SOFTWARE TEST APPROACH   
FOR   
Search Weather Feature of “OpenWeatherMap” site

TABLE OF CONTENTS

Section Page

Contents

[1 SCOPE 2](#_Toc83036106)

[1.1 WEB 2](#_Toc83036107)

[1.2 API 2](#_Toc83036108)

[2 SCOPE 3](#_Toc83036109)

[2.1 WEB 3](#_Toc83036110)

[2.2 API 3](#_Toc83036111)

[3 TYPES OF TESTING APPLIED 3](#_Toc83036112)

[3.1 Look and Feel: 3](#_Toc83036113)

[3.2 Functionality: 4](#_Toc83036115)

[3.3 Requirement Traceability: 4](#_Toc83036116)

[3.4 Equivalent-class partitioning and Boundary Value Analysis: 4](#_Toc83036117)

[3.5 Exploratory/AdHoc Testing: 5](#_Toc83036118)

# SCOPE

## WEB

* Given the application under test (AUT): https://openweathermap.org/.
* Feature to test: Search weather in your city

## API

* API automated tests for the feature "search weather in your city" API.
* API end point: api.openweathermap.org/data/2.5/weather?q={city name},{state code}&appid={APIkey}

# SCOPE

## WEB

* Since document for “Search weather” feature is not available, I assume the current AUT at 20-Sep-2021 as baseline AUT and I will use the sense of user experience to suggest for the AUT improvement.

## API

* API Spec for Current weather data – OpenWeatherMap: (<https://openweathermap.org/current#name>)

# TYPES OF TESTING APPLIED

## Look and Feel:

* GUI match to wired frame (if wired frame is unavailable, I should focus to layout, design caused good feeling of AUT usage.
* GUI resize smoothly and responsive.
* API endpoint should be named friendly and systematically so that users remember and use easily.
* API returned result should be designed friendly so that users get information effectively for their work.

## Graphical user interface, application, website Description automatically generated

## Functionality:

* Assess “search weather of a city” feature perform correctly as design.
* Focus to non-functionality was not mentioned in design, and it will cause unpleasure feeling of user when use AUT.(EX: transition of controls smoothly, respond of search result is quick enough, notification friendly when user input incorrectly and help them complete with clearly instruction …)
* API returned result should work as document mentioned in <https://openweathermap.org/current#name> .

## Requirement Traceability:

* Test design must covered all AUT requirements and make requirement traceability matrix effectively so that whenever changing is tracked, updated accordingly and effectively.
* User acceptance test should be referred strictly to validate AUT performed as contract.

## Equivalent-class partitioning and Boundary Value Analysis:

* Equivalent-class partitioning and Boundary Value Analysis is powerful test design technique to discover common defects.
* Help to reduce test cases with good test data and support data driven testing effectively.
* Refer AUT document carefully to get quality test data.

## Exploratory/AdHoc Testing:

* Exploratory testing is a process of examining the product and observing its behavior, as well as hypothesizing what its behavior is.
* Going from known test cases to unknown area help us discover AUT mystery area not covered when design. (EX: various language all around the world is hard be covered when using for testing this feature )
* Finding bugs is ultimate goal so that put tester in user shoes to discover defects cause unpleasure feeling of user.