**Software Test Plan**



Attendance System

Table of Contents

Table of Contents 3

1 Introduction 4

2 Goals and exit criteria 5

1. Quality goals that needs to be met for the test phase to exit 5

2. Schedule goals of the project 5

3. Performance and efficiency goals of the product 5

3 Items to be tested/ inspected 6

1. Executables such as modules and components 6

2. Non-executable such as requirements specification or design specification 7

3. Items Not to be Tested 7

4 Test process/ methodologies 8

1. Unit test/ functional test/ acceptance test/ regression test/ and so on, methodologies. 8

2. Inspection/ reviews methodologies 9

3. Black-box testing (e.g., input domain test, boundary value testing 9

4. White-box testing (e.g., control path testing, data flow testing ) 9

5. Test metrics(e.g., code coverage, branch coverage, number of problem by severity) 9

6. Test-bug report-fix-retest process 9

5 Resources 11

1. People 11

2. Tools 11

3. Systems 11

6 Schedule 12

1. Test-case development 12

2. Test execution 12

3. Problem reporting and fixing 12

7 Risks 13

1. Missing goals 13

2. Back-up resources needed 13

8 Major test scenarios and test cases 14

1. Boundary value and input domain test cases 14

8.1.1 Checkpoint Web App Performance Test Cases 14

8.1.2 Host test cases 15

8.1.3 User test cases 25

8.1.4 Native Test Cases 30

# Introduction

The CheckPoint project is a web based attendance system capable of registering and verifying attendance to a wide range of event types. The system will allow anyone hosting an appointment or event to efficiently track attendance levels. In addition, it will allow potential attendees to easily discover and sign up for local and national events through a web based user panel.

The nature of the application is therefore not critical to human life or health and will not need to be subject to the most stringent testing procedures. However, the application does have the potential to cause considerable disruption and frustration to users if it fails to function according to the specification requirements.

Acquiring evidence that a product is of sufficient quality with a reasonable level of confidence is the primary objective of the testing process.

This test plan will describe the necessary test procedures that must be undertaken and passed in order for the project to meet this demand. A selection of tests will be chosen to target specific aspects of the projects functionality. Each test will be documented in an auditable fashion such that it may be used to assess the quality of the product. Once all tests have achieved a satisfactory pass mark the results will provide a level of assurance to the customer and developer that the product is fit for purpose.

In order for the tests to be considered valid, they must be performed with certain constraints. The tests must be performed under conditions that reflect those similar to an end user. In this instance that will mean a clean operating system devoid of developer tools and libraries. To achieve this goal the testing will be performed from inside a virtual machine environment with a clean install of the specified operating system.

The specific functional and non-functional requirements that should be targeted by the tests are described in the Software Development Plan and the Software Requirements and Design Document.

# Goals and exit criteria

## Quality goals that needs to be met for the test phase to exit

The following goals are essential for the test phase to exit:

* All test cases have been performed.
* No critical or high impact faults should remain. These are defined in chapter 6 Test-bug report-fix-retest process.
* 95% of medium impact bugs must be resolved.
* All bugs and defects have been properly recorded and signed off.

All test goals should preferably be fulfilled before the test phase may be closed and implementation can proceed, however there will be some flexibility and the criteria for exit may be reviewed in the event that certain tests do not pass.

## Schedule goals of the project

Testing should be executed at least once in each iteration of the project, as shown in figure 2 chapter 7 in the Software Development Plan.

The software should also be tested according to the regression testing which is described in chapter 4.1.

## Performance and efficiency goals of the product

As outlined in chapter 3.2 of the Software Requirements and Design document.

* The reading terminal should be responsive enough to process and register an attendance within 4 seconds.
* The system should have an uptime of 22 hours per day.
* The GUI should not require the user to have any prior technical experience in order to navigate the system.

# Items to be tested/ inspected

## Executables such as modules and components

The items to be tested are the Checkpoint web application software, the Checkpoint native application software and the associated Reading-terminal hardware. The number of individual components and modules that would ordinarily be subject to both unit testing and integration testing under far exceed the capabilities and resources available to the development team on this project. With this considered function testing will be the primary focus of the testing phase.

The specific functions to be tested are as shown in *Table 1*.

