Online Social Bookstore Application

Sprint Report

Ομάδα 4935-4967-5085

Χρήστος Μπίτσιος ΑΜ 4935

Γεώργιος Θεοδωρόπουλος ΑΜ 4967

Δημήτριος Γώγος ΑΜ 5085

VERSIONS HISTORY

Date	Version	Description	Author
28/02/2024	0.1	Setting up Entity Classes and DB	4935-4967-5085
03/03/2024	1.0	Adding functionalities for user and user profile	4935-4967-5085
07/03/2024	1.0.1	Fixing bugs in functionalities	4935-4967-5085
17/03/2024	2.0	Adding functionalities for book ,bookcategory,bookauthor,bookoffer and bookrequest	4935-4967-5085
20/03/2024	2.0.1	Fixing bugs in functionalities	4935-4967-5085
29/03/2024	3.0	Adding services for usercontroller and authcontroller	4935-4967-5085
10/04/2024	3.0.1	Fixing bugs in service	4935-4967-5085
20/04/2024	4.0	Adding mores services , implementing Services and adding search strategy	4935-4967-5085
01/05/2024	4.0.1	Fixing bugs in service and strategy	4935-4967-5085
05/05/2024	5.0	Adding UI login functionality and multiple user support	4935-4967-5085
10/05/2024	6.0	Testing Implementation	4935-4967-5085
15/05/2024	6.0.1	Fixing bugs found	4935-4967-5085
18/05/2024	6.0.2	Fixing more bugs	4935-4967-5085
20/05/2024	6.0.3	Final Version of project	4935-4967-5085

1 Introduction

This document provides information concerning the **<X>** sprint of the project.

1.1 Purpose

To provide users of the application a way to manage their books and find the ones they like through the search in the online bookstore through various parameters such as the author, the category, etc.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies this Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team.

Product Owner	4935-4967-5085
Scrum Master	4935-4967-5085
Development Team	4935-4967-5085

2.2 Sprints

<List below the sprints that you performed and the user stories that have been realized in each Sprint>

Sprint No	Begin Date	End Date	Number of weeks	User stories
	26/02/2024	**02/03/2024**	**1**	Create Database and Required Tables
	02/03/2024	**16/03/2024**	**2**	Add Entities for the project
	16/03/2024	**30/03/2024**	**2**	Add services for UserController
	30/03/2024	**06/04/2024**	**1**	Add more services for UserController and search strategy
	06/04/2024	**13/04/2024**	**1**	Make login page for user
	13/04/2024	**20/04/2024**	**1**	Testing

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

3.1 CreateAccount

Use case ID	1
Actors	User
Pre conditions	1. The account doesn’t exist.
Main flow of events	1. The use case starts when the user starts the webpage. 2. The user is requested to create a new account.
Post conditions	The user has created a new account to log in the online social bookstore app.

3.2 LoginPage

Use case ID	2
Actors	User
Pre conditions	Knowing his username and password to login.
Main flow of events	1. The use case starts when the user starts the webpage. 2. The user is requested to provide his username and his password. 3. The system checks if username and password provided are correct. 4. If username and password are correct, the system returns the books of the user who logged in
Alternative flow	If username/password are incorrect, the system shows a message “Bad credentials”, letting the user know that the credentials he provided are incorrect.

Post conditions	The user can manage his books safely.

3.3 Logout

Use case ID	3
Actors	User
Pre conditions	User must have successfully logged in.
Main flow of events	1. The use case begins when the user decides to logout from their account. 2. The system terminates the user session and redirects them to the logout page.
Alternative flow	If the user decides not to logout, they can navigate away from the logout
Post conditions	The user can no longer access any functionalities that require them to be logged in.

3.4 CreateProfile

Use case ID	4
Actors	User
Pre conditions	User must have successfully logged in.
Main flow of events	1. The use case begins when the user clicks the button “Create Profile ” button. 2. The system shows the user profile ,including full name, address, age, phone number, preferred book categories, and favorite authors.

Post conditions	User has created his profile in online social bookstore.

