**COMPX-374**

**TESTING PLAN  
\_\_\_\_\_\_\_\_\_\_\_\_**Testing Plan for the MEDIC2MEDIC project section 2

Group L  
Natch Sadindum, Tristan Brynildsen, Shashank Mylarapu, Taran Kern

# CONTENTS

1. – **Test Plan Overview**
   1. **– Introduction**
   2. **– References**
   3. **– Project Overview**
   4. **– Scope**
   5. **– Testing Roles**
2. **– Test Discussion**
   1. **– Test Levels**
   2. **– Success Criteria**
   3. **– Suspension Criteria**
   4. **– Test Completeness**
   5. **– Test Criteria**
      1. **– Functionality**
      2. **– Usability**
      3. **– Performance**
      4. **– Scalability**
   6. **– Test Approach**
   7. **– Test Deliverables**
   8. **– Environment/Software/Testing Needs**
   9. **– Training Needs**
3. **– Tests**
   1. **– Functionality Tests – Unit Tests**
      1. **– Current Student Management**
      2. **– Donor Management**
      3. **– Donor Login**
      4. **– Prospective Student Management**
      5. **– Application Form**
      6. **– Graduate Student Management**
   2. **– Functionality Tests – Integration Tests**
   3. **– Functionality Tests – System Tests**
   4. **– Usability Tests**
   5. **– Performance Tests**
   6. **– Scalability Tests**

**4.0 – Meeting Minutes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current Version** | **Modified by** | **Date** | **Changes** |
| **1.0** | **Natch Sadindum** | **6/10/20** | **Initial Version** |
| **1.1** | **Natch Sadindum** | **7/10/20** | **Added Sections 1.0-2.8** |
| **1.2** | **Natch Sadindum** | **10/10/20** | **Sections 1.0-2.8 Complete, added section 3.0 and tests** |

# 1.0 – Test Plan Overview

## 1.1 - Introduction

The purpose of this document is to outline the tests required to ensure the Medic2Medic project is fully functional.

This document will also outline the staff and testers required, their roles, and their purpose.

The document will also describe the resources, environmental and software requirements for performing the tests listed within, the expected deliverables resulting from the testing process, the test methodology and the pass/failure thresholds and their consequences for the development process.

## 1.2 – References

Medic2Medic Software Requirements Specification

Medic2Medic Software Design Specification Document

Medic2Medic Software request document  
- located in project list

## 1.3 – Project Overview

The Medic2Medic project (section 2) is a data management system for the Medic2Medic program. The program provides a clean, easy-to-use system for staff members to access, control, modify and transfer data of students involved in the Medic2Medic education program.

The program is divided into 6 primary sections:  
- Graduate Management:   
Obtaining, storing, processing and modification of graduate data. Staff can modify graduate data directly. Graduate type users can send in survey data which a staff user can then use to make publications  
  
- Current Student Management  
NOTE: not programmed by our team, will be tested by the other team assigned to the sister project.

- Donor Management  
Creation of new donor type users, staff management of donor user data, assignment of students to donors, handling of donations, ability for donor users to view assigned student profiles.

- Donor Login  
Ability for donor users to log in and thus access the donor user functions in donor management. Ability for donor users to change their login details.

- Prospective Student Management  
Organisation and approval of prospective students and their data. Creation of new profiles for prospective students and uptake from the application form.

- Application Form  
Website form that records and sends student data to the database for use in the Prospective Student Management component.

## 1.4 – Scope

The test plan will contain tests that check against four main criteria:  
- Functionality  
- Usability  
- Performance  
- Scalability

The functionality tests must be performed at 3 levels (described in section 2.1) and ensure that all six components described in section 1.3 function correctly, and that the system also functions correctly with no unexpected interactions or code failures that prevent the program from fulfilling its functional requirements.

The usability tests will cover all common user functions for all user types.

The performance and scalability tests will cover many common and uncommon scenarios while testing effects on functionality with increasing load to find possible load-related faults.

A short example of some basic functionality tests is included below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COMPONENT – APPLICATION FORM**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **TEST ID** | **User Type** | **Target** | **Instructions** | **Pass Criteria** | | F-AF-1 | Prospective Student | User can fill text forms | Access page as Prospective Student User.  Enter data of correct format into forms | Text forms are selectable and fillable, text appears in text forms | | F-AF-2 | Prospective Student | Text forms only accept correct input | Access page as Prospective Student User.  Enter data of incorrect format into forms  Attempt to submit data | Warning text appears next to forms incorrectly filled | |

## 1.5 – Testing Roles

The tester types and their roles will be described below.

|  |  |  |
| --- | --- | --- |
| Tester Type | Purpose | Assigned to: |
| Test Manager | Manages all test members, collects data from tests | Any group member |
| Test Builder | Develops tests for the four categories | Any group member, ideally one with good knowledge of program code |
| Functionality Tester | Performs tests created by test builder targeting program functionality. Provides a filled test report to test manager upon test plan completion. | Any group member, possible to outsource to others. |
| Usability Tester | Performs tests created by test builder targeting usability. Commentary on difficulty of using program required. | Non-group members, project stakeholders and possible users. |
| Usability Test Supervisor | Supervises usability testers, records results from the usability tests | Any group member, can be outsourced if needed, so group members can continue development work. |
| Performance Tester | Tests performance and responsiveness of program | Any group member. |
| Scalability Tester | Performs tests targeting scalability and program load. | Any group member, ideally one experienced with supporting components and able to modify size/memory and familiar with testing software. |

# 2.0 – Test Discussion

In this section, we will provide an overview of our testing methodology, and elaborate on details such as the testing levels, success criteria for our tests, test completeness, the four criteria in which we are testing our program, the approach for the different tests performed, the software/environment and other needs in order to perform our tests as well as the expected deliverables resulting from testing our program.

## 2.1 – Test Levels

In this test plan, tests will be divided into multiple levels:

**Unit Tests** – target single functions, subcomponents or larger blocks up to component size. Ensures that smallest blocks of code work correctly on their own.

