

BANKE BAGESHWORI CAMPUS

Tribhuvan University

Institute of Science and Technology



A PROJECT REPORT

ON

ONLINE LIBRARY MANAGEMENT SYSTEM USING

BINARY SEARCH ALGORITHM

Submitted to:

Department of Computer Science and Information Technology

Banke Bageshwori Campus

Nepalgunj, Banke, Nepal

*In partial fulfillment of the requirements for the Bachelor's Degree in
Science and Computer Science and Information Technology*

Under the supervision of

Mr. Ashok Chand

Submitted by:

Bishwas Bagale (17017/074)

Nisha Shah (17027/074)

Arjun Nepali (17012/074)

March, 2022

Acknowledgment

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. We are profoundly grateful to our supervisor Mr. ASHOK CHAND whose help, simulating suggestions and encouragement helped us in all time of fabrication process and in writing this report. We also sincerely thanks for time spent proofreading and correcting our mistakes. His continuous inspiration and reviews has make us complete this project and achieve its target.

We would also like to express our gratitude and deepest appreciate to Mr. BIRENDRA SINGH SAUD. Chief of Banke Bageshwori College, Department of Computer Science and Information Technology, for his constant motivation, support and providing us with a suitable working environment.

We would also like to acknowledge with much appreciation to all staffs members of BSC. CSIT department who directly and indirectly extended their hands on completion of this project in its presently.

Abstract

Library management system is a project whose main aims to remove manual library system which is more time consuming and difficult to maintain also. Here we proposed this system for easy and fast access and availability of books online. Mainly dedicating in developing a computerized system to maintain and manage 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, a facility of librarian login and admin login .It also has a facility of admin login through which the admin can monitor the whole system .It also has facility of group chat on which everyone in the system such as user, admin and librarian can see and send any kind of message online. It also has facility of user feedback which is only see and reply be admin. It has also a facility where user after logging in their accounts can see list of books they also can search according to title. On feedback users can recommend and suggest admin for adding book as requirements. The librarian after logging into his account then he has access to group chat, add book, recent added books, manage books, delete request but admin have two extra access to system user feedback and manage user. 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. Finally, this proposed system is very easy to use and even a novice users can use the system in an efficient and error free manner. Online Library Management System is a system which maintains the information about the books present in the library, books titles, book id and book cover. This is very difficult to organize manually. It reduces the workload of management as most of the manual work done is reduced.

Contents

Acknowledgment.....	i
Abstract.....	ii
List of Abbreviations	v
List of Figures.....	vi
List of Tables	vii
Chapter 1: Introduction	1
1.1 Introduction.....	1
1.2 Problem Statement.....	2
1.3 Objectives.....	2
1.4 Scope and Limitation.....	3
1.4.1 Scope.....	3
1.4.2 Limitation	3
1.5 Development Methodology	4
1.6 Report Organization	5
Chapter 2: Background Study and Literature Review	6
2.1 Literature Review	6
Chapter 3: System Analysis	7
3.1 System Analysis.....	7
3.1.1 Requirement analysis.....	7
3.1.2 Feasibility Study.....	11
3.1.3 Analysis.....	13
Chapter 4: System Design	19
4.1 Design.....	19
4.1.1 Deployment Diagrams	19
4.2 Algorithm.....	20
Chapter 5: Implementation and Testing.....	24
5.1 Implementation	24
5.1.1 Tools Used.....	24
5.1.2 Implementation Details of Modules.....	24
5.2 Testing.....	27
5.2.1 Test Cases for Unit Testing	27
5.2.2 Test Cases for System Testing.....	28
5.3 Result Analysis.....	28

Chapter 6: Conclusion and Future Recommendation	29
6.1 Conclusion	29
6.2 Future Recommendation.....	29
References.....	30
Appendices.....	31
Screenshots	31
Coding.....	34

List of Abbreviations

• DLMS	Digital Library Management System
• VS code	Visual Studio Code
• MVT	Model-View-Template
• UML	Unified Modeling Language
• PC	Personal Computer
• CPU	Control Processing Unit
• GB	Gigabyte
• RAM	Random Access Memory
• LED	Light-emitting diode
• TB	Terabyte
• HFT	High Frequency Trading
• TC	Test Cases
• HTML	Hyper Text Markup Language
• CSS	Cascading Style Sheet

List of Figures

Figure 1: Waterfall Model.....	4
Figure 2: Use Case Diagram Of Student.....	8
Figure 3: Use Case Diagram Of Librarian	9
Figure 4: Use Case Diagram Of Librarian	9
Figure 5: Class Diagram.....	14
Figure 6: Activities Diagram Of Admin.....	16
Figure 7: Activities Diagram Of Librarian.....	17
Figure 8: Activities Diagram Of Student	18
Figure 9: Deployment Diagram	19
Figure 10: Flow Chart Of Insertion Sort.....	21
Figure 11: Flow Chart Of Binary Search	23
Figure 12: Student Registration	31
Figure 13: Login Interface	32
Figure 14: Student Interface	32
Figure 15: Librarian Interface	33
Figure 16: Admin Interface.....	33

List of Tables

Table 1: Gantt Chart	12
Table 2: Tools And Technology	24
Table 3: Unit Testing	27
Table 4: System Testing.....	28

Chapter 1: Introduction

1.1 Introduction

As all are living in the digital age, everyone prefers quick and accurate service. Online Library Management System is an automated library system that handles the various functions of the library. Library plays an important role in all schools and colleges, no educational institution can exist without Library Administration Software. It is an important part of every school, college and organization and it helps the librarian to keep records of available books as well as issued books. Library Management System software helps in different ways by providing students the facility to learn, gather resources, promote group learning and improve knowledge and skills. Admin and librarian can add book to the system as well as student can read and download any book [1].

In this system three users are in system admin, librarian and third one is student. Admin has all authority of group chat, add book, recent added book, manage books, delete request, user feedback and manage user. librarian has all access on the system except user feedback and manage user. We used binary search for searching books by its title. group message can be seen by all users of system with message date. this system is also easy to understand and use. This system can operate a library with efficiency and at reduced costs. The system entirely automated streamlines all the tasks also operations of the library. Such software eliminates the need for repetitive manual work and minimizes the chances of errors. Without any cost student can download required books. The system saves time for both the user and the librarian. With just a click the user can search for the books available in the library. System provided information for the availability of books that provides book if it available. Adding, removing or editing the database is a simple process done by admin and librarian. Adding new members or cancelling existing memberships can be done with ease by admin.

The library management system software makes the library a smart one by organizing the books systematically by title and id. This enables users to search for books quickly and effortlessly.

1.2 Problem Statement

In existing system all the transaction of books are done manually, So taking more time for a transaction like borrowing a book or returning a book and also for searching of members and books.

Going to physical library to searching books and reading books manually is so complicated and time consuming, also publishers of book haven't any better platform to add their book so according to this consequences our system would be solution.

We need to maintain the record of books and retrieve the details of books available in the library which mainly focuses on basic operations in a library like adding new member, new books, searching books. It features a familiar and well thought-out, an attractive user interface, combined with strong searching, insertion. makes users possible to download and add books. User can easily register, read and download the available books from the, this system library.

Keeping each record and providing books to each required users manually is so complicated and time consuming.

1.3 Objectives

The objectives of online library management system as below:

- Online book reading.
- A search column to search availability of books.
- Facility to download required book.

1.4 Scope and Limitation

1.4.1 Scope

The scope of online library management system as:

- Provide the list of books the student can download.
- It provides “better and efficient” service to student.
- Reduce the workload of librarian.
- Faster retrieval of information about the desired book.
- Provide facility for proper monitoring reduce paper work and data security
- All details will be available on a click.
- Fast access to database (quick transaction)
- Search facility
- The student can take of use his required books.

1.4.2 Limitation

The limitation of online library management system as:

- Requires internet connection.
- Requires technical knowledge to use the application.
- Costly and expensive
- Complicated to operate
- Risk of computer virus

1.5 Development Methodology

The waterfall model is a classical model used in system development life cycle. It creates system with a linear and sequential approach. It's each phase start with one after another. This model is divided into different phases and the output of one phase is used as the input of next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases. This model takes the fundamental process activities of specification development, validation and evolution and represents them as separate process phase such as requirement specification, design, implementation, testing and so on.

The waterfall model emphasizes a logical progression of steps. Goals are set for each phase of development and can't be revisited after completion.

