Bookie

Software Requirements Specification

12.04.2021

Talha Bayburtlu 150118066
Zeynep Alıcı 150119517
Ahmet Akıl 150118038

Prepared for
CSE3044 Software Engineering Term Project

1. Introduction

This section introduces the requirement specification document for the Bookie mobile application. It provides the purpose and the scope of the system. Any definitions and references are listed in this section as well as an overview of the remaining requirements specification document.

1.1 Purpose

The purpose of the requirements specification document describes the functions and requirements for Bookie mobile application. This application allows people to exchange books and meet on common pleasures. By using this application people can eliminate the requirements like a membership to a library or having enough money to read it. Also it is easier to find and get contact with someone that has common pleasures than real life.

1.2 Scope

Bookie is a mobile application that allows people to exchange books and meet on common pleasures. This application eliminates the needs for a membership to a library or having enough money to buy it and read it. People may get in contact and share common interests on books that are inside their virtual libraries by just exploring people and books on their phone which is easier than in real life. Although to be consistent, users are able to explore people and books only if they are located in the same city.

1.3 Definitions, Acronyms, and Abbreviations

PostgreSQL - A database management system that is supported with Structured Query Language (SQL).
pgAdmin - PostgreSQL administration and development tool for management of PostgreSQL database.
Spring Boot - A Java framework that enables developers to build enterprise applications.
Web Container - A component for a web server that interacts with servlets.
Tomcat - Open source application for Java Servlet Container.
API (Application Programming Interface) - It is a software intermediary that allows applications to talk to each other.
Google Books API - An API service served by Google that covers information of most of the books that have been written.
Flutter: Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform compatible applications
Dart: The programming language flutter uses.
UI: The user interface that the users of the application will interact with consists of views
View: Pages of the application that is displayed on the device screen.
Material Design: A modern design language that uses mostly grid based layouts
MVVM: Model- View - ViewModel paradigm that facilitates the separation of the development of the graphical user interface and the business logic.

Megabyte - A unit of computer memory or data storage equals 1,084,576 bytes or one million bytes.

1.4 References

- Software Engineering for Internet Applications - 2006 - Eve Andersson, Philip Greenspun, and Andrew Grumet, MIT Press
- Object-oriented Software Engineering: Practical Software - 2001 - Timothy C Lethbridge and Robert Laganiéré
- Requirements Specification Template - Sent From Birol Hoca
- Flutter (https://flutter.dev/)
- Material Design (https://material.io/design)
- Spring Boot (Getting Started | Building an Application with Spring Boot)
- PostgreSQL (PostgreSQL: About)

1.5 Overview

This document briefly gives a high level understanding of how Bookie works. It provides a general idea of how Bookie works, what are the functionalities, who are the users that are going to use Bookie, general constraints and assumptions & dependencies that affects usage of application for users and development for developers in the second section of this document. In the third section of this document specific requirements that are interface related information, functional requirements that explain the mechanism of product with a detailed pattern, non functional requirements that affect product not in a specific context but in a general context of the product, logical database requirements which supported with ER diagram takes place. In the last section, the use cases of our program, objects and their properties & functions, sequence diagram of the project has been covered.

2. General Description

2.1 Product Perspective

2.1.1 System Interfaces

- Bookie is dependent on Google Books API for searching and storing book information when users create their virtual library.

2.1.2 Hardware Interfaces

- It is a cross platform supported application which can run on Android and iOS devices.

2.1.3 User Interfaces

- It is determined that the user interface will include a section for exploring other users' libraries, a section for their profile and a section that allows modifications for their virtual library.

2.1.4 Communication Interfaces

- Communication of users will be handled by two sections. First one is a comment section that users may post comments to a book that a user has. The second one is a section that redirects to by calling phone, sending sms and sending a message through Whatsapp. Communication section may get an upgrade after the first release of application.

2.2 Product Functions

- Login/Registration must be done before accessing all functionalities
- Users can explore other users' virtual libraries in the homepage of the application.
- A user's virtual library is filled with books that are available to exchange. Users can interact with it by clicking the user's name or by directly clicking at the book.
- A book can be chosen to view in the user's virtual library or on the homepage. Book page covers related information of the book and status of the book (Active for trading or passive).
- A book can be requested via a comment or it is possible to get contact with the owner of the book by the communicate button.
- A user can add books to his/her virtual library by choosing the books that exist in Google Books API. It is possible to remove the books that are added to the virtual library by choosing the delete book option.
- Users can update their personal information, the city that they live etc.

2.3 User Characteristics

The users are formed from the people who want to borrow books. This reason might be related to cost because of high inflation, the book they are searching for might be removed from bookstores etc. . People these days prefer getting books from libraries if they have membership but lots of people aren't able to find the books that they are searching for from libraries. Plus, for some locations it is not possible to reach a library because of the distance. That's where our application comes in. Users may borrow books from other people by exploring their virtual libraries. Also they can lend other people their books and get a chance to meet with a bookworm.

2.4 General Constraints

2.4.1 User Related Constraints

In order to run this mobile application on a mobile device, for android devices the **android version must be higher than 5** and for iOS devices the **iOS version must be higher than 9.0**. There should be at most **~50 megabyte free space** on the disk of the device for storage purposes. Like so, our application uses internet connection in order to find a book or get contact with another user. **An internet connection** is required to run the application.

2.4.2 Developer Related Constraints

There are 3 different development sections for this project. Those are mobile, backend and database development sections.

- For database development it is required an installation of **PostgreSQL** database management system and **pgAdmin 4.0** tool for managing database with GUI.
- For backend development it is required an installation of **Java JDK 11 or higher version** of java, **Apache Tomcat 8 or higher version** of Tomcat.
- For mobile development it is required to install **Android Studio** (or similar IDE) **and Flutter SDK (**Flutter 1.26 developer channel**).**

2.5 Assumptions and Dependencies

2.5.1 User Related Assumptions and Dependencies

- It is required to have knowledge about installation of applications to mobile phones and having familiarity with general material design UI concepts such as buttons and forms.

2.5.2 Developer Related Assumptions and Dependencies

- It is required to have knowledge about data manipulation of an SQL database, Java programming language and Spring Boot framework for backend and Dart programming language and Flutter framework for mobile development.
- As told in 2.4.2, developers must have knowledge about installation of all required software applications for development purposes.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface will follow the basic principles of Material design the first interface that the user will see is the login screen. Where the user can either login to an existing account by entering their email and password or press the register button and navigate to the register page.

In the registration page the user needs to provide the: name, email, phone number, password, city after that by clicking the register button the user will be navigated to the home page.

Home page will contain a bottom navigation bar with 3 tabs: the home tab, the library tab and the profile tab.

Home page will contain an infinite scroll view where the users that live in the same city will be displayed in cards where each card will contain the user's name and basic information and also a snapshot of the library of that given user.

