Online Social Bookstore Application

Sprint Report



TEAM

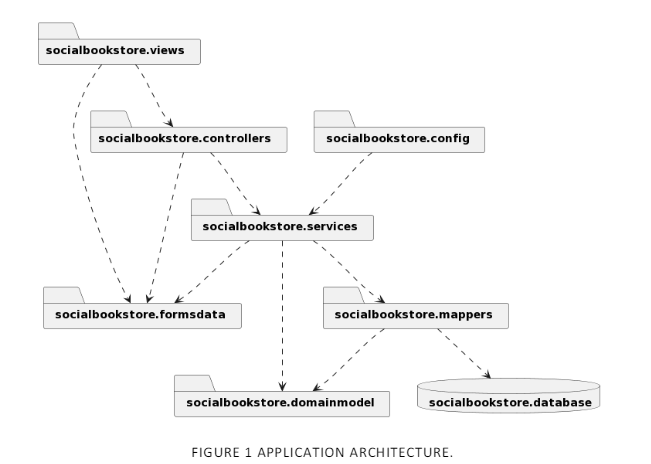
Ιωάννης Τσόχλας ΑΜ:4993 Κωσταντίνος Λαμανιάκου ΑΜ:5110 Σωτήριος Πασχάλης ΑΜ:0000

**Versions History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 18/5/2024 | 1.0 | Final stable social bookstore application | Ιωάννης.Τσόχλας Κωσταντίνος.Λαμανιάκου Σωτήριος Πασχάλης |

# Summary

***The objective of this project is to develop an online social bookstore application that allows individuals to exchange used books for free***. The objective was to be fulfilled using a Scrum approach to resolve organizational issues and technologies such as the Spring Boot Framework and MySQL to satisfy functional requirements. Non-functional requirements were specified in the form of design quality and best coding practices. Incomplete class diagrams describing the application architecture was the starting point. Testing was also among the requirements.



This document provides information concerning the 4 sprints of development for the Courses Web App. The development process has also been documented under the description of each commit on GitHub.

## 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.

# Scrum team and Sprint Backlog

## Scrum team

|  |  |
| --- | --- |
| **Product Owner** | Ioannis Tsochlas,Kostantinos Lamaniakou,Swtirios Pasxalis |
| **Scrum Master** | Ioannis Tsochlas |
| **Development Team** | Ioannis Tsochlas,Kostantinos Lamaniakou,Swtirios Pasxalis |

## 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** |
| **0** | **25 Feb** | **16 Mar** | **3** | **\*Preparation** |
| **1** | **17 Mar** | **10 Apr** | **3.5** | **User stories (US1 till US4)** |
| **2** | **25 Apr** | **6 May** | **2** | **User stories (US6 till US9)** |
| **3** | **7 May** | **21 May** | **2** | **User stories (US10, US11), \*Presentation** |

**\*Preparation** refers to tasks revolving around familiarization with new technologies such as Spring, Spring boot, MySQL Workbench, as well as building mock projects and a functional application prototype. The following preparation tasks were completed in the first sprint

* Setup MySQL
* Study Spring, Spring Boot, Maven Project Architecture
* Setup Application Architecture

**\*Presentation** refers to tasks revolving around the documentation and presentation of the application. The following presentation tasks were completed in the 4th sprint:

* Complete the report (includes: CRC cards, UML package and class diagrams, database schema)
* Write GitHub Readme
* Prepare Demo Video

Regarding the implementation of the user stories, each one was split into a checklist of 4 items:

* UI Design – HTML Styling
* Functionality Implementation
* Test (Validation)
* Use case

Once all 4 items of the checklist were completed, the user story was considered to be done.

