# Library Management System



Group B

**Table of Contents**

[**Library Management System 0**](#_buqxfj8ygqcg)

[**Chapter 1 Project Introduction 3**](#_kz9dvlhm8e92)

[**Chapter 2 System Capabilities, Enhancements, and Development Scope 4**](#_iccsui8fhojd)

[2.1 Current & existing systems 4](#_vtkz5dh4dajr)

[2.1.1 Features of the Current System 4](#_1uyzj1k1jylx)

[2.1.2 Book Management 4](#_3247kpd2zd1s)

[2.1.3 Library Management Features 4](#_7ujmmmvq6v99)

[2.1.4 Problems with the current system 5](#_rw27lsx7nl9m)

[2.2 Features of Existing Systems 5](#_kcmlhdqvqdvs)

[2.2.1 Problems with Existing Systems 5](#_nhep41od9hvj)

[2.3 Proposed enhancements & Upgrades 6](#_7de87onpn7b)

[2.3.1 Enhancements for the current system 6](#_7rpvcin6sv6b)

[2.3.2 Enhancements inspired by the existing system 6](#_knwr8q2nfczq)

[2.4 Scope 7](#_yoafgfogy7xk)

[2.4.1 Realistic Scope 7](#_k49jvpq2zh1q)

[2.4.2 Primary features: 7](#_6zx94utof9e6)

[2.4.3 Enhancement features: 7](#_g5noxpnm6ys0)

[2.4.4 Scalability 8](#_396b673c68ra)

[2.4.5 Security 8](#_kevynnfgtt4j)

[2.4.6 Intelligence 8](#_jknj3tjmiz1i)

[**Chapter 3 Requirements 10**](#_25vxcj6fxhex)

[3.1 Functional Requirements 10](#_122fkxvoylbo)

[3.2 Non-functional Requirements 12](#_1iv7ixcw4nvj)

[**Chapter 4 Development Methodology and Work Plan 13**](#_47fkr0d5qlii)

[4.1 What Method Are We Using? 13](#_p82pknbghc13)

[4.2 What are our work packages? 13](#_lnnr0okbaj6g)

[4.3 Prioritization of work packages for the first Increment 14](#_wqf4571tn8zx)

[4.3.1 Prioritization Approach 14](#_gdkdz9g7j4p8)

[4.3.2 Selected Work Packages for First Increment 14](#_cng6q7swc3rj)

[4.3.3 Conclusion 14](#_6z6s3hqcmkch)

[**Chapter 5 Team Skill Analysis and Role Allocation 15**](#_6qhmrqej565a)

[**Chapter 6 Work Pipeline and Collaboration Tools 16**](#_fw2umwxgklb7)

[6.1 Work Pipeline 16](#_dnemby73e208)

[6.2 Collaboration & Productivity Strategies 17](#_2zjoqcx0yotn)

[6.3.1 GitHub: Collaboration and Version Control 17](#_fpa0krfvkw8h)

[Figure1: Github Setup 18](#_ptsxq4mh1gx2)

[6.3.2 Docker: Environment Consistency and Easy Setup 18](#_wnjdry2yydts)

[Figure 2: socker setup and running 18](#_m9gpyhotfqbm)

[6.3.3 ESLint: Code Quality and Validation 18](#_scenjucpt55c)

[**Chapter 7 System Design & Security Considerations 20**](#_qslyb9pfwywh)

[7.1 Use Cases 20](#_dnvhu9cpbv1g)

[7.2 Threat Modelling 20](#_upuyue5nsurd)

[7.2.4 Conclusion 24](#_dh5vfs9b5o1u)

[7.3 Refined Use Cases 24](#_kjibsehuny77)

[7.4 System Architecture & Models 28](#_ibxtmjwg9fpr)

[7.6 Refined models and test cases 33](#_em4v936i4k2v)

[**Chapter 8 References 38**](#_ogq90ayp0m8l)

[‌ 38](#_re198zx2z4mr)

[**Chapter 9 Appendices 39**](#_9kvg3enqf00j)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Chapter 1 Project Introduction

Our project will be a library management system designed to provide users remote access to everyday library services. The system will enable students to perform tasks ranging from checking book availability to reserving library resources. It will also cater to library staff who need to manage borrowed books.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Chapter 2 System Capabilities, Enhancements, and Development Scope

##### **2.1 Current & existing systems**

This section will examine our current system and the University of Portsmouth Library Website, an existing system that offers library services to students and staff members. By analysing the features of both, we can gain insight into their strengths and limitations and develop an enhanced solution that offers enhanced functionality and more.

The current system is built using OOP(Object-Oriented Programming) and uses two main classes, the book class and the library class. These classes define the core functionalities of the system

###### **2.1.1 Features of the Current System**

###### **2.1.2 Book Management**

The book class is made to represent individual books in the library and provides the following features:

Initialization - The system starts with four default books that have been instantiated as objects of the book class.

Displaying Book Information - Book.display info method allows for info about the book to be displayed .

Book availability - BookgetStatus(), which returns whether a book is available or currently borrowed.

Returning a book - Book.returnBook() method returns the status of a book as returned

###### **2.1.3 Library Management Features**

The library class contains a collection of books held in an array and provides the following features:

Adding new books - Library.addBook() method allows users to add new books to the library’s collection

Borrowing books - Book.borrowBook(index) method allows the user to borrow books based on an index in the library collection.

Returning books - Book.returnbook(index) allows the users to return borrowed books.

###### **2.1.4 Problems with the current system**

* Some of the features, like the returning book method exist in both the library and book classes and they have unreported bugs as a result these features do not work.
* There is no way to handle input in the current system, making it impossible for users to interact with the system.
* There are no search or filter options which are vital for a library system
* The books are not in a database they only exist in the current code so we need a place to store them

##### **2.2 Features of Existing Systems**

* User Authentication System — The website has a login feature that allows university **students** to log into the library website.
* Device Availability Tracker — A real-time availability tracker which tells users the number of laptops and PCs that are available in the library, allowing them to plan their visits and ensuring efficient use of library resources
* Search and Advanced Search Features — A search feature that allows users to search for books and materials in the library’s catalogue. The advanced search feature enables more refined searches using filters, keywords, etc.
* Filtering options — Users can further refine their search based on factors like genre, subject, etc
* Book availability Status — Their website provides an availability feature simply displaying whether a book is available or not
* Account dashboard — The website has an account dashboard feature that displays useful information like loan history, reservations & charges.
* Reading List and Save Features — Users can save books to a reading list, allowing them to refer back to them for future reference
* Library Location system—The website has a location feature that uses a map to guide users to the physical location of a book in the library.
* Referencing & Blogging -The website provides useful information to assist students with blogging and referencing using indicators like ‘B’ & ‘R’

###### **2.2.1 Problems with Existing Systems**

**Navigation complexity**: The university library website can be a maze, with information and resources spread across many pages. Students may feel overwhelmed by the large amount of information available and find it difficult to navigate these systems effectively because web pages link to other web pages with different buttons and navigation styles.

**Access to resources**: The existing system fails to adhere to the consistency and standards heuristic. The search tab unexpectedly leads to a new page with a lot of information. While the system provides a map feature to locate physical books in the library, it is poorly designed and unclear.

**Lack of Personalisation:** There is a lack of personalisation. Despite the login feature, there is little to no visible customisation when a user is logged in. As a result, the login feels unnecessary. A better use of this would be to offer personalised access to library resources tailored specifically to an individual's study needs.

**Mapping Feature:** The current mapping feature provides a basic floor plan, which may be easier for some users to follow but could be challenging for others.

##### **2.3 Proposed enhancements & Upgrades**

###### **2.3.1 Enhancements for the current system**

Fix Return Book Functionality -

* Fix the issue where the returnBook() method conflicts between Book and Library classes.
* Fix the isBorrowed property updates correctly when a book is returned.

Implement a way to handle user inputs:

* Implement a simple way for users to interact with the system to make use of its current features this can be done via buttons, textboxes etc

Add Search and Filter Features:

* Implement a search function that allows users to find books based on the current and future attributes of the book or library( title, author, or genre)
* Add filtering options to refine searches based on what users need.

Store Books in a Database:

* Instead of having the books only exist when the code is run, store them in a database in a database (e.g., SQLite, MySQL, or sqlite3). This will make it easier to add and delete books without having to change the code as often.

###### **2.3.2 Enhancements inspired by the existing system**

User Authentication

* Implement a login system that will provide a distinct change from the basic page.
* Store user data in the database, allowing personalized interactions.
* It will provide users with a more personalized experience, as well as making it easier for library staff to manage library services based on users.

Account Dashboard:

* Create a dashboard displaying a user’s borrowed books, due dates, and borrowing history.
* Book Availability Tracker: Provide book status updates, showing which books are borrowed and which are available.

Reading List & Save Feature:

* Allow users to save books to a reading list for future reference

Improved Navigation & UI:

* Develop a user-friendly interface with clear navigation buttons.
* Have the most useful services displayed instead of being scattered
* Reduce confusion by ensuring the search results and other functions are presented clearly and concisely.

Mapping Feature Upgrade:

* Adding keys and a ‘you are here’ icon makes it easier for users to navigate
* Adding images from the library when a certain part of the map is pressed.

Personalized Recommendations:

* Introduce book recommendations based on what a user studies.

Device Availability Tracker

* If possible, include a feature that displays available laptops or other devices in real-time.

AI feature:

* If possible an AI feature which would be able to answer questions and point users to the service providing the most relevant solution to their needs

##### **2.4 Scope**

###### **2.4.1 Realistic Scope**

Our Library management system contains multiple features, but to scope realistically, we’ve segregated these core features into primary features and enhancement features.