*Table 1: Checkpoint web app functionality*

|  |  |
| --- | --- |
| Checkpoint Web App  Feature / Functionality | SRD Design Requirement Reference  Use Case Documents |
| Client Registration | Chapter 8.2.1 Register |
| Log In | Chapter 8.2.2 Log In |
| Become Attendee | Chapter 8.2.3 Become Attendee |
| Manage Profile | Chapter 8.2.4 Manage Profile |
| Create Appointment | Chapter 8.2.5 Create Appointment |
| Manage Appointment | Chapter 8.2.6 Manage Appointment |
| Create Course | Chapter 8.2.7 Create Course |
| Manage Course | Chapter 8.2.8 Manage Course |
| Manage Attendance | Chapter 8.2.9 Manage Attendance |
| Create Report | Chapter 8.2.10 Create Report |

Table 2: Checkpoint web app performance

|  |  |
| --- | --- |
| Checkpoint Web App  Performance | SRD Design Requirement Reference  Technical Requirements |
| Browser Support Chrome, Firefox, Microsoft Edge, Opera | Chapter 3.1.1 Functional Requirements |
| Support Access by Multiple Simultaneous Users | Chapter 3.1.1 Functional Requirements |

*Table 3: Checkpoint Native App & hardware reading terminal functionality*

|  |  |
| --- | --- |
| Checkpoint Native App  & Hardware Reading-terminal  Feature / Functionality | SRD Design Requirement Reference  Use Case Documents |
| Attend Appointment | Chapter 8.2.9 Attend Appointment |

*Table 4: Checkpoint native app and hardware reading-terminal performance*

|  |  |
| --- | --- |
| Checkpoint Native App  & Hardware Reading-terminal  Performance | SRD Design Requirement Reference |
| Attendance Registration Time (within 3 secs) | Chapter 3.2 Non-Functional Requirements |

## Non-executable such as requirements specification or design specification

## Items Not to be Tested

Excluded aspects of the application which will not be tested due to time constraints, lack of resources or being beyond the scope of this iteration include:

* Stress testing
* Recovery testing
* Security testing
* Database integrity testing
* Access testing
* Configuration testing (various OS and hardware configurations)

# Test process/ methodologies

## Unit test/ functional test/ acceptance test/ regression test/ and so on, methodologies.

Unit testing consists of testing every individual class and its methods in isolation with the use of mock-up classes with which to provide data for the isolated class to consume. This involves the creation of a duplicate mock class for every single class in the application and both black-box and white-box testing to be performed. Due to the limited time constraints of the project it will not be possible to complete unit testing of the project. It may be possible to unit test a small number of critical classes.

Integration testing is carried out following the unit tests being satisfied and involves bringing the individual classes together and testing their behavior once again when they are interacting with each other and not the mocks. Once again, the schedule does not permit us to complete a full integration test but it may be possible to test a small chain of critical classes.

Function testing involves testing the application according to the software design requirements and specifications as outlined in the development documentation. These tests directly address the ability of the application to fulfill the use cases. As such the function testing will be prioritized and will form the most significant part of the testing procedure. Documentation of each function under test will be recorded and signed by the tester. A list of each function and a reference to the requirements can be found in chapter *3 Items to be tested/ inspected*.

User Interface testing concerns verifying that the actual graphical user interface and the interaction with the reading terminal function conform to the specifications laid out in the software design and requirements documentation.

Regression testing will be carried out manually after any bug fixes have been applied to ensure the modifications themselves have not lead to the creation of more bugs. Testing of related functions will be postponed until the regression checks are completed to minimize side-effects. Regression tests must be completed before any big changes to the project are made and will follow the functional test cases in this document.

System testing follows integration testing and is concerned with testing the whole application in its entirety. System testing looks to discover defects that may arise between the inter-assembly items and hardware components that might not become apparent with integration testing. System testing will only be viable once all components of the project have been written.

## Inspection/ reviews methodologies

## Black-box testing (e.g., input domain test, boundary value testing

Black-box testing does not require the tester to have any knowledge about the underlying system. The main focus is to discover defects which impair the functionality of the software as specified by the requirements and use case documents. Black-box testing will most closely match the experience of the majority of the users of the Checkpoint software. As such, this approach will be a critical aspect of the testing phase and form the main body of the test cases.

