# INTRODUCTION

In the current scenario there isn’t any application that would help a tourist to get information about the place they are currently visiting in their mobile phone. Our application Sight Seeing is aimed to solve this problem. In current tourism system, whenever a tourist visits famous spots, to know more about the place he hires a guide. The hired guide then narrates history of the place. The proposed system doesn’t require a physical guide. The Mobile application installed on the mobile of tourist can act as a guide. Without having a guide, it will help one to get information of the place in their mobile and check out the nearest places also with the help of our application. Mobile Technology is now set to help boost tourism in various fields. It will be used to apply mobile applications technology to create specific visitor information for tourist attractions to effectively turn the tourist's mobile phone into their very own personalized tour guide. Mobile phone technology presents huge potential for the future of visitor interpretation at tourist attractions. People want fast, efficient and easy access to information of all kinds and tourism is no different. While tourism presents considerable potential for the use of new mobile technologies, we currently have little understanding of how tourists organize their activities or of the problems they face. This project presents an ethnographic study of city tourists' practices that draws out a number of implications for designing tourist technology

# TECHNOLOGY PROFILE

## 2.TECHNOLOGY PROFILE JSP

**Java Server Pages** (**JSP**) is a technology that helps [software developers](https://en.wikipedia.org/wiki/Software_developer) create [dynamically generated](https://en.wikipedia.org/wiki/Dynamic_web_page) [web pages](https://en.wikipedia.org/wiki/Dynamic_web_page) based on [HTML,](https://en.wikipedia.org/wiki/HTML) [XML,](https://en.wikipedia.org/wiki/XML) or other document types. Released in 1999 by [Sun](https://en.wikipedia.org/wiki/Sun_Microsystems) [Microsystems,](https://en.wikipedia.org/wiki/Sun_Microsystems) JSP is similar to [PHP](https://en.wikipedia.org/wiki/PHP) and [ASP](https://en.wikipedia.org/wiki/Active_Server_Pages), but it uses the [Java programming language.](https://en.wikipedia.org/wiki/Java_(programming_language))

To deploy and run Java Server Pages, a compatible web server with a [servlet container](https://en.wikipedia.org/wiki/Servlet_container), such as [Apache Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat) or [Jetty,](https://en.wikipedia.org/wiki/Jetty_(web_server)) is required.

JSP pages use several delimiters for [scripting](https://en.wikipedia.org/wiki/Server-side_scripting) functions. The most basic is **<% ... %>**, which encloses a JSP *scriptlet.* A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include **<%= ... %>** for *expressions,* where the scriptlet and delimiters are replaced with the result of evaluating the expression, and *directives*, denoted with **<%@ ... %>**[.[5]](https://en.wikipedia.org/wiki/JavaServer_Pages#cite_note-syntax-5)

Java code is not required to be complete or self-contained within a single scriptlet block. It can straddle markup content, provided that the page as a whole is syntactically correct. For example, any Java *if/for/while* blocks opened in one scriptlet must be correctly closed in a later scriptlet for the page to successfully compile.

### JAVA SCRIPT

Java Script is a high-level, dynamic and interpreted programming language. It has been standardized in the ECMA Script language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is prototype-based with first class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphic facilities, relaying for these upon the host environment in which it is embedded.

Because Java Script code can run locally in a user’s browser (rather than on a remote server), the browser can respond to user actions quickly, making an application more responsive. Furthermore, Java Script code can detect user actions that HTML alone cannot, such as individual key strokes. Applications such as Gmail take advantage of this: much of the user-interface logic is written in Java

Script, and Java Script dispatches requests for information (such as the content of an email message) to the server. The wider trend of Ajax programming similarly exploits this strength.

#### MySQL

MySQL is an open-source relational database management system (RDBMS); it is the world’s second most widely used RDBMS, and the most widely used open-source client-server model RDBMS. The SQL acronym stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a popular choice of database for using web applications, and is a central component of the widely used LAMP open-source web application software stack (and other "AMP “stacks). LAMP is an acronym for “Linux, Apache, MySQL, Pearl, PHP, and Python ". Free- software open-source projects that require a full-featured database management system often use MySQL.

#### HTML

**Hypertext Markup Language**, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

HTML can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. HTML markup can also refer the browser to Cascading Style Sheets (CSS) to define the look and layout of text and other material.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, of CSS over explicit presentation HTML markup.

# SYSTEM STUDY

## 3.SYSTEM STUDY

System study is an important phase of any system development process. The system is studied to the minute detail and analyzed. The system is studied to the minute detail and analyzed. The system analyst dwelled deep into the working of the present system. The system was viewed as a whole and the input of the system are identified. During analysis phase for each problem identified many alternative solutions were evaluated and selected the most feasible one. A feasibility analysis was performed to evaluate possible solutions to recommend the most feasible one.

### EXISTING SYSTEM

Placement Management System is to develop a student and company information system. Although such a project has a very wide scope, this system provides the facility to add the student’s and company’s details into database. Student details includes their academic information and personal information. Company details includes their company information and also the placement details such as date of placement, eligible criteria etc.

### PROPOSED SYSTEM

The proposed system gives a user-friendly operation i.e. No need to collect an information in paper. The student directly give their information in the system by login with their own username and password. Similarly the companies add their information in the system by login with their own username and password. The proposed system is designed to eliminate all the drawbacks of the existing system.

### MODULE DESCRIPTION

* + 1. Login module
    2. Update module
    3. Admission management module
    4. User management module
    5. Display module
    6. Feedback module

### Login module

Login is provided for both administrator and users. Administrators can manage the site only if logged in with their user id and password. Also for the students who have already created an account can manage their account by logging in with their id and password.

### Update module

This module is used to update the contents and status of the admission.

### Admission Management module

This module deals with the admissions of students which includes register admission and making fee payments etc.

### User management module

The users and their access to product and their accounts are managed in this module.

### Display module

This module deals with the display of admissions in the website.

### Feedback module

This module is used to manage the feedback provided by the users.

# SYSTEM SPECIFICATION

#### 4.1 SOFTWARE REQUIREMENTS

A software requirement specification (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non- functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints) the software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

Android Studio

Web Browser: Internet Explorer/Google Operating System: WINDOWS 8 or above for better performance

Front end: Python (For web application), Android (Mobile Application)

Back end: MYSQL

Software: SubLime Text, WAMP, Chrome/Firefox Web Server: Apache

#### 4.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

Processor: Intel Pentium or above. Hard Disc: 320GB.

Display Type: PC Display.

# FEASIBILITY ANALYSIS

## 5.FEASIBILITY ANALYSIS

The feasibility study proposes one or more conceptual solutions to the problem set for the project. The conceptual solution gives an idea of what the new system will look like. They define what will be done on the computer and what will remain manual. It also indicates what input will be needed by the system and what outputs will be produced. These solutions will be proven feasible and a preferred solution is accepted.

All projects are feasible if it is offered with unlimited resources and infinite time. The development of project is done by scarcity of resources and difficulty in completion dates.

The feasibility can be defined in three major areas namely.

* Economic
* Technical
* Functional

In the economic side, it is generally the bottom line consideration of the project. It will increase the efficiency and decrease the man-hour to achieve the result. It will provide timely and up to date to the administrative and individual departments. Since all the information is available in few seconds the system performance will be substantially increased.

In the technical side, it is most difficult area to access because objectives, functions performance are somewhat hazy; anything seems to be possible if right assumptions are made. The considerations that are normally associated with technical include development risk, technology and resource availability.

In the functional side, it is faster and efficient than the existing system. From all these, we can conclude that this system is feasible economically, technically and functionally.

Normally, the central endeavor of feasibility study is the cost benefit analysis of various alternatives. It can be defined as a systematic comparison between the cost of carrying out a service or activity and the value of that service or activity. The main benefits are quantitative and qualitative.

A feasibility study is really a small-scale system analysis. The job of analyst is to put on information together and present it to the client in the form of a coherent report.

The main section of feasibility report includes:

Background

* + Terms of reference
  + Reasons for study

The current situation

* + - Overview of current situation
    - Problem and requirements identified

Proposed solution

A description of the requirements of a new system along with a number of options explaining how the solution might be implemented.

# SYSTEM DESIGN

**6.1 1NPUT DESIGN**

Input design is the process of converting user oriented inputs to the computer base format. The input media is keyboard and mouse .details are entered through different data entry screens. Also each click events generated are converted and processed to generate outputs for those input. The following are the features of the data entry screen of the proposed system .

