Online Social Bookstore Application

Sprint Report

Μάρκος Ακρίβος 5057

Άγγελος Νέσσης 5029

VERSIONS HISTORY

Date	Version	Description	Author
25/3/2024	V.1	Done Use Cases for US1, US2 and US3.	Μάρκος, Άγγελος
8/4/2024	V.2	Done Use Cases for US5, US4 and US10.	Μάρκος, Άγγελος
21/4/2024	V.3	Done Use Cases for US11, US6 and US7.	Μάρκος, Άγγελος
23/4/2024	V.4	Done Use Cases for US8, and US9.	Μάρκος, Άγγελος
7/5/2024	V.5	Done small changes to the Use Cases.	Μάρκος, Άγγελος
11/5/2024	V.6	Done final versions of UML Diagrams.	Μάρκος, Άγγελος
12/5/2024	V.7	Done CRC Cards for service, service.search, service.recommendations, model and dao packages.	Μάρκος, Άγγελος
14/5/2024	V.8	Done CRC Cards for formsdata and controller packages and rewrite the “Purpose” part of the report.	Μάρκος, Άγγελος
15/5/2024	V.9	Final Version of the Report	Μάρκος, Άγγελος

Introduction

This document provides information concerning all sprints of the project.

1.1 Purpose

The purpose of our project (called BookstoreApplication) is to create an online application where people can offer books to anyone or request any book that they may be interested in, while making that procedure simple and free of charge. The application provides its users with the ability to request books either by searching for books that fulfill their criteria, or by looking at the recommendations field. Additionally, the users can check their book offers and requests from their profile page, in which they can also enter their personal information and customize it to receive books based on their favorite book authors and categories.

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

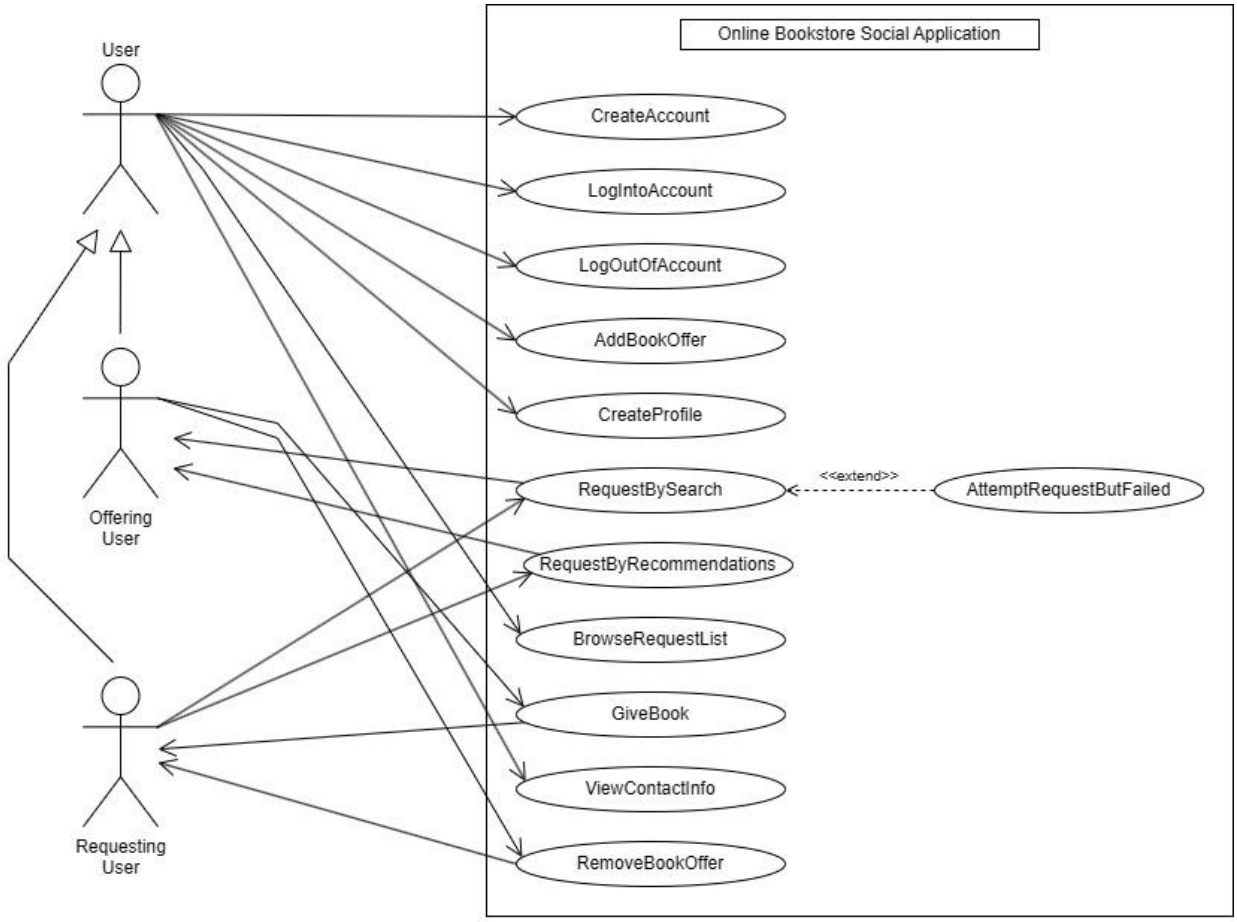
Product Owner	Άγγελος Νέσσης
Scrum Master	Μάρκος Ακρίβος
Development Team	Άγγελος Νέσσης, Μάρκος Ακρίβος

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**25/3**	**7/4**	**2**	**1,2,3**
2	**8/4**	**22/4**	**2**	**4,5,10**
3	**23/4**	**6/5**	**2**	**6,7,8,9,11**

3 Use Cases

The **UML Use Case Diagram** is given in the following picture and the **Detailed Use Case Descriptions** are below it.



