

Library Management System

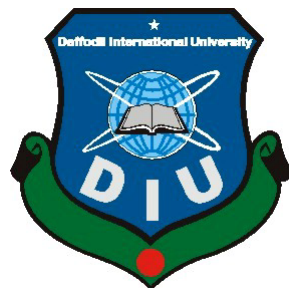
(using Python and OOP concepts)

Submitted By

Student Name	Student ID
Ramesha Rawnok Haque	0242220005101665
Kh Sadman Sakib	0242220005101951

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE222: Object Oriented Programming II Lab** in the **Computer Science and Engineering Department**



DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 15, 2024

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Nasima Islam Bithi, Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.



Submitted To:

Nasima Islam Bithi

Lecturer

Department of Computer Science and Engineering Daffodil
International University

Submitted by

	
<hr style="width: 30%; margin: 10px auto;"/> <p>Ramesha Rawnok Haue 0242220005101665 Dept. of CSE, DIU</p>	<hr style="width: 30%; margin: 10px auto;"/> <p>Kh Sadman Sakib 0242220005101951 Dept. of CSE, DIU</p>

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:.

Table 1: Course Outcome Statements

CO's	Statements
CO1	Define and Relate classes, objects, members of the class, and relationships among them needed for solving specific problems
CO2	Formulate knowledge of object-oriented programming and Java in problem solving
CO3	Analyze Unified Modeling Language (UML) models to Present a specific problem
CO4	Develop solutions for real-world complex problems applying OOP concepts while evaluating their effectiveness based on industry standards.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section 4.3.1, 4.3.2 and 4.3.3.

Table of Contents

Declaration	i
Course & Program Outcome	ii
1 Introduction	6
1.1 Introduction	6
1.2 Motivation	6
1.3 Objectives	6
1.4 Feasibility Study	7
1.5 Gap Analysis	7
1.6 Project Outcome	7
2 Proposed Methodology/Architecture	8
2.1 Requirement Analysis & Design Specification	8
2.1.1 Overview	8
2.1.2 Proposed Methodology/ System Design	8
2.1.3 UI Design	10
2.2 Overall Project Plan	10
3 Implementation and Results	12
3.1 Implementation	12
3.2 Performance Analysis	15
3.3 Results and Discussion	17
4 Engineering Standards and Mapping	18
4.1 Impact on Society, Environment and Sustainability	18
4.1.1 Impact on Life	18
4.1.2 Impact on Society & Environment	18
4.1.3 Ethical Aspects	18
4.1.4 Sustainability Plan	18
4.2 Project Management and Team Work	18
4.3 Complex Engineering Problem	19
4.3.1 Mapping of Program Outcome	19
4.3.2 Complex Problem Solving	19
4.3.3 Engineering Activities	20

Table of Contents

Table of Contents

5 Conclusion	22
5.1 Summary	22
5.2 Limitation	22
5.3 Future Work	22
Github Project Folder Link	23
References	24



Chapter 1

Introduction

This chapter provides an overview of the Library Management System project, including its background, motivation, objectives, feasibility study, gap analysis, and expected outcomes.

1.1 Introduction

Libraries are essential [1] for the preservation and dissemination of knowledge, serving as gateways to information for students, researchers, and the general public. However, traditional methods of library management often involve manual processes, which are time-consuming, prone to errors, and inefficient when handling a large collection of books, members, and administrative tasks.

This project aims to address these challenges by developing a terminal-based Library Management System that streamlines core operations such as book inventory management, member registration, borrowing and returning books, and administrative oversight. The system demonstrates the practical application of Object-Oriented Programming (OOP) principles and Python skills, showcasing the team's learning process in the OOP II course. While the current implementation is terminal-based, there is potential for future upgrades into a full-fledged desktop or web application.