**Integration Tests** – targets multiple components or modules and the interfaces between them. Ensures that components interact with each other appropriately and finds interface faults.

**System Tests –** targets all components and interfaces together as a single unit. Checks whether the entire application meets all functional requirements set in the software requirements and design specification.

## 2.2 – Success Criteria

Each tested section will need to pass a minimum test success rate to be considered acceptably functional. This threshold may vary based on the testing level, type of tests and purpose of tests.

The pass threshold for a functionality test set will be set to **80%**.

Once the test set is complete, tester reports should be sent to the test manager and checked.

If the testing success rate exceeds the threshold, then testing should continue another testing set or component. Any failed tests should be recorded and referred to developers to be addressed when possible.

Usability testing requires observation of users achieving tasks based on time taken and number of actions. Each usability test set should have these parameters recorded. If these parameters exceed a reasonable threshold based on the task given, the user interface or action sequence should be reviewed and possibly changed by the development team.

Performance tests can be measured based on the percentage of requests meeting the set response time requirement. An example is that “prospective student data requests to the database should be completed within 2 seconds 95% of the time”.

Scalability tests will be measured in the same way as performance tests, with different scenarios simulating varying load. An example is that “prospective student data requests to the database should be completed within 2 seconds 95% of the time while above 90% maximum database request load.

If performance testing and scalability testing does not meet the minimum threshold set, then the test manager should refer this issue to the development team. If the performance and scalability tests are more than 20% below the threshold, the issue should be marked as urgent and immediately addressed by a member of the development team.

## 2.3 – Suspension Criteria

If the testing success rate for any testing set for functionality tests is below the minimum threshold of 80%, testing should be halted immediately, and the testing report be compiled and sent to the test manager. The development team should be reassigned as necessary to address and fix the faults discovered.

If any fatal errors are discovered through functionality testing, testing should also be halted, and the error should be addressed.

It is not expected for the testing success rate to reach 100%. Once the success rate exceeds 90% and the only errors found are minor, developers can be reassigned to other tasks.

For usability testing, the test data should be reviewed by the team upon test completion. Testing should not be suspended until the test set is complete. If the team, testers and tester feedback indicate that the interface and usability level is unsuitable, the developers can be alerted after the tests are performed.

For Performance and Scalability testing, if the tested component does not reach the set threshold, testing for that component will immediately be stopped. The development team should be notified and be reassigned as required.

## 2.4 – Test Completeness

For functionality testing, all tests in each testing set should be run, i.e. test run rate should be 100% unless a clear reason is given, for example, if an issue/fault has been documented but not fixed.

At the integration level, all test sets included in each integration test should be successful to declare an integration test set completed and successful.

At the system level, all test sets must be complete without critical errors.

For usability, performance and scalability, all tests are expected to be run unless clear reason is given. Exceptions can be made for and scalability tests at low program load. Usability tests may also be skipped if the team and testers are in agreement that the usability is at an acceptable level after extensive testing.

## 2.5 – Testing Criteria

### 2.5.1 – Functionality

Functionality tests for this application are to ensure that all components and subcomponents fulfil their intended purpose and function without bugs or errors that may interfere with the running of the program.

This is the most important set of tests in this master plan, as basic functionality is a prerequisite for all other test types.

The objective of the functionality tests is to ensure that there are no faults in the system, and that any possible faults or errors are handled gracefully.

### 2.5.2 – Usability

Usability tests are to test how easy/complicated it is for user types to use the functionality provided by our program. Given that our application runs without faults, we need to check that the functionality provided by our code can easily be accessed and executed by the user in simple, intuitive ways.

These tests will target the application’s user interface, its design and appearance, as well as the action sequences required to achieve tasks.

Ideally a user should be able to perform any given task and access all user-appropriate functionality in as fewest steps as reasonable and be given controls that make it easier to navigate and find the required functions.

### 2.5.3 – Performance

Performance tests are to check whether our application can handle requests within a given timeframe. Fast performance and responses are essential on the usability front and will require changes that may affect the functionality front.

Due to the nature of our application (a data management application without need for massive data transfer), we can afford to set modest performance targets. We have no need for maximum speed responses

### 2.5.4 – Scalability

Scalability tests are to check if our application and its supporting systems are capable of handling increasing/decreasing amounts of load without unacceptable decreases in performance.

As we gain more concurrent users of this application and increasing number of requests at any time, this could place a strain on our support systems which can cause performance drops for all users, which is frustrating on a performance and usability standpoint. These tests may also expose possible faults or functionality problems that may not be revealed by earlier functionality tests.

## 2.6 – Test Approach

### 2.6.1 – Functionality Test Method

A tester should be given a test list and test report document (digital or physical) and should be stationed at a computer with the current build of the application connected to the database. The tester should manually perform each test according to the instructions given and record the results on the test report document and pass the results to the test manager or supervisor once finished.

Functionality tests at the unit and integration level should be performed with a white-box approach, as we need to know what happens to the data as it traverses our program. It would make finding errors significantly easier.

The system tests should be performed in a black-box manner, simulating real-time use by user types. They will be used to evaluate whether our program meets its functional requirements.

### 2.6.2 – Usability Test Method

A test supervisor and a tester must be stationed in the same room. The supervisor will provide a list of tasks which the tester must perform. The tester may have to perform several task lists.

Each task list has a set of tasks related to a user type. A tester may be assigned tasks lists resembling the tasks expected to be performed by selected user types.

For example, a tester may be expected to fill in a prospective student application form, in which they take the role of a prospective student. They may also be provided a list of tasks directing them to find a graduate student in the database, modify their details and save, in which they take the role of a staff user.

The test supervisor should record how long it takes for a user to complete the given tasks, and take feedback from the tester in terms of difficulty and suggestions for changes if needed.

### 2.6.3 – Performance Test Method

A tester will be provided with test instructions and test scenarios in which the tests are simulating. The tester can manually record performance via timer and an active application build.

The tester may alternatively use dedicated software to measure response time and other data.

