**TEST APPROACH**

**Feature**: Search weather in your city

**1.TEST ANALYSIS**

**Application under test (AUT)** - search weather in your city

**Graphical user interface, text, application

Description automatically generated**

**Test Basis:**

* Input : the city's name, comma, 2-letter country code (ISO3166)
* Output: The weather info of the input city

Graphical user interface, text, application, email, website

Description automatically generated

Related page: View detail of weather in a city

Graphical user interface, application, email

Description automatically generated

**Valid data:**

* name of city (town, village), comma, 2-letters country code
* comma, 2-letters country code
* name of city (town, village), comma, 2-letters country code: which include white space

**Invalid data:**

* invalid format
* non-existing city name
* Numbers (1,2,3,4,5,6,7,8,9,0)
* Special characters, except comma

BOUNDARY VALUES:

* 1 letter
* 2 letters
* 3 letters
* 500 letters
* 500 letters

**STATE TRANSACTION ANALYSIS:**

* Check response valid data entered and send is Enter key pressed
* Check response invalid data entered and send is Enter key pressed

**Designed API**

* URL: <domain>/data/2.5/find?q=<search keyword>&appid=<API Key>
* Required parameter:
* q: input query (city name, comma, country code)
* appid: API Key
* Optional parameters:
* type: like
* sort: population
* cnt: Number of cities around the point that should be returned. The default number of cities is 5, the maximum is 50
* units: metric, imperial, default
* Request header:

Access-Control-Allow-Credentials: true  
Access-Control-Allow-Methods: GET, POST  
Access-Control-Allow-Origin: \*  
Connection: keep-alive  
Content-Encoding: gzip  
Content-Type: application/json; charset=utf-8  
Date: Tue, 29 Dec 2020 08:34:05 GMT  
Server: openresty/1.9.7.1

* Response body:

{

"message": "accurate",

"cod": "200",

"count": 1,

"list": [

{

"id": 7302259,

"name": "Glenwood",

"coord": {

"lat": -33.7331,

"lon": 150.9282

},

"main": {

"temp": 294.92,

"feels\_like": 296.3,

"temp\_min": 294.82,

"temp\_max": 295.37,

"pressure": 1014,

"humidity": 70

},

"dt": 1609304944,

"wind": {

"speed": 0.89,

"deg": 123

},

"sys": {

"country": "AU"

},

"rain": **null**,

"snow": **null**,

"clouds": {

"all": 91

},

"weather": [

{

"id": 804,

"main": "Clouds",

"description": "overcast clouds",

"icon": "04d"

}

]

}

]

}

* Methods accepted: GET, POST

**2.TEST LEVEL requires:**

* Unit test
* API test
* Integration test: checking how backend & frontend work together
* Compatibilities test: checking on multiple browser types & responsive ability
* Acceptance testing

**3.TEST TYPES:**

* Manual UI testing - UI automation testing
* Explore testing
* API testing - API automation testing
* Stress testing

**4.Test ENVIRONMENT:**

* Window OS, Mac OS
* Browsers: Chrome/ Firefox/ Safari

**5.How and who run test:**

* Developer implements unit test and will run unit test whenever they have new changes on code
* QA prepare API test cases and implements API automation test using Mock-API and then will execute automated API test when API is ready on DEV env
* QA or Developer will execute automated API test on CI when there is new change from backend site vs each environment: DEV/ STAGING/ PRODUCTION
* QA prepare UI test cases which includes integration test. Developer will follow this test suite for checking his code before merging it.
* QA will execute UI test and explore testing on DEV/ STAGING env
* QA will run regression test on each environment when there is new change

**Team responsibilities in the testing flow:**

* QA has responsibilities for bug report, bug tracking on JIRA
* Developer has responsibilities for fixing bugs; deploy the fix into each environment.
* QA has responsibilities for verify and update status of bug in each environment

**6. Test coverage**

1. Unit test code coverage
2. Feature testing metrics :

* Requirements coverage

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

* Test cases by the requirement

**Graphical user interface, application

Description automatically generated**

* Requirements without Test Coverage

**Exit Criteria:**

* Ensuring all critical Test Cases are passed
* Achieving complete Functional Coverage
* Identifying and fixing all the high-priority defects
* Fixing all the ‘Show Stopper defects’ or ‘Blockers’ and ensuring that none of the identified Critical/Severity 1 defects are in Open Status
* Re-testing and closing all the high-priority defects to execute corresponding Regression scenarios successfully