

# CSCI 4131 – Spring 2024

## Internet Programming Assignment 3

**Due Date: *Sunday, March 3<sup>rd</sup> at 11:59 pm***

**Late Submission Deadline: *Friday, March 8<sup>th</sup> at 11:59 pm***

### 1 Description

For this assignment, you will add Google Maps to the functionality you have developed on your Schedule and Form Web pages through homework 2. You will add a 3<sup>rd</sup> page to your website named: stockQuotes.html. The page will use a free stock market Application Programmer Interface (API) for JavaScript to obtain quotes on stock prices (see: the README file found at <https://github.com/wagenaarje/stocks.js/> for instructions on how to set up and use the API to obtain stock quotes. ) There are 11 pages in this write-up/requirements specification.

The objective of this assignment is to continue to develop your JavaScript skills and introduce you to different Application Programmer Interfaces: The free stocks API, and the Google Maps JavaScript API along with the Google Places JavaScript library, the Google Maps Directions Service, and geolocation.

While the Google Maps API will be introduced in subsequent lectures, developing and/or bolstering the ability to read API and library documentation and then use it to develop functionality is an essential skill for today's web developer (or any software developer). New libraries with new APIs are introduced, and existing libraries and their APIs are updated regularly. Teaching a specific API is counterproductive in today's rapidly changing environment. Instead, one of the objectives of this assignment is to require you to develop and/or bolster your skills to learn and use new libraries and APIs.

You will update your Schedule page as follows:

1. You will write code to dynamically mark the addresses of the places specified in your schedule web page on an embedded Google map.
2. You will develop additional functionality to search for specific places near your current location and display them on the map (*Optional, Bonus*)
3. You will also add functionality to calculate directions and display a route between your current location and destination on your Google map via various modes of transportation.
4. Add a Google map to your form page. Then, use the click on a point of interest (POI) capability to fill in the address field on your form.
5. You will add a new page named: **stockQuotes.html** to your website – and use the stocks.js API found at: <https://github.com/wagenaarje/stocks.js/> to obtain stock quotes and display the results in a Textarea element on your stockQuotes.html page. You will also update your server from homework 2 to return the stockQuotes.html page when your browser requests it, and update your Navigation bar on all the web pages on your website to enable navigation to the stockQuotes.html page.

## 2 Preparation/Reference

### **Conditions on using your google maps API key: READ CAREFULLY.**

We will provide you with a Google Maps API key via email (an announcement). ***Do not share the key with anyone - ever.*** Each student is expected to limit their use of the API key to, at most, 75 queries per day (***on average over 3 weeks – so a total of approximately 1500 queries per person - maximum*** ). Should the key be over-utilized, we will turn it off, and the entire class will have to wait for a new key.

You are free to use your own key, and can obtain one as specified in the directions given at the link: <https://developers.google.com/maps/documentation/javascript/get-api-key#:~:text=Go%20to%20the%20Google%20Maps%20Platform%20%3E%20Credentials%20page.&text=On%20the%20Credentials%20page%2C%20click,Click%20Close.>

The advantage of getting your own key is that you will be able to use it for your own web applications and it will be active after we turn the key for the assignment off. Google gives you a 200 dollar allowance – and for this assignment, even if you go WAY over 1500 total queries, you will use only a few dollars.

**We will disable the key after the late due date for the assignment until the grades are released, and then enable it for 3 days so you can evaluate your submission.**

**ALSO NOTE: The late submission period has been extended for this assignment However, it occurs over spring break – so there will be no Instructional staff available to answer questions on issues during that time. For this assignment only: 1 day late 5% deduction; 2 days late 10% deduction, and, after 2 days late 15% deduction through the late submission date.**

### 2.1 Google API Setup

Some setup is required to use the Maps API – you need to get your API key (see above). Google provides an excellent tutorial for setting up your map using the key, and you should complete the tutorial on how to do so. The tutorial can be found at the following link:

<https://developers.google.com/maps/documentation/javascript/tutorial>

### 2.2 Google Places JavaScript Library Reference

You can refer to the following link to obtain more information about the places library:

<https://developers.google.com/maps/documentation/javascript/places>

### 2.3 Google Maps Directions Service Reference

You can refer the following link to obtain more information about the direction service:

<https://developers.google.com/maps/documentation/javascript/directions>

### 2.4 Geolocation Reference

You can refer the following link to obtain more information about geolocation:

<https://developers.google.com/maps/documentation/javascript/geolocation>

2.6 Google Click on Points of Interest (POI) - used to fill address field on Form when points of interest are selected/clicked on the map next to it:

<https://developers.google.com/maps/documentation/javascript/examples/event-poi>

2.7 Documentation on how to get a key for and use the stock API can be found in the README file at the following link: <https://github.com/wagenaartje/stocks.js/>

### 3 Functionality

#### Update functionality on your Schedule page

1. Start by modifying the address column of the table from homework 2: include the complete address information in table entries for those events, e.g., 100 Union St. SE, Minneapolis. MN 55455. Add some physical events to the table if you have less than 5 physical events (with an actual address).
2. Embed a Google map under the table. (see figure 4.1)
3. The map should be centered on University coordinates (44.9727, -93.23540000000003) with a zoom level of 14 (or any zoom level that you find appropriate). University coordinates can also be obtained using the following link: <https://www.gps-coordinates.net/>
4. You should write JavaScript code to **dynamically** extract the names, times, and locations of your physical events from the DOM objects in your Schedule table. Your code should then place a custom marker on the map for each location extracted. The markers should display the name and time of the event and also display the complete address of this event (e.g. Room number 2-209 in Keller or Shepherd 383) upon being clicked. **Note, your JavaScript should not contain hard-coded latitude and longitude positions to do this (this is a requirement of this assignment)**. Specifically, create JavaScript to obtain a list of addresses from your physical events and use the geocoding or places library to obtain a latitude and longitude for each. Then, create a marker for each unique latitude and longitude. This can be achieved using an information window (see figures 4.2 and 4.3 below) – **this is a requirement**.
5. The next step is to insert an input area on the right of the map. The elements in this input area will require you to use ‘Google Places JavaScript library’ and ‘Google Maps Directions Service’. See figure 4.4 below.
6. **(NOTE, the first row of the input area, and the rest of the functionality specified in this item is OPTIONAL and can be completed for a BONUS)**. The first row of input area (created as per step 5 above) should enable a user to search for places, e.g., restaurants near the user’s current geolocation. It consists of:
  - A drop-down field that lists the category of places to search for. The default categories are restaurants, hospitals, parking, and supermarkets. You can replace the default categories with additional categories present at [https://developers.google.com/places/supported\\_types](https://developers.google.com/places/supported_types). You should ensure that at least a few places exist in the search results for each category you include
  - A input element that accepts the numeric radius around your current location in which the search results will be shown.
  - An input element- that allows users to enter the keywords of other places that they may search for. This textbox should be disabled by default. The drop down field specified in the first bullet in item 6 above should include an extra option named: “Other”. When “Other” is selected, the textbox is enabled. Try one of the following keywords when you are testing this - burger, bus, library, laundromat, etc.

- A button labelled ‘Search’: Upon clicking this button, your code finds all the nearby places within the specified radius using ‘Google Places JavaScript library’ and displays the results by placing a marker on the map for each of the found places. The old markers (e.g. the markers for your events location, and markers for any other category) should be cleared before placing new markers on the map. When clicking on a marker, the name and address of the location should be displayed.
  - Refer to figures 4.4 and 4.5 for an illustration of this functionality.
7. The second search component that should be added is the functionality to get directions from a user’s current location (as determined by HTML Geolocation – see [https://www.w3schools.com/html/html5\\_geolocation.asp](https://www.w3schools.com/html/html5_geolocation.asp)) to the destination they provide. The group of elements you should use to implement this functionality are as follows:
- A input element that allows the user to input the location of the destination. The source location will be the current location of the user’s computing device
  - and it can be obtained using geolocation.
  - A radio button group: Three modes of travel should be specified: DRIVING, WALKING, and TRANSIT. One of these must be selected at all times.
  - A button labelled ‘Go’: Upon clicking this button, the route between the current location and the destination should be displayed on the map. The displayed route will be based on the mode of travel selected by the user. The directions associated with this route should be displayed on a scrollable side panel floating to the left of the map. Directions involving higher number of instructions should be wrapped into this fixed dimension panel with the scrollable feature. The source for the directions is the user’s current location extracted using Google geolocation services. The source coordinates should not be hard-coded; they should be extracted dynamically using code (JavaScript). Make use of ‘Google Maps Directions Service’ for this functionality.
  - Refer to figures 4.6 and 4.7 below for an illustration of this functionality

## Update Functionality on your Forms page

- a.) Add a Google Map to the right of your form
- b.) Use Google Map’s autocomplete functionality to fill in the address field on the form from locations selected from the Google map to the right of your form AND/ OR autofill the address field with a click on a point of interest on the map. See Figures 4.8 and 4.9 below for an example of click-on POI functionality. See the code snippet on Page 8 to help you incorporate autofill functionality

