CMP5387
Backend Web Services and Database Engineering

D2 Planning and Design Document

Group Members		
Student Name	**Student ID**	**Student Email**
Abraham Sharkey	S24142251	abraham.sharkey@mail.bcu.ac.uk
Rushai Blackwood	S23163023	rushai.blackwood@mail.bcu.ac.uk
Haashim Kabeer	S	
Malaika Noor	S24120358	malaika.noor3@mail.bcu.ac.uk
Jacob Hope	S	Jacob.hope@mail.bcu.ac.uk

Contents

Table of Figures

Table of Tables

Table of Code Snippets

Software Requirements

Summary of the target problem

Me and my group members are attempting a:

Project Idea type	Y/N (Select One)
Default Project Idea provided via Moodle	Y
Approved Project Idea Pitch	N

Table 1 Project type identification table

In most cases, local authorities have several library branches, each storing and lending books independently. This creates several challenges, such as:

Each library branch has its own book collection and lending. Several issues arise from this:

- No central real-time view of item availability across branches
- Manual or delayed coordination of reservations, loans, and inter-library transfers
- Poor tracking of borrowing histories and late fees between branches
- Limited ability for members to borrow or reserve items outside their home branch

To address these issues, the planned system will enable a central web service in the background. This will allow all the branches in a governing body to handle:

- Member information
- Items
- Loans
- Reservations
- Real-time access to item availability

This unified system will promote efficiency for library staff, improve usability for library members, and facilitate seamless resource-sharing across all libraries in the network.

Summary of the Default project idea

The default project idea describes a system for managing loans across multiple library branches under a local authority. Although the brief outlines the importance of a centralised loan system, further evaluation shows that the issue is more complicated than just keeping track of items. A multi-branch library environment introduces challenges such as synchronising inventories, ensuring reservation queues remain fair, and applying the same loan rules regardless of which branch a member enters. Staff also need to have access to accurate and up-to-date information so that they do not post duplicate copies of a book or create conflicting reservations.

Additionally, members expect extra freedom; they want to be able to borrow a book at one branch and be able to return it at another, check online availability, and reserve an item that is currently on loan elsewhere.

Older or branch-only systems often fail to meet these demands. A new modern centralised backend can streamline day-to-day operations, reduce manual contact between branches, and improve the reliability of the borrowing and reservation process.

A closer look at the default idea shows that the goal reaches past item tracking. The aim is to build a unified, scalable as well and trustworthy system that lets every branch see the same data in real time and that gives every user the same experience throughout the library network.

Summary of Project Idea Pitch

Since our group is using one of the default project ideas provided via Moodle, we did not submit a separate project idea pitch. Therefore, this section is not applicable.

Identification of target stakeholders and users

User/Stakeholder	Description	Problem(s)	Potential Benefit(s)
Library Staff	Librarians are responsible for managing items, loans, returns, and reservations.	P1: Difficulty checking real-time availability across branches. P2: Manual tracking of loan history and late fees.	Faster workflows, centralised data, fewer errors, improved efficiency.

Library Members	Individuals who borrow or reserve items from any library in the local authority.	P3: Unable to reserve items from other branches. P4: No real-time visibility of item status.	Cross-branch borrowing, better access, transparent reservation queues.
Local Authority Administrators	Oversee performance and usage data across the entire library network.	P5: Hard to analyse system-wide usage and stock distribution.	Network-wide reporting, insights into resource allocation.
IT / System Maintainers	Staff responsible for maintaining the backend, database, and infrastructure.	P6: Need a clean, maintainable, scalable architecture.	Easier maintenance, modular system, clear documentation.

Table 2 Stakeholder identification table

The identification of target stakeholders and users, as outlined in **Table 2**, highlights the key groups involved in the proposed multi-branch library management system and the problems they currently face. The two most important groups in this project are Library Staff and Library Members. They are the most affected by the operational problems and use the system the most. Library Staff are currently using different systems or performing manual work when checking availability, tracking loans, and managing reservations. This causes delays, errors, and added workload. A centralised, real-time system would not only allow the library staff to be more productive but also ensure that all branches provide a consistent level of service.

Library members also benefit significantly from this change. In many local areas, members can only borrow or reserve items from their home branch or depend on slow communication between branches. A unified system would allow members to check availability from any branch, make reservations for items that are currently loaned out, and borrow items from any location. This not only makes accessing resources easier for them but also increases their satisfaction.

Local authority administrators can see performance metrics for the whole system, while IT maintainers enjoy a structured, scalable, and secure backend. Although these groups are less involved in the day-to-day operations, the new system helps them support and manage the library network more effectively.

Stakeholder Analysis Matrix

Stakeholder	Role	Interest	Influence	Impact	Key Concern	Engagement Strategy
Library Staff	Operational users	High	High	High	Workflow disruption	Involve early, iterative feedback
Library Members	End users	High	Medium	High	Access & availability	Clear UI, system guidance
Local Authority	Oversight	Medium	High	Medium	Reporting accuracy	Periodic summaries
IT Maintainers	Technical support	Medium	Medium	Medium	Maintainability	Documentation & modular design

Figure 1 Stakeholder influence, impact, and engagement strategies for the proposed multi-branch library management system (Microsoft Excel, 2025).

This stakeholder analysis matrix, presented in **Figure 1**, looks at the main user groups involved in the proposed multi-branch library management system. Stakeholders have been assessed based on their interest, influence, impact on the system and their primary concerns, as well as the best engagement strategies to manage them. No stakeholders have been marked as low impact, as all identified groups play an important role in the functioning and the success of the proposed system. This analysis will support decision-making during system design and will ensure that the stakeholders' needs are addressed during development and implementation.

Power/Interest matrix

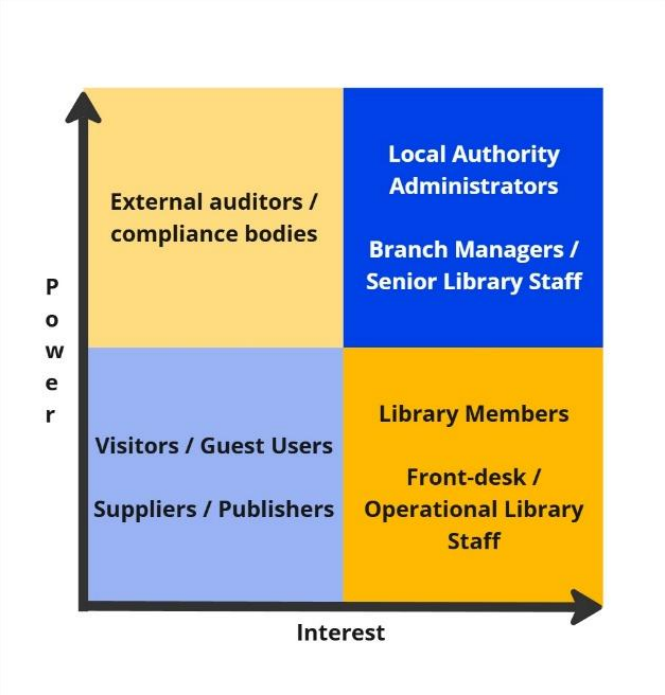


Figure 2 Power–Interest matrix showing stakeholder influence and engagement for the multi-branch library management system (Miro, 2024).

A Power/Interest matrix was used to evaluate stakeholder influence and engagement within the proposed system, as shown in **Figure 2**. Administrators of local authorities and senior staff of libraries were positioned in the high power, high interest quadrant because they can affect policy decisions and manage day-to-day operations across branches. Library members were classified as high interest with low power. They are the major users of facilities regarding borrowings, reservations, and searching. However, they have little power in decision-making. This analysis helped in prioritising stakeholders and project communication planning so that project efforts would be channelled to the ones most likely to impact system success.

This stakeholder analysis had consequences in the decisions related to feature prioritisation. This was true for the Must-Have features identification according to the MoSCoW framework presented in **Table 3**.

Salience model diagram

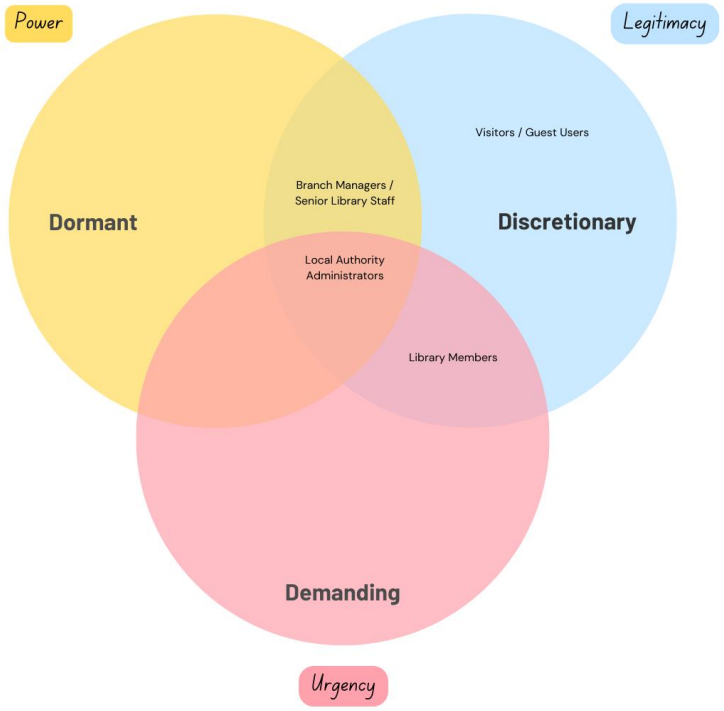


Figure 3 Stakeholder salience model showing stakeholder priority based on power, legitimacy, and urgency (created using Canva, 2024).

The stakeholder salience model assisted in prioritisation by considering power, legitimacy, and urgency as illustrated in **Figure 3**. At this stage, local authority administrators were classified as definitive stakeholders because they have the power to make decisions, they have formal accountability and have time-sensitive interests in the performance of the system. For their operational authority, branch managers were classified as dominant stakeholders because of their valid and influential position in the library network. Library users were classified as dependent stakeholders because

they have legitimacy and urgency in their needs, but have limited power. Visitors were discretionary stakeholders as they have legitimate but low-priority influence. This model helped to clarify the prioritisation of stakeholders and directed the focus of engagement.

Identification of feature set

The system's features directly address the main issue that multi-branch library networks face: the absence of a unified system for managing members, items, loans, reservations, and real-time availability across all branches. Each feature addresses a specific need and solves a problem which has been identified during the analysis of issues and stakeholders.

To prioritise development, the features were categorised using the MoSCoW method. This method organises requirements into four groups: Must Have, Should Have, Could Have, and Won't Have. This ensures that the focus is first placed on the foundational system needs before moving on to additional system enhancements (**Clegg and Barker, 1994**).

