Bookshop And Sales Management Software : Report

CSD325 - Software Engineering

Aditya Raghuram, Arnav Aditya, Sanskar Sugandhi, Rachit Anand, Arnav Jalan

2210110126, 2210110189, 2210110898, 2210110487, 2210110898

May 5, 2025

Contents

List of Figures

1 Problem Statement (as given)

Bookshop Inventory and Sales Management System (BAS): It should help the customers query whether a book is in stock. The users can query the availability of a book either by using the book title or by using the author's name. If the book is not currently sold by the bookshop, then the customer is asked to enter full details of the book for procurement of the book in future. The customer can also provide his e-mail address, so that he can be intimated automatically by the software as and when the book copies are received. If a book is in stock, the exact number of copies available and the rack number in which the book is located should be displayed. If a book is not in stock, the query for the book is used to increment a request field for the book. The manager can periodically view the request field of the books to arrive at a rough estimate regarding the current demand for different books. BAS should maintain the price of various books. As soon as a customer selects his books for purchase, the salesclerk will enter the ISBN numbers of the books. BAS should update the stock and generate the sales receipt for the book. BAS should allow employees to update the inventory whenever a new supply arrives. Also, upon request by the owner of the book shop, BAS should generate sales statistics (viz., book name, publisher, ISBN number, number of copies sold, and the sales revenue) for any period. The sales statistics will help the owner to know the exact business done over any period and also to determine inventory level required for various books. The inventory level required for a book is equal to the number of copies of the book sold over a period of one week multiplied by the average number of weeks it takes to procure the book from its publisher. Every day the book shop owner would give a command for the BAS to print the books which have fallen below the threshold and the number of copies to be procured along with the full address of the publisher.

2 Software Requirements Specification (SRS)

Attached as SRS.pdf. Kindly refer to the file in the archive.

3 Functional Requirements

3.1 Customer Interface Requirements

- Book availability query system with real-time status display
- Search functionality by title, author, genre, and publication date
- Request system for books not in stock
- Automated notification system for book availability updates
- Form for capturing detailed book information for procurement

3.2 Sales Management Requirements

- Sales transaction processing with receipt generation
- Support for both printed and digital receipts
- Integration with inventory for automatic stock updates after sales

- Batch processing for new stock entries

3.3 Inventory Management Requirements

- Real-time inventory tracking and updates
- Automated low-stock alerts and reports
- Book procurement request management
- Stock level verification against purchase orders

3.4 Reporting Requirements

- Export functionality in multiple formats (PDF, CSV)
- Daily reports for low-stock books

4 Non-Functional Requirements

4.1 Usability Requirements

- Consistent UI design across all web pages
- Product images in catalog

4.2 Security Requirements

- Secure socket implementation for confidential transactions
- Automatic session timeout after inactivity
- Password masking and encryption
- Role-based access control
- Encrypted database storage
- Limited display of sensitive information (e.g., tokens)

4.3 Interface Requirements

- RESTful API architecture
- Integration with payment gateways such as Stripe

5 DFD (Data Flow Diagram) Model

5.1 Level 0 DFD (Context Diagram)

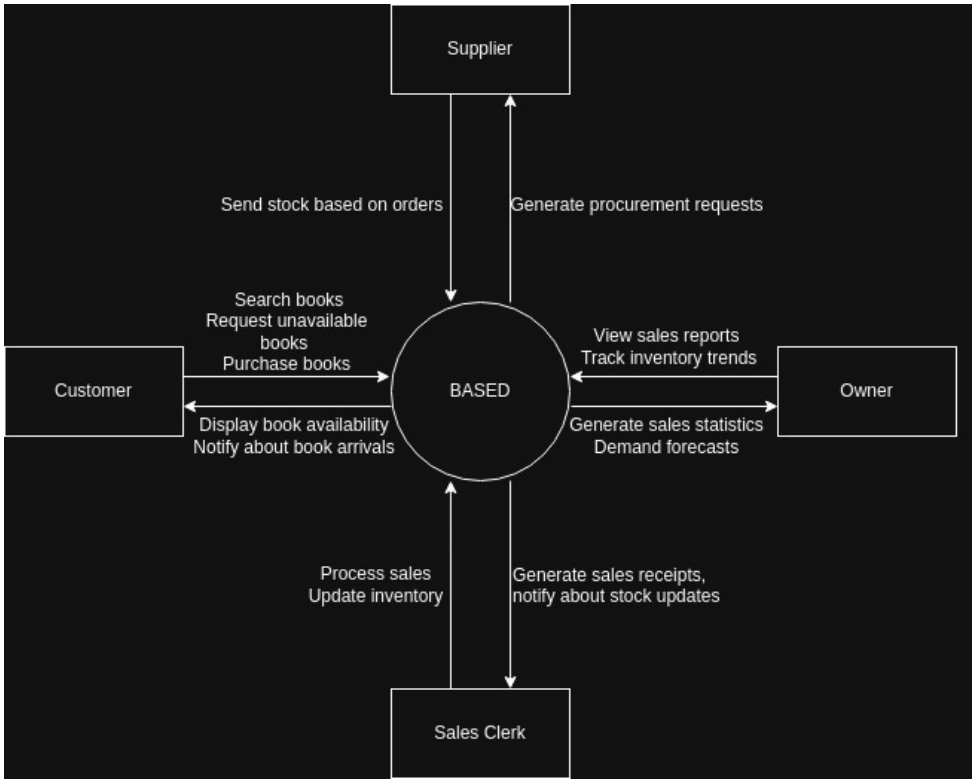

Figure 1: Level 0 Data Flow Diagram (Context Diagram).

5.2 Level 1 DFD

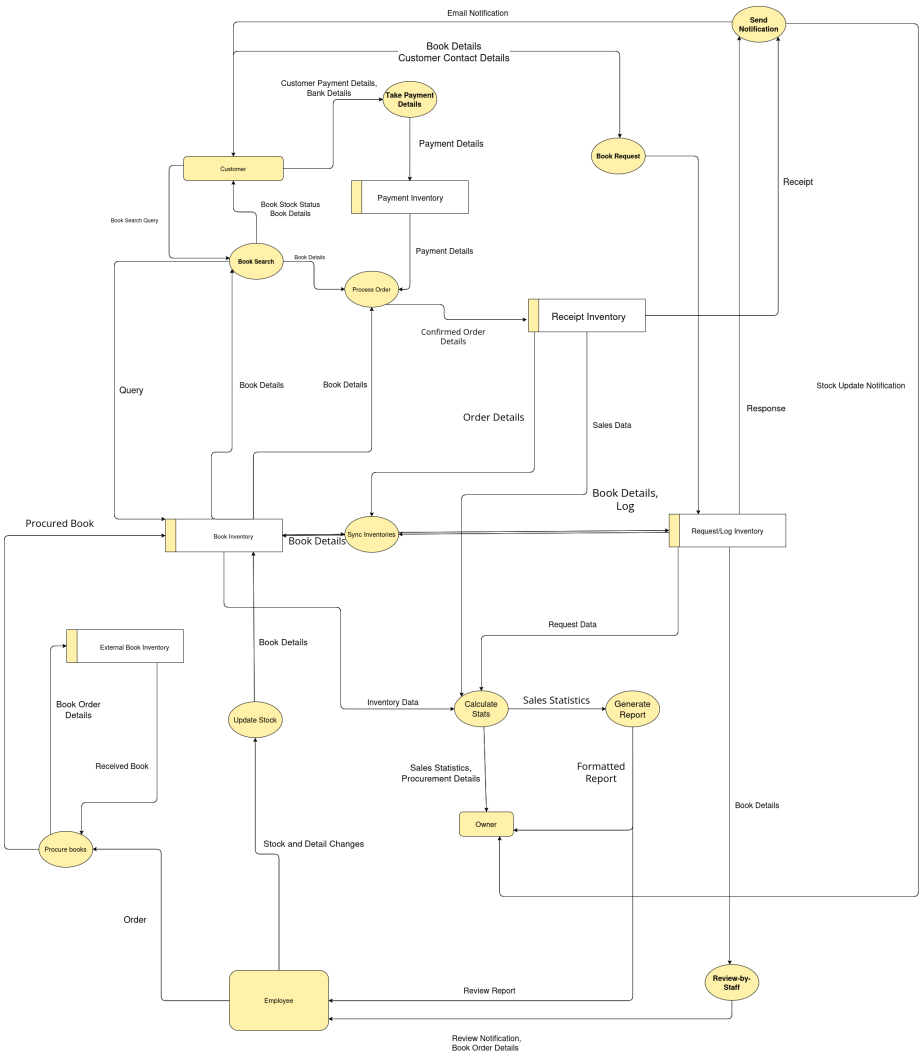

Figure 2: Level 1 Data Flow Diagram.

6 Use Case Documentation

Kindly refer to the Use Case document attached in the archive.

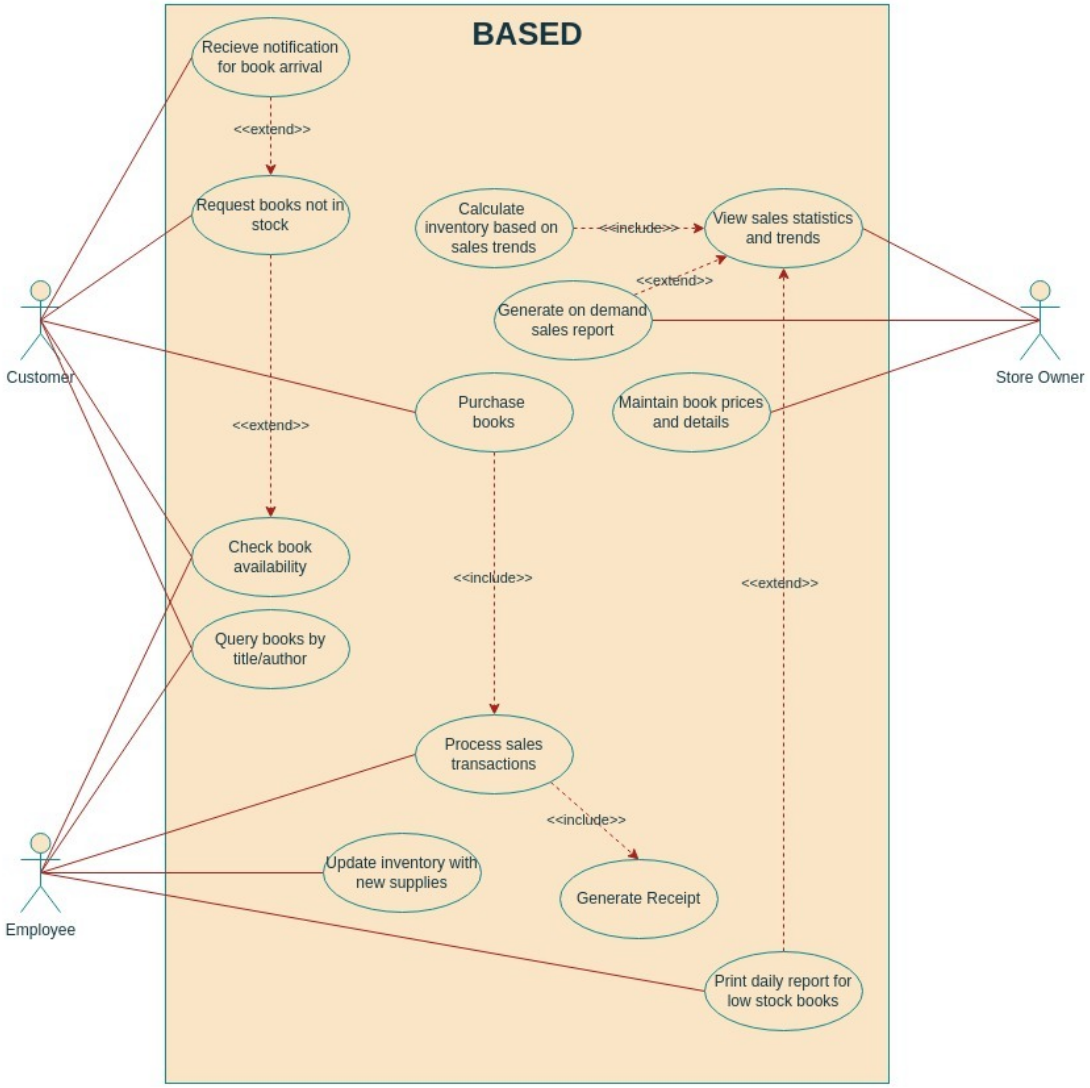

Figure 3: Use Case Diagram

6.1 Use Case Documentation

Use Case: Check Book Availability

ID: UC-001
Use Case Name: Check Book Availability
Actor(s): Customer, Employee
Description: Allows a customer to search for a book in the system to check its availability. The system provides real-time status updates on whether a book is in stock, low stock, or out of stock.