## Add a new page to your website which you can use to obtain stock quotes

Stocks.js is an easy-to-use stock market API for JavaScript. For this part of the assignment, you will learn how to use – and then use, the stocks.js API to create a web page that utilizes the stocks.js API to fetch stock data based on user input. After the user input is obtained, the stock quote (that is, the price of the stock) will be displayed in a Textarea element

1. Please read the directions in the README file in the github directory:  
<https://github.com/wagenaarartje/stocks.js/> **carefully!!!!**
2. Create an HTML file named: **stockQuotes.html** containing an input element to get a stock’s “ticker symbol” from the user and a textarea element to display the data received in response to the request for

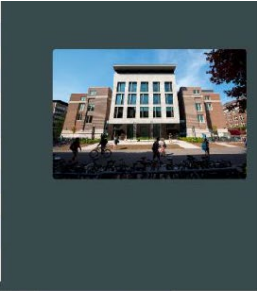
data on the stocks current price.

3. Get an API key from: <https://www.alphavantage.co/support/#api-key>
4. Write the JavaScript necessary to obtain a stock ticker symbol from the user, use the API in the file stocks.js to retrieve the stock price data, and then display the result in the textarea element.
5. Make sure the stocks.js file is referenced properly via a link in your stockQuotes.html page, your new html file and stock.js are in the proper directories, and that you add the code necessary to return them in your Python web server

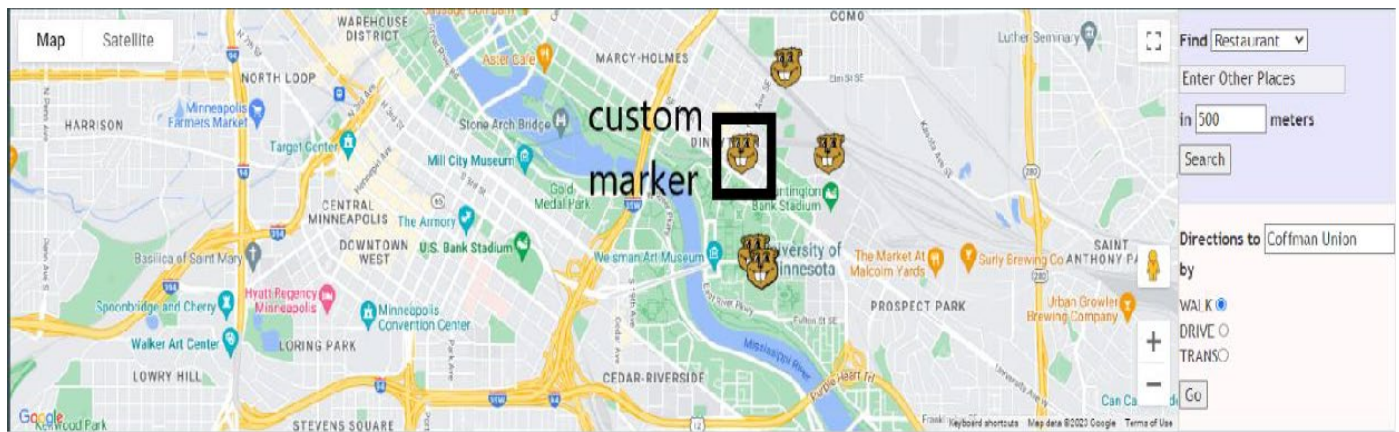
## 4 Screenshots

### 4.1 Table with a list of events, google map, and input area floating to the right of the map

Monday	Csd 4101 Lecture	11:15am - 11:30pm	110 Tate Hall	See Class Information	Csd 4121 Interaction
Monday	Csd 3081 Lecture	2:30pm - 3:00pm	302 Live Hall	Class Information	Csd 3081 Game Information
Tuesday	Csd 4101 Lecture	10:00pm - 11:00pm	110 Tate Hall	See Class Information on Canvas	Csd 4121 Info
Wednesday	Csd 4101 Lecture	11:15am - 11:30pm	105 Tate Hall	Class Information	Csd 4121 Info
Wednesday	Csd 3081 Lecture	2:30pm - 3:00pm	302 Live Hall	Class Information	Csd 3081 Information
Wednesday	Dr C's Virtual Office Hours	7:15pm - 8:15pm	Virtual (Zoom)	Zoom Meeting Info with Office Hours on Canvas	Class Information
Thursday	NBA Dog	7:30 - 9:00am	601 18th Ave SE Minneapolis, MN 55414	36121 270-4054	Dog club info
Thursday	Csd 1121 Office Hours	2:30pm - 3:00pm	322 Live Hall	See Class Information on Canvas	Csd 4121 Info
Friday	Csd 1081 Lecture	2:30pm - 3:00pm	302 Live Hall	Class Information	Csd 1081 Interaction
Saturday	Breakfast	12:00pm - 11:00am	413 14th Ave SE Minneapolis, MN 55414	36121 331-9991	Breakfast Spot
Sunday	Jog	12:00pm - 11:00am	1828 16th SE SE Minneapolis, MN 55445	None	Info on where to Jog