1.2 Motivation

The primary motivation for developing the Library Management System is to overcome the inefficiencies of traditional manual library systems. Manual record-keeping can lead to misplaced records, errors in book allocation, and challenges in tracking borrowing histories. By digitizing these processes, we aim to reduce these issues and ensure a smooth, hassle-free library experience for both users and administrators.

Additionally, this project serves as an excellent opportunity to apply programming and database management skills in solving real-world problems. The successful implementation of this system will not only help improve library operations but also benefit the team members by enhancing their software development and project management expertise.

1.3 Objectives

The key objectives of the Library Management System are:

- To design and implement a centralized system for managing library resources.
- To automate book inventory management, member registration, and borrowing/return processes.
- To provide a secure and efficient terminal-based interface for administrators and members.
- To ensure secure access and authentication for administrative and user roles.
- To demonstrate proficiency in Object-Oriented Programming concepts and Python programming.

1.4 Feasibility Study

Several existing systems and applications serve as inspiration for this project. Web-based systems such as Koha [2] and mobile applications like Libib [3] demonstrate the potential of digital solutions for library management. However, these systems often require significant customization or may not align with the specific needs of smaller libraries. This project aims to bridge that gap by creating a system tailored to the unique requirements of academic libraries.

From a technical perspective, the project is feasible as it leverages widely available technologies such as Python and MySQL, using terminal-based operations to focus on core functionality without the overhead of graphical interfaces. This ensures the project remains manageable within the scope of the course.

1.5 Gap Analysis

While many digital library systems exist, they often lack affordability or customization options for smaller institutions. Moreover, some solutions are overly complex for libraries with limited resources or do not provide user-friendly interfaces for administrators. Our project addresses these gaps by focusing on a cost-effective, maintainable, and customizable solution that caters to the specific needs of academic libraries. Currently, the project showcases terminal-based functionality to demonstrate its working principles, leaving room for future expansion into more sophisticated platforms.

1.6 Project Outcome

The expected outcomes of the Library Management System are:

- A fully functional terminal-based software application for managing library resources efficiently.
- Improved workflow for library staff, reducing manual effort and errors.
- Enhanced user experience for library members through easy access to book catalogs and borrowing histories.
- A secure and scalable system that can be extended or modified based on future requirements.
- Comprehensive documentation that highlights the practical implementation of OOP concepts and Python programming skills learned during the OOP II course.

Chapter 2

Proposed Methodology/Architecture

This chapter outlines the proposed methodology and design architecture for the Library Management System project. It includes a detailed analysis of the requirements, design specifications, and system implementation plans.

2.1 Requirement Analysis & Design Specification

2.1.1 Overview

The Library Management System is a terminal-based application designed to simplify and automate library operations. It includes features such as book inventory management, member registration, borrowing and returning books, and administrative functions. The focus is on implementing Object-Oriented Programming principles to create a scalable and efficient system.

2.1.2 Proposed Methodology/ System Design

The proposed system follows a modular design approach, breaking down the functionalities into key components:

1. **User Management:** Handles member and administrator authentication, registration, and role-based access.
2. **Book Management:** Manages the catalog of books, including adding, updating, and retrieving book details by categories.
3. **Transaction Management:** Tracks borrowing and returning of books, ensuring accurate record-keeping and preventing conflicts.
4. **Administrative Functions:** Provides additional controls for administrators to manage library staff, librarians, and inventory.

Here is UML diagram:

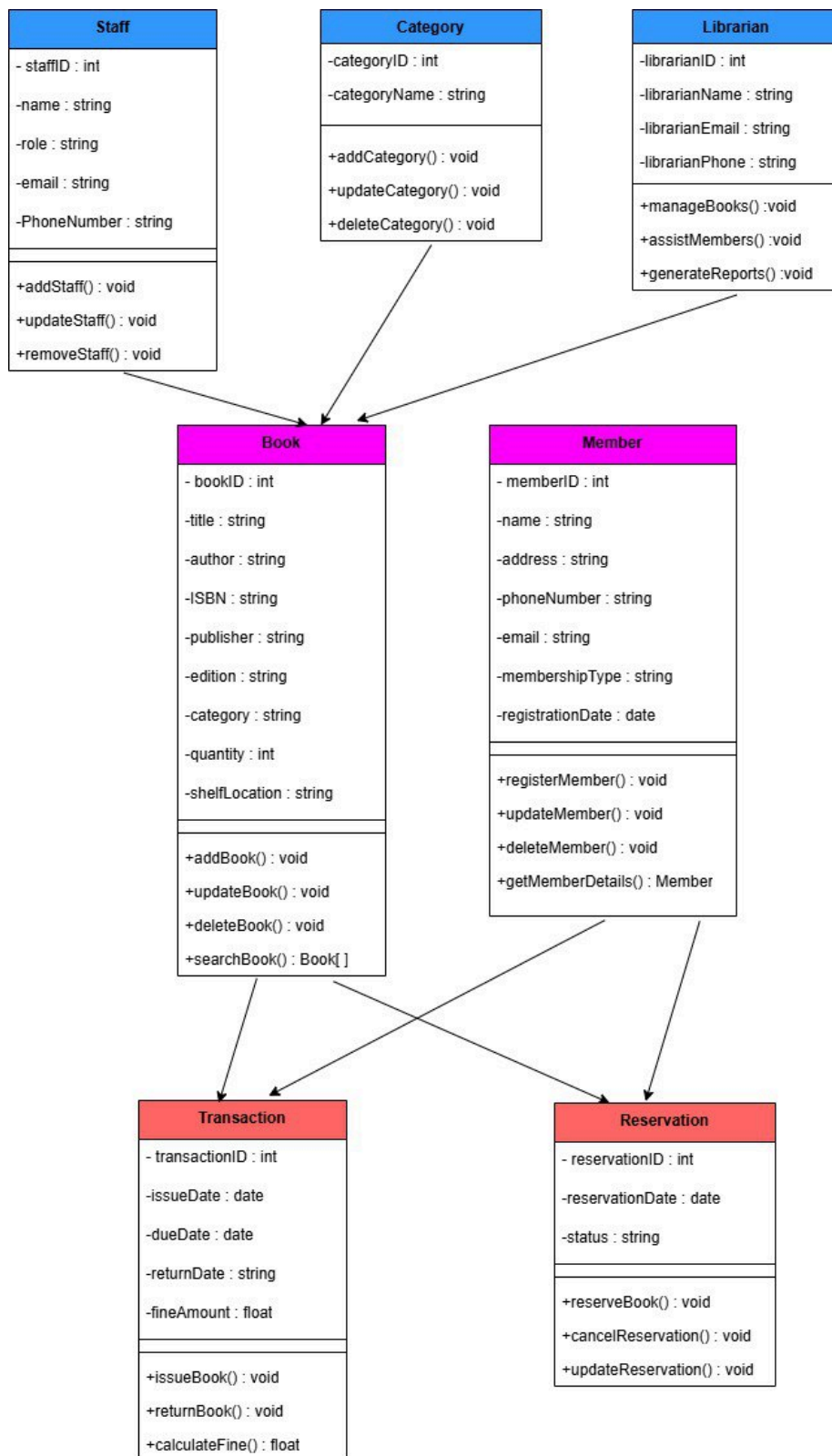


Fig 2.1: UML Diagram

2.1.3 UI Design

As this project is terminal-based, the user interface relies on simple text-based menus and prompts to guide users. Key design considerations include:

- **Clear Navigation:** Organized menu structures for both members and administrators.
- **Error Handling:** User-friendly error messages to ensure smooth interactions.
- **Role-Based Views:** Separate menu options for members and administrators.

