Software Test Description

By: Dominic Crofoot, Travis Early, Jordan Fok, Sean Hill, & Nicholas Ho

Created December 2, 2018

Revisions:

- **10Dec2018: First draft (by: Dominic, Jordan, Nicholas, Sean, and Travis)**
  - First draft of SRS document created.

Table of Contents:

Scope……….……….……….……….……….……….……….……….4
Identification……………………………………………………………….. 4
System Overview……….……….……….……….……….……….……….... 4
Document Overview……….……….……….……….……….……….…….. 4

Referenced Documents……….……….……….……….……….……….…….. 4
Government……….……….……….……….……….……….……….....4
Non-Government……….……….……….……….……….……….……... 4

Test Preparations……….……….……….……….……….……….……….... 5
Hardware Preparation……….……….……….……….……….……….… 5
3.2. Software Preparation……….……….……….……….……….……….....5

Test Description……..……….……….……….……….……….……….…....
5
Providing Limited Results from Google Maps API (T-1)……..……….……….…..... 5
Display Results and Populate Map with Results (T-2)……..……….……….…........6
Recording Search History (T-3)……..……….……….……….……….…….....8
Filtered Results from Google Maps API (T-4)……..……….……….……….….... 9
Providing User Authentication (T-5)……..……….……….……….……….. 10

Requirements Traceability……..……….……….……….………………………11

1. Scope

1.1. Identification

1.1.1. SSDD: System/Subsystem Design Description

1.1.2. SRS: System Requirements Specification

1.1.3. Date planner: unnamed application team is designing

1.1.4. FR1,2,3: Functional requirements 1, 2, & 3. Further specified in SRS

1.1.5. EIR1,2,3: External Interface Requirements 1, & 2. Further specified in SRS

1.1.6. Table1: transition state diagram found in SRS

1.1.7. Table2: Qualification procedures for each requirement found in SRS

1.1.8. API: Application Program Interface

1.1.9. MIL-STD: Military Standard

1.2. System Overview

1.2.1. The system will provide users the ability to search through date ideas based on parameters (filters) they have chosen. The results will be displayed on a map and a results tab.

1.3. Document Overview

1.3.1. This document will cover the testing methods that will be used for each requirement found in the SRS to ensure functionality, reliability, and efficiency of the system.

2. Referenced Documents

2.1. Government

2.1.1. MIL-STD 498, 5 December 1994

2.1.1.1. Template for entire SRS document

2.1.1.2. Link: MIL-STD 498

2.2. Non-Government

2.2.1. Loopback

2.2.1.1. A node.js framework used for back-end services. Loopback will be used by the application for user authentication and retrieving of the users filter history.

2.2.1.2. Link: https://loopback.io/

2.2.2. Google Maps API

2.2.2.1. Data supplied by google that will be used for filter, results tab, and map.

2.2.2.2. Link: https://cloud.google.com/maps-platform

3. Test Preparations

3.1. Hardware Preparation

There is extensive hardware preparation that must be done in order to use or test NoWut. The hardware environment must be capable of running one of these browser versions: Google Chrome version 70.0.3538.110 / November 19, 2018, and Mozilla FireFox 63.0.3 / November 15, 2018.

3.2. Software Preparation

The tests may be conducted on either of the browser versions specified in section 3.1. No specific setup is required to run the tests. The site is located at localhost:3000.

3.3. Other Preparations

N/A

4. Test Description

4.1. Providing Limited Results from Google Maps API (T-1)

Test case ID: T-1.1
Requirement Addressed: The application shall provide users with a limited number of results from Google Maps API that fit the filters given by the user
Description: Verify the application will output the number of results in the results tab and map based on code.

Test Environment: Default (Refer to github repo's README.md on setup and run instructions)
Prerequisite Conditions: Valid Filters must be selected in the fields of the filters component before searching. A valid address must be input into the address field prior to searching.
Inputs: Valid filter input from user. The user must also press the search button to initiate and confirm the search parameters.
Expected Results: The results tab and map will have a number of results displayed that matches the number the team has set.
Test Procedures:

Step #	User Action	Expected Result	Pass = P
			Fail = X

1.	**Navigate to Nowut web application**	**Web application opens properly**	
2.	**User enters an address into address bar**	**Valid address is entered into address bar**	
3.	**User selects filters from the various**	**Valid filters are chosen according to user preference**	
4.	**User clicks on the search button**	**App takes in filters and address, and begins culminating results from Google Maps API. The app then outputs it to the user via Results Tab and Map**	

Result Evaluation Criteria: Results must be limited to 20 in both the results tab and on the map. There must be 20 total pins on the map. Results must correspond with the filters the user specified. Results must be within a certain distance based on the user filters and address given.

Assumptions and Constraints: The device the application is running on must have access to the SQL database through network or local means. User knows/has a proper address to use.

4.2. Display Results and Populate Map with Results(T-2)

Test Case ID: T-2.1

Requirement Addressed: The application shall display and populate a map with pins denoting the location of activities returned by the software.

Description: Verify that every result returned by the software is represented by a map pin at the geographically correct location on the map.

Test Environment: Default (Refer to github repo's README.md on setup and run instructions)

Prerequisite Conditions: Valid Filters must be selected in the fields of the filters component before searching. A valid address must be input into the address field prior to searching.

Inputs: Valid filters selected in the filter component. The user must press the search button to commence a search and confirm the current filters to be stored.

Expected Results: Every result returned by the software is represented by a map pin at the geographically correct location on the map.