Upon clicking one of the cards the application will navigate to user's detailed profile page where all of the books that the user has on the library will be displayed also there will be contact information button with options such as whatsapp, call, sms and each option will open up the

appropriate application. And after the user clicks on a book the application will navigate to book details page.

Book details page will display all of the information about the given book such as author, description, image etc. also each book details page has a public chat room where users can ask questions or talk about that particular book.

On the home page the 2nd tab of the bottom navigation bar is the library page where user can see/edit all of the books on their library. A floating action button on the bottom right of the screen with + icon will appear and upon clicking that button a new page will appear to allow the user to add a new book. To add a new book user will search the name of the book and the application will make a request to our spring backend and after that google books api to get the results for that given query.

After that the matched books will be displayed on a list and the user can click on one of them to select and add to the library.

Last but not least, the profile page which can be accessed from the bottom navigation bar will display the information of the current user such as name, surname, email and city.

3.1.2 Hardware Interfaces

The application can run on Android 5.0+ and ios 9.0+ mobile devices and we anticipate it will not take more than 50 mb of storage. The application does not use any local storage from the device but communicates with our Spring backend using basic networking protocols. Any request sent from a mobile application to the Spring backend will get/modify information from PostgreSQL server. For server requirements it is needed an installation of Java SE 11 or higher version, Apache Tomcat 7.0 or higher for web server container, PostgreSQL database and a tool (pgAdmin 4.0 or higher) to modify database needs.

3.1.3 Software Interfaces

The application can be used by a simple android phone that has an android version of 5.0 or higher that also can access the google play store. Also the application needs internet connection to communicate with the spring backend

3.1.4 Communications Interfaces

All the communication between the backend and the mobile application with a REST Api that the application will send requests that also connection and auth token that the user gets after authenticating to the application.

3.2 Functional Requirements

3.2.1 Login To The Application

3.2.1.1 Introduction / Description

Bookie requires the user to login to the application for the features it provides. For the user to login to the application, the software behind our application must securely check the user's email and password. In addition, while the user is logging into the system, the user's information must be taken from the database correctly, and the user-specific pages must be correct.

3.2.1.2 Inputs / Display

The user shall fill the login form includes user's email and password for log into the application.

3.2.1.3 Processing

- Users shall enter email and password and click the login button.
- An http post request shall be sent to the user-login controller.
- The system shall authenticate the user details.

3.2.1.4 Outputs

The system shall return a message to the user whether the user details are verified or not.

3.2.1.5 Constraints
The user's email and password must match a record in the database.

3.2.1.6 Error/Data Handling

3.2.2 Sign-up For The Application

3.2.2.1 Introduction / Description

Since our application is an application that allows and requires that the user login, users shall register to the system.

3.2.2.2 Inputs / Display

The user shall fill the register form including its name, email, address and password for log into the application.

3.2.2.3 Processing

- Users shall enter name, email, address and password, then click the login button.
- An http post request shall be sent to the user-register controller.
- The system shall check user details for compliance with the constraints.
- The system shall return the status of the register operation as a message.

3.2.2.4 Outputs

A message is given to the user whether the user has successfully registered or not. If the user is registered successfully, the user's homepage appears on the screen.

3.2.2.5 Constraints

- The email that the user enters must be a valid e-mail.
- The password must be at least 6 characters long.

3.2.3 Homepage Feature

3.2.3.1 Introduction / Description

The most important feature of the Bookie is that the user can see the libraries of other users living in the same city. It shall be enabled to use this feature through the homepage.

3.2.3.2 Inputs / Display: -

3.2.3.3 Processing

- The user shall login to the application or click the homepage button.
- An http get request shall be sent to the relevant controller.
- Required data shall be acquired from the relevant tables of the database.

3.2.3.4 Outputs

Users who live in the same city as the user and books in their library shall be displayed on the homepage.

3.2.3.5 Constraints
Only the library of other users who live in the same city of the user should be displayed.

3.2.4 User's Personal Library

3.2.4.1 Introduction / Description

Bookie allows users to create a virtual library of their own books. By giving details of the book, the user shall be allowed to add a book to its personal library. Each user has a virtual library of their own books. This area can be updated by adding a new book and deleting the book.

3.2.4.2 Inputs / Display

- Add book button
- Remove book button

3.2.4.3 Processing

To add a book:

- The user shall click the add button.
- The user shall give the details of its book.
- A book API shall be used to access an image and other details of the book.
- The system shall record the book to the users library table on the database.

To remove a book:

- The user shall click the remove button.
- The user shall give the name of the book.
- The book's with that name shall be displayed to the user by getting the users library and the book tables on the database.
- The user shall select a book.
- The record of that book of the users library shall be removed.

3.2.4.4 Outputs

- A message containing the status of the transaction shall be given to the user.
- Users shall view its own library page.

3.2.4.5 Constraints
-

3.2.5 Viewing Other Users' Libraries and Books

3.2.5.1 Introduction / Description

A user can access other users' libraries and the books in that library. This access shall be provided from the homepage.

3.2.4.2 Inputs / Display

- The button for a user's library (at homepage)
- The button for a user's book (at homepage, the user's library)

3.2.5.3 Processing

To access a user’s library:

- The user shall click the button to a user's library on the homepage.

- With the request sent to the controller, the system obtains the details from the tables on the database. (book, user_books tables)

To access a user’s book:

- The user shall click the button to a user's book on the homepage or that users library

- With the request sent to the controller, the system obtains the details of the books from the database.

3.2.5.4 Outputs

User’s library page containing books and some user information. (accessing a user’s library)

The book details includes title, image, author and description of the book (accessing a user’s book)

3.2.5.5 Constraints

-

3.2.6 Contacting The Book Owners

3.2.6.1 Introduction / Description

In addition to enabling users to change their books, the application shall also prioritize the communication of the users while allowing this. One of the primary steps of the application to fulfill its purpose is to allow its users to communicate. The way users communicate varies as user specific and book specific.

.

3.2.6.2 Inputs / Display

- Contact button on library page for telephone call, sms and whatsapp message

- Contact button on the page of the book page

3.2.6.3 Processing

User specific:

- The user shall click the contact button on the library page.
- The user shall choose one of the options: telephone call, sms and whatsapp message.

Book specific:

- The user shall click the contact button on the book page.
- The user shall write a message.
- The user shall click the send button.
- This message shall be recorded in the comments table of the relevant book.

3.2.6.4 Outputs

Information on whether the message was delivered to the owner successfully or not.

3.2.6.5 Constraints

The book should be available: There shall be information on the book details page that indicates whether the book is available or not. If the book is available, the user should be able to contact the owner of the book.

