|  |
| --- |
|  |
| BOOK COLLECTION MANAGMENT SYSTEM  Enhancing Personal Library Management Through Structured Data Insights and Optimized Relationships |
| |  |  |  | | --- | --- | --- | | Mavericks | 9/16/24 | IA502001 Studio 1 | |

Table of Contents

[Executive Summary 2](#_Toc177571475)

[Initial Analysis of the Problem Domain 2](#_Toc177571476)

[System Development Overview 3](#_Toc177571477)

[Design Analysis Process 4](#_Toc177571478)

[System Development Life Cycle (SDLC) Followed: Agile Approach 5](#_Toc177571479)

[Functional Requirements and Non-Functional Requirements 6](#_Toc177571480)

[Actors 6](#_Toc177571481)

[Use Cases 7](#_Toc177571482)

[Use Case Diagram 9](#_Toc177571483)

[User Stories 10](#_Toc177571484)

[Entity-Relationship Diagram (ERD) 11](#_Toc177571485)

[Table Designs – Data Dictionary 14](#_Toc177571486)

[Contributions: 19](#_Toc177571487)

[References 21](#_Toc177571488)

[GITHUB 21](#_Toc177571489)

# Executive Summary

The name given to the system is Book Collection Management System (BCMS) which is aimed at enabling users to manage their collections. The overall intent of the system is to enable the user to store the books and follow the progress of books that they are reading as well as the books they have loaned to friends and the due dates required for the books to be returned. SQLite will be used for the actual storage of data and as a result, BCMS will be designed to be a simple personal book manager, in which the emphasis will not be placed on elaborate functionality. In the process of developing the given equipment, the client-oriented approach was employed, and the acceptance of the system means simplicity, adaptability, and expandability.

(Libib, n.d.)

(Library Management System Project | Software Development, 2024)

# Initial Analysis of the Problem Domain

* **Entities Identified:**
  + **USER**: Represents the user who possesses the books as well as takes the responsibility of organizing his/her collection.
  + **BOOK**: Represents each book in the collection as a separate record.
  + **GENRE**: Represents several genres to which Books belong.
  + **FRIEND**: Represents friends who can borrow books.
  + **LOAN**: Represents borrowed books by friends, keeping records like due dates.
  + **READING\_PROGRESS**: Represents how far the users have read through their books.
* **Problem Domain Boundaries:**
  + It aims at organizing personal bookshelves and following up on books being read, and at handling book loans to friends.
  + It has no capability or components for any type of monetary exchange mechanism, more comprehensive friend management provisions, or any sort of connection to another book or library management system.
  + The scope is restricted to the entities and the relationships required for personal collection management, reading tracking, and book loaning.

# **System Development Overview**

* **Initial System Proposal:**
  + The original schema included the following entities:
    - **UserData:** User management with fields such as UserID, UserName, Email address, and Password details.
    - **BookData:** All the records that are to be stored for the management of the books are names such as Title, Author, category, DateOfIssue, ISBN, Description, CoverImage, and Publisher.
    - **ReadingProgressData:** An approach about reading, and the development of reading.
    - **LoanData:** Other attributes that can be of use while relating to book loans include LoanDate, DueDate, and ReturnDate.
    - **RecommendationData (Dynamic):** Products based on attributes like Genres, Ratings, and the related books.
    - **CollectionStatistics (Dynamic):** Developed attributes namely total books read and currently reading, reading trends of the user’s data.
* **Changes Made:**
  + **Removed Attributes:**
    - **A table named BookData shows categories that have been deleted and Status from the invalid table ReadingProgressData.**
    - **Dynamic Data Entities**: Such dynamic formulas were not used to develop the omittance of RecommendationData and CollectionStatistics.
  + **Added Attributes:**
    - **TotalBooksRead** and **CurrentlyReading** were added to **GenreData** for clearer insights into reading habits.
  + **Modified Relationships:**
    - Modified many-to-many relations which were redesigned into associative entities to ensure space and caliber of queries.
* **Design Choices:**
  + Limited to the basic functions in a library, which may include orders, loans as well as the progress of the readers.
  + Advanced the schema by decreasing the optional attributes of the schema, also those that are generated dynamically.
  + Reserved a strict and constant organization schema to stay as fluid and adaptable as needed.
* **Explanation for Differences from Proposal:**
  + **Simplified Structure:** The previous attributes like Category and Status were removed as they are static and few dynamic data entities also were removed to make the data set easier.
  + **Core Functionality Focus:** Erased fields TotalBooksRead and added fields like CurrentlyReading to make clear that data modifications are possible.
  + **Consolidated Design:** Made sure that the schema does not become complicated, and it does not diverge from the laid down goals and objectives, this minimized on the creation of more complexities and enhanced extendibility.

