

A PROJECT REPORT
ON
DIGITAL LIBRARY SYSTEM
USING FULL STACK WEB DEVELOPMENT

Project Report Submitted in the Partial Fulfilment of represents to

Jawaharlal Nehru Technological University, Kakinada

For the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

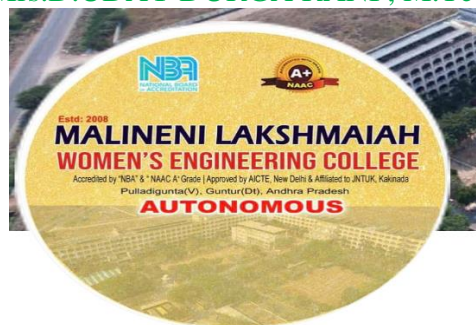
COMPUTER SCIENCE AND ENGINEERING

Submitted by

P.PRAVALLIKA	(20KE1A0584)
K.SRI LIKHITHA	(20KE1A0542)
R.SAI SNEHA	(20KE1A0591)
CH.AMRUTHA VANI	(20KE1A0511)
Y. VEERAMMA	(20KE1A05B1)

under the guidance of

Mrs.D.UDAY DURGA RANI , M.Tech



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALINENI LAKSHMAIAH WOMEN'S ENGINEERING COLLEGE

(Accredited by NBA(CSE&ECE),NAAC A+, Approved by AICTE, NEW DELHI, Affiliated to JNTU

Kakinada)Pulladigunta(V), Vatticherukuru(Md), Guntur(Dt),AP,-522017.

2020 – 2024

MALINENI LAKSHMAIAH WOMEN'S ENGINEERING COLLEGE

Pulladigunta(V), Vatticherukuru(Md), Guntur(Dt),AP,-522017.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled “ **DIGITAL LIBRARY SYSTEM USING FULL STACK WEB DEVELOPMENT** ” is the bonafide work carried out by

P.PRAVALLIKA	(20KE1A0584)
K.SRI LIKHITHA	(20KE1A0542)
R.SAI SNEHA	(20KE1A0591)
CH.AMRUTHA VANI	(20KE1A0511)
Y.VEERAMMA	(20KE1A05B1)

B.Tech , Affiliated to JNTU Kakinada in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY with the specialization in COMPUTER SCIENCE AND ENGINEERING during the Academic year 2020-2024.

Signature of the Guide

Mrs.D . Udaya Durga rani,M.Tech

Head of the Department

Dr.G.Rama swamy, M.Tech, Ph.D

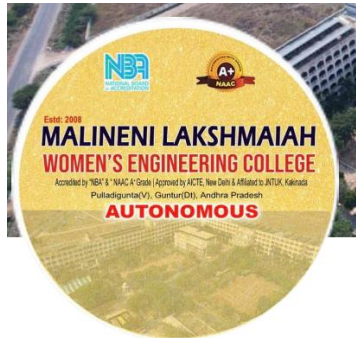
Viva – voice Held on

External Examiner

MALINENI LAKSHMAIAH WOMEN'S ENGINEERING COLLEGE

Pulladigunta(V), Vatticherukuru(Md), Guntur(Dt),AP,-522017.

DEPARTMENT OF COMPUTERS SCIENCE AND ENGINEERING



DECLARATION

We hereby declare that the project work entitled “**DIGITAL LIBRARY SYSTEM USING FULL STACK WEB DEVELOPMENT**” is entirely our original work carried out under the guidance of Mrs. D. Uday Durga rani, M. Tech, Assistant Professor in Malineni Lakshmaiah Women's Engineering College, Pulladigunta, Guntur, JNTU Kakinada, A.P, India for the award of the DEGREE OF BACHELOR OF TECHNOLOGY with the specialization in COMPUTER SCIENCE AND ENGINEERING. The results carried out in this project report have not been submitted in a part or full for the award of any degree or diploma of this or any other university or institute.

Submitted By

P.PRAVALLIKA	(20KE1A0584)
K.SRI LIKHITHA	(20KE1A0542)
R.SAI SNEHA	(20KE1A0591)
CH.AMRUTHA VANI	(20KE1A0511)
<u>Y.VEERAMA</u>	<u>(20KE1A05B1)</u>

ACKNOWLEDGEMENT

All endeavors over a long period can be successful only with the advice and support of many well-wishers. I take this opportunity to express my gratitude and appreciation to all of them.

We wish to express deep sense of gratitude to my beloved and respected guide **Mrs.D.Uday Durga Rani , M.Tech**, Assistant Professor in Malineni Lakshmaiah Women's Engineering College, Guntur, for her valuable guidance, suggestions and constant encouragement and keen interest enriched throughout the course of project work.

We extend sincere thanks to the **Principal**, Prof. **Dr.J.APPARAO** for his kind co-operation in completing and making this project a success.

We would like to thank the **Management** for their kind co-operation and for providing infrastructure facilities.

We extend thanks to all the **Teaching staff** of the Department of CSE for their support and encouragement during the course of my project work. I also thank the **non-Teaching staff** of CSE department for being helpful in many ways in successful completion of my work.

Finally, we thank all those who helped me directly or indirectly in successful completion of this project work.

P.PRAVALLIKA	(20KE1A0584)
K.SRI LIKHITHA	(20KE1A0542)
R.SAI SNEHA	(20KE1A0591)
CH.AMRUTHA VANI	(20KE1A0511)
Y.VEERAMMA	(20KE1A05B1)

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 admin login. It also has a facility of admin login through which the admin can monitor the whole system. 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.

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.

INDEX

S.NO	CONTENT	PAGE.NO
1.	INTRODUCTION	1 - 4
	Project aims and objectives.	
	Background of the project.	
	User characteristics.	
	Product function overview.	
2.	SYSTEM ANALYSIS	5 - 10
	a. Existing system	
	b. Proposed system	
	c. Requirement Analysis	
	d. Input and Output design	
3.	SOFTWARE TOOLS USED	11 - 14
	a. Frontend : html, css ,java-script	
	b. Backend : python with Flask	
	c. Database : MongoDB	
4.	SYSTEM DESIGN	15 - 19
	a. Dataflow diagrams.	
5.	SOFTWARE ENVIRORNMENT	20 - 37
	a. Python	

b. Flask

6.	IMPLEMENTATION	38
	6.1 Module description	
7.	SOURCE CODE	39 - 75
8.	CODE OUTPUTS	76 - 81
9.	CONCLUSION	82
10.	REFERENCES	83 - 84

1. INTRODUCTION

In the present age of information', automation has been making tremendous impact on different sectors of Information Technology. In fact, 'automation' is an indispensable part of any field's development, organization, management and services. Automation can improve productivity and quality.

In this aspect, nature of modern librarianship has also changed considerably with the advent of new technologies. Library automation is the general term for information and communications technologies that are used to replace manual systems in the library. Hence automation has become an indispensable part of modern library's information systems development, organization, management and services. It has been making tremendous impact on different sectors of the libraries.

Borrowing books, returning books or viewing the available books at the Library of our college is currently done manually where the student has to go to the Lib the list of books available and borrow the books if the book is a borrow book otherwise it is of waste for the student to come to the library to come to check for the books if the student doesn't get the book. Then the librarian checks the student id and allows the member to check out the book and the librarian then updates the member database and also the books database. These all takes a lot of time. Altogether current manual library management system is time consuming and tedious job for the librarian as well as the user. Realizing the importance of implementing automation in existing manual library management system, here through this project "DIGITAL LIBRARY SYSTEM" we are automating the library management system. This results in numerous specializations and of non-stop information decreasing comprehensive acquisition of documents for libraries, growing demands of information, increasing number of users, decreasing time consumption, reducing the difficulties of the librarian to manage the library etc. Automation has been playing a vital role in improving the capabilities of libraries/information centers towards attaining satisfaction Mechanization of their users. library management operations, predominantly by computerization, is known as library automation. Library automation is a sophisticated software developed for processing

enormous amount of raw data and check the available books at the Library. Students check the list of books available and borrow the books if the book is available otherwise it is a waste for the student to come to the library to check for the books if the student doesn't get the book. Then the librarian checks the student id and allows the member to check out the book and the librarian then updates the member database and also the books database. This all takes a lot of time. Altogether current manual library management system is time consuming and tedious job for the librarian as well as the user. Realizing the importance of implementing automation in existing manual library management system, here through this project "DIGITAL LIBRARY SYSTEM" we are automating the library management system. This results in numerous specializations and of non-stop information decreasing comprehensive acquisition of documents for libraries, growing demands of information, increasing number of users, decreasing time consumption, reducing the difficulties of the librarian to manage the library etc. Automation has been playing a vital role in improving the capabilities of libraries/information centers towards attaining satisfaction Mechanization of their users. library management operations, predominantly by computerization, is known as library automation. Library automation is a sophisticated software developed for processing enormous amount of raw data into meaningful and useful form of information with speed, accuracy and reliability. Library automation may be defined as the application of automatic and semi- automatic data processing machines (computers) to perform traditional library house keeping activities such as acquisition, circulation, cataloguing and reference and serial control. Finally, library automation is the process of performing all information operations/ activities in library with the help of computers and related information technologies.

1.1 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 are as follows:

- Online book issue

- Request column for librarian for providing new books

- Student login page where student can find books issued by him/her and date of return. A

- search column to search availability of books.

1.2 BACKGROUND OF THE PROJECT

Digital Library 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.

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.

1.3 USER CHARACTERSTICS

Digital Library System contains mainly three types of users: Librarian. Students and Faculties.

Librarian has the overall control of the system. The main function of Librarian is to add books as well as new users. Librarian is the person who performs the major tasks such as issuing, returning, fine calculating etc...

Students have more benefits in using this system. They can avoid the wastage of time in searching an unavailable book on library. Many other facilities like viewing the due book details, fine amount details are provided for the users. Reservation is another important feature included in the system which is highly beneficial for the user. Faculties are also provided with the same facilities as that of student.

1.4 . Product function overview

A library is a collection of resources and services, and the structure in which it is housed: it is organized for use and maintained by a public body, an institution, or a private individual. In the more traditional sense, a library is a collection of books.

This project of "DIGITAL LIBRARY SYSTEM" of gives us the complete information about the library. We can enter the record of new books and retrieve the details of books available in the library. We can issue the books to the students and maintain their records and can also check how many books are issued and stock available in the library. In this project we can maintain the late fine of students who returns the issued books after the due date.

The Digital Library System is designed & developed for a receipt and issuance of books in the library along with the student's details. The books received in the library are entered in Books Entry form and the new student is entered in the student entry form. When the student wants to get the desired book the same is issued on the availability basis to the student. The issuance and due date for the returning of the book is also entered into the Book Issue form under menu Book Issue. The student has to pay the fine if any on the basis of no. of days delayed deposit of the book in the library.

2. SYSTEM ANALYSIS

2.1. Existing System

presently we are using manual card system for library management. The staff of the library itself should note every transactions occurring at the library like issuing the book, adding new books, returning of books etc. For each student a member ship form will be provided. Then after filling that form they get a member deforms to issue a book, student has to bring their library card. Then the librarian will do every transaction. Librarian will keep a register for to store the entire details of the books like, name of the book, author name, accession number, classification number etc.

After thoroughly examining the features and working of current library system, the following drawbacks have been identified, which severely affect the performance of the existing system.

