

**CSCI 4131 – Spring 2022**  
**Internet Programming Assignment 3**  
(Version 2, Feb 11<sup>th</sup>) changes highlighted in *Red*  
***Due Date: Friday February 25<sup>th</sup> at 11:59pm***  
***Late Submission Deadline: Sunday, at 5:59am***

## 1 Description

For this assignment, you will add Google Maps to the functionality you have developed on your Contacts and Form Web pages through homework 2; enhance the styling of your Widgets Page (CSS / HTML); and eliminate errors. There are **13** pages in this assignment specification.

The objective of this assignment is to continue to develop your JavaScript skills, introduce you to the Google Maps JavaScript application programming interface (API), 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 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, on a regular basis. Teaching a specific API is counterproductive in such an environment. Instead, one of the objectives of this assignment is to motivate you to develop and/or bolster your skills to learn and use new libraries and APIs.

You will update your Contacts page as follows:

1. You will write code to ***dynamically*** mark the addresses of your contacts specified in the contacts 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.
3. ***You will also add functionality to calculate directions and display a route between your selected location and selected destination (which is the location of one the people on your contacts list) on your Google map via various modes of transportation.***
4. ***You will rework the “large image” functionality you implemented in assignment to calculate and display the distance of a place to a target location.***

You will update your form page as follows:

1. You will add a Google map on which you can select a location on the Google map and use the click on point of interest (POI) capability to fill in the address field on your form.

Finally, if you have not done so already, you should restyle the widgets on your widgets page, so they are better positioned and error-free (so you will have to fix any errors due to the embedded widgets you obtained from YouTube, Twitter, and Instagram).

## 2 Preparation/Reference

Sign up for the Google Maps JavaScript API – see:

<https://developers.google.com/maps/documentation/javascript/get-api-key>

**Note, you need a credit or debit card to sign up for the Google Maps Application Programmer Interface. The uses required by this course will never necessitate a charge for use of Google Maps – it is only required should you deploy your Maps to a production website. See the instructor if you have questions or concerns about this.**

### 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 obtaining your API key and setting up your map, and it is recommended to complete the tutorial. The tutorial can be found at the following link:

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

When signing up for your API key, use your UMN x500 account

### 2.2 Google Places JavaScript Library Reference

You can refer the following link to obtain more information about 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 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 (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>

### 3 Functionality

#### Update functionality on your Contacts page

1. Add a new functionality to the “large-image” display: a distance calculator. See figure 4.2 below.
2. Start by modifying the address column of the table from homework 2: include the complete address information in table entries for those contacts, e.g., 100 Union St. SE, Minneapolis. MN 55455
3. Embed a Google map under the table.
4. The map should be centered on University coordinates (44.9727, -93.23540000000003) with 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/>
5. You should write JavaScript code to *dynamically* extract the names, contact information, figure and addresses of your contacts from the DOM objects that are in your Contacts table. Your code should then place a custom marker on the map for each location extracted. The markers should display the name and contact information of the contact, and also display the complete address of this contact (e.g. Room number 2-209 in Keller or Shepherd 383) upon being clicked. **Note, do not use hard-coded latitude and longitude positions to do this.** More specifically, create JavaScript to obtain a list of addresses from your contacts and use the geocoding or places library to obtain a latitude and longitude for each of them. Then create a marker for each unique latitude and longitude. This can be achieved using an information window (see figure 4.3 below). If more than one contact has the same location, you only need to generate a marker for one of them.
6. 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 figures below.
7. The first row of input area (created as per step 6 above) should enable a user to for search places, e.g., restaurant 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 for this homework are restaurant, hospital, parking, supermarket. You can replace these 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 of the categories you include.
  - A text box that specifies the numeric radius around your current location in which the search results will preferably be shown.
  - A textbox that allows users to enter the keywords of other places that they may search for. This textbox is by default disabled. The drop down field should include an extra option named: “Other”. When “Other” is selected, the textbox is enabled.

Try one of the following keywords (or another keyword of your choice) when you are testing this – for example: burger, bus, library, laundromat, or 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 places found. The old markers (e.g. the markers for your contact locations, 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.
8. 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)) or another provided location to the destination they select (the location of one of the contacts). The group of elements you should use to implement this functionality are as follows:
- A textbox that allows the user to input the original location. The default original location will be the current location of the user and it can be obtained using geolocation.
  - A drop down list that contains the contacts’ name extracted from the table.
  - 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 original location and the destination should be displayed on the map. The route displayed 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, 4.7 and 4.8 for an illustration of this functionality

## Update Functionality on your Forms page

- a.) Add a Google Map to the right of your form
- b.) ***To earn up to 10 bonus points***, 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 4.9 and 4.10 below for an. See the code snippet after the figures to help you incorporate autocomplete functionality.

**Update the Style on your widgets page and get rid of the errors caused by the embedded widgets obtained from YouTube, Instagram, and Twitter.**

See figure 4.12 below for an illustration of how to improve styling your widgets page (if you have not already done so).