3.5 AddBook

Use case ID	5
Actors	User
Pre conditions	User must have successfully logged in.
Main flow of events	1. The use case starts when the user clicks at the “Add Book” button 2. The system provides the user with a form to fill with information about the book (Title/Author/Category/Summary) 3. The user fills the form with the necessary information and the clicks the “Save” button. 4. The system returns the list of books with the new book added.
Alternative flow	If the user clicks the "Back to Book " link instead of adding a new book , they are redirected to the list of book offers without making any changes
Post conditions	A new book is added in the list

3.6 MadeOffer

Use case ID	6
Actors	User
Pre conditions	1.User must have successfully logged in
Main flow of event	1. The use case starts when the user makes a book offer. 2. The system show a list of book offers made by the user. 3. The user selects a specific book offer from his list. 4. The system show a list of requests from other users who are interested in the book offer.
Alternative Flow	If the user decides not to give the book to any of the requesting users, they can choose to keep the book offer
Post conditions	User can manage the requests for his book offers

3.7NotifyUser

Use case ID	7
Actors	User
Pre conditions	User must have successfully logged in and keep a list of the users who have requested a specific book .
Main flow of events	1.The use case starts when the user selects a user from the list to whom they want to offer the book 2. The user sends a notification to the selected user, informing them that the book is available.
Alternative flow	If there are no users who have requested the book, the user keeps the book ,without notification.

Postconditions	User notifies with the selected user to take the book.

3.8ContactUser

Use case ID	8
Actors	User
Pre conditions	User must have successfully logged in and keep a list of the users who have requested a specific book .
Main flow of events	1.The use case starts when the user selects a user from the list to whom they want to offer the book. 2. The user contacts the selected user to arrange the delivery of the book.
Alternative flow	If there are no users who have requested the book, the user keeps the book without contact.
Postconditions	The user contacts with the selected user to arrange the delivery of the book

3.9 DeleteBook

Use case ID	9
Actors	User
Pre conditions	User must have successfully logged in and select a book from his list of offers.
Main flow of events	1. The use case starts when the user clicks at the “Delete” button . 2. The system shows an alert to confirm the deletion of the book . 2.1. If user clicks “OK”, the book is removed from the list.

	2.2. If user clicks “Cancel”, nothing happens
Postconditions	1.The selected book is removed from the user's personal list of book offers. 2. The book is removed from the list of book requests of other users who requested it

