LIBRARY MANAGEMENT USING PYTHON

ABSTRACT

Library management system is a project which aims in developing a computerized system to maintain all the daily work of library .This project has many features which are generally not available in normal library management systems like facility of user login and a facility of teachers login .It also has a facility of admin login through which the admin can monitor the whole system .It also has facility of an online notice board where teachers can student can put up information about workshops or seminars being held in our colleges or nearby colleges and librarian after proper verification from the concerned institution organizing the seminar can add it to the notice board . It has also a facility where student after logging in their accounts can see list of books issued and its issue date and return date and also the students can request the librarian to add new books by filling the book request form. The librarian after logging into his account i.e. admin account can generate various reports such as student report , issue report, teacher report and book report Overall this project of ours is being developed to help the students as well as staff of library to maintain the library in the best way possible and also reduce the human efforts.

INTRODUCTION

This chapter gives an overview about the aim, objectives ,back ground and operation environment of the system.

1.PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives areas follows:

- Online book issue
- Online book return
- Request column for librarian for providing new books
- A separate column for digital library
- Insertion of new book by librarian
- Student login page where student can find books issued by him/her and date of return.
- A search column to search availability of books
- A teacher login page where teacher can add any events being organized in the college and important suggestions regarding books.

2. BACKGROUND OF PROJECT

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students etc. Books and student maintenance modules are also included in this system which would keep track of the students using the library and also a detailed description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non computerized system is used. In addition, report module is also included in Library Management System. If user's position is admin, the user is able to generate different kinds of reports like lists of students registered, list of books, issue and return reports.

All these modules are able to help librarian to manage the library with more convenience and in a more efficient way as compared to library systems which are not computerized.

3.SPECIFICATIONS

Processor	Intel core for better performance
Operating system	Windows
memory	8GB or more
Hard disk space	Minimum 3GB for data base usage for future
language	Python
Version	3.9.6

4.SOFTWARE REQUIREMENTS

- Operating system-Windows 7 is used as the operating system as itis stable and supports more features and is more user friendly.
- Database MYSQL-MYSQL is used as database asset easy to maintain and retrieve records by simple queries which are in English language which are easy to understand and easy to write.
- Development tools and Programminglanguage-HTMLis used to write the whole code and develop webpages with CSS, JavaScript for styling work and php for sever side scripting.
-

5.HARDWARE REQUIREMENTS

- Intelcorei5 2ndgeneration is used as a processor because it is fast than other processors an provide reliable and stable and we can run our pc for long time. By using this processor, we can keep on developing our project without any worries.
- Ram 1gb is used as it will provide fast reading and writing capabilities and will intern support in processing.

6.USER LOGIN

Description of feature :

This feature used by the user to login into system. They are required to enter user id and password before they are allowed to enter the system. The user id and password will be verified and if invalid
id is their user is allowed to not enter the system.

Functional requirements :

- User id is provided by the register.
- The system must only allow user with valid id and password to enter the system.
- The system performs authorization process which decides what user level can access to.
- The user must be able to logout after they finished using system.

7. REGISTER NEW USER

Description of feature:

This feature can be performed by all users to register new user to create account.

Functional requirements:

- System must be able to verify information.
- System must be able to delete information if information is wrong.

8.REGISTER

New Book Description of feature:

- This feature allows to add new books to the library

Functional requirements:

- System must be able to verify information.
- System must be able to enter number of copies into table.
- System must be able to not allow two books having same book id.

9.SEARCH BOOK

Description Feature:

This feature is found in book maintenance part . we can search book based on book id ,
book name , publication or by author name.

Functional requirements:

- System must be able to search the database based on select search type.

- System must be able to filter book based on keyword entered.
- System must be able to show the filtered book in table view.

10.ISSUE BOOKS AND RETURN BOOKS

Description feature:

This feature allows to issue and return books and also view reports of book issued.

