1. Technical Description of the Final Product

1.1 Architecture and System Design

Our team created a web-based eBook loaning platform that would allow users to loan eBooks from a centralised database. The backend was implemented in Java (JDK 17). This JDK was specifically used as it integrates well with the Spring Boot framework, ensuring that backend functionality was reliable and that robust APIs could be created. The frontend was implemented with HTML, CSS, and JavaScript to provide a clean, responsive interface that promoted a smooth navigation experience and easy interaction with the websites core features. The system follows a three-tier MVC (Model-View-Controller) architecture to ensure maintainability, scalability, and clear separation of concerns.

System Architecture Layers:

- Presentation Layer (Frontend): The system provides several user interfaces accessible through a web browser. These are comprised of key components such as the registration and login pages, home display, browsing of books categorised by genres, a user dashboard and an admin panel.
- Application Layer (Backend Business Logic): This system was built using Spring Boot REST APIs which manage core functionalities to handle user authentication, book borrowing functionality, email notifications and review commenting.
- Data Layer (Database): The system utilises a MySQL database to store user accounts, eBook metadata, loan history, and reviews. Database connections are managed via Java database connectivity.

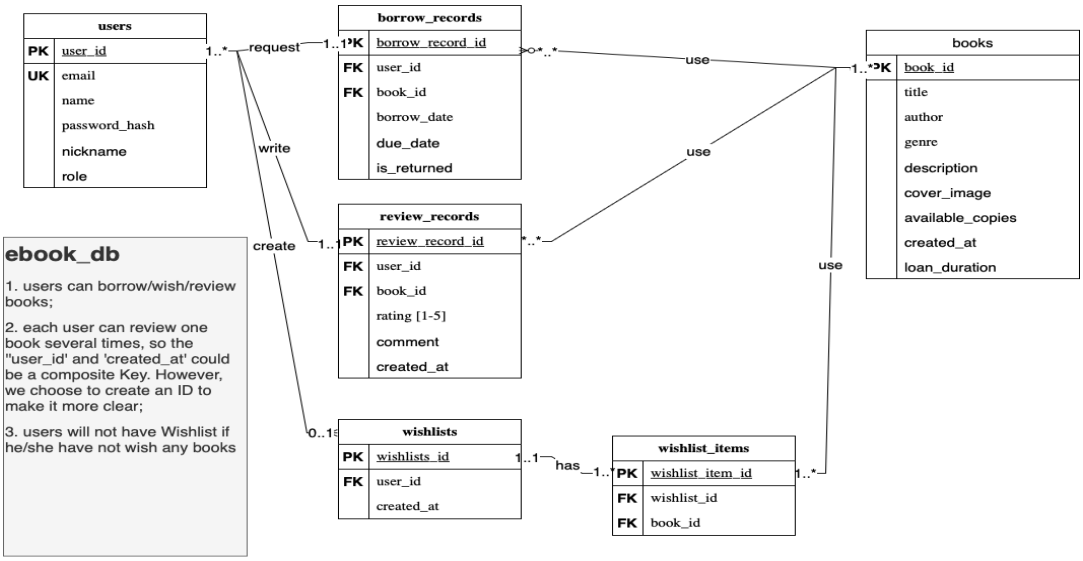

Figure - *Showing the UML entity relationship diagram of the database design*

1.2 Codebase and Complexity

The total number of lines of code is approximately 7,500, covering both the backend and frontend. It is a multi-layered system with various components, such as services, user interactions, data management and security mechanisms.

The service components have a modular architecture, each managing a different part of the systems functionality

Service Components:

- AuthService: Handles login, registration, password hashing, and reset
- BookService: Manages eBook data, search, and filtering
- BorrowService: Controls borrowing, return limits, and auto-return features
- ReviewService: Manages user reviews and ratings
- EmailService: Integrates Gmail SMTP to send system emails
- AdminService: Provides admin-specific functions with role-based access control

The system also includes role-based access control, input validation, and exception handling as part of its core security features.

1.3 Extent of functionality

Requirement	Status	Notes
User Registration/Login	✅ Completed	With hashed passwords and confirmation email
Book Browsing/Search	✅ Completed	Supports category filtering and keyword search
eBook Loan & Return (auto-expiry)	✅ Completed	10-book loan limit enforced
Wishlist Management	✅ Completed	Linked to the user dashboard
Review	✅ Completed	Users can review and rate books
Admin Book Management	✅ Completed	Admin-only, supports add/edit/delete
Email Reminders (Loan Expiry)	✅ Completed	Sent 3 days before loan expiration
Responsive Design	✅ Completed	Tested on both mobile and desktop browsers
OAuth/Payment/Recommendation	❌ Not Included	Clearly defined as out of scope
Online Reading (DRM)	❌ Not Included	Reading mechanism not implemented

1.4 Testing Strategy and Coverage

We adopted the following testing strategies to ensure the stability and functionality of the system:

- Unit Testing: Core business logic tested using JUnit.

- Integration Testing: REST API endpoints tested via Postman.
- Manual UI Testing: Covered all major UI functions across supported devices.
- Edge Case Testing: Scenarios include exceeding borrow limits, zero book availability, and duplicate registrations.
- End-to-end testing: Interactive functionality and layout behaviour of HTML, CSS, and Javascript tested using Cypress
- HTML/CSS validation: W3C Markup Validation Service and W3C CSS Validation Service was used to ensure best validity and standards compliance.

Code Coverage: Approximately 82%
Bug Fix Rate: Final version has fewer than 3% unresolved defects
Known Unresolved Issues: No critical missing functionality

Figure (left) - The back end tests some interfaces through postman.
Figure (right) - W3C CSS Validation Service depicting no errors within the HomePage.css for example.

2.

Student full name	Work contributed	Team participation	Professional Behaviour	Communication	**Final Peer Review**	
Avin Jayatilake	17	16.66666667	16.66666667	16.66666667	**16.75**	
Yunyi jiang	17	16.66666667	16.66666667	16.66666667	**16.75**	
Peilin li	17	16.66666667	16.66666667	16.66666667	**16.75**	
Grace Brown	17	16.66666667	16.66666667	16.66666667	**16.75**	
Guanyuan	17	16.66666667	16.66666667	16.66666667	**16.75**	
Nangialai Amin	15	16.66666667	16.66666667	16.66666667	**16.25**	Please leave at 0 if there are not 6 people in your team
Student 7	0	0	0	0	**0**	Please leave at 0 if there are not 7 people in your team
Student 8	0	0	0	0	**0**	Please leave at 0 if there are not 8 people in your team
Student 9	0	0	0	0	**0**	Please leave at 0 if there are not 9 people in your team
Student 10	0	0	0	0	**0**	Please leave at 0 if there are not 10 people in your team
Total	100	100	100	100	**100**	Boxes B10-F10 MUST = 100

If everyone has done an equal amount of work, your scores will be 12.5 (8 person team), 11.1 (9 person team) or 10 (10 person team) etc.

Any disputes over peer review scores MUST be resolved before submission. The submission of this document means that ALL team members have seen and agreed the peer review scores. The scores will not change once they have been submitted. Column L is your final score.

Criteria for the peer review

Work contributed - includes planning & organisation, documentation, design, programming and testing,

Team participation - suggesting ideas to the team, participating in team discussions, e.g. group online chats

Professional Behaviour - includes meeting deadlines (or requesting help if issues) for both hard and soft deadlines plus all material covered in the Professional Behaviour lecture,

Communication - includes reporting progress, communicating difficulties, responding to messages sent via the communication channel of your choice, etc.