Swerve Self-Start
Software Testing Plan (TP) and Test Case Specifications (TCS)

Version 1.2

Revision History

Date	Version	Description	Author
April 4th , 2018	1.0	Software Testing Plan Document	Swerve
April 5th, 2018	1.1	Added Test Case Specifications	Swerve
April 6th, 2018	1.2	Final Edit and Completed Document	Swerve

Table of Contents

Software Testing Plan (TP)

1. Introduction

The objective of this test plan is to describe the testing approach and overall framework that will be used by managers and testers to drive the testing of the SWERVE Self Start system. The objectives of the tests are to verify that the functionality works according to the specifications provided by the client and as defined in the project's scope and requirements definition.

2. Relationship to other documents

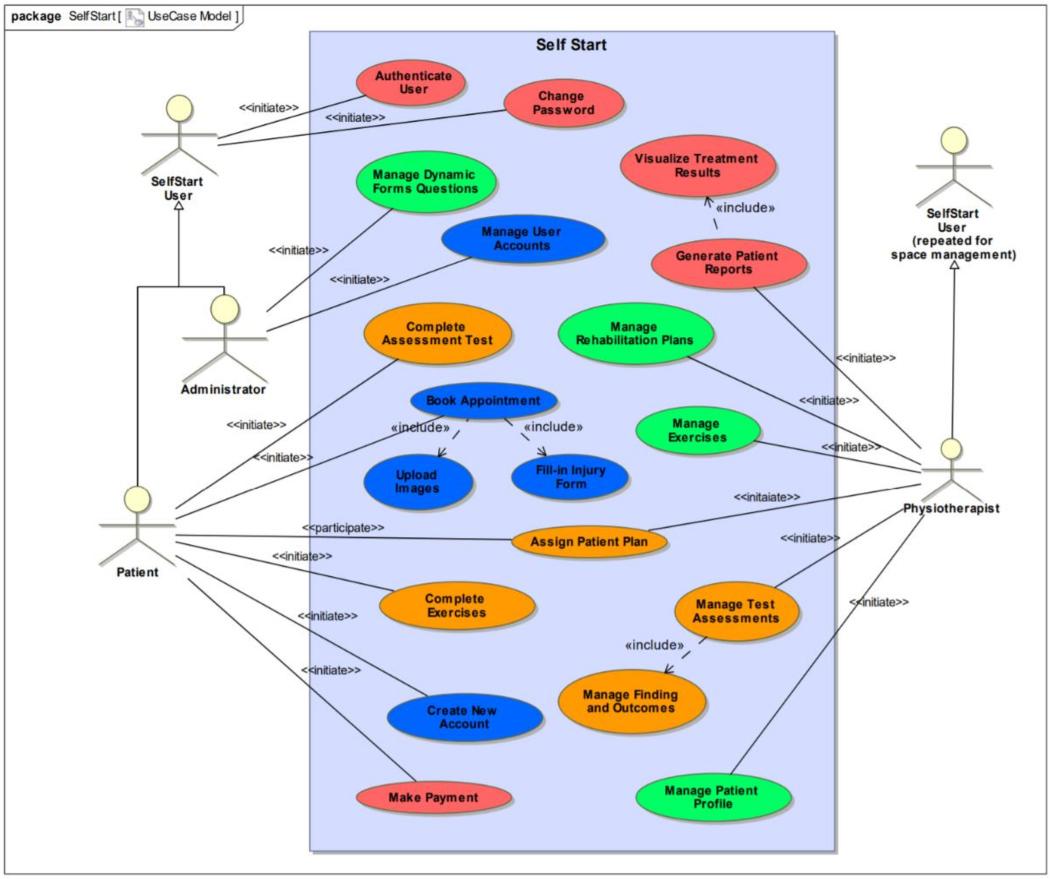
The use case diagrams in the Software Requirements Analysis Document (RAD) as well as the functional requirements developed from the RAD are used in the Grey Box unit testing.