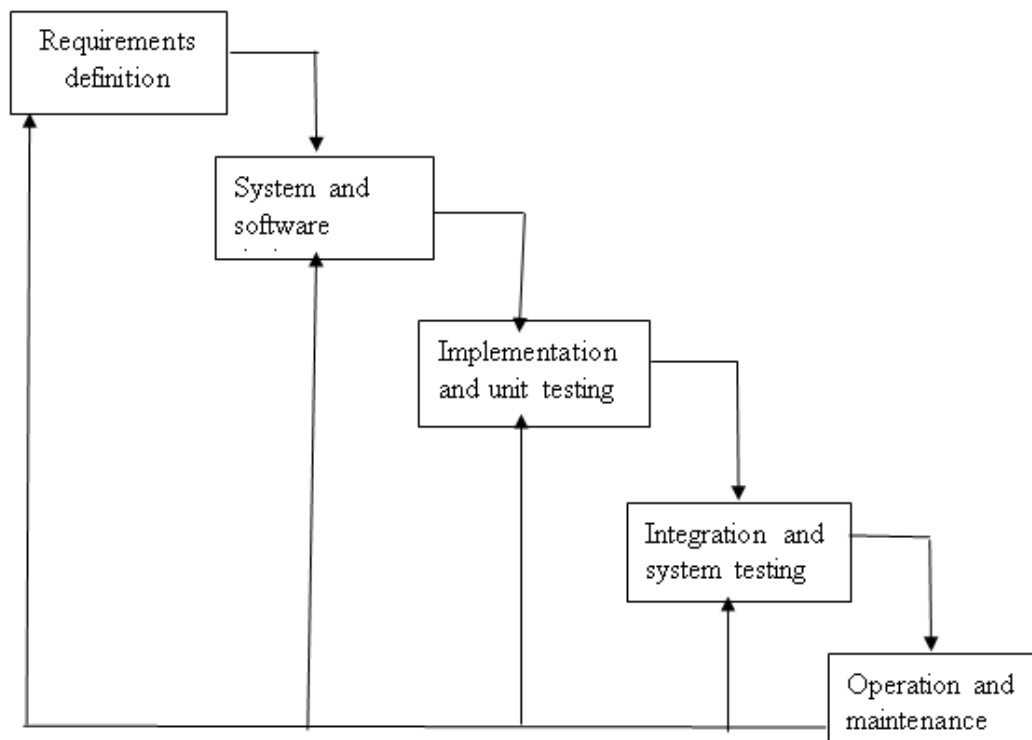


Figure 1: Waterfall Model

1.6 Report Organization

This report is organized into 6 chapters, which are as follows

In the first chapter, the project is introduced in detail along with the problem statement, its objectives and scope.

The second chapter consists of the background study and literature review of the system

The third chapter contains the functional and non-functional requirements of the project. Feasibility analysis provides analysis and evaluation of the project. In addition, data modeling and process modeling of the project are done to analyze the data and working mechanism of the system in detail.

The fourth chapter looks into detail about system design of the project that includes interface design and process design among others.

The fifth chapter consists of implementation of the system and testing that system.

The sixth chapter consists the conclusion and recommendation of this project.

Chapter 2: Background Study and Literature Review

2.1 Literature Review

Library Management System provides knowledge and practical perspectives all aspects of management of libraries and information services, which will prove invaluable to managing a library or information services cost effectively, while meeting the needs of its users. Library Management System is one of the most tedious processes, which involves the regular updating of many files and records [2].

A University Library Management System is a software that manages and stores book information electronically according to library managers', staff's, and student's needs. The system helps them to keep a constant track of all books available in the library; be it the ones borrowed or not. It provides the ability to search for the desired books as it is usually necessary for universities to keep a continuous check on the books issued by providing the issue date, last book return date, and even calculate fine which would be a tedious task to achieve with the manual system and the chance of various types of mistake occurrence would be high [3].

The design of DLMS has evolved in the past decade with the introduction of low-barrier systems, robust and flexible opensource repository systems, and more specialized options available to specific user communities like archives and museums [4].

Chapter 3: System Analysis

3.1 System Analysis

System analysis is the process of studying at the system, breaking apart the parts, and figuring out how it works in order to achieve a goal. In system analysis we observe the system for troubleshooting or development purpose

3.1.1 Requirement analysis

Requirement analysis is the process of analysis, studying and identifying the user requirement and expectations towards the systems. It also means to analyze, document, validate and manage software or system requirement. There are two types of requirements, which are below:

I. Functional Requirement

Functional requirement is mandatory requirement of the system. That must be having on the system, this describes the features of the systems. It defines what a product must do, what its features and functions are. These requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So It's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions.

a) User registration:

Before login into the system, student need to register it's self by providing his/her username, email and password. Registration of librarian is done by admin.

b) User login:

This feature used by the user(student/librarian/admin) to login onto the system. User must enter user id and password before entering in the system given during the registration phase. The user id and password will be verified according to registered details. If registered and entering id

password get match then user get access into the system otherwise error message is present.

c) Add book:

This feature allows adding new books to the system. Only admin and librarian have authority to add book to the system.

d) Search Book:

In this system search book based on book title. System must be able to filter book based on keyword entered. System must be able to show the filtered book in table view.

e) Group Chat:

In this system group chat is function as normal group chat. Everyone in system can sent message on the group chat and can view message also.

f) feedback:

Feedback which are sent from the student can only see by admin.

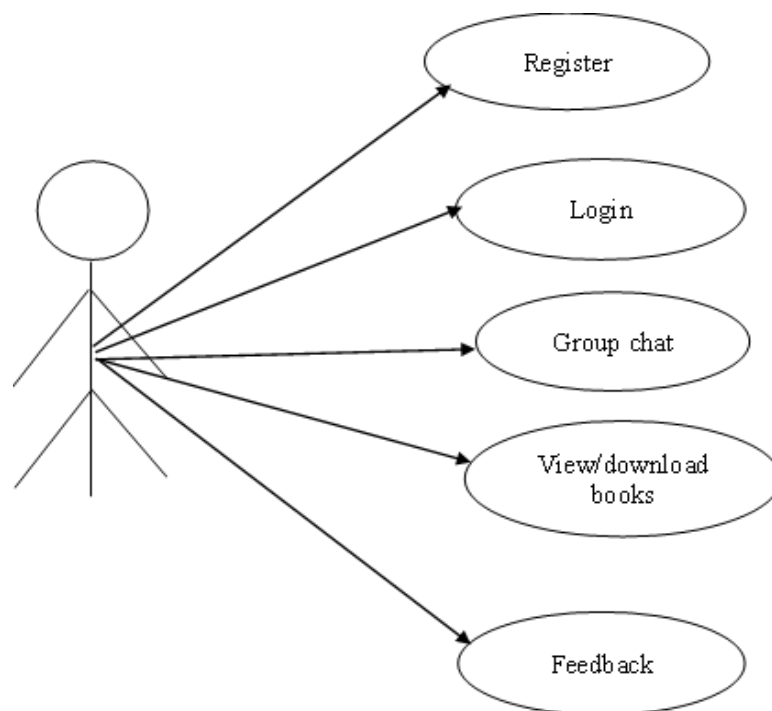


Figure 2: Use Case Diagram Of Student

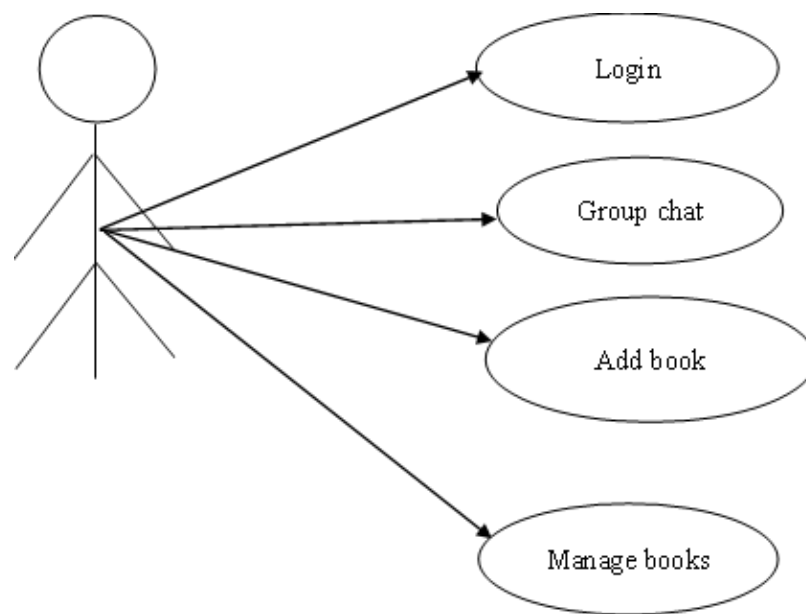


Figure 3: Use Case Diagram Of Librarian

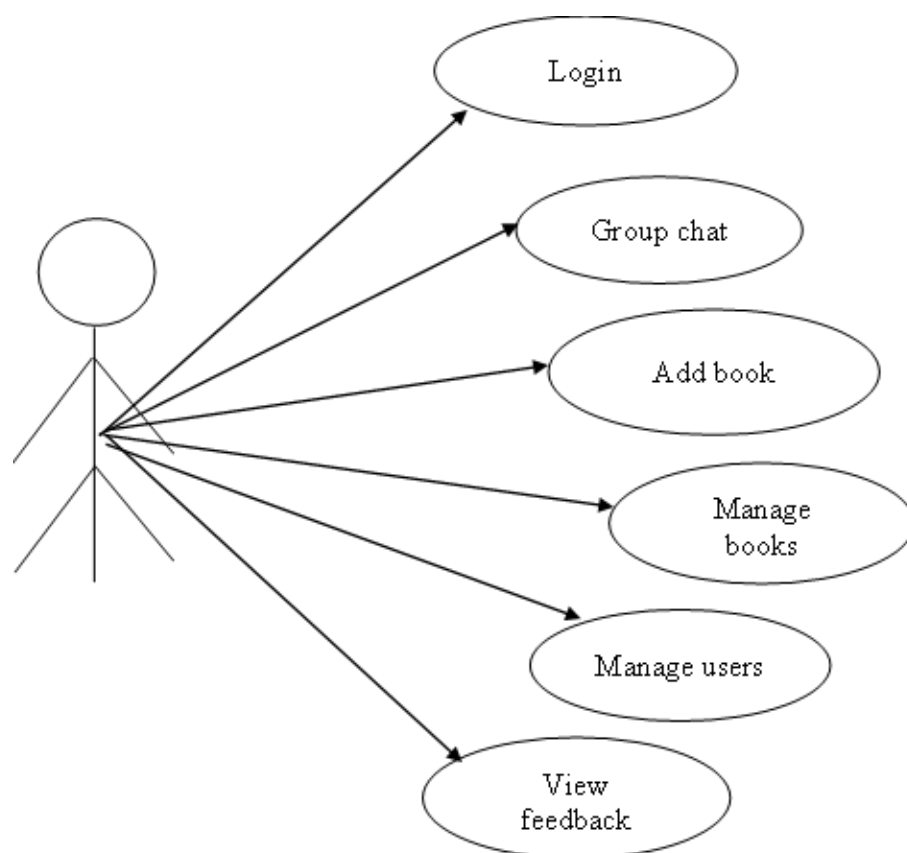


Figure 4: Use Case Diagram Of Librarian

II. Non-Functional Requirement

Non-functional requirements are requirements that are not directly concerned with specific functions of the system. It represents the quality attribute of the system. With the requirement, it judges the Responsiveness, Usability, Security, Portability and other non-functional standards of the software. They may define constraints on the system such as the capabilities of input/output device and data representation used in interface systems. These are the requirements that specify a function that a system or system component must be able to perform. It also describes the system qualities. It ensures the usability, effectiveness, efficiency and robustness of the system. If a system is failing to meet any one of them, as a result, it fails to satisfy internal business, user and market need. Non-functional requirements describe system behavior under specific conditions.