Use Case: Request Books Not in Stock

ID: UC-002
Use Case Name: Request Books Not in Stock
Actor(s): Customer, Inventory Manager
Description: This use case describes the process by which a customer requests a book that is currently out of stock. The system captures the customer's contact details, logs the request for future procurement, and notifies the customer upon availability.

Use Case: Query Books by Title and/or Author

ID: UC-003
Use Case Name: Query Books by Title and/or Author
Actor(s): Customer, Employee
Description: This use case allows a customer or employee to search for books in the system by entering a title, author, or both. The system will return a list of books that match the search criteria, displaying relevant details such as publication year, price, and availability status.

Use Case: Receive Notifications for Book Arrival

ID: UC-004
Use Case Name: Receive Notifications for Book Arrival
Actor(s): Customer, Book Inventory System
Description: This use case allows a customer to receive notifications when a previously unavailable book arrives in stock. The system monitors book inventory and sends an automated alert (via email) to customers who requested the book.

Use Case: Purchase Books

ID: UC-005
Use Case Name: Purchase Books
Actor(s): Customer
Description: This use case describes the process by which a customer purchases books from the system. The system processes the transaction, updates the inventory, and generates a receipt.

Use Case: Process Sales Transaction

ID: UC-006
Use Case Name: Process Sales Transaction
Actor(s): Employee, Inventory Management System
Description: Allows an employee to complete a book sale transaction in the system. The system verifies the selected books, calculates the total cost (including taxes and discounts), and processes the payment. Upon successful completion, a receipt is generated for the customer, and the inventory is updated accordingly.

Use Case: Generate Receipt

ID: UC-007
Use Case Name: Generate Receipt
Actor(s): Employee, System
Description: This use case allows bookstore staff to generate a receipt after a successful purchase. The receipt includes details such as book titles, quantities, prices, taxes, total amount, and payment method. It can be printed or sent digitally to the customer.

Use Case: Update Inventory with New Supplies

ID: UC-008
Use Case Name: Update Inventory with New Supplies
Actor(s): Employee, System
Description: This use case allows employees to update the book inventory when new stock arrives. The system will adjust stock levels, record batch numbers, and update availability status to reflect the latest inventory changes.

Use Case: Print Daily Reports for Low-Stock Books

ID: UC-010
Use Case Name: Print Daily Reports for Low-Stock Books
Actor(s): Employee
Description: This use case describes the process by which an employee generates and prints a daily report of books that have fallen below a predefined stock threshold.

7 Domain Model

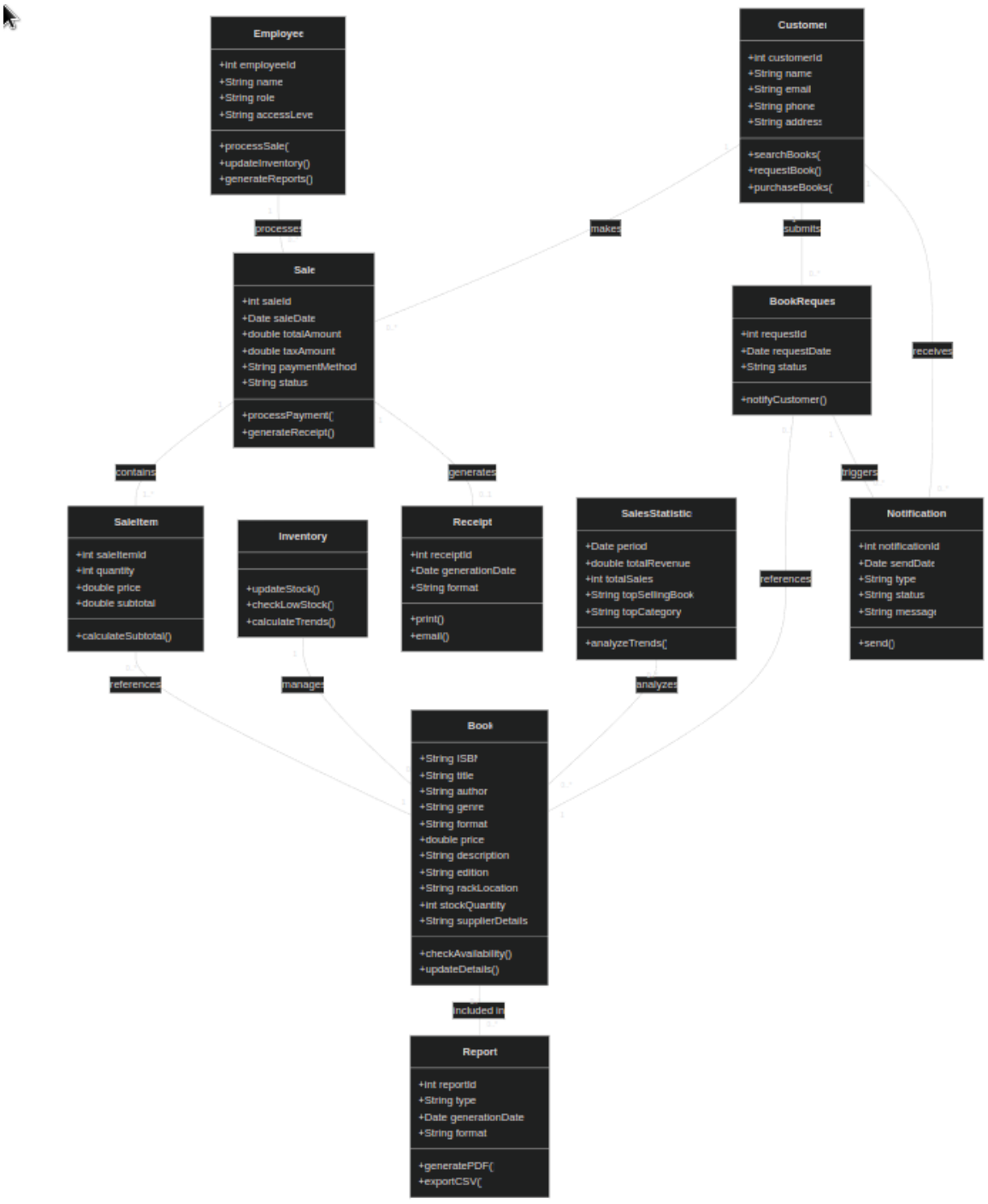

Figure 4: Domain Model Diagram.

8 General Sequence Diagram

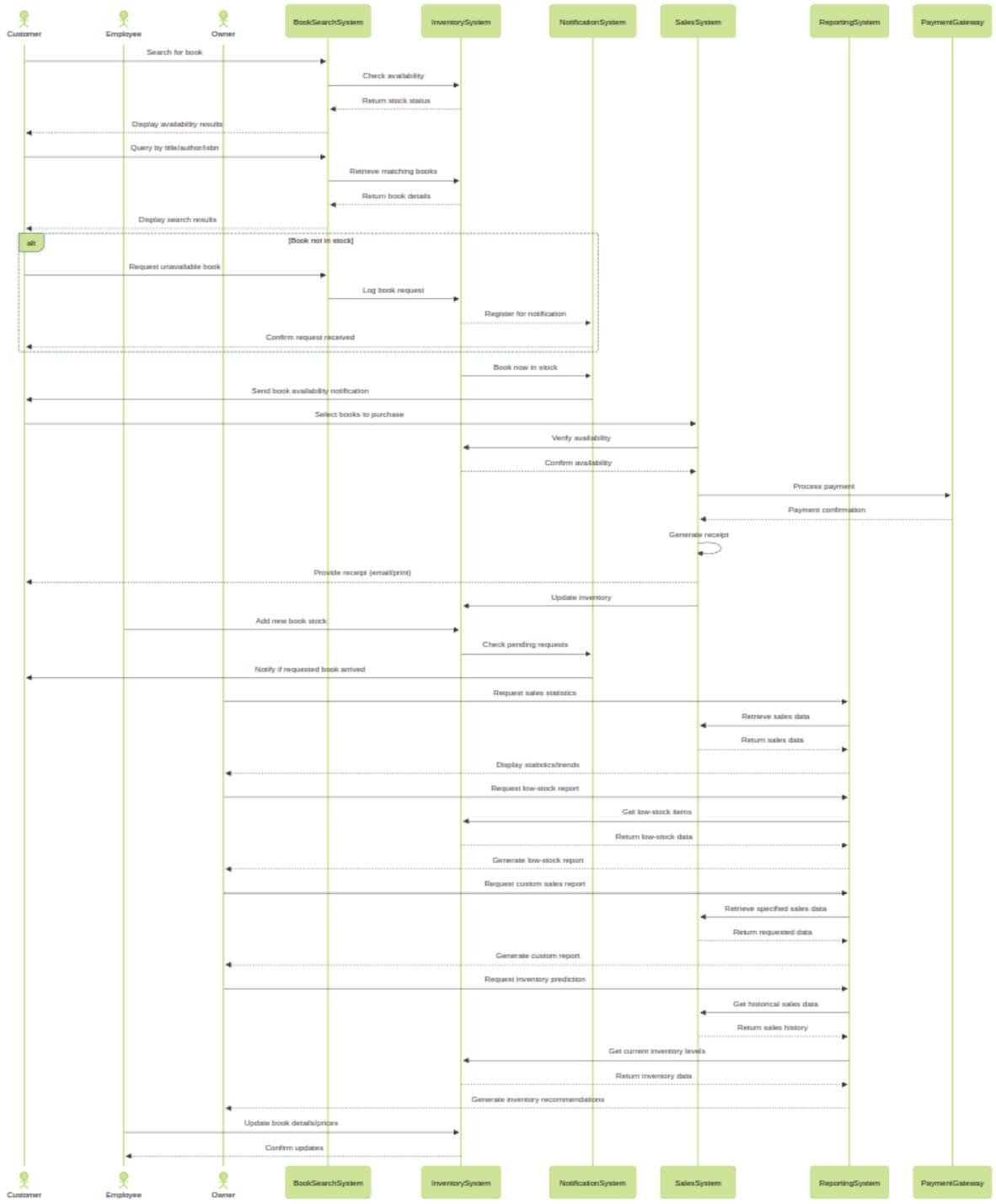

Figure 5: Sequence Diagram.

9 System Architecture and Implementation

9.1 Technologies Used

The backend is built using Java and the Spring ecosystem, while the frontend utilises React.

