UAH Fit Vault Software Test Plan

CPE 656/658 Software Studio

Timothy R. Wilkins

Whit J. Sisulak

Glen L. Riden

James J. Duggan IV

12/6/2015

Revision History

Revision #	**Revision Date**	Description of Change	Author
0.1	10/11/15	Initial Draft	T. Wilkins
0.2	10/12/15	Added cover page, revision history, headers, and footers.	J. Duggan
0.3	10/13/15	Added section **Error! Reference source not found.**	T. Wilkins
0.4	10/20/15	Refactored to reflect design change. Added up to section 4.2.3	T. Wilkins
0.5	10/25/15	Added up to section 5.1.2	T. Wilkins
0.6	11/15/15	Finished the first draft	T. Wilkins
0.7	11/22/15	Addressed comments from Program Manager	T. Wilkins
0.8	11/26/15	Further defined CM plan for Mock data, moved unit test requirement from 5.1.3 to 4.1.5	T. Wilkins
0.9	12/03/15	Redefined the CM plan for Mock patient data. Edited a few tests to cover holes that were present in the traceability matrix.	T. Wilkins
0.10	12/6/15	Updated title and approved all changes to the document. Added embedded traceability matrix excel file. Updated table of contents.	J. Duggan
0.11	12/6/15	Removed test and updated traceability matrix.	J. Duggan

Table of Contents

Software Test Plan

1 Introduction

This document shall be used to create a test plan for the medical project.

1.1 Objectives

The main objective of this document is to come up with all the necessary parts of a fully-fledged test plan. This includes identifying the scope of testing that will be done, identifying any references needed to perform the testing, defining the environment in which the tests will be run, and identifying the classifications of the tests themselves. This document will also explore any risks involved with meeting the test plan. Approvals will be noted at the end of the document.

2 Scope

2.1 Identification

This test plan covers the testing of two key pieces of software, the data collection software and the data analysis software. All testing outlined in this document must be run against the final version of these software. The final version of the application containing both pieces of software that passes all testing shall be known as version UAH Fit Vault 1.0.

2.2 System Overview

Below is a linked document containing the Project Proposal. This can be referenced to get a good idea of the overall system.

The UAH Fit Vault software package will be a web application that will accept medical data from users and display the data in a meaningful way. There are two major components to this software. The first is the data collection tool that is used by the users to upload their medical data that is recorded by one of the supported wearable medical devices. There are three different medical devices supported for this project that record various types of data. The data provided by these devices consists of different file formats, and the data is different from device to device. The software will have to determine the contents of each file and how to process them. The software needs to able to take in files that a user has downloaded from their medical devices, process those files, and store the data in a database. The software should have the ability to process multiple files at a time as well as individual files and allow for an activity to be assigned to them by date and time.

The other major component of the web application is the data analysis tools used to analyze the data that is captured from the data collection tool mentioned above. The software needs to perform data analysis over different intervals of time such as one week, one month, etc. There will need to be some way to manage user access to the various medical data that has been inserted into the database that this software will access. Below are some proposed data analysis ideas that can be incorporated into the project.

- Simple Moving Average
- Data correlation discovery between the multiple devices.
- Simply display data that was uploaded to the customer in a graphical format.
- Calculate the user's activity.

The data analysis possibilities will likely not fully be realized until the project team understands the different types of data that are available. Also, there will need to be collaboration with the customer for additions or changes to the data measurements provided by this software. The web application will have to have different levels of user access which will be defined later in this document.

2.3 Document Overview

This document contains information relevant only to the software, testing, and planning associated with such. None of the material in this document is to contain actual names, medical data, or information about actual people living or dead. Medical information is private and no actual data from actual people will be included in this document. As such, this document may be distributed freely and made publicly available.

2.4 Relationships

This document is closely related to the Software Development Plan (SDP). As the SDP is created, this document will need to be maintained to keep its relevance. Any tests described by this document will be closely related to the features, classes, or functions described in the SDP and thus as these elements are updated or changed, this document will need to be updated or changed. Updates to this document will be under revision control as described in the Configuration Management Plan.

3 References

Software Development Plan:

SDP.docx

Project Proposal:

The UAH Fit Vault software package will be a web application that will accept medical data from users and display the data in a meaningful way. There are two major components to this software. The first is the data collection tool that is used by the users to upload their medical data that is recorded by one of the supported wearable medical devices. There are three different medical devices supported for this project that record various types of data. The data provided by these devices consists of different file formats, and the data is different from device to device. The software will have to determine the contents of each file and how to process them. The software needs to able to take in files that a user has downloaded from their medical devices, process those files, and store the data in a database. The software should have the ability to process multiple files at a time as well as individual files and allow for an activity to be assigned to them by date and time.

