Vietnamese-German University

Faculty of Engineering

Vietnamese-German University

Programming Exercise Project

BIBLIOTECH

Online Library Management Platform

Contributing Students	Student Numbers
Huynh Le An Phu	10421100
Do Minh Quang	10421051
Le Cong Nguyen	10421043
Duong Thien Huong	10421019
Phan Tam Nhu	10421122
Nguyen Minh Anh	10421068

Contents

Executive summary

This report details the development and implementation of the Bibliotech online library management platform. The primary objective was to create a user-friendly system that facilitates book borrowing, reviewing, and efficient library administration. By employing the Scrum Agile methodology, the project team was able to adapt to feedback and iteratively enhance the platform.

Key features of the platform include a secure user registration and authentication system, comprehensive book browsing and filtering options, and a robust moderator management process. The project also addressed challenges such as integration issues between the frontend and backend, employing effective communication and rigorous testing to resolve them.

The outcomes demonstrate significant improvements in user experience and administrative efficiency. Future enhancements, such as payment integration and multilingual support, are planned to further enhance the platform's capabilities and user engagement.

Overall, the Bibliotech platform represents a significant advancement in digital library management, setting the stage for continued growth and innovation.

1 Introduction

1.1 Context and Purpose

This report outlines the development of an online library app for users to borrow and review books, while also providing library administration tools for managing inventory, tracking borrowings, and user communication.

The main objectives of this report include:

1. Documenting Initial Planning, Goals, and Resource Allocation: This section outlines the initial phase of the project, where the scope was defined, objectives were set, and resources were allocated. It includes the identification of key stakeholders, setting of project timelines, and allocation of necessary resources such as personnel, tools, and technologies.

2. Explaining Design and Architectural Decisions: This part of the report delves into the design and architectural choices made during the project. It covers the rationale behind selecting specific technologies, frameworks, and design patterns. It also describes how the application's user interface (UI) and user experience (UX) were crafted to ensure ease of use and efficiency for both users and administrators.

3. Describing Implementation and Development Process: Here, the report provides a detailed description of the implementation phase. It includes task distribution among team members, the development methodologies employed (e.g., Agile, Scrum), and the tools and environments used for coding and testing. This section highlights how the project team collaborated to build the application and ensure its functionality and performance.

4. Highlighting Challenges and Solutions: This section discusses the various challenges encountered during the project's lifecycle. It covers technical, operational, and logistical issues and the strategies employed to overcome them. It provides insights into problem-solving approaches and the lessons learned from these experiences.

5. Detailing Deployment Strategy and Post-Deployment Activities: This part of the report outlines the deployment strategy adopted to launch the application. It covers pre-deployment testing, staging, and final

rollout. Additionally, it describes post-deployment activities such as monitoring, user feedback collection, and ongoing maintenance and support.

6. Evaluating Outcomes and Reflecting on Learning Experiences: The final section evaluates the overall outcomes of the project, assessing whether the initial goals and objectives were met. It reflects on the project's success, the effectiveness of the strategies used, and the team's performance. This section also identifies areas for improvement and provides recommendations for future projects based on the lessons learned.

By documenting each phase of the project in detail, this report aims to serve as a comprehensive guide and reference for understanding the development process of the online library management application. It is intended to inform stakeholders, provide transparency, and contribute to the knowledge base for future projects in similar domains.

1.2 Scope

This report is structured to comprehensively cover all aspects of the development of an online library management application, providing detailed insights from initial planning through to post-deployment activities. The scope of the report is defined as follows:

Within Scope

1. Project Planning and Goals
    - *Detailed Project Goals and Objectives*: Clear articulation of the project's purpose, including specific objectives the application aims to achieve.
    - *Initial Planning and Requirement Gathering*: Documentation of the early stages of the project, focusing on identifying user needs and defining system requirements.
    - *Timeline with Milestones and Deadlines*: Presentation of the project timeline, highlighting key milestones, deadlines, and deliverables.

- *Resource Allocation*: Allocation of resources, including human resources (team roles and responsibilities), technical resources (tools and technologies), and financial resources (budget and expenditures).

2. Design and Architecture
    - *System Architecture Overview with Diagrams*: Visual representation of the system architecture, illustrating the structure and components of the application.
    - *Choice of Technologies and Tools with Justification*: Explanation of the technologies, frameworks, and tools selected for the project, along with reasons for their selection.
    - *Functional and Non-Functional Requirements*: Detailed description of both functional requirements (features and functionalities) and non-functional requirements (performance, security, usability).
3. Implementation Process
    - *Development Methodology*: Description of the development methodology (e.g., Agile, Waterfall) used, along with the rationale for its choice.
    - *Task Distribution Among Team Members*: Outline of how tasks were distributed among team members to ensure efficient workflow and collaboration.
    - *Code Management Practices and Tools*: Overview of code management practices, including version control systems and collaboration tools.
    - *Testing Strategies*: Description of testing strategies employed, including unit tests, integration tests, and user acceptance tests to ensure quality and functionality.
4. Deployment Strategy
    - *Step-by-Step Deployment Plan*: Detailed plan of the steps taken to deploy the application.

- *Description of the Deployment Environment*: Overview of the deployment environment, including any cloud platforms or servers used.
- *Rollout Strategy*: Explanation of the rollout strategy, whether it was a phased rollout or a big bang deployment.
- *Post-Deployment Monitoring and Maintenance*: Description of the monitoring and maintenance activities carried out after deployment to ensure the application's smooth operation.

5. Challenges and Solutions
    - *Challenges Faced During the Project*: Identification of the main challenges encountered during the project's lifecycle.
    - *Solutions Implemented to Address These Challenges*: Explanation of the solutions and strategies implemented to overcome the challenges.
6. Results and Evaluation
    - *Outcomes of the Project*: Assessment of the project's outcomes against the initial goals and objectives.
    - *Evaluation of Whether the Project Goals Were Met*: Critical evaluation of the extent to which the project goals were achieved.
    - *Feedback Received and How It Was Incorporated*: Summary of user and stakeholder feedback and how it was used to improve the application.
7. Learning Outcomes
    - *Technical Skills Acquired*: Documentation of the technical skills and knowledge gained by the team during the project.
    - *Insights on Teamwork and Collaboration*: Reflection on the teamwork and collaboration experiences throughout the project.
    - *Project Management Lessons Learned*: Identification of key project management lessons and best practices learned.
8. Future Enhancements

- *Planned Future Features and Improvements*: Description of future features and improvements planned for the application to enhance its functionality and user experience.

9. Appendices
    - *Supporting Documents*: Inclusion of relevant supporting documents such as code snippets, detailed diagrams, and raw data.

Outside Scope

1. Unrelated Technical Details: Exclusion of technical details or methodologies not directly relevant to the project.
2. Extensive Literature Review: Avoidance of a comprehensive literature review that does not directly relate to the project objectives or outcomes.
3. Unrelated Theoretical Background: Omission of broad theoretical discussions that do not pertain to the practical implementation and outcomes of the project.
4. Detailed Financial Analysis: Exclusion of an in-depth financial breakdown and cost analysis beyond the basic resource allocation mentioned in the planning section.

2 Planning Phase

2.1 Project Goals

The primary objective of this project is to develop a robust digital platform that facilitates the process of borrowing, reviewing, and managing books for users, moderators, and admins. This platform aims to deliver an exceptional user experience, improve book accessibility, and enhance administrative efficiency. The specific, measurable goals of the project include:

1. **User Registration and Authentication**: Implement a secure user registration and authentication system.
2. **Book Browsing and Filtering**: Enable users to browse and filter books by category, author, and publication.
3. **Book Borrow System**: Develop a system for users to borrow books for a specific duration.
4. **Review and Rating System**: Allow users to review and rate books.
5. **Moderator Management**: Provide functionality for moderators to list, manage, and track books.
6. **Admin Approval Process**: Create a system for admins to approve books submitted by moderators and users' moderator applications.
7. **User Application to Moderator**: Allow users to apply to become moderators.
8. **Feedback and Improvement**: Collect user feedback and iteratively improve the system.

2.2 Timeline

Week 1: Receive the topic, and discuss tools, architecture, and research.

- Front End: No specific tasks.
- Backend: No specific tasks.

Week 2: Choose technology, analyze requirements, create diagrams.

- Front End: Internet-based research, learn web UI/UX, make prototypes.
- Backend: Project initialization, setup files, database schema, basic pages, and functions.

Week 3: Draw sequence diagrams, update basic functionalities.

- Front End: Continue learning web design, create UI prototypes.
- Backend: Add and update functions, create "Book detail" page, update book object.

Week 4: Draw ER diagrams, and update functionalities.

- Front End: Continue learning UI design, update prototype, and create a plan.
- Backend: Change directory structure, add new functions, update existing ones.

Week 5: Design component and deployment diagrams, continue development.

- Front End: Learn advanced HTML, CSS, JS, and create layouts.
- Backend: Create new views, implement new functions, Docker containerization, and fix bugs.

Week 6: Understand DevOps, use Docker.

- Front End: Adjust pages, create pages for Moderator role.
- Backend: Create new profile page, implement functions, maintenance, and bug fixing.

Week 7: Finalize frontend designs, and maintain backend functions.

- Front End: Finalize UI pages for different roles.
- Backend: Implement new functions, and update existing ones.

Week 8: Implement and maintain backend functions, link frontend and backend.

- Front End: Update notifications pages, create rating and review elements.

- Backend: Update user profiles.

Week 9-12: Modify existing web pages, link frontend and backend.

- Front End: Various updates to profiles, elements, and UI consistency.
- Backend: Standby for finalization from frontend, update and fix bugs, maintain functions.

Week 13: Modify web pages, link frontend and backend, maintenance.

- Front End: Create new UI forms, update pagination.
- Backend: Update database, resolve conflicts, fix bugs.

Week 14: Modify web pages, link frontend and backend, maintenance, deployment.

- Front End: Create additional pages, design logo, fix minor errors, review pages.
- Backend: Linking and finalizing integration, update constraints, deployment with Railway.

Week 15: Finalize project, bug fixing, minor updates, make website responsive.

- Front End: Commit UI template, write report, prepare presentation.
- Backend: Finalize linking, update functions, make webpages responsive.

2.3 Resources

This section outlines the human and technical resources required for the project and their allocation.

2.3.1 Human Resources

Project Manager (1 person):

- Phu: Responsible for overall project coordination, timeline management, and stakeholder communication.

Backend Developers (3 people):

- Nhu, Quang, Phu: Tasked with developing the server-side logic, integrating with the database, and ensuring the application's performance and security.

Front-end Developer (4 people):

- Quang, Nguyen, Huong, Minh Anh: Responsible for implementing the user interface and ensuring a responsive and engaging user experience.

Database Administrators (3 people):

- Quang, Phu: Manage database design, optimization, and maintenance to ensure data integrity and availability.

UI/UX Designers (3 people):

- Nguyen, Minh Anh, Huong: Design intuitive and user-friendly interfaces, focusing on user experience and accessibility.

DevOps Engineers (2 people):

- Phu, Quang: Manage the development, testing, and production environments, ensuring continuous integration and delivery.

2.3.2 Technical Resources

Development Tools:

- Backend:
  - Django: A high-level Python web framework for rapid development and clean, pragmatic design.
  - Python: The programming language used for backend development.
- Frontend:
  - Webflow: A web design tool, mainly used for UX/UI designs.
  - HTML, CSS, JavaScript: Standard technologies for creating the user interface.
- Version Control:
  - Git: A distributed version control system.

  - GitHub: A platform for version control and collaboration.

Database:

- Primary:
  - SQLite: Used during development for simplicity and ease of setup.

Infrastructure:

- Servers, Storage, Networking:
  - Railway Production: The production environment for hosting the application, providing necessary server, storage, and networking resources.

2.4 Risk Management

Risk management is a crucial aspect of the project planning phase. This section identifies potential risks and outlines strategies to mitigate them.

2.4.1 Potential Risks

Integration Issues Between Backend and Frontend
Problems may arise when integrating the backend services with the frontend interface, potentially leading to functionality breakdowns or performance issues.

Delays in Deployment Due to Unforeseen Technical Challenges
Unexpected technical challenges, such as compatibility issues or infrastructure failures, could delay the deployment schedule.

2.4.2 Mitigation Strategies

Regular Meetings and Updates to Track Progress
Schedule frequent meetings to discuss progress, identify any emerging issues, and ensure alignment among team members. Regular updates help in early detection and resolution of potential problems.

