

UNIVERSITY LIBRARY MANAGEMENT SYSTEM

By

KILLADA SUSHMITHA SHINY (23VV1A1220)

MRS.MADHUMITA CHANDA
Department of Information Technology
JNTU-GV CE, Vizianagaram



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJADA-VIZIANAGARAM

VIZIANAGARAM – 533003 ANDHRA
PRADESH, INDIA MARCH
2025.

Table of Contents:

DJANGO:

- Introduction
- Objectives
- Technologies used
- Python Libraries
- Django Installation
- Project & App Creation
- View's , Url's Creation
- Templates
- Form's & Model's

DESIGN THINKING:

- Definition
- Why is Design Thinking Important
- Key Principles of Design Thinking • Phases of Design Thinkng
 - Empathize – Understanding the User
 - Define – Identifying the Problem □ Ideate – Brainstorming Solutions
 - Prototype – Creating a Model
 - Test – Evaluating the Solution □ Implement (Evolve) – Deploying & Improving for using AI-Tools..
- Design Thinking Analysis
- Data Flow Diagram.
- PPT



**DEPARTMENT OF INFORMATION TECHNOLOGY JNTU-
GURAJADA VIZIANAGARAM
COLLEGE OF ENGINEERING VIZIANAGARAM (A)
VIZIANAGARAM**

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD

Email: hod. it@intugvcev.edu.in

-
1. Name of the Laboratory :
 2. Name of the Student :
 3. Roll No :
 4. Class :
 5. Academic Year :
 6. Name of Experiment :
 7. Date of Experiment :
 8. Date of Submission of Report :

S.No	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

DATE :

Signature of Faculty :

Django :A web framework for python

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Here are some key features:

1. MVT Architecture:

Django follows the **Model-View-Template (MVT)** architectural pattern, which is similar to the **ModelView-Controller (MVC)** pattern used in other frameworks.

Model: Represents the database structure and handles data-related logic.

View: Manages business logic and processes user requests.

Template: Defines the front-end UI with dynamic content rendering.

2. Built-in Admin Interface

Automatically generates an admin panel to manage application data without extra coding.

3. ORM (Object-Relational Mapping)

Django provides an **ORM (Object-Relational Mapper)** that allows developers to interact with databases using Python code instead of raw SQL queries. This makes working with databases easier and supports multiple database systems like PostgreSQL, MySQL, SQLite, and Oracle.

4. Security Features

Django comes with built-in security features, including:

CSRF Protection (Cross-Site Request Forgery)

XSS Protection (Cross-Site Scripting)

SQL Injection Prevention

Secure Authentication System

Clickjacking Protection

5. Scalability & Performance

Designed to handle high traffic with caching, database optimizations, and asynchronous capabilities.

6. URL Routing

Clean and readable URL patterns using Django's `urls.py` instead of relying on complex server configurations.

7. Templating Engine

Django's template engine allows dynamic content rendering with filters and template tags.

8. Form Handling

Django simplifies form creation, validation, and processing using built-in form classes.

Why Use Django?

- **Fast Development** – Reduces development time.
- **Scalability** – Handles high traffic efficiently.
- **Security** – Protects against common web threats.
- **Extensibility** – Supports third-party apps and plugins.

Conclusion

Django is a powerful framework that simplifies web development by offering built-in tools for database management, authentication, security, and more. It is a great choice for building modern web applications, from small projects to enterprise-level systems.

University Library Management System

Introduction:

A **University Learning Management System (ULMS)** is a web-based platform that facilitates the administration, documentation, tracking, and delivery of educational courses and training programs in a university setting. It serves as a **centralized hub** for students, faculty, and administrators to manage academic activities, access course materials, submit assignments, conduct assessments, and engage in online learning.

Purpose of ULMS

A ULMS is designed to:

- Enhance the learning experience for students.
- Provide a structured way to manage courses and materials.
- Automate administrative tasks like enrollment, grading, and reporting.
- Enable seamless communication between students and faculty.
- Offer a digital library for e-books and research materials.

Objectives:

1. User management
2. Book management
3. Borrow and return
4. Search and Catalogue
5. Reports and analytics
6. System Integration

Technologies Used:

Backend Technologies

- **Django (Python)** – Framework for handling business logic and database interactions. **Database Management**

SQLite – Lightweight option for smaller applications.

Frontend Technologies:

- **HTML5** – For structuring the web pages.
- **CSS3** – For styling and layout design.
- **JavaScript** – For adding interactivity and enhancing user experience..

Libraries:

1. Python Collections - Container Datatypes:

Purpose: Provides specialized container datatypes that support efficient handling of data.

- **Key Types:**
 - **List:** Ordered, mutable, allows duplicates.
 - **Tuple:** Ordered, immutable, allows duplicates.
 - **Set:** Unordered, no duplicates, fast membership testing.
 - **Dictionary:** Unordered, key-value pairs, fast lookups.
- **Common Use:** Data manipulation, storing and accessing collections of data in web apps (like user data or API responses).

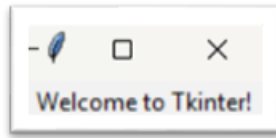
2. Tkinter - GUI Applications:

- **Purpose:** Python's standard library for creating graphical user interfaces (GUIs).
- **Key Features:**
 - **Widgets:** Buttons, labels, text boxes, etc.
 - **Event handling:** Respond to user interactions like clicks or key presses.
 - **Simple layout management.**
- **Common Use:** Build desktop applications or tools for local interaction with a web app backend.

Code:

```
Basic > New folder > Books > Tkinter > ...
1  from tkinter import Tk, Label
2  # Create a window
3  root = Tk()
4  root.title("Hello Window")
5
6  # Add a label to display text
7  Label(root, text="Welcome to Tkinter!").pack()
8
9  # Run the application
10 root.mainloop()
11
```

Output:



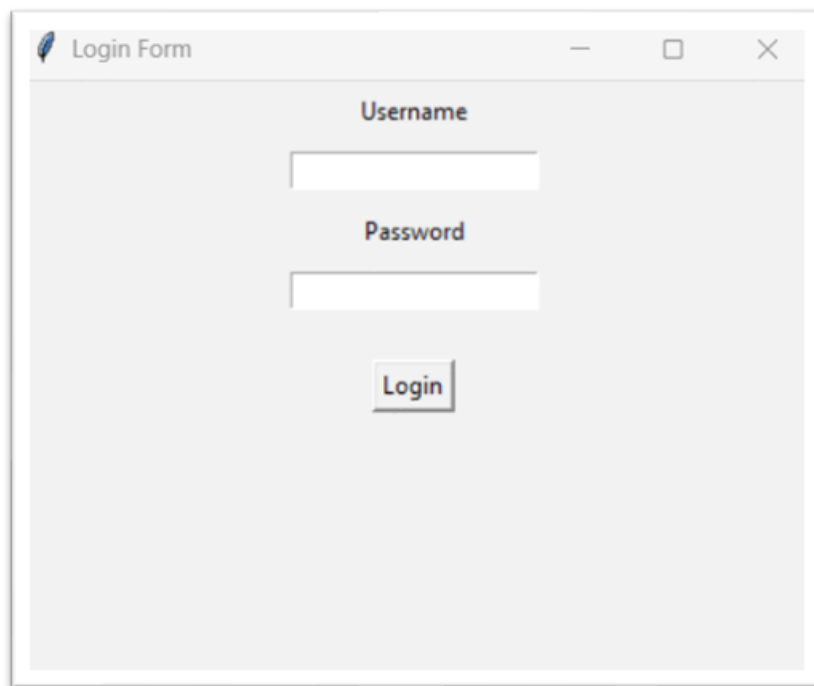
3. Requests - HTTP Requests:

- **Purpose:** Simplifies HTTP requests to interact with web APIs.
- **Key Features:**
 - Send GET, POST, PUT, DELETE requests easily.
 - Handle request parameters, headers, and cookies.
 - Simple error handling and response handling.
- **Common Use:** Interact with REST APIs, download content from the web

Code:

```
Basic > New folder > Books > request.py > ...
1  # Install tkinter (not needed for standard Python installations)
2  # pip install tk # This is not necessary for Tkinter, as it comes with Python by default
3
4  import tkinter as tk
5  from tkinter import messagebox
6
7  # Function to validate login
8  def validate_login():
9      username = username_entry.get()
10     password = password_entry.get()
11
12     # Example credentials
13     if username == "user" and password == "password":
14         messagebox.showinfo("Login Success", "Login Successful!")
15     else:
16         messagebox.showerror("Login Failed", "Invalid username or password")
17
18 # Create the main window
19 root = tk.Tk()
20 root.title("Login Form")
21 root.geometry("400x300") # Adjusted to a more reasonable size
22
23 # Create username and password labels and entry widgets
24 username_label = tk.Label(root, text="Username")
25 username_label.pack(pady=5)
26
27 username_entry = tk.Entry(root)
28 username_entry.pack(pady=5)
29
30 password_label = tk.Label(root, text="Password")
31 password_label.pack(pady=5)
32
33 password_entry = tk.Entry(root, show="*") # 'show' hides the password characters
34 password_entry.pack(pady=5)
35
36 # Create the login button
37 login_button = tk.Button(root, text="Login", command=validate_login)
38 login_button.pack(pady=20)
39
40 # Run the Tkinter event loop
41 root.mainloop()
```

Output:



4. Scrapy:

- **Purpose:** An open-source web crawling framework for large-scale web scraping.
- **Key Features:**
 - Fast, extensible, and asynchronous web scraping.
 - Supports handling requests, data extraction, and storing results.
 - Built-in handling for logging, retries, and sessions.
- **Common Use:** Web crawling and scraping projects that require high performance.

5. BeautifulSoup4 - Web Scraping:

- **Purpose:** Parses HTML and XML documents to extract data.
- **Key Features:**
 - Easy navigation and searching within HTML.
 - Supports different parsers like html.parser, lxml, and html5lib.