* User friendly .
* Data evaluation .
* Menu driven
  1. **OUTPUT DESIGN**

Computer output is the most important and direct source of information to the user. Designing the computer output should proceed in an organized,well- throughout manner. The correct output must be developed while ensuring that each output element is designed so that people will find the system easy to use efficiently. During the analysis computer output identify the specific output is needed to meet the information requirements.

* 1. **DATABASE DESIGN**

One of the most important tasks involved in the design phase is the design of data storage. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently.The general objective is to make access easy, quick and inexpensive and flexible for the users. Relationships are maintained between the data terms. Normalization is done to get an internal consistency of data and have to minimum redundancy and maximum stability. This ensures minimizing data storage required ,minimizing chances of data consistencies and optimizing for updates. That is visible for users of the system .the overall objective is development of database and treats data as original resources.

**TABLES**

USER

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes** | **Data type** | **constrains** | **Descriptions** |
| Username | Varchar(20) | Primary key | User name of user |
| Password | Varchar(20) | Not null | Password |
| Usertype | Varchar(20) | not null | Type of user |

STAFF

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Staff\_id | Int(12) | primary key | id of staff |
| Username | Varchar(20) | Not null | User name of staff |
| Firstname | Varchar(20) | Not null | First name of staff |
| Lastname | Varchar(20) | Not null | Last name of staff |
| Place | Varchar(20) | Not null | Place of staff |
| Phone | Varchar(20) | Not null | Phone number |
| Email | Varchar(20) | Not null | email |
| House | Varchar(20) | Not null | address |
| City | Varchar(20) | Not null | city |
| District | Varchar(20) | Not null | District |
| Pincode | Varchar(20) | Not null | pincode |

CUSTOMER

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Customer\_id | Int(12) | Primary key | Id of  customer |
| Username | Varchar(20) | Not null | User name of staff |
| Firstname | Varchar(20) | Not null | First name of staff |
| Lastname | Varchar(20) | Not null | Last name of staff |
| Phone | Varchar(20) | Not null | Phone number |
| Email | Varchar(20) | Not null | Email |
| House | Varchar(20) | Not null | Address |
| District | Varchar(20) | Not null | District |
| Pincode | Varchar(20) | Not null | pincode |

PUBLISHER

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Publisher\_id | Int(12) | Primary key | Id of publisher |
| Email | Varchar(20) | Not null | email |
| Publisher | Varchar(20) | Not null | Publisher |
| Phone | Varchar(20) | Not null | Phone number |
| City | Varchar(20) | Not null | City |
| District | Varchar(20) | Not null | District |
| pincode | Varchar(20) | Not null | pincode |

LANGUAGE

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Language\_id | Int(11) | Primary key | Id of language |
| language | Varchar(20) | Not null | language |

GENRE

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Genrename | Varchar(20) | Not null | genrename |
| Genre\_id | Int(11) | Primary key | Genre id |

BOOK

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Book\_id | Int(11) | Primary key | Id of book |
| Language\_id | Int(11) | Primary key | Id of language |
| Genre\_id | Int(11) | Primary key | Id of genre |
| Bname | Varchar(20) | Not null | Book name |
| Author | Varchar(20) | Not null | Author name |
| Description | Varchar(20) | Not null | Description of book |
| Status | Varchar(20) | Not null | status |

PURCHASE MASTER

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Pmaster\_id | Int(11) | Primary key | Id of purchase master |
| Publisher\_id | Int(11) | Primary key | Id of publisher |
| Staff\_id | Int(11) | Primary key | Id of staff |
| Purdate | Varchar(20) | Not null | Purchase date |
| Total amount | Varchar(20) | Not null | Total amount |

PURCHASE CHILD

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Pchild\_id | Int(11) | Primary key | Id of pchild |
| Pmaster\_id | Int(11) | Primary key | Id of pmaster |
| Book\_id | Int(11) | Primary key | Id of book |
| Amount | Varchar(20) | Not null | Amount of book |
| Quantity | Varchar(20) | Not null | Quantity of book |

RENT MASTER

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **constrains** | **descriptions** |
| Rmaster id | Int(11) | Primary key | Id of rmaster |
| Customer\_id | Int(11) | Primary key | Id of customer |
| Rdate | Varchar(20) | Not null | Return date |
| Date | Varchar(20) | Not null | Date |
| Status | Varchar(20) | Not null | Status of book |

RENT CHILD

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes** | **Data type** | **Constrains** | **descriptions** |
| Rchild\_id | Int(11) | Primary key | Id of rchild |
| Rmaster\_id | Int(11) | Primary key | Id of rmaster |
| Book\_id | Int(11) | Primary key | Id of book |

PAYMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **Constrains** | **descriptions** |
| Payment\_id | Int(11) | Primary key | Id of payment |
| Rmaster\_id | Int(11) | Primary key | Id of rmaster |
| Amount | Varchar(20) | Not null | Amount |
| Date | Varchar(20) | Not null | Date |

FINE CATEGORY

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **Constrains** | **descriptions** |
| Fcategory\_id | Int(11) | Primary key | Id of f category |
| Category | Varchar(20) | Not null | category |

FINE

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **Data type** | **Constrains** | **descriptions** |
| Fine\_id | Int(11) | Primary key | Id of fine |
| Fcategory\_id | Int(11) | Primary key | Id of fcategory |
| Rmaster\_id | Int(11) | Primary key | Id of rmaster |
| Fineamount | Varchar(20) | Not null | Amount of fine |
| Status | Varchar(20) | Not null | status |

* 1. PROCESS DESIGN

DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the 'flow’ of the data through an information system, modeling its process aspects. Often they are preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

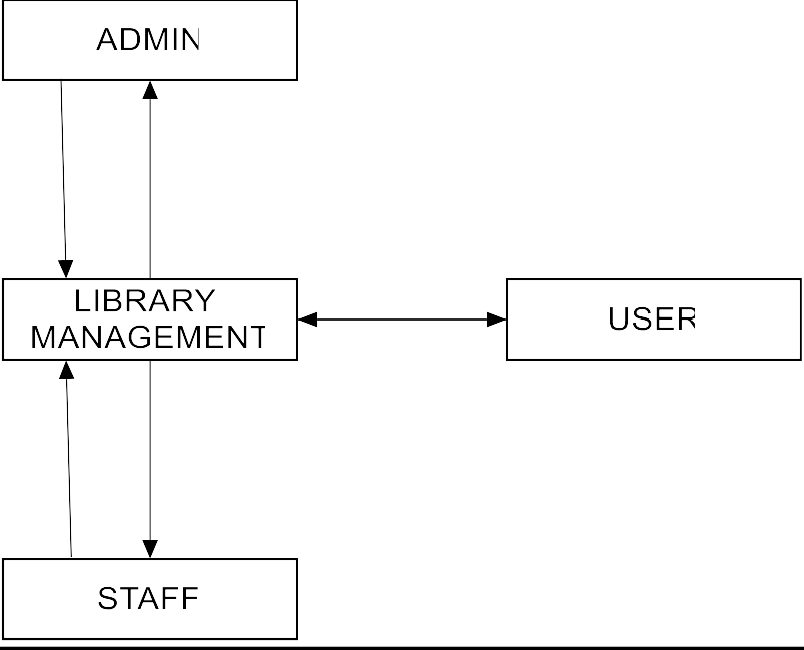
A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel. Data Flow Diagram is a logical representation of the data flow of the project. The DFD is drawn using various symbols. It has a source and destination. The process is represented using circles and source and destination using squares. The data flow is represented using circles and source and destination are represented using squares. The data flow is represented using arrows. One reader can easily get the idea about the project through Data Flow Diagram.

-Source rectangle, which defines or destination.

* Arrow, which shows dataflow.
* Circle, which represents a process that transforms incoming data into outgoing flow.
* Open rectangle, which shows a data storage.

