2015

Test Plan & Results Document

YYST

JUST DO IT!

TAE HYUN JIN, GEORGE OSCAR MILLINGTON, JOON XIAN NG, KEVIN HOH

TEAM 15

Contents

Introduction

This document's purpose is to outline the methods of testing that were used in order to develop the application to an acceptable standard, to make sure to the best of our abilities that the application is bug free.

Testing the application ranged from functions built into the code that calculate details such as average speed and distance, to making sure that APIs are correctly integrated and that local storage systems are in working order.

At the end tests were run to ensure that the UI was intuitive and easy to familiarise with for people who were not tech savvy and met the needs of the users who were in the user stories.

Obviously, in an ideal world there are no failures in testing, however it is through the failures that the application can begin to move in the right direction.

Types & Approaches of Testing

Approach / Time Allocation

Each member of the team is allocated about an hour a week to testing various parts of the project using the basic methods outlined below. Furthermore, the higher level black box testing will occur nearer to the end of the project to ensure smooth running of the web application.

Testing Styles:

Unit Testing

Testing to see whether individual bits of code work on their own in isolation, e.g. DOM element or a javascript function.

Integration Testing

Integration testing is used in the opposite way to unit testing where all parts that are affected are tested to ensure that there are no errors occurring due to the changes made. For this testing we will be working our way up from individual snippets of code to larger and larger systems until the whole application runs smoothly including the change.

Acceptance Testing

Acceptance testing comes as a part of the agile methodology and will include meetings and feedback from the client to ensure that the task is staying on track and remains on the path that the client wishes for.

Stress Testing

Stress testing is used to ensure that inputs that are out of the ordinary or overloading of inputs will translate into prompted errors or dealt with in an according manner such that the user will be given feedback on how to avoid the problem and so that every instance is accounted for.

Test Plan

The sections below are the results of the tests conducted on specific parts of the application based on the part that it is affecting.

Each of the tests shown are fundamental to the successful operation of the functional requirements and running of the application.

The tests are performed as the functionality is added meaning that there is no escalation of issues and problems are dealt with in a systematic and controlled approach. This makes it easier to isolate problems when they occur and pinpoint the source of the errors.

The Separate Tests will be organised accordingly:

"Section"

- "Functional Requirement"
    - "Test Case" (Test Data)**(Test Type)**

Calculation Functions

These will all share a test data set consisting of paired Path Objects and Options Objects. The test data will be chosen to cover a range of run speeds and lengths with differing options in order to ensure all feasible inputs that a user may supply throughout the use of the application.

Each test consists of entering the function argument data to the corresponding function and logging the returned statement to the console and analysing it.

- distanceTotal() **(Parameter testing)**
- timeTotal() **(Parameter testing)**
- calorieBurnt() **(integration testing as it time is a paramter)**

View Stored Runs

- Stored runs are displayed in a list
    - Test that a large number of runs display correctly, by inputting premade path object to local data. (test data ranges from 0 to 10 runs). (**Stress testing)**
- Rename saved Run
    - Change the name and check it applies in the list. This will be implemented via actual use of the applications DOM (test data will include both numbers, letters and symbols to cover all cases.)**(Acceptance Testing)**
- Delete Saved Run

- After deleting a run, a check to see that it has disappeared from the archives is necessary (test data will include deleting one run in the middle of the list and deleting the last run in the list) **(Unit testing)**

Record New Run

- Displaying GPS position.
  - Test that the GPS position displays as accurately as possible for various locations(altitude, indoors etc). App will be booted in an outside location and restarted at various locations to ensure wide coverage. (**Acceptance testing)**
- Save Run
  - Test to see whether runs are stored locally by creating random runs and storing them **(integration testing of the save run function with different local storage states.)**

Test Cases and Results

Calculation Functions

The test data set comprises of some contrived cases such as an empty run but also just a variety of runs physically recorded on the test run. The options object is merely changed each time to cover each option which can single out the problematic toggle. This test Data set applies to all blue coloured tests.