# Design Analysis Process

* **Information Gathering:**
  + Gathered the requirements from the potential users on how they use the application for personal book collection administering, reading progress, and controlling the books loans.
  + Listened to such systems to establish the critical characteristics of the developed system.
* **Brainstorming:**
  + Engaged the team members in collaborative meetings to determine the most important features, attributes, and functions that the System should contain.
  + Entire features and functions were designed in a priority obeying users’ expectations, minimizing complexity of interface and extra features, MFC color, etc.
* **User Stories:**
  + Developed as-built user stories that provide requirements from the user’s point of view. Some of them were book listing, recording the reading session, and being able to get a notification on the return dates of books.
  + Iterated on the user stories till their features could match the expectations and requirements set up by the users.
* **Use Cases:**
  + Derived a set of use cases to describe more detailed scenarios with the User to the System; basic operations are managing the collections, tracking the progress, loaning function, reporting, etc.
  + Utilized these use cases to identify all functional requirements and to confirm that the primary activities of the user were all supported by the system.
* **Diagrammatic Representation (ERD):**
  + Developed an initial Entity-Relationship Diagram (ERD) to identify relations between such entities as UserData, BookData, GenreData, LoanData, and ReadingProgressData**.**
  + Reconfigured the ERD to eliminate the redundant attributes and dynamic data entities to enhance clarity in data organization.
* **Testing and Modification:**
  + Another part of assessing the quality was conducting unit testing and integration testing of the design and of the schema to ensure everything worked well.
  + Modified where applicable, any feedback that was received from various stakeholders and results from testing aiming at improving data clarity, usability, and system.
* **Continuous Improvement:**
  + Maintained the design more open for future additions and changes so that the system can be adapted to the users’ feedback and the new requirements.

(The Digital Transformation of Libraries: How Library Management Systems are Shaping the Future, 2023)

# System Development Life Cycle (SDLC) Followed: Agile Approach

We followed an **Agile** methodology for the development of the **Book Collection Management System (BCMS)** to ensure flexibility, iterative progress, and continuous user feedback throughout the project.

1. **Planning:**
   * Defined the project scope, objectives as well as core functionalities to be delivered by the project.
   * Developed the first/initial list of product backlog which includes: the ability to launch and organize book collections, update progress, and administer books on loans.
2. **Analysis:**
   * Gathered requirements weekly with potential users several times.
   * Created user stories and use cases that depict a variety of users and their requirements or utilization of the application.
3. **Design:**
   * Proposed an initial conception and an ERD to identify the main entities (e. g. , UserData, BookData, LoanData).
   * With the help of feedback, the design was gradually improved by the simplification of the schema and stressing the main important functions.
4. **Implementation:**
   * For database management we used SQLite, integrating the features: of adding/editing/deleting books, reading progress, and loans.
5. **Testing:**
   * Conducted ongoing testing.
6. **Maintenance:**
   * sure to check on the system for any bugs and other problems that may arise.