Drawbacks of present manual system :

- Fast report generation is not possible
- Retrieving books from a library is very important for many peoples. The main problem is to define whether this book is exactly what the user wants and where this book resides in the library.
- Information about issue/return of the books are not properly maintained.
- Students can't get information about books easily.
- Wasting a lot of time looking for the book in the wrong location in the library
- Search process consume al lot of the librarian time as well as the visitor and it could be overwhelming for the librarian specially if they have to deal with more than one visitor at the time.

2.2 Proposed System

The proposed Library Management System that we had designed will overcome the drawbacks of the existing manual system. It is a browser-based Library Automation system that offers many flexible and convenient features, allowing librarians and library users to maximize time and efficiency. On a networked system this library software can enable users to save on their time too, as well as automate several manual processes. This system can be used to search for books/magazines, reserve books, find out who is having a particular book, put in requests to buy a new book etc... All details about the library like name of the members, name of books, categories of books, who has issued a book currently etc..., will be stored in a database. If any transactions has occurred in the library it will be reflected into this database so that all the data's in the database is correct and up-to-date. The librarian should be able to include new books/journals or remove some books from the inventory add new users to the system.

The main use of the Online Library Management System it converts the manual application to the online application. The different areas where you can use this system are: any education institution can make use of it providing information about author, content of the available books. It can be used in offices and modifications can be easily done according to requirements. It provides benefit to the user as well as the library.

The main advantages for the proposed system are:

- It provides "better and efficient" service to members
- Reduce the workload of employee
- Complete Books and Member Management
- Faster retrieval of information about the desired book
- Provide facility for proper monitoring reduces paper work and provide data security.
- All details will be available on a click.
- Online records of all the books in the library.

- Add / Edit / Delete book description.
- Online records of all the transaction made by students
- Record issued books details
- Search for books by Book name
- Search for books by Author name
- Provides advanced search also.
- ☒ Can view the books which has crossed returning date.
- Get book information online
- Can provide free as well restricted access to user.

2.3 REQUIREMENT ANALYSIS

Designing the input and output for a library management system using full stack web development involves creating user interfaces for various functionalities. Here's an outline of the input and output design for the Distil Library Management System:

- **Functional Requirements**

Graphical User interface with the User.

- **Software Requirements**

- For developing the application the following are the Software Requirements:

1. Python
2. Flask

- **Operating Systems supported:**

- Windows 10 64bit OS
- **Technologies and Languages used to Develop:**
 - Python
- **Debugger and Emulator:**

Any Browser (Particularly Chrome)
- **Hardware Requirements:**
 - For developing the application the following are the Hardware Requirements:
 - High Performance servers for hosting.
 - Ram : 16GB+
 - Rom : Min 500GB SSD
 - Internet Adapter : 1Gbps

2.4 INPUT AND OUTPUT DESIGN:

2.4.1 INPUT DESIGN :

User Registration: Allow users to register with the system, providing necessary information like name, email, and password.

Book Entry: Provide a form for librarians to input book details such as title, author, ISBN, genre, publication date, etc.

Member Information: Allow librarians to register new members by collecting their details such as name, contact information, and membership type.

Search Functionality: Implement a search interface where users can input keywords, author names, or ISBN to find books easily.

2.4.2 OUTPUT DESIGN :

Search Results: Display search results in a clear and organized manner, showing book titles, authors, availability status, and relevant details.

Book Details: Show detailed information about each book including title, author, ISBN, availability status, and any associated images or descriptions.

Member Profile: Present members' profiles with details such as name, contact information, membership status, and borrowing history.

Reports: Generate reports such as overdue books, popular genres, and member activity summaries in a readable format for librarians to analyze.

2.4.3 USER INTERFACE DESIGN PRINCIPLE :

Intuitive Navigation: Design an easy-to-use interface with clear navigation paths to different sections of the system.

Consistency: Maintain consistent design elements and layout across all screens for cohesive user experience.

Responsiveness: Ensure the system is responsive and works well on various devices including desktops, tablets, and smartphones.

Accessibility: Make the interface accessible to all users, including those with disabilities, by adhering to accessibility standards and providing options for screen readers and keyboard navigation.

Error Handling: Display helpful error messages and provide guidance to users in case of incorrect inputs or system errors.

By incorporating these design principles, the input and output components of the digital library management system can offer a user-friendly experience for both librarians and patrons.

3. SOFTWARE TOOLS USED

2.1 FRONTEND:

HTML- HTML or Hyper Text Markup Language is the main mark up language for creating web pages and other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like

), within the web page content. HTML tags most commonly come in pairs like and although some tags represent empty elements and so are unpaired, for example . The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, further tags, comments and other types of text-based content. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

CSS- Cascading Style Sheets(CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts.

This separation can improve content accessibility, provide more flexibility and control in the specification. of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design). CSS can also allow the same markup page to be

presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the author or the reader did not link the document to a specific style sheet the default style of the browser will be applied. CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

JAVA SCRIPT- JavaScript(JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications. JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multiparadigm language, supporting object-oriented, imperative, and functional programming styles. The application of JavaScript to use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but just-in-time compilation is now performed by recent (post-2012) browsers.

2.2 BACKEND

PYTHON :

- Python is a widely used general-purpose, high level programming language.
- It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
- Python is a programming language that lets you work quickly and integrate systems more efficiently.
- Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.
- Python is Interpreted Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive - You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented - Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python's Features includes-

- **Easy-to-learn-** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** - Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** - Python's source code is fairly easy-to-maintain.
- A broad standard library Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** - Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable You can add low-level modules to the Python interpreter. These modules

enable programmers to add to or customize their tools to be more efficient.

- Databases Python provides interfaces to all major commercial databases.
- GUI Programming Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows, MFC, Macintosh, and the X Window system of Unix.
- Python provides a better structure and support for large programs than shell scripting.

2.3 DATABASE

MONGODB :

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

A simple MongoDB document Structure:

```
{  
  title: 'Geeksforgeeks',  
  by: 'Harshit Gupta',  
  url:  'https://www.geeksforgeeks.org',  
  type: 'NoSQL'  
}
```

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

4. SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM:

Data Flow Diagram(DFD) depicts the flow of information and the transformation applied when the data moves in and out of a system . The overall system is represented and described using input, processing and output in the DFD. The inputs can be :

1. Book Request : When a student requests for a book.

2. Library Card : When the student has to show or submit his/her identity as proof. The overall processing unit will contain the following output that a system will produce or generate :

The book will be the output as the book demanded by the students will be given them. Information on the demanded book should be displayed by the library information system that can be used by the student while selecting the book which makes it easier for the student.

Level-1 DFD :