###### **2.4.2 Primary features:**

* **Book Management** – This system covers tasks related to books such as acquisition, categorisation, and availability tracking.
* **User Management** – Creating and managing user accounts, handling user information, and providing authentication for library services
* **Circulation Management** – Automating the process of book check-in, check-out, and reservation to streamline circulation activities.
* **Search & Retrieval** – Implementing a robust search mechanism for users to quickly locate books, authors, and other library resources equipped with filters.
* **Security & Access Control** – Ensuring the security of sensitive data and implementing access controls to manage user privileges.

###### **2.4.3 Enhancement features:**

* **Visualisation and Dashboard Analytics** -Our system offers graphical representations of trends for reporting, such as bar charts and pie charts. These statistics could highlight insights like the most borrowed books within the same course or the most popular books of the week. This helps users discover relevant books more easily and encourages them to explore a broader range of reading options.
* **Library Usage Reports** - Our system will keep track of the number of active users at a given time for a specific day of the week.
* **Automated Report Generation -** Offering a feature such as a scheduled automated report sent to specific stakeholders and then also flagging unusual trends, such as an increase in overdue books.

###### **2.4.4 Scalability**

To ensure the system can handle growth in users, books, and transactions, we need to ensure our database is dynamic, allowing a growing number of users and/or books to be added to the system consistently:

* **Database Design** - Use a relational database (MySQL/PostgreSQL) with normalized tables for efficient book and user management.
* Prevent multiple users from booking the same book simultaneously using **transaction locking mechanisms**.
* **External Referencing** - Our system could also keep track of books that aren’t available in our library but can be found elsewhere. This would help students locate books in external libraries or partner institutions, saving them time and effort. For example, if a student searches for a book that isn’t in our collection, the system could suggest nearby libraries or online resources where they can find it.

###### **2.4.5 Security**

Security is crucial to protect user data, prevent unauthorized access, and secure transactions. Realistically, we can not implement some complex and costly methods and practices, so here are some Secure Software Development Practices we can implement:

* **Threat Modeling** *(To identify and solve potential threats)* - An example of us using this is identifying a threat such as unauthorised Access, and following the threat modeling methodology, we can highlight potential risks that can come from this being ‘Users trying to modify book loans or admin privileges’. We can reduce/eliminate this risk by introducing Role-Based Access, which can be implemented in PSQL using roles.
* **Password Protection and Standards**: To reinforce password safety and ensure confidentiality, enforce password regulations such as minimum length and character diversity. This prevents dangerous events such as brute force attacks. This will be simple to implement and with no cost, only requiring a fairly simple algorithm.

###### **2.4.6 Intelligence**

Integrating AI and automation will enhance system usability and efficiency, however, not all ideal features are realistically implementable. Here is a list of features that we can realistically implement, given our time frame, finances, and technical capability:

**Core Features**

* Automated Notifications
* Overdue Reminders: Automatically send alerts via email/SMS when books are overdue.
* Reservation Alerts: Notify users when a reserved book is available.

**Future Intelligence Enhancements**

* Implement machine learning-based search ranking to show relevant results first.
* Suggest books based on borrowing history (e.g., "Users who borrowed this also borrowed...").

**Analytics & Reporting**

* Graphical reports on popular books, peak borrowing hours, and overdue trends.
* Automated Report Generation for library admins on circulation trends and flagged unusual patterns.

# 

# 

# 

# 

# 

# Chapter 3 Requirements

##### **3.1 Functional Requirements**

| Priority | Requirement |
| --- | --- |
| High | **Book Management**: The system should cover tasks related to books, including:  **Book Acquisition** — The system will allow students to borrow books by linking them to their University ID for a designated loan period.  **Time Alerts** — If students go over their allotted time, the system will automatically contact them and alert them of this.  **Cataloguing** — Books will be categorised based on topic and content, enabling easier searchability and accessibility. And have unique IDs for easier processing.  **Availability** — The system will be able to keep track of book availability by keeping data of factors such as the number of books, book reservations, books overdue, lost/damaged, etc. |
| High | **User Management**: Creating and managing user accounts, handling user information, and providing authentication for library services.  **Accounts** – The system will store user information in a database, such as student ID, email, etc. This will be linked to any books or reservations the user makes. |
| High | **Circulation Management**: Automating the process of book check-in, check-out, and reservation to streamline circulation activities.  **Check in -** Users can return a book using their University ID. When a book is returned by a user, the system will update its status, marking it as “Available”. The system will notify users with reservations once the book is available.  **Check-out -** Users can borrow a book using their University ID. When a book is borrowed, the system will update its status, marking the book as “Checked out”. A due date will automatically be assigned, and the details will be sent to the user.  **Reservations-** Users can reserve books using their student ID and will be automatically notified when their reservations are available/ready to be collected. |
| Medium | **Search and retrieval**: Implementing a robust search mechanism for users to quickly locate books, authors, and other library resources.  **Enhanced Search** — Searching by filtering should allow users to look for resources based on multiple parameters, such as book title, author name, category, and availability. A basic search should match user queries with book titles, author names, or keywords while also enabling searches based on a book’s description or content. The search should return and display the book/s in a clear, structured format with the following information:  **Retrieval Data**: Book title, Author name, Publication Year, Availability of the book (show whether it is available, reserved, or checked out), Category, and also the physical location of the book in the library (shelf number/section name). |
| Low | **Reporting and analytics**: Generating reports on library usage, circulation trends, and popular genres to aid decision-making.  **Library Usage Reports** - Keeping track of the number of active users at a given time for a specific day of the week.  Providing details about peak hours and days of visits to the library  **Popular Genres and Books** - The system should be able to generate reports/information on the most and least borrowed genres of books  Displaying the latest trending book/genre for the day based on circulation reports  **Inventory, Reports, & Circulation** - The system should be able to monitor the total number of books in each genre.  Utilising data from the availability to look for patterns in stock/inventory & circulation  **Customizable reports** - Admins should be able to filter and customise reports based on various factors, e.g., date, genre, etc  **Visualisation and Dashboard Analytics** - The system should provide graphical representations of trends for reporting,e.g., bar charts, pie charts, etc  **Automated Report Generation -** Offering a feature such as a scheduled automated report sent to specific stakeholders  Flagging unusual trends, such as an increase in overdue books. |
| High | **Security and Access**: Ensuring the security of sensitive data and implementing access controls to manage user privileges.  All passwords will be hashed to ensure password protection in case of a database leak.  All users will have an associated role within the database to ensure they only access the information they are allowed to. |

##### **3.2 Non-functional Requirements**

| Priority | Requirement |
| --- | --- |
| High | **Usability**: Ensuring a user-friendly interface that promotes ease of navigation and a positive user experience for both library staff and patrons.  Making our system more usable by people with various disabilities, this may include increasing text size for people with visual issues, color coding buttons and text for the colour blind, etc. |
| Medium | **Scalability**: Designing the system to accommodate growth in the library's collection and user base over time. The system will utilise dynamic data structures to allow this growth. |
| Medium | **Performance**: Meeting performance standards to ensure timely response and efficient processing of library transactions. |
| High | **Reliability**: Building a reliable system that minimizes downtime and ensures the continuous availability of library services. |
| High | **Security**: Implementing strong security measures to safeguard against unauthorized access, data breaches, and other cyber threats. These precautions may include *two-factor authentication*, *regular password updates*, and other advanced security protocols. |

# 

# 

# 

# 

# 

# 

# Chapter 4 Development Methodology and Work Plan

##### **4.1 What Method Are We Using?**

We believe the Agile methodology (Scrum) is the best choice for our system. Since our Library Management System has multiple interdependent features with different priorities, Agile allows for iterative development, continuous feedback, and easy adaptation to changes.

##### **4.2 What are our work packages?**

**Work Packages**

* **Set up a version control system (e.g Git)**

Regarding the book management Requirement being the system should cover tasks related to book acquisition, cataloguing, and organization within the library.

Package 1: Basic Setup & User Management

* Set up the development environment, meaning we need to decide on the core technologies for the project. Examples are the language for the backend and the language we may use for the front end.
* Create user accounts with login/logout functionality.
* Implement role-based access (admin, staff, student). This can be achieved by roles within PSQL.

Package 2: Book & Circulation Management

* Add books to the system with categories and details. Categories would be things such as topic area, authors, etc.
* Implement book check-in/check-out, similar to the existing library check-in/check-out system.
* Enable book reservations and overdue alerts.

Package 3: Search & Security Features

* Add search functionality (by title, author, or a course subject category).
* Implement basic security (user authentication, data protection). Simply check the DB for existing users and then grant existing access and allow entrance based on this.

Package 4: Reports & UI Improvements

* These are low-priority requirements: Generate reports (popular books, user activity).
* Improve UI for easy navigation and usability.

Package 5: Testing & Deployment

* Test all features and fix bugs.
* Optimize performance.
* Deploy the system and gather feedback.

##### **4.3 Prioritization of work packages for the first Increment**

###### **4.3.1 Prioritization Approach**

The packages were prioritized based on the following:

* Core functionality
* Feasibility within a two-week sprint
* Logical dependencies.

###### **4.3.2 Selected Work Packages for First Increment**

1. Book & Circulation Management

Tasks **-** Categories for books, reserving/checking in books, and alerting users.  
 Reasoning **-** The book management features are essential for user interaction and ensuring that the system can handle fundamental library operations like reserving books.

1. User Management Tasks **-**  Creating input fields for account creation, buttons to validate fields, and authentication.  
    Reasoning **-**  Account creation and management are critical to system security and data integrity. They are prioritized after book & circulation management since basic functionality should come first.
2. Work Pipeline Setup Tasks **-** Role allocation, Planning, forming and reviewing work packages.  
    Reasoning **-** Setting up the work pipeline ensures efficient task management and clear development stages. It is important, but it’s placed last since it supports ongoing development rather than the core functionality of the system.

###### **4.3.3 Conclusion**

This approach ensures the development of core library features while maintaining a smooth and efficient workflow for the System's early stages.

