Test Plan & Results Document

Tom Gu, Alex luc, Nir Nikolaevsky

TEam 68

2015

Contents

[Introduction 2](#_Toc431642758)

[Types & Approaches of Testing 3](#_Toc431642759)

[Test Plan 4](#_Toc431642760)

[Test Cases and Results 7](#_Toc431642761)

[Conclusion 7](#_Toc431642762)

# Introduction

With the creation of our running application, it is crucial to conduct methods of Software Quality Assurance (QA) to ensure that all components of the application are functional and meet set criteria. QA will be conducted via methods of testing. These tests will cover all functions within the application code such as button functionality, route tracking, run saving, history viewing, as well as other components of the application which will be discussed in detail in the test plan. The most crucial components to test are the route recoding function, the route saving functionality and the route viewing functionality, as they are the basis of our app’s operation. The criteria for testing involves testing every functionality five times and requiring a 80% success rate.

# Types & Approaches of Testing

The types of tests to be conducted fall under two approaches, Requirements-based and Code-based. The Requirements-based approach breaks the components of the app into specific requirements (such as the record page button should open the recording page when pressed). The requirements are then assessed based on the knowledge of what the outcome of that individual requirement is and whether it fulfils this outcome. This approach however is a ‘black box’ approach and does not consider how the requirement works and thus does not examine the actual code. Code-based approach examines the actual code and creates one test case for each path of the code (e.g for each branch of a if…else if statement) in order to make sure the code is bug-free. A requirements-based approach was selected for testing the application to ensure all user requirements are met.

The types of testing being utilised are:

* Unit testing - which stress tests each unit of code such as individual functions, or a method of a class in the case of object-oriented code. This test is necessary to determine whether each unit of code is able to run properly and without bugs.
* Integration testing - which tests how individual units perform in cooperation with each other in other to test whether any bugs arise. This is done after unit testing has been completed. This will be necessary in order to determine how each unit of code operated in conjunction with other units and whether bugs arise.
* System Testing – tests the complete application in order to check that it satisfies its requirements. This is done based on user stories and activity diagrams as is intended to check for correct behavior of the app as well as to see if it does anything unwanted such as crash. This test is necessary to determine whether the entire app operates as planned and as required by the user.

The table below gives an overview of the functionalities that require testing, when they will be tested and the member responsible for testing.

|  |  |  |
| --- | --- | --- |
| **Test** | **Time** | **Team Member** |
| * Testing Launch Page functionalities | Wednesday 21st October 2015 | Tom |
| * Testing Instruction’s Page Functionality | Wednesday 21st October 2015 | Tom |
| * Testing all Record Page functionalities | Wednesday 21st October 2015 | Alex |
| * Testing all View Page functionalities | Wednesday 21st October 2015 | Nir |

**Test Plan**

Functional Requirements

* Launch Page - Initial page for users and contains links to all features of the apps
* Instructions Page – Displays instructions on how the app operates
* Record Page
  + Start, Stop, Clear Route recording
  + When recording, user’s locations are displayed on the map marked by a current position marker and start position marker.
  + User’s route between start and current location is marked by a polyline.
  + Run, Walk, Cycle modes are displayed and can be selected
  + Timer starts when record button is pressed and stops when paused.
  + The route recorded can be saved and user is prompted to enter a description before each route is saved.
* View Page
  + The user should be able to select a route to display
  + The route should be shown on the screen with the start and end points labelled.
  + Only one selected route should be shown at any time.
  + The average speed, total distance, activity mode and total time taken should be shown for each route
  + The user should be able to delete a route
  + If there are no recorded routes, the record buttons should not work.

