BOOKSWAP APPLICATION FUNCTIONAL SPECIFICATION

Author
Ivan Yaremko
C00239239

Supervisor
Dr Chris Staff

Submission date
26/11/2021

1 ABSTRACT

The purpose of this document is to investigate the functionalities of BookSwap.
This document uses the FURPS model to clarify the functionality, usability, reliability, performance, and supportability of BookSwap.
This document illustrates the use case diagram of BookSwap along with brief, and detailed use cases. The document also investigates functionalities of similar application.

2 CONTENTS

3 TABLE OF FIGURES

4 INTRODUCTION

This document focuses on the functionality specifications of the BookSwap project. This document discusses the purpose of BookSwap and who are the user groups. The document uses the functionality, usability, reliability, performance, and supportability (FURPS) model to outline the specifications of BookSwap. The requirements for the implementation of the application are outlined, and a use case diagram is provided along with brief, and detailed use cases. The document also investigates the functionalities of similar applications.

5 PURPOSE

BookSwap is a web application developed for the second-hand book market. The application allows members to trade books with each other, this brings the benefit of reducing the cost for reading a new book and re-using old books.

Members of the application can:

- Upload books that they intend to trade.
- Search the marketplace for books.
- Request to swap books with other members.
- Update their profiles

The application will be able to:

- Register and authenticate members.
- Allow members to add books by using an external ISBN API.
- Allow members to add photos by using an external cloudinary API storage.
- Allow members to search the marketplace by querying the backend database.
- Facilitated swapping books.

6 USERS

The target user groups for this application are for people who like to read. Ideally, the user group of this application would consist of like-minded people who want to create a local community, save money on reading new books by swapping books with other users, and who want to be environmentally friendly by reusing books.

7 FURPS+

FURPS stands for functionality, usability, reliability, performance, and supportability. FURPS represents a checklist of software quality. The BookSwap project utilises the FURPS model to map out the functionalities required.

7.1 FUNCTIONALITY

Functionality describes the features and capabilities of the application, the general functions and capabilities that will be delivered.

7.1.1 Core

The core functionalities of BookSwap are the must-have features for the application to work.

• **Authenticate** - Users must register with the application and be logged in to access features.

• **Search** - Members must be able to search the marketplace for books they would like to trade for.

• **CRUD Books** - Members must have a create, read, update, and delete (CRUD) functionality of books linked to their account.

• **CRUD Profile** – Members must be able to update their profile.

• **Swap books** - Members must be able to swap their books with other members of the application.

• **Chat** - Members must be able to communicate with each other to arrange a swap.

7.1.2 Non-core

The non-core functionalities of BookSwap represent the nice to have features. They are not critical for the implementation of the project but will enhance it.

• **Wishlist** - Members may create a wish list of books they want to trade for.

• **Novel recommendation** - A system for members to see recommended novels that they might like to read currently available in the marketplace.

• **Google Map** - Google maps integrated into the application to aid members in arranging a location for the swap.
• **Member Rating** - Members can rate other member with whom they swapped books with. After enough ratings, members get generated an overall rating that is visible to other members.

• **Member book review** - Members can write a personal book review on books they wish to exchange. Personal reviews can increase the

7.2 Usability

Usability is assessed by the overall user experience with the application. This includes answering questions such as, how effective is the product from the end user point of view? Is the user interface aesthetically acceptable? Is the documentation accurate?

• **Authenticating** - The registration and logging into the application should be a simple standard process. The required details for authentication from the user need to be clearly stated. Problems authenticating need to be documented to the end-user.

• **Search** - The search function in the application should be responsive, easy to navigate and show the optimal results.

• **Swapping** - The swapping functionality should be a clear step by step process for the end user to understand.

• **CRUD Books** - The information needed to perform CRUD operations must be clearly outlined to the end user. The constraints and errors must be documented for the end-user.

• **Chat** - End users must be able to understand who they are talking to, and which messages are being sent by whom.

7.3 Reliability

Reliability considers the availability, accuracy, and recoverability of the application.

• **Authenticating** The process to authenticate with valid credentials should not take longer than 5 seconds 99% of the time.

