CAPSTONE PROJECT REPORT

(Project Term January-May 2023)

Library Management System

Submitted by

Name of Student : Lankadasu Naga Venkata Yaswanth
Registration Number : 12104843
Roll No. : K21CHA10
Section : CH

Course Code INT216

Under the Guidance of

Waseem Ud Din Wani

School of Computer Science and Engineering

DECLARATION

We hereby declare that the project work entitled **Library Management System**

is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Waseem Ud Din Wani, during January-May 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Name of Student : Lankadasu Naga Venkata Yaswanth
Registration Number : 12104843

(Signature of Student)

Date:

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in **Computer Science and Engineering** from Lovely Professional University, Phagwara.

Waseem Ud Din Wani
Signature and Name of the Mentor

Designation TA

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date : 3 May 2023

ACKNOWLEDGEMENT

I would like to express my sincere thanks and gratitude to my supervisor, (name), for his/her valuable guidance and support throughout this project. I would also like to thank my friends and family for their encouragement and motivation. I am grateful to the staff and members of the library for providing me with the necessary resources and information for this project. Lastly, I would like to thank God for giving me the strength and wisdom to complete this project.

Table of Contents

1. Introduction

2. Profile of the Problem. Rationale/Scope of the study (Problem Statement)

3. Existing System

- Introduction

- Existing Software

- DFD for present system

- What's new in the system to be developed

4. Problem Analysis

- Product definition

- Feasibility Analysis

- Project Plan

5. Software Requirement Analysis

- Introduction

- General Description

- Specific Requirements

6. Design

- System Design

- Design Notations

- Detailed Design

- Flowcharts

- Pseudo code

7. Testing

- Functional testing

- Structural testing

- Levels of testing

- Testing the project

8. Implementation

- Implementation of the project

- Conversion Plan

- Post-Implementation and Software Maintenance

9. Project Legacy

- Current Status of the project

- Remaining Areas of concern

- Technical and Managerial lessons learnt

10. Source Code (where ever applicable) or System Snapshots

11. Bibliography

1. Introduction

A library is a place where books and other materials are stored and made available for reading, study, or reference. A library can be of different types such as academic, public, school, special, etc., depending on its purpose and audience. A library contains various resources such as books, journals, magazines, newspapers, etc., that are organized according to certain criteria such as subject, author, title, etc. A library also has various members such as students, teachers, researchers, professionals, etc., who borrow or access these resources for their needs.

Managing a library involves various tasks such as acquiring, cataloguing, classifying, circulating, issuing, returning, reserving, renewing, etc. these resources; maintaining records of books, members, transactions, etc.; generating reports and statistics; providing services such as reference, interlibrary loan, etc.; ensuring security and maintenance of the library; etc.

These tasks require a lot of time, effort, and accuracy from the library staff.
To make these tasks easier and faster, a library management system is used. A library management system is a software application that automates and simplifies the process of managing a library. It helps to store and retrieve information related to books, authors, members, transactions, etc. in a database. It also provides a graphical user interface (GUI) that allows the users to interact with the system and perform various operations such as adding, removing, issuing, returning, searching, and viewing books; registering and logging in members; generating reports; etc. A library management system can improve the efficiency, effectiveness, and quality of the library services. It can also reduce the errors, costs, and workload of the library staff.

In this project, we have designed and implemented a library management system using python and tkinter programming and mysql as database. Python is an interpreted, high-level, *general-purpose programming language that can be used for various applications such as web development, data analysis, machine learning, etc. Python has a simple and elegant syntax that makes it easy to read and write. Python also has a large and rich set of libraries and modules that provide various functionalities and features. Tkinter is a module that can be used to create graphical user interfaces (GUI) in python. Tkinter is Python's standard GUI interface and it is easy to learn. Tkinter provides various widgets such as buttons, labels, entries, etc. that can be used to create and customize the GUI. Mysql is a relational database management system (RDBMS) that can be used to store and manipulate data in a structured and organized way. Mysql supports various data types, operators, functions, commands, etc. that can be used to perform various operations on the data. Mysql also provides various features such as security, scalability, reliability, etc.*

We have used these technologies to create a library management system that has the following features:

- Adding books: The system allows the librarian to add new books to the library by entering their details such as name, id, author, etc.
- Removing books: The system allows the librarian to remove existing books from the library by entering their id.

- Issuing books: The system allows the librarian to issue books to the members by entering their id and the member's name.
- Returning books: The system allows the librarian to return books from the members by entering their id.
- Viewing books: The system allows the librarian and the members to view all the books in the library along with their details and availability status.

The system also has a user-friendly and attractive GUI that makes it easy for the users to interact with the system and perform various tasks. The system also has a secure and reliable database that stores and retrieves all the information related to books, authors, members, transactions, etc. The system also has a robust and efficient code that ensures the smooth functioning of the system.

The project is significant because it demonstrates how python and tkinter programming and mysql database can be used to create a real-world application that can solve a practical problem. The project also helps to learn and apply various concepts and skills related to python programming, tkinter GUI development, mysql database management, etc. The project also helps to enhance creativity and problem-solving abilities.

2. Profile of the Problem. Rationale/Scope of the study

The library management system is a crucial system for managing the library's resources, including books, journals, magazines, and other reference materials. The primary goal of the system is to automate various library functions and provide quick and easy access to information for library staff and patrons.

In this project, we aim to develop a library management system using Python programming language and Tkinter library for building the user interface. The system will use MySQL as a database to store information on books, patrons, borrowing and returning, fines, and other relevant data.

The current library management system at most libraries is manual, time-consuming, and error-prone. The librarians are required to keep track of every book in the library, maintain a record of every borrower, and manage the overdue fines manually. This process is inefficient and leads to a lot of errors, which can cause delays and inconvenience for both the library staff and patrons.

Therefore, the scope of this study is to develop a library management system that can automate various library functions, including book entry, patron entry, borrowing and returning, fines, and other related tasks. The system will provide a user-friendly interface for library staff and patrons to access the library's resources quickly and efficiently. Additionally, the system will be developed using Python and MySQL database, which are easy to learn and widely used in the software development industry.