The other major component of the web application is the data analysis tools used to analyze the data that is captured from the data collection tool mentioned above. The software needs to perform data analysis over different intervals of time such as one week, one month, etc. There will need to be some way to manage user access to the various medical data that has been inserted into the database that this software will access. Below are some proposed data analysis ideas that can be incorporated into the project.

- Simple Moving Average
- Data correlation discovery between the multiple devices.
- Simply display data that was uploaded to the customer in a graphical format.
- Calculate the user’s activity.

The data analysis possibilities will likely not fully be realized until the project team understands the different types of data that are available. Also, there will need to be collaboration with the customer for additions or changes to the data measurements provided by this software. The web application will have to have different levels of user access which will be defined later in this document.

Git Hub installation: https://help.github.com/desktop/guides/getting-started/

4 Software Test Environment

This section describes the test environment. The environment includes the software and hardware needed to test the product. This section will also cover the installation, testing, and control processes of the test software and hardware.

4.1 Local Testing Environment

This is the environment local to the developer's machine. Testing here will include mostly unit testing, but may also include some component or end-to-end testing.

4.1.1 Software

Tools to be used during testing in the local environment are listed below. A detailed description of the tool, its intended use, where or who to get it from and any other relevant details are below. These will be added to as we develop the design. As the design changes new or different tools may become necessary.

- Visual Studio Enterprise 2015 – This tool will be utilized for local development of the software. Visual Studio Enterprise 2015 and the unit test plugin can be obtained here: http://e5.onthehub.com/d.ashx?s=vx43ohwn8y
- Windows Server 2008 – This will be the developing OS. All development and testing should be performed on this OS for this piece of software. It can be downloaded here: http://e5.onthehub.com/d.ashx?s=vx43ohwn8y
- GitHub Desktop – This is the repository software to be used to store all tests. GitHub is located at www.github.com.
- NUnit 2.6.4 – This will be used for unit test development. It can be downloaded here: www.nunit.org/index.php?p=download
- Mock Classes – A mock of each class necessary for performing unit tests will need to be created. It will be the responsibility of the team members to create these mock classes. A mock class has no functionality of the class it is mocking. It shall only respond with a hard coded response needed to perform the test being run.

4.1.2 Hardware and Firmware

Hardware to be used during testing in the local environment is listed below. A detailed description of the hardware, its intended use, where or who to get it from and any other relevant details are below.

- PC – a PC will be required for testing. This will be the responsibility each team member to provide for himself.
- Internet Connection – an internet connection will be required for saving and retrieving the tests. It is also the responsibility of each team member to provide this for himself.

4.1.3 Other Materials

Any other materials to be used during testing on the local environment are listed below. A detailed description of the material, its intended use, where or who to get it from and any other relevant details are below.

- Mock patient data – This is made up data that can be imbedded in the tests for automation purposes. This is to be provided by the team. This data shall be created as needed and is not under configuration management.

4.1.4 Proprietary Nature, Acquirer’s Rights, and Licensing

4.1.5 Installation, Testing, and Control

Each team member shall be responsible for downloading and installing Visual Studio Enterprise 2015 and Windows Server 2008. Code under test and the tests themselves can be downloaded off of GitHub. In order to download from GitHub, the GitHub Desktop software needs to be installed. Follow this tutorial to perform the installation: https://help.github.com/desktop/guides/getting-started/

All the mock classes will need to be developed to perform unit testing. The mock classes should be developed at the same time or before the classes they are needed to test. Following the CM plan for software development performs testing of a mock class.

Mock patient data will need to be created to preform some unit testing. This data does not need to accurately represent specific events. The data should be within the bounds of realism, but does not need to correspond to any action. It will be used to unit test the ability to parse data from each device. It could also be used to test the ability to tag data with an action. The data will be imbedded inside the test. This is the only management that will be performed for this data. It is not considered a configuration management item.

A PC is required to be provided by each team member for testing. Each team member will also need an Internet connection in order to download the tests and code under test.

GitHub and the Internet connection are under test and control by the provider. Windows Server 2008 and Visual Studio are under test by third parties and the versions are under control by the configuration management plan. The PC is assumed to be working and maintained by the team member.

Before code is committed to each team member’s development branch, all unit tests must be passing.