Backend Technologies

- **Java 21:** The core programming language used for backend development, specified by the ‘¡java.version¿‘ property.
- **Spring Boot 3.4.3:** The core framework providing rapid application development features for building RESTful APIs and web services. Version specified in the ‘¡parent¿‘ tag.
- **Spring Boot Starters:** Leveraging modules for specific abilities:
  - *spring-boot-starter-web:* For building web applications.
  - *spring-boot-starter-data-jpa:* For data persistence using the Java Persistence API (JPA), simplifying database interactions.
  - *spring-boot-starter-security:* For handling authentication, authorisation, and other security aspects.
  - *spring-boot-starter-validation:* For applying bean validation constraints.
  - *spring-boot-starter-websocket:* For enabling real-time, bidirectional communication channels.
  - *spring-boot-starter-mail:* For integrating email sending capabilities.
- **MySQL Connector/J:** The JDBC driver enabling the application to connect and interact with a MySQL database at runtime.
- **JSON Web Token (JJWT) 0.11.5:** Libraries used for creating and validating JWTs (API security).
- **Project Lombok 1.18.38:** Utility annotation library.
- **OpenPDF 1.3.26:** For creating and manipulating PDF documents, used for generating sales receipts & reports.
- **OpenCSV 5.9:** Parsing and writing CSV (Comma Separated Values) files.
- **Stripe Java Client 29.0.0:** Interacting with the Stripe API, for payment processing.
- **Apache Maven:** Build automation and dependency management tool.

Frontend Technologies

- **React 19:** Core JavaScript library for building user interfaces.
- **Vite 6.2:** Frontend build tool and development server.
- **Tailwind CSS 4.0:** CSS framework used for rapid styling the application components.
- **ShadCN UI Components:** Utilises a collection of composable UI components built on top of Radix UI primitives.
- **Lucide React 0.487:** SVG icones used throughout.

- **Axios 1.8:** Promise-based HTTP client used for API requesting to the backend services.
- **Framer Motion 12.6:** Animation library.
- **Stripe.js 7.1:** Integrating Stripe payment elements in the frontend.

Frontend Development Tools

- **Vite React Plugin ('@vitejs/plugin-react'):** Enabling React optimisations within the Vite environment.

Testing Technologies

- **Spring Boot Starter Test:** Core testing utilities:
  - **JUnit:** Primary framework for writing and running unit & integration tests.
  - **Spring Test & Spring Boot Test:** Support for loading application context and testing Spring components.
- **Spring Security Test:** Provides utilities specifically for testing Spring Security configurations and secured endpoints.
- **H2 Database:** An in-memory database used specifically during the testing phase to run tests without needing a full external MySQL instance.

10 Testing and Results

10.1 Testing Strategy

The testing strategy adopted in this project primarily revolves around **unit testing** of the repository layer using the `@DataJpaTest` annotation provided by the Spring Boot Test framework. This approach ensures that the JPA-based repositories interact correctly with the in-memory database (H2 in this case) and accurately reflect the state and behavior of the actual database layer.

JUnit 5 (via `org.junit.jupiter`) was used as the core testing framework, while **AssertJ** assertions were employed for fluent and readable test validations. The testing also leverages Spring Boot's auto-configuration for JPA tests, allowing fast and isolated tests of the repository logic.

Each repository was tested with a focus on verifying the correctness of custom query methods, entity relationships, and data persistence/retrieval logic. The tests simulate realistic scenarios, including both positive (data exists) and negative (data does not exist) cases, to ensure robustness.

10.2 BlacklistedJWTToken Repository Tests

This repository manages the blacklisted JWT tokens used for security purposes. The following scenarios were tested:

- **Token Existence Check:** Verified that the repository method `existsByToken(String token)` returns `true` when a token exists and `false` otherwise.

- **Expiry Date Persistence:** Ensured that the expiry date is correctly saved and retrieved without time drift, using millisecond comparison for accuracy.

10.3 Book Repository Tests

The Book repository supports complex search functionality. The tests covered:

- **Search by ISBN:** Checked that the method `findByIsbn(String isbn)` returns the expected book or empty result.
- **Case-Insensitive Search:** Confirmed that both `findByAuthorNameContainingIgnoreCase` and `findByBookNameContainingIgnoreCase` return results regardless of input case.
- **Combined Filtering:** Verified that `findByAuthorNameContainingIgnoreCaseAndBookNameCon` accurately filters books by both author name and book name.

10.4 Order Repository Tests

The Order repository involves more complex relationships, such as users and order items. The tests included:

- **Session ID Check:** Confirmed that `existsBySessionId(String sessionId)` identifies existing sessions correctly.
- **Order Sorting:** Ensured that `findAllByUserOrderByCreatedDateDesc(User user)` returns orders in descending order of creation time.
- **User-Order Association:** Validated that orders are properly linked to their respective users.
- **Order Items Retrieval:** Tested that all items related to a specific order are retrieved and linked to the correct books and order.

Overall, these tests validate the correctness of repository logic, data persistence, and relationships across entities in a controlled test environment, increasing confidence in data access layer integrity.

11 References

References

[1] Spring Team / VMware, *Spring Boot Reference Documentation*. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/

[2] Spring Team / VMware, *Spring Data JPA Reference Documentation*. [Online]. Available: https://docs.spring.io/spring-data/jpa/reference/html/

[3] Spring Team / VMware, *Spring Security Reference Documentation*. [Online]. Available: https://docs.spring.io/spring-security/reference/index.html

[4] Jakarta EE Maintainers, *Jakarta Bean Validation Specification*. [Online]. Available: https://beanvalidation.org/specification/

[5] Mozilla Developer Network, *MDN Web Docs - CSS: Cascading Style Sheets*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS

[6] Tailwind Labs, *Tailwind CSS Documentation*. [Online]. Available: https://tailwindcss.com/docs

[7] Oracle Corporation, *Java SE 21 Documentation*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/

[8] shadcn, *shadcn/ui Documentation*. [Online]. Available: https://ui.shadcn.com/docs

[9] React Team / Meta Platforms, Inc., *React Documentation*. [Online]. Available: https://react.dev/

[10] Stack Exchange Inc., *Stack Overflow*. Community Q&A Website. [Online]. Available: https://stackoverflow.com/ (Note: Citing specific influential Q&A threads is preferable if applicable.)

[11] Medium, *Medium.com*. Online Publishing Platform. [Online]. Available: https://medium.com/

A Application Screenshots

A.1 Commit Tree

A.2 Demonstration

A.2.1 Backend Screenshots

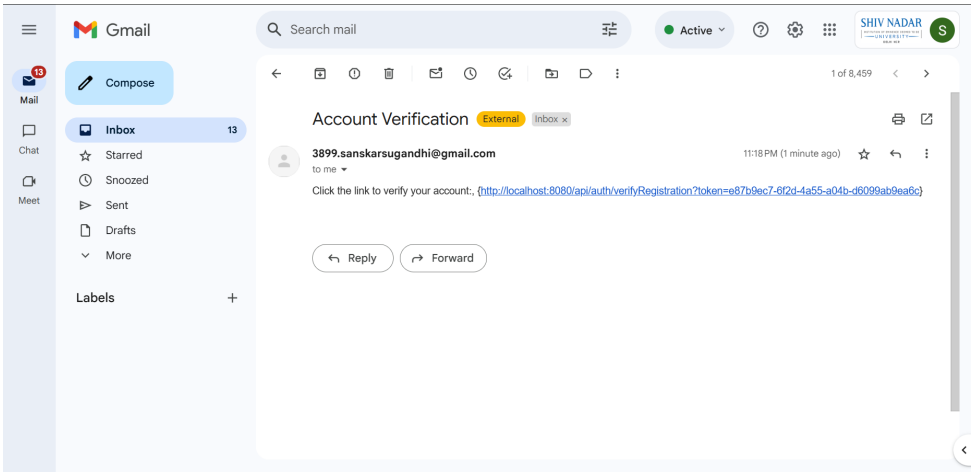

Figure 7: Account verification email

```
mysql> select * from users;
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
| id | email                       | enabled | password                                                     | profile_pic
ture | status | username |
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
|  1 | 3899.sanskarsugandhi@gmail.com | 0x00    | $2a$11$VmwF/abLssrVoEQloEQg2uHIWkoYd8P/k5uHK8yayS9O8tZtrBz4e | NULL
     | NULL   | Sanny    |
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
1 row in set (0.03 sec)

mysql> select * from users;
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
| id | email                       | enabled | password                                                     | profile_pic
ture | status | username |
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
|  1 | 3899.sanskarsugandhi@gmail.com | 0x00    | $2a$11$VmwF/abLssrVoEQloEQg2uHIWkoYd8P/k5uHK8yayS9O8tZtrBz4e | NULL
     | NULL   | Sanny    |
|  2 | ss736@snu.edu.in            | 0x01    | $2a$11$760tR7r9m2ZsQzOIuDiJOORoo4cXHs3iKKex5jvEFIF2CAK8nZzt6 | NULL
     | NULL   | Sanskar  |
+----+-----------------------------+---------+--------------------------------------------------------------+-------------
-----+--------+----------+
2 rows in set (0.00 sec)
```

Figure 8: All users (enabled and not enabled)

