Bao Nguyen Tran Nguyen

Ntnguyenbao92@gmail.com

VDC Tehcnical test

Part 1: Test Design and Bug Challenge

Contents

[I. Feature introduction 2](#_Toc67164849)

[II. Assumption requirement 2](#_Toc67164850)

[III. Test approach 2](#_Toc67164851)

[IV. Test cases 3](#_Toc67164852)

[A. Unit Test 3](#_Toc67164853)

[B. Functional test 6](#_Toc67164854)

[C. Performance test 9](#_Toc67164855)

[D. Usability test 11](#_Toc67164856)

[V. Bugs Report 11](#_Toc67164857)

# Feature introduction

“Search weather in your city” is the feature which allow user search the current weather statistic of a specific city. It provide user with Web UI where user can input searching string and get the result base on their inputted string. Moreover, this feature also allow users get current weather statistic via REST API.

# Assumption requirement

**Req-001:** When user input a string and type Enter, the page which displays search result is display. If the inputted City is available in system, found City is displayed, otherwise, a message will be displayed to inform user that inputted City cannot be found

**Req-002**: The searching string has the format <city name>, <2-letter country code (ISO3166)> One of two parameters can be missed Example – “London, GB” or “New York” or “,VN”

**Req-003:** the user can search weather of specific city via REST API with City name is parameter

**Req-004:** Any keyboard character is accepted by search field, including alphabet letters, numeric and special characters (e.g.: ~,!,#,$,%,^,&,\*,(),\_,+,=,-,`,|)

**Req-005:** If user only input city name into search field, a city is return as searching result only if all characters of its name match inputted city name but ignore case (A=a, B=b)

**Req-006:** If user only input 2-letter country code into search field (ex: “VN”), all city which belong to that country are return as searching result

**Req-007:** The search field always can be found even if user change browser windows size

**Req-008:** All cities which match searching condition will be displayed in link format in search result page. If users click a link, they will be redirected to a page which contain detail for selected city

**Req-009:** the feature search weather in your city work on Chrome and Firefox

**Req-010**: In case there is no network issue, the search result have to be return within 10 seconds

**Req-011:** this feature can be handle at least 500 concurrent searching request. Each request get the response within 10 seconds

# Test approach

The following testing will be done for this feature

**Unit test**: Test REST API which use to query weather statistic with positive and negative scenarios

* Positive scenarios:
  + Input correct city name
  + Input correct country code
  + Input correct city name and country code
* Negative scenarios:
  + Input incorrect city name and country code
  + Input special characters as searching string
  + Input incorrect API key
  + Input empty search string
  + Input very long string

**Functional test**: Test functions of the feature on Web UI with below scenarios by using both **Firefox** and **Chrome**

* Positive scenarios:
  + Input correct city name into search field
  + Input correct country code
  + Input both correct city name and country code
* Negative scenarios:
  + Input incorrect city name or country code
  + Input special character
  + Input very long string (larger than 200 characters)
  + Input empty string

**Non-functional test**

* Performance test:
  + Load test: Test response time of 500 concurrent searching request in 1 hour, increase number of concurrent request if the server still work stably to find the maximum number of simultaneous requests
  + Stress test: Test response time of server under stress (number of request is higher than maximum supported number) in 1 hour and after stressful hour
* Usability test:
  + Test visibility of search field in case the browser windows size is changed

# Test cases

## Unit Test

**Test Case ID**: Unit-001

**Objective:** Validate REST API with correct city name

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read list of city names from data file
3. Select a random city name from above list
4. Read expected result of selected city from data file
5. send GET request “api.openweathermap.org/data/2.5/weather?q={city name}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
6. Validate responded JSON file by comparing actual result and expected result from data file

**Expected result:** Receiving JSON format response which contains weather statistic of all cities which match searching name

**Test Case ID**: Unit-002

**Objective:** Validate REST API with correct country code

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read list of country code from data file
3. Select a random city name from above list
4. Read expected result of selected country from data file
5. send GET request “api.openweathermap.org/data/2.5/weather?q={country code}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
6. Validate responded JSON file by comparing actual result and expected result from data file

**Expected result:** Receiving JSON format response which contains weather statistic of all cities which belong to searching country

**Test Case ID**: Unit-003

**Objective:** Validate REST API with correct country code and city name

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read list of cities and their country codes from data file
3. Select a city name from above list randomly
4. Read expected result of selected city from data file
5. send GET request “api.openweathermap.org/data/2.5/weather?q={country code}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
6. Validate responded JSON file by comparing actual result and expected result from data file

