returning the output mentioned in the docstrings. We began coding this way, but eventually we all integrated into working on every part together. Since everyone’s code had certain errors and each team member had their own competencies with various areas of programming, we decided it would be better to go through each function as a team rather than individually. This was much more efficient because debugging each other’s code took a lot of extra time when working separately.

For unit testing we wrote test code into a main() function. We tested each function separately by manually typing in the input expected in each function and printing each output separately. The reason we chose to use manual inputs instead of calling previous functions while testing is because we wanted to be certain that each function is performing its own task without errors before integrating them all into a program where they can call each other. When we were done with the coding and debugging, we tested the location retrieval by applying a location that will yield correct results and one which we knew should not return a proper result. However, one of the things that would have helped with implementation would have been better documentation of the limitation of the API search restrictions for an unregistered key access. As we were testing our code, we first received no output when we knew that there was a station near but because we were unaware of the “one mile” rule we wasted a bit of time trying to find error in our code, when indeed there was none. Simply changing the location to a location nearer to an MBTA stop, made everything work perfectly. To avoid this error by the user, we decided to include a validation step for locations that are too far from MBTA stops. Another limitation of the API is that it does not distinguish train and bus stations, so the user will not be able to know what type of station they are being directed to.