National College of Ireland

Software Quality and Testing

Group 1 – WhatsOn - Project Report

5th May 2024

2023/2024

Contents

1.0 Introduction

“WhatsOn” is a map-based event discovery web application. The application enables users to browse events on a map-based user interface, filter events by type, and even randomly select events when they’re unsure of what to choose. Users can follow recurring events and view personalized events based on their preferences. They can also contact WhatsOn for inquiries or support.

The application is designed to overcome common barriers to socializing such as decision fatigue, anxiety, and FOMO (Fear of Missing Out), by encouraging spontaneity and action.

In addition, “WhatsOn” facilitates ticket purchasing for events and provides users with directions to events, making the process of discovering and attending events seamless and straightforward.

The project was undertaken with the belief in its great potential to provide a welcoming user experience and reduce digital anxiety. The application is designed with a singular vision that permeates the entire application, from colour scheme and layout to font, copy, and features.

2.0 Test Plan

2.1 Overview

The test plan encompasses a systematic approach to testing the various components of the WhatsOn application. It outlines the references, testing scope, levels, design techniques, completion criteria, and conclusions. The plan includes both manual and automated testing procedures to ensure thorough coverage of the application's functionality and performance.

If you would like a more in-depth understanding of our application, please refer to the Additional Material folder where you will find our Final Presentation Slides and Demo video.

2.2 References

- System Requirements Specification document
- Use Case Documentation

2.3 In-Scope

1. **Account Creation/Login:**
    - Validate the account creation process, ensuring that users can successfully register and log in to the system.
    - Technique: Equivalence Partitioning, Statement Coverage, Unit Testing
2. **Map Initialization:**
    - Verify the proper initialization of the map component, including loading, display, and interaction functionalities.
    - Technique: Use Case Testing, Statement Coverage, Unit Testing
    - Use Cases:
        - User initializes the map
        - User interacts with map controls
        - User searches for a location on the map
3. **User Location:**
    - Test the functionality related to identifying and displaying the user's location on the map.
    - Technique: Use Case Testing
4. **Browse Events:**
    - Ensure users can browse and view events displayed on the map.
    - Technique: Use Case Testing

5. **Random Event:**
    - Verify the system's ability to display random events to users.
    - Technique: Use Case Testing
6. **Filter Events:**
    - Test the event filtering feature based on event type.
    - Technique: Use Case Testing, Branch Coverage
7. **Event Source:**
    - Validate the functionality related to linking to the source/organizer of events.
    - Technique: Use Case Testing
8. **Event Directions:**
    - Verify the system's ability to link to directions to events from the user's current location.
    - Technique: Use Case Testing
9. **Event Icon Creation:**
    - Test the generation and display of event icons on the map.
    - Technique: Decision Table Testing
10. **Contact Form:**
    - Validate the functionality of the contact form, ensuring users can submit inquiries or feedback.
    - Technique: Equivalence Partitioning
11. **Toggle Modal:**
    - Validate the functionality of the Following, Random, Legend and Contact-buttons.
    - Technique: Branch Coverage
12. **Loading Map with Encoded key:**
    - Validate the functionality of the encrypted Map API key.
    - Technique: Unit Testing

2.4 Out-of-Scope

Ticket Purchasing: The ticket purchasing feature, which involves linking to the event source, is out of scope as it has not yet been implemented.

Following Feature and Personalized Events: These features are not yet implemented and are therefore out of scope.

Offline Functionality: Features that rely on an internet connection are out of scope. The application does not support offline backups, so it is assumed that the user will always have an internet connection.

API Testing: The APIs used in the application are assumed to be working correctly within this testing criteria. The API providers are responsible for testing their APIs.

Integration Testing: The integration of the application with external vendors for ticket purchasing is out of scope. The focus is on the application's functionality and security.

Non-Functional Testing: Aspects like compatibility and localization are not included in the scope of this project.

2.5 Test Levels

- Unit Testing: Validate individual components and functionalities.
- System Testing: Evaluate the integrated system as a whole.

2.6 Test Design Techniques

- Unit Testing:
    - Black-box testing techniques: Equivalence partitioning, boundary value analysis
    - White-box testing techniques: Statement coverage, branch coverage
- Use Case Testing: Validate system behaviours from an end-user perspective.
- Decision Table Testing: Evaluate different combinations of inputs and expected outputs.

2.7 Test Completion Criteria

- Unit Testing: Achieve 100% code coverage for critical functionalities.
- System Testing: Cover all user stories and acceptance criteria outlined in the requirements specification document.

3.0 Test Cases Designed Using Black-Box Testing Techniques

Tester's Name	Feature	Technique
Eoin Fitzsimons	Account Creation	Equivalence Partitioning
Conor Judge	Map Initialisation	Use Case Testing
Conor Judge	User Location	Use Case Testing
Conor Judge	Browse Events	Use Case Testing
Conor Judge	Random Event	Use Case Testing
Conor Judge	Filter Events	Use Case Testing
Conor Judge	Event Source	Use Case Testing
Conor Judge	Event Directions	Use Case Testing
Conor Judge	Event Icon Creation	Decision Table Testing
David O Connor	Contact Form	Equivalence Partitioning

Eoin Fitzsimons - Account Creation - Email restrictions

Test Case ID	**WO_001**				
Test Case Description	Test the email parameters for creating an account				
Created By	Eoin				
Date Tested	26/04/2024				
Tester's Name	Eoin				
			Equivalence Partitioning		
#	**Prerequisites**		**Condition**	**Valid Equivalence Classes**	**Invalid Equivalence Classes**
1	Access to Edge Browser		email contains @ and then a character	`@g	gg@
2	WhatsOn web application is available		email contains @	gg@	ggg
			email contains character then @	gg@	`@g
#	**Test Data**				
1	Password = g				
Test Scenario	Verify on creating an account, the user must enter a valid string				
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**	
1	Press a character on the keyboard	Character appears in form	As Expected	Pass	
2	Press sign up	App verifies the email	As Expected	Pass	
3	Press sign up with valid email	App sends confirmation.	As Expected	Pass	
Test Case (Pass/Fail/Not Executed)					
Pass					

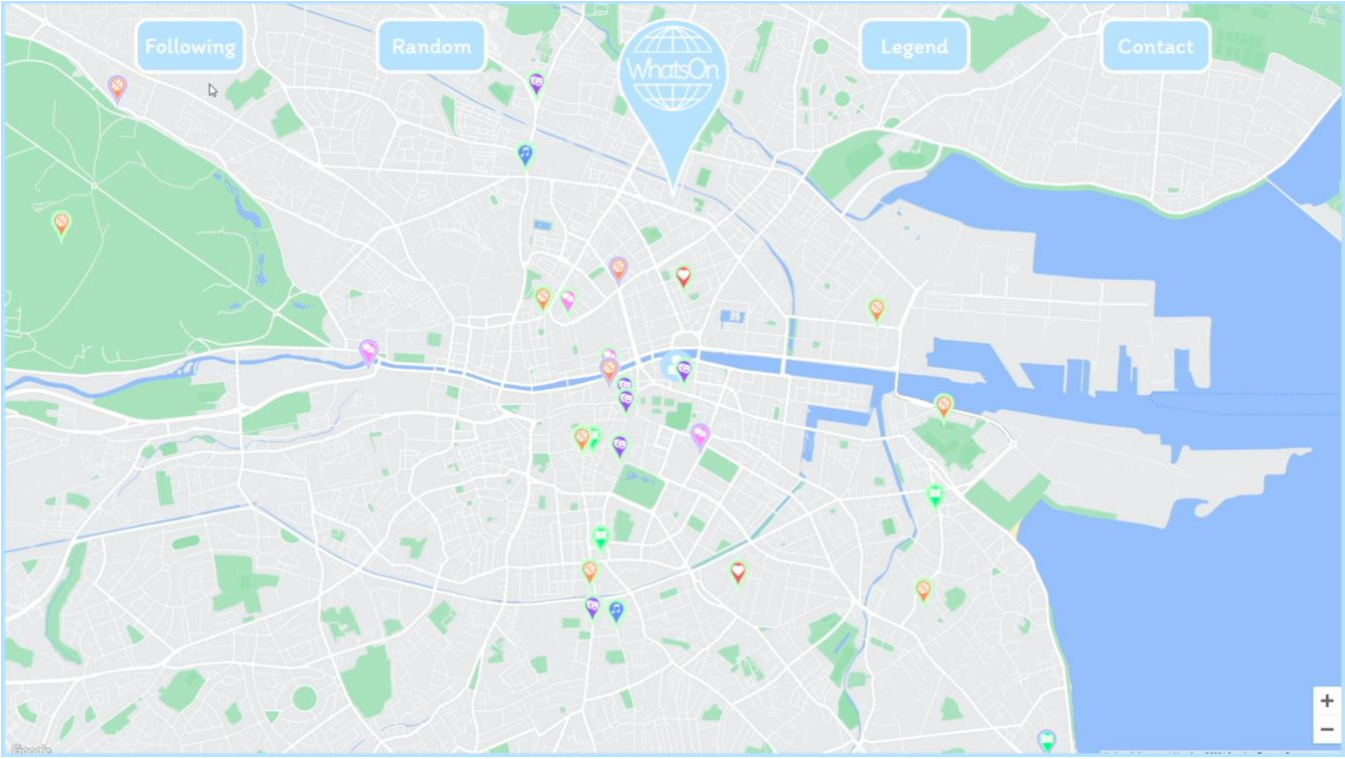
Conor Judge - Use Case 1 - Map Initialization

Test Case ID	BU_001	**Version**	1.0	**Test Case Description**	Use Case 1 - Map Initialization
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Map Initialisation Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API

S #	Step Details	Expected Results	Actual Results	Status
1.1	Verify that the map initializes correctly with the specified center coordinates.	Map is centered at lat: 53.349076911151386, lng: -6.242441879918039.	As Expected	Pass
1.2	Verify that the map initializes with the correct zoom level.	Map zoom level is 14.	As Expected	Pass
1.3	Verify that the map initializes with the correct mapId.	MapId is “13185b1ffbbba3af”.	As Expected	Pass
1.4	Verify that the map displays correctly in the “map-div” element.	Map is displayed in the “map-div” element.	As Expected	Pass
1.5	Verify that the correct map controls are disabled.	mapTypeControl, fullscreenControl, and streetViewControl are disabled.	As Expected	Pass

1.1-1.5



Conor Judge - Use Case 2 - User Location Data

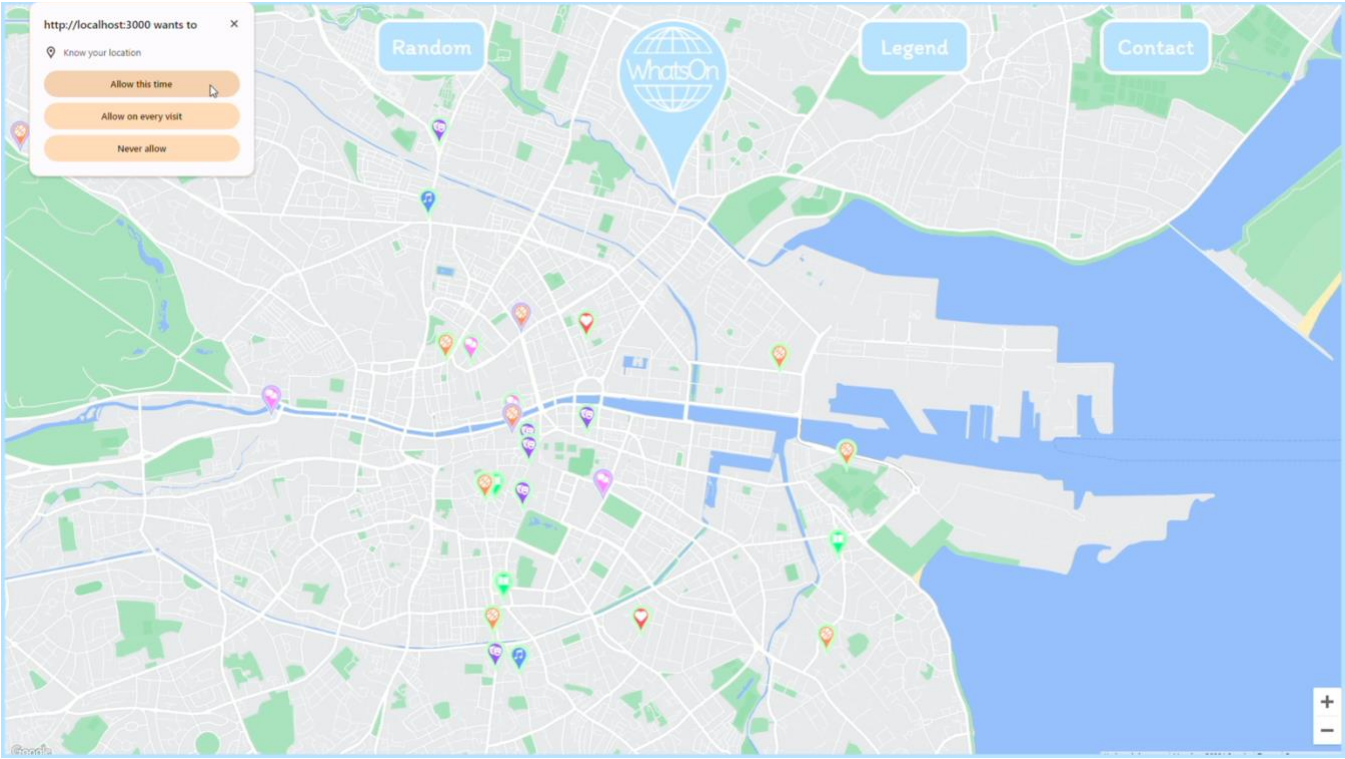
Test Case ID	BU_002	**Version**	1.0	**Test Case Description**	Use Case 2 - User Location
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking User Location Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:
1	Access to WhatsOn Web Application
2	WhatsOn is correctly loaded on the browser

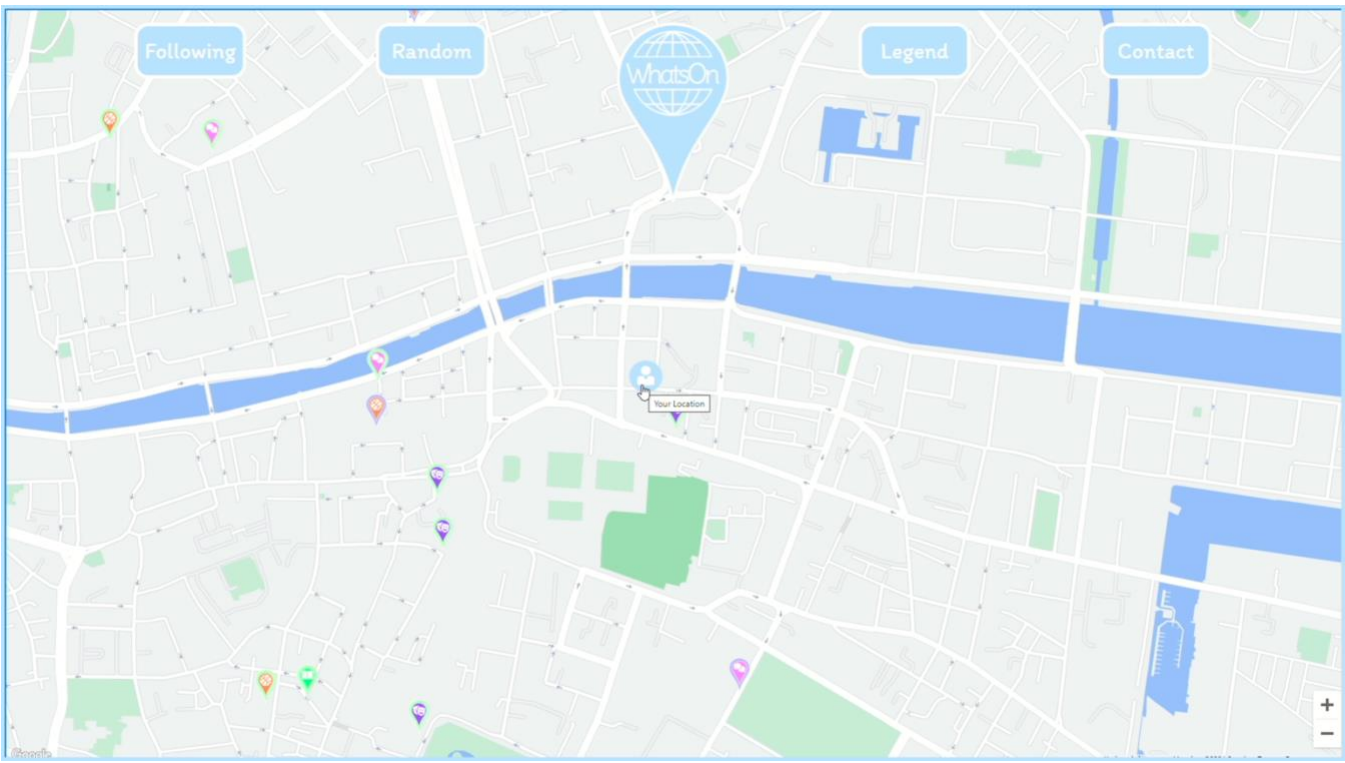
S #	Test Data
1	Google Maps API
2	Maps JavaScript API
3	Google Cloud Console
4	Maps JavaScript API
5	User's Geolocation Data

