Reflective Report on the Weather App’s Architecture

STRENGTH

The weather app effectively performs smaller jobs because it has several useful features. The application adopts a straightforward structure because PHP processes data retrieved from MySQL database and OpenWeather API. The implementation methodology makes data storage and retrieval effective for weather data while replacing stale data with updated information on demand. This database includes a straightforward structure that connects each entry through city names and date definitions as main keys thus maintaining easy weather information retrieval. Users can set any city name through a URL parameter without needing modifications to the app's core code. The system conducts database research for current weather data in advance of API contact to optimize efficiency and minimize pointless calls. The weather application verifies the database for fresh data before using the OpenWeather API API calls. Recent and valid data found within the database prompts the app to continue using it thus preventing extra API calls and improving total application performance. Recorded weather information from the OpenWeather API and the system's built-in data allows the platform to provide users with dependable real-time weather projections because of its easy-to-use design.

WEAKNESS

Several conditions may affect both security and performance features of this application. A major security issue arises when user input interacts with SQL queries because it establishes an opportunity for SQL injection attacks. The introduction of prepared statements can eliminate this security risk. A big safety weakness exists because the API key is included as a literal string directly in the source code. Security of the key requires storing it into an environment variable or separate config file which belongs outside the main application code base. Network issues affecting the OpenWeather API will cause the app to stop working properly. The application requires

additional error management systems to become more stable in the face of failures. The present app functionality remains strong for minimal use but shows potential for poor performance as user numbers increase. Independently stored cache data would enhance scalability by improving performance under heavy usage. The application suits its purpose but needs security upgrades as well as reliable performance and ability to scale.