3.3 Non-Functional Requirements

3.3.1 Performance

- Users will be able to see other user's library information within 1-2 seconds on their main screen.
- Users will be able to add new books to their libraries within 2-3 seconds (due to usage of another API which is Google Books). The search for a book to add to the library will take approximately 1 second because of fetching and rendering issues.
- Users will be able to look at other user's libraries within 1 second for each pageable set. Also the book information related to a user can be seen in less than half a second because fetching and rendering a book's information takes less effort than other requirements.
- Users will be able to send and receive message within approximately 1-2 seconds (because of the chat side of the application will be related with two users and a book will result lots of records to post and fetch)

3.3.2 Reliability

- The database will be updated for each request that comes from the mobile side of the application and if the mobile side of the application is not in maintenance. Database gets an update when a user adds a new book to their library which if the book information is not kept inside the database it will be updated with the information that is coming from Google Books API. Also sending and receiving messages, registering a new user etc. will result in an update in the database.
- If an error occurs or there is any maintenance the user will be notified from the mobile application.

3.3.3 Availability

- System is available to use as long as no maintenance occurs in our backend and Google Books API.

3.3.4 Security

- Users can be registered with their information. Their password is hashed and encrypted.
- Any functionalities that are not registration and login is secured. In order to use them the user should login with a password which is hashed and compared with the user's password in the database. After the login process the mobile part of the application receives a JWT (JSON Web Token) which is a key of communication between user (via mobile side of the application) and API (via backend side of the application.). For each call to the API other than registration and login, JWT must be placed in the "Authorization" header in order to communicate with it.
- User's password should be minimum 6 characters long.
- Users are not able to access nor modify other user's personal information like password, phone, mail etc.

3.3.5 Maintainability

- The implementation will be placed in GitHub in order to share code and modify it all together. After the first release of the implementation it is possible for us to use other

branches than the main branch which gives us flexibility to fix/produce features by not damaging nor modifying the live application.

- Incase of an error to occur on the live application, it is quite easy to fix it in another branch and merge or revert to an old version of the application.

3.3.6 Portability

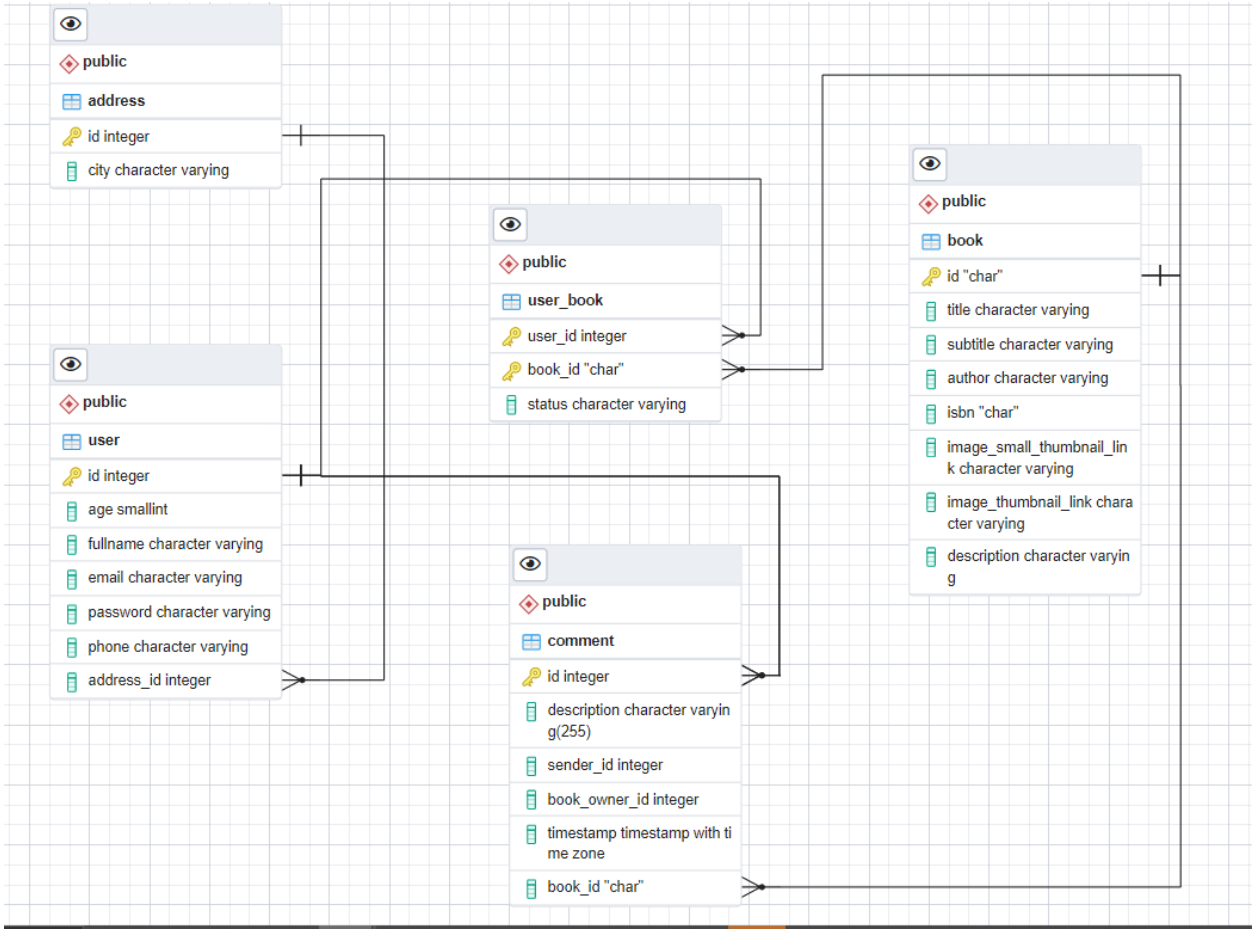
- The application can be installed and run with android devices that have version 5 or higher and iOS devices that have version 9.0 or higher.
- It is foreseen that application will not take storage higher than 50 megabyte.

3.3.7 Localization

- Users will be able to explore other users and their libraries within their city. It is not possible to reach someone that is not in the user's city.