Picture 1: UML Use Case Diagram

3.1 CreateAccount

Use case ID	UC1 (US1)
Actors	The User
Main flow of events	1. The use case starts when the user selects “Create Account” after opening the application. 2. The user may enter his first and last name, his age and a username and password. 3. The user presses the “Confirm” button. 4. While any of the fields are left blank or an already used username is given 4.1. The system informs the user accordingly. 4.2. The user corrects the necessary fields appropriately. 4.3. The user presses the “Confirm” button. 5. The system creates the account. 6. The use case terminates.
Post conditions (MF)	A new account has been added to the system.
Alternative flow 1	The user may exit from the account creating process at any time.
Post conditions (AF1)	There is no change in the system.

3.2 LogIntoAccount

Use case ID	UC2 (US2)
Actors	The User
Pre conditions	The user should already have an account.
Main flow of events	1. The use case starts when the user opens the application. 2. The user enters his username and password. 3. The user presses the “Confirm” button. 4. While the credentials are invalid 4.1. The system informs the user accordingly. 4.2. The user may enter the correct username and password. 4.3. The user presses the “Confirm” button. 5. The system logs the user into his account. 6. The use case terminates.
Post conditions (MF)	The user has logged into his account.
Alternative flow 1	The user may exit from the login process at any time.
Post conditions (AF1)	There is no change in the system.

3.3 LogOutOfAccount

Use case ID	UC3 (US3)
Actors	The User
Pre conditions	The user should be logged into his account.
Main flow of events	1. The use case starts when the user selects “Logout” from the toolbar. 2. The system opens a confirmation window. 3. If the user selects “No” 3.1. The confirmation window closes. 4. Else 4.1. The confirmation window closes. 4.2. The system logs the user out of the account and returns to the login screen. 5. The use case terminates.
Post conditions (MF)	The user is logged out of the system.

3.4 AddBookOffer

Use case ID	UC4 (US5)
Actors	The User
Pre conditions	The user should be logged into his account.
Main flow of events	1. The use case starts when the user selects “Add Book Offer” from the toolbar. 2. The system redirects the user to another page. 3. The user fills in the book’s title, the book’s authors, the book’s category and the book’s summary. 4. The user presses the “Confirm” button. 5. While any field is blank 5.1. The user fills in the necessary fields. 5.2. The user presses the “Confirm” button. 6. The system creates a new book offer. 7. The system adds the book offer to a personal offer list of the user. 8. The system returns a message to the user that “The book has been added in your Offer List”. 9. The use case terminates.
Post conditions (MF)	A new book offer is available for users to request.
Alternative flow 1	At any point, the user may move to a different page.
Post conditions (AF1)	There is no change in the system.

3.5 CreateProfile

Use case ID	UC5 (US4)
Actors	The User
Pre conditions	The user should be logged into his account
Main flow of events	1. The use case starts when the user selects “Profile” from the toolbar. 2. The system redirects the user to his profile page. 3. The user can update or fill his first and last name, age, phone number, address as well as his favorite authors and book genres. 4. The user presses the “Save” button. 5. The system updates the user’s profile. 6. The system returns a message to the user that “Your Profile has been updated!”. 7. The use case terminates.
Post conditions (MF)	The user’s profile has been updated.
Alternative flow 1	At any point, the user may move to a different page.
Post conditions (AF1)	There is no change in the system.

3.6 RequestBySearch

Use case ID	UC6 (US10)
Actors	The Requesting User, The Offering User
Related Use Cases	Extended by: AttemptRequestButFailed
Pre conditions	The Requesting User should be logged into his account
Main flow of events	1. The use case starts when the requesting user clicks on the search bar. 2. The requesting user fills in the title and authors of the book(s) they want to search. 3. The requesting user can choose if they want an exact or approximate search. 4. The requesting user presses the “Search” button. 5. The system returns to the requesting user all the books that fulfill their criteria. 6. The requesting user chooses a book from the books in the search result. 7. The requesting user presses the “Request Book” button. 8. The system sends the Request to the offering user. 9. The system returns a message to the requesting user that “The book has successfully been requested!”. 10. The use case terminates here.
Post conditions (MF)	The requesting user has successfully requested a book.
Alternative flow 1	AttemptRequestButFailed
Post conditions (AF1)	There is no change in the system.
Alternative flow 2	1. The Alternative Flow starts at step 5 of the Main Flow. 2. The system cannot find any books to show in the results. 3. The system returns a message to the requesting user that “No books were found with this book title and authors”.
Post conditions (AF2)	There is no change in the system.
Alternative flow 3	At any point, the requesting user may move to a different page.
Post conditions (AF3)	There is no change in the system.

3.7 AttemptRequestButFailed

Use case ID	UC7
Actors	The User
Related Use Cases	Extension of: RequestBySearch
Pre conditions	The user executes UC6
Main flow of events	1. The use case starts at step 6 of UC6. 2. The user chooses a book that is not available or is offered by them or has already been requested by them. 3. The system does not display the “Request Book” button.
Post conditions (MF)	The main flow of UC6 continues at step 10.