Test Plan

Launch Page:

* Unit testing will be conducted on the individual buttons on the Launch Page to see if individual lines of code run as expected. An example of this is the button effects that are activated when the button is pressed. This test is initiated by having the user press a certain button on the page. This test will run 5 times to ensure consistent display of the button effects. This test is crucial for ensuring that the buttons is fully functional and can be “pressed”. This will not test an explicit functional requirement but contributes to the requirement of linking users users to all features of the app by ensuring that the buttons are functional.
* Unit testing will also be conducted on the html layout to see if ever aspect of the page appears as desired and that all desired text is written out. This will be done by opening the page and checking if all html elements are present. This will also be done 5 times for consistency. This test is in response of the user story which requires that the developer to “utilise MDL in order to provide a better user experience”.
* System testing will also be conducted on the buttons in the form of integration testing to test whether the buttons actually link to the desired page as per the activity diagrams. This test will be conducted by pressing the individual buttons in order to see if it opens the desired page. This will be tested 5 times for consistency and ensures that the functional requirement of linking users to other feature of the app from the launch page.

Instructions Page:

* Unit testing will also be conducted on the HTML layout to see if ever aspect of the page appears as desired and that all desired text for the instructions are written out. This will be done by opening the page and checking if all html elements are present and will be done 5 times for consistency. Although not an explicit functional requirement, this test is to examine whether the desires aesthetics of the page are displayed.
* Integration testing will be conducted on the HTML by opening the instructions page from the launch page button in order to ensure that no bugs arise when this unit of code is run in conjunction with code from the launch page. This will run 5 times to ensure consistency.
* Both these tests are to ensure the functional requirement of having an instructions page that instructs users on how the app operates is met.

Record Page:

Integration tests were more applicable form of testing in the record page as all individual units are linked with and work in cooperation with other units of code. These tests for will be conducted on the Start, Stop, Clear functionalities; the mode selection functionality; the route tracking functionality; the description prompt; and the save function:

* The start, stop, clear functionalities will be tested by having a user press the start, pause, clear buttons to see if the desired outcomes run (e.g position located when start button pressed and timer starts, pause button pauses route recording and timer stops, clear button deletes previously recorded route and time resets to zero). This will be done 5 times for each button to ensure the function runs consistently. The application of this test is necessary to satisfy the functional requirement of being able to start, stop, and clear recording.
* The mode selection functionality will be tested by having the user select a particular mode (run, walk, or cycle) and seeing if the selected mode is displayed on the activity box. This will be tested five times in order to determine whether it runs consistently.
* The route tracking functionality will be tested by having the user pressing the play button to initiate route tracking. For the test to pass, the map must display the user’s current location with start and current position markers within a range of 20m. When the user moves, the route of movement must be tracked by a polyline within a range of 20m. This will be conducted 5 times to ensure accuracy is consistently met within the set parameters.
* The description prompt is tested by having by having the user pause the run and pressing the save button. To pass the test, once the save button is pressed; the user must be prompted to enter a description of the run in a text field within a pop up. This is conducted 5 times to ensure that the user is consistently prompted. This test is required to fulfil the functional requirement of being able to enter a description for each recorded route.
* The save functionality is tested by having the user pause the run and pressing the save button and entering a description. This test will pass if when you see the object containing properties relating to: the date, the activity mode, the time of the run, the array of all coordinates recorded for the route, the distance of the route, the average speed of the route; are displayed within local storage which is viewed using developer tools on Google chrome. Five routes will be saved during testing to ensure the function consistently saves all object properties. This test is required to satisfy the functional requirement of being able to save recorded routes.

The View Page

* Unit testing can be conducted on the delete function of the view page. This will be tested by first saving recorded route. The route is then deleted by pressing the delete button. In order to pass this test, the route must be deleted from the table containing saved runs and all data related to that run must be deleted from local storage. This is tested 5 times to ensure that the function runs consistently without bugs. This test also ensures the functional requirement of being able to delete saved routes.
* Unit testing can also be conducted to ensure that the delete button is not enabled if there are no existing routes. This will be tested by deleting all runs from the view page and then pressing the delete button again once all runs are deleted. For this test to pass, the delete button must not be enabled, that is it can be pressed and does not run. This will be tested 5 times to ensure consistency and should satisfy the functional requirement of disabling the delete button if there are no recorded routes.
* Integration testing will be run to see if individual routes can be selected to display a mapped route from start to finish. This will be done by recoding a route and saving it. The saved run is then accessed from the view page. In order to pass the test, a map of the recorded route must be displayed with start and finished points marked and the route highlighted by a polyline. This will be run 5 times in order to ensure it operates consistently. This test will satisfy the functional requirements of being able to display recorded routes and showing start and end points of the route.
* Integration tests will also be run to ensure that distance, average speed, time, date, and activity mode are recorded as part of the recorded route. This will be tested by recording and saving a route and viewing the saved route on the view page. The test is passed if distance, average speed, time, date, and activity mode are shown for the recorded route on the table of saved routes. This test will be conducted 5 times to ensure consistency and meets the functional requirement of showing the distance, average speed, time, date, and activity mode for each route.