3.10SearchStrategy

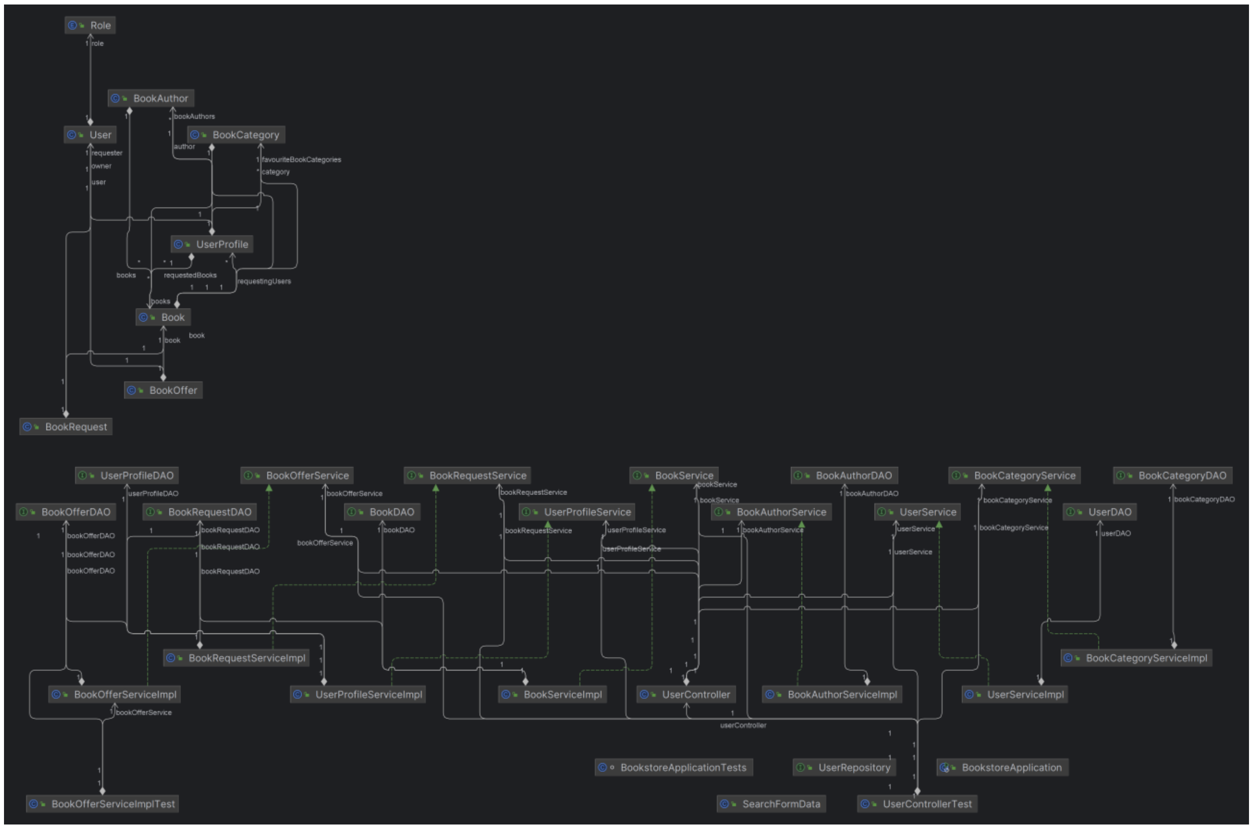
Use case ID	10
Actors	User
Pre conditions	1.User must have successfully logged in.
Main flow of event	1.The use case starts when the user search for book offers. 2. The user specifies the search criteria (Title/Author/ Exact or Approximate) 3. The user selects the book from the search results. 4. The user sends a request for the selected book.
Alternative flow	If the user search with no results, then changes his search criteria and repeat the process again.
Post conditions	User can search for book offers based on specified criteria.

3.11RecommendationsStrategy

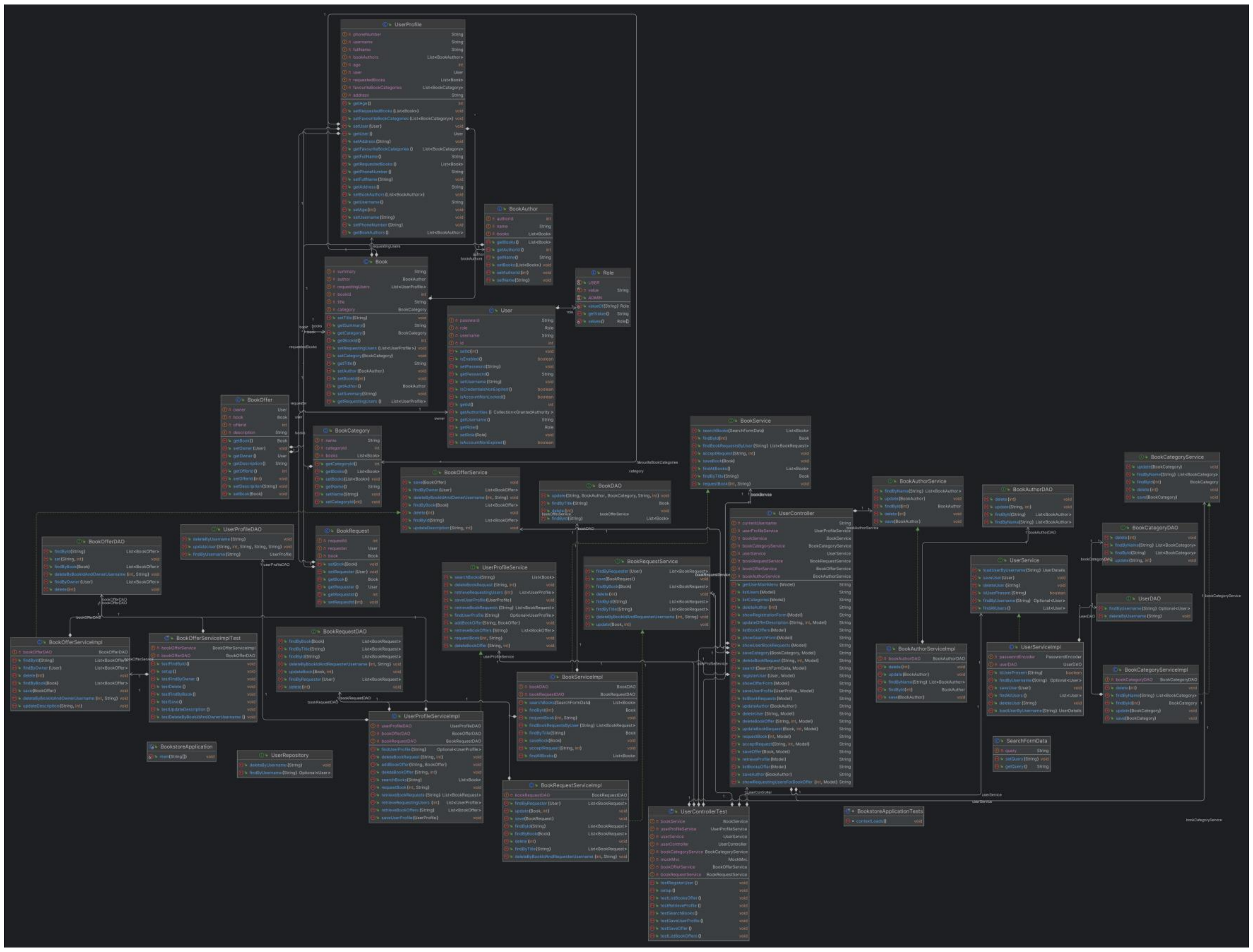
Use case ID	11
Actors	User
Pre conditions	User must have successfully logged in.
Main flow of event	1.The use case starts when the user accesses the list of recommended book offers 2. The user selects a recommendation strategy.
Alternative flow	If the recommended list is empty, the user chooses a different recommendation strategy.
Post conditions	The user selects a recommended book offer.

