Software Requirements Specification

for

Book Bridge

Version 1.0

Prepared by

Group Name: The Catalysts

Manjunath Lakkundi	<1RVU23CSE257>	manjunathl.btech23@rvu.edu.in
Magala Chetan Kumar	<1RVU23CSE247>	magalac.btech23@rvu.edu.in
B Pranith Reddy	<1RVU23CSE102>	bpranithr.btech23@rvu.edu.in
Harish P C	<1RVU23CSE181>	harishpc.btech23@rvu.edu.in

Instructor: Harikumar Vasudevan Pillai Santhibhavan

Course: Agile Software and DevOps

Lab Section: *Personal laptops, local setup*

Teaching Assistant: Harikumar Vasudevan Pillai Santhibhavan

Date: 27-01-2024

INDEX

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Harish P C	Initial draft of SRS document, including lab section and project overview.	01/16/2025
1.1	M Chetan Kumar	Added functional requirements and detailed system design.	01/18/2025

Version	Primary Author(s)	Description of Version	Date Completed
1.2	Manjunath Lakkundi	Refined user roles, testing procedures, and finalized the lab section.	01/21/2025
1.3	B Pranith Reddy	Updated revision table and added more details to the system features.	01/25/2025

1 Introduction

1.1 *Book Bridge is an innovative platform designed to connect readers and book lenders in a seamless and efficient way. The system aims to create a community-driven solution where users can borrow, buy and rate books, providing access to a wide variety of literature without the need for purchasing every book. The platform features an easy-to-use interface that allows users to register, search for available books, and request or lend books to others. The project is built using React for the frontend, Node.js, Express js for the backend, and MongoDB as the database, ensuring a robust and scalable solution.*

1.2 Document Purpose

*This Software Requirements Specification (SRS) document describes the software requirements for **BookBridge**, a platform designed to connect readers and book lenders. The current release is version **1.0**, which includes the fundamental features of the platform, such as user registration, book borrowing, and lending functionalities. This document outlines the system's architecture, functional and non-functional requirements, and user roles, ensuring a comprehensive understanding of the software.*

*The scope of this SRS covers the entire **BookBridge** system, including both frontend and backend components. It details the core features necessary to enable users to interact with the platform, access books, and manage transactions. While this SRS captures the foundational elements of the system, future revisions will expand the scope to include additional features such as recommendations, advanced search functionalities, and social integration.*

1.3 Product Scope

***BookBridge** is a digital platform that facilitates the exchange of books between readers and book lenders, aiming to create a sustainable and community-driven solution for accessing literature. The purpose of the software is to provide users with a convenient and user-friendly interface to register, borrow, and lend books. This platform eliminates the need for purchasing every book, allowing users to share books within a trusted network, promoting both cost-saving and environmentally friendly practices.*

*The key objectives of **BookBridge** include offering a wide selection of books for users to borrow, ensuring secure transactions between borrowers and lenders, and creating an easy-to-use platform that encourages community interaction. The benefits of this product include fostering a sense of community, increasing access to books without financial burden, and reducing the environmental impact associated with new book production. Additionally, **BookBridge** helps create an ecosystem where users can connect over shared interests and discover new reading material in a collaborative and socially responsible manner.*

1.4 Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is primarily intended for multiple stakeholders involved in the development and evaluation of the ***BookBridge*** *project. The key readers include:*

1. ***Client****: This SRS provides the client (in this case, the professor) with an overview of the project's features, objectives, and benefits. It outlines how the system will meet their requirements and expectations, helping to ensure the product aligns with the vision and scope.*
2. ***Developers****: Developers will find detailed functional and non-functional requirements in the document, as well as system architecture and technical specifications to guide the implementation of the project.*
3. ***Project Managers****: This section aids project managers by providing insight into the project's scope, timeline, and milestones, helping them manage the project effectively.*
4. ***Testers****: Testers can refer to the functional requirements, test cases, and validation criteria to ensure the system behaves as expected during testing phases.*
5. ***Documentation Writers****: Writers who are responsible for maintaining the documentation will find a structured and clear reference to help them create user manuals, technical documentation, and support materials.*

Document Organization

The document is organized in the following way to cater to different reader types:

1. ***Introduction and Overview****: The first sections introduce the* ***BookBridge*** *project, its purpose, and scope. These sections are essential for understanding the overall vision and goals of the product and are relevant to all readers.*
2. ***System Requirements****: This section includes detailed functional and non-functional requirements. Developers and testers will focus on this section to understand what the system must do and the constraints under which it must operate.*
3. ***System Design****: This section provides architectural details of the system, including the components and their interactions. Developers and project managers will find this part critical for the implementation phase.*
4. ***Testing and Validation****: A comprehensive description of how the system will be tested. Testers will find this section valuable for planning and executing test cases.*
5. ***Appendices and Glossary****: Any supplementary material or definitions used throughout the document will be available in the appendices. This is useful for all readers to clarify terms and abbreviations.*

1.5 Definitions, Acronyms and Abbreviations

Abbreviation	**Definition**
API	Application Programme Interface

DB	Database
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JSON	Javascript Object Notation
SRS	Software Requirement Specification
UI	User Interface
UX	User Experience

1.6 Document Conventions

This document follows the IEEE formatting requirements for Software Requirements Specifications (SRS). The conventions used in this document are as follows:

Formatting Conventions

- ***Font**: All text is written in **Arial**, with a font size of **11** or **12** throughout the document.*
- ***Line Spacing**: The document uses **single spacing** for text.*
- ***Margins**: A standard margin of **1"** is maintained on all sides.*
- ***Comments**: Text marked as comments or additional notes is written in **italics** to distinguish it from the main content.*
- ***Headings**: Section and subsection titles are formatted according to the IEEE template for consistent organization and easy navigation.*

Naming Conventions

- ***System Components**: System components (such as "BookBridge,", "User Interface," and "Database") are capitalized to indicate their importance as key elements of the system.*
- ***Technical Terms and Acronyms**: Abbreviations and technical terms are defined the first time they are used and are written in their full form, followed by the acronym in parentheses.*
- ***Variables and Code**: Code snippets, variable names, and technical references are written in **monospace font** for clear differentiation from regular text.*

1.7 References and Acknowledgments

*The following documents and resources have been referenced in the preparation of this Software Requirements Specification (SRS) for the **BookBridge** project:*

1. ***IEEE Std 830-1998*** *- IEEE Recommended Practice for Software Requirements Specifications.*
2. ***MongoDB Documentation*** *- Official documentation for MongoDB, used as the database for* ***BookBridge****. https://www.mongodb.com/docs*
3. ***Node.js Documentation*** *- Official documentation for Node.js, used as the backend runtime. https://nodejs.org/en/docs*
4. ***Express.js Documentation*** *- Official documentation for Express.js, used for creating the server-side API. https://expressjs.com/en/starter/installing.html*
5. ***React Documentation*** *- Official documentation for React, in case it's used for frontend development or additional components. https://reactjs.org/docs/getting-started.html*

2 Overall Description

2.1 Product Overview

BookBridge *is a new, self-contained product that aims to create a platform for book sharing and borrowing within a community. The idea originated from the need for an eco-friendly, cost-effective, and socially responsible way to access books without the need to purchase new copies. It is not a follow-on member of a product family or a replacement for any existing systems but a new initiative designed to cater to the growing interest in community-driven resources and collaborative consumption.*

The ***BookBridge*** *platform will serve as a connecting point for users looking to borrow and lend books. The system will allow users to sign up, browse available books, and borrow or lend books in a secure and user-friendly manner. The platform will integrate a backend system using* ***Node.js*** *with a* ***MongoDB*** *database, and a frontend built with* ***Angular*** *to provide a smooth and intuitive user experience. The system will be accessible via web browsers, making it easy for users to interact with it from anywhere.*

- ***Frontend (Angular)****: The frontend serves as the client-side interface, handling user interactions and displaying data.*
- ***Backend (Node.js)****: The backend handles business logic, API requests, and interactions with the database.*
- ***Database (MongoDB)****: The database stores user information, book listings, transaction data, etc.*
- ***API (Express.js)****: An Express.js-based API handles communication between the frontend and backend, ensuring smooth data flow.*
- ***External Services****: These services may include email notifications for user activity, such as borrowing a book or lending a book, and potential integrations with payment systems for any premium services offered.*

- *This product operates independently but may have integrations with external systems for specific functionalities. The product will be continuously enhanced to add more features as required by users, such as recommendations, reviews, and social features.*

2.2 Product Functionality

*The **BookBridge** platform will provide the following major functions:*

- ***User Registration and Login**: Users can register, log in, and manage their profiles.*
- ***Book Listing**: Users can list books they want to lend or borrow, including adding book details like title, author, and genre.*
- ***Search and Filter Books**: Users can search for available books based on title, author, genre, or availability and filter the search results accordingly.*
- ***Book Borrowing and Lending**: Users can borrow available books or lend their own books to others.*
- ***Transaction Management**: Track borrowed and lent books, including due dates and return status.*
- ***Book Recommendations**: Provide book recommendations based on user preferences, borrowing history, or popular books.*
- ***User Feedback and Ratings**: Allow users to rate books and leave feedback on their borrowing/lending experiences.*
- ***Notifications**: Send email or in-app notifications for important updates (e.g., book due dates, new listings, etc.).*
- ***Admin Management**: Admins can manage user accounts, book listings, and overall system settings.*

*These functions serve as the core operations of the **BookBridge** platform and will be expanded with additional features in future releases.*

2.3 Design and Implementation Constraints

*The design and implementation of the **BookBridge** platform must adhere to the following constraints, which will limit the options available to developers and influence the overall approach to building the system:*

- ***Software Design Methodology**: The **COMET (Component-Oriented Modeling and Evaluation Technique)** method will be used for software design. This methodology focuses on the decomposition of the system into components, allowing for better modularization and easier maintenance. All design elements, including major components and interactions, must be modeled using COMET.*
  - ***Reference**: COMET Method - [Link to Reference]*
- ***UML Modeling**: The system design will use **Unified Modeling Language (UML)** for visual modeling of the system architecture, components, and workflows. UML diagrams, such as Use Case Diagrams, Class Diagrams, and Sequence Diagrams, will be employed to document the design.*
  - ***Reference**: UML Modeling - [Link to Reference]*