The developed system will help libraries to manage their resources efficiently, reduce errors, and improve the quality of service to patrons. The system will provide an easy-to-use interface for library staff and patrons to manage and access the library's resources, thus enhancing their overall experience.

3. Existing System

Introduction:

The existing library management system is often manual and paper-based, leading to inefficiencies and errors. It involves maintaining physical records of books, borrowers, transactions, and fines. This traditional approach is time-consuming, prone to mistakes, and lacks scalability. To overcome these challenges, a new system using Python and MySQL is proposed for efficient library management.

Existing Software:

The existing software for library management varies based on the library's requirements and size. Some popular commercial systems include Aleph, Koha, Symphony, and Evergreen. These software solutions offer modules for book cataloguing, borrower management, circulation management, and reporting. However, they can be costly and may require specialized training for implementation and maintenance.
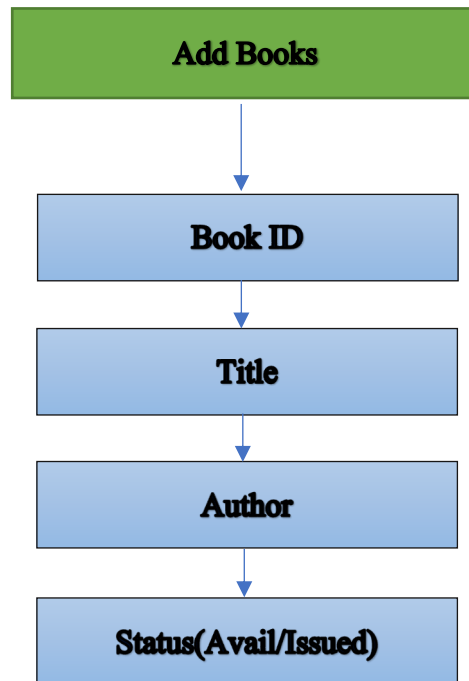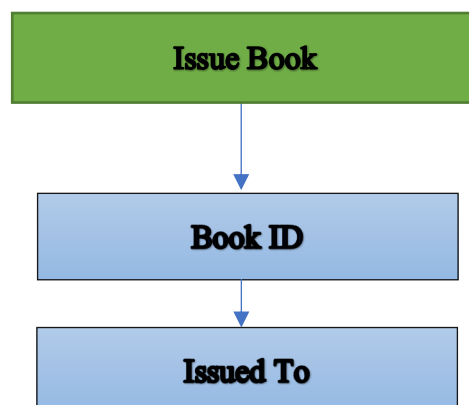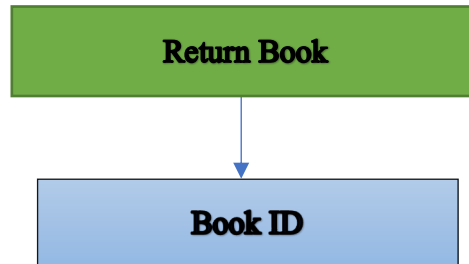
DFD for LMS:
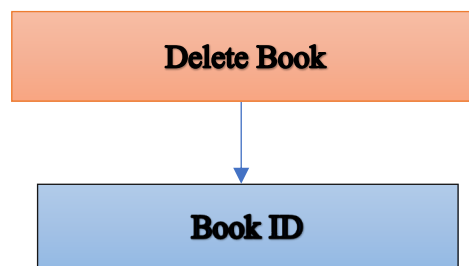
Level 0 DFD:
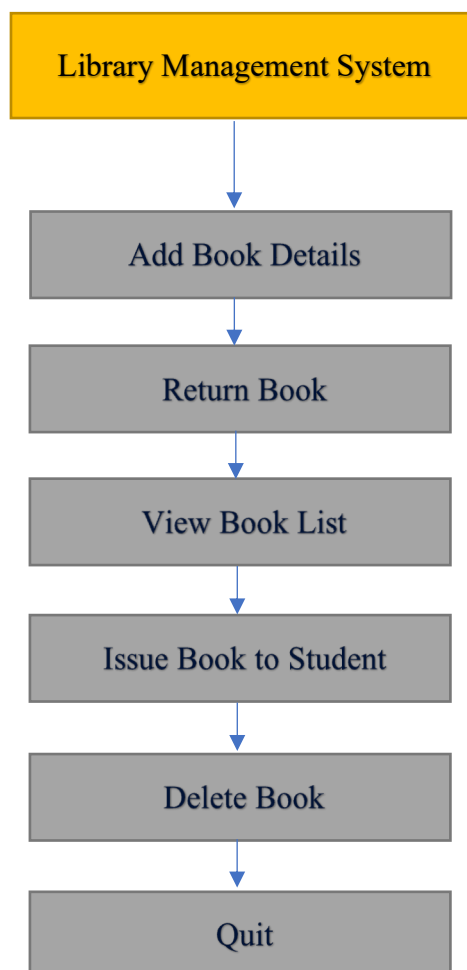
Level 1 DFD:

Add Book Details:

```
        ┌─────────────────────────────┐
        │         Add Books           │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │          Book ID            │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │           Title             │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │          Author             │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │    Status(Avail/Issued)     │
        └─────────────────────────────┘
```

Issue Book to Student:

```
        ┌─────────────────────────────┐
        │         Issue Book          │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │          Book ID            │
        └─────────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │         Issued To           │
        └─────────────────────────────┘
```

Return Book:

Delete Book:

What's New in the System to be Developed:

The proposed system will leverage Python programming language and MySQL database to develop a new library management system. It will offer several enhancements compared to the existing system:

1. Automation: The new system will automate various tasks such as book information management, patron details, book borrowing, and returning, eliminating manual processes and reducing errors.
2. User-Friendly Interface: The system will feature a user-friendly interface using the Tkinter library, making it easier for librarians to navigate and perform tasks efficiently.
3. Centralized Database: The system will utilize a MySQL database to store and manage book information, patron details, transactions, and fines. This centralized approach will ensure data integrity, scalability, and ease of access.
4. Reporting and Analytics: The system will provide reporting and analytics capabilities, allowing librarians to generate reports on book availability, overdue books, fines, and other relevant metrics. This data-driven approach will aid in decision-making and resource planning.
5. Streamlined Processes: The new system will streamline processes such as book entry, patron entry, borrowing, returning, and fine calculation, saving time and improving overall efficiency.