# Chapter 5 Team Skill Analysis and Role Allocation

| Name | Role | Reason |
| --- | --- | --- |
| up2016662 | Programmer | More confident in programming than in business analysis. Prefers to implement solutions rather than planning them. |
| up942028 | Software Architect | Has experience and is comfortable with various roles within technology, and also has the soft skills required to maintain good communication with stakeholders. |
| up2063908 | Business Analyst | Good communication skills, acquired skills in complex problem-solving regarding time management and organisation. Experienced in finding trends and analysing data. |
| up2157114 | Programmer | Lots of years of programming experience, prefers to focus on their strengths rather than acquiring new skills. |
| up2119625 | Business Analyst | Acquired key desirable qualities which make them suitable for this role, able to predict or identify the strengths of the systems |
| up2111087 | Software Architect | Because of their background ( Software Engineering), they have developed the qualities needed to move on to software architecture. |

# 

# Chapter 6 Work Pipeline and Collaboration Tools

##### **6.1 Work Pipeline**

To enhance our Library Management System, we will follow a structured software development pipeline, ensuring scalability, security, and intelligence while maintaining collaboration, productivity, and real-time monitoring.

| Stage | Process | Tools |
| --- | --- | --- |
| Requirement Analysis | * Gather customer and stakeholder needs * Define system enhancements (scalability, security, AI-based recommendations, etc.) * Identify technical constraints | Jira Google Docs |
| System Design | * Define architecture (microservices, cloud-based, etc.) * Identify integrations (KYC, Decision Support Systems, etc.) * API Design | Lucidchart  Swagger / Postman |
| Development & Implementation | * Backend Development (Scalable DB, APIs) * Frontend Development (UI/UX improvements) * Implement security features (authentication, encryption) | GitHub VS Code Docker |
| Testing & QA | * Unit, Integration, and Security Testing * Load Testing (Big Data Readiness) * User Acceptance Testing |  |
| Deployment & Monitoring | * Deploy to cloud/on-prem servers * Monitor performance and security | AWS / Azure |
| Maintenance & Continuous Improvement | * Collect user feedback * Monitor system performance * Implement new features based on analytics | Google Analytics  Datadog |

##### **6.2 Collaboration & Productivity Strategies**

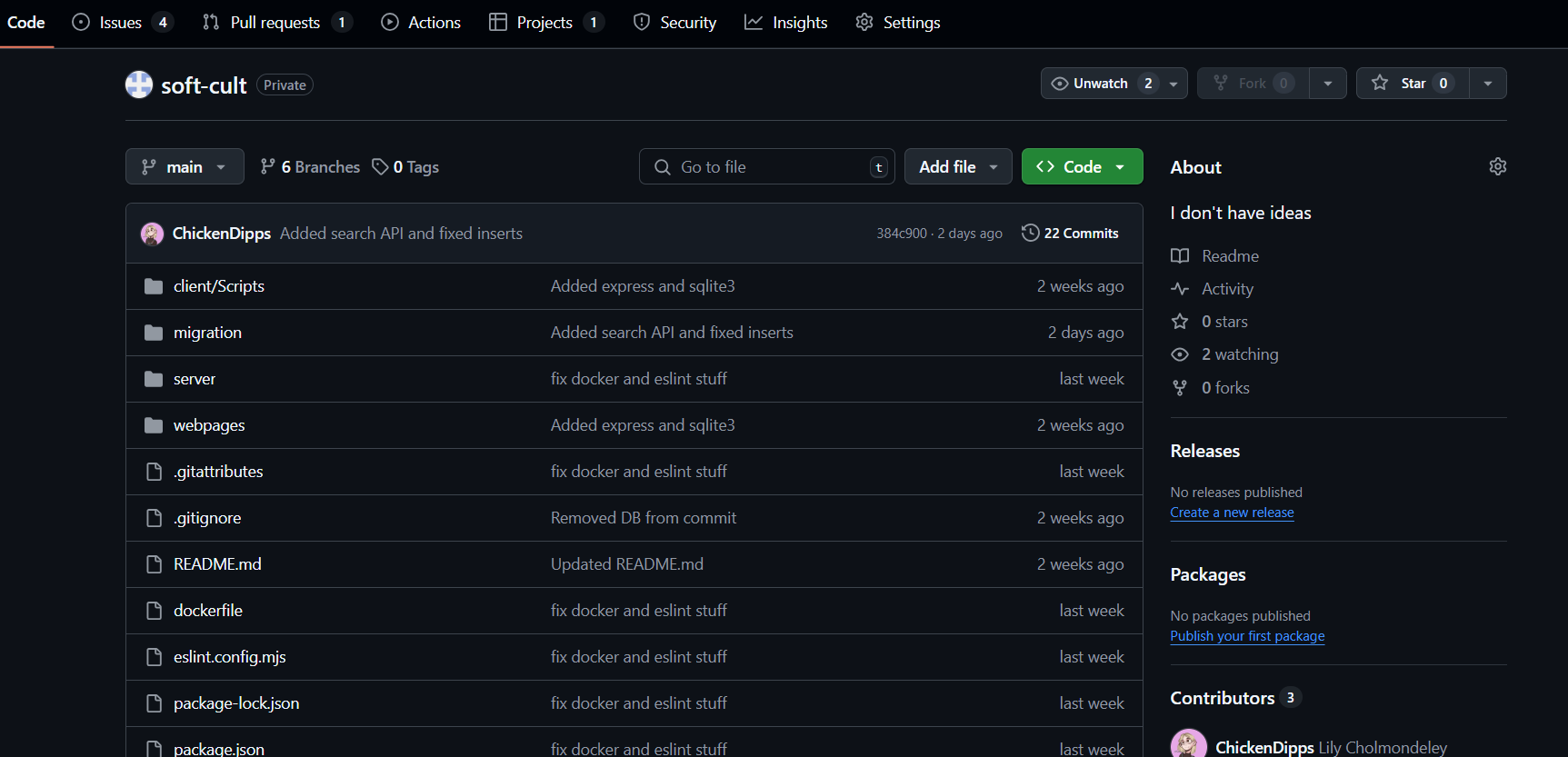
To ensure efficient teamwork, continuous delivery, and high-quality outcomes throughout the development of the Library Management System, we will adopt a combination of agile methodologies, modern development tools, and secure collaboration practices.

* **Agile & Scrum Methodology**  
   We will adopt Agile and Scrum practices to support iterative development and continuous improvement. This includes structured sprint planning, daily stand-up meetings for progress updates, and sprint retrospectives for post-sprint evaluation.  
   Tools: Jira, Discord
* **Version Control & Code Review**  
   All code contributions will be managed through a Git-based version control system, with dedicated feature branches and pull request (PR) workflows. Peer reviews will ensure code quality, consistency, and knowledge sharing among team members.  
   Tools: GitHub
* **Continuous Integration & Deployment (CI/CD)**  
   To streamline development and maintain code reliability, automated pipelines will be implemented for testing and deployment. These will ensure rapid integration of changes, early bug detection, and seamless deployment to test or production environments.  
   Tools: GitHub Actions, Jenkins, Docker
* **Security & Compliance Measures**  
   Security will be integrated throughout the development lifecycle. Measures such as data encryption, role-based access control, and vulnerability scanning will be implemented to ensure compliance with best practices and data protection standards.  
   Tools: OWASP ZAP, SonarQube, Snyk
* **Customer & Stakeholder Involvement**  
   End users and stakeholders will be actively involved through regular feedback loops. This includes scheduled demonstrations, surveys, and usability sessions to validate features, capture requirements, and improve the system based on real-world needs.  
   Tools: Google Forms, Surveys, Live Demos

**6.3 Setup Tools and suggest relevancy**

As part of the system development process for our Library Management System, we made use of several industry-standard tools to streamline development, improve collaboration, and maintain code quality. The tools we chose—GitHub, Docker, and ESLint—each served a specific purpose and contributed meaningfully to our workflow.

###### **6.3.1 GitHub: Collaboration and Version Control**

GitHub was used to host our codebase and manage collaboration across the team. It allowed us to work on different features at the same time using branches, without interfering with each other’s work. Through pull requests, we were able to review and merge changes in an organised and traceable way. We also linked our GitHub repository to a Kanban-style project board to help manage tasks and track progress. This setup kept the team aligned and reduced the chance of conflicts or duplicated work.  
  
https://github.com/54d1q/soft-cult

###### 

###### 

###### 

###### 

###### 

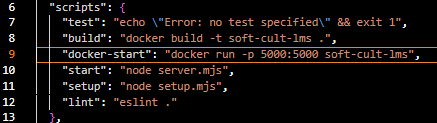
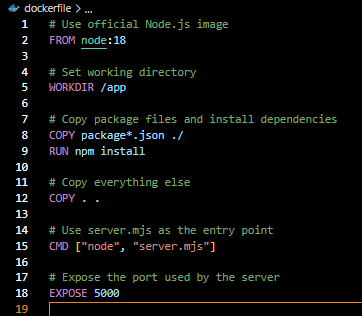
###### 

###### 

###### Figure1: Github Setup

###### 

###### **6.3.2 Docker: Environment Consistency and Easy Setup**

Docker was introduced to make it easier to set up and run the project on any machine, without worrying about local dependencies or operating system differences. By containerising the application, we ensured that it runs in the same environment for every team member and even in future deployment scenarios. This also makes the project easier to share with others, as they can spin it up quickly using a single command, without having to manually configure anything.

###### 

###### 

###### 

###### 

###### Figure 2: socker setup and running

###### 

###### **6.3.3 ESLint: Code Quality and Validation**

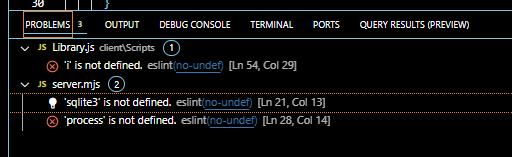
To help maintain a consistent coding style and catch issues early, we set up ESLint as a validation tool. This helped flag common errors and bad practices during development, especially in our backend code. It gave us instant feedback on potential bugs or formatting problems, allowing us to fix them before they became more serious. This has been useful not only for improving the quality of the codebase but also for keeping it readable and consistent across the team.