- ***Frontend and Backend Technologies**:*
    - ***Frontend**: The frontend will be built using **Angular** for the user interface. React's capabilities will limit the use of other frontend frameworks, and specific design conventions for Angular applications must be followed (e.g., modular architecture, reusable components).*
    - ***Backend**: The backend will use **Node.js** and **Express.js** to handle API requests and business logic. The system will follow **RESTful API design** principles for communication between the frontend and backend.*
- ***Database**: **MongoDB** will be used as the database for storing book listings, user information, transactions, and other data. The NoSQL structure of MongoDB imposes a design constraint on how data will be stored and queried, limiting the use of relational database techniques.*
- ***Security Considerations**:*
    - ***User Authentication and Authorization**: The system will implement **JWT (JSON Web Tokens)** for secure user authentication.*
    - ***Data Encryption**: All sensitive user data (such as passwords) must be encrypted before being stored in the database using industry-standard encryption techniques (e.g., bcrypt).*
    - ***HTTPS Protocol**: The application must enforce HTTPS for all communications to ensure data privacy and security.*
- ***Communication Protocols**: All communication between the frontend and backend must be done via **HTTP/HTTPS** using **RESTful API**. Data will be transferred in **JSON** format.*
- ***Hardware and Performance**: The system must be optimized to run on typical web servers with standard memory and storage capacities. The expected traffic load is moderate, but scalability should be considered during the implementation.*
- ***Design Conventions**: The software must adhere to common programming standards and best practices for both frontend (e.g., component-based architecture) and backend (e.g., MVC pattern). Code documentation and comments should follow industry standards to ensure maintainability.*
- ***Cross-platform Compatibility**: The web application should be compatible with all modern browsers, including Chrome, Firefox, Safari, and Edge, with a focus on responsiveness for mobile devices.*

*These constraints shape the design and development of **BookBridge** and should be strictly followed to ensure system consistency, security, and scalability.*

2.4 Assumptions and Dependencies

*The following assumptions have been made during the design and development of the **BookBridge** platform. These factors are not confirmed but are expected to hold true for the successful completion of the project. If any of these assumptions are incorrect or change, it may significantly affect the design and implementation:*

- ***Availability of Required Technologies**: It is assumed that the necessary technologies, such as **React**, **Node.js**, **MongoDB**, and **Express.js**, will be available and stable*

throughout the development process. Any changes to these technologies or their support could impact the project.

- ***External Service Integrations****: The project assumes the availability of external services, such as email notifications and potential payment gateways, for user notifications and future monetization features. The integration with these services depends on their APIs being accessible and stable.*
- ***User Internet Access****: It is assumed that users will have reliable internet access to interact with the platform, as it is a web-based application. If internet access is limited or unreliable, the performance and usability of the system may be affected.*
- ***Data Consistency and Integrity****: The system assumes that data entered by users (e.g., book details, user information) will be accurate and consistent. Any discrepancies or errors in data input could result in incorrect listings, transactions, or user details.*
- ***Web Browser Compatibility****: The project assumes that users will primarily access the platform using modern web browsers (Chrome, Firefox, Safari, Edge). Compatibility issues with older browsers or unsupported browsers may arise if not properly addressed.*
- ***Scalability Needs****: It is assumed that the system will handle a moderate number of users at the outset, but it may need to be scaled in the future to accommodate more users and transactions. Adequate scalability provisions should be made during the development phase to ensure future expansion.*
- ***Security Compliance****: The platform assumes that data protection laws (such as GDPR, if applicable) will be adhered to, and the necessary security measures (e.g., encryption, token-based authentication) will be implemented.*

Dependencies:

- ***External Libraries and Frameworks****: The platform will depend on third-party libraries and frameworks, such as* ***Node.js****,* ***MongoDB****,* ***Express.js****, and potentially* ***React*** *(if used for specific features). The successful operation of the system depends on these external components being up-to-date and compatible.*
- ***Hosting and Server Infrastructure****: The application's deployment depends on a suitable hosting provider and server infrastructure. The server must be capable of supporting Node.js applications, handle user requests, and store data securely.*
- ***Payment Gateway Integration (Future)****: If a payment feature is introduced in the future, the project will be dependent on integrating with a third-party payment gateway like* ***Stripe*** *or* ***PayPal****, which must be accessible and compliant with relevant regulations.*

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

*The **BookBridge** platform will have a web-based user interface, accessible through modern web browsers. The interface is designed to be simple and intuitive for users, making it easy to register, list, borrow, and search for books. Below is a description of the main user interface for the application, which includes:*

User Interface Overview:

- ***Login/Registration Page**: The main entry point for users where they can either log in or create a new account. Users will be required to input their email and password for login, while registration will involve email, username, password, and user preferences.*
- ***Dashboard**: After logging in, users will land on the dashboard. This page will provide quick access to important features such as book search, book listings, and user profile management.*
- ***Book Listing Page**: Users can view and add books they wish to lend. Each book will be displayed with details such as the title, author, genre, and availability status. There will be options to edit or delete books as well.*
- ***Search and Filter Section**: A search bar allows users to search for books by title, author, or genre. Filters can be applied to refine results based on availability or category.*
- ***Book Borrowing Page**: When a user decides to borrow a book, they can view detailed information about the book and see the availability status. There will be a button to confirm borrowing.*
- ***Profile Page**: Users can view and update their personal information, including their name, contact details, and borrowing history.*
- ***Admin Interface (if applicable)**: An admin panel will allow administrators to manage user accounts, books, and other system settings.*

Interaction Mechanisms:

- ***Touchscreen Support**: For tablet or mobile users, the application will support touchscreen navigation, allowing users to tap on buttons, scroll through pages, and interact with menus directly.*
- ***Navigation Menus**: The main sections of the application (e.g., Dashboard, Listings, Profile) will be accessible through a top navigation bar or a sidebar menu.*
- ***Forms and Input Fields**: Users will interact with input fields for submitting information such as book details, user profile updates, and search criteria. Buttons and dropdowns will be used to submit or select options.*