# Test Cases and Results

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Activity Selection functionality** | | | |
| Description: | This test case tests to check that the activity selection functionality prompts the user before recording a run and selects an activity for a recorded run. | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Open the record page by clicking the Recording Page button on the launch page | The application open the recording page | Pass |  |
| 2 | Press play button | The application prompts the user to select an activity before recording run. | Pass |  |
| 3 | Select a activity mode | The activity box shows that a particular activity mode is selected. | Pass |  |
|  |  |  |  |  |
| Test Data | |  |  |  |
| 1 | Running selected |  | Pass |  |
| 2 | Cycling selected |  | Pass |  |
| 3 | Walking selected |  | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Test route cancelling functionality** | | | |
| Description: | This test case tests to check that the route cancelling functionality works only when pressed after a run has been recorded and stopped | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Open Recording Page | The application open the recording page | Pass |  |
| 2 | Press the cancel route button | Nothing should happen, as the delete button should not run if there is no route to the delete. | Pass |  |
| 3 | Click the blue tick, or hit enter | The application should show an error stating that "The input was invalid" | Pass |  |
| 4 | Select a activity mode | The activity box shows that a particular activity mode is selected. | Pass |  |
| 5 | Press play | The application should show the user’s current location with a start marker, the timer also starts running. The status box is undated to recording. | Pass |  |
| 6 | Press the cancel route button | Nothing should happen as the delete button should not run if the route recording is not stopped | Pass |  |
| 7 | Stop Recording | The timer stops, the user’s location is no longer tracked, and his recorded route is displayed on the map. The status is updated to paused | Pass |  |
| 8 | Press Cancel Run button | The recorded… |  |  |
| 9 | Start New Run | The map should initialise with new sets of markers |  |  |
|  |  |  |  |  |
| Test Data | |  |  |  |
| 1 | Run selected |  | Pass |  |
| 2 | Walk selected |  | Pass |  |
| 3 | Cycle selected |  | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Route tracking functionality** | | | |
| Description: | This test case tests to check that the route tracking functionality works completely when activated | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Start the application opening the URL in Google Chrome | The application initializes with the launch page | Pass |  |
| 2 | Open the record page by clicking the Recording Page button on the launch page | The application open the recording page | Pass |  |
| 3 | Select a activity mode | The activity box shows that a particular activity mode is selected. | Pass |  |
| 4 | Press play | The application should show the user’s current location with a start marker within a 20m range, the timer also starts running. The status box is undated to recording. | Pass | The user’s location was not very accurate but still fell within the 20m parameter that was set for testing. |
| 5 | Walk Around | The user’s location is displayed by a current location marker and the user’s route is highlighted by a polyline | Pass |  |
| 6 | Stop Recording | The timer stops, the user’s location is no longer tracked, and his recorded route is displayed on the map. The status is updated to paused | Pass |  |
| 7 | Resume Recording | The timer restarts, the user’s location is tracked and updated on the map with the current position marker. The status box is updated to recording. | Pass |  |
|  |  |  |  |  |
| Test Data Set | |  |  |  |
| 1 | Running selected |  | Pass |  |
| 2 | Cycling selected |  | Pass |  |
| 3 | Walking selected |  | Pass |  |
| 4 | Move 100m |  | Pass |  |
| 5 | Indoors |  | Pass |  |
| 6 | Outdoors |  | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Description and Extra Comments Prompt** | | | |
| Description: | This test case tests to check that Enter Description and Extra Comments prompts and text fields appear when the user saves a run | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Record a route | The application should have recorded the user’s route marked by a polyline with a start location and current location. Time of the recorded route and activity mode is also displayed. | Pass |  |
| 2 | Press the save button | User should be prompted with a pop up to enter a description. | Pass |  |