Figure 6: GitHub commits

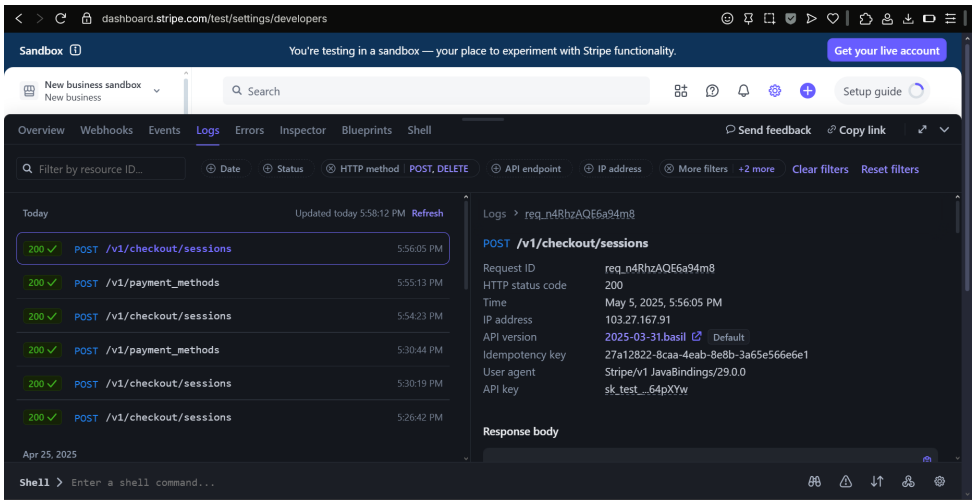

Figure 9: Stripe dashboard JSON output 1

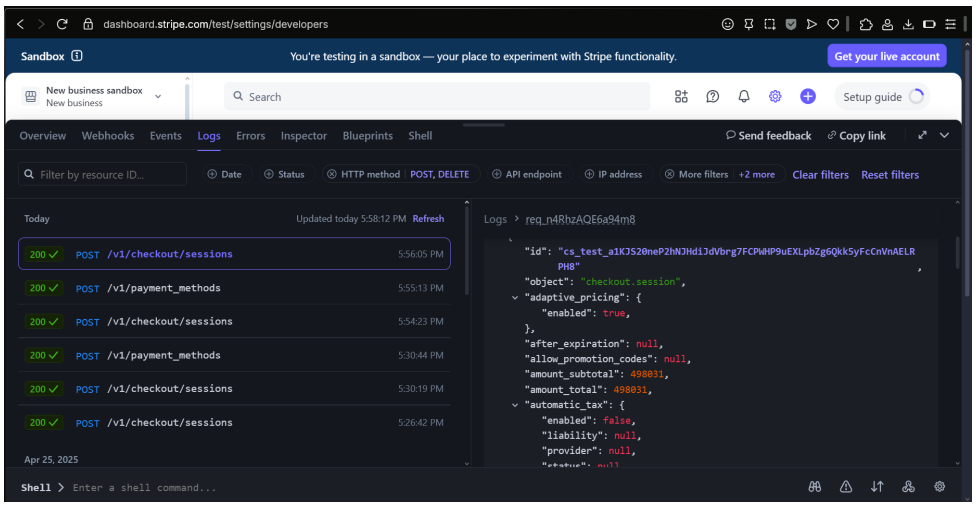

Figure 10: Stripe dashboard JSON output 2

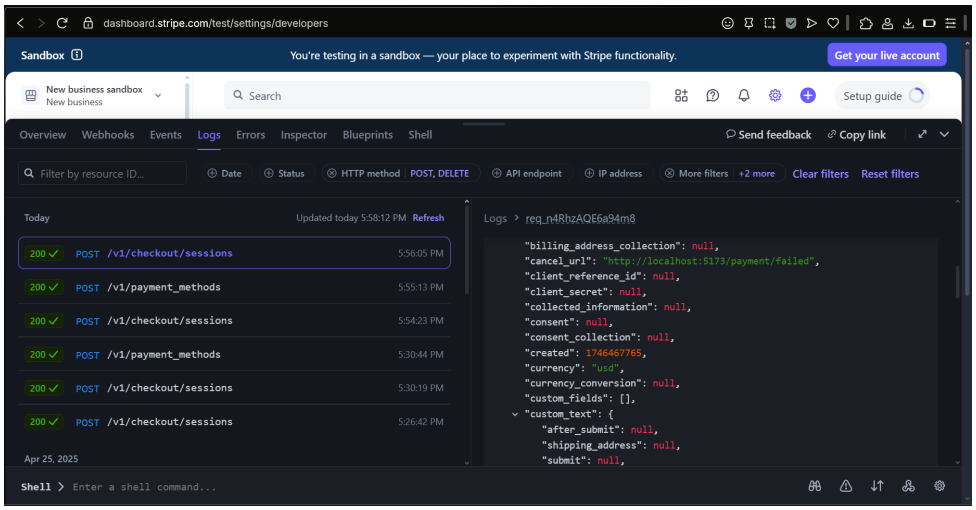

Figure 11: Stripe dashboard JSON output 3

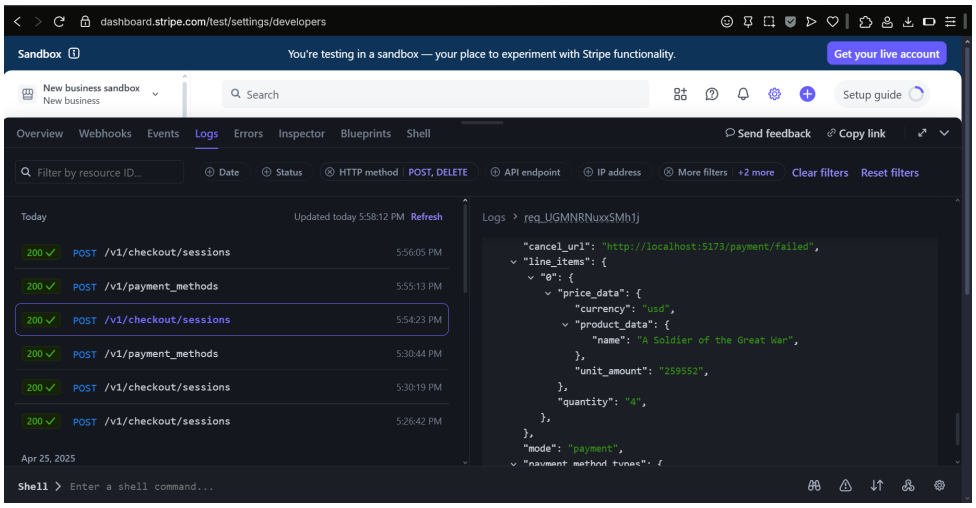

Figure 12: Stripe dashboard JSON output 4

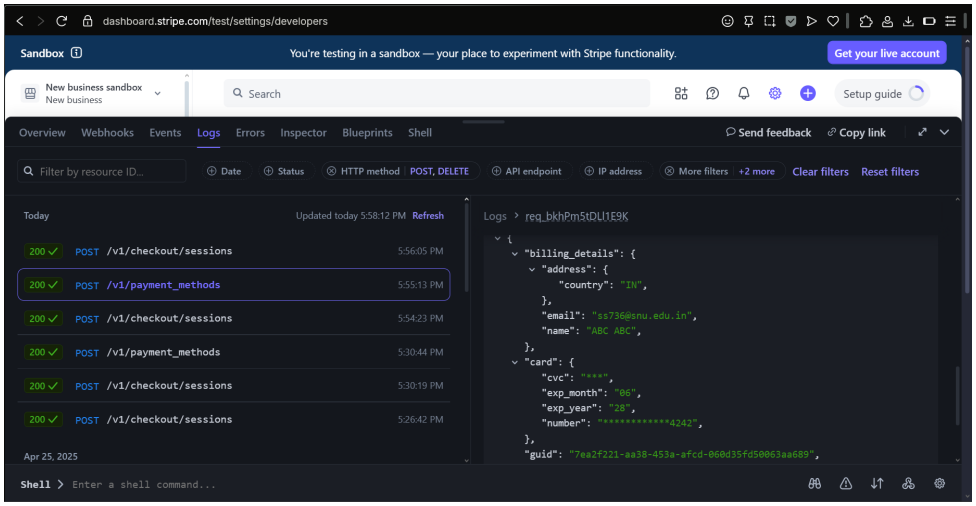

Figure 13: Stripe dashboard JSON output 5

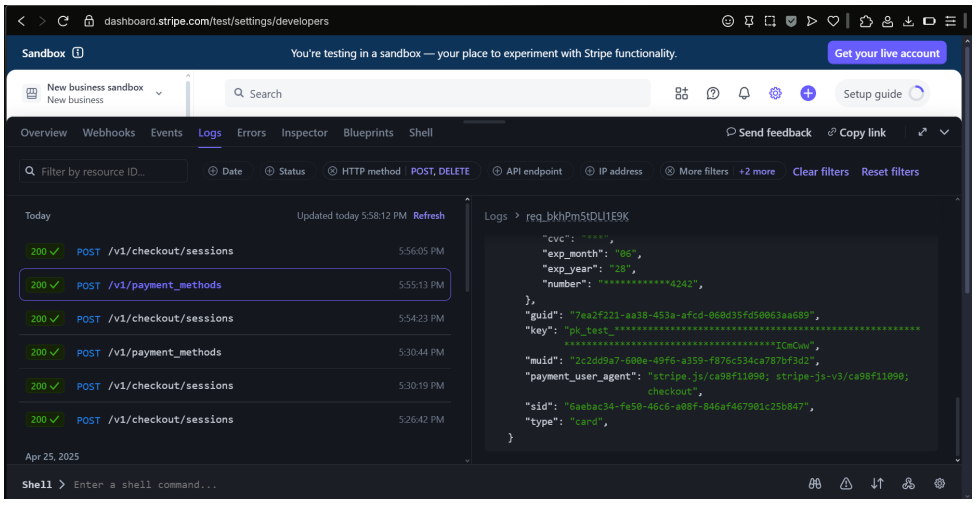

