kodisnowhere

SHELFBUTLER

Software Test Documentation

14.04.2013

1 Introduction

1.1 Problem Definition

This software product will be a mobile application that is to be used by collectors gathering books. It helps the users arranging their collection by providing features such as recording locations of the items, keeping a wishlist, readlist and a favorite list in which the users can save their wished, read/unread of favorite books, respectively. Moreover, they can see book reviews, recommendations and if they want, they can also review books they own or buy some. They also can share their lists in social media, if they like. The application will also has a web interface. Users will be able to edit their data through it, which makes the application easily accessible.

More precisely, the software employs a mobile application which is capable of adding books to the collection by either entering their ISBN number or scanning their barcode via the camera on the phone or deleting them, adding their physical locations, searching books on the web according to their title, author, press or category, giving an opportunity to the user to buy this book online and recommending books alike. Moreover, it keeps a list of favorite books, read/unread ones and a wishlist which the user can share on social media. The user can review books online. The application also keeps the data of lent items. Furthermore, the mobile application has Turkish and English language support. The project also has a web application that the users can access and edit their library. So the application will be reachable anywhere with an internet connection.

1.2 Purpose and Scope

The purpose of this document is to provide the test cases of the shelfbutler project. It defines the objective, scenario, expected outcomes and procedural requirements for each test case. It also includes a table showing which test case is related to which one. The software will be tested using guidance of this document. Although it covers all the test cases specifically in detail, a little portion of the details is subject to change in test phase.

1.3 Definitions, Acronyms & Abbreviations

Term	**Definition**
User	Person who runs and uses the application.
IDE	Integrated Development Environment
OS	Operating system.
iOS	iPhone Operating System.
ISBN	International Standard Book Number

1.4 Overview

This document includes four chapters. One of which is the introduction chapter, which briefly explains the project and scope of this document.

The following chapter provides all functional test cases with details. Each test case is described with its objective, scenario, expected outcome and procedural requirements.

Third chapter provides the unit tests for synchronization feature. Each case is explained in detail.

Remaining two sections define the testing environment and testing tools.

2 Scenarios

2.1 Functional Test Case: Register - 01

Objective
To test the register component of the system.
Scenario
1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields (name, surname, email, password) with valid data. 3. User clicks “Register” button.
Expected Outcome(s)
• A new user is created in the system. • User is redirected to the application launch page.
Procedural Requirements
• The application shall be started.

2.2 Functional Test Case: Register - 02

Objective
To test the register component of the system with an invalid e-mail.
Scenario
1. User clicks “Create New Account” button on application launch page. 2. User fills the fields name, surname, and password with valid data, e-mail with an invalid data. 3. User clicks “Register” button.
Expected Outcome(s)
• The user is warned to enter a valid e-mail on register page.
Procedural Requirements

- The application shall be started.

2.3 Functional Test Case: Register - 03

Objective
To test the register component of the system with a not unique e-mail.
Scenario
1. User clicks “Create New Account” button on application launch page. 2. User fills the fields name, surname, and password with valid data, e-mail with a previously registered user’s e-mail data. 3. User clicks “Register” button. User clicks “Log In” button on application launch page. 2. User fills the required fields (email, password) with valid data. 3. User clicks “Log In” button.
Expected Outcome(s)
• The user is warned that entered e-mail address is already registered.
Procedural Requirements
• The application shall be started.

2.4 Functional Test Case: Register - 04

Objective
To test the register component of the system with an invalid password.
Scenario
1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields (name, surname, email) with valid data. 3. User fills the “password” field with a password consisting of an invalid character. 4. User clicks “Register” button.
Expected Outcome(s)
• “Invalid Password” error is displayed and a new user is not created.
Procedural Requirements
• The application shall be started.

2.5 Functional Test Case: Register - 05

Objective
To test the register component of the system with an invalid password.
Scenario
1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields (name, surname, email) with valid data. 3. User fills the “password” field with a password shorter than six characters. 4. User clicks “Register” button.
Expected Outcome(s)
• “Invalid Password” error is displayed and a new user is not created.
Procedural Requirements
• The application shall be started.