Functional requirements :

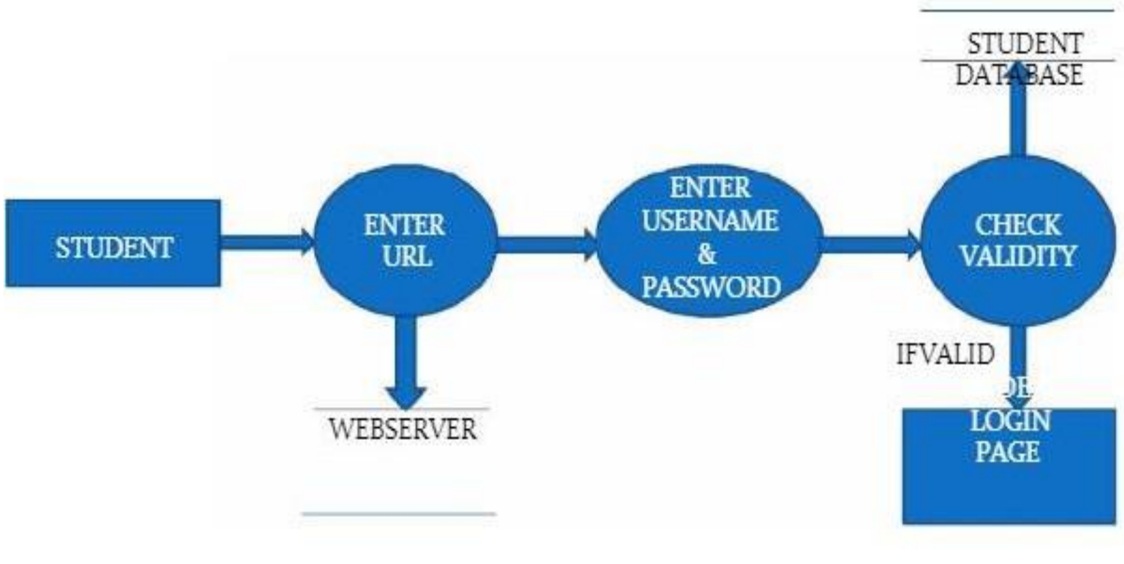
- System must be able to enter issue information in database.
- System must be able to update number of books.
- System must be able to search if book is available or not before issuing books
- System should be able to enter issue and return date information.

11. DATAFLOW DIAGRAMS

After entering to the home page of the website, teacher can choose the TEACHER LOGIN option where they are asked to enter username &password ,and if he/she is a valid user then a Teacher login page will be displayed.

