1.0 PREAMBLE

- Main technologies: Bootstrap, CSS, HTML, JavaScript
- Project Duration: 3 **work-weeks**

2.0 OBJECTIVES

In this **Industry Project**, you are to create an *interactive experience using Leaflet Map, HTML/CSS, Bootstrap* and vanilla JavaScript. Your project will test you on the following categories:

1. HTML and Mobile Responsive Design
2. JavaScript and DOM manipulation proficiency
3. Interaction Design
4. User Experience / User Interface
5. Work and Best Practices

By the end of this duration, you should have a project that you can show to future employers in your portfolio.

3.0 REQUIREMENTS

1. Build a **mobile first, responsive** web application using HTML, CSS and JavaScript
2. Decide on one of the following themes:
    a. Feature **Leaflet** as the front and center of your application and it must solve the **primary need** and **pain points** of a target audience group.
    b. Feature a **user interface** that utilizes one or more **RESTFul API**, and you must be able to demonstrate CRUD.
3. Make use of AJAX requests, either to your own JSON files or to the endpoints of an API, as part of your app.
4. You are limited to vanilla JavaScript. You may not use other frameworks such as React, Vue, Angular etc.
5. **Do not follow an online tutorial to complete your project, from YouTube or otherwise.**

PROJECT DESCRIPTION

Create a web app where an interactive map (using Leaflet) is the centerpiece. Make use of markers, map layers, drawings etc. to provide useful information to the user. The user must be able to interact with the map in meaningful ways, such as clicking on markers to activate a process or being able to manipulate the map display with HTML form elements.

3.1 Suggested RESTFul APIs

- Leaflet: You can consider using the FourSquare API together with your Leaflet project
- User interface: You can consider using the https://spacetraders.io/ , a RESTFu API that simulates a trading game set in space. It has CRUD endpoints, and also other endpoints for performing actions in the games.

4.0 ACADEMIC INTEGRITY

4.1 Plagiarism

Plagiarism is the act of passing off someone else's work as your own. The original work could have been modified with slight modifications but if at least 80% of the original work remains, it can be deemed plagiarism. You have to acknowledge all content that is not original (see point 6 above).

Plagiarism does not apply to boilerplate code, such as <meta> tags in <header>. Bootstrap components do not count too, **but a specific arrangement or order of Bootstrap components may**. Using a template **counts** however if you modify the layout and the type of components used significantly, then it will be counted as original work.

Using examples given in class and making slight modifications count as plagiarism.
When in doubt, attempt to re-do the code and find different ways of achieving the same result.

If you use code snippets from coding websites, and could not duplicate the effect using your own methods or couldn't rewrite it, you have to acknowledge the use of the code. You will not get the technical marks for that feature but it will still count towards other categories.

Effects that are achieved by using code from Github or other sources (such as Bootstrap carousel) **must** always be acknowledged and does not count towards your scoring for JavaScript or HTML/CSS proficiency.

4.2 Adapting from other sources

You are **not encouraged** to develop your project from following a complete project tutorial/walkthrough from YouTube, social media, Udemy etc, for the following reasons:

- Many of those walkthrough only **show the steps and outcomes**, but does not explain or give reasoning
- Following a tutorial does not prove competence in the eye of potential employers
- The technology or steps that followed are not universal and our TAs might be unfamiliar with them and hence unable to provide support

You are **encouraged** to **learn** from other sources, but you have to **understand their concepts and then apply it, not simply co-opt it.**

4.3 Use of AI

You may use **AI**, such as ChatGPT for:

- Explanation of concept
- Generation of mock data (but please include attribution)
- Generation of mock images (but please include attribution)

You have to take responsibility if generative AI provides code that you **cannot explain or understand**. You have to decide to keep those in and risk future employers quizzing you on them, or work on understanding them.

5.0 PROJECT SUBMISSION

The project is due 2 months when this hand-out is given. **Submission** is only via **Github**.

Each criteria is scored between 1 to 5 points, with 1 being **unsatisfactory** and 5 being **distinction**. To score a 5, the criteria must be achieved on a level acceptable for an **experienced developer**.

