Manage.Mentorship Test Plan

Jaisal Friedman, Daniel Watson, Gabriel Garcia
Software Engineering

Table of Contents

1 Introduction

This test plan describes the appropriate strategies, processes, workflows and methodologies used to plan, organize, execute and manage testing of the Hospital Management software system.

1.1 Scope

The scope of the test plan is to evaluate the in-scope metrics listed below for the Manage.Mentorship System.

1.1.1 In Scope

The Manage.Mentorship System *Test Plan* defines the unit, integration, regression, and validation, and system testing approach. The test scope includes the following:

- Perform 4 use case validation tests
  - Upload Data
  - Login/Logout all Users
  - Display Dashboard
  - Recommend Matches
- Devise overall test strategy including:
  - Role Assignments
  - Objective of Testing
  - Unit, Integration, Validation, and System test details
  - Test Schedule
  - Resources

1.1.2 Out of Scope

The following are considered out of scope for the Manage.Mentorship test plan and testing scope:

- End-to-end testing and testing of interfaces of all systems that interact with the Manage.Mentorship System.
- Functional requirements testing for systems outside Manage.Mentorship System.
- Validation tests for use cases outside of the 4 use cases provided in the in-scope section.

1.2 Objective

The objective of the test plan is to ensure that the system meets the full requirements, including non-functional requirements and satisfies the use case scenarios and maintains the quality of the product.

Specifically, the objective is to thoroughly validate the 4 use cases functions listed above - Upload Data, Login/Logout Users, Display Dashboard, and Recommend Matches.
.

1.3 Roles and Responsibilities

Roles and responsibilities are listed below and are pertinent for the scheduling and role assignment of certain tests. For the goals of this project: Daniel Watson is the lead developer. Gabriel Garcia is the project manager. Jaisal Friedman is both a developer and project manager.

1.3.1 Developers

Developer undertakes software or solution development activities. Responsible for the following:

- Developing the system/application
- Developing use cases and requirements in collaboration with the Adopters
- Conducting unit, system, regression and integration testing
- Developing automated build testing for stable software delivery
- Supporting user validation testing errors

1.3.2 Project Manager

- Conducting validation, regression, and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts and reporting test results
- Liaising with End Users to conduct alpha testing and facilitate beta testing

1.3.3 Users

Mentorship Network board members will undertake formal adoption, testing, validation, and application of products or solutions developed by developers and project guidelines submitted by project managers. Two of the members of the team are also users as members of the Mentorship Network board. Users are responsible for providing feedback for:

- Contributing to use case, requirement development through review
- Contribute to validation and system test specifications through feedback
- Participate in alpha testing and administer beta testing.

1.4 Major Constraints

Test cases for acceptance and system tests will be limited to users feedback and specification. Project managers will be responsible for obtaining these pieces of feedback. This could be a major testing constraint if there is not enough compliance.
Time is a major constraint as the feasibility study yielded a 5.5 month development cycle. For this reason, the team has decided to rely on an agile development cycle with rollouts of features. The system will remain in beta mode by the project deadline.

2 Testing Strategy

This section provides the overall test plan for unit, integration, acceptance, system, and tests. This section also details the automatic build and regression test systems that will be in place during development. The development of new features will take place on local branches. Once the developer is finished with the feature, he or she will run the regression test and submit a pull request. The branch with the new feature will be code reviewed by another developer. Once this is complete and all integration testing is passed, the branch will be merged with master. The master branch will contain the final validated and system tested code. The master branch will be delivered to the end user.

2.1 Unit Test

Unit Testing is conducted by the developers during the development process to ensure that proper functionality has been achieved by each developer both during coding and in preparation for integration testing. These tests must be conducted in the development environment. Since the system is a web-based application, unit testing will be divided between the front and back-end.

The following are the example areas of the project must be unit-tested and signed-off before being passed on to integration and regression testing:

- Firebase Database Procedures
- .jsx, .py, and other executables
- Each use case feature must include 2-3 unit tests, defined by developer and overseen by project manager. These will follow: expected execution path, alternate path with empty system, alternate path with overloaded system.

2.1.1 Selection Criteria

Unit test selection is left to the discretion of the developer. Developers are strongly urged to devise unit tests for all new features and introduction of source code. The tests will often be “white-box” tests. Thus the developer will ensure that all loop conditions have been checked, etc.

2.1.2 Standard Unit Tests

The following standard unit tests are the bare minimum for each new feature a developer roles out. Developers are responsible for providing these unit tests in code reviews, before integration happens.

- Interfacing to unit test
- Local data structures test
- Boundary conditions test
- Independent path test
- Error handling test

Front-End Unit Testing

- Conducted using the Mocha.js library (stub & driver library)
- Unfortunately, this will not be implemented in time for the current project proposal timeline.

Back-End Unit Testing

- Conducted using unittest python library (stub & driver library)
- Unfortunately, this will not be implemented in time for the current project proposal timeline.

2.2 Integration Test

The integration test will be done in 2 parts: a regression test with the newest feature added and a bottom-up test beginning with the added feature. The second part will only be executed if the first regression test is passed. Otherwise the developer will need to fix the regression bugs first. The integration tests will be conducted by pairs of developers during a code review of new features or code changes. These will be overseen and signed-off on by the project manager.