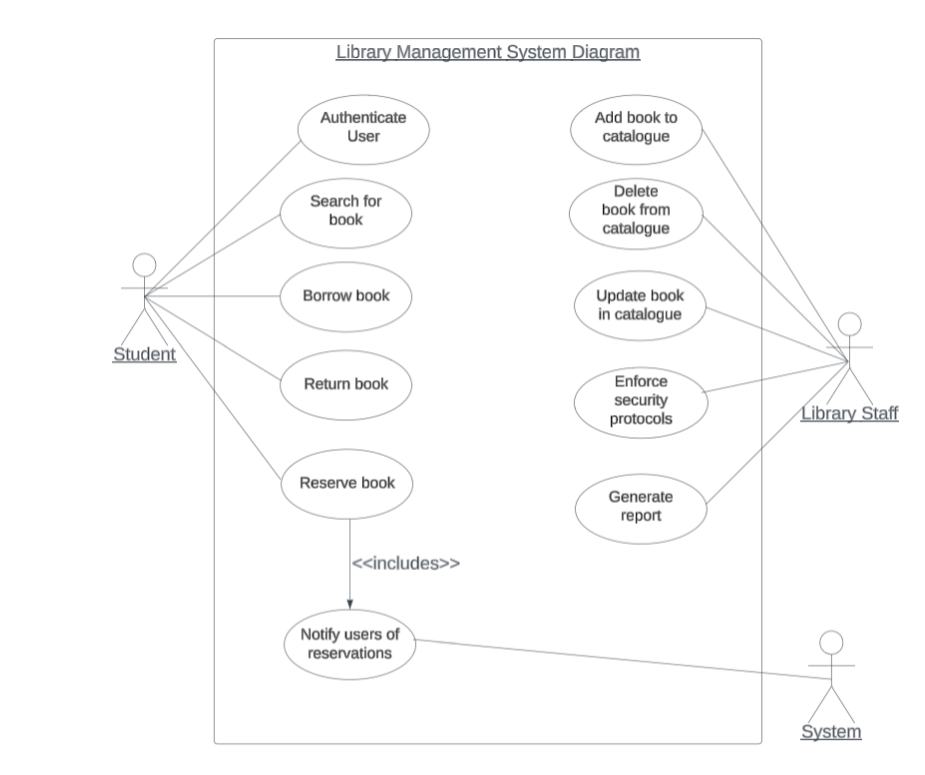
Figure 3: eslint showing problems

# 

# 

# Chapter 7 System Design & Security Considerations

##### **7.1 Use Cases**

  
Figure 4: use case diagram

This use case diagram showcases how users interact with the system.

*See the appendix for a full break down of use cases.*

##### **7.2 Threat Modelling**

**Assets:** Before identifying threats within the system, it is important to recognise all assets at risk. These assets vary in importance, but all require protection to maintain the system’s integrity:

**User Data**

* Student and staff records
* Login credentials
* Borrowing and reservation history
* Fines or penalties for overdue/lost books *(potentially card details)*

**Transaction Data**

* Loan history *(who borrowed what, due dates, overdue status)*
* Reservation and renewal records
* Book return and check-out logs

**Physical Hardware Assets**

* Library Computers → These are used by librarians and students to access the system.
* Library Servers → Store and manage the database and web applications.
* Networking Equipment → Routers, switches, and firewalls secure system acces

###### 

###### 

###### 

###### 

###### 

###### 

###### 

###### 

###### 

###### 

###### 

| **TITLE** | **THREAT** | **OUTCOMES** | **MITIGATION** |
| --- | --- | --- | --- |
| Weak Authentication & Weak Password Security | Using the **STRIDE** model, we identified spoofing as a potential risk. Weak authentication mechanisms or poor password security can allow a malicious user to impersonate legitimate users and gain unauthorised access to the system. Brute-force attacks, credential stuffing, and phishing attempts can compromise accounts, giving attackers access to sensitive library data. | Attackers can repeatedly attempt to guess login credentials, gaining access to student, librarian, or admin accounts. Once inside, they can view or alter sensitive information, such as personal details, borrowing histories, and system settings. Additionally, compromised librarian or admin accounts could lead to manipulated book records, fraudulent book loans, and unauthorised system modifications. In severe cases, an attacker might disable accounts or change security settings, escalating their control over the system. | To prevent unauthorised access, the system must enforce strict password policies, requiring a minimum of 12 characters, a mix of uppercase and lowercase letters, numbers, and special symbols. Implementing account lockout mechanisms after multiple failed login attempts can thwart brute-force attacks. Additionally, multi-factor authentication **(MFA)** should be mandatory for all librarian and administrative accounts, requiring a second verification step via email, SMS, or an authentication app. Enabling session timeouts and automatic logouts after inactivity further reduces the risk of being hijacked. Finally, users should be educated on recognising phishing attacks to prevent credential theft. |
| **SQL Injection Attacks** | SQL injection is a tampering risk where an attacker exploits input fields to execute malicious SQL queries that interact directly with the database. This attack compromises system integrity by allowing unauthorised database changes. | Attackers can manipulate book records, alter loan statuses, delete data, or insert fake data into the database. A successful SQL injection attack can also bypass authentication, allowing an attacker to log in as a higher role within the system without valid credentials. Attackers can gain access to confidential user information, such as hashed passwords, emails, and personal details, leading to identity theft and system-wide security breaches. | To prevent SQL injection, the system must use custom queries and statements, ensuring that user inputs are treated strictly as data, not executable code. Input validation should be enforced on all fields by simply limiting inputs to certain characters and lengths (e.g., only numbers for book IDs, only valid email formats for login). Also, database permissions should be restricted, regular users should only have read access, while write privileges are to be limited to authorised accounts. |
| **Insider Threats and Privileged Misuse** | Elevation of privilege is a major risk within an LMS, where users, particularly librarians, staff, or students, can gain unauthorised access to system privileged functions. This could happen through misuse of assigned privileges, credential theft, or unauthorised role escalation. | A librarian or staff member with privileged access may manipulate book records, extend borrowing periods, or override loan limits without authorisation. If an attacker steals admin credentials, they could gain access to higher-level controls, such as creating fake accounts, altering loan histories, or disabling security logs to cover their tracks. Additionally, internal staff with malicious intent could delete, modify, or leak sensitive user data, disrupting library operations. | To mitigate insider threats, the system must implement *role-based access control* ***(RBAC)***, where permissions are strictly assigned based on job roles. For example, students can only borrow books, librarians can update catalog records, and only senior admins can modify user accounts.  The database should log all privileged actions, ensuring that any unauthorised modification attempts can be traced back to specific users. This can be achieved by utilising role functions within PSQL in the database. Implementing multi-factor authentication **(MFA)** for high-privilege actions, such as modifying book loans or changing user roles, can prevent unauthorised privilege escalations. |

###### **7.2.4 Conclusion**

By identifying and mitigating these threats using **STRIDE**, we can strengthen the security of our Library Management System **(LMS).** Implementing authentication measures, preventing SQL injection, and restricting privileged access will significantly reduce the risk of cyberattacks and data breaches. Regular security assessments and system monitoring will further ensure that the LMS remains secure, reliable, and resilient against evolving threats.

##### **7.3 Refined Use Cases**

We’ve revised and updated the previously defined use cases to include appropriate security requirements. The main vulnerabilities identified such as weak password practices, SQL injection, and misuse of privileges have been addressed in this refinement through various industry-standard security mechanisms. These include Multi-Factor Authentication (MFA), Role-Based Access Below is a breakdown of the updated use cases reflecting these improvements.

1. Book Acquisition

| Attribute | Details |
| --- | --- |
| Use Case | Book Acquisition |
| Primary Actor | Student |
| Goal | Borrow a book securely |
| Preconditions | Student has no overdue books or fines |
| Trigger | Student initiates book borrowing |
| Enhancements | - Authentication: MFA enforced before borrowing  - Access Control: Only eligible users proceed  - Input Validation: Prevents SQL injection  - Audit Logging: Records transaction with user ID and timestamp |

2. Time Alerts

| **Attribute** | **Details** |
| --- | --- |
| Use Case | Time Alerts |
| Primary Actor | System |
| Goal | Notify students when books are overdue |
| Preconditions | A book's due date has passed |
| Trigger | System detects overdue status |
| Enhancements | - Secure Notification: Sent via verified channels (e.g., email)  - Escalation Handling: Unacknowledged alerts go to staff via RBAC  - Logging: All alerts and escalations are logged |

3. Cataloguing

| **Attribute** | **Details** |
| --- | --- |
| Use Case | Cataloguing |
| Primary Actor | Library Staff |
| Goal | Add or modify book entries securely |
| Preconditions | Staff is authenticated with correct permissions |
| Trigger | New book added or edited |
| Enhancements | - Access Restriction: RBAC enforced  - Input Validation: Prevents injection  - Duplicate Check: Prevents duplicate entries  - Logging: All changes logged with user identity |

4. User Management

| **Attribute** | **Details** |
| --- | --- |
| Use Case | User Management |
| Primary Actor | Librarian, Admin |
| Goal | Manage user accounts securely |
| Preconditions | Staff logged in via MFA |
| Trigger | Create, update, or delete a user |
| Enhancements | - Secure Authentication: MFA required  - RBAC Enforcement: Role-specific access  - Data Validation: Prevents injection  - Audit Trails: Tracks all changes made to accounts |