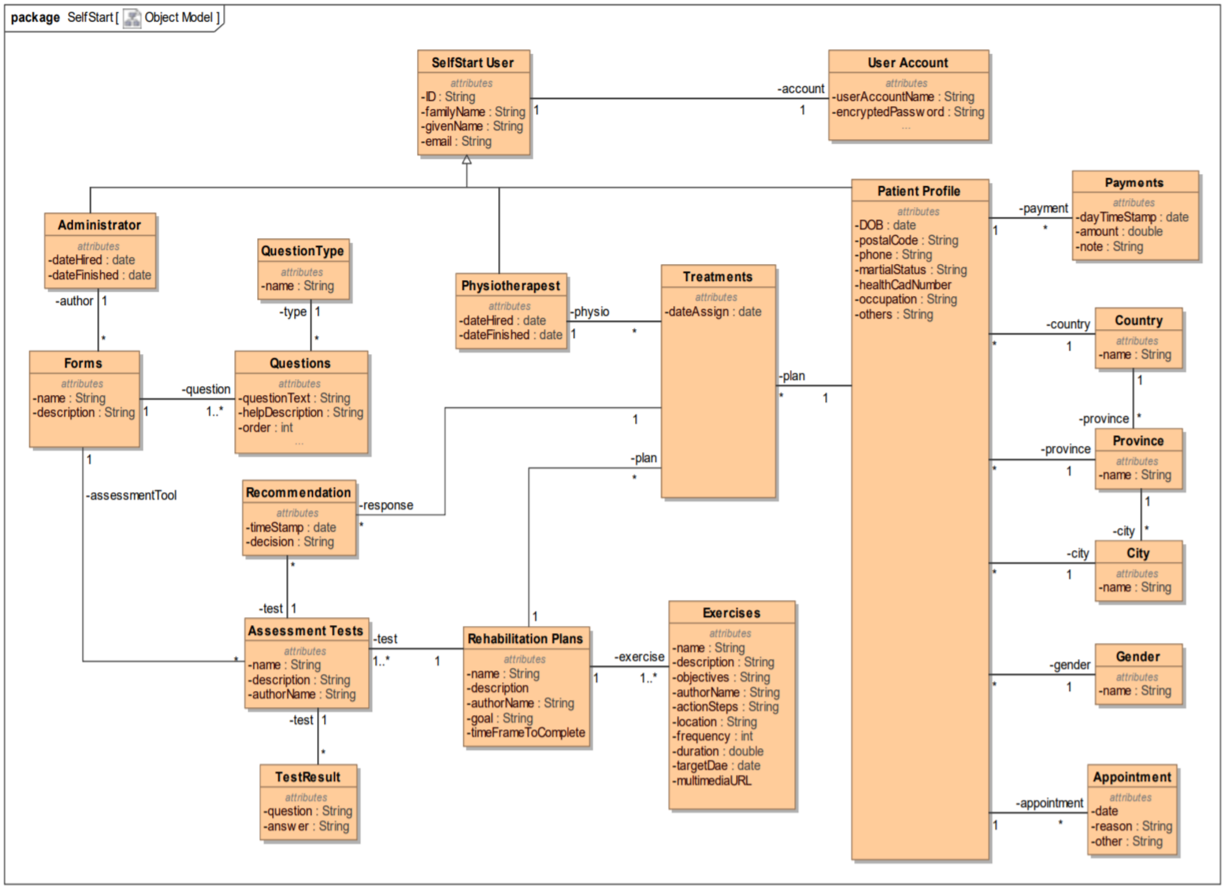
Performance testing is derived from the non-functional requirements section of the RAD.

The structured component of the Grey Box unit testing arises from the Entity Class Diagram and Object Design Document.

The integration and hardware/software environment tests are derived from the SRS and SDS, as these tests come from the overall package diagram for the system architecture.

3. System overview





4. Features to be tested/not to be tested

The features of the system that will be tested:

All components generated by Swerve's developer's.
Any components that might be outsourced by Swerve to be custom made for this project will be the responsibly for Swerve to test and validate, regardless of whether they were tested before by the vendors. Both in-house and outsourced components will be tested using the approach outlined in this document, including unit, integration and system testing.

