Assignment 3: Ajax, JSON, Responsive Design and Node.js

Weather Search

(AJAX/JSON/HTML5/Bootstrap/Angular /Node.js/Cloud Exercise)

1. Objectives

- Achieve familiarity with the AJAX and JSON technologies.
- Create the front-end by integrating and combining HTML5, CSS, Bootstrap, and Angular**.**
- Create the back end using JavaScript on Node.js.
- Both front-end and back-end must be implemented in one of the approved cloud environments.
- Build responsive web design with Bootstrap to enhance UX across multiple screens.
- Deploy your Web app on Google Cloud Platform/Amazon Web Services/Microsoft Azure.
- Leverage APIs [illegible] th include tomorrow.io API, Google M [illegible] API, HighCharts and X (aka Twitter) A [illegible]
- Learn how to r [illegible] Atlas, in the cloud.

2. Background

2.1 AJAX and JSON

AJAX (Asynchronou [illegible] logies which include Standards-based prese [illegible] and interactions using the Document Object [illegible] sing XML and JSON, Asynchronous data re [illegible] cript binds everything together. Peruse the cl [illegible] iled information.

JSON, short for JavaS [illegible] ge format. Its primary application for this tasl [illegible] serves as an alternative to the XML format fo [illegible] e class slides covering JSON on D2L Brights [illegible]

2.2 Bootstrap

Bootstrap is a free collection of tools for creating responsive websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. **We recommend Bootstrap 5.3 and ng-bootstrap (v 17.x.x) in this assignment.** In general, you can use **Bootstrap 5.2.3** through **5.3.2**, **Angular 15** through **17**, **ng-bootstrap 14** through **16,** and **Node.js 18 or 20** in this assignment. To learn more details about Bootstrap please refer to the lecture material on Responsive Web Design (RWD). Peruse the class slides covering Responsive Design on D2L Brightspace.

2.3 Google Cloud Platform (GCP)

Google Cloud Platform (GCP) applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With GCP, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and NoSQL databases, Memcached, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable in GCP. Peruse GCP support information for Node.js in the ungraded assignment **Cloud Setup (NodeJS)** in the Assignment folder on D2L Brightspace.

To learn more about GAE support for Node.js visit this page:

https://cloud.google.com/appengine/docs/standard/nodejs/

2.4 Amazon Web S

AWS is Amazon's in VS is Amazon Elastic Compute Cloud (EC2) pacity in the cloud, and AWS Elastic Beanstall pplications in the AWS cloud. You simply u matically handles the deployment details of and application health monitoring. Elastic B as the Apache HTTP Server, PHP, and Pyth omcat for Java. Peruse AWS support informa **Setup (NodeJS)** in the Assignment folder on

To learn more about A

https://aws.amazon.co

2.5 Microsoft Azur

Microsoft Azure is a for building, testing, deploying, and mana etwork of Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems. Peruse Azure support information for Node.js in the ungraded assignment **Cloud Setup (NodeJS)** in the Assignment folder on D2L Brightspace.

To learn more about Azure support for Node.js visit this page:

https://docs.microsoft.com/en-us/javascript/azure/?view=azure-node-latest

2.6 Angular

Angular is a toolset for building the framework most suited to your application development. It is fully extensible and integrates well with other libraries. Every feature can be modified to suit your

unique development feature needs. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

For this homework, **Angular 18 is recommended,** but Angular 12, through 17 can be used. However, please note Angular can be difficult to learn if you are not familiar with Typescript and component-based programming.

To learn more about Angular, visit these pages:

https://angular.io/ (v 17 and earlier)
https://angular.dev/ (v 18)

Note: AngularJS (a.k.a Angular 1.0) cannot be used in this project, and will result in a score of zero (0) in the assi

2.7 Node.js

Node.js is a JavaScrip Node.js uses an event-driven, non-blocking ent. Node.js package ecosystem, **npm**, is ries in the world. Peruse Node.js suppo Frameworks on D2L Brightspace.