Here is an example of the UI:

```

$$ Welcome to Library Management System $$

Choose what you want to do:
1. Sign In
2. Sign Up
3. About Us
4. Exit

3

Library Management System (v.01)

Developed by:
Kh Sadman Sakib (63_M2, ID: 1951)
Ramesha Rawnok Haque (63_M2, ID: 1665)

Submitted to:
Nasima Islam Bithi
Lecturer
Daffodil International University

Redirecting to Home...

```

Fig 2.2: UI Example

2.2 Overall Project Plan

The project plan is structured into the following phases:

1. Requirement Analysis (Week 1-2):

- Identify key features and functionalities.
- Define system requirements.

(Done by Ramesha Rawnok Haque)

2. System Design (Week 3):

- Develop class diagrams and define module interactions.
- Finalize terminal-based UI flow.

(Done by Ramesha Rawnok Haque)

3. Implementation (Week 4-6):

- Code individual modules (User Management, Book Management, etc.).
- Integrate modules into a cohesive system.

(Done by Ramesha Rawnok Haque & Kh Sadman Sakib)

4. Testing & Debugging (Week 7):

- Test individual modules and overall system flow.
- Resolve errors and ensure proper error handling.

(Done by Kh Sadman Sakib)

5. Documentation & Final Submission (Week 8):

- Prepare project report.
- Document the codebase.

(Done by Kh Sadman Sakib)

This phased approach ensures the project remains on track and aligns with the course's learning objectives.

Chapter 3

Implementation and Results

This chapter presents the detailed implementation of the Library Management System, including the core functionality developed and the performance of the system. The section also highlights the results achieved during the implementation phase, followed by a discussion of the outcomes.

3.1 Implementation

The implementation of the Library Management System (LMS) involves designing and developing various modules required to manage books, members, librarians, and staff.

The system consists of multiple modules:

- **User Authentication:** A login and sign-up system for library members and administrators is implemented. The system ensures security by storing user credentials and validating them during login.
- **Book Management:** This module allows administrators to manage book records, including adding new books, updating existing information, and deleting outdated records. It also supports the categorization of books based on genres like Fiction, Science, Technology, etc.
- **Member Management:** The members' details, including their personal information and membership type, are managed. New members can be added, and existing ones can be deleted if needed.
- **Librarian Management:** The system manages librarians, including their personal information and role.
- **Staff Management:** This module handles various staff roles, such as janitors, library assistants, and IT specialists.

The system is basically divided into two interfaces:

- **User Interface:** This section is for users, they can view or demand books.
- **Admin Interface:** This section is for administrators to manage everything else.

Below are some screenshots that describe the UI and codes.

```
1

username: sakibsidha
password: sakib2003

<< User Interface >>

Choose what you want to do:
1. View Books
2. Demand New Books
3. Log Out
```

Fig 3.1: Signed in as a User

```

1

username: rawnok-haque
password: 1234

<< Admin Interface >>

Choose what you want to do:
1. Manage Books
2. Manage Members
3. Manage Librarians
4. Manage Staffs
5. Log Out

```

Fig 3.2: Signed in as an Admin

```

from datetime import datetime, timedelta
from abc import ABC, abstractmethod

# Base class using abstraction
class Person(ABC):
    def __init__(self, name, email, phone_number):
        self._name = name
        self._email = email
        self._phone_number = phone_number

    @abstractmethod
    def display_info(self):
        pass

```

Fig 3.3: Abstraction (done by Ramesha Rawnok Haque)

```

# Custom exceptions
class LibraryException(Exception):
    pass

class BookNotAvailableException(LibraryException):
    pass

```

Fig 3.4: Exception classes (done by Ramesha Rawnok Haque)

```

# Adding staffs (dictionary)
staffs = {
    1: Staff(1, "Rakibul Islam", "raki",
    2: Staff(2, "Joy Hasan", "joyhasan",
    3: Staff(4, "Lucky Akhand", "lucky",

}

# Adding librarians (dictionary)
librarians = {
    1: Librarian(1, "Azad Hossain", "a

}

# Adding categories (tuple)
categories = (
    Category(1, "Fiction"),
    Category(2, "Science"),
    Category(3, "History"),
    Category(4, "Technology"),
)

# Adding members (list)
members = []
members.append(Member(1, "Abul Hossain",
members.append(Member(2, "Karim Ahmed"