ID	Feature	Description	Problem it helps solve	Users/Stakeholders involved	Importance
F1	Management	Allows staff to add, update, and remove items across branches.	P1: No centralised visibility of inventory.	Library Staff	Must Have
F2	Member Management	Register, update and track library members.	P2: Difficulty maintaining consistent member records.	Library Staff, Admin	Must Have
F3	Loan Creation & Tracking	Issue loans, track due dates, return dates, and borrowing history.	P2: Inefficient tracking of borrowing histories.	Staff, Members	Must Have
F4	Reservation System	Allows users to reserve items that are currently on loan.	P3: Limited ability to reserve outside home branch.	Members, Staff	Must Have
F5	Real-Time Item Availability	Shows item availability across all branches instantly.	P1, P4: Missing real-time availability.	Staff, Members	Must Have
F6	Late Fee Calculation	Automatically calculates fees for overdue returns.	P2: Manual calculation of fees is error-prone.	Staff, Members	Should Have

F7	Member Summary View	Displays loans, reservations, and outstanding fees.	P2: Hard to view full member status.	Staff	Should Have
F8	Search Functionality	Search books by title, author, ISBN, or branch.	P4: Hard to find resources across branches.	Members, Staff	Could Have

Table 3 Feature identification table

Feature Discussion and Justification

The Must Have features described in **Table 3** (**F1–F5**) cover the most important functions necessary for resolving the key issues in a multi-branch library network. With Item Management (**F1**) and Member Management (**F2**) functions, library employees can maintain accurate information across all branches. Functions such as Loan Creation & Tracking (F3) and the Reservation System (**F4**) enable important lending activities, ensuring borrowing and reservations are simple for library members. Real-Time Item Availability (F5) provides real-time visibility of stock levels across all branches in the network.

Should Have Features (**F6-F7**), such as Late Fee Calculation and Member Summary View, will help in increasing efficiency by automating many tasks. While they will make for a better interface, they are not necessary for a simplified model to work.

The Could Have feature (**F8**), Search Functionality, adds functionality by making it simpler for customers to access materials in other branches. However, they are not critical for the system to function at a basic level.

By using the MoSCoW prioritisation method, it is possible for the project team to focus on fulfilling the most important task requirements simultaneously while maintaining an element of flexibility in incorporating additional improvements in the future (**Clegg and Barker, 1994**). These necessary features directly influenced the design of the data model and the API operations needed to support lending across multiple branches.

Responsibility Assignment Matrix (RACI)

The Responsibility Assignment Matrix (RACI) outlines how responsibilities for each feature of the Library Management System may be divided among the team. The RAM below provides a clear structure for role distribution while remaining flexible, allowing adjustments as the project progresses to support collaboration and balance workload.

Feature ID	Feature Name	Responsible (R) 🛠	Accountable (A) ⏳	Consulted (C) 💬	Informed (I) 📢
F1	Item Management	Abraham, Hashim	Abraham	Jacob, Rushai	Malaika
F2	Member Management	Abraham	Abraham, Rushai	Jacob, Hashim	Malaika
F3	Loan Creation & Tracking	Hashim, Rushai	Jacob	Abraham	Hashim, Malaika
F4	Reservation System	Rushai	Hashim	Abraham, Jacob	Malaika
F5	Real-Time Item Availability	Hashim	Abraham	Jacob	Rushai
F6	Late Fee Calculation	Jacob	Abraham	Rushai, Hashim	Malaika
F7	Member Summary View	Malaika	Jacob, Malaika	Abraham, Hashim	Rushai
F8	Search Functionality	Jacob, Rushai	Abraham	Hashim, Abraham	Malaika

Table 4 Responsibility Assignment Matrix

Explanation

Table 4 presents the Responsibility Assignment Matrix based on the RACI model, which *defines responsibility, accountability, consultation, and communication for each system feature, following **PMI (2017)** guidance. The matrix provides clarity while remaining flexible, allowing responsibilities to adapt as development progresses. Role assignments reflect each member's demonstrated strengths across planning, design, and implementation, supporting balanced workload distribution and effective collaboration.*

MoSCoW Prioritisation Overview

Must Have

- F1 Item Management
- F2 Member Management
- F3 Loan Creation & Tracking
- F4 Reservation System
- F5 Real-Time Availability

These features resolve the primary challenges and form the system's core functionality.

Should Have

- F6 Late Fee Calculation
- F7 Member Summary View

These improve efficiency but are not fundamental to the MVP.

Could Have

- F8 Search Functionality

Adds convenience and a better user experience.

Won't Have (for now)

- Mobile app
- Analytics dashboard
- Automated interlibrary transfer workflows

Must Have feature prioritisation was directly impacted by the analysis of the stakeholder power–interest matrix (see **Figure 2**). High-power stakeholders, such as library employees and administration, impacted management and loan core functions. On the other hand, high-power stakeholders, such as members, played a major role in shaping the availability and reservation features of the system.

Planning and Design

Work Division and timeline

The planning documents outline the intended division of work and scheduled responsibilities, as summarised in **Table 5**. Like most project plans, individual input is likely to change over the course of development.

Work Division Overview

Project Area	Description	Lead	Contributor
Backend API Development	Design and implementation of RESTful API endpoints for items, members, loans, and reservations. Includes business logic, validation, error handling, and integration with the database layer.	**Abraham**	All members
Database Design and Implementation	Development of ER diagrams, data normalisation, SQL schema creation, constraints, relationships, and preparation of sample datasets and queries.	**Rushai**	All members
Client-Side Prototype	Construction of a simple functional HTML, CSS, and JavaScript prototype, including UI wireframes, page structure, and API integration.	**Jacob**	All members

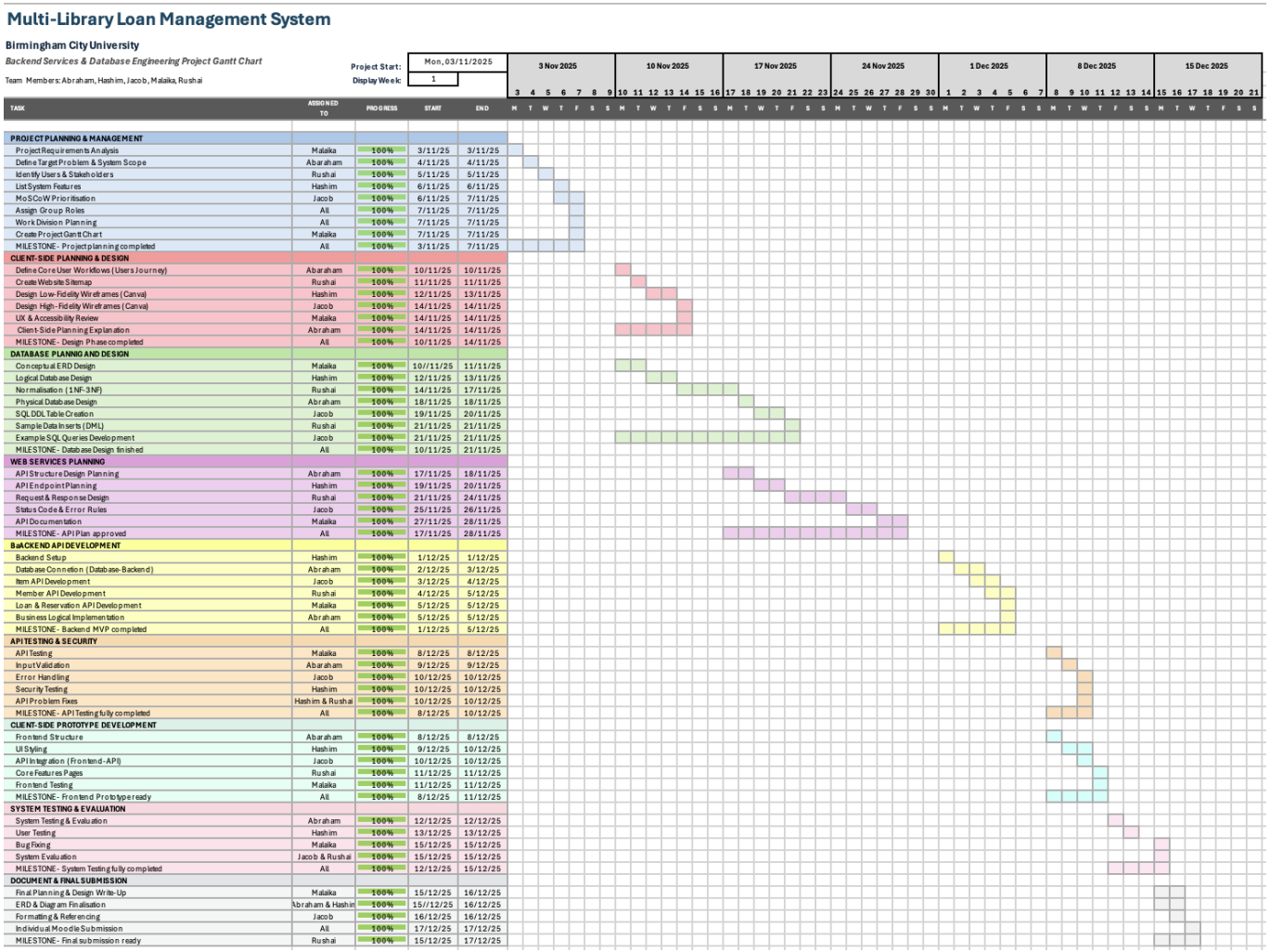
Documentation (D2 and D3)	Writing the planning, design, and evaluation documentation, including requirements analysis, diagrams, and final assembly.	**Malaika**	All members
Testing and Quality Assurance	Unit testing of backend endpoints, integration testing with the client, data validation, bug fixing, and evaluation against functional and non-functional requirements.	**Hashim**	All members

Table 5 Work Division Plan

Explanation

*This work division shown in **Table 5** ensures that all group members contributed across planning, design, development, testing, and documentation. While leads were assigned for each area, all members remained involved throughout the project, supporting flexibility and shared understanding of the system.*

Project Timeline (Gantt Chart)