**Data Flow Diagram:-**

**Level-0 context diagram**

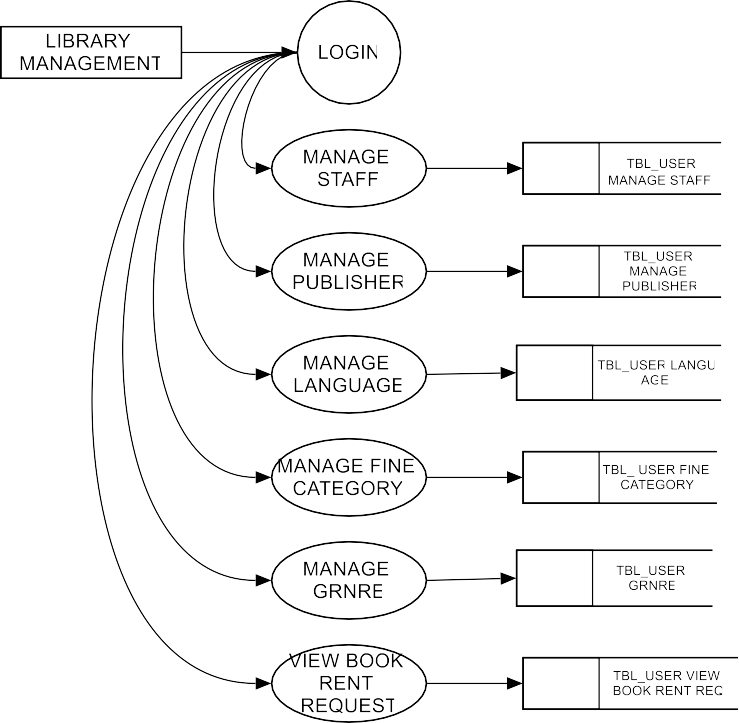


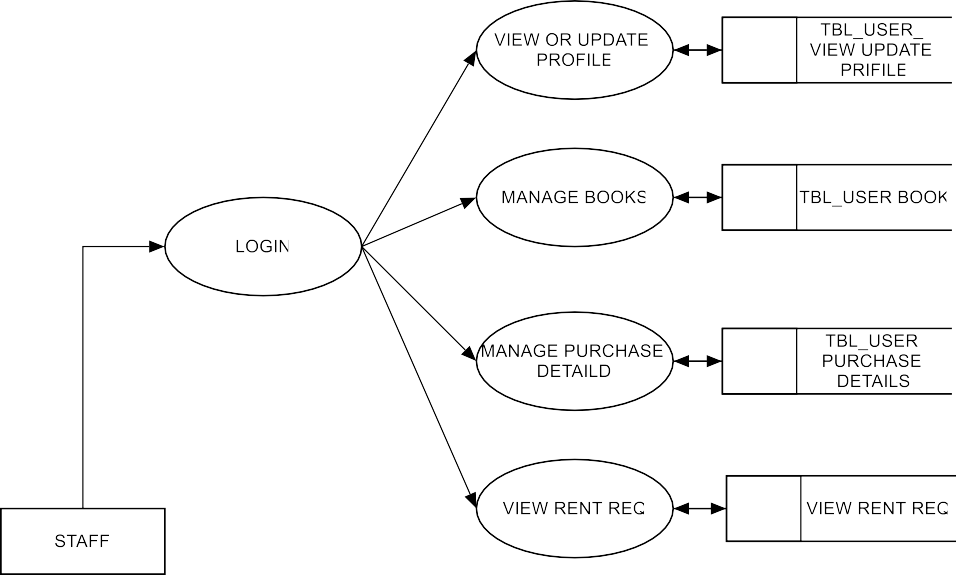
**Level-1 context diagram**

A picture containing computer, set, night sky

Description automatically generated

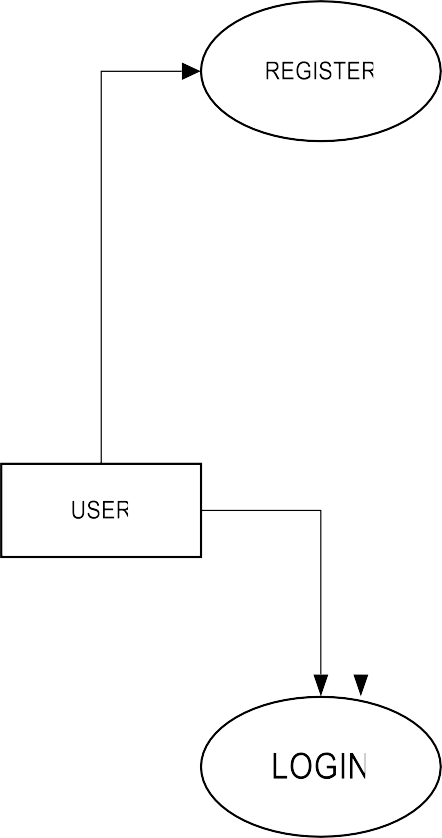
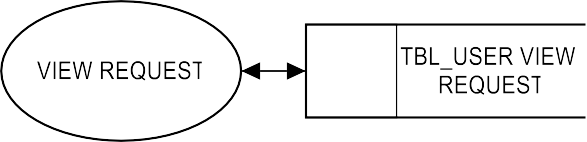
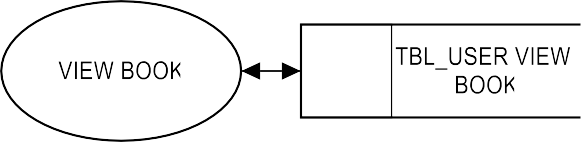
### LEVEL 2





* + **ACCEPT\REJECT**
  + **SELECT RETURN DATE**
  + **ADD FINE**
  + **VIEW PAYMENT**

**LEVEL 2**



# SYSTEM TESTING

## 7.SYSTEM TESTING

System Testing is a level of the software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the systems compliances with the specified requirements.

**System Testing** :The process of testing an integrated system to verify that it meets specified requirements.

**ANALOGY** : During the process of manufacture a ballpoint pen, the cap, the body, the tail, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready they are assembled and integration test is performed. When the complete pen is integrated, system testing is performed.

System Testing is performed after integration testing and before acceptance testing. Normally, independent Testers perform System Testing.

### 7.1ACCEPTANCE TESTING

**Acceptance Testing** is a level of the software testing where a system is tested for acceptability. The purpose of this test is to evaluate the systems compliance with the business requirements and access whether it is acceptable for delivery. ANALOGY : During the process of manufacturing a ballpoint pen, the cap, the body, the tail, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, system Testing is performed. Once System Testing is performed so as to confirm that the ballpoint pen is ready to be made available to the end-users.

METHOD: Usually , Black box Testing method is used in Acceptance Testing. Testing does not normally follow a strict procedure and is not scripted but is rather ad-hoc.Acceptance Testing is performed after System Testing and before making the system available for actual use.

Internal Acceptance Testing is performed by members of the organization that developed the software but who are not directly involved in the project(Development or Testing). Usually, it is the members of Product Management, Sales or Customer Support.

External Acceptance Testing is performed by people who are not employees of the organization that developed the software.

# FUTURE ENHANCEMENT

## FUTURE ENHANCEMENT

Nothing can exist as still end of the world. There may occur changes to all the system. This system also be subjected to similar changes to meet the ever changing tests and requirements of the users. While looking to the future, there will be many enhancements to the system, Most noticeable one will be the advanced controls that will be the advanced controls that will satisfy the user’s need.

# CONCLUSION

## 9.CONCLUSION

After we completed the project we sure the problems in the existing system are overcome. The “LIBRARY MANAGEMENT SYSTEM” process made computerized to reduce human error and to increase the efficiency. the main focus of this project is to lesson human effort, the maintenance of the records is made efficient, as all the records are stored in access database, through which data can retrieved easily. the editing is also made simpler. The user has to just type in the required field and update the desired field.