3.6 Logical Database Requirements

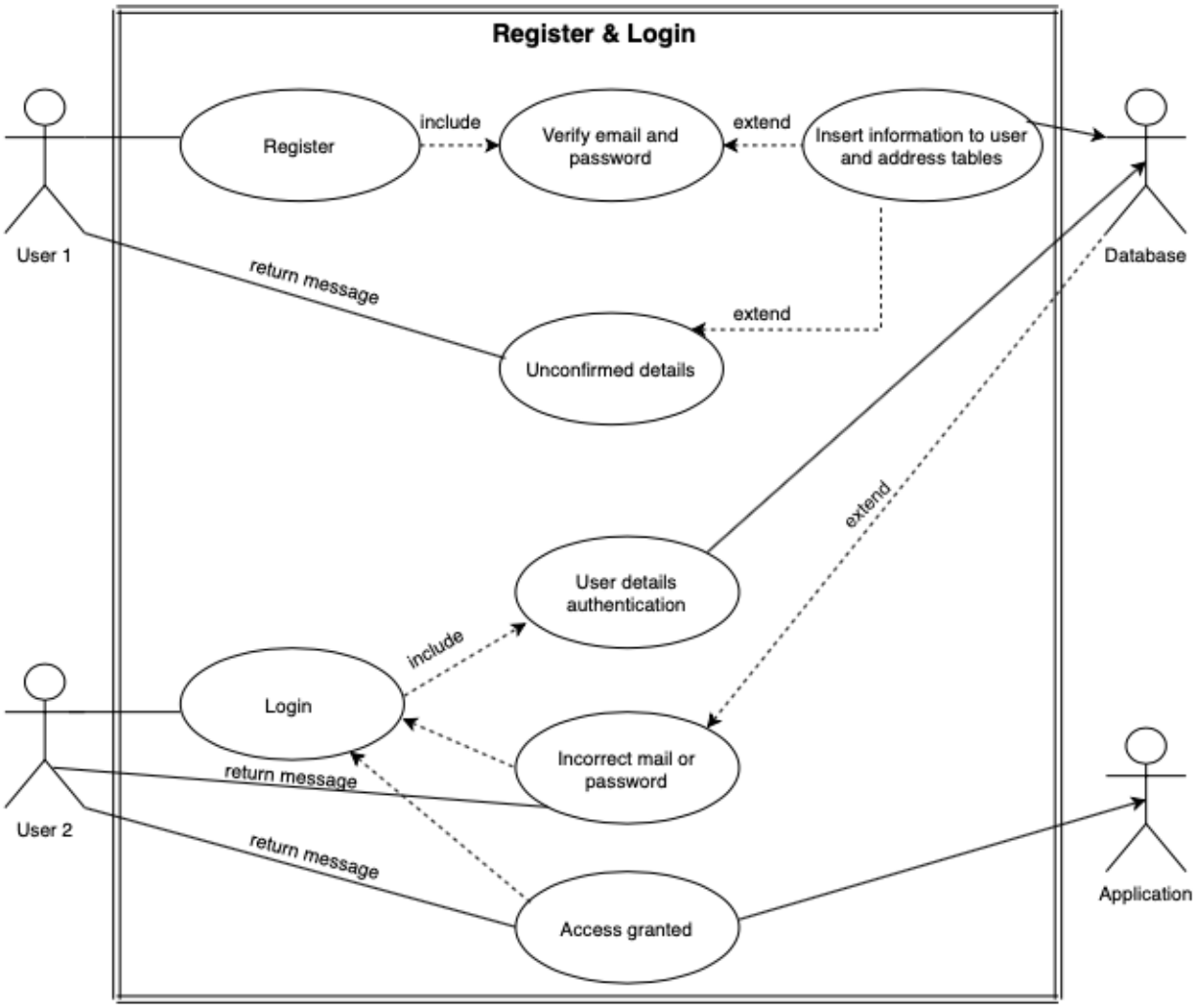
- Our application will use a PostgreSQL database.
- This database will hold information about users, books, comments that are related with books and addresses.
- ER diagram of the project is shown below with necessary data types of columns for each table.