Multi-Library Loan Management System

Birmingham City University

Backend Services & Database Engineering Project Gantt Chart

Team Members: Abraham, Hashim, Jacob, Malaika, Rushai

Project Start: Mon, 03/11/2025

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
PROJECT PLANNING & MANAGEMENT				
Project Requirements Analysis	Malaika	100%	3/11/25	3/11/25
Define Target Problem & System Scope	Abaraham	100%	4/11/25	4/11/25
Identify Users & Stakeholders	Rushai	100%	5/11/25	5/11/25
List System Features	Hashim	100%	6/11/25	6/11/25
MoSCoW Prioritisation	Jacob	100%	6/11/25	7/11/25
Assign Group Roles	All	100%	7/11/25	7/11/25
Work Division Planning	All	100%	7/11/25	7/11/25
Create Project Gantt Chart	Malaika	100%	7/11/25	7/11/25
MILESTONE - Project planning completed	All	100%	3/11/25	7/11/25
CLIENT-SIDE PLANNING & DESIGN				
Define Core User Workflows (Users Journey)	Abaraham	100%	10/11/25	10/11/25
Create Website Sitemap	Rushai	100%	11/11/25	11/11/25
Design Low-Fidelity Wireframes (Canva)	Hashim	100%	12/11/25	13/11/25
Design High-Fidelity Wireframes (Canva)	Jacob	100%	14/11/25	14/11/25
UX & Accessibility Review	Malaika	100%	14/11/25	14/11/25
Client-Side Planning Explanation	Abraham	100%	14/11/25	14/11/25
MILESTONE - Design Phase completed	All	100%	10/11/25	14/11/25
DATABASE PLANNING AND DESIGN				
Conceptual ERD Design	Malaika	100%	10//11/25	11/11/25
Logical Database Design	Hashim	100%	12/11/25	13/11/25
Normalisation (1NF-3NF)	Rushai	100%	14/11/25	17/11/25
Physical Database Design	Abraham	100%	18/11/25	18/11/25
SQL DDL Table Creation	Jacob	100%	19/11/25	20/11/25
Sample Data Inserts (DML)	Rushai	100%	21/11/25	21/11/25
Example SQL Queries Development	Jacob	100%	21/11/25	21/11/25
MILESTONE - Database Design finished	All	100%	10/11/25	21/11/25
WEB SERVICES PLANNING				
API Structure Design Planning	Abraham	100%	17/11/25	18/11/25
API Endpoint Planning	Hashim	100%	19/11/25	20/11/25
Request & Response Design	Rushai	100%	21/11/25	24/11/25
Status Code & Error Rules	Jacob	100%	25/11/25	26/11/25
API Documentation	Malaika	100%	27/11/25	28/11/25
MILESTONE - API Plan approved	All	100%	17/11/25	28/11/25
BaCKEND API DEVELOPMENT				
Backend Setup	Hashim	100%	1/12/25	1/12/25
Database Connection (Database-Backend)	Abraham	100%	2/12/25	3/12/25
Item API Development	Jacob	100%	3/12/25	4/12/25
Member API Development	Rushai	100%	4/12/25	5/12/25
Loan & Reservation API Development	Malaika	100%	5/12/25	5/12/25
Business Logical Implementation	Abraham	100%	5/12/25	5/12/25
MILESTONE - Backend MVP completed	All	100%	1/12/25	5/12/25
API TESTING & SECURITY				
API Testing	Malaika	100%	8/12/25	8/12/25
Input Validation	Abaraham	100%	9/12/25	9/12/25
Error Handling	Jacob	100%	10/12/25	10/12/25
Security Testing	Hashim	100%	10/12/25	10/12/25
API Problem Fixes	Hashim & Rushai	100%	10/12/25	10/12/25
MILESTONE - API Testing fully completed	All	100%	8/12/25	10/12/25
CLIENT-SIDE PROTOTYPE DEVELOPMENT				
Frontend Structure	Abaraham	100%	8/12/25	8/12/25
UI Styling	Hashim	100%	9/12/25	10/12/25
API Integration (Frontend-API)	Jacob	100%	10/12/25	10/12/25
Core Features Pages	Rushai	100%	11/12/25	11/12/25
Frontend Testing	Malaika	100%	11/12/25	11/12/25
MILESTONE - Frontend Prototype ready	All	100%	8/12/25	11/12/25
SYSTEM TESTING & EVALUATION				
System Testing & Evaluation	Abraham	100%	12/12/25	12/12/25
User Testing	Hashim	100%	13/12/25	13/12/25
Bug Fixing	Malaika	100%	15/12/25	15/12/25
System Evaluation	Jacob & Rushai	100%	15/12/25	15/12/25
MILESTONE - System Testing fully completed	All	100%	12/12/25	15/12/25
DOCUMENT & FINAL SUBMISSION				
Final Planning & Design Write-Up	Malaika	100%	15/12/25	16/12/25
ERD & Diagram Finalisation	Abraham & Hashim	100%	15//12/25	16/12/25
Formatting & Referencing	Jacob	100%	16/12/25	16/12/25
Individual Moodle Submission	All	100%	17/12/25	17/12/25
MILESTONE - Final submission ready	Rushai	100%	15/12/25	17/12/25

Figure 4 Gantt chart showing project schedule, dependencies, and milestones for the multi-library loan management system (Microsoft Excel, 2025).

Figure 4 shows a clear project timeline. It highlights the dependencies among planning, design, development, testing, and documentation activities. Tasks were arranged to allow parallel work where possible and to ensure that prerequisite activities were completed first. This approach lowers project risk, supports good team coordination, and helps make sure that all deliverables meet the module deadlines.

Task Allocation

*Task allocation followed the planned schedule outlined in the Gantt chart (see **Figure 4**) and the responsibility distribution defined in the RACI matrix (see **Table 4**). While individual members led specific areas based on their strengths, tasks such as requirement analysis, system testing, and final submission were completed collaboratively to ensure shared understanding and alignment with learning outcomes.*

Planning and Design: Client side

The client-side part of the system will be a basic prototype interface that shows the main functions of the backend web service. Its purpose is not to behave as a full library portal for members or staff, but rather to demonstrate how users would interact with the key features of the system. This includes features such as searching for items, borrowing books, returning books, and making reservations.

The interface will be built using HTML, CSS and some basic JavaScript, drawing on what we learned in this module and the previous web development module from the first year. The design will mainly focus on clarity, simplicity, and ease of use, ensuring each action performed by the user has a corresponding backend API call. This approach aligns with the original problem statement, where users and staff require easy access to real-time item availability and loan management.

The prototype will include a few pages or views to support the main workflows:

- Search Items: Allows the user to look for items by their title, author, ISBN, or library branch.
- View Item Availability: Shows an item's current status (available, loaned, reserved) across all library branches.
- Borrow Item: Offers a form for entering a member ID and item ID to create a loan.

- Return Item: Lets users record the return of an item and start late fee calculation if needed.
- Reserve Item: Allows users to put a reservation on an item that is currently on loan.
- Member Summary: Displays a member's active loans, reservations, and outstanding fees.

The user experience will focus mainly on clearly labelled buttons, simple forms, and minimal layouts. To ensure the prototype is functional and easy to use while also highlighting the backend system's functionality.

Client-Side Planning: Sitemap

The sitemap offers a clear outline of how users move between the pages of the proposed library loan system, as shown in **Figure 5**. It makes sure that the layout helps users easily find what they need. This allows members, staff, and administrators to easily access primary functions of the system, like searching for items, viewing item details, managing loans, and using staff tools. The final sitemap differentiates between areas for users and areas for staff, improving the system's overall accessibility and navigation. The sitemap was created with Canva to provide a clear visual model of the client-side structure (**Canva, 2024**).

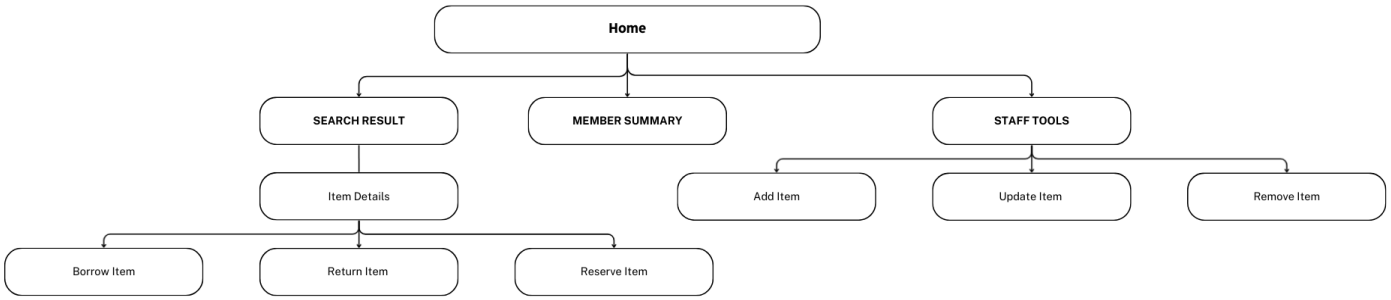


Figure 5 System Sitemap Diagram (created using Canva, 2024).

Client-Side Planning: Wireframes

*The wireframes illustrate the planned layout and structure of the systems user interface, designed for a simple and intuitive experience. Clarity and minimalism are prioritised to support easy navigation of key workflows. Canva sketches of the homepage, search results, and member summary with staff tools (see **Appendix A, Figures A1-A3**), explored layout options and workflows, which were then refined into high-fidelity Figma wireframes (**Appendix B**).*

*The homepage highlights the search bar as the primary interaction point (**Figure 6**), while the search results and item details pages present book information with contextual actions (**Appendix B, Figures B1-B2**). Borrowing, reservation, and return confirmation pages provide concise feedback, with dates and status information (**Appendix B, Figures B3-B5**). The member summary page shows current loans, reservations, and late fees in a structured format (**Figure 7**).*

*Consistent styling, spacing, and a light beige background support readability and focus. To avoid visual clutter in the main design section, only the homepage and member summary wireframes are presented here; other screens are in **Appendix B**. These wireframes were created using **Figma (2024)** to help connect the conceptual design with the final implementation. Staff management tools shown in the initial Canva sketches were not developed further in Figma to focus on core member workflows.*

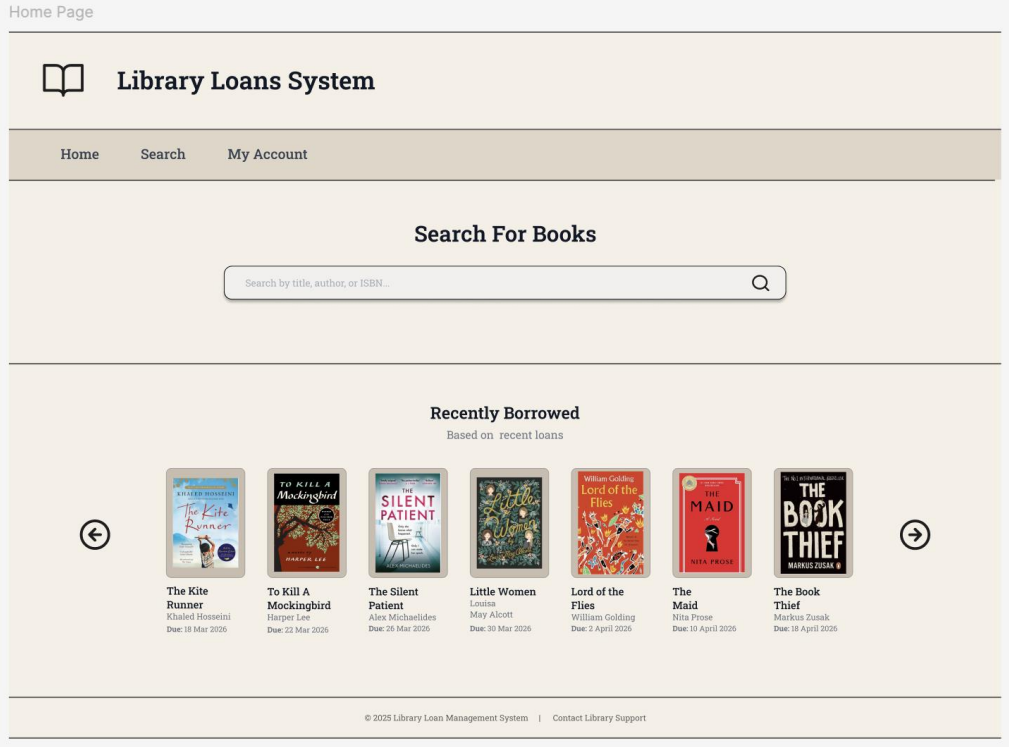


Figure 6 Home Page Wireframe (created using Figma, 2024).