**Expected result:** Receiving JSON format response which contains weather statistic of city which match searching condition - city name is correct and country name is correct

**Test Case ID**: Unit-004

**Objective:** Validate REST API with incorrect country code and city name

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read expected message from data file
3. Generate a random string for city name and country code
4. send GET request “api.openweathermap.org/data/2.5/weather?q={generated string}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
5. Validate responded JSON file by comparing actual result and expected message from data file

**Expected result:** Receiving JSON format response which contains message “city not found”

**Test Case ID**: Unit-005

**Objective:** Validate REST API with special characters

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read expected message from data file
3. Generate a random string which contain special characters for city name and country code
4. send GET request “api.openweathermap.org/data/2.5/weather?q={generated string}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
5. Validate responded JSON file by comparing actual result and expected message from data file

**Expected result:** Receiving JSON format response which contains message “city not found”

**Test Case ID**: Unit-006

**Objective:** Validate REST API with incorrect API key

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Generate an incorrect API key
2. Read expected message from data file
3. Read list of cities from data file
4. Select a city from above list randomly
5. send GET request “api.openweathermap.org/data/2.5/weather?q={city name}&appid=[{incorrect API key}](https://home.openweathermap.org/api_keys)”
6. Validate responded JSON file by comparing actual result and expected message from data file

**Expected result:** Receiving JSON format response which contains message “Invalid API key. Please see http://openweathermap.org/faq#error401 for more info.”

**Test Case ID**: Unit-007

**Objective:** Validate REST API with empty search string

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read expected message from data file
3. send GET request “api.openweathermap.org/data/2.5/weather?q=&appid=[{API key}](https://home.openweathermap.org/api_keys)”
4. Validate responded JSON file by comparing actual result and expected message from data file

**Expected result:** Receiving JSON format response which contains message “Nothing to geocode"

**Test Case ID**: Unit-008

**Objective:** Validate REST API with very long string (more than 200 characters)

**Requirement:** Req-003

**Tag:** unit

**Test procedure:**

1. Read API key from data file
2. Read expected message from data file
3. Generate random very long string (more than 200 characters)
4. send GET request “api.openweathermap.org/data/2.5/weather?q={generated string}&appid=[{API key}](https://home.openweathermap.org/api_keys)”
5. Validate responded JSON file by comparing actual result and expected message from data file

**Expected result:** Receiving JSON format response which contains message “city not found”

## Functional test

**Test Case ID**: Func-001

**Objective:** Validate searching weather by city name via Web UI

**Requirement:** Req-001, Req-002, Req-005, Req-008, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Read the list of cities from data file
5. Select a city from above list randomly
6. Input a random correct city name (e.g. ho chi minh)
7. Press Enter/Click searching icon 
8. Read expected result of selected city from data file
9. Wait for searching result is displayed (timeout 10 seconds)
10. Validate the result match searching condition by comparing with expected result from file
11. Click a link in list of cities
12. Wait for detail page is display

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #9: links of matching city are displayed within 10 seconds
  + Step #10: list of displayed cities matches searching condition
  + Step #12: detail page of selected city is displayed

**Test Case ID**: Func-002

**Objective:** Validate searching weather by 2-letters country code via Web UI

**Requirement:** Req-001, Req-002, Req-006, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Read the list of countries code from data file
5. Select a country from above list randomly
6. Input a random correct country code (e.g. “,US” , “,vn”)
7. Press Enter/Click searching icon 
8. Read expected result of selected city from data file
9. Wait for searching result is displayed (timeout 10 seconds)
10. Validate the result match searching condition by comparing with expected result from file

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #9: links of cities which belong to searching country are displayed within 10 seconds
  + Step #10: list of displayed cities matches searching condition

**Test Case ID**: Func-003

**Objective:** Validate searching weather by city name and country code combination via Web UI

**Requirement:** Req-001, Req-002, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Read the list of cities and their country code from data file
5. Select a city from above list randomly
6. Input a random correct city name and its country code(e.g. ho chi minh,vn)
7. Press Enter/Click searching icon 
8. Read expected result of selected city from data file
9. Wait for searching result is displayed (timeout 10 seconds)
10. Validate the result match searching condition by comparing with expected result from file

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #9: links of cities which match searching condition are displayed within 10 seconds
  + Step #10: list of displayed cities matches searching condition