Figure 14: Stripe dashboard JSON output 6

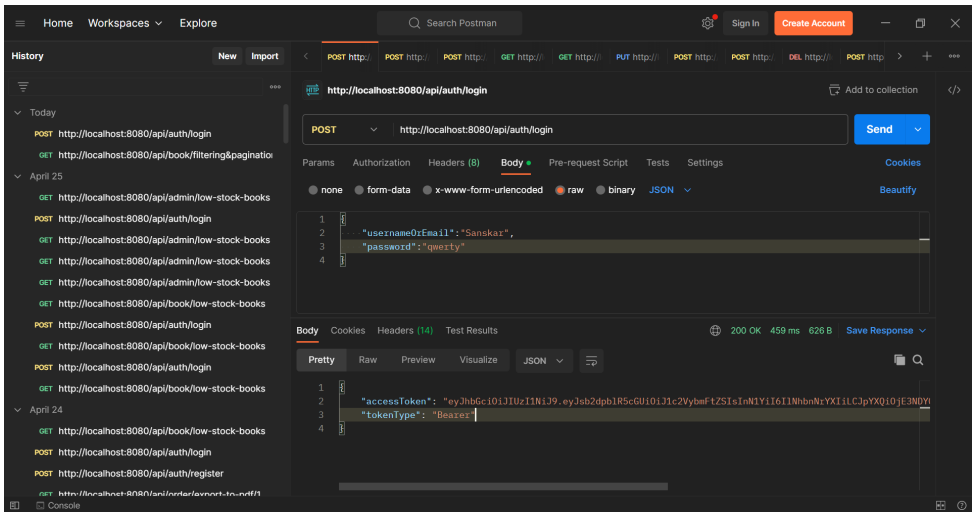

Figure 15: Postman login JWT token response

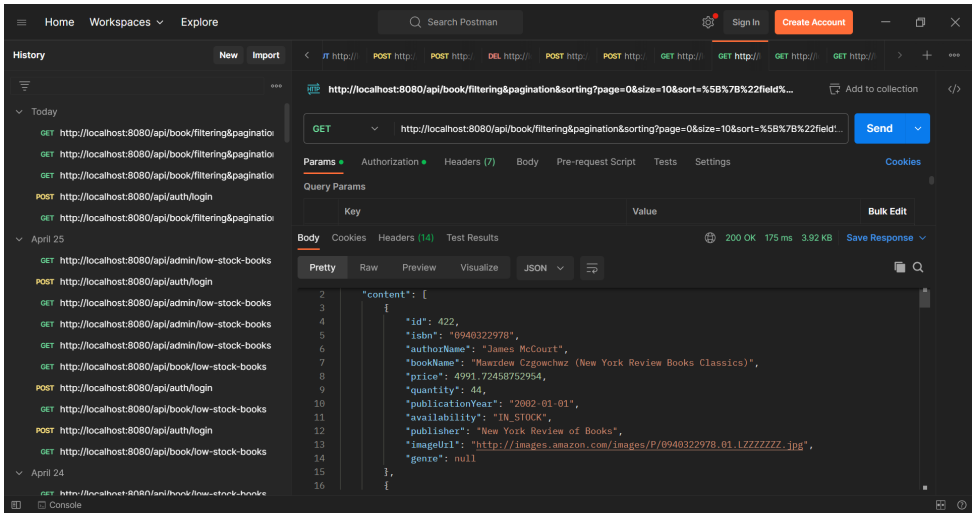

Figure 16: Postman pagination, filtering and sorting response

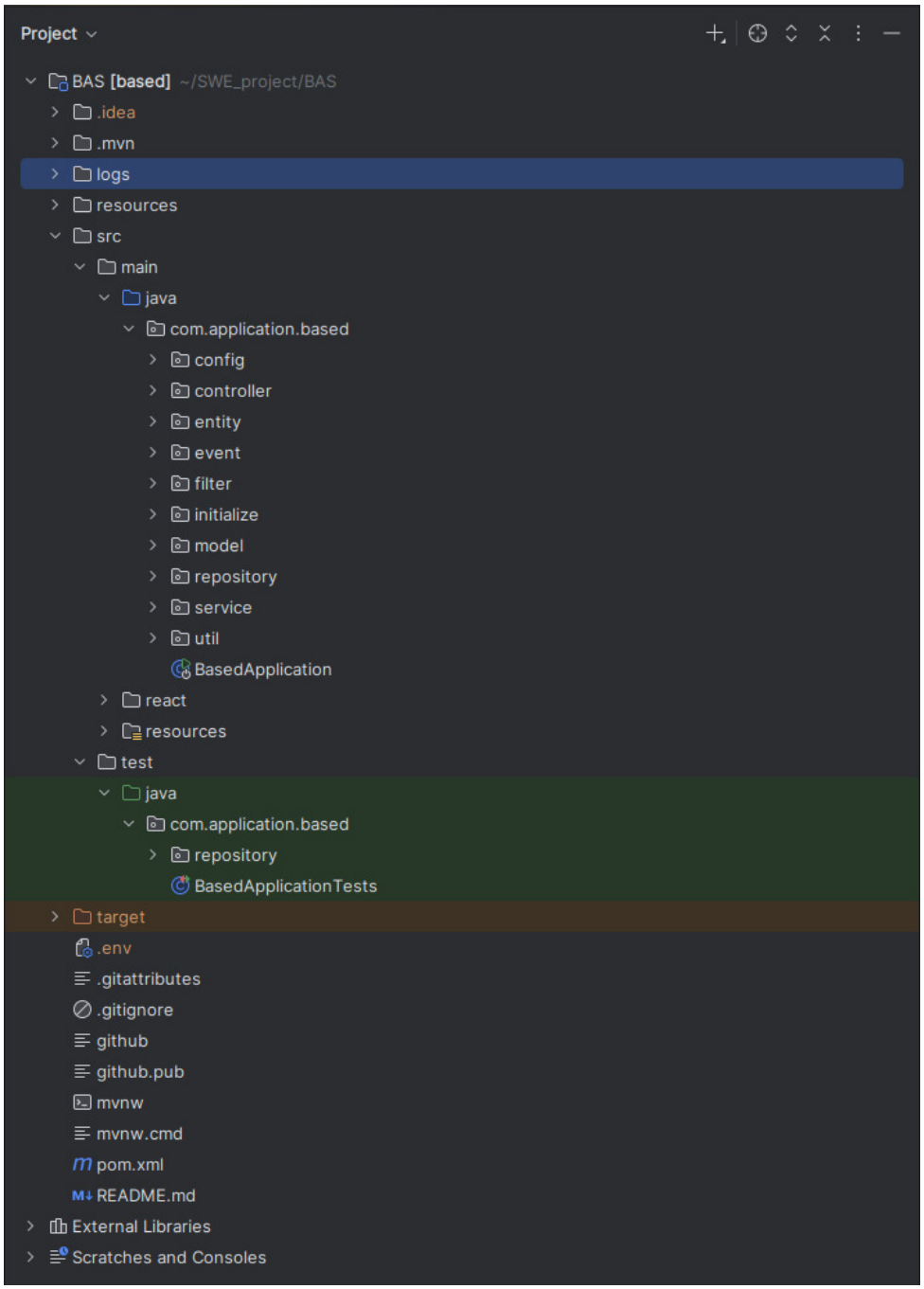

Figure 17: project structure

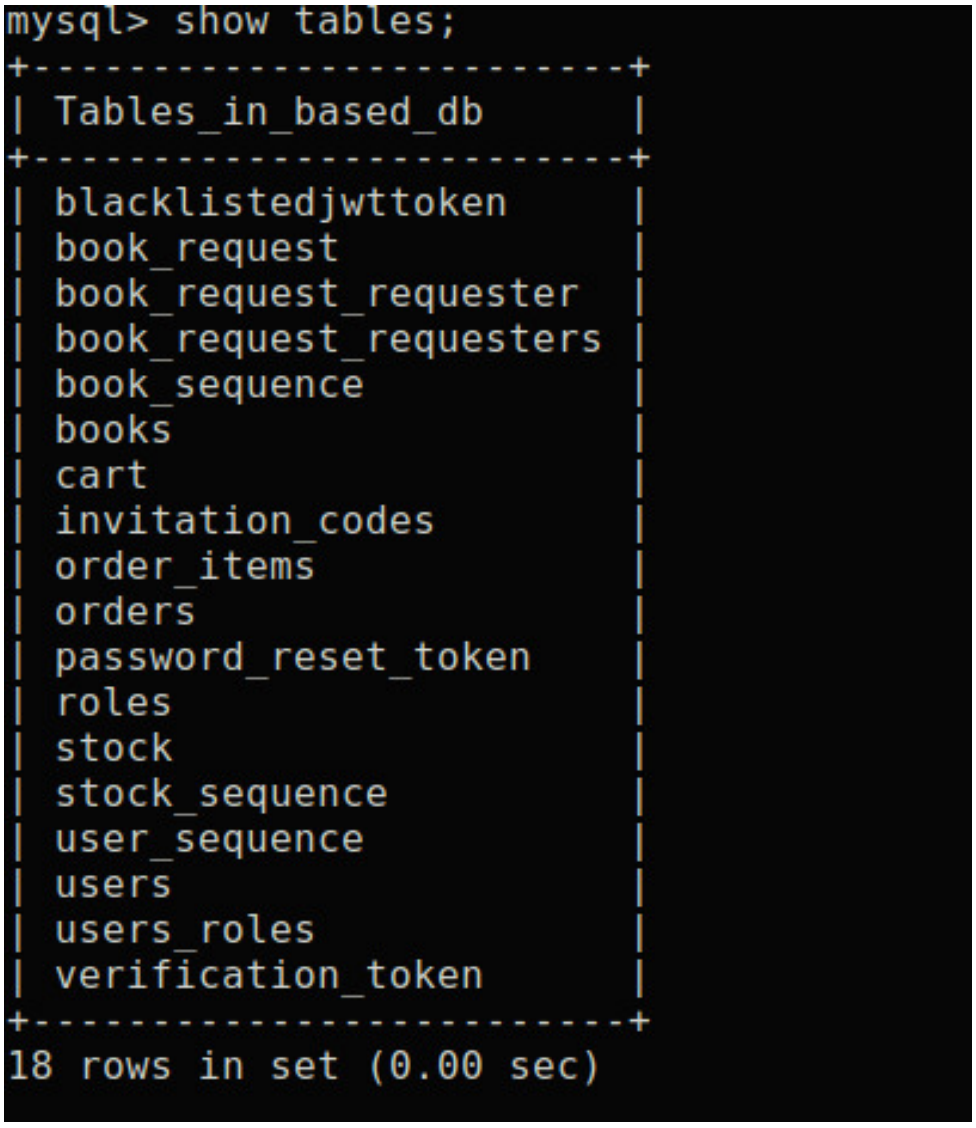

Figure 18: All DB relations

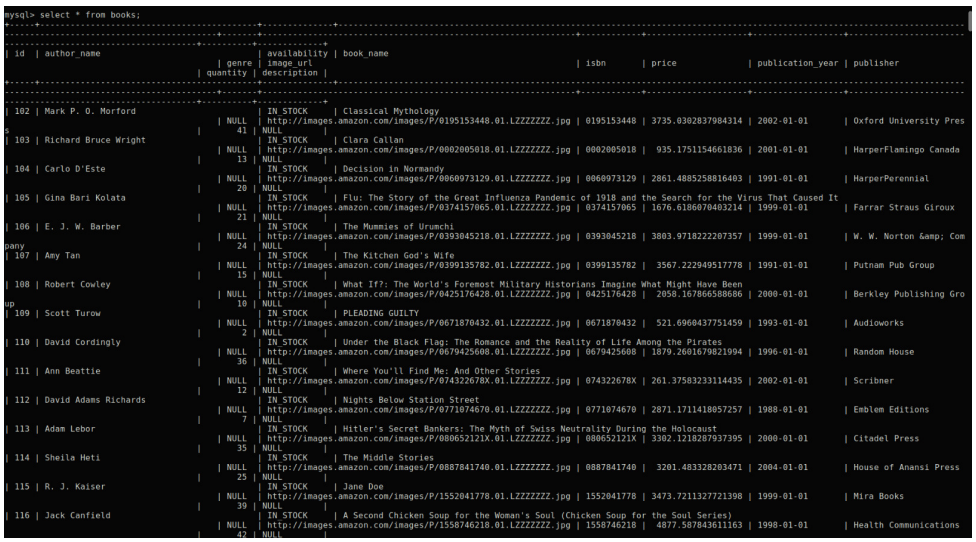

Figure 19: All available books in DB

A.2.2 Frontend Screenshots

