Digital Library Specification Document

PennWest California University

CMSC 4900: Senior Project

Specification Document

Dr. Chen

11 / 14 / 2025

Group Members:

Joshua Watson

Camron Mellott

Luke Joseph

Instructor Comments/Evaluation

Table of Contents

Abstract.....5

Document Description.....6

Purpose and Use.....6

Intended Audience.....6

System Description.....7

Overview.....7

Environment and Constraints.....8

End user Profile.....8

User Interaction.....8

Hardware Constraints.....8

Software Constraints.....9

Time Constraints.....10

Cost Constraints.....10

Other Concerns.....11

Acceptance Criteria.....11

Testers.....11

Criteria for User Acceptance.....12

Integration of Separate Parts and Installation.....12

System Modeling.....13

Functional: Use Cases and Scenarios.....13

Normal Scenario.....

Exception Scenario.....

Dynamic State chart.....

Dataflow.....17

Components / Tools Needed.....18

Appendix: Glossary of Terms.....19

Appendix: Team Details.....21

Appendix: Writing Center Report.....22

Appendix: Workflow Authentication.....23

Abstract

The Digital Library is an intelligent library management system designed to help users organize, discover, and track their reading journeys. The system allows users to manually upload their personal book catalog, scan ISBN barcodes for easy entry, and receive AI-powered book recommendations based on their individual interests. Users have the ability to place their books in 3 different categories: “Want to Read”, “Currently Reading”, and “Completed”. This specification document serves as a comprehensive technical blueprint for our development team.

Description of Document

Purpose and Use:

The purpose of this document is to establish the scope and requirements for the development team to use as an outline. The document provides detailed descriptions of the project's functionality. One aspect that will be described is the acceptance criteria. For this project, the acceptance criteria will go as follows: A user can: log books under the three defined categories, scan books in through ISBN codes, create a profile to save progress.

Intended Audience:

The intended audience of this document is the members of the development team and the client. The client will inspect this document to ensure that the project will meet all of their requirements. If the client determines the specifications laid out in this document, it will need to be revised to find agreeable terms between the client and the development team.

System Description

Overview:

The Digital Library Application is designed to give users a convenient and interactive way to manage their personal reading experience. The main function of this Digital Library is to provide users with a space to log their books that they are reading or have already read. It will also act as a tool to manage and organize their reading history. Users will be able to mark books as “Currently Reading” or “Completed”.

This application will operate as both a web and IOS mobile platform, allowing users to access their library on any iOS device, such as an iPhone or iPad. This application will also include a recommendation engine that will make use of backend algorithms to analyze user reading habits and suggest new books personalized to each user's interests.

The user interface and application logic will be built using JavaScript for web use, and Capacitor to allow for support on both iOS, Android, and web platforms. Backend operations, including data communication, authentication, and integration with the SQL database, will be managed by a server-side React based framework such as NextJS, while the recommendation engine will run separately using Python machine learning libraries.

Not only will this application track books, but the user will also have the option to record notes and reflections for each book, which will make the library more personalized to each user. The combination of these features will provide users with an organized, engaging, and user-friendly platform.

Environment and Constraints

End User Profile:

The majority of End users for The Digital Library application will be avid readers and users of smartphones. The users of the app will need to have access to an IOS device supported by the application and will also need to know how to use basic applications on said device. An internet connection will also be required to use aspects of the application. As long as a user has a supported device and an internet connection, they will be able to use all services of the application.

User Interaction:

The user will interact with software by using a supported device through a screen. They will be able to add books to their applications that they have read. Users will also be able to search through a database of many books using the application. They can then add these books to their page. There will also be an artificial intelligence recommendation feature for users to interact with when searching for a new book.

Hardware Constraints:

The hardware constraints for the Digital Library application require users to have access to a compatible device capable of running the application, such as a phone, tablet, or computer. The device must also be connected to the internet for both the mobile app and the web version.

The mobile version of the Digital Library will require a device running IOS 16 or higher, and Android version 6 or higher. The web version will just require access to a modern browser such as Google Chrome, Microsoft Edge, Safari, or Firefox. Devices should also have sufficient storage in order to install the app and allow for adequate processing power to efficiently handle the database communication and recommendation features.

Devices will also need a stable internet connection due to the system relying on cloud-based data storage. This will be necessary for users to sync their data, access their personalized library, and receive book recommendations. Some features like book recommendations and synchronization across devices may not be available without internet access.

Software Constraints:

The software constraints for the Digital Library are mainly related to the operating systems, frameworks, and programming languages used to develop and run the system. The application will be designed to operate on multiple platforms, such as IOS, Android, and web browsers.