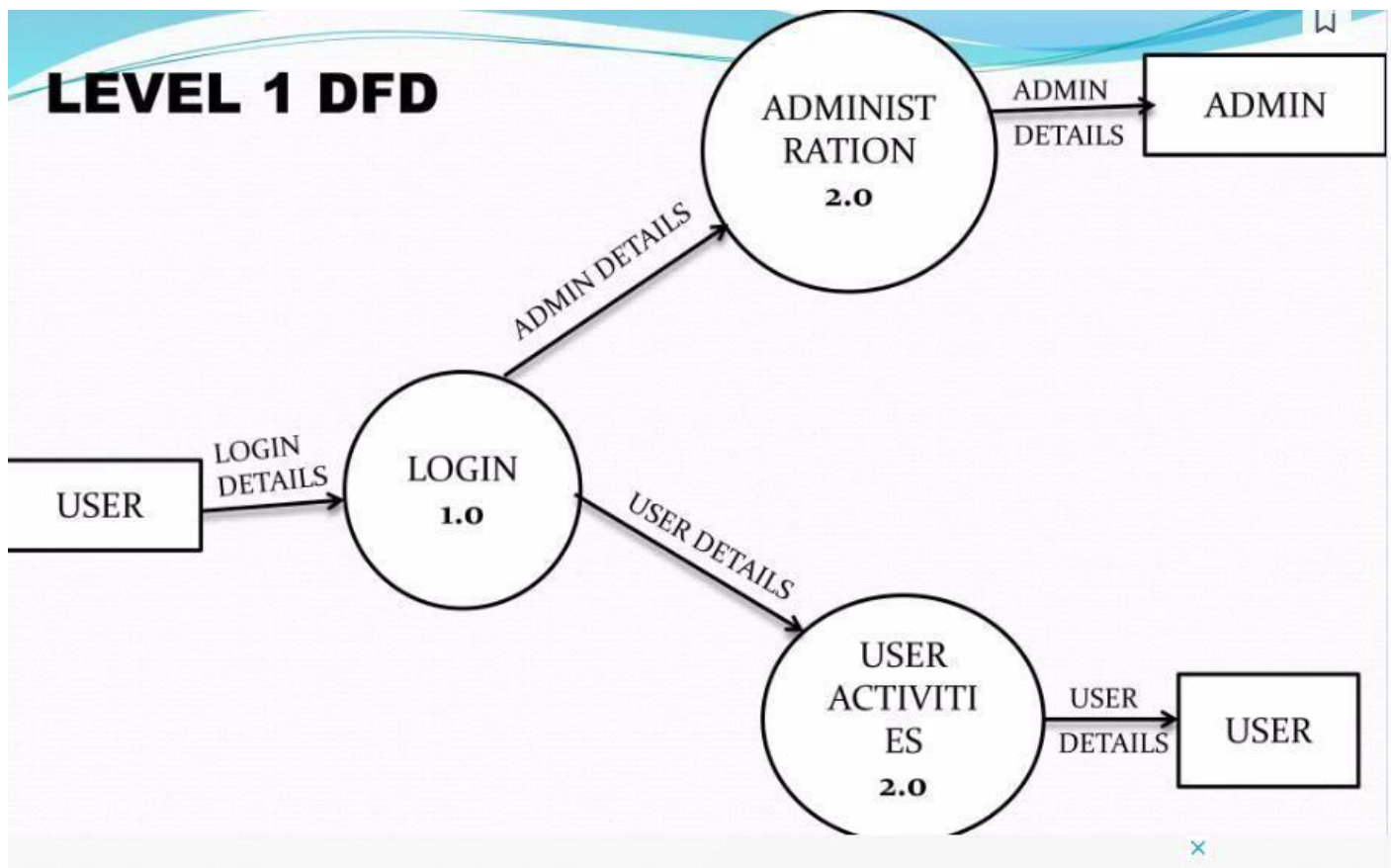


Fig4.1

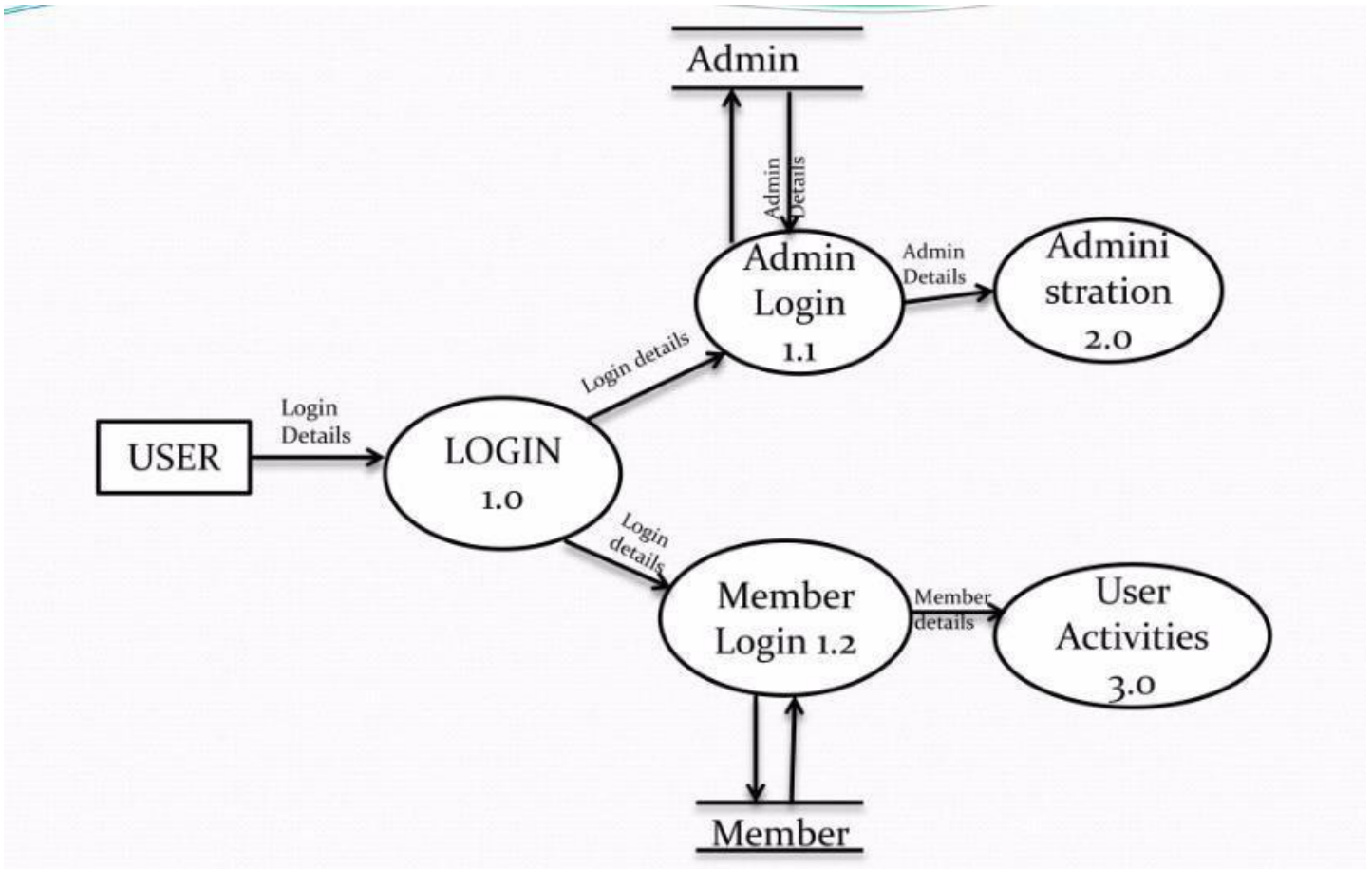


Fig4.2

Level 2 DFD – At this level, the system has to show or exposed with more details of processing. The processes that are important to be carried out are:

- Book Delivery.
- Search by topic.

Fig4.3

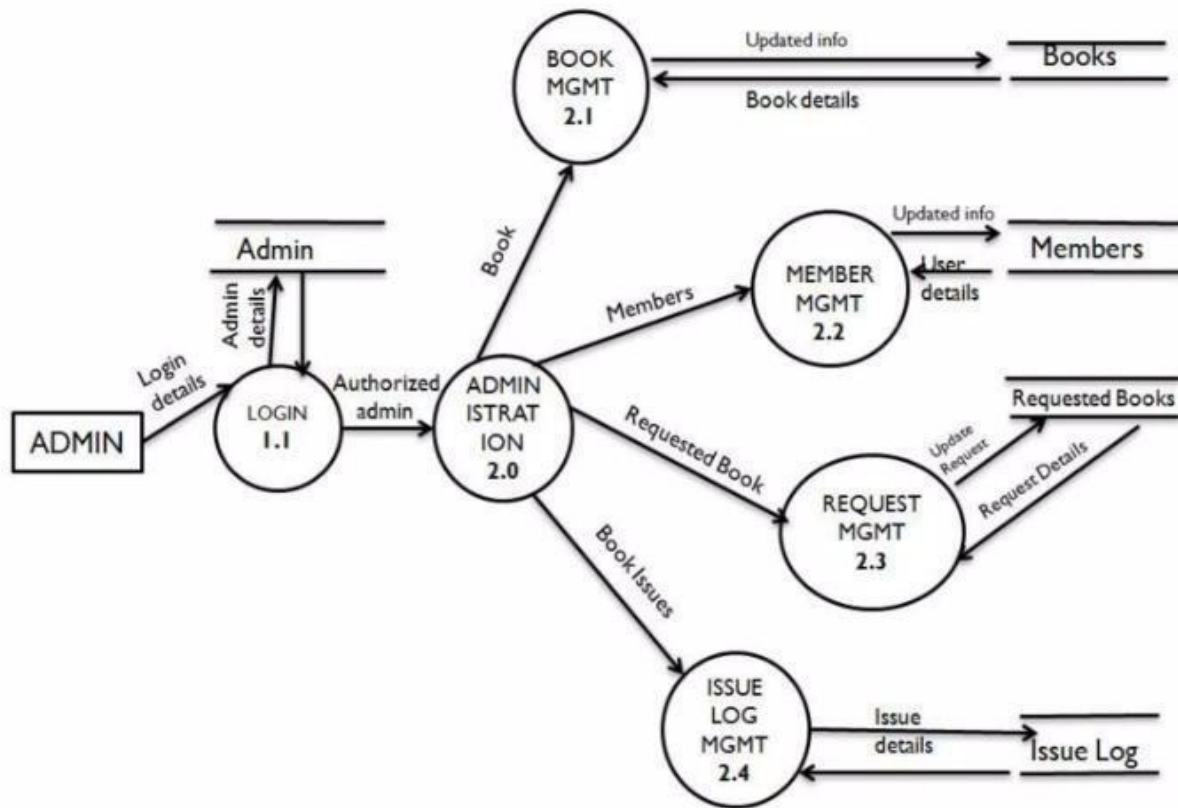
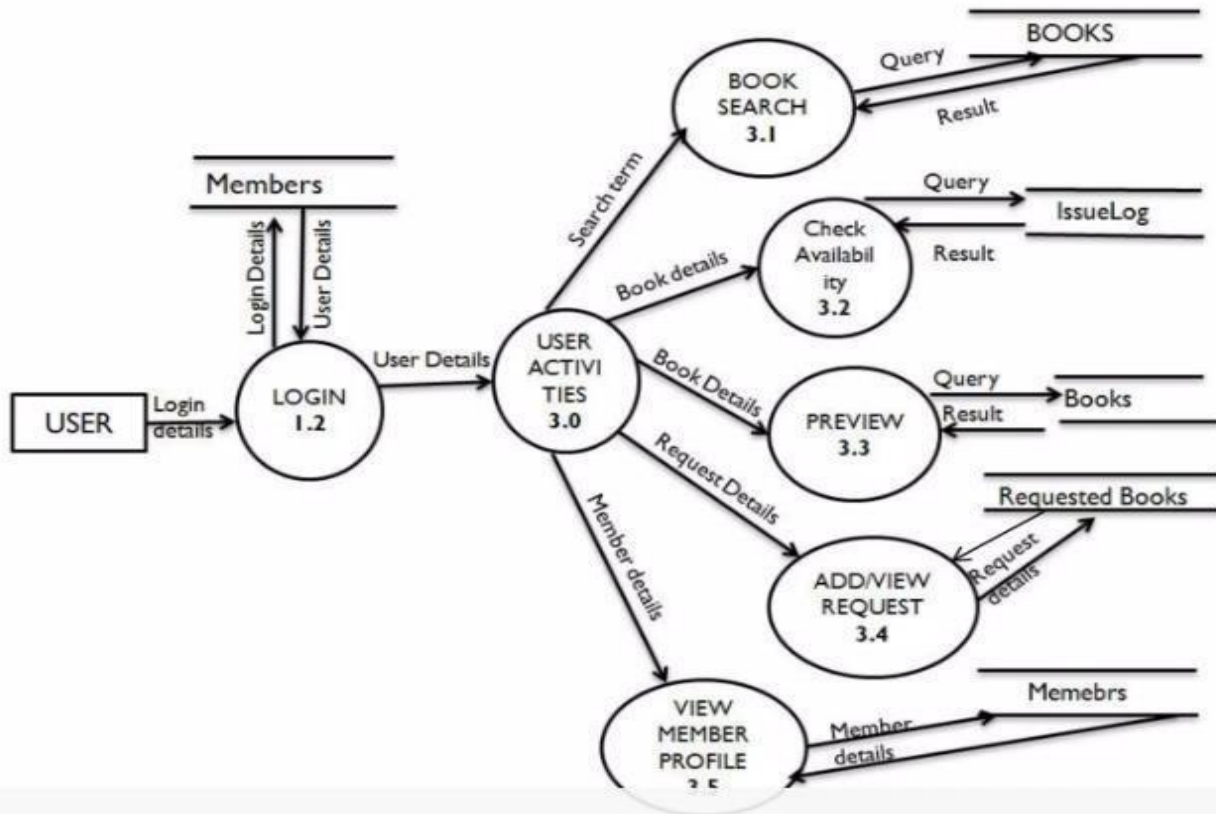


Fig4.4



5. SOFTWARE ENVIRONMENT

5.1 PYTHON

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems.

Scripting Language

A scripting or script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted (rather than compiled). Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting include software applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, as well as numerous games. A scripting language can be viewed as a domain-specific language for a particular environment; in the case of scripting an application, this is also known as an **extension language**. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction, or as control languages.

Object Oriented Programming Language

Object Oriented Programming Language Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known

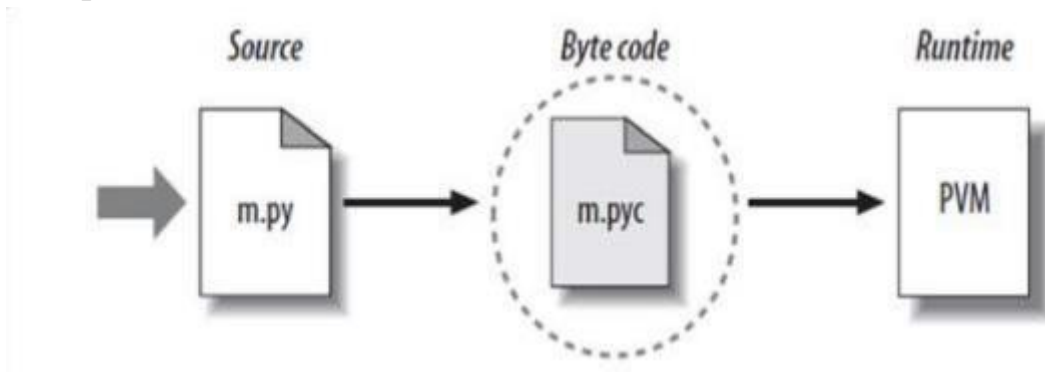
as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.

History

Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (BDFL).