Thorough Testing at Each Stage of Development
Implement a rigorous testing regimen that includes unit testing, integration testing, and user acceptance testing. Thorough testing at each development

stage ensures that components work correctly both independently and when integrated.

Backup Plans for Critical Components
Develop backup plans for essential parts of the project. This includes having alternative solutions for key technologies and creating contingency plans for critical infrastructure. Having backups in place can minimize downtime and ensure continuity in case of technical failures.

3 Design Phase

3.1 System Architecture

This section provides a comprehensive overview of the system architecture, detailing its primary components, data flow, and considerations for security and scalability.

3.1.1 Overview

The system is designed as a robust digital platform for borrowing, reviewing, and managing books. It consists of three main components: the client-side (frontend), the server-side (backend), and the database. In addition, we also use Google OAuth service for authentication and use GitHub version control for continuous integration and continuous deployment (CI/CD).

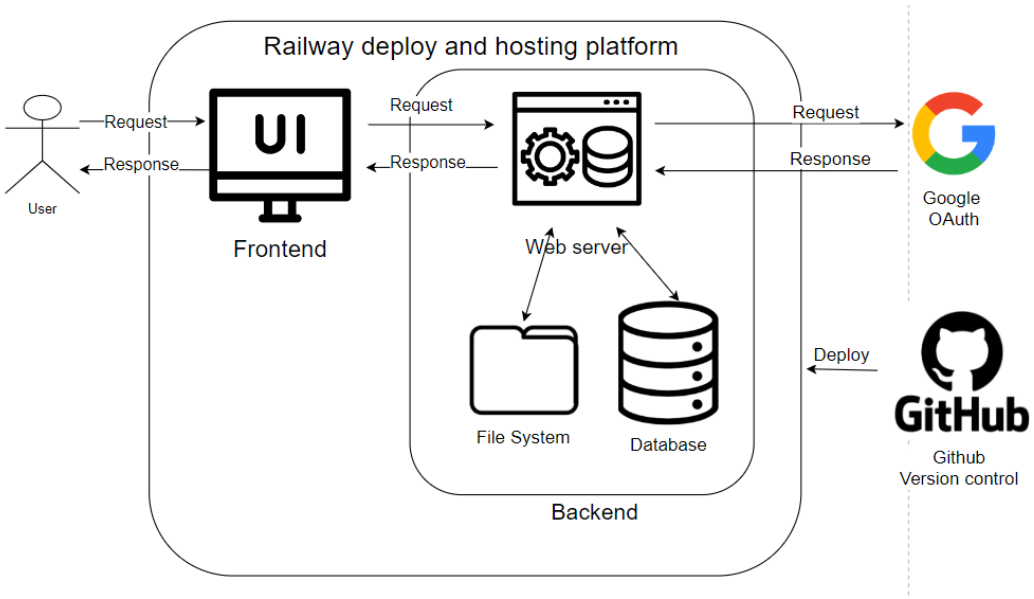

Figure 1: System Architecture

3.1.2 Component Breakdown

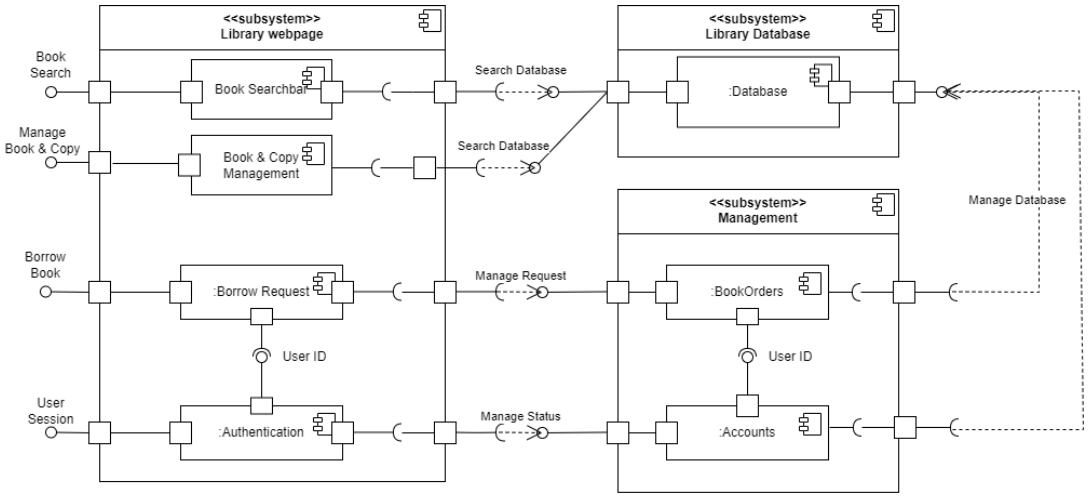

Figure 2: Component Diagram

Client-side (Frontend):

- Technologies: Webflow, HTML, CSS, JavaScript
- Role**:** Handles user interactions and presents data to the user through a responsive and engaging user interface.

Web Server (Backend):

- Technologies: Django, Python
- Role: Manages application logic, user authentication, and database interactions. The backend processes requests from the frontend, applies business logic, and communicates with the database to retrieve or store data.

Database (Backend):

- Technology**:** SQLite for development
- Role: Stores all persistent data, including user information, book details, and borrowal records.

File System (Backend):

- Technology: OS File System
- Role: Stores static files including html, css, js and images files.

Google OAuth:

- Role: Use authenticaion with google mail account.

Github:

- Role: Manage project version and deploy with incoming commits.

3.1.3 Data Flow

Data flows through the system in the following manner:

- User Interaction: Users interact with the application through the frontend, performing actions such as browsing books, registering, logging in, borrowing books, and leaving reviews.
- HTTP Requests: The frontend sends HTTP requests to the backend server, containing user actions and data inputs.
- Backend Processing: The backend server processes these requests, applying the necessary business logic. For example, when a user borrows a book, the backend verifies the user's session, checks the book's availability, and records the borrowal transaction in the database.
- Database Interaction: The backend communicates with the SQLite database to store or retrieve data as required by the application logic.
- Response Generation: The backend generates a response based on the processed data and sends it back to the frontend.
- Data Presentation: The frontend receives the response and updates the user interface accordingly, presenting the data to the user.

3.1.4 Functionalities

This sections describes the overall workflow or activity in each of the functionalities of the project.

- User Registration

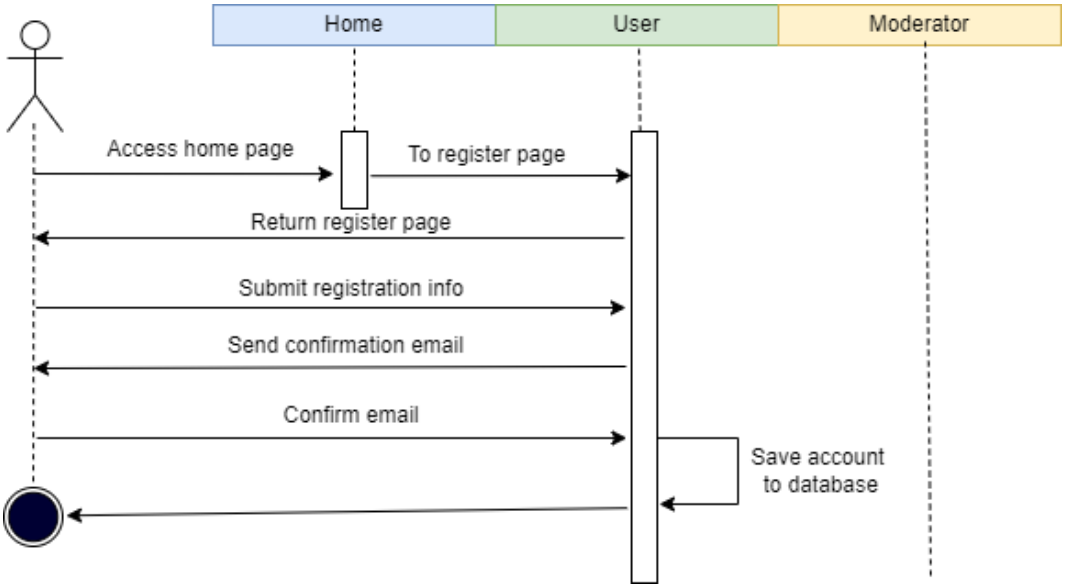

Figure 3: User Registration Workflow

- User Authentication (login)

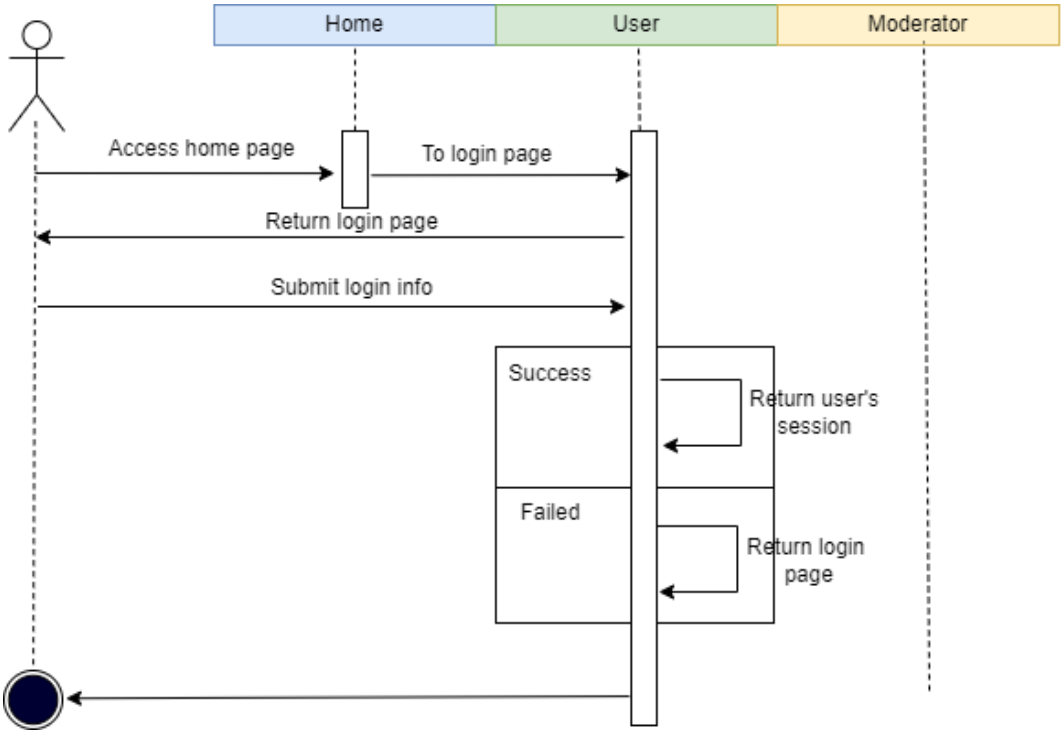

Figure 4: User Login Workflow

Apart from web login, our website also supports Google OAuthentication.