A| HTML AND MOBILE RESPONSIVE WEB DESIGN

No	Title	Criteria	Guidelines for scoring
1	**Use of HTML elements**	You demonstrate proficiency with designing web pages with HTML, making use of varied HTML elements.	• You use different kinds of HTML elements, such as `<ul>`, `<ol>`, `<div>`, `<p>` etc. • You incorporate the use of semantic HTML elements: `<section>`, `<article>`, `<main>`, `<header>`, `<footer>` etc. • Your use of semantic HTML is effective and is well-structured
2	**Layout**	You must use at least one layout tool, such as Bootstrap Grid, flex box or CSS Grid	• You have at least one layout that demonstrate your understanding of the tool • Your layout is effective and shows contrast • You show that you can mix different kind of layout tools (such as flexbox with CSS grids, or flexbox with Bootstrap Grid)
3	**Mobile Responsive**	The website must be mobile responsive	• The layout works for a mobile device, a tablet device and desktop device. • Additionally, adjust the font size, spacing, image size etc. so that they

			look good on the three different services • You can score higher by making sure it is mobile responsive for one additional device (such as widescreen monitors)
4	**CSS Selectors**	Are you able to select by class and id? Those are the basics. Try to make use of more advanced ones if possible, such as parent/child selector, pseudo-classes etc.	• You are able to select by class, by id and by tag name • You are able to incorporate advanced selectors • You demonstrate the use of `!important` and/or the use of **CSS specificity** • If you are able to write reusable, modular CSS rules (such as the ones used in Bootstrap), you can score higher than pass.
5	**CSS Box Model**	Adjust the padding, margin, borders using the CSS Box model, and use blocks, inline-blocks and inline when needed	• Does your design have ample white space from manipulating the CSS box model? • Are you able to use shorthands for adjusting padding, margin etc? • Are you able to change certain inline elements to inline-blocks or blocks to achieve certain effects?
6	**CSS Typography**	Adjust font family, font size, line spacing, line height etc. to present text in a visually pleasing manner.	• Demonstrate changing font family, font size etc. • Make use of line height, spacing to make sure your text are presented in an easy to read manner • Make use of text alignment effectively (tip: do not align everything center!)
7	**CSS Framework**	You must use components or widgets from a CSS framework of your choice.	• Use a variety of components, such as cards, list group, carousel etc. from a CSS framework of your choice. We recommend Bootstrap • Customize the components with your own visual styling and content structure • Make sure those components are used effectively to enhance the user experience
8	**CSS Positioning**	You use the various CSS positioning type: absolute, relative, fixed and sticky, as well as using transforms.	• Try to use at least one of each to score at least a `4` • Creative uses of positioning may net you a `5`
9	**Bonus**	You get to earn bonus points from 1 to 5 here.	• Do you include accessibility features, such as `alt` for images • Do you use semantic HTML elements? • You can try to embed social media pages or videos • You can try to make use of CSS animations

B| JAVASCRIPT AND DOM PROFICIENCY

No	Title	Criteria	Guidelines for scoring
1	**Use of Variables and Data Types**	You are able to use variables in your project, and advanced data types, such as objects	• If you have a functional project, you most likely will score a `3` in this criteria. Be sure not use any `var`

		and arrays.	• To improve your score, use `const` whenever appropriate • Proactively use `parseInt`, `parseFloat` functions to ensure the proper data types.
2	**Branching**	You are able to use `if/else` or `if/else if/else` in your project with comparison operators and logical operators	• If you have a few `if/else` or `if/else if/else` in your project you should score a `2` in this criteria. • To improve your score, make sure you use comparison operators and logical operators • Features that can help you to score this includes search and data processing
3	**Loops**	You are able to use `while` or `for` loops	• Have at least two loops to demonstrate proficiency • To improve your score, include the use of `break` and/or `continue` when appropriate • If you can have **nested loops** (one loop within another) you will be able to score higher
4	**Array Processing**	You demonstrate that you can filter, map or reduce an array. • Filter: retrieving certain elements from the array • Map: transform one element of the array to a different one in a new array • Reduce: summarize the content of the array, such as the sum of the array or its count	• At the **bare minimal**, you must be able to demonstrate a combination of filter, map or reduce using `for` loops • Try using `forEach` to score better • To aim for the best score, explore the use of the `map`, `reduce` and `filter` functions
5	**Functions**	You can write functions that accept parameters and return values, and work comfortably with them	• You must have a few functions that accept parameters and return values. This means event listeners **do not count**. • If you follow the lab work for the **CRUD application** and implement a **data layer**, you should be able to meet the basic requirements. • For better score, use **arrow functions** when possible • Demonstrate that you are able to use **closures**. The **CRUD application** lab work uses closures to add interactivity to each todo item.. • Make use of functions as parameters to demonstrate mastery, such as the `sort` function for arrays