• **Search** - A result for a search should not take longer than 10 seconds 99% of the time.

• **Swapping** - To initiate and form a swap must work 90% of the time.

• **CRUD Books** - The end user must be able to, in a valid way, create, remove, update, and delete books from their list without occurring server errors 99% of the time.

• **Chat** - The chat functionality must show messages within 10 seconds of them being sent 99% of the time.

7.4 PERFORMANCE

The performance of an application is measured by the processing speed, response time, resource consumption, throughput, and efficiency.

The use of an external ISBN API is vital for the application, the connection to this API must be up 99% of the time excluding the down time maintenance from the API provider. Without a stable connection to this API end users will encounter problems when engaging with the application. Initiating and receiving a GET request to this API must not take longer than 5 seconds.

The back-end API of the application must also be up 99% of the time. This API contains end user information, the up time of this API is extremely important to the functionalities of the application. The HTTP requests to this API must not be longer than 10 seconds to respond.

Components within the application's front-end must not take longer than 5 seconds to render. Fluidity for the application is important, end-users must not feel that they are stuck on a loading screen. Each loading component must show reasonable progress and document to the user what they are waiting for.

7.5 METRICS

The success of this project will be measured based on the functionality metrics described in the FURPS headings above.

8 REQUIREMENTS

Requirements in this context means the necessary tools and libraries required for the application to be functional. This document gives a brief overview of the different technologies and tools the application will utilise, a more detailed explanation of these tools is available in the BookSwap research report.

8.1 FRONT END

The front end is where users interact with the application. The front-end renders information to the users and enables the user to access functionalities. This application will use the React [1] library as the front end.

React is an open-source library developed by Facebook. It is used for building user interfaces for single-page application (SPA). SPA are applications that do not need to reload the page during its use. React is commonly used for the view layer of applications. React allows developers to create reusable UI components.

Since React is only a library and not a framework, the developer is free to choose libraries to integrate with React. Different open-source libraries are used with React to create a front end “stack”.

Axios [2] is a HTTP client library that enables clients to make requests to given endpoints. Since Axios is promise-based this gives the ability to use JavaScript’s async and await commands. Axios hosts several functions to allow simple HTTP requests. The front-end part of the application will utilise Axios to connect with the different API endpoints.

To finalise the front end, React stack, MobX and React Router libraries will also be integrated. MobX [3] is a tool to allow state management within the React application. React Router [4] enables navigation among components within a React application.

8.2 BACK END

The back end of the application is used by the front end to render information into the browser. The back end contains the database where all the necessary application information is stored. This database is exposed through controllers and is accessed by HTTP requests.

.NET [5], pronounced dot net, is an open-source framework developed by Microsoft. The framework includes a host of open-source libraries such as:

- **Web API.** Used for building HTTP services
- **Entity.** Used for accessing databases through objects of domain specific classes.
- **Identity**: Supports authentication by login information.

8.3 ISBN API

ISBNdb API [6] provides a catalogue of books that can be requested via HTTP GET requests by using books ISBN numbers. This external API provider will be integrated in the BookSwap application to achieve functionality goals.

8.4 Cloudinary API

The Cloudinary API provides a way to store photos and access those photos via public URL. This external API provider will be integrated into BookSwap.

9 USE CASES AND DIAGRAM

A use case diagram is used to represent the functionalities of the system, its actors, and the keyways it will be used. A use case is a written description of how users will perform functionalities within the application.
Brief use cases are one-paragraph statements, that outline the main success scenario. These are used to get a quick understanding of the subject and scope.
Detailed use cases outline all steps and variations in detail. They also include supporting sections such as preconditions. Detailed use cases are created after use cases have been identified and written in a brief format

