High Level Description:

- This project creates a webpage that allows users to search for weather information using the Forecast.io API and display the results on the same page below the form
- The user provides the location information such as Street address, City and State for which they would want to find the detailed weather information or provide their current location.
- Once the user has provided the data and clicks on the Search button, validation must be done to check that the entered data is valid. Once the validation is successful, 3 tabs should be displayed
- The 3 tabs correspond to Current tab, Hourly tab, and Weekly tab.
- The webpage should also support adding cities to and removing cities from the Favorites tab and sharing the weather info with Twitter

Search Form:

- **Design:**
  - There are 3 fields in the search form which are required if the Current Location is not checked:
    - Street
    - City
    - State
  - The search form has two buttons:
    - Search: The "Search" button should be disabled whenever either of the required fields is empty or validation fails, or the user location is not obtained yet
    - 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
- **AutoComplete:**
  - The city field must support autocomplete feature
- **Validation:**
  - The application should check if the "Street" and "City" edit boxes contain something other than spaces or blank. If not, then it is invalid, and an error message should be shown
  - If the Current Location checkbox is checked, then the Street, City and the State controls should be disabled with their values retained
- **Search Execution:**
  - Once the validation is successful and the user clicks on "Search" button, the application should make an AJAX call to the Node.js script hosted on GAE/AWS/Azure
  - The Node.js script on GAE/AWS/Azure will then make a request to forecast.io API to get the weather information

Results Tab:

- This section has 3 tabs:
- **Current Tab:**
  - This tab has a card that displays the following weather information:
    - City name
    - Time zone
    - Temperature
    - Weather summary
    - Humidity
    - Pressure

        - ❖ Wind Speed
        - ❖ Visibility
        - ❖ Cloud Cover
        - ❖ Ozone
        - ❖ State seal
- **Hourly Tab:**
    - This tab provides a bar chart for all the parameters of the weather. There are 6 parameters:
        - ❖ Temperature (Fahrenheit) vs Time (Hourly)
        - ❖ Pressure (Millibars) vs Time (Hourly)
        - ❖ Humidity (%) vs Time (Hourly)
        - ❖ Ozone (Dobson Units) vs Time (Hourly)
        - ❖ Visibility (Miles) vs Time (Hourly)
        - ❖ Wind Speed (Miles per hour) vs Time (Hourly)
    - These details are provided by the DarkSky API
    - The bar charts are shown using Chats.js: https://www.chartjs.org/docs/latest/charts/bar.html
- **Weekly Tab:**
    - This tab provides a range bar chart of the minimum and maximum temperature for the next 7 days in the week
    - On clicking on any one of the range bar chart rows, a corresponding modal window is displayed for that date. The modal window provides the detailed weather information for that date
    - The modal will contain the following information:
        - ❖ Date
        - ❖ City
        - ❖ Temperature
        - ❖ Summary
        - ❖ Icon
        - ❖ Precipitation
        - ❖ Chance of rain
        - ❖ Wind Speed
        - ❖ Humidity
        - ❖ Visibility
- **Favorite button and Twitter button:**
    - The **Favorite button** (star) provides the user the ability to add or remove the city to their favorites tab
    - The **Twitter button** allows the user to compose a Tweet and post it to Twitter. Once the button is clicked, a new dialog should be opened and display the default Tweet

Favorites Tab:

- In the Favorites tab, the favorite cities are listed in a table format. The user can search for weather information for that city by clicking on the City name in the “City” column
- The information displayed in the Favorites tab is saved in and loaded from the local storage of the browser; the buttons in the “Favorites” column of the Favorites tab is only used to remove a city from the list and has a “trash” icon for it to be removed from the Favorite
- The columns in this tab are:
    - **Counter:** Just a counter value indicating the number of cities in the favorites list
    - **Image:** The state seal provided by the Google Custom search API

- **City:** The favorite city the user put in their favorites list. On clicking it should provide the weather details for that city
- **State:** The State abbreviations

Server side:

- The server side is implemented in Node.js and hosted on AWS elastic beanstalk
- **Places Autocomplete API:**
  - To get the API key for this, follow the steps provided here: https://developers.google.com/places/web-service/get-api-key
  - To use the Autocomplete API, the below URL is used: https://maps.googleapis.com/maps/api/place/autocomplete/json?input=<query>&types=(cities)&language=en&key=[YOUR_GOOGLE_AUTOCOMPLETE_API]
  - An example of an HTTP request to the Places Autocomplete API Get Suggestion that searches for the city text field "Los" is shown below: https://maps.googleapis.com/maps/api/place/autocomplete/json?input=LOS&types=(cities)&language=en&key=[YOUR_GOOGLE_AUTOCOMPLETE_API]

- **Getting Google API key:**
  - Go to the Google Developers Console (https://console.developers.google.com/flows/enableapi?apiid=geocoding_backend&keyType=SERVER_SIDE&reusekey=true)
  - Create or select a project
  - Click Continue to Enable the API
  - Go to Credentials to get a Server key (and set the API Credentials)
  - Copy this key and use it in the code of this project in place of <GOOGLE_API_KEY>

- **Google Geocode API:**
  - The Google geocode API call looks like this: https://maps.googleapis.com/maps/api/geocode/json?address=[STREET,CITY,STATE]&key=[GOOGLE_API_KEY]

- **Getting user's current location:**
  - This is obtained using this url: http://ip-api.com/json

- **Getting DarkSky API key:**
  - Go to https://darksky.net/dev
  - Click on "Try For Free".
  - Fill out the form to get the key
  - Copy this key and use it in the code of this project in place of <YOUR_DARKSKY_API_KEY>

- **DarkSky API:**
  - This web service URL is constructed as follows: https://api.forecast.io/forecast/[YOUR_DARKSKY_API_KEY]/[LATITUDE,LONGITUDE]
  - Here LATITUDE and LONGITUDE corresponds to the values extracted from Google Geocode API. The response is in JSON format. Required details are obtained from the parsed JSON file

- **Getting Google Custom Search API:**

- Follow the below documentation to use the Google custom search API key: https://developers.google.com/custom-search/json-api/v1/overview
- The URL to be used further to get state seal is this: https://www.googleapis.com/customsearch/v1?q=[STATE]%20State%20Seal&cx=[YOUR_SEARCH_ENGINE_ID]&imgSize=huge&imgType=news&num=1&searchType=image&key=[YOUR_GOOGLE_CUSTOM_API_KEY]

- All the icons used are from: https://cdn3.iconfinder.com/data/icons
- Initially, the front-end is run on localhost:4000 and server is run on localhost:8081. Once everything is tested, it can be deployed on any cloud platform
- Bootstrap libraries are used to make the web page responsive for mobile screens too