The features of the system that will not be tested:

Off-the shelf modules and components, included those given by Dr. Ouda, that have been installed for use in our project. These will be rigorously chosen and assumed to be functional, they will however be tested upon integration, but not component wise. Additionally, the build platform and associated systems used for development, hosting and storage will not be tested for functionality. This includes, things such as MongoDB, Cloud9, Node Modules, NodeJS, Angular, PayPal Module, Ouda Authentication's Module etc. As well, no testing will be done to specifically isolate the connection protocol's performance used for the backend.

5. Pass/Fail criteria

The generic pass/fail criteria for the tests covered in this plan are as follows:

Component Pass/Fail criteria:

Tests on the various components are only given a pass when they satisfy the requirements specified by the client as stated at the in the initial vision document, and when they fulfill satisfactory non-functional benchmarks.

If a test results in a failure to meet the objectives of the design document, an issue will be logged in Swerve’s defect logging book to be later reviewed by the developers associated with that component.

Integration Pass/Fail criteria:

Tests executed on the integrated components only pass when they fulfill all the features and objectives outlined in the vision document by the client, including constraints and interfaces laid out by the system architecture design. This will also include passing tests that single out various interface environments (such as HTTP requests and database connections) to ensure functionality.

If a test leads to a failure in the program, where the objectives of the object design specifications or system architecture are not met, an issue will be logged in the Swerve’s defect logging book to be later reviewed by the developers associated with that component.

System Pass/Fail criteria:

To pass the Systems test, the project will be tested against the functional and non-functional requirements, as well as the use cases outlined at the outset.

If a test leads to a product failure in any of the objectives or features outlined by the functional and non-functional requirements or use cases, an issue will be logged in the Swerve’s defect logging book to be later reviewed by the developers associated with that component.

6. Approach

The approach that will be used in the testing process will be Unit Testing and Integration Testing. We will perform Gray Box Testing to focus on the internal structure of the program as well as the functionality of the program. This testing will allow Swerve to analyze the module, program or object levels to determine if the specific functions are working properly. Next, we will carry out Integration Testing. This tests weather a set of logically related units work together properly after unit testing is complete. Systems testing allows us to test the system as a whole and in an operating environment to verify functionality and fitness for use. It will include tests to validate usability, performance, stress, compatibility and documentation. Finally, we have Acceptance Testing, which certifies that the system satisfies the end user’s or customer’s scope and detailed requirements after systems testing is complete.

7. Suspension and resumption

As the components are being developed, unit tests will be developed alongside them to test them as well. This will also include detailed testing down to the level of the methods of the various classes and components in the build.

Before the Build Acceptance Test (BAT) can be done, the unit test suite should be run on each build. If a unit-test suite reports a failure, then it must be resolved before testing on the build can occur. When a build passes the unit testing suite then BAT's can resume.

After a build is deemed ready to test, then it will undergo BAT's, which are high level tests to measure the overall stability of the build and see if further testing is needed. However, if the BAT fails on a build, then the testing will be suspended until a new build without BAT failures is available, which will be verified by re-running the BAT. If a build passes the BAT, testing will resume.

After each major adjustment or build, regression testing will also be performed to make sure that these changes did not adversely affect the system.

If any issues raised in the defect log book require a design change to the system to be addressed, all testing will be suspended. If needed, changes to the requirements and design will be made, and subsequent test specifications will need to be revised. Tests that occurred around the change will also be run again, in addition to regression testing to make sure nothing unexpected occurs.

8. Testing materials (hardware/software requirements)

Facilities:

We will be working in TEB 244 Software Engineering Laboratory, as it has workstations preinstalled with the software we would need, there is also plenty of desk space and power outlets to make use of our Personal Computers. This is also an open concept space that allows for easy communication when developing and troubleshooting.

Hardware:

The hardware needed for this assignment is minimal, as it is a software development project, we will simply be using our PC's or MacBook Laptops.