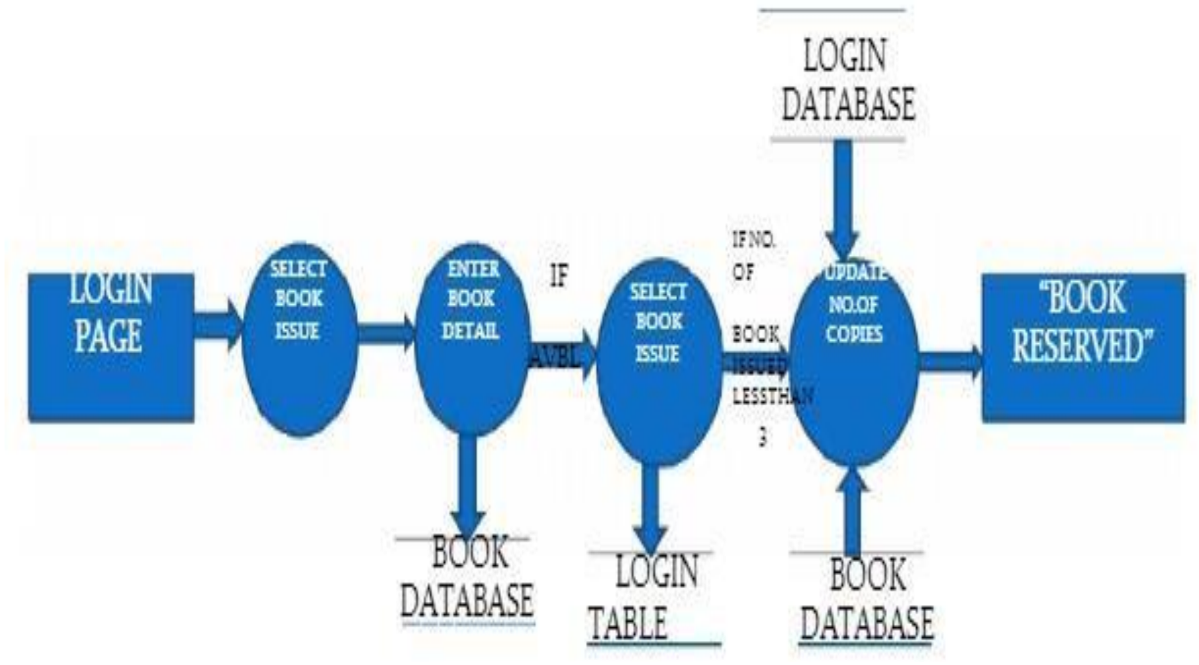
12. DATA FLOW DIAGRAM FOR STUDENT LOGIN

After entering to the home page of the website, student can choose the STUDENTLOGIN option where they are asked to enter username &password ,and if he/she is a valid user then a student login page will be displayed.



13.DATA FLOW DIAGRAM FOR BOOK ISSUE

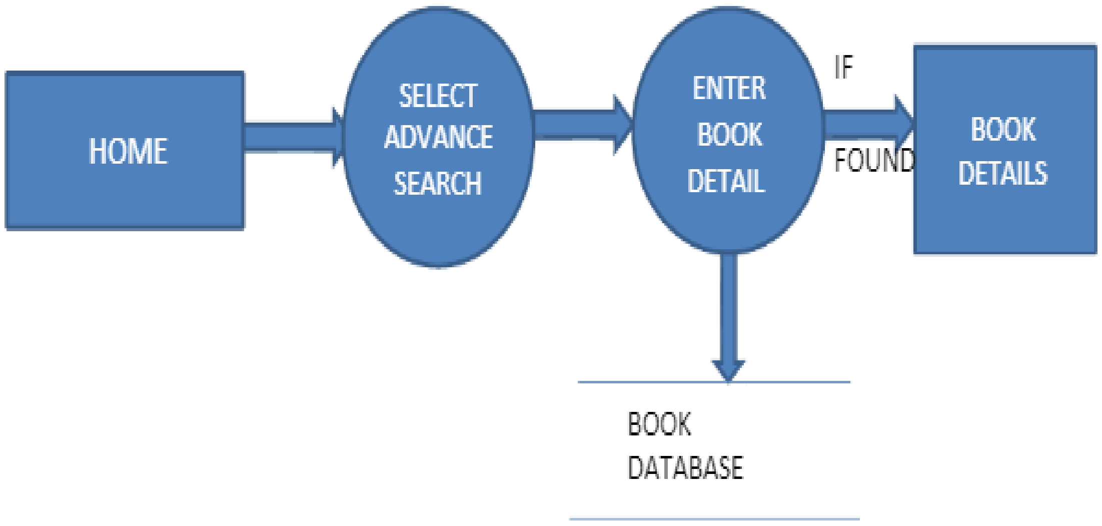
It is the second level Data Flow Diagram where after entering STUDENTLOGIN page he/she can select a book issue option where after entering the book detail, he/she can select the book issue option and if the maximum no of books issued limited books then request will be sent to the librarian who will approve the book issue.



14.DATA FLOW DIAGRAM FOR BOOK ISSUE

Data Flow Diagram is the flow of data of a system, or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow andno loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. itis useful for analyzing existing as well as proposed system.

After the home page login there will be an option of the book search whereafter entering book detail like author name, publication, book name etc. book details will be displayed.