**Why Our Implementation is Better**

* **Simplicity and Usability:**
  + Thus, by keeping the preexisting schema simple it can be made sure that the system remains easy to navigate through.
* **Reduced Complexity:**
  + We do not burden our structure with superfluous attributes or kinetic objects that may cause errors; data handling is made easy.
* **Core Functionalities Supported:**
  + Presently it affords complete support to core features contributing to book circulation, reading progress, and loan management that was found to respond to each strategic goal.

**Supporting and Justifying the Design**

* **Focus on Data Integrity and Clarity:**
  + It brings out clear relationships of the entities reducing the use of ambiguities hence improving the reliability of data.
* **Efficient and Logical Data Flow:**
  + Manages data connections with opportunities to operate the queries and data processing swiftly and accurately.
* **Maintains Flexibility for Future Growth:**
  + Favors easy extensibility, which means that in the future, new functionalities can be added with little hassles in changing the basic form.

(Lin, 2024) (Breeding, 2020)

# Functional Requirements and Non-Functional Requirements

**Functional Requirements**

1. The system has to include functionalities such as creating books, modifying books and also deleting books from the library.
2. The system of use must allow users to follow and edit the progress of reading every book.
3. A user should be able to classify the books according to genre to simplify the process of searching for a particular book.
4. The system must include the ability to issue books to friends for borrowing and a system of loan status.
5. It must be designed in such a way that it requires the users to receive notification when the due date for the borrowed book is near.
6. The system should come with an option through which users would be able to search books by titles, authors, or categories.
7. The system needs to generate analyses and lists of the habits of reading of the user as well as the books that he or she owns.

**Non-Functional Requirements**

1. The system should be able to allow many users to manage their collections at the same time and a faster rate.
2. **Reliability:** The system should be highly available and must be able to pass from one state to another smoothly.
3. **Security:** It should also provide the means of preserving the data and preventing any unauthorized access to that data.
4. **Scalability:** There should be increments in the number of users and the database through time in the system.

(Library Management 101: A Practical Guide, Second Edition)

# Actors

The system identifies two primary actors:

• **User**: The main user who employs the system to track the books they have, want, or have read. Loans, progress, new books, modifications, and deletions, as well as reading information, can all be accessed by the users.

• **Friend**: A second party, which communicates with the system only through loan transactions and return reminders. In the system, friends are controlled by the user.