9.1 Context Diagram

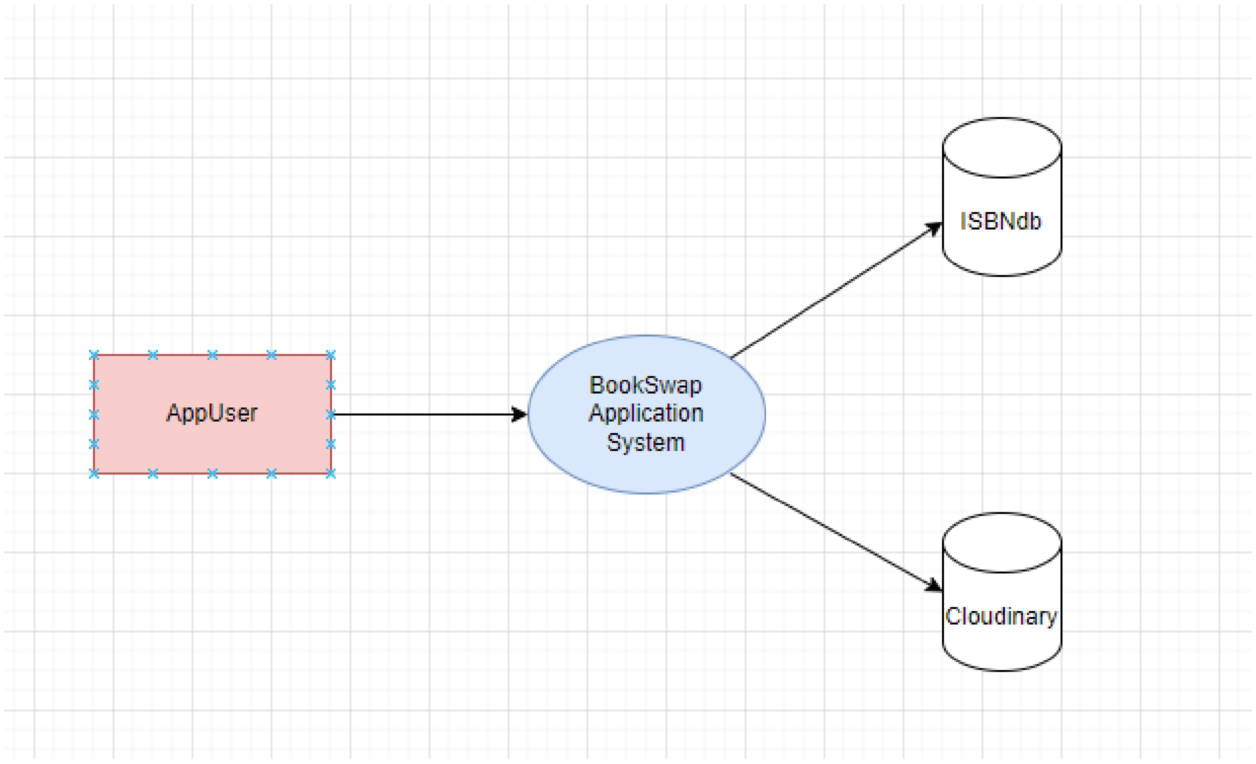

Figure 1 Context Diagram

9.2 Use Case Diagram

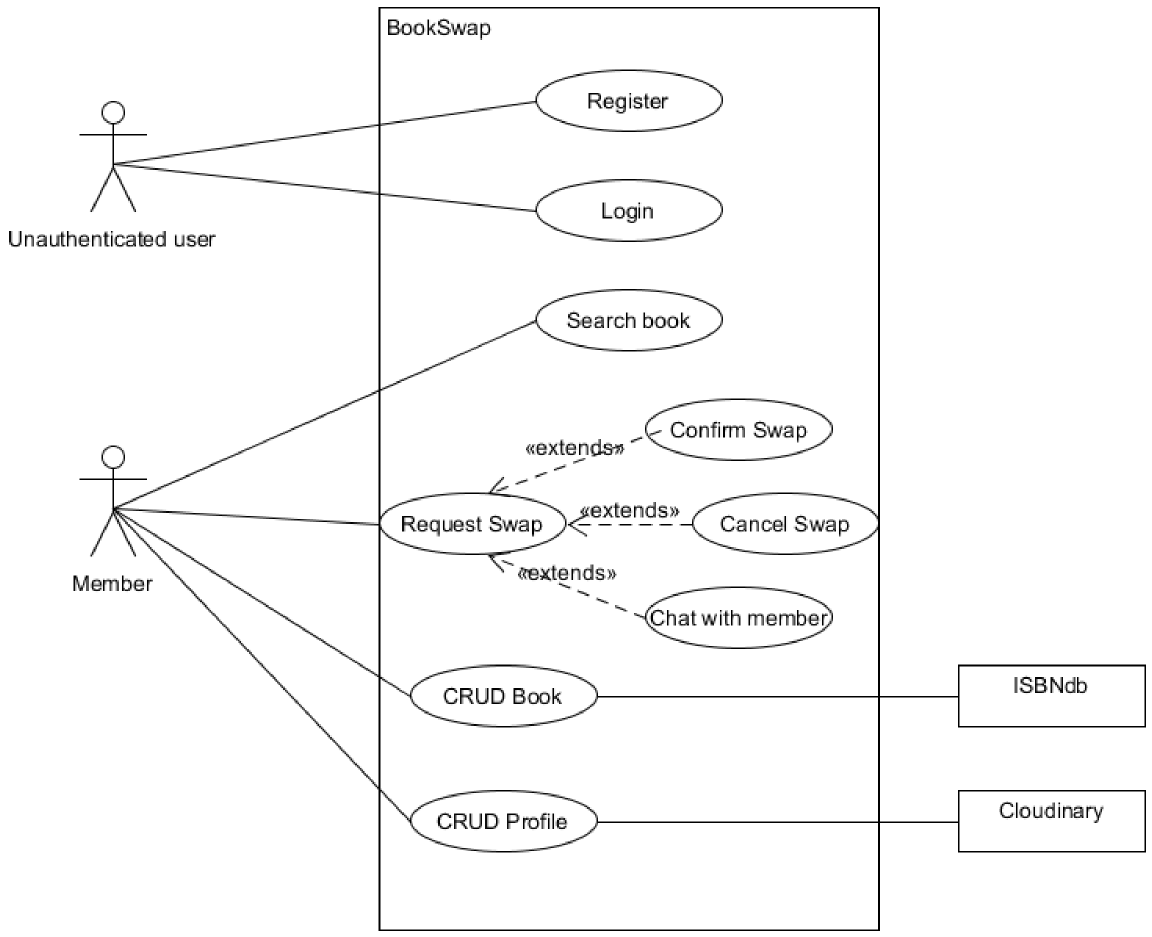

Figure 2 Use case diagram of BookSwap

9.3 Brief Use Cases

9.3.1 Register

Use Case Name: Register

Actor(s): Unauthenticated User

Description: This use case begins when a user visits the website and wishes to register. The user enters the required details into a form. After a successful registration the user may login.

9.3.2 Login

Use Case Name: Login

Actor(s): User

Description: This use case begins when a user visits the website and wishes to login. The user enters the required details into a form. After a successful login the user may continue with the application.

9.3.3 CRUD Book

Use Case Name: CRUD Book

Actor(s): Member, ISBNdb API

Description: This use case begins when a member wishes to use of the create, read, update, and remove functionalities on their books. When an actor wants to create a new book, the actor enters the book’s ISBN, and the ISBNdb API pulls in the book’s information.

9.3.4 Search book

Use Case Name: Search Book

Actor(s): Member

Description: This use case begins when a member wishes search for a book in the marketplace. The member enters the book’s name into the search bar. The BookSwap application will query the database for the book and display information back to the member.

9.3.5 Swap book

Use Case Name: Request book swap

Actor(s): Member

Description: This use case begins when a member wishes to swap a book with another member. The requester member initiates the request by selecting a book from the marketplace and requesting a swap with member who has the book. The requestee member gets to select a book from the requester’s list of offers, when a member selects a book from the list the swap is confirmed.

9.3.6 Confirm Swap

Use Case Name: Confirm swap

Actor(s): Member

Description: This use case begins when a member wishes to confirm a swap for a book with another member. The member picks a books to swap with from a list of books of the member requesting the swap.

9.3.7 Cancel Swap

Use Case Name: Cancel swap

Actor(s): Member

Description: This use case begins when a member wishes to decline a swap request.

9.3.8 Chat with member

Use Case Name: Chat with member

Actor(s): Member

Description: This use case begins when a swap is confirmed between members. Members are brought into a chat window to organize a meeting place to swap their books.