S #	Step Details	Expected Results	Actual Results	Status
2.1	Verify that the user is prompted to allow location permissions	The user is prompted to allow location permissions	As Expected	Pass
2.2	If the user's location is granted, verify that the user's location is retrieved and displayed correctly.	If the user's location is granted, a user icon displays at the user's location and the map centres at that location	As Expected	Pass
2.3	If the user's location is denied/unavailable, verify that the map displays as normal.	If the user's location is denied/unavailable, the map displays and centres at standard long/lat.	As Expected	Pass

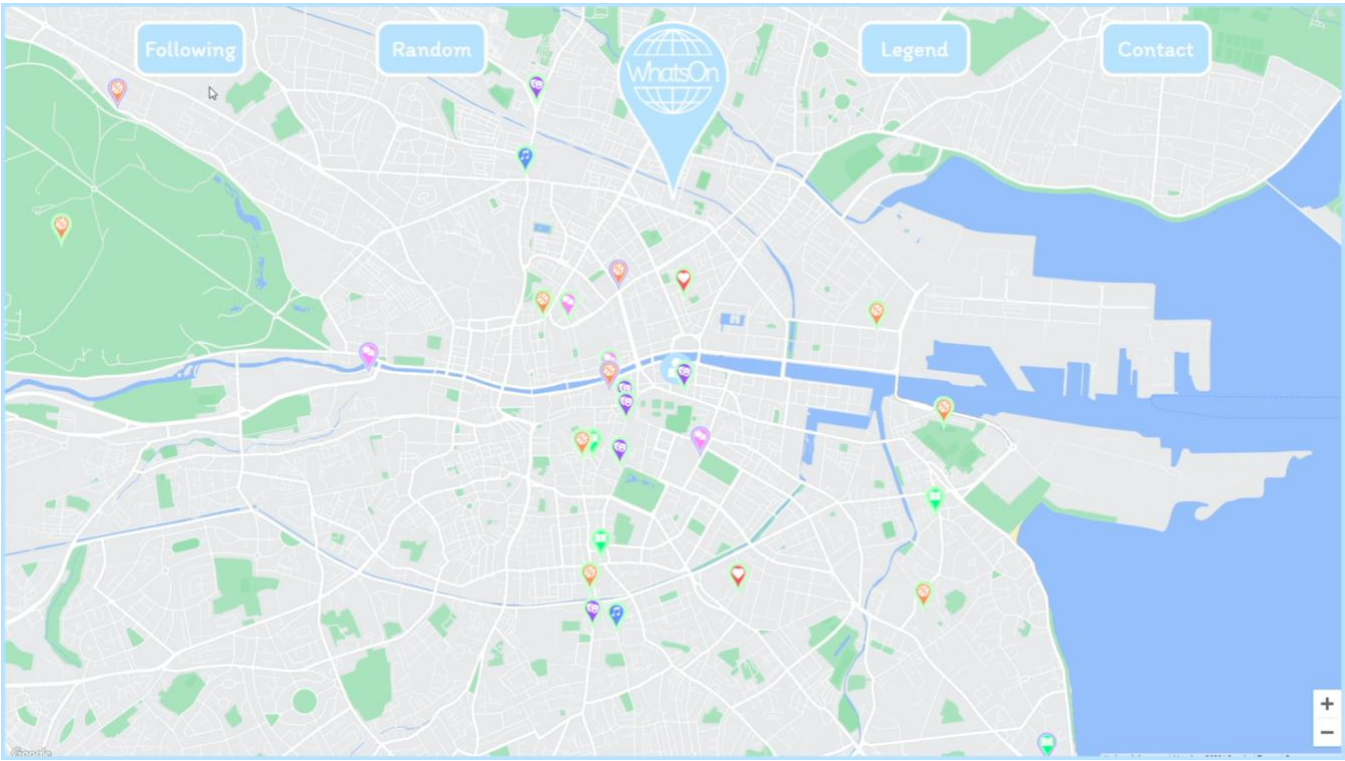
2.1


http://localhost:3000 wants to
Know your location
Allow this time
Allow on every visit
Never allow
Random
WhatsOn
Legend
Contact

2.2


Following
Random
WhatsOn
Legend
Contact
Your Location

2.3


Following
Random
WhatsOn
Legend
Contact

Conor Judge - Use Case 3 - Browse Events

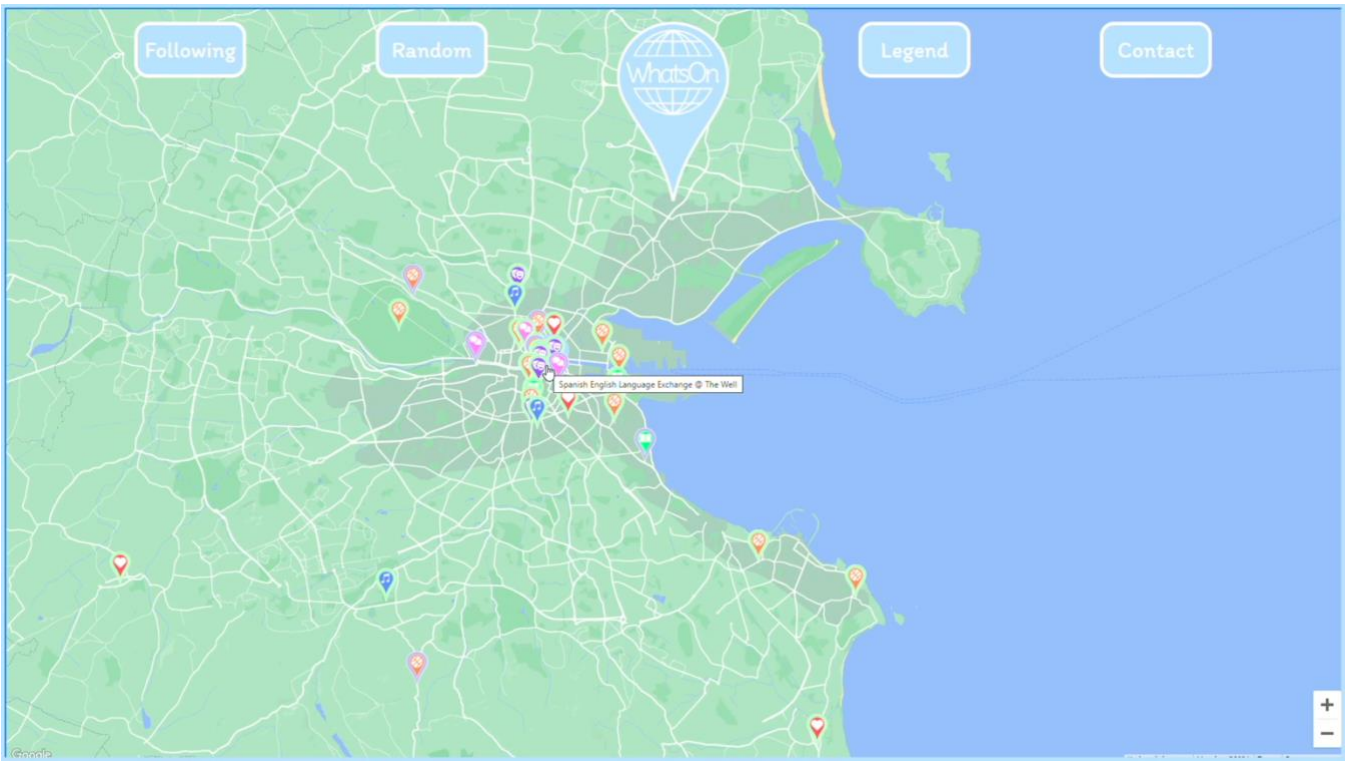
Test Case ID	BU_003	**Version**	1.0	**Test Case Description**	Use Case 3 - Browse Events
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Browse Events Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

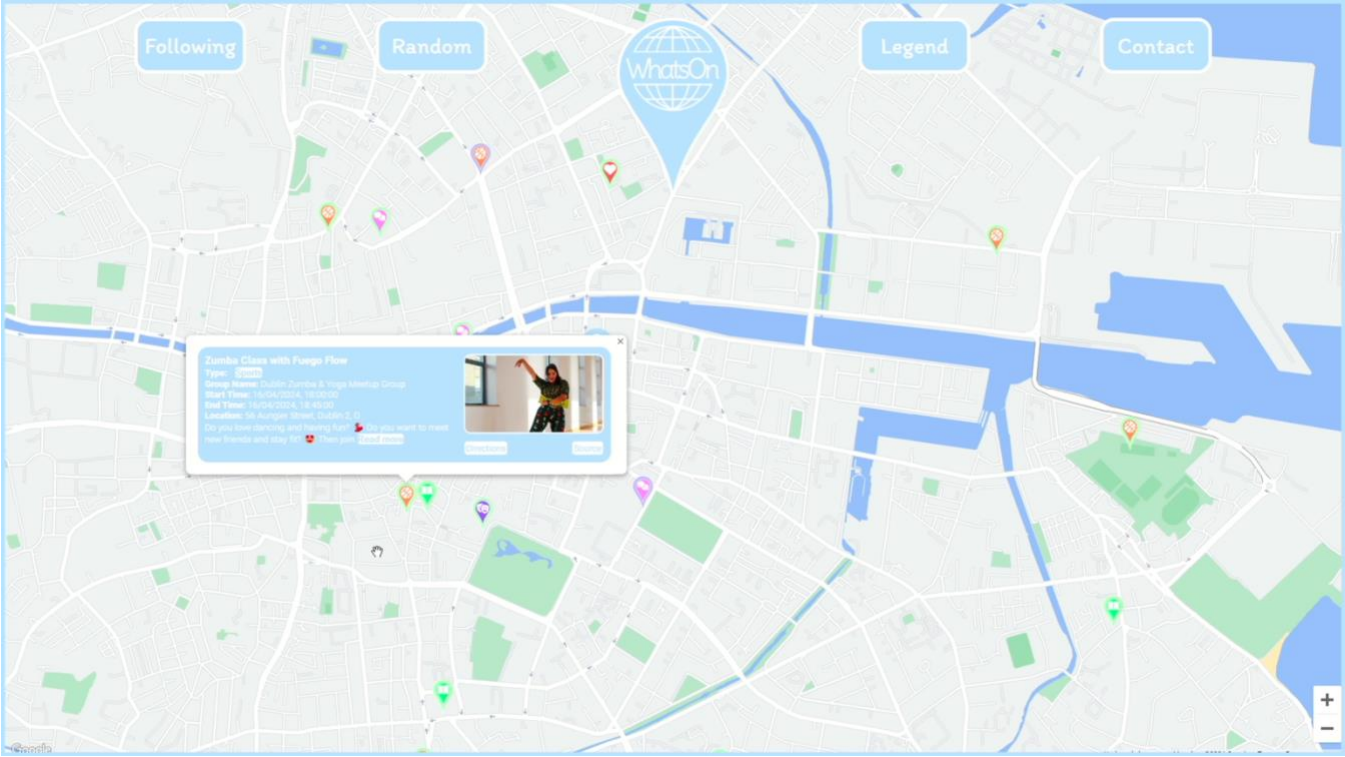
S #	Step Details	Expected Results	Actual Results	Status
3.1	For each event, verify that a marker is created with the correct location, icon, and title.	Markers are created correctly for each event.	As Expected	Pass
3.2	Verify that clicking on a marker opens the correct info window.	Clicking on a marker opens the correct info window and closes any previously opened info window.	As Expected	Pass
3.3	Verify that clicking on the map closes any previously opened info window.	Clicking on the map closes any previously opened info window.	As Expected	Pass
3.4	Verify that clicking on a marker opens the correct info window and	Clicking on a marker opens the correct info window and closes any previously opened info window.	As Expected	Pass

	closes any previously opened info window.			
3.5	For each event, verify that an info window is created with the correct content.	Info windows are created with correct Name, Type, Group Name, Start Time, End Time, Location, Directions, Source, Event Image.	As Expected	Pass