# Use Cases

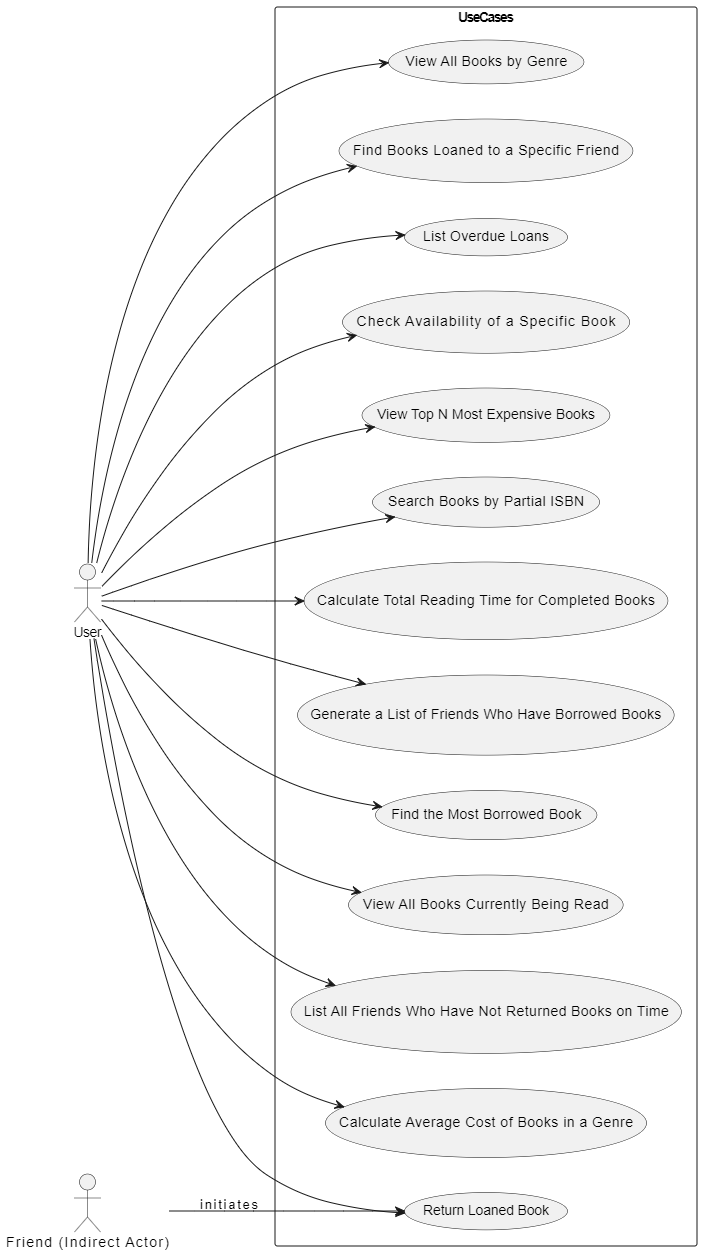
**User**

1. **View All Books by Genre**
   * **Description:** A user opens a section to find out what books fall under a particular category of books, for instance, fiction.
   * **Primary Actor:** User
2. **Find Books Loaned to a Specific Friend**
   * **Description:** A user searches for all books that have been borrowed by a particular friend to identify books that a friend is currently using.
   * **Primary Actor:** User
3. **List Overdue Loans**
   * **Description:** A user realizes all the books borrowed from him/her are due so they can send a message to all friends informing them that he/she needs the books back.
   * **Primary Actor:** User
4. **Check the Availability of a Specific Book**
   * **Description:** A user verifies whether a particular book is on his or her list to determine whether it should be loaned out or read.
   * **Primary Actor:** User
5. **View Top N Most Expensive Books**
   * **Description:** A user has a look at the N priciest books in a collection to evaluate the value of the collection.
   * **Primary Actor:** User
6. **Search Books by Partial ISBN**
   * **Description:** A user searches for books using partial ISBN: This lets the user find books quickly if one only knows a part of the ISBN code.
   * **Primary Actor:** User
7. **Calculate Total Reading Time for Completed Books**
   * **Description:** A user comes up with the total time she/he uses in reading the books which are already complete to know how she/he spends his/her time.
   * **Primary Actor:** User
8. **Generate a List of Friends Who Have Borrowed Books**
   * **Description:** A user sees friends with borrowed books to monitor all loans keeping track at any given time.
   * **Primary Actor:** User
9. **Find the Most Borrowed Book**
   * **Description:** A user identifies the most borrowed book in their collection to recognize popular books among friends.
   * **Primary Actor:** User
10. **View All Books Currently Being Read**
    * **Description:** A user sees all the books that he is currently reading to be able to plan for them in terms of time.
    * **Primary Actor:** User
11. **List All Friends Who Have Not Returned Books on Time**
    * **Description:** The user says who among his friends has not returned books on time to ask them to return the same.
    * **Primary Actor:** User
12. **Calculate the Average Cost of Books in a Genre**
    * **Description:** A user wants to know the price of books in some category and for this, they find out the average of the given set of book prices.
    * **Primary Actor:** User

**Friend (Indirect Actor)**

1. **Return Loaned Book**
   * **Description:** A friend brings back a book borrowed from a user, which the user modifies within his/ her account.
   * **Primary Actor:** User