The tester should automate this process if possible, allowing them to run tests more efficiently.

The tester will record performance data down on a test report and send to the test supervisor or manager once finished.

### 2.6.4 – Scalability Test Method

As above in 2.6.3 – Performance Test Method, the tester is provided test instructions and scenarios which they will simulate using an active build of the application. They are to use dedicated software to simulate increasing load on each scenario provided and record response time and other data similar to in 2.6.3.

This process can be achieved manually but it is recommended for the tester to use dedicated software to automatically perform these tests.

## 2.7 – Test Deliverables

The following documents are expected as part of completing the test plan:

**Test Lists –** Contains all tests grouped into lists for testing certain components, or the system. Should be passed to testers. They should contain the instructions for the tests and the pass criteria/expected outcome.

**Test Set Summary** – Gives percentage of tests attempted/successful, marks tests failed for the functionality tests.

**Test Log** – Test and test sets that have been performed, when they were performed and who by. Includes the Test Set Summary for that test set.

**Test Data** – Contains tabled data from the scalability and performance tests, such as response time, load level and

**Usability Test Responses –** Contains feedback and time taken for usability testers to complete task lists.

## 2.8 – Environment/Software/Testing Needs

In this section we will discuss the testing environment and requirements for performing the tests.

All tests will require a computer with sufficient hardware performance to run any of our tests. The developers already have such hardware available to them, but outsourced testers may not. Outsourced testers will need to be provided hardware in this case.

### 2.8.1 – Functionality Test Needs

In the case on manually performed functionality tests, only a computer with an active build of our application.

Alternatively, dedicated testing software could be used to automate the process. Each time the program is updated, the application is passed through a test-pipeline (for our build, we could use Microsoft Azure’s pipeline options) which checks if the build passes a set of critical and non-critical tests. If it fails the pipeline, the build is rejected and the developer is sent a report of tests passed and failed.

### 2.8.2 – Usability Test Needs

For usability tests, no specific environments are required, only a usable build of the application with access to a prefilled database.

### 2.8.3 – Performance and Scalability Test Needs

Performance tests can be performed manually with minimal software if needed, using an active build of the application and a timer at minimum.

Software such as Apache JMeter or NeoLoad can be used to perform automated performance tests on our software. Considering that performance tests must be performed many times for many scenarios, this is a preferable solution.

Scalability testing will require external software to force loading on our software. Apache JMeter and NeoLoad can also perform in these roles.

## 2.9 – Training Requirements

Testers working on the functional tests will need to be familiar with the application in order to discover and diagnose faults. It is recommended to use a developer as a tester or a supervisor for these tests.

Testers working on usability do not have to be familiar to the application. It is recommended for them to be unfamiliar, as this allows the test supervisor and manager to collect truthful data about application usability.

Performance and scalability testers should be familiar with the program and should be trained to use external testing software (in this case Apache JMeter or NeoLoad) to make the testing process more efficient.

When the program is released fully, staff users will need training or a training manual as a guide for using the application. Documentation should be released at program release to aid this process. In-program can be implemented if required, but is not strictly necessary.

Non-staff users do not need training, but should be provided with in-program or external help online if they get stuck. Given the simplicity of operating the program and its general design, we do not foresee users having trouble with the program to the degree where dedicated training is required.

# 3.0 – Tests

This section will contain many test lists, containing tests that should be performed to ensure that our application works correctly and to a high standard across many scenarios.

Each test will be denoted with a unique identification number. The format will vary between categories, but will be described in each section.

Note that some sections may have significantly less tests than others. The tests currently in the list reflect the current state of development of those components, the expected functionality of those components and the purpose of those components as set out in the Software Design Document. As development continues, this plan will be updated as required to ensure functionality.

## 3.1 – Functionality Tests – Unit Tests

This section will contain a list of functionality tests, at unit level, divided into the sections relating to each component. The test list will also contain a section for testers to report test success and provide feedback.

Note that some tasks in this section are classed as both unit and integration tests, as some of these tests check the interaction between itself and the database. These tests will be listed in both sections, but will be fully described only in this section.

#### 3.1.1 – Current Student Management

Ignore this section, testing of this section is performed by another group and will be included in their testing plan.