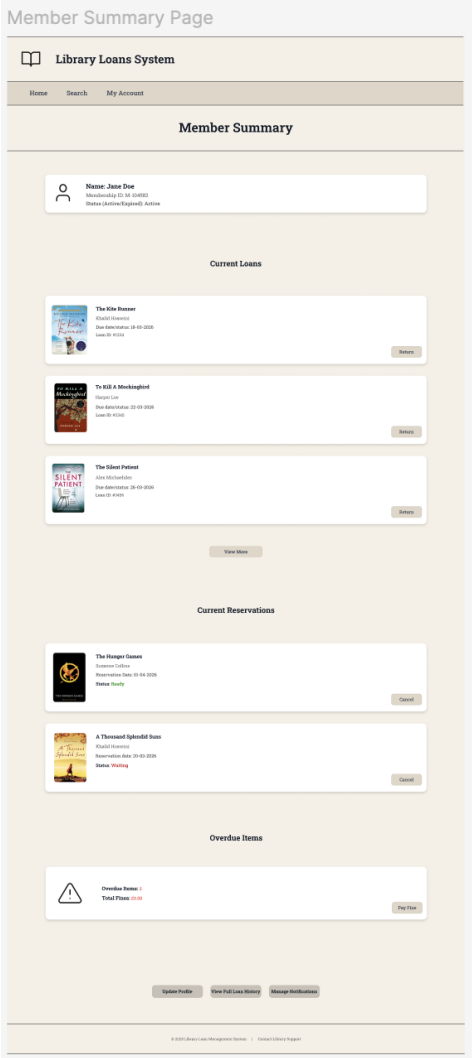


Figure 7 Member summary and staff tools wireframe illustrating active loans, current reservations, overdue items, and available account management actions (Figma, 2024).

Planning and Design: Database

Conceptual design

At the conceptual level, the database needs to support a local authority that manages multiple library branches, each holding many items that members can borrow or reserve. The main elements in the real world include:

- Library branches that own physical copies of items.
- Items that represent logical works (for example, a specific book title).
- Item copies that represent the individual physical copies stored at branches.
- Members who borrow and reserve items.
- Loans that represent a member borrowing a specific copy for a certain time.
- Reservations that represent a member waiting to borrow an item when no suitable copy is available.
- Staff users who manage the catalogue and carry out actions like adding or removing items.

These entities and relationships are shown in the conceptual ERD in **Figure 8**.

Key relationships include:

- A branch holds many item copies.
- An item can have many copies, possibly across different branches.
- A member can have many loans and many reservations.
- A loan always connects one member to one item copy.
- A reservation connects one member to one item and can include a preferred pickup branch.
- A staff user belongs to one branch and can manage many items or copies.

This modelling approach reflects common database design practices for identifying main entities and their relationships before choosing data types or setting up the physical schema (**Connolly and Begg, 2015**). This model directly influenced the logical and physical database designs mentioned later in this report. It also supports the main system features outlined earlier.

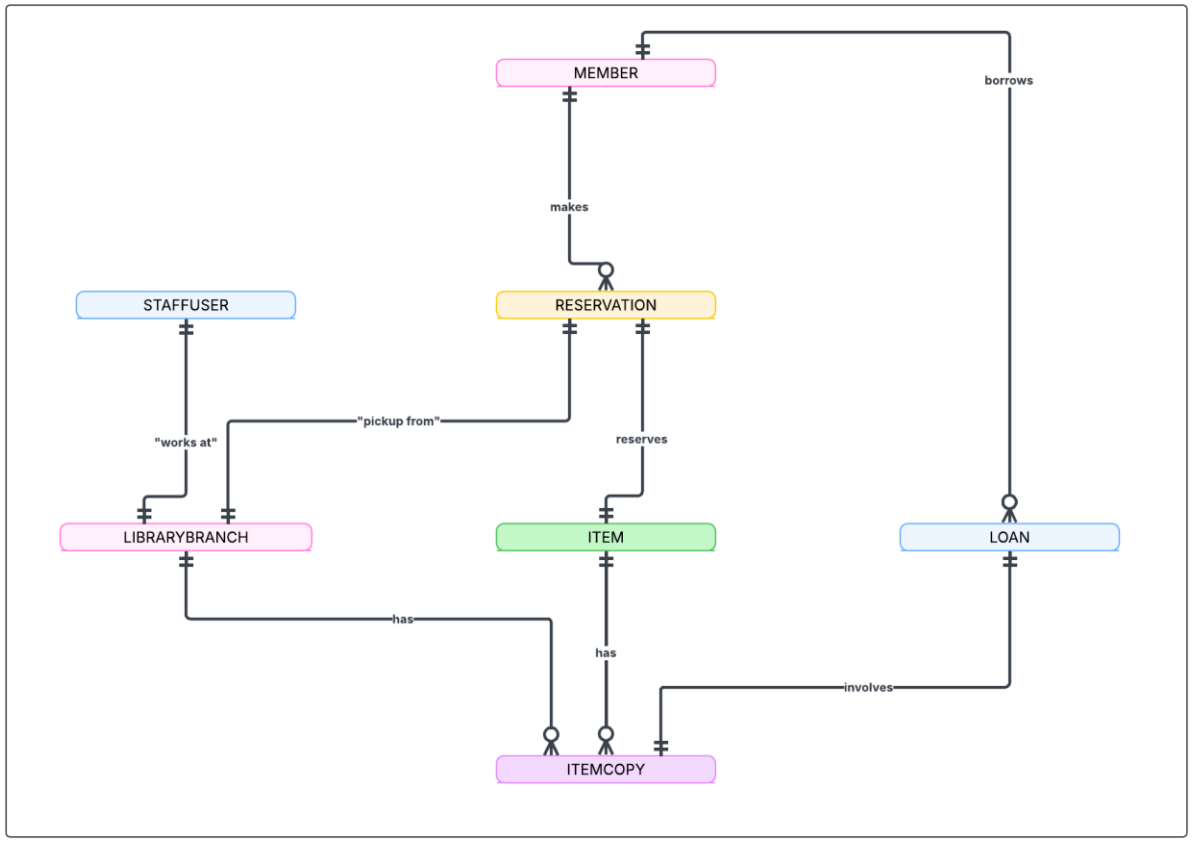


Figure 8 Conceptual Entity-Relationship Diagram created using diagrams.net (Lucidchart, 2024).

Logical design

In the logical design, the conceptual entities are mapped to relational tables. Primary keys, foreign keys, and cardinalities are clearly defined. The design is normalised to reduce redundancy.

Proposed main tables:

- LibraryBranch
    - *BranchID (PK)*
    - *Name*
    - *Address*
    - *City*
    - *Postcode*
- Item
    - *ItemID (PK)*
    - *Title*
    - *Author*
    - *ISBN*
    - *Category*
    - *PublicationYear*
- ItemCopy
    - *CopyID (PK)*
    - *ItemID (FK → Item)*
    - *BranchID (FK → LibraryBranch)*
    - *Status (for example, Available, OnLoan, Reserved, Lost)*
    - *ShelfLocation*
- Member
    - *MemberID (PK)*
    - *FirstName*

    - *LastName*
    - *Email*
    - *Phone*
    - *HomeBranchID (FK → LibraryBranch)*
- Loan
    - *LoanID (PK)*
    - *CopyID (FK → ItemCopy)*
    - *MemberID (FK → Member)*
    - *LoanDate*
    - *DueDate*
    - *ReturnDate (nullable)*
    - *LateFeeAmount (calculated and stored when the item is returned)*
- Reservation
    - *ReservationID (PK)*
    - *ItemID (FK → Item)*
    - *MemberID (FK → Member)*
    - *PickupBranchID (FK → LibraryBranch)*
    - *ReservationDate*
    - *Status (for example, Active, Fulfilled, Cancelled)*
    - *QueuePosition*
- StaffUser
    - *StaffID (PK)*
    - *FirstName*
    - *LastName*
    - *Email*
    - *BranchID (FK → LibraryBranch)*
    - *Role*

These tables and their relationships are illustrated in the logical ERD shown in **Figure 9**.

Logical relationships and cardinalities:

- One LibraryBranch to many ItemCopy (one to many).
- One Item to many ItemCopy (one to many).
- One Member to many Loans (one to many).
- One Member to many Reservations (one to many).
- One ItemCopy to many Loans over time (historical loans), but a copy can only be on one active loan at a time.
- One Item to many Reservations (many members can queue for the same title).
- One LibraryBranch to many StaffUser.

This logical model directly supports the system's required features. Real-time availability comes from the ItemCopy.Status attribute. Loan histories and member summaries use the Loan and Reservation tables, while late fees are based on LoanDate, DueDate, and ReturnDate. These structures follow accepted principles of relational database design and normalisation. They help ensure consistency and efficient querying across the system (**Coronel and Morris, 2019**).

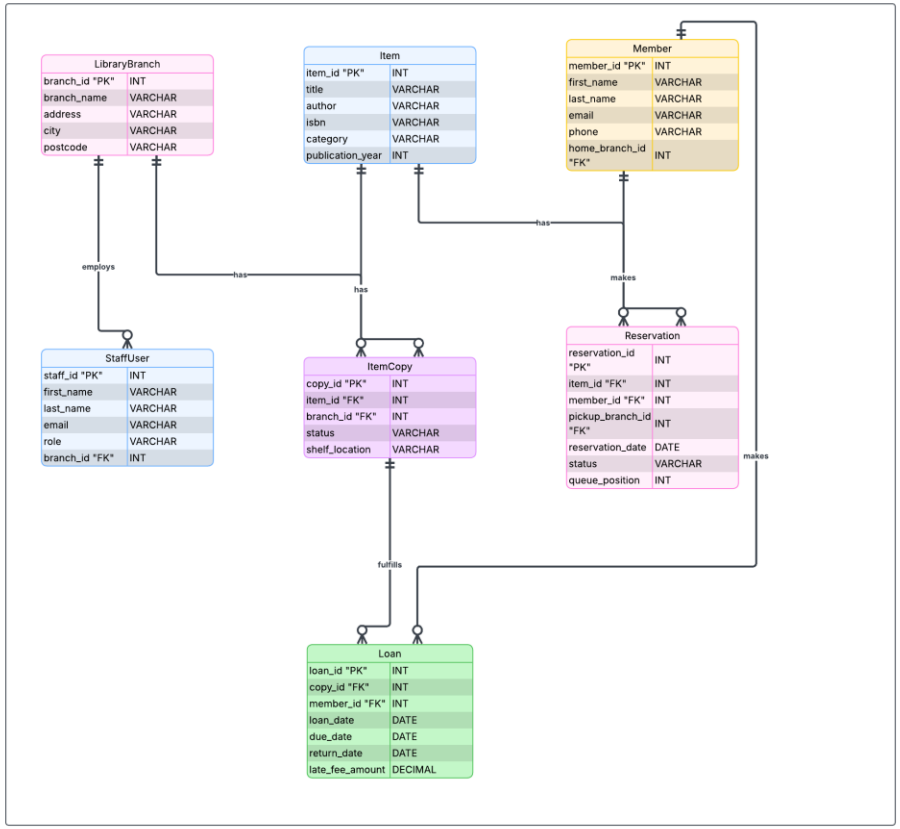


Figure 9 Logical Entity-Relationship Diagram showing relational tables and cardinalities (Lucidchart, 2024).