9.3.9 CRUD Profile

Use Case Name: CRUD Profile

Actor(s): Member, Cloudinary

Description: This use case begins when a member wishes update details. The member may upload a profile image via cloudinary.

9.4 Detailed Use Cases

9.4.1 Register

Use Case Name:	Register
Actor(s):	User
Main success scenario:	This use begins when a user wishes to register with the application. 1) The user visits the application. 2) The user is prompted with options to login or register. 3) The user selects the register option. 4) The user is redirected to the register form. 5) The user enters their email, password, username. 6) The user submits the register form. 7) The user is redirected to the login form page.
Alternatives:	5a) The user enters invalid email, password, or username. The user gets prompted that their details are invalid.

9.4.2 Login

Use Case Name:	Login
Actor(s):	User
Main success scenario:	This use begins when a user wishes to login with the application. 1) The user visits the application. 2) The user is prompted with options to login or register. 3) The user selects the login option. 4) The user is redirected to the login form. 5) The user enters their email and password. 6) The user submits the login form. 7) The user is redirected to the home page of the application.
Alternatives:	5a) The user enters invalid email or password. The user gets prompted that their details are invalid.

9.4.3 CRUD Book

Use Case Name:	CRUD Book
Actor(s):	Member, ISBNdb API
Preconditions:	The actor must be authenticated with the application. The application and its dependent API's are not down.
Main success scenario:	This use case begins when an actor wishes to use of the create, read, update, and remove functionalities on their books. 1) The actor navigates to their list of offers. 2) The actor is given the options to add, remove or update existing books in the list. 3) The actor selects one of the options. 4) The actors change to the offers are saved.
Alternatives:	3a) The actor selects to add a book to their offers. The member is prompted with a form to add a new book. The actor enters the books ISBN, the application utilises the ISBNdb API to pull in information about the book. 3b) The actor selects to remove a book from their offers. The member selects the book they want to remove from the list of offers, they then select the remove option. 3c) The actor selects to update one of the books from their list of offers. The member is prompted with a form that contains information about the book. The member can edit information about the book.
Postconditions:	1) The CRUD operations on the books from the members list of offers is saved.

9.4.4 Swap book

Use Case Name:	Book Swap
Actor(s):	Member(s)
Preconditions:	The member must be authenticated with the application. The application and its dependent APIs are not down. The member selected an offer that exists in the marketplace.
Main success scenario:	This use case begins when a member initiates a book swap with another member. 1) The member requests a book from the marketplace. 2) The member initiates a swap for the offer with the member who is offering the book. 3) The requestee member gets notified that another member has initiated a swap for one of their offers. 4) The requestee member is provided with a list of offers that are owned from the requester member. 5) The requestee member selects an offer from that list.
Alternatives:	5a) The requestee member declines the offer from the requester. 5b) The requestee member selects a book to swap with, the application confirms the swap.
Postconditions:	The books that were swapped between members are removed from the marketplace.

9.4.5 Chat with member

Use Case Name:	Chat with member
Actor(s):	Member(s)
Preconditions:	Two members have agreed and confirmed a swap of books.
Main success scenario:	The use case begins when a swap has been confirmed between two members. 1) The requester and requestee member are prompted to chat to each other.
Alternatives:	1a) The chat functionality is offline.
Postconditions:	Two members can chat with each other to organize a meeting for the swap.

9.4.6 CRUD Profile

Use Case Name:	CRUD Profile
Actor(s):	Member, Cloudinary
Preconditions:	Member is authenticated with the application.
Main success scenario:	The use case begins when an member wishes to change their profile details. 1) Member navigates to profile section. 2) Member updates their profile information and or image.
Alternatives:	2a) Cloudinary is offline
Postconditions:	The profile changes for the member are changed.

10 SIMILAR APPLICATIONS

A similar application exists in the United Kingdom market, BookSwap.co.uk [7] allows members to give away books that they no longer need and exchange them for new ones that they want to read. Detailed research was conducted on BookSwap.co.uk in the BookSwap research document. This document will examine the similarities and differences of functionalities between BookSwap.co.uk and BookSwap.