3.1.2 Hardware Interfaces

*Since **BookBridge** is a web-based application, there are no direct hardware interfaces involved in the basic operation of the system. The software interacts with standard hardware components, such as:*

- ***User Devices (Desktops, Laptops, Tablets, and Smartphones)**: Users will access the **BookBridge** platform through a web browser on their personal devices. The platform will be compatible with devices running modern operating systems (e.g., Windows and Android) and web browsers (e.g., Chrome, Firefox, Safari, Edge).*
- ***Server Hardware**: The backend of the system will run on a web server, which will host the application, manage databases, and handle user requests. The server will need adequate CPU, memory, and storage to support the expected number of users.*
- ***Networking Hardware**: The system will depend on standard networking hardware (routers, switches, and modems) for communication between the client devices and the server. Internet connectivity is required for users to access the platform.*

Data and Control Interactions:

- ***Data Transmission**: The communication between user devices and the server will involve HTTP requests and responses, typically in the form of RESTful API calls. The interaction will involve sending and receiving data in JSON format (e.g., book details, user information, etc.).*
- ***Input Devices**: Users will interact with the system via input devices such as keyboards, mice, and touchscreens. These devices will be used to enter text (e.g., login credentials, book details) and navigate the application.*
- ***Output Devices**: Output will be displayed on monitors (for desktops, laptops, or tablets) or screens (for mobile devices). The content will include HTML pages rendered by the browser, showing the book listings, user profiles, and other system pages.*

3.1.3 Software Interfaces

*Currently, the **BookBridge** platform is a web-based application with no direct mobile app integration. However, future expansions may include a mobile app that interacts with the platform. Below are the key software interfaces:*

Web Application and Mobile App Interface:

- ***API Communication**: The web-based application will expose RESTful APIs that can be consumed by the future mobile app. These APIs will allow the mobile app to interact with the platform for functionalities such as:*
  - *User registration and authentication*
  - *Searching for books*
  - *Borrowing and returning books*
  - *Viewing user profiles and book listings*
- ***Data Format**: The data exchanged between the web application and the mobile app will primarily be in **JSON format** for consistency and ease of integration. API calls will be made to perform CRUD (Create, Read, Update, Delete) operations on user and book data.*
- ***Authentication**: The mobile app will use **OAuth2** or **JWT (JSON Web Tokens)** for secure authentication. Once a user logs in through the mobile app, a token will be generated and used to authorize subsequent API requests.*

Third-Party Software Integration (Future Considerations):

- ***Payment Gateway Integration****: If a payment feature is added to the platform, both the web application and the mobile app will integrate with a third-party payment service like* ***Stripe*** *or* ***PayPal*** *to facilitate financial transactions in future. This integration will require API connections with the payment gateway and secure handling of user payment data.*
- ***Push Notifications (For Mobile App)****: In the future, the mobile app may integrate with a push notification service (e.g.,* ***Firebase Cloud Messaging****) to send real-time updates and alerts to users (e.g., notifications about book availability, due dates, or new book listings).*

Mobile App (Future Expansion):

- ***Platform Support****: The mobile app will be designed for both* ***iOS*** *and* ***Android*** *platforms. It will interact with the backend web application via APIs and render the same core features as the web application, optimized for mobile devices.*

3.2 Functional Requirements

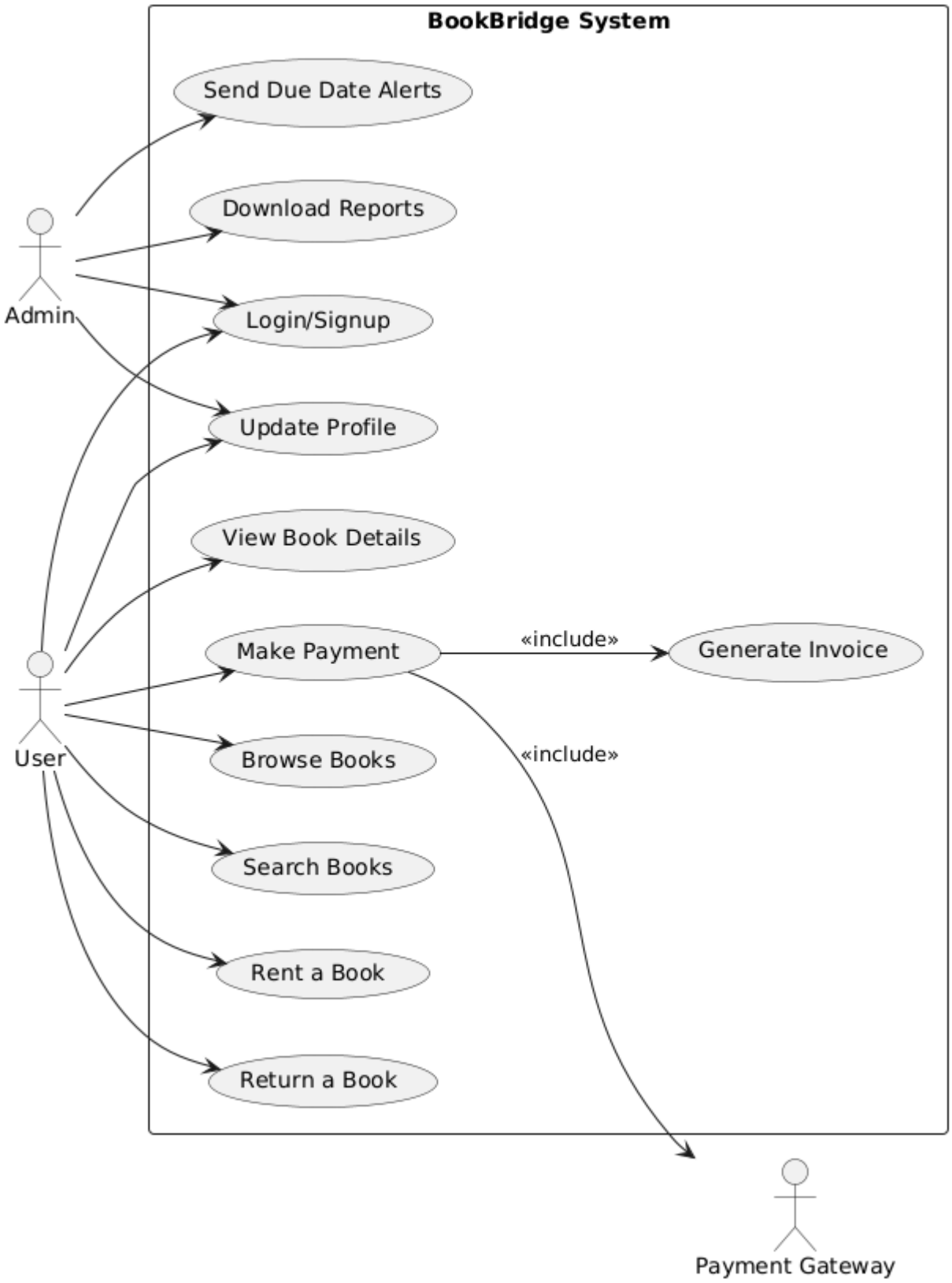
The ***BookBridge*** *platform provides a range of functionalities to enable efficient management of books, user accounts, and transactions. Below is a detailed list of the system's functional requirements:*