| 3 | Enter the test data in B1 | Desired text should be inputted into text field | Pass |  |
| 4 | Press ok | User should be prompted with a pop up to enter any extra comments. | Pass |  |
| 5 | Enter the test data in B1 | Desired text should be inputted into text field | Pass |  |
| 6 | Press ok | User should be prompted to that run is saved | Pass |  |
| 7 | Open View page | Inputted Description and Extra Comments should be correctly displayed under it’s particular run. | Pass |  |
|  |  |  |  |  |
| Test Data Set B1 | |  |  |  |
| 1 | "" - blank input |  | Pass |  |
| 2 | " " - single space |  | Pass |  |
| 3 | 999 |  | Pass |  |
| 4 | 10000 |  | Pass |  |
| 5 | 1 |  | Pass |  |
| 6 | -1 |  | Pass |  |
| 7 | abcd |  | Pass |  |
| 8 | @$ |  | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Testing reliability of buttons in the navigation of the application’s pages and the consistency of the HTML of pages when loaded** | | | |
| Description: | This test case checks that the UI pages are navigating as intended | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Start the application opening the URL in Google Chrome | The application initializes with the launch page with the correct MDL layout. | Pass |  |
| 2 | Press Instructions | The application should open the instruction’s page with the correct MDL layout. | Pass |  |
| 3 | Press back | The user should be taken back to the launch page. | Pass |  |
| 4 | Press Record Button | The application should open the recording page with the correct MDL layout. | Pass |  |
| 5 | Press back | The user should be taken back to the launch page. | Pass |  |
| 6 | Press History | The application should open the view page with the correct MDL layout. | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Testing the Saving functionality** | | | |
| Description: | This test case tests to check that the route cancelling functionality works only when pressed after a run has been recorded and stopped | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Save a recorded run using the test data below | Once the user has pressed the save button and entered a description and additional comments, he/she should be prompted that the route has been saved | Pass |  |
| 2 | View saved route on recording page | A table should show the saved routes with data about the distance, average speed, activity mode, time and date of the run also shown | Pass | Average speed displays as undefined however. |
| 3 | Press on recorded route | A map of the recorded route should pop up highlighted by start and end markers and the route marked by a polyline. | Fail | This functionalitywas not achieved |
|  |  |  |  |  |
| Test Data | |  |  |  |
| 1 | Run selected |  | Pass |  |
| 2 | Walk selected |  | Pass |  |
| 3 | Cycle selected |  | Pass |  |
| 4 | Walk 100m |  | Pass |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | **Testing the delete functionality for saved routes** | | | |
| Description: | This test case tests to check that the route cancelling functionality works only when pressed after a run has been recorded and stopped | | | |
| Step | Description | Expected Result | Pass/Fail | Remarks |
| 1 | Save a recorded run using the test data below | Once the user has pressed the save button and entered a description and additional comments, he/she should be prompted that the route has been saved | Pass |  |
| 2 | View saved route on recording page | A table should show the saved routes with data about the distance, average speed, activity mode, time and date of the run also shown | Pass | Average speed displays as undefined however. |
| 3 | Press the delete button | Route is deleted from the table and when local storage is accessed, the data related to that route is deleted. | Pass |  |
| 4 | Clear all routes and then press delete button | Delete button should be diabled | Fail |  |
|  |  |  |  |  |
| Test Data | |  |  |  |
| 1 | Run selected |  | Pass |  |
| 2 | Walk selected |  | Pass |  |
| 3 | Cycle selected |  | Pass |  |
| 4 | Walk 100m |  | Pass |  |

# Conclusion

In conclusion, a majority of the set tests were passed according to the set criteria and thus the application met most of the functional requirements and specifications. Those that failed were mainly due to time constraints, which resulted in the test criteria not being met. This was the case for the occasional inaccuracies in position given from WatchLocation, which resulted in inconsistent position coordinates, 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 and set a wide testing parameter of within 20m. Further testing may be undertaken utilising a Code-based approach in order to test the running of individual JavaScript functions and methods for greater identification of potential bugs.