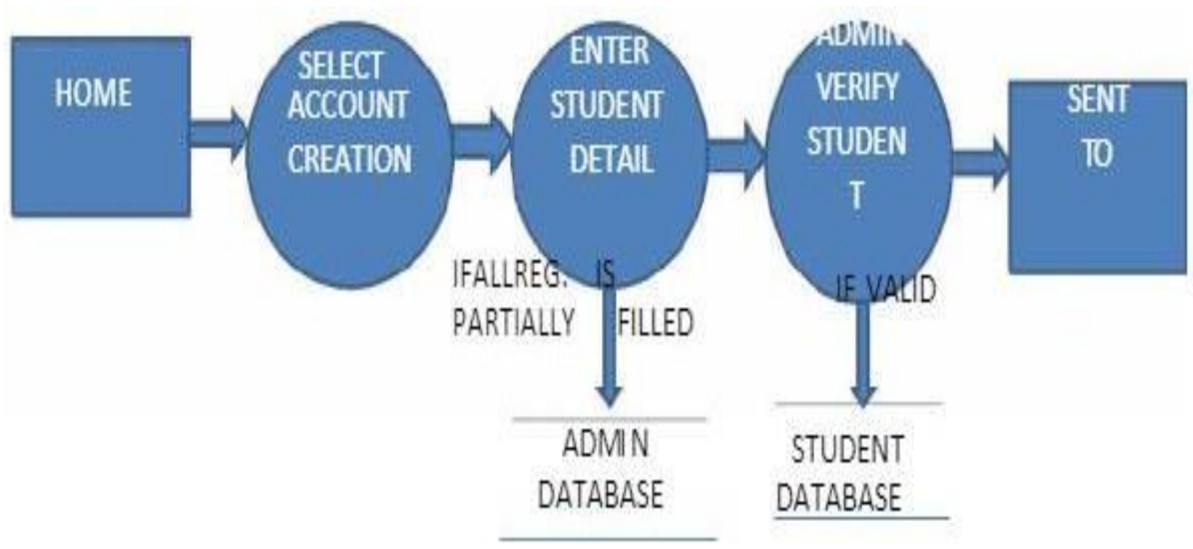


15.DATA FLOW DIAGRAM FOR ACCOUNT CREATION

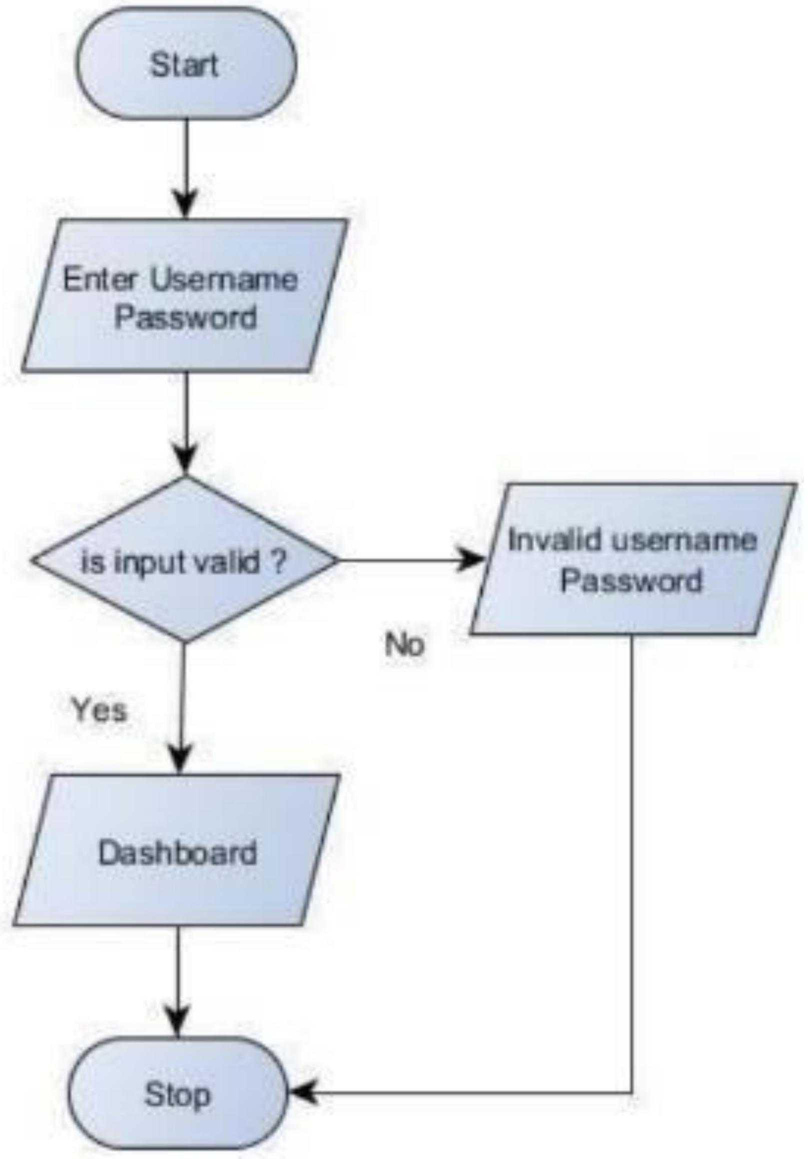
One of the methods used for login and registration system development is the DFD (data flow diagram). It represents the system's major processes and alternatives that generate the internal flow of data.

