

***Software Requirement and Design Specifications***

***Library Management System***

***Version: 1.0***

| *Course Code* | *CS 3004* |
| --- | --- |
| *Instructor* | *Miss Nida Munawar* |
| *Project Team* | *Insha Javed (21K-3279)*  *Sabika Shameel (21K-4606)* |
| *Submission Date* | *1st – December-2023* |

*Table of Contents*

[*1. Introduction 3*](#_heading=h.gjdgxs)

[1.1.](#_heading=h.30j0zll) Purpose of Document 4

[1.2.](#_heading=h.1fob9te) Intended Audience 4

[*2. Overall System Description 5*](#_heading=h.3znysh7)

[2.1.](#_heading=h.3j2qqm3) Project Background 5

[2.2.](#_heading=h.1y810tw) Project Scope 5

[2.3.](#_heading=h.4i7ojhp) Not In Scope 5

[2.4.](#_heading=h.2xcytpi) Project Objectives 5

[2.5.](#_heading=h.1ci93xb) Stakeholders 5

[2.6.](#_heading=h.3whwml4) Operating Environment 5

[2.7.](#_heading=h.2bn6wsx) System Constraints 5

[2.8.](#_heading=h.qsh70q) Assumptions & Dependencies 5

[*3. External Interface Requirements 6*](#_heading=h.2et92p0)

[3.1.](#_heading=h.3as4poj) Hardware Interfaces 6

[3.2.](#_heading=h.1pxezwc) Software Interfaces 6

[3.3.](#_heading=h.49x2ik5) Communications Interfaces 6

[*4. Functional Requirements 7*](#_heading=h.tyjcwt)

[*4.1. Functional Hierarchy 7*](#_heading=h.3dy6vkm)

[4.2.](#_heading=h.1t3h5sf) Use Cases 7

[4.2.1.](#_heading=h.4d34og8) [Title of use case] 7

[*5. Non-functional Requirements 8*](#_heading=h.2s8eyo1)

[5.1.](#_heading=h.2p2csry) Performance Requirements 8

[5.2.](#_heading=h.147n2zr) Safety Requirements 8

[5.3.](#_heading=h.3o7alnk) Security Requirements 8

[5.4.](#_heading=h.23ckvvd) User Documentation 8

[*SDS 10*](#_heading=h.17dp8vu)

[***6. System Architecture 11***](#_heading=h.3rdcrjn)

[*6.1.*](#_heading=h.26in1rg) *System Level Architecture 11*

[*6.2.*](#_heading=h.ihv636) *Software Architecture 11*

[***7. Design Strategy 11***](#_heading=h.lnxbz9)

[***8. Detailed System Design 12***](#_heading=h.lnxbz9)

[*8.1.*](#_heading=h.35nkun2) *Database Design 12*

[*9. Application Design 13*](#_heading=h.1ksv4uv)

[*10. References 14*](#_heading=h.44sinio)

[*11. Appendices 15*](#_heading=h.z337ya)

## 1. Introduction

#### Purpose of Document

#### This Software Requirements Specification (SRS) document is designed to offer a detailed overview of the Library Management System project. It presents the problem that needs to be addressed, the proposed solution, the scope of the project, its features and functions, the data model, the tools and techniques to be used, the project timeline, and a conclusion. The document is intended to serve as a reference for the development team, stakeholders, and any other parties involved in the project, providing them with a clear understanding of the project’s requirements and goals.

#### 1.2. Intended Audience

The document is intended for different groups of people:

1. Developers: It provides technical details for software engineers, developers, and programmers working on the library management website. This includes information about how the system is structured, how the database is designed, and how the different parts of the website work together.
2. Project Managers: It is important for project managers who are in charge of planning, allocating resources, and monitoring the progress of the library management website development. The document outlines the specific requirements for the website in the context of a library, which helps with making informed decisions.
3. Stakeholders: It offers a clear understanding of the goals, benefits, and potential risks associated with the library management website. This helps stakeholders make decisions and ensure that the website aligns with the overall goals of the organization.
4. End Users: It focuses on the needs of librarians, staff, and patrons who will be using the website. This includes information about how the website should be designed and how it should be easy to use, so that it meets the specific needs of the people who will be using it.

***1.3 Definition of Terms, Acronyms and Abbreviations***

*[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly. ]*

| ***Term*** | ***Description*** |
| --- | --- |
| *ASP* | *Active Server Pages* |
| *DD* | *Design Specification* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***1.4 Document Convention***

*[Describe the font and font size that this document will be using]*

**2. Overall System Description**

**2.1 Project Background**

The current library system is manual and inefficient, leading to difficulties in providing timely information to library patrons. The need for an automated Library Management System was identified to enhance library operations and availability.

**2.2 Project Scope**

The library management website aims to streamline and enhance the book management process for both students and administrators within the educational institution. The scope of the project encompasses the development of a user-friendly web platform that facilitates efficient cataloging, tracking, and management of the library's book inventory.

**2.1.1 Inclusions**

**Student Functionality:**

* Browse Book Catalog: Students will have the ability to explore the library's book collection through an intuitive and categorized catalog.
* Request Book: Students can request books online, initiating a streamlined process for book issuance.
* View Book Record: Access to individual book records for detailed information.

**Admin Functionality:**

* Add Book: Administrators can add new books to the system, including details such as title, author, genre, and quantity.
* View Book Report: Generate comprehensive reports on the library's book inventory, aiding in decision-making and inventory management.
* Manage Book Requests: Administrators will have tools to efficiently handle book requests, including approval and rejection functionalities.
* Add Student: Capability to add new students to the system, capturing essential details for user management.
* View Student Report: Access to student-related reports for administrative purposes.
* Issue Book to Students: Administrators can manage the issuance of books to students, maintaining a record of transactions.
* View Issue Record: Track and review the history of book issuances.

**2.3 Not In Scope**

The project scope does not include functionalities outside the realm of library management, such as financial transactions, external e-commerce integration, or unrelated administrative tasks.

**2.4 Project Objectives**

The primary objectives of the library management website are to:

* Provide an intuitive and efficient interface for students to interact with the library catalog.
* Enable administrators to effectively manage the book inventory, student records, and book transactions.
* Enhance the overall accessibility and usability of the library services through an online platform.

**2.5 Stakeholders**

Stakeholders in the system include:

* Managers**:** Responsible for task assignment and overseeing task completion.
* Employees**:** Users who will interact with the dashboard to view and complete assigned tasks.
* Database Administrators**:** Responsible for maintaining and managing the underlying database.
* Project Managers**:** Oversee the overall progress and success of the project.

**2.6 Operating Environment**

The system will function within a web-based setting, utilizing standard computers and internet-connected devices as its hardware platform. It is compatible with popular operating systems like Windows and Linux, and can seamlessly coexist with web browsers like Chrome and Firefox.

**2.7 System Constraints**

The system aims to provide a complete solution, but there are some limitations to consider. These include budget, time, and possible technology restrictions. However, the system will be designed to be scalable, so it can easily handle future improvements and updates.

**2.8 Assumptions & Dependencies**

Assumptions and dependencies for the system include:

**Assumptions:**

* Users have reliable internet access,
* The system supports modern web browsers.
* Administrators maintain accurate book information.
* Standard security measures are implemented.
* Administrators manage system updates responsibly.

**Dependencies:**

* System functionality relies on the chosen DBMS.
* Project success depends on timely user feedback.
* Stability of the technology stack is vital.
* Adherence to budgetary limits is essential.