2.6 Functional Test Case: Login - 01

Objective
To test the login component of the system with valid data.
Scenario
1. User clicks “Log In” button on application launch page. 2. User fills the required fields (email, password) with valid data. 3. User clicks “Log In” button.
Expected Outcome(s)
• The user is logged in to the application. • New session is created for the user. • The user is redirected to collection page.
Procedural Requirements
• The application shall be started and user must be registered.

2.7 Functional Test Case: Login - 02

Objective
To test the login component of the system with invalid credentials.
Scenario
1. User clicks “Log In” button on application launch page. 2. User fills the required fields (email, password) with invalid data. 3. User clicks “Log In” button.
Expected Outcome(s)
• The user is not logged in to the application. • A warning indicating that login credentials are wrong is shown. • The user stays on the login page.
Procedural Requirements
• The application shell be started.

2.8 Functional Test Case: Forgot Password - 01

Objective
To test the reset password feature of the system with valid data.
Scenario
1. User clicks “Log In” button on application launch page. 2. User fills the email field with valid data. 3. User click “Forgot Password?” button on login page.
Expected Outcome(s)
• New password is generated randomly and login credentials are updated in the database. • An email with randomly generated new password is sent to user. • A warning indicating that password has changed is shown. • The user stays on the login page.
Procedural Requirements
• The application shall be started and user must be registered.

2.9 Functional Test Case: Forgot Password - 02

Objective

To test the reset password feature of the system with invalid data.
Scenario
1. User clicks “Log In” button on application launch page. 2. User fills the email field with invalid data. 3. User click “Forgot Password?” button on login page.
Expected Outcome(s)
• New password is not generated. • A warning indicating that credentials are wrong is shown. • The user stays on the login page.
Procedural Requirements
• The application shall be started and user must be registered.

2.10 Functional Test Case: Change Password - 01

Objective
To test the change password functionality.
Scenario
1. User clicks “Settings” tab in application launch page. 2. User clicks “Change Password” button. 3. User is redirected to a new page. 4. User enters a new password in the required field. 5. User clicks “Change Password” button.
Expected Outcome(s)
• The password associated with the user is changed in the database.
Procedural Requirements
• The application shall be started. • User is logged in.

2.11 Functional Test Case: Change Password - 02

Objective
To test the change password functionality.

Scenario
1. User clicks “Settings” tab in application launch page. 2. User clicks “Change Password” button. 3. User is redirected to a new page. 4. User enters an invalid new password in the required field. 5. User clicks “Change Password” button.
Expected Outcome(s)
• “Invalid Password” error should be displayed.
Procedural Requirements
• The application shall be started. • User is logged in.

2.12 Functional Test Case: Logout - 01

Objective
To test the logout functionality.
Scenario
1. User clicks “Logout” tab in collection page.
Expected Outcome(s)
• The session should be invalidated. • The user is redirected to application launch page.
Procedural Requirements
• The application shall be started. • User shall be logged in. • Collection page is open.

2.13 Functional Test Case: Collection Management - 01

Objective
To test adding a book to the collection.
Scenario
1. User clicks add to collection button on corresponding book’s page.

Expected Outcome(s)
• The corresponding book is added to the user’s collection.

Procedural Requirements
• The application shall be started. • User shall be logged on. • User shall be on a searched/scanned book’s page.

2.14 Functional Test Case: Collection Management - 02

Objective
To test removing a book from collection.
Scenario
1. User clicks remove button from corresponding book’s page.
Expected Outcome(s)
• The corresponding book is removed from user’s collection. • The user is redirected to the collection page.
Procedural Requirements
• The application shall be started. • User shall be logged on. • The user shall be on desired book’s page.

2.15 Functional Test Case: Collection Management - 03

Objective
To test changing location of a book from collection.
Scenario
1. User click “Location Information” button. 2. User is redirected to a new page. 3. User fills location row and column fields. 4. User clicks save button.
Expected Outcome(s)
• The book entry is updated in the database. • The location information is displayed on the book page.

Procedural Requirements

- The application shall be started.
- User shall be logged in.
- The book page is open.

2.16 Functional Test Case: Wishlist - 01

Objective
To test adding a book to the wishlist.
Scenario
1. User clicks add to wishlist button on corresponding book's page.
Expected Outcome(s)
• The corresponding book is added to the user's collection.
Procedural Requirements
• The application shall be started. • User shall be logged on. • User shall be on a searched/scanned book's page.