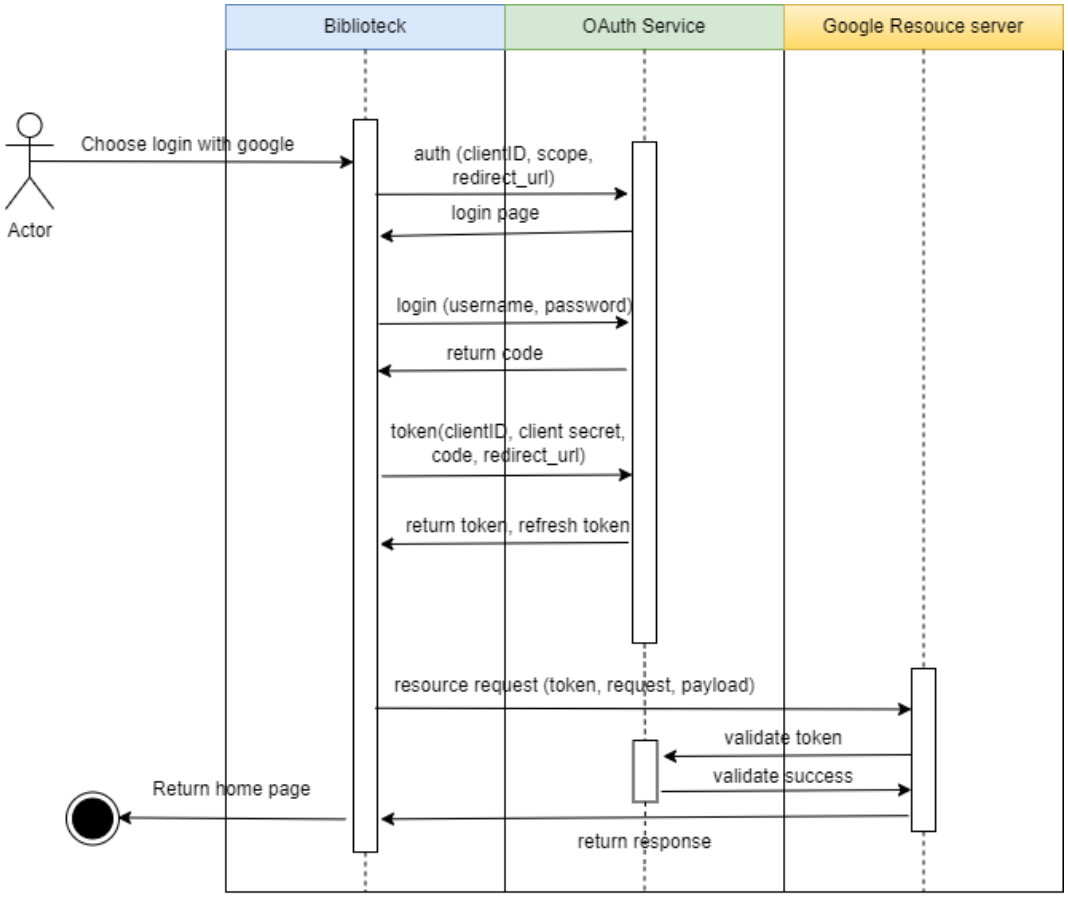

Figure 5: Google OAuth Workflow

Once users are logged in successfully (either by web login or google Oauth), on the next step, the role of the user must be checked for access control.