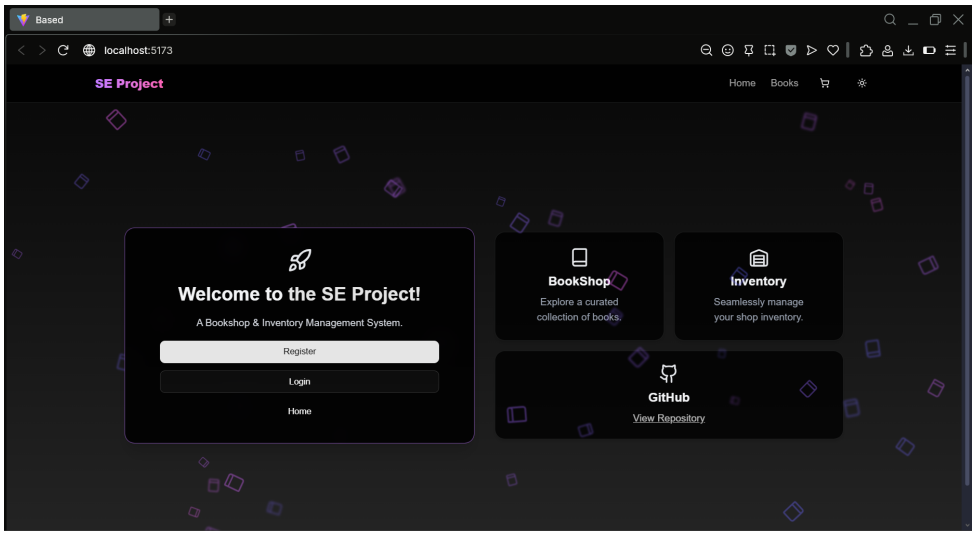

Figure 20: Landing page interface

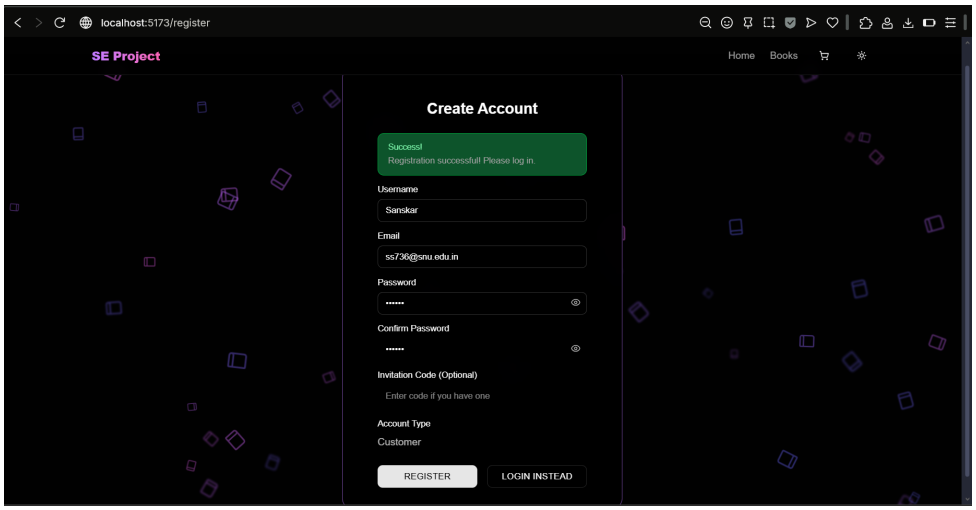

Figure 21: Register page interface

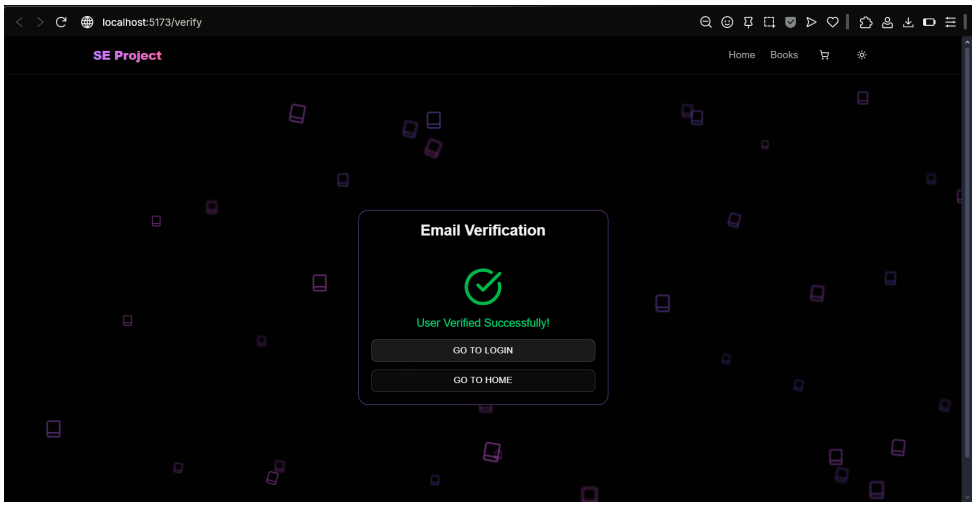

Figure 22: Email verification success page

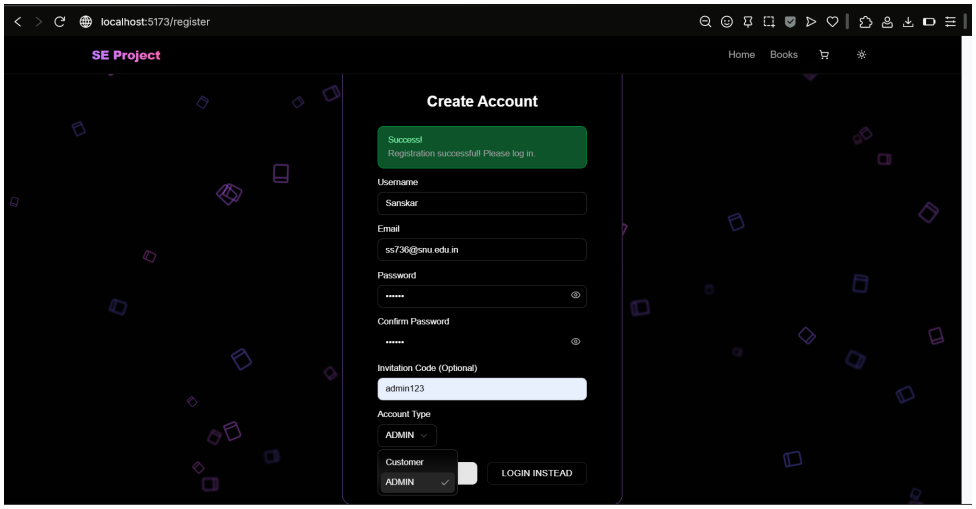

Figure 23: Creating an account with invitation code for admin

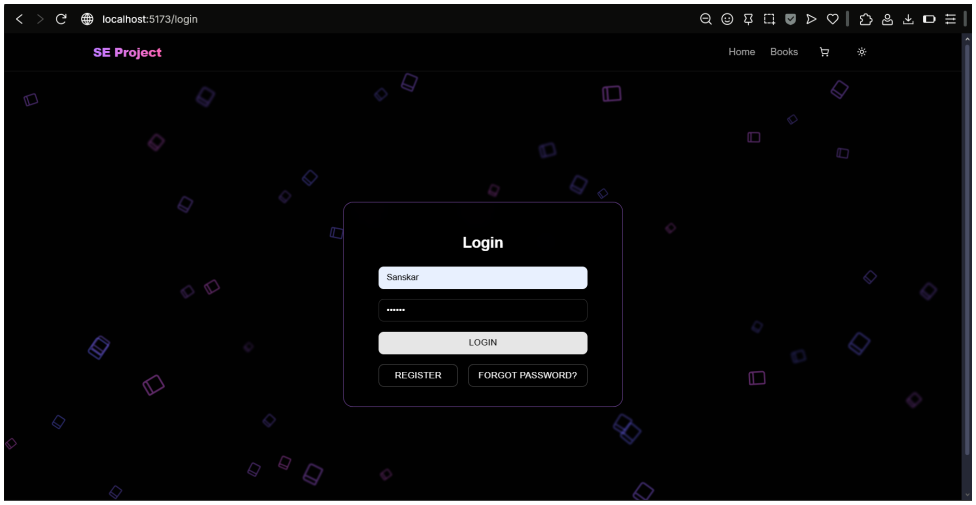

Figure 24: Login page

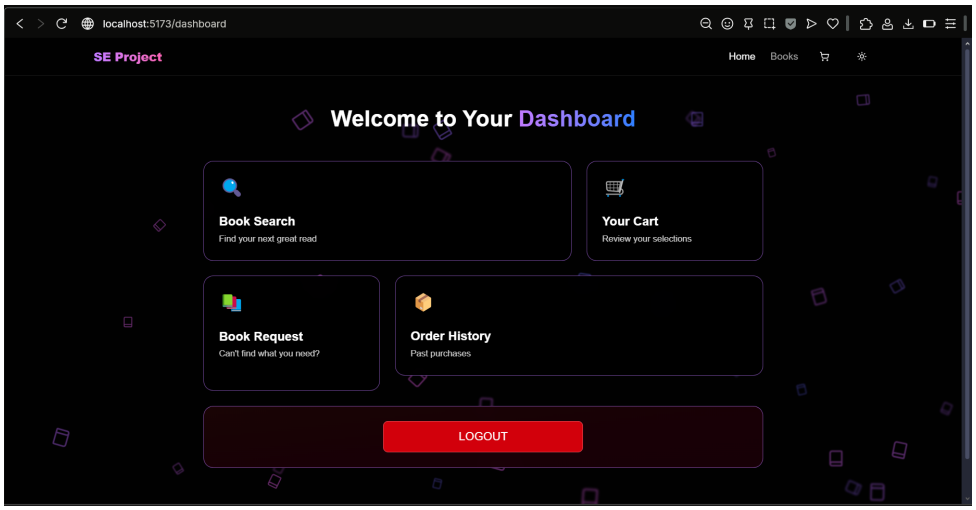

Figure 25: User dashboard