Our main aim of the project is to get the collect information about a particular person and book available in the library.The problems, which existed, have been removed to large extent. A d it is expected that this project will go a long way satisfying user’s requirements . the computerization of the library management will also reduce human stress thereby indirectly improving human resourses

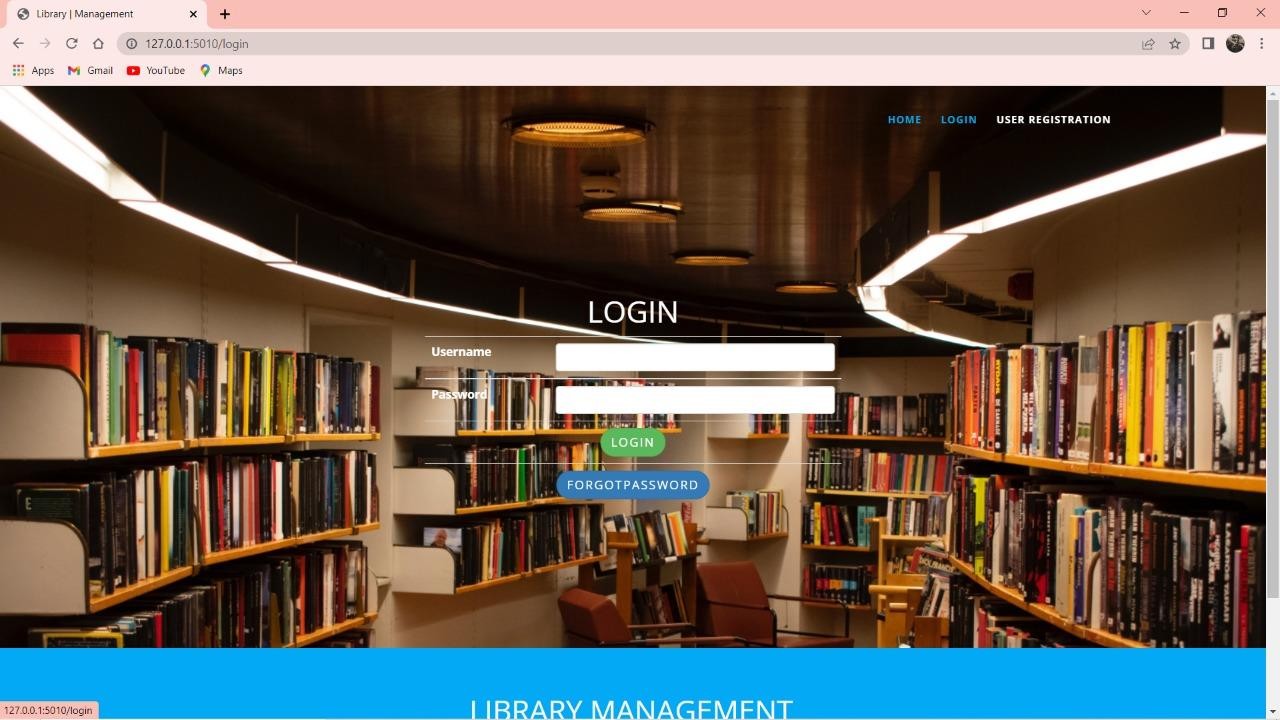
# REFERENCES

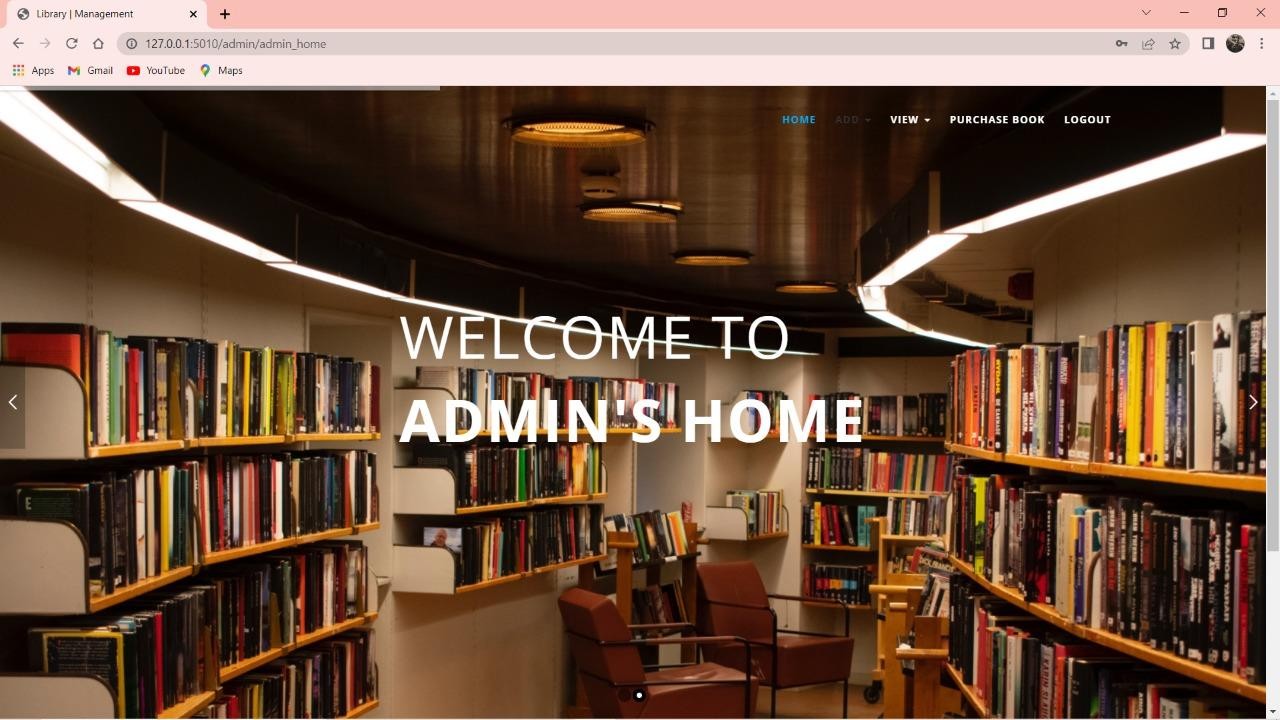
## REFERENCES

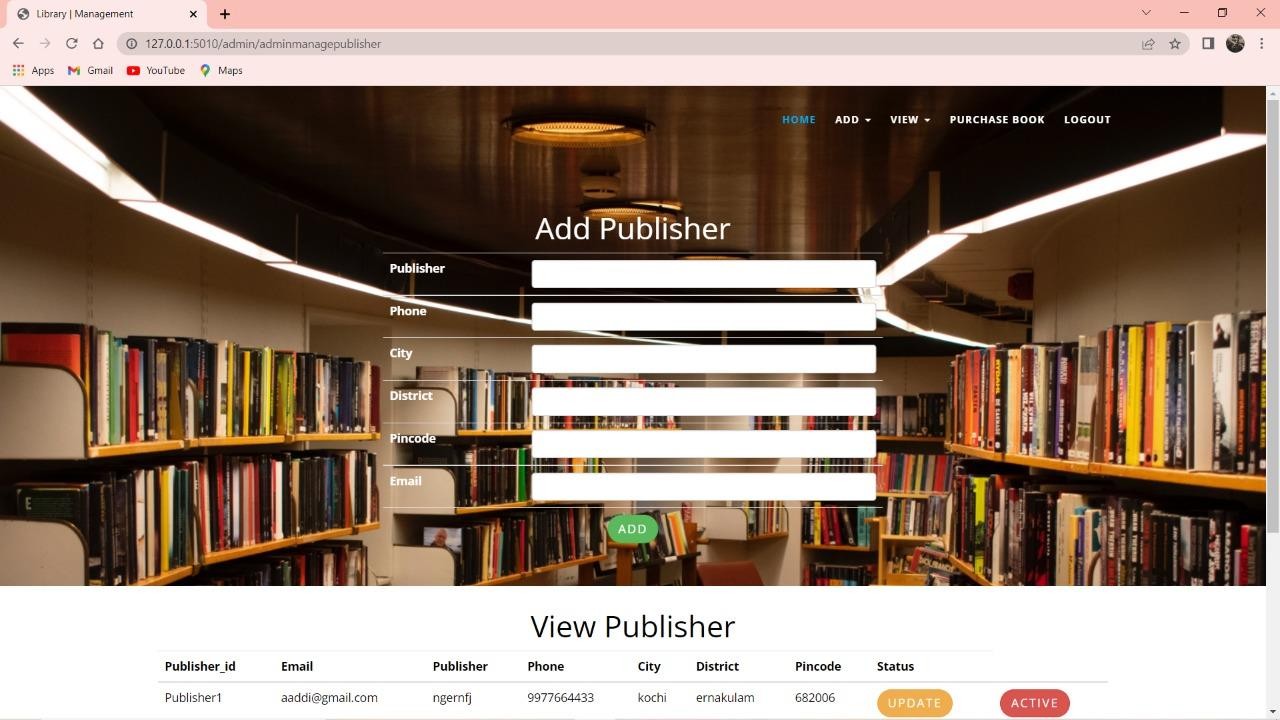
1. <http://www.w3schools.com/html/html_intro.asp>
2. <http://www.w3schools.com/css/css_background.asp>
3. <http://www.w3schools.com/js/js_datatypes.asp>
4. <http://www.w3schools.com/sql/sql_insert.asp>
5. <http://www.w3schools.com/sql/sql_update.asp>
6. <http://www.w3schools.com/php/php_forms.asp>