4.1.6 Participating Organizations

Organizations involved in the testing of the data collection software are listed below. Their roles are described with each role.

- Med656 Team – The Med656 team will be responsible for writing unit tests, acceptance tests, and running both of them.
- Customer – The customer will be responsible for customer acceptance tests.

4.1.7 Orientation Plan

All team members are to go through the following tutorial for training purposes for GitHub: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. There will also be a training session for C#/Visual Studio let by James Duggan.

4.1.8 Tests To Be Performed

4.2 Staged Testing Environment

This will be the testing environment on our controlled UAH server. Testing here will be focused on end-to-end tests but may also cover some component testing.

4.2.1 Software

Tools to be used during testing in the staged environment are listed below. A detailed description of the tool, its intended use, where or who to get it from and any other relevant details are below. These will be added to as we develop the design. As the design changes new or different tools may become necessary.

- Windows Server 2008 – This will be the OS running on the stage server. The OS on the server should be running on top of a VM. The customer should provide the OS and the VM.
- Visual Studio Enterprise 2015 – This will be used to develop the tests for this environment.
- Selenium 2.48.0 for C# – This tool is good for automating web interaction and will be used to execute the end-to-end tests that exercise the web GUI. It can be obtained here: http://www.seleniumhq.org/download/.

4.2.2 Hardware and Firmware

Hardware to be used during testing in the staged environment is listed below. A detailed description of the hardware, its intended use, where or who to get it from and any other relevant details are below.

- Server – There will be server hardware provided by the customer. The maintenance and control of this hardware is up to the customer.
- Hypervisor – There will be a hypervisor installed on the server of the customer's choice. Maintenance and control of the host OS and hypervisor are up to the customer.
- VPN Access – The customer will provide VPN access (or other form of remote access) to the guest OS in order for work to be done remotely. This will be up to the customer to provide and the team to maintain.

4.2.3 Installation Testing and Control

The Server, hypervisor, guest OS, and VPN access shall be installed by the customer. The Server, hypervisor, and VPN access shall be tested and controlled by the customer. Issues with these three components will be brought to the customer for resolution. The guest OS shall be tested and controlled by the team.

Visual Studio and Selenium shall be installed by the team onto the guest OS. Issues with these components will be resolved by the team.

4.2.4 Participating Organizations

Organizations involved in the testing of the data collection software are listed below. Their roles are described with each role.

- Team – The team will be involved in writing and performing tests. The team will also be in charge of installing, configuring, and maintaining Visual Studio Enterprise 2015 and Selenium 2.48.0.
- Customer – The customer will be involved with reviewing this document, test case coverage, and test results. The customer will be responsible for signing off on this document. The customer will also be responsible for signing off on the test results of the final release of the software.

5 Test Identification

This section identifies all the tests that will apply to this software test plan. It gives a description of each test. Each test number can be used to identify which requirement it belongs to in the test traceability matrix.

5.1 General Information

This section describes general information about the tests in this test plan. The information includes test levels, test classes, test conditions, test progression, and data recording/reduction/analysis.

5.1.1 Test Levels

The following is a list of levels at which the software will be tested.

- Procedure level – This level will be covered by a minimum of one unit test per procedure. This shall apply to both class procedures as well as any global procedures.
- Class level – This level will be covered by acceptance tests. Each of these tests will be written to directly or indirectly, partially or fully test the software with regard to functionality or non-functional requirements.
- UI level – This level will be covered by end-to-end tests. These tests will be developed directly in relation to a functional or non-functional requirement.

5.1.2 Test Classes

The following is a list of classes/types of tests that will be performed.

- Functional – These will test the behavior of a specific procedure/class/feature with respect to its intended behavior.
- Timing – Tests of this category will check the responsiveness of the system at key points in the software.
- Scale – Tests of this category will check the capacity of the system at key points in the software.
- Erroneous Input – Tests of this category will check the ability of the system to handle invalid input.
- Service Interruption – Tests of this category will check the ability of the system to recover from a loss of some resource (power, network, database…).
- Installation/Setup – Tests of this category will check the ability of the functionality of the setup and installation scripts.
- Security – Tests of this category will check the ability of the system to limit users to their respective features.