By incorporating these new features, the proposed library management system aims to enhance the overall management of library resources, optimize workflows, and provide an improved

4. Problem Analysis

In the library management system project using Python and MySQL, it is important to conduct a thorough problem analysis to understand the requirements, constraints, and challenges involved. The problem analysis phase typically includes the following key components:

1. Product Definition:
   - Identify the objectives and goals of the library management system.
   - Define the scope of the system, including the key functionalities it should provide.
   - Determine the target audience, such as librarians, staff, and patrons.
2. Feasibility Analysis:
   - Conduct a feasibility study to assess the viability of the project.
   - Evaluate technical feasibility, considering the required technologies, resources, and infrastructure.
   - Assess economic feasibility, including the project's cost, return on investment, and potential benefits.
   - Analyse operational feasibility, considering the impact on existing library operations and staff.
3. Project Plan:
   - Develop a detailed project plan outlining the tasks, timelines, and resource allocation.
   - Define the project milestones and deliverables.
   - Identify the roles and responsibilities of the project team members.
   - Create a risk management plan to identify and mitigate potential risks.
   - Determine the project's budget and procurement requirements.

By conducting a comprehensive problem analysis, you can gain a clear understanding of the project's requirements, feasibility, and project plan, laying a solid foundation for the successful development of the library management system.

Note: For the purpose of this response, the subsequent sections (Product Definition, Feasibility Analysis, and Project Plan) provide an overview of the steps involved in each component. In practice, these sections would require a more detailed analysis specific to the library management system project.

5. Software Requirement Analysis

Introduction:

Software Requirement Analysis is a crucial phase in the development of a library management system using Python and MySQL. It involves gathering, analysing, and documenting the functional and non-functional requirements of the system. This analysis ensures that the system meets the needs of the library and its users.

General Description:

The general description provides an overview of the library management system, its purpose, and its users. It sets the context for the specific requirements and helps understand the system's overall functionality and scope.

Specific Requirements:

The specific requirements focus on the detailed functionalities and features of the library management system. These requirements can be categorized into functional and non-functional requirements.

Functional Requirements:

1. Book Management:
   - Add, update, and delete book information.
   - Assign unique identifiers (ISBN, barcode) to books.
   - Categorize books by genre, author, and publication.
   - Track availability and location of books.
2. Borrowing and Returning:
   - Allow patrons to borrow books by scanning their library cards and book barcodes.
   - Record the borrowing date and due date for each book.
   - Send automated notifications for overdue books.
   - Enable the return of books and update the availability status.

Non-functional Requirements:

1. User Interface:
   - Design a user-friendly interface using Tkinter library for easy navigation and interaction.
2. Security:
   - Implement secure access control mechanisms to protect sensitive data.
   - Ensure data privacy and confidentiality.
3. Performance:
   - The system should handle a large volume of books, patrons, and transactions efficiently.
   - Response times should be quick to provide a seamless user experience.

4. Reliability:
    - The system should be reliable, minimizing the risk of data loss or system failures.
    - Backup and recovery mechanisms should be in place to safeguard data.
5. Scalability:
    - The system should be scalable to accommodate future growth and increasing library resources.

These specific requirements form the basis for the development and implementation of the library management system using Python and MySQL. They ensure that the system meets the needs of the library, its staff, and patrons, enabling efficient management of library resources.

6. Design

Design is a crucial phase in the development of a library management system using Python and MySQL. It involves creating a systematic plan and structure for implementing the system based on the identified requirements. The design phase includes system design, design notations, detailed design, flowcharts, and pseudo code.

1. System Design: System design focuses on defining the overall architecture and structure of the library management system. It includes the selection of appropriate technologies, databases, and frameworks. The system design ensures that the system is scalable, maintainable, and meets the functional and non-functional requirements.

2. Design Notations: Design notations help represent the system's design in a standardized and visual manner. UML (Unified Modelling Language) can be used to create diagrams such as Use Case Diagrams, Class Diagrams, and Sequence Diagrams. These notations help in understanding the relationships between different components of the system.

3. Detailed Design: Detailed design involves breaking down the system into smaller components or modules. Each module is designed in detail, including the data structures, algorithms, and interfaces. The detailed design ensures that each module performs its intended functions efficiently and effectively.

4. Flowcharts: Flowcharts provide a visual representation of the logic and flow of the system's processes. They illustrate the sequence of steps, decision points, and loops involved in different functionalities. Flowcharts help in understanding the system's behaviour and can be used as a reference during the implementation phase.

Start

[Display Menu] --> [User Input]

[Add Book]   ----> [Search Book]

[Update Book] ----> [Delete Book]
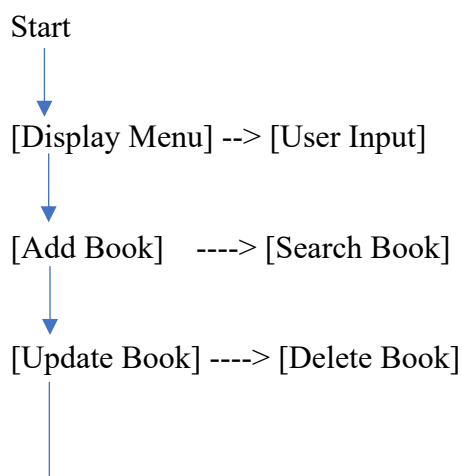
[Display Books] ----> [Exit]

End

5. Pseudo code: Pseudo code is a high-level description of the algorithms and logic used in the system. It uses a combination of natural language and programming constructs to outline the steps involved in performing a specific task. Pseudo code helps in communicating the design logic to developers and serves as a blueprint for coding the system.

Function displayMenu():
    Print "1. Add Book"
    Print "2. Search Book"
    Print "3. Update Book"
    Print "4. Delete Book"
    Print "5. Display Books"
    Print "6. Exit"

Function addBook():
    Read book details from user (title, author, ISBN, etc.)
    Connect to the MySQL database
    Execute SQL query to insert book details into the "Books" table
    Display success message

Function searchBook():
    Read search criteria from user (title, author, ISBN, etc.)
    Connect to the MySQL database
    Execute SQL query to search for books based on the criteria
    Display the search results

Function updateBook():

    Read book details from user (title, author, ISBN, etc.)