Python Code Execution

Python's traditional runtime execution model: source code you type is translated to byte code, which is then run by the Python Virtual Machine. Your code is automatically compiled, but then it is interpreted.



code extension is .py

Byte code extension is .pyc (compiled python code)

Data Type

Data types determine whether an object can do something, or whether it just would not make sense. Other programming languages often determine whether an operation makes sense for an object by making sure the object can never be stored

somewhere where the operation will be performed on the object (this type system is called static typing). Python does not do that. Instead it stores the type of an object with the object, and checks when the operation is performed whether that operation makes sense for that object.

Python has many native data types. Here are the important ones:

- **Booleans** are either True or False.
- **Numbers** can be integers (1 and 2), floats (1.1 and 1.2), fractions (1/2 and 2/3), or even complex numbers.
- **Strings** are sequences of Unicode characters, e.g. an HTML document.
- **Bytes and byte arrays**, e.g. a JPEG image file.
- **Lists** are ordered sequences of values.
- **Tuples** are ordered, immutable sequences of values.
- **Sets** are unordered bags of values.

Variables

Python Variable is containers that store values. Python is not “statically typed”. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. It is the basic unit of storage in a program. In this article, we will see how to define a variable in Python

```
EX : counter = 100
      assignment miles = 1000.0
      name = "John" # A string
```

String

In programming terms, we usually call text a string. When you think of a string as a collection of letters, the term makes sense.

All the letters, numbers, and symbols in this book could be string. For that matter, your name could be a string, and so could your address.

Creating Strings

We can create a string by enclosing the characters in single-quotes or double-quotes. Python also provides triple-quotes to represent the string, but it is generally used for multiline string or docstrings.

Ex :

```
str1 = 'Hello Python'
print(str1)
str2 = "Hello Python"
print(str2)
str3 = """Triple quotes are generally used for
        represent the multiline or docstring"""
print(str3)
```

Output :

Hello Python

Hello

Python

Triple quotes are generally used for

represent the multiline or

docstring

PYTHON OPERATORS

- **ARITHMETIC OPERATORS** : Python Arithmetic Operators are used to perform mathematical operations like addition, subtraction, multiplication, and Division.

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a + b = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator	$a * b = 200$
/ Division	Divides left hand operand by right hand operand	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 0$
** Exponent	Performs exponential (power) calculation on operators	$a ** b = 10$ to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$

• LOGICAL OPERATORS

In Python, logical operators are used to combine multiple conditions together and evaluate them as a single boolean expression.

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

• COMPARISION OPERATORS

Comparison operators can compare numbers or strings and perform evaluations. Expressions that use comparison operators do not return a number value as do arithmetic expressions. Comparison expressions return either 1, which represents true, or 0, which represents false.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

- **ASSIGNMENT OPERATOR**

Operators are used to perform operations on values and variables. These are the special symbols that carry out arithmetic, logical, bitwise computations. The value the operator operates on is known as **Operand**.

Operator	Description	Example
=	Assigns values from right side operands to left side operand	$c = a + b$ assigns value of $a + b$ into c
$+=$ Add AND	It adds right operand to the left operand and assign the result to left operand	$c += a$ is equivalent to $c = c + a$
$-=$ Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	$c -= a$ is equivalent to $c = c - a$
$*=$ Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	$c *= a$ is equivalent to $c = c * a$
$/=$ Divide AND	It divides left operand with the right operand and assign the result to left operand	$c /= a$ is equivalent to $c = c / a$ $c /= a$ is equivalent to $c = c / a$
$\%=$ Modulus AND	It takes modulus using two operands and assign the result to left operand	$c \% a$ is equivalent to $c = c \% a$
$**=$ Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	$c ** a$ is equivalent to $c = c ** a$
$//=$ Floor Division	It performs floor division on operators and assign value to the left operand	$c //= a$ is equivalent to $c = c // a$

- BITWISE OPERATORS**

Bitwise operators are used to perform bitwise calculations on integers.

The integers are first converted into binary and then operations are performed on each bit or corresponding pair of bits, hence the name bitwise operators. The result is then returned in decimal format.

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> = 15 (means 0000 1111)

- **MEMBERSHIP OPERATORS**

In Python, the membership operators for strings are "in" and "not in".

These operators allow you to check the presence or absence of a substring within a given string. They return a boolean value - True if the substring is present, and False otherwise.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Tuples

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas.
- Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as read-only lists.

EXAMPLE :

```
tuple = ( 'abcd', 198 , 2.23, 'john', 70.2 ) tinytuple =
```

```
(123, 'john')
```

```
print tuple print
```

```
tuple[0] print
```

```
tuple[1:3] print
```

```
tuple[2:]
```

```
print tinytuple * 2
```

```
print tuple + tinytuple
```

Output:

```
('abcd', 198, 2.23, 'john', 70.2)
abcd (18, 2.23)
(2.23, 'john', 70.2)
(123, 'john', 123, 'john')
('abcd', 198, 2.23, 'john', 70.2, 123, 'john')
```

PYTHON DICTIONARY

- Python's dictionaries are kind of hash table type.
- They work like associative arrays or hashes found in Perl and consist of key- value pairs.
- A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- Example:

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
print dict['one']
print dict[2]
print tinydict
print tinydict.values() # Prints all the values
```

Output :

```
This is one
This is two
{'dept': 'sales', 'code': 6734, 'name': 'john'}
['dept', 'code', 'name']
['sales', 6734, 'john']
```

LISTS

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square

brackets ([]).

- To some extent, python lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Example:

```
list = [ 'abcd', 198 , 2.23, 'john', 70.2 ] tinylist
= [123, 'john']
print list
print list[0]
print list[1:3]
print list[2:]
print tinylist * 2
print list + tinylist
```

Output:

```
['abcd', 198, 2.23, 'john', 70.2]
abcd
[198, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 198, 2.23, 'john', 70.2, 123, 'john']
```

PYTHON FUNCTIONS

Functions are a convenient way to divide your code into useful blocks, allowing us to order our code, make it more readable, reuse it and save some time. Functions are a key way to define interfaces so programmers can share their code.

Function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

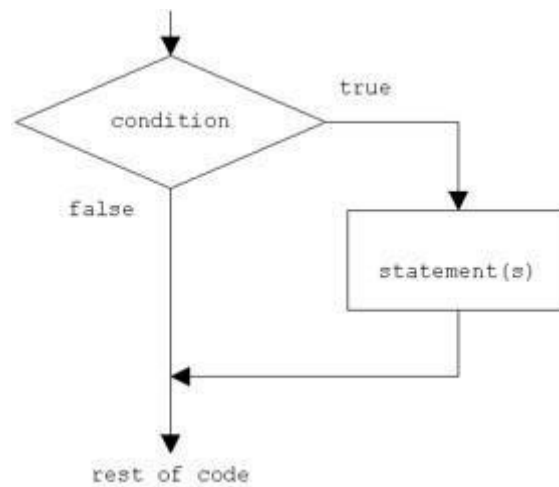
Function	Description
print()	Displayed function.
input ()	Get value from user.
int()	Convert integer data type.
float()	Convert float data type.
str()	Convert string data type.
type()	Displayed types of variables.
round()	Returns the nearest integer to its input.
max()	Returns the maximum value in a list.
min()	Returns the minimum value in a list.
sum()	Returns the sum of value in a list.
pow()	Return the power of values.

CONTROL STATEMENTS

control statements in Python are used to control the flow of execution of a program. The three types of control statements are break, continue, and pass. These statements allow us to selectively execute specific parts of the code based on certain conditions, optimize performance, and handle errors. By using control statements in Python effectively, we can write more efficient and error-free code.

- **IF STATEMENT**

In Python, the "if" statement assesses the test expression enclosed within parentheses. Should the test expression resolve to a true value (nonzero), the statements housed within the "if" block are executed. Conversely, if the test expression evaluates to a false value (0), the statements within the "if" block are bypassed.



- **SYNTAX :**

If expression :

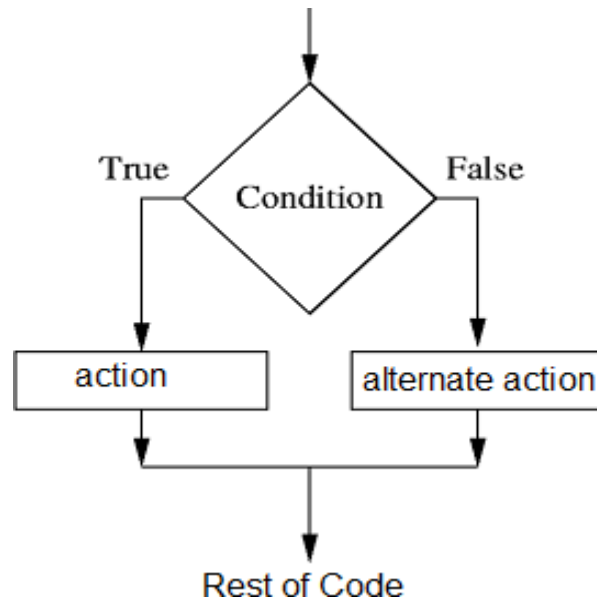
Statements

- **Example :**

```
x=2
0
y=1
0
if x > y :
    print(" X is bigger ")
```

IF – ELSE STATEMENT

The "else" statement serves the purpose of defining a code block that is to be executed when the condition within the "if" statement turns out to be false. As a result, the “else” clause guarantees the execution of a predetermined sequence of statements.



- **SYNTAX**

if expression:

 statements

else:

 Statements

- **Exampe**

X = 10

Y = 20

If

X>Y :

 Print("X is Bigger")

Else :

 Print ("Y is bigger")

- **NESTED IF :**
- **SYNTAX**

```

if condition:
    if condition:
        Statements
    else:
        statements
else:
    statements

```

- **Ex :**