JavaScript will be the primary language used to build the system. We will be utilizing a cross-platform native runtime called Capacitor to easily allow us to build for web and mobile use.. The framework depends on Node.js and npm for package management and backend integration. The backend server will utilize Express.js running with the Node.js environment in order to manage requests, authentication, and data transfer between the frontend and the database.

To ensure secure and efficient data handling, an SQL database such as MySQL or PostgreSQL will serve as the central storage for user information, book progress, and notes. The recommendation system will rely on machine learning libraries, which will require the appropriate runtime environment and dependencies to function properly. For users, IOS 16 or higher will be required to ensure all the applications features run efficiently and properly.

Time Constraints:

The time constraints for the Digital Library Application are based on the academic semester schedule. There will be about fifteen weeks until completion. During this time, team members will spend a few hours each week planning, developing, testing, and documenting the project. The team will establish goals throughout the semester to stay on track and monitor

progress. Some limitations to these time constraints are additional coursework and responsibilities for each member that limit the total time available for the project.

Cost Constraints:

The cost constraints for the Digital Library application will mainly involve software development and hosting requirements. This project will utilize open-source tools and frameworks, such as React Native, Node.js, Express.JS,NextJS, MySQL, Capacitor, and Visual Studio Code so there will be no direct cost associated with software licensing.

Although, cloud hosting and data storage may be initially free during testing through the use of services like AWS, Azure, or Google Cloud. The cost may increase if the application scales or requires additional storage capacity.

There will be no additional hardware purchase due to development being performed on existing personal computers. The main expense is expected for hosting and optional premium cloud features during deployment and testing.

Other Concerns:

The other concern with the Digital Library application is the user's internet access. A reliable internet connection is required to fully utilize the features of the Digital Library application. Without this stable connection users may experience downtime or connection loss and will only have access to previously downloaded content.

Acceptance Testing

Testers:

The testers and quality assurance team of the Digital Library Application will be comprised of all 3 members of the development team creating the digital library project. Each member of the team will perform their own methods of testing each aspect of the project. This will include testing the software on appropriate devices. They will also be testing the functionality and performance of each individual feature of the project and all of their related systems. The team will continuously test these aspects of the project throughout all development of the project which will ensure stability throughout development and in the final product.

Criteria for User Acceptance:

Successful user acceptance for the Digital Library application will occur when users are using all features of the application. These features include the user being able to easily use the application to track the books that they are reading and have already read. It also includes users searching the database to find more books to read or add to their page.

Integration of Separate Parts and Installation

The Digital Library software will be installed on the main server, which will handle all communication between the user interface and the database. The server will connect to the online database system, where all user accounts are stored. The front-end website or mobile app will communicate with the server through a free-web-based application. All digital content, such as search operations, and user permissions, will be managed on the server to ensure fast, centralized control.

System Modeling

Functional: Use Cases and Scenarios:

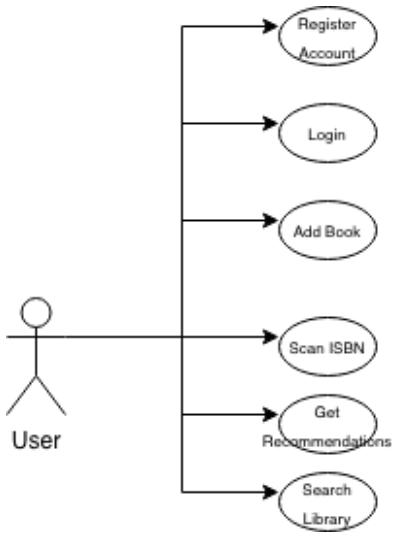

Figure 1: Flow of Digital Library App

Figure 1 displays a general Use Case Diagram for when a user first opens the application. Register Account will allow users to create an account to track progress across sessions. Login will allow users to enter their credentials to pick up a saved session. Add books will allow users to add a specific title to one of the 3 defined categories. Scan ISBN will allow for the direct scanning of ISBNs from physical books. Getting recommendations will use AI to give titles similar to ones that the user has entered. Search library will allow the user to search for a specific title without adding it to their personal library.