Software:

We will be using a cloud-based development platform, Cloud 9. This may require us to upgrade to a premium account if we reach the storage or processing limits of the free option. As well, for some of the documentation we will need to use Magic Draw Software which is preinstalled into the workstations in the lab. We will be using Angular 4, whose framework will be installed from online into the Cloud9 workspace. We will also be able to easily install any additional node packages through the command line. The database will be a local Mongo data base installed on our virtual machine in Cloud9.

Other:

We will also be given a large amount of content by our client to populate our project with data so that it meets the requirements and specifications. Additionally, we will be using randomly generated data and tests to stress test our system, which will be developed by scripts or from third party services. This data however will not be shipped with the product, and only used for testing purposes.

9. Test cases

Test Specifications derived from nonfunctional requirements:

nfrs_3.3.2usability_EaseOfNavigation
nfrs_3.3.3usability_UserLoginRedirect
nfrs_3.3.6_usability_EaseofUse
nfrs_3.4.3_reliability_MeatTimeToRepair
nfrs_3.4.5_reliability_MaxAllowableDefectRate
nfrs_3.5.1_performance_SystemResponse3Seconds
nfrs_3.5.3_performance_DegredationMode
nfrs_3.6.2_supportability_AngularJSAndCross-BrowserSupport
nfrs_3.6.3_supportability_MobileSupport

Test Specifications derived from use cases in the functional requirements:

uc_VisionDocumentFeat1Obj10_PatientLog-in
uc_VisionDocumentFeat1Obj1_WelcomeVisitors
uc_VisionDocumentFeat2_ContactClinic
uc_VisionDocumentFeat3_BookAppointment
uc_VisionDocumentFeat4Obj3_FillOutForm
uc_VisionDocumentFeat5Obj3_EditForm
uc_VisionDocumentFeat6Feat28Obj8_CreateAccount
uc_VisionDocumentFeat9Feat11_CreateStandardRehabPlan
uc_VisionDocumentFeat10Feat14_CreateNewExercise
uc_VisionDocumentFeat10Feat11Obj4Obj5Obj7_CreateCustomRehabPlan
uc_VisionDocumentFeat12_AssignExercise
uc_VisionDocumentFeat13_AssignAssesmentTests
uc_VisionDocumentFeat16Obj5_AssignRehabPlan
uc_VisionDocumentFeat17_GeneratePlanSummary
uc_VisionDocumentFeat17_Print
uc_VisionDocumentFeat17_Send
uc_VisionDocumentFeat14Feat18_ViewExercises
uc_VisionDocumentFeat15Feat19Feat20_FillOutAssesment
uc_VisionDocumentFeat21Obj6_CreateProgressReport
uc_VisionDocumentFeat22_AssignFollow-up
uc_VisionDocumentFeat22_CloseCase
uc_VisionDocumentFeat23Obj9_GenerateProfileSummary
uc_VisionDocumentFeat30_ProcessPayment
uc_VisionDocumentFeat8Obj2_ManageAccountsAndProfiles
uc_VisionDocumentFeat7_SetPassword
uc_VisionDocumentFeat29_ResetPassword

Test Case Specifications

9.1 Test case specification identifier

Account-Login

9.1.1 Test items

Components under Test:
AccountLoginComponent
Features being exercised:
Authentication Service
- JSON Token verification

Admin Login Redirect
Patient Login Redirect
Email Not Empty
Password Field Not Empty

9.1.2 Input specifications

Email
Password
Submit Button
Create Account Redirect Link

9.1.3 Output specifications

Upon valid Login admin will be redirected to admin dashboard
Upon valid Login patient will be redirected to view their profile

9.1.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.1.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.1.6 Intercase dependencies

None

9.2 Test case specification identifier

Assign Patient Plan

9.2.1 Test items

Components under Test:

Profile-View

Features under Test:

Authenticate User
Retrieve Profile
Manage Rehab Plan
Save Files
Take Test
Assign Plan
Remove Plan
Send Message

9.2.2 Input specifications

The Admin chooses a specific patient and selects a plan from a list of plans and assigns it to a specific patient with the Assign to User button.

