|  |
| --- |
|  |
| Test design Specification |
|  |
|  |
|  |
|  |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

**Table of Contents**

[1 Introduction 2](#_Toc38196917)

[2 Test Objective 2](#_Toc38196918)

[3 Detailed Testing Strategy 2](#_Toc38196919)

[3.1 Unit Testing 2](#_Toc38196920)

[3.1.1 White Box Testing 2](#_Toc38196921)

[3.1.1.1 Basis Path Testing – Tree Repository Module 2](#_Toc38196922)

[3.1.2 Test Cases 3](#_Toc38196923)

[3.1.3 [TC001] Install mobile application 3](#_Toc38196924)

[3.1.4 [TC002] MA User login 3](#_Toc38196925)

[3.1.5 [TC003] MA User logout 4](#_Toc38196926)

[3.1.6 [TC004] Mobile Application Settings 5](#_Toc38196927)

[3.1.7 [TC005] Start Transmitting 6](#_Toc38196928)

[3.1.8 [TC006] Start Transmitting 6](#_Toc38196929)

[3.1.9 [TC007] Resume Transmitting 7](#_Toc38196930)

[3.1.10 [UC008] Stop Transmitting 8](#_Toc38196931)

[3.1.11 [TC009] WA User Login 8](#_Toc38196932)

[3.1.12 [TC010] WA User Logout 9](#_Toc38196933)

[3.1.13 [TC011] WA User Registration 9](#_Toc38196934)

[3.1.14 [TC012] WA View Route 10](#_Toc38196935)

[3.1.15 [TC013] WA Rename Route 11](#_Toc38196936)

[3.1.16 Black Box Testing 12](#_Toc38196937)

[3.2 Integration Testing 12](#_Toc38196938)

[3.2.1 Incremental Testing 12](#_Toc38196939)

[3.3 System Testing 13](#_Toc38196940)

[3.3.1 Function Validation Testing 14](#_Toc38196941)

[3.3.2 Performance testing 14](#_Toc38196942)

[4 Pass/Fail Criteria 14](#_Toc38196943)

[4.1 Test Log 14](#_Toc38196944)

[4.2 Shipping or Live Release 14](#_Toc38196945)

[4.2.1 Shipping/Live Release Entry Criteria 14](#_Toc38196946)

[4.2.2 Shipping/Live Release Exit Criteria 15](#_Toc38196947)

[5 References 16](#_Toc38196948)

# Introduction

This document provides the test documentation that will facilitate the technical tasks of testing including the detailed test cases for both white box and black box testing. Each test case specifies who will be performing the test, the preconditions required to execute each test case, the specific item to be tested, the input, expected output or results, and procedural steps where applicable.

# Test Objective

The objective of this document is to expand on the test plan and provide specific information needed to actually perform the necessary tests. By providing detailed test information, we hope to reduce the probability of overlooking items and improve test coverage. Testers will be able to use each test case provided in this document to move forward and begin testing. Test results will be logged in a database and a complete bug report generated for each test failure.

# Detailed Testing Strategy

## Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

### White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. The test cases that have been generated shall cause each condition to be executed at least once. To ensure this happens, we are applying Basis Path Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

//Todo

Each function of the backend is executed independently ; therefore, a program flow for each function has been derived from the code. The development team will be performing all white box testing.

#### Basis Path Testing – Tree Repository Module

Using the program flow graph for each function in our tree repository module, we were be able to determine all of the paths that will need to tested and have developed the corresponding test cases. In order to test the success of each path, return values were added to verify successful completion. Any preconditions needed to exercise a path have been included in the test case.If the expected result/output is not achieved, the test will be considered a failure and a bug report filed.