## 3. External Interface Requirements

**3.1 Hardware Interfaces**

The web-based Library Management System is devoid of specific hardware dependencies, functioning seamlessly on conventional computing devices such as desktops, laptops, and tablets. Its hardware interfaces are universally applicable, accommodating devices equipped with web browsers and internet connectivity.

**3.2 Software Interfaces**

The library management website seamlessly integrates various technologies to deliver a robust and dynamic user experience. The software interfaces employed in this system include:

1. **HTML (Hypertext Markup Language):**

* Utilized for structuring and presenting content on the user interface.
* HTML ensures a standardized and accessible foundation for web page design.

1. **CSS (Cascading Style Sheets)**

* Employed to enhance the visual presentation and layout of HTML elements.
* CSS provides a consistent and aesthetically pleasing design across the library management website.

1. **PHP (Hypertext Preprocessor):**

* Powering the backend functionality, PHP is employed for server-side scripting.
* Enables dynamic content generation, form handling, and database interactions for a responsive and interactive user experience.

1. **MySQL Database:**

* Serves as the relational database management system for storing and retrieving data.
* MySQL ensures efficient data management, integrity, and security for the library management system.

**3.3 Communications Interfaces**

The Library Management System relies on standard web communication protocols. The communication interfaces include:

HTTP/HTTPS Protocols: The system uses HTTP/HTTPS protocols for communication between the client (web browser) and the server (Node.js/Express backend). HTTPS is employed to ensure secure data transmission.

RESTFUL API: The backend exposes RESTful API endpoints to facilitate communication between the frontend and backend. This includes endpoints for task management, user authentication, and other system functionalities.

Communication Security: Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols will be implemented to ensure the security of data during transmission.

Data Transfer Rates: The data transfer rates will depend on the internet connection speed of the user. However, efforts will be made to optimize data transfer for efficient performance.

Synchronization Mechanisms: The system will employ asynchronous communication for non-blocking interactions, providing a responsive user experience.

## 4. Functional Requirements

### *4.1. Functional Hierarchy*

The functional hierarchy provides a comprehensive overview of the library management website's overall system functionality, breaking down key modules and features into a structured hierarchy. This high-level perspective allows stakeholders, project managers, and development teams to grasp the fundamental components of the system before delving into detailed use cases. The functional hierarchy is designed to facilitate a clear understanding of the library management website's main functionalities:

1. **Library Administration:**

* User Management:
* Create, update, and delete user accounts for librarians and staff.
* Assign and manage user roles and permissions.
* Resource Management:
* Add, edit, and remove books, journals, and other library resources.
* Categorize resources and update availability status.
* Inventory Control:
* Track and manage the library's inventory, including book circulation and returns.
* Generate reports on resource utilization and availability.

1. **Patron Services:**

* *User Accounts:*
  + Allow patrons to create accounts, update profiles, and manage their borrowing history.
  + Provide personalized recommendations based on user preferences.
* *Search and Browse:*
  + Enable patrons to search and browse the library catalog with ease.
  + Implement filters for refining search results.
* *Borrowing and Returns:*
  + Facilitate the borrowing and return of resources through a user-friendly interface.
  + Send notifications for overdue items and manage fines.

1. **Communication and Notifications:**
   * *Alerts and Notifications:*
     + Implement a system for sending alerts on new arrivals, overdue items, and library announcements.
     + Allow users to opt-in for communication preferences.
   * *Messaging System:*
     + Provide a messaging platform for communication between librarians and patrons.
     + Send automated reminders for upcoming due dates.
2. **Reporting and Analytics:**
   * *Usage Reports:*
     + Generate reports on resource usage, popular items, and library traffic.
     + Provide insights for informed decision-making and resource allocation.
   * *Financial Reports:*
     + Compile financial reports related to fines, fees, and resource purchases.
     + Support budgeting and financial planning.

#### 

#### 

#### 4.2. Use Cases

##### 4.2.1. E-Library

Use Cases for Library Management system

Actors: Student, Teacher, Admin

Database

Use Cases:

For the Admin:

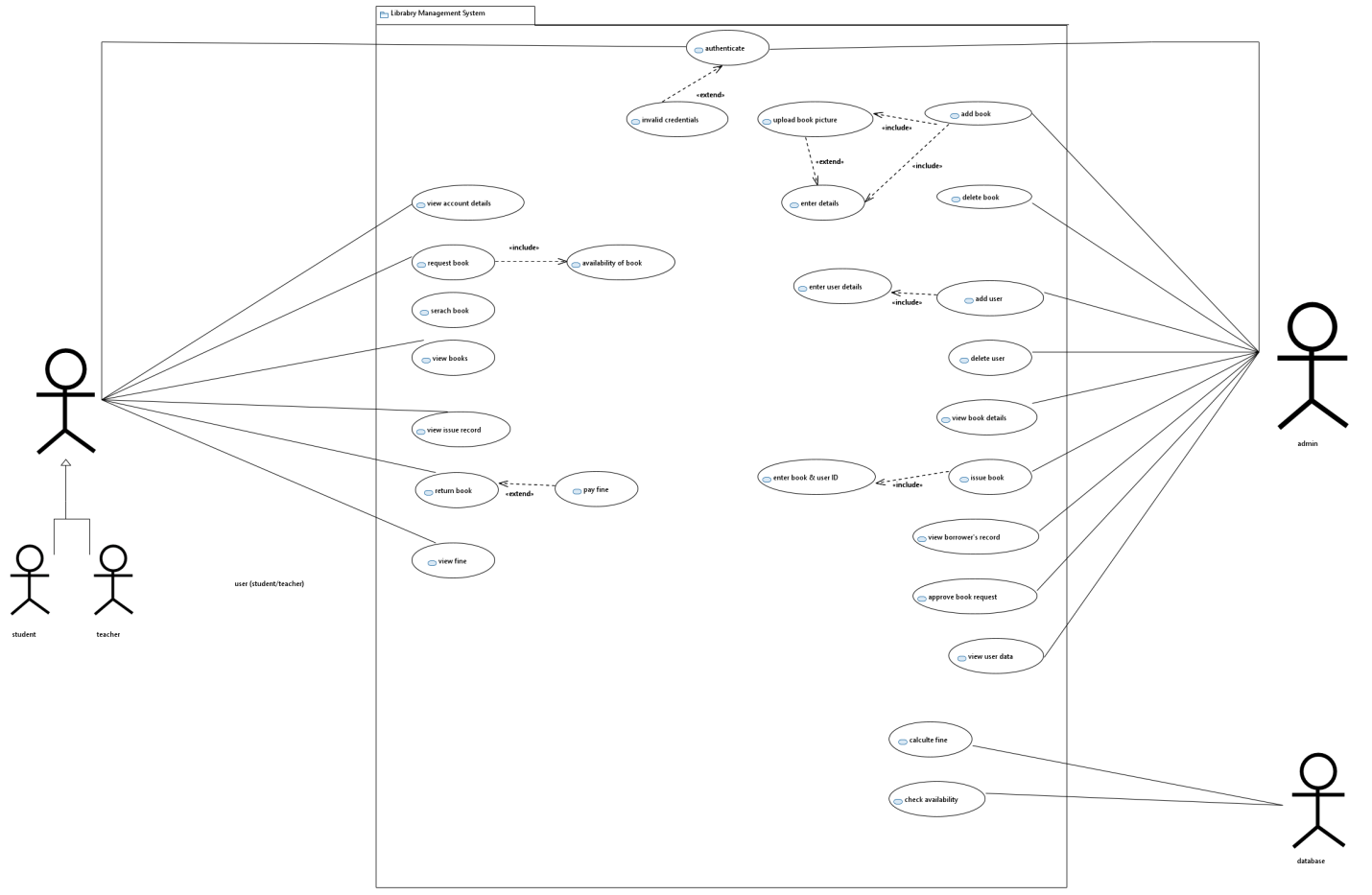
1. Add student: The admin can add new student record.
2. Add Book: The admin can add new book record.
3. View student record: The admin can view the record of students.
4. View book record: The admin can view the record of books.
5. View borrowers record: The admin can view the borrower's and book issue report.
6. Delete student: The admin can delete a student record when required. if it is not relevant anymore.
7. Issue book: The admin can issue books for the user on their request.