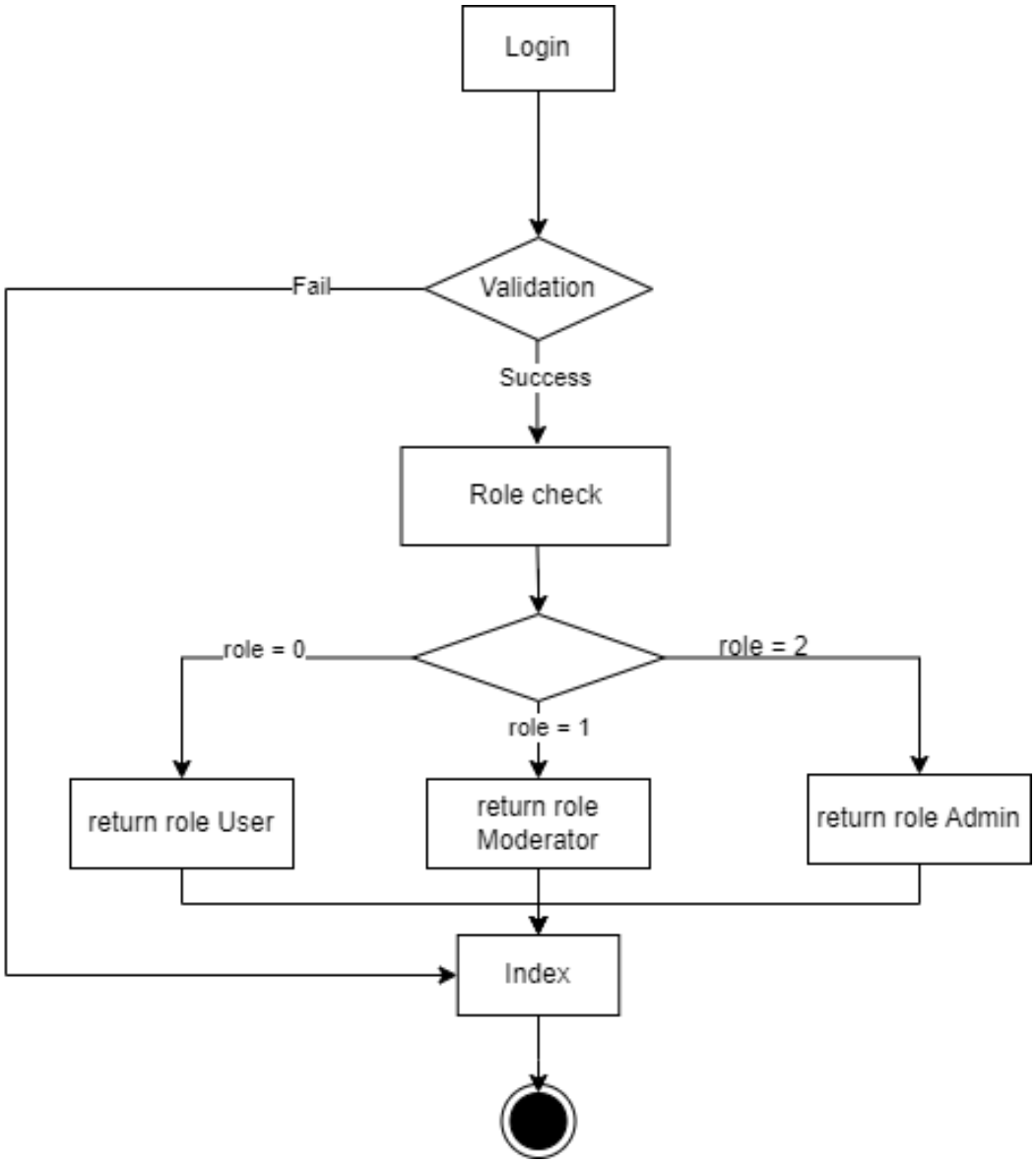

Figure 6: Role-based access control

- Book Browsing and Filtering

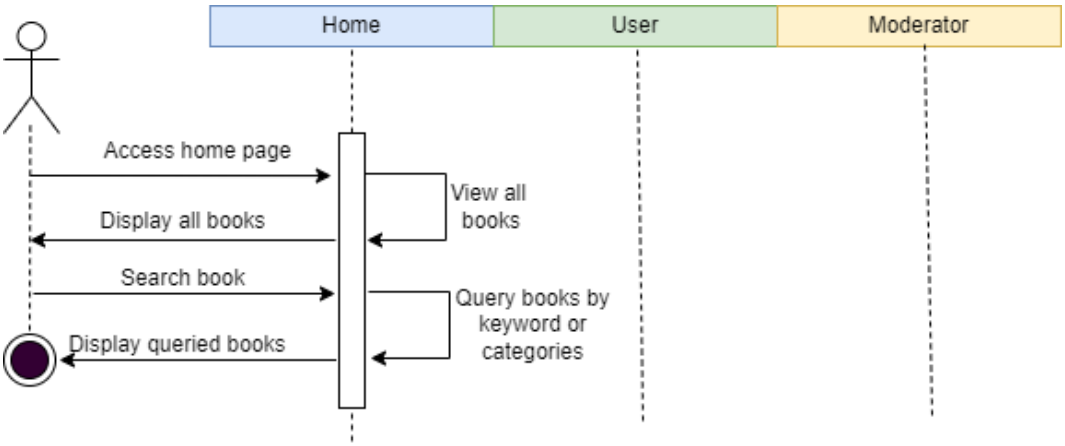

Figure 7: Book Browsing and Filtering workflow

- Book Borrow System

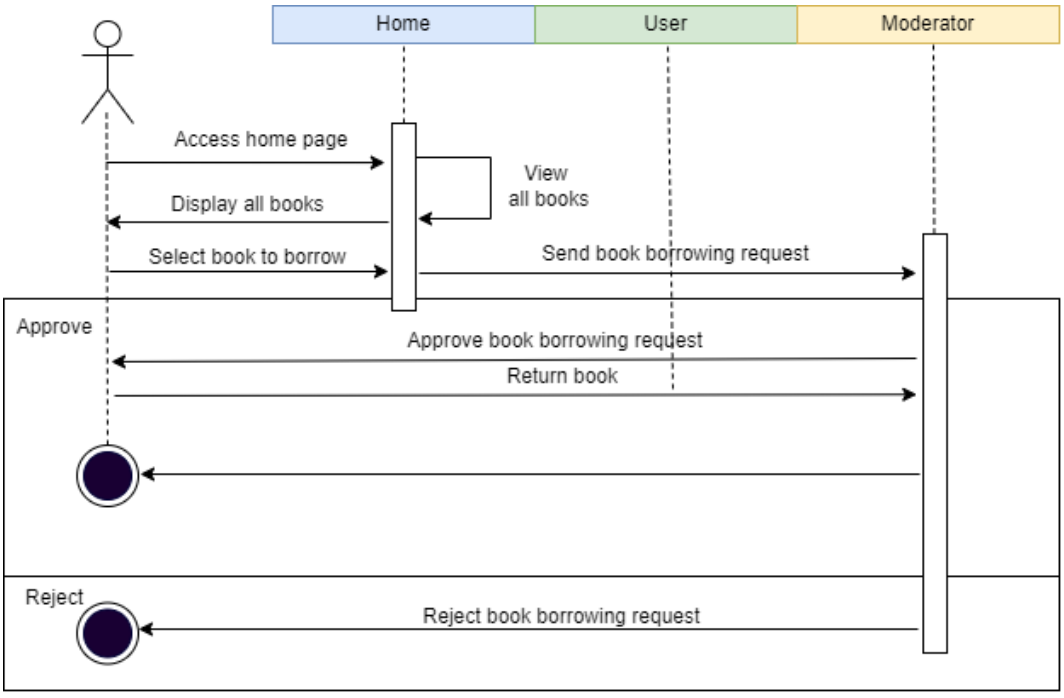

Figure 8: Book Borrowing System

- Review and Rating System

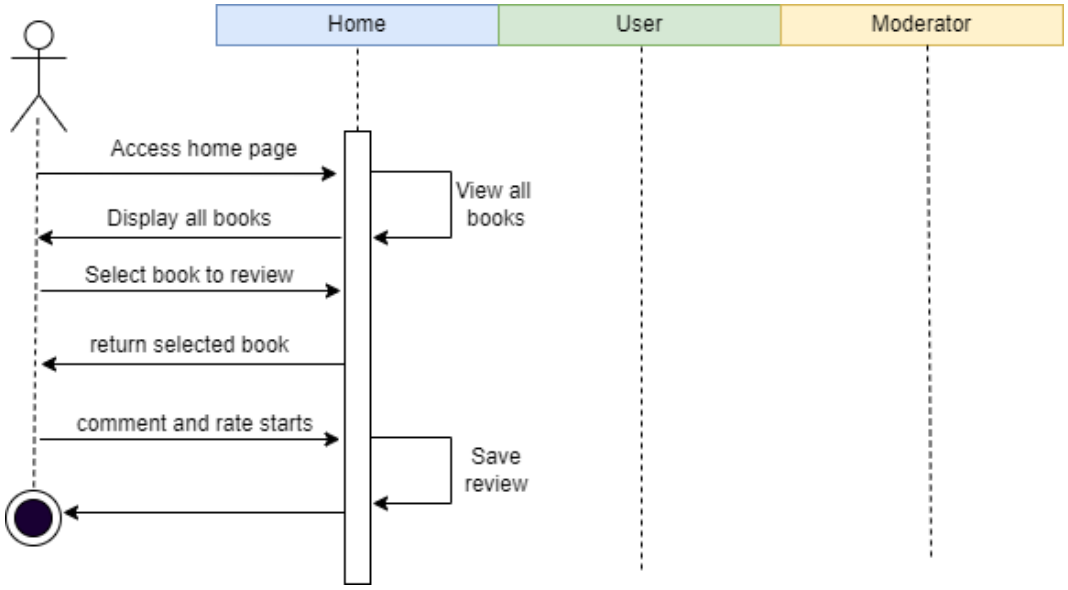

Figure 9: Review and Rating Workflow

- Moderator Management

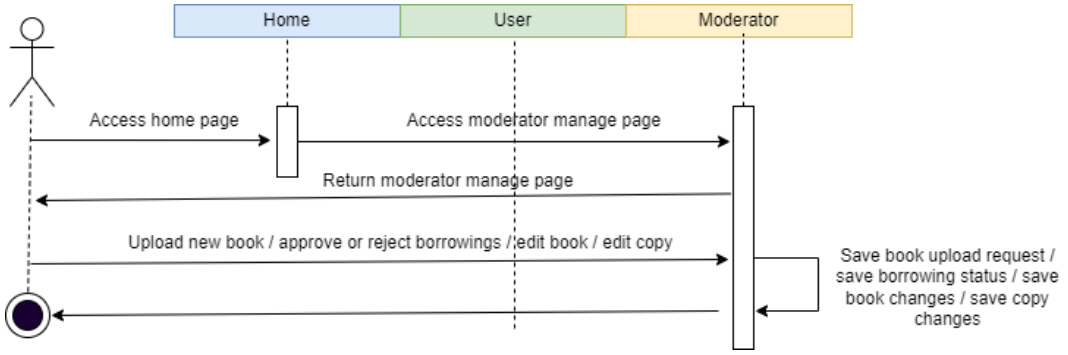

Figure 10: Moderator Management Workflow

- Admin Approval Process

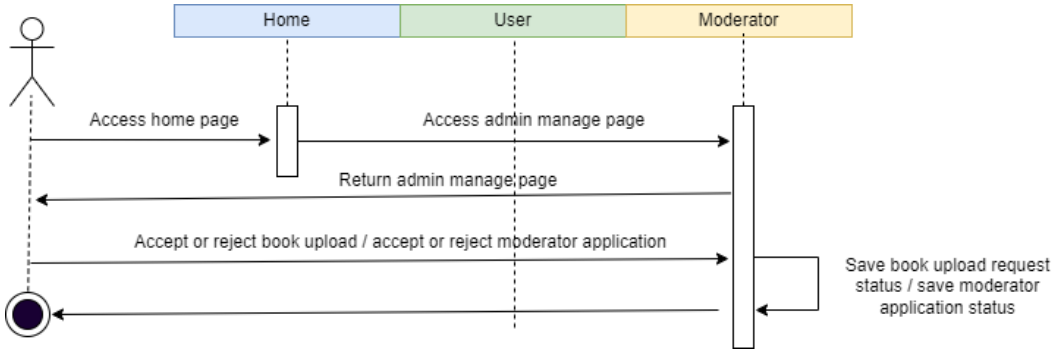

Figure 11: Admin Approval Process Workflow

- User Application to Moderator

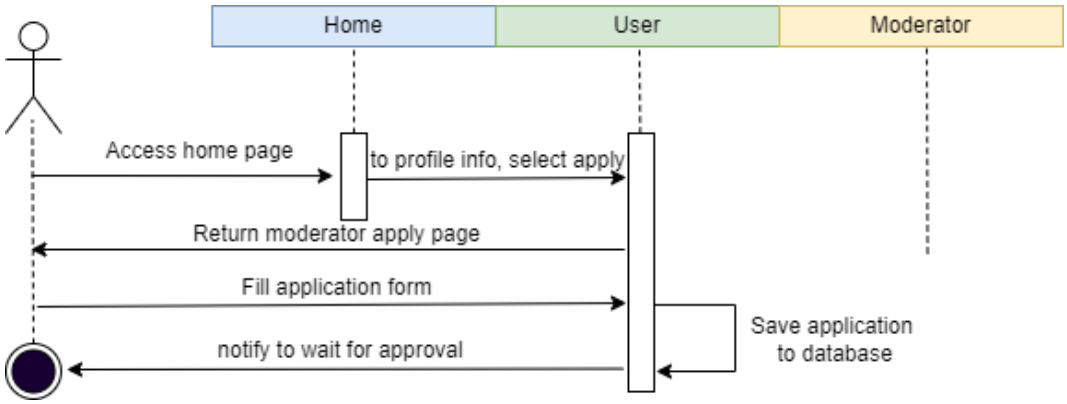

Figure 12: User Application to Moderator Workflow

3.1.5 Database design

Database schemas:

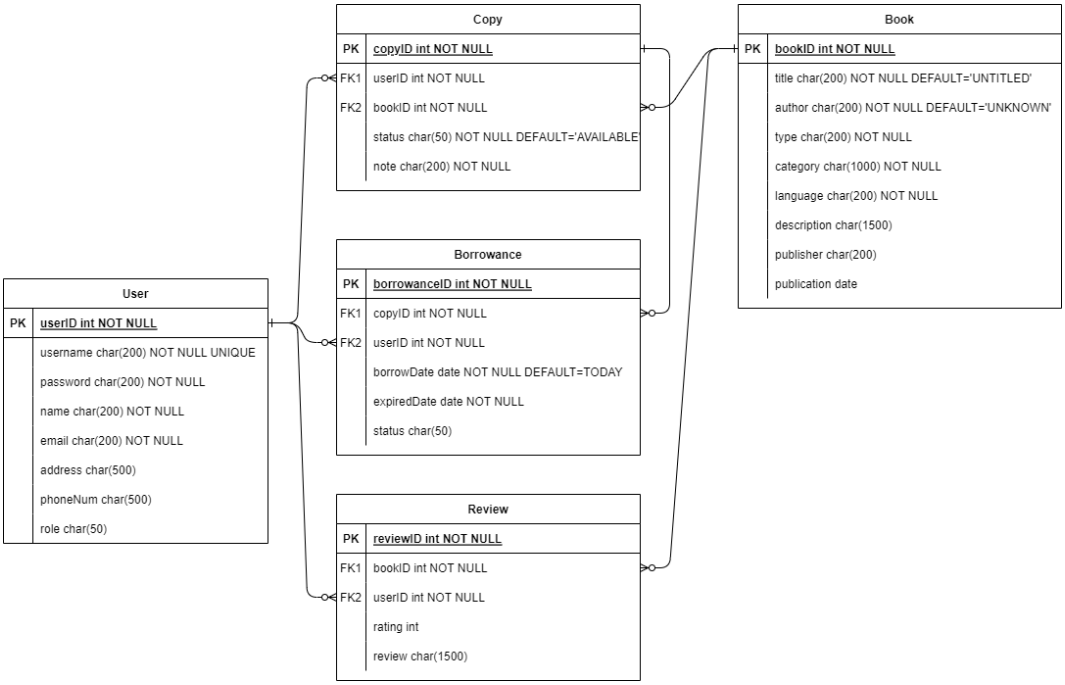

Figure 13: User Application to Moderator Workflow

- Book(**bookID**,title, author, type, liteCate, socioCate, naturCate, techCate, poliCate, romanCate, enterCate, otherCate, language, description, coverImage, publisher, publication, codeISBN, status)
- User(**userID**,avatar,username,password, first_name, last_name, birthdate, gender, email, address, phoneNum, role, description, availableWorkingHour)
- Copy(**copyID**,bookID, userID, status, note, regDate)

  FOREIGN KEY (bookID) REFERENCES Book(bookID)

  FOREIGN KEY (userID) REFERENCES User(userID)
- Borrowance(**borrowanceID**, copyID, userID, borrowDate, expiredDate, status)

FOREIGN KEY (copyID) REFERENCES Copy(copyID)

FOREIGN KEY (userID) REFERENCES User(userID)

- Review(**reviewID**,bookID, userID, rating, review)

  FOREIGN KEY (bookID) REFERENCES Book(bookID)

  FOREIGN KEY (userID) REFERENCES User(userID)

- ModApplication(**modApplicationID**, applicant, applicantDocument, applicantText, adminComment, status, created_at)

  FOREIGN KEY (applicant) REFERENCES User(userID)

- BookApplication(**bookApplicationID**,bookID, uploader, status, created_at)

  FOREIGN KEY (bookID) REFERENCES Book(bookID)

  FOREIGN KEY (uploader) REFERENCES User(userID)

3.1.6 Security Measures

Authentication:

- Tokens: Use tokens to ensure secure user sessions. Tokens authenticate users and maintain secure communication between the client and server.

Data Protection:

- Implement necessary data protection measures to ensure the integrity and confidentiality of user data.

Role-based Access Control: In this library management system, access control is implemented to ensure that users have appropriate permissions based on their roles. The system includes four distinct roles: Guest, Registered User, Moderator, and Admin. Each role has specific access levels and capabilities.

1. *Guest Role:* Guests are unregistered users who have the lowest access level in the system. *Permissions:*

   - *View Books:* Guests can browse the library's catalog and view book details.

- *Search Books:* Guests can use the search functionality to find books in the catalog.
- *Access Public Information:* Guests can access public information such as library hours, events, and general announcements.

Restrictions:

- *No Borrowing:* Guests cannot borrow books.
- *No Account Management:* Guests do not have access to account management features.
- *No Moderation or Administration:* Guests cannot perform any moderation or administrative tasks.

2. *Registered User Role:* Registered users have logged in to the system and have access to book borrowing services. *Permissions:*
   - *All Guest Permissions:* Registered users inherit all permissions available to Guests.
   - *Borrow Books:* Registered users can borrow books from the library, subject to availability and borrowing limits.
   - *View Borrowing History:* Registered users can view their borrowing history and due dates.
   - *Renew Books:* Registered users can renew borrowed books if they are not reserved by others.

   Restrictions:

   - *No Moderation:* Registered users cannot moderate or manage books.
   - *No Administration:* Registered users cannot perform administrative tasks or access the system database.
3. *Moderator Role:* Moderators are book lenders who own book collections and have control over whether a certain user can borrow their books. *Permissions:*
   - *All Registered User Permissions:* Moderators inherit all permissions available to Registered Users.

- *Manage Own Collection:* Moderators can add, edit, and delete books in their personal collection.
- *Approve/Reject Borrow Requests:* Moderators can approve or reject borrow requests for the books they own.
- *View Borrowers:* Moderators can view the list of users who have borrowed their books.

Restrictions:

- *No System-Wide Administration:* Moderators cannot manage the overall system or access the global database.
- *Limited to Own Collection:* Moderators can only moderate and manage books they own, not those owned by other moderators or the library.

4. *Admin Role:* Admins have the highest level of access in the system and can manage users, moderators, and the overall library database. *Permissions:*
   - *All Moderator Permissions:* Admins inherit all permissions available to Moderators.
   - *Manage Users:* Admins can add, edit, and delete user accounts, including promoting or demoting users to/from Moderator status.
   - *Approve/Reject Moderator Applications:* Admins can approve or reject applications from users to become Moderators.
   - *Manage Books:* Admins can approve or reject requests from Moderators to add books to the library's collection.
   - *System Database Access:* Admins have access to the library's database for maintenance and management purposes.
   - *Global Administration:* Admins can perform system-wide administrative tasks, including setting policies and configuring system settings.

   Restrictions:

   - *None:* Admins have full access and no specific restrictions within the scope of the system.

3.1.7 Scalability and Considerations

Scaling Plan: The scaling plan for the library management system leverages the built-in features of the Railway production environment, which facilitate scalability to handle increased user load and data volume as the application grows. The plan primarily focuses on vertical autoscaling, with considerations for the project's current scale and nature.

1. **Vertical Autoscaling:**
    - *Automatic Resource Allocation:* Railway automatically scales the service up to the specified vCPU and memory limits of the chosen plan. This ensures that as the system experiences higher user loads or processes more data, additional computational resources are allocated to maintain performance and stability.
    - *Resource Limits:* Each plan on Railway comes with predefined limits for vCPU and memory. The vertical autoscaling mechanism ensures that the service scales within these limits, providing a balance between performance and cost-efficiency.
    - *Performance Optimization:* By scaling vertically, the system can handle more simultaneous users and larger datasets without degrading performance. This is particularly important for database operations, search functionalities, and real-time user interactions.
2. **Built-in Scalability Features:**
    - *Load Balancing:* Railway's production environment includes built-in load balancing features that distribute incoming traffic across available resources, preventing any single instance from becoming a bottleneck.
    - *Monitoring and Alerts:* The environment offers monitoring tools and alert systems to track performance metrics and resource usage. This allows administrators to proactively manage the system and make informed decisions about scaling.
    - *Seamless Scale-Up:* As the application grows, Railway's infrastructure supports seamless scaling. This means the system can scale up without requiring significant downtime or manual intervention, ensuring continuous availability and reliability.

3. **Horizontal Scaling:**

   - *Current Implementation:* Due to the small scale of testing and the developmental nature of this project, horizontal scaling is not currently implemented.
   - *Future Considerations:* Should the number of users accessing the system increase suddenly, horizontal scaling will be considered and implemented. This will involve adding more instances of the application to handle the increased load, ensuring that the system remains responsive and reliable.

3.2 Technologies and Tools

In this section, we specify the technologies, frameworks, and tools chosen for the project and provide justifications for their selection.