4 Design

4.1 Architecture



4.2 Design


UserProfile
BookAuthor
Book
Role
User
BookOffer
BookCategory
BookService
BookOfferService
BookDAO
UserController
BookAuthorService
BookAuthorDAO
BookCategoryService
BookCategoryDAO
UserProfileDAO
BookOfferDAO
BookRequest
UserProfileService
BookRequestService
UserService
UserDAO
BookOfferServiceImpl
BookOfferServiceImplTest
BookRequestDAO
BookServiceImpl
BookAuthorServiceImpl
UserServiceImpl
BookCategoryServiceImpl
UserProfileServiceImpl
SearchFormData
BookstoreApplication
UserRepository
BookRequestServiceImpl
UserControllerTest
BookstoreApplicationTests

Class Name: **BookstoreApplication**	
Responsibilities: ▪ Main Class, starts the SpringBoot Application	**Collaborations:**

Class Name: SecurityConfig	
Responsibilities: ▪ Secure login	**Collaborations:**

Class Name: UserController	
Responsibilities: ▪ Controls the UI ▪ Shows the information the user requests	**Collaborations:** ▪ UserService ▪ UserProfileService ▪ BookService ▪ BookCategoryService ▪ BookOfferService ▪ BookRequestService ▪ BookAuthorService

Class Name: UserService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ UserServiceImpl

Class Name: UserServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ UserService ▪ UserDAO

Class Name: UserProfileService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ UserProfileService

Class Name: UserProfileServiceImpl	
Responsibilities: ▪ Implements the Interface	**Collaborations:** ▪ UserProfileService

▪ Business Logic of the application	▪ UserProfileDAO

Class Name: BookService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ BookService ▪ BookRequest

Class Name: BookServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ BookService ▪ BookDAO ▪ BookRequestDAO
Class Name: BookCategoryService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ BookCategory

Class Name: BookCategoryServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ BookService ▪ BookCategoryDAO

Class Name: BookAuthorService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ BookAuthor

Class Name: BookAuthorServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ BookAuthorService ▪ BookAuthorDAO