    Connect to the MySQL database

    Execute SQL query to update book details in the "Books" table

    Display success message

Function deleteBook():

    Read book ID from user

    Connect to the MySQL database

    Execute SQL query to delete the book from the "Books" table

    Display success message

Function displayBooks():

    Connect to the MySQL database

    Execute SQL query to fetch all books from the "Books" table

    Display the list of books

Main program
Display "Welcome to the Library Management System!"

Repeat:

    Call displayMenu() to show the options

    Read user's choice

    If user's choice is 1, then

        Call addBook() function

    Else if user's choice is 2, then

        Call searchBook() function

13

Else if user's choice is 3, then

Call updateBook() function

Else if user's choice is 4, then

Call deleteBook() function

Else if user's choice is 5, then

Call displayBooks() function

Else if user's choice is 6, then

Break the loop

End of program

During the design phase, it is important to consider modularity, reusability, and maintainability. The design should be well-documented and reviewed to ensure it aligns with the project requirements and best practices.

Note: Due to the text-based nature of this platform, providing specific flowcharts and pseudo code examples might be challenging. However, you can find numerous resources and tutorials online that demonstrate the design process and provide examples specific to library management systems implemented in Python and MySQL.

7. Testing

Testing is a crucial phase in the development of a library management system using Python and MySQL. It ensures that the system functions as intended, meets the specified requirements, and is free from errors and bugs. Testing can be categorized into functional testing, structural testing, and different levels of testing.

1. Functional Testing: Functional testing focuses on verifying that the system's functionalities work correctly and meet the specified requirements. It involves testing each function, feature, and user interaction to ensure they produce the expected results. For a library management system, functional testing may include scenarios like adding books, borrowing books, returning books, and generating reports.

2. Structural Testing: Structural testing, also known as white-box testing, focuses on testing the internal structure and code of the system. It ensures that the code is well-structured, follows best practices, and meets coding standards. Structural testing techniques include code reviews, unit testing, and code coverage analysis.

3. Levels of Testing: Different levels of testing are performed to ensure comprehensive testing coverage. These levels include:
- Unit Testing: It focuses on testing individual components or modules of the system in isolation to ensure they function correctly.
- Integration Testing: It verifies that the different components of the system work together seamlessly and communicate correctly.
- System Testing: It tests the entire system as a whole to ensure it meets the specified requirements and functions properly.
- Acceptance Testing: It involves testing the system with real-world scenarios and data to ensure it meets the end-users' expectations.

4. Testing the Project: To test the library management system project using Python and MySQL, you can follow these steps:

- Identify the test scenarios and create test cases for each functionality.
- Execute the test cases and document the test results, including any defects or issues found.
- Perform functional testing by simulating different user interactions and workflows.
- Conduct structural testing by reviewing the code, performing unit tests, and ensuring proper code coverage.
- Perform different levels of testing, including unit testing, integration testing, system testing, and acceptance testing.
- Use automated testing frameworks or tools to streamline the testing process and ensure consistent and repeatable results.
- Continuously monitor and track the testing progress, addressing any issues or bugs discovered during testing.

It is important to create a comprehensive test plan and test cases that cover all aspects of the library management system. Regularly reviewing and updating the test plan throughout the development process will help ensure that the system is thoroughly tested and of high quality.

8. Implementation

The implementation phase involves translating the design specifications into a working library management system using Python and MySQL. It includes writing the code, integrating components, and ensuring the system functions as intended. Additionally, a conversion plan and post-implementation activities are crucial for a successful deployment and maintenance of the system.

1. Implementation of the Project:
   - Write the code: Develop the library management system using Python programming language, incorporating the defined functionalities, database interactions, and user interface using a library like Tkinter.
   - Integrate components: Integrate the different modules and components of the system to ensure they work together seamlessly. This includes connecting the application with the MySQL database and handling data retrieval, storage, and updates.
   - Test the system: Perform thorough testing to verify the functionality, usability, and performance of the system. This includes functional testing, structural testing, and testing at different levels to identify and address any issues or bugs.
2. Conversion Plan:
   - Data Migration: If transitioning from an existing system, develop a plan to migrate data from the old system to the new library management system. Ensure that the data is accurately transferred and validated.
   - User Training: Develop a training plan to educate library staff and users on how to use the new system effectively. Conduct training sessions and provide documentation or user guides to facilitate a smooth transition.
   - Rollout Strategy: Define a strategy for deploying the new system, considering factors such as timing, user impact, and system availability. Gradually roll out the system in stages or implement it all at once, depending on the library's needs and capacity.

3. Post-Implementation and Software Maintenance:

    - User Support: Provide ongoing support to users by addressing their queries, issues, and feedback. Maintain a support system to handle user inquiries and provide timely resolutions.

    - Bug Fixing and Updates: Continuously monitor the system for any bugs or errors reported by users or identified during maintenance. Implement bug fixes and updates to enhance system stability and functionality.

    - System Enhancements: Analyse user feedback and suggestions for system improvements. Plan and implement enhancements to the library management system based on evolving needs and requirements.

    - Regular Maintenance: Perform routine maintenance activities, such as database backups, security updates, and performance optimizations, to ensure the system operates smoothly and securely.

The implementation phase requires coordination between developers, database administrators, and end-users to ensure a successful deployment and smooth transition to the new library management system. Effective project management and communication are essential to keep stakeholders informed and involved throughout the process.

9. Project Legacy

1. Current Status of the Project: Provide an update on the current status of the library management system project. Include details such as whether the system has been successfully implemented, any ongoing maintenance or support activities, and the level of user adoption.

2. Remaining Areas of Concern: Identify any unresolved issues or areas of concern that still need attention in the library management system project. This could include known bugs, performance issues, user feedback, or pending enhancements.

3. Technical and Managerial Lessons Learned: Reflect on the technical and managerial aspects of the project and highlight the lessons learned throughout the development process. This can include:
   - Technical Lessons Learned:
     - Identify any challenges encountered during the implementation and how they were addressed.
     - Discuss any technical limitations or issues faced with Python, MySQL, or other technologies used in the project.
     - Share any insights gained about best practices for developing a library management system or similar applications.
   - Managerial Lessons Learned:
     - Discuss the effectiveness of the project management approach used, including planning, coordination, and communication.
     - Reflect on the team dynamics, collaboration, and any challenges encountered.
     - Share insights on managing user expectations and gathering feedback during the project.