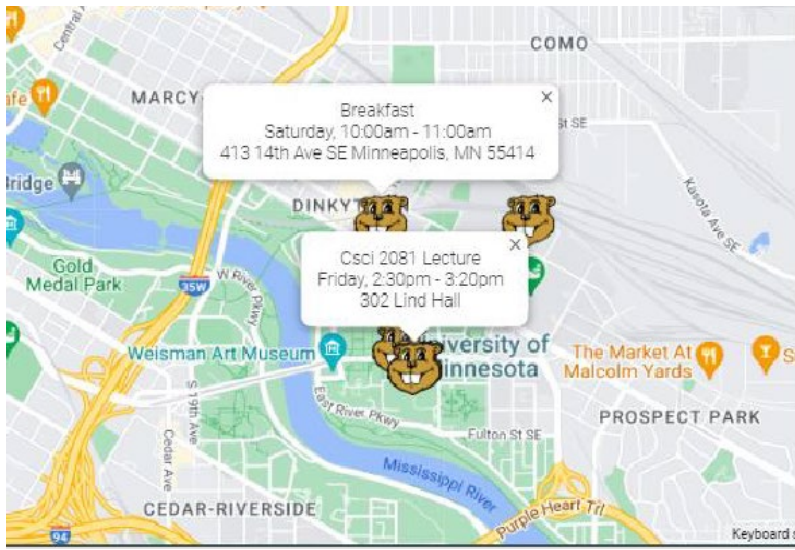


### 4.2 Markers of the physical events in the schedule on the Google map.

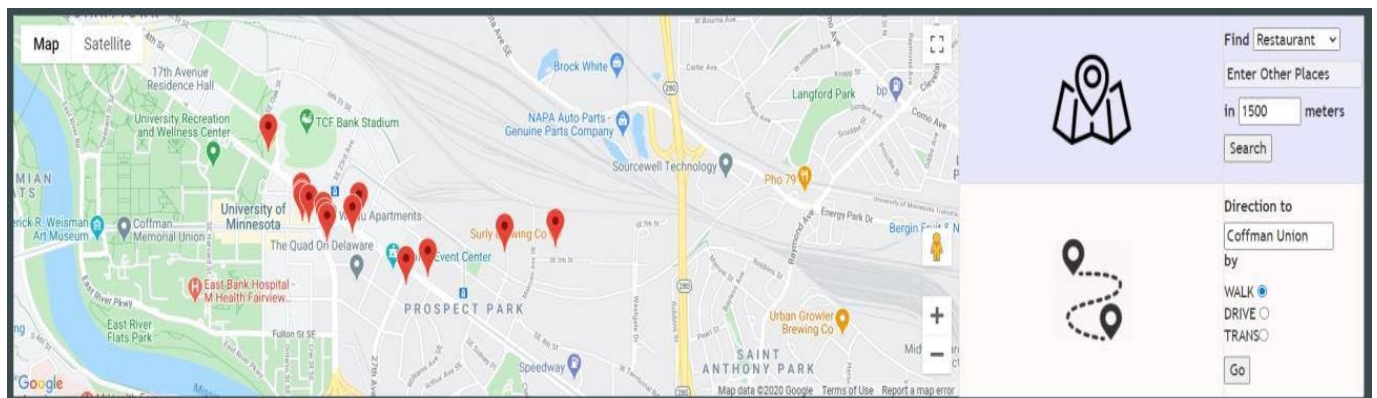




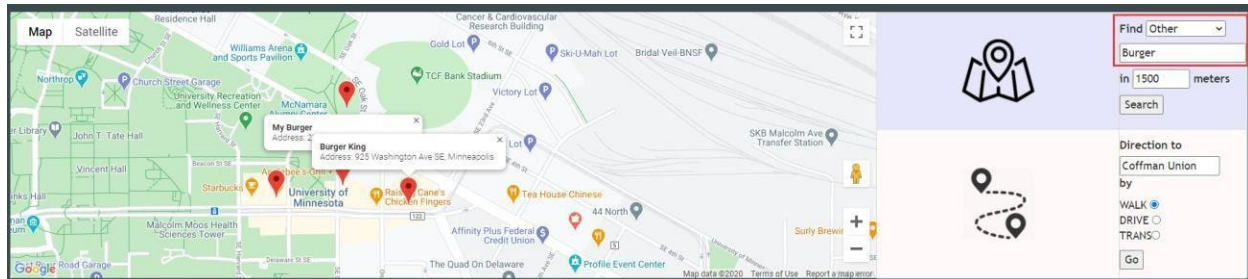
4.3 Marker shows the name, time, and address of a physical event on click  
(you should use JavaScript to extract that information from your schedule table; in the screenshot, two markers have been clicked)



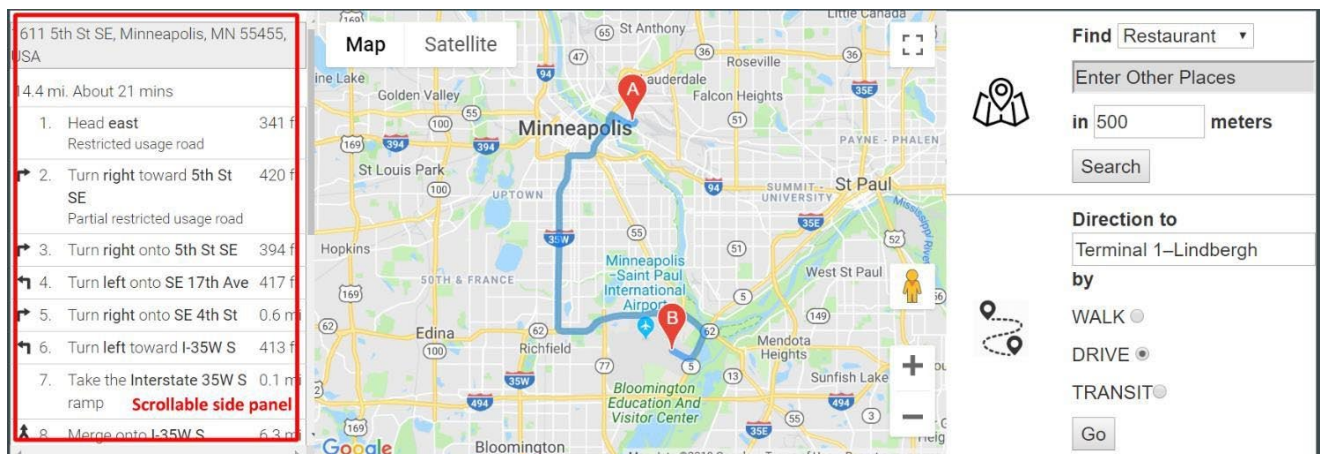