For the User:

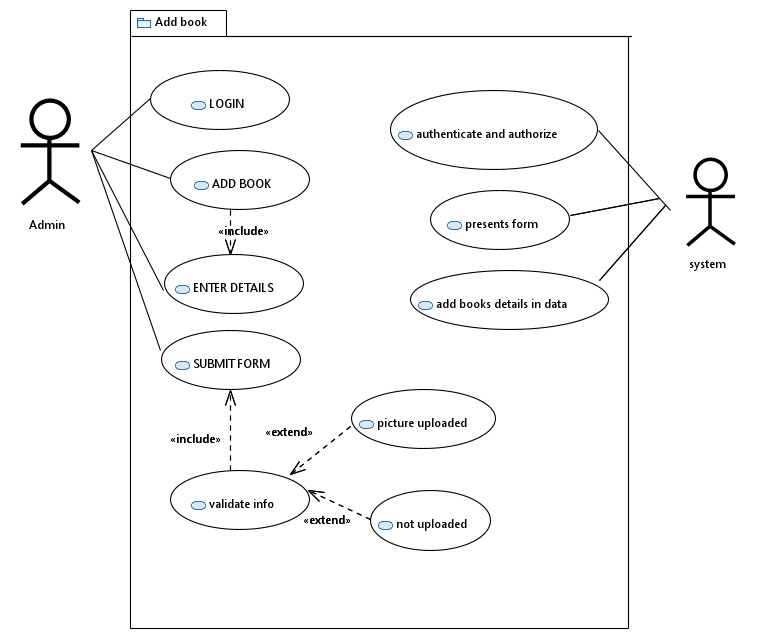
1. Request book: The student or teacher can request for a specific book from a list of the available books.
2. View account info: The user can see their account details.
3. View user’s issue record: A user can view their book issue record.
4. View all books: The student or teacher can view the list of the books.

For the System:

1. Update records from database: Whenever a new record is added or deleted by the admin, the system updates the database.
2. Calculate fine: The system can calculate fine using issue date and return date details from the database.



**Use Case Name 01: Add book**



| USE CASE ID: | UC-1 |
| --- | --- |
| USE CASE NAME: | Add a book |
| USE CASE DESCRIPTION: | The system will allow the admin to add book records. |
| AUTHORS: | Insha Javed  Sabika Shameel |
| PRIMARY ACTOR: | User(student/teacher) |
| SUPPORTING ACTOR: | Admin  Database system |
| PRECONDITIONS: | User or admin must be logged in. |
| POSTCONDITIONS: | Book is issued. |
| MAIN SUCCESS SCENERIO: | 1. Admin logs in. 2. He clicks “add book” button 3. He enters the book details in the form. 4. Uploads the image. 5. Clicks ”submit” button. 6. The record is added to the database. 7. Admin clicks “book record” button and the list of available books is displayed. |
| EXTENSIONS: | 1.Admin enters wrong email or password.  1.a. prompt displayed.  1.b rewrite the correct credentials.  4. If picture no uploaded.  4.a. prompt displayed “no file uploaded”.  4.b go back and add the picture. |

**Use Case Name 02: Issue Book**

| USE CASE ID: | UC-2 |
| --- | --- |
| USE CASE NAME: | Issue a book |
| USE CASE DESCRIPTION: | The system will allow the users to conveniently request and issue books. |
| AUTHORS: | Insha Javed  Sabika Shameel |
| PRIMARY ACTOR: | User(student/teacher) |
| SUPPORTING ACTOR: | Admin  Database system |
| PRECONDITIONS: | User or admin must be logged in. |
| POSTCONDITIONS: | Book is issued. |
| MAIN SUCCESS SCENERIO: | 1. Admin logs in. 2. Admin select “Issue book” button. 3. He chooses a book. 4. He chooses the student name. 5. He clicks “submit” button on the form. |
| EXTENSIONS: | 1.Admin enters wrong email or password.  1.a. prompt displayed.  1.b rewrite the correct credentials.  4.He chooses a student name already deleted.  4.a. the system logs out the admin. |

**Use Case Name 03: Delete Book**

| USE CASE ID: | UC-3 |
| --- | --- |
| USE CASE NAME: | View/delete student record |
| USE CASE DESCRIPTION: | The system will allow the admin to view student and make deletions. |
| AUTHORS: | Insha Javed  Sabika Shameel |
| PRIMARY ACTOR: | User(student/teacher) |
| SUPPORTING ACTOR: | Admin  Database system |
| PRECONDITIONS: | User or admin must be logged in. |
| POSTCONDITIONS: | Book is issued. |
| MAIN SUCCESS SCENERIO: | 1. Admin logs in . 2. Admin selects“student records” button. 3. A list of students is shown. 4. He views the students. 5. He clicks the “delete” button. 6. Admin clicks on the “students records” button again. 7. The record is successfully deleted. |
| EXTENSIONS: | 1.Admin enters wrong email or password.  1.a. prompt displayed.  1.b rewrite the correct credentials. |

## 5. Non-functional Requirements

**5.1 Performance Requirements**

The library management website is expected to demonstrate efficient performance to ensure a seamless user experience. Specific performance requirements include:

* + Response Time, the system should respond to user requests within a maximum of 2 seconds.
  + Scalability, the platform must handle a concurrent user load of at least 500 users without significant degradation in performance.

**5.2 Safety Requirements**

* Data Integrity: All user and transaction data must be securely stored and protected against unauthorized access or loss.
* System Reliability: The system should have a minimum uptime of 99%, ensuring continuous availability for users.

**5.3 Security Requirements**

Security is paramount to protect user information and maintain system integrity. Key security requirements include:

* Authentication**:** Users must authenticate securely, with the use of strong password policies and potentially multi-factor authentication.
* Authorization**:** Access controls will be implemented to ensure that users only have access to the features and data relevant to their roles.
* Data Encryption**:** All sensitive data, including user credentials and transaction information, should be encrypted during transmission.

**5.4 User Documentation**

To guarantee effective system navigation and utilization for all users, including students and administrators, a comprehensive set of user documentation will be provided. The documentation will encompass the following aspects:

* User Guides: Step-by-step guides for common tasks, such as browsing the catalog, making requests, and managing book inventory.
* FAQs: Frequently Asked Questions to address common queries and issues.

## SDS

# 6. System Architecture

### 6.1. System Level Architecture

The Library Management System is a web-based application that follows a client-server architecture. It is divided into the following important parts:

1. **Client-Side (Frontend):**

The frontend of the system is created using HTML and CSS. It provides a user-friendly interface for the Library Management System. It includes different components for various functions such as reserving books, managing user profiles, tracking borrowing history, user authentication, and dashboard management. This well-designed frontend improves the user experience and helps with efficient library system management.