3.2.1 Backend Technologies

Django

- *High-Level Framework:* Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It helps developers write less code while achieving more functionality.
- *Security:* Django comes with built-in security features to protect against common web vulnerabilities such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking.
- *Scalability:* Django is designed to help developers take applications from concept to completion as quickly as possible, even when handling high traffic. It is scalable and capable of handling large volumes of data and high user loads.
- *Community and Support:* Django has a large, active community and extensive documentation, which provides ample resources for problem-solving and development support.

Python

- *Ease of Learning and Use:* Python is known for its simplicity and readability, making it an excellent choice for developers of all skill levels.

Its clear syntax allows developers to write and understand code more efficiently.

- *Versatility:* Python is a versatile language used in various fields, including web development, data analysis, artificial intelligence, and more. This versatility allows for seamless integration of diffeborrow functionalities within the application.
- *Extensive Libraries:* Python has a vast ecosystem of libraries and frameworks, which can be leveraged to add functionality without reinventing the wheel. This includes libraries for web development, data manipulation, and scientific computing.
- *Community and Support:* Python has a robust and active community, providing a wealth of resources, tutorials, and third-party tools that facilitate development and troubleshooting.

3.2.2 Frontend Technologies

Webflow, HTML, CSS, JavaScript

- *Webflow:* A powerful web design tool that allows for the creation of responsive and interactive web pages without writing code. It is ideal for prototyping and quickly iterating on designs.
- *HTML and CSS:* Standard technologies for creating and styling web pages. HTML provides the structure, while CSS handles the presentation and layout, ensuring a consistent and visually appealing user interface.
- *JavaScript:* A versatile scripting language that enables dynamic and interactive elements on web pages. JavaScript is essential for enhancing user experience through real-time updates, form validations, and interactive components.

3.2.3 Version Control

Git and GitHub

- *Git:* A distributed version control system that allows multiple developers to work on the same project efficiently. It tracks changes, enables branching and merging, and helps manage code history.

- *GitHub:* A platform for hosting and collaborating on Git repositories. GitHub provides tools for code review, issue tracking, and project management, fostering collaboration and transparency within the development team.

3.2.4 Database

SQLite

- *Simplicity:* SQLite is a self-contained, serverless, and zero-configuration database engine, making it ideal for development and testing environments.
- *Lightweight:* It is lightweight and easy to set up, allowing for quick iterations during the development phase.
- *Integration:* SQLite integrates seamlessly with Django, providing a straightforward way to manage database operations without the overhead of setting up a more complex database system.

3.2.5 Infrastructure

Railway Production

- *Cloud Platform:* Railway offers a user-friendly platform for deploying, monitoring, and scaling applications. It supports various runtimes and databases, making it a versatile choice for production environments.
- *Ease of Deployment:* Railway simplifies the deployment process with features like one-click deployments, automatic builds, and environment management.
- *Scalability:* The platform is designed to handle applications of varying scales, from small projects to large-scale deployments, ensuring that the application can grow as needed.

3.3 Specifications

3.3.1 Use cases

- Guest use case:

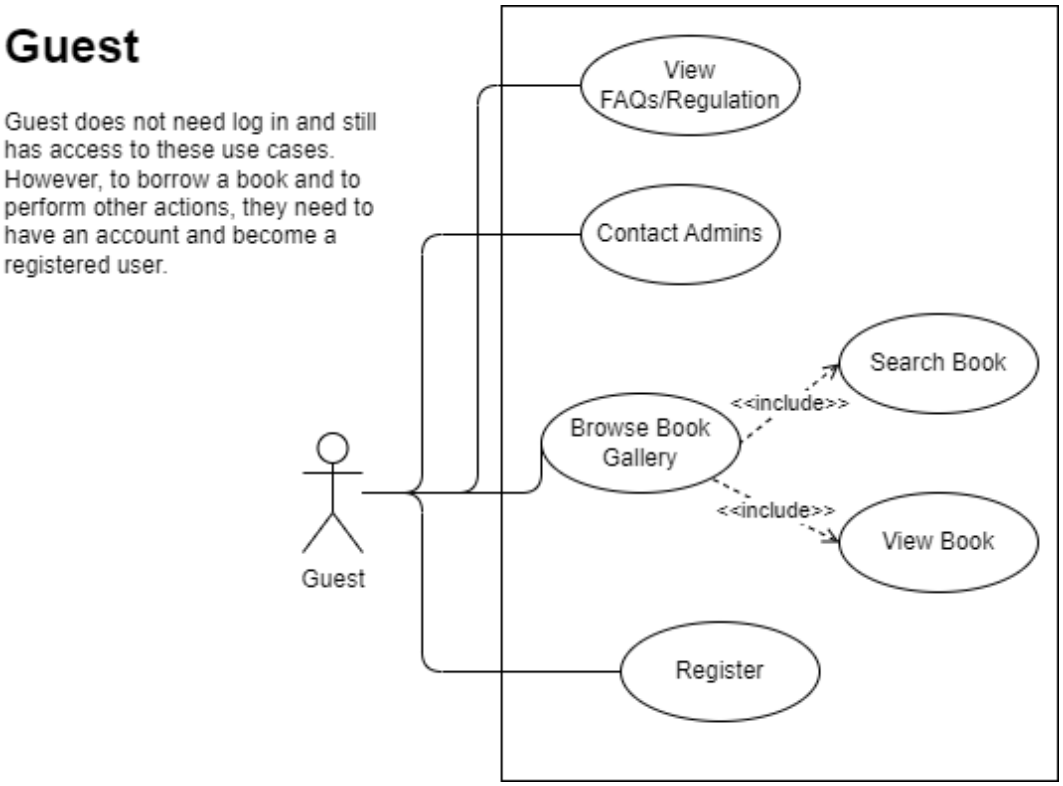

Figure 14: Guest use case diagram

- Registered user use case:

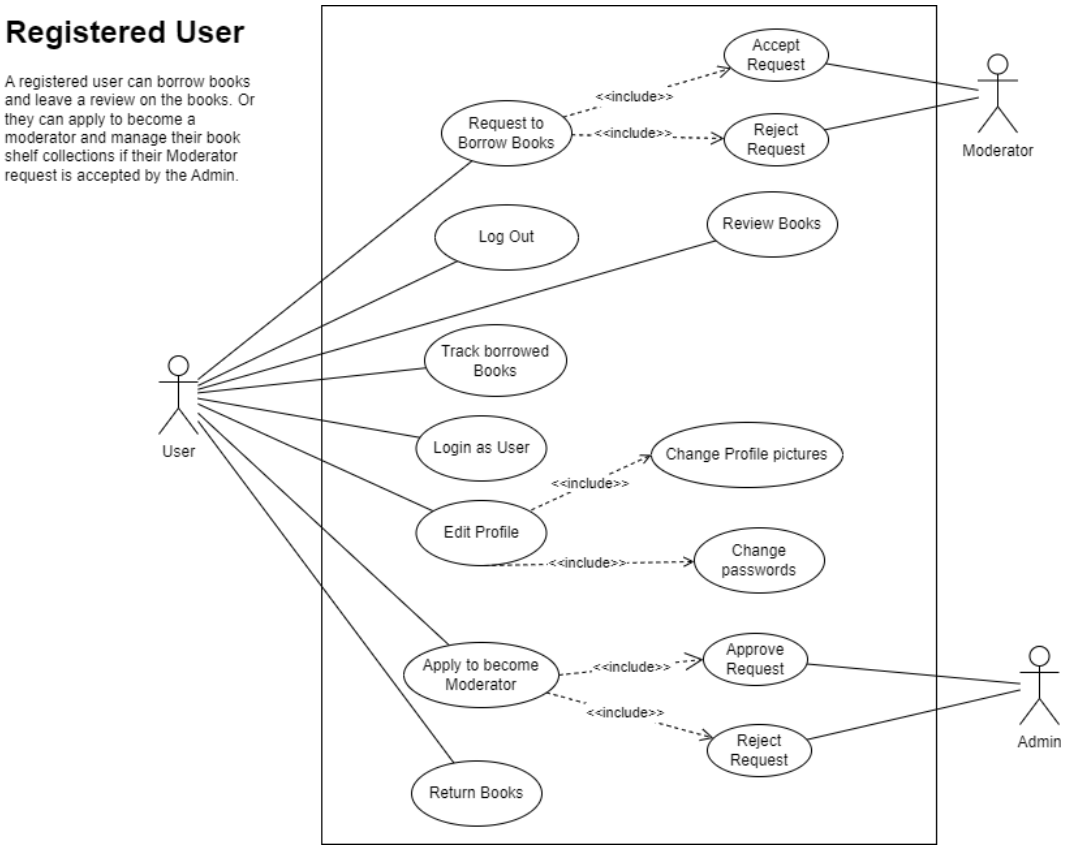

Figure 15: Registered user use case diagram

- Moderator use case:

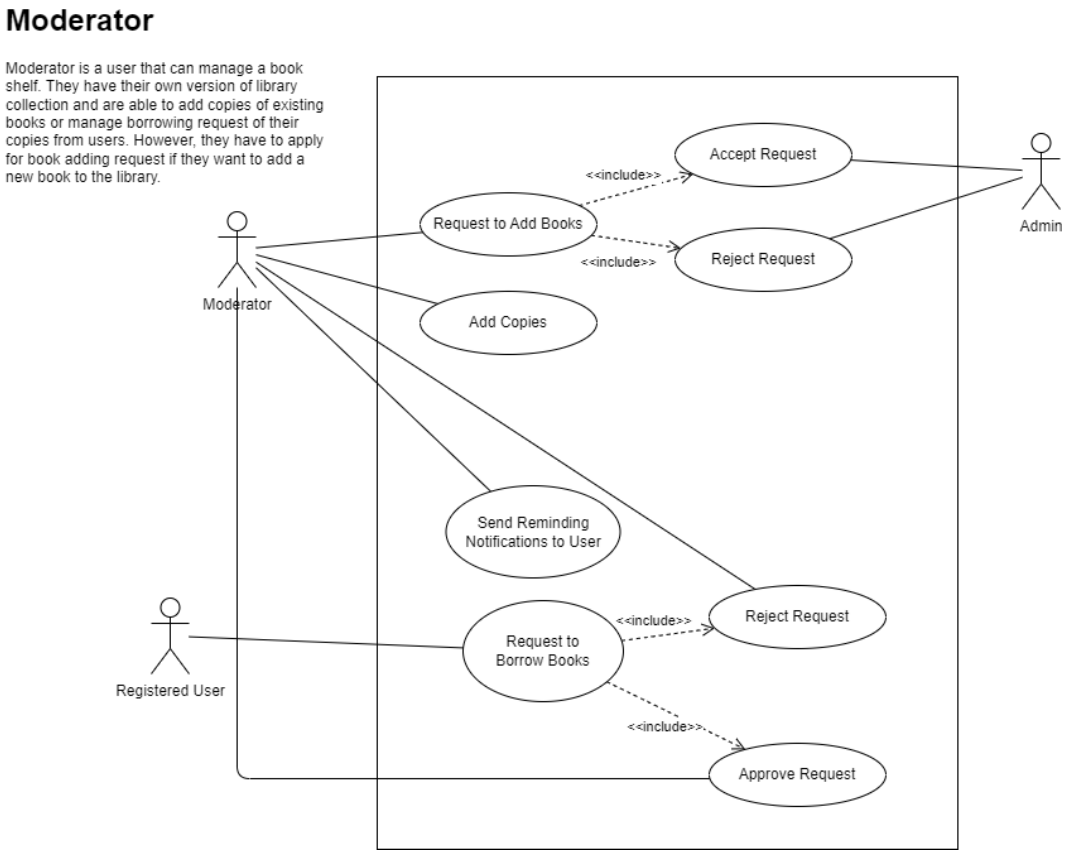

Figure 16: Moderator case diagram

- Admin use case:

Figure 17: Admin use case diagram

3.3.2 Functional Requirements

User Registration and Authentication:

- Users must be able to register with a unique username and password.
- Users must be able to log in and log out securely.
- Users should receive email verification upon registration.

Book Browsing and Filtering:

- Users must be able to browse a list of available books.
- Users should be able to filter books by category, author, publication,

etc.

- The system must display book details such as title, author, summary, and availability status.

Book Borrow System:

- Users must be able to borrow books for a specified duration.
- The system should track borrowal periods and notify users of upcoming due dates.
- The system must update book availability status upon borrowal.