1. ***User Registration and Authentication***
    - *The system must allow users to register by providing their name, email, and password.*
    - *Users must be able to log in using their credentials and log out securely.*
    - *The system should support password recovery through email verification.*
2. ***User Roles and Permissions***
    - *The system must distinguish between different user roles:* ***Admin*** *and* ***Regular Users****.*
    - *Admins should have the ability to manage user accounts, books, and other platform settings.*
    - *Regular users should have access to view books, borrow books, and manage their profiles.*
3. ***Book Management (Admin Functionality)***
    - *Admins must be able to add, edit, and delete books in the catalog.*
    - *The system should allow admins to categorize books based on genre, author, and availability status.*
    - *Admins must be able to view reports on book usage and transactions.*
4. ***Book Search and Browsing (User Functionality)***
    - *Users must be able to search for books by title, author, genre, or ISBN.*
    - *The system should provide filters and sorting options (e.g., by availability, popularity, or date added).*
5. ***Borrowing and Returning Books***
    - *Users must be able to borrow books by selecting a book and confirming the transaction.*

- *The system should track the borrowing duration and send reminders for return deadlines.*
- *Users must be able to return books, and the system should update the book's availability status.*

6. ***Profile Management***
   - *Users must be able to view and update their personal information (e.g., name, email, password).*
   - *The system should display borrowing history and any overdue books in the user profile.*
7. ***Notifications and Alerts***
   - *The system must send email notifications for successful registration, book borrowing, return reminders, and overdue alerts.*
8. ***Reports and Analytics (Admin Functionality)***
   - *Admins must be able to generate reports on book inventory, borrowing trends, and overdue books.*
   - *The system should provide visual analytics (e.g., graphs and charts) for better insights.*
9. ***System Security***
   - *All user data must be stored securely, ensuring compliance with data protection standards.*
   - *The system must validate all inputs to prevent SQL injection, XSS, and other security vulnerabilities.*
10. ***Scalability and Responsiveness***
    - *The system must handle multiple users accessing the platform simultaneously without performance degradation.*
    - *The platform must be responsive and work seamlessly on desktops, tablets, and smartphones.*

3.3 Use Case Model



4.1.1 Use Case #1 (use case name and unique identifier – e.g. U1)

Author

- *Harish*

Purpose

- *The primary objective of this use case is to allow users (Admin and Users) to securely authenticate themselves into the system. It ensures only authorized access to BookBridge functionalities.*

Requirements Traceability

- ***FR1**: The system must provide secure login functionality.*
- ***FR2**: Users must be able to create new accounts.*
- ***FR3**: Both admin and users should access specific features post-login.*

Priority

- ***High**: Critical for system deployment as all functionalities depend on authenticated access.*

Preconditions

- *The user must have an internet connection.*
- *The user must already be registered (for login) or have the necessary details to register (for signup).*

Postconditions

- *The user is authenticated, and a session is created.*
- *The user is redirected to their respective dashboard (Admin/User).*

Actors

- ***User**: Individual accessing the system for book rentals and returns.*
- ***Admin**: Manages system operations and generates reports.*

Extends

- *None.*

Flow of Events

Basic Flow

1. *The actor opens the system and is presented with the login/signup screen.*
2. *For login:*
    - *The actor enters their username and password.*
    - *The system verifies credentials with the database.*
    - *Upon successful verification, the actor is redirected to their dashboard.*
3. *For signup:*
    - *The actor clicks "Sign Up."*
    - *The system prompts for necessary details (e.g., Name, Email, Password).*
    - *Upon successful registration, the actor is redirected to the login screen.*