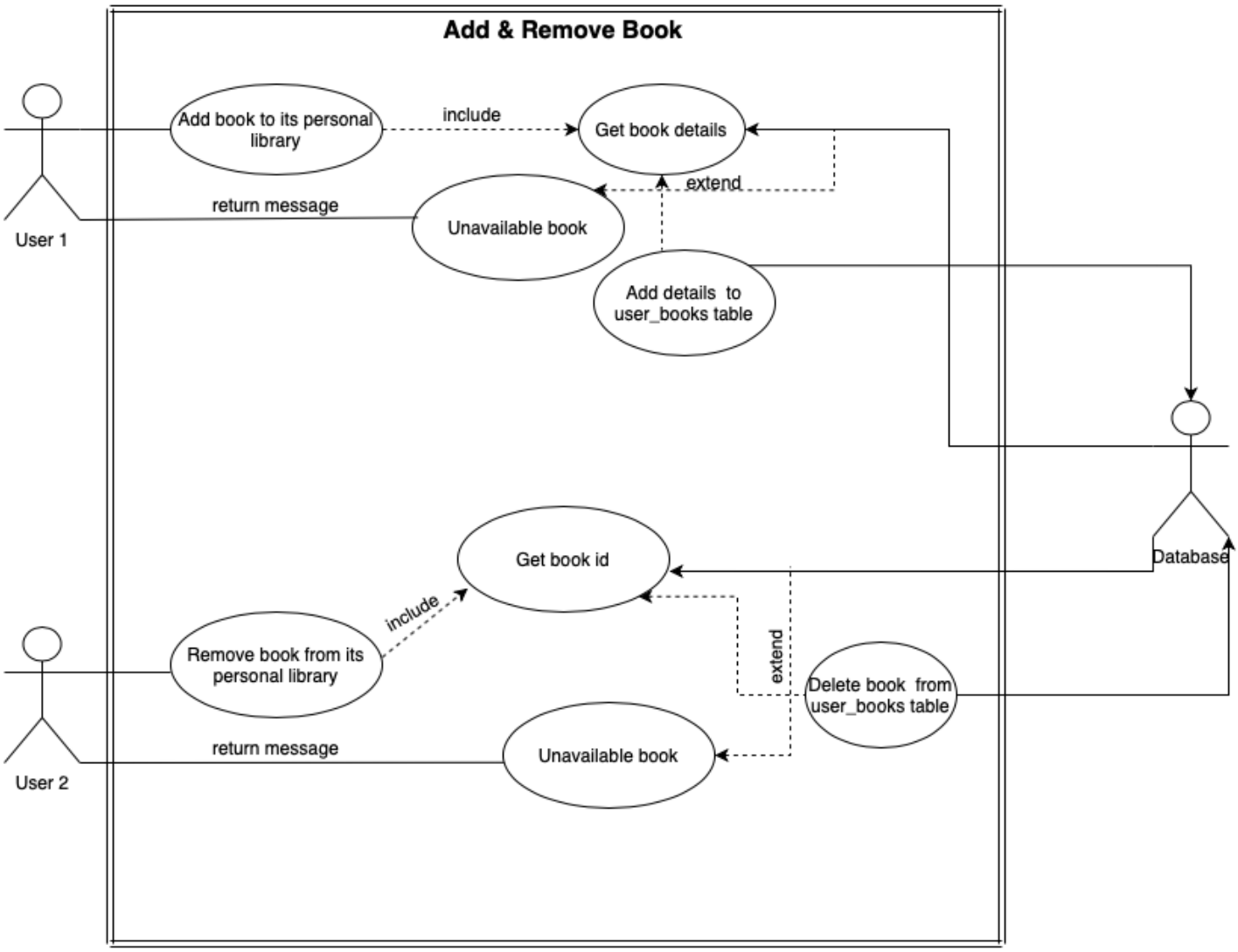
4. Use Case Diagrams

4.1 Use Cases

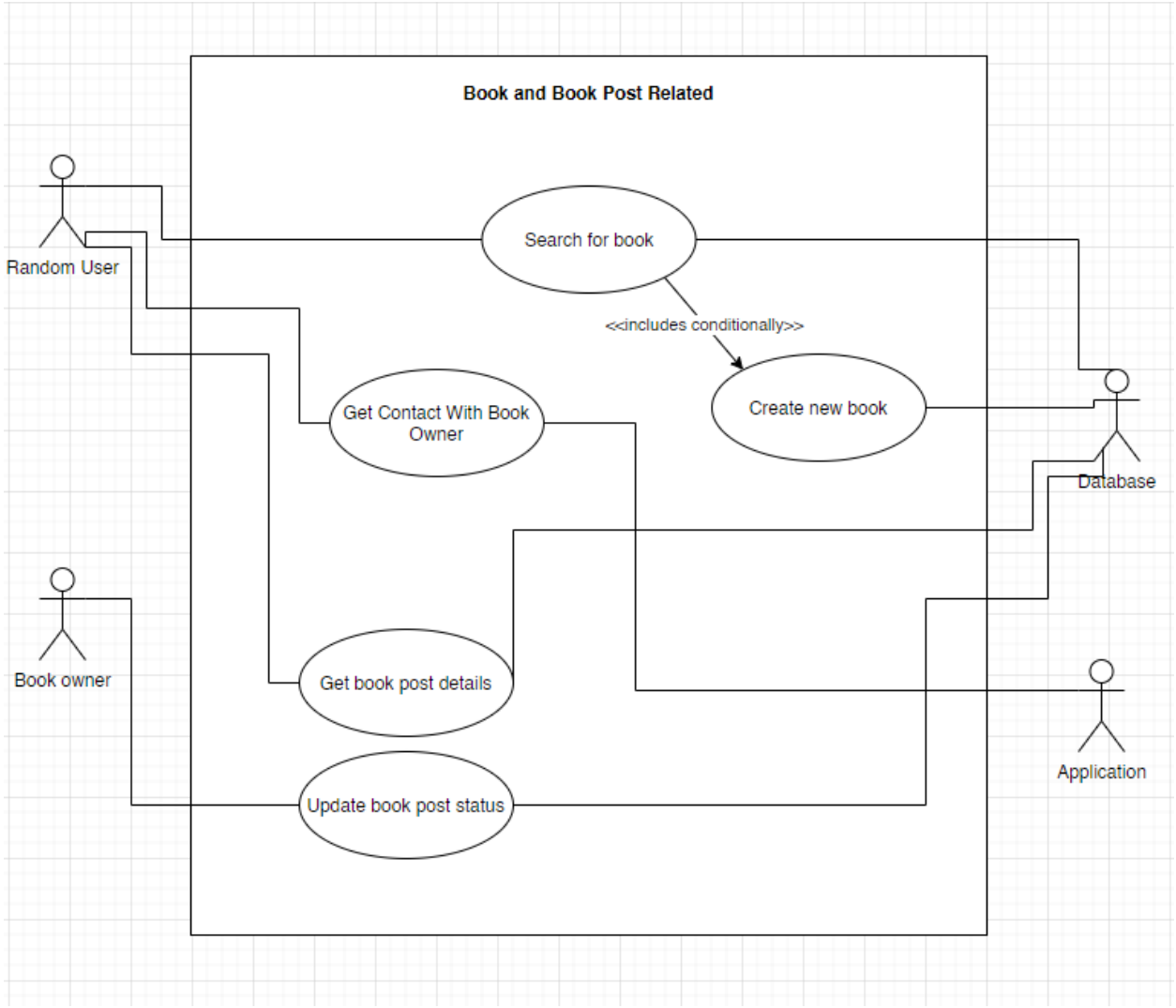
4.1.1 Authorization Related Use Cases



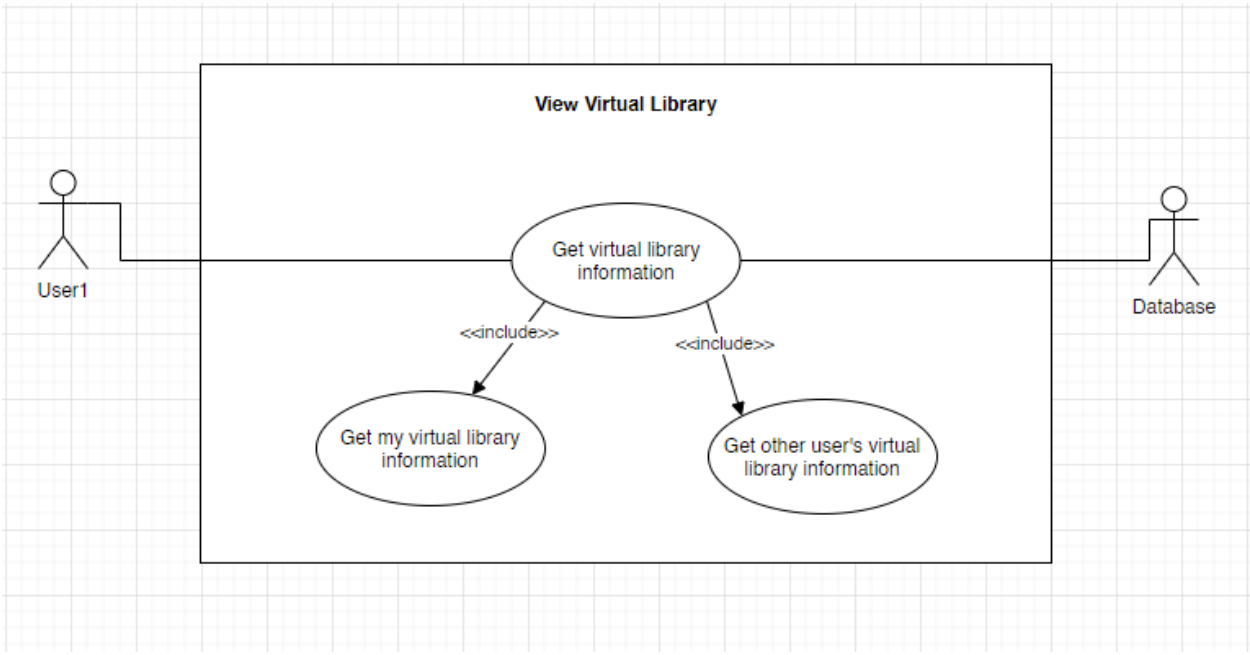
4.1.2 Book and Virtual Library Related Use Cases