5.1.3 General Test Conditions

The following conditions will apply to all tests that are run in this test plan:

- All tests shall pass a minimum of three consecutive test runs for their results to be marked as passed in the customer acceptance test report.
- Before code is committed to each team member’s development branch, all unit tests must be passing.
- Unit tests shall maintain a minimum of 90% code coverage. Unit test coverage is the first chance to uncover bugs and helps code stability, but requiring 100% code

coverage without development team buy in can cause a lack of quality in unit tests. Therefore the 90% was selected as a minimum requirement to encourage high quality unit test development with the goal of 100% code coverage.
- Acceptance tests shall cover 100% of the functional requirements. Properly written acceptance tests are one of the best ways to regularly check that the product being developed will match what the customer wants. Therefore 100% code coverage has been required to verify that all the customers' requests are met.

There shall be three categories of tests: CI tests, customer acceptance tests, and regression tests. CI tests shall consist of very fast tests. All tests in this category must complete running within 5 min. This helps to guarantee that each developer will run the test before or immediately after commitment of any changes. Tests that will push the test run time beyond 5 min. will be put into the customer acceptance category. These tests will be run before code is released to the customer at a minimum. The customer acceptance category shall be subdivided into functional categories that can be run independent of each other to test specific features when hotfixes have been applied. Customer acceptance tests will include tests for current features. Any tests related to outdated, deprecated, or existing features that are no longer being developed should be moved to the regression category. The regression category tests shall be run before each major point release.

5.1.4 Test Progression

N/A

5.1.5 Data Recording, Reduction, and Analysis

Unit test results shall not be delivered to the customer. Acceptance test results and regression test results shall be delivered to the customer for evaluation. All test results and test artifacts generated during the execution of automatic testing of acceptance or regression tests shall be zipped together and delivered to the customer. Test results of semi-automatic or manual tests shall be filled into an appropriate spread sheet. Any failures in all test results delivered to the customer shall be documented and brought to the attention of the customer.

5.2 Planned Tests

5.2.1 Fit Vault Software

5.2.1.1 User Login Customer Acceptance Tests

a) Objective – These tests will verify the requirements in section 3.1.1 of the Software Requirements document have been met. There shall be 2 tests per user. One positive and one negative test to verify the user account can see what he should and cannot see what he should not.
b) Test Level – These tests will be at the UI level.
c) Test Class – These tests will include classes: Functional, Scale, Erroneous Input, and Security.
d) Qualification method – Automated pass/fail analysis.
e) Requirements - See Requirements Traceability.
f) Special requirements – N/A.

g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – Tests related to security must pass with no exceptions.

5.2.1.2 User Information Tests

a) Objective – These tests shall verify all the information required is stored in the user account. There needs to be 1 test per user account type.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.3 User Privacy Tests

a) Objective – These tests shall be used to verify that there are no data that can be considered personal identifiable information available for a patient. This shall be a single test. This could be a simple test consisting of a user visually looking at the types of data that are stored in the database and determining if they could be used to identify an individual. This is not a easily automated task (because human judgment is involved) therefore this should be a visual inspection/human analysis.
b) Test Level – UI level.
c) Test Class – Security.
d) Qualification method – Visual Inspection/Human Analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes by the tester recording what information can be seen and noting if that information can or cannot be used to identify the patient.
i) Assumptions/Constraints – N/A.
j) Safety, security, and privacy considerations – No information revealing the identity of the patient may be included in the system. The system cannot be released under any circumstances with this test failing.

5.2.1.4 Account editing Tests

a) Objective – These tests shall verify that the correct users can edit the proper account information. This will include 1 test case for the physician and 2 test cases for the system administrator. Each test case should test a subset of all possible account information edits.

b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.5 Account Management Tests

a) Objective – These tests shall verify the appropriate accounts can be added, removed, enabled and disabled by the system administrator and physician. This will consist of 4 tests. 1 for each function.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – TBD.

5.2.1.6 Username Tests

a) Objective – These tests shall verify that the usernames created are unique and an error is thrown if attempts are made to copy usernames between accounts. This shall consist of 1 test to verify that users with unique names can be created (a minimum of 3 usernames must be used) and 1 test to verify that users with identical names cannot be created (a minimum of 2 usernames must be used).
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis (steps will be created in test details that automation shall follow).
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.7 Patient ID Tests

a) Objective – These tests shall verify that the patient IDs that are created when a patient account is created are all different. No patient ID number may be the same. This shall be verified by a single test that creates at 5 users at a minimum and verifies their IDs are all unique.

b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.8 Password Tests

a) Objective – These tests shall verify that only passwords that meet the security criteria can be created. This will need 4 separate tests that each attempt to create a password missing one of the 4 requirements. It will also need a single test verifying that it can create a password matching all the requirements.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – TBD.