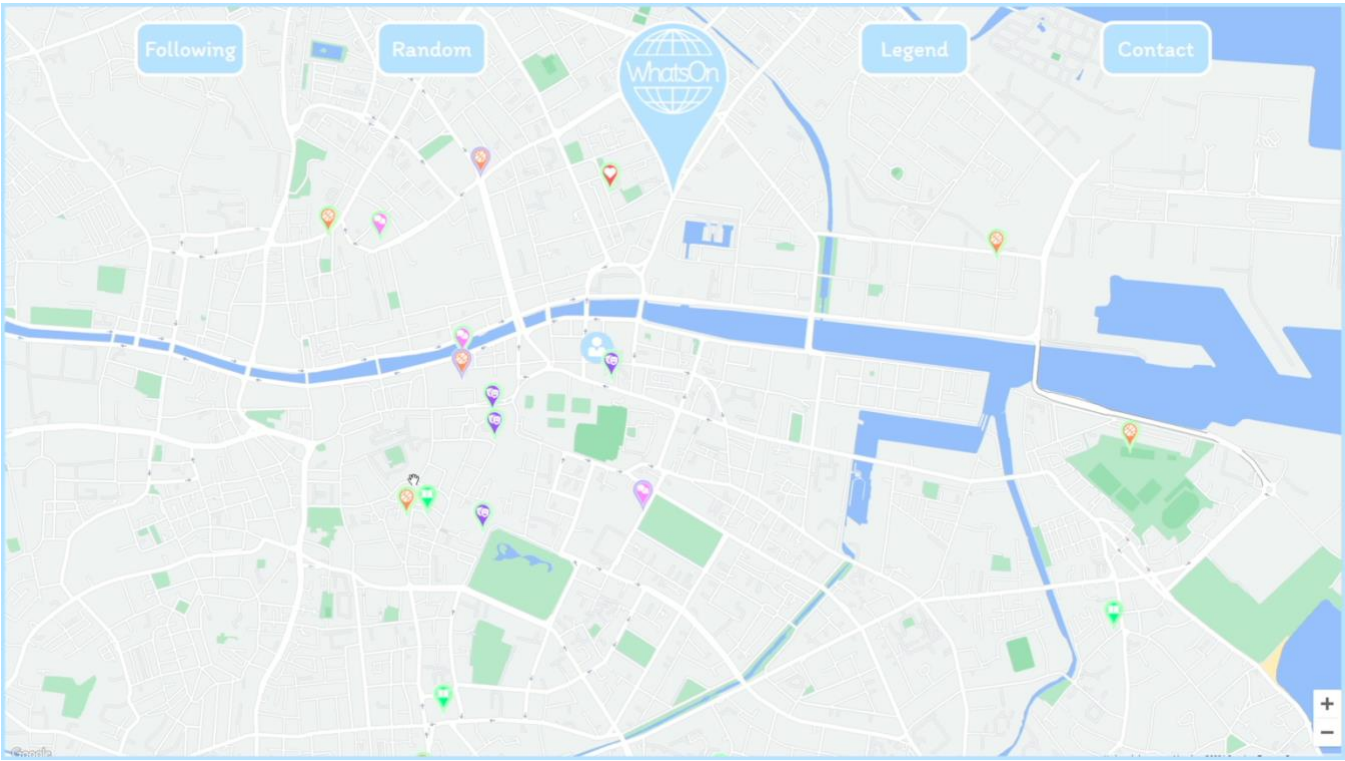
3.1



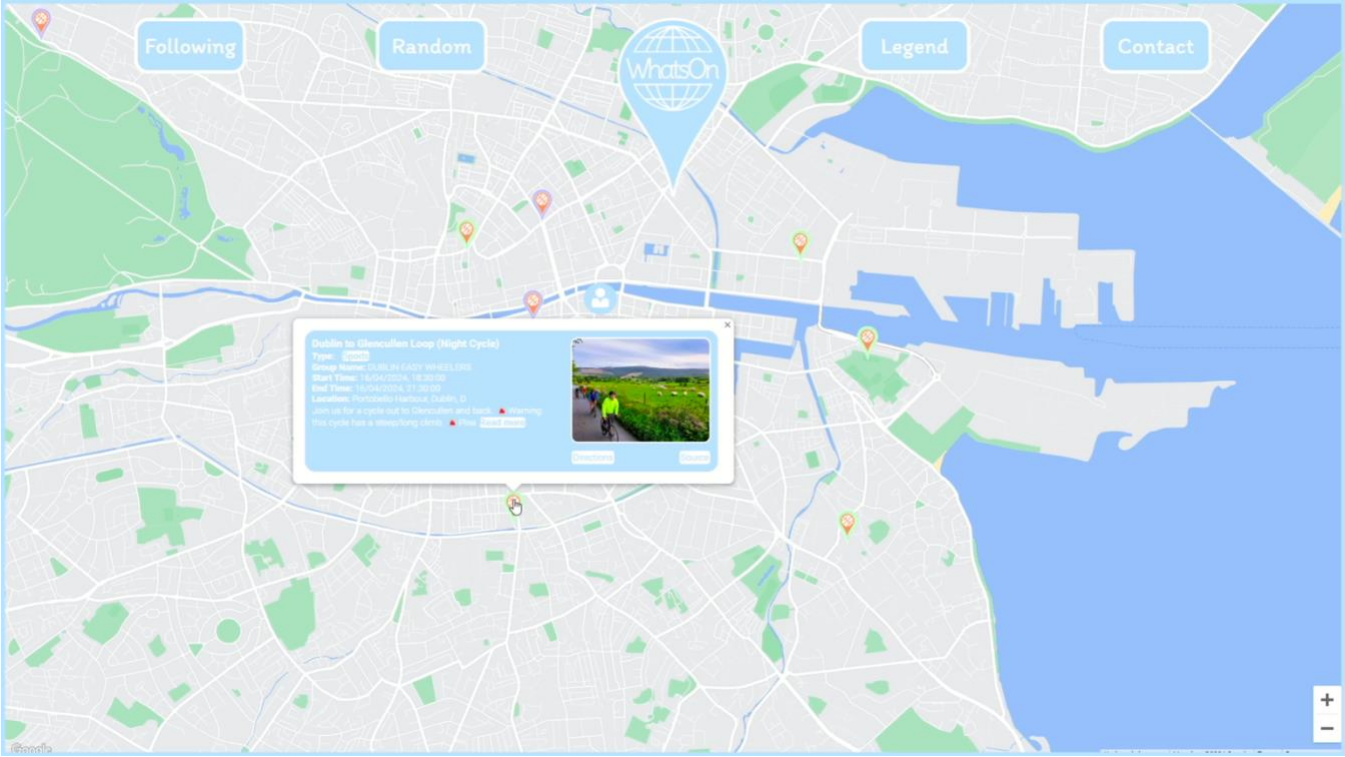
3.2, 3.5


Following
Random
WhatsOn
Legend
Contact
Zumba Class with Fuego Flow
Type:
Group Name: Dublin Zumba & Yoga Meetup Group
Start Time: 16/04/2024, 18:00:00
End Time: 16/04/2024, 18:45:00
Location: 56 Aungier Street, Dublin 2, IE
Do you love dancing and having fun? Join us as we want to meet new friends and stay fit! Then join
Google

3.3


Following
Random
WhatsOn
Legend
Contact
Google

3.4



Conor Judge - Use Case 4 - Random Event

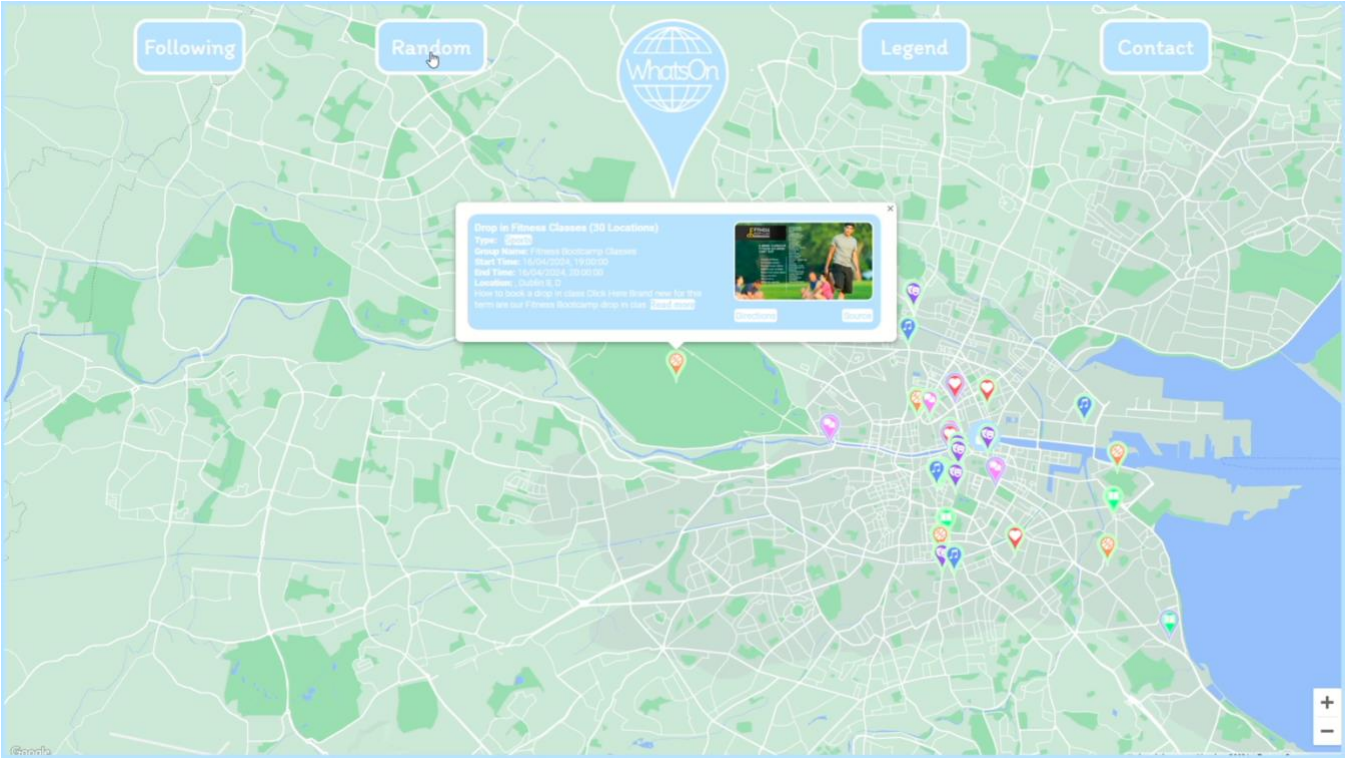
Test Case ID	BU_004	**Version**	1.0	**Test Case Description**	Use Case 4 - Random Event
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Random Event Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

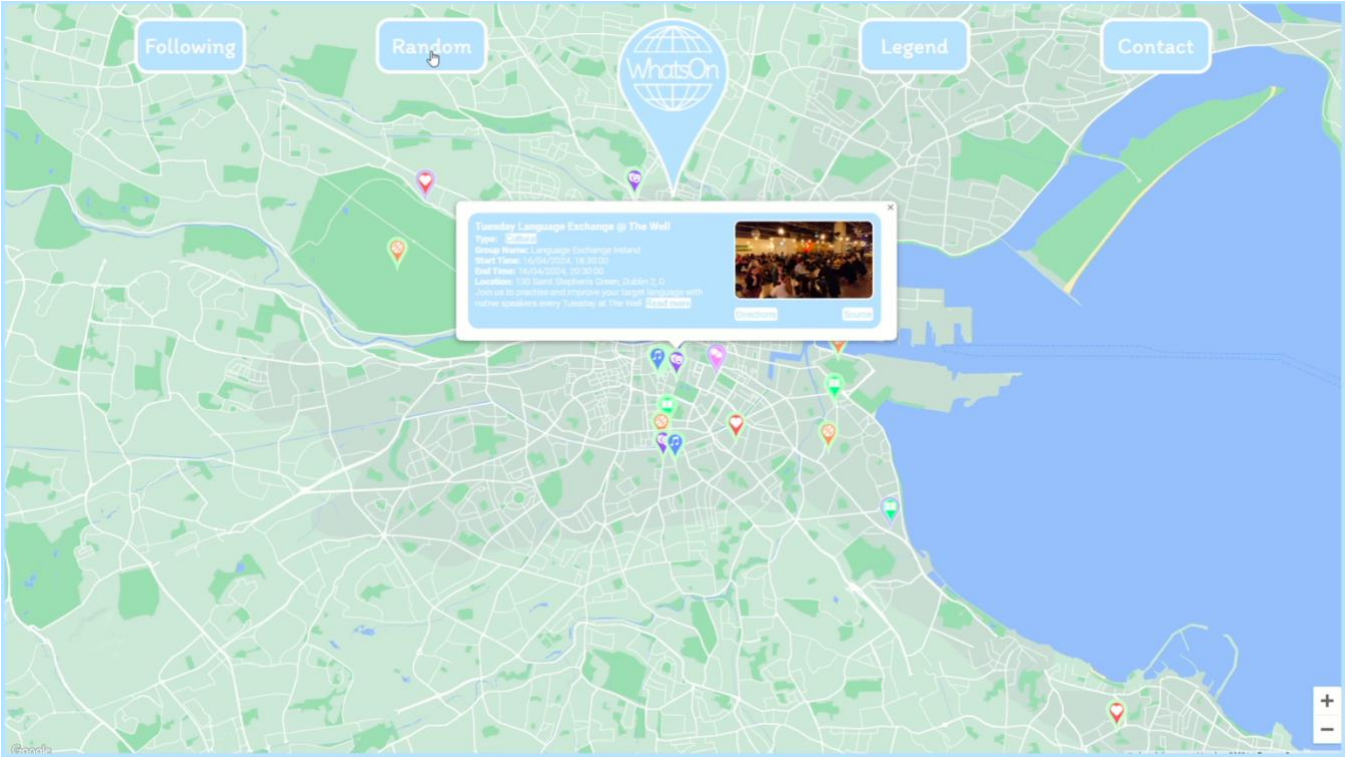
S #	Step Details	Expected Results	Actual Results	Status
4.1	Verify that clicking the random event button selects a random event	Clicking the random event button selects a random event	As Expected	Pass
4.2	Verify that clicking the random event button opens a random events info window	Clicking the random event button opens a random events info window	As Expected	Pass

4.3	Verify that clicking the random event button centers the map on a random events location	Clicking the random event button centers the map on a random events location	As Expected	Pass
4.4	Verify that clicking the random event button multiple times selects different events each time.	Clicking the random event button multiple times selects different events each time.	As Expected	Pass

4.1-4.3



4.4



Conor Judge - Use Case 5 - Filter Events

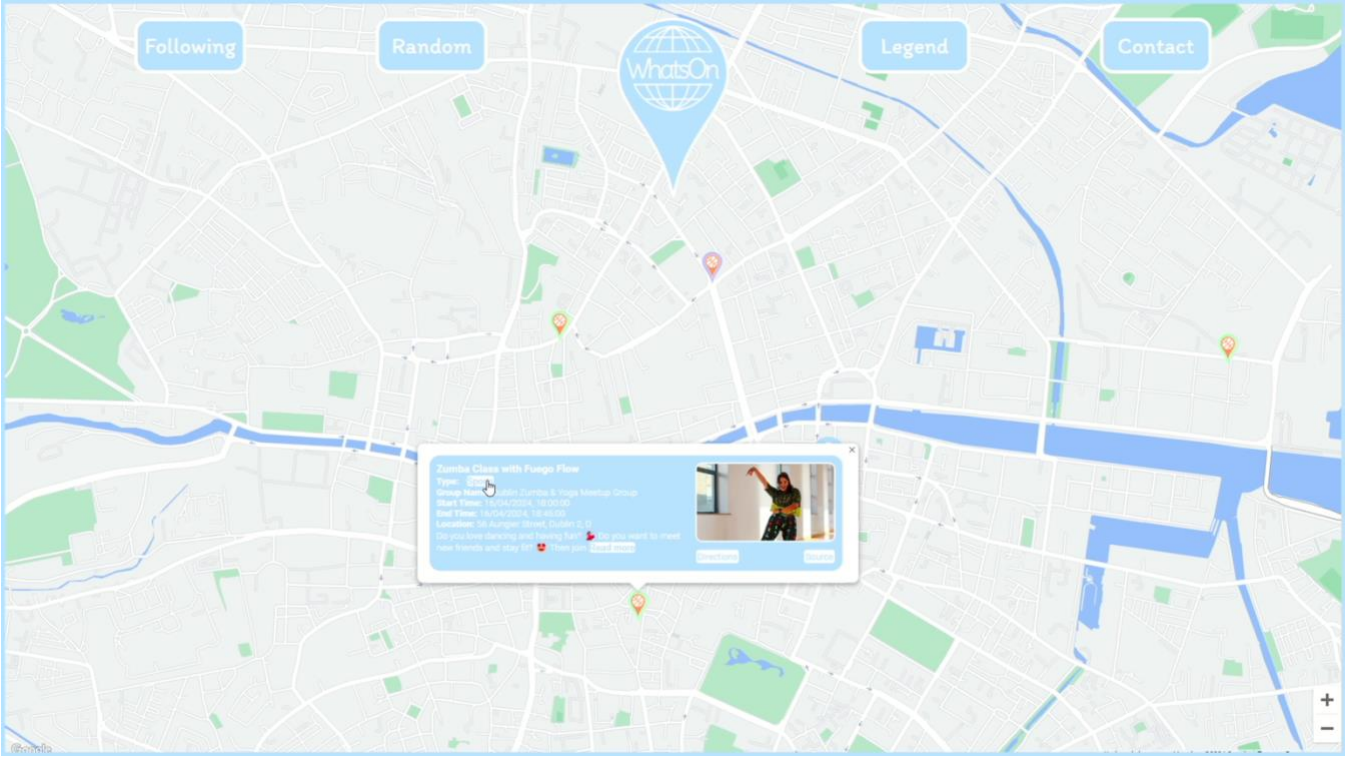
Test Case ID	BU_005	**Version**	1.0	**Test Case Description**	Use Case 5 - Filter Events
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Filter Events Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