a) Efficiency:

When this system will be implemented, librarians and users will easily access the library as searching and book transaction will be very faster. Users can search books according to title.

b) Reliability:

The system should accurately perform member registration, member validation, book search and download.

c) Usability:

The system aims to be designed for a user-friendly environment so that users, admin and librarians of the library system can perform their own tasks easily and in an effective way.

d) Performance: The performance of the system is easy. To use the system, it needs basic technical knowledge. It is more user-friendly.

- e) Extensibility: The develop system is easy to extend. The code is built for the application to allow testing of different functions. The new algorithm can be easy add and remove from the code of system.
- f) Look and feel: The attractive color is used and the application is desktop.
- g) Maintainability: The system can be easy to maintain if any change required. System can be maintained according to user needs.

3.1.2 Feasibility Study

Feasibility analysis is process of investigating whether the system is worth to build or not. feasibility study Is done Before starting to build the system. cost and effort to build the system are also predicted here. If the system become not valuable and possible to accomplish the developers can stops their project. It makes easy for further planning and scheduling. it is a measure of the software product in terms of how much beneficial product development will be for the organization. Feasibility study is done for ensuring that the software product will be right in terms of development, implantation, contribution of project to the organization.

a) Technical

It considers the technical requirements of the proposed project. It measures that system Is technically capable or not. The technical requirements are then compared to the technical capability of the organization. This developed system is technically feasible. The software and hardware requirements for the development of this application are not many more and already available as free as open source. All technical aspects that are to be used in this system are easily available. The text editor as VS code, database as db.SQLite3 and python high level programming language are to be used to complete this system.

b) Operational

It is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Operational feasibility reviews the willingness of the organization to

support the proposed system. Users do not need to have any knowledge of the logics used in the system. With a simple interface, users can searching book by title and download the book, admin and librarian have authority to add book. Admin has some extra authority on system that user feedback and manage user. This system performs normally and on so simple way. It performs according to user needs and expectations. it does operations very well and satisfied way.

c) Economic

Economic feasibility analysis could also be referred as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a stakeholder. It measures whether the financial limitations are available or not for designing developed systems. The application is economically feasible because this was made using a VS code editor, which is open source and free available. A little money is 9 invested in printing the document. And take little amount for internet connections This project also uses SQLite as a local server to access data from a server with internet connection. The developed system is simple and dynamic. This system will use a minimal specification server, hardware during its development using the system will provide more profit for colleges.

d) Schedule

Table 1: Gantt Chart

Activates	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9
Literature review and proposal writing									
System Analysis									
System Design									
Implementation and Testing									
Documentation									

3.1.3 Analysis

Systems analysis is the process by which an individual (s) studies a system such that an information system can be analyzed, modeled, and a logical alternative can be chosen.

3.1.3.1 Class Diagram

The class diagram is a static diagram which represents the static view of an application. Diagram is not only for visualizing, describing and documenting different aspects of system but also for constructing executable code of software application.

Purpose of class diagram

- i. Analysis and design static view of application.
- ii. Describe responsibilities of a system.
- iii. Base for component and development diagrams.
- iv. Forward and reverse engineering.

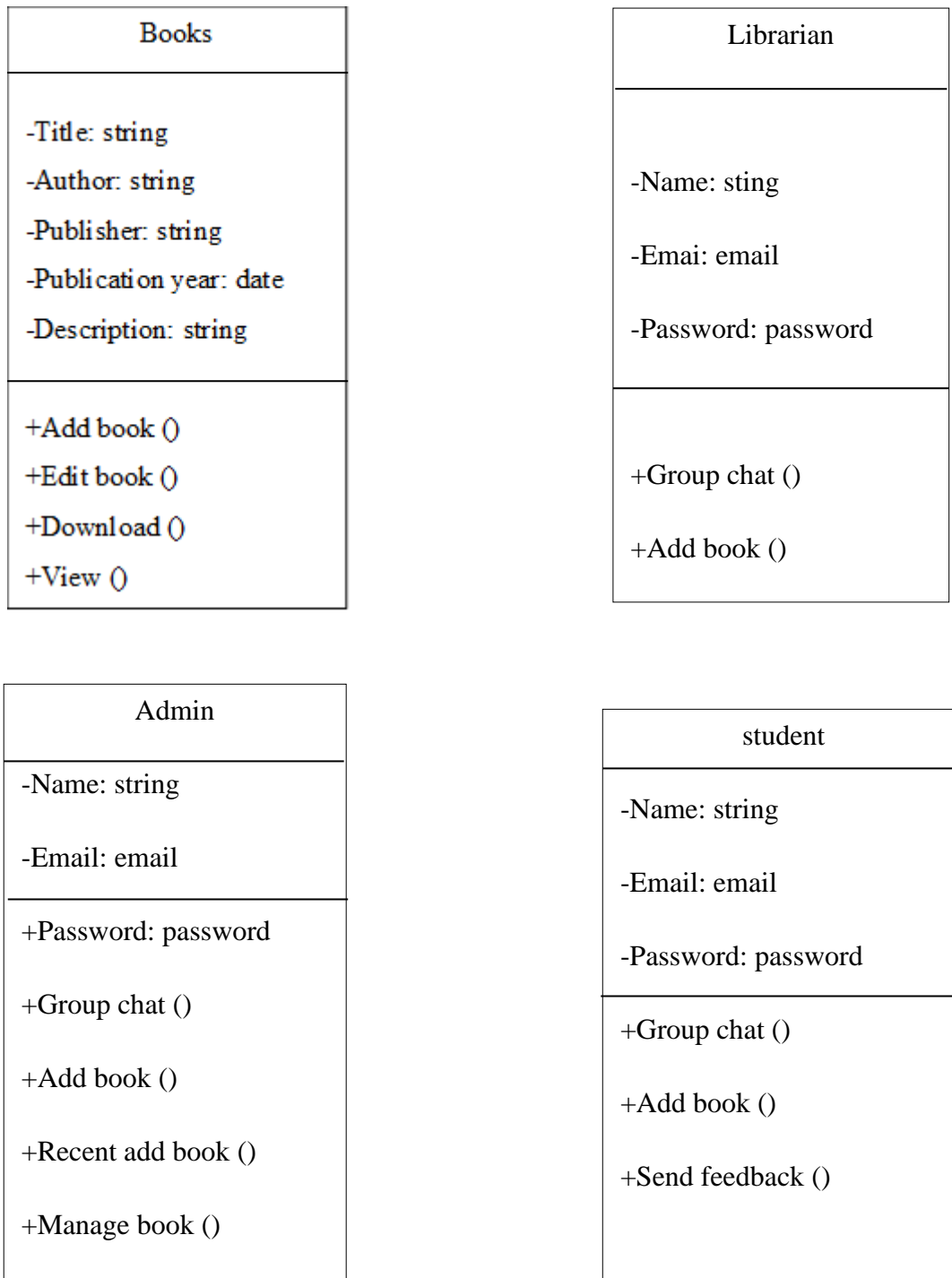


Figure 5: Class Diagram

3.1.3.2 Activities diagram

It is dynamic view of the system the show the interactions between the system objects. Dynamic aspect can be described as the changing and moving parts of the system.

It is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

Purpose of activity diagram

- i. Draw the activity flow of a system.
- ii. Draw the sequence from one activity to another.