### Test Cases

### [TC001] MA User Register

|  |  |
| --- | --- |
| [ID] Name | [TC001] MA User Register |
| Summary | The User is registered to MA which is installed on a device |
| Items to be tested | Module Account: Registration |
| Users | All unregistered users |
| Pre-conditions | The User has valid email and non-taken username and MA that already installed to device. |
| Basic Course of Events | 1. The user launches the MA  2. The user clicks the sign up button and move to user register page and starts the registration process.  3.The user inputs the unique username, full name, valid email and passwords then clicks the register button.  4. The user gets a verification email and he/she must verify his/her account to be able to login the MA. |
| Input | The parameters that required in registration page which are unique username, full name, valid email and passwords. |
| Expected output | The user account is created and ready to use for login fucntion. |

### [TC002] MA User login

|  |  |
| --- | --- |
| [ID] Name | [TC002] MA User login |
| Summary | The user logs in to the MA |
| Items to be tested | Module Account: Login |
| Users | All registered users |
| Pre-conditions | The user has opened an account using the MA’s register function and verified his/her email. |
| Basic Course of Events | 1. The user launches the MA. 2. The user enters his email and password on the Login Page. 3. The user authenticates and access the MA. 4. The MA session remains open until the user explicitly logs out. The user logs in once and gains access to the MA without being prompted to log in again. |
| Alternative Paths | 1. In Step 2 the email the user enters does not exist. In this case the MA shows an error message and information. Then user can continue with the sign up function.  2. If In Step 2 the password for the username doesn’t match the system records. In this case the user is shown a message and gets an error. User can continue with forgot password function. |
| Input | The parameters that required by user on the login page which are email and password. |
| Expected Output | The user is authenticated and has access to the MA functions, the MA is an idle state. |

### [TC003] MA User forgot password

|  |  |
| --- | --- |
| [ID] Name | [TC003] MA User forgot password |
| Summary | The user reset his/her password by MA’s using forgot password function |
| Items to be tested | Module Account: Forgot password |
| Users | All registered users |
| Pre-conditions | The user has registered to MA. |
| Basic Course of Events | 1. User launches the MA 2. User clicks forgot password button on the login page. 3. User enters his email as an input to get an recover email. 4. User gets an recover email and click the link. 5. User enters his/her new password on the website opened by clicking the link. 6. User clicks the change button and so his/her password is changed. |
| Input | The parameter that required by user on the forgot password page which is his/her email. |
| Expected Output | The user password is recreated and is able to login to the MA. |

### [TC003] MA User change password

|  |  |
| --- | --- |
| [ID] Name | [TC004] MA User Change Password |
| Summary | The user changes his/her password using the change password function. |
| Items to be tested | Module Account: ChangePassword |
| Users | All users |
| Pre-Conditions | The user has an account and user signed in. |
| Basic Course of Events | 1. The user navigates to profile page and opens change password page from profile menu. 2. The user is presented a form that asks for current password, new password and new password again. 3. The user enters his/her current password , new password and new password again and clicks on “Done” button. 4. The Photonest system validates this information and changes the user’s password. |
| Alternative Paths | 1. In the 4th step the new password and new password(again) fields do not match and the system displays a message that says “New password and New Password(again) must be the same”. |
| Input | 1. Current Password 2. New Password 3. New Password(Again) |
| Expected Output | The user has got a new password. |

### [TC003] MA User logout

|  |  |
| --- | --- |
| [ID] Name | [TC005] User Logout |
| Summary | User logs out of Photonest |
| Items to be tested | Module Account: Logout |
| Users | All users |
| Pre-Conditions | User must be logged in to the system. |
| Basic Course of Events | 1. The user opens the menu on the profile page and presses on logout button. 2. The Photonest system ends the user session and redirects the user to login page. |
| Alternative Paths |  |
| Input |  |
| Expected Output | User signed out of the system and sees the login page. |

### [TC003] MA User edit profile

|  |  |
| --- | --- |
| [ID] Name | [TC006] User EditProfile |
| Summary | User changes account information. |
| Items to be tested | Module Account : EditProfile (ProfileActivity, EditProfileFragment) |
| Users | All users |
| Pre-Conditions | User must be logged in to the system. |
| Basic Course of Events | 1. The user presses the “Edit Profile” button on the profile page and opens the edit profile fragment. 2. The user enters new Username, Fullname, Website link and Bio and presses the check button. 3. The Photonest system updates the user information on the firebase. |
| Alternative Paths | 1. In step 2 the user presses the cancel button and the system closes the edit profile fragment. |
| Input | 1. New username 2. New fullname 3. New bio 4. New website link. |
| Expected Output | User’s account information in firebase has been changed. |