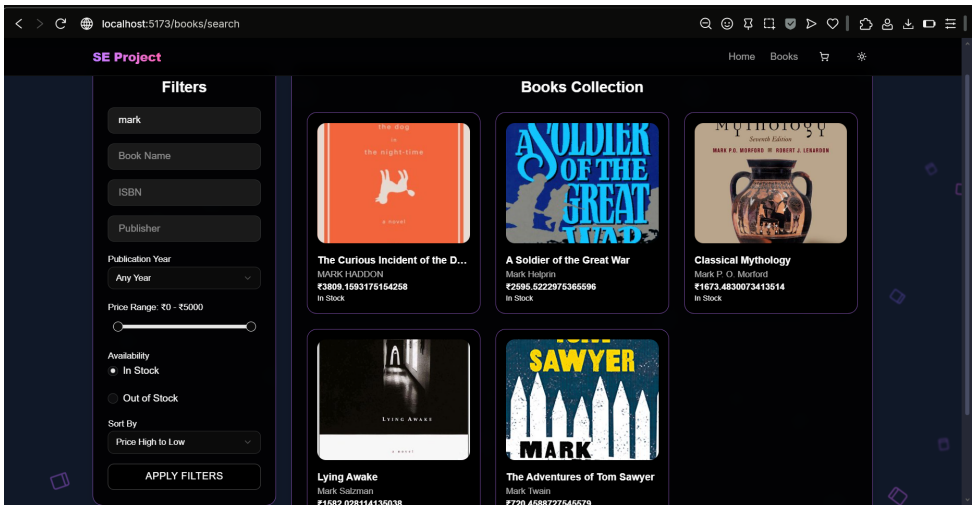

Figure 26: Book search and filter page

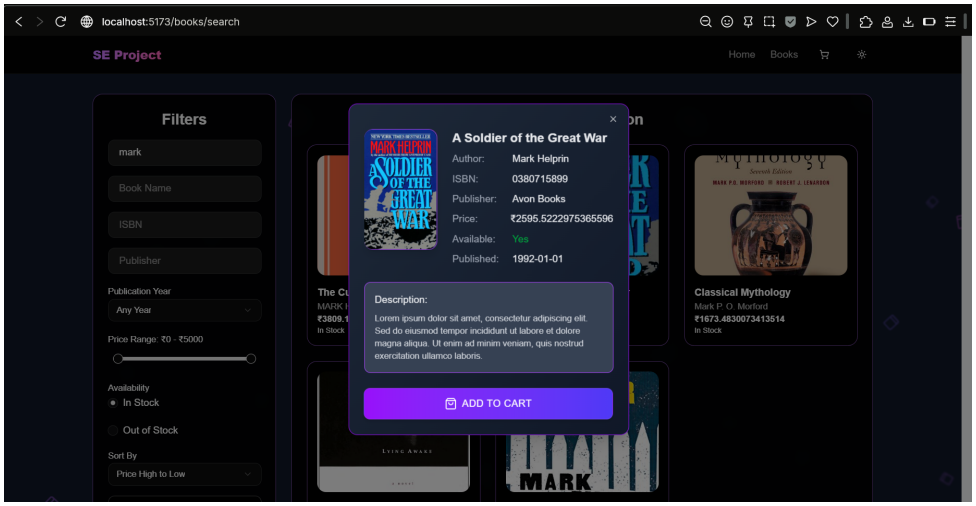

Figure 27: Add to cart dialog

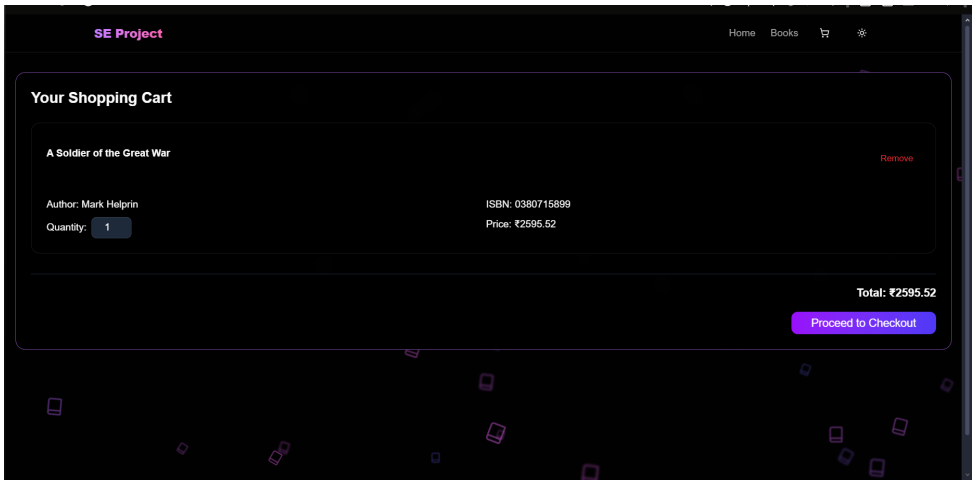

Figure 28: Shopping cart with quantity 1

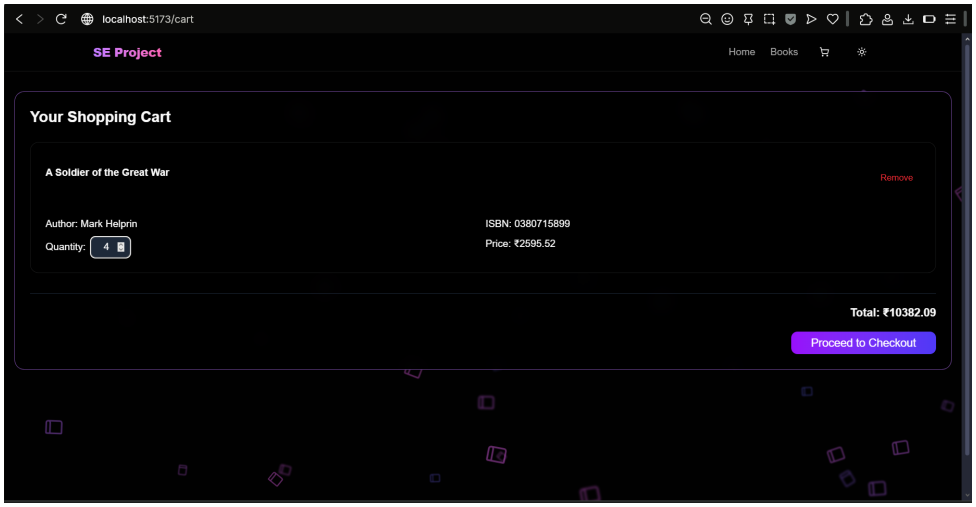

Figure 29: Shopping cart with quantity 4

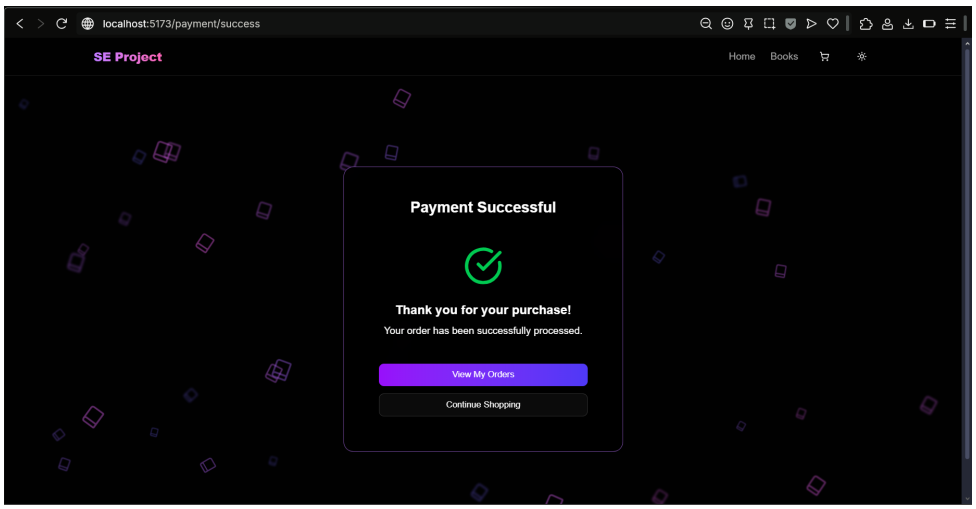

Figure 30: Payment success page

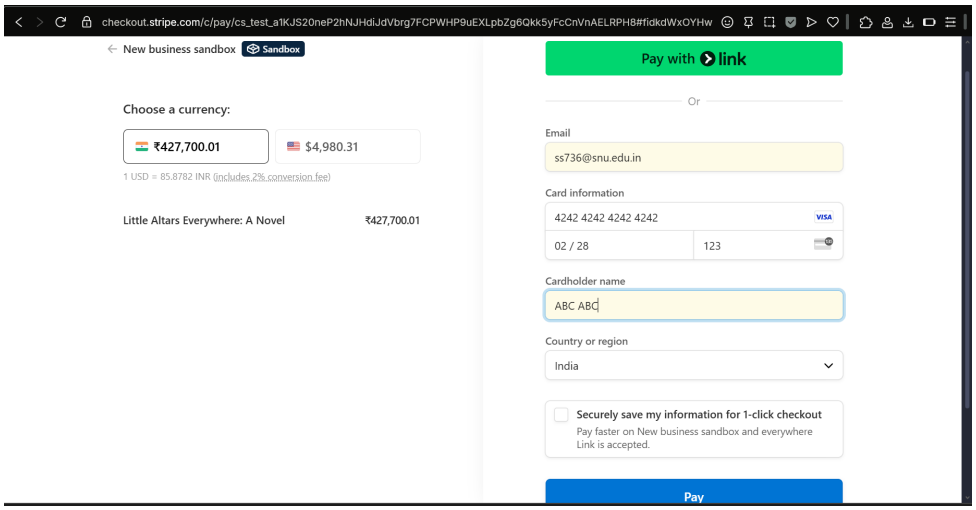

Figure 31: Stripe Payment redirect with fake card info.