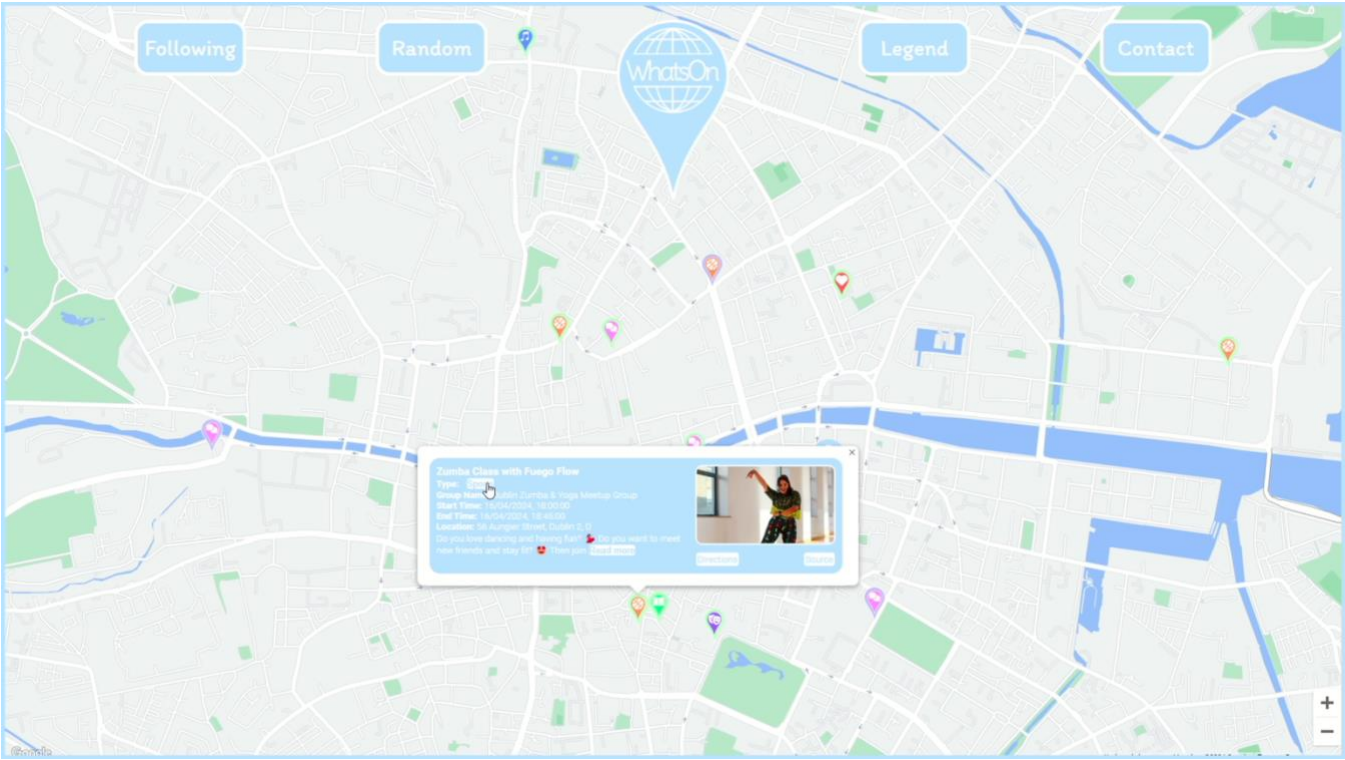
S #	Step Details	Expected Results	Actual Results	Status
5.1	For each event type, verify that clicking the type button displays only markers of that type.	Clicking the type button displays only markers of that type.	As Expected	Pass

5.2	Verify that clicking the type again button again removes the filter and shows all markers.	Clicking the type again button again removes the filter and shows all markers.	As Expected	Pass

5.1


Following
Random
WhatsOn
Legend
Contact
Zumba Class with Fuego Flow

5.2


Following
Random
WhatsOn
Legend
Contact
Zumba Class with Fuego Flow

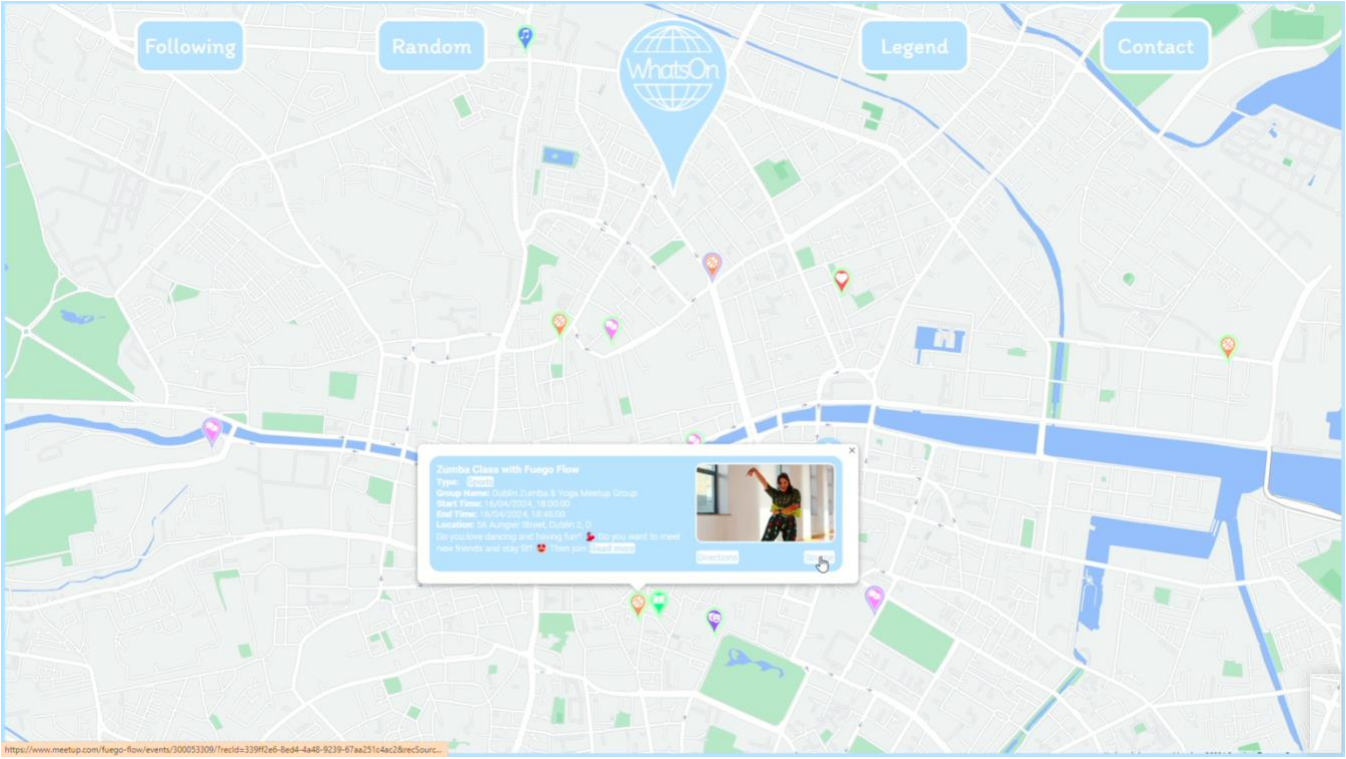
Conor Judge - Use Case 6 - Event Source

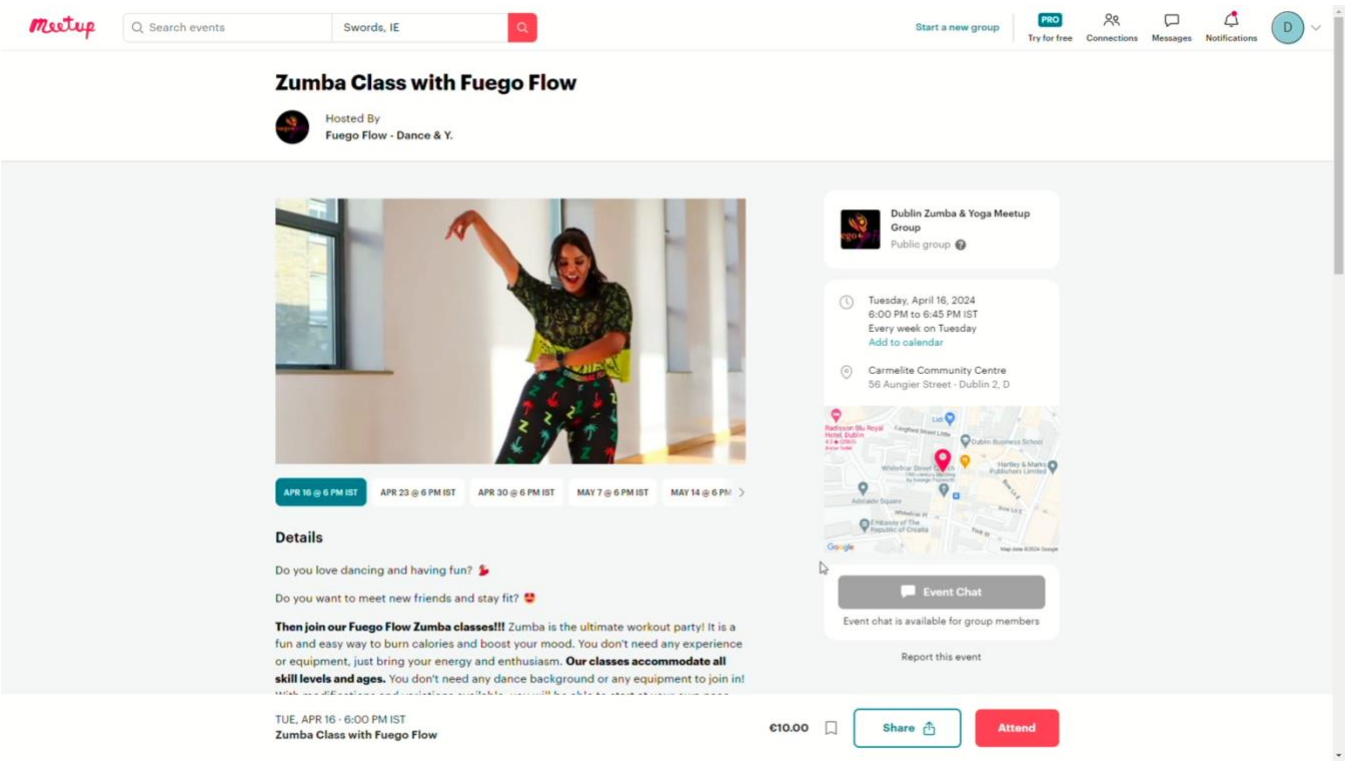
Test Case ID	BU_006	**Version**	1.0	**Test Case Description**	Use Case 6 - Event Source
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Event Source Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

S #	Step Details	Expected Results	Actual Results	Status
6.1	Within an info window, verify that clicking the source button redirects to the correct event source site.	Clicking the source button redirects to the correct event source site.	As Expected	Pass

6.1


Following
Random
WhatsOn
Legend
Contact
Zumba Class with Fuego Flow


meetup
Search events
Swords, IE
Start a new group
Try for free
Connections
Messages
Notifications
Zumba Class with Fuego Flow
Hosted By
Fuego Flow - Dance & Y.
Dublin Zumba & Yoga Meetup Group
Public group
Tuesday, April 16, 2024
6:00 PM to 6:45 PM IST
Every week on Tuesday
Add to calendar
Carmelite Community Centre
56 Aungier Street · Dublin 2, D
APR 16 @ 6 PM IST
APR 23 @ 6 PM IST
APR 30 @ 6 PM IST
MAY 7 @ 6 PM IST
Details
Do you love dancing and having fun?
Do you want to meet new friends and stay fit?
Then join our Fuego Flow Zumba classes!!! Zumba is the ultimate workout party! It is a fun and easy way to burn calories and boost your mood. You don't need any experience or equipment, just bring your energy and enthusiasm. Our classes accommodate all skill levels and ages. You don't need any dance background or any equipment to join in!
Event Chat
Event chat is available for group members
Report this event
TUE, APR 16 · 6:00 PM IST
Zumba Class with Fuego Flow
€10.00
Share
Attend

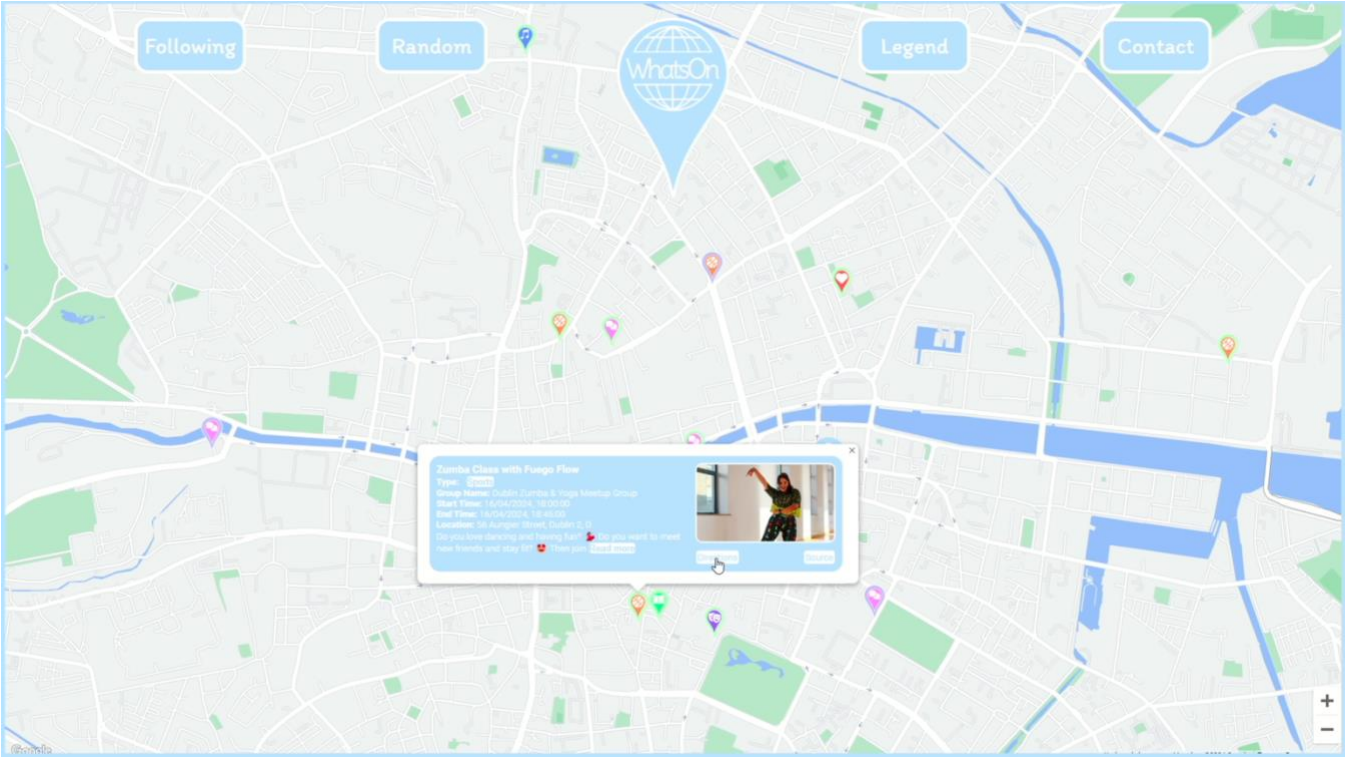
Conor Judge - Use Case 7 - Event Directions