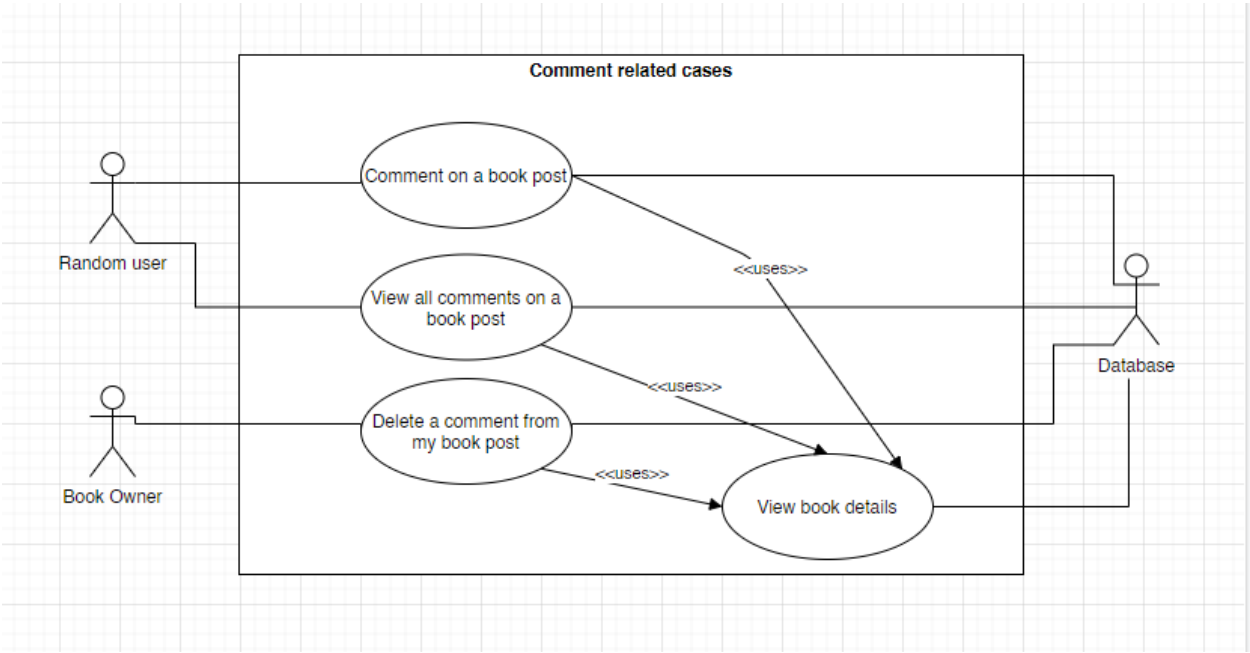
4.1.3 Book and Book Post Related Use Cases