Physical design

The physical design describes how the logical model will be organised on the chosen relational database management system, such as MySQL or PostgreSQL. At this stage, decisions are made about data types, indexing, constraints, and storage structures to make sure the database functions properly and keeps data safe. These decisions follow the guidelines from the MySQL 8.0 Reference Manual (**MySQL, 2024**).

Key physical design decisions:

- Data types
    - *IDs such as BranchID, ItemID, CopyID, MemberID and LoanID implemented as INT (or SERIAL / AUTO_INCREMENT depending on DBMS).*
    - *Names and titles stored as VARCHAR with appropriate length (for example, VARCHAR(100) for names, VARCHAR(255) for titles).*
    - *Dates stored as DATE and timestamps as TIMESTAMP.*
    - *Status fields implemented as VARCHAR(20) or as a small lookup table if it is needed.*
- Primary and foreign keys
    - *Each table has a single-column primary key.*
    - *Foreign key constraints are added to ensure referential integrity between ItemCopy, Loan, Reservation and their parent tables.*
    - *Cascading rules can be used carefully, for example, preventing deletion of items that still have active copies or loans.*
- Indexes
    - *Index on Item.Title, Item.Author and Item.ISBN to speed up search.*
    - *Index on ItemCopy.ItemID and ItemCopy.BranchID for availability queries.*
    - *Index on Loan.MemberID and Reservation.MemberID to support member summary views.*
- Derived attributes
    - *Availability is not stored as a separate table. It is derived from ItemCopy records with Status = Available for each branch.*
    - *LateFeeAmount may be stored in the Loan table after calculation, to make reporting simpler.*
- Physical ERD

  A physical ERD in Crow's Foot notation was created to show the final table structures, including columns, primary keys, and

foreign keys. This ERD served as the blueprint for the SQL data definition language (DDL) script that was used to create the database schema.

A prototype SQL script was built based on this design and filled with sample data for a small number of branches, items, and members. Example queries were run to demonstrate key system functionality, including:

- Finding available copies of an item across all branches.
- Listing all current loans and reservations for a specific member.
- Identifying overdue loans and calculating the associated late fees

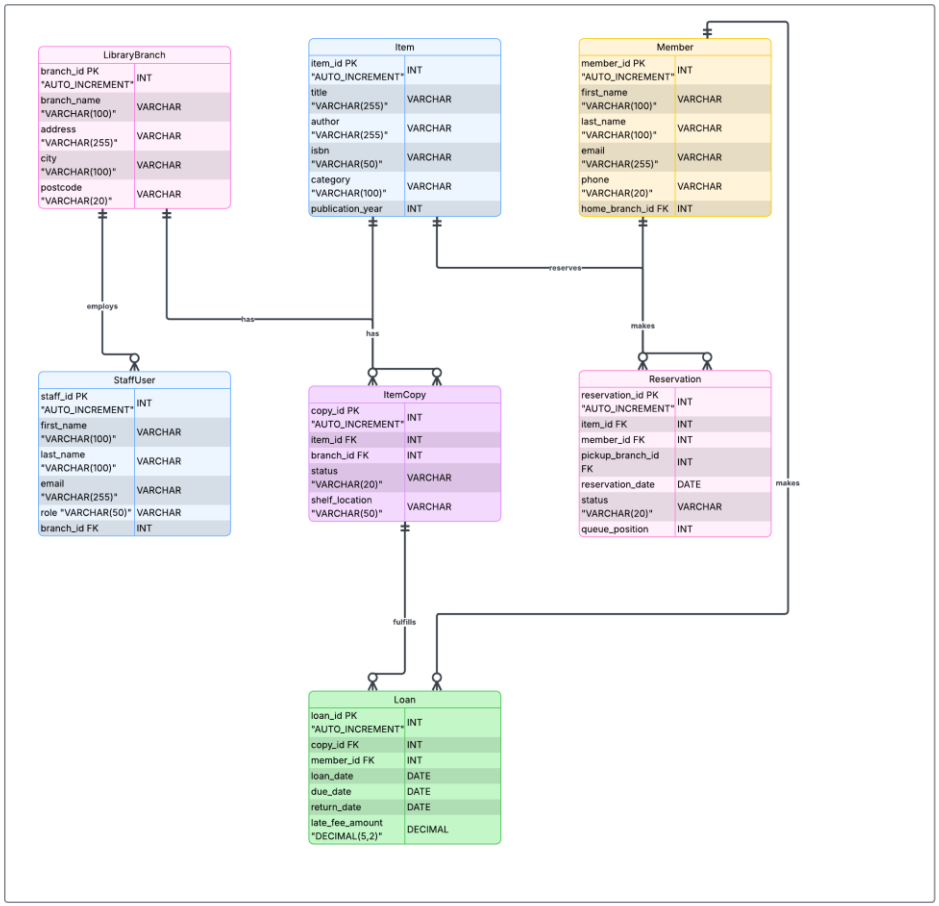


Figure 10 Physical Entity-Relationship Diagram in Crow's Foot notation showing final table structures and relationships (Lucidchart, 2024).

The physical ERD in **Figure 10** directly guided the SQL schema used in the physical prototyping stage. It mapped the tables, keys, and constraints clearly into the prototype database.

This database design offers a strong base for the backend web service. It directly supports the main features identified for the multi-library loan management system.

A relational database model was chosen instead of a NoSQL approach because of the well-defined relationships between members, items, loans, and reservations. There is also a need for strong data consistency.

Physical Prototyping

This subsection shows how to take the physical database model and implement it using SQL. To do this, this section contains sample Data Definition Language (DDL) statements that focus on the construction of the database schema, optional inserts using the Data Manipulation Language (DML) for testing purposes, and example SQL queries that support key functionalities of the system, which include checking the availability of items, managing loans, and processing reservations. The SQL statements follow standard MySQL practices for schema creation, indexing, and querying (**MySQL, 2024**).

Prototype SQL (DDL)

Below is an SQL implementation of the Physical ERD using MySQL-style syntax.

```sql
-- 1. Library Branch Table
CREATE TABLE LibraryBranch (
    branch_id INT AUTO_INCREMENT PRIMARY KEY,
    branch_name VARCHAR(100) NOT NULL,
    address VARCHAR(255),
    city VARCHAR(100),
    postcode VARCHAR(20)
);

-- 2. Item Table
```

```sql
CREATE TABLE Item (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    isbn VARCHAR(50),
    category VARCHAR(100),
    publication_year INT
);

-- 3. ItemCopy Table
CREATE TABLE ItemCopy (
    copy_id INT AUTO_INCREMENT PRIMARY KEY,
    item_id INT NOT NULL,
    branch_id INT NOT NULL,
    status VARCHAR(20) NOT NULL,
    shelf_location VARCHAR(50),
    FOREIGN KEY (item_id) REFERENCES Item(item_id),
    FOREIGN KEY (branch_id) REFERENCES LibraryBranch(branch_id)
);

-- 4. Member Table
CREATE TABLE Member (
    member_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(255),
    phone VARCHAR(20),
    home_branch_id INT,
    FOREIGN KEY (home_branch_id) REFERENCES LibraryBranch(branch_id)
);

-- 5. Loan Table
CREATE TABLE Loan (
    loan_id INT AUTO_INCREMENT PRIMARY KEY,
    copy_id INT NOT NULL,
```

```sql
    member_id INT NOT NULL,
    loan_date DATE NOT NULL,
    due_date DATE NOT NULL,
    return_date DATE,
    late_fee_amount DECIMAL(5,2),
    FOREIGN KEY (copy_id) REFERENCES ItemCopy(copy_id),
    FOREIGN KEY (member_id) REFERENCES Member(member_id)
);

-- 6. Reservation Table
CREATE TABLE Reservation (
    reservation_id INT AUTO_INCREMENT PRIMARY KEY,
    item_id INT NOT NULL,
    member_id INT NOT NULL,
    pickup_branch_id INT NOT NULL,
    reservation_date DATE NOT NULL,
    status VARCHAR(20) NOT NULL,
    queue_position INT,
    FOREIGN KEY (item_id) REFERENCES Item(item_id),
    FOREIGN KEY (member_id) REFERENCES Member(member_id),
    FOREIGN KEY (pickup_branch_id) REFERENCES LibraryBranch(branch_id)
);

-- Indexes to improve performance
CREATE INDEX idx_item_title ON Item(title);
CREATE INDEX idx_itemcopy_item ON ItemCopy(item_id);
CREATE INDEX idx_itemcopy_branch ON ItemCopy(branch_id);
CREATE INDEX idx_loan_member ON Loan(member_id);

CREATE INDEX idx_reservation_member ON Reservation(member_id);
```

Code Snippet 1 SQL DDL for Physical Database Prototype

Sample Data Inserts (DML)

```sql
-- Insert Library Branches
INSERT INTO LibraryBranch (branch_name, address, city, postcode)
VALUES
('Library of Birmingham', 'Centenary Sq', 'Birmingham', 'B1 2ND'),
('Curzon Library', '4 Cardigan St', 'Birmingham', 'B4 7BD');

-- Insert Items (Books)
INSERT INTO Item (title, author, isbn, category, publication_year)
VALUES
('Meditations', 'Marcus Aurelius', '9780140449334', 'Philosophy', 180),
('Harry Potter and the Philosopher's Stone', 'J.K. Rowling', '9780747532743', 'Fantasy', 1997);

-- Insert Item Copies
INSERT INTO ItemCopy (item_id, branch_id, status, shelf_location)
VALUES
(1, 1, 'Available', 'PHI-A12'),
(1, 2, 'OnLoan', 'PHI-B03'),
(2, 1, 'Available', 'FAN-C07'),
(2, 2, 'Available', 'FAN-D14');

-- Insert Members
INSERT INTO Member (first_name, last_name, email, phone, home_branch_id)
VALUES
('Abraham', 'Sharkey', 'abraham.sharkey@mail.bcu.ac.uk', '07000000000', 1),
('Charles', 'Leclerc', 'charles.leclerc@mail.bcu.ac.uk', '07000000001', 2);

-- Insert Loan
INSERT INTO Loan (copy_id, member_id, loan_date, due_date)
VALUES
(3, 2, '2025-02-01', '2025-02-15');

-- Insert Reservation
INSERT INTO Reservation (item_id, member_id, pickup_branch_id, reservation_date, status, queue_position)
```

```
VALUES

(1, 1,  1, '2025-02-05', 'Active', 1);
```

Code Snippet 2 SQL DML Sample Data Inserts

Example SQL Queries

Query 1: Find all available copies of a specific item

```
SELECT ic.copy_id, lb.branch_name, ic.shelf_location
FROM ItemCopy ic
JOIN LibraryBranch lb ON ic.branch_id = lb.branch_id
WHERE ic.item_id = 1 AND ic.status = 'Available';
```

Code Snippet 3 SQL Query to Retrieve Available Copies

Query 2: Get a full loan summary for a member

```
SELECT l.loan_id, i.title, l.loan_date, l.due_date, l.return_date, l.late_fee_amount
FROM Loan l
JOIN ItemCopy ic ON l.copy_id = ic.copy_id
JOIN Item i ON ic.item_id = i.item_id
WHERE l.member_id = 1;
```