- **Common Use:** Extract data from websites for analysis, e.g., for building data-driven applications

Code:

```
Basic > New folder > Books > beautiful.py > ...
1  import requests
2  from bs4 import BeautifulSoup
3
4  # The URL of the website to scrape
5  url = 'https://example.com' # Replace with the actual website URL
6
7  # Set user-agent headers to avoid getting blocked
8  headers = {
9      'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36'
10 }
11
12 try:
13     # Send a GET request
14     response = requests.get(url, headers=headers, timeout=10)
15     response.raise_for_status() # Raise an error for HTTP errors (e.g., 404, 500)
16
17     # Parse the HTML content with BeautifulSoup
18     soup = BeautifulSoup(response.text, 'html.parser')
19
20     # Extract and print the page title
21     title = soup.title.string if soup.title else "No title found"
22     print(f"Title of the page: {title}\n")
23
24     # Extract and print all headings (h1, h2, h3)
25     headings = soup.find_all(['h1', 'h2', 'h3'])
26     if headings:
27         print("Headings:")
28         for heading in headings:
29             print(f" - {heading.name}: {heading.text.strip()}")
30     else:
31         print("No headings found.")
32
33     # Extract and print all links
34     links = soup.find_all('a', href=True)
35     if links:
36         print("\nLinks:")
37         for link in links:
38             print(f" - {link['href']}")
39     else:
40         print("No links found.")
41
42 except requests.exceptions.RequestException as e:
43     print(f"Error fetching the webpage: {e}")
44
```

Output:

```
✓ TERMINAL

PS C:\Users\yerra\OneDrive\Desktop\python> python "c:/Users/yerra/OneDrive/Desktop/python/Basic/New folder/Books/beautiful.py"
>>
Title of the page: Example Domain

Headings:
- h1: Example Domain

>>
Title of the page: Example Domain
```

6. Zappa:

- **Purpose:** Deploy Python web applications to AWS Lambda and API Gateway.
- **Key Features:**
 - Supports frameworks like Flask and Django for serverless deployments.
 - Manages serverless architecture and deployment configurations.
- **Common Use:** Build scalable, serverless web apps without maintaining servers.

7. Dash:

- **Purpose:** Web application framework for building interactive data visualization applications.
- **Key Features:**
 - Built on top of Flask, React, and Plotly.
 - Integrates seamlessly with data science libraries (e.g., Pandas, Plotly).
- **Common Use:** Building dashboards and data-driven web applications.

8. TurboGears:

- **Purpose:** Full-stack web framework built on top of WSGI.
- **Key Features:**
 - Modular: Mix and match components like SQLAlchemy, Genshi, and others.

- Focus on rapid development and scalability.
 - **Common Use:** Develop scalable, enterprise-level web applications.
-

9. CherryPy:

- **Purpose:** Minimalistic web framework for building web applications.
- **Key Features:**
 - Provides a simple and fast HTTP server.
 - Handles routing, cookies, sessions, and file uploads.
- **Common Use:** Building web applications with a lightweight framework.

Code:

```
Basic > New folder > Books > 📁 cherry.py > ...
1  import cherrypy
2
3  class HelloWorld:
4      @cherrypy.expose
5      def index(self):
6          return "Hello, World! This is a CherryPy web page."
7
8  if __name__ == '__main__':
9      # Server Configuration (Custom Port & Logging)
10     cherrypy.config.update({
11         'server.socket_host': '127.0.0.1', # Bind to localhost
12         'server.socket_port': 9090,       # Change port (default is 8080)
13         'log.error_file': 'cherrypy_error.log', # Log errors
14         'log.access_file': 'cherrypy_access.log'
15     })
16
17     print("Server is running on http://127.0.0.1:9090/")
18
19     try:
20         cherrypy.quickstart(HelloWorld())
21     except Exception as e:
22         print(f"Error starting CherryPy: {e}")
23
```

Output:

```

CherryPy is installed!
PS C:\Users\yerra\OneDrive\Desktop\python> & c:\Users\yerra\OneDrive\Desktop\python\.venv\Scripts\python.exe "c:\Users\yerra\OneDrive\Desktop\python\Basic\New folder\Books\cherry.py"
Server is running on http://127.0.0.1:9090/
[15/Mar/2025:22:12:16] ENGINE Listening for SIGTERM.
[15/Mar/2025:22:12:16] ENGINE Bus STARTING
CherryPy Checker:
The Application mounted at '' has an empty config.

[15/Mar/2025:22:12:16] ENGINE Set handler for console events.
[15/Mar/2025:22:12:16] ENGINE Started monitor thread 'Autoreloader'.
[15/Mar/2025:22:12:16] ENGINE Serving on http://127.0.0.1:9090
[15/Mar/2025:22:12:16] ENGINE Bus STARTED
127.0.0.1 - - [15/Mar/2025:22:12:44] "GET / HTTP/1.1" 200 42 "" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0"
127.0.0.1 - - [15/Mar/2025:22:12:44] "GET /favicon.ico HTTP/1.1" 200 1406 "http://127.0.0.1:9090/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0"

```

After run the server :-



Hello, World! This is a CherryPy web page.

10. Flask:

- **Purpose:** Lightweight micro-framework for building web applications.
- **Key Features:**
 - Simple to learn and use, but highly extensible.
 - Supports extensions for database integration, form handling, authentication, etc.
- **Common Use:** Small to medium web applications, APIs, or microservices

Code:

```

Basic > New folder > Books > images > app.py > ...
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def hello():
7      return "Hello, World!"
8
9  if __name__ == '__main__':
10     app.run(debug=True)
11

```

Output:

```
C:\Users\yerra\OneDrive\Desktop\python\Basic\New folder\Books>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 112-327-991
127.0.0.1 - - [15/Mar/2025 22:55:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Mar/2025 22:55:52] "GET /favicon.ico HTTP/1.1" 404 -
C:\Users\yerra\OneDrive\Desktop\python\Basic\New folder\Books>
```

After run the server:



Hello, World!

.

11. Web2Py:

- **Purpose:** Full-stack framework for rapid web application development.
- **Key Features:**
 - Includes a web-based IDE for development.
 - Built-in ticketing system and database integration.
- **Common Use:** Enterprise web applications with minimal setup.

12. Bottle:

- **Purpose:** Simple and lightweight WSGI micro-framework.
- **Key Features:**
 - Single-file framework, minimalistic, and fast.
 - No dependencies, supports routing, templates, and form handling.
- **Common Use:** Small web applications, APIs, and prototypes.

Code:


```

Basic > New folder > Books > bottle.py > ...
1  from bottle import Bottle, run
2
3  # Create a Bottle application instance
4  app = Bottle()
5
6  @app.route('/')
7  def index():
8      return "Hello, World! This is a Bottle web page."
9
10 # Run the Bottle application
11 if __name__ == '__main__':
12     run(app, host='localhost', port=8080, debug=True)
13

```

Output:

```

C:\Users\yerra\OneDrive\Desktop\python\Basic\New folder\Books>python bottle_app.py
Bottle v0.13.2 server starting up (using WSGIRefServer())...
Listening on http://localhost:8080/
Hit Ctrl-C to quit.

```

After run the server:



13. Falcon:

- **Purpose:** High-performance framework for building APIs.
- **Key Features:**
 - Focuses on speed and minimalism.
 - Supports RESTful API development and is optimized for large-scale deployments.
- **Common Use:** Building fast, high-performance APIs.

14. CubicWeb:

- **Purpose:** Web application framework based on an entity-relation model.
- **Key Features:**

- Uses a highly modular architecture for development.
 - Focus on building web apps with rich data models.
- **Common Use:** Semantic web applications or data-driven web apps.

15. Quixote:

- **Purpose:** A web framework designed for simplicity and scalability.
- **Key Features:**
 - Full support for Python's object-oriented programming.
 - Easily extensible, with minimalistic core.
- **Common Use:** Scalable and customizable web applications.

16. Pyramid:

- **Purpose:** Full-stack web framework that can scale from simple to complex applications.
- **Key Features:**
 - Highly flexible with support for routing, templating, authentication, and authorization.
 - Allows for small and large applications, with fine-grained control.
- **Common Use:** Building large, enterprise-grade web applications and REST APIs.

SUMMARY:

- **Flask, Django, Pyramid:** Popular web frameworks, each offering flexibility and scalability.
- **Scrapy, BeautifulSoup4:** Specialized for web scraping and data extraction.
- **Requests, Zappa, Dash:** Tools for making HTTP requests, serverless apps, and interactive data visualizations.
- **Tkinter, Bottle, CherryPy:** Libraries for building lightweight desktop and web applications.

Installation process of PYTHON_DJANGO :

Step-1 : Checking the installation & version of Python & PIP

```
python --version
```

```
pip --version
```

Step-2 : Installation of Virtual Environment `pip install`
`virtualenvwrapper-win`

Step-3 : Creation of Virtual Environment `mkvirtualenv`
`(name)`

Step-4 : Installation of Django in Virtual environment
`pip install Django`

Step-5 : Create a folder to store all the projects
`mkdir proj_folder_name`

Step-6 : start new_project `django-admin startproject`
`project_name`

```
django-admin startapp app_name
```

Step-7 : Run the server
`python manage.py runserver`

Step-8 : Open the browser and check the homepage of Django

COMMANDS :

```
C:\Users\vccl>python --version
```

```
Python 3.13.1
```

```
C:\Users\vccl>pip --version
```

```
pip 25.0.1 from
```

```
C:\Users\vccl\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\pip (python 3.13)
```

```
C:\Users\vccl>pip install virtualenvwrapper-win
```

```
Successfully installed virtualenvwrapper-win-1.2.7
```

```
C:\Users\vccl>mkvirtualenv pythondjango  created virtual  
environment CPython3.13.1
```

```
(pythondjango) C:\Users\vccl>pip install django
```

```
Successfully installed asgiref-3.8.1 django-5.1.4 sqlparse-0.5.3 tzdata-2024.2
```

```
(pythondjango) C:\Users\vccl>mkdir Django_Projects
```

```
(pythondjango) C:\Users\vccl>cd Django_Projects
```

```
(pythondjango) C:\Users\vccl\Django_Projects>django-admin startproject university
```

```
(pythondjango) C:\Users\vccl\Django_Projects>django-admin startapp university_app
```

```
(pythondjango) C:\Users\vccl\Django_Projects>cd university
```