3.8 RequestByRecommendations

Use case ID	UC8 (US11)
Actors	The Requesting User, The Offering User
Pre conditions	The Requesting User should be logged into his account.
Main flow of events	1. The use case starts when the requesting user is on the homepage. 2. The requesting user selects the criteria which the recommendations will be based on. 3. The system recommends books based on the selected criteria. 4. The requesting user chooses a book from the recommended books. 5. The requesting user presses the “Request Book” button. 6. The system sends the Request to the offering user. 7. The system returns a message to the requesting user that “The book has successfully been requested!”. 8. The use case terminates here.
Post conditions (MF)	The requesting user has successfully requested a book.
Alternative flow 1	1. The Alternative Flow starts at step 3 of the Main Flow. 2. The system cannot find any books to recommend. 3. The system returns a message to the requesting user that “There are not any books to recommend for you!”.
Post conditions (AF1)	There is no change in the system.
Alternative flow 2	At any point, the requesting user may move to a different page.
Post conditions (AF2)	There is no change in the system.

3.9 BrowseRequestList

Use case ID	UC9 (US6)
Actors	The User
Pre conditions	The user should be logged into his account and offer at least one book.
Main flow of events	1. The use case starts when the user selects “View Offers” from the toolbar. 2. The system returns a list of books that the user offers. 3. The user chooses one of the books. 4. The system returns information about that book. 5. The user presses on “Show Requesting Users” button. 6. The system returns a list with the users who requested the book. 7. The use case terminates.
Post conditions (MF)	There is no change in the system.
Alternative flow 1	At any point, the user may move to a different page.
Post Conditions (AF1)	There is no change in the system.

3.10 GiveBook

Use case ID	UC10 (US7)
Actors	The Offering User, The Requesting Users
Pre conditions	The Offering User should be logged in and offer at least one book. The Requesting Users should have requested at least one of the Offering User’s books.
Main flow of events	1. The use case starts when the offering user browses the request list of one of his book offers. 2. The offering user selects one of the requesting users. 3. The offering user presses the “Give” button. 4. The system notifies the requesting user, selected by the offering user, that his request was accepted. 5. The system notifies the rest of the requesting users that their request was declined. 6. The system notifies the offering user that the book has been given. 7. The use case terminates.
Post conditions (MF)	The status of the requests for the book offer is updated.
Alternative flow 1	At any point, the offering user may move to a different page.
Post conditions (AF1)	There is no change in the system.

3.11 ViewContactInfo

Use case ID	UC11 (US8)
Actors	The User
Pre conditions	The user should be logged in, offer at least one book and have at least one request for an offered book.
Main flow of events	1. The use case starts when the user browses the request list of one of his book offers. 2. The user selects one of the requesting users. 3. The user presses the “Show Contact Info” button. 4. The system returns the selected user’s contact info. 5. The use case terminates.
Post conditions (MF)	There is no change in the system.
Alternative flow 1	At any point, the user may move to a different page.
Post conditions (AF1)	There is no change in the system.

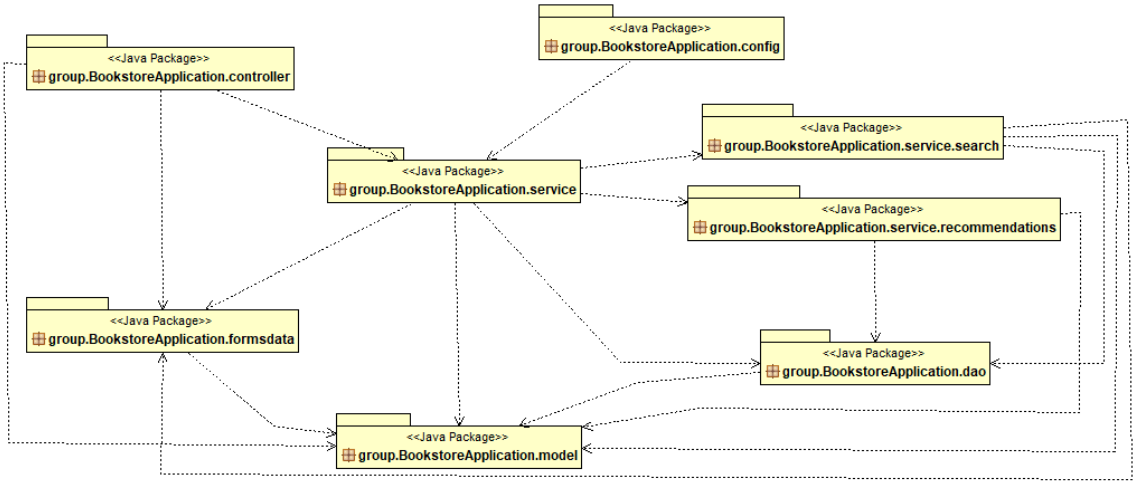
3.12 RemoveBookOffer

Use case ID	UC12 (US9)
Actors	The Offering User, The Requesting Users
Pre conditions	The Offering User should be logged in and have at least one book offered.
Main flow of events	1. The use case starts when the offering user browses his personal book offer list. 2. The offering user selects the book they want to delete. 3. The offering user presses the “Delete” button. 4. The system removes the selected book from the list and from the request list of the requesting users. 5. The use case terminates.
Post conditions (MF)	The selected book has been removed entirely from the database.
Alternative flow 1	At any point, the offering user may move to a different page.
Post conditions (AF1)	There is no change in the system.