Code Snippet 4 SQL Query for Member Loan Summary

Query 3: List all overdue loans

```
SELECT l.loan_id, m.first_name, m.last_name, i.title, l.due_date
FROM Loan l
```

```sql
JOIN Member m ON l.member_id = m.member_id
JOIN ItemCopy ic ON l.copy_id = ic.copy_id
JOIN Item i ON ic.item_id = i.item_id
WHERE l.return_date IS NULL AND l.due_date < CURDATE();
```

Code Snippet 5 SQL Query to Identify Overdue Loans

Query 4: View reservation queue for an item

```sql
SELECT r.queue_position, m.first_name, m.last_name, r.status
FROM Reservation r
JOIN Member m ON r.member_id = m.member_id
WHERE r.item_id = 1
ORDER BY r.queue_position ASC;
```

Code Snippet 6 SQL Query for Reservation Queue

Query 5: Count how many available copies exist per branch

```sql
SELECT lb.branch_name, COUNT(*) AS available_copies
FROM ItemCopy ic
JOIN LibraryBranch lb ON ic.branch_id = lb.branch_id
WHERE ic.status = 'Available'
GROUP BY lb.branch_name;
```

Code Snippet 7 SQL Query for Branch Availability Count

Planning and Design: Web Services

The web service for the multi-library loan system uses a REST-based API structure. Each endpoint supports a specific feature, such as searching for items, checking availability, issuing and returning loans, managing reservations, and maintaining catalogue data for staff. This design follows

standard REST principles, which emphasize stateless communication, resource-oriented URIs, and predictable interaction patterns (**Fielding, 2000**). The behaviour of HTTP methods like GET, POST, PUT, and DELETE follows widely accepted standards for web communication. This ensures consistent and interoperable endpoint interactions (**MDN Web Docs, 2023**). Additionally, the use of semantic HTTP status codes follows definitions in RFC 7231. This ensures clear communication between the client and server (IETF, 2014).

The following pages present the complete resource inventory. This includes endpoint descriptions, expected request and response formats, and status code behaviours, as summarised in **Table 6**. These are followed by a protocol specification and a discussion explaining the design decisions.

Endpoint	**HTTP** Method	**Description**	**Example Input (Request)**	**Example Output (Response)**	**Status** Code	Happy/**Sad**
/api/v1/items	GET	Returns a list of all items.	Request: N/A (GET request has no body)	{ "items": [...], "page": 1 }	200	Happy
/api/v1/items/{item_id}	GET	Returns full details of a single item.	Request: /api/v1/items/1	{ "item_id": 1, "title": "Meditations", ... }	200	Happy
/api/v1/items/{item_id}	GET	Item not found.	Request: /api/v1/items/9999	{ "error": "Item not found" }	404	Sad
/api/v1/search	GET	Searches items by title, author or ISBN.	Request: /api/v1/search?query=harry	{ "results": [...] }	200	Happy
/api/v1/items/{item_id}/copies	GET	Lists all item copies with availability.	Request: /api/v1/items/1/copies	{ "copies": [...] }	200	Happy
/api/v1/members/{member_id}	GET	Shows member details, active loans, reservations and fines.	Request: /api/v1/members/1	{ "active_loans": [...], "reservations": [...], "outstanding_fines": 0 }	200	Happy
/api/v1/members/{member_id}	GET	Member not found.	Request: /api/v1/members/9999	{ "error": "Member not found" }	404	Sad
/api/v1/loans	POST	Creates a new loan.	Headers: content-type: application/jsonBody: { "copy_id": 3, "member_id": 2 }	{ "message": "Loan created", "loan_id": 12 }	201	Happy
/api/v1/loans	POST	Copy unavailable.	Headers: content-type: application/jsonBody: { "copy_id": 2, "member_id": 1 }	{ "error": "Copy not available" }	409	Sad

/api/v1/loans/{loan_id}/return	PUT	Returns a borrowed item.	Headers: content-type: application/jsonBody: { "return_date": "2025-02-16" }	{ "message": "Item returned", "late_fee_amount": 0.50 }	200	Happy
/api/v1/loans/{loan_id}/return	PUT	Loan not found.	Headers: content-type: application/jsonBody: { "return_date": "2025-02-16" }	{ "error": "Loan not found" }	404	Sad
/api/v1/reservations	POST	Creates a reservation.	Headers: content-type: application/jsonBody: { "item_id": 1, "member_id": 1, "pickup_branch_id": 1 }	{ "message": "Reservation created" }	201	Happy
/api/v1/reservations	POST	Already in queue.	Headers: content-type: application/jsonBody: { "item_id": 1, "member_id": 1 }	{ "error": "Already in queue" }	409	Sad
/api/v1/reservations/{item_id}	GET	Shows reservation queue for an item.	Request: /api/v1/reservations/1	{ "queue": [...] }	200	Happy
/api/v1/items	POST	Adds a new catalogue item.	Headers: content-type: application/jsonBody: { "title": "New Book", "author": "Example", ... }	{ "message": "Item added" }	201	Happy
/api/v1/items/{item_id}	PUT	Updates item details.	Headers: content-type: application/jsonBody: { "category": "Fantasy" }	{ "message": "Item updated" }	200	Happy
/api/v1/items/{item_id}	DELETE	Removes item with no copies left.	Request: /api/v1/items/8	{ "message": "Item removed" }	200	Happy
/api/v1/items/{item_id}	DELETE	Cannot delete item because copies still exist.	Request: /api/v1/items/1	{ "error": "Cannot delete item" }	400	Sad

Table 6 Resource URI inventory summary table

Protocol Specification

The protocol specification describes the expected behaviour of each endpoint, including request structure, data validation rules and response formats.

Request format

All requests that include a body use JSON formatting. POST and PUT operations must consist of the Content-Type: application/json header to ensure the server interprets the data correctly. Path parameters, such as /items/{item_id}, must be validated to confirm they represent valid resource identifiers. Input data should also be checked for required fields and correct data types before the request is processed.

Response Format

The API uses a consistent JSON response structure and standardised status codes, as shown in **Table 6**:

- 200 OK — Successful retrieval or update
- 201 Created — Successful creation of a resource (loan, reservation or item)
- 400 Bad Request — Missing or invalid input
- 404 Not Found — Resource does not exist
- 409 Conflict — Valid request but cannot be completed due to system rules (e.g., copy unavailable)

These codes follow HTTP standards and are used to make interactions predictable and machine-readable (IETF, 2014).

Validation Rules

Examples of rule enforcement include:

- A loan can only be created for a copy with status = 'Available'.
- A reservation cannot be made if the member is already in the queue.
- An item cannot be deleted if copies still exist or if any active loans reference it

These rules ensure system integrity and reflect the relationships defined in the ERD and logical model (**Connolly and Begg, 2015**).

Discussion and Justification

The API design follows a clear REST structure with resource-focused endpoints that each represent a single, meaningful action. This method makes it easier to maintain while also making it more readable and easier to test. Furthermore, using well-defined HTTP methods makes sure the client behaviour is more predictable and consistent with standard definitions (**MDN Web Docs, 2023**).

Clear status codes improve error handling while also reducing vagueness. For instance, 201 Created indicates the successful creation of a resource, while 409 Conflict tells the client that the request was valid but could not be completed, for example, when trying to loan a book that has already been checked out. These distinctions help the client decide what to do. Whether that is to retry, modify the input, or display an error message.

A RESTful API architecture was selected for its simplicity, scalability, and compatibility with standard HTTP methods. It does, however, introduce some overhead compared to a more tightly coupled design. While this adds some extra work compared to more closely linked designs, it improves compatibility and long-term upkeep.

The member summary endpoint was designed to return all essential information (loans, reservations, fines) in a single request. This makes the client-side less complex and limits the number of API calls needed to create a complete user profile.

Overall, the REST API supports all main system features, including searching for books, checking availability, issuing loans, managing returns, handling reservations, and maintaining catalogue data. The structure also keeps the door open for future improvements such as user authentication, rate limiting, and role-based access control. The design follows standard RESTful and HTTP principles, which boost scalability and future readiness (**Fielding, 2000; MDN Web Docs, 2023**).

References

Canva *(2024) Online design tool. Available at:* https://www.canva.com/ *(Accessed: 14 December 2025).*

Clegg, D. and Barker, R. *(1994) Case Method Fast-Track: A RAD Approach. Wokingham: Addison-Wesley.*

Connolly, T. and Begg, C*. (2015) Database Systems: A Practical Approach to Design, Implementation, and Management. 6th edn. Harlow: Pearson.*

Coronel, C. and Morris, S. *(2019) Database Systems: Design, Implementation, and Management. 13th edn. Boston: Cengage Learning.*

Fielding, R. *(2000) Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.*

Grammarly *(2024) Grammar and writing assistant tool. Available at:* https://www.grammarly.com/ *(Accessed: 15 December 2025).*

IETF *(2014) RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. Internet Engineering Task Force. Available at:* https://datatracker.ietf.org/doc/html/rfc7231 *(Accessed: 16 December 2025).*

Lucidchart *(2024) Online diagramming and visual modelling tool. Available at: https://www.lucidchart.com (Accessed: 14 December 2025).*

MDN Web Docs *(2023) HTTP request methods—MDN Web Docs. Available at:* https://developer.mozilla.org/ *(Accessed: 12 December 2025).*

MySQL *(2024) MySQL 8.0 Reference Manual. Oracle. Available at:* https://dev.mysql.com/doc/refman/8.0/en/ *(Accessed: 13 December 2025).*

W3Schools *(2025) SQL Tutorial. Available at:* https://www.w3schools.com/sql/ *(Accessed: 11 December 2025).*

Figma *(2024) Collaborative interface design tool. Available at:* https://www.figma.com/ *(Accessed: 14 December 2025).*

Microsoft *(2025) Microsoft Excel. Spreadsheet software. Available at: https://www.microsoft.com/excel (Accessed: 10 December 2025).*

Miro *(2024) Online collaborative whiteboard platform. Available at: https://miro.com/ (Accessed: 13 December 2025).*

Project Management Institute *(PMI) (2017) A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* 6th edn. Newtown Square, PA: Project Management Institute.

Diagrams.net (2024) online diagramming tool. Available at: https://www.diagrams.net/ (Accessed: 14 December 2025).

Appendices

Appendix A: Canva Sketches (low-fidelity wireframes)

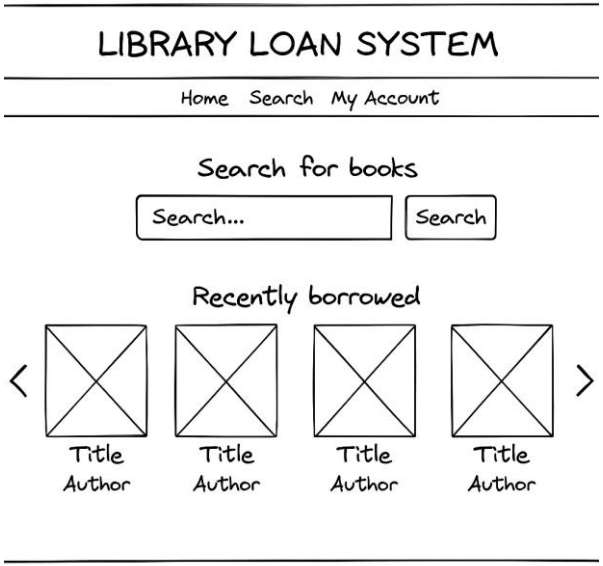


Figure A1: Search Results Wireframe (created using Canva, 2025).