## White-box testing (e.g., control path testing, data flow testing )

White-box testing requires the tester to have detailed knowledge of the programs source code and structure. Rather than focusing on the functional requirements from the perspective of an end user, white-box testing looks at the data flow of the code and seeks to discover aspects which might be vulnerable to failure or fragile. This might include the complexity of conditional logic, the structure of the inter-class communication or the general implementation of the code. Most often performed on the same level as unit testing and integration testing, white-box testing is costly in terms of resources requiring time and testers with a proficiency in programming. Due to the tight schedule of the CheckPoint project white-box testing will not be prioritized.

## Test metrics(e.g., code coverage, branch coverage, number of problem by severity)

The results obtained from the test case documents will be reviewed and assessed according to severity and the quality goal requirements.

## Test-bug report-fix-retest process

Any bugs or failures during testing will be recorded and logged with Microsoft VSTS development tools. The bug will be allocated to a developer who will be tasked with correcting the behavior and regression testing the associated function along with related functions. Once the regression test is complete the bug will be marked as resolved on the bug tracker.

The severity of the bug or failure will be recorded according to the criteria in *Table 5*.

*Table 5: Severity of bug criteria*

|  |  |
| --- | --- |
| Severity | Impact |
| 1.Critical | System crash, potential data loss or corruption.  Necessitates a reboot or restart to recover. |
| 2.High | Functionality is disrupted or unable to be completed. Impacts other areas of the program. |
| 3.Medium | Functionality not working as specified but can be achieved via other means. |
| 4.Low | Functionality works as specified except under certain outlying conditions. |
| 5.Cosmetic | Experience impacted by confusing feedback or inappropriate graphical response. Functionality is otherwise correct |

Bug fixing will be prioritized according to the level of severity.

Each bug must be reported with a description of both the bug and how to reproduce the undesired behavior.

*Table 6: Bug description and steps to reproduce*

|  |  |
| --- | --- |
| Bug | Steps to Reproduce |
| Description of bug | Description of how to reproduce the bug. |

# Resources

This chapter contains information about which resources are used during testing of the Checkpoint system.

## People

The people in *Table 7* will be responsible for executing and documenting the tests described in this document.

*Table 7: Test personnel and contact information*

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Email | Phone number | Role |
| Brueland, Kevin | [Kevin.brueland@gmail.com](mailto:Kevin.brueland@gmail.com) | 4128376 | System developer |
| Liknes, Morten | mliknes@hotmail.com | 47882901 | System developer |
| Lam, Chi Mon Noel | tonje\_@hotmail.co.uk | 99297412 | System developer |

## Tools

Microsoft VSTS Bug tracking tools will be used for documenting, storing and tracking bugs.

## Systems

To achieve a clean environment when testing the software, the testing will take place on a virtual computer through VMware Workstation Player 12.5.

The virtual computer will have installed Windows 10 Education (Version 1703) and SQL-server 2016 express.

# Schedule

The test schedule will cohere with the Gantt chart in the SDP (Software Development Plan) chapter 7.

## Test-case development

## Test execution

## Problem reporting and fixing

Team foundation server

# Risks

This chapter tackles all the

## Missing goals

The risks are shown in *Table 8*.

*Table 8: Risk analysis*

|  |  |  |
| --- | --- | --- |
| Risk | Probability | Consequence |
| Project not completed to the degree that all functionalities have been implemented. | medium | All functions cannot be tested or system tested. |
| Testing not completed due to disease. | low | All functions will not be tested or system tested. |
| Testing not completed due to time limitation | medium | All functions will not be tested or system tested. |

## Back-up resources needed

No back-up resources are available.

# Major test scenarios and test cases

## Boundary value and input domain test cases

### Checkpoint Web App Performance Test Cases

The following test cases are for the Checkpoint Web Application.