#### 3.1.2 – Donor Management

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-UDM-1 | Donor List Display | Donor list function displays donors in the database | Staff User | Login as staff user  Access donor function via navigation bar  Check table for database content | Data loaded successfully, donor data present in table |  |  |
| F-UDM-2 | Donor List Sort | Donor data can be sorted when using sort function on informative columns | Staff User | While accessing donor management function as staff user, use the sort function to sort the donor data by category | Donor data should be sorted correctly by selected category |  |  |
| F-UDM-3 | Donor List Sort | Donor data is not sorted when using sort function on non-informational columns | Staff User | While accessing donor management function as staff user, use the sort function to sort the donor data by category | Trying to sort donor data using non-informative columns should not sort data. |  |  |
| F-UDM-4 | Donor List Search – informative columns | Informative donor data columns can be searched via search function | Staff User | While accessing *donor management function* as staff user, use search bar to type for donor name “Test Tester”.  Test should be repeated with data from all informative columns. | Donor with name “Test Tester” should appear.  Donor with data searched should appear if donor is present. |  |  |
| F-UDM-5 | Donor List Search – Non-informative columns | Donor data search functions correctly by not searching non-informative columns | Staff User | While accessing *donor management function* as staff user, use search bar to type for any data found in non-informative columns. | No donors should appear, if data searched is only found in the non-informative columns. |  |  |
| F-UDM-6 | Donor List – Add new donor | Using add new donor function redirects user to new page | Staff User | While accessing *donor management function*, use the *add new donor* function. | User should be redirected to a new page. |  |  |
| F-UDM-7 | Donor List – Add page exit with data and without changes | Exiting the donor add page does not change database data | Staff User | While accessing *donor management function* and the *add donor page*, exit the page with “Name” in the first name field | No new donors with the first name “Name” should appear in the database |  |  |
| F-UDM-8 | Donor List – Add page exit empty without changes | Exiting the donor add page does not change database data | Staff User | While accessing *donor management function* and the *add donor page*, exit the page with no data in any fields | No new donors with the should have been added to the database, |  |  |
| F-UDM-9 | Donor List – Add donor incomplete | Error/field checking when adding new donors | Staff User | While accessing *add donor function*, attempt to add a new donor with fields empty or within incorrect input types | Error/data verification message should appear, no new donors added |  |  |
| F-UDM-10 | Donor List – Add page success | Successful addition of a new donor to database | Staff User | While accessing the *add donor page*, fill all fields with correct input and confirm adding a new donor. Check that donor is added back in donor list. | A new donor with the details entered should be present in the database and visual donor list. |  |  |
| F-UDM-11 | Donor List – Delete donor redirection | Deleting a donor in the database | Staff User | While accessing *donor list* function, attempt to delete a donor using the delete control | User should be redirected to a confirmation page |  |  |
| F-UDM-12 | Donor List – Delete donor details | Deleting a donor in the database | Staff User | While accessing *donor list* function, attempt to delete a donor using the delete control, check the donor details in the list and on confirmation page | Redirected page should display the details of the donor selected for deletion |  |  |
| F-UDM-13 | Donor List -Delete cancel | Deleting a donor in the database | Staff User | While deleting a donor, cancel the deletion process using the cancel button | Donor selected should not be deleted, they should still be present in database with no details changed |  |  |
| F-UDM-14 | Donor List – Delete Complete | Deleting a donor in the database | Staff User | While deleting a donor, confirm delete on redirected page. | Donor deleted should disappear from database, user should be redirected to a new page. |  |  |
| F-UDM-15 | Donor List – Edit Redirect | Editing donor details in the database | Staff User | While accessing *donor management,* use edit function to edit a donors details | User should be redirected to a new page to edit donor details |  |  |
| F-UDM-16 | Donor List – Edit Correct information displayed | Editing donor details in the database | Staff User | While on the donor details editing page, , check if the donor details here match the details in the donor list | The details should be identical |  |  |
| F-UDM-17 | Donor List – Edit information editable | Editing donor details in the database | Staff User | While in the donor details editing page,, check that details are in editable fields | The details should be freely editable |  |  |
| F-UDM-18 | Donor List – Non save edited info | Editing donor details in the database | Staff User | While in donor details editing page, change some details and exit without saving and check if any details are changed in the donor list | No change in donor details |  |  |
| F-UDM-19 | Donor List – Save edited info | Editing donor details in the database | Staff User | While in donor details editing page, change some details and exit while saving. Check that donor details have changed successfully. | Donor details should change in the donor list. |  |  |
| F-UDM-20 | Donor List – Update after edit | Editing donor details in the database/synchronisation of changes | Staff User | While in donor list page, edit a donor and save, check that donor details have changed in the donor list. Perform this concurrently with another user. | Donor details should update every time someone edits and saves details. |  |  |
| F-UDM-21 | Donor List – Assign students | Editing donor details in the database | Staff User | While in donor details editing page, a staff member should add a new student to the donor’s assigned students and save. | A new student should be assigned to the donor user, confirm via donor list and donor details edit page. |  |  |

#### 3.1.3 – Donor Login

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-UDL-1 | Donor Login | Donor user can log in to system | Donor User | As donor user, access the login page and use provided login details to log in | User can log in and is redirected to the donor user portal |  |  |
| F-UDL-2 | Donor Login Failure | Donor user can log in to the system | Donor User | Attempt to log in as a donor user but with incorrect details | User does not login, error message is shown and user is left on the login page |  |  |
| F-UDL-3 | Donor donation microservice redirect | Donor user can donate money | Donor User | When logged in as donor user, access the donate function and attempt to use microservice to donate money | User should be redirected to a page dedicated to the donation microservice |  |  |
| F-UDL-4 | Donor donation microservice page exit | Donor user can donate money | Donor User | When on payment microservice page, exit the page | User should be redirected back to the donor user portal |  |  |
| F-UDL-5 | Donor donation microservice incorrect details | Donor user can donate money | Donor User | When on payment microservice page, enter incorrect details | Error messages should appear informing user that their details are incorrect |  |  |
| F-UDL-6 | Donor donation microservice correct details confirmation | Donor user can donate money | Donor User | When on payment microservice page, enter correct details and confirm | User should be given a confirmation dialog |  |  |
| F-UDL-7 | Donor donation microservice complete donation | Donor user can donate money | Donor User | Complete test F-UDL-6 and confirm. | User should be provided with a receipt via email and an appropriate message. |  |  |
| F-UDL-8 | Donor student assignment redirect | Donor user can check assigned students | Donor User | Donor user should access assigned student profile function via donor portal | Donor user will be moved to assigned student profile page |  |  |
| F-UDL-8 | Donor student assignment empty | Donor user can check assigned students | Donor User | Check assigned student profiles while the donor has no assigned students | No students and their profiles should appear on the page |  |  |
| F-UDL-9 | Donor student assignment non-empty | Donor user can check assigned students | Donor User | Check assigned student profiles while the donor has more than one student assigned to them | Handles representing the assigned student should appear on the page |  |  |
| F-UDL-10 | Donor student assignment open profile | Donor user can check assigned students | Donor User | Click on student profile handles found in test F-UDL-9 | Clicked handle should bring up page or dialog showing user details of assigned student |  |  |
| F-UDL-11 | Donor access change log in details | Donor user can modify login details | Donor User | While logged in, access the change details function | User should be redirected to a page with appropriate form controls for changing password |  |  |
| F-UDL-12 | Donor change log in details error check | Donor user can modify login details | Donor User | Access login details change function, enter new password that does not meet requirements and enter and confirm | Error message should appear and prevent a password change, forms are cleared |  |  |
| F-UDL-13 | Donor change log in details success | Donor user can modify login details | Donor User | Continue from F-UDL-12, enter acceptable password and confirm change | User should receive a positive message and email, and be logged out and sent to login page. |  |  |
| F-UDL-14 | Donor change log in details re-test | Donor user can modify login details | Donor User | Use log in page to log in using new password changed in F-UDL-14 | User should be able to successfully log in again. |  |  |