Alternative Flow

- ***Forgot Password**:*
    1. *The actor clicks "Forgot Password."*
    2. *The system prompts for the actor's registered email.*
    3. *A password reset link is sent via email.*
    4. *The actor resets their password and logs in.*

Exceptions

- *Invalid username/password:*
    - *The system displays an error message: "Invalid Credentials. Please try again."*
- *Email already in use during signup:*
    - *The system displays an error message: "Email already exists. Try logging in."*
- *Server/database downtime:*
    - *The system displays an error: "Service unavailable. Please try again later."*

4.1.2 Use Case #2

Author

- *Harish*

Purpose

- *The purpose of this use case is to allow a **User** to rent a book from the **BookBridge** platform. The system should ensure that the book is available for rent, update the book's status, and record the rental details.*

Requirements Traceability

- ***FR4**: The system must allow users to rent books.*
- ***FR5**: The system must check if the book is available before renting it.*
- ***FR6**: The system must update the status of the book once rented.*
- ***FR7**: The system should maintain the rental history for each user.*

Priority

- ***High**: This is a core functionality of the system. If this does not work, users cannot rent books.*

Preconditions

- *The user must be logged into the system.*
- *The book to be rented must be available in the catalogue.*
- *The user must have sufficient account balance or a valid payment method if applicable.*

Postconditions

- *The book is marked as "rented" in the system.*
- *The rental details (e.g., user, book, rental duration) are stored in the database.*
- *The user's account is updated with the rental transaction.*

Actors

- ***User**: The actor who interacts with the system to rent books.*

Extends

- ***Make Payment (U8)**: If payment is required for the rental, this use case will extend to the "Make Payment" use case.*

Flow of Events

Basic Flow

1. *The user browses the available books or searches for a specific book.*
2. *The user selects a book they wish to rent.*
3. *The system checks if the book is available.*
    - *If the book is available, the system proceeds to the next step.*
    - *If the book is not available, the system notifies the user that the book is currently unavailable.*
4. *The user specifies the rental duration (e.g., 1 week, 1 month).*
5. *The system displays the rental fee based on the duration.*
6. *The user confirms the rental details and clicks "Rent Book."*
7. *If payment is required, the system redirects the user to the **Make Payment** use case.*
8. *Upon successful payment (if applicable), the system updates the book's status to "rented."*
9. *The system records the rental details (e.g., user, book, rental duration) in the database.*

Alternative Flow

- ***Book Already Rented**:*
    - *If the book has already been rented by another user, the system notifies the user with a message: "The book is currently unavailable for rent."*
    - *The user is given the option to check back later or rent a different book.*

Exceptions

- ***Payment Failure**:*

- *If the payment fails, the system displays an error message: "Payment could not be processed. Please try again."*
- *The book is not marked as rented, and the user is prompted to retry the payment.*

- ***System Error**:*
  - *If there's an issue with the database or system, the system displays: "An error occurred. Please try again later."*
  - *The user is advised to contact support if the issue persists.*

Includes

- ***Make Payment (U8)**: If applicable, the "Rent a Book" use case will include the "Make Payment" use case for processing payment.*

Notes/Issues

- *The system must ensure that rented books are not double-booked.*
- *The rental durations and pricing details should be configurable by the admin.*
- *Ensure that the rental process is simple and intuitive for the user.*

5 Other Non-functional Requirements

5.1 Performance Requirements

- ***P1. Book Search Response Time**: The system must return book search results within **2 seconds** after the user submits a search query. This ensures a responsive user experience while browsing or searching for books.*
- ***P2. Book Availability Check**: The system must check the availability of a book and return the status within **1 second** of the user's request. This quick check is critical for providing real-time availability to the user.*
- ***P3. Rental Confirmation**: The system must process and confirm the rental of a book within **3 seconds** after the user confirms the rental. This quick response is essential for maintaining a seamless user experience, especially when dealing with large book catalogs.*
- ***P4. Payment Processing Time**: If payment is required, the system should process and confirm the payment within **5 seconds**. Delays longer than this could result in user frustration, especially during transactions.*
- ***P5. Load Time for Book Details Page**: The detailed book page (with descriptions, reviews, and rental options) should load within **2 seconds** to ensure the user has a quick and smooth navigation experience.*
- ***P6. System Scalability**: The system must be able to handle up to **500 concurrent users** performing book rentals, searches, and payments without significant performance degradation. This is essential for maintaining a positive experience even during peak usage times.*
- ***P7. Rental History Access**: The system should allow users to retrieve their rental history within **2 seconds** of accessing the "My Rentals" page. Users expect fast access to their transaction history.*
- ***P8. System Uptime**: The system should have a **99.9% uptime** during peak periods (e.g., weekends or holidays), ensuring minimal downtime and maintaining availability for users.*
- ***P9. Response Time for Notifications**: The system must send rental confirmation or error notifications to users within **5 seconds** of completing the transaction.*