# Use Case Diagram



# User Stories

**User**

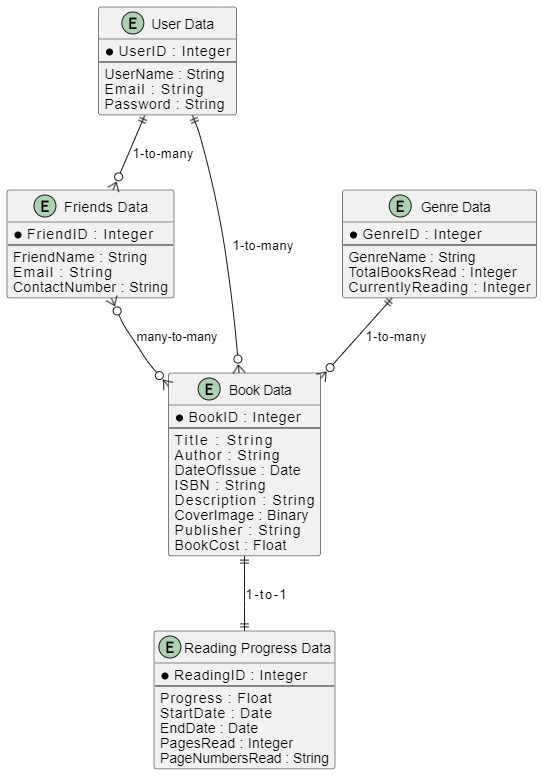
* As a user, I want to view all books in a specific genre so that I can see what books are available in that category.
* As a user, I want to find all the books that I have loaned to a specific friend so that I can track which books they currently have.
* As a user, I want to list all overdue loans so that I can remind my friends to return the borrowed books on time.
* As a user, I want to check if a specific book is available in my collection so that I know if I can loan it out or read it myself.
* As a user, I want to see the top N most expensive books in my collection so that I can assess the value of my collection.
* As a user, I want to search for books using a partial ISBN so that I can quickly find books when I only remember part of the ISBN.
* As a user, I want to calculate the total time I have spent reading completed books so that I can better understand my reading habits.
* As a user, I want to generate a list of friends who currently have borrowed books from me so that I can track all loans.
* As a user, I want to find out which book in my collection has been borrowed the most so that I can identify popular books among my friends.
* As a user, I want to see a list of all books I am currently reading so that I can manage my reading schedule effectively.
* As a user, I want to identify which friends have not returned books on time so that I can send them reminders.
* As a user, I want to know the average cost of books in a specific genre so that I can understand the cost distribution in my collection.

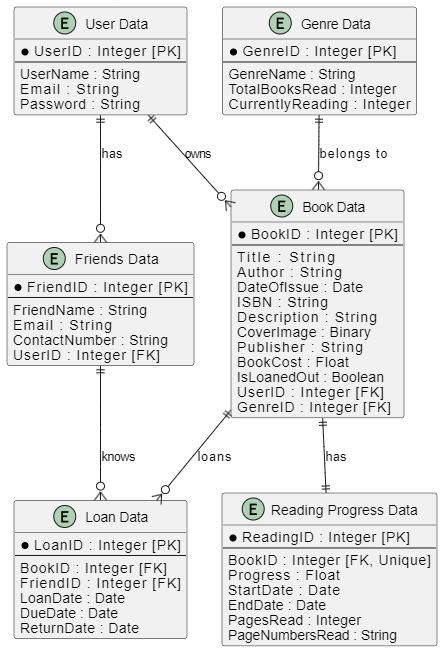
**Friend (Indirect Actor)**

* As a friend, I want to return a loaned book to the user so that the user can update their collection accordingly.