By evaluating the current status, identifying areas of concern, and reflecting on the lessons learned, you can gain valuable insights for future projects and ensure continuous improvement in the library management system.

10. Source Code (where ever applicable) or System Snapshots

Description of Project Files

Below are the project files you will get once you download and extract the Library project:

- **main.py** – which does function call to all other python files
- **AddBook.py** – To add the book
- **ViewBooks.py** – To View the list of books in the library
- **DeleteBook.py** – To Delete a book from library
- **IssueBook.py** – To Issue a book from library
- **ReturnBook.py** – To Return a book to the library

Description of Tables
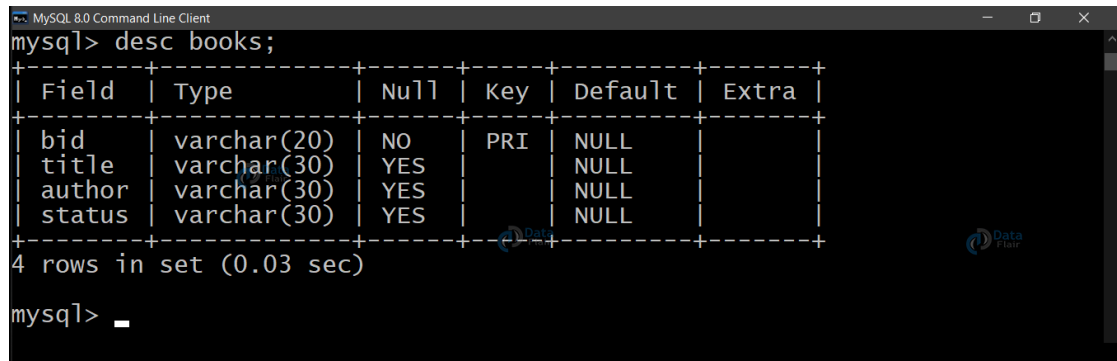
Create Tables

create database LMS;

create table books(bid varchar(20) primary key, title varchar(30), author varchar(30), status varchar(30));

create table books_issued(bid varchar(20) primary key, issuedto varchar(30));
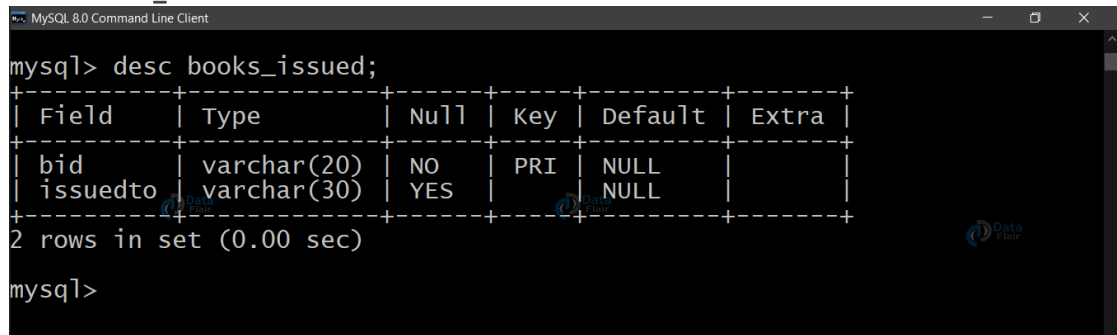
1. books

2. issued_books

1. main.py

```python
from tkinter import *
from PIL import ImageTk, Image
import pymysql
from tkinter import messagebox
from AddBook import *
from DeleteBook import *
from ViewBooks import *
from IssueBook import *
from ReturnBook import *

# Add database name and password
mypass = "yaswanth"
mydatabase = "LMS"

con = pymysql.connect(host="localhost", user="root", password=mypass,
database=mydatabase)
cur = con.cursor()

root = Tk()
root.title("Library Management System")
root.minsize(width=400, height=400)
root.geometry("684x634")

# Take n greater than 0.25 and less than 5
same = True
n = 0.25

# background image
background_image = Image.open("lpulib.jpg")
[imageSizeWidth, imageSizeHeight] = background_image.size

newImageSizeWidth = int(imageSizeWidth * n)
if same:
    newImageSizeHeight = int(imageSizeHeight * n)
else:
    newImageSizeHeight = int(imageSizeHeight / n)

# background_image =
background_image.resize((newImageSizeWidth,newImageSizeHeight),Image.
ANTIALIAS)

img = ImageTk.PhotoImage(background_image)

Canvas1 = Canvas(root)

Canvas1.create_image(300, 340, image=img)
Canvas1.config(bg="white", width=newImageSizeWidth,
height=newImageSizeHeight)
Canvas1.pack(expand=True, fill=BOTH)

headingFrame1 = Frame(root, bg="green", bd=2)
headingFrame1.place(relx=0.2, rely=0.1, relwidth=0.6, relheight=0.16)

headingLabel = Label(headingFrame1, text="Welcome to \n Yaswanth's
Library", bg='black', fg='white', font=('Times', 15))
headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

btn1 = Button(root, text="Add Book Details", bg='black', fg='green',
command=addBook)
```

21

```python
btn1.place(relx=0.28, rely=0.35, relwidth=0.45, relheight=0.1)

btn3 = Button(root, text="View Book List", bg='black', fg='green',
command=View)
btn3.place(relx=0.28, rely=0.55, relwidth=0.45, relheight=0.1)

btn4 = Button(root, text="Issue Book to Student", bg='black',
fg='green', command=issueBook)
btn4.place(relx=0.28, rely=0.65, relwidth=0.45, relheight=0.1)

btn5 = Button(root, text="Return Book", bg='black', fg='green',
command=returnBook)
btn5.place(relx=0.28, rely=0.45, relwidth=0.45, relheight=0.1)

btn2 = Button(root, text="Delete Book", bg='black', fg='red',
command=delete)
btn2.place(relx=0.28, rely=0.75, relwidth=0.45, relheight=0.1)


# function to quit the application
def close():
    root.after(100, root.destroy())


btn6 = Button(root, text="Quit", bg='black', fg='red', command=close)
btn6.place(relx=0.28, rely=0.85, relwidth=0.45, relheight=0.1)

root.mainloop()
```