2.17 Functional Test Case: Wishlist - 02

Objective
To test removing a book from the wishlist.
Scenario
1. User clicks remove button from corresponding book's page.
Expected Outcome(s)
• The corresponding book is removed from user's wishlist. • The user is redirected to the wishlist page.
Procedural Requirements
• The application shall be started. • User shall be logged on • User shall be on desired book's page.

2.18 Functional Test Case: Barcode Scan - 01

Objective

To test barcode scanning of a book.

Scenario

1. User clicks scan button.
2. User scans barcode of a book from camera.

Expected Outcome(s)

- The corresponding book page is displayed.

Procedural Requirements

- The application shall be started.
- User shall be logged on.
- User shall be on add page.

2.19 Functional Test Case: Search - 01

Objective

To test scanning barcode of a book.

Scenario

1. User clicks scan button.
2. User scans barcode of a book from camera.

Expected Outcome(s)

- The corresponding book page is displayed.

Procedural Requirements

- The application shall be started.
- User shall be logged on.
- User shall be on add page.

2.20 Functional Test Case: Search - 02

Objective

To test searching a book with its name.
Scenario
1. User clicks search field and fills it with book’s name.
Expected Outcome(s)
• A list containing search results is displayed.
Procedural Requirements
• The application shall be started. • User shall be logged on. • User shall be on add page.

2.21 Functional Test Case: Search - 03

Objective
To test searching a book with its author.
Scenario
1. User clicks search field and fills it with book’s author.
Expected Outcome(s)
• A list containing search results is displayed.
Procedural Requirements
• The application shall be started. • User shall be logged on. • User shall be on add page.

2.22 Functional Test Case: Search - 04

Objective
To test searching a book with its isbn.
Scenario
1. User clicks search field and fills it with book’s isbn.
Expected Outcome(s)

- A list containing search results is displayed.

Procedural Requirements

- The application shall be started.
- User shall be logged on.
- User shall be on add page.

2.23 Functional Test Case: Book Evaluation - 01

Objective

To test rating a book.

Scenario

1. User clicks on a star.

Expected Outcome(s)

- The corresponding star determine the rate.
- All stars from left to the clicked star turn to yellow.

Procedural Requirements

- The application shall be started.
- User shall be logged on.
- User shall be on book evaluation page.

2.24 Functional Test Case: Book Evaluation - 02

Objective

To test reviewing a book.

Scenario

1. User clicks review field and fills it with a review.

Expected Outcome(s)

- The corresponding review is added.

Procedural Requirements

- The application shall be started.
- User shall be logged on.
- User shall be on book evaluation page.

2.25 Functional Test Case: Add to Favorites - 01

Objective
To test if a book is added to the favorite list successfully.
Scenario
1. User clicks “Add to Favorites” button in the book page.
Expected Outcome(s)
• The book is added to the favorite list in the system database. • The heart figure in the page turns to red.
Procedural Requirements
• The application shall be started. • User shall be logged in. • Page of a book in collection shall be opened.

2.26 Functional Test Case: Remove from Favorites - 01

Objective
To test if a book is removed from the favorite list.
Scenario
1. User clicks “Remove from favorites” button in the book page.
Expected Outcome(s)
• The book is removed from the favorite list in the database. • The red heart figure turns into light grey.
Procedural Requirements
• The application shall be started. • User shall be logged in. • The book is already in the favorite list of the user. • The book page is open.

2.27 Functional Test Case: Display Favorites - 01

Objective
To test if the favorite list is displayed correctly.
Scenario
1. User clicks “Favorites” tab.
Expected Outcome(s)
• The favorite list of the user is displayed.
Procedural Requirements
• The application shall be started. • User shall be logged in.

2.28 Functional Test Case: Add to Readlist - 01

Objective
To test if a book is added to the readlist successfully.
Scenario
1. User clicks “Add to Readlist” button in the book page.
Expected Outcome(s)
• The book is added to the readlist in the system database. • The bookmark figure in the page turns to green.
Procedural Requirements
• The application shall be started. • User shall be logged in. • Page of a book in collection shall be opened.

2.29 Functional Test Case: Remove from Readlist - 01

Objective
To test if a book is successfully removed from the readlist.
Scenario
1. User clicks “Remove from readlist” button in the book page.
Expected Outcome(s)

- The book is removed from the favorite list in the database.
- The green bookmark figure turns into light grey.