```

mark = 72
if mark > 50:
    if mark >= 80:
        print ("You got A Grade !!")
    elif mark >= 60 and mark < 80 :
        print ("You got B Grade !!")
    else:
        print ("You got C Grade !!")
else:
    print("You failed!!")

```

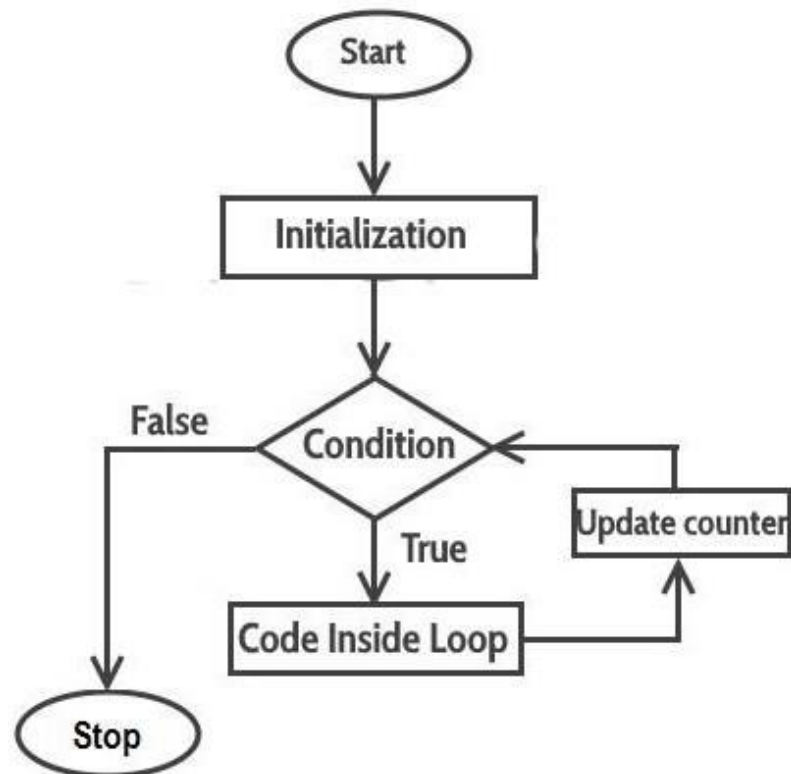
- **Output**
:

You got B Grade !!

LOOPING STATEMENTS

A loop constitutes a foundational programming concept frequently employed in software development. It entails a sequence of instructions that is reiterated until a specific condition is met. Within a for loop, there are two components: a header defining the conditions for iteration and a body executed

with each iteration. Typically, the header includes a designated loop counter or variable, enabling the body to discern the ongoing iteration.



5.2 FLASK

Flask is a small framework by most standards, small enough to be called a “micro framework.” It is small enough that once you become familiar with it, you will likely be able to read and understand all of its source code. But being small does not mean that it does less than other frameworks. Flask was designed as an extensible framework from the ground up; it provides a solid core with the basic services, while extensions provide the rest. Because you can pick and choose the extension packages that you want, you end up with a lean stack that has no bloat and does exactly what you need. Flask has two main dependencies. The routing, debugging, and Web Server Gateway Interface (WSGI) subsystems come from Werkzeug, while template support is provided by Jinja2. Werkzeug and Jinja2 are authored by the core developer of Flask. There is no native support in Flask for accessing databases, validating web forms, authenticating users, or other high-level tasks. These and many other key services most web applications need are available through extensions that integrate with the core packages. As a developer, you have the power to cherry-pick the extensions that work best for your project or even write your own if you feel inclined to. This is in contrast with a larger framework, where most choices have been made for you and are hard or sometimes impossible to change.

Using Virtual Environments :

The most convenient way to install Flask is to use a virtual environment. A virtual environment is a private copy of the Python interpreter onto which you can install packages privately, without affecting the global Python interpreter installed in your system. Virtual environments are very useful because they prevent package clutter and version conflicts in the system’s Python interpreter. Creating a virtual environment for each application ensures that applications have access to only the packages that they use, while the global interpreter remains neat and clean and serves only as a source from which more virtual environments can be created. As an added benefit, virtual environments don’t require administrator rights.

1.2 Installing Python Packages with pip

Most Python packages are installed with the pip utility, which virtual env automatically adds to all virtual environments upon creation. When a virtual environment is activated, the location of the pip utility is added to the PATH.

To install Flask into the virtual environment, use the following command:

```
(venv) $ pip install flask
```

With this command, Flask and its dependencies are installed in the virtual environment.

You can verify that Flask was installed correctly by starting the Python interpreter and trying to import it:

```
(venv) $ python  
>>> import flask  
>>>
```

If no errors appear, you can congratulate yourself: you are ready for the next chapter, where you will write your first web application.



SYNTAX -

1. Import Flask :

Import the Flask class from Flask library

From flask import Flask

2. Create an Application :

Create a Flask Application instance :

```
app = Flask( __name__ )
```

3. Define Routes :

Use the `@app.route` decorator to define routes(URLs) and the associated view functions (Python functions that handle requests):

```
@app.route('/')  
  
def home():  
  
    return 'Hello, World!'
```

4. Run the Application :

At the end of your script, add the following code to run the Flask

```
if __name__ == '__main__':  
    app.run(debug=True)
```

Complete Example :

```
from flask import  
  
app = Flask(  
    __name__  
  
    @app.route('/')  
  
    def home():  
  
        return 'Hello, World! '  
  
    if __name__ == '__main__':  
  
        app.run(debug=True)
```

You can run this Script and when you access the root URL of the server in your web browser, it will display 'Hello, World!'.

The `debug=True` argument is useful during development as it automatically reloads the server when code changes are detected.

6. IMPLEMENTATION

6.1MODULE DESCRIPTION

6.1.1 USER MANAGEMENT MODULE :

This module handles user authentication, registration, and management functionalities. Users can create accounts, login, reset passwords, and manage their profiles.

6.1.2 CATALOG MODULE :

This module is responsible for organizing digital resources within the library. It includes features for adding, editing, and deleting metadata associated with each resource. Metadata may include title, author, publication date, keywords, and other descriptive information.

6.1.3 SEARCH AND DISCOVERY MODULE :

This module enables users to search for digital resources based on various criteria such as title, author, subject, keyword, or category. It provides advanced search capabilities including filters, sorting options, and relevance ranking.

6.1.4 CONTENT MANAGEMENT MODULE :

This module handles the storage and retrieval of digital content including documents, images, videos, audio files, and other multimedia resources. It may integrate with a digital asset management system for efficient storage and organization.

6.1.5 ADMIN DASHBOARD MODULE :

This module provides administrators with a centralized dashboard for managing system settings, monitoring user activities, resolving issues, and performing administrative tasks such as backups, updates, and maintenance.

7. SOURCE CODE

App.py

```
from flask import Flask,render_template,request,session,send_file
from pymongo import MongoClient
import json
from werkzeug.utils import secure_filename
import os

cluster=MongoClient('mongodb+srv://Likitha:1234567890@cluster0.qygnado.mongodb.net/')

db=cluster['digitallibrary']

users=db['users']

books = db['books']

article = db['articles']

journal = db['journals']

app=Flask(__name__)

app.secret_key='50000'

UPLOAD_FOLDER = 'uploads'

app.config['UPLOAD_FOLDER'] =

UPLOAD_FOLDER

@app.route('/')

def land():

    return render_template('land.html')
```



```

@app.route('/register',methods=['post','get'])

def register():

    username=request.form['username']

    email=request.form['email']

    password=request.form['password']

    confirmpass=request.form['cpassword']

    k={}

    k['username']=username

    k['mail']=email

    k['password']=password

    res=users.find_one({"username":username})

    mail=users.find_one({"email":email})

    if res:

        return render_template('register.html',status="Username already exists") else:

    if mail:

        return render_template('register.html',status='Email already exists')

    elif password != confirmpass:

        return render_template('register.html',status='Passwords do not match') elif

    len(password)<8:

        return render_template('register.html',status="Password must be greater than 7

```

```

characters")

    else:

        users.insert_one(k)

        return render_template('register.html',stat="Registration successful")


@app.route('/login',methods=['post','get'])
def login():

    user=request.form['username']

    password=request.form['password']

    res=users.find_one({"username":user})

    if res and res['password']==password:

        session['name']=user

        return render_template('uhome.html')

    else:

        return render_template('login.html',status='User does not exist or wrong password')


@app.route('/log')
def log():

    return render_template('login.html')


@app.route('/reg')
def reg():

    return render_template('register.html')

```

```

@app.route('/adminlog')

def adlog():

    return render_template('adminlogin.html')

@app.route('/about')

def about():

    return render_template('about.html')

@app.route('/book')

def book():

    data = books.find()

    return render_template('book.html',books=data)

@app.route('/addBook')

def bookR():

    return render_template('booksform.html')

@app.route('/bookS',methods=['post'])

def bookS():

    # Get form data

    title = request.form['title']

    author = request.form['author']

    genre = request.form['genre']

    file = request.files['file']

    image = request.form['image']

```

```

# Save PDF file

filename = secure_filename(file.filename)

file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename)) # Insert data into MongoDB

book_data = {

    'title': title,

    'author': author,

    'genre': genre,

    'file_path': os.path.join(UPLOAD_FOLDER, filename),

    'image':image

}

books.insert_one(book_data)

return render_template('booksform.html',status='Book added successfully!')

@app.route('/journal')

def journal():

    data= journal.find()

    return render_template('journal.html',journals=data)

@app.route('/addjournal')

def journalR():

    return render_template('journalform.html')

@app.route('/journals',methods=['post'])

```

```

def journalS():
    # Get form data

    title = request.form['title']

    author = request.form['author']

    file = request.files['file']


    filename = secure_filename(file.filename)

    file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

    journal_data = {
        'title': title,
        'author': author,
        'file_path': os.path.join(UPLOAD_FOLDER, filename)
    }

    journal.insert_one(journal_data)


    return render_template('journalform.html', status='journal added successfully!')

@app.route('/article')

def articl():

    data = article.find()

    return render_template('article.html', articles=data)

```

```

@app.route('/addarticle')

def articleR():

    return render_template('articleform.html')


@app.route('/articles',methods=['post'])

def articleS():

    # Get form data

    title = request.form['title']

    author = request.form['author']

    file = request.files['file']

    filename = secure_filename(file.filename)

    file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

    article_data = {

        'title': title,

        'author': author,

        'file_path': os.path.join(UPLOAD_FOLDER, filename)

    }

    article.insert_one(article_data)

    return render_template('articleform.html',status='Article added successfully!')

@app.route('/view_pdf')

def view_pdf():

```

```

pdf_filename = request.args['book'] # Change this to the filename of the PDF you want to view
print(pdf_filename)

pdf_path = pdf_filename
print(pdf_path)

if os.path.exists(pdf_path):

    return send_file(pdf_path, as_attachment=False)

else:

    return "PDF file not found"

@app.route('/logout')

def logout():

    session['name']="

    return render_template('land.html')

@app.route('/adminlogin',methods=['post']) def

adminlogin():

    user=request.form['username']

    password=request.form['password']

    if user=='admin' and password=='1234567890':

        return render_template('home.html')

    return render_template('adminlogin.html',status='Wrong credentials')

    @app.route('/adminlogout')

def adminlogout():

```

```

    return render_template('land.html')

@app.route('/ubook')
def ubook():
    data=books.find()

    return render_template('ubook.html',books=data)

@app.route('/ujournal')
def ujournal():
    data=journal.find()

    return render_template('ujournal.html',journals=data)

@app.route('/uarticle')
def uarticle():
    data=article.find()

    return render_template('uarticle.html',articles=data)

@app.route('/y',methods=['post'])
def y():
    filter=request.form['filter']

    result = article.aggregate([
        {
            '$search': {
                'text': {
                    'query': filter,

```



```

        'path': [
            'author', 'genre', 'title'
        ]
    }
}
}
D)

```

```

v = article.aggregate([
    {
        '$search': {
            'text': {
                'query': filter,
                'path': [
                    'author', 'genre', 'title'
                ]
            }
        }
    }
])

```

```

if list(v)==[]:

```

```

        return render_template('uarticle.html',status="No results found") return
render_template('uarticle.html',articles=result)

@app.route('/x',methods=['post']) def
x():

    filter=request.form['filter']

    result = books.aggregate([

        {

            '$search': {

                'text': {

                    'query': filter,

                    'path': [

                        'author', 'genre', 'title'

                    ]

                }

            }

        }

    ])

    v = books.aggregate([

        {

            '$search': {

                'text': {

```

```

        'query': filter,
        'path': [
            'author', 'genre', 'title'
        ]
    }
}
}
})

```

```

if list(v)==[]:
    return render_template('ubook.html',status="No results found")
return render_template('ubook.html',books=result)

```

```

@app.route('/jsearch',methods=['post'])

```

```

def z():
    print('jsearch')
    filter=request.form['filter']
    result=journal.aggregate([
        {
            '$search': {
                'text': {

```

```

        'query': filter,
        'path': [
            'author', 'genre', 'title'
        ]
    }
}

}

D)
v = journal.aggregate([
    {
        '$search': {
            'text': {
                'query': filter,
                'path': [
                    'author', 'genre', 'title'
                ]
            }
        }
    }
}