Class Name: BookOfferService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ BookOffer

Class Name: BookOfferServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ BookOfferService ▪ BookOfferDAO

Class Name: BookRequestService(Interface)	
Responsibilities: ▪ Provides an Interface for the User controller	**Collaborations:** ▪ UserController ▪ BookRequest

Class Name: BookRequestServiceImpl	
Responsibilities: ▪ Implements the Interface ▪ Business Logic of the application	**Collaborations:** ▪ BookRequestService ▪ BookRequestDAO

Class Name: BookDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByTitle/FindById/Update/Delete	**Collaborations:** ▪ BookServiceImpl

Class Name: BookCategoryDAO(Interface)	
Responsibilities: ▪ Interacts with the DB	**Collaborations:** ▪ BookCategoryServiceImpl

▪ FindByName/FindById/Update/Delete	

Class Name: BookAuthorDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByName/FindById/Update/Delete	**Collaborations:** ▪ BookAuthorServiceImpl

Class Name: BookOfferDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByBook/FindById/FindByOwner/DeleteByBookIdAndOwnerUsername/Delete	**Collaborations:** ▪ BookOfferServiceImpl

Class Name: BookRequestDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByBook/FindByRequester/FindByTitle/FindById/UpdateBook/Delete/DeleteByBookIdAndRequesterUsername	**Collaborations:** ▪ BookRequestServiceImpl

Class Name: UserProfileDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByUserName/UpdateUser/DeleteByUsername	**Collaborations:** ▪ UserProfileServiceImpl

Class Name: UserDAO(Interface)	
Responsibilities: ▪ Interacts with the DB ▪ FindByUserName/DeleteByUsername	**Collaborations:** ▪ UserServiceImpl

Class Name: Book	
Responsibilities: ▪ Make Objects from the database table book. ▪ Entity Class	**Collaborations:** ▪ BookAuthor ▪ BookCategory ▪ UserProfile ▪ UserController

Class Name: BookAuthor	
Responsibilities: ▪ Make Objects from the database table bookauthor. ▪ Entity Class	**Collaborations:** ▪ Book ▪ UserController

Class Name: BookCategory	
Responsibilities: ▪ Make Objects from the database table bookcategory. ▪ Entity Class	**Collaborations:** ▪ Book ▪ UserController

Class Name: BookOffer	
Responsibilities: ▪ Make Objects from the database table bookoffer. ▪ Entity Class	**Collaborations:** ▪ Book ▪ User ▪ UserController

Class Name: BookRequest	
Responsibilities: ▪ Make Objects from the database table bookrequest. ▪ Entity Class	**Collaborations:** ▪ Book ▪ User ▪ UserController

Class Name: Role	
Responsibilities: ▪ Entity Class	**Collaborations:** ▪ User ▪ UserController

