Online Social Bookstore Application

Sprint Report

<Fill here the Team Name>
Ορφανίδης Παύλος 4134, Θανάσης Τζαβαλάς 4221

VERSIONS HISTORY

Date	Version	Description	Author
20/05/2024	0.0.1	Created a first version of a working application	Ορφανίδης Παύλος 4134 Θανάσης Τζαβαλάς 4221

1 Introduction

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

1.1 Purpose

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the 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	Ορφανίδης Παύλος 4134 Θανάσης Τζαβαλάς 4221
Scrum Master	Ορφανίδης Παύλος 4134 Θανάσης Τζαβαλάς 4221
Development Team	Ορφανίδης Παύλος 4134 Θανάσης Τζαβαλάς 4221

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
1	**03/05/2024**	**07/05/24**	**1**	**US1, US2, US3**
2	**07/05/24**	**14/05/24**	**1**	**US4, US5, US6**
3	**14/05/24**	**20/05/24**	**1**	**US7, US8, US9, US10**

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**.>

Use Case ID	US1 User Register
Actors	User
Preconditions	User is not logged in. User has internet access.
Main flow of events	1. The use case starts when the user navigates to the sign-up page. 2. The user fills in the registration form with their details (full name, email, password, etc.). 3. The user submits the registration form. 4. The system validates the input data. 5. The system creates a new user account. 6. The system displays a confirmation message and redirects the user to the login page.
Alternative flow 1	3.1. If the user enters invalid data, the system displays an error message and prompts the user to correct the input.
Post conditions	The user account is created, and the user can now log in.

Use Case ID	US2 Log In
Actors	User
Preconditions	User has an existing account. User is not logged in. User has internet access.
Main flow of events	1. The use case starts when the user navigates to the login page. 2. The user enters their email and password. 3. The user submits the login form. 4. The system validates the login credentials. 5. The system logs the user in and redirects them to the home page.
Alternative flow 1	3.1. If the user enters incorrect credentials, the system displays an error message and prompts the user to re-enter the credentials.
Post conditions	The user is logged in and can access the application's functionalities.

Use Case ID	US3 Log Out
Actors	User
Preconditions	User is logged in.
Main flow of events	1. The use case starts when the user clicks the logout button. 2. The system logs the user out. 3. The system redirects the user to the login page.
Post conditions	The user is logged out, and the session is terminated.

Use Case ID	**US4 Profile**
Actors	User
Preconditions	User is logged in.
Main flow of events	1. The use case starts when the user navigates to the profile page. 2. The user fills in the profile form with their details (full name, address, age, phone number, preferred book categories, favorite authors). 3. The user submits the profile form. 4. The system validates the input data. 5. The system creates the user profile. 6. The system displays a confirmation message and redirects the user to their profile page.
Alternative flow 1	
Post conditions	The user's profile is created and stored in the system.