5. Return a Book

| **Attribute** | **Details** |
| --- | --- |
| Use Case | Return a Book |
| Primary Actor | Student |
| Goal | Secure return of a borrowed book |
| Preconditions | Student is logged in |
| Trigger | Book return action initiated |
| Enhancements | - Authentication Requirement: Only logged-in users can return  - Late Return Handling: Flags and notifies staff  - Logging: Logs return action with timestamp |

6. Search and Retrieval

| **Attribute** | **Details** |
| --- | --- |
| Use Case | Search and Retrieval |
| Primary Actor | Student |
| Goal | Search for available books securely |
| Preconditions | User is authenticated |
| Trigger | Search initiated |
| Enhancements | - Input Sanitisation: Prevents SQL injection  - Access Control: Authenticated access only  - Abuse Prevention: Throttling to limit spam/abuse |

7. Reporting and Analytics

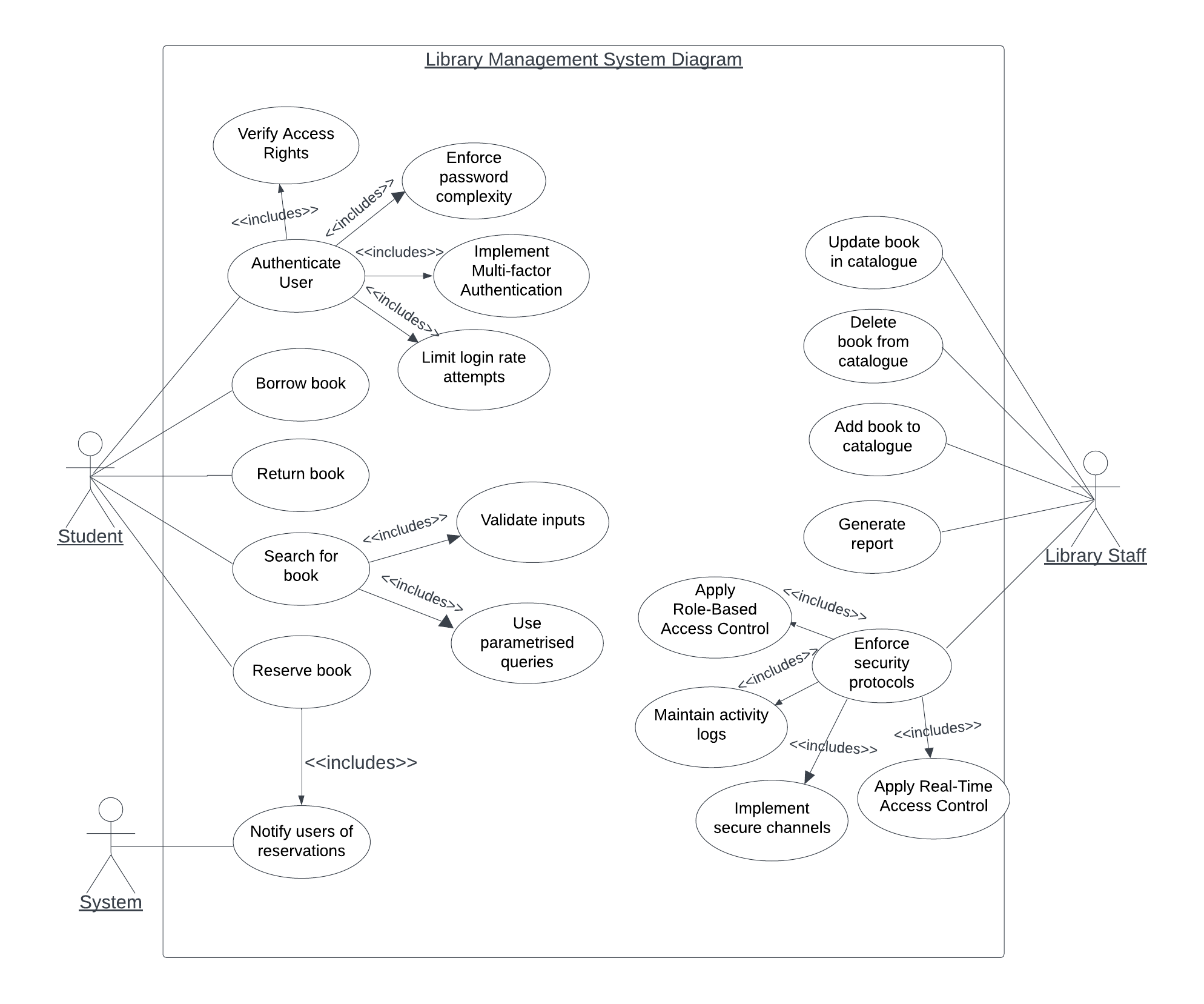
| **Attribute** | **Details** |
| --- | --- |
| Use Case | Reporting and Analytics |
| Primary Actor | Staff Member |
| Goal | Generate reports on usage, borrowing, and inventory |
| Preconditions | User has proper staff role |
| Trigger | Report generation request |
| Enhancements | - Access Control: Staff-only access via RBAC  - Data Privacy: Sensitive data masked or excluded  - Audit Logging: Logs all reporting activities |

8. Security and Access

| **Attribute** | **Details** |
| --- | --- |
| Use Case | Security and Access |
| Primary Actor | All Users |
| Goal | Enforce secure access system-wide |
| Preconditions | User attempts access to restricted service |
| Trigger | Login or privilege escalation attempt |
| Enhancements | - MFA Implementation: Required for sensitive actions  - Access Monitoring: Logs access attempts  - Access Revocation: Immediate admin controls  - Least Privilege Principle: Users get minimum necessary access |

**Summary of Security Controls Added**

| **Threat** | **Mission Strategy** |
| --- | --- |
| Weak Passwords | Enforce password complexity + MFA |
| Brute Force | Login rate limiting + lockouts |
| SQL Injection | Input validation + parameterized queries |
| Insider Threats | RBAC + logging + limited access |
| Data Breaches | Secure channels + least privilege principle |
| Unauthorized Access | Authentication & real-time access control |

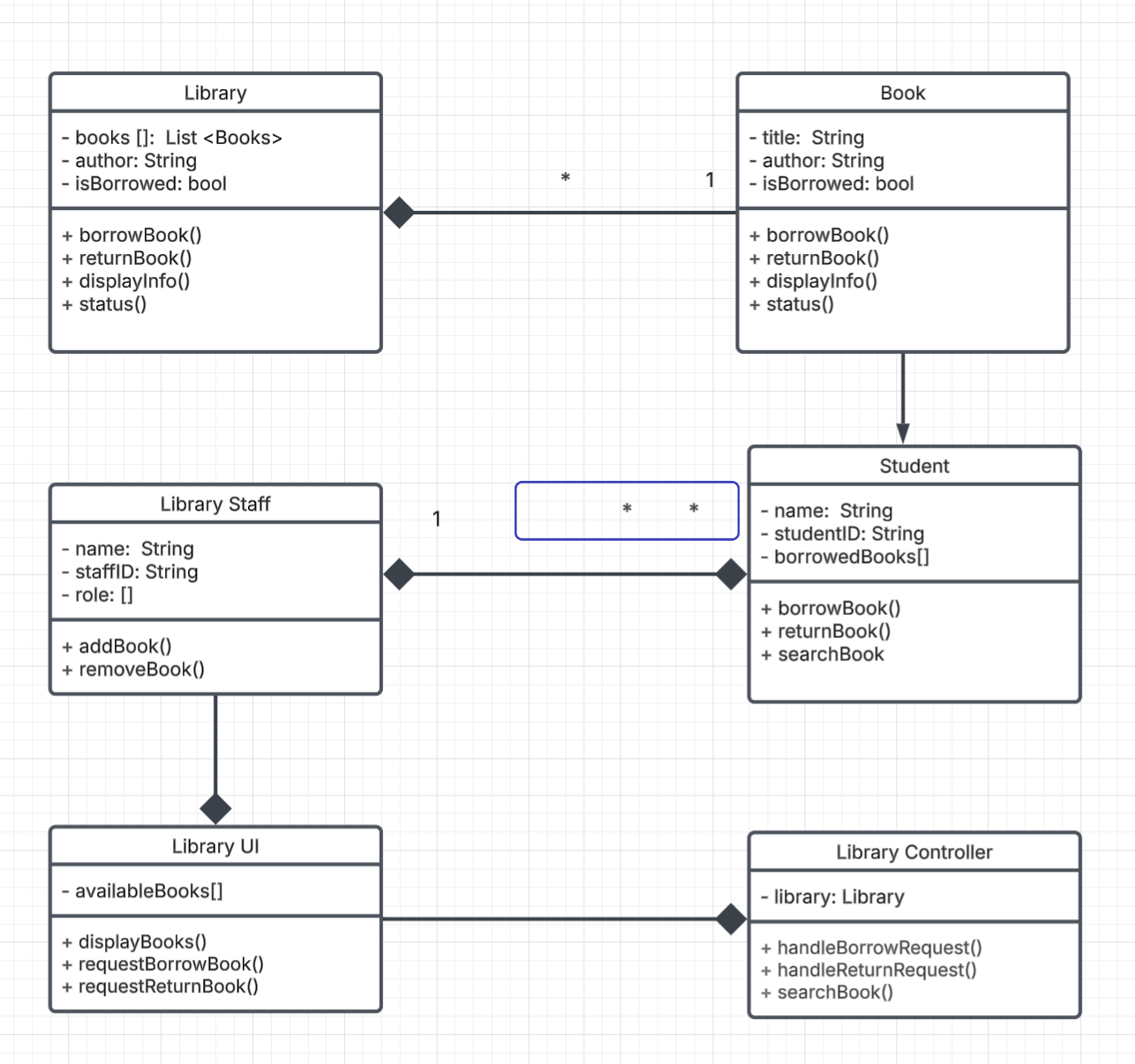
  
Figure 5: refined use case diagram

Taking these security requirements into account, our use case diagram was also refined to accommodate them.