Total Distance function test

Test Case:	Test for **distanceTotal Function Reliability**			
Description:	Test to check that the function returns the distance of the path object in a string format.			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Parse local storage data	A path object with latlng property and time property	Pass	
2	Call the distance calculation function.	Distance function executes, returning either the numeric distance in units specified by the optionsObject, or an error in which case the input was invalid.	Pass	Some errors occurred for the empty object path
3	console.log() the output and compare to expected result.	The numeric distance in units specified is then displayed through the console and then compared to the expected result.	Pass	No movement returned a 0

Time function test

Test Case:	Test for **timeTotal Function Reliability**			
Description:	Test to check that the function keeps track of the ongoing run.			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Parse the local storage objects	A path object with latlng property and time property	Pass	
2	Call timeTotal function.	Time function executes, returning the time as a string	Pass	
3	console.log() the output and compare to expected result.	The time is displayed as a string in format hh:mm:ss.	Pass	

Calories function test

Test Case:	Test for **calorieBurnt Function Reliability**			
Description:	Test to check that the function is correctly calculating the number of calories burned			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Parse the local storage objects	A path object with time property and weight property.	Pass	
2	Call the function	Calories function executes, returning in calories specified by the optionsObject.	Pass	
3	console.log() the output and compare to expected result.	The total calories burned is displayed in a string format.	Pass	

Tests that require Path and Option objects as their test data:

Save run test

Test Case:	**Test for saving run function reliability**			
Description:	Tests the ability to save runs			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Record the current run.	Run is recorded	Pass	
2	Click the save button	Run is saved	Pass	
3	View the run in local storage.	Run saved as JSON data	Pass	

Stored runs display correctly test

Test Case:	**Stored Runs Display Correctly**			
Description:	Tests the ability to view stored runs and display them correctly			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Insert the current testing run to the end of the local storage array and store it.	The path is stored as an array inside the path property of storageObject. The format is JSON.	Pass	
2	Navigate to the archive menu	Navigates to archive menu	Pass	
3	See whether the list displays correctly	Name and date displayed as strings	Pass	

Delete saved run test

Test Case:	**Test for deleting run function reliability**			
Description:	Tests whether local data is deleted properly			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Navigate to the record page.	Navigates to record page	Pass	
2	Record the run then stop	Run is recorded	Pass	
3	Click the ‘clear’ button	Run is deleted from memory	Pass	

Display GPS position.

Test Case:	**Testing that geolocation.getCurrentPosition updates**			
Description:	Tests whether GPS position is correct			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Navigate to the new run page	Navigates to new run page	Pass	
2	Check the red marker on the map and compare with the surroundings if the position is accurate	Marker indicating location of user is similar to that of real life location	Pass	
3	Move to a direction and check if the marker is being updated correctly.	The position of the red marker is being updated and moved to a new current position.	Pass	There is a delay sometimes on getting a geoposition and the app does not work well indoors but this is an accepted constraint of GPS. Possibly the performance could be improved by integrating accelerometer values also.
Test Data Set A				
1	Outside stationary		Pass	
2	Outside mobile tracking		Pass	
4	Inside tracking		Fail	10-30 second - intervals between updates

5	Inside mobile tracking	Fail	10-30 second - intervals between updates

DOM elements resize according to browser size

Test Case:	**Checking that the UI resizes correctly.**			
Description:	Test to check that DOM elements are displayed correctly independent of screen size			
Step	**Description**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Open the Application from a various devices	All elements on the page will be stretched out to accommodate the width of the larger screen size.	pass	
2	Resize the window to large.	All elements are located in the same position but resized to large size.	pass	Elements stretched to fit larger sizes
3	Resize the window to small.	All elements are located in the same position but resized to small size.	pass	Some fonts were not legible at extremely small screen sizes

Conclusion

Overall the testing was successful as where errors occurred, we edited the code to accommodate for required changes.

For some cases such as the occasional inaccuracies in position given from GeoLocation which spike the run path, it was not feasible in the time limit to implement a fix of suitable quality. Therefore we let the inaccuracies simply be a constraint of our current application. Also for weight and stride length input, the user was unable to input decimal numbers due to the way that the input box defines numeric characters. This is an acceptable error because generally the decimals will not have any noticeable impact on calculations.

In reflection of these cases it is apparent that further testing and tweaking would benefit the app if the clients were willing to extend time and resources but for the moment the application is of acceptable quality.