2.1.1 Testing Environment

Integration tests will be run as part of the testing environment thus ensuring that all new features are automatically tested in a breakable environment. Upon passing of the integration tests, the new changes will be added to the acceptance environment.

2.1.2 Regression Test

Bug Regression is a central tenant in the integration test phase. All bugs that are resolved will be regressed before the issue is closed. Furthermore, all new features or code changes will be regressed before they pass the code review and integration testing phase.

2.3 Validation Test

The validation tests will be conducted in accordance with the users provided feedback. There will be both alpha and beta testing as well to ensure that patient's feedback is also included. These will be conducted upon the rollout of new features that have passed the integration tests. New validation requirements will need to be provided for new features. This will be owned and managed by the project managers. The validation tests will take place in the feature branch environment.

2.3.1 Alpha & Beta Testing

In alpha testing the users are provided access to the system to confirm certain features are correctly working. In beta testing, the system access is released to the admin in the mentorship network to test on the board members. The feedback is then used to iterate.

2.3.1.1 Alpha Tests

Alpha testing will occur for all new features including Upload Data, Login/Logout Users, Display Dashboard, and Recommend Matches. The alpha tests will occur iteratively, every 2 weeks after new bugs have been fixed. Refer to the schedule for more clarity on how this will work. Alpha testing will take in the final Sprint of the project agile cycle.

2.3.1.2 Beta Test

Beta testing will occur with the final release of the newly added features in each iterative process. Beta testing will occur with all four features: Upload Data, Login/Logout Users, Display Dashboard, and Recommend Matches.at one time in a productionized product delivered to the end user to test with the patients. For the scope of this project, no beta testing will take place by the project deadline.

2.3.2 Requirements

The following specific requirements will need to be tested through the 7 validation tests below:

- Upload Data
    - Test Case 1.1
    - Test Case 1.2
- Login/Logout Users
    - Test Case 2.1
    - Test Case 2.2
    - Test Case 2.3
- Display Dashboard
    - Test Case 3.1

    - Test Case 3.2
- Recommend Matches
    - Test Case 4.1
    - Test Case 4.2

These are black-box tests overseen by the project manager. They are first alpha tests and once they are packaged, merged, and finalized in acceptance, beta testing occurs at the end user site.

2.3.3 Objective & Pass/Fail Criteria

Upload Data

- Test Case 1.1
    -
    - **Pass:**.
    - **Fail:**
- Test Case 1.2
    -
    - **Pass:**
    - **Fail:**

Login/Logout

- Test Case 2.1
    - **Pass:**
    - **Fail:**
- Test Case 2.2
    -
    - **Pass:**
    - **Fail:**
- Test Case 2.3
    -
    - **Pass:**
    - **Fail:**

Display Dashboard

- Test Case 3.1
    -
    - **Pass:**
    - **Fail:**
- Test Case 3.2
    -
    - **Pass:**
    - **Fail:**

Recommend Matches

- Test Case 4.1
    -

    - **Pass:**
    - **Fail:**
- Test Case 4.2
    -
    - **Pass:**
    - **Fail:**

2.4 System Test

System testing will be the final step before roll-out of a new feature and will be conducted in 4-phases using a stress, recovery, security, and build tests. All tests will be automated and will be the last tests in the acceptance environment. The project managers will oversee these tests. The developers will develop automated testing procedures associated with the two-part test.

2.4.1 Recovery Test

The recovery test will force the software to fail and then test that is recovered in the correct way and in the correct amount of time. This will be developed and executed in conjunction with the developers.

2.4.2 Stress Test

The stress test will stimulate an overloaded and exhausted system through intensive system requests and database overloading. The stress test will signal to the project manager, the load that the system can handle. It will be in the hands of the project manager to decide whether this load is an acceptable value.

2.4.1 Security Test

The security test will try and break the database and system encryption and security. It will attempt to compromise sensitive patient information and pertinent system algorithms.

2.4.2 Build Test

The build test will first perform a regression test to ensure that the previous three tests did not break the previous test cases passed. Then the build test will perform a build test to ensure that the system is properly functioning and the requirements have been met by the end-user. The build test will largely test the database and system functionality together through a series of black-box tests specified by the users and project manager.

Once all four system tests pass and the project manager signs off on it, the developers will migrate the new feature/code changes into the production environment and release them to the users.