5.2 Safety and Security Requirements

- ***S1. Secure User Authentication****: All users must authenticate via a secure login system using* ***username and password****. Passwords must be stored securely using* ***hashing algorithms*** *(e.g., bcrypt) to protect user credentials from unauthorized access.*
- ***S2. Two-Factor Authentication (2FA)****: For high-value actions like* ***payment transactions*** *and* ***account modifications****, two-factor authentication (2FA) should be enabled. This can be done via an SMS or email code to ensure an added layer of security.*
- ***S3. Secure Payment Processing****: All transactions involving user payment data must be processed via a* ***PCI-DSS compliant payment gateway*** *(e.g., Stripe, PayPal). This ensures that payment details are securely transmitted and stored according to industry standards.*
- ***S4. Data Encryption****: All sensitive user data (including personal information, rental history, and payment details) must be encrypted using* ***AES-256 encryption*** *while stored in the database and during transmission over the network. This protects users' data from potential breaches.*
- ***S5. Session Timeout****: User sessions should automatically timeout after* ***15 minutes*** *of inactivity. This reduces the risk of unauthorized access to a user's account in case of an unattended device.*
- ***S6. Secure Mobile Connection****: The mobile app must use* ***HTTPS (SSL/TLS)*** *for secure communication between the app and the server. All API calls, particularly for book rentals and payments, should be protected to prevent* ***man-in-the-middle (MITM)*** *attacks.*
- ***S7. Protection Against SQL Injection and XSS Attacks****: The system must implement input validation and sanitization to prevent* ***SQL injection*** *and* ***Cross-Site Scripting (XSS)*** *attacks. All user inputs should be properly sanitized before being processed or stored in the database.*
- ***S8. Access Control and Role-Based Authorization****: The system must implement* ***role-based access control (RBAC)*** *to ensure that only authorized personnel (e.g., admins) can access sensitive features such as user data, rental history, and payment processing.*
- ***S9. Data Privacy Compliance (GDPR/CCPA)****: The system must comply with relevant data privacy regulations such as the* ***General Data Protection Regulation (GDPR)*** *for users in the EU and the* ***California Consumer Privacy Act (CCPA)*** *for users in California. Users must be able to request access to their data, request deletions, and control how their data is used.*
- ***S10. Secure User Data Backup****: The system should perform* ***encrypted backups*** *of user data on a regular basis (e.g., daily). These backups must be stored securely to ensure data recovery in case of system failure or data corruption.*
- ***S11. Prevent Fraudulent Activities****: The system must monitor for* ***suspicious activity*** *such as multiple failed login attempts, irregular rental patterns, or multiple accounts using the same payment details. Alerts should be sent to administrators for review, and actions such as locking accounts or requesting additional verification may be taken.*
- ***S12. User Education on Security Best Practices****: Users should be provided with educational material or pop-ups within the app, informing them about* ***secure password practices****, avoiding phishing attacks, and recognizing potential security risks.*

5.3 Software Quality Attributes

5.3.1 Reliability

- ***Requirement**: The system must operate without failure under normal operating conditions for **at least 99.9%** of the time.*
- ***Goal**: The system must be designed to handle expected traffic loads and user requests without crashing or encountering errors. This will be achieved through thorough testing, error handling mechanisms, and failover systems (e.g., redundant servers).*
- ***Verification**: Stress testing and uptime monitoring will be used to validate that the system meets the reliability standard.*

5.3.2 Maintainability

- ***Requirement**: The software should be maintainable, meaning that it can be easily updated, bug-fixed, and extended by the development team.*
- ***Goal**: Maintainability will be achieved through:*
  - ***Modular architecture**: Code should be broken down into smaller, reusable modules with clear boundaries and responsibilities.*
  - ***Code comments and documentation**: Functions and methods will be well-commented to clarify their purpose and behavior, making it easier for future developers to understand and modify the code.*
  - ***Version control**: All code will be tracked in a version control system (e.g., Git) to allow rollback and proper versioning.*
- ***Verification**: The maintainability of the system will be assessed by evaluating the ease of applying bug fixes and the complexity of adding new features, following the above principles.*

5.3.3 Usability

- ***Requirement**: The software must be user-friendly and allow users to achieve their goals efficiently without requiring extensive training.*
- ***Goal**: The system will be designed with the following principles to ensure usability:*
  - ***Clear navigation**: The app's user interface will have an intuitive design with easy-to-navigate menus and buttons.*
  - ***Responsive design**: The system must be compatible with multiple screen sizes and devices (smartphones, tablets, desktops).*
  - ***Accessibility**: The UI should meet accessibility standards, such as providing text alternatives for images and ensuring that the app is navigable via keyboard and screen readers.*
- ***Verification**: Usability will be validated through user testing with a representative group of customers, gathering feedback on their ease of interaction with the system.*

5.3.4 Adaptability

- ***Requirement**: The software must be designed with the flexibility to integrate with new sensors or systems in the future (e.g., different payment gateways, inventory management tools, or heating/cooling units).*
- ***Goal**: The system will be designed using:*
  - ***Plug-in architecture**: New sensors, gateways, and other components will be treated as plug-ins that can be added or removed without affecting the core system.*
  - ***APIs**: Integration with external systems will be done via well-documented APIs, allowing future systems to be plugged into the platform without changing the core logic.*
- ***Verification**: Adaptability will be verified by successfully integrating a new component or sensor into the system without requiring significant changes to the existing codebase.*

5.3.5 Performance

- ***Requirement**: The software should handle at least **500 concurrent users** without significant slowdowns, and all actions (e.g., searching for a book, completing a transaction) should take less than **2 seconds**.*
- ***Goal**: Performance will be optimized by:*
  - ***Efficient database queries**: Indexing and query optimization will be employed to reduce database load.*
  - ***Caching**: Frequently accessed data, such as book details or user profiles, will be cached to reduce latency.*
  - ***Load balancing**: The system will employ load balancing across multiple servers to distribute user requests evenly.*
- ***Verification**: Load testing and response time benchmarks will be used to ensure that performance requirements are met.*

6 Other Requirements

*This section outlines additional requirements that are not covered under the previous sections but are still essential to the **BookBridge** project.*