Test Procedures:

Step #	**User Action**	**Expected Result**	**Pass = P Fail = X**
1.	**Navigate to the login screen of Nowut located at the top right of the main page.**	**Login page opens.**	
2.	**Sign in to an existing account using login credentials (username and password).**	**Login and authentication is successful, bringing the user to the main page.**	
3.	**Select filters in the filters menu located at the top of the page.**	**Valid filters fill in the fields of the component.**	
4.	**Enter a valid address into the address bar located below of the filters.**	**A valid address is entered into the address bar.**	
5.	**Click the search button.**	**A query is run using the currently selected filters and address in the address bar. The map is then populated with pins denoting the locations of all of the results returned by the software.**	

Result Evaluation Criteria: All results must have a correlating map pin, and all map pins must have a correlating result. These pins must be geographically accurate and must contain the same relevant information as in the result list. Any deviation from these constraints result in a test result failure.

Assumptions and Constraints: The Google Maps API must be connected to a network or local means.

4.3. Recording Search History (T-3)

Test Case ID: T-3.1
Requirement Addressed: The application shall store the search history of users who are logged into an account. If a user wants a previous result or set of results, this feature allows them to do so. (FR3)
Description: Verify the storage of user searches in the SQL database

Test Environment: Default (Refer to github repo's README.md on setup and run instructions)
Prerequisite Conditions: A user must currently be logged into the application. Valid Filters must be selected in the fields of the filters component before searching. A valid address must be input into the address field prior to searching.
Inputs: Valid filters selected in the filter component. The user must press the search button to commence a search and confirm the current filters to be stored.
Expected Results: All filters currently selected are stored in a row in the SQL database.

Test Procedures:

Step #	**User Action**	**Expected Result**	**Pass = P Fail = X**
1.	**Navigate to the login screen of Nowut located at the top right of the main page.**	**Login page opens.**	
2.	**Sign in to an existing account using login credentials (username and password).**	**Login and authentication is successful, bringing the user to the main page.**	
3.	**Select filters in the filters menu located at the top of the page.**	**Valid filters fill in the fields of the component.**	
4.	**Enter a valid address into the address bar located below of the filters.**	**A valid address is entered into the address bar.**	

5.	**Click the search button.**	**A query is run using the currently selected filters and address in the address bar. The currently selected filters are stored into a new row in the SQL database.**	

Result Evaluation Criteria: All selected filters must be stored in a row of the SQL database. Strings do not have to be case sensitive. Numbers must be rounded to the nearest tenth place decimal. Values must be stored within 10 seconds of the user pressing the search button. No errors, exceptions or flags may be raised. Any deviation from these constraints result in a test result failure.
Assumptions and Constraints: A connection to the database must have been established. The device the application is running on must have access to the SQL database through network or local means.

4.4. Filtered Results from Google Maps API (T-4)

Test case ID: T-4.1
Requirement Addressed: The application shall query the Google Maps API upon a set of given filters to develop results that align with the filters.
Description: Verify that results make sense with respect to the supplied filters

Test Environment: Default (Refer to github repo's README.md on setup and run instructions)
Prerequisite Conditions: Valid Filters must be selected in the fields of the filters component before searching. A valid address must be input into the address field prior to searching.
Inputs: Valid filter input from user. The user must also press the search button to initiate and confirm the search parameters.
Expected Results: The results tab and map will have a number of results displayed that matches the number the team has set.
Test Procedures:

Step #	User Action	Expected Result	Pass = P
			Fail = X

1.	**Navigate to Nowut web application**	**Web application opens properly**	
2.	**User enters an address into address bar**	**Valid address is entered into address bar**	
3.	**User selects filters from the various**	**Valid filters are chosen according to user preference**	
4.	**User clicks on the search button**	**App takes in filters and address, and begins culminating results from Google Maps API. The app then outputs it to the user via Results Tab and Map**	

Result Evaluation Criteria: Each result should be relevant to the filters which were supplied during the search. No results that contradict the filters should be displayed.

Assumptions and Constraints: The device the application is running on must have access to the SQL database through network or local means. User knows/has a proper address to use.

4.5. Providing User Authentication (T-5)

4.5.1. Test Case 1: HTTP Status of the request

Test Case ID: T-5.1

Requirement Addressed: The application shall utilize the IBM Loopback framework to develop user accounts and store data using MySQL.

Description: Verify that the http request was valid by comparing the status of the request and the data returned

Test Environment: Default (Refer to github repo's README.md on setup and run

instructions)
Prerequisite Condition: N/A
Inputs: Loopback Explorer provides input based request, as well as valid user data
Expected Results: Valid user data, and HTTP response status 200
Requirement: Loopback Explorer API

Step #	User Action	Expected Result	Pass = P Fail = X
1.	**Navigate to http://localhost:3000/explorer /**	**Loopback Explorer Testing API open**	
2.	**Select /user_models/{id} (GET) and input an id**	**If id is valid it will return the user data and HTTP status 200**	
3.	**Select /user_models (POST) and follow input directions**	**The same user data should return but with a valid id and HTTP status 200**	
4.	**Select /user_models/findOne (GET) and follow filter input directions**	**The user data corresponding to the filter setting should be returned and HTTP status 200**	

Result Evaluation Criteria: Each request should result with an HTTP response status code of 200, and valid data.

Assumptions and Constraints: A connection to the database must have been established. The device the application is running on must have access to the SQL database through network or local means.

5. Requirements Traceability

The following table describes the traceability between the Functional Requirements as described in the SRS and the CSCIs as described.

Requirement	Capability	Related CSCIs
FR1	Provide search results based on user filters	**C4.1.3, C4.1.4**

FR2	Provide a user-friendly location display	**C4.1.3**
FR3	Allow registered users to store searches	**C4.1.1**
EIR1	Allow the user to customize search options	**C4.1.4**
EIR2	Store and authenticate user-specific accounts	**C4.1.1, C4.1.2**