#### 3.1.4 – Prospective Student Management

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-UPSM-1 | PSM Accessible | Function is accessible by staff user | Staff User | Log in as a staff user, use navigation menu to access prospective student management portal | Component is accessible, all prospective students in database should be visible in on screen table/list |  |  |
| F-UPSM-2 | PSM data sort informative columns | PSM data can be sorted when using sort function on informative columns | Staff User | While accessing PSM function as staff user, use the sort function to sort the data using an informative column (like name) | PSM data should be sorted correctly by selected category |  |  |
| F-UPSM-3 | PSM data sort uninformative columns | PSM data is not sorted when using sort function on non-informational columns | Staff User | While accessing PSM function as staff user, use the sort function to sort the donor data by category using a non-informative column as target (such as the column with edit student button) | PSM data should not be sorted and rearranged |  |  |
| F-UPSM-4 | PSM List Search – informative columns | Informative PSM data columns can be searched via search function | Staff User | Use search bar to type for prospective student name “Test Tester”.  Test should be repeated with data from all informative columns. | Prospective student with name “Test Tester” should appear.  Student with data searched should appear if donor is present. |  |  |
| F-UPSM-5 | PSM List Search – Non-informative columns | Non-informative PSM data search functions correctly by not searching non-informative columns | Staff User | Use search bar to type for any data found in non-informative columns. | No prospective students should appear if data searched is only found in the non-informative columns. |  |  |
| F-UPSM-6 | PSM List – Add new prospective student | Staff user can add new prospective students | Staff User | Use the add new prospective student button located on the page in the PSM portal | User should be redirected to a new page, similar to the application form and donor add page, but for prospective students |  |  |
| F-UPSM-7 | PSM List – Add page exit with data and without changes | Exiting the PSM add page does not change database data | Staff User | Exit the *add prospective student* page with “Name” in the first name field | No new prospective students with the first name “Name” should appear in the database |  |  |
| F-UPSM-8 | PSM List – Add page exit empty without changes | Exiting the PSM add page does not change database data | Staff User | Exit the *add prospective student* page while leaving fields empty | No new prospective students have been added to the database |  |  |
| F-UPSM-9 | PSM List – Add donor incomplete | Error/field checking when adding new prospective students | Staff User | While accessing *add prospective student function*, attempt to add a new prospective student with fields empty or within incorrect input types | Error/data verification message should appear, no new prospective students added |  |  |
| F-UPSM-10 | PSM List – Add page success | Successful addition of a new donor to database | Staff User | While accessing the *add prospective student page*, fill all fields with correct input and confirm adding a new prospective student. Check that student is added back in the PSM list. | A new prospective student with the details entered should be present in the database and visual prospective student list. |  |  |
| F-UPSM-11 | PSM List – Delete prospective student redirection | Deleting a prospective student in the database | Staff User | While accessing the *prospective student list*, attempt to delete a student using the delete control | User should be redirected to a confirmation page |  |  |
| F-UPSM-12 | PSM List – Delete student details | Deleting a prospective student in the database | Staff User | While accessing the *prospective student list*, attempt to delete a student using the delete control, check the student details in the list and on confirmation page | Redirected page should display the details of the student selected for deletion |  |  |
| F-UPSM-13 | PSM List -Delete cancel | Deleting a prospective student in the database | Staff User | While deleting a prospective student, cancel the deletion process using the cancel button | Prospective student selected should not be deleted, they should still be present in database with no details changed |  |  |
| F-UPSM-14 | PSM List – Delete Complete | Deleting a prospective student in the database | Staff User | While deleting a prospective student, confirm delete on redirected page. | Student deleted should disappear from database, user should be redirected to back to the prospective student portal and list. |  |  |
| F-UPSM-15 | PSM List – Edit Redirect | Editing prospective student details in the database | Staff User | While accessing the *prospective student management,* use edit function to edit a student’s details | User should be redirected to a new page to edit prospective student details |  |  |
| F-UPSM-16 | PSM List – Edit Correct information displayed | Editing prospective student details in the database | Staff User | While on the prospective student details editing page, check if the student details here match the details in the student list | The details should be identical |  |  |
| F-UPSM-17 | PSM List – Edit information editable | Editing prospective student details in the database | Staff User | While in the prospective student details editing page, check that details are in editable fields | The details should be freely editable |  |  |
| F-UPSM-18 | PSM List – Non save edited info | Editing prospective student details in the database | Staff User | While in prospective student details editing page, change some details and exit without saving and check if any details are changed in the prospective student list | No change in prospective student details |  |  |
| F-UPSM-19 | PSM List – Save edited info | Editing donor details in the database | Staff User | While in prospective student details editing page, change some details and exit while saving. Check that student details have changed successfully. | Prospective student details should change in the prospective student list. |  |  |
| F-UPSM-20 | PSM List – Update after edit | Editing prospective student details in the database/synchronisation of changes | Staff User | While in prospective student list page, edit a prospective student and save, check that student details have changed in the prospective student list. Perform this concurrently with another user. | Prospective student details should update every time someone edits and saves details. |  |  |
| F-UPSM-21 | PSM List – Approve student success | Change a prospective student to a current student | Staff User | While in the prospective student list, edit a prospective student’s details and use the controls to approve a prospective student and confirm. | After the confirmation screen, the user should receive a affirming message and the prospective student will no longer be in the prospective student list, having being moved to the current students section. |  |  |
| F-UPSM-22 | PSM List – Approve student cancel | Change a prospective student to a current student but cancel | Staff User | While in the prospective student list, edit a prospective student’s details and use the controls to approve a prospective student except do not confirm and exit the page back to the prospective student list. | The prospective student selected should still be present in the prospective student list. No changes to the student’s status. |  |  |