**Test Case ID**: Func-004

**Objective:** Validate searching weather by incorrect city name or country code via Web UI home page

**Requirement:** Req-001, Req-002, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Generate a random searching string
5. Input a random incorrect city name
6. Press Enter/Click searching icon 
7. Wait for searching result is displayed (timeout 10 seconds)
8. Validate the result message “Not Found” is displayed

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #7: Search result page is displayed within 10 seconds
  + Step #8: Message is “Not Found” is displayed

**Test Case ID**: Func-005

**Objective:** Validate searching weather by string which contains special characters via Web UI

**Requirement:** Req-001, Req-004, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Generate a random searching string which contain special character(e.g. ~!@#$%^&\*)
5. Input a random incorrect city name
6. Press Enter/Click searching icon 
7. Wait for searching result is displayed (timeout 10 seconds)
8. Validate the result message “Not Found” is displayed

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #7: Search result page is displayed within 10 seconds
  + Step #8: Message is “Not Found” is displayed

**Test Case ID**: Func-006

**Objective:** Validate searching weather by very long string (more than 200 characters) via Web UI

**Requirement:** Req-001, Req-004, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Generate a random very long string which contain more than 200 characters
5. Input a random incorrect city name
6. Press Enter/Click searching icon 
7. Wait for searching result is displayed (timeout 10 seconds)
8. Validate the result message “Not Found” is displayed

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #7: Search result page is displayed within 10 seconds
  + Step #8: Message is “Not Found” is displayed

**Test Case ID**: Func-007

**Objective:** Validate searching weather by empty string via Web UI

**Requirement:** Req-001, Req-004, Req-009, Req-010

**Tag:** Functional

**Browser:** Chrome, Firefox

**Test procedure:**

1. Open browser
2. Access URL https://openweathermap.org/
3. Wait for search field is displayed
4. Input an empty string into search field
5. Press Enter/Click searching icon 
6. Wait for searching result is displayed (timeout 10 seconds)
7. Validate the result message “Not Found” is displayed

**Expected result:**

* + Step #2: home page is displayed
  + Step #3: search field is displayed
  + Step #6: Search result page is displayed within 10 seconds
  + Step #7: Message is “Not Found” is displayed

## Performance test

**Test Case ID**: Perf-001

**Objective:** Find the maximum number of simultaneous requests that server can handle

**Requirement:** Req-012

**Tag:** Performance

**Test procedure:**

1. Using multithread mechanism to simulate 500 simultaneous searching requests
2. Run loop to send request and check response in 1 hour
3. If after 1 hour, the response time of each request is still in 10 seconds, add 50 more requests and continue validating in 1 hour. Repeat until response time of 50% request is higher than 10 seconds

**Expected result:** feature can support at least 500 simultaneous requests

**Test Case ID**: Perf-002

**Objective:** Validate feature working in and after stressful hour

**Requirement:** Req-012

**Tag:** Performance

**Test procedure:**

1. Using multithread mechanism to simulate amount of simultaneous searching requests which higher than supported number
2. Run loop to send request and check response in 1 hour
3. Reduce number of request to lower supported number
4. Run loop to send request and check response in 1 hour

**Expected result:** feature may be slow in stressful hour but after that the response time is back to normal - within 10 seconds for each request

## Usability test

**Test Case ID**: Usability-001

**Objective:** Validate visibility of search field in case browser windows is changed

**Requirement:** Req-007

**Tag:** Usability

**Test procedure:**

1. Open browser
2. Access to <https://openweathermap.org/>
3. Change browser windows size
4. Validate menu button  is displayed
5. Click menu button
6. Validate search field is displayed in dropdown menu
7. Read list of city from data file
8. Select a city from above list randomly
9. Input selected city name into search filed
10. Press Enter/Click searching icon 
11. Read expected result of selected city from data file
12. Wait for searching result is displayed (timeout 10 seconds)
13. Validate the result match searching condition by comparing with expected result from file

**Expected result:**

* Step #4: menu button is displayed
* Step #6: search field is displayed
* Step #12: search result is displayed within 10 seconds
* Step #13: list of displayed cities matches searching condition

# Bugs Report

**Bug ID:** Bug-001

**Title:** The searching icon doesn’t work

**Build:** 20/03/2021

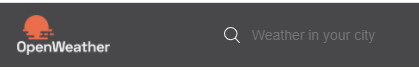
**Severity:** Medium

**Priority:** Medium

**Testing environment:**

* Browser: Chrome 89.0.4389.90, Firefox 86.0
* OS: Windows 10

**Step to reproduce:**

1. Open Browser
2. Access <https://openweathermap.org/>
3. Input any string into search field
4. Click icon  to start searching