4 Design

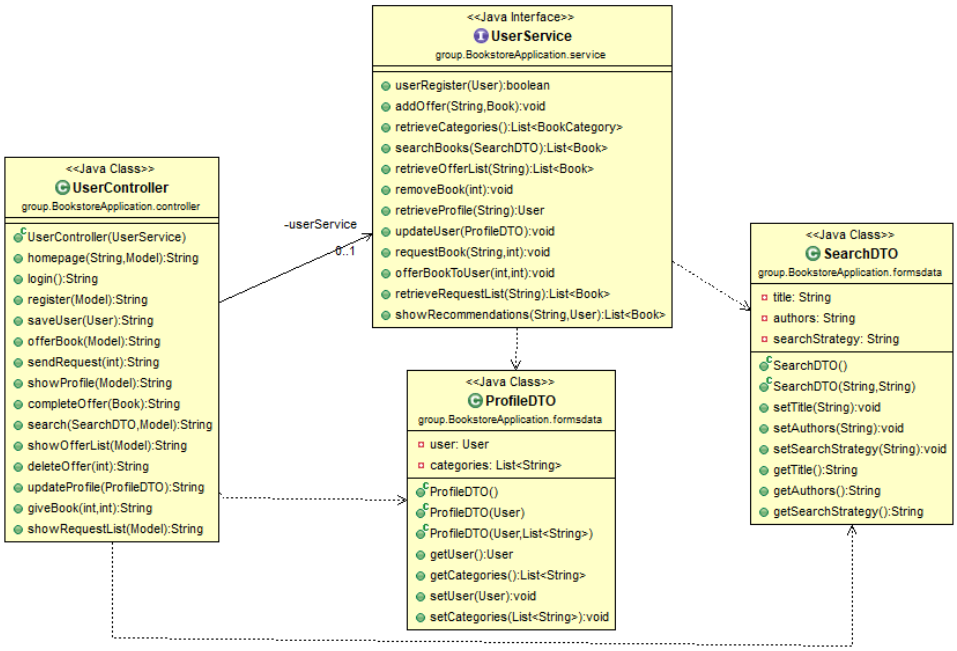
4.1 Architecture

The **UML Package Diagram** is given in the following picture and the **UML Class Diagrams** for each package are below it.

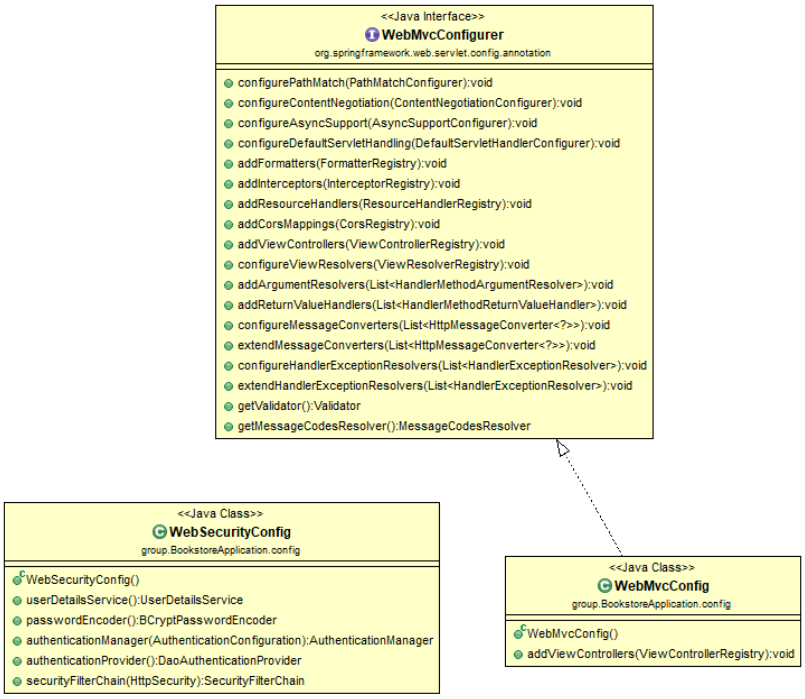


Picture 2: UML Package Diagram

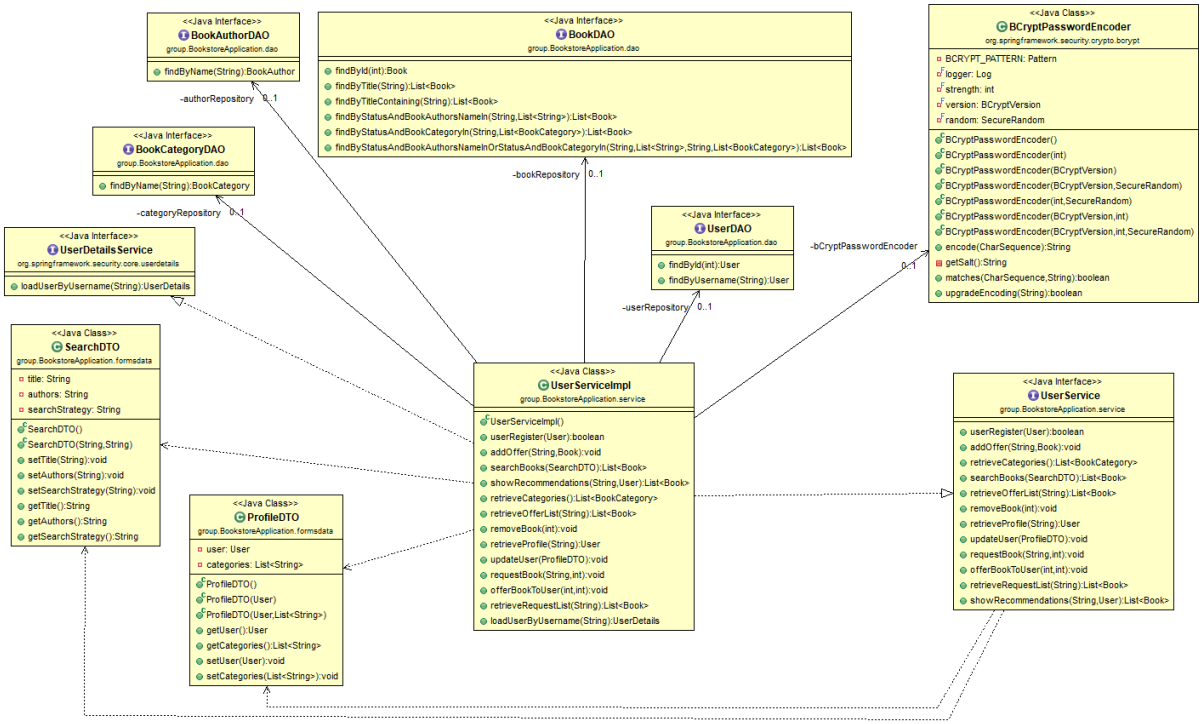
package group.BookstoreApplication.controller



