# SDA Lab Project

**Submitted By:** Ubaidullah yousaf (sp22-Bse-104)

**Submitted To:** Sir Mukhtiar Zamin

**Date:** 11th November 2024

## Book Wala

## Report:

**Overview:**
We are developing an online platform that offers licensed books for users to explore. This web application will allow readers to enjoy books, bookmark their progress, and return seamlessly to where they left off. Additionally, users can share their thoughts through reviews, ratings, and comments on the titles they've read.

**Advantages:**

- This web application will significantly enhance users' reading habits.
- It will aid in improving users' verbal communication skills.
- Users will have the opportunity to engage with one another through reviews and discussions.

**Key Features:**

- A wide variety of books are available for selection.
- Tools for highlighting key passages.
- Bookmarking functionality for easy return to previous reading points.
- Tailored book recommendations based on individual preferences and reading history.

**Target Users:**

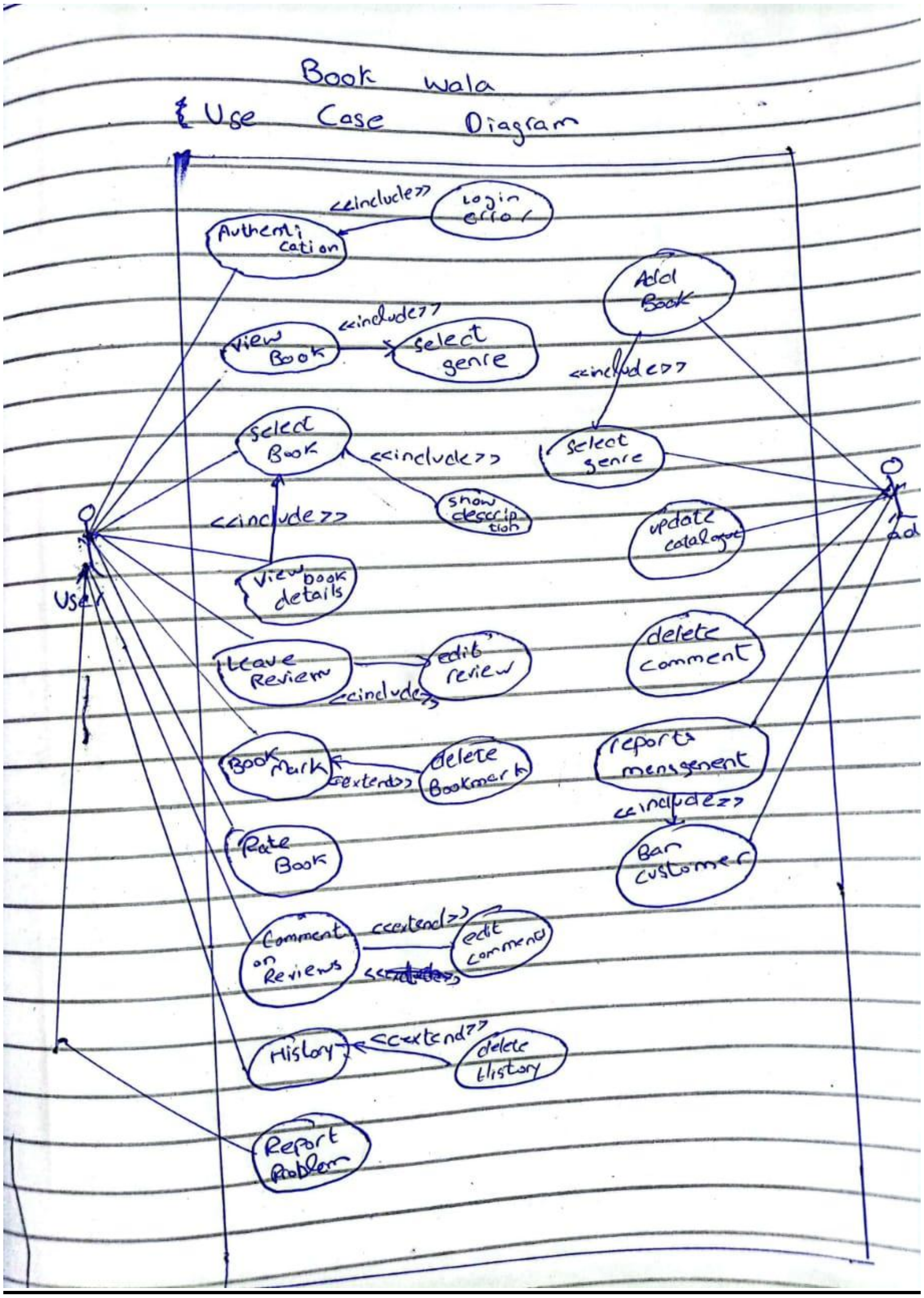
- Students aiming to enhance their reading abilities.
- Avid readers interested in book discussions.
- Individuals learning a new language.

**Possible Challenges:**

- Creating a welcoming and enjoyable environment for all users.
- Keeping users motivated to read consistently.
- Regular upkeep of the web application to ensure it operates smoothly.

**Conclusion:**
In summary, our web application aims to make a meaningful difference in users' reading and speaking skills, promoting literacy and effective communication in the process.

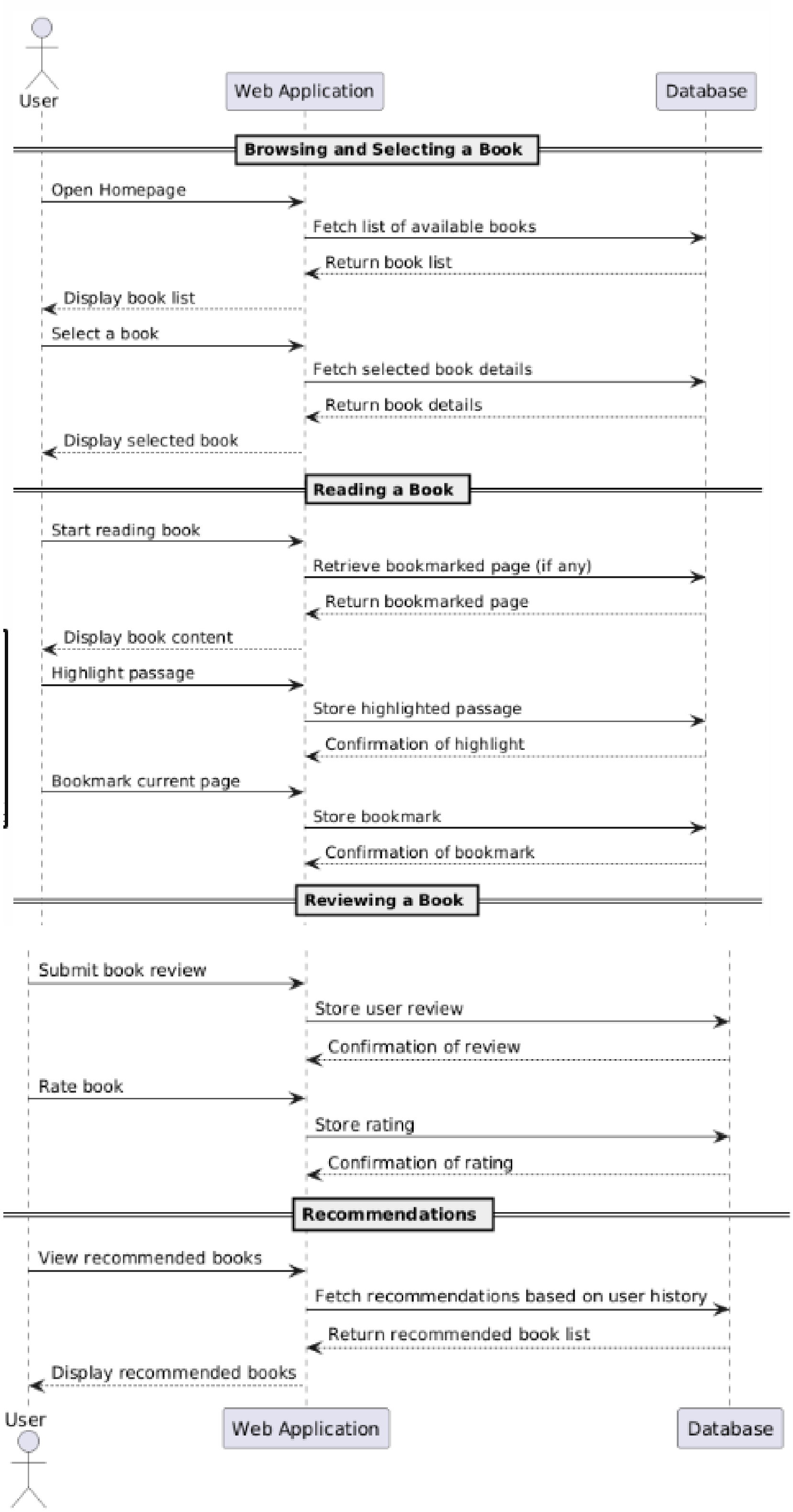
## Use Case Diagram:



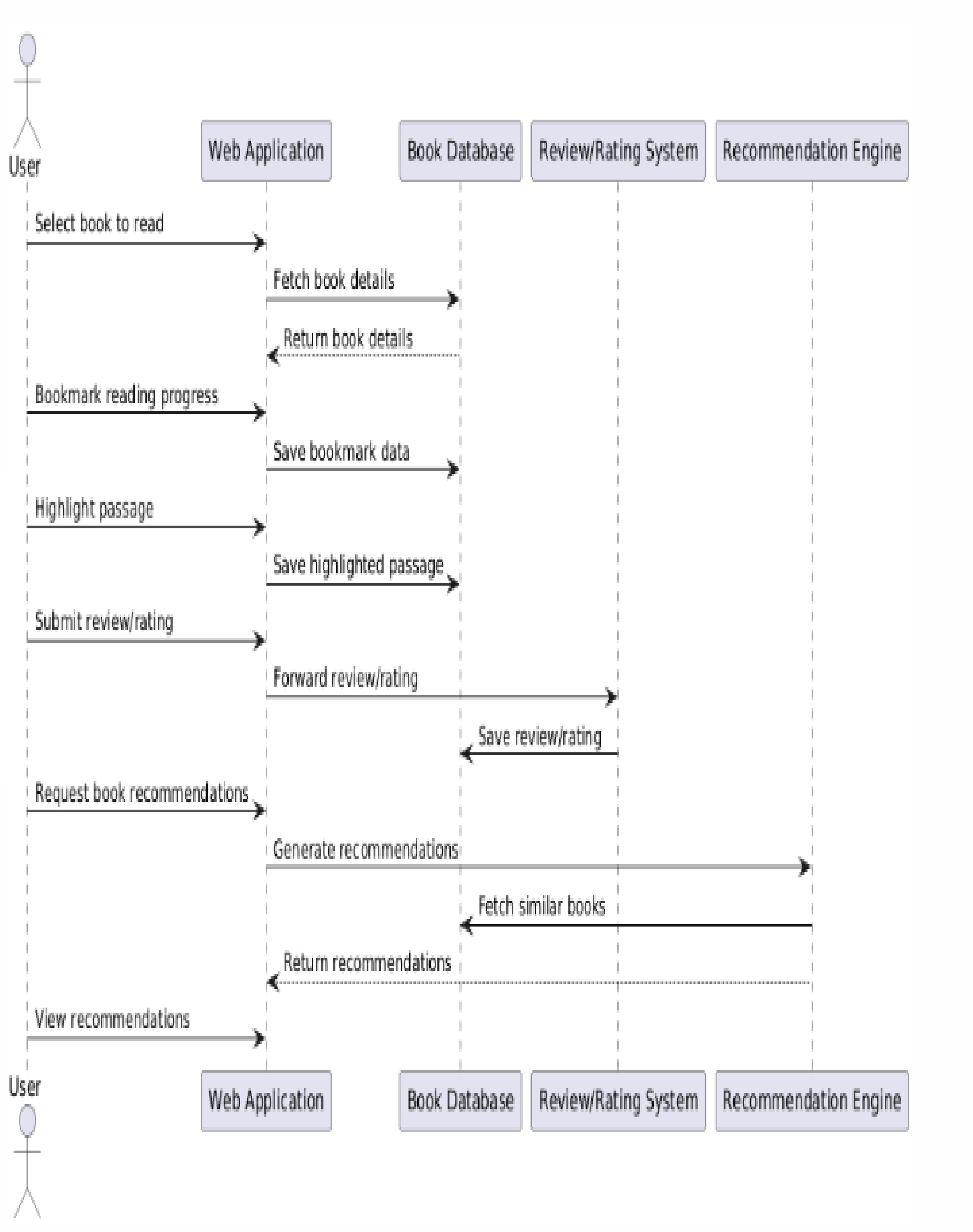
## Fully Dressed Use Case:

| Use Case | Submit and Review Feedback |
|---|---|
| **Actors** | Customer, Admin |
| **Precondition** | Customer is logged in and has read or interacted with a book. The Admin is logged in. |
| **Trigger** | Customer finishes reading a book and wants to leave feedback. |
| **Basic Flow** | 1. Customer navigates to the book's page.<br>2. Customer clicks on the "Leave a Review" button.<br>3. Customer writes a review, rates the book, and submits it.<br>4. The system stores the review and links it to the book and customer profile.<br>5. Admin logs into the system and navigates to the "Reviews" section.<br>6. Admin views the submitted review. |
| **Post Condition** | The review is available on the book's page for other users to read. The admin has viewed the review. |
| **Alternative Flow** | 3a. Customer tries to leave a review without being logged in.<br>- System prompts the customer to log in or create an account.<br>3b. Customer submits an incomplete review.<br>- System displays an error, asking for all required fields to be filled (e.g., rating, review text). |
| **Exception Flow** | 5a. Admin encounters an error while trying to view reviews (e.g., server issue).<br>- System shows an error message and logs the issue. |
| **Assumptions** | Customer is a valid registered user. The admin has the necessary permissions to read reviews. |
| **Priority** | Medium |
| **Frequency of Use** | Common (Multiple reviews daily, depending on traffic) |
| **Stakeholders** | Customers, Admin |
| **Success Guarantee** | The customer's review is successfully posted, and the admin has access to read it without errors. |