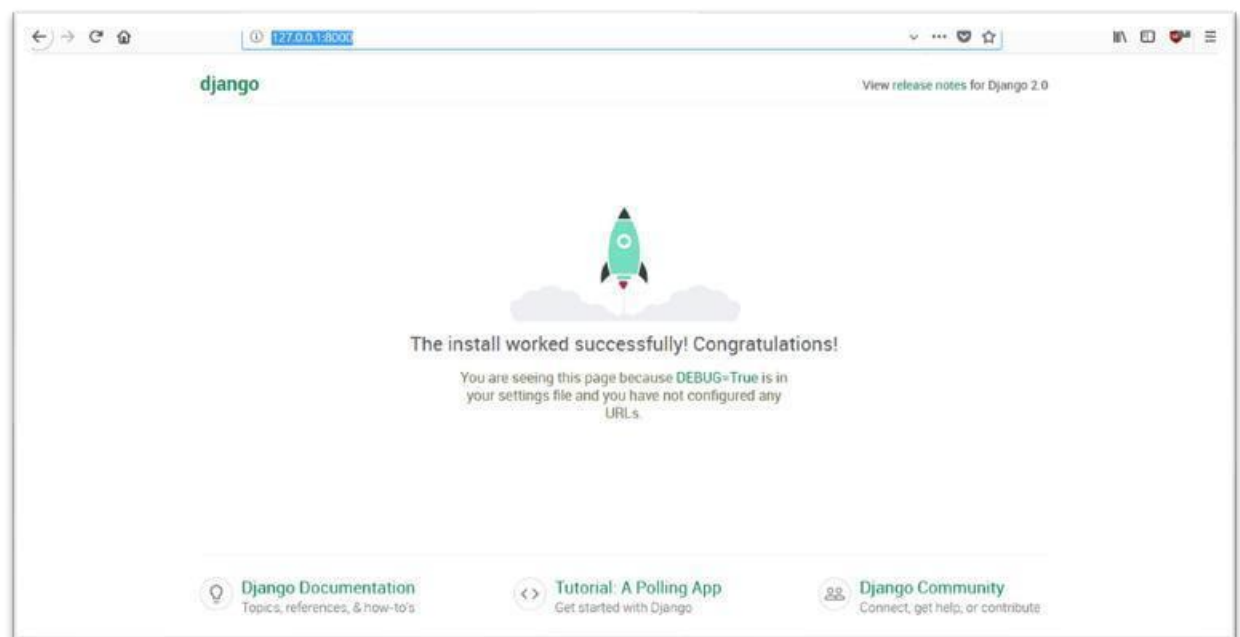
(pythondjango) C:\Users\vccl\Django_Projects\university\python manage.py runserver
Django version 5.1.4, using settings 'university.settings' Starting development server at
http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

To Come out from the ENVS - Ctrl + C

To Come back - Cd..

Output:



PROJECT CREATION

Command used : `django-admin startproject project_name`

Connecting views and urls:

Apps.py: `from django.apps import`

`AppConfig class`

`LibraryConfig(AppConfig):`

`name = 'library'`

Create a Sample View:

`from django.http import HttpResponse`

`def home(request):`

`return HttpResponse("<h1>Welcome to My Django App!</h1>")`

Run Migrations:

`python manage.py migrate`

Run the Server and Test:

`python manage.py runserver`

urls.py:

Each Django app should have its own urls.py file to define app-specific routes.

Steps to Create urls.py in a Django App

- Inside your Django app folder (myapp1), create a file named **urls.py**.
- Define URL patterns to map URLs to views.

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.index, name="index"),
    —
    # Book Management
    path("add book/", views.add_book, name="add_book"),
    path("view books/", views.view_books, name="view_books"),
    path("delete_book/<int:myid>/", views.delete_book, name="delete_book"),

    # Student Management
    path("view students/", views.view_students, name="view_students"),
    path("delete_student/<int:myid>/", views.delete_student, name="delete_student"),
    —
    # Book Issuing & Viewing
    path("issue book/", views.issue_book, name="issue_book"),
    path("view issued book/", views.view_issued_book, name="view_issued_book"),
    path("student issued books/", views.student_issued_books,
name="student issued books"),

    # Student Profile
    path("profile/", views.profile, name="profile"),
    path("edit_profile/", views.edit_profile, name="edit_profile"),

    # Authentication & Registration
    path("student_registration/", views.student_registration,
name="student_registration"),
    path("student_login/", views.student_login, name="student_login"),
    path("admin_login/", views.admin_login, name="admin_login"),
    path("change_password/", views.change_password, name="change_password"),
    path("logout/", views.Logout, name="logout"),
    ]
```

Views.py:

In Django, views.py is the file where you define functions or classes that handle requests and return responses. Views act as the logic layer of a Django web application, controlling how data is processed and which HTML templates are displayed.

```
from django.shortcuts import redirect, render, HttpResponseRedirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from datetime import date
from .models import *
from .forms import IssueBookForm

def index(request):
    return render(request, "index.html")

@login_required(login_url='/admin_login')
def add_book(request):
    if request.method == "POST":
        name = request.POST.get('name')
        author = request.POST.get('author')
        isbn = request.POST.get('isbn')
        category = request.POST.get('category')

        if name and author and isbn and category:
            Book.objects.create(name=name, author=author, isbn=isbn, category=category)
            return render(request, "add_book.html", {'alert': True})

    return render(request, "add_book.html")
```



```
@login_required(login_url='/admin_login')
def view_books(request):
    books = Book.objects.all()
    return render(request, "view_books.html", {'books': books})
```

```
@login_required(login_url='/admin_login')
def view_students(request):
    students = Student.objects.all()
    return render(request, "view_students.html", {'students': students})
```

```
@login_required(login_url='/admin_login')
def issue_book(request):
    form = IssueBookForm()
    if request.method == "POST":
        form = IssueBookForm(request.POST)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.save()
            return render(request, "issue_book.html", {'obj': obj, 'alert': True})

    return render(request, "issue_book.html", {'form': form})
```

```
@login_required(login_url='/admin_login')
def view_issued_book(request):
    issued_books = IssuedBook.objects.select_related('student', 'book').all()
    details = []
```

```
for issued_book in issued_books:

    days_overdue = (date.today() - issued_book.issued_date).days
    fine = max(0, (days_overdue - 14) * 5)

    details.append((
        issued_book.student.user,
        issued_book.student.user_id,
        issued_book.book.name,
        issued_book.book.isbn,
        issued_book.issued_date,
        issued_book.expiry_date,
        fine
    ))

return render(request, "view_issued_book.html", {'issuedBooks': issued_books, 'details': details})

@login_required(login_url='/student_login')
def student_issued_books(request):
    student = Student.objects.filter(user=request.user).first()
    if not student:
        return HttpResponse("Student not found.")

    issued_books = IssuedBook.objects.filter(student=student)
    li1, li2 = [], []

    for issued_book in issued_books:
        li1.append((
            request.user.id,
```

```
        request.user.get_full_name(),
        issued_book.book.name,
        issued_book.book.author
    ))
```

```
days_overdue = (date.today() - issued_book.issued_date).days
fine = max(0, (days_overdue - 14) * 5)
```

```
li2.append((
    issued_book.issued_date,
    issued_book.expiry_date,
    fine
))
```

```
return render(request, "student_issued_books.html", {'li1': li1, 'li2': li2})
```

```
@login_required(login_url='/student_login')
```

```
def profile(request):
```

```
    return render(request, "profile.html")
```

```
@login_required(login_url='/student_login')
```

```
def edit_profile(request):
```

```
    student = Student.objects.get(user=request.user)
```

```
    if request.method == "POST":
```

```
        student.user.email = request.POST.get('email', student.user.email)
```

```
        student.phone = request.POST.get('phone', student.phone)
```

```
        student.branch = request.POST.get('branch', student.branch)
```

```
        student.classroom = request.POST.get('classroom', student.classroom)
```

```
student.roll_no = request.POST.get('roll_no', student.roll_no)
```

```
student.user.save()
```

```
student.save()
```

```
return render(request, "edit_profile.html", {'alert': True})
```

```
return render(request, "edit_profile.html")
```

```
@login_required(login_url='/admin_login')
```

```
def delete_book(request, myid):
```

```
    Book.objects.filter(id=myid).delete()
```

```
    return redirect("/view_books")
```

```
@login_required(login_url='/admin_login')
```

```
def delete_student(request, myid):
```

```
    Student.objects.filter(id=myid).delete()
```

```
    return redirect("/view_students")
```

```
@login_required(login_url='/student_login')
```

```
def change_password(request):
```

```
    if request.method == "POST":
```

```
        current_password = request.POST.get('current_password')
```

```
        new_password = request.POST.get('new_password')
```

```
        user = request.user
```

```
        if user.check_password(current_password):
```

```
            user.set_password(new_password)
```

```
            user.save()
```

```

        return render(request, "change_password.html", {'alert': True})
    else:
        return render(request, "change_password.html", {'currpaswrong': True})

return render(request, "change_password.html")

def student_registration(request):
    if request.method == "POST":
        username = request.POST.get('username')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        email = request.POST.get('email')
        phone = request.POST.get('phone')
        branch = request.POST.get('branch')
        classroom = request.POST.get('classroom')
        roll_no = request.POST.get('roll_no')
        image = request.FILES.get('image')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')

        if password != confirm_password:
            return render(request, "student_registration.html", {'passnotmatch': True})

        user = User.objects.create_user(username=username, email=email, password=password,
first_name=first_name, last_name=last_name)

        Student.objects.create(user=user, phone=phone, branch=branch, classroom=classroom,
roll_no=roll_no, image=image)

        return render(request, "student_registration.html", {'alert': True})

```

```
return render(request, "student_registration.html")
```

```
def student_login(request):
```

```
    if request.method == "POST":
```

```
        username = request.POST.get('username')
```

```
        password = request.POST.get('password')
```

```
        user = authenticate(username=username, password=password)
```

```
    if user:
```

```
        login(request, user)
```

```
        return redirect("/profile") if not user.is_superuser else HttpResponseRedirect("You are not a student!!")
```