1. **Server-Side (Backend):**

The backend of the system uses PHP, a strong scripting language for servers. It handles API requests efficiently, manages user authentication to ensure only authorized access, and acts as a connection between the frontend and the database. It processes and validates user requests, ensures data integrity, and sends the data back to the frontend. This architecture ensures efficient data management and provides a solid foundation for the system.

1. **Database Management System (DBMS):**

For the Library Management System, MySQL is used as the Database Management System. It helps store and manage data about students, administrators, and books effectively. The database is organized into tables for users, books, and reservations. These tables make it easy to retrieve and modify data, ensuring the system can quickly access the information it needs. This strong database design is important for the smooth functioning of the Library Management System.

1. **Communication Protocols:**

The system uses HTTP/HTTPS protocols for communication between the web browser (client-side) and the PHP backend (server-side). This ensures secure and reliable data transmission. Restful API endpoints are defined to enable smooth interaction between the frontend and backend. These endpoints handle various functions like book reservation and user authentication. This design choice enables efficient data exchange and improves the functionality and performance of the system.

**System Decomposition:**

* **Client (Frontend):**
  + Components: Book reservation, Borrower’s record, accessing books, User Authentication, Dashboard UI.
  + Responsibilities: Render the UI, capture user interactions, send requests to the server, and display information.
* **Server (Backend):**
  + Components: API Endpoints, User Authentication Logic, Database Interface.
  + Responsibilities: Process client requests, handle user authentication, interact with the database, and send responses to the client.
* **Database:**
  + Tables: userdata, issue record, admin, book, requestbook
  + Responsibilities: Store and manage book information, user information, admin information, book records, books issued records.

**Interfaces:**

* **Client-Server Interface:**
  + The frontend communicates with the backend through defined API endpoints, sending JSON-formatted data.
* **Server-Database Interface:**
  + The backend interacts with the MySQL database using SQL queries for tasks such as inserting, retrieving, updating, and deleting data.

**Execution Environment:**

* The client-side code (HTML & CSS) is executed in users' web browsers.
* The server-side code (PHP) is executed on a web server.
* The database (MySQL) is hosted on a database server.

**Global Design Strategies:**

Error Handling: A centralized error-handling mechanism will be implemented to ensure consistent and informative error messages are returned to the client in case of issues.

Security Measures:Security measures, such as data encryption, user authentication, and authorization, will be enforced at both the client and server sides

**6.2 Software Architecture**

The architecture of the library management website is built on a combination of HTML for the user interface, CSS for styling, PHP for server-side scripting and business logic, and MySQL for database management. This four-tier architecture includes the User Interface Layer, the Middle Tier (implemented in PHP), the Data Access Layer (MySQL), and the Database Layer.

**Three-Tier Architecture:**

1. **User Interface Layer :** The User Interface Layer, constructed with HTML and styled using CSS, provides a visually appealing and user-friendly front end. It includes web pages for catalog browsing, book requests, and user interactions.

.

1. **Middle Tier (PHP):** The Middle Tier, implemented in PHP, handles server-side scripting and houses the business logic of the application. It acts as an intermediary between the User Interface Layer and the Data Access Layer, processing user requests and dynamically generating HTML pages.
2. **Data Access Layer (MySQL):** The Data Access Layer utilizes MySQL as the relational database management system. It is responsible for storing and retrieving data. PHP scripts interact with the MySQL database to execute queries and manage the library's book inventory and user information.

**Interactions between Layers:**

1. **User Interface Layer to Middle Tier:**

* User interactions through HTML forms and pages are processed by the Middle Tier (PHP).
* PHP scripts dynamically generate HTML pages based on user input.

1. **Middle Tier to Data Access Layer:**

* PHP scripts in the Middle Tier communicate with the Data Access Layer (MySQL) to perform database operations.
* SQL queries are executed to retrieve or update data in the MySQL database.

# 7. Design Strategy

The library management website's design approach is based on principles that focus on promoting system flexibility, maintainability, and ensuring an excellent user experience. Several important design choices have been implemented to tackle different aspects of system organization:

**1. Future System Extension or Enhancement:**

To make it future-proof and easy to improve, we designed the system with a modular and scalable architecture. We chose PHP because it allows us to easily add new features and functions using its server-side scripting capabilities. The system is organized in layers, so changes in one layer won't affect the others, making it easier to adapt and develop in the future.

**2. System Reuse:**

The design focuses on reusing components whenever possible. We created reusable PHP modules for common functions like user authentication and data validation. This makes the code more efficient, reduces repetition, and makes maintenance easier.

**3. User Interface Paradigms:**

The User Interface Layer is designed to be responsive and user-friendly. We use HTML and CSS to create an intuitive interface that works well on any device. Our aim is to provide a consistent and seamless user experience for students and administrators interacting with the system.

**4. Data Management (Storage, Distribution, Persistence):**

The Data Access Layer uses MySQL as the database system for storing and retrieving data efficiently. The database schema is designed to represent the relationships between entities like books, students, and transactions. We follow normalization principles to minimize data duplication and ensure data integrity. To handle data transfer between the User Interface Layer and the Database Layer, the system relies on server-side processing.

**5. Concurrency and Synchronization:**

Concurrency and synchronization are taken care of by the database itself. We use MySQL transactions to ensure that data remains consistent and reliable, especially when multiple users are accessing and updating the system at the same time. We carefully define transaction boundaries to avoid conflicts and maintain a clear and accurate picture of the data.

**6. Trade-offs:**

While our design choices prioritize flexibility and ease of maintenance, we have also made some optimizations. For instance, relying on server-side processing for dynamic content can introduce delays during peak user activity. However, this trade-off is justified because it ensures data integrity and consistency.

Overall, these design decisions align with our goal of creating a strong and adaptable library management system. By consistently following these strategies, we can easily enhance the system, promote reusability, offer a user-friendly interface, efficiently manage data, and handle concurrency effectively.