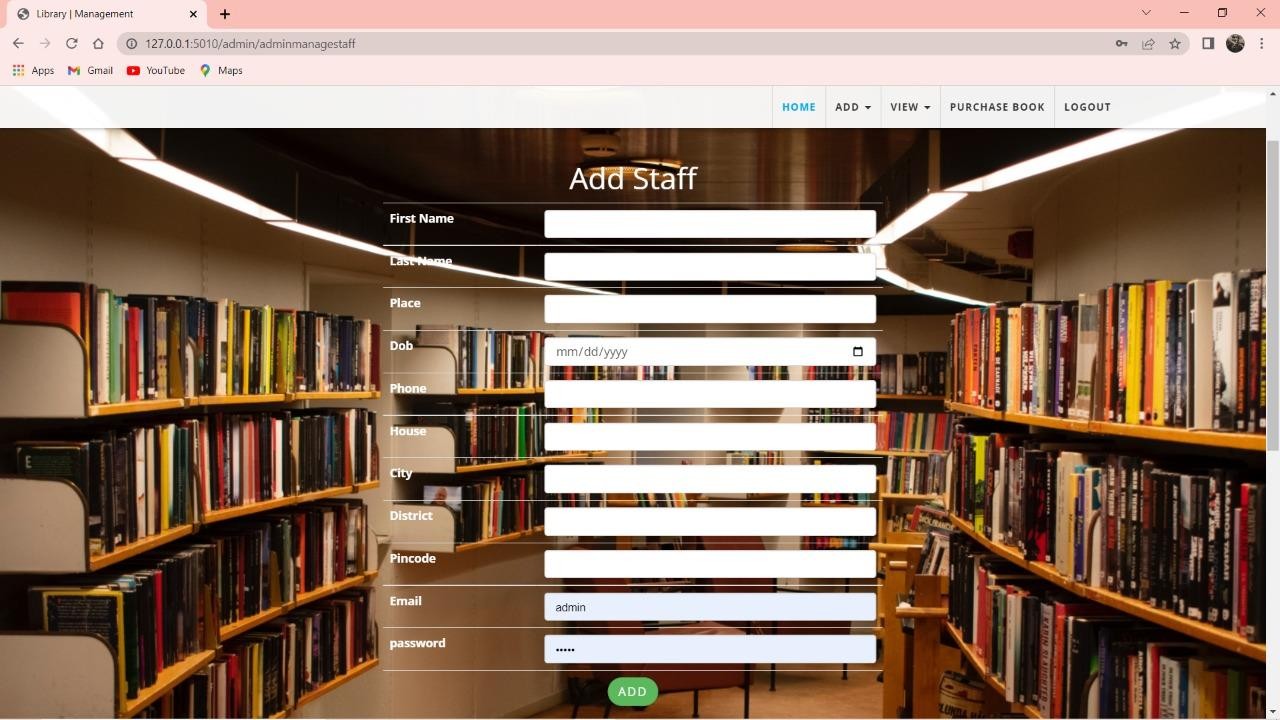
# APPENDIX

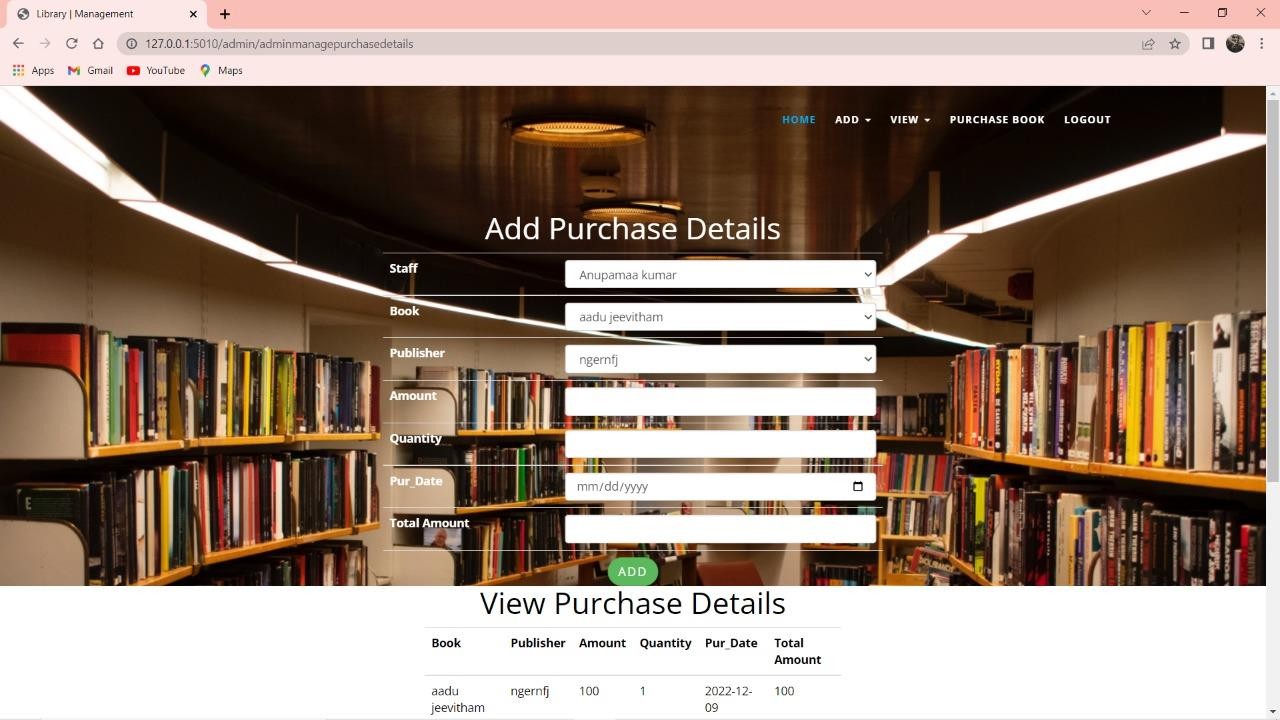
m

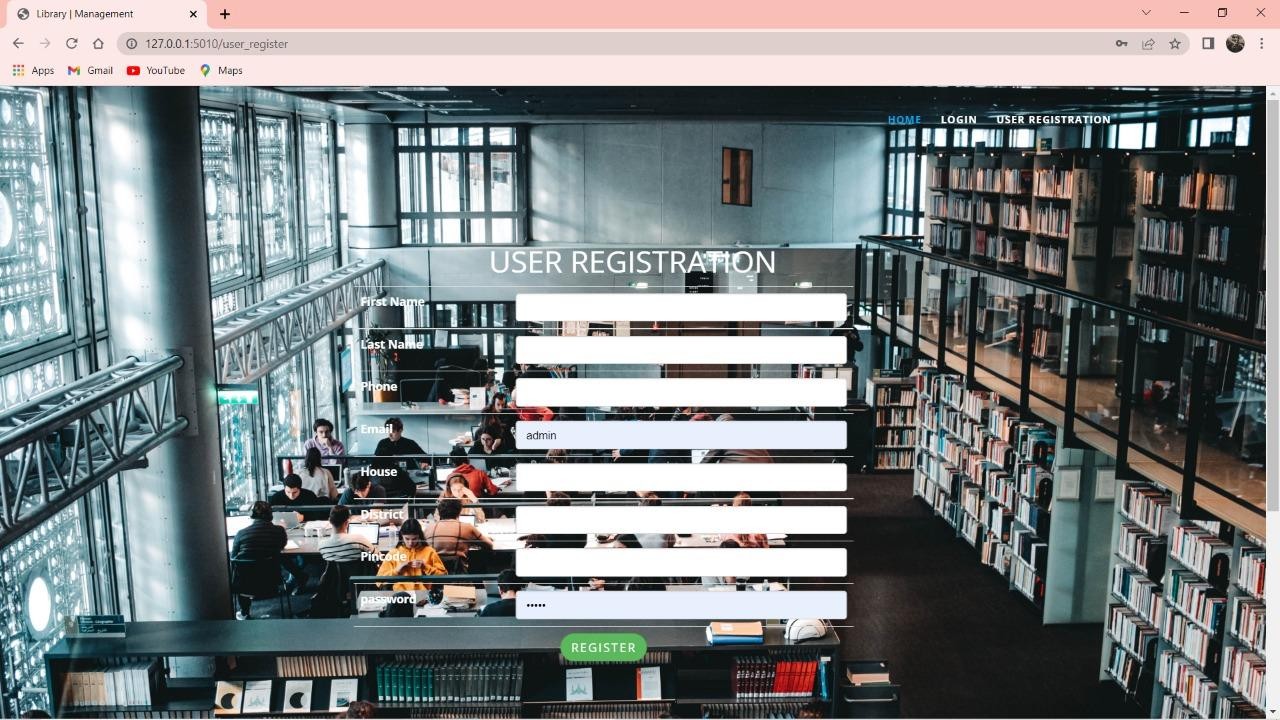


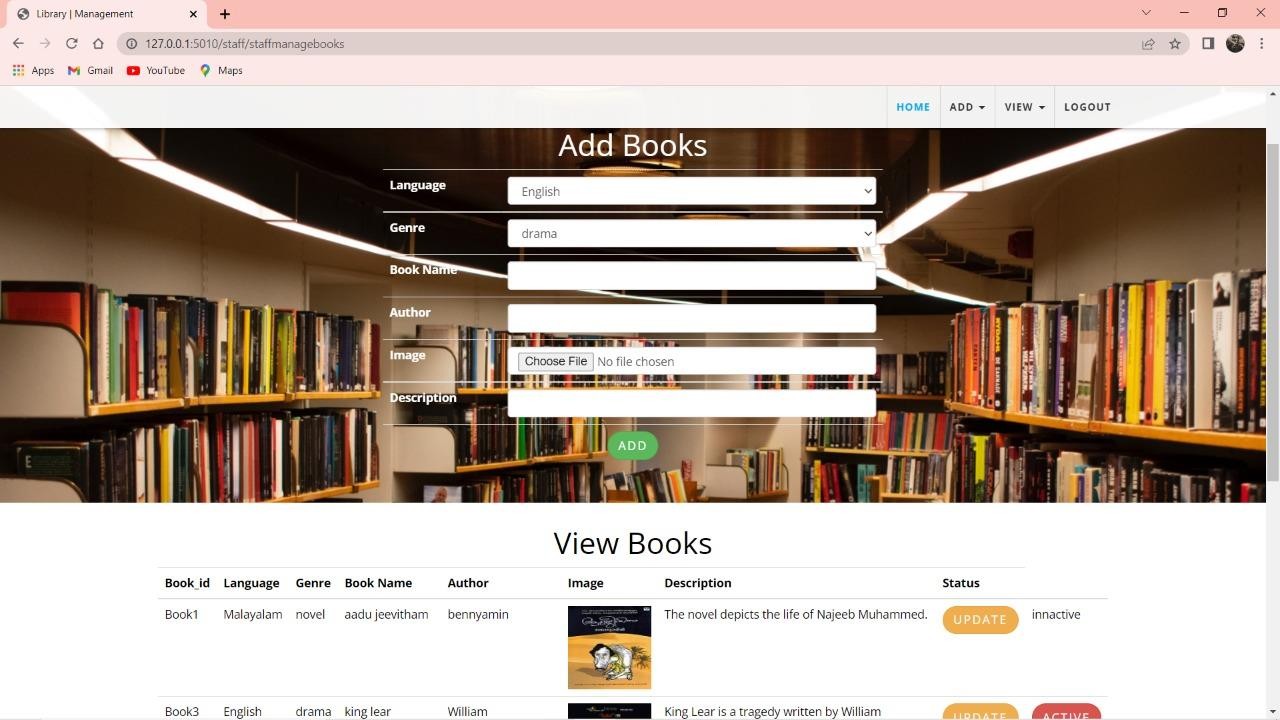


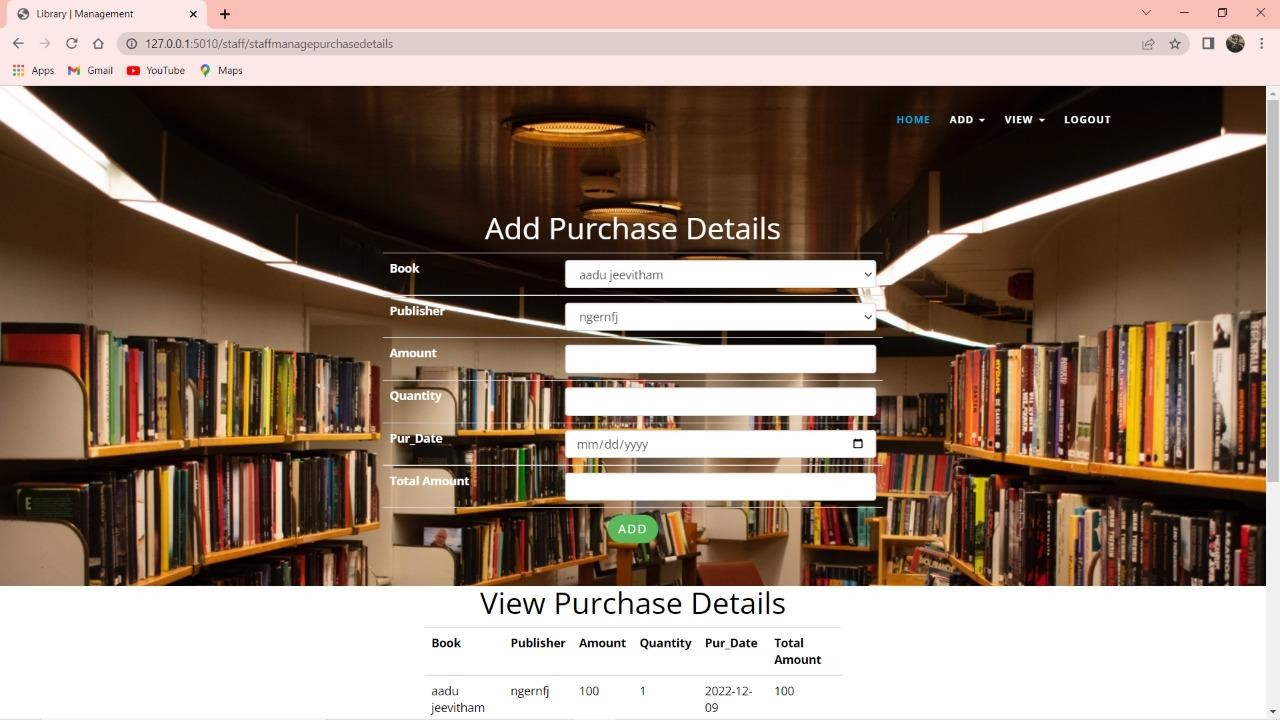


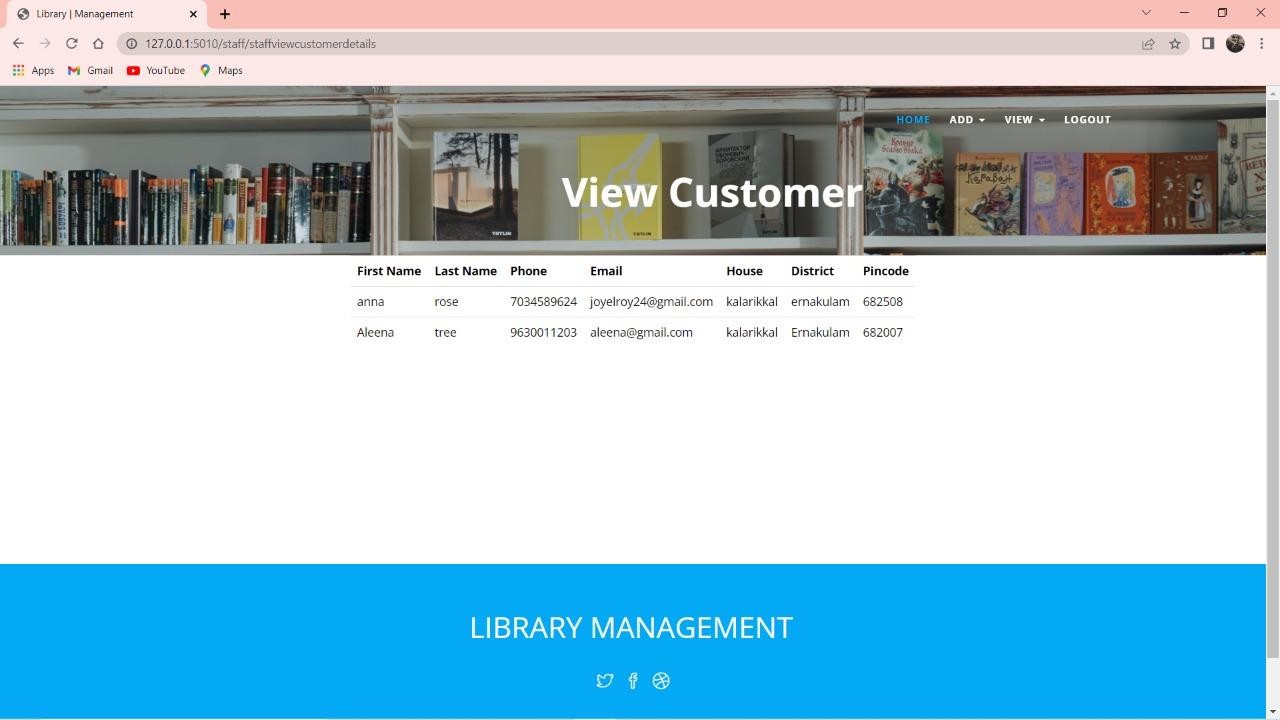


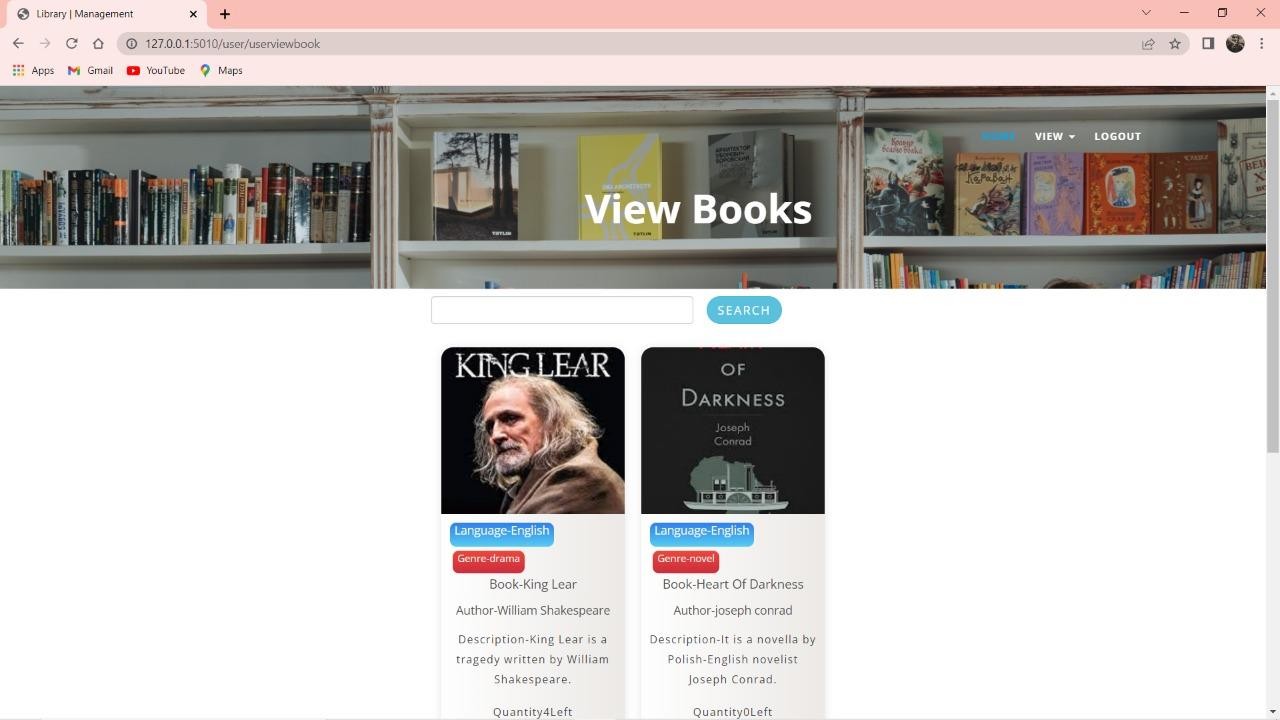


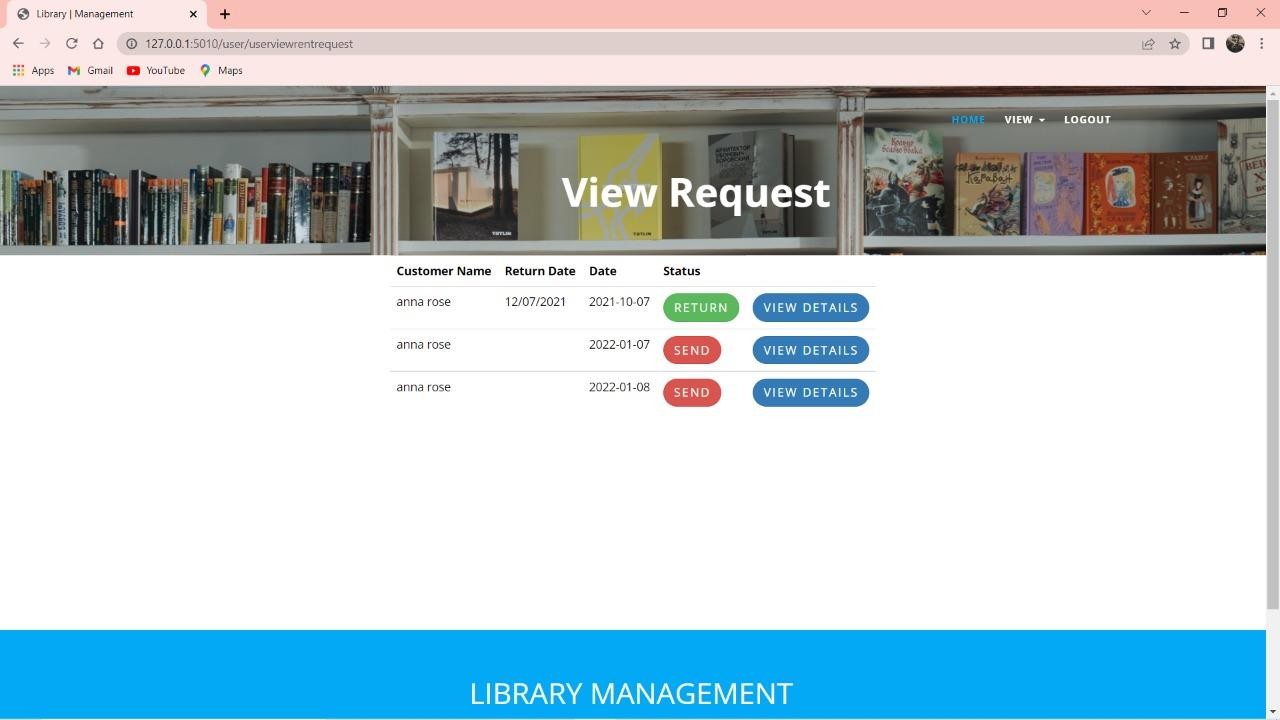












### ADMIN.PY

flask import \* database import \*

from datetime import date from

admin=Blueprint('admin',\_name\_)

@admin.route('/admin\_home') def admin\_home():

return render\_template('admin\_home.html')

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

data={} today=date.today() data['today']=today

if "add" in request.form:

fna=request.form['fn'] lna=request.form['ln'] pla=request.form['pl'] d=request.form['d'] pho=request.form['ph'] em=request.form['em'] hn=request.form['hn'] dis=request.form['dis'] c=request.form['c'] pin=request.form['pin'] pwd=request.form['p']

q="select \* from login where username='%s'"%(em) res=select(q)

if res:

flash("username is already exist")

else:

ql="insert into login values('%s','%s','staff')"%(em,pwd) rl=insert(ql)

qs="insert into staff values(null,'%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','active')"%(em,fna,lna,pla,d,pho,em,hn