Run Scenario

- User enters app
- User chooses one of the following options:

- Register Account
- Login
- Add Book
- Scan ISBN
- Get Recommendations
- Search Library

1. If Register Account is chosen, an account registration page will be loaded on to the screen
2. If Login is chosen, the user will be directed to a page that allows them to enter their credentials
3. If Add Book is chosen, the user will be directed to a page that has a search bar. Upon entering the title of the book, they would like to add and hit enter, a list of books will appear for selection.
4. If Scan ISBN is chosen, the user will be directed to a page that accesses their camera, allowing them to scan the ISBN barcode of a physical book.
5. If Get Recommendations is chosen, the user will be directed to a page that utilizes AI to analyze their current library and suggest similar titles.
6. If Search Library is chosen, the user will be directed to a page with a search bar. Upon entering the title of a book, they can view, but not add, books.

Class Diagrams:

The following sections contain the diagrams and descriptions of entity classes to control classes, boundary classes, and association classes. The entity classes will represent real world objects such as the user and books. The control classes contain the logic and algorithms used for the search engine as well as the recommendation logic. The Boundary classes deal with external

systems such as the ISBN scanner and API services. The association classes represent the relationships between entities.

Class categories:

The system consists of several classes organized into different categories:

- Entity Classes – The core data objects. Objects such as the User, Book, and UserBook all store data critical to the operation of the application.
- Control Classes – The business logic of the application. The SearchEngine and RecommendationEngine objects give the system key functionalities.
- Boundary Classes – The external systems. The ISBNScanner and ExternalAPI classes will communicate with external services to provide their functionalities to the core system.

Class Diagrams:

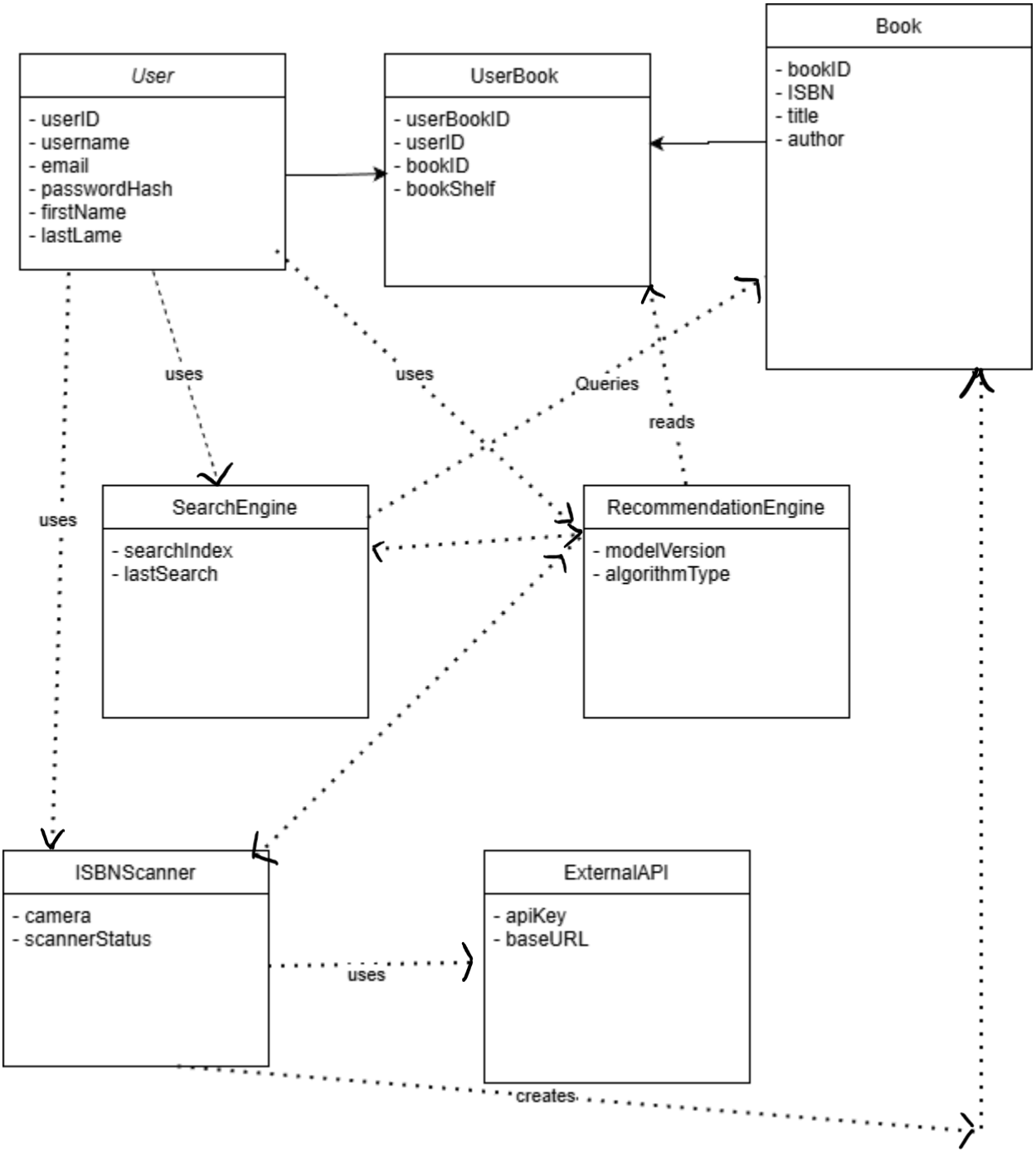

Figure 2: Class Diagrams

Class Descriptions:

User Class

The user class holds the user's credentials including username, password, and email, as well as the user's name, all variables are of string type. All information is used to track progress across sessions.

Book Class

The book class holds book information including the bookID, ISBN number, title, and author of the book, all variables will be of string type.

ISBNScanner class

The ISBNScanner class contains the camera stream, which will use the user's phone camera to scan ISBN barcodes. Additionally, the class will have an enum variable that determines whether the scanner is currently scanning something or not.

SearchEngine Class

The SearchEngine class contains the search index object for fast and efficient searching. It will also contain the last search attribute as a string variable to keep track of the most recent search.

ExternalAPI class

The ExternalAPI class will be used to contain various API interfaces that will be used during development. It will have an attribute to keep the API key, the base URL for the API with both variables being strings. As well as keep track of how many requests are being made to the API by keeping an integer variable to count requests.

RecommendationEngine class

The RecommendationEngine class will be used to analyze the user's reading patterns and generate personalized book recommendations. It will contain a string to denote the algorithm type being used in our machine learning model. In addition, it will contain a string to denote which version of the machine learning model we are using.

Dynamic State Chart & Transitions

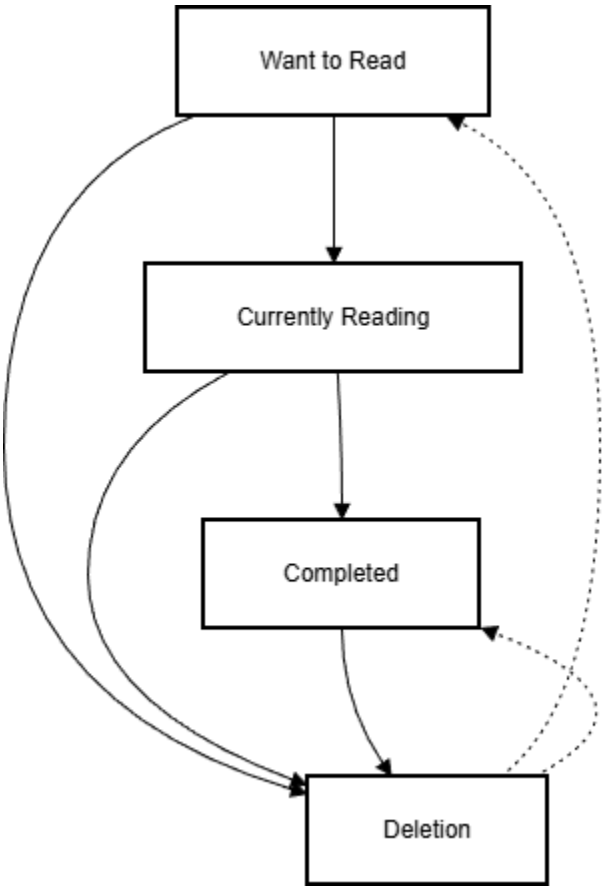

Figure 3: Dynamic State Chart

States: The following will describe the lifecycle of a UserBook object as it moves through different stages

- Initial state: When a user adds a book to their library through either manual entry or ISBN scanning, the UserBook is created

- State 1- Want to Read: The book is added to the user's "Want to Read" shelf. The UserBook remains in this state until the user changes to any of the other states
- State 2 – Currently Reading: When the user starts reading, they can mark a book in their library accordingly. The UserBook will be moved to the user's "Currently Reading" shelf
- State 3 – Completed: Once the user finishes the book, the user can manually mark the book as finished. The UserBook object will be moved to the user's "Completed" shelf
- State 4 – Deletion: If the user chooses to permanently delete a book, the UserBook record is permanently removed from the database.

Transitions:

When the application is loaded onto the user's mobile device, the initial state will always be the home screen. From The home screen they will have various options to choose from. The user can choose to:

- Register an account
- Login
- Add a book (manual search)
- Scan ISBN (automatic search)
- Get Recommendations
- Search Library

The user can loop through the options as many times as they would like.

