|  |
| --- |
| Photonest  Prepared for  SE302 Software Engineering |
| Test design Specification |
| Version 1.0 |
| 15.05.2020  Dilara Ünbay  Nursena Karakulah  Feyzullah Berkay Danış  Özay Ezerceli |

Contents

[1 Introduction 2](#_Toc40476060)

[2 Test Objective 2](#_Toc40476061)

[3 Detailed Testing Strategy 2](#_Toc40476062)

[3.1 Unit Testing 2](#_Toc40476063)

[3.1.1 White Box Testing 2](#_Toc40476064)

[3.2 Test Cases 3](#_Toc40476065)

[3.2.1 [TC001] User Register 3](#_Toc40476066)

[3.2.2 [TC002] User login 4](#_Toc40476067)

[3.2.3 [TC003] User forgot password 5](#_Toc40476068)

[3.2.4 [TC004] User change password 6](#_Toc40476069)

[3.2.5 [TC005] User logout 6](#_Toc40476070)

[3.2.6 [TC006] User edit profile 7](#_Toc40476071)

[3.2.7 [TC007] User create new post 8](#_Toc40476072)

[3.2.8 [TC008] User create new post 9](#_Toc40476073)

[3.2.9 [TC009] User edit profile photo 10](#_Toc40476074)

[3.2.10 [TC010] Add Comment 10](#_Toc40476075)

[3.2.11 [TC011] Delete Post 11](#_Toc40476076)

[3.2.12 [TC012] Edit Post 11](#_Toc40476077)

[3.2.13 [TC0013] Like Post 12](#_Toc40476078)

[3.2.14 [TC0014] Unlike Post 13](#_Toc40476079)

[3.2.15 Black Box Testing 14](#_Toc40476080)

[3.3 Integration Testing 14](#_Toc40476081)

[3.3.1 Big Bang Approach 14](#_Toc40476082)

[References 16](#_Toc40476083)

# 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. But we implemented instrumented unit test classes so it means that we should run our test on real devices so that UI could be tested. 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.

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] 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] 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] 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. |

### [TC004] 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. |

### [TC005] 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. |

### [TC006] 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. |

### [TC007] 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. |

### [TC008] 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, CameraFragment, 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. |

### [TC009] User edit profile photo

|  |  |
| --- | --- |
| [ID] Name | [TC009] 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. |

### [TC0013] Like Post

|  |  |
| --- | --- |
| [ID] Name | [TC0013] Like Post |
| Summary | User likes a post whose is her/him or someone else |
| Items to be tested | Module Post: Like Post |
| Users | All registered users |
| Pre-conditions | The user has clicked on like post button on a post. |
| Basic Course of Events | 1. The user clicks on like post button on main page, on her/him profile page or other users’ profile page. 2. Post liked by the user. Number of like of that post is increased by one. 3. User is added into the list of users who liked that post. 4. The like post button’s shape changes, return unlike post button. |
| Input |  |
| Expected Output | The post liked by the user and like post button’s shape returns unlike post button. |

### [TC0014] Unlike Post

|  |  |
| --- | --- |
| [ID] Name | [TC0014] Unlike post |
| Summary | The user clicks on unlike post button of the post which was liked by the user. |
| Items to be tested | Module Post: Unlike post |
| Users | All registered users |
| Pre-conditions | The user has clicked on unlike post button on a post. |
| Basic Course of Events | 1. The user clicks on unlike post button on main page, on her/him profile page or other users’ profile page. 2. Post unliked by the user. Number of like of that post is decreased by one. 3. User is removed from the list of users who liked that post. 4. The unlike post button’s shape changes, return like post button. |
| Input |  |
| Expected Output | The post unliked by the user and unlike post button’s shape returns like post button. |

### 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

### Big Bang Approach

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

There is one module, in this module we will test Graphic User Interface (GUI) and backend function of the system.

In this application program, the integration testing will be performed by development team. We will be employing a Big Bang approach strategy to complete integration of whole system. The Database interface provides the storage of the data, implementation of GUI functions.

After implementation of the interface and backend function of whole system, their functionality will be tested together. To see whether register button is functioning properly, creating a new account, and by the Register button, the button is functioning or not will be seen. When user clicks the Register button, the account must be created. As the same way, to test functionality of the Login button, by using an existing account, whether it is logged into the system by the Login Button will be tested. These two functions will be tested back to back, and to test Login Button and its function, the account that created Register Button will be used for Login Button. These buttons reads the data given from user, login button searches data, register button inserts into database. With Upload Photo/Post Buttons, when it clicks on these buttons, uploaded photo/post will be stored in database. When these functions are completed, uploaded photo/post will be shown by user interface, on profile page and main page. Delete Photo/Post/Account Buttons deletes and searches, after creating a post or account or changed profile photo, these are deleted by the delete button. After execution, post, photo or account will be deleted from database. Results will be printing by the user interface again. When click on Edit Post Button, given current data of the post by the user interface will be changed after received the new data from dialog box and this button will read new data, searches on database and writes into the database. After execution, new data of the post will be given on user interface. Results will be shown on user interface. While log in into the system, by clicking Forgot Password Button, to write new data, a new page will be provided to the user. This functionality will be tested by Login page interface. After entering new data, by Save Button the data will be stored in database and writes on that account’s information. Its results are shown on database. By Change Password Button, the same execution and test will be played.

**Application**

|  |  |
| --- | --- |
| **GUI Function** | **DB Interface Function(s)** |
| Register Button | Read  Search  Insert |
| Login Button | Read  Search  Insert |
| Upload Photo/Post Button | Store  Write |
| Delete Photo/Post/Account Button | Delete  Search |
| Edit Post Button | Read  Search  Write |
| Search Button | Search |
| Forgot Password Button | Read |
| Save Button | Store  Write |
| Change Password Button | Read  Search  Insert |

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

# References

[1] “Test UI for a Single App  :   Android Developers.” *Android Developers*, developer.android.com/training/testing/ui-testing/espresso-testing.

[2] <http://www.vogella.com/tutorials/AndroidTestingEspresso/article.html>,

[3] <https://github.com/codepath/android_guides/wiki/UI-Testing-with-Espresso>

[4] Example test documentations and SE302 course materials.

[5] SE301 Software Engineering course materials about testing.