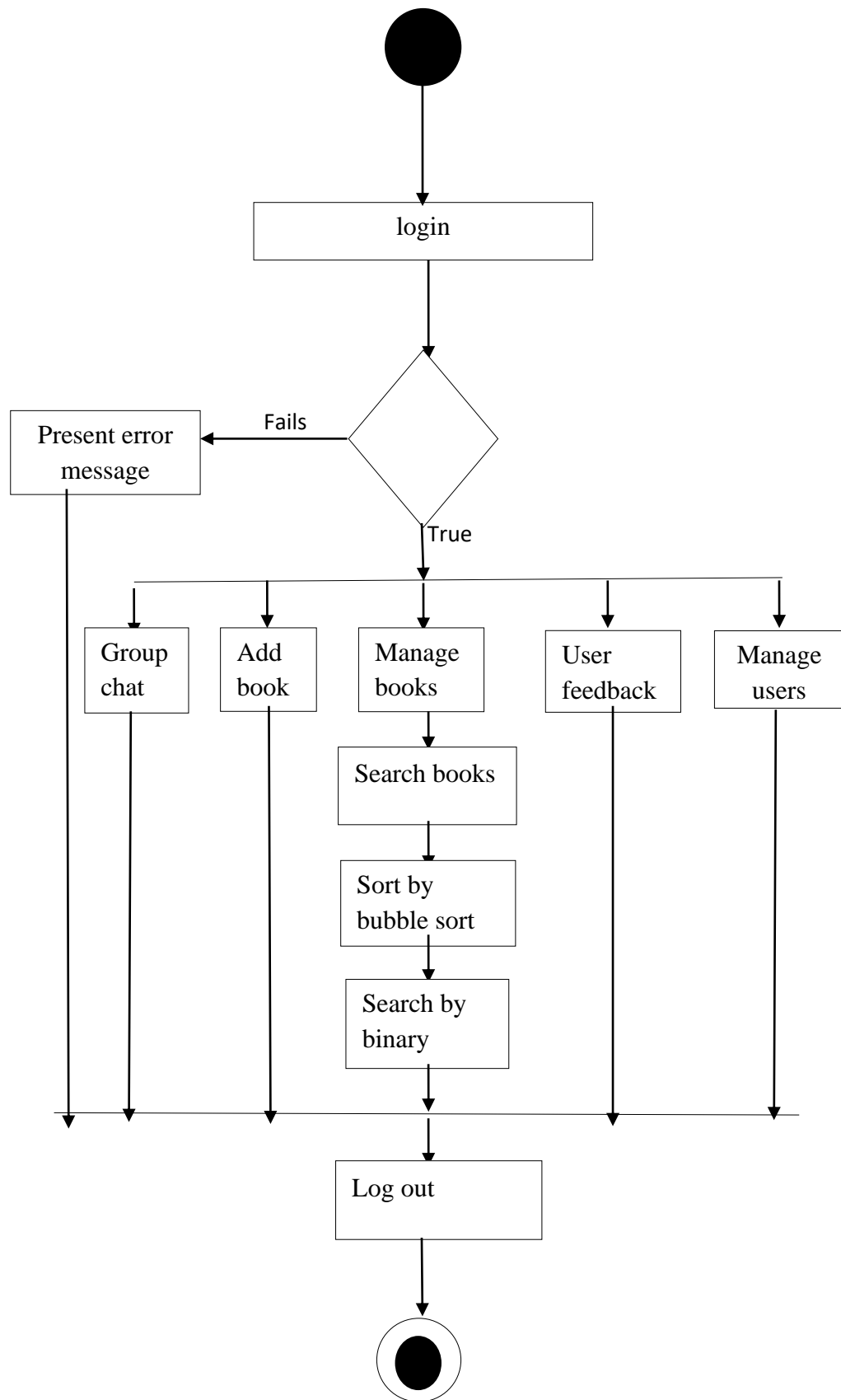


Figure 6: Activities Diagram Of Admin

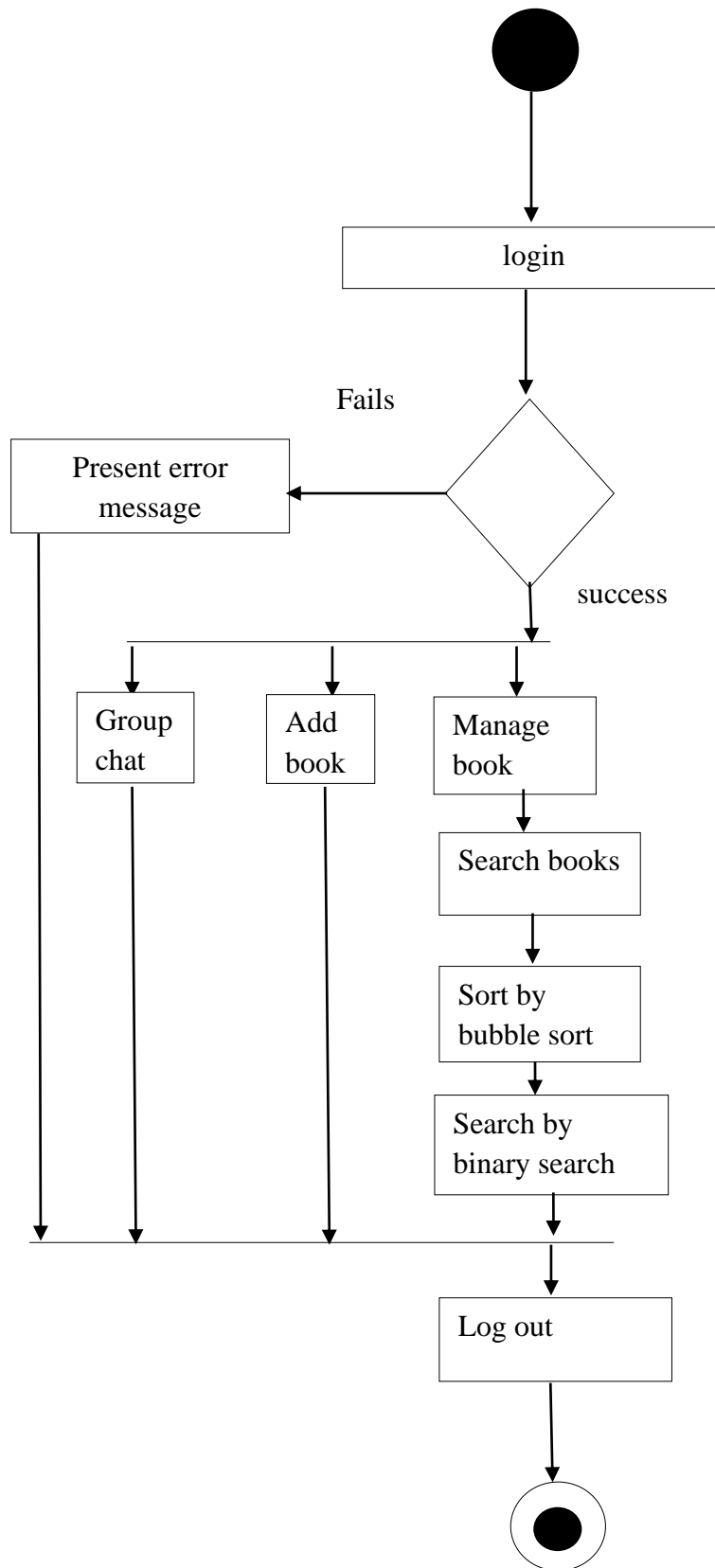


Figure 7: Activities Diagram Of Librarian

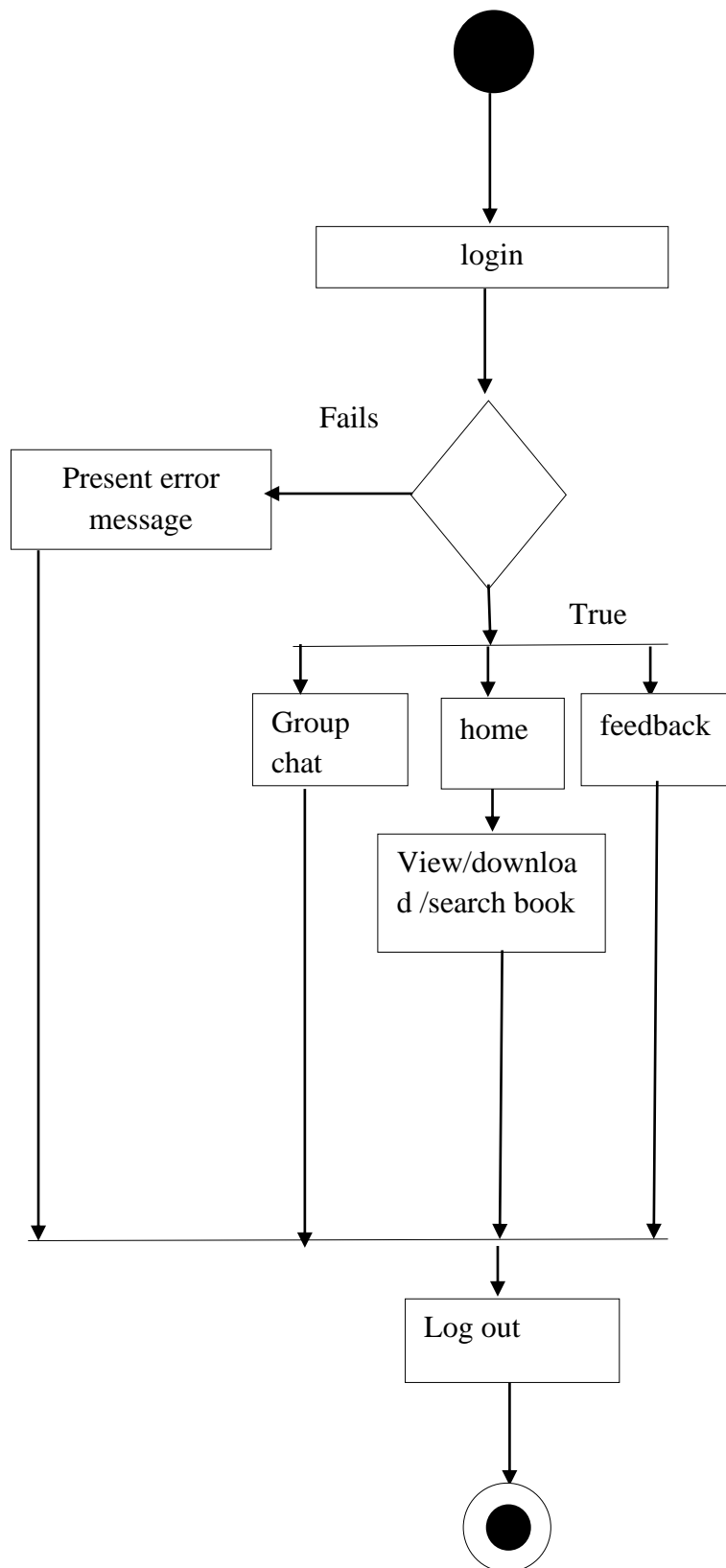


Figure 8: Activities Diagram Of Student

Chapter 4: System Design

4.1 Design

4.1.1 Deployment Diagrams

UML offers a way to visualize a system's architectural blueprints in a diagram. A deployment diagram in the unified modeling language models the physical deployment of artifact on nodes. It is used to represent visualize topology or organization of your physical component of your system upon which your system software is deployed for execution. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and database), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g.db.sqlite3, MVT).