D)

```

```
if list(v)==[]:

    return render_template('ujournal.html',status="No results found")

return render_template('ujournal.html',journals=result)

if __name__=='main_':

    app.run(debug=True)
```

Home.html :

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Digital Library</title>

<style>

    /* Reset */

    * {

        margin: 0;

        padding: 0;

        box-sizing: border-box;

    }
```

```
body {  
    font-family: Arial, sans-  
    serif; background-color:  
    #f9f9f9; color: #333;  
}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    text-align: center;  
    padding: 20px 0;  
}
```

```
nav {  
    background-color: #444;  
    text-align: center;  
    padding: 10px 0;  
}
```

```
nav ul {  
    list-style-type: none;
```

```
}

nav ul li {

    display: inline;

    margin: 0 10px;

}

nav ul li a {

    color:

    #fff;

    text-decoration: none;

    padding: 5px 10px;

}

nav ul li a:hover {

    background-color: #555;

}

.container {

    max-width: 800px;

    margin: 20px auto;

    padding: 0 20px;

}
```

```
.intro {  
    text-align: center;  
    margin-bottom: 30px;  
}
```

```
.advantages {  
    display: flex;  
    flex-wrap: wrap;  
    justify-content: space-between;  
}
```

```
.advantage {  
    width: calc(50% -  
    20px); background-  
    color: #fff; padding:  
    20px;  
    margin-bottom: 20px;  
    border-radius: 5px;  
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
}
```

```
.advantage h3 {
```



```
margin-bottom: 10px;  
}
```

```
.advantage p {  
    font-size: 0.9em;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<h1>Digital Library</h1>
```

```
</header>
```

```
<nav>
```

```
<ul>
```

```
<li><a href="#">Home</a></li>
```

```
<li><a href="/about">about</a></li>
```

```
<li><a href="/book">Books</a></li>
```

```
<li><a href="/journal">Journals</a></li>
```

```
<li><a href="/article">Articles</a></li>
```

```
<li><a href="/adminlogout">Logout</a></li>
```

```
</ul>
```

</nav>

<div class="container">

<section class="intro">

<h2>Welcome To Our Digital Library</h2>

<p>Access knowledge anytime and anywhere!</p>

</section>

<section class="advantages">

<div class="advantage">

<h3>Convenience</h3>

<p>Access to a vast collection of resources from your comfortable place.</p>

</div>

<div class="advantage">

<h3>24/7 Access</h3>

<p>No more limited library hours. Access information at any time of day and night.</p>

</div>

<div class="advantage">

<h3>Searchability</h3>

<p>Easily search and find specific information with just a few clicks.</p>

</div>

```
<div class="advantage">
```

```
<h3>Space Saving</h3>
```

```
<p>No need to worry about storing physical books. Digital materials take up no physical space.</p>
```

```
</div>
```

```
<div class="advantage">
```

```
<h3>Environmentally Friendly</h3>
```

```
<p>Reduce paper usage and environmental impact by accessing digital resources.</p>
```

```
</div>
```

```
</section>
```

```
</div>
```

```
</body>
```

```
</html>
```

ABOUT.html :

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>About Digital Library</title>
```

<style>

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
    background-color: #f7f7f7;  
}  
  
.container {  
    max-width: 800px;  
    margin: 20px auto;  
    padding: 20px;  
    background-color:  
    #fff; border-radius:  
    5px;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}  
  
h1 {  
    text-align: center;  
    color: #333;  
}  
  
p {  
    line-height: 1.6;  
    color: #666;  
}
```

```

    </style>

</head>

<body>

<div class="container">

    <h1>About Digital Library</h1>

    <p>Welcome to our digital library! We are dedicated to providing access to a wide range of
digital resources including books, articles, journals, and more. Our mission is to make knowledge
accessible to everyone, anytime, anywhere.</p>

    <p>Our library offers:</p>

    <ul>

        <li>Extensive collection of e-books </li>

        <li>Access to academic journals </li>

        <li>Easy search and navigation features</li>

        <li>User-friendly interface</li>

        <li>24/7 access from any device with an internet connection</li>

    </ul>

    <p>Whether you're a student, researcher, or simply love to rea

</div>

</body>

</html>

```

ARTICLE.html :

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Digital Library</title>
```

```
  <style>
```

```
    body {
```

```
      font-family: Arial, sans-serif;
```

```
      margin: 0;
```

```
      padding: 0;
```

```
    }
```

```
    header {
```

```
      background-color: #333;
```

```
      color: #fff;
```

```
      padding: 20px;
```

```
      text-align: center;
```

```
    }
```

```
    #books {
```

```
      display:
```

```
      flex;
```

```
flex-wrap: wrap;
justify-content: space-around;
margin-top: 20px;
}
```

```
.book {
width: 200px;
margin-bottom: 20px;
padding: 10px;
border: 1px solid #ddd;
border-radius: 15px;
margin: 15px;
}
```

```
.book img {
width:
100%;
height: auto;
}
```

```
.book h2 {
font-size: 18px;
margin-top: 10px;
}
```

```

.book p {
    margin: 5px 0;
}

.book a{
    text-decoration: none;
    color: white;
}

</style>

</head>

<body>

    <header>

        <h1>Digital Library</h1>

        <button><a href="/addarticle">Add Article</a></button>

    </header>

    <div style="display: flex;margin: 50px;">

        { % for i in articles % }

        <article class="book">

            <h2> {{ i['title'] }} </h2>

            <p>Author : {{ i['author'] }} </p>

            <center><button

```



```
style="margin-top: 10px;height: 30px;width: 100px;background: rgb(16, 160, 218);border-radius: 10px;"><a
```

```
href="/view_pdf?book={{ i['file_path'] }}"> Read</a></button>
```

```
</center>
```

```
</article>
```

```
{ % endfor % }
```

```
</div>
```

```
</body>
```

```
</html>
```

UARTICLE.html :

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Digital Library</title>
```

```
<style>
```

```
body {
```

```
font-family: Arial, sans-serif;
```

```
margin: 0;
```

```
padding: 0;
```

```
}
```

```
header {
```

```
    background-color: #333;
```

```
    color: #fff;
```

```
    padding: 20px;
```

```
    text-align: center;
```

```
}
```

```
#books {
```

```
    display:
```

```
    flex;
```

```
    flex-wrap: wrap;
```

```
    justify-content: space-around;
```

```
    margin-top: 20px;
```

```
}
```

```
.book {
```

```
    width: 200px;
```

```
    margin-bottom: 20px;
```

```
    padding: 10px;
```

```
    border: 1px solid #ddd;
```

```
    border-radius: 15px;
```

```
    margin: 15px;
```

```
}
```

```
.book img {  
  width:  
  100%;  
  height: auto;  
}
```

```
.book h2 {  
  font-size: 18px;  
  margin-top: 10px;  
}
```

```
.book p {  
  margin: 5px 0;  
}
```

```
.book a{  
  text-decoration: none;  
  color: white;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<h1>Digital Library</h1>
```

```

<form action="/y" method="post">
    <input type="text" placeholder="search" name="filter">
    <input type="submit" value="Search">
</form>
</header>
<div>
    <center style="margin-top: 50px;">
        {{ status }}
    </center>
</div>
<div style="display: flex;margin: 50px;">
    {% for i in articles %}
    <article class="book">
        <h2> {{ i['title'] }} </h2>
        <p>Author : {{ i['author'] }} </p>
        <center><button
            style="margin-top: 10px;height: 30px;width: 100px;background: rgb(16, 160,
218);border-radius: 10px;"><a
                href="/view_pdf?book={{ i['file_path'] }}"> Read</a></button>
        </center>
    </article>
    {% endfor %}
</div>

```

</body>

</html>

UBOOK.html :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Digital Library</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-color: #333;

color: #fff;

padding: 10px;

text-align: center;

}

#books {

```
display: flex;
flex-wrap: wrap;
justify-content: space-around;
margin-top: 20px;
}

.book {
width: 200px;
margin-bottom: 20px;
padding: 20px;
border: 1px solid #ddd;
border-radius: 25px;
margin: 10px;
}

.book img {
width:
100%;
height: auto;
}

.book h2 {
font-size: 18px;
margin-top: 10px;
}

.book p {
margin: 5px 0;
```

```

    }

    .book a{
        text-decoration: none;
        color: white;

    }
</style>
</head>
<body>
    <header>
        <h1>Digital Library</h1>
        <form action="/x" method="post">
            <input type="text" placeholder="search" name="filter">
            <input type="submit" value="Search">
        </form>
    </header>
    <div style="display: flex;margin: 50px;">
        <div>
            <center style="margin-top: 50px;">
                {{ status }}
            </center>
        </div>
        { % for i in books % }

```

```

<article class="book">

    <h2> {{i['title']}} </h2>

    <p>Author : {{i['author']}} </p>

    <p> Genre : {{i['genre']}} </p>

    <button style="margin-top: 10px;margin-left: 20px;height: 30px;width: 100px;background:
rgb(16, 160, 218);border-radius: 10px;"><a href="/view_pdf?book={{i['file_path']}}" >
Read</a></button>

</article>

{% endfor %}

</div>

</body>

</html>

```

UJOURNAL.html :

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Digital Library</title>

    <style>

```



```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    padding: 20px;  
    text-align: center;  
}
```

```
#books {  
    display:  
    flex;  
    flex-wrap: wrap;  
    justify-content: space-around;  
    margin-top: 20px;  
}
```

```
.book {  
    width: 200px;  
    margin-bottom: 20px;
```

```
padding: 20px;
border: 1px solid #5c5353;
border-radius: 15px;
margin: 10px;
}
```