# BACKLOG - USER STORIES

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **AS A**  **<User Type>** | **I WANT**  **<An Action>** | **SO THAT**  **<A Benefit/Value>** |
| US1 | user | I want to create a new account | I have access to the functionalities of the social bookstore application |
| US2 | user | I want login to my account | t I have access to the functionalities of the social bookstore application |
| US3 | user | I want to logout from my account | to terminate my interaction with the social bookstore application. |
| US4 | user | , I want to create a profile that includes my full name, address, age, phone number, the categories of books that I prefer (e.g., Art, Comic, Fantasy, Fiction, Biographies, History, Science, Literature, Adventure, Crime, Other) and my favorite authors | to facilitate the search/recommendation of books that may be interesting for me. |
| US5 | user | I want to be able to add a book offer in personal list of book offers with a description that includes the book title, the author(s), the category that the book belongs to and a summary of the book | to facilitate the search/recommendation of books by/to other users. |
| US6 | user | I want to browse a list of requests from other users who are interested in a book offer that I have made | so that I can decide to whom I shall give the book |
| US7 | user | I want to select a user who requested a book from my list and notify him that he can take the bookI also want to notify the rest of the users who requested the book that the book has been taken by another user. | To inform the users of the availability of books |
| US8 | user | I want to have access to the contact information of a selected user who requested a book that I offer | so that I can contact him to arrange the delivery of the book |
| US9 | user | I want to remove a book that is no longer available from my personal list of book offers and from the book request lists of other users who requested for this book | to enable the accurate book search/recommendation. |
| US10 | user | I want to be able to search for book offers with a certain title and authors to find interesting books. Besides specifying the search criteria, I want to choose between an exact or an approximate search strategy. Then, I want to be able to make a request for a book that is included in the search results to the user who offers the book | so that he becomes aware that I want to get the book. |
| US11 | user | I would like to browse a list of recommended book offers from the social bookstore application, to find interesting books. It would be nice to be able to choose among various recommendation strategies that consider information given in my profile |  |

# USE CASE

## User Registration

|  |  |
| --- | --- |
| **Use case ID** | US1 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host.  2. Database initialized and running on respective host.  3. Application successfully connected to database. |
| **Main flow of events** | 1. The use case starts when an unauthenticated user navigates to the registration page. 2. The user enters the required registration information and clicks the "Register" button. 3. The application validates the entered information for completeness and correctness. 4. The application attempts to create a new user account with the provided credentials in the database. 5. The application redirects to the home page. |
| **Alternative flow 1** | 1. The alternative flow begins after step 4. 2. The application determines that the username or email is already in use. 3. The application displays a "Username or Email is already in use" message. |
| **Post conditions** | The user is registered and can proceed to log in with the newly created credentials. |

## User Login

|  |  |
| --- | --- |
| **Use case ID** | US2 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host.  2. Database initialized and running on respective host.  3. Application successfully connected to database. |
| **Main flow of events** | 1. The use case starts when an unauthenticated user attempts to access any of the applications' pages that require authentication, including the login page itself.  1.1 If the user requests a page other than the login page.  1.2 The application redirects to the login page.  2. The user enters his/her username and password and clicks the "log in” button.  3. The application attempts to authenticate the users credentials (username and password with the respective credentials in the database.  4. The application redirects to the home page. |
| **Alternative flow 1** | 1. The alternative flow begins after step 3.  2. The application determines the given user credentiais as invalid.  3. The application displays a "Username or Password is invalid." message. |
| **Post conditions** | The user is authenticated and can access any of the applications' pages. |

## User Logout

|  |  |
| --- | --- |
| **Use case ID** | US3 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. |
| **Main flow of events** | 1. The use case starts when an authenticated user clicks the "logout" button or link in the application. 2. The application invalidates the user's session. 3. The application redirects the user to the login page. |
| **Post conditions** | The user's session is invalidated, and the user is logged out of the application.  The user is redirected to the login page and can no longer access authenticated pages without logging in again. |

## User Profile submission

|  |  |
| --- | --- |
| **Use case ID** | US4 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to the "Profile" page. 2. The user enters their full name, address, age, phone number, preferred book categories, and favorite authors. 3. The user clicks the "Submit" button. 4. The application validates the entered information. 5. The application saves the profile information in the database. 6. The application redirects the user to the profile view page. |
| **Post conditions** | If the profile creation is successful, the user’s profile information is stored in the database, and the user can view and edit their profile. |