9.2.3 Output specifications

Upon submit, the plan will be assigned to the patient and the admin will be returned to the profile view page for that patients.

9.2.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.2.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.2.6 Intercase dependencies

Authenticate User
Manage Patient Profile
Manage Rehabilitation Plan

9.3 Test case specification identifier

Change Password

9.3.1 Test items

Components under Test:
Create-profile
Features under Test:
Validate User
Retrieve Profile
Edit Profile
Change Password
Change Admin Password if Admin

9.3.2 Input specifications

User enters first and last names, age, country and other related information to create a new profile if they don't have one.
If they are they are verified as admin they can then change their password by entering a new password and confirming it.
If they are verified as a non-admin account holder, they will also be given the option to change password where they enter their new password and confirm it.

9.3.3 Output specifications

After changing their password or entering their new account information, they will be rerouting to their profile page associated with their unique ID.

9.3.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.3.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.3.6 Intercase dependencies

Authenticate User

9.4 Test case specification identifier

Create Exercise

9.4.1 Test items

Components under Test:
- Exercise-Dialog
- CreateExerciseComponent

Features under Test:
- Upload files
- Exercise Service
- Exercise Form Field Validation
- Verify Admin Status
- Verify Physician Status

9.4.2 Input specifications

Exercise Name
Exercise Description
Exercise Objectives
Exercise Author
Exercise Action Steps
Exercise frequency
Exercise Duration
Exercise Target Data
Exercise Image

9.4.3 Output specifications

Upon Create Exercise Submit, exercise will appear in the exercise list and user will be return to the exercises page.
If user inputs invalid content (i.e. too long, too short or empty), they will be alerted of the error and unable to create exercise until the issue is resolved.

9.4.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.4.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.4.6 Intercase dependencies

Exercise Component

9.5 Test case specification identifier

Create New Account

9.5.1 Test items

Components under Test:

Create-Account

Features under Test:

Create Profile
Validate E-mail Format
Save New Profile
Check if Admin
Check if Patient
Accept Terms and Conditions

9.5.2 Input specifications

User Enters an email address
User Enters a password

9.5.3 Output specifications

User's email is validated, and they are presented with the terms and conditions.
If user agrees to terms and conditions they proceed to the next dialogue form to enter their information about themselves.
After the user enters information about themselves in the valid format, including their date of birth they will be notified that their account has been created successfully and they will be redirected to the login page where they can then login page, where they will now be able to login to their account.

9.5.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.5.5 Special procedural requirements

MongoDB must be running and connected
JS Server Must be Running and connected

9.5.6 Intercase dependencies

None

9.6 Test case specification identifier

Make Payment

9.6.1 Test items

Components under Test:

PaymentComponent

Features being exercised:

Pay Now
Authorize Payment
Render PayPal Container

9.6.2 Input specifications

The user selects a service from a menu with the type of assessment or consultation they want and the associated price.

9.6.3 Output specifications

Upon Buyer Authorization, the user is redirected to the checkout container as part of the PayPal API, where they are then prompted to proceed through the payment process.

9.6.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.6.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running
User Must have an account and be Logged in

9.6.6 Intercase dependencies

Create New Account

9.7 Test case specification identifier

Manage Test Assessments

9.7.1 Test items

Components under Test:
WriteTestComponent
Features being exercised:
FindTest
FindTestQuestions
SubmitTest
AssementMaker

9.7.2 Input specifications

Physical Wellness Rating Value

9.7.3 Output specifications

If test cannot be found, then alter user that error finding form to edit occurred.
If successful, display the selected questions with their names and descriptions.
On Submit Test, push the assessment form and completed test along with the data and reroute the user back

9.7.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.7.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.7.6 Intercase dependencies

Manage Finding and Outcomes

9.8 Test case specification identifier

Manage Exercises

9.8.1 Test items

Components under Test:

ExerciseComponent

Features being exercised:

Validate User
Refresh Exercise List
Delete Exercise
Edit Exercise
Display Exercise Columns in Table

9.8.2 Input specifications

User Can choose to create a new exercise, delete or edit an exercise from this page.