```
.book img {
  width:
  100%;
  height: auto;
}
```

```
.book h2 {
  font-size: 18px;
  margin-top: 10px;
}
```

```
.book p {
  margin: 5px 0;
}
```

```
.book a{
  text-decoration: none;
  color: white;
}
```

```

</style>

</head>

<body>

  <header>

    <h1>Digital Library</h1>

    <form action="/jsearch" method="post">

      <input type="text" placeholder="search" name="filter">

      <input type="submit" value="Search">

    </form>

  </header>

  <div>

    <center style="color: red;margin-top: 30px;">

      {{ status }}

    </center>

  </div>

  <div style="display: flex;margin: 50px;">

    {% for i in journals %}

      <article class="book">

        <h2> {{ i['title'] }} </h2>

        <p>Author : {{ i['author'] }} </p>

        <center><button style="margin-top: 10px;height: 30px;width: 100px;background: rgb(16,
160, 218);border-radius: 10px;"><a href="/view_pdf?book={{ i['file_path'] }}" >
Read</a></button>

```

</center>

</article>

{% endfor %}

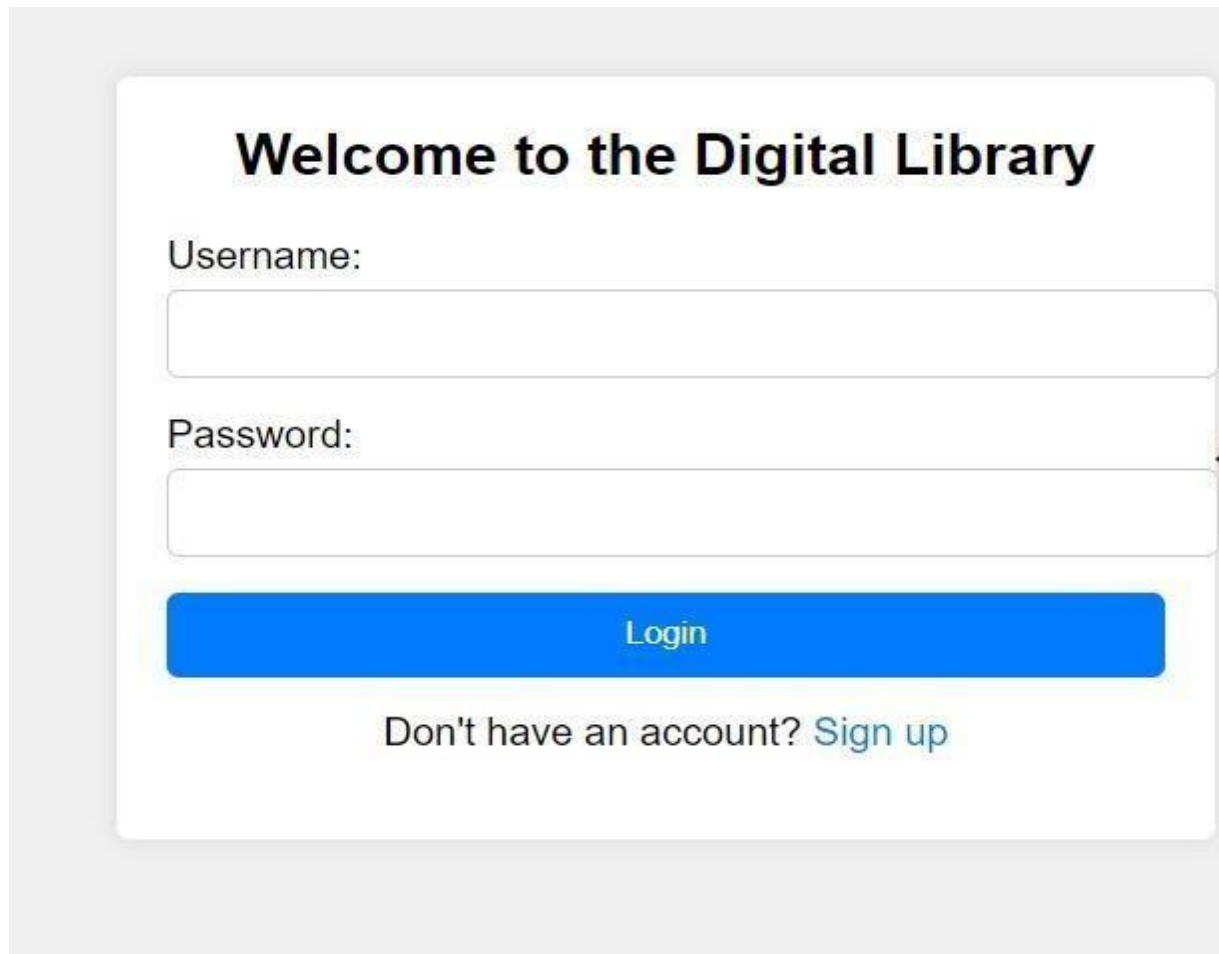
</div>

</body>

</html>

8. SCREENSHOTS

LOGIN PAGE :



The screenshot shows a login interface with a white card on a light gray background. The card has a title, two input fields, a login button, and a sign-up link.

Welcome to the Digital Library

Username:

Password:

[Login](#)

Don't have an account? [Sign up](#)

BOOKS PAGE :

WhatsApp

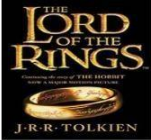
Digital Library

127.0.0.1:5000/ubook

GoogleNew TabNew folderGmailYouTubeMaps

Digital Library

Search




the lord of rings

Author : J. R. R. Tolkien

Genre : Fiction

Read

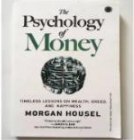


Bhagavad-Gita As It Is

Author : Vysa

Genre : Non-Fiction

Read

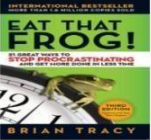


The Psychology Of Money

Author : Morgan

Genre : Science Fiction

Read

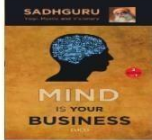


Eat That Frog

Author : Brain Tracy

Genre : Fantasy

Read



Mind Your Business

Author : Sadhguru

Genre : Fiction

Read

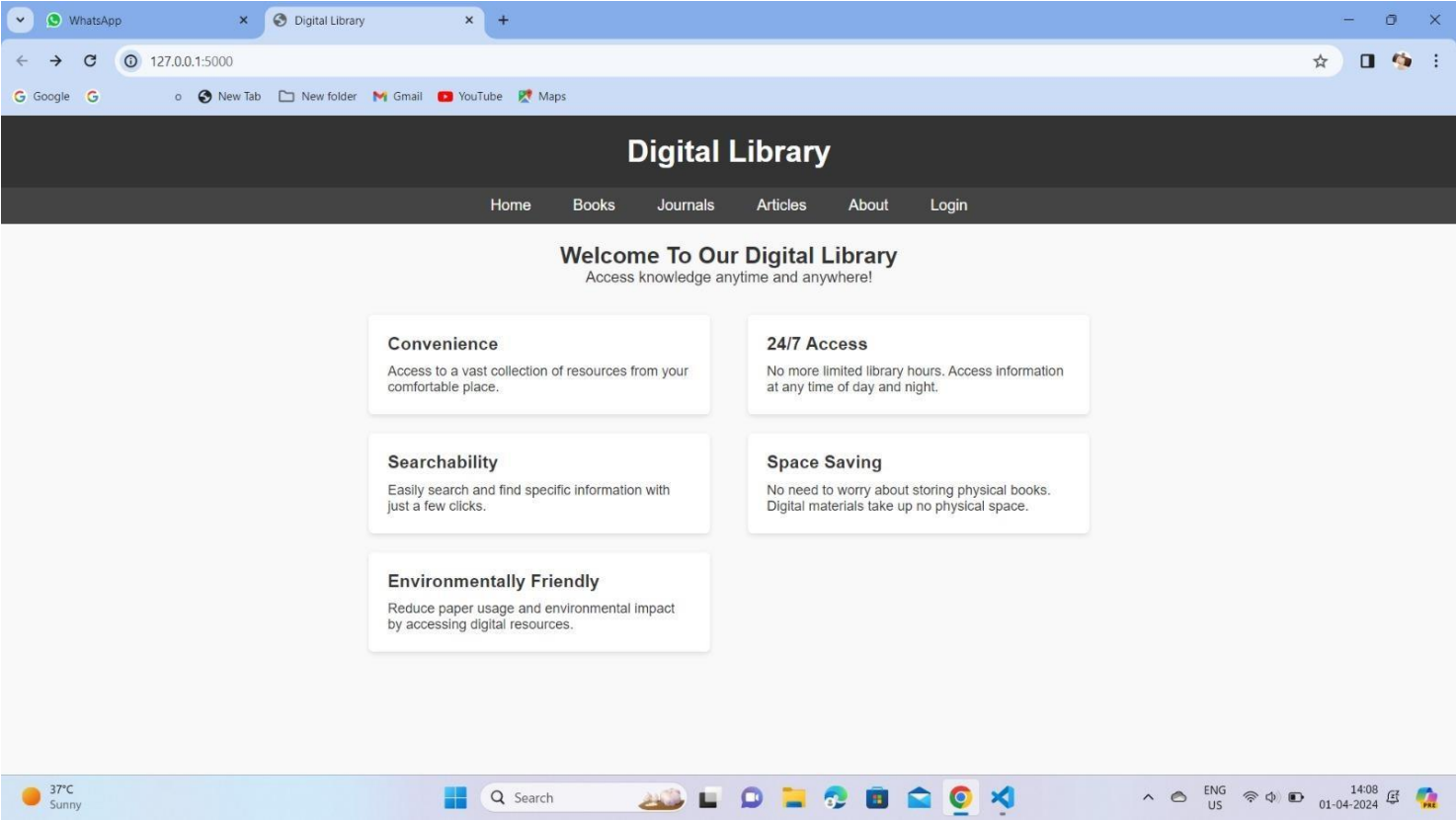
BSE smicap
+2.59%

Search

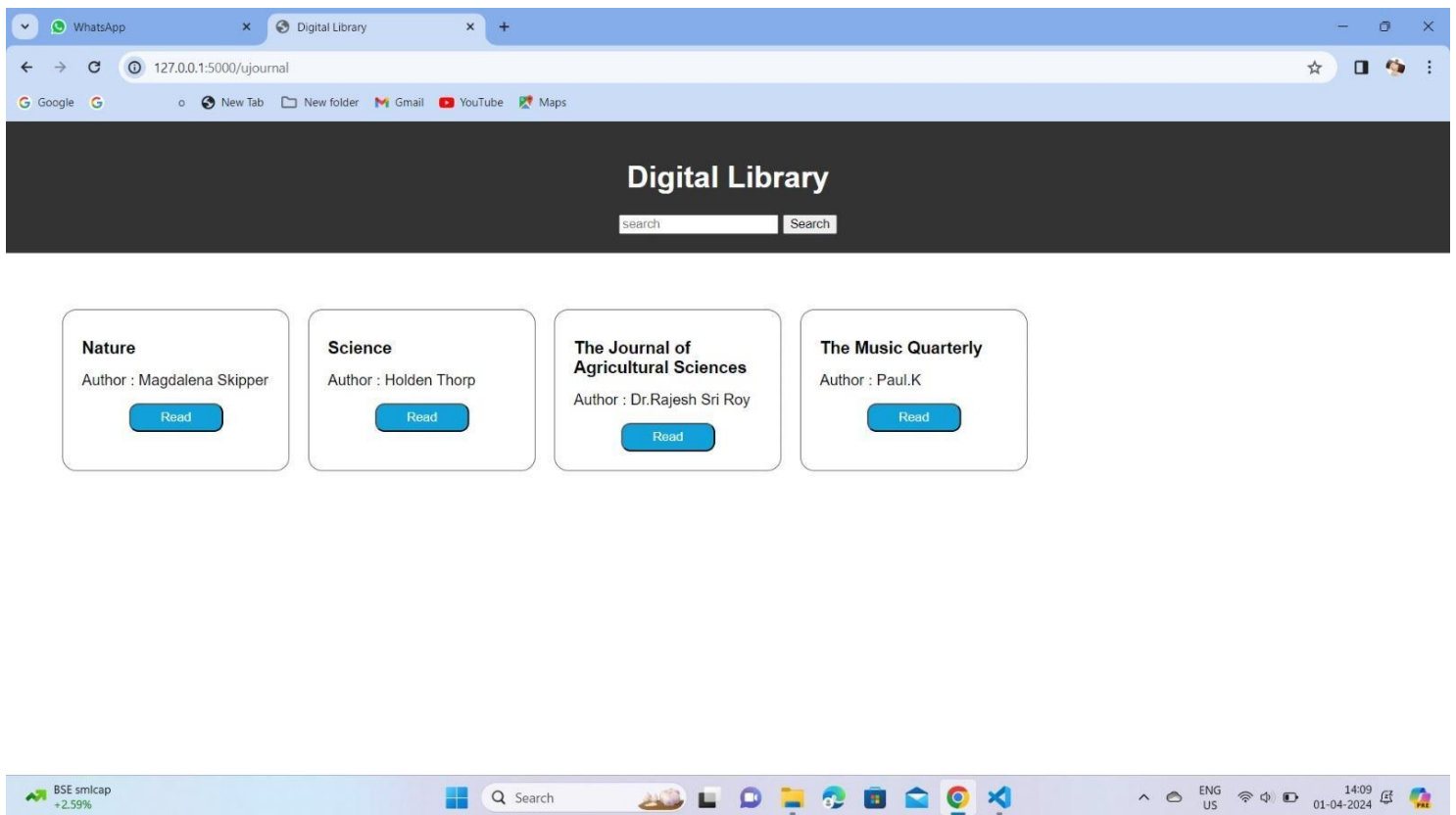
ENG
US

14:08
01-04-2024

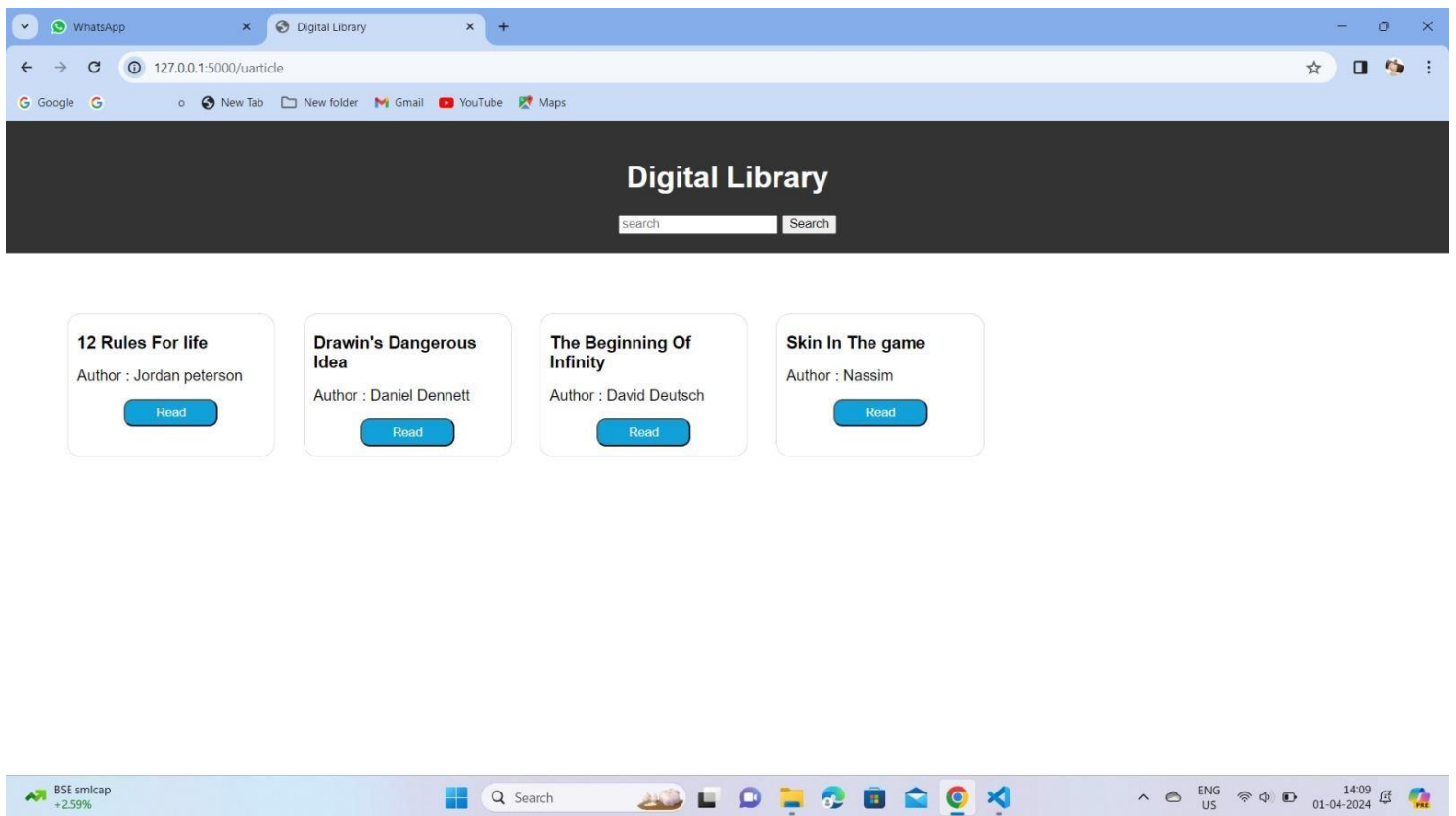
HOME PAGE :



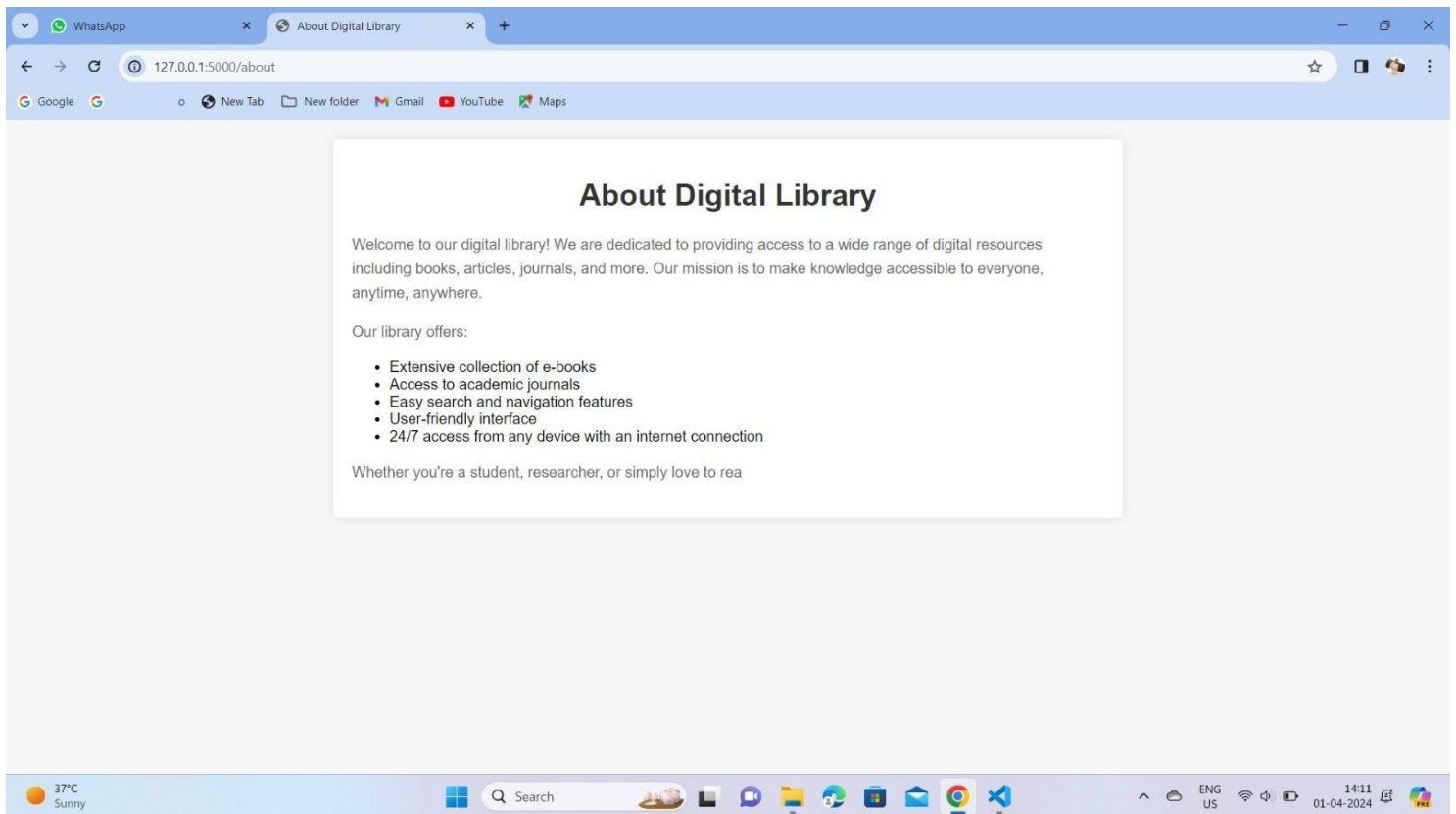
JOURNALS PAGE :



ARTICLES PAGE :



ABOUT PAGE :



9. CONCLUSION

In conclusion, the development of a full-stack digital library system involves a comprehensive process of analysis, design, implementation, and deployment. By leveraging a full-stack approach, we have successfully integrated frontend and backend technologies to create a robust and user-friendly platform for accessing and managing digital resources.

Throughout the project, we prioritized user requirements and feedback, ensuring that the system meets their needs effectively. The frontend interface provides an intuitive and responsive user experience, allowing users to search, browse, and access digital content with ease. Meanwhile, the backend infrastructure supports data management, authentication, authorization, and other core functionalities necessary for a digital library system.