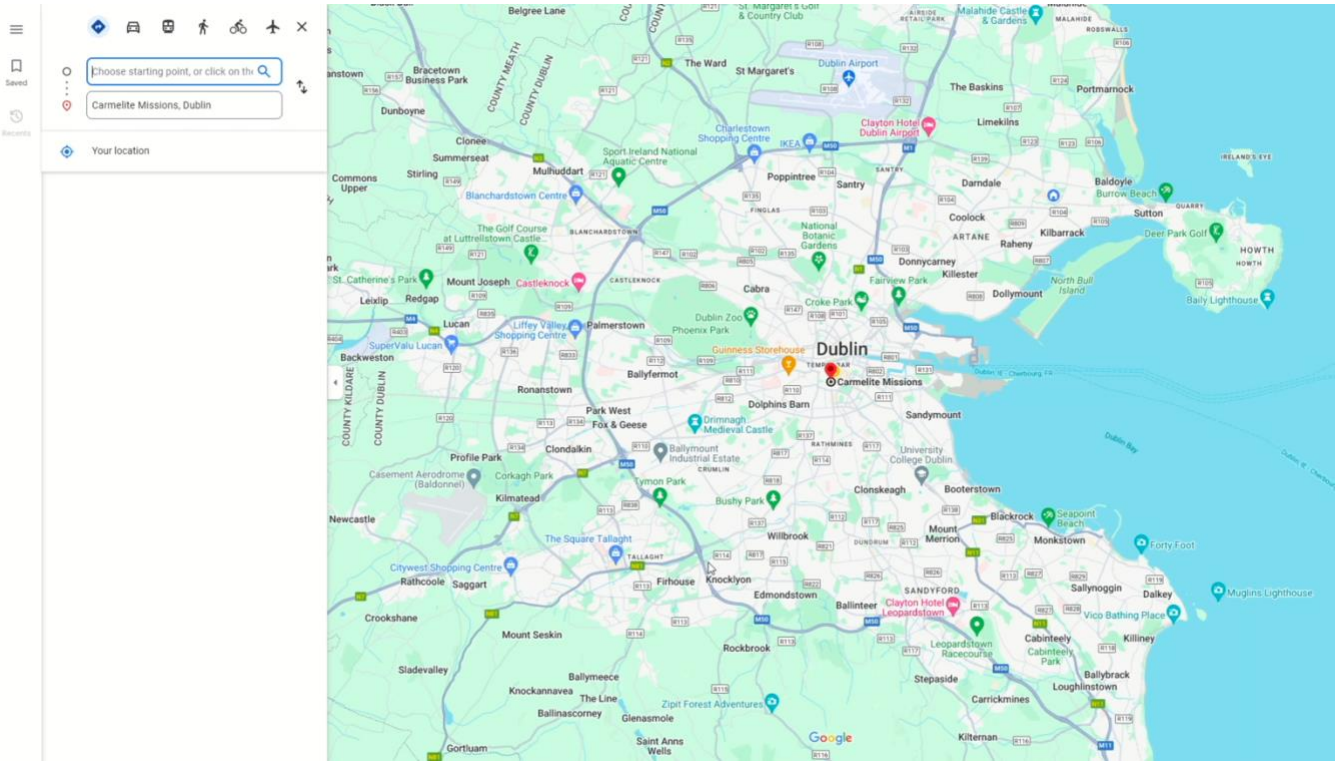
Test Case ID	BU_007	**Version**	1.0	**Test Case Description**	Use Case 7 - Event Directions
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Event Directions Functionality
Tester's Name	Conor	**Date Tested**	01/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

S #	Step Details	Expected Results	Actual Results	Status
7.1	Within an info window, verify that clicking the directions button redirects to Google Maps at the correct location.	Clicking the directions button redirects to Google Maps at the correct location.	As Expected	Pass

7.1


Following
Random
WhatsOn
Legend
Contact
Zumba Class with Fuego Flow


Choose starting point, or click on the
Carmelite Missions, Dublin
Your location
Dublin
Carmelite Missions
Google

Conor Judge - Event Icon Creation- Decision Table Testing

Test Case ID	DT_001	**Version**	1.0	**Test Case Description**	Event Icon Creation
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking Event Icon Creation functionality
Tester's Name	Conor	**Date Tested**	02/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API
		5	eventsData.json

	1.1	2.1	3.1	4.1	5.1	6.1
Event Type	Charity	Cultural	Education	Entertainment	Social	Sports
Time < 16 hrs	T	T	T	T	T	T
Time 16-24 hrs	F	F	F	F	F	F
Time > 24 hrs	F	F	F	F	F	F
Expected Icon URL	ChS.png	CuS.png	EdS.png	EnS.png	SoS.png	SpS.png
Actual Icon URL	As Expected	As Expected	As Expected	As Expected	As Expected	As Expected
Status	Pass	Pass	Pass	Pass	Pass	Pass

	1.2	2.2	3.2	4.2	5.2	6.2
Event Type	Charity	Cultural	Education	Entertainment	Social	Sports
Time < 16 hrs	F	F	F	F	F	F
Time 16-24 hrs	T	T	T	T	T	T
Time > 24 hrs	F	F	F	F	F	F
Expected Icon URL	ChO.png	CuO.png	EdO.png	EnO.png	SoO.png	SpO.png
Actual Icon URL	As Expected	As Expected	As Expected	As Expected	As Expected	As Expected
Status	Pass	Pass	Pass	Pass	Pass	Pass

	1.3	2.3	3.3	4.3	5.3	6.3
Event Type	Charity	Cultural	Education	Entertainment	Social	Sports
Time < 16 hrs	F	F	F	F	F	F
Time 16-24 hrs	F	F	F	F	F	F
Time > 24 hrs	T	T	T	T	T	T
Expected Icon URL	ChE.png	CuE.png	EdE.png	EnE.png	SoE.png	SpE.png
Actual Icon URL	As Expected	As Expected	As Expected	As Expected	As Expected	As Expected
Status	Pass	Pass	Pass	Pass	Pass	Pass

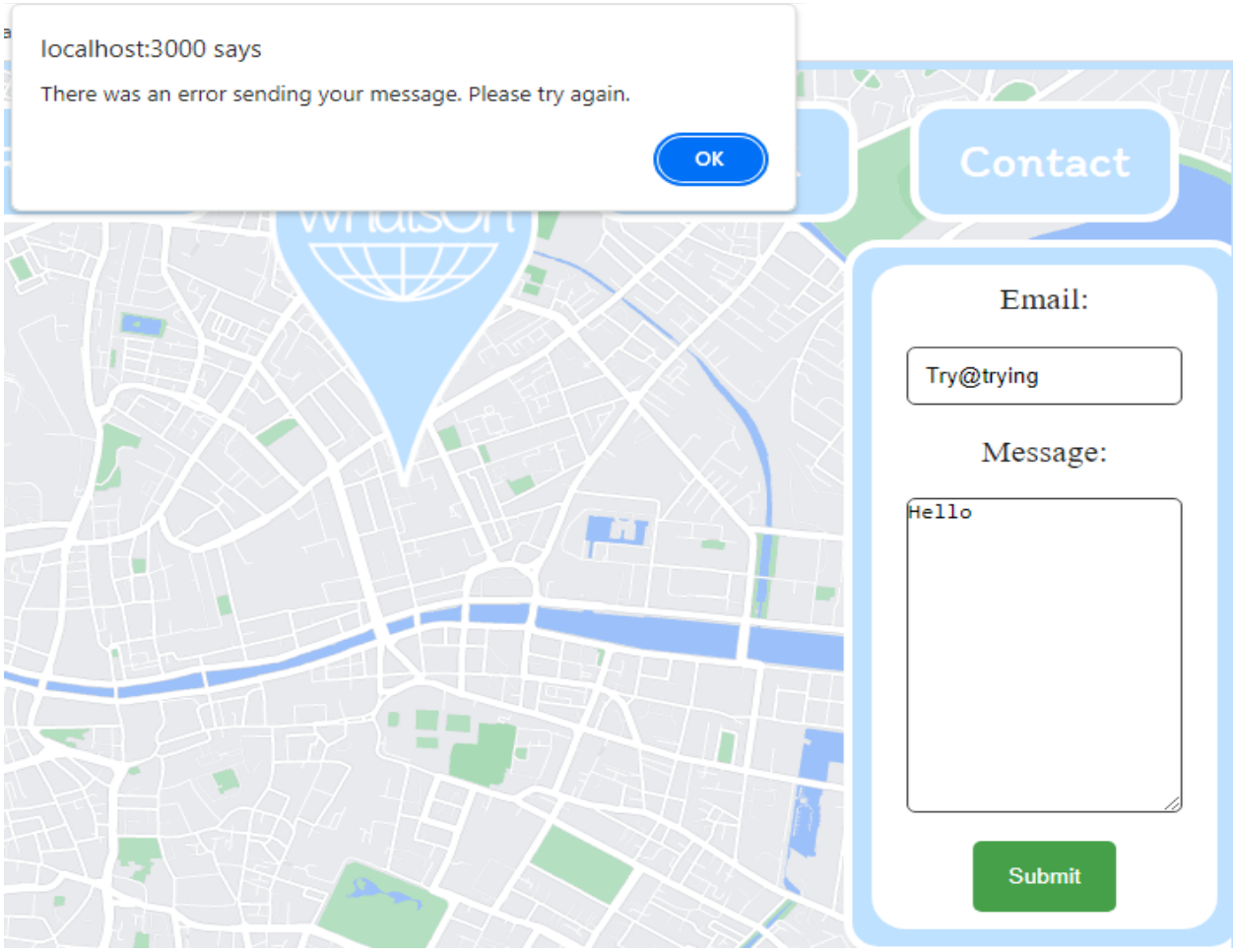
David O'Connor - Contact Form Field - Equivalence Testing

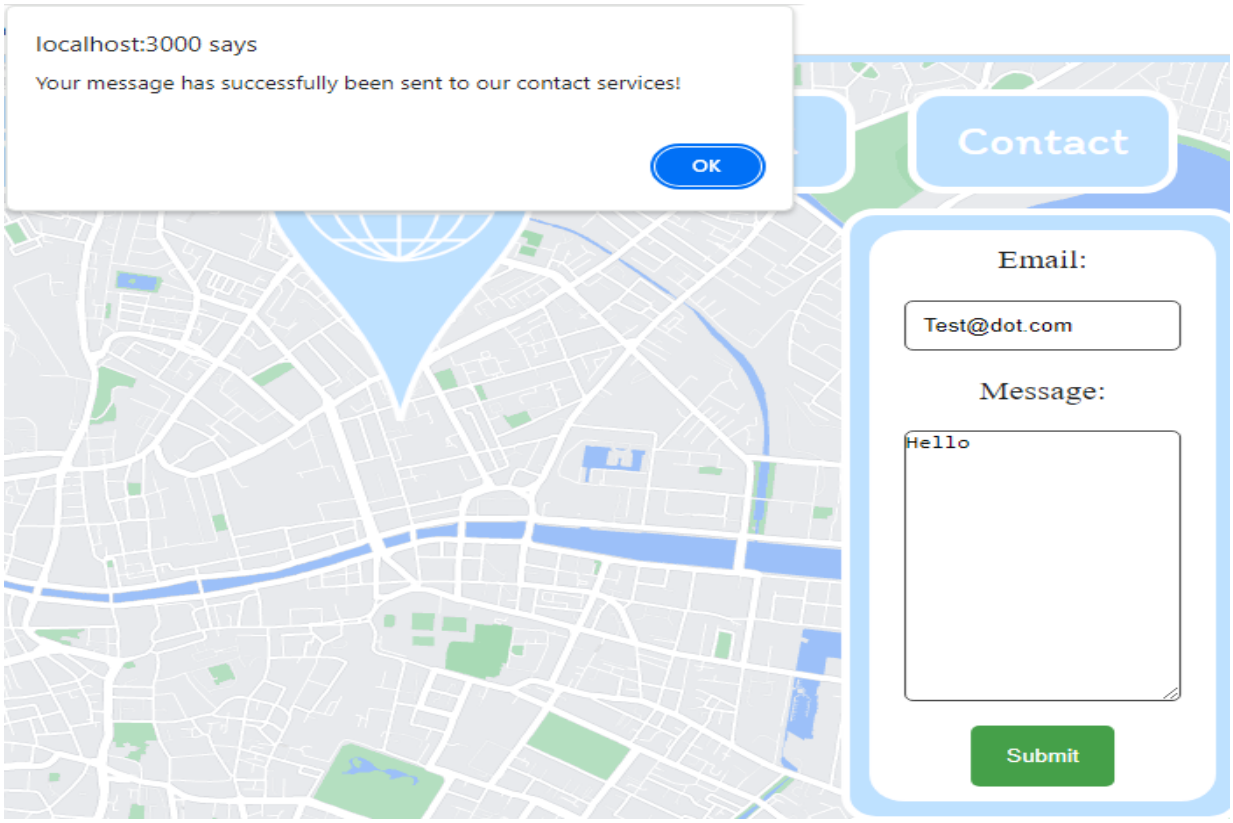
Test Case	Description	Steps
D01	Testing the parameters of the email field in the contact form module	The user must load the page The user must press onto contact button The user must enter their email

ID	Condition	Valid Equivalence Classes	Invalid Equivalence Class
E01	Email has a word then@followed by an email address then .com	Test@dot.com	Try@trying
E02	Email must contain @	Flow@	Flow
E03	Email must contain a word or address after @	Man@United	Man@

Results of E01:

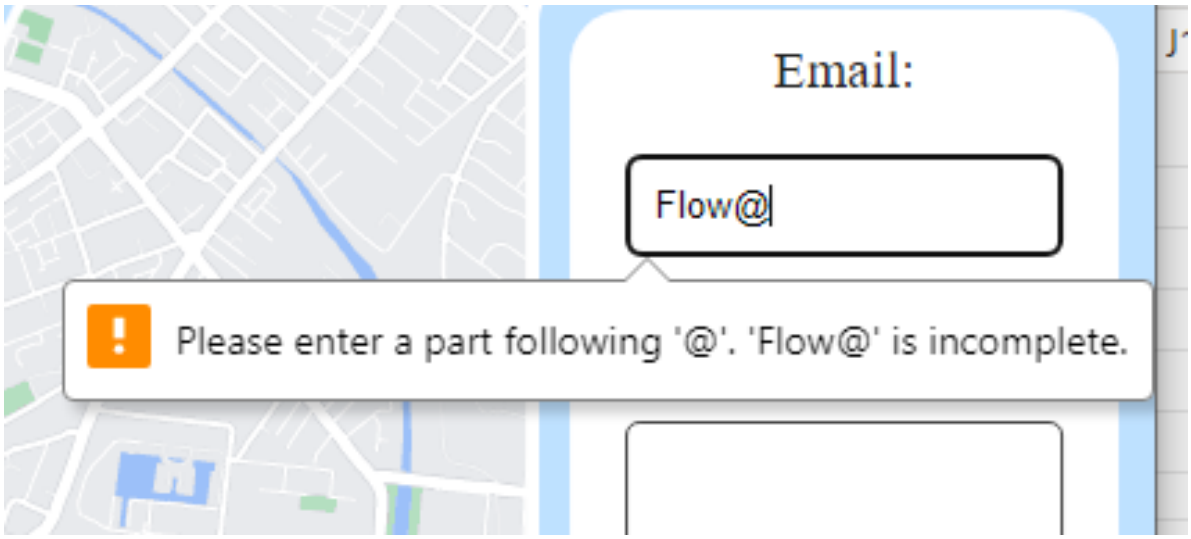
Expected	Actual	Pass/Fail
Will not accept an email an unfinished email address	As Expected	Pass

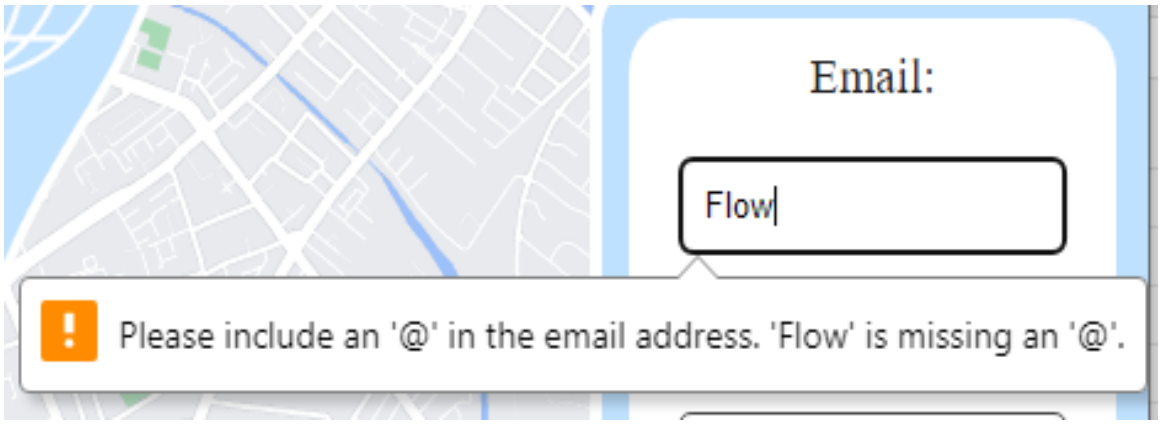



localhost:3000 says
Your message has successfully been sent to our contact services!
OK
Contact
Email:
Test@dot.com
Message:
Hello
Submit

Results of E02:

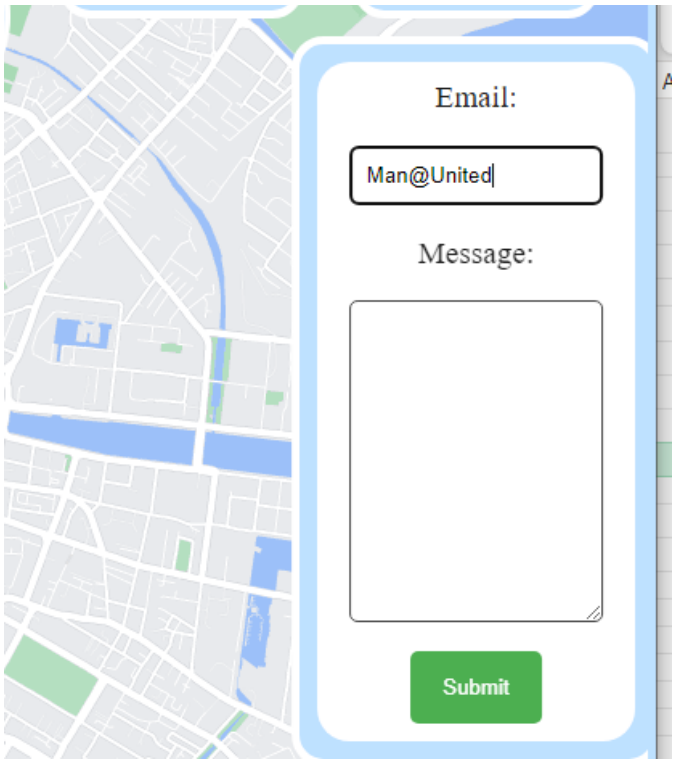
Expected	Actual	Pass/Fail
A message will be thrown if an @ is not in the email	As Expected	Pass

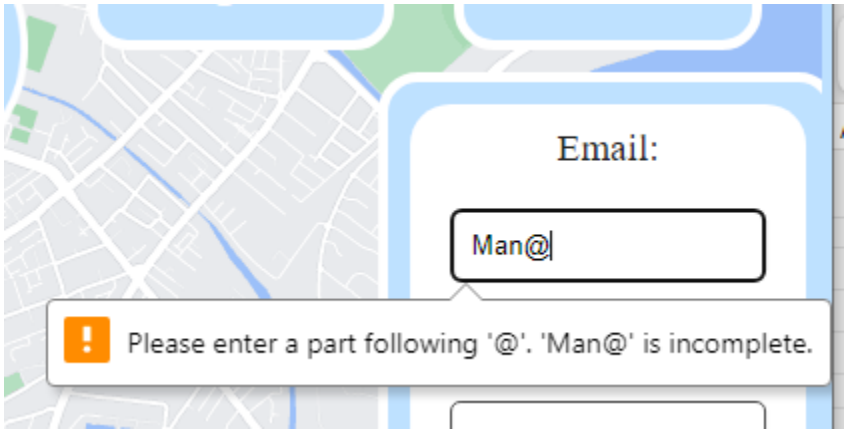




Results of E03:

Expected	Actual	Pass/Fail
A message will be thrown if word/address is not added after the @	As Expected	Pass



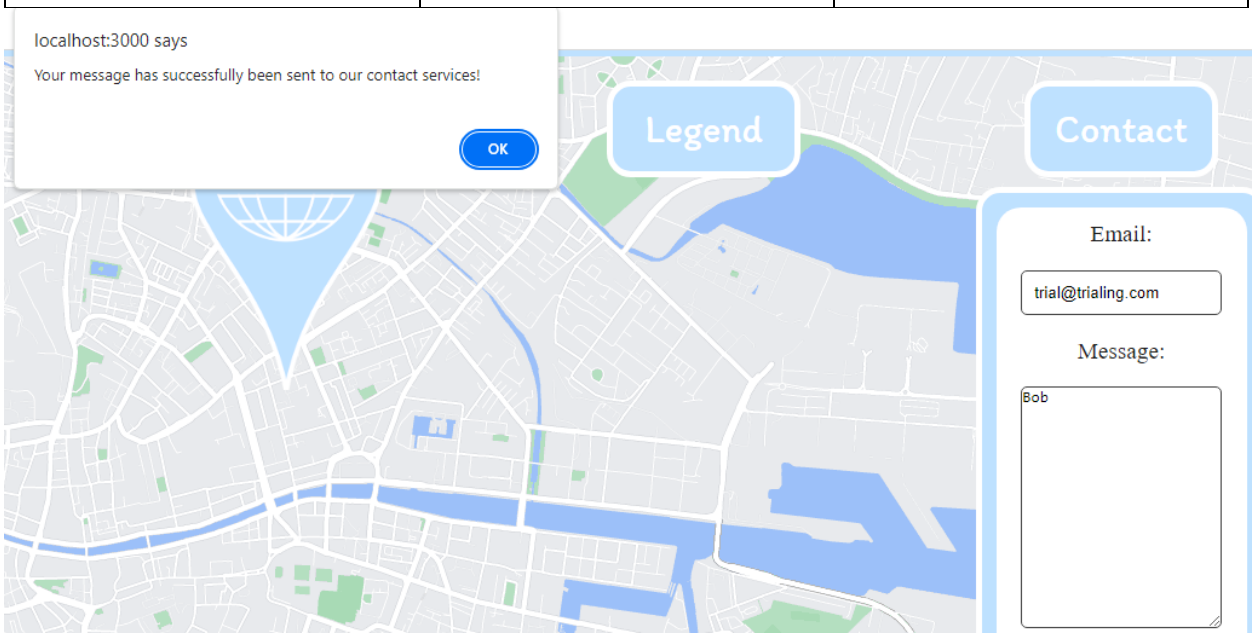


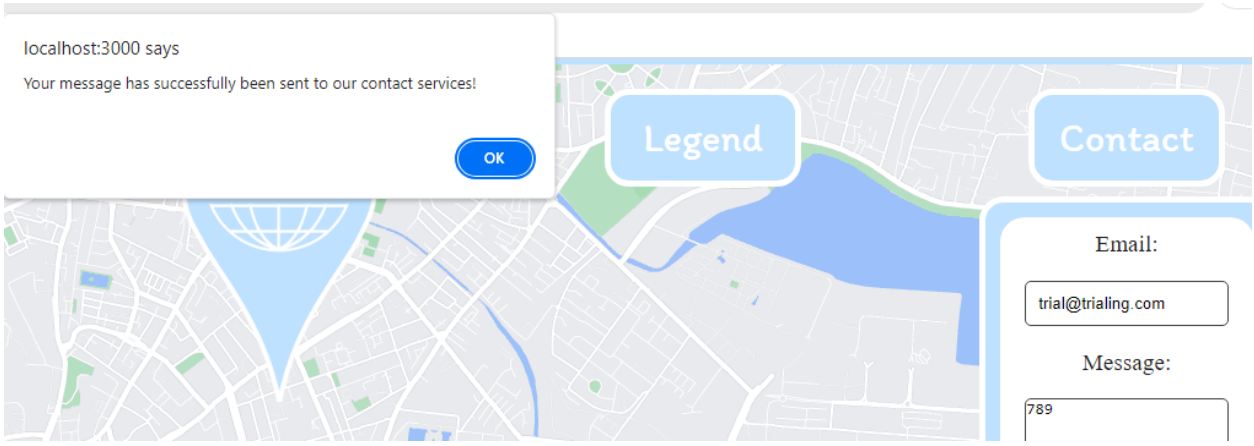
Test Case	Description	Steps
D02	Testing the parameters of the message field in the contact form module	The user must load the page The user must press onto contact button The user must enter their email The user must enter their message

ID	Condition	Valid Equivalence	Invalid Equivalence
M01	The message contains just letters	Bob	789
M02	The message contains just numbers	89	Eight
M03	The message box contains both alphabet letters and numerical values	Hello 1234	(blank)

Results of M01:

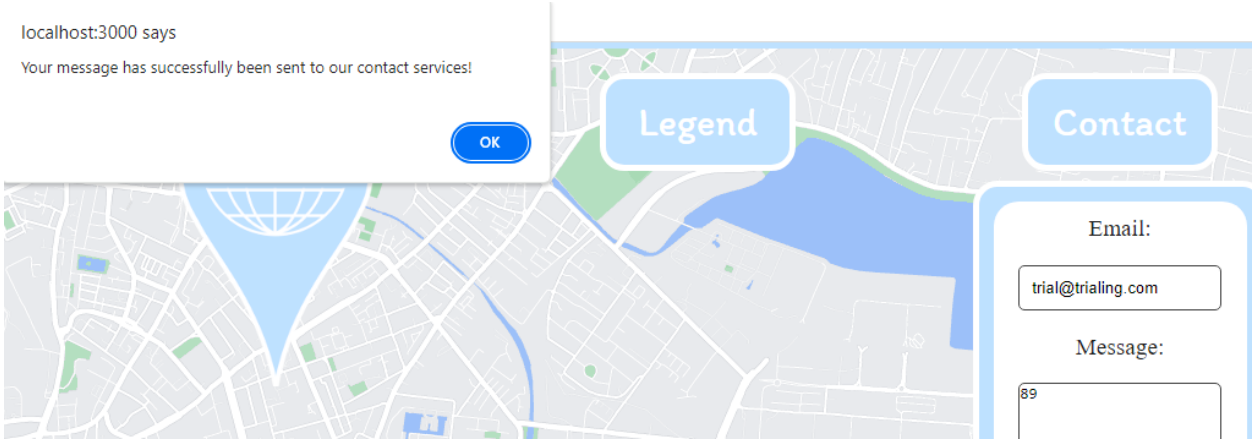
Expected	Actual	Pass/Fail
Won’t accept numbers	Accepts Numbers	Fail

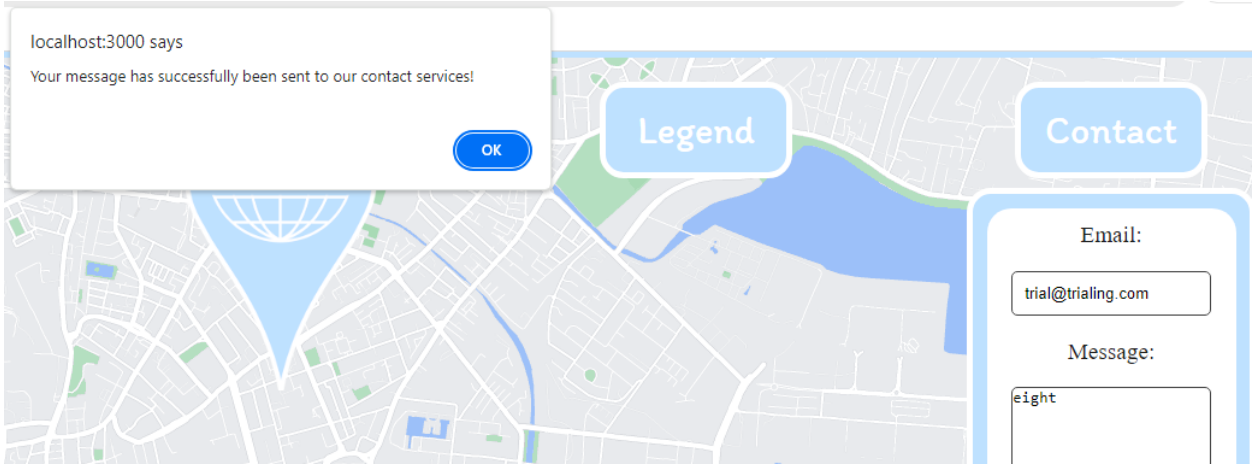




Results of M02:

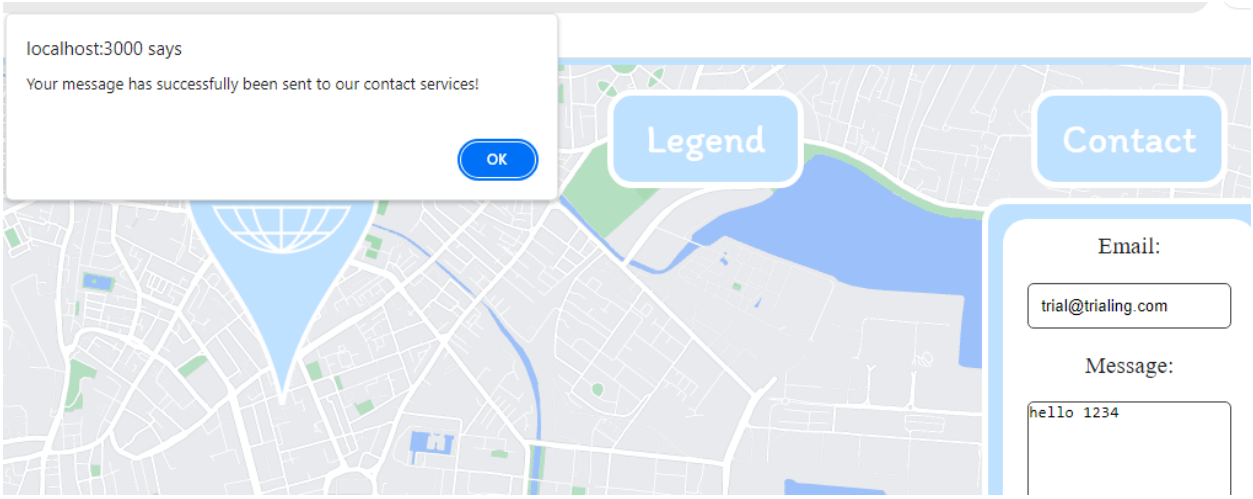
Expected	Actual	Pass/Fail
Won’t accept Alphabetic letters	Accepts Alphabetic letters	Fail

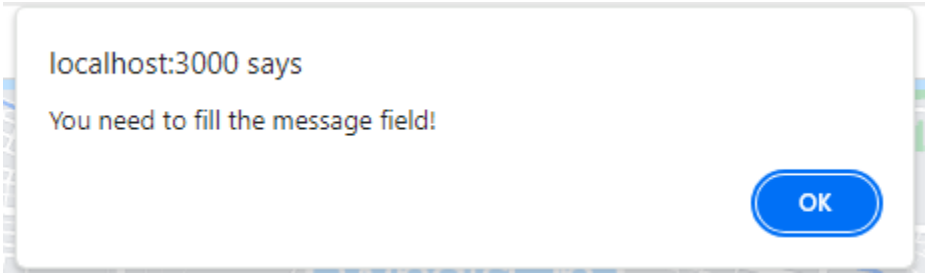




Results of M03:

Expected	Actual	Pass/Fail
Won't accept an empty field	As expected,	Pass





4.0 Test Cases Designed Using White-Box Testing Techniques

Tester's Name	Feature	Technique
Eoin Fitzsimons	Toggle Modal	Branch Coverage
Conor Judge	Map initialisation	Statement Coverage
Conor Judge	Filter Events	Branch Coverage
David O'Connor	Sign up (register/user route)	Statement Coverage

Eoin Fitzsimons - Test the functionality of the toggleModal function – Branch Coverage

Test Case ID	**WO_003**		
Test Case Description	Test the functionality of the toggleModal function		
Created By	Eoin		
Date Tested	05/05/2024		
Tester's Name	Eoin		
Test Data	modalId values		
Test Scenario	Verify that the toggleModal function correctly opens and closes the modal based on user interactions		
Test Case (Pass/Fail/Not Executed)	Pass		
1	Run the Cypress script	Script starts running	Pass
2	Script navigates to the web application	Web application loads	Pass
3	Script clicks the '.item.following' button with viewport 500 x 600	The 'Following' modal should be visible and centered on the screen	Pass
4	Script clicks the '.item.legBTN' button with viewport 500 x 600	The 'Following' modal should not be visible and the 'Legend' modal should be visible and centered on the screen	Pass
5	Script clicks the '.item.following' button with viewport 800 x 600	The 'Following' modal should be visible and positioned below the button	Pass
6	Script clicks the '.item.legBTN' button with viewport 800 x 600	The 'Following' modal should not be visible and the 'Legend' modal should be visible and positioned below the button	Pass
7	Script clicks the '.item.following' button with viewport 500 x 600 and currentModal not null	The 'Following' modal should be visible and centered on the screen	Pass
8	Script clicks the '.item.legBTN' button with viewport 500 x 600 and currentModal not null	The 'Following' modal should not be visible and the 'Legend' modal should be visible and centered on the screen	Pass
9	Script clicks the '.item.following' button with viewport 800 x 600 and currentModal not null	The 'Following' modal should be visible and positioned below the button	Pass
10	Script clicks the '.item.legBTN' button with viewport 800 x 600 and currentModal not null	The 'Following' modal should not be visible and the 'Legend' modal should be visible and positioned below the button	Pass