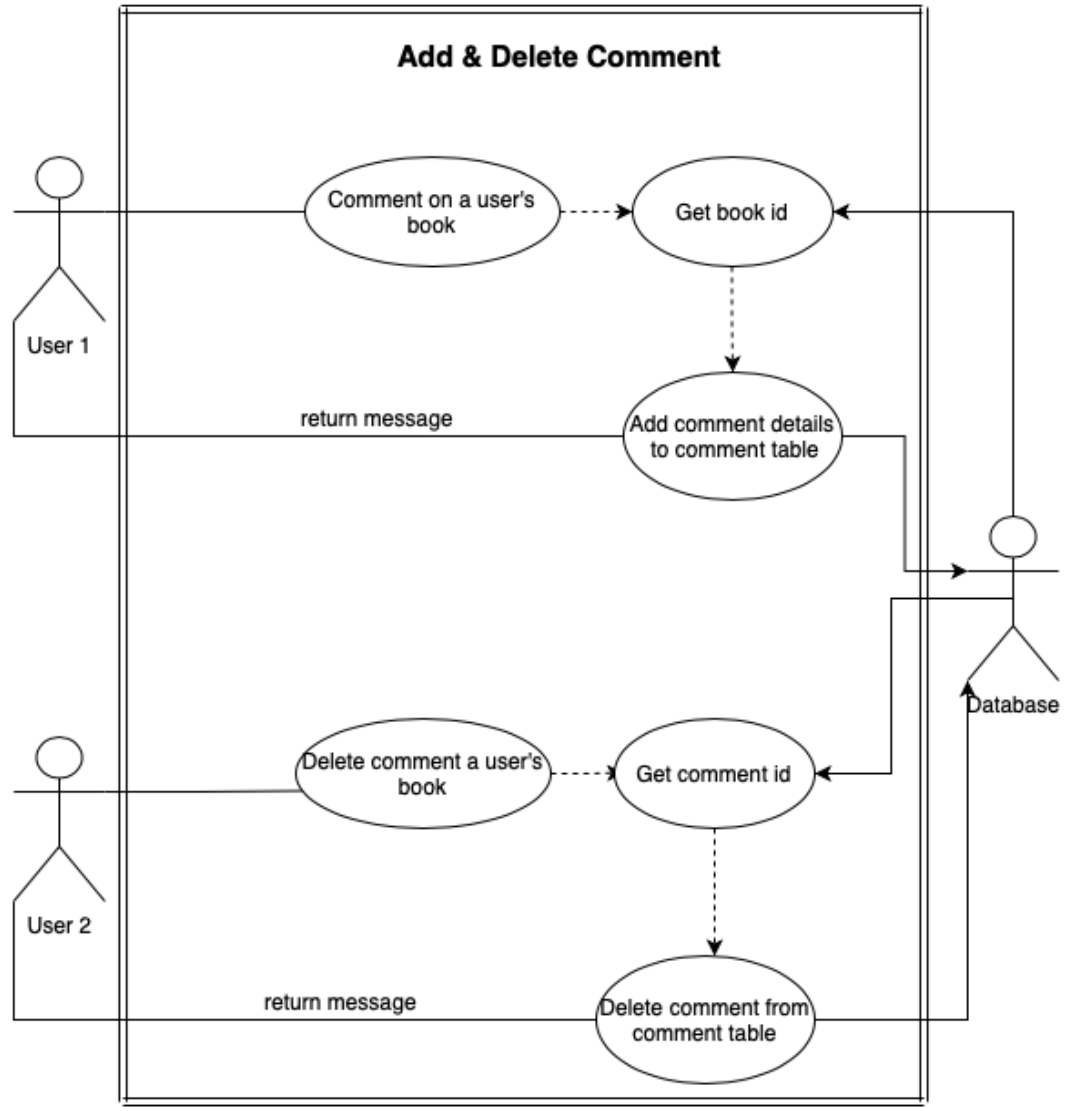


4.1.4 Viewing Virtual Library Related Use Cases

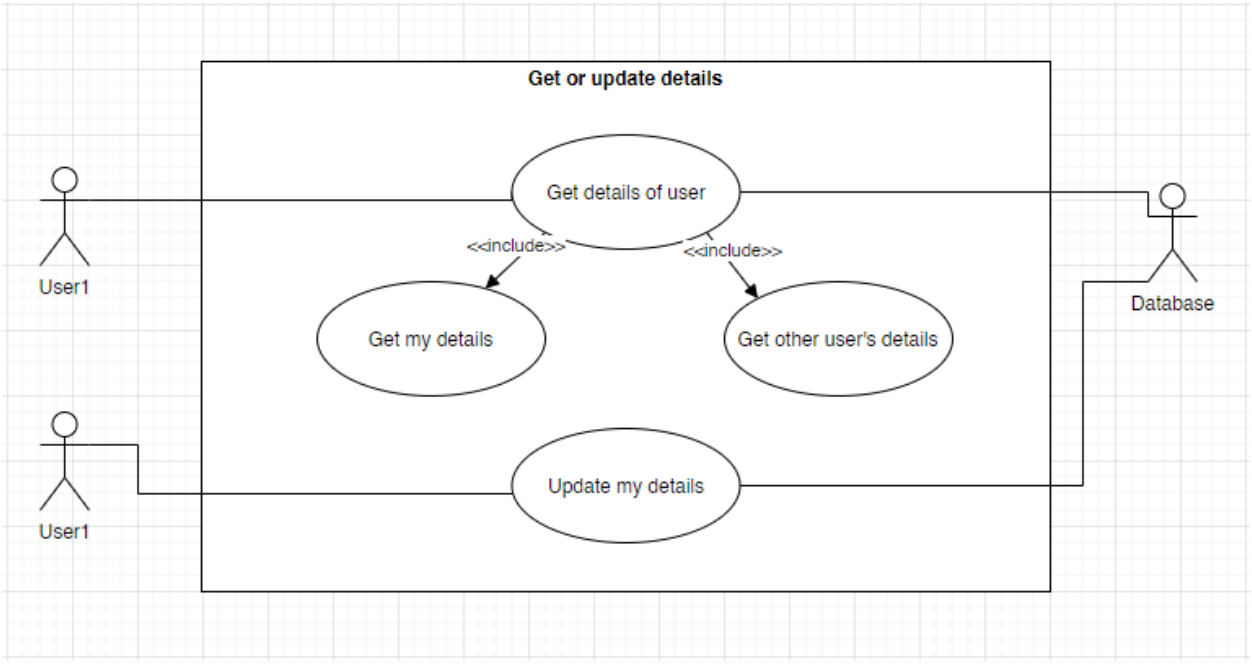


4.1.5 Comment Related Use Cases




Add & Delete Comment
Comment on a user's book
Get book id
User 1
return message
Add comment details to comment table
Database
Delete comment a user's book
Get comment id
User 2
return message
Delete comment from comment table

4.1.6 User Details Related Use Cases



4.2 Classes / Objects

4.2.1 Authentication

4.2.1.1 Attributes

- None (because it is a service)

4.2.1.2 Functions

- **login(params):** This function gets credentials of the user and converts the giving password to a hashed version. Then it compares the given password with the password in the database and if it is matched, it returns a JWT to the user. If credentials are not matched with the information in the database, then simply an error is thrown to the one upper layer (if this function gets placed in the service layer, the error will be thrown to the controller layer which will handle it by itself.) This function is related with **use case 4.1.1** and **3.2.1 of the functional requirements.**
- **register(params):** This function gets user information and after multiple relevance checks, the user will be created with given information. If the relevance checks gone wrong, then simply an error is thrown to the one upper layer. This function is related with **use case 4.1.1** and **3.2.2 of the functional requirements.**
- **verifyAccount(params):** This function checks a given token's valid status and if the token is matched with the created token then the account becomes verified and can use a

login function with no error thrown. This function just enables access to other functionalities after the register mechanic. **use case 4.1.1** and **3.2.2 of the functional requirements.**

4.2.2 User

4.2.2.1 Attributes

- **id:** The identifier of user.
- **fullname:** Full name of the user.
- **email:** The email of the user (should be unique).
- **password:** The hashed version of user's password (security purpose).
- **phone:** Phone number of the user (should be unique).
- **address :** Address object which includes city.
- **age:** Age of the user.

4.2.2.1 Functions:

- **getMyDetails():** Returns the details of the currently authenticated user.
- **updateMyDetails(params):** Updates currently authenticated user's information. This function is related with **use case 4.1.6**
- **getMyLibrary():** Returns the pageable list of currently authenticated user's virtual library that includes books.This function is related with **use case 4.1.4 and 3.2.4 of the functional requirements.**
- **getUserDetails(params):** Returns the details of the specified user.This function is related with **use case 4.1.6**
- **getUserLibrary(params):** Returns the pageable list of specified user's virtual library that includes books. This function is related with **use case 4.1.4 and 3.2.4 of the functional requirements.**
- **getOtherUsersWithLibraries():** Based on the city of the currently authenticated user, random users and their virtual libraries that contain books are returned.This function is related with **use case 4.1.4 and 3.2.5 of the functional requirements.**
- **addNewBookToTheLibrary(params):** Adds a new specified book with it's id to the virtual library to the currently authenticated user. Throws error if the book is already contained by the user's virtual library or the book with that id does not exist.This function is related with **use case 4.1.2** and **3.2.4 of the functional requirements.**
- **removeBookFromTheLibrary(params):** Removes a specified book with it's id from the virtual library of the currently authenticated user. Throws error if the book is not contained by the user's virtual library or the book with that id does not exist.This function is related with **use case 4.1.2** and**3.2.4 of the functional requirements.**
- **Getter/Setter functions for attributes.**

4.2.3 Book

4.2.3.1 Attributes

- **id:** The identifier of the book (same with the identifier of the books in Google Books).
- **title:** The title of the book.
- **subtitle:** The subtitle of the book.
- **author:** The author of the book.
- **isbn:** The ISBN 13 of the book (should be unique).
- **image_small_thumbnail_link:** The link for the book's small thumbnail image.

- **image_thumbnail_link:** The link for the book's thumbnail image.

4.2.3.2 Functions

- **searchForBook(params):** Searches a book with book credentials and returns a list of best matched ones from Google Books API.This function is related with **use case 4.1.3** and **3.2.5 of the functional requirements.**
- **getBookDetails(params):** Gets book details on someone's virtual library.This function is related with **use case 4.1.2, use case 4.1.3** and **3.2.5 of the functional requirements.**
- **createNewBook(params):** Creates and stores the newly created book. This function will be called in addNewBookToTheLibrary function when the selected book is not contained by the PostgreSQL database.This function is related with **use case 4.1.2, use case 4.1.3** and **3.2.4 of the functional requirements.**
- **updateStatus(params):** Updates the status of a book in a currently authenticated user's virtual library. If the book specified does not exist or that book is not owned by the authenticated user, the function will throw an error to the upper layer. This function is related to **use case 4.1.3.**
- **Getters/Setters for attributes.**