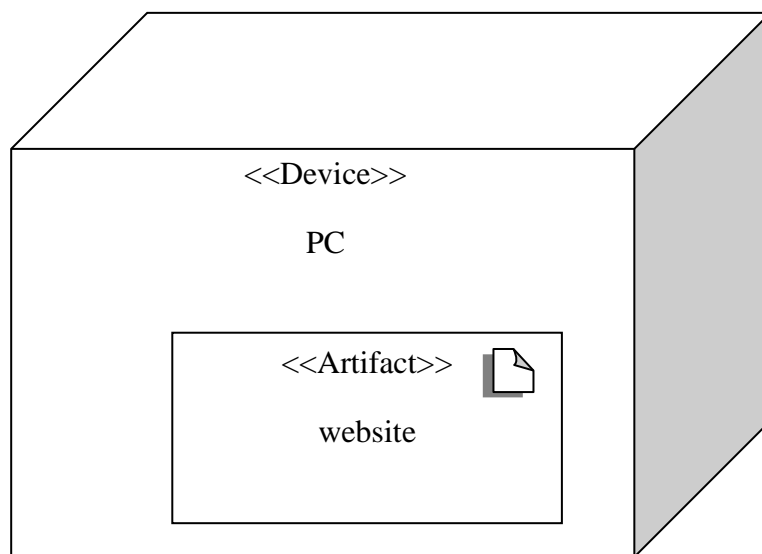


Figure 9: Deployment Diagram

4.2 Algorithm

In this system we use two algorithm one is insertion sort for sort the books title and another is binary search for search the book by title.

Insertion sort

Insertion sort iterates, consuming one input element each repetition, and grows a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

Algorithm

To sort an array of size n in ascending order:

- i. Compare the element with its adjacent element.
- ii. If at every comparison, we could find a position in sorted array where the element can be inserted, then create space by shifting the elements to right and insert the element at the appropriate position.
- iii. Repeat the above steps until you place the last element of unsorted array to its correct position.

Time complexity: $O(n^2)$

Space complexity: $O(1)$

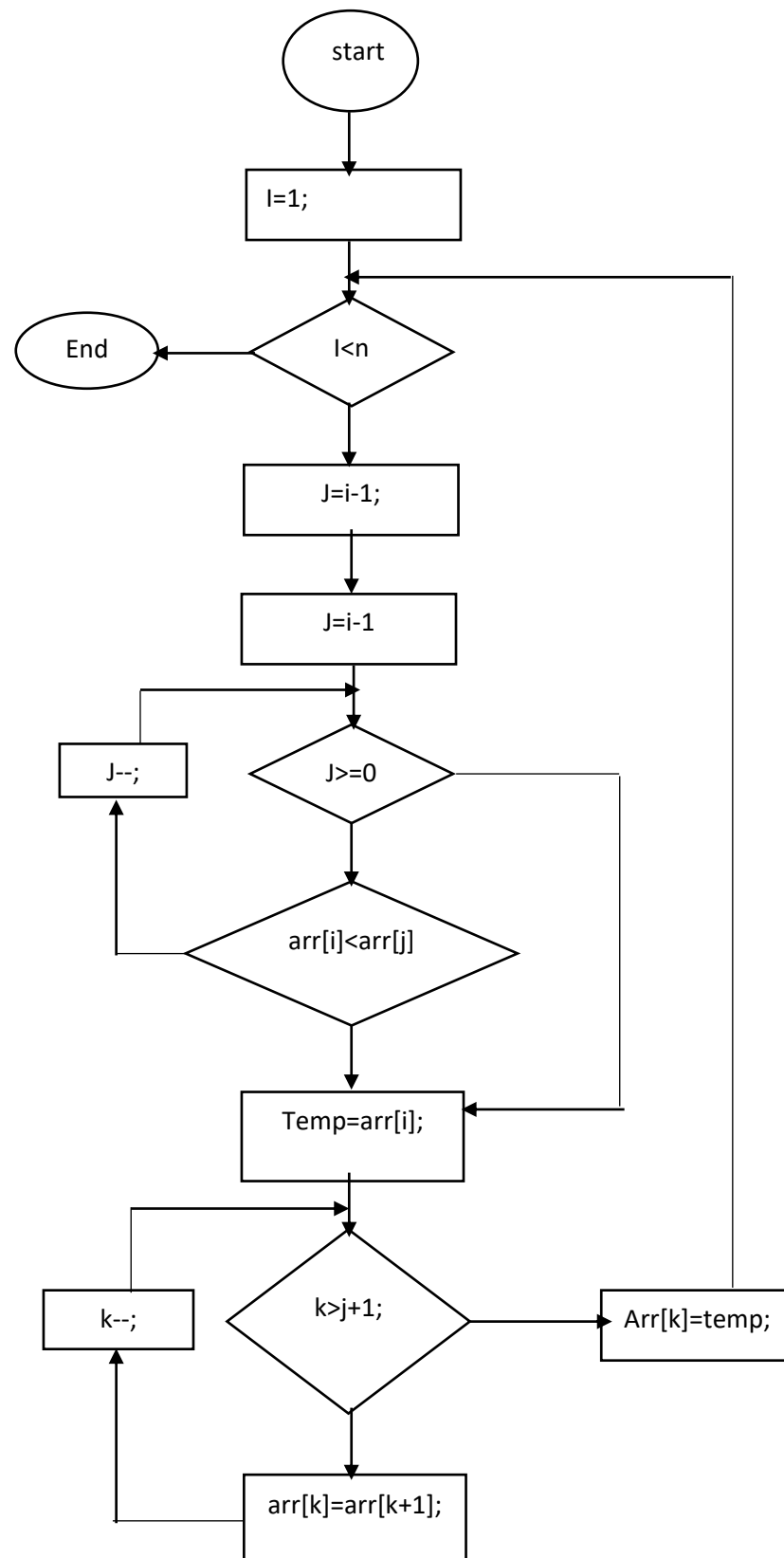


Figure 10: Flow Chart Of Insertion Sort

Binary search

Binary search works on sorted arrays. Binary search begins by comparing an element in the middle of the array with the target value. If the target value matches the element, its position in the array is returned. If the target value is less than the element, the search continues in the lower half of the array. If the target value is greater than the element, the search continues in the upper half of the array. By doing this, the algorithm eliminates the half in which the target value cannot lie in each iteration.

Algorithm

- i. Set two pointers low and high at the lowest and the highest position respectively.
- ii. Find the middle element mid of the array ie. $\text{arr}[(\text{low}+\text{high})/2]$.
- iii. If $x == \text{mid}$, then return mid else, compare the element to be searched with m.
- iv. If $x > \text{mid}$, compare x with the middle element of the elements on the right side of mid. This is done by setting low to $\text{low} = \text{mid} + 1$.
- v. Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high to $\text{high} = \text{mid} - 1$.
- vi. Repeat steps ii to v until low meets high.

