Test Plan: Weather Forecast Application

Objective:

The objective of testing the weather forecast application is to ensure that the application functions as intended, displaying accurate weather information as per the OpenAPI schema.

Test Environment:

- Technology: Docker, VS Code, Postman, Swagger
- Operating Systems: Windows, macOS, iOS, Android
- Devices: Desktop, mobile (iOS, Android), different browsers
- Network: Test across different network speeds (3G, 4G, 5G, Wi-Fi)

Test Scenarios:

1. **Functional(API) Testing:**
   1.1. Validate that the API response conforms to the schema provided in the OpenAPI documentation.
   1.2. Check the response for various weather conditions (mild, chilly, warm, balmy, scorching).
   1.3. Verify the correctness of the temperature unit conversion from Fahrenheit to Celsius.

2. **Non-functional Testing**
   2.1. Performance Testing:
      2.1.1. Test the application's performance by analyzing the API response time
   2.2. Compatibility Testing:
      2.2.1. Test the application across different browsers (Chrome, Firefox, Safari, Edge) and ensure functionality.
      2.2.2. Test on different operating systems (Windows, macOS, iOS, Android) to check for any OS-specific issues.

Test Cases:

Test Case 1: API Response Verification

Objective: To validate if the API response matches the OpenAPI schema.

Steps:

1- Send a GET API request to the application server.

2- Ensure the response code is as expected (e.g., 200 for success).
3- Validate that the response contains the expected fields as per the schema
4- Check for the correctness of the data types (ex: temperature being integer).

Test Data:

- Request URL: http://localhost:8080/WeatherForecast
- Expected fields:
  date → string
  temperatureC → integer
  temperatureF → integer
  summary → string
- Expected date format: YYYY-MM-DD (2023-11-04)

Test Case 2: Unit Conversion Verification

Objective: To verify the temperature data is converted between Celsius and Fahrenheit correctly. Use temperature conversion formula for crosschecking.

Steps:

1- Send a GET API request to the application server.
2- Ensure the response code is as expected (e.g., 200 for success).
3- Validate that conversion is done according to temperature conversion formula.

Test Data:

- Request URL: http://localhost:8080/WeatherForecast
- Temperature Conversion Formula C = (F – 31) / 1.8

Test Case 3: Performance Testing

Objective: To evaluate the API's behavior under load by sending multiple concurrent requests.

Steps:

1- Simulate a load by sending multiple simultaneous API requests for weather data retrieval.
2- Monitor and record the API's response time and its ability to handle concurrent requests.
3- Validate that API Gracefully handle concurrent requests without significant increase in response time or failures.

Test Data:

- Request URL: http://localhost:8080/WeatherForecast
- Exact time intervals TBD instead of significant increase in response time statement.

Conclusion:

These test cases cover a range of scenarios and aspects to ensure comprehensive testing of the weather forecast application, addressing API conformity, functionality, UI, performance, and compatibility.