,dis,c,pin)

insert(qs)

flash("added successfully")

return redirect(url\_for('admin.adminmanagestaff')) if "action" in request.args:

action=request.args['action'] sid=request.args['sid']

else:

action=None

if "update" in request.form: fna=request.form['fn'] lna=request.form['ln'] pla=request.form['pl']

d=request.form['d'] pho=request.form['ph'] em=request.form['em'] hn=request.form['hn'] dis=request.form['dis']

from c=request.form['c'] pin=request.form['pin']

q="select \* from login inner join staff using(username) where staff\_id='%s'"%(sid) print(q)

res=select(q) preuname=res[0]['username']

q="update staff set username='%s', firstname='%s',lastname='%s',place='%s',dob='%s',phone='%s',email='%s',house='%s',district='%s',c ity='%s',pincode='%s' where staff\_id='%s'"%(em,fna,lna,pla,d,pho,em,hn,dis,c,pin,sid)

r=update(q)

q="update login set username='%s' where username='%s'"%(em,preuname) r=update(q)

flash("update successfully")

return redirect(url\_for('admin.adminmanagestaff')) if action=="update":

q="select \* from staff where staff\_id='%s'"%(sid) r=select(q)

data['updatestaff']=r if action=="active":

q="update staff set ststatus='inactive' where staff\_id='%s'"%(sid) update(q)

flash("inactive successfully")

return redirect(url\_for('admin.adminmanagestaff')) if action=="inactive":

q="update staff set ststatus='active' where staff\_id='%s'"%(sid) update(q)

flash("active successfully")

return redirect(url\_for('admin.adminmanagestaff')) q="select \* from staff "

r=select(q) data['staff']=r

return render\_template('adminmanagestaff.html',data=data)

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

data={} today=date.today() data['today']=today

q="select \* from book " r=select(q) data['book']=r

q="select \* from staff " r=select(q) data['staff']=r

q="select \* from publisher where pstatus='active' "

r=select(q) data['pub']=r

q="SELECT \* FROM `purchase\_master` INNER JOIN`purchase\_child` USING(`pmaster\_id`)INNER JOIN `book` USING(`book\_id`)INNER JOIN `publisher` USING(`publisher\_id`) "

r=select(q) data['view']=r

if "add" in request.form:

sid=request.form['staffid'] bid=request.form['bid'] pid=request.form['pid'] amt=request.form['amt'] data['amt']=amt q=request.form['q'] data['q']=q d=request.form['d'] t=request.form['t']

qs="insert into purchase\_master values(null,'%s','%s','%s','%s')"%(pid,sid,d,t) id=insert(qs)

qp="insert into purchase\_child values(null,'%s','%s','%s','%s')"%(id,bid,amt,q) insert(qp)

q="update book set qty='%s' where book\_id='%s'"%(q,bid) update(q)

q="update book set status='active' where book\_id='%s'"%(bid) update(q)

flash("Purchased successfully")

return redirect(url\_for('admin.adminmanagepurchasedetails')) return render\_template('adminmanagepurchasedetails.html',data=data)

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

data={}

if "add" in request.form:

p=request.form['p'] pho=request.form['ph'] em=request.form['em'] dis=request.form['dis'] c=request.form['c'] pin=request.form['pin']

qs="insert into publisher values(null,'%s','%s','%s','%s','%s','%s','active')"%(em,p,pho,c,dis,pin)

insert(qs)

flash("added successfully")

return redirect(url\_for('admin.adminmanagepublisher')) if "action" in request.args:

else:

action=request.args['action'] pid=request.args['pid']

action=None

if "update" in request.form: p=request.form['p'] pho=request.form['ph'] em=request.form['em'] dis=request.form['dis'] c=request.form['c'] pin=request.form['pin']

q="update publisher set email='%s',publisher='%s',phone='%s',city='%s',district='%s',pincode='%s' where publisher\_id='%s'"%(em,p,pho,c,dis,pin,pid)

r=update(q)

flash("update successfully")

return redirect(url\_for('admin.adminmanagepublisher')) if action=="update":

q="select \* from publisher where publisher\_id='%s'"%(pid) r=select(q)

data['updatepublisher']=r if action=="active":

q="update publisher set pstatus='inactive' where publisher\_id='%s'"%(pid) update(q)

flash("inactive successfully")

return redirect(url\_for('admin.adminmanagepublisher')) if action=="inactive":

q="update publisher set pstatus='active' where publisher\_id='%s'"%(pid) update(q)

flash("active successfully")

return redirect(url\_for('admin.adminmanagepublisher')) q="select \* from publisher "

r=select(q) data['publisher']=r

return render\_template('adminmanagepublisher.html',data=data)

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

data={}

if "add" in request.form:

l=request.form['l']

qs="insert into language values(null,'%s','active')"%(l) insert(qs)

flash("added successfully")

return redirect(url\_for('admin.adminmanagelanguage')) if "action" in request.args:

action=request.args['action'] lid=request.args['lid']

else:

action=None

if "update" in request.form: l=request.form['l']

q="update language set language='%s' where language\_id='%s'"%(l,lid) r=update(q)

flash("update successfully")

return redirect(url\_for('admin.adminmanagelanguage')) if action=="update":

q="select \* from language where language\_id='%s'"%(lid) r=select(q)

data['updatelan']=r if action=="active":

q="update language set lstatus='inactive' where language\_id='%s'"%(lid) update(q)

flash("inactive successfully")

return redirect(url\_for('admin.adminmanagelanguage')) if action=="inactive":

q="update language set lstatus='active' where language\_id='%s'"%(lid) update(q)

flash("active successfully")

return redirect(url\_for('admin.adminmanagelanguage')) q="select \* from language "

r=select(q) data['lan']=r

return render\_template('adminmanagelanguage.html',data=data)

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

data={}

if "add" in request.form:

f=request.form['fc'] fa=request.form['fa'] m=request.form['m']

qs="insert into fine\_category values(null,'%s','%s','%s','active')"%(f,fa,m) insert(qs)

flash("added successfully")

return redirect(url\_for('admin.adminmanagefinecategory')) if "action" in request.args:

action=request.args['action'] cid=request.args['cid']

else:

action=None

if "update" in request.form: f=request.form['fc']

q="update fine\_category set category='%s' where fcategory\_id='%s'"%(f,cid)

r=update(q)

flash("update successfully")

return redirect(url\_for('admin.adminmanagefinecategory')) if action=="update":

q="select \* from fine\_category where fcategory\_id='%s'"%(cid) r=select(q)

data['updatecat']=r if action=="active":

q="update fine\_category set fcstatus='inactive' where fcategory\_id='%s'"%(cid) update(q)

flash("inactive successfully")

return redirect(url\_for('admin.adminmanagefinecategory')) if action=="inactive":

q="update fine\_category set fcstatus='active' where fcategory\_id='%s'"%(cid) update(q)

flash("active successfully")

return redirect(url\_for('admin.adminmanagefinecategory')) q="select \* from fine\_category "

r=select(q) data['cat']=r

return render\_template('adminmanagefinecategory.html',data=data)

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

data={}

if "add" in request.form:

g=request.form['g']

qs="insert into genre values(null,'%s','active')"%(g) insert(qs)

flash("added successfully")

return redirect(url\_for('admin.adminmanagegenre')) if "action" in request.args:

action=request.args['action'] gid=request.args['gid']

else:

action=None

if "update" in request.form: g=request.form['g']

q="update genre set genre\_name='%s' where genre\_id='%s'"%(g,gid) r=update(q)

flash("update successfully")

return redirect(url\_for('admin.adminmanagegenre')) if action=="update":

q="select \* from genre where genre\_id='%s'"%(gid) r=select(q)

data['updategen']=r if action=="active":

q="update genre set gstatus='inactive' where genre\_id='%s'"%(gid) update(q)

flash("inactive successfully")

return redirect(url\_for('admin.adminmanagegenre')) if action=="inactive":

q="update genre set gstatus='active' where genre\_id='%s'"%(gid) update(q)

flash("active successfully")

return redirect(url\_for('admin.adminmanagegenre')) q="select \* from genre "

r=select(q) data['gen']=r

return render\_template('adminmanagegenre.html',data=data)

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

data={}

q="SELECT \* FROM rent\_master INNER JOIN customer USING (customer\_id) " r=select(q)

data['book']=r

return render\_template('adminviewbookrentrequest.html',data=data)

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

data={} rid=request.args['rid']

q="SELECT \* FROM fine INNER JOIN fine\_category USING (fcategory\_id)INNER JOIN rent\_master USING (rmaster\_id) where rmaster\_id='%s' "%(rid)

r=select(q) data['book']=r

return render\_template('adminviewfine.html',data=data)

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

data={} rid=request.args['rid']

q="SELECT \* FROM book INNER JOIN LANGUAGE USING (language\_id) INNER JOIN

genre USING(genre\_id) INNER JOIN `rent\_child` USING(`book\_id`) where rmaster\_id='%s' and lstatus='active'and gstatus='active'"%(rid)

r=select(q) data['book']=r

return render\_template('adminviewdetails.html',data=data)

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

data={}

q="SELECT \* FROM customer" r=select(q)

data['view']=r

return render\_template('adminviewcustomerdetails.html',data=data)