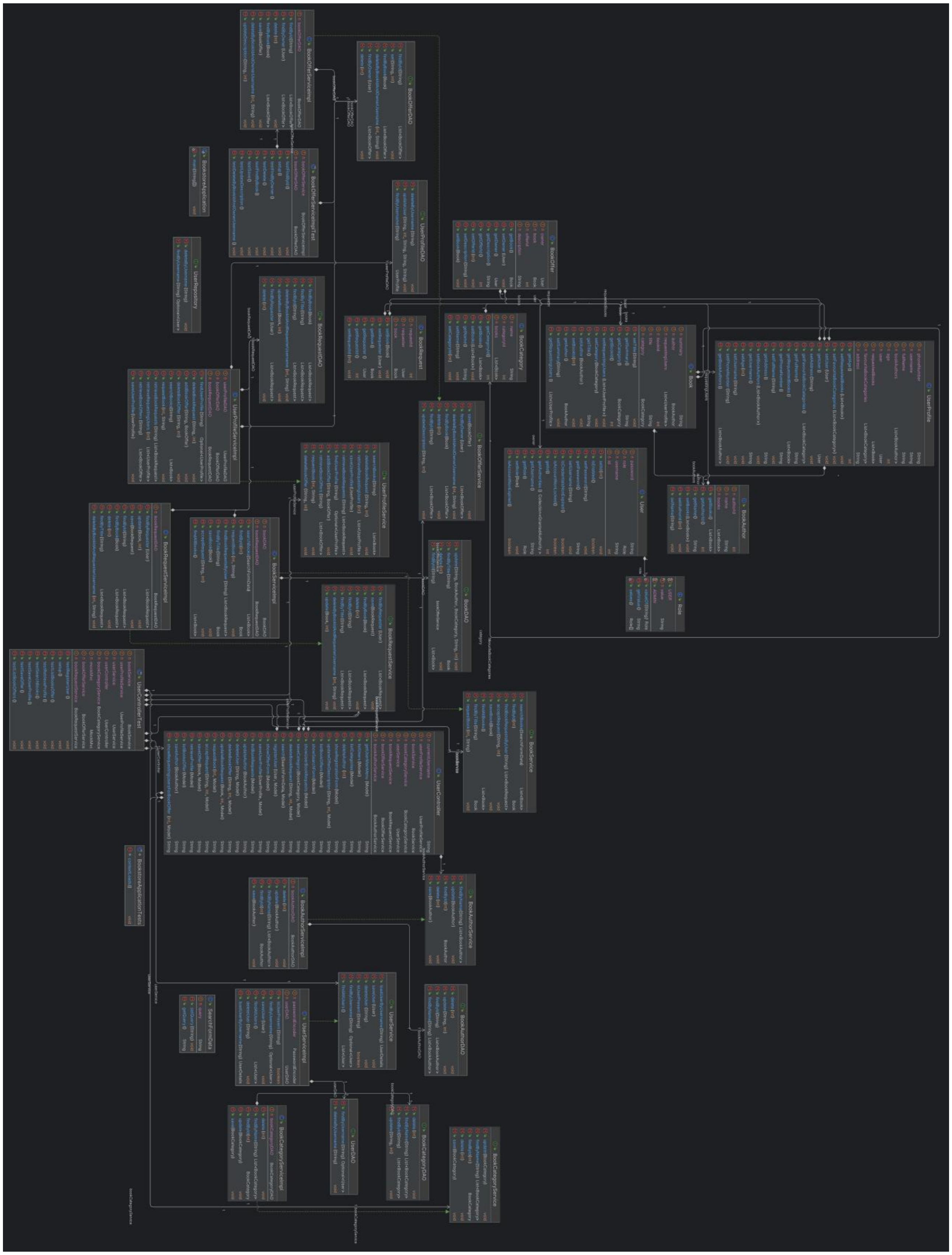
Class Name: UserProfile	
Responsibilities: ▪ Make Objects from the database table userprofile. ▪ Entity Class	**Collaborations:** ▪ Book ▪ BookCategory ▪ BookAuthor ▪ User ▪ UserController

Class Name: User	
Responsibilities: ▪ Make Objects from the database table user. ▪ Entity Class	**Collaborations:** ▪ Role ▪ UserDetails ▪ UserController

Class Name: SearchFormData	
Responsibilities: ▪ Controls SearchStrategy	**Collaborations:** ▪ UserController

Comments

We did not complete the ReccomendationsFormData class. The securityconfig is in comments because we had a problem with the functions used via SpringBoot Web Security, this is also the problem why the app is running in the Log In html and not in the main-menu. Furthermore, The html named signin and sing up in the directory named auth in templates are responsible for showing the Users log in and register but there was a problem with the name, but the html keep doing the work properly.


BookOfferServiceImpl
BookOfferDAO
BookstoreApplication
BookOfferServiceImplTest
UserProfileDAO
UserRepository
BookOffer
BookRequestDAO
BookCategory
BookRequest
Book
UserProfile
UserProfileServiceImpl
UserProfileService
BookOfferService
User
BookAuthor
BookRequestServiceImpl
BookServiceImpl
BookRequestService
BookDAO
Role
UserControllerTest
UserController
BookService
BookstoreApplicationTests
BookAuthorServiceImpl
BookAuthorService
UserService
UserServiceImpl
SearchFormData
BookAuthorDAO
BookCategoryServiceImpl
UserDAO
BookCategoryDAO
BookCategoryService