# Entity-Relationship Diagram (ERD)

**Logical ERD:**



**Physical ERD:**  


**Entities and Relationships:**

* **User Data**
  + One-to-Many with **Friends Data**: A user can have one or more friends.
  + One-to-Many with **Book Data**: A user can have one or more books.
  + One-to-Many with **Reading Progress Data**: A user can have one or more reading progress records because of different books.
* **Friends Data**
  + Many-to-One with **User Data**: Each friend can have one user (the owner).
  + One-to-Many with **Loan Data**: A friend can have one or more books loaned.
* **Genre Data**
  + One-to-Many with **Book Data**: A genre can have one or more books in it.
* **Book Data**
  + Many-to-One with **User Data**: Each book is owned by the user (the owner).
  + Many-to-One with **Genre Data**: Each book is of a specific genre.
  + One-to-Many with **Loan Data**: A book can be loaned multiple times.
  + One-to-Many with **Reading Progress Data**: A book can have one or more reading progress entries because of different users.
* **Loan Data**
  + Many-to-One with **Book Data**: Each loan entry is associated with a specific book.
  + Many-to-One with **Friends Data**: Each loan entry is associated with a specific friend.
* **Reading Progress Data**
  + Many-to-One with **Book Data**: Each reading progress entry is associated with a specific book.
  + One-to-One with **User Data**: Each reading progress entry belongs to one user (the owner).

(Library Management Systems, n.d.)

(19 - Library management systems)

# Table Designs – Data Dictionary

**User Data Table**

Stores information about each user of the system, including unique identifier, name, email address, and password.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| UserID | Unique ID for each user | Integer | PK | Unique, Not Null | 1 |
| UserName | Name of the user | String |  |  | John Doe |
| Email | Email address of the user | String |  | Unique, Not Null | john.doe@example.com |
| Password | Password of the user | String |  | Not Null | securepassword123 |

**Genre Data Table**

Stores information about different genres available in the system, including unique identifier, genre name, total books read, and currently being read.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| GenreID | Unique ID for each genre | Integer | PK | Unique, Not Null | 101 |
| GenreName | Name of the genre | String |  | Not Null | Science Fiction |
| TotalBooksRead | Total number of books read in this genre | Integer |  |  | 25 |
| CurrentlyReading | Number of books currently being read in this genre | Integer |  |  | 3 |

**Book Data Table**

Stores information about each book in the collection, including unique identifier, title, author, publication details, cost, and loan status.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| BookID | Unique ID for each book | Integer | PK | Unique, Not Null | 1001 |
| Title | Title of the book | String |  | Not Null | Dune |
| Author | Author of the book | String |  |  | Frank Herbert |
| DateOfIssue | Date when the book was issued | Date |  |  | 2024-01-15 |
| ISBN | ISBN number of the book | String |  | Unique, Not Null | 978-0441013593 |
| Description | Description of the book | String |  |  | A science fiction novel about politics and power. |
| CoverImage | Cover image of the book | Binary |  |  | [Image Data] |
| Publisher | Publisher of the book | String |  |  | Ace Books |
| BookCost | Cost of the book | Float |  |  | 9.99 |
| IsLoanedOut | Loan status of the book | Boolean |  |  | True |
| UserID | ID of the user owning the book | Integer | FK | References UserID | 1 |
| GenreID | ID of the genre the book belongs to | Integer | FK | References GenreID | 101 |

**Friends Data Table**

Stores details about friends of users, including a unique identifier, name, email, and contact number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| FriendID | Unique ID for each friend | Integer | PK | Unique, Not Null | 201 |
| FriendName | Name of the friend | String |  |  | Alice Smith |
| Email | Email address of the friend | String |  |  | alice.smith@example.com |
| ContactNumber | Contact number of the friend | String |  |  | +1234567890 |
| UserID | ID of the user associated with this friend | Integer | FK | References UserID | 1 |