```
    return render(request, "student_login.html", {'alert': True})
```

```
return render(request, "student_login.html")
```

```
def admin_login(request):
```

```
    if request.method == "POST":
```

```
        username = request.POST.get('username')
```

```
        password = request.POST.get('password')
```

```
        user = authenticate(username=username, password=password)
```

```
    if user:
```

```
        login(request, user)
```

```
        return redirect("/add_book") if user.is_superuser else HttpResponseRedirect("You are not an admin.")
```

```
    return render(request, "admin_login.html", {'alert': True})
```

```
return render(request, "admin_login.html")
```

```
def Logout(request):
```

```
    logout(request)
```

```
    return redirect("/")
```

Templates :

Templates are the third and most important part of Django's MVT Structure. A template in Django is basically written in HTML, CSS, and Javascript in a .html file.

- Django framework efficiently handles and generates dynamic HTML web pages that are visible to the end-user. Django mainly functions

with a backend so, in order to provide a frontend and provide a layout to our website, we use templates.

- There are two methods of adding the template to our website depending on our needs.

We can use a single template directory which will be spread over the entire project.

- For each app of our project, we can create a different template directory.

Index.html:

```
{% extends 'basic.html' %}
```

```
{% load static %}
```

```
{% block title %} Library Management System {% endblock %}
```

```
{% block css %}
```

```
<style>
```

```
/* Styling the marquee */ .marquee-
container {          width: 100%;
overflow: hidden;    background-color:
#f7f7f7;    padding: 10px;    border-
radius:
10px;
}

.marquee-text {    font-size: 2rem;    font-weight: bold;
color: #ff6f61; /* Change the color of the sentence */
white-space: nowrap;    display:
inlineblock;    animation: marquee-animation 15s linear
infinite;
}

/* Define the scrolling animation */
@keyframes marquee-animation {
    0% {          transform:
translateX(100%);
    }
    100% {          transform:
translateX(-100%);
    }
}

/* Additional style to make the page layout nice */ .container-fluid {    background-
color: #fff3e0; /* Light background for better contrast */
```



```
}
```

```
.container {
```

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

```
}
```

```
</style>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<div class="p-4 bg-light">
```

```
<!-- Marquee Section for Welcome Message -->
```

```
<div class="container-fluid py-3">
```

```
<div class="marquee-container">
```

```
<p class="marquee-text">Welcome to University Library Management System</p>
```

```
</div>
```

```
<p class="col-md-8 fs-4">You can register as a student. You will be able to issue any book through the admin. After that, you need to return the book within 14 days, or else a fine of Rs 20 will be charged per day till the book is returned.</p>
```

```
<a href="/student_registration/" class="btn btn-primary btn-lg">Register as Student</a>
```

```
</div>
```

```
<br>
```

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col-lg-6">
```

```
<br><br>
```

```
<a class="btn btn-outline-primary btn-lg" style="text-align:center" href="/admin_login/">Admin</a>
```

```
</div>
```

```
<div class="col-lg-6">

    <br><br>

    <a class="btn btn-outline-primary btn-lg" href="/student_login/">Student</a>

</div>

</div>

</div>

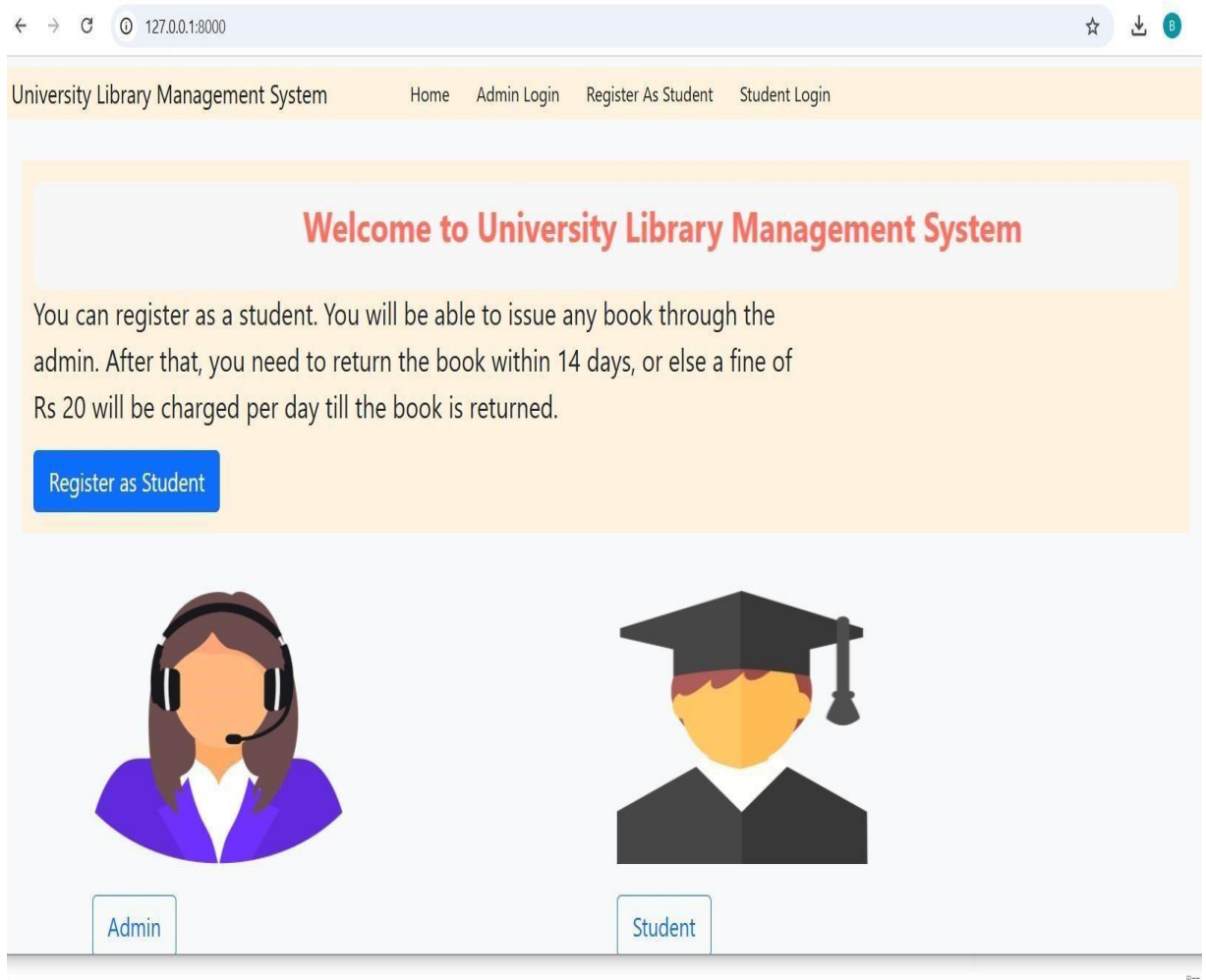
</div>

{% endblock %}
```

Output:

Description:-

*This is the homepage of a **University Library Management System (ULMS)**. It allows students to register, borrow books, and return. The page includes options for **Admin Login**, **Student Registration**, and **Student Login**, along with buttons for admin and student access.*



Admin_login.html:

```
{% extends 'basic.html' %}
```

```
{% load static %}
```

```
{% block title %} Library Management System {% endblock %}
```

```
{% block css %}
```

```
<!-- Add custom CSS styles here -->
```

```
<style>
```

```
/* Overall container styling with a blue background */
```

```
.container {    max-width: 600px;    margin: 0 auto;    padding: 30px;    background-color: #007bff; /* Blue background */    color:
```

```
white; /* White text to contrast with the blue background */    border-radius:
8px;    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
```

```
/* Title styling */  h1 {    font-size:
2.5rem;    color: white; /* White text for the
title */    font-weight: bold;    text-align:
center;    margin-bottom: 20px;    text-
decoration: underline;
}
```

```
/* Note paragraph */
p {    text-align: center;    font-size: 1.1rem;    color: #f8f9fa;
/* Lighter text color for better readability */    margin-bottom:
20px;
}
```

```
/* Form field labels */  label {    fontsize:
1rem;    font-weight: bold;    color:
#f8f9fa; /* Lighter text for labels */
}
```

```
/* Form fields styling */
.formcontrol {    border-radius: 5px;
box-shadow: none;    border: 1px solid
#ccc;    padding: 12px;    fontsize:
1rem;    transition: border-color 0.3s
ease;
```

```
}
```

```
.form-control:focus { border-color: #f8f9fa; /* White focus border for better  
contrast */ boxshadow: 0 0 8px rgba(248, 249, 250, 0.5);  
}
```

```
/* Submit button styling */ .btn-outline-primary { width:  
100%; padding: 12px; font-size: 1.2rem; fontweight:  
bold; border-radius: 5px; border-color:  
#f8f9fa; /* Lighter border for the button */ color: #f8f9fa; /*  
Light text color */ background-color: transparent;  
/* Transparent button */ transition:  
background-color 0.3s, color 0.3s;  
}
```

```
.btn-outline-primary:hover { background-color: #f8f9fa; /*  
White background on hover */ color: #007bff; /* Blue text on  
hover */  
}
```

```
/* Alert message styling */ .alert  
{ font-size:  
1rem; padding: 10px; marginbottom:  
20px; background-color:  
#f8d7da; border-color:  
#f5c6cb; color: #721c24; border-radius:  
5px;  
}
```

```
/* Spacing improvements */

.mt-4 {    margintop:
30px;
}

.mb-4 {    marginbottom:
30px;
}

</style>
{% endblock %}
{% block body %}
<div class="container">

    <br>

    <h1><u>Admin Login</u></h1><br>

    <p><b>Note:</b> Only the admin can login from here. No student will be allowed to
login.</p>

    <form method="POST">

        {% csrf_token %}

        {% if alert %}

            <div class="alert alert-danger" role="alert">

                Invalid Username or Password.

            </div>

        {% endif %}

    </form>

</div>
```

```

<div class="row mt-4">
    <div class="form-group col-md-12">
        <label><i style="font-weight: bold;">Username</i></label>
        <input type="text" class="form-control mt-2" name="username" placeholder="Enter
Username" required>
    </div>
</div>

<div class="row mt-4">
    <div class="form-group col-md-12">
        <label><i style="font-weight: bold;">Password</i></label>
        <input type="password" class="form-control mt-2" name="password"
placeholder="Enter Password" required>
    </div>
</div>

<button type="submit" class="btn btn-outline-primary mt-4">Login</button>
</form>
</div>
{% endblock %}
{% block js %}
<script>
    {% if alert %} alert("Invalid
Username or Password.") document.location
= "/admin_login"
    {% endif %}
</script>

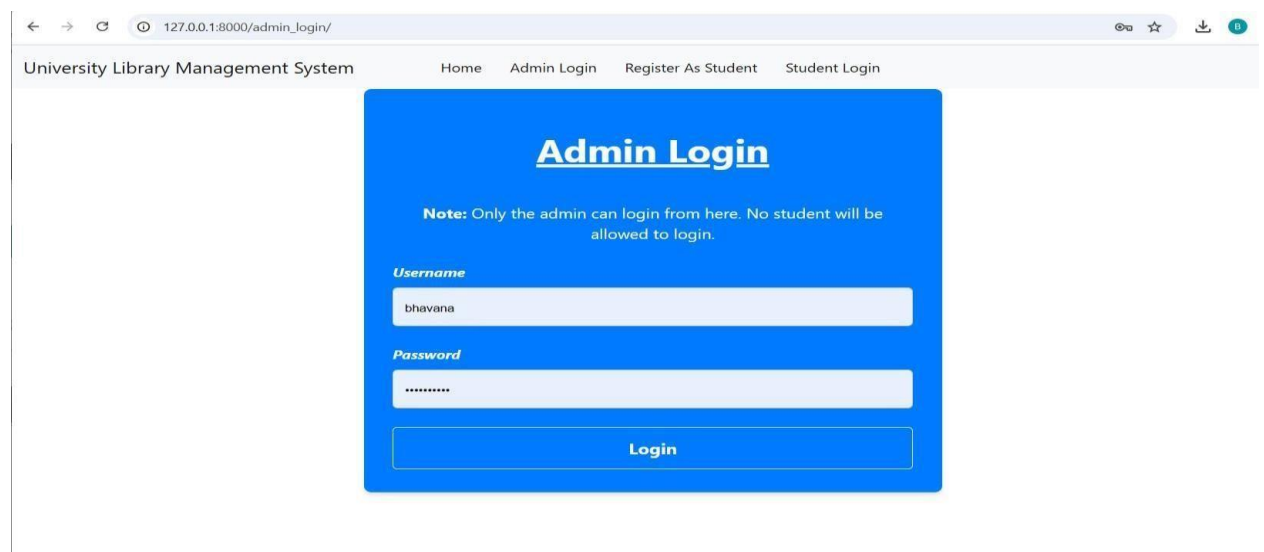
```