Dataflow Diagrams

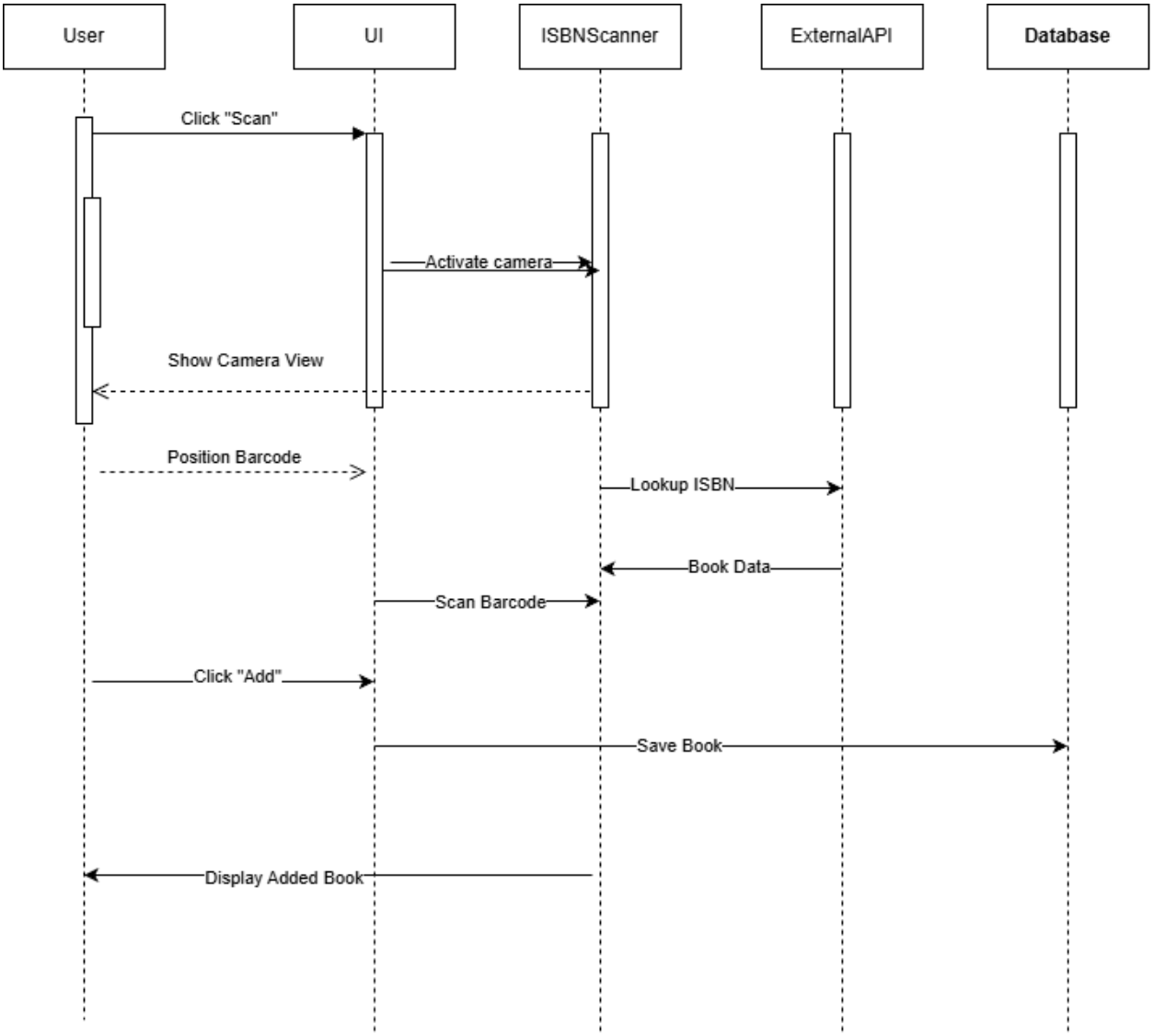

Figure 4: Scan ISBN use case diagram

Description: The ISBN use case shows a common interaction between user and the ISBN Scanner. The diagram shows the interactions from user to UI, UI to the ISBN Scanner, and the ISBN Scanner to the ExternalAPI. It also displays how book data will be stored in the database upon scanning.