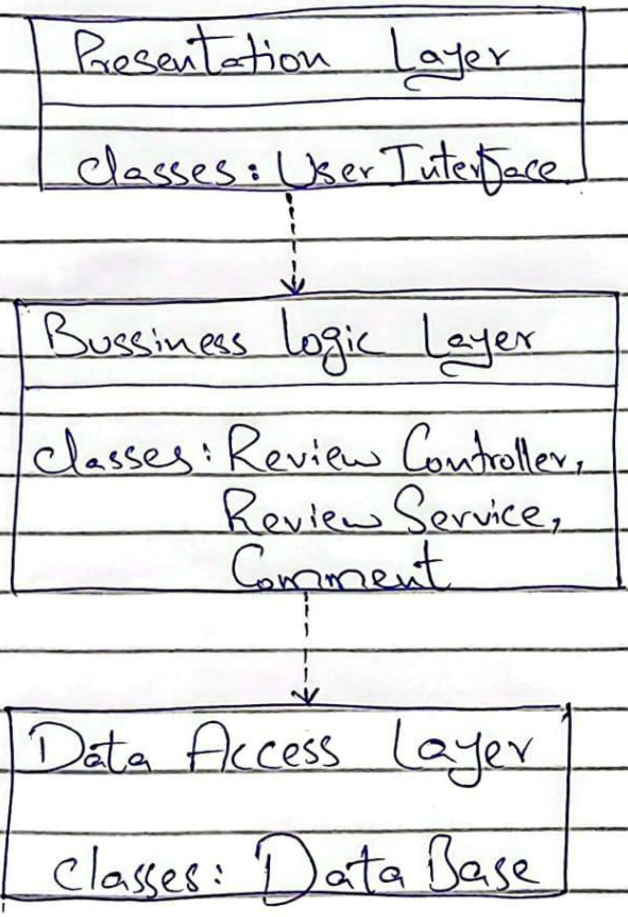
## **SSD:**


User
Web Application
Database
Browsing and Selecting a Book
Open Homepage
Fetch list of available books
Return book list
Display book list
Select a book
Fetch selected book details
Return book details
Display selected book
Reading a Book
Start reading book
Retrieve bookmarked page (if any)
Return bookmarked page
Display book content
Highlight passage
Store highlighted passage
Confirmation of highlight
Bookmark current page
Store bookmark
Confirmation of bookmark
Reviewing a Book
Submit book review
Store user review
Confirmation of review
Rate book
Store rating
Confirmation of rating
Recommendations
View recommended books
Fetch recommendations based on user history
Return recommended book list
Display recommended books
User
Web Application
Database