##### **7.4 System Architecture & Models**

**Class Diagram**

The system’s class diagram model shows all the essential entities and their interactions within the system. It consists of all the systems main classes: Book, Library, Staff, Student etc.

  
Figure 6: The system’s class diagram model

**Component Diagram**

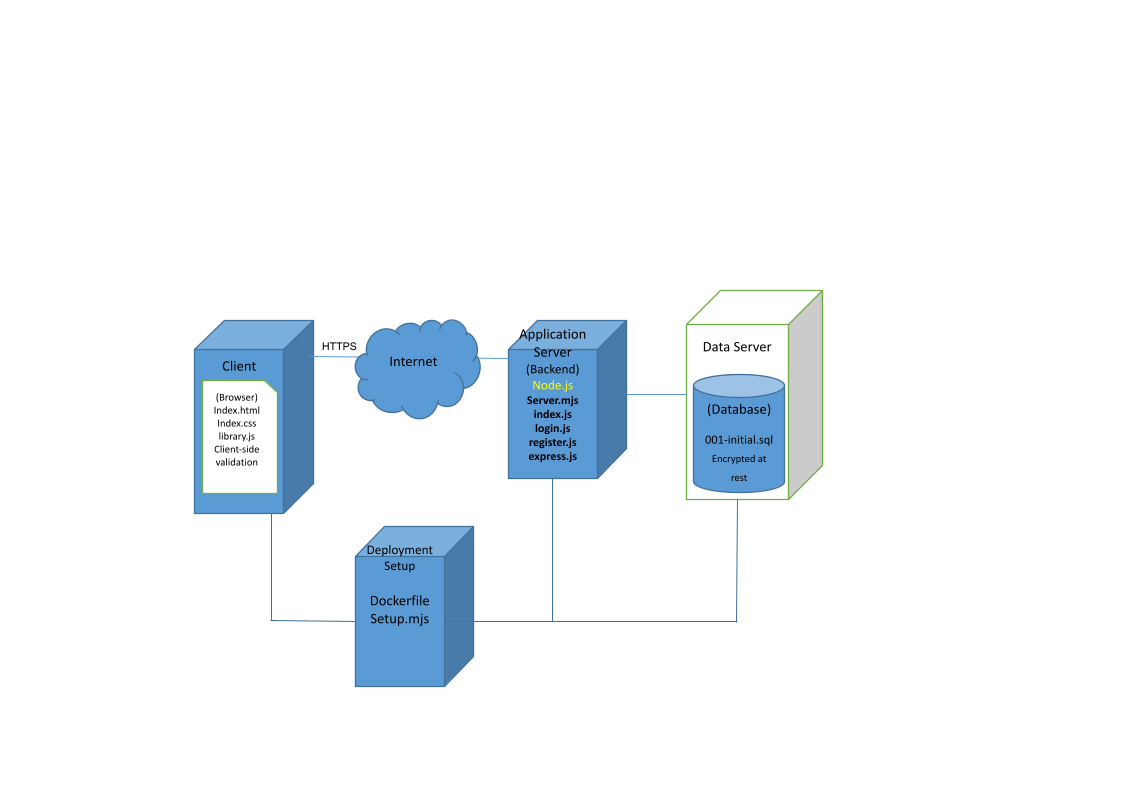
The component diagram shows all the components within our system including security modules, showing how different parts of the system interact and work together.



Figure 7: Component diagram

**Deployment Diagram**

This diagram shows how our system would be deployed whilst also considering various security measures.

  
Figure 8: component diagram

**7.5 Test cases**   
  
**Test Case 1: Verify secure login functionality**

| Attribute | Description |
| --- | --- |
| Purpose | Verify that users can securely log in, ensuring proper authentication. |
| Preconditions | Valid user credentials must be registered and available. |
| Test Steps | 1. Enter valid username and password, then log in.  2. Attempt login repeatedly using incorrect credentials. |
| Expected Results | - Successful login using correct credentials.  - User receives a notification of failed attempts; account locks temporarily after multiple unsuccessful attempts. |

**Test Case 2: Confirm proper book borrowing process**

| Attribute | Description |
| --- | --- |
| Purpose | Validate that the book borrowing process functions as expected. |
| Preconditions | User must be logged in, have no overdue books, and no outstanding fines. |
| Test Steps | 1. Search for the desired book.  2. Select the available book from search results.  3. Confirm the borrowing action. |
| Expected Results | - Book status is updated to 'unavailable'.  - User receives borrowing confirmation with the due date clearly specified. |

**Test Case 3: Handle borrow request with overdue books or fines**

| Attribute | Description |
| --- | --- |
| Purpose | Ensure that borrowing is blocked for users who have unresolved issues such as overdue books or unpaid fines. |
| Preconditions | User logged in with existing overdue books or unpaid fines. |
| Test Steps | 1. Attempt to borrow another book. |
| Expected Results | - Borrowing action is denied.  - Clear notification is presented to inform the user of overdue books or outstanding fines. |

**Test Case 4: Ensure book return updates correctly**

| Attribute | Description |
| --- | --- |
| Purpose | Confirm accurate system updates upon book return. |
| Preconditions | User logged in and currently has a borrowed book. |
| Test Steps | 1. Select the borrowed book from the user's borrowed list.  2. Confirm the book return action. |
| Expected Results | - The returned book status changes to 'available'.  - Notifications are sent to the user and staff if the return is late. |

**Test Case 5: Validate system security against SQL injection**

| Attribute | Description |
| --- | --- |
| Purpose | Verify the resilience of the system against SQL injection attacks. |
| Preconditions | User has access to input fields (such as login and search functions). |
| Test Steps | 1. Enter malicious SQL injection code into various input fields. |
| Expected Results | - The system properly sanitizes inputs.  - Security alerts are triggered immediately upon detecting malicious input. |

**Test Case 6: Test Role-Based Access Control (RBAC) to prevent unauthorized privileged actions**

| Attribute | Description |
| --- | --- |
| Purpose | Confirm RBAC restricts access to privileged functions based on user roles. |
| Preconditions | Staff member is logged in with restricted or limited permissions. |
| Test Steps | 1. Attempt to perform privileged actions beyond assigned permissions, such as modifying book statuses or adjusting due dates. |
| Expected Results | - Unauthorized actions are clearly denied.  - System administrator receives notifications of any attempted breaches. |

**Test Case 7: Verify accurate cataloging of new books**

| Attribute | Description |
| --- | --- |
| Purpose | Ensure books are cataloged correctly with accurate and complete information. |
| Preconditions | Staff member logged into the system. |
| Test Steps | 1. Navigate to the cataloging section.  2. Enter all required book details accurately.  3. Attempt to submit with incomplete information. |
| Expected Results | - Book successfully added when complete information is provided.  - Prompts appear clearly guiding completion of missing fields when details are incomplete. |

**Test Case 8: Confirm accurate report generation and retrieval**

| Attribute | Description |
| --- | --- |
| Purpose | Verify that the system generates accurate reports for analysis and management purposes. |
| Preconditions | Staff member is logged in. |
| Test Steps | 1. Access the report generation section.  2. Select appropriate filters and generate the report. |
| Expected Results | - Generated reports accurately reflect data based on selected filters.  - Reports are accessible, clear, and available for immediate viewing or download. |

##### **7.6 Refined models and test cases**

Refining Behavioural Models with Newly Considered Security and Other Concerns in Light of Desired Test Cases (TDD Focused)

Test Case 1: Verify Secure Login Functionality

| Category | Security & Authentication |
| --- | --- |
| Purpose | Prevents unauthorised access by enforcing authentication policies. |
| Preconditions | Valid user credentials must be registered and available. |
| Test Steps | 1. Enter valid username and password, then attempt login.  2. Enter incorrect credentials multiple times to test account lockout.  3. Attempt login from an unrecognised device and verify MFA (if applicable). |
| Expected Results | • Successful login using correct credentials.  • Failed login attempts trigger error messages.  • Account temporarily locked after repeated failures.  • MFA challenge prompts verification on new devices. |

Test Case 2: Confirm Proper Book Borrowing Process

| Category | Functionality & User Experience |
| --- | --- |
| Purpose | Ensures users can borrow books without issues. |
| Preconditions | User must be logged in, have no overdue books, and have no outstanding fines. |
| Test Steps | 1. Search for a book by title, author, or category.  2. Select an available book.  3. Confirm the borrowing action. |
| Expected Results | • Book status updates to 'unavailable'.  • User receives borrowing confirmation with the due date.  • System prevents simultaneous borrowing conflicts. |

Test Case 3: Handle Borrow Requests with Overdue Books or Fines

| Category | Business Logic & Access Control |
| --- | --- |
| Purpose | Reinforces library policies and prevents system misuse. |
| Preconditions | User is logged in with overdue books or unpaid fines. |
| Test Steps | 1. Attempt to borrow another book while having an overdue book or outstanding fine. |
| Expected Results | • Borrowing request is denied.  • A clear notification informs the user of their outstanding issues. |

Test Case 4: Ensure Book Return Updates Correctly

| Category | System Integrity & Accuracy |
| --- | --- |
| Purpose | Ensures system records remain accurate for book availability. |
| Preconditions | User is logged in and has at least one borrowed book. |
| Test Steps | 1. Select a borrowed book from the user’s account.  2. Confirm the book return action. |
| Expected Results | • The returned book status changes to 'available'.  • Notifications are sent to the user and staff if the return is late.  • System ensures accurate due date calculations. |

Test Case 5: Validate System Security Against SQL Injection

| Category | Security & Data Protection |
| --- | --- |
| Purpose | Ensures input fields properly handle malicious SQL queries. |
| Preconditions | User has access to input fields such as login and search functions. |
| Test Steps | 1. Enter common SQL injection strings into input fields.  2. Observe system response to the injected inputs. |
| Expected Results | • The system properly sanitizes inputs and prevents SQL injection.  • Security alerts are triggered upon detecting malicious input.  • Database queries execute safely without unintended modifications. |