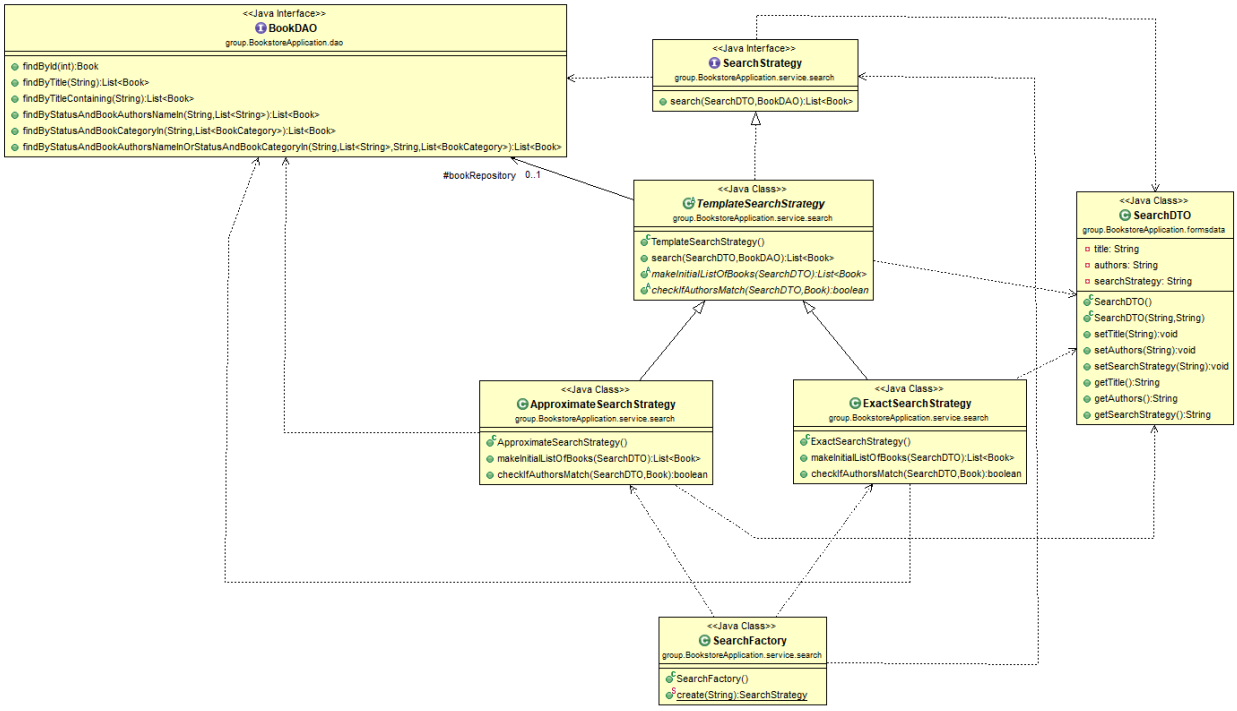
package group.BookstoreApplication.config



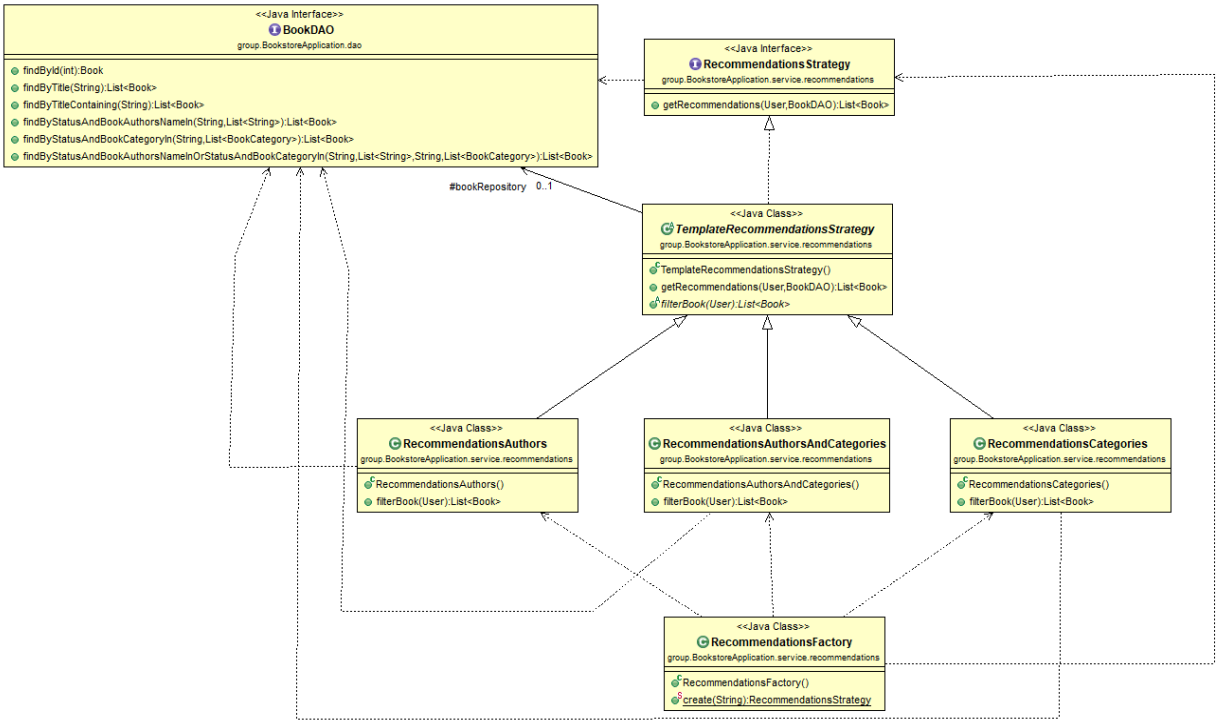
package group.BookstoreApplication.service