**Expected result:** the search result page is displayed

**Actual result:** nothing happens

**Bug ID:** Bug-002

**Title:** The search function doesn’t work if searching string has less than or equal 2 character

**Build:** 20/03/2021

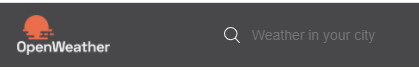
**Severity:** High

**Priority:** High

**Testing environment:**

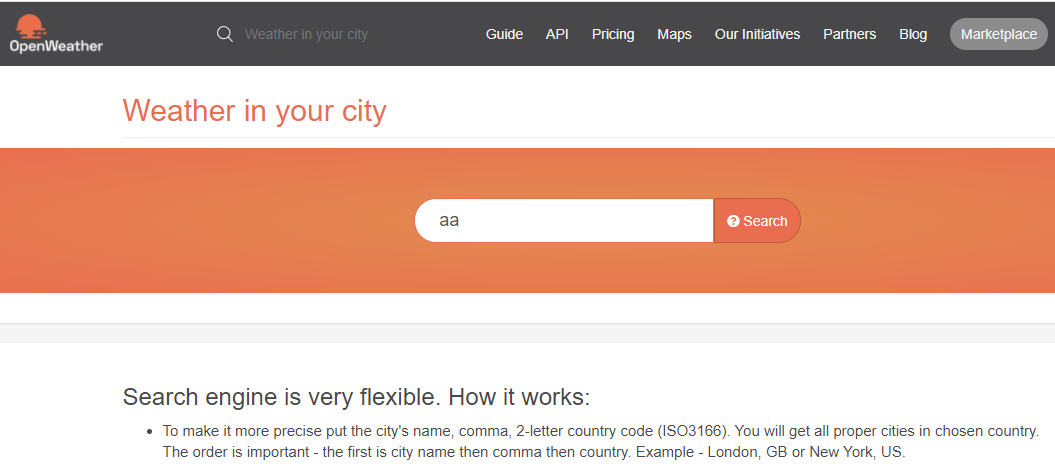
* Browser: Chrome 89.0.4389.90, Firefox 86.0
* OS: Windows 10

**Step to reproduce:**

1. Open Browser
2. Access <https://openweathermap.org/>
3. Input less than or equal 2 characters into search field
4. Click icon  to start searching

**Expected result:** the search result page with links of cities or “Not Found” message is displayed

**Actual result:** neither links of cities nor “Not Found” message is displayed



**Bug ID:** Bug-003

**Title:** More than 1 results are return for one city

**Build:** 20/03/2021

**Severity:** High

**Priority:** High

**Testing environment:**

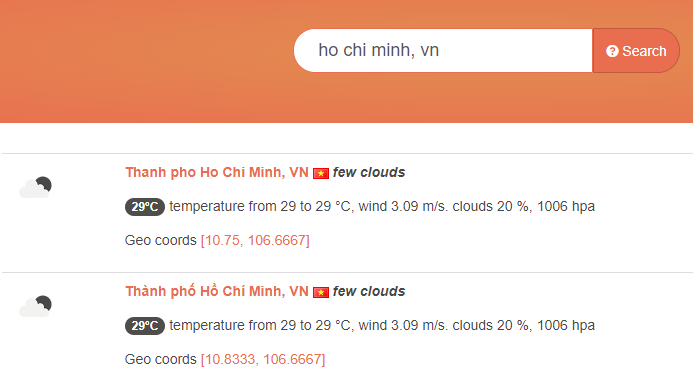
* Browser: Chrome 89.0.4389.90, Firefox 86.0
* OS: Windows 10

**Step to reproduce:**

1. Open Browser
2. Access <https://openweathermap.org/>
3. Input any string into search field “ho chi minh, vn”
4. Press Enter
5. Validate search result

**Expected result:** Only one record is return because there is only one city called Ho Chi Minh in Viet Nam

**Actual result:** 2 records are return for same city



**Bug ID:** Bug-004

**Title:** Search result page cannot display all city of a country

**Build:** 20/03/2021

**Severity:** High

**Priority:** High

**Testing environment:**

* Browser: Chrome 89.0.4389.90, Firefox 86.0
* OS: Windows 10

**Step to reproduce:**

1. Open Browser
2. Access <https://openweathermap.org/>
3. Input country code search field “,US”
4. Press Enter
5. Validate search result

**Expected result:** All cities of searching country should be displayed in result page

**Actual result:** only 5 cities is displayed