**Communication Diagram:**

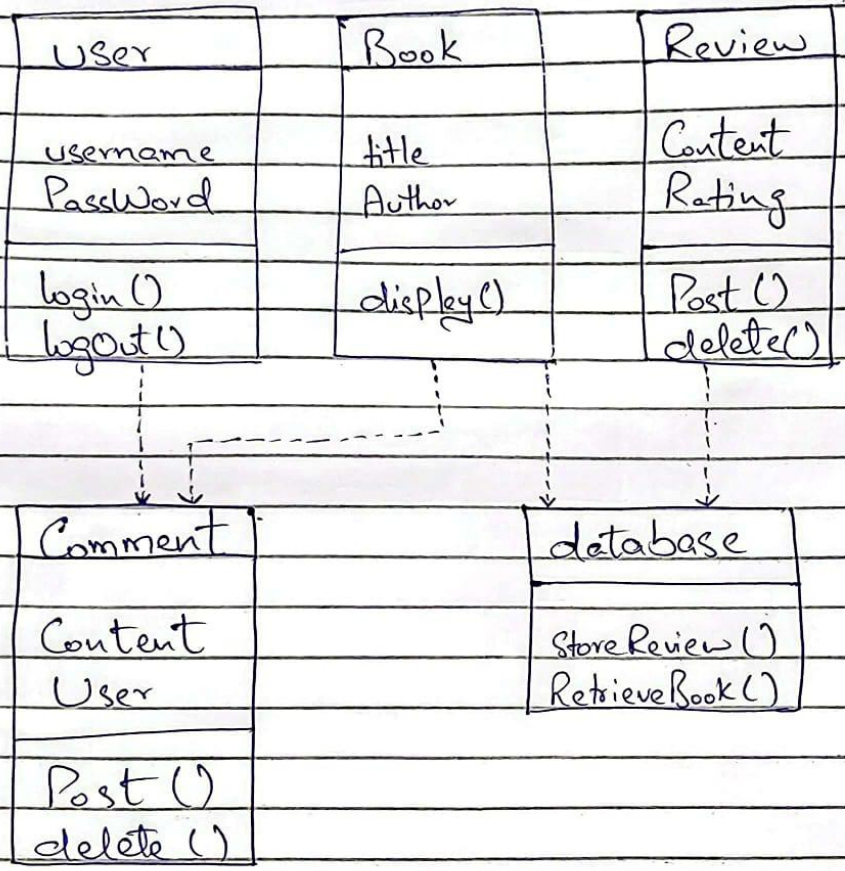


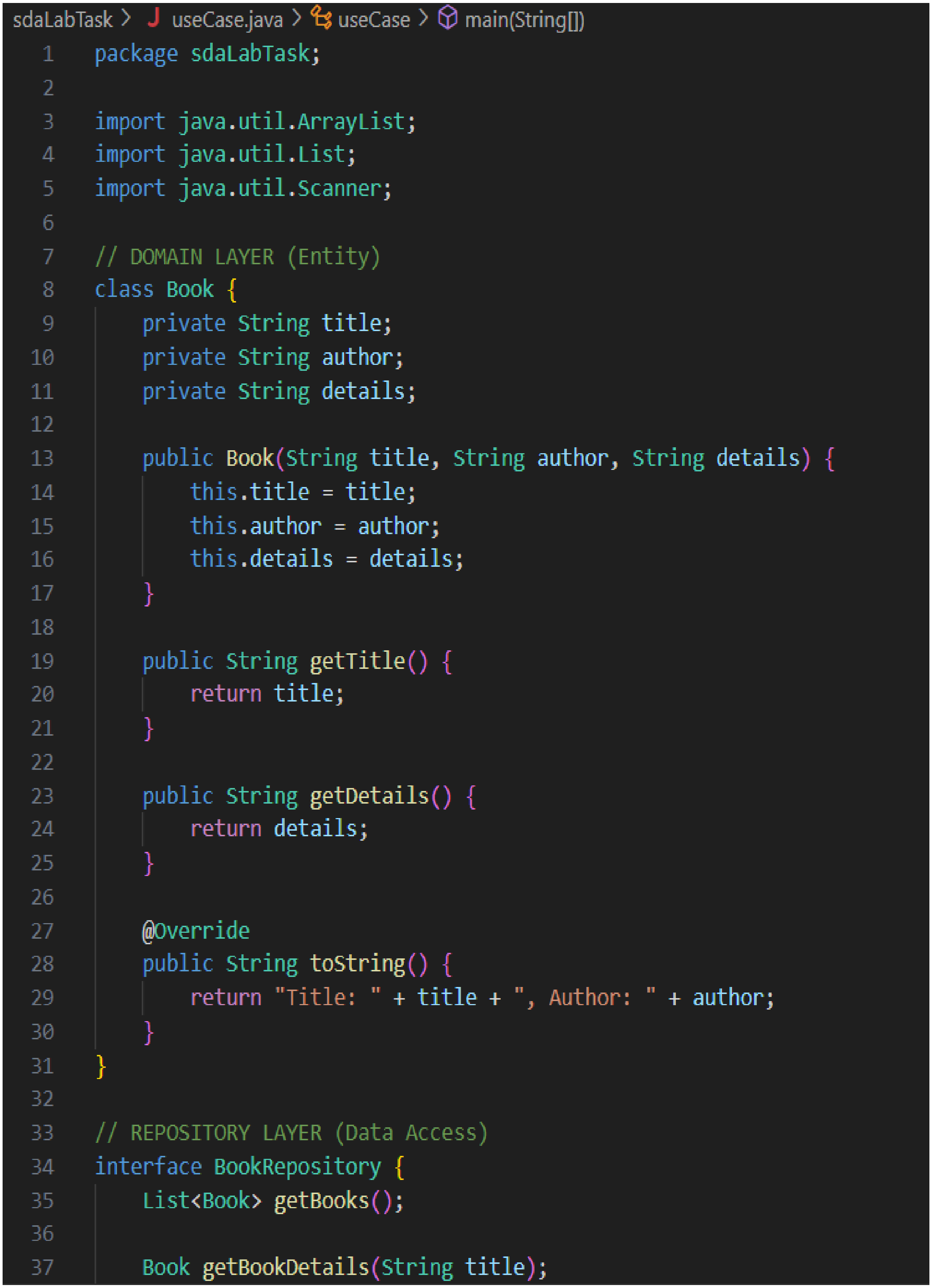
# Package Diagram



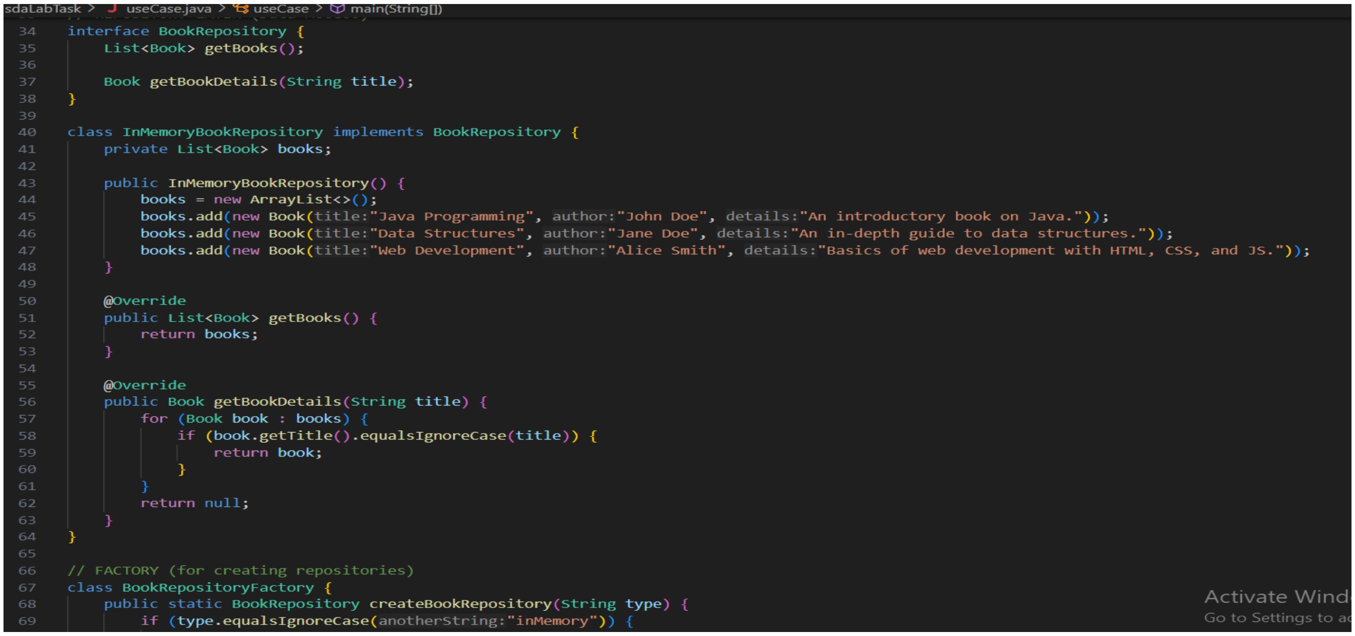
Huzaifa Ahmed Khan /FA22-BSE-079

## class Diagram



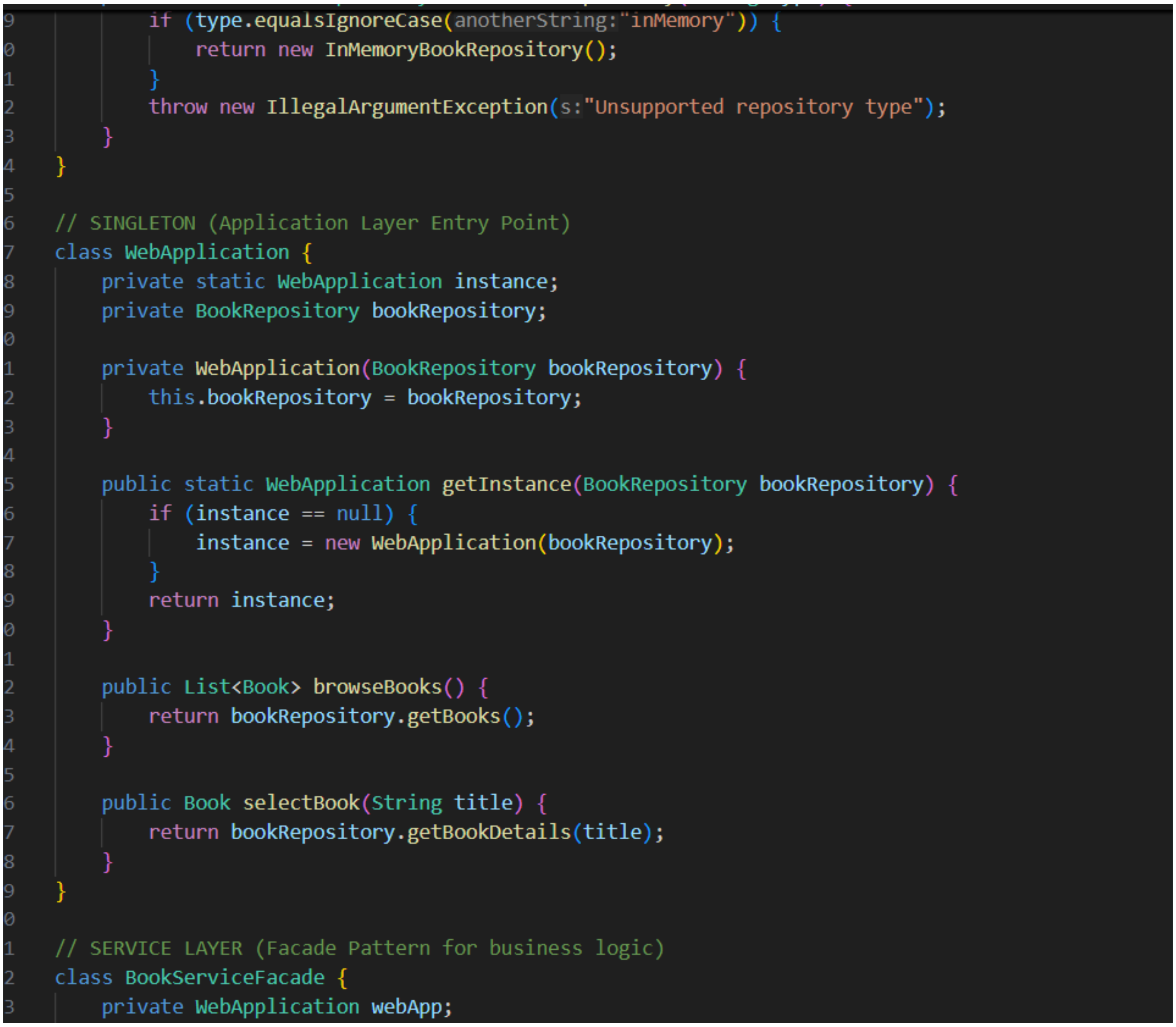


```java
package sdaLabTask;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// DOMAIN LAYER (Entity)
class Book {
    private String title;
    private String author;
    private String details;

    public Book(String title, String author, String details) {
        this.title = title;
        this.author = author;
        this.details = details;
    }

    public String getTitle() {
        return title;
    }

    public String getDetails() {
        return details;
    }

    @Override
    public String toString() {
        return "Title: " + title + ", Author: " + author;
    }
}

// REPOSITORY LAYER (Data Access)
interface BookRepository {
    List<Book> getBooks();

    Book getBookDetails(String title);
```