## 4 Screenshots

### 4.1 Overall structure of your new design.

The screenshot shows a web application with a dark blue header containing 'My Contacts', 'Form Input', and 'My Widgets'. Below the header is a 'My Contacts' section with a table listing contacts. To the right of the table is a sidebar featuring a large yellow 'M' logo with a cartoon mascot, a map of Minneapolis, and search controls. The table has columns for Name, Contact Category, Location, Contact Information, email, and Website. The sidebar includes a search bar, a dropdown for 'Find Restaurant', and a 'Take me to:' section with a dropdown for 'Dr. Dan Chailou' and radio buttons for 'WALK', 'DRIVE', and 'TRANS'. A 'Go' button is at the bottom of the sidebar.

Name	Contact Category	Location	Contact Information	email	Website
Tianming Cui	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Csci 4131 TA	cuiux327@umn.edu	CS Department Home Page
President Joan T.A. Gabel	Industry	202 Morris Hall 100 Church Street SE Minneapolis, MN 55455	President of the University of Minnesota System	ugpres@umn.edu	Home Page
Dr. Dan Chailou	Personal	383 Shepherd Labs 100 Union Street SE Minneapolis, MN 55455	Professor of Computer Science	chail000@umn.edu	Home Page
Thomas Nguyen	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Csci 4131 TA	nguy3817@umn.edu	CS Department Home Page
Peter Wang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Csci 4131 TA	wang5959@umn.edu	CS Department Home Page
Zejun Zhang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Csci 4131 TA	zhan756@umn.edu	CS Department Home Page

4.2 When your mouse is hovering over a contact, show its distance to the target location (the default latitude and longitude of the target location is the University coordinates (44.9727, -93.23540000000003)). Note: computing the distance using latitude and longitude coordinates can be non-trivial! Please review the instructions at the link below to assist you:  
<https://www.movable-type.co.uk/scripts/latlong.html>

This screenshot shows the same web application as the previous one, but with the 'Distance to the target location (metres): 359.87834773303047' displayed below the map. The table and sidebar are identical to the previous screenshot.



You can modify the target location, and the result should change.


My Contacts

Form Input

My Widgets

My Contacts

Name	Contact Category	Location	Contact Information	email	Website
Tanning Cul	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	cuiux327@umn.edu	CS Department Home Page
President Joan T.A. Gabel	Industry	202 Morrill Hall 100 Church Street SE Minneapolis, MN 55455	President of the University of Minnesota System	upres@umn.edu	Home Page
Dr. Dan Challou	Personal	383 Shepherd Labs 100 Union Street SE Minneapolis, MN 55455	Professor of Computer Science	chal0006@umn.edu	Home Page
Thomas Nguyen	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	ngay3817@umn.edu	CS Department Home Page
Peter Wang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	wang5959@umn.edu	CS Department Home Page
Zejun Zhang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	zhan7565@umn.edu	CS Department Home Page




Latitude/Longitude of the Target location: 43.1 -95.3

Distance to the target location (metres):265999.70184599276

Map

Satellite



Keyboard shortcuts Map data ©2022 Google Terms of Use Report a map error

Find Restaurant

Enter Other Places

In 500 meters

Search

Take me to:

Dr. Dan Challou

From Default (current location)

by

WALK

DRIVE

TRANS

Go

Result for another contact


My Contacts

Form Input

My Widgets

My Contacts

Name	Contact Category	Location	Contact Information	email	Website
Tanning Cul	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	cuiux327@umn.edu	CS Department Home Page
President Joan T.A. Gabel	Industry	202 Morrill Hall 100 Church Street SE Minneapolis, MN 55455	President of the University of Minnesota System	upres@umn.edu	Home Page
Dr. Dan Challou	Personal	383 Shepherd Labs 100 Union Street SE Minneapolis, MN 55455	Professor of Computer Science	chal0006@umn.edu	Home Page
Thomas Nguyen	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	ngay3817@umn.edu	CS Department Home Page
Peter Wang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	wang5959@umn.edu	CS Department Home Page
Zejun Zhang	Adademic	2-209 Keller Hall 200 Union Street SE Minneapolis, MN 55455	Spring 22 Caci 4131 TA	zhan7565@umn.edu	CS Department Home Page




Latitude/Longitude of the Target location: 43.1 -95.3

Distance to the target location (metres):265976.27731037553

Map

Satellite



Keyboard shortcuts Map data ©2022 Google Terms of Use Report a map error

Find Restaurant

Enter Other Places

In 500 meters

Search

Take me to:

Dr. Dan Challou

From Default (current location)

by

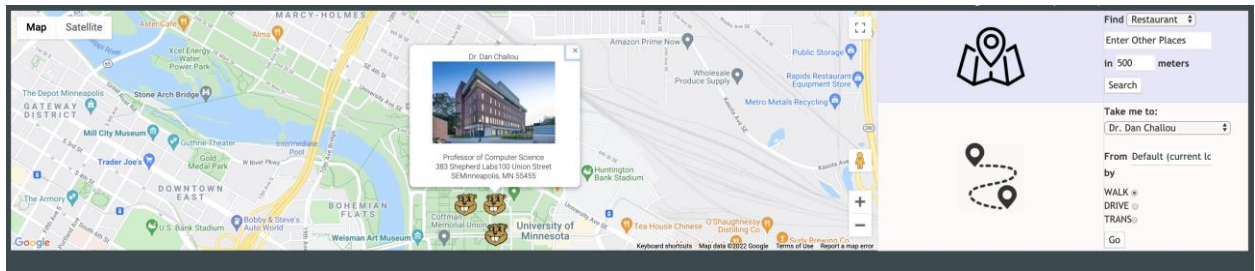
WALK

DRIVE

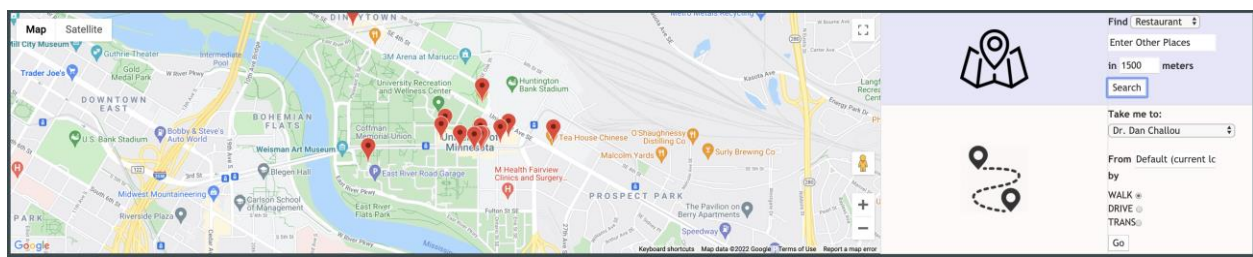
TRANS

Go

4.3 Marker shows the name, contact information, figure and address of a contact on click (you should use JavaScript to extract that information from your contact table)



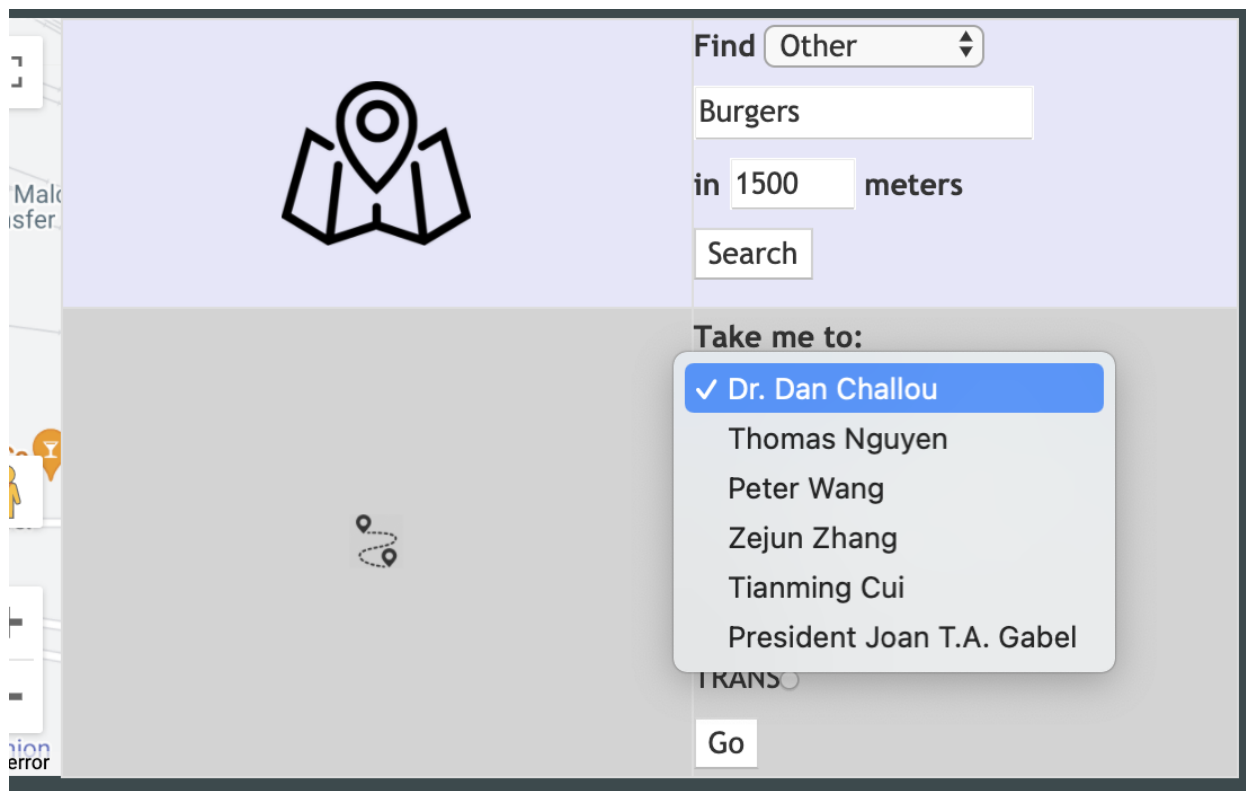
4.4 Map showing the result of nearby restaurants in a radius of 1500 meters