Additionally, the data was properly categorized to illustrate the login and registration system structure. Take note that DFD is not part of the Login and Registration System UML Diagrams, but they complement each other in explaining the project activities, behaviours, interactions, and structure.

After the home page login there will be an option of CREATE AN ACCOUNT where after entering student detail, if all the fields are filled then a request will be sent to the librarian who will approve him as a registered member of the library.



16.PROCEDURAL DESIGN



CODE FOR IMPLEMENTION OF LIBRARY MANAGEMENT USING PYTHON

```
import tkinter as tk
from tkinter import messagebox

class LibraryManagement:
    def __init__(self, master):
        self.master = master
        self.master.title("Library Management System")
        self.master.geometry("400x400")
        self.master.config(bg='#708090')

        self.books = []
        self.lend_list = []

        # Labels
        self.login_label = tk.Label(self.master, text="Library Management System",
font=("Helvetica", 16), bg='#708090', fg='white')
        self.login_label.pack()
        self.username_label = tk.Label(self.master, text="Username", font=("Helvetica", 12),
bg='#708090', fg='white')
        self.username_label.pack()
        self.username_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.username_entry.pack()
        self.password_label = tk.Label(self.master, text="Password", font=("Helvetica", 12),
bg='#708090', fg='white')
        self.password_label.pack()
        self.password_entry = tk.Entry(self.master, font=("Helvetica", 12), show="*")
        self.password_entry.pack()

        # Login
        self.login_button = tk.Button(self.master, text="Login", command=self.login,
font=("Helvetica", 12))
        self.login_button.pack()

        # Register
        self.register_button = tk.Button(self.master, text="Register", command=self.register,
font=("Helvetica", 12))
        self.register_button.pack()

        self.username = ""
        self.password = ""
        self.librarians = []

    def login(self):
        self.username = self.username_entry.get()
        self.password = self.password_entry.get()
        for librarian in self.librarians:
            if self.username == librarian[0] and self.password == librarian[1]:
                self.username_entry.delete(0, tk.END)
                self.password_entry.delete(0, tk.END)
                self.login_label.destroy()
                self.username_label.destroy()
                self.username_entry.destroy()
                self.password_label.destroy()
                self.password_entry.destroy()
                self.login_button.destroy()
                self.register_button.destroy()
                self.library_management_screen()
                return
        messagebox.showerror("Error", "Invalid username or password")

    def register(self):
        self.username = self.username_entry.get()
        self.password = self.password_entry.get()
        self.librarians.append([self.username, self.password])
        self.username_entry.delete(0, tk.END)
        self.password_entry.delete(0, tk.END)
    def library_management_screen(self):
        # add books
        self.add_book_label = tk.Label(self.master, text="Add Book", font=("Helvetica", 16),
bg='#708090', fg='white')
        self.add_book_label.pack()
        self.add_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.add_book_entry.pack()
        self.add_book_button = tk.Button(self.master, text="Add Book", command=self.add_book,
font=("Helvetica", 12))
        self.add_book_button.pack()
        # remove books
        self.remove_book_label = tk.Label(self.master, text="Remove Book", font=("Helvetica",
16), bg='#708090', fg='white')
        self.remove_book_label.pack()