```

Fig 3.5: Using multiple types of data structures (done by Kh Sadman Sakib)

```

# Member class
class Member(Person):
    def __init__(self, member_id, name, address, phone_number, email, memb
        super().__init__(name, email, phone_number)
        self.member_id = member_id
        self.address = address
        self.membership_type = membership_type
        self.registration_date = registration_date

    def get_name(self):
        return self._name

    def display_info(self):
        return f" ID: {self.member_id}, Name: {self._name}, Membership: {s

```

Fig 3.6: A class demonstrating inheritance, polymorphism, encapsulation (done by Kh Sadman Sakib)

```

Choose what you want to do:
1. View Books
2. Add New Book
3. Issue Book
4. Return Book
5. Show All Transactions
6. Show Demanded Books
7. Exit

`1

Error occurred: invalid literal for int() with base 10: ``1'
Try again.

```

Fig 3.7: Exception handling (done by Ramesha Rawnok Haque)

The code also contains various Python concepts like loop, recursion, conditional statements, slicing techniques and uses various data structures.

The whole code can be found here in our Github repository: [\[Source Code Link\]](#)

3.2 Performance Analysis

Performance analysis is a crucial part of evaluating the effectiveness and responsiveness of the Library Management System. The system's performance was analyzed based on several parameters, including:

- **Response Time:** The time it takes to load data, perform CRUD operations (Create, Read, Update, Delete), and execute search queries (e.g., searching for books or members).
- **System Stability:** The ability of the system to handle multiple operations without crashing or experiencing errors.
- **Scalability:** How well the system handles increasing amounts of data, such as adding hundreds of books or thousands of members.

To analyze these factors, the system was tested using various datasets of different sizes. Performance was measured before and after implementing certain optimization techniques.

More screenshots of code's performance is added below:

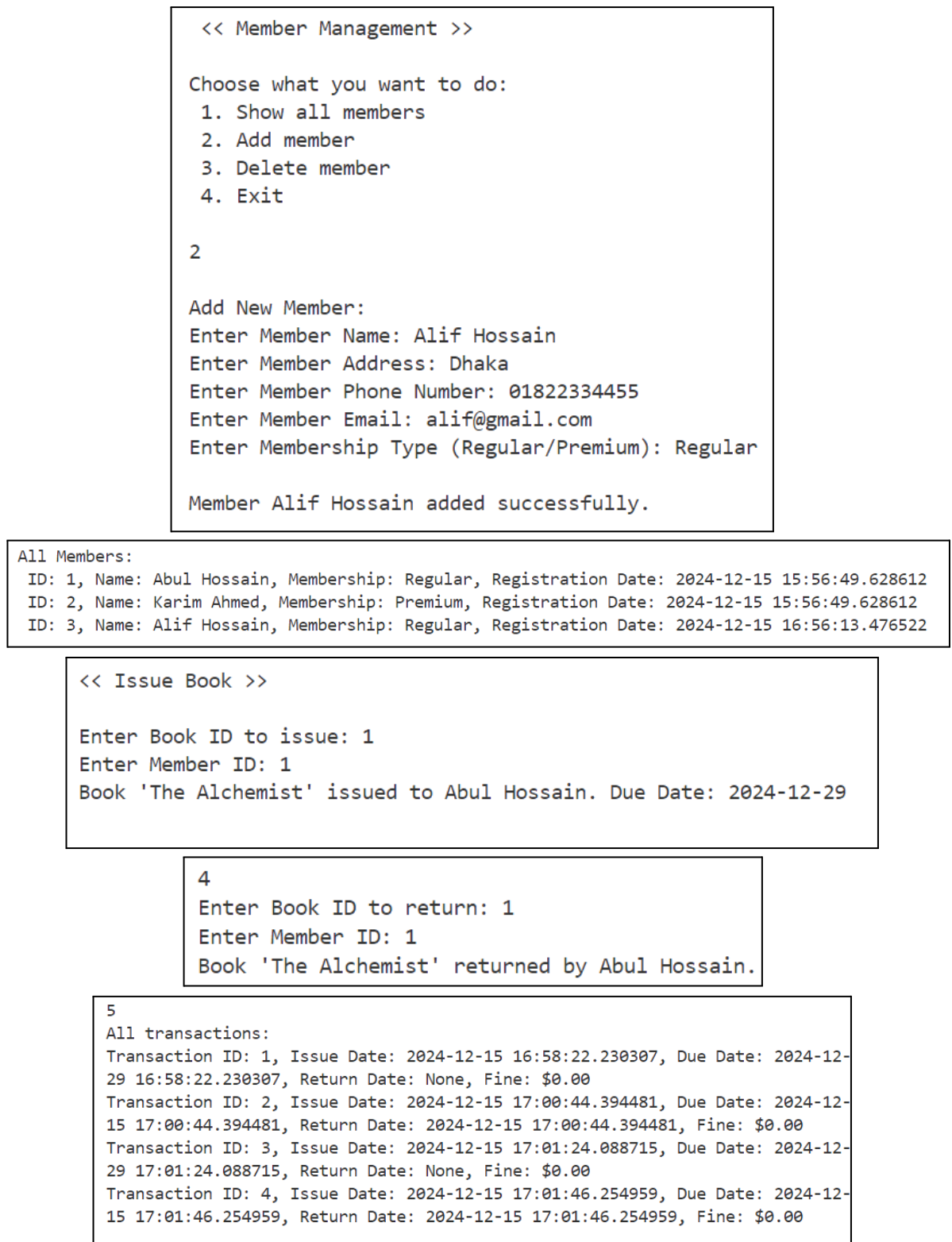


Fig 3.8: Adding, Deleting, Borrowing, Transaction Tracking (done by Kh Sadman Sakib)


```

1
username: sakib
password: kio

Credentials not found.

1. Try Again
2. Sign Up
3. Exit

2

Sign-up as:
1. Library Member
2. Administration
3. Exit

2

Enter the secret key (hint: OOPII): OOPII

Enter a username: x
Enter password: 1234
Re-enter password: 1234

Account Created Successfully! You can now Sign In with the credentials.

```

Fig 3.9: Sign-in / Sign-up in action (done by Kh Sadman Sakib)

3.3 Results and Discussion

The Library Management System was successfully implemented, and the core functionalities were tested thoroughly. Here are the key results:

- **User Authentication:** The sign-in and sign-up features were functional without errors, providing secure access to both members and administrators.
- **Book Management:** The system efficiently added, updated, and deleted books. The search functionality for categories worked as expected, and filtering by genres produced accurate results.
- **Member and Staff Management:** New members and staff were successfully added, with the system automatically assigning unique IDs. The deletion of members was performed without data integrity issues.
- **Performance:** The response time for operations was optimal for datasets of moderate size. However, as the data set grew larger (e.g., adding 1,000+ records), there was a slight increase in response times, which can be improved with database optimization in future versions of the system.

In the discussion, it is important to address any challenges encountered during the development process. For instance, ensuring the consistency of the data when deleting members or books was a critical aspect that required careful handling. Additionally, performance bottlenecks were identified when the system had to handle a large volume of data, which could be addressed through future optimizations such as indexing and more efficient searching algorithms.