## Add Book Offer

|  |  |
| --- | --- |
| **Use case ID** | US5 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to the "My Books/AddBook" page. 2. The user enters the book title, author(s), category. 3. The user clicks the "Submit" button. 4. The application saves the book offer information in the database. 5. The application redirects the user to their personal list of book offers. |
| **Post conditions** | If the book offer is added successfully, the book offer information is stored in the database, and the user can view it in their personal list of book offers. |

## Browse Requests for Book Offer

|  |  |
| --- | --- |
| **Use case ID** | US6 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. A User has at least one book offer listed. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to their list of book offers”My Books”. 2. The user select “Requests” a specific book offer from their list. 3. The application retrieves a list of requests from other users interested in the selected book offer from the database including actions such as take the book and Contact information . 4. The user reviews the list of requests and can make a decision on whom to give the book. |
| **Post conditions** | The user is able to see a list of requests from other users interested in their book offer and can make an informed decision on whom to give the book. |

## Notify Users about Book Offer

|  |  |
| --- | --- |
| **Use case ID** | US7 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. User has at least one book offer listed. 6. There are requests from other users for the book offer. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to their list of book offers. 2. The user selects a specific book offer from their list. 3. The application retrieves a list of requests from other users interested in the selected book offer from the database. 4. The user selects a specific request from the list. 5. The user clicks the "Take the book" button for a specific user. 6. The application sends a notification to the selected user informing them that they can take the book. 7. The application sends notifications to the other users who requested the book, informing them that the book has been taken by another user. |
| **Post conditions** | If the notifications are sent successfully, the selected user is informed that they can take the book, and the rest of the users are informed that the book has been taken by another user. |

## Access Contact Information of Requester

|  |  |
| --- | --- |
| **Use case ID** | US8 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. User has at least one book offer listed. 6. There are requests from other users for the book offer. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to their list of book offers. 2. The user selects a specific book requests action from their list. 3. The application retrieves a list of requests from other users interested in the selected book offer from the database. 4. The user selects a specific contact information from the list. 5. The application displays the contact information of the selected user (e.g., name, email, phone number). 6. The user uses the contact information to arrange the delivery of the book. |
| **Post conditions** | The user successfully accesses the contact information of the selected requester and can contact them to arrange the delivery of the book. |

## Remove Book Offer

|  |  |
| --- | --- |
| **Use case ID** | US9 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. User has at least one book offer listed. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to their list of book offers. 2. The user selects a specific book offer they want to remove. 3. The user clicks the "Delete" button. 4. The application deletes the book offer from the user's personal list of book offers in the database. 5. The application deletes the book offer from the request lists of other users who requested the book. |
| **Post conditions** | If the book offer is removed successfully, it is deleted from both the user's personal list of book offers and the request lists of other users.that is neceseary for accurate book search/recommendation |

## Browse Search and Request Book Offers

|  |  |
| --- | --- |
| **Use case ID** | US10 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. A User has at least one book offer listed. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to the "Search for Books" page. 2. The user enters search criteria, such as book title and author. 3. The user selects either an exact or an approximate search strategy. 4. The user clicks the "Search" button. 5. The application validates the search criteria. 6. The application performs the search based on the criteria and selected search strategy. 7. The application displays the search results, which include book offers that match the search criteria. 8. The user reviews the search results and selects a specific book offer. 9. The user clicks the "Request" button for the selected book offer. 10. The application displays a " Are you sure you want to request for this book?" message. 11. The application sends a request to the user who offers the book |
| **Post conditions** | If the search and request are successful, the user is able to find book offers that match the search criteria and send a request to the user offering the book.. |