**Loan Data Table**

Stores details of books loaned to friends, including unique loan identifier, book and friend references, and loan, due, and return dates.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| LoanID | Unique ID for each loan | Integer | PK | Unique, Not Null | 301 |
| BookID | ID of the book being loaned | Integer | FK | References BookID | 1001 |
| FriendID | ID of the friend to whom the book is loaned | Integer | FK | References FriendID | 201 |
| LoanDate | Date when the book was loaned | Date |  |  | 2024-03-01 |
| DueDate | Due date for the book to be returned | Date |  |  | 2024-04-01 |
| ReturnDate | Date when the book was actually returned | Date |  |  | 2024-03-30 |

**Reading Progress Data Table**

Stores the progress of each book being read, including unique identifier, book reference, progress percentage, and dates.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type** | **Key Type** | **Constraints** | **Example** |
| ReadingID | Unique ID for each reading progress record | Integer | PK | Unique, Not Null | 401 |
| BookID | ID of the book being read | Integer | FK | References BookID | 1001 |
| Progress | Reading progress percentage | Float |  |  | 75.0 |
| StartDate | Start date of reading | Date |  |  | 2024-02-15 |
| EndDate | End date of reading | Date |  |  | 2024-03-15 |
| PagesRead | Total number of pages read | Integer |  |  | 300 |
| PageNumbersRead | Specific page numbers read during the session | String |  |  | 1-50, 100-150 |

# Contributions:

* **Leader (Bhanu)**:
  + All sections
  + Oversight, Graphs, Integration
* **Member 1 (Jaspreet)**:
  + Initial Analysis
  + Design Analysis
  + Functional Requirements
* **Member 2 (Sahib)**:
  + System Development Overview
  + SDLC Documentation
  + Use Cases, User Stories
* **Member 3 (Mandeep)**:
  + Entity-relationship diagram (ERD)
  + Table Designs, Data Dictionary
  + Non-Functional Requirements

**Section Assignments:**

1. **Executive Summary**
   * Bhanu, Jaspreet, Sahib, Mandeep
2. **Initial Analysis of the Problem Domain**
   * Jaspreet, Bhanu
3. **System Development Overview**
   * Sahib, Bhanu
4. **Design Analysis Process**
   * Jaspreet, Bhanu
5. **System Development Life Cycle (SDLC)**
   * Sahib, Bhanu
6. **Functional and Non-Functional Requirements**
   * Jaspreet (Functional), Mandeep (Non-Functional)
7. **Actors**
   * Mandeep, Bhanu
8. **Use Cases**
   * Sahib, Bhanu
9. **Use Case Diagram**
   * Bhanu
10. **User Stories**
    * Sahib, Bhanu
11. **Entity-Relationship Diagram (ERD)**
    * Mandeep, Bhanu
12. **Table Designs – Data Dictionary**
    * Mandeep, Bhanu
13. **Contributions**
    * Bhanu
14. **References & GitHub**
    * Bhanu

# References

(n.d.). In L. K. Velasquez, *Library Management 101: A Practical Guide, Second Edition.*

(n.d.). In G. G. Chowdhury, *19 - Library management systems.*

Breeding, M. (2020, May 1). *2020 Library Systems Report*. Retrieved from American Libraries Magazine: https://americanlibrariesmagazine.org/2020/05/01/2020-library-systems-report/

*Libib*. (n.d.). Retrieved from Libib: https://www.libib.com/

*Library Management System Project | Software Development*. (2024, September 13). Retrieved from Geeks For Geeks: https://www.geeksforgeeks.org/library-management-system/

*Library Management Systems*. (n.d.). Retrieved from TrustRadius: https://www.trustradius.com/library-management

Lin, M. (2024, August 22). *Top 10: Library Management Software (LMS)*. Retrieved from Bookrunch: https://www.bookrunch.com/overview/library\_management\_software/

*The Digital Transformation of Libraries: How Library Management Systems are Shaping the Future*. (2023, December 12). Retrieved from loqqat: https://blog.loqqat.com/blog/library-management-software/the-digital-transformation-of-libraries:-how-library-management-systems-are-shaping-the-future

# GITHUB

<https://github.com/bhanuGupta1/BCMS>