Time complexity: $O(\log n)$

Space complexity: $O(1)$

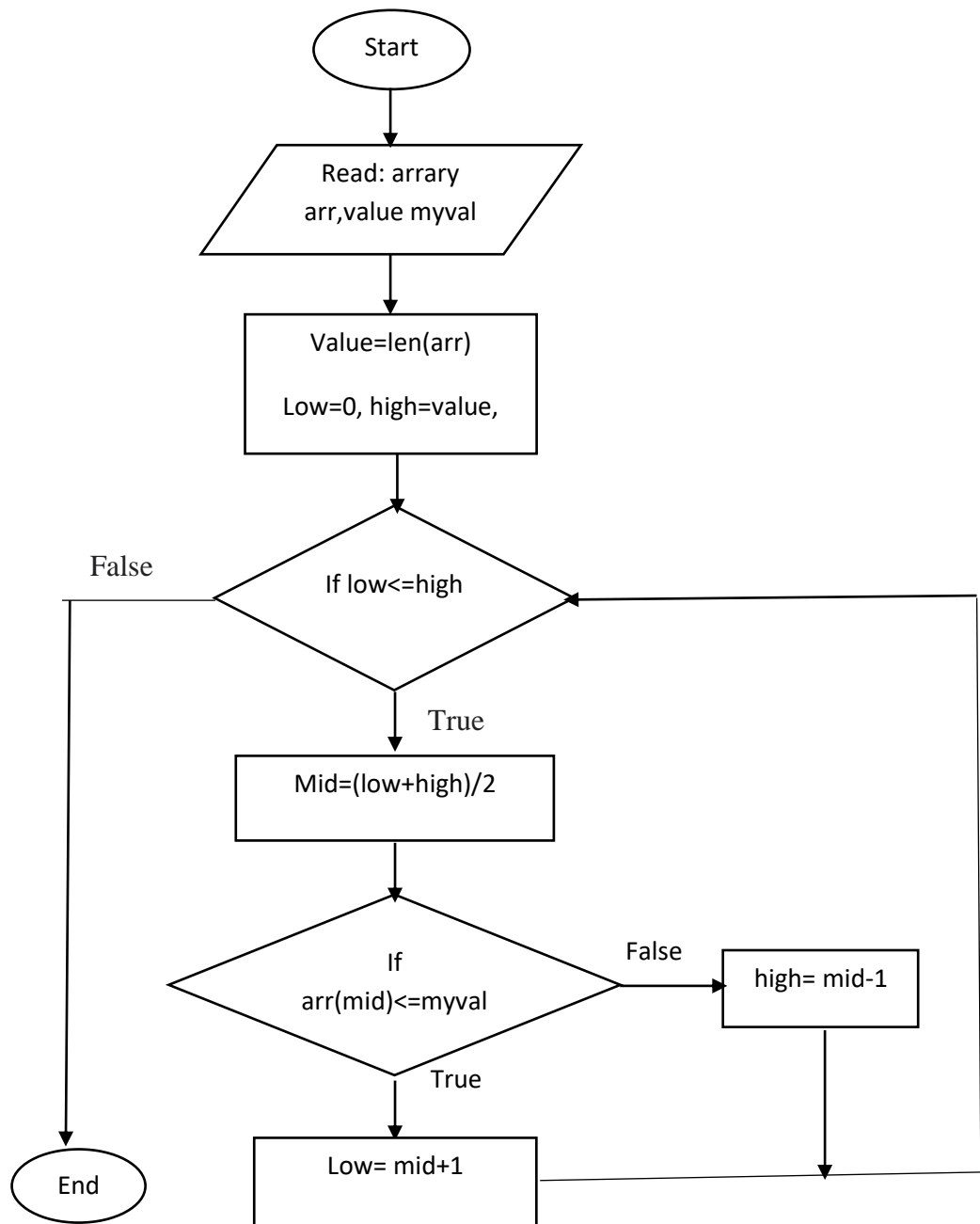


Figure 11: Flow Chart Of Binary Search

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

Table 2: Tools And Technology

Software tools	Hardware tools
Text editor: VS code Programming language: python Frame work: Django Database: db.sqlite3 Front end: HTML, CSS, JAVA Script	Processor (CPU): Intel Core i3 Memory: 4 GB RAM Storage: 1 TB internal storage device Monitor/Display: 14 LED monitors

5.1.2 Implementation Details of Modules

A. Modules

There are five modules in this project such as:

i. Users

This module helps us to create users. In this system there are three types of users: student, librarian, admin.

ii. Books

This module is used to add books in the system. Only admin and librarian have authority to add books. When books are added, need to provide book titles, author name, publication years, publisher name, description, upload by, user id, pdf files and cover page of book.

iii. Delete

This module is used to delete books. Only admin and librarian have authority to delete books.

iv. Chat

This module is used to send and receive messages.

v. Feedback

This module is used to send feedback from student to admin.

B. Algorithm

An algorithm is a set of instructions for solving a problem or accomplishing a task. Algorithm have become important in developing automated and high-frequency-trading (HFT) system as well as in the pricing of sophisticated financial instrument like derivatives. In this system we used insertion sort to sort the title of book and binary search to search the book.

i. Insertion sort algorithm:

Insertion sort is simple sorting algorithm. It sorted the given array in the ascending order.

Insertion sort iterates, consuming one input element each repetition, and grows a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

```
def insertion_sort(arr):
```

```
    #loop over all the elements in the list
```

```
    for i in range(1, len(arr)):
```

```
        val = arr[i]
```

```
        # move elements of list [0..i-1], that are
```

```
        # greater than val, to one position ahead
```

```
        # of the current position
```

```
        j = i-1
```

```
        while j >=0 and val < arr[j] :
```

```
            arr[j+1] = arr[j]
```

```
            j -= 1
```

```
        arr[j+1] = val
```

```
    return arr
```

ii. Binary search algorithm

Binary search works on sorted arrays. Binary search begins by comparing an element in the middle of the array with the target value. If the target value matches the element, its position in the array is returned. If the target value is less than the element, the search continues in the lower half of the array. If the target value is greater than the element, the search continues in the upper half of the array. By doing this, the algorithm eliminates the half in which the target value cannot lie in each iteration.

```
def binary_search(arr, low, high, x):
    # Check base case
    if high >= low:
        mid = (high + low) // 2
        # If element is present at the middle itself
        if arr[mid] == x:
            return mid
        # If element is smaller than mid, then it can only
        # be present in left subarray
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)
        # Else the element can only be present in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)
    else:
        # Element is not present in the array
        return -1
```

5.2 Testing

Testing is a set of activities that can be planned in advance and conducted systematically. Testing assesses the quality of the product. Software testing is a process that should be done during the development process.

5.2.1 Test Cases for Unit Testing

Unit testing is the process of taking a module and running it in isolation from the rest of the software product by using prepared test cases and comparing actual results with the result predicted by the specifications and design of the Module.

Table 3: Unit Testing

Test case ID	Test scenario	Test step	Input test data	Excepted result	Actual result	Status
TC-1	User interface	How look like interface	-----	Look like more attractive	Look like more attractive	Success
TC-2	Student registration and login	Registration and login	Username, email and password	Registration success and login success	Registration success and login success	Success
TC-3	Login admin interface	Login	Admin username and password	Login success	Login success	Success
TC-4	Login librarian interface	Register by admin and login	Librarian username and password	Registration success and login success	Registration success and login success	Success

5.2.2 Test Cases for System Testing

System testing is the phase in software testing in which individual software modules are combined and tested as a group. Here entire system has been tested as per requirements and has been applied after unit testing only.

Table 4: System Testing

Test case ID	Test scenario	Test step	Input test data	Excepted result	Actual result	Status
TC-1	Manage books	Add, edit and delete books	Add new book, edit Book information and delete	Add, edit and delete book success	Add, edit and delete books success	Success
TC-2	Manage users	Add, edit and delete users	Add new book, edit user information and delete	Add, edit and delete user success	Add, edit and delete books success	Success

5.3 Result Analysis

The implementation and testing of online library management system is satisfying the required design specification. Table 3 and table 4 show the successful testing of the system as unit testing and system testing.

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion

This online library management system is for computerizing the working in a library. There is storage of books then users can easily download and read required books online. It provides an easy and effective way. The main aim of online library management system is to make it easy for user who wants to download books with easy and simple process and interface. This system gives access only registered and validate users. The System has been developed using number of technologies in which Python has been used as main programming language. The system is developed by considering all issues related to all users which are included in this system. This system mainly focused on users to provides easy way for find out and download the required books. Various issues related to old way of Service have been solved by providing them a full-fledged system.

Based on the result of this research, it can be concluded: It helps users in searching, ,reading and download required books easily; most of organizations carrying manual way of library management , these can easily adopt this system for organization. It will make library related work more effective and quickly. The online library management system mainly developed for school, can help school in storing book online and easy access for student, also easy for management as well.

6.2 Future Recommendation

Currently use of online library management system is limited. In the upcoming future, the organizational library management will be digitized using an online library management system. In the future, the developed web applications can be developed as mobile app as well.

References


- [1] [Online]. Available: https://en.wikipedia.org/wiki/Library_management.
- [2] PRASANNA, NAMBURI SAI NAGA LAKSHMI ; GUPTA, B.N SRINIVASA, "ONLINE LIBRARY MANAGEMENT SYSTEM," *International Journal of Emerging Technologies and Innovative Research*, vol. Vol.7, no. Issue 5, pp. page no.117-122, May-2020.
- [3] "Development of an Online Integrated Library Management Information," *International Journal of Scientific Research in Computer Science and Engineering*, vol. Vol.8, no. Issue.2, pp. 65-76, 2020.
- [4] Roy, Abir ; Mridha, Anindita; Paul, Dibyajyoti ; Dutta , Jewel; Mondal , Subhojyoti ; Giri, Susmita ;, "E-Library Management System," 2018.

Appendices


Screenshots

Library Management


Register Here



username



email



password


Login

Register


Figure 12: Student Registration

Library Management

Login In Here



username



password


Sign In


[Register](#)

Figure 13: Login Interface

BookApp

Home Group Chat Send Feedback About Us


Search for... 