By adopting best practices in software development, including modular design, agile methodologies, and continuous testing and integration, we have delivered a reliable and scalable solution. The system architecture is designed for flexibility and extensibility, allowing for future enhancements and integrations with external systems and services.

10. REFERENCES

- 1.** "Introduction to Digital Libraries" by G. Chowdhury and Sudatta Chowdhury - This book provides an overview of digital libraries, including their architecture, content, and management.
- 2.** "Digital Libraries: Principles and Practices" edited by Sangeeta Namdev Dhamdhare and Sunita Barve - This book covers various aspects of digital libraries, including digitization, metadata, and preservation.
- 3.** "The Digital Library Toolkit" by Terry Reese - This resource offers practical guidance on building and managing digital libraries, including software tools and best practices.
- 4.** "Digital Libraries: Implementation and Management" by Judith M. Nixon and Johnathan P. Palmer - This book focuses on the practical aspects of implementing and managing digital libraries, including selection, acquisition, and user services.
- 5.** "Digital Library Development: The View from Kanazawa University" edited by Atsuyuki Morishima and Fuyuki Yoshikane - This book presents case studies and insights from the development of digital libraries at Kanazawa University, providing real-world examples and lessons learned.
- 6.** "Digital Libraries: Challenges and Opportunities" edited by Aparajita Suman and Sanjay Kataria - This book explores the challenges and opportunities in the field of digital libraries, including issues related to access, preservation, and interoperability.
- 7.** "Building Digital Libraries: A How-To-Do-It Manual" by Terry Reese - This practical guide provides step-by-step instructions for building digital libraries, covering topics such as digitization, metadata creation, and user interface design.
- 8.** "Digital Preservation for Libraries, Archives, and Museums" by Edward M. Corrado and Heather Moulaison Sandy - This book focuses on the important aspect of preserving digital content in libraries, archives, and museums, including strategies for long-term access and sustainability.
- 9.** "Digital Libraries and Archives" by Donald J. Waters and Kim M. Thompson - This comprehensive text covers the theory and practice of digital libraries and archives, including

collection development, metadata standards, and digital preservation.

10. "The Handbook of Digital Library Economics" edited by Wendy Pradt Lougee and Eileen G. Abels - This handbook examines the economic aspects of digital libraries, including funding models, cost-benefit analysis, and sustainability strategies.