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., “burgers”. The nearby places with the keyword in their names will show up when “search” button is clicked (See the results for “burgers” below)

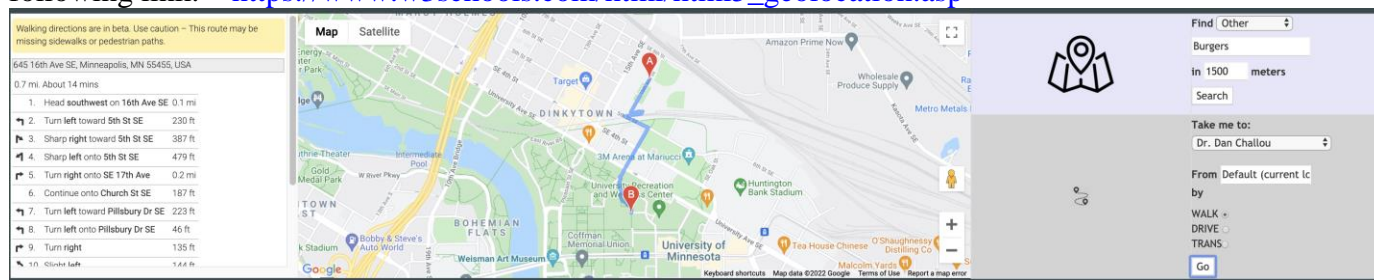


4.6 There should be a “take me to” panel next to the nearby search, which contains a drop-down list with options extracted from the contacts table. (Note, this should also be generated with JavaScript by extracting information from the table, you should not hard code those options)



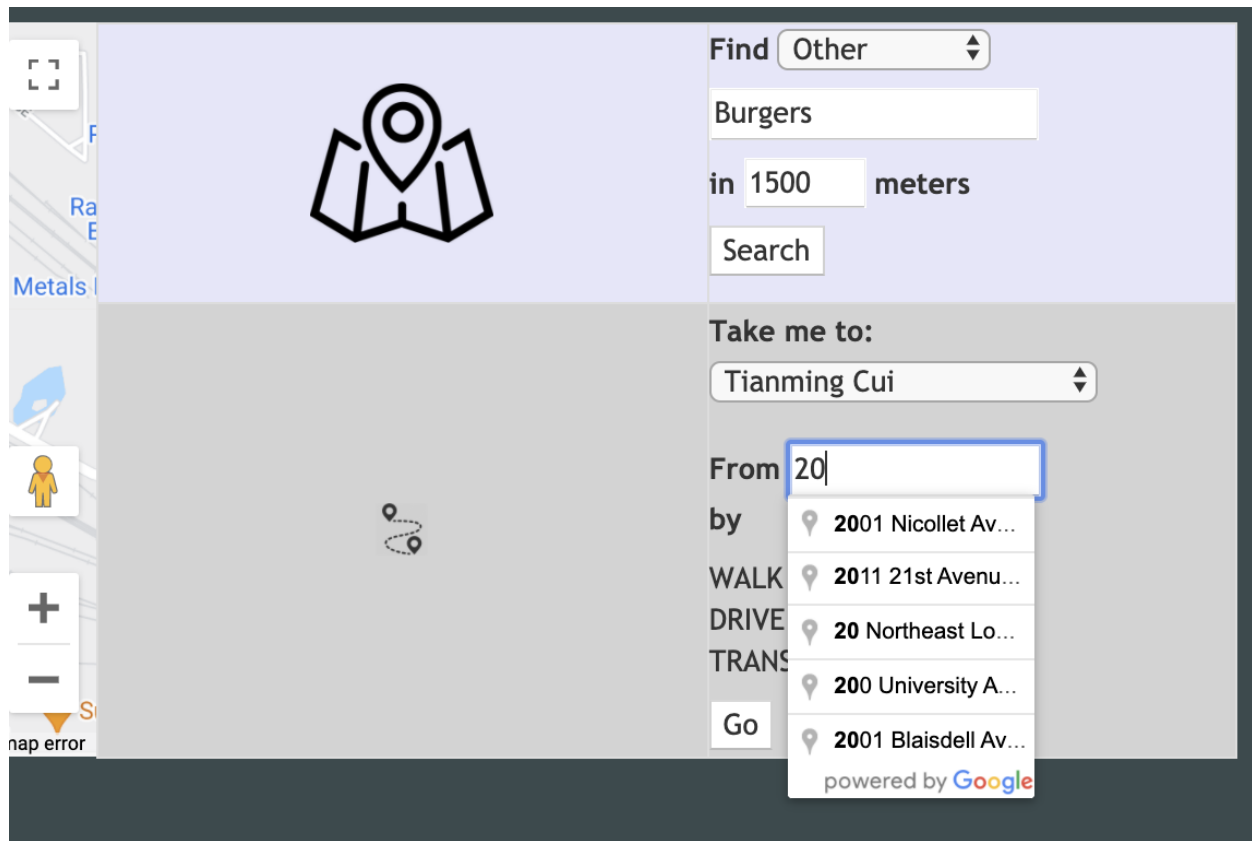
4.7 After selecting target contact, original location (by default, it is your current location) and a method (walk/drive/trains), the map calculates and displays a route:

**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)

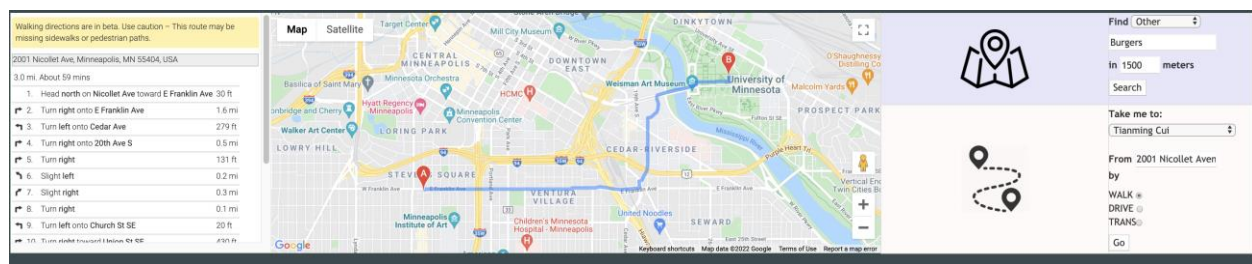




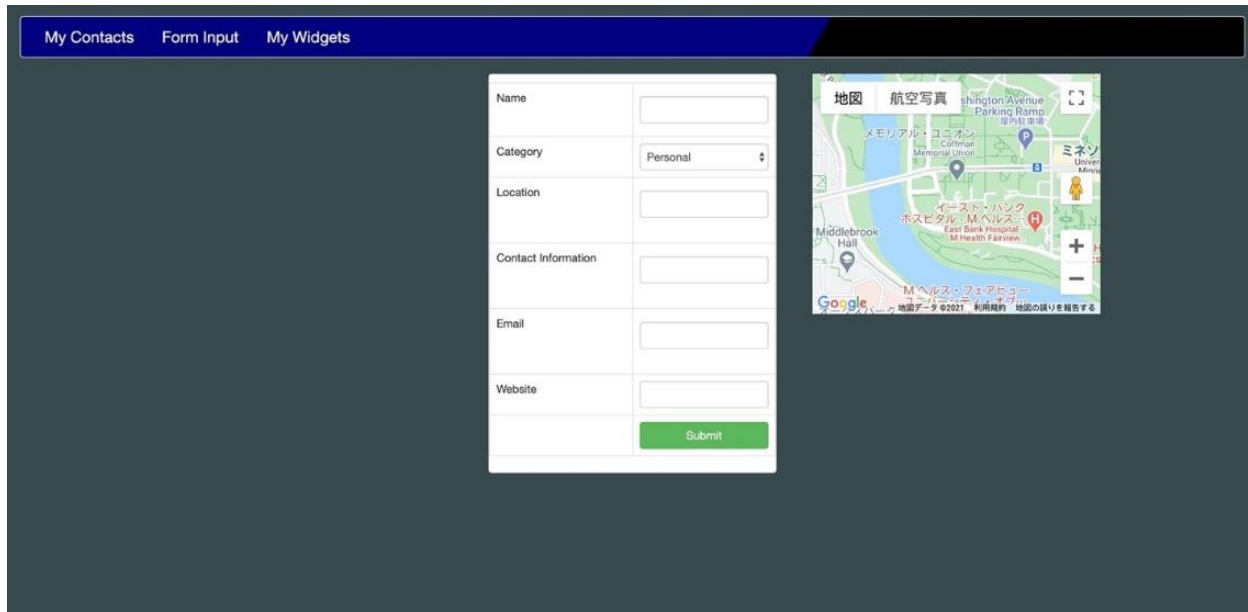
4.8 You can select another contact, and modify your original location:



Then a new route will be displayed:

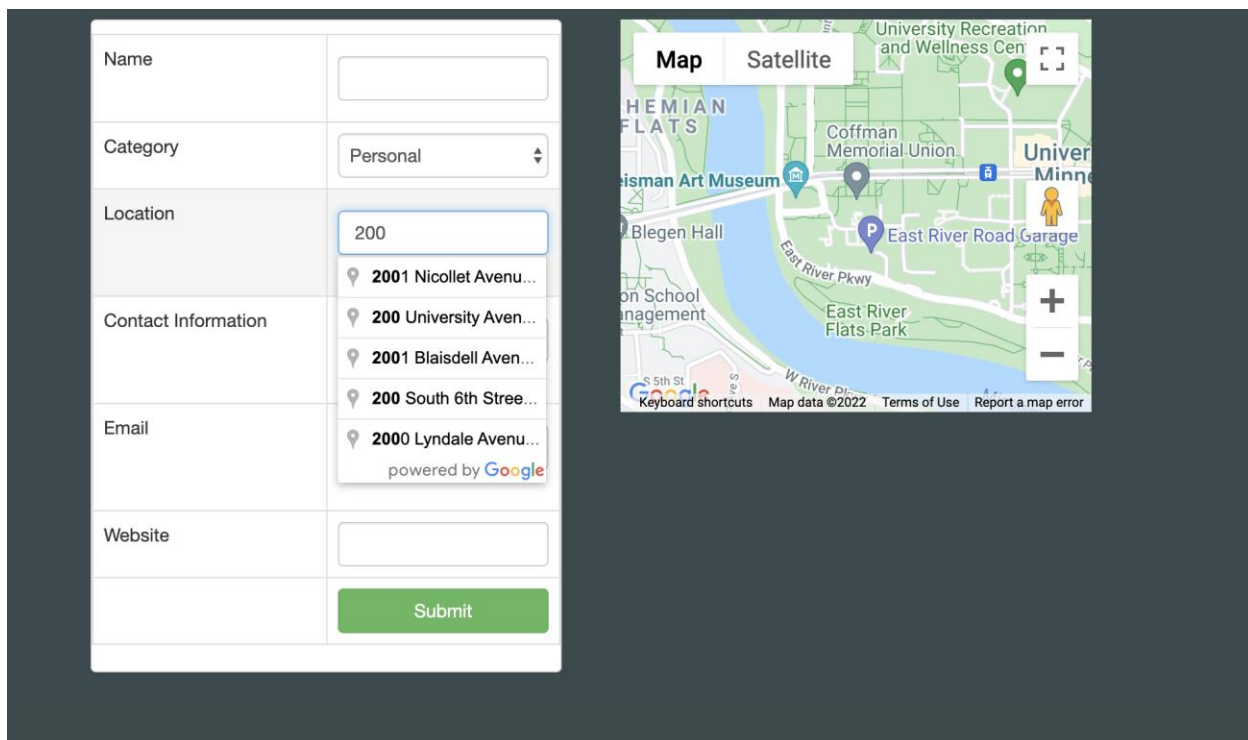


#### 4.9 Add a Google Map to your Form Page



The screenshot shows a form page with a dark blue header containing the tabs "My Contacts", "Form Input", and "My Widgets". The form itself is white and contains the following fields: "Name" (text input), "Category" (dropdown menu set to "Personal"), "Location" (text input), "Contact Information" (text input), "Email" (text input), and "Website" (text input). A green "Submit" button is at the bottom right of the form. To the right of the form is a Google Map widget showing a map of Minneapolis, Minnesota, with labels in Japanese and English for various locations like "Middlebrook Hall", "East Bank Hospital", and "M Health Fairview".

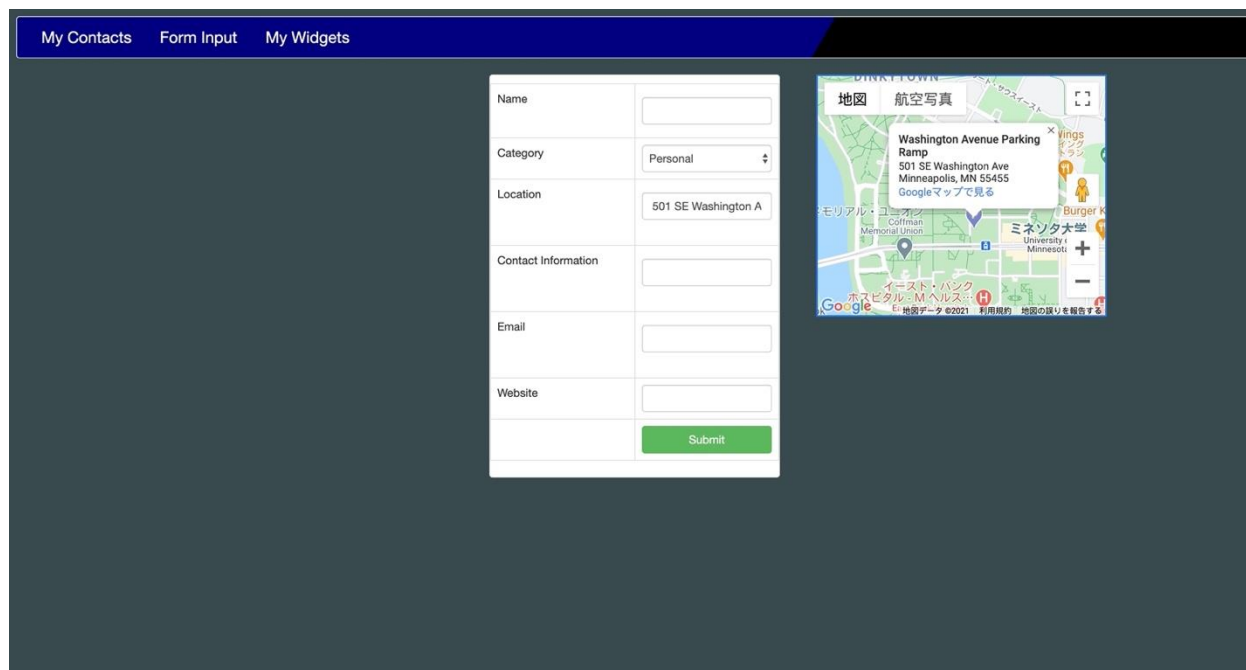
4.10 The location input box (on both form page and contact page) should support “auto-complete”, which helps the user input their locations easier, shown as following, you should do the same thing on your contact page when inputting the original location:



The screenshot shows a form page with a white background. The form fields are: "Name" (text input), "Category" (dropdown menu set to "Personal"), "Location" (text input with an auto-complete dropdown), "Contact Information" (text input), "Email" (text input), and "Website" (text input). A green "Submit" button is at the bottom right. The auto-complete dropdown for the "Location" field is open, showing a list of suggestions: "200", "2001 Nicollet Avenue...", "200 University Avenue...", "2001 Blaisdell Avenue...", "200 South 6th Street...", and "2000 Lyndale Avenue...". The text "powered by Google" is at the bottom of the dropdown. To the right of the form is a Google Map widget showing a map of Minneapolis, Minnesota, with labels in English for various locations like "University Recreation and Wellness Center", "Coffman Memorial Union", "East River Road Garage", and "East River Flats Park".

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

(e.g., 501 SE Washington Ave Minneapolis, MN, 55455 is shown in the screenshot below)

The screenshot shows a web application interface. At the top is a dark blue navigation bar with three tabs: "My Contacts", "Form Input", and "My Widgets". Below this is a form with several input fields: "Name" (text), "Category" (dropdown menu showing "Personal"), "Location" (text field containing "501 SE Washington A"), "Contact Information" (text), "Email" (text), and "Website" (text). A green "Submit" button is at the bottom right of the form. To the right of the form is a Google Map. The map shows a street view of Minneapolis, MN, with a red pin and a label "Washington Avenue Parking Ramp" pointing to "501 SE Washington Ave Minneapolis, MN 55455". Other nearby locations like "University of Minnesota" and "Burger King" are also visible on the map.

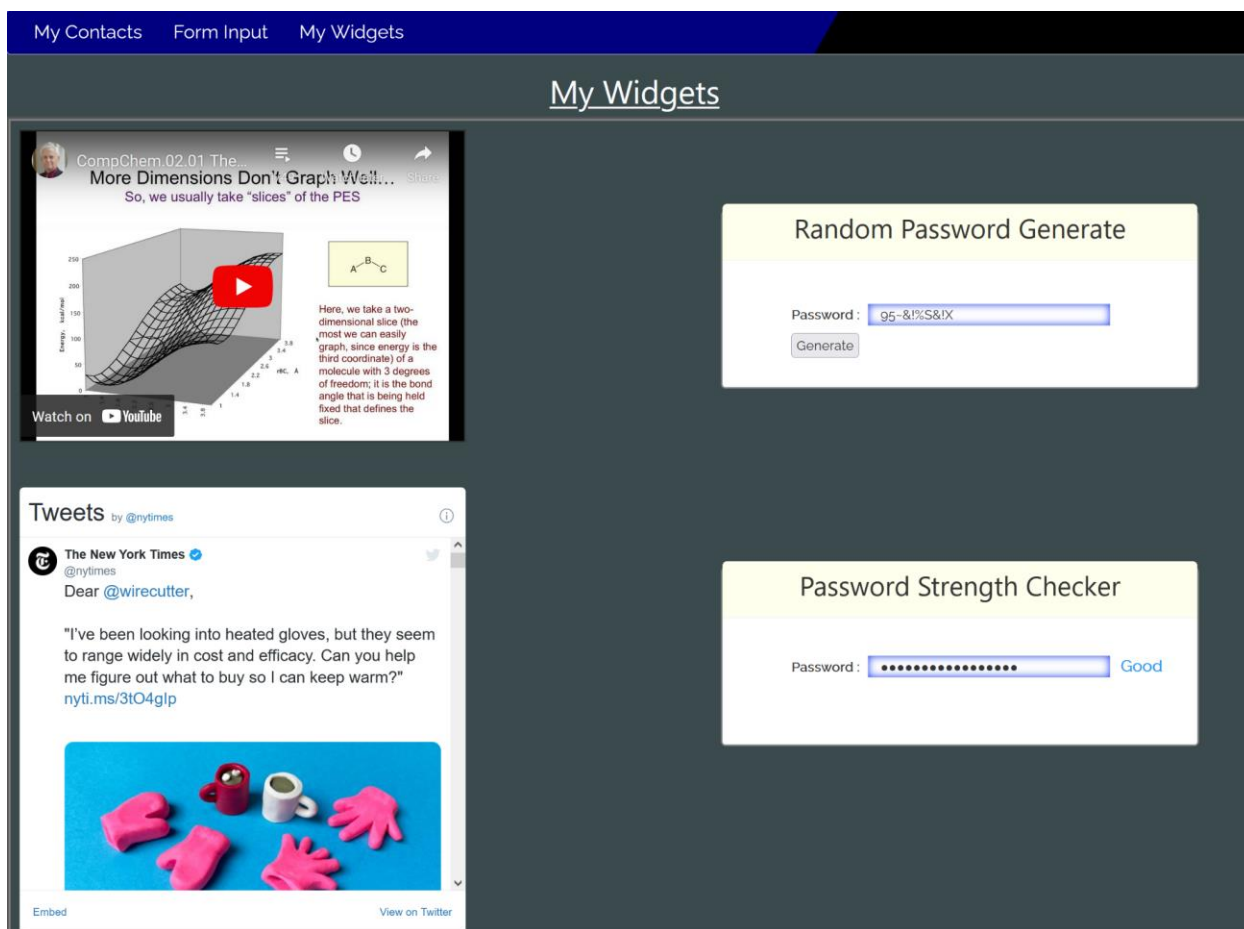
The “snippet” of HTML and JavaScript below can be used to help incorporate Google Maps Auto-complete functionality into the location text field on your form.

```
<input id="locationTextField" type="text" size="50">
  <script>
    function init() {
      var input =      I
        document.getElementById('locationTextField');
      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>
```