package group.BookstoreApplication.service.search

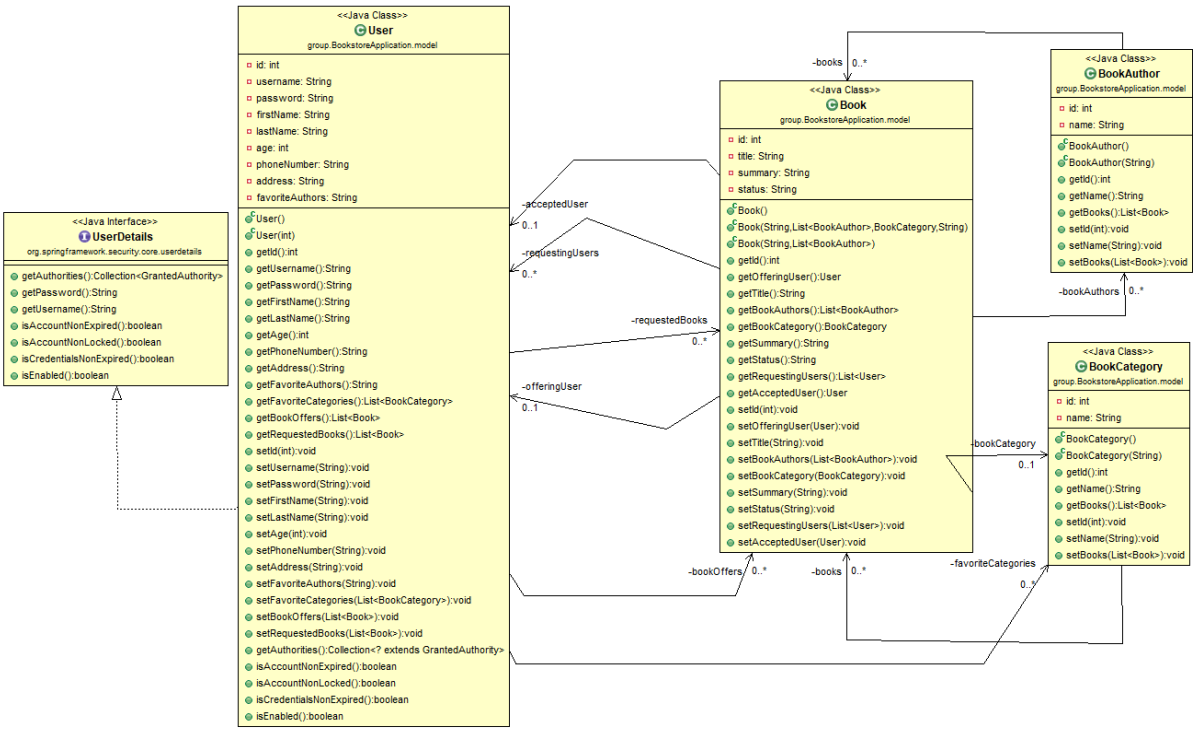


package group.BookstoreApplication.service.recommendations

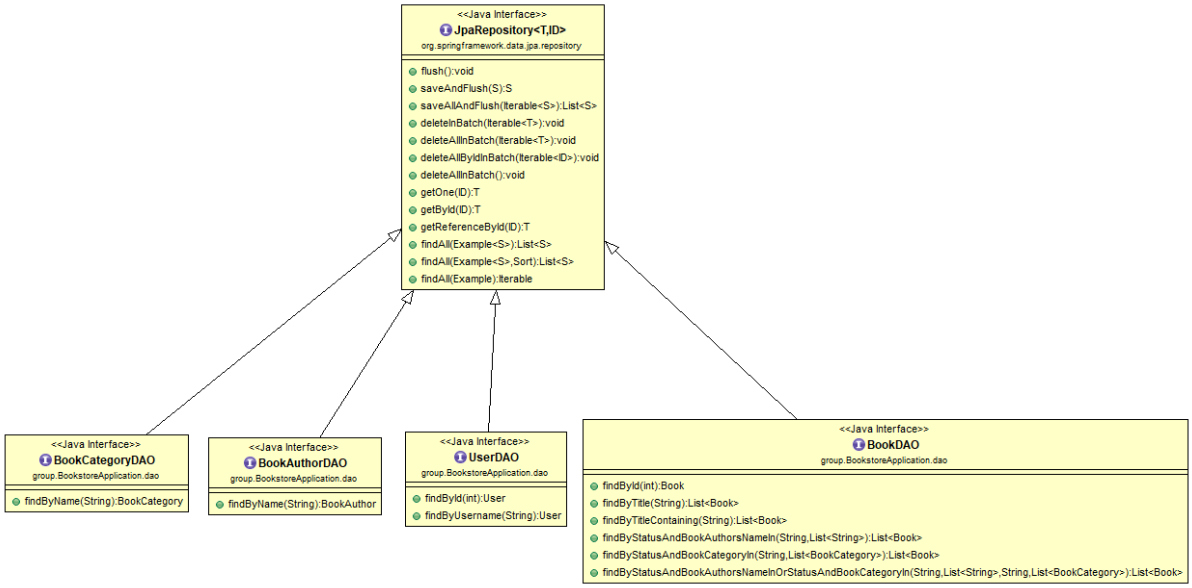


package group.BookstoreApplication.formsdata

package group.BookstoreApplication.model



package group.BookstoreApplication.dao



4.2 Design

The following CRC Cards provide a detailed description of each class included in the packages depicted in the UML Class Diagrams.

4.2.1 PACKAGE: GROUP.BOOKSTOREAPPLICATION.CONTROLLER

Class Name: UserController	
Responsibilities: ▪ Handles requests that relate to the user's interaction with the application. ▪ Redirects the user to the appropriate page, considering the user's given input. ▪ Receives data from the user and transfers it to UserService. ▪ Receives data from UserService and manipulates it. ▪ Updates the view to display the manipulated data properly.	**Collaborations:** ▪ UserService ▪ SearchDTO ▪ SecurityContextHolder ▪ Principal ▪ User ▪ Book ▪ BookAuthor ▪ BookCategory ▪ ProfileDTO ▪ Model

4.2.2 PACKAGE: GROUP.BOOKSTOREAPPLICATION.DAO

Interface Name: BookCategoryDAO	
Responsibilities: ▪ Extends JPARepository interface. ▪ Defines a query method that retrieves book categories based on the book category's name.	**Collaborations:** ▪ JpaRepository ▪ BookCategory

Interface Name: BookAuthorDAO	
Responsibilities: ▪ Extends JPARepository interface. ▪ Defines a query method that retrieves book authors based on the book author's name.	**Collaborations:** ▪ JpaRepository ▪ BookAuthor

Interface Name: UserDAO	
Responsibilities: ▪ Extends JPARepository interface. ▪ Defines query methods that retrieve users based on their ID or their username.	**Collaborations:** ▪ JpaRepository ▪ User

Interface Name: BookDAO	
Responsibilities: ▪ Extends JPARepository interface. ▪ Defines query methods that retrieve books based on criteria such as book title, authors, category, and status.	**Collaborations:** ▪ JpaRepository ▪ Book ▪ BookCategory

4.2.3 PACKAGE: GROUP.BOOKSTOREAPPLICATION.FORMSDATA

Class Name: ProfileDTO	
Responsibilities ▪ Stores the user's personal information. ▪ Holds the names of the user's favorite categories inside a list. ▪ Is used to transfer data from view layer to service layer. ▪ Provides methods that allow to set and get the user's personal information. ▪ Provides methods that allow to set and get the contents of the list which contains the names of the user's favorite categories.	**Collaborations:** ▪ User ▪ BookCategory

Class Name: SearchDTO	