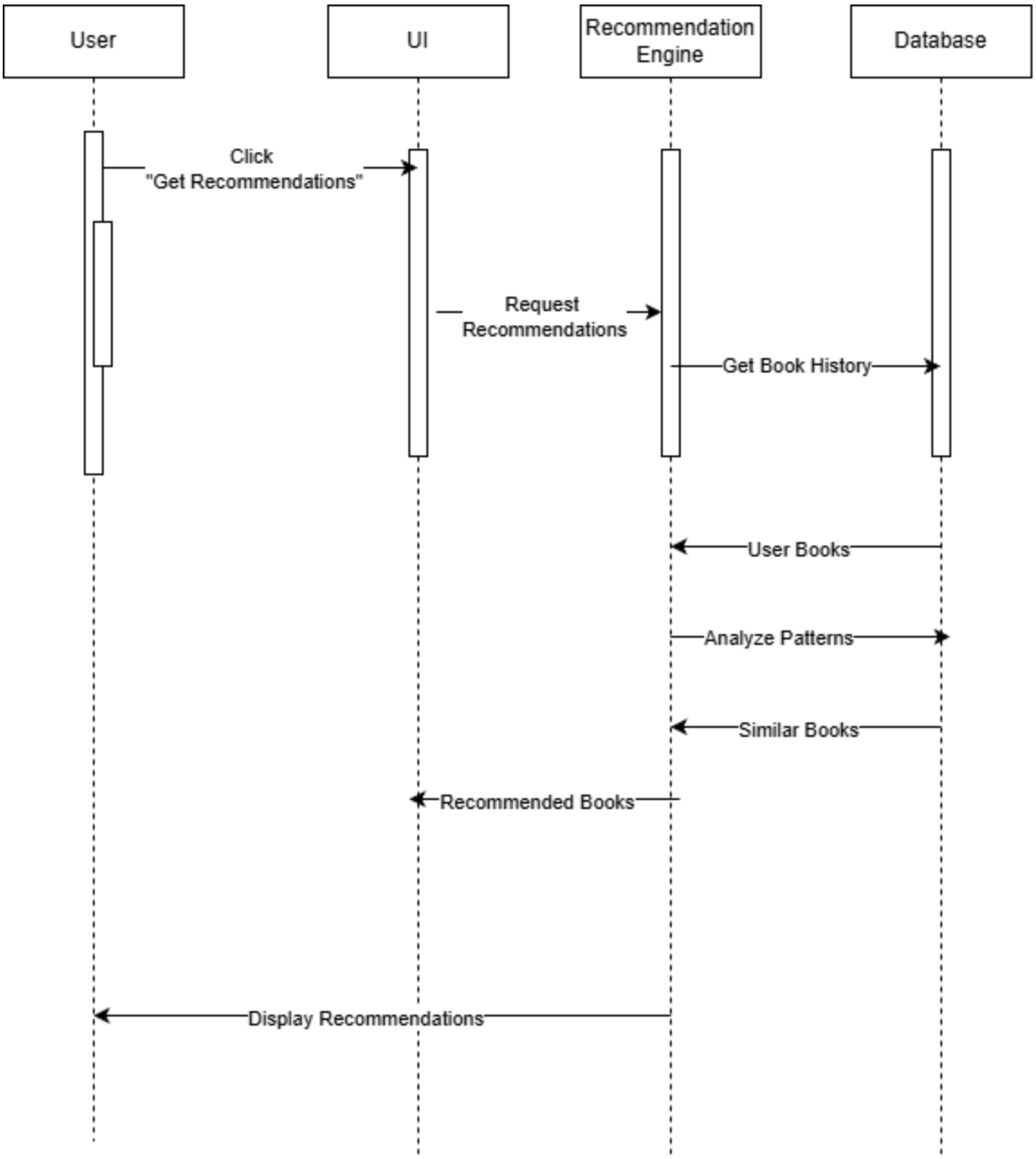

Figure 5: Find recommendations use case diagram

Description: The recommendations use case diagram shows what happens when a user asks the system for book recommendations. The UI will interact with the Recommendation Engine, which will access the database to gain insight into the user's book preferences. The collected data will be displayed back to the user.