Books list

Show 10 entries

Search:

Book ID	Book Cover	Book Title	Read	Download
10		Python Book 2	View PDF	Download PDF

Showing 1 to 1 of 1 entries

[Previous](#) [1](#) [Next](#)

Figure 14: Student Interface

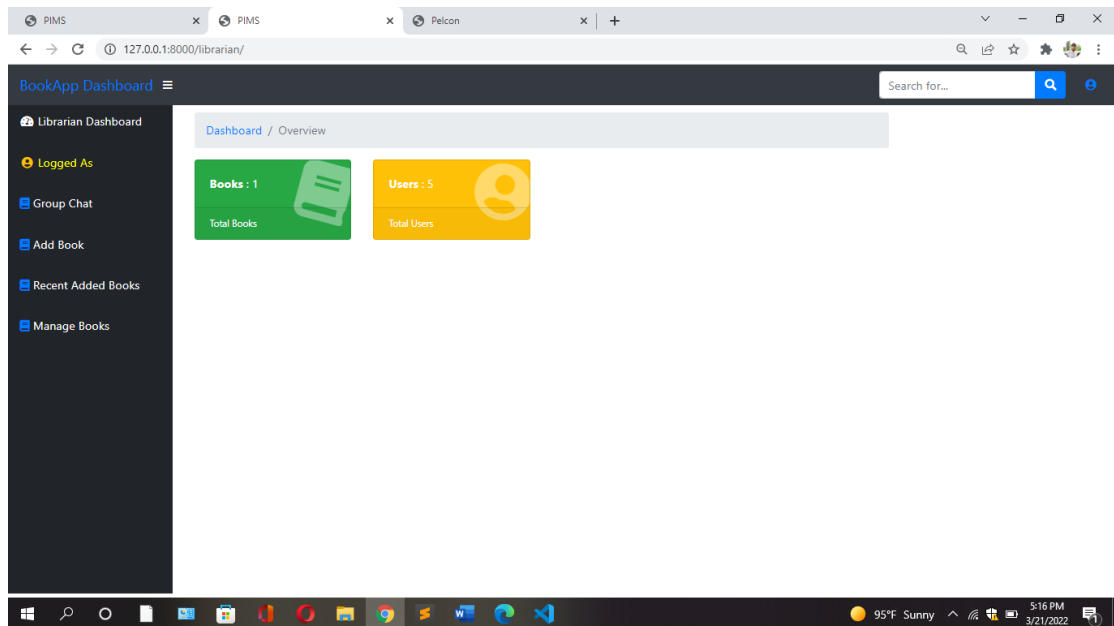


Figure 15: Librarian Interface

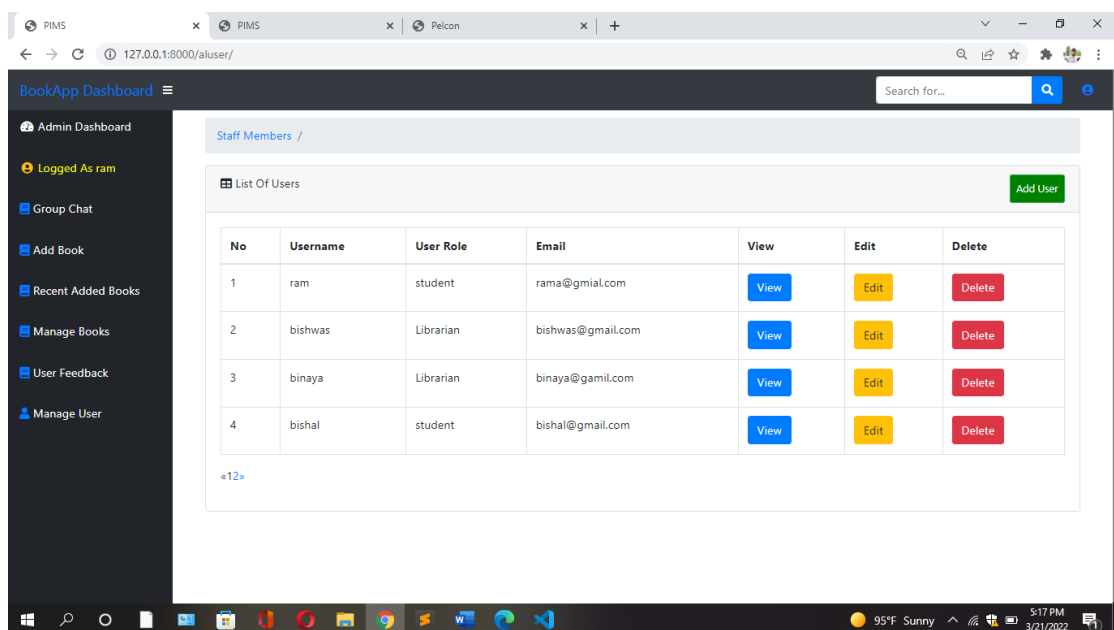


Figure 16: Admin Interface

Coding

View.py

```
from asyncio.windows_events import NULL
from django.shortcuts import redirect, render
from django.contrib.messages.views import SuccessMessageMixin
from django.urls import reverse_lazy
from django.views import generic
from bootstrap_modal_forms.mixins import PassRequestMixin
from .models import User, Book, Chat, Feedback
from django.contrib import messages
from django.db.models import Sum
from django.views.generic import CreateView, DetailView, DeleteView,
UpdateView, ListView
from .forms import ChatForm, BookForm, UserForm
from . import models
import operator
import itertools
from django.core.paginator import Paginator, EmptyPage,
PageNotAnInteger
from django.contrib.auth import authenticate, logout
from django.contrib import auth, messages
from django.contrib.auth.hashers import make_password
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.decorators import login_required
from django.utils import timezone

# Shared Views
def login_form(request):
    return render(request, 'bookstore/login.html')

def logoutView(request):
    logout(request)
    return redirect('home')

def loginView(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        if username=="":
            error="enter username"

            return
    render(request, 'bookstore/login.html', {'error': error})
    # elif len(password)>=5:
    #     error="enter password more than five charater"
```

```

        # return
render(request, 'bookstore/login.html', {'error': error})
        user = authenticate(request, username=username,
password=password)
        if user is not None and user.is_active:
            auth.login(request, user)
            if user.is_admin or user.is_superuser:
                return redirect('dashboard')
            elif user.is_librarian:
                return redirect('librarian')
            else:
                return redirect('student')
        else:
            error="Invalid username or password"
            return
render(request, 'bookstore/login.html', {'error': error})
        # messages.info(request, "Invalid username or password")
        # return redirect('home')

def register_form(request):
    return render(request, 'bookstore/register.html')

def registerView(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        password = make_password(password)
        if username=="":
            error="enter username"

            return
render(request, 'bookstore/register.html', {'error': error})
        elif len(password)>=5:
            error="please enter password more than five character"
            return
render(request, 'bookstore/register.html', {'error': error})

        a = User(username=username, email=email, password=password,
is_student=True)
        a.save()
        error="Account was created successfully"
        return render(request, 'bookstore/login.html', {'error': error})
        # messages.success(request, 'Account was created successfully')
        # return redirect('home')
    else:
        error="please enter more than five character"

```

```

        return
    render(request, 'bookstore/register.html', {'error': error})
    # messages.error(request, 'Registration fail, try again later')
    # return redirect('regform')

# student views
@login_required
def student(request):
    return render(request, 'student/home.html')

@login_required
def uabook_form(request):
    return render(request, 'student/add_book.html')

@login_required
def feedback_form(request):
    return render(request, 'student/send_feedback.html')

@login_required
def about(request):
    return render(request, 'student/about.html')

# This function takes unsorted array as an input
# and returns sorted array.
def insertion_sort(arr):

    # loop over all the elements in the list
    for i in range(1, len(arr)):

        val = arr[i]

        # move elements of list [0..i-1], that are
        # greater than val, to one position ahead
        # of the current position
        j = i-1
        while j >= 0 and val < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = val

    return arr

```

```

# Python 3 program for recursive binary search.
# Modifications needed for the older Python 2 are found in comments.

# Returns index of x in arr if present, else -1
def binary_search(arr, low, high, x):

    # Check base case
    if high >= low:

        mid = (high + low) // 2

        # If element is present at the middle itself
        if arr[mid] == x:
            return mid

        # If element is smaller than mid, then it can only
        # be present in left subarray
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)

        # Else the element can only be present in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)

    else:
        # Element is not present in the array
        return -1

@login_required
def usearch(request):
    query = request.GET['query']
    Context = {}
    searchElem = Book.objects.all().filter(title = query)
    if(len(searchElem)==0):
        Context['message'] = "NO Such Element in Database"
        return render(request, "student/result.html", Context)

    else:
        for item in searchElem:
            searchId = item.id
            value = Book.objects.all()
            myVal = []
            for item in value:
                myVal.append(item.id)
            # print(myVal)
            myVal = insertion_sort(myVal)

```

```

        myIndex = binary_search(myVal,0,len(myVal),searchId)
        # print(myIndex)
        mytitle = myVal[myIndex]
        searchedValue = Book.objects.all().filter(id = mytitle)
        Context['files'] = searchedValue
        return render(request,'student/result.html',Context)

@login_required
def send_feedback(request):
    if request.method == 'POST':
        feedback = request.POST['feedback']
        current_user = request.user
        user_id = current_user.id
        username = current_user.username
        feedback = username + " " + " says " + feedback

        a = Feedback(feedback=feedback)
        a.save()
        messages.success(request, 'Feedback was sent')
        return redirect('feedback_form')
    else:
        messages.error(request, 'Feedback was not sent')
        return redirect('feedback_form')

class UBookListView(LoginRequiredMixin,ListView):
    model = Book
    template_name = 'student/book_list.html'
    context_object_name = 'books'
    paginate_by = 2

    def get_queryset(self):
        return Book.objects.order_by('-id')

class UCreateChat(LoginRequiredMixin, CreateView):
    form_class = ChatForm
    model = Chat
    template_name = 'student/chat_form.html'
    success_url = reverse_lazy('ulchat')

    def form_valid(self, form):
        self.object = form.save(commit=False)
        self.object.user = self.request.user
        self.object.save()
        return super().form_valid(form)

```

```

class UListChat(LoginRequiredMixin, ListView):
    model = Chat
    template_name = 'student/chat_list.html'

    def get_queryset(self):
        return
Chat.objects.filter(posted_at__lt=timezone.now()).order_by('posted_at')

# Librarian views
def librarian(request):
    book = Book.objects.all().count()
    user = User.objects.all().count()

    context = {'book':book, 'user':user}

    return render(request, 'librarian/home.html', context)

@login_required
def labook_form(request):
    return render(request, 'librarian/add_book.html')

@login_required
def labook(request):
    if request.method == 'POST':
        title = request.POST['title']
        author = request.POST['author']
        year = request.POST['year']
        publisher = request.POST['publisher']
        desc = request.POST['desc']
        cover = request.FILES['cover']
        pdf = request.FILES['pdf']
        current_user = request.user
        user_id = current_user.id
        username = current_user.username