5.2.1.9 Username/Password Retrieval Tests

a) Objective – These tests shall verify that there exists a utility to retrieve a forgotten username or password. This will be a single test.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Visual Inspection for any processes that might not be automated, otherwise Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information; or automated test generated artifacts and test results.
h) Type of data recording/reduction/analysis – Notes by the tester reporting what mechanism the system has implemented and if it works. If automated, Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – TBD.

5.2.1.10 Data File Processing Interface Tests

a) Objective – These tests shall verify data files can be uploaded. This shall consist of 6 tests. Upload single file with no assigned activity, upload multiple files with no assigned activity, upload single file with assigned activity, upload multiple files with assigned activity, upload single file with at a minimum 3 activities, and upload multiple files with at minimum 3 activities each. These tests with in combination with the other tests listed in this document should provide adequate coverage, but for any failures that are found after passing these tests, a new test shall be created.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.11 File Upload Interface Tests

a) Objective – These tests shall verify the interface for uploading a file meets the requirements (specifically 3.1.2.3). This can be a single visual inspection. Since the uploads have been tested, this test is purely to check the visual web GUI.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Visual Inspection of the web GUI.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester recording the test status and pertinent information about the system and how it hits or misses the requirements.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.12 Data Type Tests

a) Objective – These tests shall verify all the types of data required can be processed. This shall consist of 3 positive tests (1 for each data file type) and a single negative test. As failures are identified new tests shall be created.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.

h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.13 Database Function Test

a) Objective – These tests shall verify that when a file is processed, information shows up in the database. This can be a single test.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.14 Experiment Tests

a) Objective – These tests shall verify that an experiment can be created and viewed correctly. There should be 1 test case to create an experiment with a complex query with experiment administrator and delete the experiment with the experiment administrator. There shall also be 1 test procedure to create an experiment with a simple query and delete the experiment with the system administrator. Both of those can be automated. There should also be a couple visual inspection tests checking that other user types cannot create experiments and that the experiment looks correct when viewed by the experiment administrator. As failures are found, new tests shall be created to catch those failures.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis and visual inspection.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results, and tester information.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report, and notes by the tester verifying the inability to create experiments by incorrect users and notes by the tester verifying the correct view of an experiment.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.15 Data Exportation Tests

a) Objective – These tests shall verify that all data exportation features work. There shall be one test for an experiment, one test for a patient, and one test for a physician. Graphical exports will need Visual inspection.
b) Test Level – Class level.
c) Test Class – Functional.
d) Qualification method – Automated pass/fail analysis, Visual Inspection of any graphical exports.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – automated test generated artifacts, test results, tester information.
h) Type of data recording/reduction/analysis – Logger (Info and Debug), test framework test report, notes for the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.16 Medical Data View Tests

a) Objective – These tests shall verify that there is an interface for displaying medical data. This test may be manual visual verification because there may be problems with the UI that cannot be caught by automation.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Visual Inspection of the GUI.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.17 OS Test

a) Objective – These tests shall verify the correct OS the system is installed on.
b) Test Level – UI level.
c) Test Class – Non-Functional.
d) Qualification method – Visual Inspection.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.18 SQL Database Test

a) Objective – These tests shall verify the correct version of the database.

b) Test Level – UI level.
c) Test Class – Non-Functional.
d) Qualification method – Visual Inspection.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.19 Network Connection Test

a) Objective – These tests shall verify the network/internet connection of the machine hosting the system.
b) Test Level – UI level.
c) Test Class – Non-Functional.
d) Qualification method – Visual Inspection that the network/internet connection of the host machine is working.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

5.2.1.20 Graph View Tests

a) Objective – These tests shall verify that the graphs are correctly displayed.
b) Test Level – UI level.
c) Test Class – Functional.
d) Qualification method – Visual Inspection.
e) Requirements – See Requirements Traceability.
f) Special requirements – N/A.
g) Recorded data type – test results, tester information.
h) Type of data recording/reduction/analysis – Notes from the tester regarding the visual inspection.
i) Assumptions/Constraints – N/A.
j) Safety, security, privacy considerations – N/A.

6 Test Schedule

	Jan	Feb	Mar	Apr	May
On-site		🟩	🟩	🟩	🟩
Pretest	🟩	🟩			
Operational Data					

Collection					
Testing		🟩	🟩	🟩	🟩
STR Approval					🟩
STD	🟩	🟩	🟩	🟩	

7 Requirements Traceability

See Tractability Matrix.

TraceabilityMatrix.xls
x

8 Notes

None.

9 Annexes

None.

10 Approvals