6	**Selecting DOM elements**	You are able to select DOM elements for manipulation	• At the minimal you must be able to:
			  ○ select by `id` or `class`
			  ○ Use both `querySelector` and `querySelectorAll`
			• If you wish, you can also demonstrate selecting with `getElementById` or `getElementsByClassName` etc.
			• To prove mastery, use more complicated selectors with `querySelector`/`querySelectorAll`, such as parent/child selector or selecting by attributes.

7	**Manipulating DOM elements**	You are able to manipulate the properties of selected DOM elements, such as their `innerHTML` , or adding new child elements with `appendChild`.	• You must be able to modify the `innerHTML` at the very least • You must demonstrate modifying `.style` **for selected elements**, usually in response to an interaction • You must demonstrate at least one instance of adding child elements to an existing parent element • For more points, consider changing `class` or explore using the `data` properties.
8	**Handling Events**	You must be able to handle events, such as `click`	• Demonstrate you can handle the `click` events and `DOMContentLoaded` event • Explore handling events not covered in class, such as `change` (when a form field is changed), `submit` (when a form is submitted), `mouseover`/`mouseout` (when the mouse cursor moves over or out from an element) • If possible, also demonstrate `preventDefault` and make use of the `event` parameter if possible.
9	**Asynchronous JavaScript**	You must be able to use `await` and `async` in your project.	• Being able to use `await` and `async` in your application is the minimal baseline. • To score higher, try to execute multiple `await` calls in parallel.
10	**RESTFUL API**	Make sure to use at least **one RESTFUL API** in your project	• Consume at least one endpoint (two is better, to show that you can do it) in your project. • Demonstrate you can use data from different RESTFul API • Demonstrate that you process the data from the endpoints meaningfully and display them in a useful manner to the user.
11	**Form Processing and Data Validation**	You must demonstrate your ability to work with forms and form controls such as text field, radio buttons, check boxes and select drop-down	• Incorporate at least **one each of the following: text box, radio buttons, checkboxes and select dropdown** in a form • You must show that you can retrieve the values from the form controls and process them (for example, as part of a search engine) • Consider adding data validation to improve your score.
12	**(Optional) CRUD**	Add in CRUD abilities to your project	• You can do this with JSONBin or MongoDB, once you have learnt how to do it. • You can also demonstrate this with `localStorage` • Some features: add to favorites or allow users to submit new points of interest.

C1| INTERACTION DESIGN (IF USING LEAFLET)