#### Browser Compatibility

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Browser Compatibility | OK | FAIL | Description | Severity |
| Functionality in Chrome |  |  |  |  |
| Functionality in Firefox |  |  |  |  |
| Functionality in Opera |  |  |  |  |
| Functionality in Microsoft Edge |  |  |  |  |
| GUI is rendered correctly in Chrome |  |  |  |  |
| GUI is rendered correctly in Firefox |  |  |  |  |
| GUI is rendered correctly in Opera |  |  |  |  |
| GUI is rendered correctly in Edge |  |  |  |  |

#### Multiple Simultaneous Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Multiple Simultaneous Users | OK | FAIL | Description | Severity |
| System handles 3 or more simultaneous attempt to update the same table in the database |  |  |  |  |

### Host test cases

The following test cases are specifically for testing Host functionality.

#### Register Host Client Type

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register Host Client Function | OK | FAIL | Description | Severity |
| Registration succeeds with valid data |  |  |  |  |
| Registration fails with invalid password length (< 6 chars) |  |  |  |  |
| Registration fails with duplicate username |  |  |  |  |
| Registration fails with invalid postcode  (does not exist) |  |  |  |  |
| Registration fails with invalid email  format |  |  |  |  |
| Registration fails with invalid phone number (< 8 digits / non-integer) |  |  |  |  |
| Registration fails with null fields |  |  |  |  |
| Registration response to invalid data |  |  |  |  |
| Registration data stored correctly in database |  |  |  |  |
| Response if registration fails |  |  |  |  |
| Client is redirected to the login page after registration is successful |  |  |  |  |

#### Host Login

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login Function | OK | FAIL | Description | Severity |
| Login succeeds with valid data |  |  |  |  |
| Login fails with invalid data |  |  |  |  |
| Login recognizes correct client type |  |  |  |  |
| Login stores client to session |  |  |  |  |
| Login response to null fields |  |  |  |  |
| Login hides password |  |  |  |  |

#### Manage Host Profile

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manage Profile Function | OK | FAIL | Description | Severity |
| Page shows correct profile data |  |  |  |  |
| Profile updated with valid data is successful |  |  |  |  |
| Update fails with invalid password length (< 6 chars) |  |  |  |  |
| Response if update fails |  |  |  |  |
| Update fails with invalid postcode  (does not exist) |  |  |  |  |
| Update fails with invalid email format |  |  |  |  |
| Update fails with invalid phone number (< 8 digits / non-integer) |  |  |  |  |
| Update fails with null fields |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Create Appointment Function | OK | FAIL | Description | Severity |
| New Appointment creation with valid data is successful |  |  |  |  |
| Creation fails with null fields |  |  |  |  |
| Creation fails with invalid time |  |  |  |  |
| Creation fails with invalid date |  |  |  |  |
| Creation fails with invalid postcode  (does not exist / non-integer) |  |  |  |  |
| Appointment is stored correctly in database |  |  |  |  |
| Response if creation fails |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Update Appointment Function | OK | FAIL | Description | Severity |
| Page shows correct appointment data |  |  |  |  |
| Appointment updated with valid data is successful |  |  |  |  |
| Update fails with null fields |  |  |  |  |
| Update fails with invalid time |  |  |  |  |
| Update fails with invalid date |  |  |  |  |
| Update fails with invalid postcode  (does not exist / non-integer) |  |  |  |  |
| Appointment is updated correctly in database |  |  |  |  |
| Response if update fails |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Delete Appointment Function | OK | FAIL | Description | Severity |
| Appointment is deleted |  |  |  |  |
| Deletion is updated correctly in database |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Create Course Function | OK | FAIL | Description | Severity |
| New course creation with valid data is successful |  |  |  |  |
| Creation fails with null fields |  |  |  |  |
| Course is stored correctly in database |  |  |  |  |
| Response if creation fails |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Add Existing Appointment To Course Function | OK | FAIL | Description | Severity |
| Existing appointment is added successfully |  |  |  |  |
| Adding fails with null fields |  |  |  |  |
| Adding can be abandoned successfully |  |  |  |  |
| Course is stored with existing appointment correctly in database |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Host Home Page Function | OK | FAIL | Description | Severity |
| Appointment table is displayed successfully |  |  |  |  |
| Sorting appointment data functions correctly |  |  |  |  |
| Create new appointment button navigates to correct page |  |  |  |  |
| Manage appointment button navigates to correct page |  |  |  |  |
| View courses button navigates to correct page |  |  |  |  |
| Create Report button navigates to correct page |  |  |  |  |
| Add selected appointment to course button navigates to correct page |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Host Courses Page Function | OK | FAIL | Description | Severity |
| Courses page table is displayed successfully |  |  |  |  |
| Sorting course data functions correctly |  |  |  |  |
| Create new course button navigates to correct page |  |  |  |  |
| Manage courses button navigates to correct page |  |  |  |  |
| View courses button navigates to correct page |  |  |  |  |
| Manage attendance button navigates to correct page |  |  |  |  |
| Create report button navigates to correct page |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Add appointment to course page Function | OK | FAIL | Description | Severity |
| Courses page table is displayed successfully |  |  |  |  |
| Sorting course data functions correctly |  |  |  |  |
| Create new course button navigates to correct page |  |  |  |  |
| Manage courses button navigates to correct page |  |  |  |  |
| View appointments button navigates to correct page |  |  |  |  |
| Manage attendance button navigates to correct page |  |  |  |  |
| Create report button navigates to correct page |  |  |  |  |