4.4 Map showing the result of nearby restaurants in a radius of 1500 meters (Note, this is optional, bonus functionality)



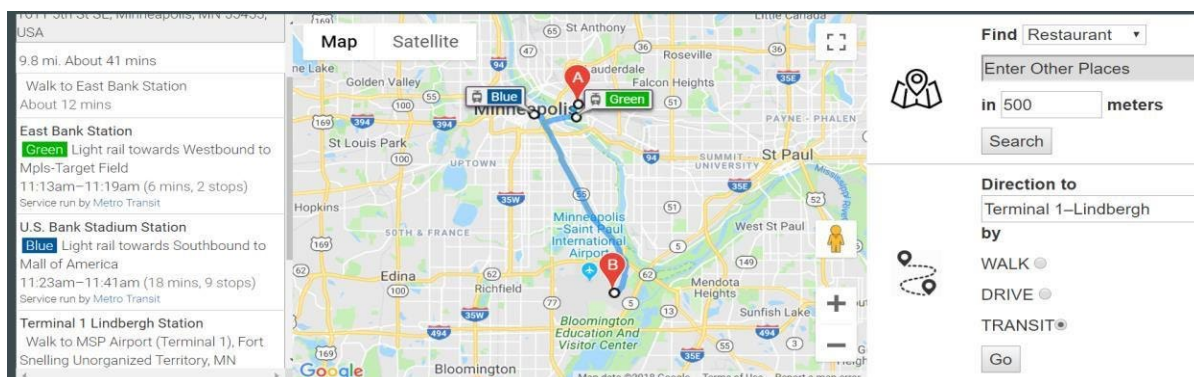
**4.5** When “Other” is selected in the dropdown list, user can enter the keyword to search for items related to places they are looking for, e.g., “burger”. The nearby places with the keyword in their names will show up when “search” button is clicked (See the results for “burger” below). **NOTE – this is optional, bonus functionality.**