No	Title	Criteria	Guidelines for scoring

1	**Leaflet Map Custom Interaction**	You must have at least one Leaflet map in your application and add **custom interaction** to it. This means in addition to the standard move the map and zooming the map, you add in other ways to interact with the map.	• You must have **ways** for the user to alter the content of the map without using the standard Leaflet controls. Some examples are shown in the class, especially those in the labs dealing with Layer Groups • Adding a search feature is the easiest way to meet this criteria. Make sure you include **multiple** form controls (radio buttons, checkboxes etc.) • You **must** customize the markers used in the Leaflet map from the standard one. • The more custom interaction you add, the higher the score for this criteria.
2	**Layers and Layer Groups**	You use a variety of layers and organize them in layer groups.	• Make use of layers such as markers, circles, lines etc. The use of those layers must make sense in context of the user experience. • Use Layer Groups to organize the layers • While you can use `L.layerControl` to use the default layer select, consider coding your own with `addLayer`, `removeLayer` and `hasLayer` for better score.
3	**Layer Interaction**	Make sure your layers aren't static. When the user clicks on a layer, there must be some interaction (feedback, more information etc.)	• Use `bindPopup` or `addEventListener` to add event listeners to layers (such as clicking on a marker for more information) • Execute complicated processes when a layer is clicked, such as making a call to a RESTFUL API endpoint.
4	**External Data**	Use external data to draw layers on your Leaflet map. This can be data from `JSON`, `GeoJSON` or RESTFUL API	• Have at least **one** external data source • Use more data sources to score higher.
5	**Leaflet Plugins**	Make sure of Leaflet plugins (such as Marker Cluster Groups) to add more features to your application	• Demonstrate at least **one** external plugin • You should try to incorporate other Leaflet plugins to score higher
6	**Leaflet Map Customization**	You customize the Leaflet map to suit your needs	• Change the tile map provider to a different one besides OpenStreetMap (which is not visually appealing) • Make sure you customize the icons • For additional score, customize the default Leaflet controls (it's via CSS).

C2 INTERACTION DESIGN (IF **NOT** USING LEAFLET)

No	Title	Criteria	Guidelines for scoring
1	**List Rendering**	Your user interface must render a list of data records. Each data record must have at least five fields of different data types.	• To score extra grades, consider adding a **search feature** to allow users to filter the list. • Also consider making use of components from Bootstrap to display each record. • The user must be able to access **edit** and **delete** functionality readily.

2	**Creating new record**	Your user interface must be able to allow users to create a new data record. The data record must be persisted.	• Provide a form for user to create a new data record • For extra grade, make sure the user's input are validated before being stored
3	**Editing existing record**	Your user interface must be able to allow users to edit an existing record.	• Minimally, provide an interface for the user to edit an existing record • For extra grade, ensure that the user's input is validated before updating the existing record • For extra grade, also consider the user experience when editing an existing record (for example, avoiding the use of `prompt`)
4	**Delete existing data**	Your user interface should allow the user to delete an existing data record.	• For extra grade, let the user confirm their decision to delete the data record.
Choose either 5A or 5B			
5A	**External data source**	Use an additional RESTFul API that to use in your project. For example, you can use the Foursquare API to allows the user to search for a location which they can add to their own bookmarks, or a financial API to get the latest exchange rate on a currency	• Demonstrate usage at least **one** other RESTFul API
5B	**Execute Process via Endpoint**	Execute a process that works on your data via an endpoint. That process must not be CRUD related. f	• Add in two more different features that execute a process via endpoints.
6	**Data Visualization**	Use one method, besides displaying data in a list, to allow the user to visualize data, such as a line bar, bar chart or other forms of graphs.	• For extra grade, consider more than one visualization.

D| USER EXPERIENCE / USER INTERFACE

No	Title	Criteria	Guidelines for scoring
1	**Primary Needs and Pain Points**	Your application must solve one or more primary needs of a target audience, and incorporate features for their pain points. .	• Minimally, your Leaflet application must be **functional** and **useful** for your target audience. • To score, add features that solve their **pain points**. For example, if your target audience are "dog owners" and their problem is "I want to find nearby dog parks", then you should have a Leaflet map to show them the dog parks nearby them. There should also be features that solve their problem (i.e, a way to find dog parks near their current location) Their pain points could be: 1. Sometimes the dog parks are closed when they reach there 2. They don't know the amenities around the dog park that they want to go to