{% endblock %}

Output:

Description:-

This is an admin login page. It's for a university library system. Only admins can access it. Users enter their username and password. They click "Login" to proceed.



The screenshot shows a web browser window with the URL `127.0.0.1:8000/admin_login/`. The page title is "University Library Management System". The navigation bar includes links for "Home", "Admin Login", "Register As Student", and "Student Login". The main content area features a blue "Admin Login" form. The form includes a note: "Note: Only the admin can login from here. No student will be allowed to login." Below the note are two input fields: "Username" (containing "bhavana") and "Password" (containing masked characters). A "Login" button is positioned at the bottom of the form.

Add_book.html:

{% extends 'admin_navbar.html' %}

{% load static %}

{% block title %} Library Management System {% endblock %}

{% block css %}

{% endblock %}

{% block body %}

<div class="container">

<form method="POST"> {% csrf_token %}

<div class="row mt-4">

<div class="form-group col-md-12">

<label><i style="font-weight: bold;">Book Name</i></label>


```
        <input type="text" class="form-control mt-2" name="name" placeholder="Enter name of the Book" required>
```

```
    </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-12">
```

```
        <label><i style="font-weight: bold;">Author Name</i></label>
```

```
        <input type="text" class="form-control mt-2" name="author" placeholder="Enter name of the Author" required>
```

```
    </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-12">
```

```
        <label><i style="font-weight: bold;">ISBN Number</i></label>
```

```
        <input type="number" class="form-control mt-2" name="isbn" placeholder="Enter ISBN number of the book" required>
```

```
    </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-12">
```

```
        <label><i style="font-weight: bold;">Category</i></label>
```

```
        <input type="text" class="form-control mt-2" name="category" placeholder="Enter Category of the book" required>
```

```
    </div>
```

```
</div>
```

```
<button type="submit" class="btn btn-outline-primary mt-4">Add Book</button>
```

```
</div>

</form>

{% endblock %}

{% block js %}

<script>

    {% if alert %}    alert("Book is added successfully.")    document.location

=

"/view_books"

    {% endif %}

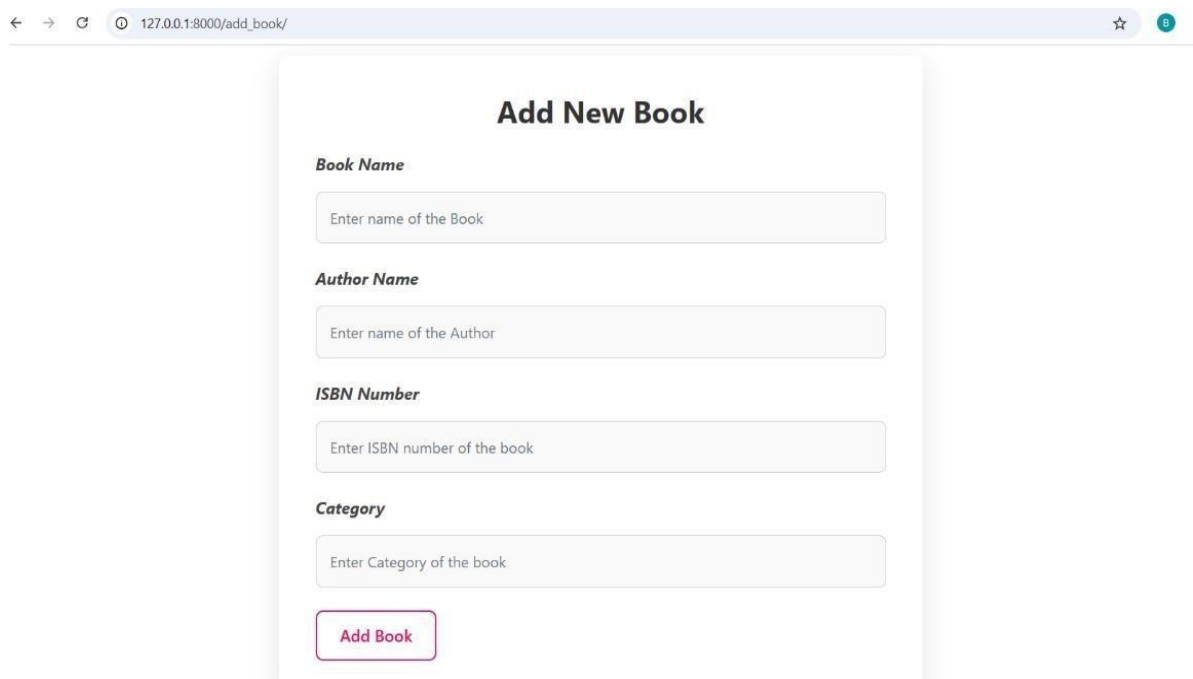
</script>

{% endblock %}
```

Output:

Description:-

This is a form to add a new book. It includes fields for Book Name, Author Name, ISBN, and Category. Users enter the book details. Clicking "Add Book" submits the information. It's likely part of a library management system.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/add_book/". The main content area features a form titled "Add New Book". The form contains four input fields, each with a label above it: "Book Name" (placeholder: "Enter name of the Book"), "Author Name" (placeholder: "Enter name of the Author"), "ISBN Number" (placeholder: "Enter ISBN number of the book"), and "Category" (placeholder: "Enter Category of the book"). Below these fields is a purple button labeled "Add Book".

View_books.html:

```
{% extends 'admin_navbar.html' %}

{% load static %}

{% block title %} Library Management System {% endblock %}

{% block css %}

{% endblock %}

{% block body %}

<div class="container mt-4">

    <h1 class="text-center"><u>All Books List</u></h1>

    <table class="table table-hover" id="example">

        <thead>

            <tr>

                <th>Sr.No</th>

                <th>Book Name</th>

                <th>Author</th>

                <th>ISBN Number</th>

                <th>Category</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            {% for book in books %}

            <tr>

                <td>{{forloop.counter}}.</td>

                <td>{{book.name}}</td>

                <td>{{book.author}}</td>
```

```
<td>{{book.isbn}}</td>

<td>{{book.category}}</td>

<td><a href="/delete_book/{{book.id}}/" class="btn btn-danger" onclick="return
confirm('Are you sure you want to delete this book?')">Delete</a></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% endblock %}
```

Output:

Description:-

This is a "View Books" page for a University Library Management System. It displays a table listing all books in the library's database. Columns show the book's serial number, name, author, ISBN, and category. Each row includes a "Delete" button for removing books. Users can export the book list to Excel, CSV, or PDF formats. A search bar allows filtering books by title or other criteria. Pagination is provided for navigating through multiple pages of books. The page indicates which entries are currently being displayed. The header shows navigation links like "View All Students" and "Issue Book."

