Student Name : Marina Salina

Deployed URL : <https://weather-app-ms-as-2.netlify.app/>

See Below for Self-Assessment Rubric

See the last page to see the sections you are aiming for.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Grade Band** | **JavaScript** | **Dev Experience** | **Submission and Deployment** | **Release** | **Release Features** |
| Starter | Understanding of best  practices:  **• let and const**  In my project, I used const for values that don’t change and let for values that do change.  I primarily used const for variable declarations. For example:  const now = dayjs();  const cities = ['berlin', 'amsterdam', 'paris', 'tromso', 'cork', 'copenhagen'];  let is used when the value needs to change later. For example:  let liveRightNowTemp; The temperature updates every hour, so was use let because the value will change.  • **Usage of arrays**  I created an array to store a list of cities:  const cities = ['berlin', 'amsterdam', 'paris', 'tromso', 'cork', 'copenhagen'];  I used forEach() to loop through this array and update the weather data for each city on the page. Inside the main weatherData object, there are many arrays such as:  temperature\_2m\_max;  temperature\_2m\_min;  time etc.  These arrays store daily or hourly values. I also used an array for weekday names to show the correct forecast days.  **• Usage of object**  The Main object I worked with is: weatherData  This object contains nested object for each city like  Amsterdam\_daily, berlin\_daily etc.  Each nested object includes weather information such as temperature, wind speed, and time. These were accessed to update the content on the web page dynamically. | Clearly laid out project  structure – HTML, CSS  and JavaScript  My project uses a clean and modular structure suitable for Eleventy and Nunjucks.  1. \_includes(nunjucks includes):  layout.njk – main layout file,  components: nav.njk reusable nav. bar;  Minus – I don’t have footer bar, cards bar.  2. \_site – output folder generated by eleventy;  2. icons:  logo.png,  sun.png;  4. js – java script files:  bulma.js – handles Bulma CSS framework,  processWeatherData.js – utility to transform weather data,  data - weather\_data.js – Raw weather data,  pages:  city.js JS logic for the city weather page,  dropdowns.js – handles dropdown interactivity,  index.js – JS logic for the dashboard.  5. .eleventy.js – Eleventy configuration file,  6. city.njk – Nunjucks template for a city weather app,  7. index.njk – Nunjucks homepage,  8. package-lock.json,  9. package.json - scripts,  10. Read.me - instruction. | Zip file to moodle  A ZIP file containing the full project was created and submitted via Moodle. | POC – 1 - City Focus 1 1 City with today’s weather  The initial goal was to display today’s weather for a single city — Berlin. The project began by hardcoding the data using Bulma components to create a basic layout and styling.  A median function was initially implemented to calculate and display average temperatures. | • Weather Code Mapped to Correct Weather  • Display City Name  • Clear it is today’s weather  • Max Temp and Wind  All this punkts is correct and display: city name, and today weather. |
| **baseline** | • Add and Remove  elements from the  DOM without the need  for a screen refresh  I used innerHTML to change the temperature and wind speed values when the city is selected. I also used createElement and appendChild() to add new weather cards for each day in the weekly forecast (const dayWeatherBox = document.createElement("div"). | DRY - Very little  Repetition of HTML, JS  and Styles  In this project, I didn’t focus on following the DRY principle, which is not ideal and is something I can improve on in the future. I understand that by following the DRY principle, my code can become cleaner, shorter, and easier to update. | + github repository link  + github commit history  OR  bulma breadcrumb  component  I used GitHub for this project, but I understand that I still have a lot to learn in order to use it confidently and effectively. While working on this, I also learned how to use the terminal to push my project to GitHub. I used three basic commands: git add ., git commit -m "", and git push — which helped me upload my files and include commit messages to track changes.  You can view my GitHub repository and commit history here:  <https://github.com/Marina19761107/weather/tree/main> | POC – 2 – City Focus 2 + current hour’s weather  The initial goal was to display today’s weather for a single city — Berlin. The project began by hardcoding the data using Bulma components to create a basic layout and styling.  A median function was initially implemented to calculate and display average temperatures. However, this function was later removed. Instead, the implementation was updated to use hourly weather data to display the current temperature for today, based on the user's local time. | • Added current hours forecast  Mapped hourly weather cod  • Correctly mapped hourly weather code. |