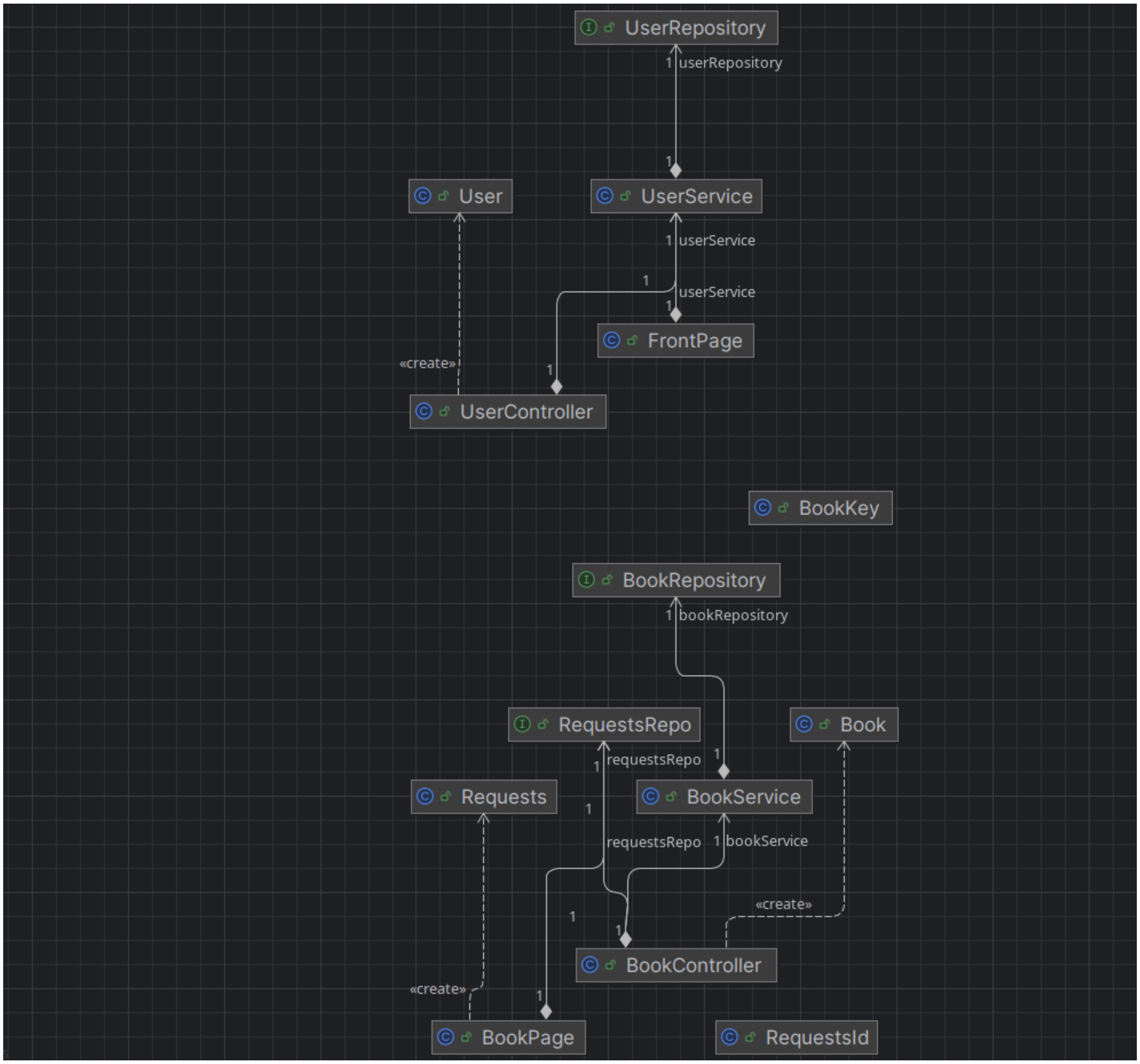
Use Case ID	**US5 Add Book**
Actors	User
Preconditions	User is logged in.
Main flow of events	1. The use case starts when the user navigates to the add book offer page. 2. The user fills in the book offer form with details (title, author, category, summary). 3. The user submits the book offer form. 4. The system adds the book offer to the user's list of book offers. 5. The system displays a confirmation message and updates the book offers list.
Alternative flow 1	3.1. If the user enters invalid data, the system displays an error message and prompts the user to correct the input.
Post conditions	The book offer is added to the user's list of book offers.

Use Case ID	**US6 Display requests**
Actors	User
Preconditions	User is logged in. User has book offers in their list.
Main flow of events	1. The use case starts when the user navigates to their book offers list. 2. The system displays the list of users who requested the book for every book.
Post conditions	The user can see the list of requests for their book offer.

Use Case ID	**US7 Accept request**
Actors	User
Preconditions	User is logged in. User has book offers in their list. User has received requests for a book offer.
Main flow of events	1. The use case starts when the user navigates to their book offers list and selects a book offer. 2. The user views the list of requests and selects a user to receive the book. 3. The system displays that the request has been accepted to both users
Post conditions	The selected user is notified to take the book

4 Design

4.1 Architecture



4.2 Design

Book

Responsibilities	Collaborations
- Holds information about a book.	- Collaborates with `BookRepository` for persistence.
- Fields include title, author, category, summary, etc.	- Collaborates with `BookKey` for composite keys.
- Provides methods to get and set book details.	