			3. It's hard to find parking near those dog parks. You can then devise features to address those pain points after taking care of their primary needs.
2	**Visual design**	Your application must use color, fonts and images that suit your target audience.	● Be **specific** about your **core** target audience. You could say anyone from 15 to 65 could use your web site but **if you only have an advertising budget to target one core group, who will it be?** ● Once you have targeted your core group, your design must **appeal** to them and must be functional for them.
3	**Typographic design**	You use fonts appropriately and are able to make use of line heights, spacing etc to create text that are pleasing to read	● Minimally, make sure to use a serif and a sans-serif font ● To score, make sure the font are easy to read and fits your visual design ● Make sure the color of the fonts are suitable to score better ● Experiment with line directions, different text size, alignments etc. for better contrast
4	**Color design**	You use colors appropriately	● Minimally, have a consistent color scheme for your website that has some effort put in (not just black and white!) ● You **must not use the default Bootstrap color scheme.** Doing so is an automatic 0. ● To score, you should the colors are pleasing and match your audience and your visual design.
5	**User journey and site structure**	The user must be able to find what they want quickly	● The user must be able to find what they need within 3 clicks ● To achieve a better score, pre-empt what your user may want and make them readily available, through a call to action or a sticky footer. ● Use suitable site structure (linear, hub, tree etc.)
6	**Visual Layout and Information Hierarchy**	The user should know where to click and where to find the information they want on the screen	● Make sure elements that can be clicked or interacted with are obviously clickable ● Make sure you use principles of information hierarchy to present information (header should have larger fonts, important information should have a border on a different background color etc.) ● Make sure you use layout and white space for contrast
7	**Polish**	How ready is the application for deployment?	● 1 point: the project is still in a raw, unfinished state ● 2 point: the project is functional but has critical bugs ● 3 point: the project is feasible as **proof of concept** ● 4 point: the project is feasible as a minimal viable product ● 5 point: the project is feasible for a soft launch ● 5+ point: the project can be launched

E| WORKFLOW AND BEST PRACTICES

No	Title	Criteria	Guidelines for scoring
1	**Folder Structure**	You organize your files across different folders. Each folder should be named appropriately. The folder structure should make it easier to find the files.	• Use two or more folders to organize your project files • Subfolders are not necessary unless you have lots of files, and they must be easier to help you to find them • Your naming convention for folders must be consistent.
2	**File Naming**	Follow a consistent naming convention for your files (for example, camel cases, kebab cases or snake cases).	• If you are following a style guide, check their guidelines on how to name files. • **Consistency** is the key.
3	**Code Construction**	Code construction is concerned with coding best practices to make sure code is readable and of high quality.	Code construction is concerned with following best practices such as: • Use of comments • Making sure functions are short • Make it easy to identify how a `while` or `for` loops to end • Clear and descriptive variable naming conventions • Clear and descriptive function naming conventions

4	**Use of Git as Source Control**	Commit to Git regularly with clear commit messages and make use of Git to manage your source code	• Minimally:
			  ○ Make sure you have regular commits and pushes to commit
			  ○ Make sure your commit messages are clear and meaningful
			  ○ As a rule of thumb, one day of work should incur at least one to two commits minimum.
			• To score more, explore using branches to develop features and then merge them back into the `main` branch
			• You can also use Github issues to track your own bugs.

5	**Readme file**	You should have a `readme.md` file that explains your project	Minimally, your `readme.md` file should: • Contain a description of your project • Use user stories to explain **who** the users are, **what** are their primary problems and pain points • Describe the **features** of your project and how it addresses their **problems** and **pain points.** • Describe the list of technologies used in your project • Make sure you use markdown language properly To score points, you should • Describe your design and thinking process for your user interface and user experience. Explain your choice of fonts, color, layout and visual theme. • Include wire frames (drawn using Balsamiq, Creately or draw.io) and site diagrams • Generate screenshots of your project showing how it looks like in different devices

			• Show test cases and testing steps • Explain how to deploy the project

6	**Software Engineering (optional but important)**	How modular and reusable are your code?	• You should be able to display **separation of concern**, having functions that have one responsibility (or sub-responsibility). The best examples are the functions in the data layer as demonstrated in the CRUD application.
			• You should break your code across multiple files (`css` or `js` files)
			• To score extra, consider:
			  ○ Create a library of functions that are reusable across different projects
			  ○ Create functions that can be customized via parameters
			  ○ Consider implementing some version of the MVC paradigm.

7	**Code Robustness**	Can your code survive user errors and exceptions?	• Use of `try/catch` in your code • Make sure to detect invalid user input for higher score • Recover when there are error in input or data