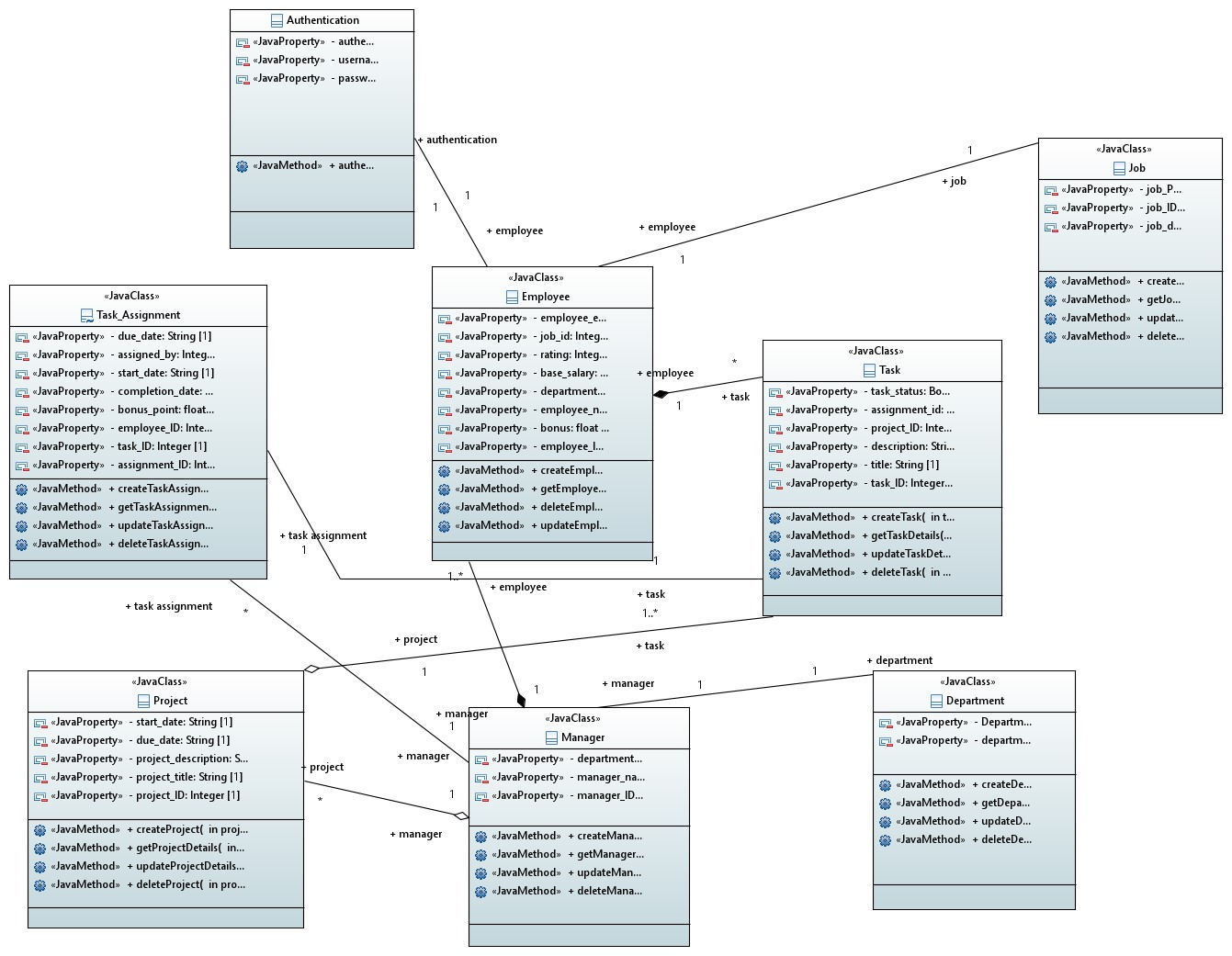
***8. Detailed System Design***

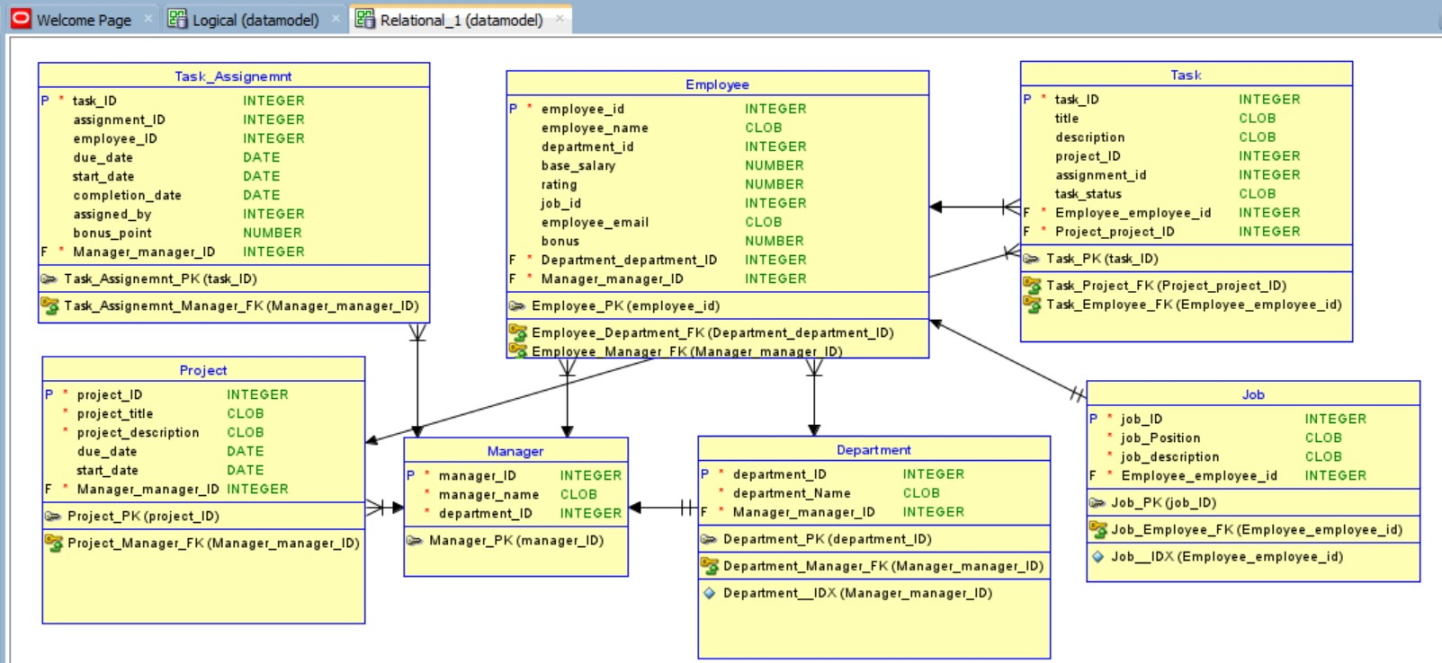
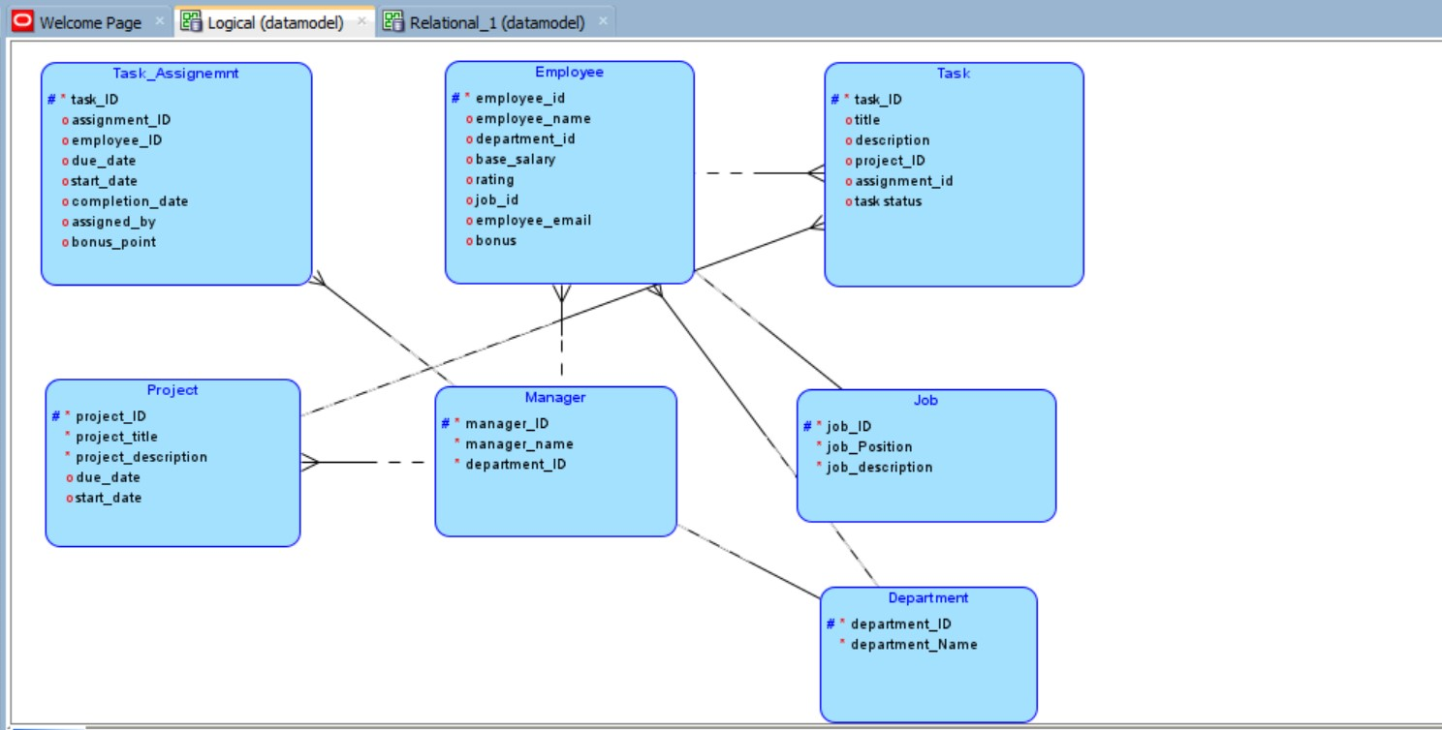
*[A detailed design should include the following:*