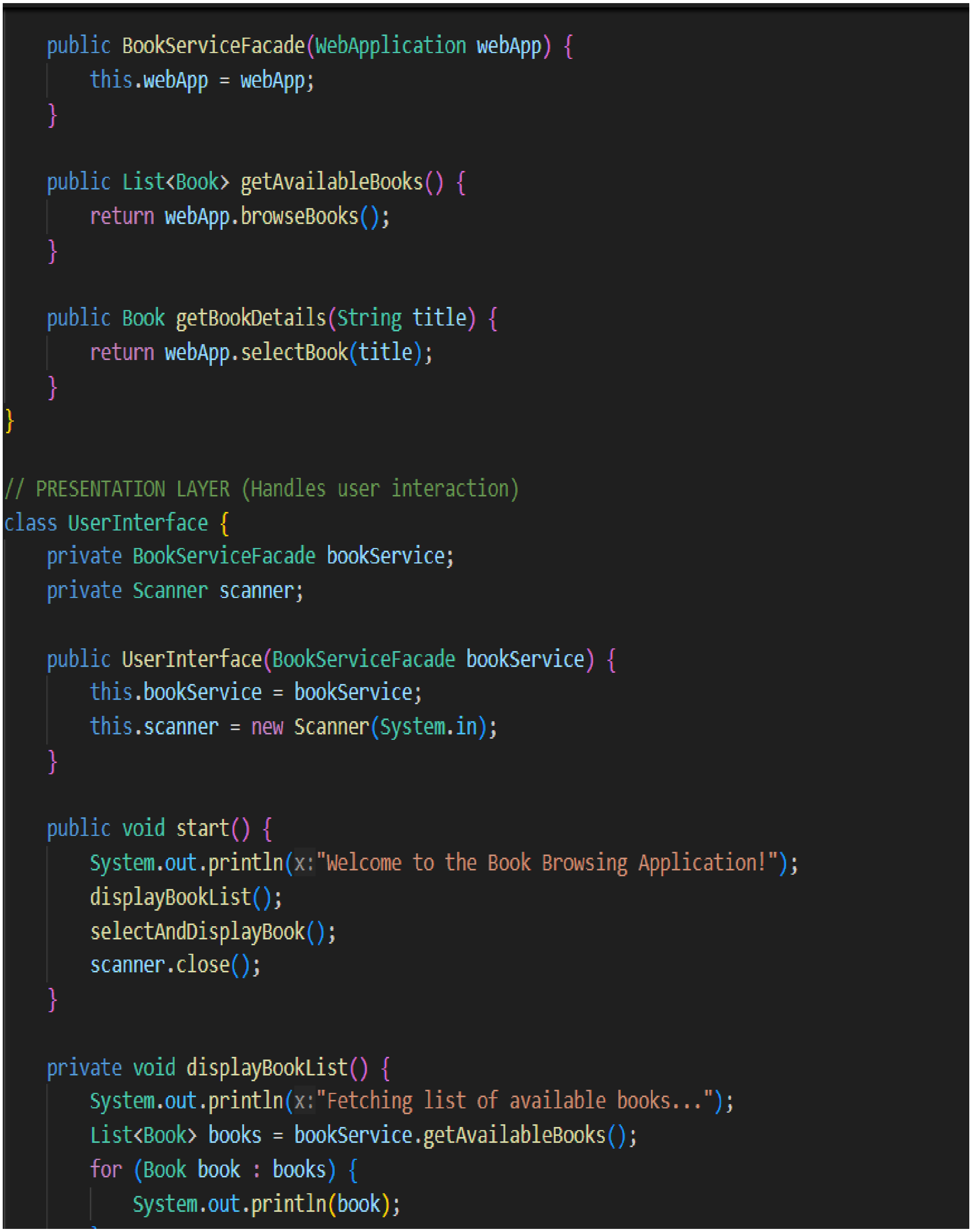


sdaLabTask > useCase.java > useCase > main(String[])

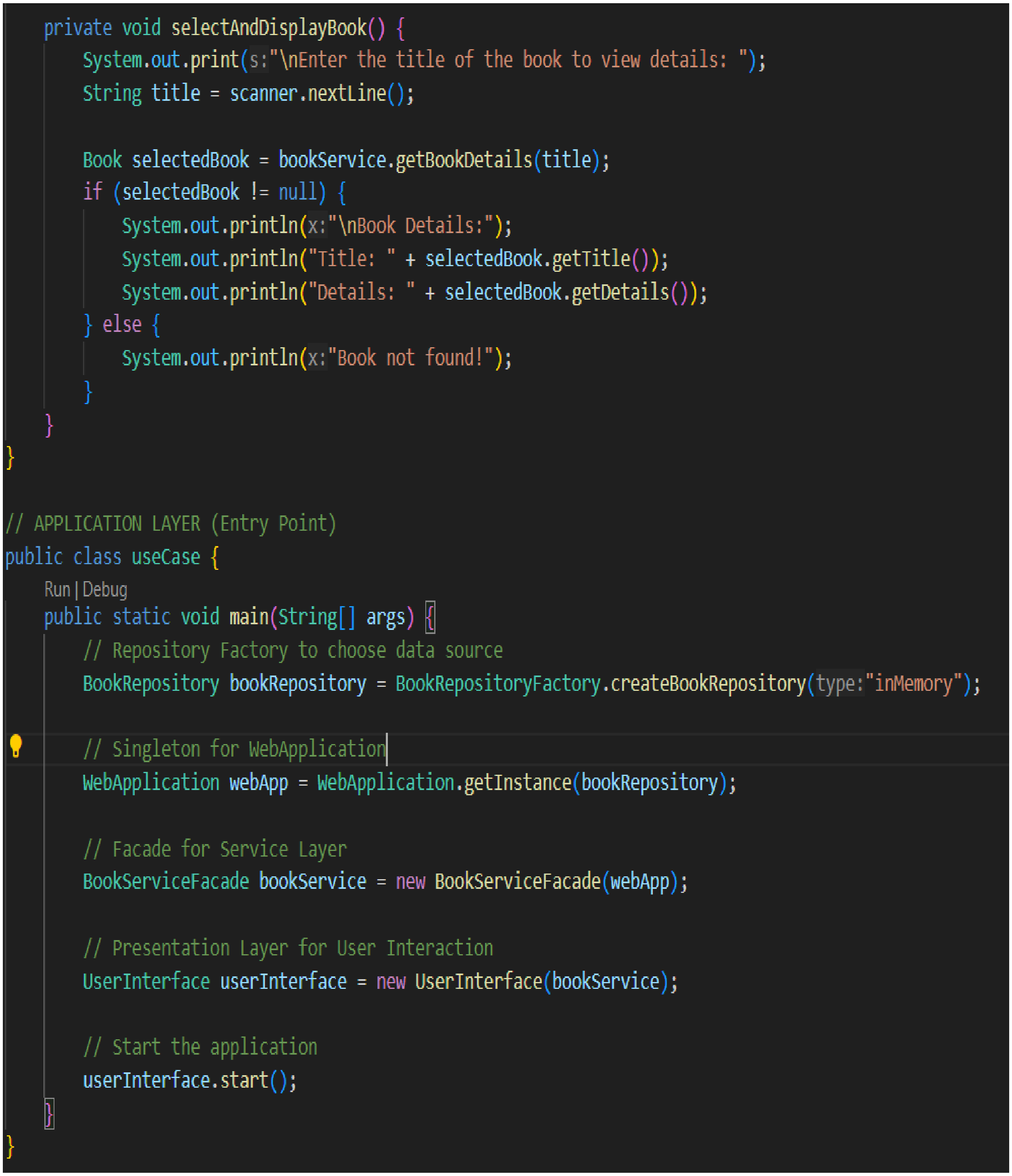
```java
interface BookRepository {
    List<Book> getBooks();

    Book getBookDetails(String title);
}

class InMemoryBookRepository implements BookRepository {
    private List<Book> books;

    public InMemoryBookRepository() {
        books = new ArrayList<>();
        books.add(new Book(title:"Java Programming", author:"John Doe", details:"An introductory book on Java."));
        books.add(new Book(title:"Data Structures", author:"Jane Doe", details:"An in-depth guide to data structures."));
        books.add(new Book(title:"Web Development", author:"Alice Smith", details:"Basics of web development with HTML, CSS, and JS."));
    }

    @Override
    public List<Book> getBooks() {
        return books;
    }

    @Override
    public Book getBookDetails(String title) {
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
                return book;
            }
        }
        return null;
    }
}

// FACTORY (for creating repositories)
class BookRepositoryFactory {
    public static BookRepository createBookRepository(String type) {
        if (type.equalsIgnoreCase(anotherString:"inMemory")) {
```