#### Manage Attendance

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manage Course Attendance Page Function | OK | FAIL | Description | Severity |
| Course table is displayed successfully and shows courses with active attendee request |  |  |  |  |
| Attendees table is displayed successfully when a course is selected |  |  |  |  |
| When a course is selected “Accept Attendance Request” and “Accept All Attendance Requests For Selected Course” buttons will show |  |  |  |  |
| “Accept attendance request” button accepts the selected attendee |  |  |  |  |
| “Accept All Attendance Requests For Selected Course” button accepts all the attendee requests |  |  |  |  |
| Attendees are updated correctly in the database |  |  |  |  |
| Sorting course data functions correctly |  |  |  |  |
| Sorting attendee data functions correctly |  |  |  |  |
| Response if updating the database fails |  |  |  |  |
| “Manage Attendance For Appointments” button navigates to correct page |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manage Appointment Attendance Page Function | OK | FAIL | Description | Severity |
| Appointment table is displayed successfully and shows appointments with active attendee request |  |  |  |  |
| Attendees table is displayed successfully when an appointment is selected |  |  |  |  |
| Appointment is selected “Accept attendance request” and “Accept All Attendance Requests For Selected Course” buttons will show |  |  |  |  |
| “Accept attendance request” button accepts the selected attendee |  |  |  |  |
| “Accept All Attendance Requests For Selected Appointment” button accepts all the attendee requests |  |  |  |  |
| Attendees are updated correctly in the database |  |  |  |  |
| Sorting appointment data functions correctly |  |  |  |  |
| Sorting attendee data functions correctly |  |  |  |  |
| Response if updating the database fails |  |  |  |  |
| “Manage Attendance For Courses” button navigates to correct page |  |  |  |  |

#### Attendance History

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Completed Appointments Page Function | OK | FAIL | Description | Severity |
| Appointment table is displayed successfully |  |  |  |  |
| Sorting appointment data functions correctly |  |  |  |  |
| “View Attendances For Selected Appointment” button redirects to correct page |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Appointment Attendance Records Page Function | OK | FAIL | Description | Severity |
| Appointment table is displayed successfully |  |  |  |  |
| Attendees table is displayed successfully |  |  |  |  |
| “View Completed Appointments” button redirects to correct page |  |  |  |  |

### User test cases

These test cases are specifically for testing User functionality.

#### Register User Client Type

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register User Client Function | OK | FAIL | Description | Severity |
| Registration succeeds with valid data |  |  |  |  |
| Registration fails with invalid password length (< 6 chars) |  |  |  |  |
| Registration fails with duplicate username |  |  |  |  |
| Registration fails with invalid postcode  (does not exist) |  |  |  |  |
| Registration fails with invalid email  format |  |  |  |  |
| Registration fails with invalid phone number (< 8 digits / non-integer) |  |  |  |  |
| Registration fails with null fields |  |  |  |  |
| Registration response to invalid data |  |  |  |  |
| Registration data stored correctly in database |  |  |  |  |
| Response if registration fails |  |  |  |  |
| Client is redirected to the login page after registration is successful |  |  |  |  |