9.8.3 Output specifications

If the user chooses to create a new exercise, they will be redirected to the exercise dialog form, where they can create a new exercise.

If the user selected the delete icon next to an exercise listed, it will be deleted from the list and in the database.

If the user selects the edit icon next to an exercise listed, that exercise dialogue form will appear along with its information from which the user can then edit and save.

9.8.4 Environmental needs

Backend must be running
Server Must be running
Website must be live

9.8.5 Special procedural requirements

MongoDB must be running
JS Server Must be Running

9.8.6 Intercase dependencies

None

9.9 Test case specification identifier

Manage Rehabilitation Plan

9.9.1 *Test items*

Components under Test:

CreatePlanComponent
EditPlanComponent
PlanSelectComponent

Features being exercised:

Validate User
Select Exercises
Select Test
Remove Tests
Remove Exercises
Create New Rehab Plan
Update Rehab Plan
Redirect to Rehab Plan Selection Page
Display Completed Forms
Delete Plan

9.9.2 *Input specifications*

Plan Name
Plan Description
Plan Goal
Plan Time Frame
Selected Assessment Test
Selected Rehab Plan
Select Action to perform on Plan, exercises or test.

9.9.3 *Output specifications*

When a user creates a new plan, test or exercise or performs any action on it such as edit or delete, the changes are saved, and they are redirected to the rehab place Selection page.

9.9.4 *Environmental needs*

Backend must be running
Server Must be running
Website must be live

9.9.5 *Special procedural requirements*

MongoDB must be running
JS Server Must be Running

9.9.6 *Intercase dependencies*

Manage Exercises

10. Testing schedule

Staffing and Training Needs:

The testing team will be a subset of the Swerve Team, and will include 3 members, one of which will be a Test Manager, all of whom are familiar with the functionality and requirements of the project. They will require 1 day of training to self-learn and group teach the testing frameworks and methodologies specific to the project and it requirements, performance and design. Testing may also be automated in certain circumstances if required, by using Quality Assurance Testing Frameworks for Angular such as *Jasmine*, an open source development and testing framework.

Responsibilities:

The Test Manager will we in charge of developing the Test Plan, concurrently with the development of Self-Start, and acquiring the necessary hardware and software resources needed for testing. He or she will recruit suitable members from the Swerve team to join the testing team. Once the testing team has been assembled, they will jointly develop the specific test cases according to the functional and non-functional requirements of the project. One testing team member will be assigned to the component testing and will be assisted by the Test Team manager as it is a large task. The other testing team member will work on the Integration testing, as the components pass the unit testing phase. Finally, upon completion of component and Integration testing, the entire Testing Team, including the Team Manager will conduct system testing, and if any changes occur regression testing, after each testing approach. The final stage of validation will be Acceptance Testing. This is where the entire Swerve Team can certify that the system satisfies the client’s scope and detail requirements, after the completion of systems testing.

Test Schedule:

Test Phase	**Time**	**Owner**
Test Plan Creation	½ day	Test Manager
Test Cases Development	½ day	Testing Team
Component Testing	½ day	Testing Team
Integration Testing	½ day	Testing Team
System Testing	½ day	Testing Team
Regression Testing	½ day	Testing Team
Validation Testing	1 day	Testing Team

Risks and Contingencies Matrix:

Risk	Probability	Type	Contingency
Inability for Testing team to grasp new testing technology and framework	LOW	Project Risk	Change testing framework or consult experts in the framework to address issues
Desired Testing Framework is faulty, unavailable or incompatible	MED	Product Risk	Create custom testing system, or conduct unit testing manually
Planned testing schedule becomes unrealizable	MED	Project Risk	Reduce Testing Scope and prioritize critical functional and performance tests.
Testing Team Turnover	LOW	Project Risk	Quickly train a Swerve Team member who is most familiar with the work of the testing team member that will be replaced, and adjust schedule as needed.