Responsibilities: ▪ Holds the search criteria (book title, authors, and selected search strategy) that the user entered in the search form. ▪ Is used to transfer data from view layer to service layer. ▪ Provides methods that allow to set and get the user's entered book title, authors, and selected search strategy.	**Collaborations:** ▪ None

4.2.4 PACKAGE: GROUP.BOOKSTOREAPPLICATION.MODEL

Interface Name: UserDetails	
Responsibilities: ▪ Defines methods that manage core user information.	**Collaborations:** ▪ User

Class Name: User	
Responsibilities: ▪ Defines the User as an Object in the program. ▪ Stores the user's ID, username, and password. ▪ Stores the user's first and last name, age, phone number and address. ▪ Stores the user's favorite book authors and categories. ▪ Stores a list with the user's book offers and a list with the user's requested books. ▪ Provides methods that allow to set and get the user's ID, username, and password. ▪ Provides methods that allow to set and get the user's first and last name, age, phone number and address. ▪ Provides methods that allow to set and get the user's favorite book authors and categories. ▪ Provides methods that allow to set and get the user's book offer list and the user's requested book list. ▪ Implements methods that manage core user information.	**Collaborations:** ▪ UserDetails ▪ Book

Class Name: Book	
Responsibilities: ▪ Defines the Book as an Object in the program. ▪ Stores the ID, the title, list of authors, category, summary and offering status. ▪ Stores a list of users who request the book. ▪ Stores the book's offering user, and the user whose book request got accepted. ▪ Provides methods that allow to set and get the book's ID, title, summary, category, offering status, and author list. ▪ Provides methods that allow to set and get the list of users who request a book. ▪ Provides methods that allow to set and get the book's offering user, and the user whose book request got accepted.	**Collaborations:** ▪ User ▪ BookAuthor ▪ BookCategory

Class Name: BookAuthor	
Responsibilities: ▪ Defines the Book Author as an Object in the program. ▪ Stores the ID and the name of a book author. ▪ Stores a list of books that are written by an author. ▪ Provides methods that allow to set and get the ID and name of an author. ▪ Provides methods that allow to set and get the list of books written by an author.	**Collaborations:** ▪ Book

Class Name: BookCategory	
Responsibilities: ▪ Defines the Book Category as an Object in the program. ▪ Stores the ID and name of a book category. ▪ Stores a list of the books that belong to a category. ▪ Provides methods that allow to set and get the ID and name of a category. ▪ Provides methods that allow to set and get the list of books that belong to the same category.	**Collaborations:** ▪ Book