All Books List

Copy Excel CSV PDF

Search:

Sr.No	Book Name	Author	ISBN Number	Category	Delete
1.	The C Programming Language	Brian Kernighan and Dennis Ritchie	6757458342134	Education	Delete
2.	Pride and Prejudice	Jane Austen	9342312345643	Humor	Delete
3.	To Kill a Mockingbird	Harper Lee	4657382745343	Novel	Delete
4.	Engineering Mathematics	Robert Davison	5609872343216	Education	Delete
5.	Python Programming Language	Guido van Rossum	9781906966140	Education	Delete
6.	Database Management System	Raghu Ramakrishnan	8769087651231	Education	Delete

Showing 1 to 6 of 6 entries

Previous

1

Next

Issue_book.html:

```
{% extends 'admin_navbar.html' %}
```

```
{% load static %}
```

```
{% block title %} Issue Book {% endblock %}
```

```
{% block css %}
```

```
{% endblock %}
```

```
{% block body %}
```

```
<div class="container mt-4">
```

```
  <form method="POST"> {% csrf_token %}
```

```
    {% for i in form %}
```

```
      <div class="form-group">
```

```
        <br>
```

```
        <label class="control-label col-xs-4">{{ i.label_tag }}</label>
```

```
        <div class="col-xs-8 mt-2">
```

```

        {{ i }}
    </div>
</div>
{% endfor %}

<button type="submit" class="btn btn-outline-primary mt-4">Issue Book</button>
</div>
</form>
{% endblock %}
{% block js %}
<script>
    {% if alert %} alert("Book Successfully
Issued.") document.location =
"/issue_book" {% endif %}
</script>
{% endblock %}

```

Output:

Description:-

This is the "Issue Book" page within a University Library Management System. It's used by admins to lend books to students. The page has two main input sections: "Book (Name and ISBN)" and "Student Details." The "Book" section requires the book's name and ISBN. Admins enter book and student details. They click "Issue Book" to complete the process. It's part of a library management system. The page is simple and focused on key information

← → ↻ ⓘ 127.0.0.1:8000/issue_book/ ☆ ⓘ ⬇️ ⓘ

University Library Management System View All Students Books ▾ Issue Book ▾ Logout Welcome Admin: bhavana

Book (Name and ISBN):

Book Name [ISBN]

Student Details:

Name [Branch] [Class] [Roll No]

Issue Book

View_issued_book.html:

```
{% extends 'student_navbar.html' %}

{% load static %}

{% block title %} All Students List {% endblock %}

{% block css %}

{% endblock %}

{% block body %}

<div class="container mt-4">

    <h1 class="text-center"><u>All Issued Books</u></h1>
```

```
<table class="table table-hover" id="example">
```

```
<thead>
```

```
<tr class="text-center">
```

```
<th>Sr.No</th>
```

```
<th>Student ID</th>
```

```
<th>Student Name</th>
```

```
<th>Book Name</th>
```

```
<th>Author</th>
```

```
<th>Issued Date</th>
```

```
<th>Expiry Date</th>
```

```
<th>Fine</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
{% for i in li1 %}
```

```
<tr class="text-center">
```

```
<td>{{forloop.counter}}.</td>
```

```
<td>{{i.0}}</td>
```

```
<td>{{i.1}}</td>
```

```
<td>{{i.2}}</td>
```

```
<td>{{i.3}}</td>
```

```
{% endfor %}
```

```
{% for i in li2 %}
```

```
<td>{{i.0}}</td>
```

```
<td>{{i.1}}</td>
```

```
<td>₹ {{i.2}}</td>
```



```
</tr>

{% endfor %}

</tbody>

</table>

</div>

{% endblock %}
```

Output:

Description:-

This page lists all issued books in a table format. It shows details like student name, ID, book title, ISBN, issue and expiry dates, and fine amount. Users can export the data to Excel, CSV, or PDF. A search bar helps find specific issued books. Pagination allows navigation through the list. It's likely part of a library management system. Each entry has a "Delete" option. The header shows navigation links and admin name. The page provides a clear overview of issued books. It helps manage and track book lending activities.

University Library Management System

View All Students

Books

Issue Book

Logout

Welcome Admin: bhavana

All Issued Books

Copy

Excel

CSV

PDF

Search:

Sr.No	Student	Student ID	Book Name	ISBN	Issued Date	Expiry Date	Fine	Delete
1.	abc	6	The C Programming Language	6757458342134	Aug. 13, 2021	Aug. 27, 2021	₹ 6435	Delete
2.	xyz	7	To Kill a Mockingbird	4657382745343	Aug. 13, 2021	Aug. 27, 2021	₹ 6315	Delete
3.	shiny	9	Engineering Mathematics	1	Aug. 13, 2021	Aug. 27, 2021	₹ 0	Delete

Showing 1 to 3 of 3 entries

Previous

1

Next

View_students.html:

```
{% extends 'admin_navbar.html' %}

{% load static %}

{% block title %} All Students List {% endblock %}

{% block css %}

{% endblock %}

{% block body %}

<div class="container mt-4">

    <h1 class="text-center"><u>Students List</u></h1>

    <table class="table table-hover" id="example">

        <thead>

            <tr class="text-center">

                <th>Sr.No</th>

                <th>Name</th>

                <th>ID</th>

                <th>Email</th>

                <th>Mobile Number</th>

                <th>Branch</th>

                <th>Class</th>

                <th>Roll Number</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            {% for student in students %}
```

```
|  |  |
| --- | --- |
| <td>{{forloop.counter}}.</td> | <td>{{student.user.get_full_name}}</td> |
| <td>{{student.user.id}}</td> |  |
| <td>{{student.user.email}}</td> |  |
| <td>{{student.phone}}</td> |  |
| <td>{{student.branch}}</td> |  |
| <td>{{student.classroom}}</td> |  |
| <td>{{student.roll_no}}</td> |  |
| <td><a href="/delete_student/{{student.id}}/" class="btn btn-danger" onclick="return confirm('Are you sure you want to delete this student?')">Delete</a></td> |  |
| </tr> |  |
| {% endfor %} |  |
| </tbody> |  |
| </table> |  |
| </div> |  |
| {% endblock %} |  |

```

Output:

Description:-

This page lists students in a table.

It shows student details like name, ID, and contact info.

Users can export or delete student records.

A search bar and pagination are available

University Library Management System

View All Students

Books

Issue Book

Logout

Welcome Admin: bhavana

Students List

Copy

Excel

CSV

PDF

Search:

Sr.No	Name	ID	Email	Mobile Number	Branch	Class	Roll Number	Delete
1.	priyanka majji	10	priyanka5@gmail.com	9999999999	Information Technology	Btech 2ndyear	71	Delete
2.	shiny sushmitha	12	shiny20@gmail.com	9492840083	Information Technology	B.tech 2ndyear	20	Delete
3.	sai kumar	13	sai13@gmail.com	6309125612	Computer Science Engineering	B.tech 1styear	52	Delete
4.	Tarun kumar	14	Tarun32@gmail.com	97067832451	Electrical and Electronics Engineering	B.tech 3rdyear	90	Delete
5.	Nihaal Chandhaka	15	nihaal09@gmail.com	7045123423	Civil Enginerring	B.tech 4thyear	121	Delete

Showing 1 to 5 of 5 entries

Previous

1

Next

Student_registration.html:

```
{% extends 'basic.html' %}

{% load static %}

{% block title %} Library Management System {% endblock %}

{% block css %}

{% endblock %}

{% block body %}

<div class="container">

    <form method="POST" enctype="multipart/form-data"> {% csrf_token %}

    <br>

    <h1 class="text-center"><u>Student Registration</u></h1>
```

```
<div class="row mt-4">
```

```
  <div class="form-group col-md-12">
```

```
    <label><i style="font-weight: bold;">Username</i></label>
```

```
    <input type="text" class="form-control mt-2" name="username"  
placeholder="Enter Username" required>
```

```
  </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
  <div class="form-group col-md-6">
```

```
    <label><i style="font-weight: bold;">First Name</i></label>
```

```
    <input type="text" class="form-control mt-2" name="first_name"  
placeholder="Enter First Name" required>
```

```
  </div>
```

```
  <div class="form-group col-md-6">
```

```
    <label><i style="font-weight: bold;">Last Name</i></label>
```

```
    <input type="text" class="form-control mt-2" name="last_name"  
placeholder="Enter Last Name" required>
```

```
  </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
  <div class="form-group col-md-6">
```

```
    <label><i style="font-weight: bold;">Email</i></label>
```

```
    <input type="email" class="form-control mt-2" name="email"  
placeholder="Enter Email" required>
```

</div>

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Mobile Number</i></label>

<input type="number" class="form-control mt-2" name="phone"
placeholder="Enter Mobile Number" required>

</div>

</div>

<div class="row mt-4">

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Branch Name</i></label>

<input type="text" class="form-control mt-2" name="branch"
placeholder="Enter Branch Name" required>

</div>

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Class Name</i></label>

<input type="text" class="form-control mt-2" name="classroom"
placeholder="Enter Class Name" required>

</div>

</div>

<div class="row mt-4">

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Roll Number</i></label>

<input type="text" class="form-control mt-2" name="roll_no"
placeholder="Enter Roll Number" required>

</div>

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Student Image</i></label>

<input type="file" class="form-control mt-2" name="image" required>

</div>

</div>

<div class="row mt-4">

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Password</i></label>

<input type="password" class="form-control mt-2" name="password" placeholder="Enter Password" required>

</div>

<div class="form-group col-md-6">

<label><i style="font-weight: bold;">Confirm Password</i></label>

<input type="password" class="form-control mt-2" name="confirm_password" placeholder="Confirm Password" required>

</div>

</div>

<button type="submit" class="btn btn-outline-primary mt-4">Register As Student</button>

</div>

</form>

{% endblock %}

{% block js %}

```
<script>

    {% if alert %}

    alert("Registration Successfull.")

    document.location = "/student_login"

    {% endif %}

</script>

{% endblock %}
```

Output:

Description: -

This is the "Student Registration" page for a University Library Management System. It allows students to create new accounts. The form collects essential student information like first and last name, email, and mobile number. It also captures academic details such as branch name, class name, and roll number. Users can upload a student image. A username and password are required for account creation. The password needs to be confirmed for accuracy. A "Register As Student" button submits the form data. Navigation links include "Home," "Admin Login," and "Student Login." The page facilitates student enrollment into the library system.