2. AddBook.py

```python
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import pymysql


def bookRegister():
    bid = bookInfo1.get()
    title = bookInfo2.get()
    author = bookInfo3.get()
    status = bookInfo4.get()
    status = status.lower()

    insertBooks = "insert into " + bookTable + " values('" + bid +
"','" + title + "','" + author + "','" + status + "')"
    try:
        cur.execute(insertBooks)
        con.commit()
        messagebox.showinfo('Success', "Book added successfully")
    except:
        messagebox.showinfo("Error", "Can't add data into Database")

    print(bid)
    print(title)
    print(author)
    print(status)
```

```python
    root.destroy()


def addBook():
    global bookInfo1, bookInfo2, bookInfo3, bookInfo4, Canvas1, con,
cur, bookTable, root

    root = Tk()
    root.title("Library")
    root.minsize(width=400, height=400)
    root.geometry("684x634")

    # Add your own database name and password here to reflect in the
code
    mypass = "yaswanth"
    mydatabase = "LMS"

    con = pymysql.connect(host="localhost", user="root",
password=mypass, database=mydatabase)
    cur = con.cursor()

    # Enter Table Names here
    bookTable = "books"  # Book Table

    Canvas1 = Canvas(root)

    Canvas1.config(bg="white")
    Canvas1.pack(expand=True, fill=BOTH)

    headingFrame1 = Frame(root, bg="green", bd=5)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5,
relheight=0.13)

    headingLabel = Label(headingFrame1, text="Add Books", bg='black',
fg='white', font=('Courier', 15))
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = Frame(root, bg='black')
    labelFrame.place(relx=0.1, rely=0.4, relwidth=0.8, relheight=0.4)

    # Book ID
    lb1 = Label(labelFrame, text="Book ID : ", bg='black',
fg='white')
    lb1.place(relx=0.05, rely=0.2, relheight=0.08)

    bookInfo1 = Entry(labelFrame)
    bookInfo1.place(relx=0.3, rely=0.2, relwidth=0.62,
relheight=0.08)

    # Title
    lb2 = Label(labelFrame, text="Title : ", bg='black', fg='white')
    lb2.place(relx=0.05, rely=0.35, relheight=0.08)

    bookInfo2 = Entry(labelFrame)
    bookInfo2.place(relx=0.3, rely=0.35, relwidth=0.62,
relheight=0.08)

    # Book Author
    lb3 = Label(labelFrame, text="Author : ", bg='black', fg='white')
    lb3.place(relx=0.05, rely=0.50, relheight=0.08)
```

```
    bookInfo3 = Entry(labelFrame)
    bookInfo3.place(relx=0.3, rely=0.50, relwidth=0.62,
relheight=0.08)

    # Book Status
    lb4 = Label(labelFrame, text="Status(Avail/issued) : ",
bg='black', fg='white')
    lb4.place(relx=0.05, rely=0.65, relheight=0.08)

    bookInfo4 = Entry(labelFrame)
    bookInfo4.place(relx=0.3, rely=0.65, relwidth=0.62,
relheight=0.08)

    # Submit Button
    SubmitBtn = Button(root, text="SUBMIT", bg='#d1ccc0', fg='black',
command=bookRegister)
    SubmitBtn.place(relx=0.28, rely=0.9, relwidth=0.18,
relheight=0.08)

    quitBtn = Button(root, text="Quit", bg='#f7f1e3', fg='black',
command=root.destroy)
    quitBtn.place(relx=0.53, rely=0.9, relwidth=0.18, relheight=0.08)

    root.mainloop()
```

3. DeleteBook.py

```
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import pymysql

# Add your own database name and password here to reflect in the code
mypass = "yaswanth"
mydatabase = "LMS"

con = pymysql.connect(host="localhost", user="root", password=mypass,
database=mydatabase)
cur = con.cursor()

# Enter Table Names here
issueTable = "books_issued"
bookTable = "books"  # Book Table


def deleteBook():
    bid = bookInfo1.get()

    deleteSql = "delete from " + bookTable + " where bid = '" + bid +
"'"
    deleteIssue = "delete from " + issueTable + " where bid = '" +
bid + "'"
    try:
        cur.execute(deleteSql)
        con.commit()
        cur.execute(deleteIssue)
```

24

```python
        con.commit()
        messagebox.showinfo('Success', "Book Record Deleted
Successfully")
    except:
        messagebox.showinfo("Please check Book ID")

    print(bid)

    bookInfo1.delete(0, END)
    root.destroy()


def delete():
    global bookInfo1, bookInfo2, bookInfo3, bookInfo4, Canvas1, con,
cur, bookTable, root

    root = Tk()
    root.title("Library")
    root.minsize(width=400, height=400)
    root.geometry("684x634")

    Canvas1 = Canvas(root)

    Canvas1.config(bg="#006B38")
    Canvas1.pack(expand=True, fill=BOTH)

    headingFrame1 = Frame(root, bg="#FFBB00", bd=5)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5,
relheight=0.13)

    headingLabel = Label(headingFrame1, text="Delete Book",
bg='black', fg='white', font=('Courier', 15))
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = Frame(root, bg='black')
    labelFrame.place(relx=0.1, rely=0.3, relwidth=0.8, relheight=0.5)

    # Book ID to Delete
    lb2 = Label(labelFrame, text="Book ID : ", bg='black',
fg='white')
    lb2.place(relx=0.05, rely=0.5)

    bookInfo1 = Entry(labelFrame)
    bookInfo1.place(relx=0.3, rely=0.5, relwidth=0.62)

    # Submit Button
    SubmitBtn = Button(root, text="SUBMIT", bg='#d1ccc0', fg='black',
command=deleteBook)
    SubmitBtn.place(relx=0.28, rely=0.9, relwidth=0.18,
relheight=0.08)

    quitBtn = Button(root, text="Quit", bg='#f7f1e3', fg='black',
command=root.destroy)
    quitBtn.place(relx=0.53, rely=0.9, relwidth=0.18, relheight=0.08)


    root.mainloop()
```