The branch coverage is 100% as it covers both landscape and portrait dimensions.

```
// Function to toggle the modal
function toggleModal(modalId, event) {
  // If a modal is currently open, close it
  if (currentModal) {
    //  Close the current modal
    currentModal.style.display = "none";
  }
  // Open the new modal
  currentModal = document.getElementById(modalId);
  //  Set the display style to block
  currentModal.style.display = "block";
  // Check if the screen width is 750px or less
  if (window.innerHeight > window.innerWidth) {
    // Set the display style to block
    currentModal.style.display = "block";
    // Set the position to fixed
    currentModal.style.position = "fixed";
    // Center the modal vertically
    currentModal.style.top = "50%";
    // Center the modal horizontally
    currentModal.style.left = "50%";
    // Translate the modal to center it
    currentModal.style.transform = "translate(-50%, -50%)";
  } else {
    // Get the position of the button
    var button = event.target;
```

```
// Get the position of the button
var rect = button.getBoundingClientRect();

// Account for the border
var border = parseInt(getComputedStyle(button).borderWidth);

// Position the modal above the viewport
currentModal.style.top = -currentModal.offsetHeight + "px";

// Center the modal horizontally
currentModal.style.left = "50%";
// Slide the modal down to its final position below the button
currentModal.style.transform = "translateX(-50%)";

// Slide the modal down to its final position below the button
var finalTop = rect.bottom + window.scrollY - 15; // default value

// If the modal extends off the right edge of the screen, move it back
var modalRight = currentModal.offsetLeft + currentModal.offsetWidth;
if (modalRight > window.innerWidth) {
  currentModal.style.right = "0px";

  // If it's the legend modal, adjust the finalTop
  if (modalId === "legendModal") {
    finalTop = rect.bottom + window.scrollY + 12;
  }
}
```

```
  var currentTop = parseInt(currentModal.style.top);
  var intervalId = setInterval(function () {
    if (currentTop < finalTop) {
      currentTop += 2;
      currentModal.style.top = currentTop + "px";
    } else {
      clearInterval(intervalId);
    }
  }, 3); // Adjust this value to change the speed of the animation
}
```

Conor Judge - Map Initialization - Statement Coverage

Test Case ID	SC_001	**Version**	1.0	**Test Case Description**	Statement Coverage for Map Initialization
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking every line of code in Map Initialization Functionality
Tester's Name	Conor	**Date Tested**	02/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API

S #	Step Details	Expected Results	Actual Results	Status
SC_001.1	Verify that the map object is created with new google.maps.Map() function.	A new map object is created.	As Expected	Pass
SC_001.2	Verify that the map options are defined including center coordinates, zoom level, and disabled controls.	Map options are defined with centre at lat:53.349076911151386, lng:-6.242441879918039, zoom level:14 and disabled controls.	As Expected	Pass
SC_001.3	Verify that the map is initialized with defined options.	Map initializes with specified options.	As Expected	Pass

```
function initMap() {
  map = new google.maps.Map(document.getElementById("map-div"), {
    center: { lat: 53.349076911151386, lng: -6.242441879918039 },
    zoom: 14,
    mapId: "13185b1ffbbba3af",
    mapTypeControl: false,
    fullscreenControl: false,
    streetViewControl: false,
  });
```

1. **function initMap() {**

   This line is covered because the initMap function is being called

2. **map = new google.maps.Map(document.getElementById("map-div"), {**

   This line is covered by the test that verifies a new map object is created. The new google.maps.Map() constructor is called with two arguments: the result of document.getElementById("map-div") and an options object.

3. **center: { lat: 53.349076911151386, lng: -6.242441879918039 },**

   This line is covered by the tests that check the map options and the initialized map. They verify that the center option is correctly set.

4. **zoom: 14,**

   This line is covered by the tests that check the map options and the initialized map. They verify that the zoom option is correctly set.

5. **mapId: "13185b1ffbbba3af",**

   This line is covered by the tests that check the map options and the initialized map. They verify that the mapId option is correctly set.

6. **mapTypeControl: false,**

   This line is covered by the tests that check the map options and the initialized map. They verify that the mapTypeControl option is correctly set.

7. **fullscreenControl: false,**

   This line is covered by the tests that check the map options and the initialized map. They verify that the fullscreenControl option is correctly set.

8. **streetViewControl: false,**

   This line is covered by the tests that check the map options and the initialized map. They verify that the streetViewControl option is correctly set.

Statement Coverage = $\frac{8}{8}$ = 100%

Conor Judge - Filter Events - Branch Coverage

Test Case ID	BC_001	**Version**	1.0	**Test Case Description**	Verify that the filterEvents function filters the visible markers on the map based on their event type. This should cover both branches of the if condition inside filterEvents.
Created By	Conor	**Reviewed By**		**Test Scenario**	Checking every line of code in Event Filtering Functionality
Tester's Name	Conor	**Date Tested**	03/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Access to WhatsOn Web Application	1	Google Maps API
2	WhatsOn is correctly loaded on the browser	2	Maps JavaScript API
3	Events are loaded and displayed as markers on the map	3	Google Cloud Console
4	Events info window is open	4	Maps JavaScript API
		5	eventsData.json

S #	Step Details	Expected Results	Actual Results	Status
BC_001.1	Call filterEvents with an event type parameter	Only markers of specified event type are visible on the map. Markers of other event types are hidden.	As Expected	Pass
BC_001.2	Verify that other event types' markers are not visible on the map	Markers of other event types are hidden. Only markers of the specified event type are visible.	As Expected	Pass
BC_001.3	Call filterEvents with a non-existent or invalid event type	No changes occur, all previously visible markers remain visible.	As Expected	Pass

```
window.filterEvents = function (eventType) {
  if (currentFilter === eventType) {
    currentFilter = null;
    markers.forEach((marker) => {
      marker.setVisible(true);
    });
  } else {
    currentFilter = eventType;
    markers.forEach((marker) => {
      marker.setVisible(marker.eventType === currentFilter);
    });
  }
};
```

1. **function filterEvents(eventType) {**

   This line is covered because we're testing if calling this function with different parameters affects marker visibility.

2. **if (currentFilter === eventType) {**

   This line is covered by BC_001.3, checking if the current filter is the same as the eventType parameter.

3. **currentFilter = null;**

   Covered by BC_001.3, where it confirms that the current filter is set to null.

4. **markers.forEach((marker) => {**

   This line is covered by looping through all available markers to check their visibility status.

5. **marker.setVisible(true);**

   Covered by BC_001.3, where it confirms that all markers become visible.

6. **} else {**

   Covered by BC_001.1 and BC_001.2, where it confirms that if a new eventType is passed, the current filter is updated and markers are filtered.

7. **currentFilter = eventType;**

   Covered by BC_001.1 and BC_001.2, where it confirms that the current filter is updated to the new eventType.

8. **markers.forEach((marker) => {**

   This line is covered by looping through all available markers to check their visibility status.

9. **marker.setVisible(marker.eventType === currentFilter);**

   Covered by BC_001.1 and BC_001.2, where it confirms that only markers of the specified eventType are visible.

Statement Coverage = $\frac{9}{9}$ = 100%

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Pass/ Fail
TU01	The user must load the site or load the local server through nodemon The user must press onto the following button The user must then choose sign up choice The user should than press the sign up When the user has press signup that should then load the register function from the server From there it will check if the email is already in use If the email is not in use, it should start generating a salt Once the salt is generate it should encrypt the password and write the user details to a json file Once Completed writing to file it should give a status code of 200	userEmail: testEmail@email.com Password: test123	User should be able to create an account and their password encryption should appear in the console of the backend	As expected,. Encrypted password $2b$10$6bkQEAOkGr493skNV3oXYOUicrhaufET/lKSMfc4GOZZxDxf0hnCi status code Post/ Users/ Register 200	Pass

The Result of Test Case TU01:

```
$2b$10$6bkQEAOkGr493skNV3oXYO
$2b$10$6bkQEAOkGr493skNV3oXYOUicrhaufET/lKSMfc4GOZZxDxf0hnCi
POST /users/register 200 58.317 ms - 58
```


```
{"email":"testEmail@email.com","password":"$2b$10$6bkQEAOkGr493skNV3oXYOUicrhaufET/lKSMfc4GOZZxDxf0hnCi"}]
```


Code Coverage of Test Case TU01

```
app.post("/users/register", async (req, res) => {
```

```
try {
  const salt = await bcrypt.genSalt();
  const hashedPassword = await bcrypt.hash(req.body.password, salt);
  console.log(salt);
  console.log(hashedPassword);
  const newUser = { email: req.body.email, password: hashedPassword };

  let users = [];

  try {
    const fileContent = fs.readFileSync(
      path.join(__dirname, "User Details", "users.json"),
      "utf8"
    );
    users = JSON.parse(fileContent);
    if (!Array.isArray(users)) {
      users = [];
    }
  } catch (error) {

    if (error.code !== "ENOENT") {
      console.error(error);
      res.status(500).json({
        success: false,
        message: "An error occurred while reading the users file.",
      });
      return;
    }
  }
```

```
const existingUser = users.find((user) => user.email === newUser.email);
if (existingUser) {
  res
    .status(400)
    .json({ success: false, message: "Email already in use." });
  return;
}
// Add the new user
users.push(newUser);
  try {
    fs.writeFileSync(
      "./User Details/users.json",
      JSON.stringify(users),
      "utf8"
    );
  } catch (error) {
    console.error(error);
    res.status(500).json({
      success: false,
      message: "An error occurred while writing to the users file.",
    });
```

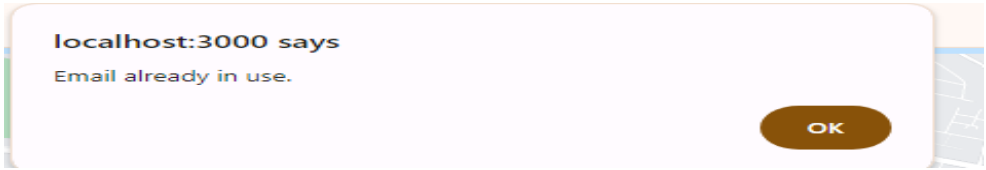
```
    return;
  }
 } catch (error) {
  res.status(500).json({
    success: false,
    message: "An error occurred during registration.",
  });
  console.log(error);
  return;
 }
 res.json({ success: true, message: "Account created successfully." });
});
```

Statement Coverage of Test Case TU01:

Statement Coverage = $\frac{26}{59}$ = 44.06%

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Pass/ Fail
TU02	The user must load the site or load the local server through nodemon The user must press onto the following button The user must then choose sign up choice The user should than press the sign up When the user has press signup that should then load the register function from the server From there it will check if the email is already in use When the server-side function grabs that the email is in use. It should notify the user the that the email is already in use. The server side should give back a status of 400	Email: Test Email@ email.com Password: Test 234	User should not be able to create an account because of the email is already in use It should notify the user through an alert Give status 404	As expected.	Pass

The result of test case TU02:




```
POST /users/register 400 58.219 ms - 51
```

```
app.post("/users/register", async (req, res) => {
  try {
    const salt = await bcrypt.genSalt();
    const hashedPassword = await bcrypt.hash(req.body.password, salt);
    console.log(salt);
    console.log(hashedPassword);
    const newUser = { email: req.body.email, password: hashedPassword };

    let users = [];

    try {
      const fileContent = fs.readFileSync(
        path.join(__dirname, "User Details", "users.json"),
        "utf8"
      );
      users = JSON.parse(fileContent);
      if (!Array.isArray(users)) {
        users = [];
      }
    } catch (error) {
      // If file does not exist, ignore the error
      if (error.code !== "ENOENT") {
        console.error(error);
        res.status(500).json({
          success: false,
          message: "An error occurred while reading the users file.",
        });
        return;
      }
```

```
  }

  const existingUser = users.find((user) => user.email === newUser.email);
  if (existingUser) {
    res
      .status(400)
      .json({ success: false, message: "Email already in use." });
    return;
  }



users.push(newUser);

  // Write the updated users back to the file
  try {
    fs.writeFileSync(
      "./User Details/users.json",
      JSON.stringify(users),
      "utf8"
    );
  } catch (error) {
    console.error(error);
    res.status(500).json({
      success: false,
      message: "An error occurred while writing to the users file.",
    });
    return;
  }
```

```
} catch (error) {
  res.status(500).json({
    success: false,
    message: "An error occurred during registration.",
  });
  console.log(error);
  return;
}
res.json({ success: true, message: "Account created successfully." });
});
```

Statement Coverage of Test Case TU02:

Statement Coverage = $\frac{26}{59}$ = 44.06%

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Pass/ Fail
TU03	The user must load the site or load the local server through nodemon Then the user must close the server after loading the page The user must press onto the following button The user must then choose sign up choice The user should than press the sign up When the user has press signup that should then load the register function from the server The function should not load because the server is down. From there the user should receive a status of 500 from the console with error messages	Email: Test3 @ email.com Password: Test 999	User should not be able to create an account because of the server being down. They should receive a status code of 500	The user was not able to create an account as I expected, But it did not give back a status code of 500 nor did it give any of the error messages I had provided in function	fail

Results of Test Case TU03

```
Failed to load resource:                    :3000/users/register:1
net::ERR_CONNECTION_REFUSED
```

Name	Status	Type	Initiator	Size	Time
GetViewportInfo	200	xhr	js?key=AIzaSyCI	3.3...	3.33 s
transparent.png	200	png	common.js:224	(di...	4 ms
vt?pb=!1m4!1m3!1i12!2i1...	200	script	common.js:43	1.7...	2.87 s
closedhand_8_8.cur	200	bmp	Other	(di...	3 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	16....	115 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	23....	231 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	26....	242 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	15....	143 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	17....	198 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	14....	207 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	12....	222 ms
tiles?map_id=13185b1ffbb...	200	png	common.js:223	15....	128 ms
vt?pb=!1m4!1m3!1i12!2i1...	200	script	common.js:43	1.7...	85 ms
register	(failed) n...	fetch	(index):877	0 B	2.37 s

```
app.post("/users/register", async (req, res) => {
  try {
    const salt = await bcrypt.genSalt();
    const hashedPassword = await bcrypt.hash(req.body.password, salt);
    console.log(salt);
    console.log(hashedPassword);
    const newUser = { email: req.body.email, password: hashedPassword };

    let users = [];



    try {
      const fileContent = fs.readFileSync(
        path.join(__dirname, "User Details", "users.json"),
        "utf8"
      );
      users = JSON.parse(fileContent);
      if (!Array.isArray(users)) {
        users = [];
      }
    } catch (error) {
      if (error.code !== "ENOENT") {
        console.error(error);
        res.status(500).json({
          success: false,
          message: "An error occurred while reading the users file.",
        });
```

```
      return;
    }
  }
  const existingUser = users.find((user) => user.email === newUser.email);
  if (existingUser) {
    res
      .status(400)
      .json({ success: false, message: "Email already in use." });
    return;
  }
  // Add the new user
  users.push(newUser);

  // Write the updated users back to the file
  try {
    fs.writeFileSync(
      "./User Details/users.json",
      JSON.stringify(users),
      "utf8"
    );
  }
catch (error) {
    console.error(error);
    res.status(500).json({
      success: false,
      message: "An error occurred while writing to the users file.",
    });
    return;
  }
```

```
 } catch (error) {
   res.status(500).json({
     success: false,
     message: "An error occurred during registration.",
   });
   console.log(error);
   return;
 }
 res.json({ success: true, message: "Account created successfully." });
});
```

Statement Coverage of Test Case 3:

Statement Coverage = $\frac{0}{59}$ = 0%

Overall Statement Coverage for Test Case "TU":

This is concluded from the three conducted test cases I have provided above. The total percentage of the all the code of 59 lines of the (/users/register) route function is 55.93%.

In terms of fulfilling the main purpose of the code checking if the user email is already in use and denying the creation of an account and to create an account if the user's email is already not in use. The percentage of that is 100%.