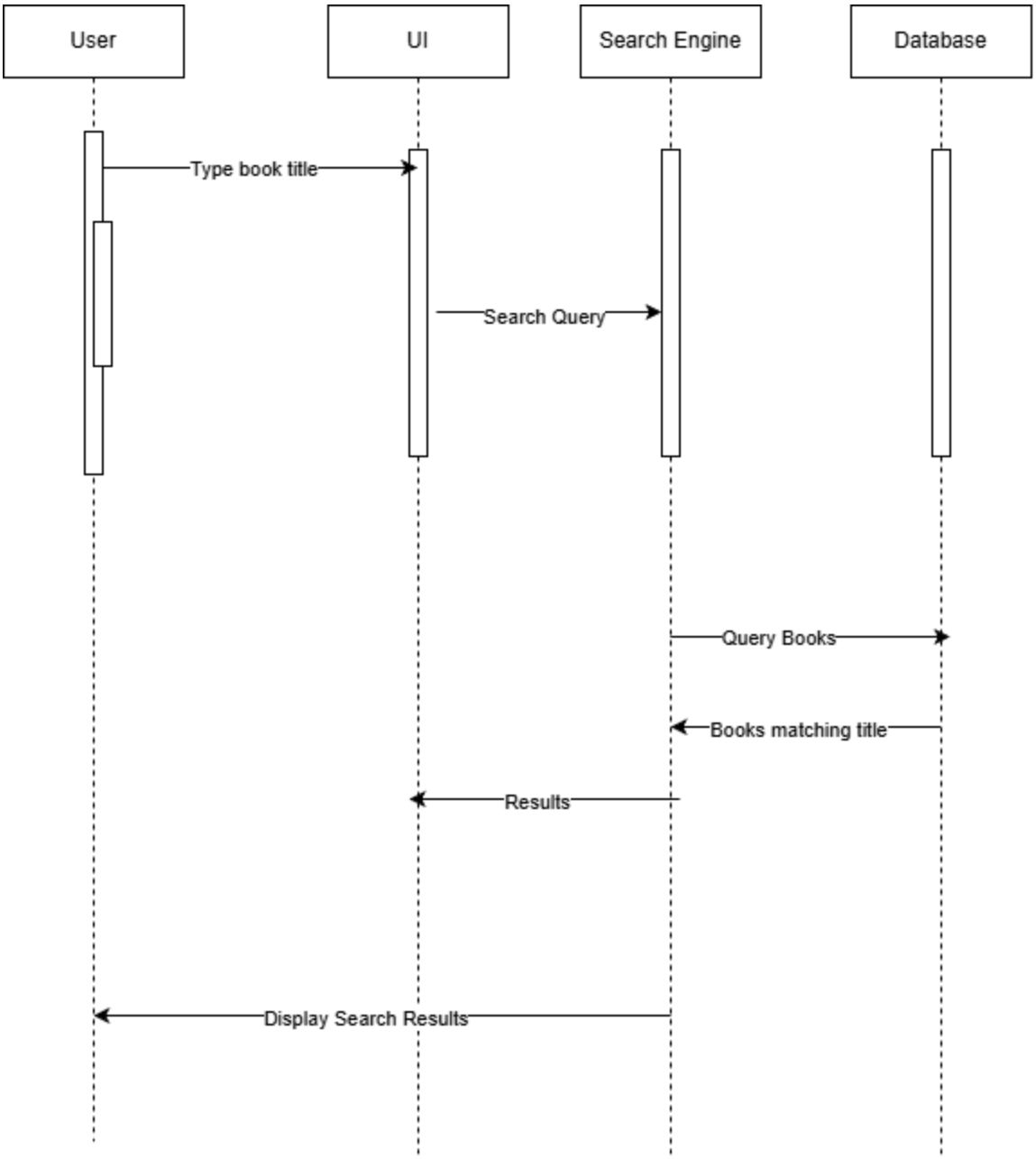

Figure 6: Search Library use case diagram

Description: The Search Library diagram represents one of the most common use cases. The user will enter a book title into the UI. Which will then use the search engine to query the database. The database returns the books matching the title to the user.

Components/ Tools Needed

The Digital Library Application will be developed as both a web and mobile platform; it will allow users to log, organize, and track their reading activity while also receiving recommendations personalized to them.

The software constraints for the Digital Library application are centered around the operating systems, frameworks, and programming languages required to develop and run the system. The application will be developed in Visual Studio Code, utilizing JavaScript frameworks such as ReactJS, NodeJS, and using Capacitor as a mobile wrapper. The frameworks we will be using will allow for easy communication of data back and forth between the server and client.

To ensure secure and efficient data handling, an SQL database such as MySQL or PostgreSQL will serve as the central storage for user information, book progress, and notes. The recommendation system will rely on Python-based machine learning libraries, such as scikit learn or TensorFlow Lite, which will require the appropriate runtime environment and dependencies to operate properly. These Python processes will be integrated with the JavaScript application through API communication or scheduled data processing.

Users accessing the web version will need a modern browser such as Google Chrome, Microsoft Edge, Safari, or Firefox. For mobile users, devices running IOS 16 or higher will be required to ensure the application performs efficiently and supports all intended features.