| **good** | • Understanding of  application name  spacing and scopes  In this project, I used clear variable names like weatherData, currentHour, processedData, and cityToday, which helped make my code easier to read and understand. I also used scope correctly—for example, variables like cityToday, dropdown, and trigger were declared inside loops, so they only worked within those blocks, keeping the logic clean and avoiding conflicts. Although I didn’t use a namespace. | Layered JavaScript  Architecture – MVC  In my project, I didn’t think about using the MVC (Model-View-Controller) architecture, but I think it’s present in my project.  **Model**: This is the weather data, stored in weather\_data.js.  **View**: The view is the user interface (UI), which displays the weather data. It’s managed through HTML templates (like index.njk and city.njk), and JavaScript updates the content.  **Controller**: The controller handles the logic. Functions like processWeatherData() and event listeners for interactions (like dropdown clicks) control the data flow and update the UI. | + github tags  OR  bulma dropdown component  to select something (city?)  I used Bulma's dropdown in my project, and the functionality is handled in the dropdown.js file, while the layout is defined in nav.njk. | POC – 3 – City Focus 3 + 7 days summary | • Clear and concise summary for the 7 days  • Weather code correctly mapped  • Additional weather information  In my project, I display a 7-day weather forecast for a selected city. Using data from the processWeatherData() function, I extract daily temperature and wind speed details and display them on the UI. The weather information for each day, including maximum and minimum temperatures, is shown in separate cards for each day of the week. |
| **excellent** | • Well maintained utility  file(s)  In this project, I created and used utility functions like processWeatherData() and getCurrentDay() to keep my code organized  • Clear and concise  loading of javascript  In this project, I organized my JavaScript files into a folder to maintain clarity and structure. I have three separate files: city.js, dropdown.js, and index.js.  • Functions utilized  Correctly  All functions are used correctly to display weather information for 7 days across six cities  Clearly laid out project structure – HTML, CSS  and JavaScript  In this project, I organized my files in separate folders for JavaScript, styles, and templates. My JavaScript code is split into multiple files. Instead of regular HTML files, I used .njk template files (like index.njk, city.njk, and components in the includes folder), which helped keep the layout reusable and clean. My CSS is managed using Bulma. | The README file is well-organized and contains all the information needed to help users get started with the app. | + Manual upload to Netlify  For the manual upload, I used the structure created by Eleventy (\_site folder) and manually uploaded it to Netlify. You can view the deployed app at <https://weather-app-ms-as-2.netlify.app/> | Release -1  + Dashboard | • Focus on user experience  Users can easier navigate between the dashboard and city views  • Ability to navigate to city view  The city view page is designed to present a clear layout of the city's weather, such as current temperatures, wind speeds, and forecast data for the upcoming days. Additionally, from the city view page, users can use the dropdown to switch to another city without having to return to the dashboard.  • Ability to navigate back to dashboard  A button at the top allows users to go back to the main page and view weather details for other cityes  • Usage of URL paths and parameters  I used URL paths and parameters to manage the navigation between different pages (dashboard and city view). The city-specific data is dynamically loaded based on the URL, allowing users to directly access weather data for any city by changing the URL parameter. |
| **outstanding** |  |  |  | Release – 2 + Configure and Persist User Preferences |  |
| **amazing** |  |  | - | Release – 3 + Build out your own features |  |

**Reflection Section**

* Please highlight anything of mention that you think should get marks that is difficult to convey in the rubric above
  + Here please
* Source of text, images, icons or anything else that was used

1. Weather Data: The data was taken from raw data inside the file weather\_data.js, and to understand this data, the API documentation was used.
2. Libraries and Frameworks: [Bulma CSS Framework](https://bulma.io); Day.js Library; Eleventy; Github
3. Icons: I used 2 icons: a sun icon (represents sunny weather conditions in the UI) and a logo icon
4. Project Hosting: [Netlify Deployment](https://weather-app-ms-as-2.netlify.app/)

A diagram with text and images

Description automatically generated with medium confidence