Student Registration

Username

bhavana

First Name

Enter First Name

Last Name

Enter Last Name

Email

Enter Email

Mobile Number

Enter Mobile Number

Branch Name

Enter Branch Name

Class Name

Enter Class Name

Roll Number

Enter Roll Number

Student Image

Choose file No file chosen

Password

Confirm Password

Confirm Password

Register As Student

Student_login.html:

```
{% extends 'basic.html' %}
```

```
{% load static %}
```

```
{% block title %} Library Management System {% endblock %}
```

```
{% block css %}
```

```
<!-- Add custom CSS styles here -->
```

```
<style>
```

```
/* Overall container styling */
```

```
.container {    max-width: 600px;    margin: 0 auto;    padding: 40px;
background: linear-gradient(to right, #007bff, #00c6ff); /* Gradient background */
color: white;    border-radius: 12px; /* Rounded corners for a modern look */
boxshadow: 0px 8px 16px rgba(0, 0, 0, 0.2); /* Soft shadow for depth */
}
```

```
/* Title styling */  h1 {
font-size: 2.8rem;    color: #ffffff;
font-weight: bold;    text-align:
center;    marginbottom: 25px;
textdecoration: underline;
}
```

```
/* Form field labels */  label {    font-size:
1rem;    font-weight: bold;    color: #f8f9fa;
/* Light color for contrast */
}
```

```
/* Form fields styling */  .form-control {    border-radius: 8px; /* Rounded
corners */    border: 1px solid #ffffff; /* White border to contrast with
background */    padding: 15px;    font-size: 1.1rem;    transition:
all 0.3s ease; /* Smooth transition for focus */
}
```

```
.form-control:focus {    border-color: #00c6ff; /* Light blue border
when focused */    box-shadow: 0 0 8px rgba(0, 198, 255,
```

```
0.6); /* Glow effect */    outline: none; /* Remove default outline
*/
}
```

```
/* Submit button styling */ .btn-outline-primary
{
    width:
100%;    padding:
15px;    font-size: 1.2rem;
fontweight: bold;    borderradius:
8px;    border-color: #ffffff;    color:
#ffffff;    background-color:
transparent;    transition: all 0.3s
ease;
}
```

```
.btn-outline-primary:hover {    background-color: #ffffff; /*
White background on hover */    color: #007bff; /* Blue text on
hover */    transform: translateY(-3px); /* Subtle lift effect on
hover */
}
```

```
/* Alert message styling */
.alert {    font-size: 1.1rem;
padding: 12px;    marginbottom:
20px;    backgroundcolor:
#f8d7da;    bordercolor:
```

```
#f5c6cb;    color: #721c24;
border-radius:

8px;    text-align: center;
}

/* Additional spacing and responsiveness */

.mt-4 {    margintop:
30px;
}

.mb-4 {    marginbottom:
30px;
}

/* Responsive Design */
@media (max-width: 767px) {
    .container {
padding: 20px;
    }
h1 {    font-size: 2rem; /* Slightly smaller title on
mobile */
    }

    .form-control {    font-size: 1rem; /*
Adjust input font size */
    }
```

```

        .btn-outline-primary {          font-size: 1rem; /*
Adjust button font size */
    }
}
</style>
{% endblock %}
{% block body %}
<div class="container">
    <br>
    <h1><u>Student Login</u></h1>
    <form method="POST">
        {% csrf_token %}
        {% if alert %}
            <div class="alert alert-danger" role="alert">
                Invalid Username or Password.
            </div>
        {% endif %}

        <div class="row mt-4">
            <div class="form-group col-md-12">
                <label><i style="font-weight: bold;">Username</i></label>
                <input type="text" class="form-control mt-2" name="username" placeholder="Enter
Username" required>
            </div>
        </div>

        <div class="row mt-4">

```

```
<div class="form-group col-md-12">

    <label><i style="font-weight: bold;">Password</i></label>

    <input type="password" class="form-control mt-2" name="password"
placeholder="Enter Password" required>

</div>

</div>

<button type="submit" class="btn btn-outline-primary mt-4">Login</button>

</form>

</div>

{% endblock %}

{% block js %}

<script>

    {% if alert %}    alert("Invalid Username or
Password.")    document.location =
"/student_login"

    {% endif %}

</script>

{% endblock %}
```

Output:

Description:

This is a student login page.

Students enter their username and password.

They click "Login" to access their account.

It's part of a university library system.

The page is simple and focused on authentication.

← → ↻ 127.0.0.1:8000/student_login/ 🔍 ☆ ⬇️ B

University Library Management System Home Admin Login Register As Student Student Login

Student Login

Username

Password

Login

Profile.html:

```
{% extends 'student_navbar.html' %}
```

```
{% block title %} Profile {% endblock %}
```

```
{% block css %} <style>
```

```
.profile{
```

```
padding: 3%; margin-top: 3%;
```

```
margin-bottom: 3%; borderradius:
```

```
0.5rem; background:
```

```
#fff;
```

```
}
```

```
.profile-img{ text-align: center;
}
```

```
.profile-img .file { position:
relative; overflow: hidden;
margin-top: -20%; width:
70%; border: none;
border-
radius: 0; font-size: 15px;
background: #212529b8;
}
```

```
</style>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<div class="container profile">
```

```
    <div class="row">
```

```
        <div class="col-md-4">
```

```
            <div class="profile-img">
```

```
                
```

```
            </div>
```

```
        </div>
```

```
        <div class="col-md-8">
```

```
            <div class="profile-tab">
```

```
                <div class="tab-pane">
```

```
                    <div class="row">
```

```
                        <div class="col-md-6">
```

```
                            <label>ID:</label>
```

```
                        </div>
```



```
<div class="col-md-6">
    <p>{{user.id}}</p>
</div>
</div>
<div class="row">
    <div class="col-md-6">
        <label>Username:</label>
    </div>
    <div class="col-md-6">
        <p>{{user}}</p>
    </div>
</div>
<div class="row">
    <div class="col-md-6">
        <label>Full Name:</label>
    </div>
    <div class="col-md-6">
        <p>{{user.get_full_name}}</p>
    </div>
</div>
<div class="row">
    <div class="col-md-6">
        <label>Email:</label>
    </div>
    <div class="col-md-6">
        <p>{{user.email}}</p>
    </div>
</div>
</div>
```

```
<div class="row">
  <div class="col-md-6">
    <label>Phone Number:</label>
  </div>
  <div class="col-md-6">
    <p>{{user.student.phone}}</p>
  </div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>Branch:</label>
  </div>
  <div class="col-md-6">
    <p>{{user.student.branch}}</p>
  </div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>Class:</label>
  </div>
  <div class="col-md-6">
    <p>{{user.student.classroom}}</p>
  </div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>Roll Number:</label>
  </div>
```




ID: 12
Username: shiny sushmitha
Full Name: shiny sushmitha
Email: shiny20@gmail.com
Phone Number: 9492840083
Branch: Information Technology
Class: B.tech 2ndyear
Roll Number: 20

[Edit Profile](#)

Edit_profile.html:

```
{% extends 'student_navbar.html' %}
```

```
{% block title %} Edit Profile {% endblock %}
```

```
{% block css %}
```

```
{% endblock %}
```

```
{% block body %}
```

```
<div class="container">
```

```
    <form method="POST"> {% csrf_token %}
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-6">
```

```
        <label><i style="font-weight: bold;">Email</i></label>
```

```
        <input type="email" class="form-control mt-2" name="email" value="{{user.email}}">
```

```
</div>
```

```
    <div class="form-group col-md-6">
```

```
        <label><i style="font-weight: bold;">Mobile Number</i></label>
```

```
        <input type="number" class="form-control mt-2" name="phone"
value="{{user.student.phone}}">
```

```
    </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-6">
```

```
        <label><i style="font-weight: bold;">Branch Name</i></label>
```

```
        <input type="text" class="form-control mt-2" name="branch"
value="{{user.student.branch}}">
```

```
    </div>
```

```
    <div class="form-group col-md-6">
```

```
        <label><i style="font-weight: bold;">Class Name</i></label>
```

```
        <input type="text" class="form-control mt-2" name="classroom"
value="{{user.student.classroom}}">
```

```
    </div>
```

```
</div>
```

```
<div class="row mt-4">
```

```
    <div class="form-group col-md-6">
```

```
        <label><i style="font-weight: bold;">Roll Number</i></label>
```

```
        <input type="text" class="form-control mt-2" name="roll_no"
value="{{user.student.roll_no}}">
```

```
    </div>
```

```
</div>
```

```
    <button type="submit" class="btn btn-outline-primary mt-5">Update Profile</button>
```

```
</form>
```

```
</div>
```

```
{% endblock %}
```

```
{% block js %}  
<script>  
    {% if alert %}    alert("Profile Updated  
Successfully.")    document.location =  
"/profile"  
    {% endif %}  
</script>  
{% endblock %}
```

Output:

Description:

This is a student profile edit page.

Students can change their contact and academic details.

They click "Update Profile" to save changes.

It's part of a library management system.

← → 🔍 127.0.0.1:8000/edit_profile/ ☆ B ⋮

University Library Management System Change Password Profile View Issued Books Logout Welcome Student: shiny sushmitha

Email	Mobile Number
<input type="text" value="shiny20@gmail.com"/>	<input type="text" value="9492840083"/>
Branch Name	Class Name
<input type="text" value="Information Technology"/>	<input type="text" value="B.tech 2ndyear"/>
Roll Number	
<input type="text" value="20"/>	

Student_issued_books.html:

```
{% extends 'student_navbar.html' %}

{% load static %}

{% block title %} All Students List {% endblock %}

{% block css %}

<style>

    /* General container styling */

.container {      maxwidth:
1000px;
margin-top: 60px;
}

    /* Table header styling */

.table thead {
backgroundcolor: #007bff;
color: white;
```

```
}
```

```
.table th, .table td {  
padding: 15px;    text-align:  
center;  
}
```

```
/* Hover effect on table rows */ .tablehover tbody  
tr:hover {    background-color:  
#f1f1f1;  
}
```

```
/* Table styling */ .table {    border-  
collapse: collapse;    width: 100%;    margin-top:  
20px;    boxshadow: 0 4px  
8px rgba(0, 0, 0, 0.1);  
}
```

```
/* Title styling */  h1 {  
font-size: 2rem;    fontweight: bold;  
color: #333;  
text-decoration: underline; }
```

```
/* Responsive design for smaller screens */  
@media (max-width: 768px) {  
    .table th, .table td {  
padding: 10px;  
}
```



```

    h1 {
        font-size: 1.5rem;
    }
}
</style>
{% endblock %}

{% block body %}
<div class="container mt-4">
    <h1 class="text-center"><u>All Issued Books</u></h1>
    <table class="table table-hover" id="example">
        <thead>
            <tr class="text-center">
                <th>Sr.No</th>
                <th>Student ID</th>
                <th>Student Name</th>
                <th>Book Name</th>
                <th>Author</th>
                <th>Issued Date</th>
                <th>Expiry Date</th>
                <th>Fine</th>
            </tr>
        </thead>
        <tbody>
            {% if li1 %}
                {% for i in li1 %}
                    <tr class="text-center">
                        <td>{{ forloop.counter }}</td>

```

```

        <td>{{ i.0 }}</td>
        <td>{{ i.1 }}</td>
        <td>{{ i.2 }}</td>
        <td>{{ i.3 }}</td>
        {% if li2 %}
            <td>{{ li2.0 }}</td>
            <td>{{ li2.1 }}</td>
            <td>₹ {{ li2.2 }}</td>
        {% else %}
            <td colspan="3">No Data Available</td>
        {% endif %}
    </tr>
{% endfor %}
{% else %}
    <tr>
        <td colspan="8" class="text-center">No records found.</td>
    </tr>
{% endif %}
</tbody>
</table>
</div>    {%
endblock %}

```

Output:

Description:

lists books issued to a student.It shows book details and issue/expiry dates.