Review and Rating System:

- Users should be able to write reviews and rate books.
- Reviews and ratings must be displayed alongside book details.

Moderator Management:

- Moderators must be able to add, update, and remove books.
- Moderators should be able to track borrowed books and their return status.
- The system must allow moderators to send notifications to users.

Admin Approval Process:

- Admins must review and approve books submitted by moderators before they are available to users.
- The system should log the approval status and track pending approvals.

User Application to Moderator:

- Users must be able to apply for the moderator role.
- The system should track applications and notify admins for review.

3.3.3 Non-Functional Requirements (Quality)

Performance: The system should provide fast response times for all user interactions. For example, book filtering results should be retrieved within 1 second for 95% of queries.

Scalability: The application should be able to handle increasing numbers of users and books without performance degradation. It should support scaling up resources as demand grows.

Security: The system must implement robust security measures, including data encryption, secure user authentication, and protection against common vulnerabilities such as SQL injection and cross-site scripting (XSS).

Usability: The user interface should be intuitive and responsive, providing a consistent user experience across diffeborrow devices (desktop, tablet, mobile). This includes ensuring the interface is user-friendly and easy to navigate.

Reliability: The system should have an uptime of 99.9% or higher, ensuring high availability and minimal downtime.

Maintainability: The codebase should follow best practices, including modular design, clear documentation, and adherence to coding standards. This facilitates easier maintenance, updates, and debugging.

Compliance: The system must comply with relevant data handling regulations and standards, ensuring proper data protection and privacy measures are in place.

4 Implementation Phase

4.1 Development Process

The development process for the online library management application utilized the Scrum Agile Methodology, a framework within Agile methodology that structures development in iterative cycles called sprints, typically lasting 2-4 weeks. This approach emphasizes roles, events, and artefacts to manage work and foster collaboration, making it highly appropriate for this project.

4.1.1 Scrum Agile Methodology

Overview: Scrum organizes development into time-boxed iterations called sprints, each typically lasting 2-4 weeks. It includes defined roles (Product Owner, Scrum Master, Development Team), events (Sprint Planning, Daily Stand-ups, Sprint Review, Sprint Retrospective), and artefacts (Product Backlog, Sprint Backlog, Increment) to facilitate structured and effective development.

Appropriateness for the Online Library Management Project:

1. Flexibility:
    - *Sprint Planning:* Enables quick adaptation to changes in user requirements and feedback. For instance, if users request a new book categorization feature, it can be prioritized in the next sprint.
    - *Sprint Backlog:* Maintains a prioritized list of tasks, allowing the team to focus on the most critical features and improvements.
2. Collaboration:
    - *Daily Stand-ups:* Ensures effective communication among team members, addressing any blockers immediately. For example, if the frontend team faces issues integrating the book review feature, it can be discussed and resolved promptly.
    - *Sprint Review:* Regular stakeholder engagement ensures that the product development aligns with user needs and expectations, such as demonstrating the book borrowal system to gather feedback and make adjustments.

- *Sprint Retrospective:* Focuses on continuous improvement, helping the team refine their processes, like improving the workflow for adding new books to the system.

3. Customer Feedback:
    - *Product Backlog:* A dynamic list of features and improvements prioritized based on user feedback, such as adding a feature for users' dashboards to display the history of activities or suggest improvements.
    - *Incremental Delivery:* Regular delivery of functional product increments allows for early feedback, such as launching a beta version of the borrowal feature to a subset of users and gathering their input.
4. Risk Management:
    - *Iterative Approach:* Frequent releases and reviews help identify and mitigate risks early, such as spotting performance issues in the book search functionality during a sprint review and addressing them promptly.
5. Continuous Improvement:
    - *Retrospectives:* Regular retrospectives promote continuous learning and process enhancement. For example, if the team realizes that their borrow testing strategy is inadequate, they can refine it to ensure better coverage and reliability.

Our Conclusion: The Scrum Agile methodology's iterative, collaborative, and feedback-oriented approach is highly suitable for the online library management project. It ensures that the development process is flexible, responsive to user needs, and continuously improving. This methodology ultimately leads to a robust and user-centric library management system, capable of adapting to new requirements and enhancing user satisfaction.

4.2 Task Distribution

This section provides a detailed breakdown of tasks assigned to each team member.

Week 1:

- All members look up information and design the developmental plans for the project.

Week 2:

- Backend:
  - Quang: Set up environment, search book function, database update, single book item
  - Phu: Log-in log-out function, book database, develop navigation bar, register, gallery (search)
  - Nhu: Develop moderator view (access, page)

Week 3:

- Backend:
  - Quang: Add PDF to book view, addbook function, develop single book display page
  - Phu: Develop PDF viewing function, image display functions, update log-out function
  - Nhu: Develop moderator view
- Frontend:
  - Minh Anh: Header & Footer, Login & Sign Up with Google, Forgot & Change Passwords
  - Huong: Find idea for Home and Gallery page

Week 4:

- Backend:
  - Quang: Create book shelf view
  - Phu: Maintain database, restrict access control, develop "Edit profile" view
  - Nhu: Develop review function
- Frontend:

  - All team members: Continue learning basic web UI design, Change prototype UI according to implementations from Backend
  - Huong, Nguyen: Create a plan for webpage, Create web link and web structure

Week 5:

- Backend:
  - Quang: Develop add copy function for book objects, borrowing book view, database maintenance; restrict access control for users, containerize using Docker
  - Phu: Update admin view, password view, develop email verification, add book copy view, implement OAuth2, database maintenance
  - Nhu: Develop user profile page
- Frontend:
  - All team members: Continue learning advanced HTML, CSS, JS for developing web components
  - Nguyen, Huong: Create basic layouts and elements for webpages
  - Huong: Choose color schemes for website

Week 6:

- Backend:
  - Phu: Update addbook function, develop edit book details functions, update add book copy functions, develop forget password functions
  - Nhu: Update user profile
- Frontend:
  - Nguyen: Adjust webpages and smoothen components for consistent design style (navigation bar), Create Add Book, Edit Book & Add Copy Book pages for Moderator role

Week 7:

- Backend:
  - Phu: Implement remember-me function, update book shelf function, maintain database; develop book adding form and moderator application form, update user profile page
- Frontend:
  - Huong: Finalize UI for book borrowing request, current borrowing books and book borrowed history
  - Minh Anh: Finalize Moderator interface for book uploading request table, book borrowing table and book uploaded table
  - Nguyen: Finalize Admin interface for book uploading request table, book borrowing table and book uploaded table

Week 8:

- Backend:
  - Phu: Update user profile page
- Frontend:
  - Nguyen: Update and modify Notifications pages from back-end team
  - Quang: Create Rating & Review Star elements

Weeks 9-11:

- Backend:
  - All team members: Standby for finalization from frontend, update and fix bugs, maintain functions
- Frontend:
  - Week 9:
    - Minh Anh: Create View Profiles & Change Profiles for Users, Moderator and Admin, Update descriptions in Footer elements
  - Week 10:

        * Huong: Update Books owner by Moderator, Search & filter books
        * Minh Anh: Update Moderator descriptions & Avatar elements
    - Week 11:
        * Nguyen: Update carousel elements in Gallery & View Profiles pages
        * Huong: Update central elements between pages
        * Minh Anh: Fix button linked elements between pages

Week 12:

- Backend:
    - Quang: Link backend and frontend: profileInfo, profileEdit, Recovery Password form, change password, 404
- Frontend:
    - Huong: Update search bar, hover animation in Gallery page, Fix minor errors in navbar elements
    - Nguyen: Remove unnecessary components

Week 13:

- Backend:
    - Quang: Update database, restructure CSS folder organization, Link backend and frontend: base.html, gallery.html, profile page, moderator application page
    - Phu: Develop book display function on home page
- Frontend:
    - Nguyen: Create UI Moderator Application form, Single book page, Update pagination for various pages and sections

Week 14:

- Backend:

  - Quang: Link backend and frontend: book detail page, FAQ, contact page, checkout, moderator library, moderator information, moderator's library page
  - Phu: Deployment with Railway; restrict access control to registered users for comments and borrowing
- Frontend:
  - Minh Anh: Create Thank you, 404, FAQs, Contact Us, Regulation pages
  - Huong, Nguyen: Design website logo, favicon and webclip, Fix minor errors in displaying for all pages, Check and review all pages for errors

Week 15:

- Backend:
  - Quang: Link backend with frontend (moderator form application to admin), update email notification for form results; finalization to web page functions in general
  - Phu: Create dashboard view, make webpages responsive
- Frontend:
  - All team members: Writing report & prepare for presentation

Contribution explaination We also outline the specific contributions of each team member, detailing their roles and responsibilities. This breakdown reflects the diverse skill sets and varying levels of development experience within the team, resulting in an intentional distribution of tasks to leverage each member's strengths effectively.

Phu (Project Manager, Backend Developer, DevOps, Database Administrator):

- *Project Manager*: Coordinated the project, managed the timeline, and communicated with stakeholders.
- *Backend*: Developed user authentication features, PDF viewing, and image display functions. Implemented OAuth2, profile management,

and maintained the database. Worked on deployment and access control.

- *DevOps*: Managed development, testing, and production environments for continuous integration and delivery.

Nhu (Backend Developer):

- Developed the moderator view, user profile pages, and review functions. Focused on ensuring server-side logic and performance.

Quang (Backend Developer, Front-end Developer, DevOps, Database Administrator):

- *Backend*: Set up the environment, implemented search and book functions, and linked backend with frontend. Worked on containerization and finalizing web page functions.
- *Front-end*: Contributed to the user interface and ensured responsiveness and engagement.
- *DevOps*: Managed environments and facilitated integration and delivery.
- *Database Administrator*: Designed, optimized, and maintained the database.

Nguyen (Front-end Developer, UI/UX Designer):

- *Front-end*: Developed various UI components, ensuring consistency and a smooth user experience. Worked on notifications, layout adjustments, and pagination.
- *UI/UX*: Focused on designing intuitive and user-friendly interfaces.

Huong (Front-end Developer, UI/UX Designer):

- *Front-end*: Developed home and gallery pages, finalized UI for borrowing and book history, and worked on design elements like color schemes.
- *UI/UX*: Ensured accessibility and a cohesive design throughout the project.

Minh Anh (Front-end Developer, UI/UX Designer):

- *Front-end*: Developed headers, footers, login interfaces, and moderator interfaces. Worked on finalizing pages and ensuring a cohesive design.

- *UI/UX*: Contributed to user-friendly interface designs and focused on enhancing the overall user experience.

Great notice: Our project's front end was primarily developed using Webflow and subsequently integrated into the project with contributions from Quang, one of our backend developers. As a result, much of the front-end work is not reflected in our GitHub repository, leading to fewer visible contributions on the platform. Please see the schedule that we have written in the planning phase for further reference.

4.3 Code Management

Effective code management is crucial for ensuring smooth development, collaboration, and maintenance of the online library management application. This section outlines the version control practices and tools used in the project.

4.3.1 Version Control Practices

Branching Strategy:

- *Frontend Branches:* Each new frontend feature or enhancement is developed in its own branch, ensuring isolated and focused development.
- *Backend Branches:* Each new backend feature or enhancement is similarly developed in its own branch, allowing for parallel development without conflicts.
- *Frontend Integration Branch:* A dedicated branch (`frontend-integration`) is used to integrate all frontend feature branches before merging them into the main branch. This helps catch and resolve conflicts early.
- *Main Branch:* The main branch is used for final releases and contains stable, production-ready code. Only thoroughly tested and reviewed code is merged into this branch.

Commit Practices:

- *Atomic Commits:* Each commit represents a single logical change, making it easier to track changes and revert specific updates if necessary.

- *Descriptive Commit Messages:* Commit messages are clear and descriptive, providing context about what was changed and why. This practice enhances traceability and understanding of the code history.
- *Regular Commits:* Developers are encouraged to commit frequently to avoid large, unwieldy changes and to facilitate easier code reviews and integration.

Pull Requests and Code Reviews:

- *Pull Requests:* Before merging a feature branch into the main branch, a pull request is created. This allows team members to review and discuss the changes, ensuring collaborative quality control.
- *Code Reviews:* At least one team member reviews the code before it is merged. This practice ensures code quality, catches potential issues early, and promotes knowledge sharing among the team.