Chapter 4

Engineering Standards and Mapping

This chapter outlines the engineering standards adhered to during the development of the Library Management System, focusing on its impact on society, the environment, and sustainability. It also discusses the application of engineering knowledge and complex problem-solving techniques while mapping the project to relevant Program Outcomes (POs) and engineering activities.

4.1 Impact on Society, Environment and Sustainability

The Library Management System (LMS) is designed to streamline and improve library operations, which, in turn, positively impacts society by providing easy access to resources and encouraging knowledge sharing. The system enhances user experience, simplifies administrative tasks, and improves the overall efficiency of managing books, members, and staff.

4.1.1 Impact on Life

The system benefits library users by making it easier to manage library resources. It provides a more efficient method of borrowing and returning books, searching for books by category, and managing member profiles. This contributes to enhancing the overall educational experience.

4.1.2 Impact on Society & Environment

By digitizing library management processes, the system reduces the need for physical records, promoting paperless operations and, as a result, reducing paper waste. This aligns with the goals of environmental sustainability by lowering the ecological footprint associated with traditional library management practices.

4.1.3 Ethical Aspects

The ethical considerations in this project primarily involve data privacy and security. Since the system stores sensitive user information, such as names, emails, and phone numbers, ensuring that this data is securely stored and protected against unauthorized access is paramount. The project adheres to ethical guidelines for information technology development, ensuring that user data is handled with the utmost respect and confidentiality.

4.1.4 Sustainability Plan

The Library Management System aims for long-term sustainability by providing a robust, scalable, and easily maintainable solution. Future improvements could include the integration of AI-powered book recommendation systems, enhanced search features, and the expansion to a cloud-based infrastructure for greater scalability. The system can be further optimized for larger libraries or educational institutions, ensuring that it can adapt to changing needs over time.

4.2 Project Management and Teamwork

This project was executed using agile project management techniques. The team worked collaboratively, with each member focusing on a specific module (e.g., user authentication, book management, etc.). Regular meetings were held to ensure progress, identify challenges, and implement solutions efficiently. Team members communicated regularly through project management tools and version control systems, ensuring smooth collaboration.

The task distribution can also be found here: [\[Bookmarked Link\]](#)

4.3 Complex Engineering Problem

This section highlights the complexity of the problem and solution, mapping it to the engineering knowledge required to address the challenge.

Knowledge Profile and Rational Thereof: The solution to the problem required knowledge in several areas of computer science, including:

- **Database Management:** To handle book records, member data, and transactions efficiently.
- **Software Engineering Principles:** To design the system architecture and ensure scalability and maintainability.
- **User Interface Design:** For creating a simple, intuitive interface for administrators and library members.

4.3.1 Mapping of Program Outcome

In this section, provide a mapping of the problem and provided solution with targeted Program Outcomes (PO's).

Table 4.1: Justification of Program Outcomes

PO's	Justification
PO1	The project demonstrates strong problem-solving skills, as it successfully addresses the need for efficient library management through software solutions.
PO2	The development process involved rigorous design and testing phases, ensuring that the system met all functional and non-functional requirements.
PO3	The team applied ethical principles related to data privacy and user security, ensuring the protection of personal information within the system.

4.3.2 Complex Problem Solving

This section maps the complex engineering problem to the categories of problem-solving and explains how each aspect of the problem was handled

Table 4.2: Mapping with complex problem solving.

EP1	EP2	EP3	EP4	EP5	EP6	EP7
Dept of Knowledge	Range of Conflicting Requirements	Depth of Analysis	Familiarity of Issues	Extent of Applicable Codes	Extent Of Stakeholder Involvement	Interdependence
The project applies advanced OOP concepts like abstraction, polymorphism, and exception handling to solve real-world banking challenges.	Conflicting requirements such as performance optimization and ensuring modularity were balanced while developing the system.	The system was designed after thorough analysis of requirements, leading to robust features like account management	Issues such as handling invalid user inputs and ensuring reliable exception management	Standard programming practices and OOP principles were strictly followed to ensure a maintainable and scalable system.	The project considered both technical and non-technical users, ensuring the interface was simple and easy to navigate.	The system modules, such as account management, transactions, and loan management, were designed to work interdependently to achieve efficiency.