4. IssueBook.py

```python
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import pymysql

# Add your own database name and password here to reflect in the code
mypass = "yaswanth"
mydatabase = "LMS"

con = pymysql.connect(host="localhost", user="root", password=mypass,
database=mydatabase)
cur = con.cursor()

# Enter Table Names here
issueTable = "books_issued"
bookTable = "books"

# List To store all Book IDs
allBid = []


def issue():
    global issueBtn, labelFrame, lb1, inf1, inf2, quitBtn, root,
Canvas1, status

    bid = inf1.get()
    issueto = inf2.get()

    issueBtn.destroy()
    labelFrame.destroy()
    lb1.destroy()
    inf1.destroy()
    inf2.destroy()

    extractBid = "select bid from " + bookTable
    try:
        cur.execute(extractBid)
        con.commit()
        for i in cur:
            allBid.append(i[0])

        if bid in allBid:
            checkAvail = "select status from " + bookTable + " where
bid = '" + bid + "'"
            cur.execute(checkAvail)
            con.commit()
            for i in cur:
                check = i[0]

            if check == 'avail':
                status = True
            else:
                status = False

        else:
            messagebox.showinfo("Error", "Book ID not present")
    except:
        messagebox.showinfo("Error", "Can't fetch Book IDs")
```

```python
    issueSql = "insert into " + issueTable + " values ('" + bid +
"','" + issueto + "')"
    show = "select * from " + issueTable

    updateStatus = "update " + bookTable + " set status = 'issued'
where bid = '" + bid + "'"
    try:
        if bid in allBid and status == True:
            cur.execute(issueSql)
            con.commit()
            cur.execute(updateStatus)
            con.commit()
            messagebox.showinfo('Success', "Book Issued
Successfully")
            root.destroy()
        else:
            allBid.clear()
            messagebox.showinfo('Message', "Book Already Issued")
            root.destroy()
            return
    except:
        messagebox.showinfo("Search Error", "The value entered is
wrong, Try again")

    print(bid)
    print(issueto)

    allBid.clear()


def issueBook():
    global issueBtn, labelFrame, lb1, inf1, inf2, quitBtn, root,
Canvas1, status

    root = Tk()
    root.title("Library")
    root.minsize(width=400, height=400)
    root.geometry("684x634")

    Canvas1 = Canvas(root)
    Canvas1.config(bg="#D6ED17")
    Canvas1.pack(expand=True, fill=BOTH)

    headingFrame1 = Frame(root, bg="#FFBB00", bd=5)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5,
relheight=0.13)

    headingLabel = Label(headingFrame1, text="Issue Book",
bg='black', fg='white', font=('Courier', 15))
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = Frame(root, bg='black')
    labelFrame.place(relx=0.1, rely=0.3, relwidth=0.8, relheight=0.5)

    # Book ID
    lb1 = Label(labelFrame, text="Book ID : ", bg='black',
fg='white')
    lb1.place(relx=0.05, rely=0.2)

    inf1 = Entry(labelFrame)
    inf1.place(relx=0.3, rely=0.2, relwidth=0.62)
```

```python
    # Issued To Student name
    lb2 = Label(labelFrame, text="Issued To : ", bg='black',
fg='white')
    lb2.place(relx=0.05, rely=0.4)

    inf2 = Entry(labelFrame)
    inf2.place(relx=0.3, rely=0.4, relwidth=0.62)

    # Issue Button
    issueBtn = Button(root, text="Issue", bg='#d1ccc0', fg='black',
command=issue)
    issueBtn.place(relx=0.28, rely=0.9, relwidth=0.18,
relheight=0.08)

    quitBtn = Button(root, text="Quit", bg='#aaa69d', fg='black',
command=root.destroy)
    quitBtn.place(relx=0.53, rely=0.9, relwidth=0.18, relheight=0.08)

    root.mainloop()
```

5. ReturnBook.py

```python
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import pymysql

# Add your own database name and password here to reflect in the code
mypass = "yaswanth"
mydatabase = "LMS"

con = pymysql.connect(host="localhost", user="root", password=mypass,
database=mydatabase)
cur = con.cursor()

# Enter Table Names here
issueTable = "books_issued"  # Issue Table
bookTable = "books"  # Book Table

allBid = []  # List To store all Book IDs


def returnn():
    global SubmitBtn, labelFrame, lb1, bookInfo1, quitBtn, root,
Canvas1, status

    bid = bookInfo1.get()

    extractBid = "select bid from " + issueTable
    try:
        cur.execute(extractBid)
        con.commit()
        for i in cur:
            allBid.append(i[0])

        if bid in allBid:
```

```python
            checkAvail = "select status from " + bookTable + " where
bid = '" + bid + "'"
            cur.execute(checkAvail)
            con.commit()
            for i in cur:
                check = i[0]

            if check == 'issued':
                status = True
            else:
                status = False

        else:
            messagebox.showinfo("Error", "Book ID not present")
    except:
        messagebox.showinfo("Error", "Can't fetch Book IDs")

    issueSql = "delete from " + issueTable + " where bid = '" + bid +
"'"

    print(bid in allBid)
    print(status)
    updateStatus = "update " + bookTable + " set status = 'avail'
where bid = '" + bid + "'"
    try:
        if bid in allBid and status == True:
            cur.execute(issueSql)
            con.commit()
            cur.execute(updateStatus)
            con.commit()
            messagebox.showinfo('Success', "Book Returned
Successfully")
        else:
            allBid.clear()
            messagebox.showinfo('Message', "Please check the book
ID")
            root.destroy()
            return
    except:
        messagebox.showinfo("Search Error", "The value entered is
wrong, Try again")

    allBid.clear()
    root.destroy()


def returnBook():
    global bookInfo1, SubmitBtn, quitBtn, Canvas1, con, cur, root,
labelFrame, lb1

    root = Tk()
    root.title("Library")
    root.minsize(width=400, height=400)
    root.geometry("684x634")

    Canvas1 = Canvas(root)

    Canvas1.config(bg="#006B38")
    Canvas1.pack(expand=True, fill=BOTH)

    headingFrame1 = Frame(root, bg="#FFBB00", bd=5)
```

```
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5,
relheight=0.13)

    headingLabel = Label(headingFrame1, text="Return Book",
bg='black', fg='white', font=('Courier', 15))
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = Frame(root, bg='black')
    labelFrame.place(relx=0.1, rely=0.3, relwidth=0.8, relheight=0.5)

    # Book ID to Delete
    lb1 = Label(labelFrame, text="Book ID : ", bg='black',
fg='white')
    lb1.place(relx=0.05, rely=0.5)

    bookInfo1 = Entry(labelFrame)
    bookInfo1.place(relx=0.3, rely=0.5, relwidth=0.62)

    # Submit Button
    SubmitBtn = Button(root, text="Return", bg='#d1ccc0', fg='black',
command=returnn)
    SubmitBtn.place(relx=0.28, rely=0.9, relwidth=0.18,
relheight=0.08)

    quitBtn = Button(root, text="Quit", bg='#f7f1e3', fg='black',
command=root.destroy)
    quitBtn.place(relx=0.53, rely=0.9, relwidth=0.18, relheight=0.08)

    root.mainloop()
```

