# A short reflective report on the strengths and weaknesses of my app's architecture

The Weather App is a web application developed by using HTML, CSS, and JavaScript to get the exact time of all around the world. The weather application will provide users with real-time weather information, forecasts, and other weather-related data, which can help them make better decisions about their day-to-day activities and many more. This report assesses the strengths and weaknesses of the application, considering its design, functionality, and user experience. First of all, starting with the introduction of anatomy of webpages. HTML (Hypertext Markup Language) serves as the foundation for structuring the content of the Weather App. It defines the elements and layout of the web page. The language has tags, elements and attributes and without styling and interaction, HTML does not look useful. That's why CSS (Cascading Style Sheets) is employed for styling and presenting the content defined by HTML, enhancing the visual appeal of the Weather App. The CSS has selectors, properties and values and styling is done by targeting the given selector.CSS Selectors can either be element, classes, ids or attributes. This allows us to add some functionality and interaction to web application. And finally, we use JavaScript to utilized for adding dynamic behavior to the Weather App, fetching real-time weather data and updating the background accordingly. JavaScript is a scripting language, which means you don't have to render or compile it like Java. It is a client-side scripting language. It runs on browser itself. The strengths of the Weather App lie in its dynamic background feature, where JavaScript leverages realtime weather conditions to dynamically alter the backdrop, creating an immersive visual experience. The integration of the OpenWeatherMap API strengthens the app's reliability by providing accurate and upto-date weather information. Additionally, the user-friendly design, crafted with HTML and CSS, fosters a well-structured and visually appealing interface. The modular nature of these technologies allows for easy scalability and maintenance, accommodating future enhancements seamlessly. However, the Weather App does exhibit certain weaknesses. User interaction is somewhat limited, potentially hindering user engagement. The app's functionality heavily relies on the OpenWeatherMap API, introducing a dependency that might affect performance and availability. While the dynamic background is a strength, the use of static images may not fully capture the nuances of certain weather conditions. Furthermore, there's a need for thorough cross-browser testing to ensure consistent performance across various browsers. In conclusion, the Weather App showcases a robust blend of HTML, CSS, and JavaScript, offering a visually appealing and functional platform for accessing weather information. Addressing the identified weaknesses, such as enhancing user interaction and mitigating external API dependencies, will be crucial for refining and elevating the overall user experience. Ongoing development and strategic improvements can pave the way for a more resilient and feature-rich application.

## Prototype 2

MySQL is used to store data in tables that map to objects. Each table has a schema defining what columns each row of the table will have. Developers can reliably store and retrieve many data types, including text, numbers, dates, times, and even JSON. Prototype 2 uses the MySql database and PHP to enable server-side caching in addition to using HTML, CSS, and JS to create the basic structure. This web application provides a basic and functional weather forecast experience. It fetches real-time weather data from a local PHP endpoint ("data.php") to retrieve the weather information stored in the database. This allows for the data to also be accessed without rendering to the 'openweathermap.org' API. By using the database, it is capable of displaying the data of the past week. In the PHP code of this app, SQL queries are used to insert and update the required weather data of cities in the database. The data in the database includes columns for all the different information that needs to be displayed in the past week's weather column. Also, MySql errors that may be encountered when inserting data, updating data, etc. are handled in the php code itself. The strength of this app's improved architecture is that it can be used even without internet access. Also, errors are handled well. This prevents the app from suddenly crashing and creates a user-friendly interface that is clear and organized. It also displays the data of the past week of the default city through the use of dynamically created cards, unlike the previous prototype. The code is organized into functions which promotes a modular structure that is easier to understand and maintain. This architecture has weaknesses as well. One of the major weaknesses of this architecture is that browser caching, and client-side caching of the weather data are not done. This gives the app access to the network or server even when it is not required. Also, the web app is not secure as API keys are not handled properly. Additionally, this architecture doesn't allow users to create their accounts with their data history which could improve user interaction.

- By Samikshhya Gurung (2417750)