To learn more about N

https://Node.js.org/en/

Express.js is strongly ode.js web application framework that provid ations. It is in fact the standard server frame nation the class slides covering JS Framewo s like Express that are covered in the lectures ment.

To learn more about Express.js, visit:

http://expressjs.com/

Important Explicit Notes:

1. In this document when you see GCP/AWS/Azure it implies that you can either use Google App Engine, Amazon Web Services or Microsoft Azure Services. However, the CSCI 571 staff will only provide full support for GCP on Piazza and only partial support for AWS and Azure, as none of the TAs use either platform.
2. All APIs calls to tomorrow.io must be done through the Node.js server, functioning as a "proxy." All other API calls should be done through your front-end JavaScript.

3. It is recommended to perform all HTTP calls to your Node backend with either fetch() or the Angular HttpClient (as opposed to using jQuery.ajax() / axios() / XMLHttpRequest()), but any asynchronous XHR functionality will be acceptable.

3. High Level Description

In this exercise, you will need to create a webpage that allows users to search for weather information using the tomorrow.io API and display the results on the same page, below the form.

There are two ways the user can provide the location information for which they are trying to look up the detailed weather information. The first way is by using the fields "Street address" and "City", and/or "State". The second way is by detecting the user's current location.

Dynamic Validation must be performed on all the form text fields – i.e., you must not wait until the submit button is hit to verify that the text input is invalid. More information on this is provided below.

Upon invoking the sea the results pane should be displayed, which y Temp. Chart", and "Meteogram" (see **Fig**

You must display appr a valid response from the API. In the time be esults / error messages you should display a oing something in the background

The webpage should h onality of adding cities and removing cities f should have a Tweet button to share the we nts and details will be explained in the next s

When a user initially c l look like **Figure 1**.