10.1 SEARCH

To search for a book on BookSwap.co.uk, members use the search function to find a book by the title. Members then select the edition of the book they want from the given list provided by the application. Upon the selection of the edition, the member is brought to the page of that edition. A list of members offering the edition of the book is displayed, the requesting member selects an offer from this list.

BookSwap has similar functionality for searching a book in the marketplace. In the current iteration, only books already added to the marketplace will show up in the search function.

10.2 CRUD BOOKS

Members of BookSwap.co.uk add books they want to give away by:

- Searching the catalogue for the book.
- Select a similar edition.
- Select the “offer this book” option.
- Select the condition of this book.
- Their offer is then published by the application

Members of BookSwap.co.uk who want to delete, or update books they have on offer:

- They navigate towards “My offers” section of the application.
- Select the offer they want to edit.
- Use the editing tool to edit the offer.

The adding books functionality of BookSwap is different in the current iteration. Members will need to enter details into a form to add an offer to the marketplace.

10.3 SWAPPING BOOKS

BookSwap.co.uk utilises a third-party courier service to administrate the swapping of books. Members get to post their offers by:

- Printing the address label generated by the application.
- Dropping off their offer to a courier drop off box or office.

This implementation ensures that there is no foul play. The third-party courier confirms when a book is dropped off and delivered. Upon a successful delivery, the sender gets rewarded and the requestee gets a new book to read.

BookSwap does not have a third-party system to authenticate swaps. The application is only concerned with allowing members to digitally confirm a swap. The members will be responsible in arranging a meeting to swap books. The application implements a chat functionality to enhance the experience of arranging a swap for the members.

10.4 CHAT

Bookswap.co.uk does not have a chat functionality. For that application a chat is not necessary since the swapping of books is done by a third-party courier.
BookSwap will utilise a chat feature to allow members to arrange meeting place for a book swap.

10.5 WISHLIST

Bookswap.co.uk has a wish list functionality for its members. A member may create a wish list and add books to it, whenever a new offer is uploaded to the marketplace, the member will automatically receive an email notification to let them know that the book is now available.

A non-core functionality for BookSwap is a wish list system. This functionality would enhance the overall user experience.

10.6 MEMBER RATING

Bookswap.co.uk implements a member rating functionality. Ratings are generated for members after enough orders are completed by them and enough ratings are given by members who received the orders. When a member rating is generated, it is displayed near all users' active offers and under the members profile.

A member rating functionality is a non-core feature for BookSwap, such a feature would also enhance the overall user experience with the application.

11 CONCLUSION

In conclusion, this document has covered the following topics:

- Utilising the FURPS model to outline BookSwap’s functionalities. The model described the core, and non-core functionalities of the application.
- Requirements for the implementation of the application.
- Illustrated a use case diagram along with brief, and detailed use cases.
- Investigated functionalities of similar applications.

12 REFERENCES

[1]. Reactjs.org. 2021. Getting Started – React. [online] Available at: <https://reactjs.org/docs/getting-started.html> [Accessed 11 November 2021].

[2]. npm. 2021. axios. [online] Available at: <https://www.npmjs.com/package/axios> [Accessed 11 November 2021].

[3]. Mobx.js.org. 2021. README · MobX. [online] Available at: <https://mobx.js.org/README.html> [Accessed 11 November 2021].

[4]. Reactrouter.com. 2021. React Router | Docs Home. [online] Available at: <https://reactrouter.com/docs/en/v6> [Accessed 11 November 2021].

[5]. G. (2021, September 15). *Overview of .NET Framework - .NET Framework*. Microsoft Docs. https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

[6]. ISBNdb API Documentation v2 | ISBNdb. (2021, August 18). Isbndb.Com. Retrieved October 31, 2021, from https://isbndb.com/apidocs/v2

[7]. Bookswap.co.uk. 2021. Bookswap. [online] Available at: <https://www.bookswap.co.uk/about> [Accessed 12 November 2021].

13 Plagiarism Declaration

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name (Printed) : Ivan Yaremko

Student Number : C00239239

Signature:

26/04/2022

X [illegible]

Ivan Yaremko

Signed by: 6c456de4-a264-4906-bbdd-cd8e10592da5