#### User Login

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login Function | OK | FAIL | Description | Severity |
| Login succeeds with valid data |  |  |  |  |
| Login fails with invalid data |  |  |  |  |
| Login recognizes correct client type |  |  |  |  |
| Login stores client to session |  |  |  |  |
| Login response to null fields |  |  |  |  |
| Login hides password |  |  |  |  |

#### Manage Profile

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manage Profile Function | OK | FAIL | Description | Severity |
| Page shows correct profile data |  |  |  |  |
| Profile updated with valid data is successful |  |  |  |  |
| Update fails with invalid password length (< 6 chars) |  |  |  |  |
| Response if update fails |  |  |  |  |
| Update fails with invalid postcode  (does not exist) |  |  |  |  |
| Update fails with invalid email format |  |  |  |  |
| Update fails with invalid phone number (< 8 digits / non-integer) |  |  |  |  |
| Update fails with null fields |  |  |  |  |

#### Become Attendee Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Find Public Appointments Page Function | OK | FAIL | Description | Severity |
| Public appointments table are displayed correctly |  |  |  |  |
| Sorting appointment data functions correctly |  |  |  |  |
| Selecting an appointment and clicking Apply To Attend The Selected Appointment button navigates to correct page. |  |  |  |  |
| Find Public Courses button redirects to correct page |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Find Public Courses Page Function | OK | FAIL | Description | Severity |
| Public Courses table are displayed correctly |  |  |  |  |
| Sorting Course data functions correctly |  |  |  |  |
| Selecting a course and clicking Apply To Attend The Selected Course button navigates to correct page. |  |  |  |  |
| Find Public Appointments button redirects to correct page |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apply To Attend Appointment Page Function | OK | FAIL | Description | Severity |
| Appointment details are displayed correctly |  |  |  |  |
| Message box with “Yes” and “No” buttons are displayed with a confirmation message. |  |  |  |  |
| “Yes” button updates database with a new attendee where status is 0 , tag Id and Appointment id is correct. |  |  |  |  |
| “No” button redirects to correct page |  |  |  |  |
| “Back To Find Appointments” button is displayed after attendance attempt |  |  |  |  |
| “Back To Find Appointments” button redirects to correct page |  |  |  |  |
| Response if attendee creation fails |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apply To Attend Course Page Function | OK | FAIL | Description | Severity |
| Course details and appointments are displayed correctly |  |  |  |  |
| Sorting Appointment data functions correctly |  |  |  |  |
| “Cancel” button redirects to correct page |  |  |  |  |
| When Apply to course button is clicked a message box with “Yes” and “No” buttons are displayed with a confirmation message. |  |  |  |  |
| “Yes” button updates databasae with a new attendee for each appointment in the couse where status is 0 , tag Id and Appointment id is correct. |  |  |  |  |
| “No” button hides the messagebox |  |  |  |  |
| “Continue” button is displayed after attendance attempt |  |  |  |  |
| “Continue” button redirects to correct page |  |  |  |  |
| Response if attendee creation fails |  |  |  |  |

#### Attendance History

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Appointment Attendance History Page Function | OK | FAIL | Description | Severity |
| Appointment table is displayed successfully |  |  |  |  |
| Attendance time and date is displayed for the selected appointment |  |  |  |  |
| Sorting appointment data functions correctly |  |  |  |  |

### Native Test Cases

#### Attend Appointment

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reading Terminal Login View | OK | FAIL | Description | Severity |
| Login succeeds with valid data |  |  |  |  |
| Login fails with invalid data |  |  |  |  |
| Login hides password |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Appointment View | OK | FAIL | Description | Severity |
| Reading terminal displays appointments successfully |  |  |  |  |
| “Start Appointments” button starts the selected appointment |  |  |  |  |

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attend Appointment Function | OK | FAIL | Description | Severity |
| Reading terminal prompts attendee to swipe his RFID tag. |  |  |  |  |
| Reading terminal stores attendance data in database successfully |  |  |  |  |
| Reading terminal displays message to attendee confirming successful registration |  |  |  |  |
| Registration fails if attendee is not signed up for the appointment |  |  |  |  |
| Response if registration fails |  |  |  |  |

#### Attendance Registration Time

Tester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reading terminal Attendance Registration time | OK | FAIL | Description | Severity |
| Registers attendee and gives attendee feedback within 3 seconds |  |  |  |  |