        self.remove_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.remove_book_entry.pack()
        self.remove_book_button = tk.Button(self.master, text="Remove Book",
command=self.remove_book, font=("Helvetica", 12))
        self.remove_book_button.pack()
        # issue books
        self.issue_book_label = tk.Label(self.master, text="Issue Book", font=("Helvetica",
16), bg='#708090', fg='white')
        self.issue_book_label.pack()
        self.issue_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.issue_book_entry.pack()
        self.issue_book_button = tk.Button(self.master, text="Issue Book",
command=self.issue_book, font=("Helvetica", 12))
        self.issue_book_button.pack()
        # return book
        self.return_book_label = tk.Label(self.master, text="Return Book", font=("Helvetica",
16), bg='#708090', fg='white')
        self.return_book_label.pack()
        self.return_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.return_book_entry.pack()
        self.return_book_button = tk.Button(self.master, text="Return Book",
command=self.return_book, font=("Helvetica", 12))
        self.return_book_button.pack()
         # Search Books
        self.search_label = tk.Label(self.master, text="Search Books", font=("Helvetica",
16), bg='#708090', fg='white')
        self.search_label.pack()
        self.search_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.search_entry.pack()
        self.search_button = tk.Button(self.master, text="Search", command=self.search_books,
font=("Helvetica", 12))
        self.search_button.pack()
        self.search_result_label = tk.Label(self.master, text="", font=("Helvetica", 12),
bg='#708090', fg='white')
        self.search_result_label.pack()
        # view book
        self.view_books_button = tk.Button(self.master, text="View Books",
command=self.view_books, font=("Helvetica", 12))
        self.view_books_button.pack()


    def add_book(self):
        book = self.add_book_entry.get()
        self.books.append(book)
        messagebox.showinfo("Success", "Book added successfully")
        self.add_book_entry.delete(0, tk.END)

    def remove_book(self):
        book = self.remove_book_entry.get()
        if book in self.books:
            self.books.remove(book)
            messagebox.showinfo("Success", "Book removed successfully")
        else:
            messagebox.showerror("Error", "Book not found")
        self.remove_book_entry.delete(0, tk.END)

    def issue_book(self):
        book = self.issue_book_entry.get()
        if book in self.books:
            self.lend_list.append(book)
            self.books.remove(book)
            messagebox.showinfo("Success", "Book issued successfully")
        else:
            messagebox.showerror("Error", "Book not found")
        self.issue_book_entry.delete(0, tk.END)
    def return_book(self):
        returned_book = self.return_book_entry.get()
        if returned_book in self.lend_list:
            self.books.append(returned_book)
            self.lend_list.remove(returned_book)
            messagebox.showinfo("Success", "Book returned successfully")
        else:
            messagebox.showerror("Error", "Book not found in the lend list")
        self.return_book_entry.delete(0, tk.END)
    def search_books(self):
        query = self.search_entry.get().lower()
        search_result = [book for book in self.books if query in book.lower()]
        if search_result:
            result_message = "\n".join(search_result)
            self.search_result_label.config(text=f"Search Result:\n{result_message}")
        else:
            self.search_result_label.config(text="No matching books found")


    def view_books(self):
        message = "\n".join(self.books)
        messagebox.showinfo("Books", message)


if __name__ == "__main__":
    root = tk.Tk()
    app = LibraryManagement(root)
    root.mainloop()
```