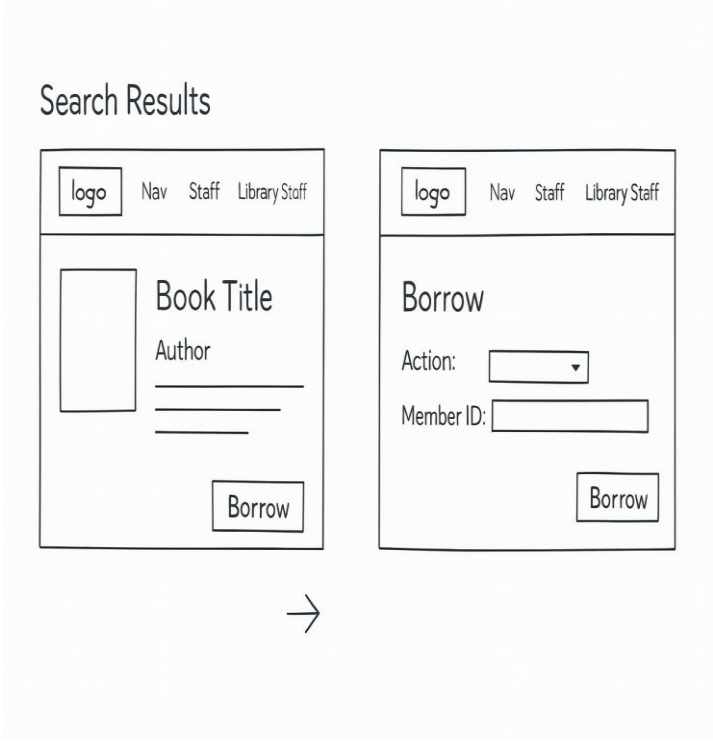


Figure A2: Search Results Wireframe (created using Canva, 2025).

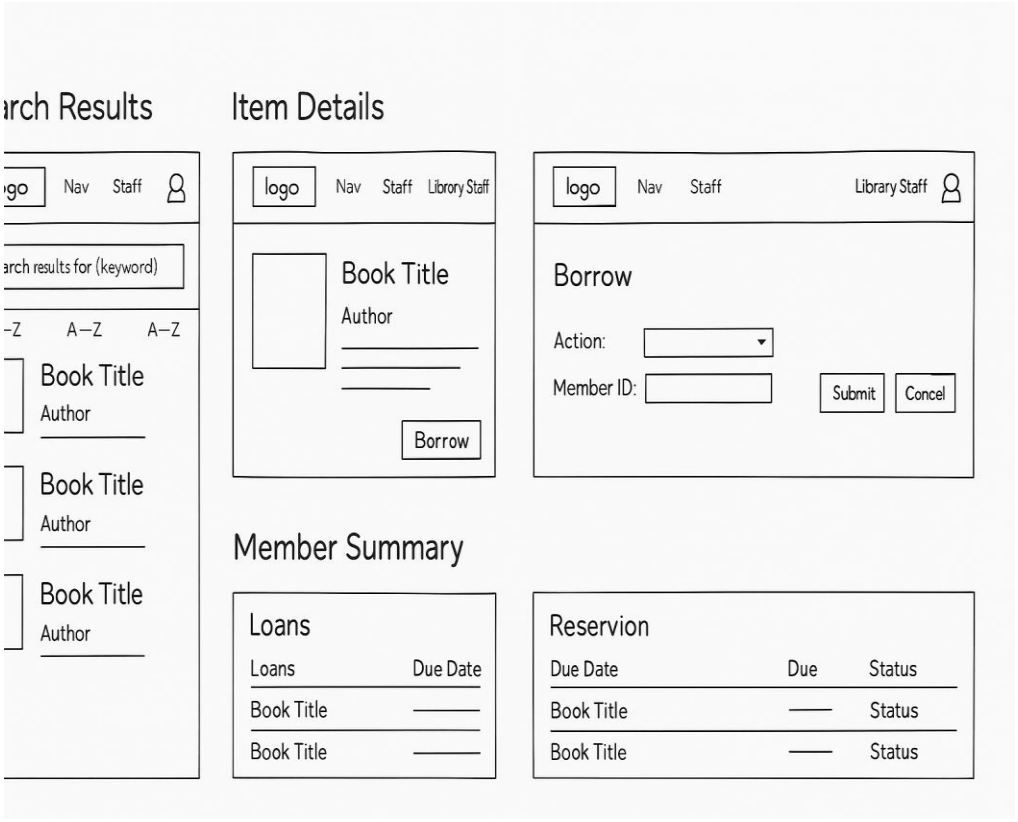


Figure A3: Search Results Wireframe (created using Canva, 2025).

Appendix B: Figma Design (high-fidelity wireframes)

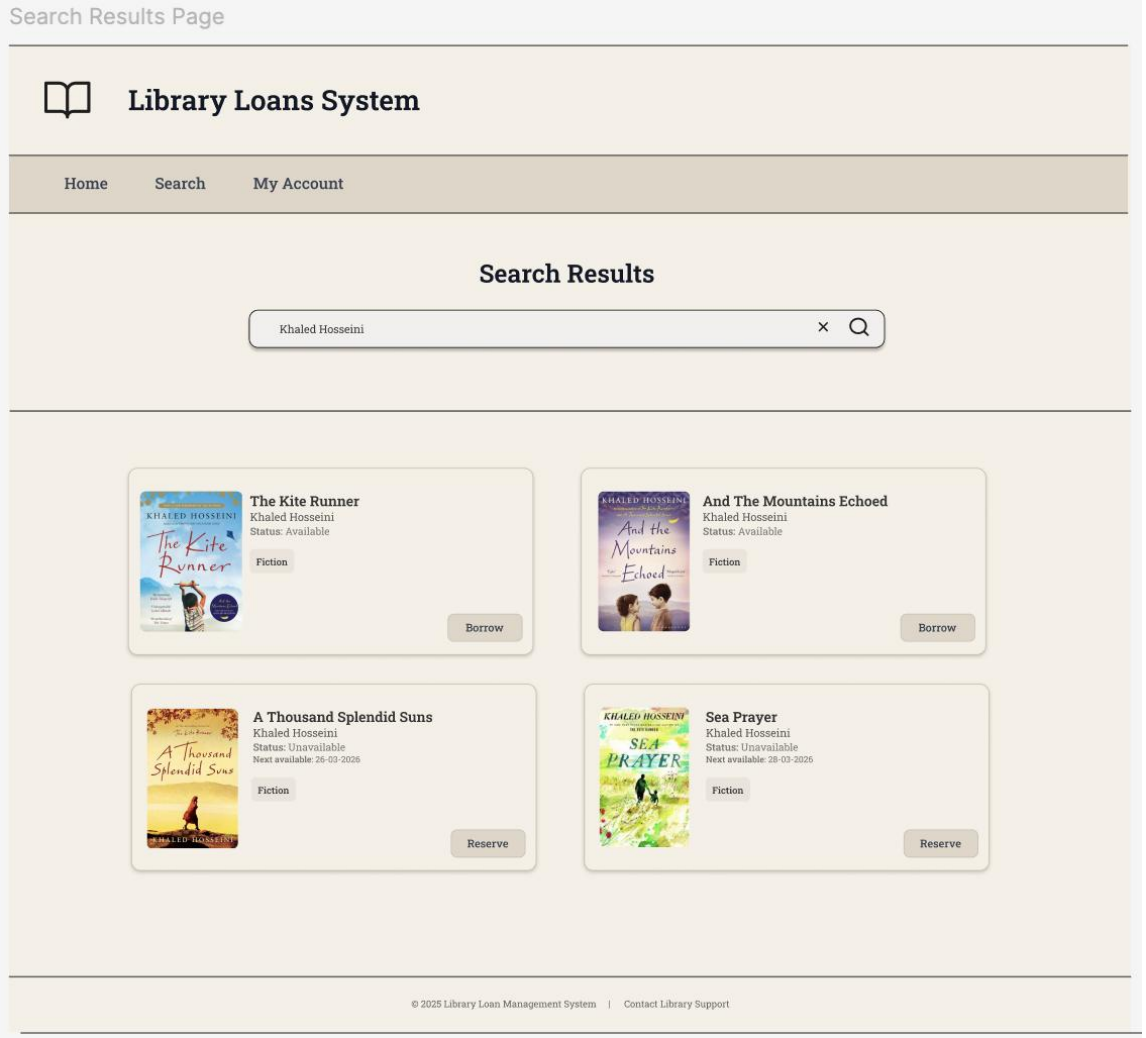


Figure B1: Search Results Wireframe (created using Figma, 2024).

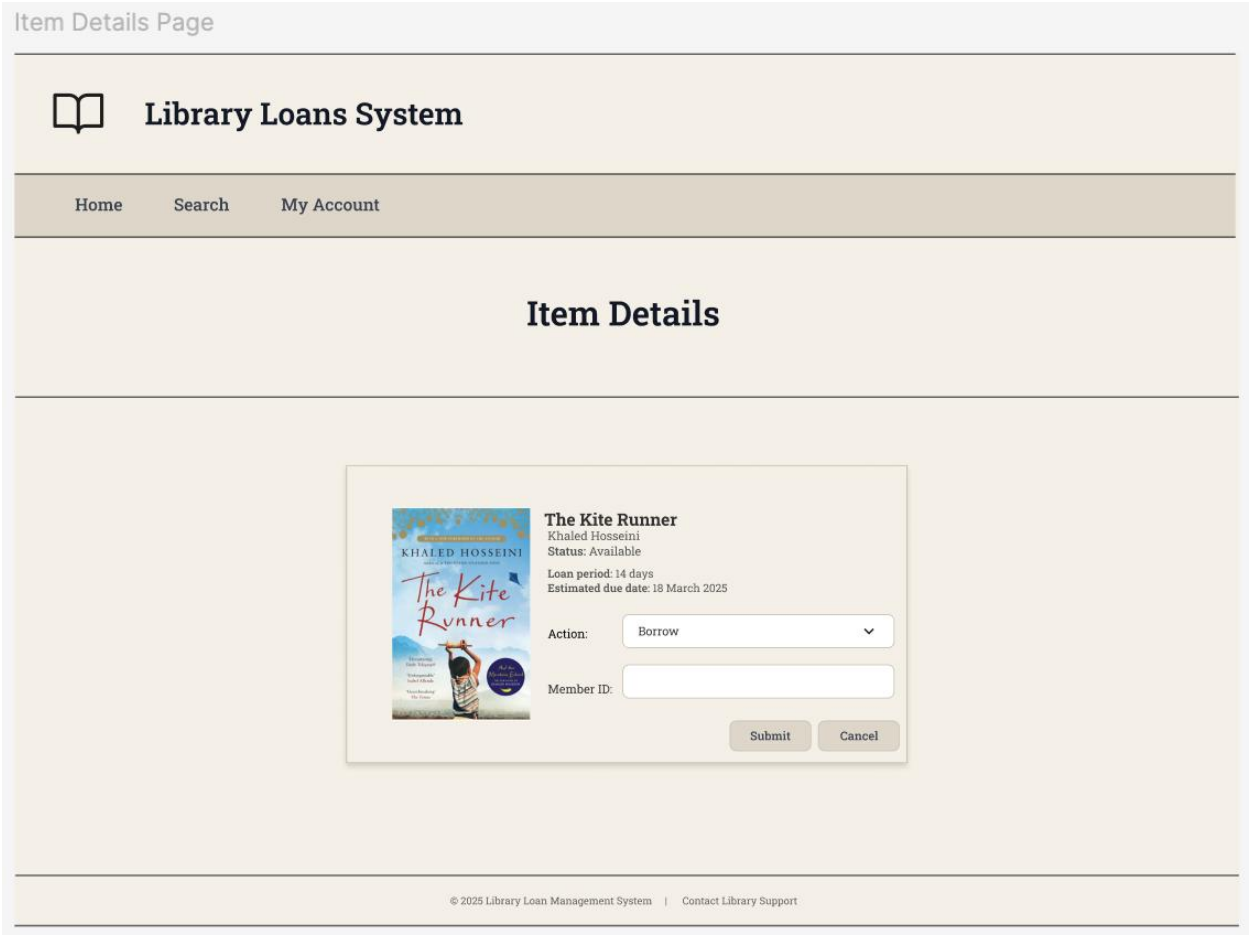


Figure B2: Item Details Wireframe (created using Figma, 2024).