* *Detailed class diagram along with a detailed description of all attributes, functions or methods specifying interactions between different classes/modules.*
* *Logical data model (E/R model)*
* *Detailed GUIs]*

### 8.1. Database Design

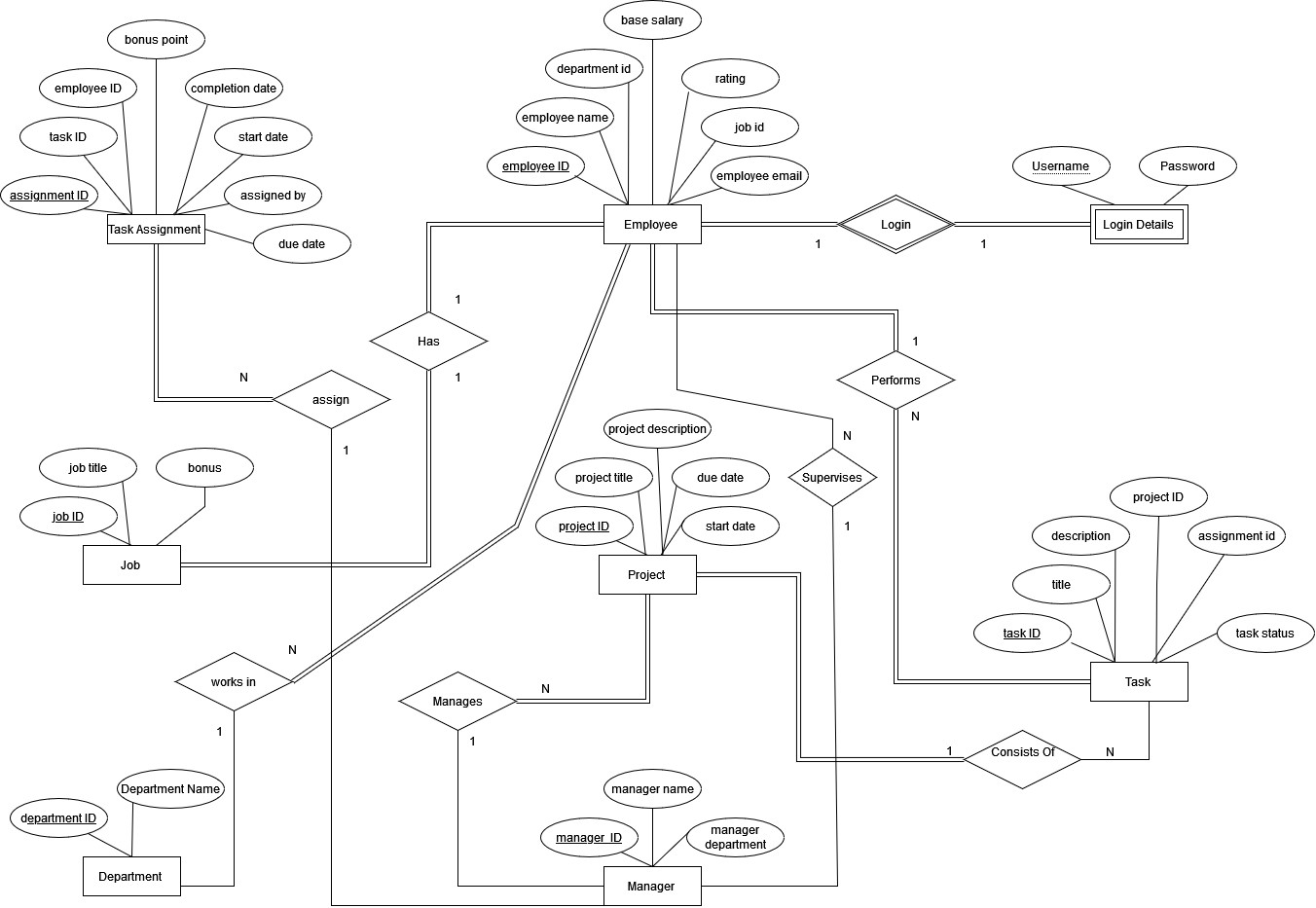
*[A detailed Database design should include the following:*

* *Logical data model (E/R model)*
* *Data dictionary]*
* ***Logical data model (E/R model)***
* **



***8.1.1. ER Diagram***

*[Entity Relationship Diagram of the system with description]*

**

***8.1.2. Data Dictionary***

*[The convention recommended for writing the data dictionary is as follows.]*

***8.1.2.1 Data 1***

*[Description (Refer to Template on next page). ]*

***8.1.2.2. Data 2***

*[Description (Refer to Template on next page). ]*

*.*

*.*

***8.1.2.3 Data n***

*[Description (Refer to Template on next page). ]*

| **< Data 1>** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Name*** | | *Give primary name of the data or control item, the data store or an external entity.* | | | | | |
| ***Alias*** | | *Write other names used for the first entry.* | | | | | |
| ***Where-used/howused*** | | *List all processes that use the data or control item and how it is used (e.g., input to process, output from the process, as a store, as n external entity)* | | | | | |
| ***Content description*** | | *Notation for representing content.* | | | | | |
|  | | | | | | | |
| ***Column Name*** | ***Description*** | | ***Type*** | ***Length*** | ***Null able*** | ***Default Value*** | ***Key Type*** |
| *[Column1*  *Name]* | *[Description of the column]* | | *[Type of column]* | *[Length of column]* | *[Is*  *Column*  *Null able]* | *[Default*  *Value]* | *[If Primary*  *Key than write*  *PK, if Foreign Key then FK, if not a key leave it blank]* |
| *[Column2*  *Name]* | *[Description of the column]* | | *[Type of column]* | *[Length of column]* | *[Is*  *Column*  *Null able]* | *[Default*  *Value]* | *[If Primary*  *Key than write*  *PK, if Foreign Key then FK, if not a key leave it blank]* |
|  |  | |  |  |  |  |  |