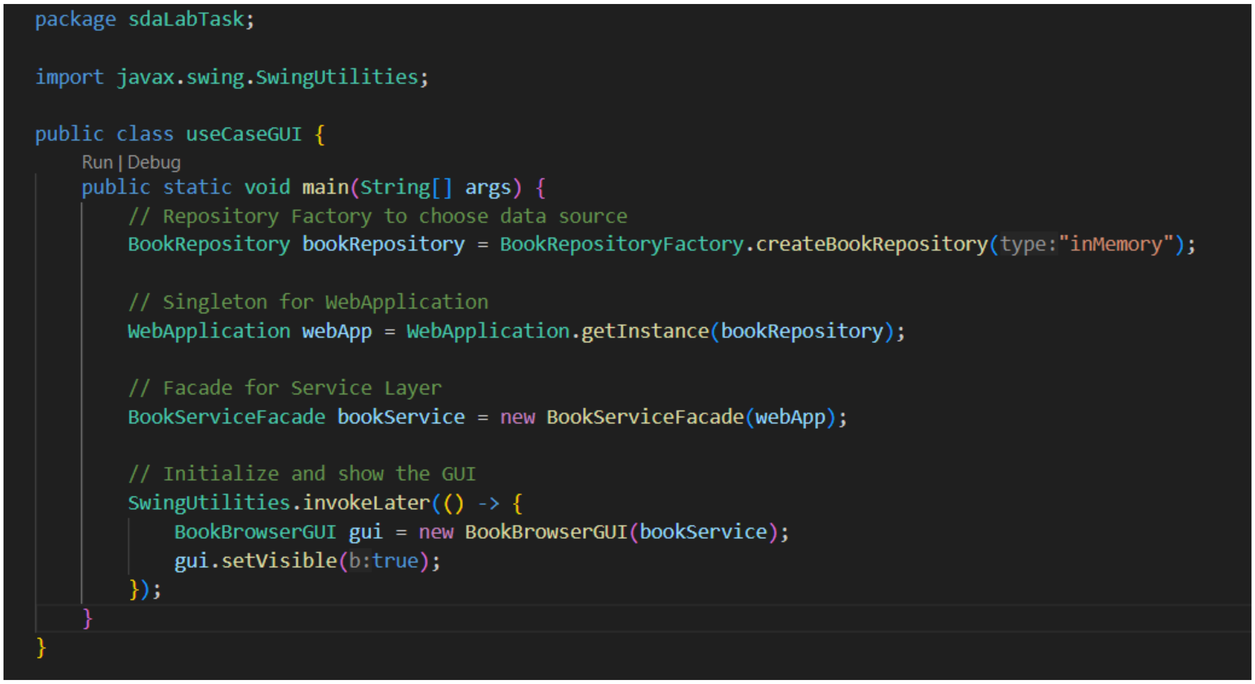
```java
        if (type.equalsIgnoreCase(anotherString:"inMemory")) {
            return new InMemoryBookRepository();
        }
        throw new IllegalArgumentException(s:"Unsupported repository type");
    }
}

// SINGLETON (Application Layer Entry Point)
class WebApplication {
    private static WebApplication instance;
    private BookRepository bookRepository;

    private WebApplication(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public static WebApplication getInstance(BookRepository bookRepository) {
        if (instance == null) {
            instance = new WebApplication(bookRepository);
        }
        return instance;
    }

    public List<Book> browseBooks() {
        return bookRepository.getBooks();
    }

    public Book selectBook(String title) {
        return bookRepository.getBookDetails(title);
    }
}

// SERVICE LAYER (Facade Pattern for business logic)
class BookServiceFacade {
    private WebApplication webApp;
```