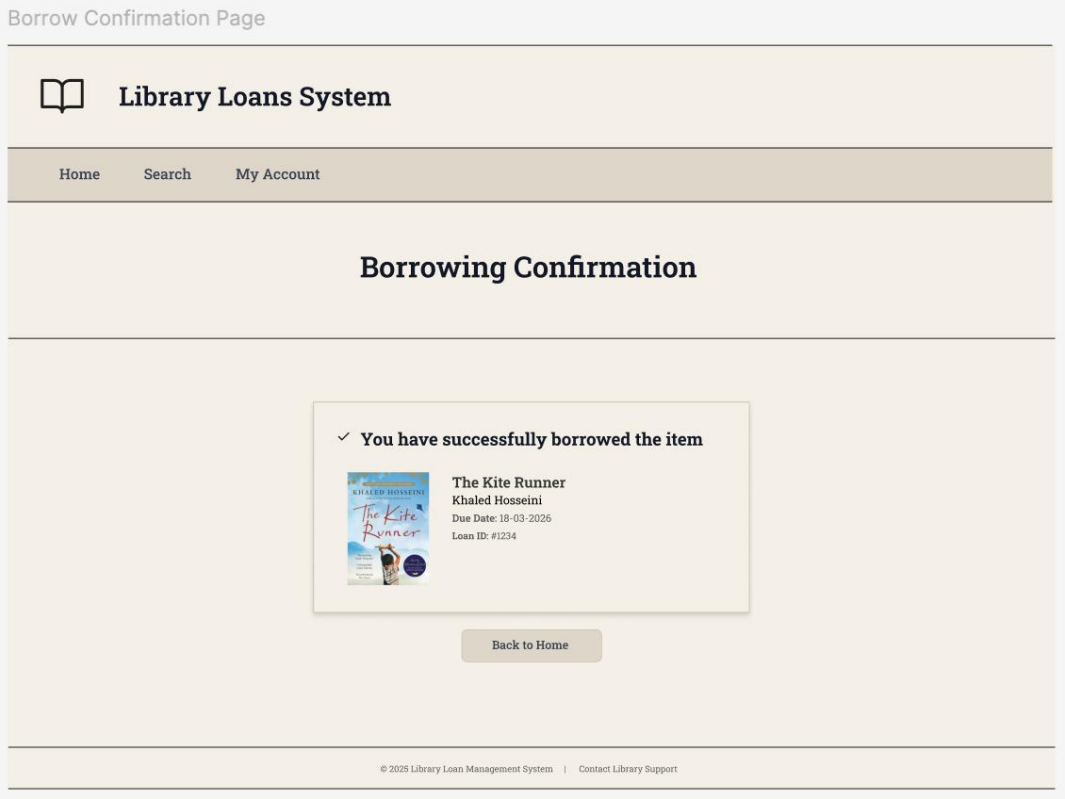


Figure B3: Borrowing Confirmation Wireframe (created using Figma, 2024).

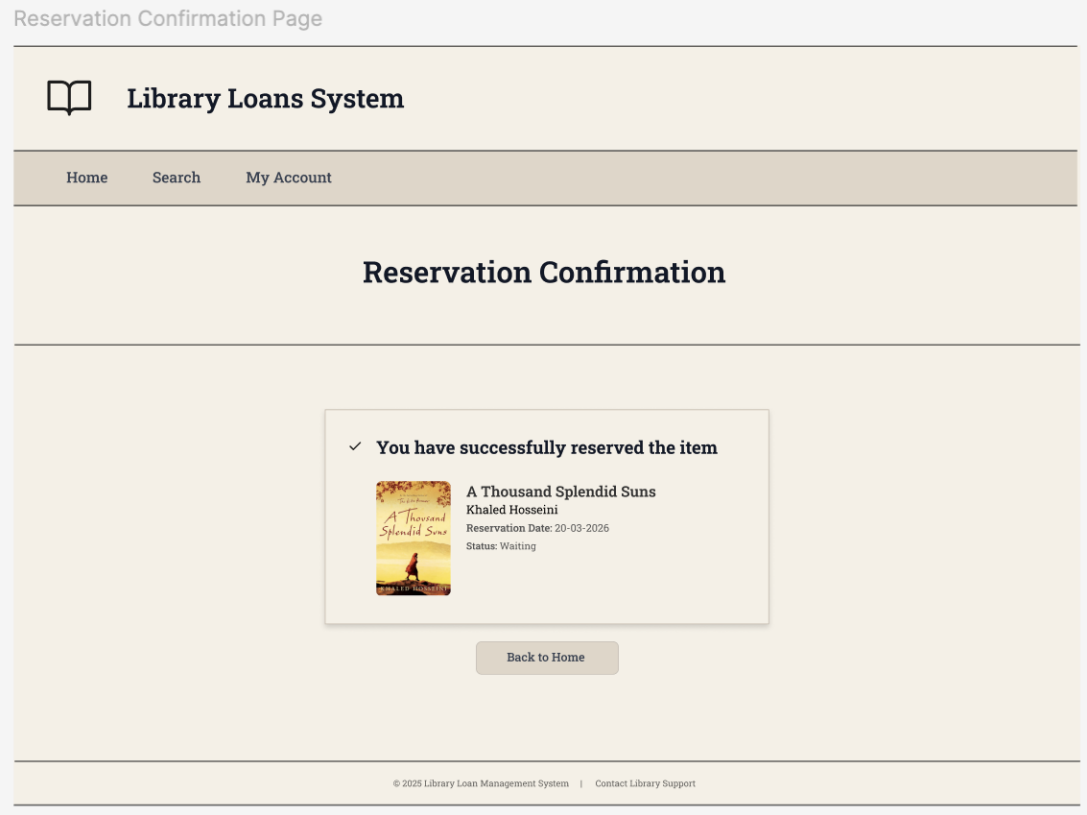


Figure B4: Reservation Confirmation Wireframe (created using Figma, 2024).

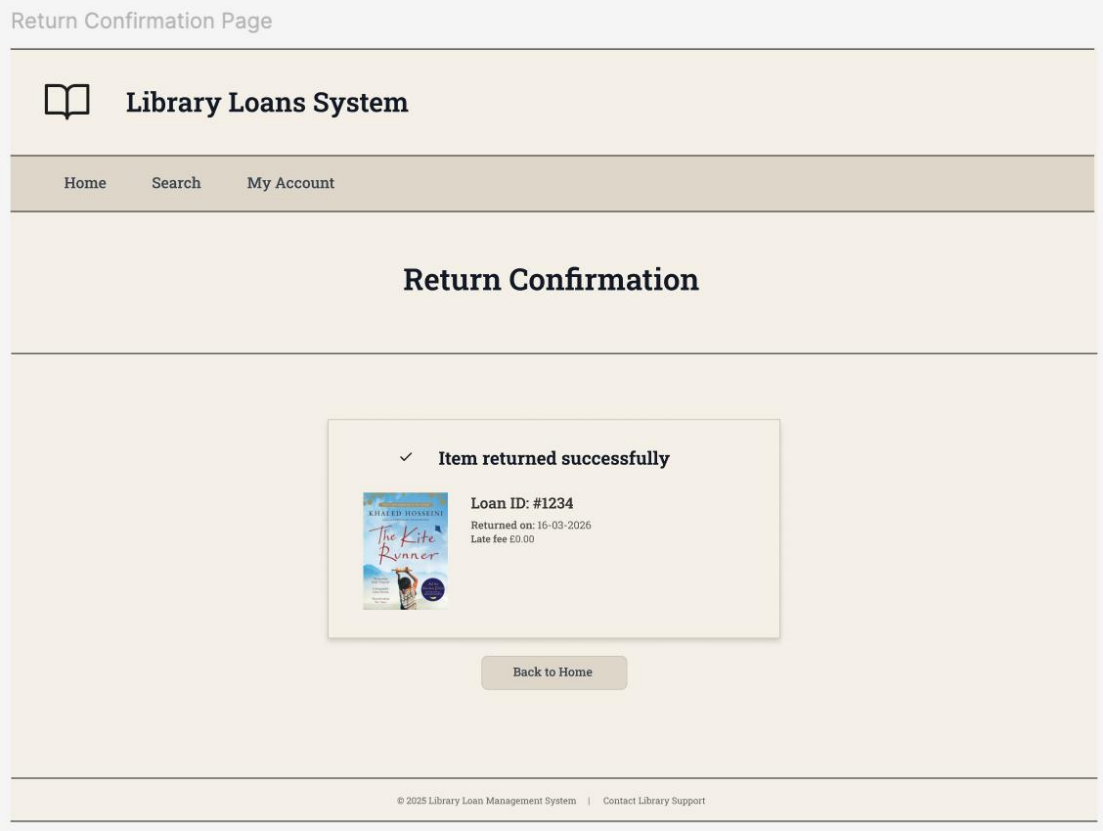


Figure B5: Return Confirmation Wireframe (created using Figma, 2024).

Group Contribution Reflection

This project was completed by the group, but the workload for Deliverable 2 was not evenly shared. Most of the planning, documentation, and technical design work was completed by me. At first, I worked mostly alone because one assigned group member did not contribute to the deliverable.

Later, three additional members joined our group after not having an assigned group before. Of these three, one member made a significant contribution by improving the Gantt chart, refining the wireframes, helping complete some planning tables, and taking part in the task allocation section. The other two members did not provide meaningful contributions, even after they were assigned tasks and asked multiple times if they could help. As a result, the remaining work was completed by myself and the one contributing member.

My contribution covered nearly the entire deliverable, including the Software Requirements section, stakeholder identification and analysis (including the Stakeholder Analysis Matrix, Power-Interest matrix, and Salience Model), identification and prioritization of the feature set using the MoSCoW method, responsibility and task allocation matrices, database design (conceptual, logical, and physical ERDs), SQL schema creation (DDL), sample data inserts (DML), example SQL queries, Planning and Design: Web Services (API endpoint design, protocol specification, validation rules, and justification), overall document structure, figure and appendix organization, captions, formatting consistency, and final assembly to meet assessment and submission requirements.