5.0 Automated Testing

Tester's Name	Feature
Eoin Fitzsimons	Loading the Map with the Encoded key
Conor Judge	Map Initialization
David O'Connor	User account login

Eoin Fitzsimons - Loading the Map with the Encoded key

Test Case ID	WO_002			
Test Case Description	Test the functionality of the application with the encoded API key			
Created By	Eoin			
Date Tested	(1/5/24)			
Tester's Name	Eoin			
Test Data	Encoded API Key = '9067dc23064fbdb794f79053211c2c4395c0d8bea8208194f8583ba42b8946d09acadcc4252e72bb'			
Test Scenario	Verify that the application functions correctly when the encoded API key is entered into the prompt			
Test Case (Pass/Fail/Not Executed)	Pass			
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Run the Selenium script	Script starts running	As Expected	Pass
2	Script navigates to the web application	Web application loads	As Expected	Pass
3	Script enters the encoded API key when prompted	Encoded API key is entered automatically	As Expected	Pass
4	Script accepts the prompt	Prompt is accepted and web application continues to load	As Expected	Pass
5	Check the functionality of the application	Application functions as expected with the entered API key	As Expected	Pass

```
import time

from selenium import webdriver

from selenium.webdriver.edge.service import Service

from selenium.webdriver.edge.options import Options

from selenium.webdriver.common.alert import Alert


# Path to Edge WebDriver

webdriver_service =
Service('C:\\Users\\eoin0\\Downloads\\edgedriver_win64\\msedgedriver.exe')
```

```
# Create EdgeOptions
options = Options()

# Create Edge WebDriver with options
driver = webdriver.Edge(service=webdriver_service, options=options)

# Navigate to a blank page
driver.get('about:blank')
# Wait for 1 second
time.sleep(11)
# Navigate to the web application
driver.get('http://localhost:3000')

# Wait for the prompt to appear
driver.implicitly_wait(10)  # waits up to 10 seconds for the alert to appear

# Switch to the alert
alert = Alert(driver)

# Enter the API key into the prompt
alert.send_keys('9067dc23064fbdb794f79053211c2c4395c0d8bea8208194f8583ba42b8946d09acadcc4252e72bb')

# Wait for 1 second
time.sleep(1)

# Accept the prompt
alert.accept()
```

```
# Wait for 5 seconds
time.sleep(2)

# Refresh the page
driver.refresh()

# Wait for the prompt to appear again
driver.implicitly_wait(10)

# Switch to the alert
alert = Alert(driver)

# Enter the wrong API key into the prompt
alert.send_keys('wrong_key')

# Wait for 1 second
time.sleep(1)

# Accept the prompt
alert.accept()

# Wait for 5 seconds
time.sleep(2)

# Don't forget to quit the driver when you're done
driver.quit()
```

Conor Judge – Map Initialization

Test Case ID	UT_001	**Version**	1.0	**Test Case Description**	Unit Testing for Map Initialization
Created By	Conor	**Reviewed By**		**Test Scenario**	Validating the correctness of every line of code in Map Initialization Functionality
Tester's Name	Conor	**Date Tested**	02/05/2024	**Test Case Status**	Pass

S #	Prerequisites:	S #	Test Data
1	Codebase access to review and test the WhatsOn Web Application map initialization function.	1	Google Maps API
2	The development environment is set up and running, including all necessary dependencies to execute the map initialization function.	2	Maps JavaScript API
		3	Google Cloud Console
		4	Maps JavaScript API

S #	Step Details	Expected Results	Actual Results	Status
UT_001.1	Run the map initialization function and check if it returns a new google.maps.Map() object without errors.	A new google.maps.Map() object is returned without any errors.	As Expected	Pass
UT_001.2	Check if the map options including center coordinates, zoom level, and disabled controls are correctly defined within the function or received as parameters.	The function should either have these options predefined or should accept them as parameters and apply them correctly to create a map object with specified attributes.	As Expected	Pass
UT_001.3	Run a series of assertions to validate that the initialized map adheres to defined options including center coordinates, zoom level, and disabled controls The assertions confirm that all defined options are applied correctly during map initialization As Expected Pass	The assertions confirm that all defined options are applied correctly during map initialization.	As Expected	Pass

These tests were implemented using Mocha, a JavaScript test framework, and Chai, an assertion library. Sinon was used for creating stubs and JSDOM to simulate the DOM in Node.js.

test > map.test.js > describe('Map Initialization') callback > after() callback

```
// Importing required libraries and tools
import { expect } from 'chai';
import { describe, it, before, after } from 'mocha';
import { JSDOM } from 'jsdom';
import sinon from 'sinon';
```

```
// Create a global window object using JSDOM
global.window = new JSDOM().window;
global.document = window.document;
```

```
// Create a global google object with a maps property
global.google = {
  maps: {
    Map: function() {
      return {
        getCenter: () => testData.center,
        getZoom: () => testData.zoom,
        getMapTypeId: () => testData.mapId,
        getMapTypeControl: () => false,
        getFullscreenControl: () => false,
        getStreetViewControl: () => false,
      };
    },
  },
};
```

```
// Sample data for testing
const testData = {
  center: { lat:53.349076911151386, lng:-6.242441879918039 },
  zoom: 14,
  mapId: "13185b1ffbbba3af",
};
```

```
describe('Map Initialization', () => {
  let mapStub, getElementByIdStub;

  before(() => {
    // Stub the google.maps.Map constructor
    mapStub = sinon.stub(google.maps, 'Map');
    mapStub.returns({
      getCenter: () => testData.center,
      getZoom: () => testData.zoom,
      getMapTypeId: () => testData.mapId,
      getMapTypeControl: () => false,
      getFullscreenControl: () => false,
      getStreetViewControl: () => false,
    });

    // Stub the document.getElementById method
    getElementByIdStub = sinon.stub(document, 'getElementById');
    getElementByIdStub.returns(document.createElement('div'));
  });

  after(() => {
    // Restore the original functions after the tests have run
    mapStub.restore();
    getElementByIdStub.restore();
  });
```

UT_001.1

```
// SC_001.1 - Verify that the map object is created
it('SC_001.1 - Verify that the map object is created', () => {
  const map = new google.maps.Map(document.getElementById("map-div"), {
    center: testData.center,
    zoom: testData.zoom,
    mapId: testData.mapId,
    mapTypeControl: false,
    fullscreenControl: false,
    streetViewControl: false,
  });
  expect(map.getCenter()).to.deep.equal(testData.center);
  expect(map.getZoom()).to.equal(testData.zoom);
  expect(map.getMapTypeId()).to.equal(testData.mapId);
  expect(map.getMapTypeControl()).to.be.false;
  expect(map.getFullscreenControl()).to.be.false;
  expect(map.getStreetViewControl()).to.be.false;
});
```

UT_001.2

```
// SC_001.2 - Verify that the map options are defined
it('SC_001.2 - Verify that the map options are defined', () => {
  const mapOptions = {
    center: testData.center,
    zoom: testData.zoom,
    mapId: testData.mapId,
    mapTypeControl: false,
    fullscreenControl: false,
    streetViewControl: false,
  };
  expect(mapOptions.center).to.deep.equal(testData.center);
  expect(mapOptions.zoom).to.equal(testData.zoom);
  expect(mapOptions.mapId).to.equal(testData.mapId);
  expect(mapOptions.mapTypeControl).to.be.false;
  expect(mapOptions.fullscreenControl).to.be.false;
  expect(mapOptions.streetViewControl).to.be.false;
});
```

UT_001.3

```
  // SC_001.3 - Verify that the map is initialized with defined options
  it('SC_001.3 - Verify that the map is initialized with defined options', () => {
    const mapOptions = {
      center: testData.center,
      zoom: testData.zoom,
      mapId: testData.mapId,
      mapTypeControl: false,
      fullscreenControl: false,
      streetViewControl: false,
    };
    const map = new google.maps.Map(document.getElementById("map-div"), mapOptions);
    expect(map.getCenter()).to.deep.equal(testData.center);
    expect(map.getZoom()).to.equal(testData.zoom);
    expect(map.getMapTypeId()).to.equal(testData.mapId);
    expect(map.getMapTypeControl()).to.be.false;
    expect(map.getFullscreenControl()).to.be.false;
    expect(map.getStreetViewControl()).to.be.false;
  });
});
```

Execution Of UT_001.1, UT_001.2, UT_001.3

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS

PS C:\Users\conor\Documents\GitHub\Team-Project> npm test

> npm-init-package@1.0.0 test
> mocha



  Map Initialization
    ✓ SC_001.1 - Verify that the map object is created
    ✓ SC_001.2 - Verify that the map options are defined
    ✓ SC_001.3 - Verify that the map is initialized with defined options


  3 passing (11ms)

PS C:\Users\conor\Documents\GitHub\Team-Project>
```

David O'Connor - User account login

Test Case: Unit_01

Test Case Description	Test Data
In this test case will supply multiple scenarios for testing the user login route function. It will test when a user has a successful login and that it should come back with a status code of 200	Email: wwe@wwe.com Password: wwe
Expected Results	Actual Results
It should give back a status code of 200 and let the user successfully login	As Expected.

Unit_01 Code:

```
const request = require("supertest");

const app = require("./login.js");

const fs = require("fs");


describe("POST /users/login", () => {

  it("should respond with success true and a status 200 code when user is found and password matches", async () => {

    // Mocking the file content for users.json

    const mockFileContent = JSON.stringify([

      { email: "wwe@wwe.com", password:
"$2b$10$dsm0qv/PHQg21PZYJ7uM8.Nni8V4E3ZPtxaQhBPpPlMKRGvmGthg." }

    ]);


    // Mocking the readFileSync function

    jest.spyOn(fs, "readFileSync").mockReturnValueOnce(mockFileContent);


    const requestBody = { email: "wwe@wwe.com", password: "wwe" };
```

```
  const response = await request(app)
    .post("/users/login")
    .send(requestBody);

  expect(response.body).toEqual({ success: true });
 });
});
```

Test Case: Unit_01

Unit_01 Results:

```
> npm-init-package@1.0.0 test
> jest --coverage

 PASS  ./successLogin.test.js
  POST /users/login
    √ should respond with success true and a status 200 code when user is found and password matches (91 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   77.77 |       50 |     100 |   76.92 |
 login.js |   77.77 |       50 |     100 |   76.92 | 32-34,39,45-48
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.775 s, estimated 1 s
Ran all test suites.
```

All files

77.77% Statements 21/27 50% Branches 2/4 100% Functions 2/2 76.92% Lines 20/26

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
login.js	77.77%	21/27	50%	2/4	100%	2/2	76.92%	20/26

Test Case: Unit_02

Test Case Description	Test Data
When a user tries to login in with an account that does not exist within the database it should respond to the user a status code of 400	Email: hello@testing.com Password: helloWorld
Expected Results	Actual Results
It should give back a status code of 400 and won't let the user login	As Expected.

Unit_02 Code:

```
const request = require("supertest");

const app = require("./login.js");

const fs = require("fs");


describe("POST /users/login", () => {

  it("should respond with 400 status when user is not found", async () => {

    // Mocking the file content for users.json

    const mockFileContent = JSON.stringify([]);


    // Mocking the readFileSync function

    jest.spyOn(fs, "readFileSync").mockReturnValueOnce(mockFileContent);


    // Creating the request body

    const requestBody = { email: "hello@testing.com", password: "helloWorld" };


    // Making the request using SuperTest

    const response = await request(app)

      .post("/users/login")

      .send(requestBody);
```

```
    // Asserting the response
    expect(response.status).toBe(400);
    expect(response.text).toBe("Cannot find user");
  });
});
```

Test Case: Unit_02

Unit_02 Results:

```
 PASS  ./userNotFound.test.js
  POST /users/login
    √ should respond with 400 status when user is not found (38 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   66.66 |       25 |      50 |   69.23 |
 login.js |   66.66 |       25 |      50 |   69.23 | 32-34,41-48
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.88 s
Ran all test suites.
```

All files

66.66% Statements 18/27 **25%** Branches 1/4 **50%** Functions 1/2 **69.23%** Lines 18/26

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
login.js	66.66%	18/27	25%	1/4	50%	1/2	69.23%	18/26

Test Case: Unit_03

Test Case Description	Test Data
When a user tries to login into their account with the wrong password. It should throw an status code of 500 and a message to the user notifying them that the password is incorrect	Email: wwe@wwe.com Password: eww
Expected Results	Actual Results
It should give back a status code of 500 and won't let the user login	As Expected.

Unit_03 Code:

```
describe("POST /users/login", () => {

  it("should respond with 500 status code when password is incorrect", async () => {

    // Mocking the file content for users.json

    const mockFileContent = JSON.stringify([

      { email: "wwe@wwe.com", password:
"$2b$10$dsm0qv/PHQg21PZYJ7uM8.Nni8V4E3ZPtxaQhBPpPlMKRGvmGthg." }

    ]);


    // Mocking the readFileSync function

    jest.spyOn(fs, "readFileSync").mockReturnValueOnce(mockFileContent);


    // Creating the request body with incorrect password

    const requestBody = { email: "wwe@wwe.com", password: "eww" };


    // Making the request using SuperTest

    const response = await request(app)

      .post("/users/login")

      .send(requestBody);


    // Asserting the response
```

```
    expect(response.status).toBe(500);

    expect(response.body).toEqual({ success: false, message: "Password incorrect"     });

  });

});
```

Test Case: Unit_03

Unit_03 Results:

```
 PASS  ./incorrectPassword.test.js
  POST /users/login
    √ should respond with 500 status code when password is incorrect (96 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   77.77 |       50 |     100 |   76.92 |
 login.js |   77.77 |       50 |     100 |   76.92 | 32-34,39,43,48
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.966 s
Ran all test suites.
```

All files

77.77% Statements 21/27 **50%** Branches 2/4 **100%** Functions 2/2 **76.92%** Lines 20/26

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
login.js		77.77%	21/27	50%	2/4	100%	2/2	76.92%	20/26

Test Case: Unit_04

Test Case Description	Test Data
When a user tries to login into their account an unexpected error occurs when trying to establish a connection to the server end of the code	Email: wwe@wwe.com Password: wwe
Expected Results	Actual Results
It should give back a status code of 500 and won't let the user login.	As Expected.

Unit_04 Code:

```
describe("POST /users/login", () => {
  it("should respond with a 500 status code when an error occurs during login", async () => {
    // Mocking the readFileSync function to throw an error
    const readFileSyncMock = jest.spyOn(fs, "readFileSync");
    readFileSyncMock.mockImplementationOnce(() => {
      throw new Error("File read error");
    });

    // Creating the request body
    const requestBody = { email: "wwe@wwe.com", password: "wwe" };

    try {
      // Making the request using SuperTest
      const response = await request(app)
        .post("/users/login")
        .send(requestBody);

      // Asserting the response
      expect(response.status).toBe(500);
```

```
    } finally {


      readFileSyncMock.mockRestore();

    }

  });

});
```

Unit_04 Results:

```
PASS ./unexpectedError.test.js
  POST /users/login
    √ should respond with a 500 status code when an error occurs during login (68 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   62.96 |        0 |      50 |   65.38 |
 login.js |   62.96 |        0 |      50 |   65.38 | 30,36-48
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.992 s
Ran all test suites.
```

6.0 Conclusions

In conclusion, the testing process for the WhatsOn application has provided valuable insights into its functionality, reliability, and performance.

Through both manual and automated testing techniques, we were able to identify and address various issues and ensure the overall quality of the application. Additionally, the testing process helped improve our understanding of the application's behaviour and provided opportunities for optimization and refinement.

Through this process, we learned the importance of unit testing in verifying the correctness of our code. We also learned how to use various testing tools like Mocha, Chai, Sinon, and JSDOM.

We found out that testing code that interacts with third-party APIs, like the Google Maps API, can be challenging. These APIs often need to be stubbed out in tests, which can make it difficult to test all aspects of the code.

Despite these challenges, we were able to achieve 100% statement coverage for almost all of the tested code. However, we understand that high code coverage doesn’t guarantee that our code is bug-free.

Overall, this was a valuable exercise in understanding the role of testing in software development and how to effectively use testing tools to improve the quality of our code.