The developed weather application connects client-side technologies (HTML, CSS, JavaScript) through a PHP implementation of a MySQL relational database structure. In the selected structure PHP serves as a dual-role system that functions as both an API client by retrieving information from OpenWeatherMap and an API server that responds to frontend requests.

Strengths

The system demonstrates its main strength through its optimized approach to API consumption. A database caching mechanism within the application retrieves data from OpenWeatherMap API only when stored data for a specific city exceeds two hours in age. The system prevents excessive external API requests through this approach which allows it to stay within the usage restrictions of the provider. The system delivers better performance because it removes procurement delays caused by multiple repeated network requests.

The application stands out because it maintains different functional areas separately from each other. Users interact with the frontend for both interface and interactions whereas the backend manages data retrieval and data processing and data storage functions. The modular approach of this design structure enables both easy code maintenance and clear presentation of the software base. Calling MySQL to manage structured data storage enables the system to keep information permanently stored while creating possibilities for adding historic weather tracking capabilities in the future.

Weaknesses

The application shows good functionality yet presents several architectural limitations. The application suffers from an inadequacy of systems that clean out outdated data storage. Performance and storage performance decrease as the weather table accumulates superfluous data from an increasing number of cities queried. The system needs automatic data management strategies that would handle outdated data storage.

A disadvantage of the application is its limited capability to handle errors. The frontend displays only basic feedback to users when API failures or data incompleteness occurs from network problems and incorrect city input. The system performance together with the user experience can improve through additional error reporting features and user notification mechanisms.

A procedural PHP setup suits educational purposes yet fails to deliver the organizational benefits and replication abilities which contemporary frameworks using Laravel and MVC offer.