### [TC003] MA User create new post

|  |  |
| --- | --- |
| [ID] Name | [TC007] CreatePostTest |
| Summary | User chooses a photo from his/her android device and upload it on Photonest |
| Items to be tested | Module Post: CreatePost(PostActivity, GalleryFragment, UploadPostActivity.) |
| Users | All users |
| Pre-Conditions | User must be logged in to the system. |
| Basic Course of Events | 1. The user opens the PostActivity tab and chooses the “Gallery” tab from the bottom menu. 2. The user chooses one of the directories using the spinner on the page. 3. The user chooses a photo from gridview and press on “Next” button. 4. The user enters a caption and adds location and presses on “Post” button. 5. The Photonest system adds the new Photo to firebase as Post. |
| Alternative Paths | 1. In step 2, 3 or 4 user press the cancel button on the page. The system closes the activity. |
| Input | 1. Photo directory name. 2. Photo to be uploaded. 3. Caption. 4. Location. |
| Expected Output | The new post is uploaded to the user’s photo list in firebase. |

### [TC003] MA User create new post

|  |  |
| --- | --- |
| [ID] Name | [TC008] CreatePostTest |
| Summary | User adds a post after taking a photo |
| Items to be tested | Module Post : CreatePost(PostActivity, GalleryFragment, UploadPostActivity) |
| Users | All users |
| Pre-Conditions | User must be logged in to the system. |
| Basic Course of Events | 1. The user opens the PostActivity tab and chooses the “Camera” tab from the bottom menu. 2. The user presses the camera icon on the page. 3. The camera application of the phone is opened. 4. The user takes a photo and confirms it. 5. The user enters a caption and adds location and presses on “Post” button. 6. The Photonest system adds the new Photo to firebase as Post. |
| Alternative Paths |  |
| Input | 1. Photo to be uploaded. 2. Caption. 3. Location. |
| Expected Output | The new post is uploaded to the user’s photo list in firebase. |

### [TC003] MA User edit profile photo

|  |  |
| --- | --- |
| [ID] Name | [TC006] User EditProfilePhoto |
| Summary | User changes profile photo. |
| Items to be tested | Module Account : EditProfile (ProfileActivity, EditProfileFragment) |
| Users | All users |
| Pre-Conditions | User must be logged in to the system. |
| Basic Course of Events | 1. The user presses the “Edit Profile” button on the profile page and opens the edit profile fragment. 2. The user presses the profile photo. 3. The user’s photos on his/her device are displayed. 4. The user chooses one of them. 5. The Photonest changes the profile photo with new photo. |
| Alternative Paths | 1. In step 2 the user presses the cancel button and the system closes the edit profile fragment. |
| Input | 1. New photo |
| Expected Output | User’s profile photo in firebase has been changed. |

### [TC010] Add Comment

|  |  |
| --- | --- |
| [ID] Name | [TC010] Add Comment |
| Summary | The User adds a comment under a post. |
| Items to be tested | Module Comment: Post |
| Users | All registered users. |
| Pre-conditions | The MA has an existing post and The User has clicked on a post’s comment section. |
| Basic Course of Events | 1. The User writes his/her desired comment.  2. The User clicks the post button and initializes the add comment function.  3. The User sees his/her comment in the comment section of the selected post. |
| Input | The text input that is required in the comment section. |
| Expected output | The comment is created and added to the selected post’s comment section. |

### [TC011] Delete Post

|  |  |
| --- | --- |
| [ID] Name | [TC011] Delete Post |
| Summary | The User deletes his/her post. |
| Items to be tested | Module Post: Post |
| Users | All registered users. |
| Pre-conditions | The User has an existing post and The User has clicked on a post. |
| Basic Course of Events | 1. The User clicks on the settings menu of the selected post.  2. The User clicks on the “Delete Post” option of the post’s menu.  3. The User confirms the deletion by clicking the OK button and initializes the delete post function.  4. The User is moved back to the profile page to see that his/her post is removed. |
| Input |  |
| Expected output | The post is deleted and removed from the User’s account. |

### [TC012] Edit Post

|  |  |
| --- | --- |
| [ID] Name | [TC012] Edit Post |
| Summary | The User edits his/her post’s caption. |
| Items to be tested | Module Post: Post |
| Users | All registered users. |
| Pre-conditions | The User has an existing post and The User has clicked on a post. |
| Basic Course of Events | 1. The User clicks on the settings menu of the selected post.  2. The User clicks on the “Edit Post” option of the post’s menu.  3. The User writes his/her desired caption and clicks on the Edit button to initialize the edit post function.  4. The User is moved back to the post’s preview page to see that the caption is changed. |
| Input | The text input that is required in the editing page. |
| Expected output | The caption of the post is changed. |