BookController

Responsibilities	Collaborations
- Handles HTTP requests related to books (e.g., viewing books, searching, adding offers).	- Collaborates with `BookService` for business logic.
- Processes user input and returns appropriate views.	- Collaborates with `Book` for displaying details.
- Validates user inputs and interacts with the service layer.	- Collaborates with `BookRepository` through the service layer.

BookKey

Responsibilities	Collaborations
- Represents the composite primary key for the `Book` entity.	- Used by `Book` for composite key representation.
- Fields include multiple keys (e.g., ISBN, edition).	- Collaborates with `BookRepository` for database operations.
- Implements `Serializable` for JPA requirements.	

BookRepository

Responsibilities	Collaborations
- Implements JPA repository interface for `Book` entity.	- Collaborates with `Book` for CRUD operations.
- Provides methods for querying the database (e.g., find by title, author, category).	- Collaborates with `BookKey` for composite key operations.
- Facilitates database access for the `Book` entity.	- Used by `BookService` for accessing and manipulating book data.

BookPage

Responsibilities	Collaborations
- Handles requests related to books (e.g., viewing book details, making requests).	- Collaborates with `RequestsRepo` for accessing request data.
- Processes user input related to book requests.	- Collaborates with `BookController` for book-related operations.
- Returns appropriate views or responses based on user actions.	- Collaborates with `Book` for displaying book details.

Requests

Responsibilities	Collaborations
- Holds information about a book request.	- Collaborates with `RequestsRepo` for persistence.
- Fields include details such as requester, requested book, date, etc.	- Collaborates with `RequestsId` for composite keys.
- Provides methods to get and set request details.	

RequestsId

Responsibilities	Collaborations
- Represents the composite primary key for the `Requests` entity.	- Used by `Requests` for composite key representation.
- Fields include multiple keys (e.g., requester ID, book ID).	- Collaborates with `RequestsRepo` for database operations.
- Implements `Serializable` for JPA requirements.	

RequestsRepo

Responsibilities	Collaborations
- Implements JPA repository interface for `Requests` entity.	- Collaborates with `Requests` for CRUD operations.
- Provides methods for querying the database (e.g., find by requester, book, date).	- Collaborates with `RequestsId` for composite key operations.
- Facilitates database access for the `Requests` entity.	- Used by `BookPage` for accessing and manipulating request data.

FrontPage

Responsibilities	Collaborations
- Handles requests related to the front page, profile page, update profile, registration, and login.	- Collaborates with `User`, `UserRepository`, and `BackendApplication` for user-related functionalities.
- Processes user input for registration, login, and profile updates.	- Collaborates with `UserRepository` for user data persistence.
- Returns appropriate views or responses based on user actions.	- Collaborates with `User` for displaying user profile details.

User

Responsibilities	Collaborations
- Holds information about a user.	- Collaborates with `UserRepository` for persistence.
- Fields include details such as username, email, password, profile information, etc.	- Collaborates with `FrontPage` for user registration and login.
- Provides methods to get and set user details.	- Collaborates with `UserRepository` for data retrieval and storage.

UserRepository

Responsibilities	Collaborations
- Implements JPA repository interface for `User` entity.	- Collaborates with `User` for CRUD operations related to users.
- Provides methods for querying the database (e.g., find by username, email).	- Collaborates with `FrontPage` for user registration and login.
- Facilitates database access for the `User` entity.	- Used by `FrontPage` for accessing and manipulating user data.

BackendApplication

Responsibilities	Collaborations
- Starts the Spring Boot application.	- Collaborates with `FrontPage` and other components of the application.
- Configures application settings, security, and endpoints.	- Collaborates with `UserRepository` for database access.
- Initializes and runs the application.	- Used by the system for application deployment and execution.