#### 3.1.5 – Application Form

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-UAF-1 | Application Form fields error check | Application form error checking is functional | Prospective Student User or Staff User | Access the application form option on the website. Enter inappropriate input in any of the fields | Fields with inappropriate input will display an error message |  |  |
| F-UAF-2 | Application Form failed submit | Application form error checking on submission is functional | Prospective Student User or Staff User | While in application form page, enter inappropriate input into fields and attempt to submit | Application form will not be submitted, error message displayed |  |  |
| F-UAF-3 | Application Form exit without submission | Application form exit does not submit form, confirmation message exists | Prospective Student User or Staff User | Access the application form option on the website. Exit the application form page without submitting | Leaving page confirmation dialog should appear, application form should not be submitted, no change in database for staff user when checked. |  |  |
| F-UAF-4 | Application Form successful submission | Application form can be submitted | Prospective Student User or Staff User | Access the application form, fill all fields appropriately and submit and confirm. | Confirmation message should appear and then success message should appear, user will be redirected out of the application form. Staff users can check if new prospective student appears in database. |  |  |

#### 3.1.6 – Graduate Student Management

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-UGSM-1 | UGSM Accessible | Function is accessible by staff user | Staff User | Log in as a staff user, use navigation menu to access graduate student management portal | Component is accessible, all prospective students in database should be visible in on screen table/list |  |  |
| F- UGSM -2 | UGSM data sort informative columns | UGSM data can be sorted when using sort function on informative columns | Staff User | While accessing UGSM function as staff user, use the sort function to sort the data using an informative column (like name) | UGSM data should be sorted correctly by selected category |  |  |
| F- UGSM -3 | UGSM data sort uninformative columns | UGSM data is not sorted when using sort function on non-informational columns | Staff User | While accessing UGSM function as staff user, use the sort function to sort the data by category using a non-informative column as target (such as the column with edit student button) | UGSM data should not be sorted and rearranged |  |  |
| F- UGSM -4 | UGSM List Search – informative columns | Informative UGSM data columns can be searched via search function | Staff User | Use search bar to type for graduate student name “Test Tester”.  Test should be repeated with data from all informative columns. | Graduate student with name “Test Tester” should appear.  UGSM with data searched should appear if graduate student is present. |  |  |
| F- UGSM -5 | UGSM List Search – Non-informative columns | Non-informative UGSM data search functions correctly by not searching non-informative columns | Staff User | Use search bar to type for any data found in non-informative columns. | No graduate students should appear if data searched is only found in the non-informative columns. |  |  |
| F- UGSM -6 | UGSM List – Add new prospective student | Staff user can add new graduate students | Staff User | Use the add new graduate student button located on the page in the UGSM portal | User should be redirected to a new page, like the application form and donor add page, but for graduate students |  |  |
| F- UGSM -7 | UGSM List – Add page exit with data and without changes | Exiting the UGSM add page does not change database data | Staff User | Exit the *add graduate student* page with “Name” in the first name field | No new graduate students with the first name “Name” should appear in the database |  |  |
| F- UGSM -8 | UGSM List – Add page exit empty without changes | Exiting the UGSM add page does not change database data | Staff User | Exit the *add graduate student* page while leaving fields empty | No new graduate students have been added to the database |  |  |
| F- UGSM -9 | UGSM List – Add student incomplete | Error/field checking when adding new graduate students | Staff User | While accessing *add graduate student function*, attempt to add a new graduate student with fields empty or within incorrect input types | Error/data verification message should appear, no new graduate students added |  |  |
| F- UGSM -10 | UGSM List – Add page success | Successful addition of a new graduate student to database | Staff User | While accessing the *add graduate student page*, fill all fields with correct input and confirm adding a new graduate student. Check that student is added back in the UGSM list. | A new prospective student with the details entered should be present in the database and visual graduate student list. |  |  |
| F-UPSM-11 | UGSM List – Delete graduate student redirection | Deleting a graduate student in the database | Staff User | While accessing the *graduate student list*, attempt to delete a student using the delete control | User should be redirected to a confirmation page |  |  |
| F- UGSM -12 | UGSM List – Delete graduate details | Deleting a graduate student in the database | Staff User | While accessing the *graduate student list*, attempt to delete a student using the delete control, check the student details in the list and on confirmation page | Redirected page should display the details of the student selected for deletion |  |  |
| F- UGSM -13 | UGSM List -Delete cancel | Deleting a graduate student in the database | Staff User | While deleting a graduate student, cancel the deletion process using the cancel button | Graduate student selected should not be deleted, they should still be present in database with no details changed |  |  |
| F- UGSM -14 | UGSM List – Delete Complete | Deleting a graduate student in the database | Staff User | While deleting a graduate student, confirm delete on redirected page. | Student deleted should disappear from database, user should be redirected to back to the graduate student portal and list. |  |  |
| F- UGSM -15 | UGSM List – Edit Redirect | Editing graduate student details in the database | Staff User | While accessing the *graduate student management,* use edit function to edit a student’s details | User should be redirected to a new page to edit graduate student details |  |  |
| F- UGSM -16 | UGSM List – Edit Correct information displayed | Editing graduate student details in the database | Staff User | While on the graduate student details editing page, check if the student details here match the details in the graduate list | The details should be identical |  |  |
| F- UGSM -17 | UGSM List – Edit information editable | Editing graduate student details in the database | Staff User | While in the graduate student details editing page, check that details are in editable fields | The details should be freely editable |  |  |
| F- UGSM -18 | UGSM List – Non save edited info | Editing graduate student details in the database | Staff User | While in graduate student details editing page, change some details and exit without saving and check if any details are changed in the graduate student list | No change in graduate student details |  |  |
| F- UGSM -19 | PSM List – Save edited info | Editing graduate student details in the database | Staff User | While in graduate student details editing page, change some details and exit while saving. Check that student details have changed successfully. | Graduate student details should change in the graduate student list. |  |  |
| F- UGSM -20 | UGSM List – Update after edit | Editing graduate student details in the database/synchronisation of changes | Staff User | While in graduate student list page, edit a graduate student and save, check that student details have changed in the list. Perform this concurrently with another user. | Graduate student details should update every time someone edits and saves details. |  |  |