**4.6** Map showing the DRIVING route between the current location and Terminal 1–Lindbergh

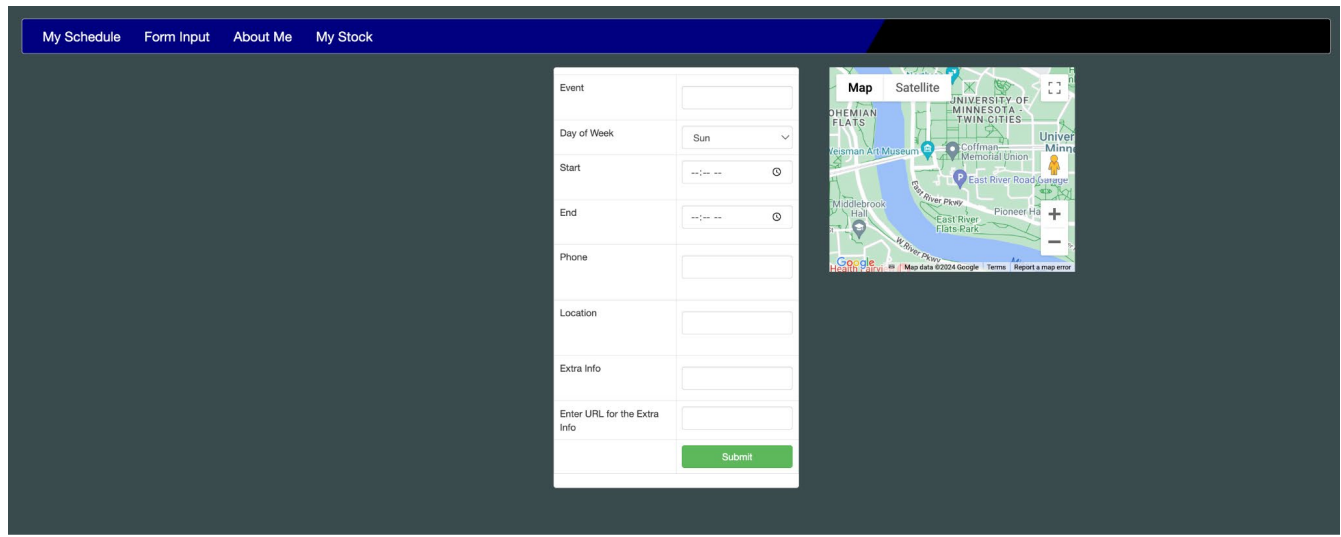


**4.7** Map showing the TRANSIT route between the current location and Terminal 1–Lindbergh



**Note:** you can use **HTML5 Geolocation** to set your current location – see the information at the following link: [https://www.w3schools.com/html/html5\\_geolocation.asp](https://www.w3schools.com/html/html5_geolocation.asp)

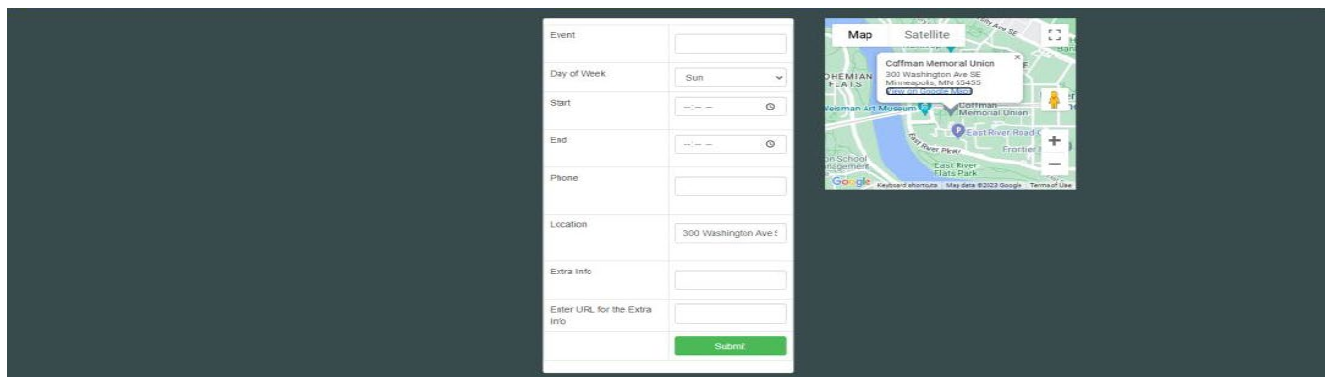
## 4.8 Add a Google Map to your Form Page