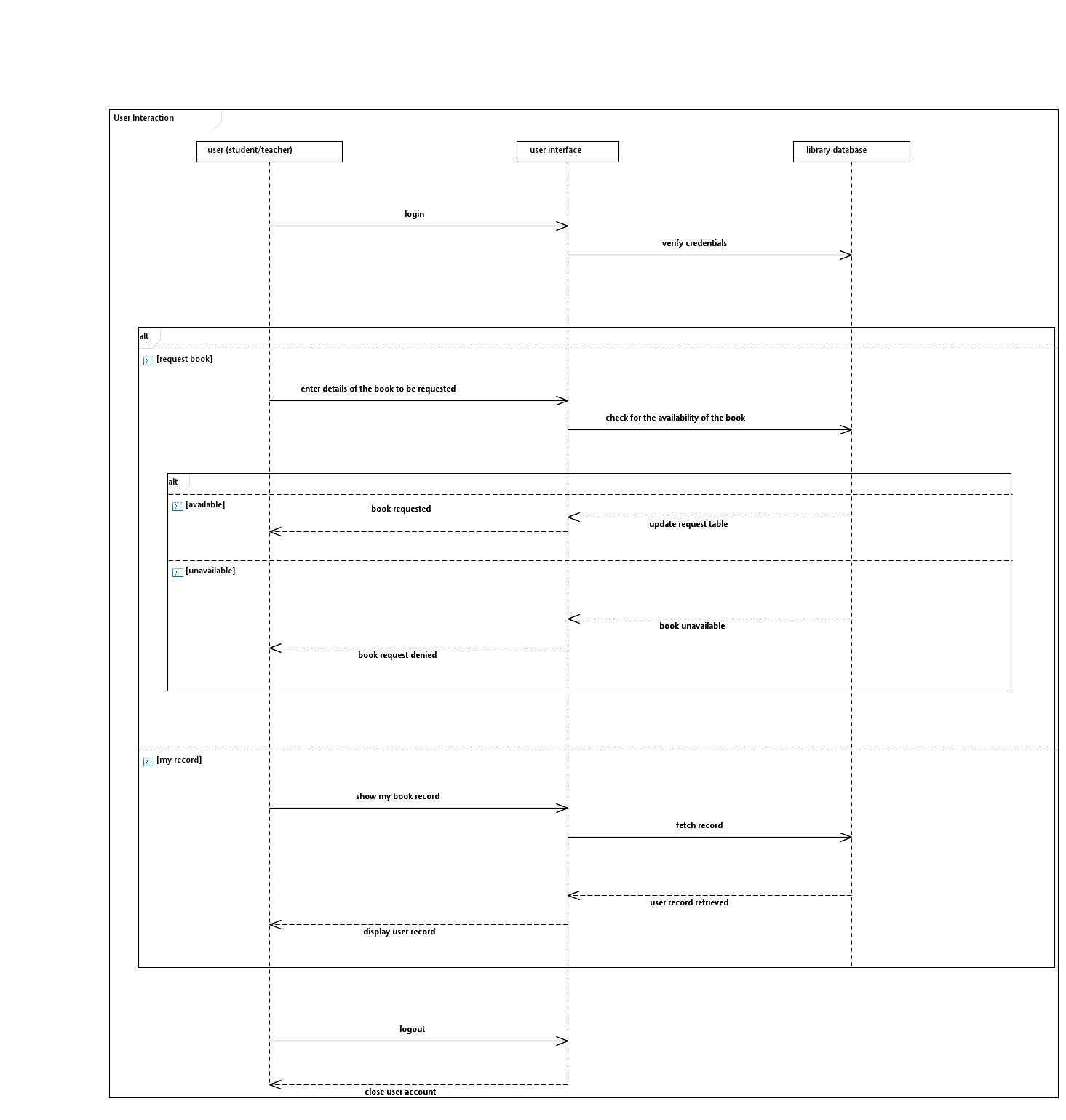
## 9. Application Design

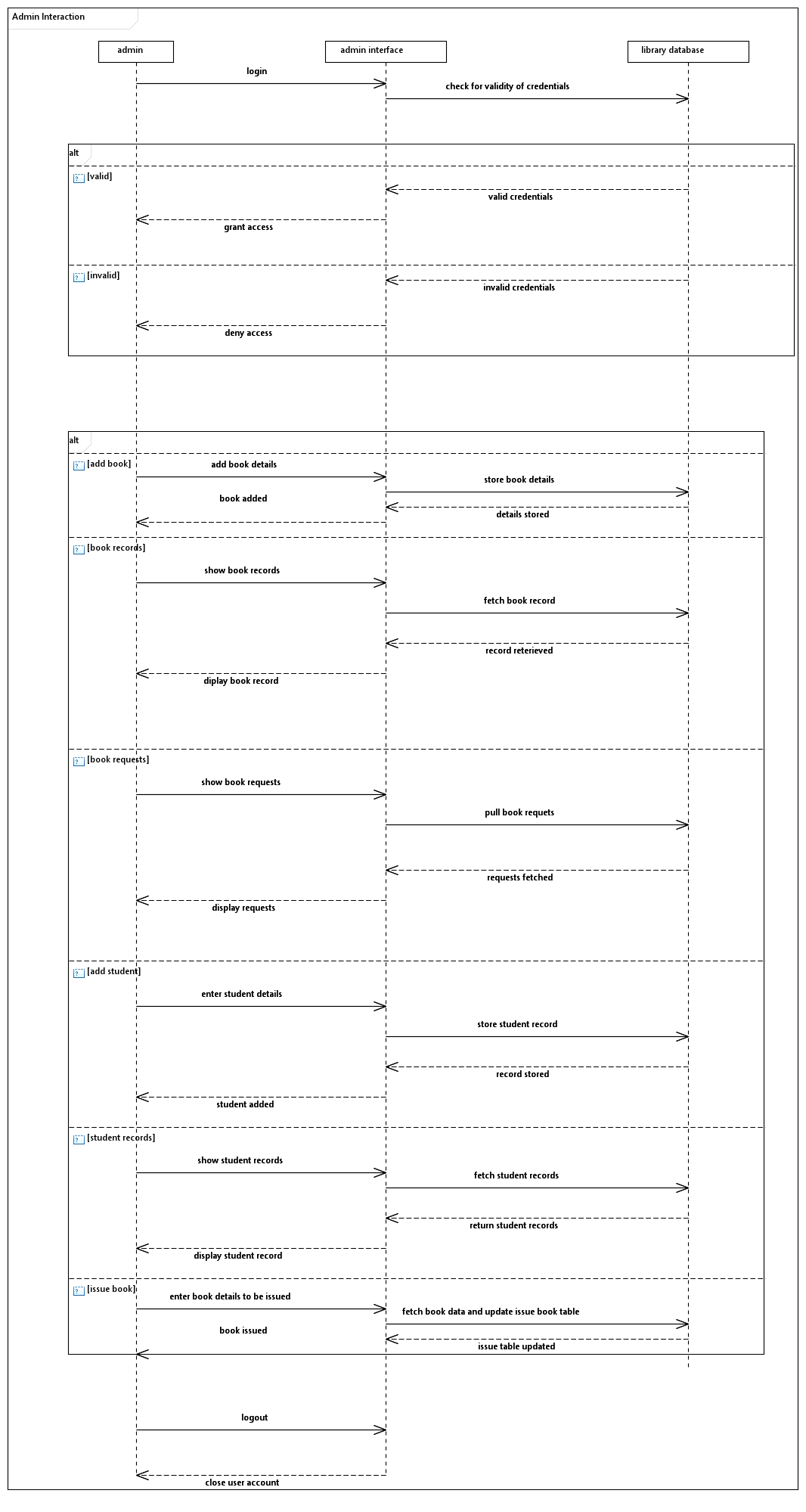
*[A detailed application design should include the following:*