Test Case 6: Test Role-Based Access Control (RBAC)

| Category | Security & Access Management |
| --- | --- |
| Purpose | Prevents users from performing unauthorised administrative actions. |
| Preconditions | Staff member is logged in with restricted permissions. |
| Test Steps | 1. Attempt to modify book statuses or adjust due dates without admin privileges. |
| Expected Results | • Unauthorized actions are blocked.  • System administrator receives notifications of attempted breaches.  • Logs capture failed privilege escalation attempts. |

Test Case 7: Verify Accurate Cataloging of New Books

| Category | Data Management & System Integrity |
| --- | --- |
| Purpose | Maintains a well-structured and searchable database. |
| Preconditions | Staff member is logged in. |
| Test Steps | 1. Navigate to the cataloging section.  2. Enter all required book details.  3. Attempt submission with missing or invalid information. |
| Expected Results | • Book successfully added when all required details are provided.  • Prompts guide users to complete missing fields.  • System prevents duplicate ISBN entries. |

Test Case 8: Confirm Accurate Report Generation and Retrieval

| Category | Reporting & Data Analysis |
| --- | --- |
| Purpose | Ensures management can access reliable reports for decision-making. |
| Preconditions | Staff member is logged in. |
| Test Steps | 1. Access the report generation section.  2. Select appropriate filters and generate a report. |
| Expected Results | • Reports accurately reflect filtered data.  • Reports are accessible and available for download.  • Data visualisation and summaries are clear and useful. |

#### 

#### 

#### 

#### 

#### 

#### 

# 

# 

# 

# 

# 

# 

# 

# 

# Chapter 8 References

geeksforgeeks. (2024, January 12). *Library Management System Project | Software Development*. GeeksforGeeks. <https://www.geeksforgeeks.org/library-management-system/>  
Refactoring Guru. (2014). *Design Patterns*. Refactoring.guru. <https://refactoring.guru/design-patterns>  
Agile and Scrum Methodology <https://asana.com/resources/agile-methodology>  
Docker. (2024). *Enterprise Application Container Platform | Docker*. Docker. https://www.docker.com/

ESLint. (2013). *ESLint - Pluggable JavaScript linter*. ESLint - Pluggable JavaScript Linter. <https://eslint.org/>  
GitHub. (2024). *GitHub Actions Documentation - GitHub Docs*. Docs.github.com. <https://docs.github.com/en/actions>  
GitHub. (2025). *GitHub*. GitHub. <https://github.com/>  
LucidChart. (2017). *Online Diagram Software & Visual Solution | Lucidchart*. Lucidchart.com. <https://www.lucidchart.com/>  
OWASP. (2020). *OWASP ZAP*. Owasp.org. <https://owasp.org/www-project-zap/>  
Postman. (2021). *Postman | The Collaboration Platform for API Development*. Postman. <https://www.postman.com/>

# 

#### **‌**

#### 

#### 

#### 

#### 

#### 

#### 

#### 

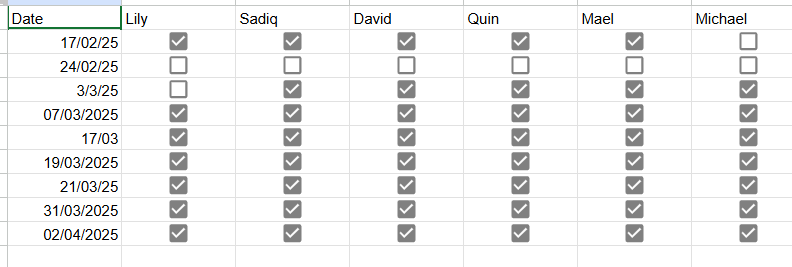
#### 

#### 

#### 

# Chapter 9 Appendices

* **Appendix A:** Meeting Logs



* **Appendix B:** Contribution Breakdown of Team Members

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Members ID** | UP2157114 | UP2119625 | UP2016662 | UP942028 | UP2063908 | UP2111087 |
| **Items** |  |  |  |  |  |  |
| **D01-T01** |  |  |  |  |  |  |
| **D01-T02** |  |  |  |  |  |  |
| **D01-T03** |  |  |  |  |  |  |
| **D02-T01** |  |  |  |  |  |  |
| **D02-T02** |  |  |  |  |  |  |
| **D02-T03** |  |  |  |  |  |  |
| **D02-T04** |  |  |  |  |  |  |
| **D02-T05** |  |  |  |  |  |  |
| **D02-T06** |  |  |  |  |  |  |
| **D03-T01** |  |  |  |  |  |  |
| **D03-T02** |  |  |  |  |  |  |
| **D03-T03** |  |  |  |  |  |  |
| **D03-T04** |  |  |  |  |  |  |
| **D03-T05** |  |  |  |  |  |  |
| **D03-T06** |  |  |  |  |  |  |
| Percentage Contribution | ***%16.6*** | ***%16.6*** | ***%16.6*** | ***%16.6*** | ***%16.6*** | ***%16.6*** |
| Signatures | O Michael  Majekodunmi | D.M. |  | Mael F. |  | Quinton I. |

* **Appendix C:** Additional Artifacts (if applicable)

**Title: Book Acquisition**

| **Primary Actor** | Student |
| --- | --- |
| **Goal/Description:** | Borrow a book from the library |
| **Preconditions:** | They are the student of University with a uni number or library card  Student musn’t have overdue books or fines |
| **Trigger:** | A student wants to borrow a book |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | 1. Login to the system 2. Search for a book they wish to borrow 3. Select the book they wish to borrow 4. Checks the student for preconditions 5. The system gives the user a due date 6. The user confirms the due date 7. The book status is changed from available to unavailable |
| **Alternate Flows/ Exceptions/ Extensions:** | *1.1- Login Failed.*  *1.2- Number of login tries expired.*  *3.1- The book is not available to borrow*  *4.1 - The student may have outstanding payments or overdue books* |
| **Exception Handling:** | 1.1.a- Notify the user of invalid credentials and ask them to input them again  2.a- request for change of schedule, call UC 3  4.1.a- Notify the user that they have to pay any outstanding fines or return overdue books in order to borrow |
| **Frequency of Use:** | Frequent |
| **Post-conditions:** | Student registers for desired courses.  Students clarify the courses they intend to study. |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Time alerts**

| **Primary Actor** | System |
| --- | --- |
| **Goal/Description:** | To notify students when their borrowed books are overdue |
| **Preconditions:** | The student has borrowed a book  If the due date of a book has passed |
| **Trigger:** | The system detects overdue books |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* The system checks all the borrowed books  *2-* The system identifies all the books past their due dates  *3-* Send notifications to students who have books past their due dates |
| **Alternate Flows/ Exceptions/ Extensions:** | *3.1- The student has already received a notification in the past about an overdue* |
| **Exception Handling:** | 3.1.a- Library staff are notified about the outstanding book being overdue despite previous notification |
| **Frequency of Use:** | Frequent |
| **Post-conditions:** | Student is notified about overdue book  Overdue book escalated to staff members if necessary |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Cataloguing**

| **Primary Actor** | Library Staff |
| --- | --- |
| **Goal/Description:** | To add a new book to the catalog |
| **Preconditions:** | Staff member has access  Necessary book details are available |
| **Trigger:** | A new book needs to be added |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* The staff member enters their login  *2-* They navigate to the book management section  *3-* The staff member enters the book details(Title,Author, Category, etc)  4- The staff member saves the book and adds it to the catalog |
| **Alternate Flows/ Exceptions/ Extensions:** | *1.1- Login Failed.*  *1.2- Number of login tries expired.*  *4.1 - The book already exists*  *4.2- Not enough details about the book has been entered* |
| **Exception Handling:** | 1.1.a- Notify the user of invalid credentials and ask them to input them again  4.1a - Notify the user the book already exists  4.2a- Ask the user to enter more details about the book |
| **Frequency of Use:** | Regular |
| **Post-conditions:** | The book is added |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: User Management**

| **Primary Actor** | Librarian, Student |
| --- | --- |
| **Goal/Description:** | Manage user accounts |
| **Preconditions:** | Student registered  Staff member has access |
| **Trigger:** | A new user needs to be added or updated |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* Login  *2-* They navigate to the user management section  *3-* Create or update the user with the correct details and access |
| **Alternate Flows/ Exceptions/ Extensions:** | *1.1- Login Failed.*  *1.2- Number of login tries expired.*  *3.1 - The user already exists*  *4.2- Not enough details about the user has been entered* |
| **Exception Handling:** | 1.1.a- Notify the user of invalid credentials and ask them to input them again  4.1a - Notify the staff member the user already exists  4.2a- Ask the user to enter more details and grant the correct access |
| **Frequency of Use:** | Regular |
| **Post-conditions:** | The account is created and the credentials are stored |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Return a Book**

| **Primary Actor** | Student |
| --- | --- |
| **Goal/Description:** | Return a book |
| **Preconditions:** | Student is a member of the uni with a library card or a uni number |
| **Trigger:** | Student wishes to return a book |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* Login  *2-* Select the book they wish to return  *3-* Confirm the return of the book  4- Update the status of the book from borrowed to available |
| **Alternate Flows/ Exceptions/ Extensions:** | *1.1- Login Failed.*  *1.2- Number of login tries expired.*  *2.1- The book is overdue* |
| **Exception Handling:** | 1.1.a- Notify the user of invalid credentials and ask them to input them again  2.1a - Notify the user and staff member of a late return |
| **Frequency of Use:** | Frequent |
| **Post-conditions:** | Book is returned and the status is correctly updated |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Search & Retrieval**

| **Primary Actor** | Student |
| --- | --- |
| **Goal/Description:** | To search for books |
| **Preconditions:** | Student has access to the system |
| **Trigger:** | Student needs to find a book |
| **Priority:** | Medium |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* Student enter the book they wish to find  *2-* Student applies relevant filters  *3-* The system finds the desired book or the books which are closely related to the desired book and displays them |
| **Alternate Flows/ Exceptions/ Extensions:** | *3.1- The book is not in the catalogue* |
| **Exception Handling:** | 3.1.a- Notify the student that there is no match for what they have typed/ searched for  3.1b - Give the user useful suggestions to resolve this issue e.g. check your spelling, check ebooks etc |
| **Frequency of Use:** | Frequent |
| **Post-conditions:** | Student retrieves book info |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Reporting and Analytics**

| **Primary Actor** | Staff member |
| --- | --- |
| **Goal/Description:** | To generate reports |
| **Preconditions:** | Staff member has access to the system |
| **Trigger:** | Reports need to be generated |
| **Priority:** | Low |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* Staff member logs in  *2-* Staff member selects the report feature  *3-* The relevant filters are applied  4- The report is generated. |
| **Alternate Flows/ Exceptions/ Extensions:** | *3.1- The book is not in the catalogue* |
| **Exception Handling:** | 3.1.a- Notify the student that there is no match for what they have typed/ searched for  3.1b - Give the user useful suggestions to resolve this issue e.g. check your spelling, check ebooks etc |
| **Frequency of Use:** | Regular |
| **Post-conditions:** | Reports are generated |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