The screenshot shows a web form on a dark background. At the top is a blue navigation bar with links: "My Schedule", "Form Input", "About Me", and "My Stock". The form itself is white and contains several input fields: "Event" (text), "Day of Week" (dropdown menu showing "Sun"), "Start" (datetime), "End" (datetime), "Phone" (text), "Location" (text), "Extra Info" (text), and "Enter URL for the Extra Info" (text). A green "Submit" button is at the bottom right of the form. To the right of the form is a Google Map showing a street view of Minneapolis, MN, with a red location pin.

4.9 When a place is selected (“clicked”) on the Map, the “Location” field on your form should be populated automatically with the address of the place you selected for click on Places of Interest (POI) functionality.

(For Example, 300 Washington Ave SE Minneapolis, MN, 55455, as shown in the screenshot below)



This screenshot is similar to the one above, but the "Location" field is now populated with the text "300 Washington Ave SE". The Google Map to the right shows a red location pin at the same spot, and a small information window is visible over the map showing the address "300 Washington Ave SE Minneapolis, MN 55455".

**The “snippet” of HTML and JavaScript below can be used to help incorporate Google Maps Auto-complete functionality into the address text field on your form (Note: this is not required)**

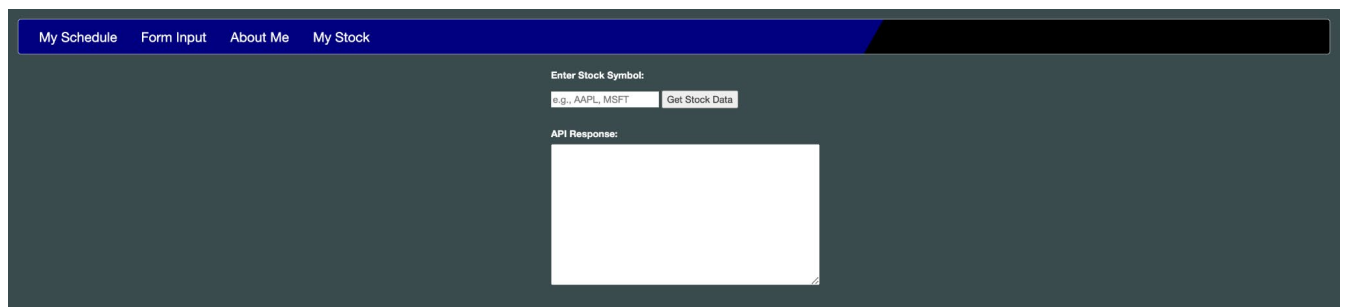
```
<input id="addressTextField" type="text" size="50">
  <script>
    function init() {
      var input =      I
        document.getElementById('addressTextField');
      var autocomplete =
        new google.maps.places.Autocomplete(input);
    }
  </script>
```



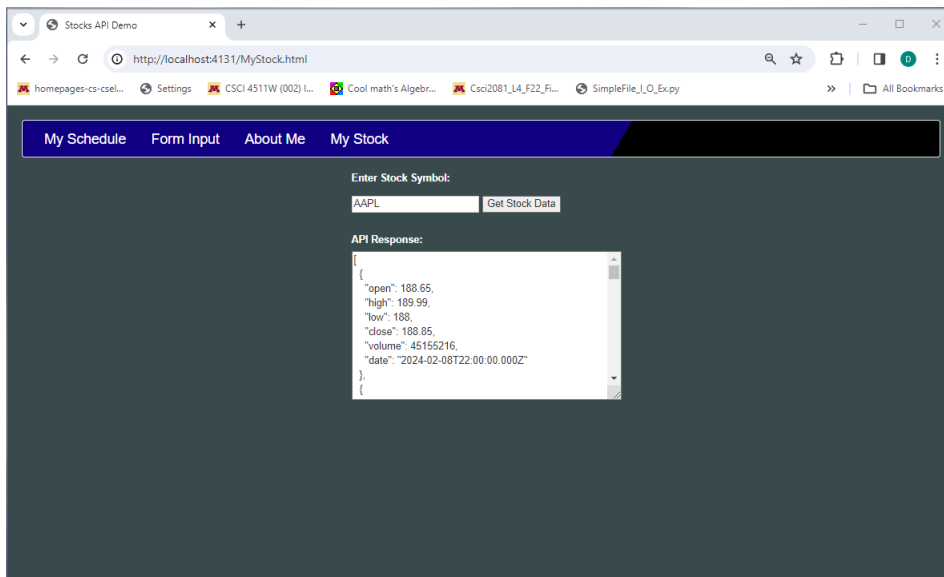
For “click on POI functionality”, you will need to add the JavaScript API (with your key) to your forms page to include another Google map on the page. See the example at the link specified in section 2.6 on page two of this document:

```
<script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=YOUR-GOOGLE-
MAPS-KEY=places&callback=initMap"></script>
```