4.2.5 PACKAGE: GROUP.BOOKSTOREAPPLICATION.SERVICE

Interface Name: UserDetailsService	
Responsibilities: ▪ Defines method that a user can be found by his username.	**Collaborations:** ▪ UserServiceImpl ▪ UserDetails

Interface Name: UserService	
Responsibilities: ▪ Defines the set of actions that a user can do. These are: ○ Register Account ○ Create Book Offer ○ Retrieve Book Categories ○ Search Books ○ Retrieve Personal Book Offer List ○ Retrieve Personal Book Request List ○ Remove Book Offer ○ Retrieve Profile Information ○ Update Profile Information ○ Request Book Offer ○ Give Book to Another User ○ Show Recommended Books	**Collaborations:** ▪ UserServiceImpl ▪ SearchDTO ▪ User ▪ Book ▪ BookCategory ▪ ProfileDTO

Class Name: UserServiceImpl	
Responsibilities: ▪ Implements the set of user's actions that are mentioned in UserService interface. These are: ○ Register Account ○ Create Book Offer ○ Retrieve Book Categories ○ Search Books ○ Retrieve Personal Book Offer List ○ Retrieve Personal Book Request List ○ Remove Book Offer ○ Retrieve Profile Information ○ Update Profile Information ○ Request Book Offer ○ Give Book to Another User ○ Show Recommended Books ▪ Implements method that a user can be found by his username.	**Collaborations:** ▪ UserService ▪ UserDetailsService ▪ BCryptPasswordEncoder ▪ UserDAO ▪ BookDAO ▪ BookCategoryDAO ▪ BookAuthorDAO ▪ SearchDTO ▪ ProfileDTO ▪ Book ▪ User

4.2.6 PACKAGE: GROUP.BOOKSTOREAPPLICATION.SERVICE.RECOMMENDATIONS

Interface Name: RecommendationsStrategy	
Responsibilities: ▪ Defines a method that returns a list of books based on the specified recommendations strategy.	**Collaborations:** ▪ User ▪ BookDAO ▪ Book ▪ TemplateRecommendationsStrategy

Class Name: TemplateRecommendationsStrategy	
Responsibilities: ▪ Implements a template method for the book recommendation algorithm. ▪ Defines one abstract method, which customizes the result of book recommendations.	**Collaborations:** ▪ RecommendationsStrategy ▪ User ▪ BookDAO ▪ Book ▪ RecommendationsAuthors ▪ RecommendationsCategories ▪ RecommendationsAuthorsAndCategories

Class Name: RecommendationsAuthors	
Responsibilities: ▪ Extends the TemplateRecommendationsStrategy class. ▪ Implements the one abstract method in such way that the recommendation algorithm returns a list of books that their authors are included in the user's favorite authors.	**Collaborations:** ▪ TemplateRecommendationsStrategy ▪ User ▪ Book ▪ BookDAO

Class Name: RecommendationsCategories	
Responsibilities: ▪ Extends the TemplateRecommendationsStrategy class. ▪ Implements the one abstract method in such way that the recommendation algorithm returns a list of books that their category is included in the user's favorite categories.	**Collaborations:** ▪ TemplateRecommendationsStrategy ▪ User ▪ Book ▪ BookDAO

Class Name: RecommendationsAuthorsAndCategories	
Responsibilities: ▪ Extends the TemplateRecommendationsStrategy class. ▪ Implements the one abstract method in such way that the recommendation algorithm returns a list of books that their category or authors are included in the user's favorite categories and authors.	**Collaborations:** ▪ TemplateRecommendationsStrategy ▪ User ▪ Book ▪ BookDAO

Class Name: RecommendationsFactory	
Responsibilities: ▪ Creates appropriate instances of RecommendationsStrategy, depending on the string we pass as argument.	**Collaborations:** ▪ RecommendationsStrategy ▪ RecommendationsAuthors ▪ RecommendationsCategories ▪ RecommendationsAuthorsAndCategories

4.2.7 PACKAGE: GROUP.BOOKSTOREAPPLICATION.SERVICE.SEARCH

Interface Name: SearchStrategy	
Responsibilities: ▪ Defines a search method that returns a list of books based on the search criteria.	**Collaborations:** ▪ SearchDTO ▪ BookDAO ▪ Book ▪ TemplateSearchStrategy

Class Name: TemplateSearchStrategy	
Responsibilities: ▪ Implements a template method for the search algorithm. ▪ Defines two abstract methods, which customize the search behavior.	**Collaborations:** ▪ SearchStrategy ▪ SearchDTO ▪ BookDAO ▪ Book ▪ ApproximateSearchStrategy ▪ ExactSearchStrategy

Class Name: ApproximateSearchStrategy	
Responsibilities: ▪ Extends the TemplateSearchStrategy class. ▪ Implements the two abstract methods in such way that the search algorithm returns a list of books that contain the given title and at least one of the authors.	**Collaborations:** ▪ TemplateSearchStrategy ▪ SearchDTO ▪ BookDAO ▪ Book ▪ BookAuthor

Class Name: ExactSearchStrategy	
Responsibilities: ▪ Extends the TemplateSearchStrategy class. ▪ Implements the two abstract methods in such way that the search algorithm returns a list of books that match exactly the given title and authors.	**Collaborations:** ▪ TemplateSearchStrategy ▪ SearchDTO ▪ BookDAO ▪ Book ▪ BookAuthor

Class Name: SearchFactory	
Responsibilities: ▪ Creates appropriate instances of SearchStrategy, depending on the string we pass as argument.	**Collaborations:** ▪ SearchStrategy ▪ ApproximateSearchStrategy ▪ ExactSearchStrategy