## 3.2 – Functionality Tests – Integration Tests

This section will contain more functionality tests set on the integration level, where we check and test the interactions between components. These tests should be performed after the unit tests are completed and successful. As mentioned in section 3.1, tests that class as both integration and unit will be listed here as their identification number to avoid double descriptions of those tests.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Expected Result | Success/Fail | Result/Feedback |
| F-I-1 | Application Form New Student added | Successful application form submission | Staff User/Prospective Student and Staff User | Access application form as staff member or prospective student. Fill form and submit successfully. Then log in as staff user and check that new prospective student is added to the list. | New prospective student should be in the prospective student list. |  |  |
| F-I-2 | Application Form New Student cancelled | Cancelled application form submission | Staff User/Prospective Student and Staff User | Access application form as staff member or prospective student. Fill form and cancel submission. Check database in prospective students section as staff member. | New prospective student should not present in the database. |  |  |
| F-I-3 | Application Form New Student error, not added | Unsuccessful application form submission | Staff User/Prospective Student and Staff User | Access application form as staff member or prospective student. Fill form with non-appropriate data and submit and confirm submission. Check database in prospective students section as staff member. | New prospective student should not present in the database. Error messages or dialog. should have appeared and prevented the submission. |  |  |
| F-I-4 |  |  |  |  |  |  |  |

## 3.3 – Functionality Tests – System Tests

Our system tests aim to confirm if our application fulfils the functional requirements set out in the software requirements and design document. All system-level tests are covered in sections 3.4, 3.5 and 3.6, as usability, performance and scalability tests.

The difference between these tests and the integration tests is that we are testing the system as a whole with no knowledge of inner workings (black box testing) whereas in the integration tests, we are only checking the interface between two or more components and can choose between a white box or black box approach.

## 3.4 – Usability Tests

Our usability tests will consist of lists of tasks a tester must complete. The supervisor or the tester themselves should record how long it takes to complete the task list and give any feedback on the user interface or action sequences for the developers to improve the program.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | User Type | Instructions | Time taken | Result/Feedback |
| U-1 | Use application form | Usability of Application Form | Prospective Student | Navigate to website  Find Prospective Student Application form  Fill in details and submit form |  |  |
| U-2 | Approve Prospective Student | Usability of Prospective Student Function | Staff User | Navigate to website  Log in as staff user  Access Prospective Student Portal  Search for prospective student “Test Tester”  Change student name to “Test Complete”  Save prospective student details  Approve prospective student “Test Complete” as current student  Access current student database and search for “Test Complete”.  Exit and log out of user |  |  |
| U-3 | Graduate Student | Usability of Graduate Student Section | Staff User | Navigate to website  Log in as staff user  Access Graduate Student Portal  Search for Graduate student “Test Tester”  Change student name to “Test Complete”  Save graduate student details  Search for “Test Complete” using search tool or sort.  Exit and log out of user |  |  |
| U-4 | Donor Management | Usability of Donor Management Section | Staff User | Navigate to website  Log in as staff user  Access Donor Management Portal  Search for Donor “Don Donor”  Change donor name to “Donor Don”  Save details  Search for “Test Complete” using search tool or sort  Exit and log out of user |  |  |
| U-5 | Donor Login | Usability of Donor Login | Donor User | Access website  Log in using given donor login details  Access assigned students list  Open one student profile  Access change password function  Change password and log out.  Log back in  Log out |  |  |
| U-6 | Mass Delete – Prospective Students | Usability of deleting multiple data instances | Staff User | Access website  Log in as staff user  Access prospective student management portal  Delete 5 prospective students in the database  Log out |  |  |
| U-7 | Mass Delete – Graduate Students | Usability of deleting multiple data instances | Staff User | Access website  Log in as staff user  Access graduate student management portal  Delete 5 students in the database  Log out |  |  |
| U-8 | Mass Delete – Donors | Usability of deleting multiple data instances | Staff User | Access website  Log in as staff user  Access donor management portal  Delete 5 donor profiles in the database  Log out |  |  |
| U-9 | Mass Modification – Prospective Students | Usability of modifying multiple data instances | Staff User | Access website  Log in as staff user  Access prospective student portal  Edit 5 students and change their first and last names to “1”  Log out |  |  |
| U-10 | Mass Modification – graduate students | Usability of modifying multiple data instances | Staff User | Access website  Log in as staff user  Access graduate student portal  Edit 5 students and change their first and last names to “1”  Log out |  |  |
| U-11 | Mass Modification – Donors | Usability of modifying multiple data instances | Staff User | Access website  Log in as staff user  Access donor management portal  Edit 5 donors and change their first and last names to “1”  Log out |  |  |
| U-12 | Assign Students | Usability of modifying data instances | Staff User | Access website  Log in as staff user  Access donor management portal  Edit 5 donors, reassign a new current student to the donor  Log out |  |  |

## 3.5 – Performance Tests

Our performance tests consist of scenarios that user types will find themselves in. The time taken for the application to respond must be recorded. Other measures we could include are database operations, amount of data transferred and hardware strain/usage.

Each test should be performed at least 100 times using automated software to get a fairly accurate result.