6.1 Database Requirements

- ***Requirement**: The system must use a **NoSQL** database (e.g., MongoDB) to store book details, user profiles, transactions, and other related data.*
  - *The database should be scalable, allowing for efficient storage and retrieval of data even as the number of books, users, and transactions grows.*
  - *The system must implement **data redundancy** and **backups** to ensure data availability and prevent data loss.*

6.2 Internationalization and Localization

- ***Requirement**: The system should support future localization for multiple languages.*
  - *The user interface must be easily adaptable to display content in different languages, including text direction (e.g., for right-to-left languages such as Arabic).*
  - *The application should provide language options for users to choose from, and this setting should be saved in the user profile.*

6.3 Legal and Compliance Requirements

- ***Requirement**: The system must comply with relevant data privacy and protection laws, such as the **General Data Protection Regulation (GDPR)** and **California Consumer Privacy Act (CCPA)**.*
  - ***Data storage**: All user data, such as personal details and transaction information, must be stored securely and must be encrypted both at rest and in transit.*
  - ***User rights**: Users must be able to request their data, correct inaccuracies, or delete their data at any time.*

6.4 Reuse Objectives

- ***Requirement**: The codebase must be designed with reusability in mind. The following components should be reusable:*
  - ***Authentication and authorization modules**: These modules should be generic enough to be reused across other projects or systems.*
  - ***Payment gateway integration**: The payment module should be designed in such a way that it can easily be extended to support other payment providers or services.*
  - ***Book catalogue management**: The component handling book management (e.g., adding, updating, and deleting books) should be reusable for future features or services that might integrate with other library systems.*

6.5 System Integration and Interoperability

- ***Requirement**: The system should be able to integrate seamlessly with third-party services such as payment gateways (e.g., Stripe, PayPal) and external APIs (e.g., book availability or shipping services).*
    - *Integration points should be standardized and well-documented to allow for smooth and efficient integration with future third-party systems.*

6.6 Sustainability

- ***Requirement**: The system should be designed with energy efficiency in mind, optimizing resource usage, particularly when deployed in cloud environments.*
    - *The system should support **auto-scaling** in the cloud to manage resources dynamically based on demand, ensuring that only the necessary resources are used.*

6.7 Accessibility

- ***Requirement**: The system must comply with the **Web Content Accessibility Guidelines (WCAG)** 2.1 at the **AA** level to ensure that it is usable by people with disabilities.*
    - *All interactive elements must be keyboard-navigable, and screen reader support must be included for visually impaired users.*
    - *Color contrast, font size, and other visual elements should be adjustable to accommodate users with visual impairments.*

Appendix A – Data Dictionary

1. Constants

Constant Name	Description	Value/Range	Related Operations/Requirements
MAX_BOOKS_PER_USER	Maximum number of books a user can borrow at any given time.	5	Limits borrowing capacity for users. Ensures fair usage.
LATE_FEE_RATE	The daily late fee applied to borrowed books if not returned on time.	5.0 rupees per day	Applied when a user returns a book after the due date.
BOOK_AVAILABILITY_TIMEOUT	Time duration for checking availability of a book in the system.	15 seconds	Ensures the system doesn't hang while checking availability.
MAX_PASSWORD_LENGTH	Maximum length for a user's password.	20 characters	Ensures security standards for user accounts.

Constant Name	Description	Value/Range	Related Operations/Requirements
MIN_PASSWORD_LENGTH	Minimum length for a user's password.	8 characters	Ensures users set a strong password.

Appendix B - Group Log

Meeting 1: Kickoff Meeting

- ***Date****: 01/16/2025*
- ***Attendees****: 4*
- ***Agenda****:*
  - *Discuss project requirements and objectives.*
  - *Assign initial tasks and responsibilities.*
  - *Define project scope and deliverables.*
- ***Activities****:*
  - *Brainstormed ideas for the project.*
  - *Assigned roles (e.g., frontend, backend, database).*
  - *Discussed the tools and technologies to be used (e.g., React, Node.js, MongoDB).*
- ***Decisions Made****:*
  - *We decided to use the COMET method for software design.*
  - *Agreed on a project timeline with deadlines.*
- ***Next Steps****:*
  - *Team members to start their individual tasks (e.g., setting up environments, gathering resources).*

Meeting 2: Progress Check-In

- ***Date****: 01/19/2025*
- ***Attendees****: 4*
- ***Agenda****:*
  - *Review the progress of the project.*
  - *Discuss any challenges or blockers.*
  - *Adjust timelines if necessary.*
- ***Activities****:*
  - *Reviewed the progress on individual tasks (frontend, backend, etc.).*
  - *Identified some integration issues between frontend and backend.*
- ***Decisions Made****:*
  - *Agreed on additional time to resolve integration issues.*
  - *Plan to have another meeting with the professor for clarification.*
- ***Next Steps****:*
  - *Resolve integration issues.*
  - *Review the codebase for potential optimizations.*

Meeting 3: Finalization and Testing

- ***Date****: 01/24/2025*
- ***Attendees****: 4*
- ***Agenda****:*

- *Finalize all project components.*
- *Prepare for testing and debugging.*
- *Plan for project submission and presentation.*

- ***Activities**:*
  - *Completed final testing of features (login, adding books, etc.).*
  - *Prepared the presentation for the project demo.*
- ***Decisions Made**:*
  - *Finalized codebase for deployment.*
  - *Created project presentation slides and assigned who will present which parts.*
- ***Next Steps**:*
  - *Submit the project and perform final review of the SRS document*