4.3.3 Engineering Activities

This section maps the project's engineering activities with categories EA1–EA5, providing rationale for each activity.

Table 4.3: Mapping with complex engineering activities.

EA1 Range of resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
Utilized open-source tools like Python and IDLE/VS Code , ensuring cost-efficiency and accessibility to all team members.	Interaction with team members included collaborative discussions, code reviews, and systematic testing to refine system functionality.	Innovative use of OOP concepts like multiple inheritance, abstract base classes, and custom exceptions enhanced system robustness.	The project minimizes the need for manual, paper-based banking systems, thereby reducing resource waste and promoting sustainability.	Ensured familiarity with industry coding standards and OOP best practices, applying rigorous testing and validation procedures.

Chapter 5

Conclusion

This chapter concludes the report by summarizing the key outcomes of the project, highlighting its limitations, and providing directions for future improvements and enhancements.

5.1 Summary

The Library Management System was developed to address the inefficiencies of traditional library management processes by providing a streamlined, digital solution. By leveraging Python and MySQL, the system offers robust features, including book management, user management, and transaction tracking. The project emphasizes scalability, user-friendliness, and security, ensuring that it meets the needs of administrators, librarians, and library members. The successful implementation of this project demonstrates the application of software engineering principles, database management techniques, and teamwork to solve a complex problem efficiently.

The project contributes positively to society by promoting digital transformation in libraries, encouraging paperless operations, and simplifying access to resources for students and other users.

5.2 Limitation

While the Library Management System fulfills its primary objectives, there are certain limitations that were identified during the development process:

- **Scalability Constraints:** The system currently operates on a local server and lacks the capability to handle large-scale, cloud-based library networks.
- **Limited Features:** Advanced functionalities, such as book recommendation systems, overdue notifications via email, or mobile app integration, are not yet implemented.
- **Security Enhancements:** While the system uses basic security mechanisms, such as password hashing, it could benefit from more robust measures like multi-factor authentication.
- **Customization:** The system has limited options for customization, which might restrict its use in libraries with unique requirements.

5.3 Future Work

To enhance the capabilities of the Library Management System, the following improvements and features can be implemented in future iterations:

- **Cloud-Based Deployment:** Migrating the system to a cloud platform will allow multiple branches of a library to operate under a unified system and enable remote access for administrators and users.
- **Mobile Application Development:** Creating a mobile app version of the system will make it more accessible to users, allowing them to search, borrow, and renew books conveniently from their smartphones.
- **Integration of AI Features:** AI-based book recommendations and predictive analytics could be added to improve the user experience and help libraries make data-driven decisions.

- **Advanced Security Features:** Implementing features like role-based access control, multi-factor authentication, and data encryption will enhance the security of the system.
- **Expanded Reporting Tools:** Adding detailed reporting and analytics features will help administrators gain insights into library operations, book popularity, and user activity trends.
- **Support for Multiple Languages:** Enabling multilingual support will allow the system to cater to a more diverse user base.

Github Project Folder Link

[\[Github Link\]](#)

References

[1] The Importance of the Use of Libraries and the Need for a Reading Culture
ResearchGate.

Available at:

https://www.researchgate.net/publication/319946753_The_Importance_of_the_Use_of_Libraries_and_the_Need_for_a_Reading_Culture

[2] Koha: Open Source Integrated Library System
Koha Community.

Available at: <https://koha-community.org/>

[3] Libib: Library Management Platform
Libib.

Available at: <https://www.libib.com/>

General Online Resources and Technical Documentation

Various websites and forums were consulted during the implementation and development phases of the project to resolve coding challenges, learn about database optimization, and understand user interface design principles.

Thanks.