Each test should achieve a minimum level of performance, having its task be completed within a set timeframe 95% of the time. (denoted as <time>/95%). The required performance can be set on a case-by-case basis but defaults to 2s/95% as a maximum.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Name | Purpose/Task | Required  Performance | User Type | Instructions | Average Time taken | Feedback/Notes |
| P-1 | Application Form Load | Load Time – Application Form | 2s/95% | Prospective Student/Staff User | Navigate to application form, monitor time taken to load |  |  |
| P-2 | Application Form submit | Submit Time – Application Form | 2s/95% | Prospective Student/Staff User | Navigate to application form, submit form with correct, appropriate data. Monitor response time |  |  |
| P-3 | Login Timer | Load Time – Application Form | 2s/95% | Staff User | Navigate to website, monitor time taken to login and reach staff portal |  |  |
| P-4 | Data Load Timer – Prospective Students | Load Time – prospective student data | 2s/95% | Staff User | Navigate to prospective student data portal, record data load time |  |  |
| P-5 | Data Load Timer – Current Students | Load Time – current students | 2s/95% | Staff User | Navigate to current student data portal, record data load time |  |  |
| P-6 | Data Load Timer – Graduate Students | Load Time – graduate students | 2s/95% | Staff User | Navigate to graduate student data portal, record data load time |  |  |
| P-7 | Data Sort Timer – Prospective Student | Data Sorting by Informative Category – Prospective Student | 2s/95% | Staff User | Navigate to prospective student portal, record sort time |  |  |
| P-8 | Data Sort Timer – Current Student | Data Sorting by Informative Category – Current Student | 2s/95% | Staff User | Navigate to current student portal, record sort time |  |  |
| P-9 | Data Sort Timer – Graduate Student | Data Sorting by Informative Category – graduate student | 2s/95% | Staff User | Navigate to graduate student portal, record sort time |  |  |
| P-10 | Modify Student Data Update Time - Prospective | Modify Data Update – prospective student | 2s/95% | Staff User | Navigate to prospective student portal, Modify first and last name of a student to 1, record update/change time of student in database |  |  |
| P-11 | Modify Student Data Update Time - Graduate | Modify Data Update – graduate student | 2s/95% | Staff User | Navigate to graduate student portal, Modify first and last name of a student to 1, record update/change time of student in database |  |  |
| P-12 | Modify Student Data Update Time - Donor | Modify Data Update Category – Donor | 2s/95% | Staff User | Navigate to donor management portal, Modify first and last name of a donor to 1, record update/change time of donor in database |  |  |
| P-13 | Delete Records – Prospective | Delete user database update speed | 2s/95% | Staff User | Navigate to prospective student portal, Select and delete a student, record update/change time of student in database |  |  |
| P-14 | Delete Records – Graduate | Delete user database update speed | 2s/95% | Staff User | Navigate to graduate student portal, Select and delete a student, record update/change time of student in database |  |  |
| P-15 | Delete Records – Donor | Delete user database update speed | 2s/95% | Staff User | Navigate to donor management portal, Select and delete a donor, record update/change time of student in database |  |  |
| P-16 | Log Out Timer | Log out/identity response time | 2/95% | Donor/Staff User | Log in as either a donor user or staff user.  Log out.  Record time taken to successfully be logged out |  |  |
| P-17 | Donor Student Assignment | Assignment of Donors | 2/95% | Staff User | Log in as staff user.  Access donor portal.  Edit a donor and reassign new student profiles to the donor. Check database update time. |  |  |
| P-18 | Donor Donation Timer | Donation Request | 2/95% | Donor User | Log in as donor user.  Access donor portal.  Donate money using donor microservice.  Check response time. |  |  |

## 3.6 – Scalability Tests

Our scalability tests will follow the same track as the performance tests, being very similar except that we will artificially increase the load on our program using dedicated testing software. The same parameters will be checked as in the performance tests.

The table below will contain the tests and scenarios from the 3.5 – Performance Test section, but will be provided a different structure that allows for stress testing at different load levels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Required  Performance | Average Time taken (Load level 10 users) | Average Time taken (Load level 100 users) | Average Time taken (Load level 1000 users) | Feedback/Notes |
| P-1 | 2s/95% |  |  |  |  |
| P-2 | 2s/95% |  |  |  |  |
| P-3 | 2s/95% |  |  |  |  |
| P-4 | 2s/95% |  |  |  |  |
| P-5 | 2s/95% |  |  |  |  |
| P-6 | 2s/95% |  |  |  |  |
| P-7 | 2s/95% |  |  |  |  |
| P-8 | 2s/95% |  |  |  |  |
| P-9 | 2s/95% |  |  |  |  |
| P-10 | 2s/95% |  |  |  |  |
| P-11 | 2s/95% |  |  |  |  |
| P-12 | 2s/95% |  |  |  |  |
| P-13 | 2s/95% |  |  |  |  |
| P-14 | 2s/95% |  |  |  |  |
| P-15 | 2s/95% |  |  |  |  |
| P-16 | 2s/95% |  |  |  |  |
| P-17 | 2s/95% |  |  |  |  |
| P-18 | 2s/95% |  |  |  |  |

# – Meeting Minutes

**Meeting 1: Meeting start 9:01am**

2 minute project summary and update of of the project

- Application form is complete

- Donor login functional

- Staff areas not accessible to donors, vice versa

- 2 roles available for staff, normal and administrator

- database is connected to application

Possible additional project available to the group after the trimester is complete

Design document seems fine

Team will need to work closely with other Medic2Medic team after the next sprint, as student database becomes linked

Main focus is the infrastructure section of the program

Project is split into multiple sections handled by each person in the team.

Text plan is next deliverable - how do we plan to achieve this?

Ideally automated testing, but we may have to achieve this manually

Don't forget to take minutes for the meetings!

**Meeting Finish - 9:11am**

**Meeting 2: minutes start 9:03am**

Basic project update, little progress made

Group is assigned to separate components

Currently dealing with database issues

Design document handed in

Examining test plan options

Coordination with other group beginning in earnest

**Meeting 2 end: 9:12am – Note, only one member attended**

**Meeting 3: minutes start 9:02am**

basic project update

Group split into test plan section and programming section

Main section will be completed soon

Looking into Azure storage?

Update on video presentation changed from live presentation

Requested a 10-15 minute video of presentation of project

End of exam week, video is due.

Both teams working on Medic2Medic project can submit a single video (can be a bit longer due to double project).

Combine images and video with voice, don't make the video boring.

Group is expected to submit a short document alongside the video.

4 pages or so, document should contain details about the application?

Design document marks released today.

Test plan needs to test functionality and optionally usability, scalability and performance

**Meeting 3 end, 9:11am**