Web Development Project Report

1. Project Report: A detailed project report must also be uploaded to the same GitHub repository(That includes what kind of software tool, APIS, what you tried and success and failure).

The Programming Languages

When creating a website there are 3 fundamental languages that are used: HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) and JavaScript. One additional language I utilised was SASS CSS which had a purpose as I tried to implement the Bootstrap Library but lost value as I dropped Bootstrap.

HTML

HTML is incharge of displaying content on the web page. It is also responsible for the general positioning of content. The <head> tag names tab and can store key data that persists throughout the HTML file. The head is also where you place your imports, in my case the reference to my external stylesheet, the url to the google fonts api, the ajax library installation (although I didn't end up using it as you will find out later) and my Echarts installation. The <body> tag is where the majority of the script is kept as it is the portion the user will be looking at and interacting with. The way I structured my website is as follows:

- I had a container div class to hold all the content. At first it was a bit redundant since it was functioning similarly to the <body> however over time as I added more features, it became a necessity.
- I had a header which contained the title and the search bar (one that is designed for the user to search the desired city to get information on).
- Had an error message div identifier to display a more web based error message rather than a browser base although it did become an annoyance due to some of the backend-esque work done in my project.
- I had a navBar class that was divided into 2 parts: top and sideMenu. The top is for where the logo and another title was kept. The sideMenu held the 3 tabs (as of the time of this report) : Dashboard, Graphs and About.
- Content container was put in place to clearly differentiate between the 3 tabs mentioned in the previous point and space for me to clearly define what content goes where.
- The dashboard content holds all the content that will be displayed when the dashboard tag is active. In my project, when you open the website, there will be cards that are displayed with weather based and air quality based data. Initially, each card will have “Loading…” on them and will change when a city is searched, reflecting the data fetched from the api.
- The graphs content was the location where the data searched will be plotted on some type of visualisation. In my project, there is a line chart that is set to show temperature data by default. There are buttons that allow you to switch between line chart, bar chart or radar chart. I wanted to omit the radar chart as I did not implement it sensibly - it was too difficult for the user to observe data property. There was also a dropdown menu so that you could switch the weather data displayed - between temperature, humidity and wind speed.

- The about content tab existed to inform the user about the website and weather. I left that tab empty as I believed in development it was the least important so decided to deal with it last.
- There is a footer that only has the copyright symbol on it.
- There is a loading indicator that exists independently in the body that appears upon certain conditions such as when the website is fetching data.
- At the bottom of the <body>, there is a reference to my JavaScript file and is placed at the bottom because I learnt that it ensures the website loads correctly before factoring the functions in JavaScript.

CSS

CSS is responsible for the styling of the website. There are a few ways to implement style, but I opted for an external CSS as working in that way makes the most sense to me and it seems a lot easier to track what I am targeting for style. I used SASS in tandem with CSS, as I quite liked the formatting options I had when using SASS. After coding in the scss file, it would be translated into my css file due function - sass --watch scss/style.scss:css/style.css - entered into the command prompt.

JavaScript

JavaScript is responsible for the functionality of the website. It is the reason why the website is interactive and operational. My API keys, stored in the js file have been assigned to variables so that they could be applied to the many different functions throughout the code. I have a multitude of fetch functions as there are at the moment 3 pieces of data that need to be fetched: the coordinates (longitudinal and latitudinal values), the weather data (temperature, humidity, wind speed) and the air quality metrics. I also implemented functions to enable the aforementioned search bar that is held in the header. The search has an auto-complete suggestion feature to help the user find the location they want faster. The suggestions are shown in the form of a dropdown menu. There are functions for the button which initiates the process of making API calls. Caching is another feature that I included as a means to be efficient and as a solution to reduce the amount of API calls. There are a couple of functions that handle the loading indicator to ensure it appears when it has been prompted to do so. Tab switching requires functions in order to change the content displayed on the web page. Echarts needs a function to initialise the visualisations, also the ability to create graphs and extract data to plot onto said graphs.

APIs

There are 4 APIs that I managed to use in this current version of the website:

- Google Fonts API: allows me to use the different designs google fonts has to off
- Tomorrow.io API: this allowed me to get the weather data (temperature, humidity, wind speed)
- AQICN API: this allowed me to get the air quality data(pm25, pm10, no2, co, so2)
- OpenCage API: this allowed me to get the coordinates of cities around the world via their longitudinal and latitudinal values.

Tools & Libraries

- Echarts: this was the library I used to make visualisations, I chose this as it had a large scope of graphs to use that I didn't get a chance to fully utilise. It also greatly supports graph interactivity.
- ChatGPT: this AI helped me learn more about web development and helped me dive deeper into more complex functionalities such as caching. It was also used as a means to help identify errors hindering the overall functionality of the website. This ultimately helps with independent learning for projects.
- YouTube: there a few YouTube videos that I borrowed bits and pieces and applied to my own project.

Tried & Failed/Succeeded

- Bootstrap: As mentioned earlier, I tried to use the Bootstrap library as an additional tool for styling but I dropped it completely because setting it became an issue and got to a point where it was straight up time wasting. In short, I couldn't get it to work.
- WeatherStack API: Initially, I wanted to use this API to get the data I needed for the project but the website was a terrible experience to navigate through and it was not clear what I got for free so I chose to stay away from it.
- Caching: This isn't a tool but a feature I implemented that at first didn't work - in fact was breaking my website - but later became an aspect of value. It helped with reducing the amount of API calls by temporarily storing data that may be reused again in a short period of time.
- Optimisation: This was more of the process of centralising a lot of my functions and reducing the number of functions required to perform a group of tasks. This ultimately supported me when it came to maintenance.
- Github: After some of the disasters I encountered when programming, Github helped me keep track of the changes I was making, preventing me from keeping, adding or changing irrelevant code. Also after repeating the commit, pull, push process I felt myself getting better at using it, so I deem this as some sort of success in terms of personal growth.

Limitations

One big limitation was the APIs. Since I was using the free version of each API, there were a limited number of calls that could be made per day or hour. This at certain points prevented me from actively testing the website.

I as an individual was a limitation as I left myself with less time due to irresponsibility and lack of organisation resulting in a more incomplete product in the end.