### [TC003] MA User logout

|  |  |
| --- | --- |
| [ID] Name | [TC003] MA User logout |
| Summary | The user logs out of the MA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All logged in users |
| Pre-conditions | The user is currently logged in to the MA |
| Basic Course of Events | 1. The taps the logout button 2. The MA closes the user session and returns to the login screen |
| Alternative Paths | 1. In Step 1 the MA is currently either on a transmitting or paused state. In this case:   1. The MA sends a message to the WA that it is about to terminate its session 2. The MA closes the user session and returns to the login screen |
| Input |  |
| Expected Output | The session is terminated and the MA returns to the login screen |

### [TC004] Mobile Application Settings

|  |  |
| --- | --- |
| [ID] Name | [TC004] Mobile Application Settings |
| Summary | The user accesses the MA settings for viewing or modification |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The user is logged in to the MA |
| Basic Course of Events | 1. The user taps the Settings icon and app Settings screen is shown. 2. The user modifies the following values:  * Location sampling interval: Defined by an integer value and unit {seconds}. 1 < t < 3600. * Location recording interval: Defined by a type {distance, time}, integer value and unit {{m},{seconds}}. 1< d < 100. 1 < t < 3600. * Web server to report the location points to {URL}.  1. After a change is made the Save and Cancel buttons are enabled 2. The user clicks the Save or Cancel buttons and the Settings screen grays out both buttons. 3. The user clicks the Close button and goes back to the Main Screen |
| Alternative Paths | 1. In Step 2 the user does not make any changes to the settings and interacts with the Close button only. In this case the system does not enable the Save and Cancel buttons. |
| Input |  |
| Expected Output | The application settings are changed (settings) |

### [TC005] Start Transmitting

|  |  |
| --- | --- |
| [ID] Name | [TC005] Start Transmitting |
| Summary | The user turns the MA transmitting function on |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The user is logged in to the MA |
| Basic Course of Events | 1. The user taps the Start button 2. The MA verifies the GPS is enabled on the device, if it is disabled it asks the user for his authorization for enabling it 3. The MA starts transmitting location points to the web server 4. The MA receives acknowledgement that the web server is receiving information 5. The MA shows a UI indicator that it is currently transmitting 6. The Start button changes its text from ‘Start’ to ‘Pause’ |
| Alternative Paths | 1. In Step 2 the user decides not to authorize enabling the GPS. In this case the application goes back tot he main screen. |
| Post-conditions | The MA starts transmitting location points and it is set to a transmitting state |

### [TC006] Start Transmitting

|  |  |
| --- | --- |
| [ID] Name | [TC006] Pause Transmitting |
| Summary | The user pauses the transmission of location points |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The MA was transmitting to the WA |
| Basic Course of Events | 1. The user taps the ‘Pause’ button. 2. The MA pauses transmitting location 3. The ‘Pause’ button changes its text to ‘Resume’ 4. The MA shows a UI indicator that it is currently on an paused state |
| Alternative Paths | None |
| Input |  |
| Expected Output | The MA is in a paused state |

### [TC007] Resume Transmitting

|  |  |
| --- | --- |
| [ID] Name | [TC007] Resume Transmitting |
| Summary | The user resumes the transmission of location points |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The MA is in a paused state |
| Basic Course of Events | 1. The user taps the ‘Resume’ button 2. The MA resumes transmitting 3. The ‘Resume’ button changes its text to ‘Pause’ 4. The MA shows a UI indicator that it is currently on an transmitting state |
| Alternative Paths | NA |
| Input |  |
| Post-conditions | The MA resumes transmitting location points and it is in a transmitting state |

### [UC008] Stop Transmitting

|  |  |
| --- | --- |
| [ID] Name | [UC008] Stop Transmitting |
| Summary | The user turns the transmitting function off |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The mobile application is initially transmitting information to the web application |
| Basic Course of Events | 1. The user taps the Stop icon. 2. The MA sends a message to the WA that it is about to stop transmitting 3. The MA stops transmitting 4. The application shows a UI indicator that it is currently on an idle state |
| Alternative Paths | None |
| Post-conditions | The application gets set to an idle state |