TESTING

INTRODUCTION TO SYSTEM TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTING:

1.Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Valid input	Identified classes of valid input must be accepted.
Invalid input	Identified classes of invalid input must be rejected.
Functions	Identified functions must be exercised.
Output	Identified classes of operation outputs must be exercised.
Procedures/systems	Interface systems/procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4. System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is outputs without considering how the software works.

Features to be tested:

- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.

- All links should take the user to the correct page.

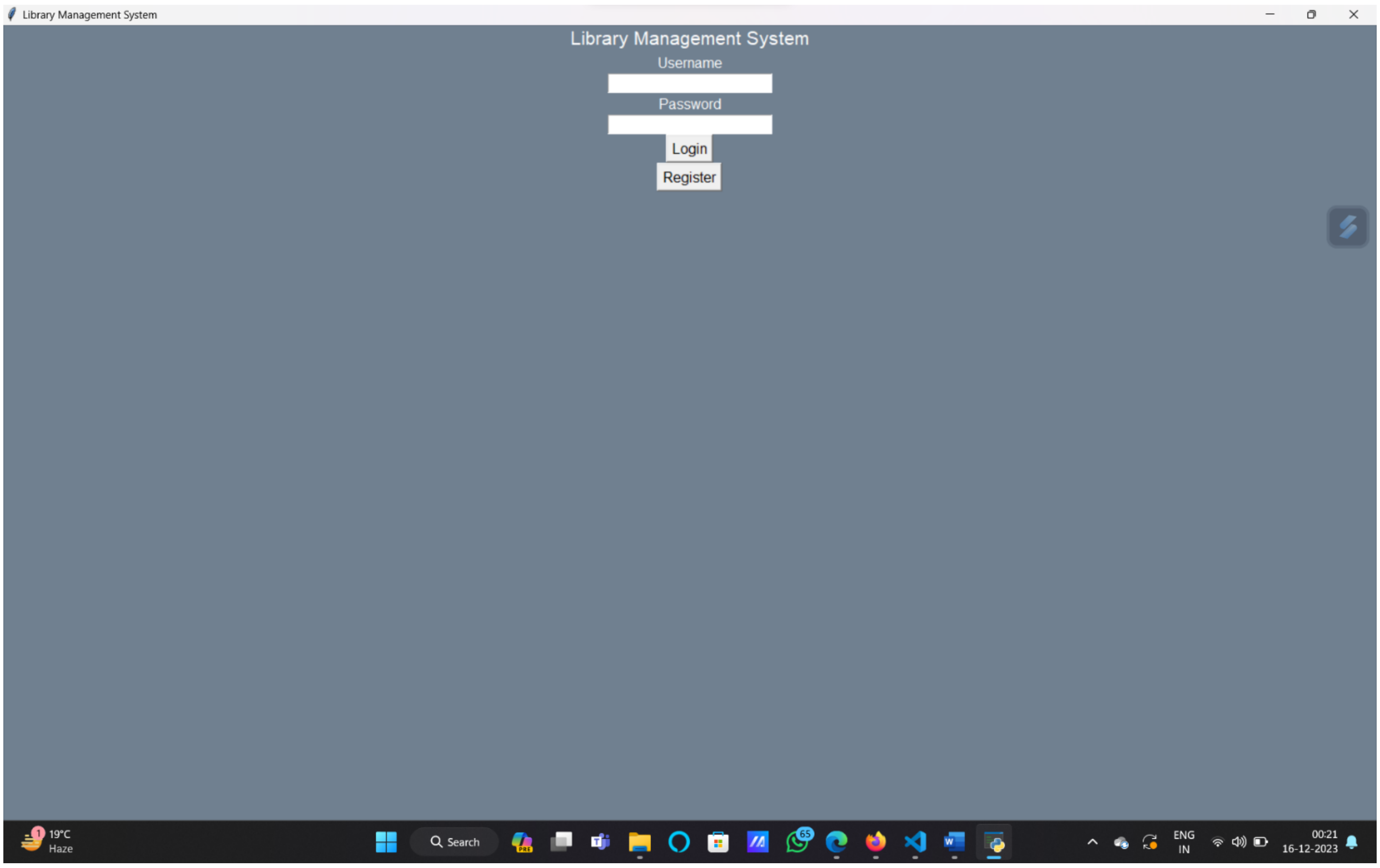
Test objectives:

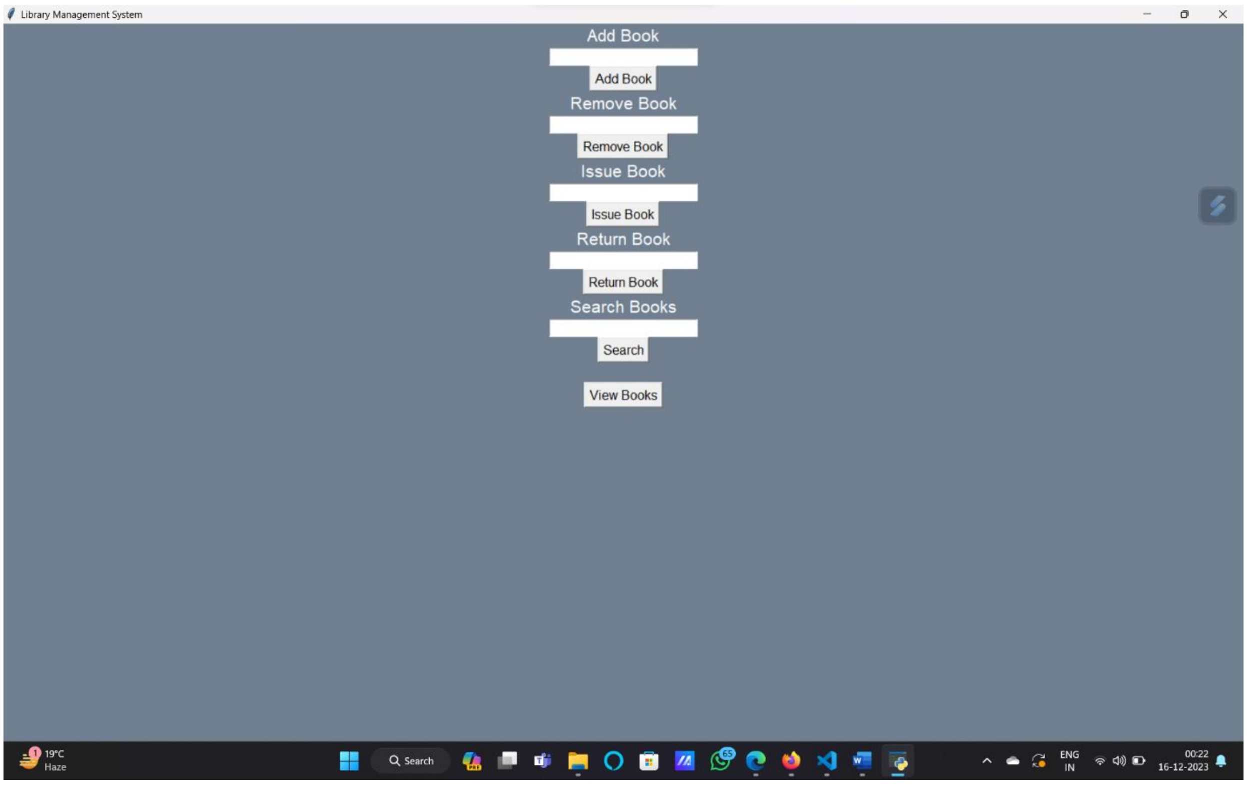
• All field entries must work properly.
• Pages must be activated from the identified link.
• The entry screen, messages and responses must not be delayed.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

RESULTS:




Library Management System
Add Book
Add Book
Remove Book
Remove Book
Issue Book
Issue Book
Return Book
Return Book
Search Books
Search
View Books

CONCLUSION

In conclusion, from proper analysis and assessment of the designed system it can be safely concluded that the system is an efficient, usable, and reliable Library Management System. Itis working properly and adequately meets the minimum expectations that were for it initially. The new system is expected to give benefits to the users and staff in terms of efficiency in the usage of library system.

Creating a library management system using Python presents an efficient and organized way to handle various library operations. The system allows for easy cataloging, tracking, and managing of books, borrowers, and transactions. This project aids librarians in streamlining tasks, enhancing user experience, and maintaining accurate records.

Through the implementation of Python, developers can harness its flexibility and extensive libraries, such as SQL Alchemy for database management, Flask or Django for web development, and Tkinter or PyQt for creating a user-friendly interface. Leveraging these tools enables the system to perform functions like adding new books, managing inventory, issuing and returning books, generating reports, and handling user accounts efficiently.

REFERENCES

[1].HonghaiKan,Zhimin Yang, Yue Wang, Nana Qi, "Research on Library Management System for CDs Attached to Books Based on Cloud Computing", in Proceedings of the 14th International Conference on Computer Supported Cooperative Work in Design 2010.
[2].Bao Sun, JiangweiFeng and Ling Liu, "A Study on How to Construct the Prediction Model of Library Lending of University Library", International Conference on Information Science andTechnology March 26-28, 2011 Nanjing, Jiangsu, China.
[3].Singh, V.,"Expectation versus experience: librarians using open source integrated library systems", The Electronic Library, 32 (5), 688-709(2014).
[4]. Ching-yu Huang and Patricia A.,"MorrealeA Web based, Self-Controlled Mechanism to Support Students Learning SQL" IEEE Integrated STEMEducation Conference (ISEC)2016.