4.10 Create a stockQuotes.html page that includes a user input text object and a textarea object.



4.11 Example of response when stock ticker symbol for Apple Computer is entered in the Input Text Object, the Get Stock Quote Button is clicked, and the response is displayed in the TextArea Object



## 5 Submission Instructions

1. For this assignment, you are expected to store the HTML, CSS, JavaScript, image files, and server.py file in their respective directory / folder as specified in Homework 2 (each of which are subdirectories to the static folder).
2. Your submission should include a **minimum** of 7 HTML, CSS, JavaScript files (5 HTML, and 1 each for JavaScript and CSS), plus one server.py file, and all the images used by your web pages. You can include more than one JavaScript and/or CSS file. However, it is required that your CSS is separated from HTML (that is, in an external file) - and you should include the majority (not necessarily all) of your JavaScript in an external file or files as well.
3. These files should all reside in the directory yourx500id\_hw3
4. Submit a zip file of all your files - and submit the zip file. The name of your zip file should be: **yourx500id\_hw3.zip**

PLEASE ENSURE TO TEST YOUR CODE Using Google Chrome and Firefox.

NOTE: As of Chrome 50, the Geolocation API will only work on secure contexts such as HTTPS. If your site is hosted on a non-secure origin (such as HTTP) the requests to get the user's location will no longer function. Use Firefox and Safari to check your directions functionality in case your computer is running Chrome 50 or later version

## 6 Evaluation

Your submission will be graded out of 115 points (100 points and 15 bonus points) on the following items:

1. Google Map is placed on webpage below the Schedule table with proper alignment as shown in pictures (**10 points**).
2. Custom markers are dynamically placed on the physical locations specified in your Schedule table. You must use JavaScript to **dynamically** obtain the event names, addresses, and time information from your Schedule table and use the Google Maps Geocoding or Places functionality to create markers and place them on the map. (**15 points**).
3. When the markers created in step 5 above are selected (via a click), the markers display the event name, address, and time information of the event. (**10 points**).
4. **(BONUS)** Nearby places can be searched within a specific radius and corresponding markers are created on map (**5 BONUS points**).
5. **(BONUS)** When clicking your mouse on each of the markers placed on map in response to a place search (done in step 4 above), each marker displays the name of the nearby place (**5 BONUS points**).
6. **(BONUS)** All previously displayed markers are removed before the results of a new search are displayed (**5 BONUS points**).
7. Functionality for finding directions to a custom destination from the user's current location functions as specified in this write-up.
  - i. Accept destination through input text box and correctly display directions on side panel (**10 points**).
  - ii. Functioning capability to switch between multiple travel modes (**5 points**).
  - iii. Style and alignment of the side panel with scroll feature as shown in picture (**5 points**).

8. The form page is modified to include a Google Map. **(10 points)**
9. On the forms page, when a location on the Google Map is selected, it will automatically populate the address field of the form (POI functionality). **(10 points)**
10. stockQuotes.html page is added to your website and can be successfully displayed by your server when the Get request: <http://localhost:4131/stockQuotes.html> is entered in your browsers address bar (when your updated Python server is running) **(10 points)**
11. When the stockQuotes.html page is displayed in your browser, it functions correctly (when a stock ticker symbol is entered in the input text object, and the Get Stock Data button is clicked with your mouse, current stock data (e.g, the stock price) is returned and displayed in the textarea object as pictured in the Screen shots 4.10 and 4.11 above. **(10 points)**
12. All your webpages should be responsive. **(5 points)**
- 13.** All the files required for your solution should be packaged in a tar or zip file, and submitted via the link on the Canvas class site. The name of the zipped file should be **yourx500id\_hw3.zip**. ***Failure to do this correctly may result in a deduction ranging from 10 points up to full credit for the assignment.***