### [TC009] WA User Login

|  |  |
| --- | --- |
| [ID] Name | [TC009] WA User Login |
| Summary | The user logs in to the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The user has opened an account using the WA |
| Basic Course of Events | 1. The user browses to the WA URL 2. The WA presents the user with a login screen with a login and a password fields. The password field will show asterisk characters only. 3. The user enters his username and password 4. The user authenticates and access the WA 5. The WA navigates to a ‘Home’ screen that shows a list of the user routes previously saved |
| Alternative Paths | 1. In Step 3 the password for the username doesn’t match the system records.In this case the user is shown a message and gets 4 more attempts for login in. If the attempts are unsuccessful the account is locked and an email is sent to the user to inform of the account locking |
| Post-conditions | The user is authenticated and has access to the WA functions |

### [TC010] WA User Logout

|  |  |
| --- | --- |
| [ID] Name | [TC010] WA User Logout |
| Summary | The user logs out of the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The user is currently logged in to the WA |
| Basic Course of Events | 1. The user clicks the log out function 2. The WA closes the user session and returns to the login screen |
| Alternative Paths | None |
| Post-conditions | The session is terminated and the WA returns to the login screen |

### [TC011] WA User Registration

|  |  |
| --- | --- |
| [ID] Name | [TC011] WA User Registration |
| Summary | The user creates a new account in the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | None |
| Basic Course of Events | 1. The user browses to the WA url 2. The WA shows the login screen 3. The user clicks the ‘Open new account’ link 4. The user enters his/her personal information including: name, email and password 5. The WA validates this info, including verifying that the username does not exist already and validating the email and password formats 6. The WA creates the new account and sends a confirmation email to the user 7. The WA browses to the login screen |
| Alternative Paths | 1. In Step 5 the user information returns errors after the validation. In this case:   * The WA indicates in the UI which fields contain errors so the user can fix the problems * Once the detected issues are fixed the WA continues with step 6 |
| Post-conditions | The WA creates a new account |

### [TC012] WA View Route

|  |  |
| --- | --- |
| [ID] Name | [TC012] WA View Route |
| Summary | The user views a previously saved route in the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The user is logged in to the WA |
| Basic Course of Events | 1. The ‘Home’ screen shows a list of the routes previously saved by the user 2. The user clicks on a route from the list 3. The WA retrieves the saved route data points on a list. Additionally it may or may not display these points on a map service provided by an external entity (e.g. Google Maps, Bing Maps). 4. The user clicks on the ‘Home’ link 5. The ‘Home’ screen shows a list of the saved routes |
| Alternative Paths | 1. On step 4 the user executes the the browser Back command. In this case the WA will execute step 5. |
| Post-conditions | The selected route data points are shown on a map |

### [TC013] WA Rename Route

|  |  |
| --- | --- |
| [ID] Name | [TC013] WA Rename Route |
| Summary | The user deletes a previously saved route in the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The is currently logged in to the WA and in the ‘Home’ screen |
| Basic Course of Events | 1. The ‘Home’ screen shows a list of the routes previously saved by the user 2. The user selects one or more routes from the list 3. The user clicks the ‘Delete’ button/link 4. The saved routes are removed from the WA back end storage area 5. The ‘Home’ screen shows an updated list of the saved routes |
| Alternative Paths | None |
| Post-conditions | The WA shows an updated ‘Home’ screen |

|  |  |
| --- | --- |
| [ID] Name | [UC014] WA Delete Route |
| Summary | The user deletes a previously saved route in the WA |
| Items to be tested | Module X : Login //ToDO what module is being tested |
| Users | All users |
| Pre-conditions | The is currently logged in to the WA and in the ‘Home’ screen |
| Basic Course of Events | 1. The ‘Home’ screen shows a list of the routes previously saved by the user 2. The user selects one or more routes from the list 3. The user clicks the ‘Delete’ button/link 4. The saved routes are removed from the WA back end storage area 5. The ‘Home’ screen shows an updated list of the saved routes |
| Alternative Paths | None |
| Post-conditions | The WA shows an updated ‘Home’ screen |

### Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application. The Equivalent Partitioning will be performed at both the unit test level and the system test level. Boundary Value analysis will only be done at the system test level. In considering the inputs for our equivalence testing, the following types will be used:

* Legal input values – Test values within boundaries of the specification equivalence classes. This shall be input data the program expects and is programmed to transform into usable values.
* Illegal input values – Test equivalence classes outside the boundaries of the specification. This shall be input data the program may be presented, but that will not produce any meaningful output.