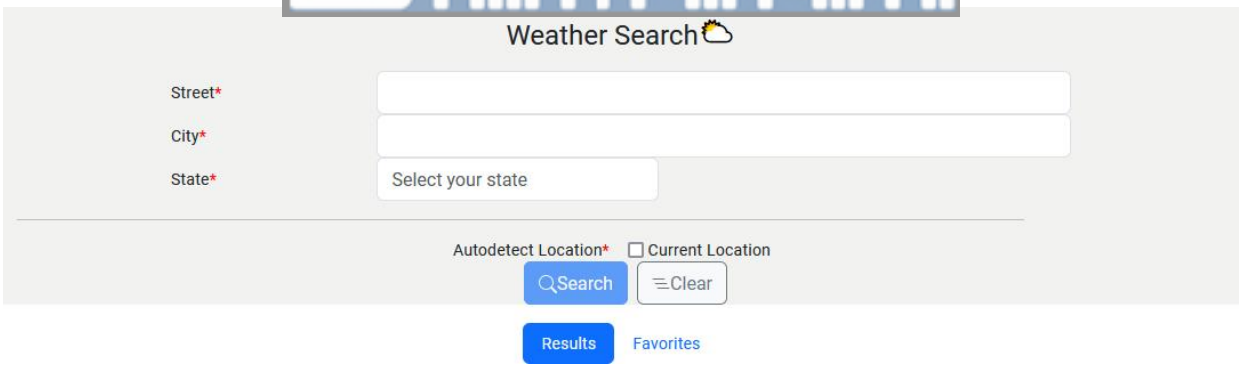


Figure 1. Initial Search Form

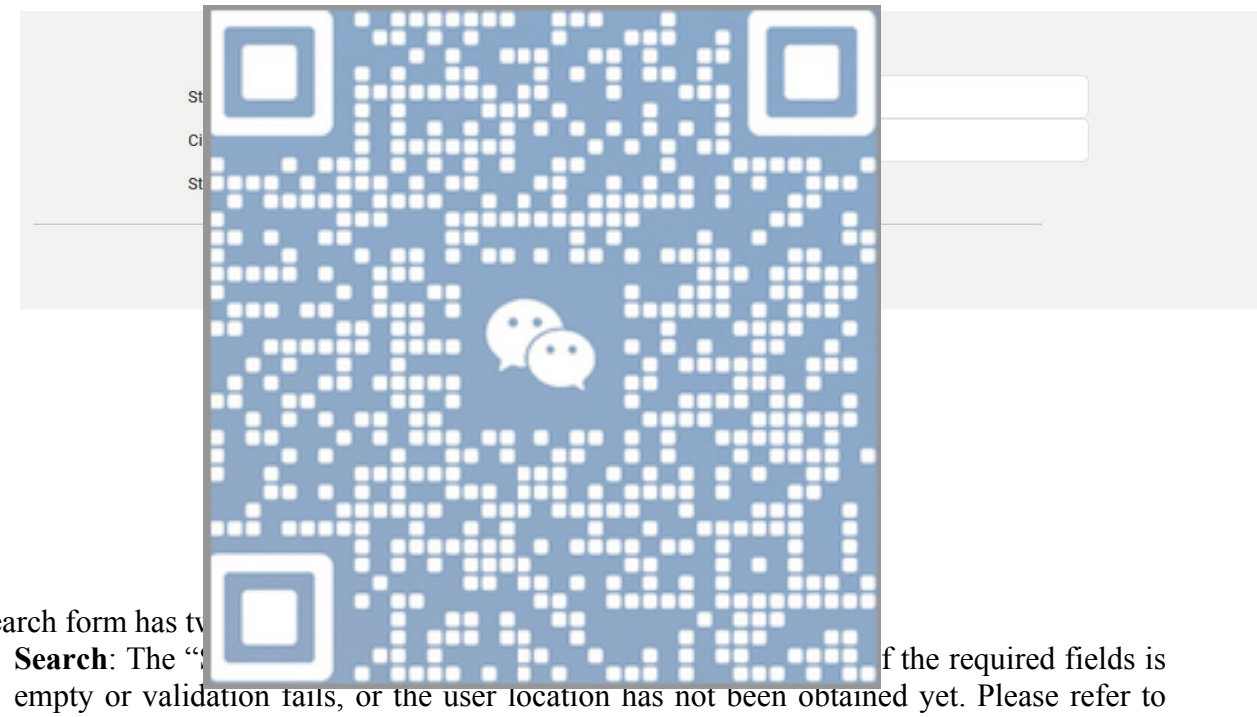
3.1 Search Form

3.1.1 Design

You must replicate the search form displayed in **Figure 1** using a **Bootstrap form**. The form fields are like the ones used in Assignment #2, but their "look" is different, as they are not implemented with default HTML controls.

There are 3 fields in the search form which are **required** if the **Current Location** is not checked:

1. **Street**: Initially, this field is left blank.
2. **City**: This input field should support "autocomplete" which is explained in section 3.1.2. Please note that the user may not necessarily select the autocomplete suggestions. Initially, this field is left blank.
3. **State**: There are multiple options for the user to select from, containing all the names of States of the US as shown in **Figure 2**.



The search form has tw

1. **Search**: The " f the required fields is empty or validation fails, or the user location has not been obtained yet. Please refer to section 3.1.3 for validation details. Please refer to section 3.1.4 for details of obtaining user location.
2. **Clear**: This button must reset the form fields, clear all validation errors if present, switch the view to the Results tab and clear the results area.

3.1.2 AutoComplete

Autocomplete for the **City** field is implemented by using the *Google Place Autocomplete API service*. To learn more about this service, you can go to this page:

https://developers.google.com/maps/documentation/places/web-service/autocomplete

The format of the HTTP request URL, is as follows:

https://maps.googleapis.com/maps/api/place/autocomplete/json?input=[location]&key=[key]

An example of an HTTP request to the *Places Autocomplete API* that searches for matches with the city text field "Los", and returns results in JSON, is shown below:

https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Los&key=[KEY]

To get the **API key** for the Goole Places API service in the above URL. please follow the steps provided in the following link:

https://developers.google.com/maps/documentation/places/web-service/get-api-key

Note:

1. **If the API doesn't work, hide the autocomplete feature. This situation should be handled and r**
2. You can try se restrict suggestions to only cities.

The HTTP response is