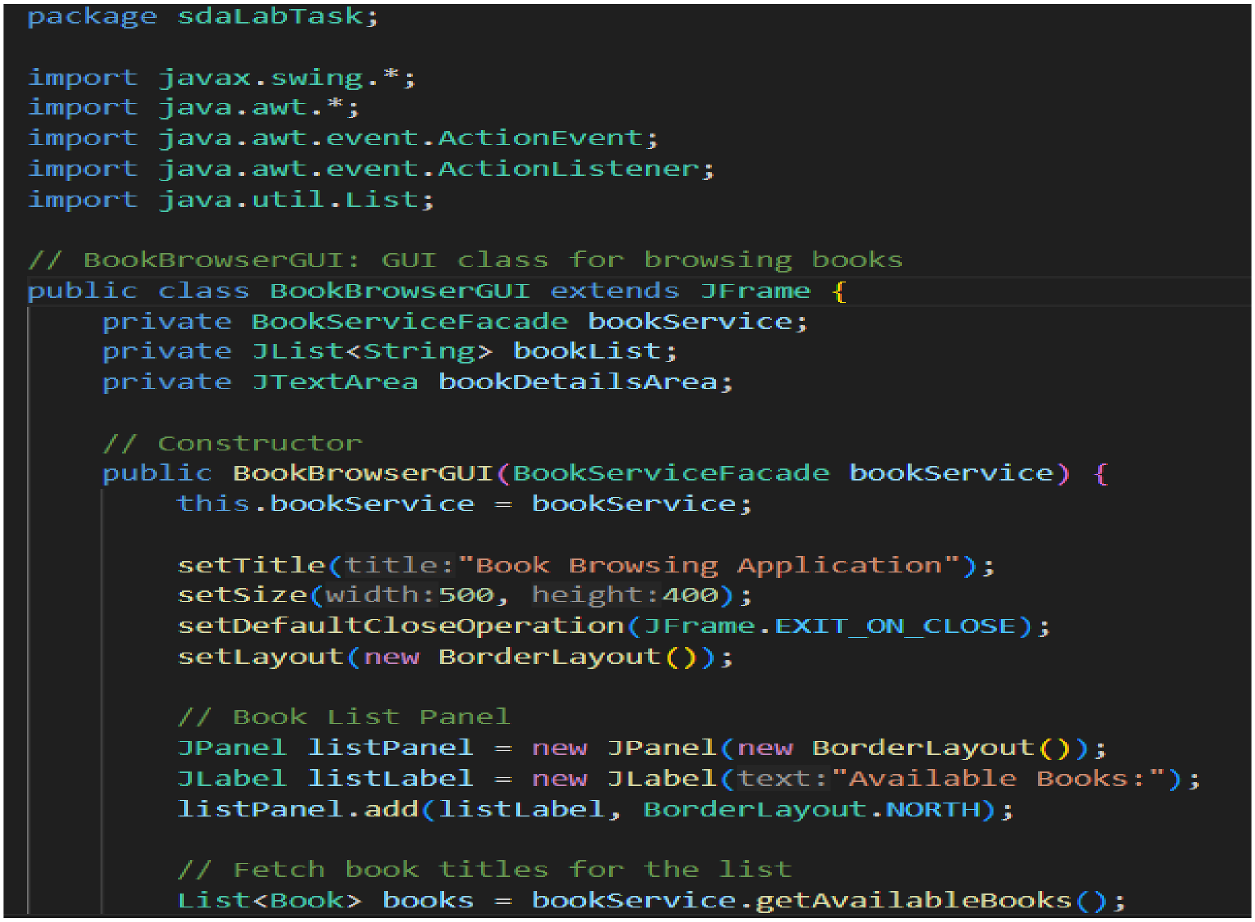
```java
    public BookServiceFacade(WebApplication webApp) {
        this.webApp = webApp;
    }

    public List<Book> getAvailableBooks() {
        return webApp.browseBooks();
    }

    public Book getBookDetails(String title) {
        return webApp.selectBook(title);
    }
}

// PRESENTATION LAYER (Handles user interaction)
class UserInterface {
    private BookServiceFacade bookService;
    private Scanner scanner;

    public UserInterface(BookServiceFacade bookService) {
        this.bookService = bookService;
        this.scanner = new Scanner(System.in);
    }

    public void start() {
        System.out.println(x:"Welcome to the Book Browsing Application!");
        displayBookList();
        selectAndDisplayBook();
        scanner.close();
    }

    private void displayBookList() {
        System.out.println(x:"Fetching list of available books...");
        List<Book> books = bookService.getAvailableBooks();
        for (Book book : books) {
            System.out.println(book);
```



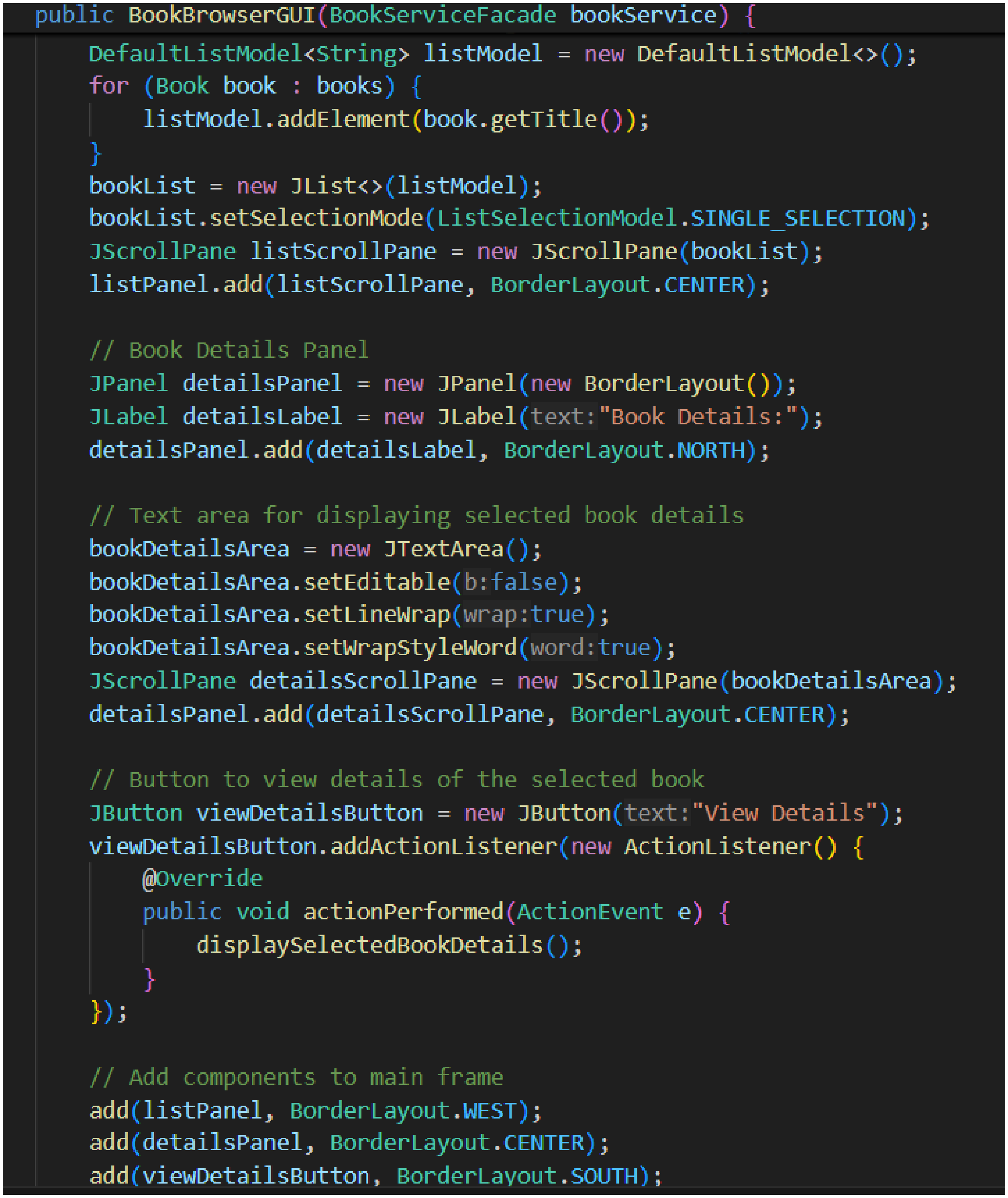
```
    private void selectAndDisplayBook() {
        System.out.print(s:"\nEnter the title of the book to view details: ");
        String title = scanner.nextLine();

        Book selectedBook = bookService.getBookDetails(title);
        if (selectedBook != null) {
            System.out.println(x:"\nBook Details:");
            System.out.println("Title: " + selectedBook.getTitle());
            System.out.println("Details: " + selectedBook.getDetails());
        } else {
            System.out.println(x:"Book not found!");
        }
    }
}

// APPLICATION LAYER (Entry Point)
public class useCase {
    Run | Debug
    public static void main(String[] args) {
        // Repository Factory to choose data source
        BookRepository bookRepository = BookRepositoryFactory.createBookRepository(type:"inMemory");

        // Singleton for WebApplication
        WebApplication webApp = WebApplication.getInstance(bookRepository);

        // Facade for Service Layer
        BookServiceFacade bookService = new BookServiceFacade(webApp);

        // Presentation Layer for User Interaction
        UserInterface userInterface = new UserInterface(bookService);

        // Start the application
        userInterface.start();
    }
}
```