**Title: Security and Access**

| **Primary Actor** | Staff member |
| --- | --- |
| **Goal/Description:** | To enforce security protocols |
| **Preconditions:** | Staff member has access to the system |
| **Trigger:** | User attempts restricted access |
| **Priority:** | High |
| **Normal Flow of Events/ Scenario/functionalities/Actions:** | *1-* Users attempts to access a restricted library service  *2-* The system verifies their details  *3-* The system grants them access |
| **Alternate Flows/ Exceptions/ Extensions:** | *3.1- Access is denied* |
| **Exception Handling:** | 3.1.a- Notify the user that access has been denied and ask them to refer to a staff member who can help them with the issue. |
| **Frequency of Use:** | Frequent |
| **Post-conditions:** | Secure access is maintained |
| **Secondary Actors:** |  |
| **Includes:** |  |
| **Special Requirements:** |  |
| **Assumptions:** |  |
| **Open Issues:** |  |
| **Notes :** | *Any other related notes.* |

Some of the first few work packages are related to organisation and planning

**Initial Team Work Package**

| Project | Library | Work Package ID/ WP ID | |  | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 17/03/2025 | |
| Increment ID |  | Date Modified | |  | |
| WP Title | **Initial Team Work Package** | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | During this phase, we had to formulate our teams and get to know each other | | | | |
| Requirements and Deliverables | * Formulate and meet team members * Assess each other’s strengths and weaknesses | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 | Meet the team | | Everyone | | 10/02/2025 |
| D002 | Join the team | | Everyone | | 10/02/2025 |
| D003 | Carry out team-building exercises | | Everyone | | 10/02/2025 |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 | Meet the team | | Completed | | |
| D002 | Join the team | | Completed | | |
| D003 | Carry out team-building exercises | | Completed | | |

Work Pipeline Package

| Project | Library | Work Package ID/ WP ID | |  | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 17/03/2025 | |
| Increment ID |  | Date Modified | |  | |
| WP Title | Work Pipeline Package | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | During this phase, we will formulate realistic work packages and suggest a suitable development methodology to accomplish them | | | | |
| Requirements and Deliverables | * Formulate realistic work packages * Look into and decide on a methodology | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 | Create and fill in work packages | | Everyone | | 19/03/2025 |
| D002 | Review and finalise work packages | | Everyone | | 19/03/2025 |
| D003 | Decide on a methodology | | Everyone | | 19/03/2025 |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 | Create and fill in work packages | | In Progress | | |
| D002 | Review and finalise work packages | | In Progress | | |
| D003 | Decide on a methodology | | Completed | | |

Tools Workpackage

# 

| Project | Library | Work Package ID/ WP ID | |  | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 17/03/2025 | |
| Increment ID |  | Date Modified | |  | |
| WP Title | Tools Work package | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Look at some tools and decide on them | | | | |
| Requirements and Deliverables | * Make use of tools * Look into and decide on a methodology | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 | Look into useful tools | | Everyone | | 19/03/2025 |
| D002 | Set up tools to use | | Everyone | | 19/03/2025 |
| D003 | Set up version control & development environment | | Everyone | | 19/03/2025 |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 | Look into useful tools | | Completed | | |
| D002 | Set up tools to use | | Completed | | |
| D003 | Set up version control & development environment | | Completed | | |

# 

| Project | Library | Work Package ID/ WP ID | | R2 | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 03/03/2025 | |
| Increment ID | R1-I1 | Date Modified | |  | |
| WP Title | User Management | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Creating and managing user accounts, handling user information, and providing authentication for library services. | | | | |
| Requirements and Deliverables | * Users should be able to create either admin or student accounts * Authentication of users | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 - T1 | Create input fields for the creation of an account | |  | |  |
| D001 - T2 | Add buttons to validate field inputs | |  | |  |
| D002 - T1 | Re-use username and password input fields for account authentication | |  | |  |
| D002 - T2 | Check usernames and passwords against the database. | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 | Account creation | | Not Started | | |
| D002 | Authentication of Users | | Not Started | | |

| Project | Library | Work Package ID/ WP ID | | R2 | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 03/03/2025 | |
| Increment ID | R2-I1 | Date Modified | |  | |
| WP Title | Book & Circulation Management | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Developing a means for users to check in,out and manage books | | | | |
| Requirements and Deliverables | * Establish clear categories and details of books * Implement a booking system where users can check in and out * Develop a reservation feature | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 - T1 | Create a list of categories for the user to choose from | |  | |  |
| D001 - T2 | Create a function which allows users to check in books | |  | |  |
| D002 - T1 | Re-use the function for checking in to create a function for checking out | |  | |  |
| D002 - T2 | Add buttons to verify the checking in and out of books | |  | |  |
|  | Create a function to notify the user of their check in or out | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 | Account creation | | Not Started | | |
| D002 | Authentication of Users | | Not Started | | |

| Project | Library | Work Package ID/ WP ID | | R2 | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 03/03/2025 | |
| Increment ID | R2-I1 | Date Modified | |  | |
| WP Title | Search & Security | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Developing a means for users to search in a secure environment | | | | |
| Requirements and Deliverables | * Add a search bar for students to be able to search from * Add an authentication feature | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 - T1 | Create an input box that will act as a search bar | |  | |  |
| D001 - T2 | Add a search or submit icon for the user to press on once they've finished searching | |  | |  |
| D001 - T3 | Create a function that takes the input and stores it as a variable to be used in a SQL query that uses LIKE to search for the given book | |  | |  |
| D002 - T1 | Create a function which checks users against the database and assigns them the correct access | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 - T1 | Input Box | | Not Started | | |
| D001 - T2 | Search bar | | Not Started | | |
| D001 - T3 | SQL Function | | Not Started | | |
| D002-T1 | Security | | Not Started | | |

| Project | Library | Work Package ID/ WP ID | | R2 | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 03/03/2025 | |
| Increment ID | R2-I1 | Date Modified | |  | |
| WP Title | Report & UI improvements | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Covers low-priority improvements to generate useful reports ) and refine the user interface for better usability and navigation | | | | |
| Requirements and Deliverables | Generate reports for:   * Most borrowed books * User activity logs * Improved UI elements (navigation bar, tooltips, cleaner layout) | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| D001 - T1 | Design a UI for the new navigation layout | |  | |  |
| D001 - T2 | Implement the new navigation layout in the front end | |  | |  |
| D001 - T3 | Add tooltips/help icons | |  | |  |
| D002 - T1 | Create a function to fetch the most borrowed books | |  | |  |
| D002-T2 | Create a function to fetch user activity | |  | |  |
| D002 -T3 | Display the reports in a user friendly dashboard | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| D001 - T1 | UIDesign | | Not Started | | |
| D001 - T2 | Front end UI | | Not Started | | |
| D001 - T3 | UI Help Features | | Not Started | | |
| D002-T1 | Popular Book Reports | | Not Started | | |
| D002-T2 | User Activity Report | | Not Started | | |
| D02-T3 | Displaying reports | | Not Started | | |

| Project | Library | Work Package ID/ WP ID | | R3 | |
| --- | --- | --- | --- | --- | --- |
| Manager |  | Date Created | | 17/03/2025 | |
| Increment ID | R5-I1 | Date Modified | |  | |
| WP Title | User Management | | | | |
| WP Type | E.G.   * Management * Implementation * Testing * Reusability and integration * Integration * QA * Deployment * Maintenance | | | | |
| WP Description | Performing functional testing, security testing, and debugging to ensure the Library Management System is free of errors and meets requirements. This includes validating user authentication, error handling, and UI/UX responsiveness. | | | | |
| Requirements and Deliverables | * Ensure all features work as expected under normal and extreme conditions * Identify and fix security vulnerabilities * Improve system stability and performance | | | | |
| Tasks definition and allocation | | | | | |
| Task ID | Task Description | | Delegated to | | Due Date |
| R5 - T1 | Conduct unit tests for user authentication | |  | |  |
| R5- T2 | Perform integration testing for database operations | |  | |  |
| R5- T3 | Debug and fix UI issues in account creation | |  | |  |
| R5- T4 | Conduct load testing for user login and authentication | |  | |  |
| R5 - T5 | Perform security testing (SQL Injection, Brute-force attack prevention) | |  | |  |
|  |  | |  | |  |
| Deliverable and Status | | | | | |
| Deliverable ID | Requirements | | Status | | |
| R5D001 | Functional testing complete | | Not Started | | |
| R5D002 | Security vulnerabilities identified and patched | | Not Started | | |
| R5D003 | Performance and load testing completed | |  | | |