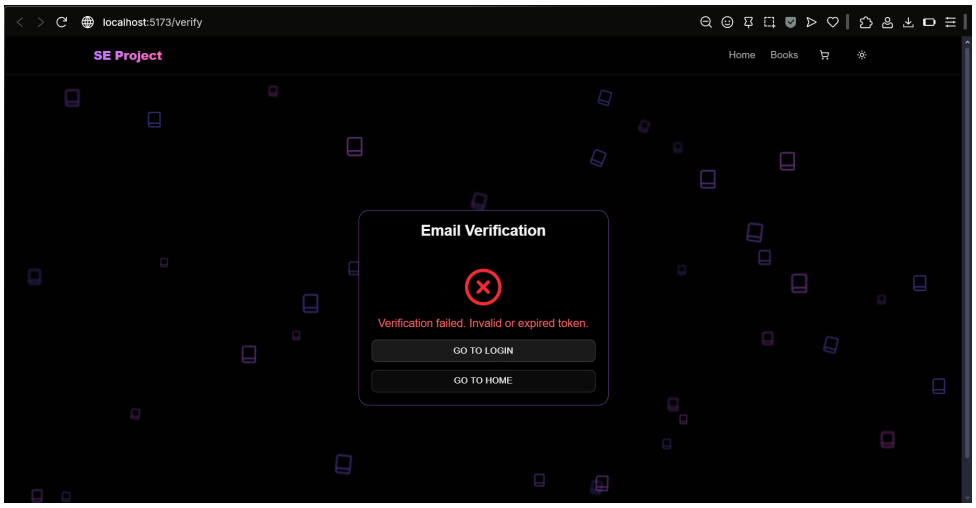

Figure 32: Email verification failure page

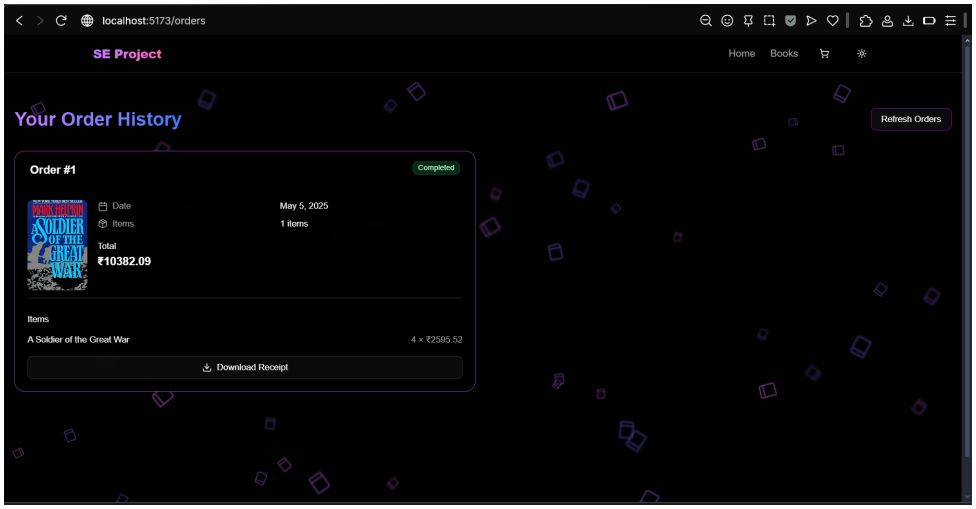

Figure 33: User order history page

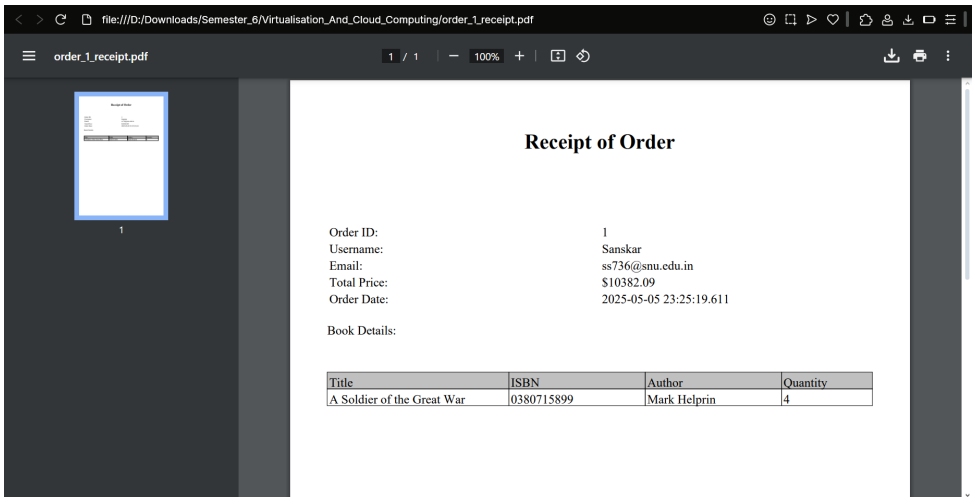

Figure 34: Receipt for order in PDF format

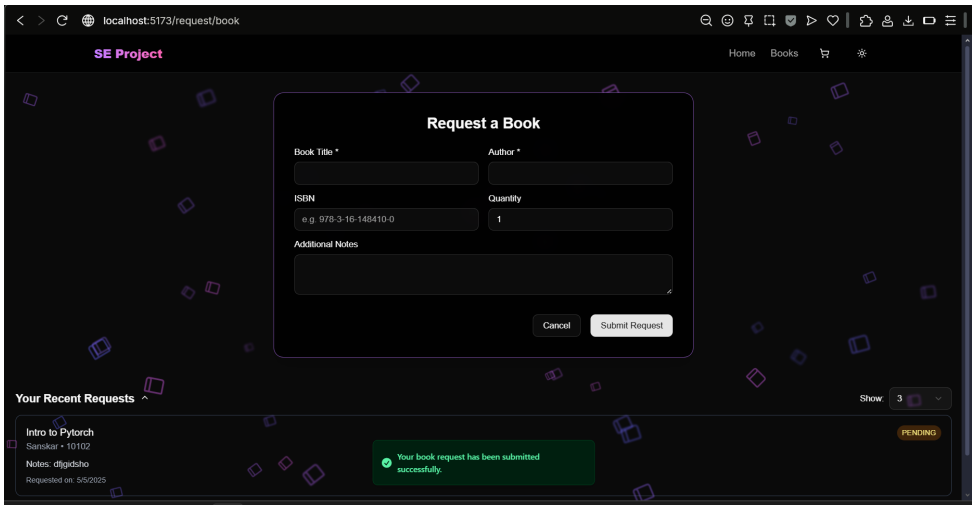

Figure 35: Request book page

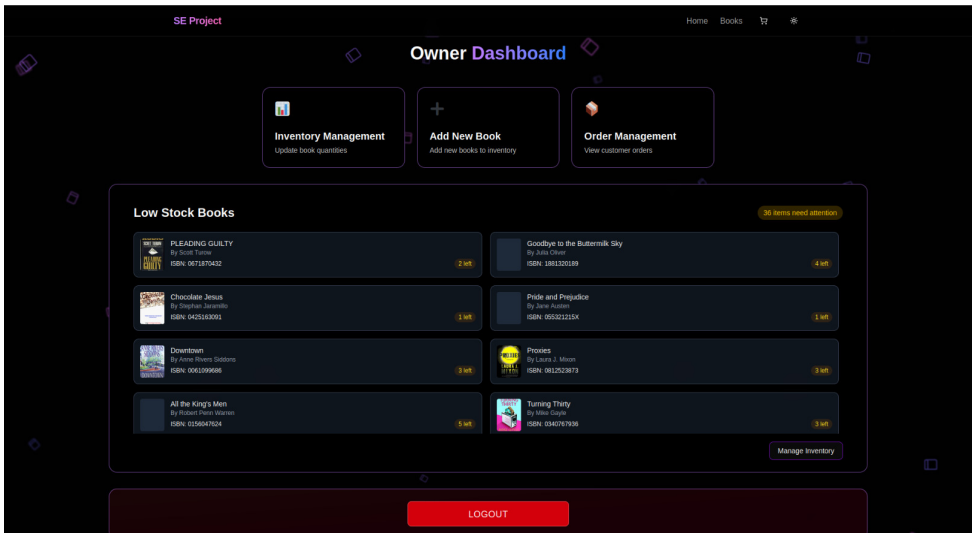

Figure 36: Low stock books and owner dashboard

A.2.3 Logging Screenshot after successful book search

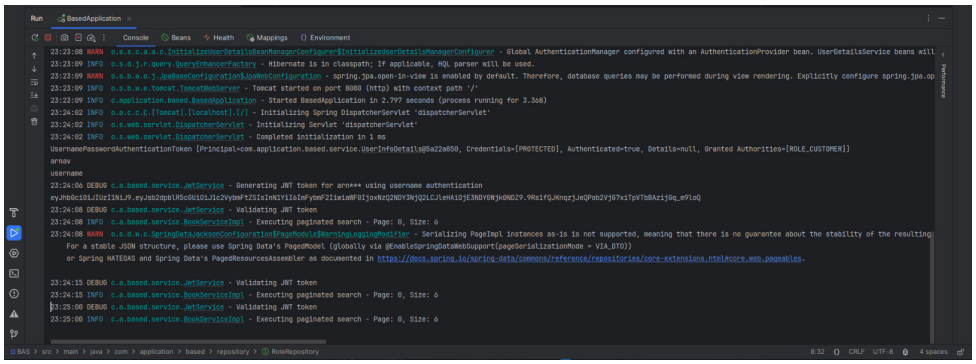

Figure 37: System logging configuration

A.2.4 Testing Screenshots

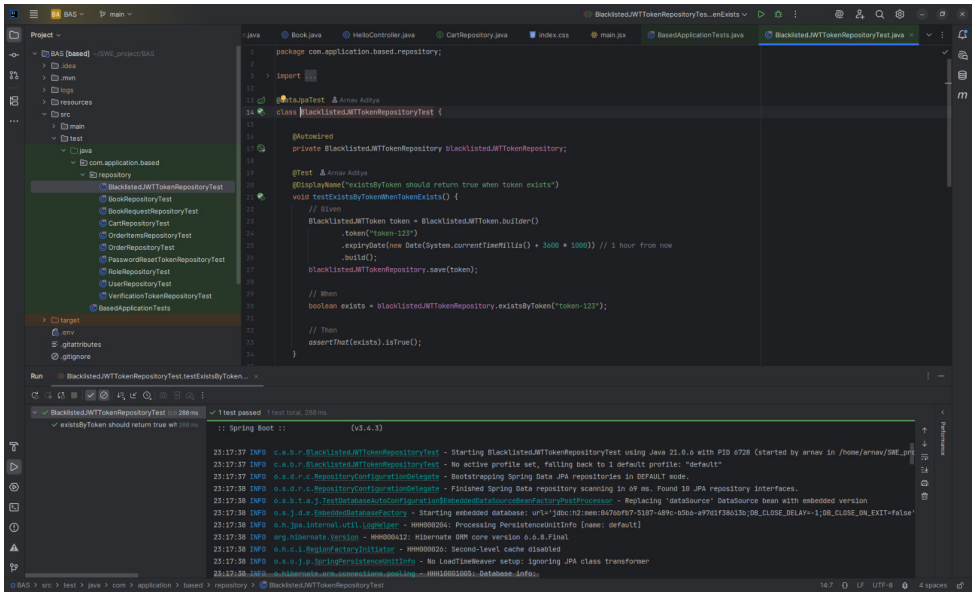

Figure 38: Unit test for repository layer (Blacklisted JWT Token repository success

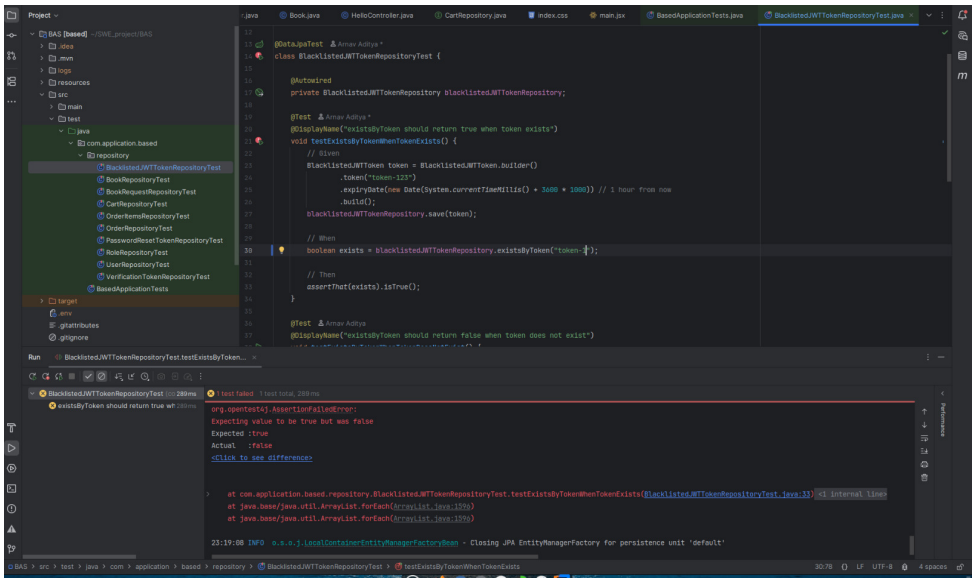

Figure 39: Unit test for repository layer (Blacklisted JWT Token repository fail

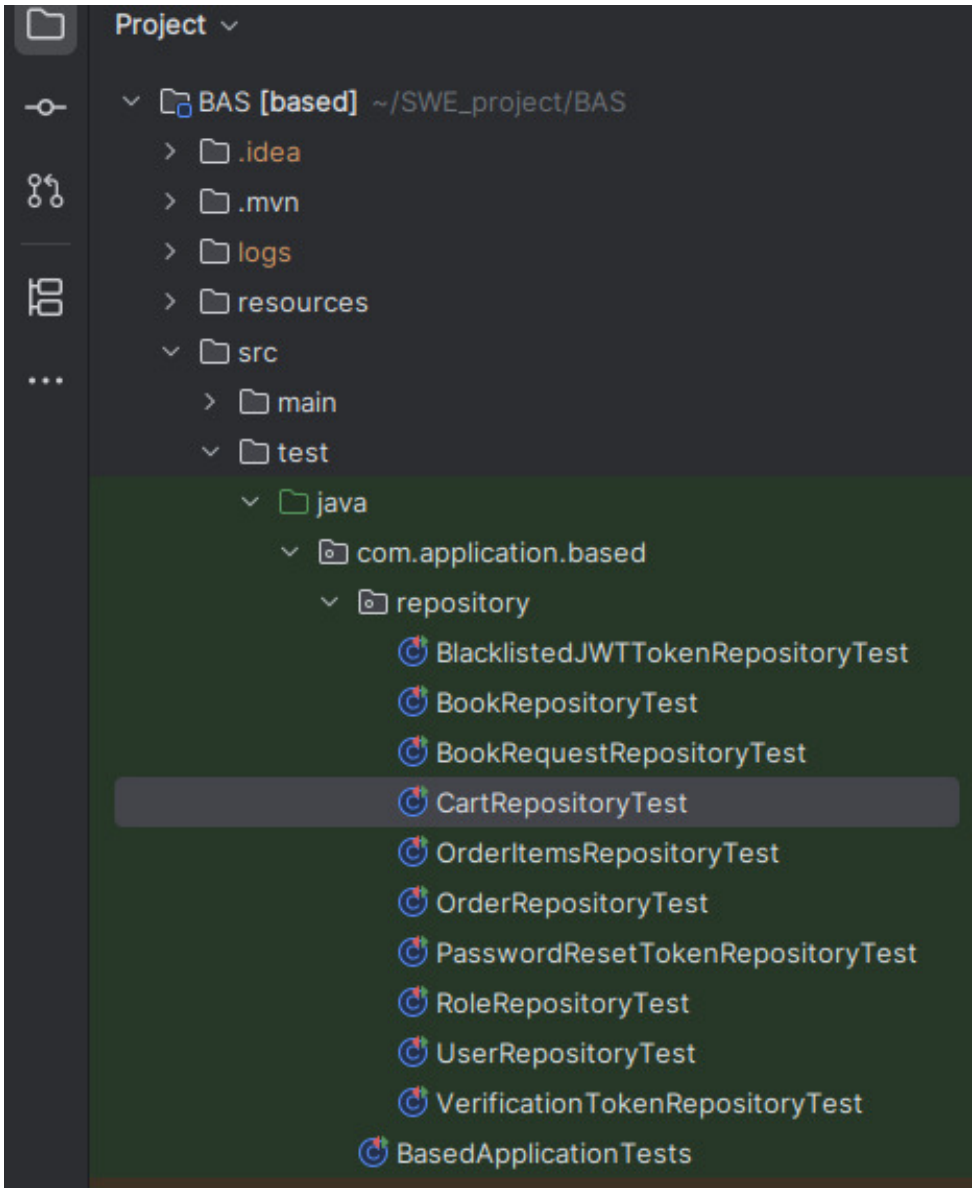

Figure 40: All repository layer tests