4.12 Improve the styling on your widgets page, and get rid of HTML and CSS errors caused by the Instagram, YouTube, and Twitter widgets (check to ensure you have done that with the HTML and CSS validators).



## 5 Submission Instructions

1. For this assignment, you are expected to separate the HTML, CSS, and JavaScript files.
2. Your submission should include a **minimum** of 5 files (3 HTML, and 1 each for JavaScript and CSS). You can include more than one JavaScript and/or CSS file. However, it is required that the CSS is separated from HTML (that is, in an external file), and you should include the majority of your JavaScript in an external file or files as well.
3. Zip these files and the name of the zipped folder should be **yourx500id\_hwk03**. PLEASE ENSURE TO TEST YOUR CODE Using Google Chrome and Firefox.



## 6 Evaluation

Your submission will be graded out of 120 points (100 points base and 20 possible bonus points) on the following items:

1. ALL HTML files pass the w3schools validator (<http://validator.w3.org>) without errors. **Warnings are accepted (5 points).**
2. All CSS files pass w3schools validator (<http://jigsaw.w3.org/css-validator/>) without errors. **Warnings are accepted (5 points).**
3. Google Map is placed on webpage below the Contacts with proper alignment as shown in pictures **(10 points).**
4. Custom markers are dynamically placed on the locations specified in your Contacts table. You must dynamically obtain the names, addresses, figure, and contacts information from your Contacts table and use the Google Maps Geocoding or Places functionality to create markers and place them on the map. **(15 points).**
5. When the markers created in step 5 above a selected (via a click), the markers display the name, address, figure and contact information of the contact. **(5 points).**
6. The “distance calculator” works correctly. **(10 bonus points)**
7. Nearby places can be searched within a specific radius and corresponding markers are created on map **(10 points).**
8. When clicking your mouse on each of the markers placed on map in response to a place search, each marker displays the name of the nearby place **(5 points).**
9. All previously displayed markers are removed before the results of a new search are displayed **(5 points).**
10. Functionality for finding directions to a contact’s location from the original location (provided by the user or default as current location) functions as specified in this write-up.
  - i. Accept destination / original location through input and correctly display directions on side panel **(15 points).**
  - ii. Functioning capability to switch between multiple travel modes **(4 points).**
  - iii. Style and alignment of the side panel with scroll feature as shown in picture **(6 points).**
11. The form page is modified to include a Google Map. **(10 points)**
12. On the forms page, when a location on the Google Map is selected, it will automatically populate the address field of the form. **(5 bonus points)**
13. Enable autocomplete functionality on the address field of the form **(5 bonus points)**
14. The Webpage should be responsive. (5 points)
15. All the files required for your solution should be packaged in a tar or zip file, and submitted via the link on the Moodle class site. The name of the zipped file should be **yourx500id\_hwk03.zip**. Failure to do this correctly may result in a deduction ranging from 10 points up to full credit for the assignment.