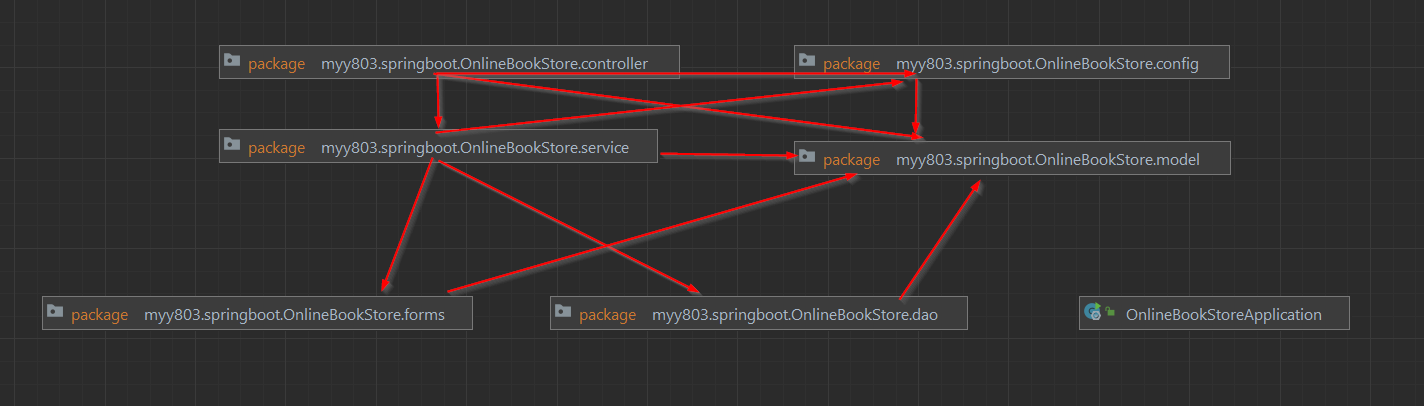
## Browse and Request Recommended Book Offers

|  |  |
| --- | --- |
| **Use case ID** | US11 |
| **Actors** | User |
| **Pre conditions** | 1. Application successfully initialized and running on respective host. 2. Database initialized and running on respective host. 3. Application successfully connected to the database. 4. User is authenticated and logged into the application. 5. User has a completed profile with favorite category. |
| **Main flow of events** | 1. The use case starts when an authenticated user navigates to the "Recommended Books Offers" page. 2. The application retrieves a list of recommended book offers based on the users favorite category. 3. The application displays the list of recommended book offers to the user. 4. The user browses the list and selects a specific recommended book offer. 5. The user clicks the "Request Book" button. 6. The application displays a " Are you sure you want to request for this book ?" message. 7. The application sends a request notification to the user who offers the book, informing them that the user is interested in getting the book. |
| **Post conditions** | If the request is sent successfully, the user offering the book is notified of the interest, and the interested user is informed that their request has been sent. |