Procedural Requirements
• The application shall be started. • User shall be logged in. • The book is already in the readlist of the user. • The book page is open.

2.30 Functional Test Case: Display Readlist - 01

Objective
To test if the readlist is displayed correctly.
Scenario
1. User clicks “Readlist” tab.
Expected Outcome(s)
• The readlist of the user is displayed.
Procedural Requirements
• The application shall be started. • User shall be logged in.

2.31 Functional Test Case: Share - 01

Objective
To test the share feature of the system with a book via Facebook.
Scenario
1. User clicks Facebook icon on book page.
Expected Outcome(s)
• User is redirected Facebook share page of iOS. • New Facebook post is filled with book data such as title, author and cover image.
Procedural Requirements

- The application shall be started.
- User is logged in.
- User is on book page.

2.32 Functional Test Case: Share - 02

Objective
To test the share feature of the system with a book via Twitter.
Scenario
1.User clicks Twitter icon on book page.
Expected Outcome(s)
• User is redirected Twitter share page of iOS. • New Twitter post is filled with book data such as title, author and cover image.
Procedural Requirements
• The application shall be started. • User is logged in. • User is on book page.

2.33 Functional Test Case: Share - 03

Objective
To test the share feature of the system with a book via email.
Scenario
1.User clicks email icon on book page.
Expected Outcome(s)
• User is redirected to email application of iOS. • New email is filled with book data such as title, author and cover image.
Procedural Requirements
• The application shall be started. • User is logged in. • User is on book page.

2.34 Functional Test Case: Share - 04

Objective
To test the share feature of the system with a list via Facebook.
Scenario
1.User clicks Facebook icon on list page.
Expected Outcome(s)
• User is redirected Facebook share page of iOS. • New Facebook post is filled with the link of intended list on web.
Procedural Requirements
• The application shall be started. • User is logged in. • User is on any list page (Favorites, Readlist, Wishlist).

2.35 Functional Test Case: Share - 05

Objective
To test the share feature of the system with a list via Twitter.
Scenario
1. User clicks Twitter icon on list page.
Expected Outcome(s)
• User is redirected Twitter share page of iOS. • New Twitter post is filled with the link of intended list on web.
Procedural Requirements
• The application shall be started. • User is logged in. • User is on any list page (Favorites, Readlist, Wishlist).

2.36 Functional Test Case: Share - 06

Objective

To test the share feature of the system with a list via email.

Scenario

1. User clicks email icon on list page.

Expected Outcome(s)

- User is redirected to email application of iOS.
- New email is filled with list content.

Procedural Requirements

- The application shall be started.
- User is logged in.
- User is on any list page (Favorites, Readlist, Wishlist).

2.37 Functional Test Case: Lend Book - 01

Objective

To test if the track information of a lent book is entered correctly.

Scenario

1. User clicks “Location Information” button.
2. User is redirected to a new page.
3. User enters the fields (to whom, lend date, comment).
4. User clicks save button.

Expected Outcome(s)

- A track entry for the book is added to the system database.
- The last track information is displayed on the book page.

Procedural Requirements

- The application shall be started.
- User shall be logged in.
- The book page is open.

2.38 Functional Test Case: Receive Book - 01

Objective

To test if the track information of a lent book is entered correctly.

Scenario

1. User clicks “Location Information” button. 2. User is redirected to a new page. 3. User fills the return date field. 4. User clicks save button.
Expected Outcome(s)
• The track entry of the book is updated in the system database. • The location information of the book is displayed on the book page.
Procedural Requirements
• The application shall be started. • User shall be logged in. • The book page is open. • The book has already a track entry.

2.39 Functional Test Case: Online Shopping - 01

Objective
To test if the online shopping functionality runs correctly.
Scenario
1. User clicks “Shop Online” button.
Expected Outcome(s)
• The user is redirected to third party shopping websites.
Procedural Requirements
• The application shall be started. • User shall be logged in. • The search result page is open.

2.40 Functional Test Case: Get Recommendation - 01

Objective
To test if the get recommendation functionality runs correctly.
Scenario
1. User shakes the phone.
Expected Outcome(s)

• A list of books are recommended to the user based on his/her choices.
Procedural Requirements
• The application shall be started. • User shall be logged in.