4.2.4 Comment

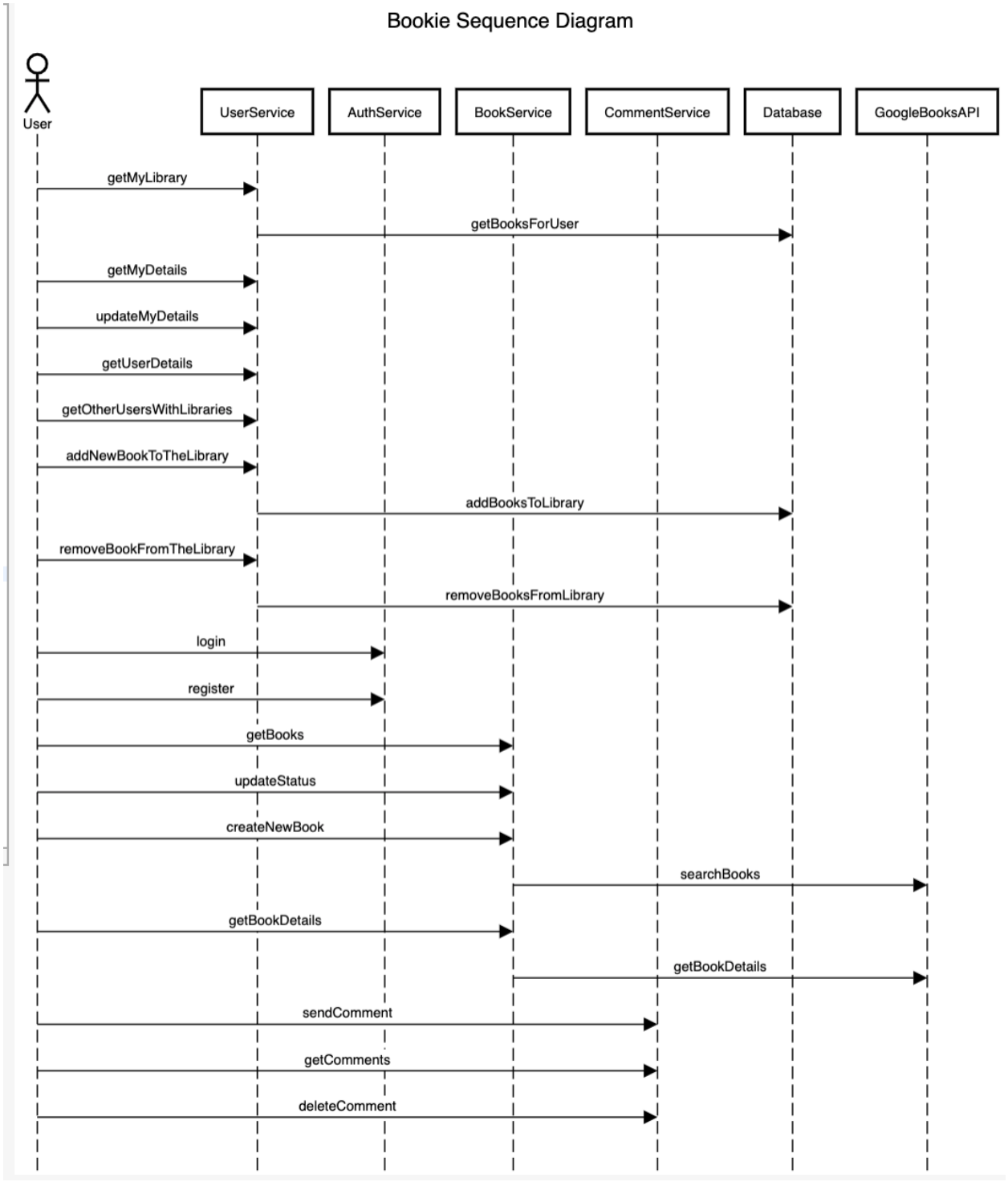
4.2.4.1 Attributes

- **id:** Identifier of the comment.
- **description:** Description of the comment.
- **sender:** Sender of the book shaped as a user object.
- **bookOwner:** Book owner as a user object.
- **book:** Book object of which comment belongs.
- **timestamp:** Timestamp of the comment.

4.2.4.2 Functions

- **sendComment(params):** Sends a comment to a book on some other user's virtual library from the currently authenticated user.This function is related with **use case 4.1.5** and **3.2.6 of the functional requirements.**
- **getComments(params):** Gets list of comments belongs to a book on some other user's virtual library.**use case 4.1.5** and **3.2.6 of the functional requirements.**
- **deleteComment(params):** Deletes a comment of a book on currently authenticated users virtual library. Throws error if the comment does not exist or the comment does not belong to a book that is owned.**use case 4.1.5** and **3.2.6 of the functional requirements.**
- **Getters/Setters for attributes.**

4.3 Sequence Diagrams



5) Contributions

	Ahmet Akıl	Talha Bayburtlu	Zeynep Alıcı
1. INTRODUCTION			
1.1 Purpose		✓	
1.2 Scope	✓	✓	✓
1.3 Definitions, Acronyms, and Abbreviations	✓	✓	
1.4 References	✓	✓	
1.5 Overview		✓	
2. GENERAL DESCRIPTION			
2.1 Product Perspective		✓	
2.2 Product Functions		✓	
2.3 User Characteristics		✓	
2.4 General Constraints	✓	✓	
2.5 Assumptions and Dependencies		✓	
3. SPECIFIC REQUIREMENTS			
3.1 External Interface Requirements	✓		
3.1.1 User Interfaces	✓		

3.1.2 Hardware Interfaces	✓		
3.1.3 Software Interfaces	✓		
3.1.4 Communications Interfaces	✓		
3.2 Functional Requirements			✓
3.2.1 Login To The Application			✓
3.2.2 Sign-up For The Application			✓
3.2.3 Homepage Feature			✓
3.2.4 User's Personal Library			✓
3.2.5 Viewing Other Users' Libraries and Books			✓
3.2.6 Contacting The Book Owners			✓
3.3 Non-Functional Requirements		✓	
3.3.1 Performance		✓	
3.3.2 Reliability		✓	
3.3.3 Availability		✓	
3.3.4 Security		✓	
3.3.5 Maintainability		✓	
3.3.6 Portability		✓	
3.3.7 Localization		✓	

3.6 Logical Database Requirements		✓	
4. USE CASE DIAGRAMS			
4.1 Use Cases			✓
4.1.1 Authorization Related Use Cases			✓
4.1.2 Book and Virtual Library Related Use Cases			✓
4.1.3 Book and Book Post Related Use Cases		✓	
4.1.4 Viewing Virtual Library Related Use Cases		✓	
4.1.5 Comment Related Use Cases		✓	✓
4.1.6 User Details Related Use Cases		✓	
4.2 Classes / Objects		✓	✓
4.2.1 Authentication		✓	
4.2.2 User	✓	✓	✓
4.2.3 Book		✓	✓
4.2.4 Comment		✓	✓
4.3 Sequence Diagrams	✓		