* *Detailed Sequence diagram and Collaboration diagram with parameter list*
* *State Transition Diagram*
* *Activity Diagram*

***9.1.2. Sequence Diagram***

***9.1.2.1 User Sequence Diagram***





***9.1.2.2 <Sequence Diagram 2>***

*[Diagram & Explanation of diagram]*

*.*

*.*

***9.1.2.3 <Sequence Diagram n>***

*[Diagram & Explanation of diagram]*

***9.1.3. State Diagram***

***9.1.3.1 <State Diagram 1>***

*[Diagram & Explanation of diagram]*

***9.1.3.2 <State Diagram 2>***

*[Diagram & Explanation of diagram]*

*.*

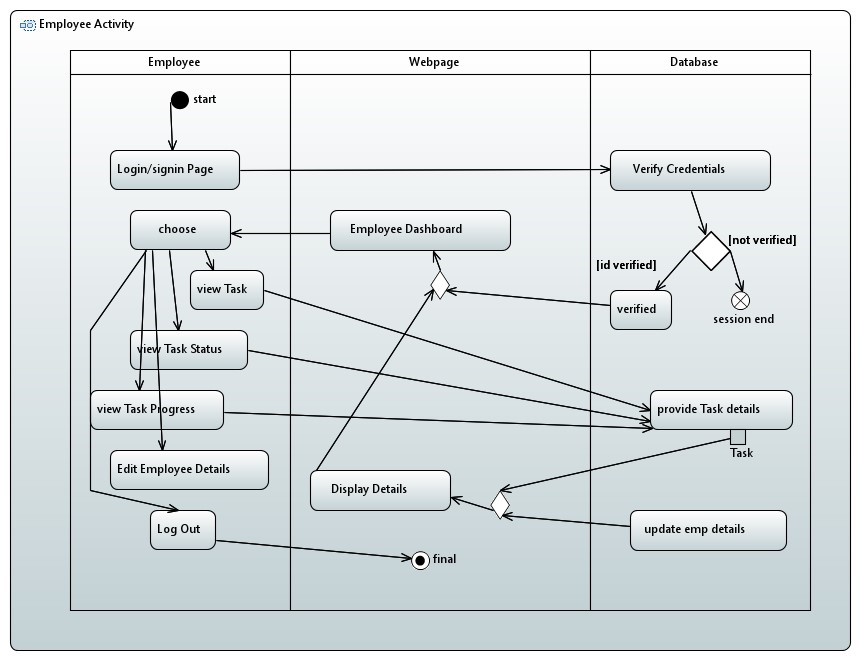
*.*

*.*

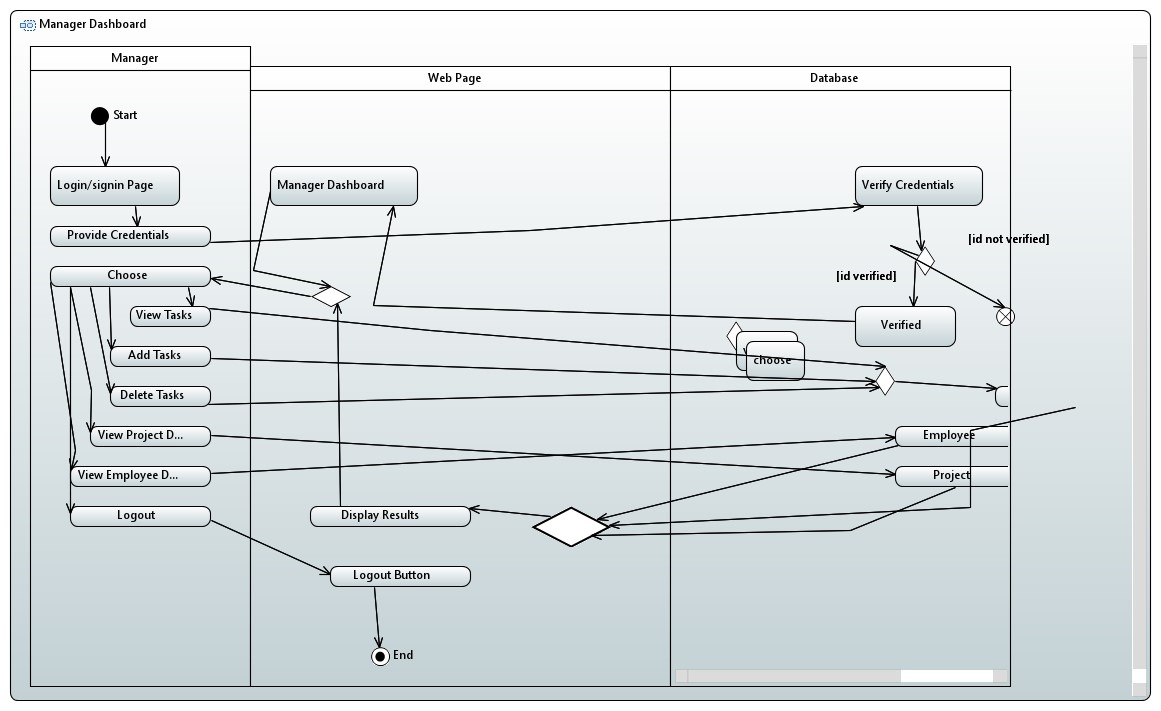
***9.1.3.3 <State Diagram n>****[Diagram & Explanation of diagram]*

***9.1.4. Activity Diagram***

***9.1.4.1 Employee Activity Diagram***

**

***9.1.4.2 Managers Activity Diagram***

**

## 10. References

1. Author, A. A., Author, B. B., & Author, C. C. (Year). Title of the article. \*Title of the Journal, volume number\*(issue number), page range. DOI or URL

2. Author, D. D., Author, E. E., & Author, F. F. (Year). Title of the second article. \*Title of the Journal, volume number\*(issue number), page range. DOI or URL

3. Author, G. G., Author, H. H., & Author, I. I. (Year). Title of the third article. \*Title of the Journal, volume number\*(issue number), page range. DOI or URL

4. Author, J. J., Author, K. K., & Author, L. L. (Year). Title of the online article. \*Title of the Journal or Website\*. Retrieved from URL

5. Author, M. M., Author, N. N., & Author, O. O. (Year). Title of another online article. \*Title of the Journal or Website\*. Retrieved from URL

6. Author, P. P. (Year, Month Day). Title of the magazine article. \*Title of the Magazine, volume number\*(issue number), page range.

7. Author, Q. Q. (Year, Month Day). Title of another magazine article. \*Title of the Magazine, volume number\*(issue number), page range.

8. Author, R. R. (Year, Month Day). Title of the newspaper article. \*Title of the Newspaper, page range\*. DOI or URL

9. Author, S. S. (Year, Month Day). Title of another newspaper article. \*Title of the Newspaper, page range\*. DOI or URL

## 11. Appendices

*[This section should include supporting detail that would be too distracting to include in the main body of the document.]*