Reflective Report on App Architecture

The architecture of the weather Forecast application demonstrates several strengths and weaknesses that influence its functionality and scalability.

Strengths:

1. Separation of concerns: The app follows a clear separation of concerns. Html Provides a clean structure, CSS handles styling, and JavaScript manages interactivity and API integration. The php backend efficiently handles data retrieval and storage.
2. Database caching: The use of a database to store weather data reduces the dependency on external Api calls. This improves performance and minimizes potential costs associated with frequent API usage, particularly when accessing the same city multiple times within two hours.
3. Responsive Design: the app is visually appealing and user-friendly, with responsive css ensuring compatibility across different devices.

Weaknesses:

1. Hardcoded Dependencies: the api key and database credentials are hardcoded in the code, posing a security risk. This could be mitigared using environment variables.
2. Limited testing: There's no mention of automated testing or error logging, Which are essential for city names or real-time updates, which could improve usability.
3. Single Endpoint: the app relies on a single php endpoint for both caching and data retrieval . this could become a bottleneck and may need to be modularized as the application grows.

Conclusion:

While the architecture effectively delivers a functional weather app, enhancing security, scalability, and advanced features will be essential to prepare it for real-world deployment.