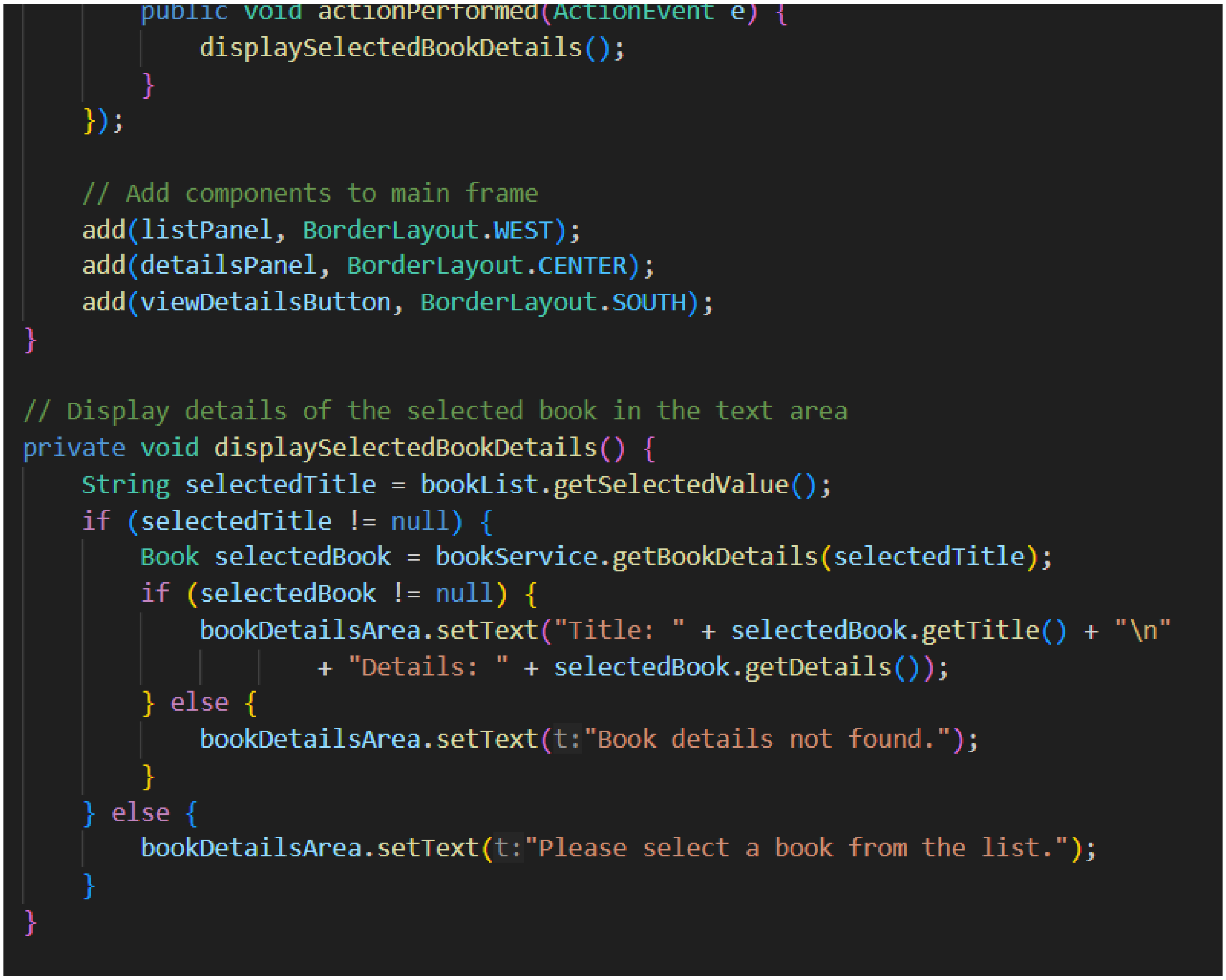
```
package sdaLabTask;

import javax.swing.SwingUtilities;

public class useCaseGUI {
    Run | Debug
    public static void main(String[] args) {
        // Repository Factory to choose data source
        BookRepository bookRepository = BookRepositoryFactory.createBookRepository(type:"inMemory");

        // Singleton for WebApplication
        WebApplication webApp = WebApplication.getInstance(bookRepository);

        // Facade for Service Layer
        BookServiceFacade bookService = new BookServiceFacade(webApp);

        // Initialize and show the GUI
        SwingUtilities.invokeLater(() -> {
            BookBrowserGUI gui = new BookBrowserGUI(bookService);
            gui.setVisible(b:true);
        });
    }
}
```



```
package sdaLabTask;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

// BookBrowserGUI: GUI class for browsing books
public class BookBrowserGUI extends JFrame {
    private BookServiceFacade bookService;
    private JList<String> bookList;
    private JTextArea bookDetailsArea;

    // Constructor
    public BookBrowserGUI(BookServiceFacade bookService) {
        this.bookService = bookService;

        setTitle(title:"Book Browsing Application");
        setSize(width:500, height:400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Book List Panel
        JPanel listPanel = new JPanel(new BorderLayout());
        JLabel listLabel = new JLabel(text:"Available Books:");
        listPanel.add(listLabel, BorderLayout.NORTH);

        // Fetch book titles for the list
        List<Book> books = bookService.getAvailableBooks();
```



```java
public BookBrowserGUI(BookServiceFacade bookService) {
    DefaultListModel<String> listModel = new DefaultListModel<>();
    for (Book book : books) {
        listModel.addElement(book.getTitle());
    }
    bookList = new JList<>(listModel);
    bookList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    JScrollPane listScrollPane = new JScrollPane(bookList);
    listPanel.add(listScrollPane, BorderLayout.CENTER);

    // Book Details Panel
    JPanel detailsPanel = new JPanel(new BorderLayout());
    JLabel detailsLabel = new JLabel(text:"Book Details:");
    detailsPanel.add(detailsLabel, BorderLayout.NORTH);

    // Text area for displaying selected book details
    bookDetailsArea = new JTextArea();
    bookDetailsArea.setEditable(b:false);
    bookDetailsArea.setLineWrap(wrap:true);
    bookDetailsArea.setWrapStyleWord(word:true);
    JScrollPane detailsScrollPane = new JScrollPane(bookDetailsArea);
    detailsPanel.add(detailsScrollPane, BorderLayout.CENTER);

    // Button to view details of the selected book
    JButton viewDetailsButton = new JButton(text:"View Details");
    viewDetailsButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            displaySelectedBookDetails();
        }
    });

    // Add components to main frame
    add(listPanel, BorderLayout.WEST);
    add(detailsPanel, BorderLayout.CENTER);
    add(viewDetailsButton, BorderLayout.SOUTH);
```



```
        public void actionPerformed(ActionEvent e) {
            displaySelectedBookDetails();
        }
    });

    // Add components to main frame
    add(listPanel, BorderLayout.WEST);
    add(detailsPanel, BorderLayout.CENTER);
    add(viewDetailsButton, BorderLayout.SOUTH);
}

// Display details of the selected book in the text area
private void displaySelectedBookDetails() {
    String selectedTitle = bookList.getSelectedValue();
    if (selectedTitle != null) {
        Book selectedBook = bookService.getBookDetails(selectedTitle);
        if (selectedBook != null) {
            bookDetailsArea.setText("Title: " + selectedBook.getTitle() + "\n"
                    + "Details: " + selectedBook.getDetails());
        } else {
            bookDetailsArea.setText(t:"Book details not found.");
        }
    } else {
        bookDetailsArea.setText(t:"Please select a book from the list.");
    }
}
```