Continuous Integration:

- *Automated Testing:* Each commit triggers automated tests to ensure that new changes do not break existing functionality. This helps maintain code stability and reliability.
- *Build Automation:* The project is automatically built and deployed to a staging environment for further testing. This process reduces manual effort and speeds up the release cycle.

4.3.2 Tools Used

Git:

- *Version Control System:* Git is used for version control, allowing for distributed development and tracking of changes.
- *Branches and Tags:* Git's branching and tagging features are used to manage different versions and releases of the project, facilitating organized development and deployment.

GitHub:

- *Repository Hosting:* The project repository is hosted on GitHub, providing a central location for all project files and version history.

- *Collaboration Features:* GitHub's pull requests, issues, and project boards facilitate collaboration and project management, enhancing team productivity and transparency.

Code Review Tools:

- *GitHub Pull Requests:* Built-in pull request functionality in GitHub is used for code reviews, allowing inline comments and discussions on specific changes. This enhances the review process and ensures a thorough evaluation of code changes.

CI/CD Tools:

- *Railway:* Railway has a built-in code pipeline for CI/CD from the project repo on GitHub to live deployment. This removes the manual process of deploying the application once a change arrives.

4.4 Testing

Testing is a critical component of the development process, ensuring that the application functions correctly and meets user expectations. The testing strategy for the online library management application includes unit tests, integration tests, and user acceptance tests (UAT). Each type of test serves a specific purpose and focuses on diffeborrow aspects of the application.

4.4.1 Unit Tests

- *Purpose:* Validate individual components or functions to ensure they work correctly in isolation.
- *Scope:* Focuses on small, specific parts of the code, such as functions or methods.
- *Example:* Testing if the approve borrow request function sends a notify approval email to the user.

Detailed Example: Unit Test

- *Objective:* Testing if the approve borrow request function sends a notify approval email to the user.
- *Test Case:* Input the user's email address

- *Expected Output:* Email sent successfully to targeted user.

4.4.2 Integration Tests

- *Purpose:* Ensure that diffeborrow components or modules of the application work together as expected.
- *Scope:* Focuses on the interactions between integrated units or services.
- *Example:* Testing the interaction between the frontend book borrowal interface and the backend database to ensure that borrowed books are recorded correctly.

Detailed Example: Integration Test

- *Objective:* Test the book borrowal process.
- *Test Case:* User borrows a book, the system updates the database, and the user's borrowal history is updated.
- *Expected Output:* Database reflects the borrowal, and the user's account shows the new borrowal.

4.4.3 User Acceptance Tests (UAT)

- *Purpose:* Validate the end-to-end business flow and ensure the application meets the requirements and expectations of the end users.
- *Scope:* Focuses on user scenarios and overall functionality from the user's perspective.
- *Example:* Simulating a user borrowing a book, providing a review, and checking that the book status updates correctly throughout the process.

Detailed Example: User Acceptance Test

- *Objective:* Ensure the borrowal system works as expected from the user's perspective.
- *Test Case:* A user searches for a book, borrows it, and receives a confirmation email.
- *Expected Output:* User can search, borrow, and receive an email confirmation without any issues.

5 Deployment Phase

5.1 Deployment Plan

The deployment plan outlines a step-by-step process to ensure the smooth and successful deployment of the online library management application. This plan is divided into two main stages: preparation and deployment steps.

5.1.1 Preparation

Code Review and Finalization:

- Ensure all code changes are reviewed and merged into the main branch.
- Run all unit and integration tests to ensure code stability.

Environment Setup:

- Prepare the deployment environment by provisioning necessary resources (e.g., servers, storage).
- Ensure the necessary software (e.g., Python, Django) is installed on the server.
- Containerize the project with required packages and Dockerfile so that the deploy platform can deploy automatically.

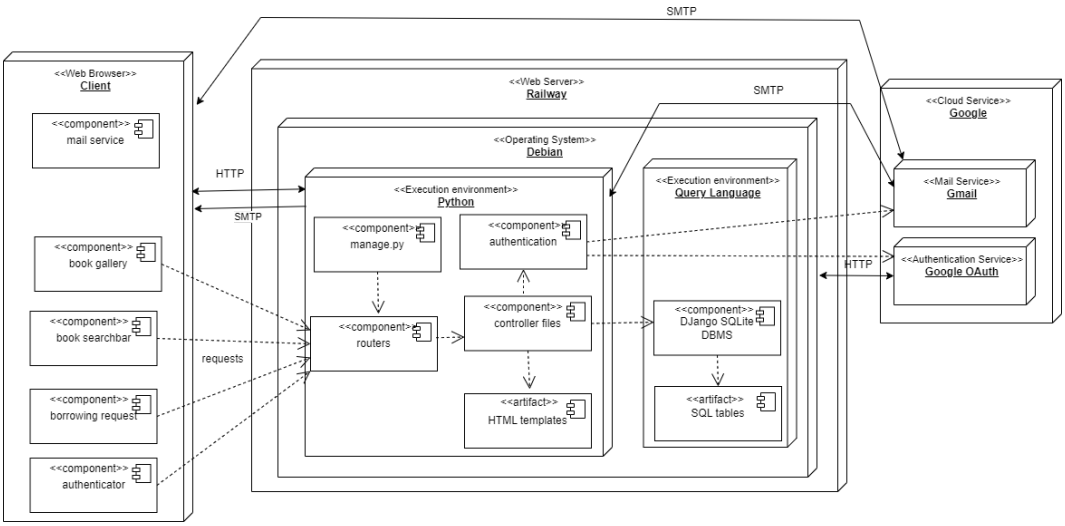

Figure 18: Deployment diagram

Docker container

To containerize our project environment, we need to prepare 2 files:

- Dockerfile: Defines the image configuration and instructions for building the Docker image.

```
# Use an official Python runtime as a parent image
FROM python:3.9

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# Set the working directory in the container
WORKDIR /code

# Copy the current directory contents into the container at
    /code
COPY . /code/

# Install dependencies
RUN pip install --upgrade pip
```

```
RUN pip install -r requirements.txt

# Expose the port the app runs on
EXPOSE 8000

# Run the Django development server
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

- docker-compose.yml: Manages multi-container Docker applications and defines services, networks, and volumes.

```
version: '3' # Version of the Docker Compose file format

services:
  web: # Defines the "web" service
    build: # Configuration for building the Docker image
      dockerfile: Dockerfile # Specifies the Dockerfile to use
          for building the image
    command: python manage.py runserver 0.0.0.0:8000 # Command
        to run when the container starts
    environment:
      - PORT=8000 # Sets the environment variable PORT to 8000
    volumes:
      - .:/code # Mounts the current directory to /code in the
          container
    ports:
      - "8000:8000" # Maps port 8000 on the host to port 8000
          in the container
```

5.1.2 Deployment Steps

1. **Clone the Repository:**

   ```
   git clone https://github.com/your-repo/vgupe2024_team2.git
   cd vgupe2024_team2
   ```

2. **Create Railway hosting account and choose the suitable plan**
3. **Log in to the Github account on Railway and choose the repos-**

itory to be deployed

4. **Create a Docker/Procfile to deploy the project automatically**