3 Test Specifications

3.1 Resources

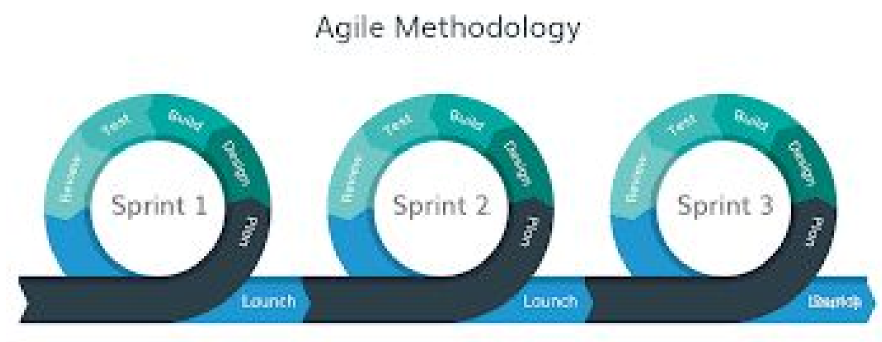
The resources deployed for testing are as follows:

- 25% of developers time
- 40% of project managers time
- 10 Laptops/Desktops for development and project manager teams.
- 5 Tablets/Laptops for Adoption Team to test
- Adoption team of End Users: 2 Administrative Staff, 1 Doctor, 1 Consulting Nurse, 1 Nurse, and 2 Receptionist.
- 1-2x per week Hospital Management (users) meeting with lead engineer and project manager
- 4 Code/Server/Database Environments: Development, Testing, Acceptance, and Production

3.2 Schedule

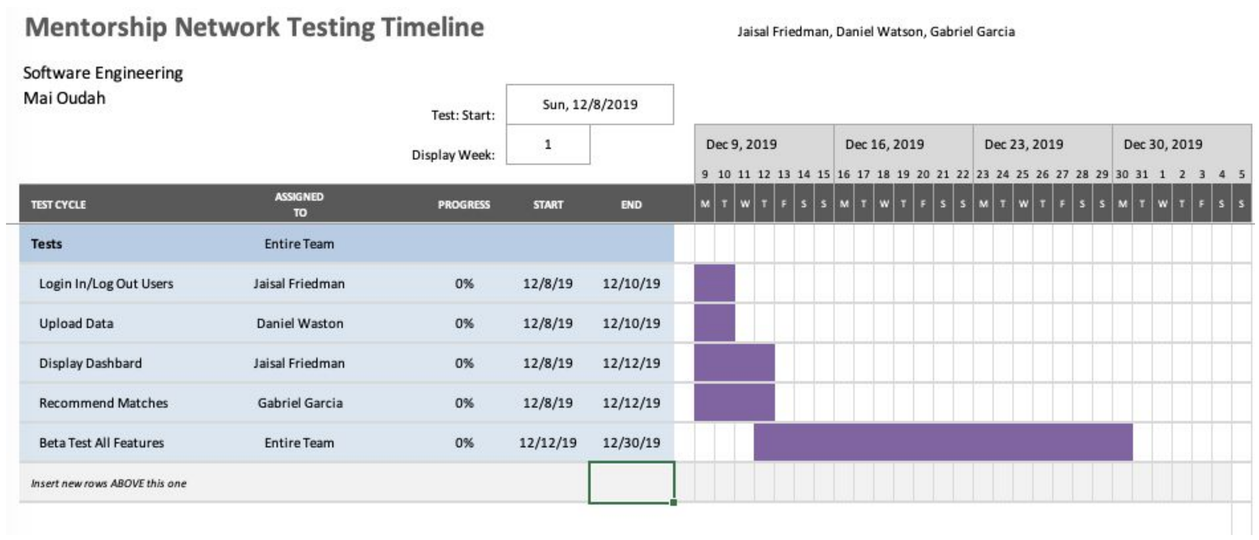
The testing schedule is displayed below and will be followed by all developers and project managers for the rollout of the 4 use cases: Upload Data, Login in/Log out, Display Dashboard, and Recommend Matches. First the alpha testing will occur in the Intermediate Versions between the Development and Validation phase. There will be several versions run as bugs are identified, thus development will continue. Finally, the beta testing will be run with all three use cases packaged into the software and shipped to the end user for use in the Hospital. The beta tests will be the same, but the information will be recorded and tested by the end users with the patients.

3.2.1 Process Model



An agile process model will be followed to roll-out the new use cases into the Manage.Mentorship system. There will be 3 Sprints conducted. These can be found on the GitHub projects at https://github.com/danielwatson6/swe-project/projects.

3.2.2 Timeline



Mentorship Network Testing Timeline

Jaisal Friedman, Daniel Watson, Gabriel Garcia

Software Engineering
Mai Oudah

Test: Start: Sun, 12/8/2019

Display Week: 1

TEST CYCLE	ASSIGNED TO	PROGRESS	START	END
Tests	Entire Team			
Login In/Log Out Users	Jaisal Friedman	0%	12/8/19	12/10/19
Upload Data	Daniel Waston	0%	12/8/19	12/10/19
Display Dashbard	Jaisal Friedman	0%	12/8/19	12/12/19
Recommend Matches	Gabriel Garcia	0%	12/8/19	12/12/19
Beta Test All Features	Entire Team	0%	12/12/19	12/30/19
Insert new rows ABOVE this one				

3.3 Deliverables

Testing will provide specific deliverables during the project. These deliverables fall into three basic categories: Documents, Test Cases / Bug Write-ups, and Reports.
Below is the list of artifacts that are process driven and should be produced during the testing lifecycle.

Deliverable
Documents
Test Plan
Test Timeline
Test Specifications
Test Case
Test Case Results
Reports
Test Results Report