# Design

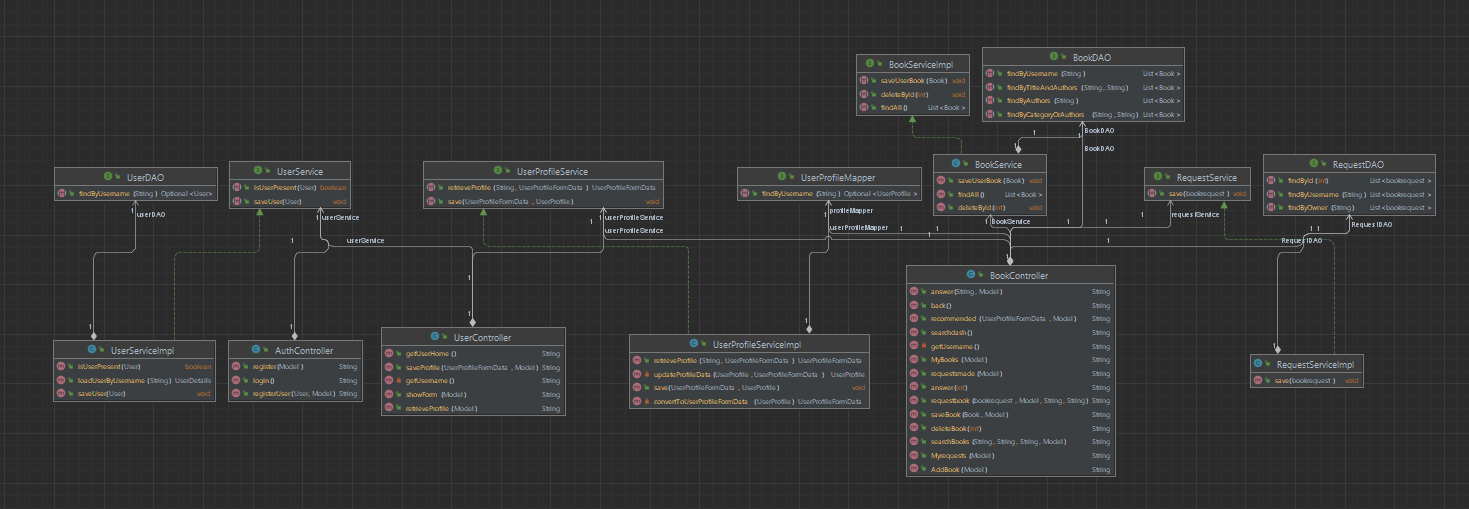
## Architecture

The overall application architecture is outlined in the following UML package diagram:



## Design

The detailed design is outlined by the following UML class diagrams.



The detailed design is further documented by the following Class-ResponsibilitiesCollaborations Cards for certain classes included in this release:

|  |  |
| --- | --- |
| **Class Name: CustomSecuritySuccessHandler** | |
| **Responsibilities:**   * Determines the url that is appropriate for the logged user based on his role | **Collaborations:** |

|  |  |
| --- | --- |
| **Class Name: WebMvcConfig** | |
| **Responsibilities:**   * ViewControllerRegistry allows to create  simple automated controllers pre-configured  with status code and/or a view | **Collaborations:** |

|  |  |
| --- | --- |
| **Class Name: WebMvcConfigWebSecurityConfig** | |
| **Responsibilities:**   * @Configuration Indicates that a class declares one or more   Spring container to generate bean definitions  and service requests for those beans at runtime.  The class may also have code that configures other  spring functionalities. | **Collaborations:** |

|  |  |
| --- | --- |
| **Class Name: AuthController** | |
| **Responsibilities:**   * Handle login requests: Respond to the "/login" endpoint and return the login page. * Handle registration requests: Respond to the "/register" endpoint, prepare the registration form, and return the registration page. * Save user registration: Respond to the "/save" endpoint, extract user registration data from the submitted form, save the user in the database, and return a success message. | **Collaborations:**   * Model * UserService |

|  |  |
| --- | --- |
| **Class Name: UserController** | |
| **Responsibilities:**   * Provide user dashboard: Respond to the "/user/dashboard" endpoint and return the user's dashboard page. * Retrieve user profile: Respond to the "/user/profile" endpoint, retrieve the user's profile data, and populate the profile form. * Retrieve logged-in username: Extract the logged-in username from the security context. * Save user profile: Respond to the "/saveProfile" endpoint, extract user profile data from the submitted form, save the user profile in the database, and redirect to the user's profile page. | **Collaborations:**   * Model: * UserProfileService: * UserService: |

|  |  |
| --- | --- |
| **Class Name: BookData** | |
| **Responsibilities:**   * Provide access to book title, authors, and category data. | **Collaborations:** |

|  |  |
| --- | --- |
| **Class Name: BookDAO** | |
| **Responsibilities:**   * Manage persistence operations for Book entities. * Provide methods for accessing and manipulating book data in the database. * Define query methods for searching books based on different criteria. | **Collaborations:**   * Book * JpaRepository |

|  |  |
| --- | --- |
| **Class Name: Book** | |
| **Responsibilities:**   * Represent a book entity with attributes such as title, authors, category, and username. * Provide getters and setters for accessing and modifying the attributes of a book. | **Collaborations:** |

|  |  |
| --- | --- |
| **Class Name: BookService** | |
| **Responsibilities:**   * Manage business logic related to book entities. * Interface with the BookDAO for database operations. * Provide methods for saving, retrieving, and deleting books. | **Collaborations:**   * BookDAO * Book |