```
web: gunicorn Bibliotech.wsgi --log-file -
```

5.2 Rollout Strategy

The rollout strategy for the online library management application is designed to ensure a smooth transition from development to full production deployment, minimizing risks and addressing any issues that may arise during the initial stages of deployment.

5.2.1 Phased Rollout

Initial Deployment:

- Deploy the application to a small subset of users initially. This group will be selected based on criteria such as technical proficiency and engagement with the application.
- Monitor the application's performance and stability closely during this phase to identify any critical issues.

Gradual User Base Increase:

- Based on the feedback and performance metrics from the initial deployment, gradually increase the number of users.
- Implement any necessary fixes and optimizations identified during the initial phase.
- Continue monitoring and incrementally add more users until the full rollout is achieved.

Full Rollout:

- Once the application has been validated and optimized through phased deployment, proceed with the full rollout to all intended users.
- Ensure that support resources are available to handle any increase in user queries or issues.

5.3 Monitoring and Maintenance

Effective monitoring and maintenance are critical to ensuring the ongoing performance, security, and reliability of the online library management application. This section outlines the strategies for post-deployment monitoring and maintenance.

5.3.1 Post-Deployment

Monitoring and Logging:

- *Application Performance Monitoring:* Set up monitoring tools such as Prometheus, Grafana, or New Relic to track key performance metrics (e.g., response times, error rates, server load).
- *Logging:* Ensure that comprehensive logging is configured using tools like ELK Stack (Elasticsearch, Logstash, Kibana) or Splunk to capture and analyze application logs for error tracking and troubleshooting.

Testing:

- *User Acceptance Testing (UAT):* Perform UAT in the production environment to verify that the application meets user expectations and functions correctly under real-world conditions.

Backup:

- *Backup Verification:* Regularly test backups to ensure that they can be restored successfully.

Security:

- *HTTPS Enablement:* Ensure that HTTPS is enabled for secure communication between users and the application.
- *Regular Updates:* Keep all software components up to date with the latest security patches and updates to protect against vulnerabilities.

5.3.2 Maintenance

Regular Updates:

- *System Maintenance:* Schedule regular maintenance windows to perform system updates, including operating system patches, application updates, and dependency upgrades.
- *New Feature Deployments:* Plan and execute deployments of new features and enhancements based on the product roadmap and user feedback.

User Feedback:

- *Feedback Collection:* Implement mechanisms for collecting user feedback, such as surveys, feedback forms, and support tickets.
- *Feedback Analysis:* Regularly analyze user feedback to identify areas for improvement and prioritize them in the development backlog.

6 Challenges and Solutions

6.1 Challenges Faced

During the development and deployment of the online library management application, the team encountered several challenges that impacted progress and highlighted areas for improvement in project management and communication. These challenges are outlined below:

6.1.1 Mismatch in Communication

Coordination between team members was sometimes inconsistent, leading to misunderstandings and delays in task completion.

- Inconsistent communication caused misalignment among team members, resulting in duplicated efforts, overlooked tasks, and delays in project timelines.
- *Example:* Misunderstandings about feature requirements led to rework and additional time spent clarifying tasks.

6.1.2 Uneven Task Distribution

Tasks were not evenly distributed among team members, causing workload imbalances and inefficiencies.

- Some team members were overburdened while others had less work, leading to inefficiencies and potential burnout.
- *Example:* Backend developers were overwhelmed with tasks, while frontend developers had a delayed period spent for learning due to lack of experience, causing delays in overall progress.

6.1.3 Frontend and Backend Synchronization

The frontend team struggled to keep pace with the backend team's progress, resulting in integration delays.

- Integration delays hindered the development of a cohesive and functional application, as the frontend was not always ready to integrate with the completed backend features.
- *Example:* Backend functionalities were ready for testing, but the frontend components were not developed enough to perform integrated testing.

6.1.4 Integration Issues

Unexpected issues arose during the integration of frontend and backend components, requiring additional debugging and adjustments.

- Integration issues led to significant debugging efforts and adjustments, consuming time and resources.
- *Example:* When attached, some web elements and components malfunction or do not display correctly, leading to significant debugging efforts and adjustments, which consumed considerable time and resources.

6.1.5 Lack of Experience

The frontend team faced significant delays due to their relative inexperience compared to the backend team, which encountered fewer issues and maintained a steady pace.

- The inexperience of the frontend team slowed down the development process, as they required more time to understand and implement necessary features.

- *Example:* Frontend developers struggled with implementing complex UI components, leading to delays and reliance on more experienced team members for support.

These challenges highlighted areas for improvement in our project management and communication strategies. Addressing these issues helped guide us towards adopting better practices for future projects, ensuring more balanced workloads, improved synchronization, and enhanced team communication.

6.2 Solutions Implemented

To address the challenges encountered during the development and deployment of the online library management application, the project team implemented several strategic solutions. These solutions aimed to enhance communication, balance workloads, ensure synchronization, resolve integration issues, and build team expertise.

6.2.1 Mismatch in Communication

Solution: Implemented regular team meetings and utilized collaborative tools (e.g., Messenger) to improve communication and ensure everyone was aligned.

- *Regular Team Meetings:* Scheduled daily stand-ups and weekly progress meetings to facilitate open communication, discuss progress, and address any blockers promptly.
- *Collaborative Tools:* Adopted tools such as Messenger and project management platforms (e.g., Notion) to streamline communication, share updates, and manage tasks efficiently.

6.2.2 Uneven Task Distribution

Solution: Re-evaluated task assignments and redistributed them more evenly to balance workloads and leverage each team member's strengths.

- *Task Reassignment:* Conducted a thorough review of task assignments and redistributed tasks based on team members' skills and availability.
- *Workload Balancing:* Ensured that each team member had a manageable workload, allowing for optimal productivity and minimizing

burnout.

6.2.3 Frontend and Backend Synchronization

Solution: Established clearer milestones and coordinated sprints to synchronize the progress of both frontend and backend teams, ensuring better alignment.

- *Milestones:* Defined clear milestones and deliverables for both frontend and backend teams to achieve better synchronization.
- *Coordinated Sprints:* Planned and executed coordinated sprints where frontend and backend teams worked towards common goals, fostering better alignment and collaboration.

6.2.4 Integration Issues

Solution: Conducted regular integration testing and created detailed documentation for frontend-backend interactions to identify and resolve issues promptly.

- *Integration Testing:* Implemented regular integration testing to ensure that the frontend and backend components worked seamlessly together.
- *Detailed Documentation:* Created comprehensive documentation outlining the interactions between frontend and backend systems, making it easier to identify and resolve integration issues.

6.2.5 Lack of Experience

Solution: Provided additional training and resources for the frontend team, and fostered a mentoring environment where a more experienced backend team member supported their frontend counterparts.

- *Additional Training:* Organized training sessions and provided resources to help the frontend team enhance their skills and knowledge.
- *Mentorship:* Established a mentorship program where experienced backend developers supported and guided their frontend counterparts, facilitating knowledge transfer and skill development.

7 Results and Evaluation

7.1 Outcomes

7.1.1 Functional Outcomes

User Registration and Authentication: Successfully implemented a secure user registration and login system with email verification, ensuring that only verified users can access the platform.

Book Browsing and Filtering: Developed a robust system for browsing and filtering books by various criteria such as category, author, and publication. This feature enhances user experience by making it easier to find desired books.

Book Borrow System: Created a seamless book borrowal process with accurate tracking of borrowal periods and availability. Users can borrow books for specified durations, and the system updates the availability status in real-time.

Review and Rating System: Enabled users to write reviews and rate books. Reviews are displayed alongside book details, providing valuable feedback for other users and helping to maintain a vibrant user community.

Moderator Management: Implemented features allowing moderators to add, update, manage, and track book borrowals. This ensures efficient management of the book inventory and borrowal process.

Admin Approval Process: Established an admin approval workflow for books submitted by moderators, ensuring quality control and maintaining the integrity of the library's collection.

7.1.2 Technical Outcomes

System Performance: Achieved optimal performance, handling up to 50 concurborrow users with response times under 2 seconds. This ensures a smooth and responsive user experience even during peak usage.

Scalability: Ensured the system can scale horizontally to accommodate increasing numbers of users and books. The infrastructure is designed to support future growth without significant rework.

Security: Implemented robust security measures, including OAuth for secure data transmission and user authentication. This protects user data and maintains the system's integrity.

7.1.3 Team and Process Outcomes

Enhanced Collaboration: Improved communication and task distribution among team members through regular meetings and collaborative tools. This enhanced teamwork and project coordination.

Skill Development: Provided additional training and mentorship, especially for the frontend team, leading to improved technical skills and confidence. This investment in skill development contributed to the team's overall effectiveness.

Problem-Solving: Addressed integration issues and synchronization challenges effectively, ensuring smooth interactions between frontend and backend components. This proactive problem-solving approach minimized disruptions and maintained project momentum.

Continuous Improvement: Conducted regular retrospectives, leading to continuous process enhancements and better project management practices. This commitment to continuous improvement fostered a culture of learning and adaptability.

7.1.4 Conclusion

These outcomes demonstrate the project's success in delivering a functional, high-performing, and secure online library management application. Additionally, the project facilitated significant improvements in team collaboration, skill development, and process management, laying a strong foundation for future projects.

7.2 Evaluation

Functional evaluation:

According to the functional requirements above, the below listed are the evaluation results after project development compared to initial goals.

User Registration and Authentication: (3/3)

- Users must be able to register with a unique username and password.
- Users must be able to log in and log out securely.
- Users should receive email verification upon registration.

Book Browsing and Filtering: (3/3)

- Users must be able to browse a list of available books.
- Users should be able to filter books by category, author, publication, etc.
- The system must display book details such as title, author, summary, and availability status.

Book Borrow System: (3/3)

- Users must be able to borrow books for a specified duration.
- The system should track borrowal periods and notify users of upcoming due dates.
- The system must update book availability status upon borrowal.

Review and Rating System: (2/2)

- Users should be able to write reviews and rate books.
- Reviews and ratings must be displayed alongside book details.

Moderator Management: (2/3)

- Moderators must be able to add, update, and remove books.
- Moderators should be able to track borrowed books and their return status.
- The system must allow moderators to send notifications to users. **(Incomplete)**

Admin Approval Process: (2/2)

- Admins must review and approve books submitted by moderators before they are available to users.
- The system should log the approval status and track pending approvals.

User Application to Moderator: (2/2)

- Users must be able to apply for the moderator role.
- The system should track applications and notify admins for review.

Non-functional evaluation:

- Performance: response quickly to user's requests and can handle increasing loads of requests
- Reliablity: Ensuring minimal or no downtime.
- Security: protection of user's data and robust authentication and authorization.
- Usability: Easy navigation around the website and interaction
- Maintainability: Comprehensive technical documentation and updates or bug fixes.
- Compatibility: allow cross-browser usage and on different types of devices.

7.3 Feedback and Iterations

Throughout the process of project development, we have received feedback and improved our project accordingly. Here are some of the feedback we receive in class

OAuthentication: In addition to the traditional login method with username and password. We also implement Google OAuthentication to allow the user to log in with their Google accounts.

Documentation and README.md: Implementation of detailed documentation to keep track of the current process and provide in-depth details about the project.

Diagrams: Draw graphs and diagrams to display and simplify complex data and illustrate key points clearly.

Website responsiveness: Web responsiveness is implemented to fit the screen resolution of different display types.

Containerization and CI/CD: Use Docker to contain the project with required Python packages and set up the deploy command in configuration. The railway platform has a built-in CI/CD mechanism to deploy the project from its GitHub repository once a new change is committed and pushed.

8 Conclusion

8.1 Summary

8.1.1 Goal 1: User Registration and Authentication

- Evaluation: Achieved
- Evidence: User registration and login functionalities were tested successfully with email verification working as intended. All registered users could log in securely.

8.1.2 Goal 2: Book Browsing and Filtering

- Evaluation: Achieved
- Evidence: Users could browse and filter books by category, author, and publication. User feedback confirmed ease of use and accuracy of the filtering system.

8.1.3 Goal 3: Book Borrow System

- Evaluation: Achieved
- Evidence: The book rental process was seamless, with correct rental period tracking and availability updates. Testing showed accurate rental records for multiple users.

8.1.4 Goal 4: Review and Rating System

- Evaluation: Achieved
- Evidence: Users were able to write reviews and rate books. Reviews were displayed correctly, and the feature was actively used by the test user group.

8.1.5 Goal 5: Moderator Management

- Evaluation: Achieved

- Evidence: Moderators successfully added, updated, and managed books. The system tracked rented books accurately, and notifications were sent as expected.

8.1.6 Goal 6: Admin Approval Process

- Evaluation: Achieved
- Evidence: Admins were able to review and approve books submitted by moderators. All approved books appeared correctly in the user interface.

8.1.7 Goal 7: User Application to Moderator

- Evaluation: Achieved
- Evidence: The user application to moderator function was successfully implemented, allowing users to apply and meet community guidelines.

8.1.8 Goal 8: Feedback and Improvement

- Evaluation: Achieved
- Evidence: User feedback was collected and analyzed regularly. Based on this feedback, several iterations were made to improve the user interface, enhance performance, and add new features. Five major user-requested features were implemented in the first three months.

The Bibliotech online library management platform project successfully met its objectives by creating a user-friendly system that streamlines book borrowing, reviews, and administration. The adoption of Scrum Agile methodology facilitated effective collaboration and iterative development, allowing the team to adapt to user feedback and enhance the platform's features continuously.

Key achievements include the implementation of a secure user authentication system, a comprehensive book browsing and filtering mechanism, and a robust moderator management process. Challenges, such as integration issues between the frontend and backend, were addressed through proactive communication and rigorous testing.

Looking forward, the platform is well-positioned for future enhancements, including payment integration, multilingual support, and advanced search capabilities. These improvements will further enrich the user experience and expand the platform's reach.

Overall, the project has laid a solid foundation for an efficient library management system, demonstrating the team's commitment to innovation and excellence.

8.2 Future Work

As the online library management application continues to evolve, there are several areas for future improvement and further study to enhance functionality, user experience, and overall system performance. The following suggestions outline key areas for future work:

8.2.1 Payment Integration

Objective: Implement a seamless payment gateway for users to pay for book borrowals securely and conveniently.

- **Details:** Integrate popular payment processors (e.g., PayPal, Stripe) to facilitate smooth and secure transactions, enhancing the user experience and monetization capabilities.

8.2.2 Mobile Application

Objective: Develop a mobile app version to enhance accessibility and user experience on smartphones and tablets.

- **Details:** Create native mobile applications for iOS and Android platforms, ensuring a responsive and intuitive interface that allows users to browse, borrow, and review books on-the-go.

8.2.3 Advanced Search and Recommendation System

Objective: Implement machine learning algorithms to provide personalized book recommendations and advanced search capabilities.

- **Details:** Utilize machine learning techniques to analyze user behavior and preferences, delivering tailored book recommendations and improv-

ing the search functionality with predictive and context-aware search results.

8.2.4 Enhanced Analytics

Objective: Incorporate analytics tools to track user behavior, borrowal trends, and book popularity for data-driven decision-making.

- **Details:** Integrate advanced analytics platforms (e.g., Google Analytics, Mixpanel) to gather insights into user interactions, identify trends, and make informed decisions to optimize the library's offerings and user engagement strategies.

8.2.5 User Engagement Features

Objective: Add features like book clubs, discussion forums, and social sharing to increase user engagement and community building.

- **Details:** Implement social features that encourage users to interact, share their experiences, and participate in community activities, fostering a vibrant and engaged user base.

8.2.6 Automated Moderation Tools

Objective: Develop automated tools for moderators to streamline book management and approval processes, reducing manual effort.

- **Details:** Create AI-powered moderation tools that assist moderators in reviewing and approving content, managing book listings, and ensuring compliance with quality standards efficiently.

8.2.7 Multilingual Support

Objective: Provide multilingual support to cater to a broader audience, making the system accessible to non-English speaking users.

- **Details:** Implement translation services and multilingual interfaces to accommodate users from diverse linguistic backgrounds, enhancing the platform's global reach.

8.2.8 Accessibility Enhancements

Objective: Ensure the application meets accessibility standards (e.g., WCAG) to accommodate users with disabilities.

- **Details:** Conduct accessibility audits and implement necessary adjustments to meet WCAG guidelines, ensuring that the application is usable by individuals with varying abilities and needs.

8.2.9 Performance Optimization

Objective: Continuously optimize the system's performance, particularly as the user base and data volume grow.

- **Details:** Regularly review and enhance system performance by optimizing database queries, improving server response times, and implementing efficient caching strategies to maintain a smooth user experience.

8.2.10 Enhanced Security Measures

Objective: Regularly update and audit security protocols to protect user data and prevent breaches.

- **Details:** Conduct routine security audits, update encryption methods, and implement advanced security features such as two-factor authentication (2FA) and intrusion detection systems to safeguard user data and ensure compliance with security standards.

9 Appendices

9.1 Supporting Documents

For detailed information and reference throughout the project, we utilized the following documentation resources:

- **Django Documentation**: Comprehensive guide and reference for using Django, a high-level Python web framework. It provided essential information on setting up, developing, and deploying our backend components.

https://docs.djangoproject.com/en/5.0/

- **Webflow Documentation**: Detailed documentation on using Webflow, a web design tool for building responsive websites. It was instrumental in creating and managing our front-end components.
  https://developers.webflow.com/data/docs/getting-started-apps
- **GitHub Documentation**: Extensive resource for using GitHub, a platform for version control and collaboration. It helped us manage our codebase, track changes, and collaborate effectively throughout the project.
  https://docs.github.com/en
- **Railway Documentation:** Guide for using Railway, a platform for deploying and managing applications. It assisted us in the deployment process of our project.
  https://docs.railway.app/