The equivalence partitioning technique is a test case selection technique in which the test designer examines the input space defined for the unit under test and seeks to find sets of input that are, or should be, processed identically. Black box testing will be performed by the test team. All procedural step have been included to assist the team in executing the various tests.

## Integration Testing

### Incremental Testing

// ToDo

There are two primary modules that will need to be integrated: the MA Interface and the backend . The two components, once integrated, will form the complete Application testing . The following describes these modules as well as the steps that will need to be taken to achieve complete integration. We will be employing an incremental testing strategy to complete the integration. The integration testing will be performed by the development team.

**Module 1 - Graphic User Interface (GUI) Module**

This module provides a simple GUI where the user can perform the different actions (functions). This module will be tested separate from the backend to check if each interface (e.g. insert button) is functioning properly, and in general, to test if the mouse-event actions are working properly. The testing will be performed by writing a stub for each element in the interface.

**Module 2 – Backend Module**

The DB interface provides the storage for the data elements and implements the algorithms and associated functionality of the. This module will be tested separate from the GUI by printing out the results to the Console????. In testing this module we will follow the incremental testing method i.e. testing one function first and then keep adding additional function and test it again until all the required functions are tested.

**Application**

|  |  |
| --- | --- |
| **GUI Function** | **DB Interface Function(s)** |
| Load Button | Read  Search  Insert |
| Store Button | Store  Write |
|  |  |
|  |  |
|  |  |
| Insert Button | Insert  Search |
| Delete Button | Delete  Search |
| Search Button | Search |

## System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

### Function Validation Testing

The integrated Application will be tested based on the requirements to ensure that we built the right application. In doing this test, we will try to find the errors in the inputs and outputs, that is, we will test each function to ensure that it properly implements the

| **Function** | **Expected Behavior** |
| --- | --- |
| Load | see Software Program Specification for detailed explanation |
| Store | see Software Program Specification for detailed explanation |
| Insert | see Software Program Specification for detailed explanation |
| Delete | see Software Program Specification for detailed explanation |
| Search | see Software Program Specification for detailed explanation |
|  |  |
|  |  |
|  |  |

### Performance testing

This test will be conducted to evaluate the fulfillment of a system with specified performance requirements. It will be done using black-box testing method. And this will be performed by:

* Storing the maximum data in the file and trying to insert, and observe how the application will perform when it is out of boundary.
* Deleting data and check if it follows the right sorting algorithm to sort the resulting data or output.
* Trying to store new data and check if it over writes the existing once.
* Trying to load the data while they are already loaded

# Pass/Fail Criteria

This section will include the master list of both white box and black box tests which will be used to track the progress of the testing. A test will be considered a failure if the expected result or output is not achieved. A bug report will be filled out for each failure and will be submitted to the development team for correction. After the bug has been fixed, the test case will be repeated.

## Test Log

## Shipping or Live Release

### Shipping/Live Release Entry Criteria

### Shipping/Live Release Exit Criteria

The Shipping/Live Release stage is when the product is ready for general availability to the public and the user documentation is final. The product must fully satisfy its release specifications and the user documentation must adequately describe the product’s functionality. Both should be ready for use by the end user.

* QA tests the final product version to verify that the product to be released to the general public is of the utmost quality and satisfies original design specifications.
* The product must receive approval from the product team.
* QA and Development must prepare Release Notes.

The product is now ready to ship or published to production environment.

HATALAR

\*Hiyerarşi (numaralandırma) yanlış. Test 3.1.2 ise ilk test case 3.1.2.1 olur 3.1.3 değil!

\*Örnekler silinmemiş.

\*Başlıklarda her kelimebüyük harfle başlamıyor (test classlar)

\*Bazı test classlar numaralandırılmamış: TC0003🡪 hepsi için.

\*SDD’ye uygun olamyan Module isimleri 🡪 CreatePost

# References