```



```

        a = Book(title=title, author=author, year=year,
publisher=publisher,
                desc=desc, cover=cover, pdf=pdf, uploaded_by=username,
user_id=user_id)
        a.save()
        messages.success(request, 'Book was uploaded successfully')
        return redirect('llbook')
    else:
        messages.error(request, 'Book was not uploaded successfully')
        return redirect('llbook')

class LBookListView(LoginRequiredMixin,ListView):
    model = Book
    template_name = 'librarian/book_list.html'
    context_object_name = 'books'
    paginate_by = 3

    def get_queryset(self):
        return Book.objects.order_by('-id')

class LManageBook(LoginRequiredMixin,ListView):
    model = Book
    template_name = 'librarian/manage_books.html'
    context_object_name = 'books'
    paginate_by = 3

    def get_queryset(self):
        return Book.objects.order_by('-id')

class LViewBook(LoginRequiredMixin,DetailView):
    model = Book
    template_name = 'librarian/book_detail.html'

class LEditView(LoginRequiredMixin,UpdateView):
    model = Book
    form_class = BookForm
    template_name = 'librarian/edit_book.html'
    success_url = reverse_lazy('lmbook')
    success_message = 'Data was updated successfully'

class LDeleteView(LoginRequiredMixin,DeleteView):
    model = Book
    template_name = 'librarian/confirm_delete.html'

```

```

        success_url = reverse_lazy('lmbook')
        success_message = 'Data was deleted successfully'

class LDeleteBook(LoginRequiredMixin, DeleteView):
    model = Book
    template_name = 'librarian/confirm_delete2.html'
    success_url = reverse_lazy('librarian')
    success_message = 'Data was dele successfully'

@login_required
def lsearch(request):
    query = request.GET['query']
    Context = {}
    searchElem = Book.objects.all().filter(title = query)
    if(len(searchElem)==0):
        Context['message'] = "NO Such Element in Database"
        return render(request, "librarian/result.html", Context)

    else:
        for item in searchElem:
            searchId = item.id
            value = Book.objects.all()
            myVal = []
            for item in value:
                myVal.append(item.id)
            # print(myVal)
            myVal = insertion_sort(myVal)
            myIndex = binary_search(myVal, 0, len(myVal), searchId)
            # print(myIndex)
            mytitle = myVal[myIndex]
            searchedValue = Book.objects.all().filter(id = mytitle)
            Context['files'] = searchedValue
            return render(request, 'librarian/result.html', Context)

class LCreateChat(LoginRequiredMixin, CreateView):
    form_class = ChatForm
    model = Chat
    template_name = 'librarian/chat_form.html'
    success_url = reverse_lazy('llchat')

    def form_valid(self, form):
        self.object = form.save(commit=False)
        self.object.user = self.request.user

```

```

        self.object.save()
        return super().form_valid(form)

class LListChat(LoginRequiredMixin, ListView):
    model = Chat
    template_name = 'librarian/chat_list.html'

    def get_queryset(self):
        return
Chat.objects.filter(posted_at__lt=timedelta.now()).order_by('posted_at')

# Admin views

def dashboard(request):
    book = Book.objects.all().count()
    user = User.objects.all().count()

    context = {'book':book, 'user':user}

    return render(request, 'dashboard/home.html', context)

def create_user_form(request):
    choice = ['1', '0', 'student', 'Admin', 'Librarian']
    choice = {'choice': choice}

    return render(request, 'dashboard/add_user.html', choice)

class ADeleteUser(SuccessMessageMixin, DeleteView):
    model = User
    template_name='dashboard/confirm_delete3.html'
    success_url = reverse_lazy('aluser')
    success_message = "Data successfully deleted"

class AEditUser(SuccessMessageMixin, UpdateView):
    model = User
    form_class = UserForm
    template_name = 'dashboard/edit_user.html'
    success_url = reverse_lazy('aluser')
    success_message = "Data successfully updated"

class ListUserView(generic.ListView):

```

```

model = User
template_name = 'dashboard/list_users.html'
context_object_name = 'users'
paginate_by = 4

def get_queryset(self):
    return User.objects.order_by('-id')

def create_user(request):
    choice = ['1', '0', 'student', 'Admin', 'Librarian']
    choice = {'choice': choice}
    if request.method == 'POST':
        first_name=request.POST['first_name']
        last_name=request.POST['last_name']
        username=request.POST['username']
        userType=request.POST['userType']
        email=request.POST['email']
        password=request.POST['password']
        password = make_password(password)
        print("User Type")
        print(userType)
        if userType == "student":
            a = User(first_name=first_name, last_name=last_name,
username=username, email=email, password=password,is_publisher=True)
            a.save()
            print(a.is_student)
            messages.success(request, 'Member was created
successfully!')
            return redirect('aluser')
        elif userType == "Admin":
            a = User(first_name=first_name, last_name=last_name,
username=username, email=email, password=password, is_admin=True)
            a.save()
            messages.success(request, 'Member was created
successfully!')
            return redirect('aluser')
        elif userType == "Librarian":
            a = User(first_name=first_name, last_name=last_name,
username=username, email=email, password=password, is_librarian=True)
            a.save()
            messages.success(request, 'Member was created
successfully!')
            return redirect('aluser')
        else:
            messages.success(request, 'Member was not created')
            return redirect('create_user_form')
    else:
        return redirect('create_user_form')

```

```

class ALViewUser(DetailView):
    model = User
    template_name='dashboard/user_detail.html'

class ACreateChat(LoginRequiredMixin, CreateView):
    form_class = ChatForm
    model = Chat
    template_name = 'dashboard/chat_form.html'
    success_url = reverse_lazy('alchat')

    def form_valid(self, form):
        self.object = form.save(commit=False)
        self.object.user = self.request.user
        self.object.save()
        return super().form_valid(form)

class AListChat(LoginRequiredMixin, ListView):
    model = Chat
    template_name = 'dashboard/chat_list.html'

    def get_queryset(self):
        return
Chat.objects.filter(posted_at__lt=timezone.now()).order_by('posted_at')

@login_required
def aabook_form(request):
    return render(request, 'dashboard/add_book.html')

@login_required
def aabook(request):
    if request.method == 'POST':
        title = request.POST['title']
        author = request.POST['author']
        year = request.POST['year']
        publisher = request.POST['publisher']
        desc = request.POST['desc']
        cover = request.FILES['cover']
        pdf = request.FILES['pdf']
        current_user = request.user

```

```

        user_id = current_user.id
        username = current_user.username

        a = Book(title=title, author=author, year=year,
publisher=publisher,
                desc=desc, cover=cover, pdf=pdf, uploaded_by=username,
user_id=user_id)
        a.save()
        messages.success(request, 'Book was uploaded successfully')
        return redirect('albook')
    else:
        messages.error(request, 'Book was not uploaded successfully')
        return redirect('aabook_form')

class ABookListView(LoginRequiredMixin,ListView):
    model = Book
    template_name = 'dashboard/book_list.html'
    context_object_name = 'books'
    paginate_by = 3

    def get_queryset(self):
        return Book.objects.order_by('-id')

class AManageBook(LoginRequiredMixin,ListView):
    model = Book
    template_name = 'dashboard/manage_books.html'
    context_object_name = 'books'
    paginate_by = 3

    def get_queryset(self):
        return Book.objects.order_by('-id')

class ADeleteBook(LoginRequiredMixin,DeleteView):
    model = Book
    template_name = 'dashboard/confirm_delete2.html'
    success_url = reverse_lazy('ambook')
    success_message = 'Data was dele successfully'

class ADeleteBookk(LoginRequiredMixin,DeleteView):
    model = Book
    template_name = 'dashboard/confirm_delete.html'

```

```

        success_url = reverse_lazy('dashboard')
        success_message = 'Data was dele successfully'

class AViewBook(LoginRequiredMixin,DetailView):
    model = Book
    template_name = 'dashboard/book_detail.html'

class AEditView(LoginRequiredMixin,UpdateView):
    model = Book
    form_class = BookForm
    # fields="__all__"
    template_name = 'dashboard/edit_book.html'
    success_url = reverse_lazy('ambook')
    success_message = 'Data was updated successfully'

class AFeedback(LoginRequiredMixin,ListView):
    model = Feedback
    template_name = 'dashboard/feedback.html'
    context_object_name = 'feedbacks'
    paginate_by = 3

    def get_queryset(self):
        return Feedback.objects.order_by('-id')

@login_required
def asearch(request):
    query = request.GET['query']
    # print(query)
    # print(type(query))

    # print(myVal)
    # print(myVal[2])

    Context = {}
    searchElem = Book.objects.all().filter(title = query)
    if(len(searchElem)==0):
        Context['message'] = "NO Such Element in Database"
        return render(request,"dashboard/result.html",Context)

    else:

```

```
for item in searchElem:
    searchId = item.id
value = Book.objects.all()
myVal = []
for item in value:
    myVal.append(item.id)
myVal = insertion_sort(myVal)
# print(myVal)
myIndex = binary_search(myVal,0,len(myVal),searchId)
# print(myIndex)
mytitle = myVal[myIndex]
searchedValue = Book.objects.all().filter(id = mytitle)
Context['files'] = searchedValue
return render(request, 'dashboard/result.html',Context)
```