6. ViewBooks.py

```
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import pymysql

# Add your own database name and password here to reflect in the code
mypass = "yaswanth"
mydatabase = "LMS"

con = pymysql.connect(host="localhost", user="root", password=mypass,
database=mydatabase)
cur = con.cursor()

# Enter Table Names here
bookTable = "books"


def View():
    root = Tk()
    root.title("Library")
    root.minsize(width=400, height=400)
    root.geometry("684x634")

    Canvas1 = Canvas(root)
    Canvas1.config(bg="#12a4d9")
```

```python
    Canvas1.pack(expand=True, fill=BOTH)

    headingFrame1 = Frame(root, bg="#FFBB00", bd=5)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5,
relheight=0.13)

    headingLabel = Label(headingFrame1, text="View Books",
bg='black', fg='white', font=('Courier', 15))
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = Frame(root, bg='black')
    labelFrame.place(relx=0.1, rely=0.3, relwidth=0.8, relheight=0.5)
    y = 0.25

    Label(labelFrame, text="%-10s%-40s%-30s-20s" % ('BID', 'Title',
'Author', 'Status'), bg='black', fg='white').place(
        relx=0.07, rely=0.1)
    Label(labelFrame, text="----------------------------------------
----------------------------------", bg='black',
        fg='white').place(relx=0.05, rely=0.2)
    getBooks = "select * from " + bookTable
    try:
        cur.execute(getBooks)
        con.commit()
        for i in cur:
            Label(labelFrame, text="%-10s%-30s%-30s-20s" % (i[0],
i[1], i[2], i[3]), bg='black', fg='white').place(
                relx=0.07, rely=y)
            y += 0.1
    except:
        messagebox.showinfo("Failed to fetch files from database")

    quitBtn = Button(root, text="Quit", bg='#f7f1e3', fg='black',
command=root.destroy)
    quitBtn.place(relx=0.4, rely=0.9, relwidth=0.18, relheight=0.08)

    root.mainloop()
```
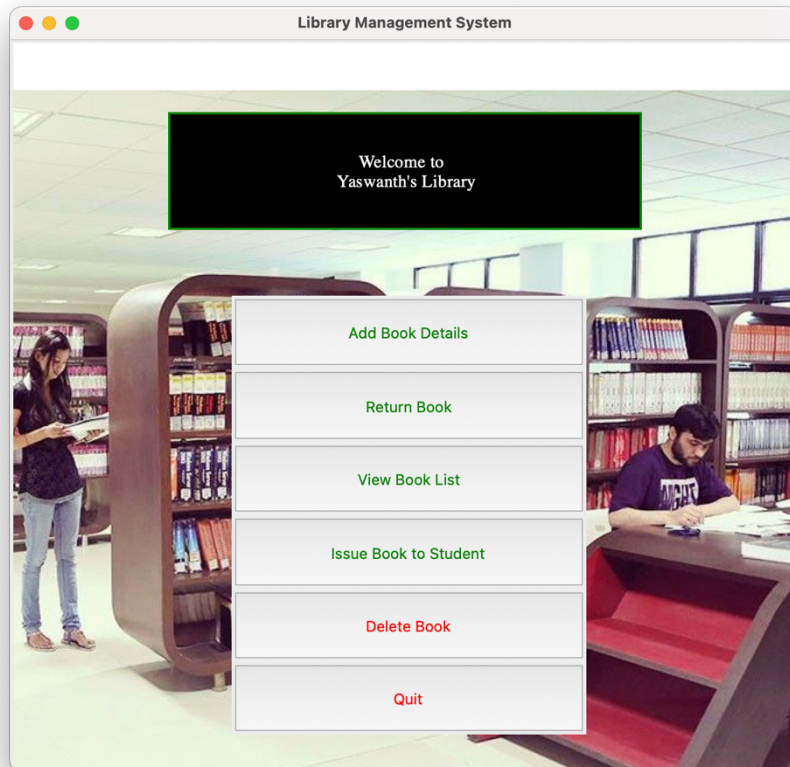
31

Input/Output

Return Book

Book ID : [   ]

[Return] [Quit]

View Books

BID	Title	Author	Status
1210	INT215	Taswa	avail
1234	jaswa TB	JASWANTH	issued

[Quit]

Library

Issue Book

Book ID : [   ]

Issued To : [   ]

[Issue]   [Quit]

Library

Delete Book

Book ID : [   ]

[SUBMIT]   [Quit]

11. Bibliography

When providing a bibliography for a project, it's important to include the sources that were consulted and referenced during the research and development process. As an AI language model, I don't have real-time access to external sources. However, I can provide you with a generic example of how a bibliography entry for a library management system project using Python and MySQL might look like:

1. IFNOSS. "Library Management Systems."at:[University College Library Management System | IFNOSS].
2. Python. "Python Programming for Beginners."[BeginnersGuide - Python Wiki].
3. MySQL Documentation. Available at: [mysql · PyPI].
4. Tkinter Documentation. Available at: [tkinter — Python interface to Tcl/Tk — Python 3.11.3 documentation].
5. Guru99. "Database Design and Implementation."at: [Database Design in DBMS Tutorial: Learn Data Modeling (guru99.com)].