Appendix I: Glossary of Terms

AI Recommendations – Automated book suggestions generated using algorithms based on user interests and reading habits.

API – Application Programming Interface. Allows different parts of the system (app, server, and database) to communicate.

Backend – The part of the system that processes data, handles authentication, and communicates with the database.

Book Categories – Organizational labels used in the app: "Want to Read," "Currently Reading," and "Completed."

Capacitor – A mobile wrapper that allows web applications to seamlessly be repackaged for mobile devices.

Cloud Hosting – Online storage and server services used to run the application and store use data.

Database (SQL) – Structured storage for user accounts, book lists, notes, and progress.

Frontend – The part of the software users interacts with on their screen.

IOS – Apple's mobile operating system used on iPhones and iPads.

ISBN – International Standard Book Number. A unique code used to identify books.

Machine Learning Model – A program that learns user preferences to generate personalized book recommendations.

Mobile Application – A version of the Digital Library that runs on portable devices such as

phones and tablets.

NodeJS – A JavaScript framework used for building the backend of web applications

Notes/Reflections – User-created text entries that can be added to books.

ReactJS – JavaScript framework used for building the frontend web applications.

scikit-learn – A Python machine learning library used to analyze user patterns and help generate book recommendations.

Synchronization – Updating user data across multiple devices so everything stays consistent.

TensorFlow Lite – A lightweight machine learning library used to power recommendation features.

User Profile – A saved account where a user's books, progress, and notes are stored.

Web Application – A browser-based version of the app that does not require installation.

WIFI – Wireless networking technology required for syncing data, searching books, and receiving recommendations.

Visual Studio Code - Free code editor

Appendix II: Team Details

Although all members contributed to the completion of this document, Camron Mellott was the leader. The following are the contributions from each team member.

Josh Watson:

- Abstract
- Description of Document
    - Purpose and Use
    - Intended Audience
- Environment and Constraints
    - Cost Constraints
    - Software Constraints
- System Modeling
    - Functional: Use Cases and Scenarios
    - Class Diagrams
    - Class Descriptions
    - Dynamic State Chart & Transitions
- Dataflow Diagrams
- Components/Tools needed
- Appendix I: Glossary of Terms
- Appendix II: Team Details

Luke Joseph:

-

Camron Mellott:

- System Description:
    - Overview
    - Hardware Constraints

    - Software Constraints
    - Time Constraints
    - Cost Constraints
    - Other Concerns
- Components / Tools Needed
- Appendix I: Glossary of Terms
- Appendix II: Team Details
- Appendix IV: Workflow Authentication

Appendix III: Writing Center Report

Cal U Writing Center Report

Client: Camron Mellott, Josh Watson, Luke Joseph

Staff or Resource: Chayse L.

Date: November 14th, 2:00-2:30

Did the student request that the instructor receive a visit report: Yes

What course was serviced by this visit?: CMCS4090 - Sen. Proj. I: Software Engineering

What goals were established for this tutoring session: Proof-read for errors and completion.

How did the process of this consulting session address the established goals:

You came in with your group for your computer science class, and wanted a simple overview of the paper. These are a few comments from what I observed and we discussed: I think that the content was good and you did a good job in explaining what each section was supposed to go over. I think for consistency's sake, make sure your titles are all in Times New Roman, but they do not have to be 12 points like the body text. Make sure the spacing above titles is consistent, either have them or not. When punctuation is used with quotation marks, make sure the punctuation is inside like this: "cats on trampolines," "dogs on bikes," and "cats meow." Correct logbooks to log books. When discussing the operation system needed to run the library, you should change the iphone and android to iphone or android. Then make sure that all your

paragraphs are indented for consistency. I think we also discussed that the glossary section needs the entries switched and then make sure your reference page is in the correct format. Outside of those details, the paper looked good and just make sure you finish up the rest of the sections missing.

Appendix IV: Workflow Authentication

I attest that I have completed all the work written within the “Appendix: Team Details” section.

Signature: Camren Mellett

I attest that I have completed all the work written within the “Appendix: Team Details” section.

Signature: Luke Joseph

I attest that I have completed all the work written within the “Appendix: Team Details” section.

Signature: Joshua Watson

Appendix V: Works Cited

Capacitorjs. (n.d.). *Capacitor by Ionic - cross-platform apps with web technology*. Capacitor. https://capacitorjs.com/

Mobile application development: Web vs. native: Communications of the ACM: Vol 54, no 5. (n.d.). https://dl.acm.org/doi/10.1145/1941487.1941504