2.41 Functional Test Case: Sync - 01

Objective
To test the synchronization component of the system.
Scenario
1. User launches the application.
Expected Outcome(s)
• All dirty records in the user's database are pushed to cloud.
Procedural Requirements
• The application shall be started. • There should be available network connection.

3 Unit Tests

3.1 SBSynchronization

Test cases presented below are to complete the unit tests of SBSynchronization utility which provides synchronization between cloud and local repository.

3.1.1 Unit Test Case: Book Create - 01

Objective
To test the book create web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)

- Book record is created on cloud database.
- A return value indicating that operation is successful.

3.1.2 Unit Test Case: Book Create - 02

Objective
To test the book create web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Book record is not created on cloud database. • A return value indicating that operation is not successful.

3.1.3 Unit Test Case: Book Update - 01

Objective
To test the book update web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Book record is updated on cloud database. • A return value indicating that operation is successful.

3.1.4 Unit Test Case: Book Update - 02

Objective
To test the book update web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Book record is not updated on cloud database. • A return value indicating that operation is not successful.

3.1.5 Unit Test Case: Book Delete - 01

Objective
To test the book delete web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Book record is deleted on cloud database. • A return value indicating that operation is successful.

3.1.6 Unit Test Case: Book Delete - 02

Objective
To test the book delete web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Book record is not deleted on cloud database. • A return value indicating that operation is not successful.

3.1.7 Unit Test Case: User Create - 01

Objective
To test the user create web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• User record is created on cloud database. • A return value indicating that operation is successful.

3.1.8 Unit Test Case: User Create - 02

Objective
To test the user create web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• User record is not created on cloud database.   • A return value indicating that operation is not successful.

3.1.9 Unit Test Case: User Update - 01

Objective
To test the user update web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• User record is updated on cloud database.   • A return value indicating that operation is successful.

3.1.10 Unit Test Case: User Update - 02

Objective
To test the user update web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• User record is not updated on cloud database.   • A return value indicating that operation is not successful.

3.1.11 Unit Test Case: Review Create - 01

Objective
To test the review create web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is created on cloud database. • A return value indicating that operation is successful.

3.1.12 Unit Test Case: Review Create - 02

Objective
To test the review create web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is not created on cloud database. • A return value indicating that operation is not successful.

3.1.13 Unit Test Case: Review Update - 01

Objective
To test the review update web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is updated on cloud database. • A return value indicating that operation is successful.

3.1.14 Unit Test Case: Review Update - 02

Objective

To test the review update web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is not updated on cloud database. • A return value indicating that operation is not successful.

3.1.15 Unit Test Case: Review Delete - 01

Objective
To test the review delete web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is deleted on cloud database. • A return value indicating that operation is successful.

3.1.16 Unit Test Case: Review Delete - 02

Objective
To test the review delete web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Review record is not deleted on cloud database. • A return value indicating that operation is not successful.

3.1.17 Unit Test Case: Track Create - 01

Objective
To test the track create web service with valid data.

Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is created on cloud database. • A return value indicating that operation is successful.

3.1.18 Unit Test Case: Track Create - 02

Objective
To test the track create web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is not created on cloud database. • A return value indicating that operation is not successful.

3.1.19 Unit Test Case: Track Update - 01

Objective
To test the track update web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is updated on cloud database. • A return value indicating that operation is successful.

3.1.20 Unit Test Case: Track Update - 02

Objective
To test the track update web service with invalid data.
Scenario

1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is not updated on cloud database. • A return value indicating that operation is not successful.

3.1.21 Unit Test Case: Track Delete - 01

Objective
To test the track delete web service with valid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is deleted on cloud database. • A return value indicating that operation is successful.

3.1.22 Unit Test Case: Track Delete - 02

Objective
To test the track delete web service with invalid data.
Scenario
1. Call the corresponding web service with a valid JSON string.
Expected Outcome(s)
• Track record is not deleted on cloud database. • A return value indicating that operation is not successful.

4 Testing Environment

All tests are to be conducted both on mobile device and server side. A mobile device with an operating system of iOS6 and higher equipped with network connection is required. For the server side, a simple computer with internet connection is enough.

5 Performance and Tools

Since shelfbutler is a mobile application and handles all server connection in the background, response time is fast enough for smooth user interaction. User interface tests are operated using Automation Tests provided by XCode IDE.