Students can search and export the list.Pagination helps navigate through the data.

It's part of a library management system.

Copy

Excel

CSV

PDF

Search:

Sr.No	Student ID	Student Name	Book Name	Author	Issued Date	Expiry Date	Fine
1.	12	shiny sushmitha	The C Programming Language	Brian Kernighan and Dennis Ritchie	(datetime.date(2025, 3, 25), datetime.date(2025, 4, 8), 0)		₹

Showing 1 to 1 of 1 entries

Previous

1

Next

Change_password.html:

```

{% extends 'student_navbar.html' %}

{% load static %}

{% block title %} Change Password {% endblock %}

{% block css %}

<style>

/* Form container with red background */

.container {

    max-width: 600px; background-color: #f44336; /* Red

background */ padding: 30px;  border-radius: 10px;  box-shadow:

0 8px 20px rgba(0, 0, 0, 0.1);  color:

white;

}

```

```
/* Label styling */ .form-  
group label { fontweight:  
bold; color: white; font-  
size:  
1.1rem;  
}
```

```
/* Input field styling */ .formcontrol  
{ padding: 12px; fontsize: 1rem;  
border-radius: 8px; border: 1px  
solid #ccc; background-color: #fff;  
color: #333; transition: bordercolor  
0.3s ease;  
}
```

```
/* Input focus effect */  
.formcontrol:focus { border-color:  
#007bff; box-shadow: 0 0 8px rgba(38,  
143, 255, 0.3);  
}
```

```
/* Button styling */ .btnoutline-primary  
{ font-size:  
1.1rem; font-weight: bold; padding:  
12px 20px; border-  
radius: 8px; transition: all 0.3s ease;  
}
```

```
/* Button hover effect */ .btn-outlineprimary:hover
{ background-color: #007bff;
color: white; bordercolor:
#007bff;
}
```

```
/* Add space between form fields */
.form-group { marginbottom: 20px;
}
```

```
/* Additional margin for the submit button */
.btn {
margin-top: 20px;
}
```

```
/* Responsive design adjustments */
@media (max-width: 768px) {
.container { padding:
20px; margin-top:
30px;
}
}
```

```
</style>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<form class="container mt-3" method="POST" name="change_password"
onsubmit="return checkPassword();">  {% csrf_token %}

  <div class="row mt-4">

    <div class="form-group col-md-6">

      <label><i style="font-weight: bold;">Username</i></label>

      <input type="text" class="form-control mt-2" name="username"
value="{{request.user}}" readonly>

    </div>

    <div class="form-group col-md-6">

      <label><i style="font-weight: bold;">Current Password</i></label>

      <input type="password" class="form-control mt-2" name="current_password"
placeholder="Current Password">

    </div>

  </div>

  <div class="row mt-4">

    <div class="form-group col-md-12">

      <label><i style="font-weight: bold;">New Password</i></label>

      <input type="password" class="form-control mt-2" name="new_password"
placeholder="Enter the new password">

    </div>

  </div>

  <div class="row mt-4">

    <div class="form-group col-md-12">

      <label><i style="font-weight: bold;">Confirm Password</i></label>

      <input type="password" class="form-control mt-2" name="confirm_password"
placeholder="Confirm the new password">

    </div>

  </div>
```

```
</div>
```

```
<input type="submit" class="btn btn-outline-primary mt-4" value="Update Password">
```

```
</form>
```

```
{% endblock %}
```

```
{% block js %} <script>
```

```
function checkPassword() {
```

```
    if (document.change_password.new_password.value !=  
document.change_password.confirm_password.value) {
```

```
        alert("New Password and Confirm Password fields do not match each other.");
```

```
document.change_password.confirm_password.focus();
```

```
        return false;
```

```
    }
```

```
    return true;
```

```
}
```

```
{% if alert %}
```

```
    alert("Password Updated Successfully."); document.location
```

```
= "/logout";
```

```
{% endif %}
```

```
{% if currpasswrong %} alert("Current  
Password is wrong."); document.location  
= "/change_password";
```

```
{% endif %}
```

```
</script>
```

```
{% endblock %}
```

Output:

Description:

This page lets students change their password. They enter their current and new passwords. Clicking "Update Password" saves the changes. It's part of a library management system.

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/change_password/". The page title is "University Library Management System". The navigation bar includes links for "Change Password", "Profile", "View Issued Books", and "Logout", along with a welcome message "Welcome Student: shiny sushmitha". The main content area is a red box containing the following fields:

- Username:** shiny sushmitha
- Current Password:** masked with dots
- New Password:** Enter the new password
- Confirm Password:** Confirm the new password

An "Update Password" button is located at the bottom of the red box.

A University Learning Management System (ULMS) is a powerful tool that modernizes university education by providing digital course management, learning resources, and student/faculty interaction. By implementing a Django-based ULMS, universities can streamline education delivery, reduce administrative workload, and improve student learning outcomes.

Deploying Django Web Application on Cloud

What is Deployment?

Deployment is the process of making a Django web application live on the internet so users can access it. This involves hosting your app on a cloud server like AWS, Google Cloud, Digital Ocean, Heroku, or PythonAnywhere.

Features:

Scalability – Handle more users without performance issues. Security – Protect user data with SSL and secure databases. Global Accessibility – Users can access your app from anywhere.

Continuous Deployment – Easily update your app with new features.

Here's a step-by-step guide to Register on GitHub, Create a Django website with login and registration pages, and Configure Django to handle static files.

Step 1: Register on GitHub

1. Go to [GitHub](https://github.com) and click Sign up.
2. Enter your Username, Email, and Password.
3. Complete the verification and click Create Account.
4. Verify your email by clicking the link in your inbox.

Step 2: Push to GitHub

Initialize Git in your project: `git init`

2.Connect to GitHub:

`git remote add origin`

<https://github.com/bhavana071/Bhavana>

3.Add and commit changes: `git add .`

`git commit -m "Initial Commit: Login and Registration App"`

4.Push to GitHub:

`git branch -M main`

`git push -u origin main`

You have successfully built a Django website with login, registration, and static file management.

Your code is now available on GitHub.

GITHUB LINK:

<https://github.com/bhavana071/Bhavana>



**DEPARTMENT OF INFORMATION TECHNOLOGY JNTU-
GURAJADA VIZIANAGARAM
COLLEGE OF ENGINEERING VIZIANAGARAM (A)
VIZIANAGARAM**

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD

Email: hod. it@intugvcev.edu.in

-
1. Name of the Laboratory :
 2. Name of the Student :
 3. Roll No :
 4. Class :
 5. Academic Year :
 6. Name of Experiment :
 7. Date of Experiment :
 8. Date of Submission of Report :

S.No	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

DATE :

Signature of Faculty :

DESIGN THINKING RECORD

Definition: -

Design Thinking is a **creative, user-focused problem-solving method** that emphasizes **understanding people's needs, brainstorming ideas, prototyping solutions, and testing them iteratively**. It helps in developing innovative and practical solutions by blending logic, intuition, and empathy. Design Thinking is a human-centered approach to problem-solving that focuses on understanding users' needs, redefining problems, and creating innovative solutions. It is widely used in product design, software development, business strategy, and user experience (UX) design.

Why is Design Thinking Important?

- Encourages innovation by focusing on user needs.
- Reduces risk by testing ideas before full implementation.
- Helps create efficient, user-friendly, and scalable solutions.
- Promotes an iterative process to refine solutions based on feedback.

Key Principles of Design Thinking:

- **User-Centric** – The process revolves around the user's needs.
- **Iterative & Non-Linear** – Phases can be revisited multiple times.
- **Collaborative** – Encourages teamwork across different disciplines.
- **Experimental** – Encourages prototyping and testing new ideas.

Phases of Design Thinking:

Design Thinking typically consists of five or six phases, depending on the approach. The most comprehensive model follows six phases:

1. Empathize – Understanding the User

2. Define – Identifying the Problem
3. Ideate – Brainstorming Solutions
4. Prototype – Creating a Model
5. Test – Evaluating the Solution
6. Implement (Evolve) – Deploying & Improving

1. Empathize – Understanding the User

Definition: Empathy is the ability to understand and share the feelings, experiences, and perspectives of others. In Design Thinking, empathy helps designers gain deep insights into users' needs, behaviors, and challenges to create solutions that truly address their problems.

Objective: Identify and understand the user's needs, emotions, and challenges.

Goal: Develop deep empathy for the people you're designing for.

Methods Used:

User Interviews – Ask users about their pain points and experiences.

Surveys & Questionnaires – Collect broad insights.

Observation – Watch users interact with similar products.

Empathy Maps – Capture what users say, think, feel, and do.

Empathy AI-Tools:

Mixpanel:

Description and Overview

Mixpanel is a powerful product analytics platform that helps businesses track, analyze, and optimize user interactions within their digital products. It provides insights into user behavior, enabling data-driven decision-making to improve engagement, retention, and overall product performance.

Key Features of Mixpanel:

1. Event-Based Tracking
2. Real-Time Data Analysis
3. User Segmentation
4. Retention and Funnel Analysis
5. A/B Testing & Experimentation
6. Automated Reports & Alerts
7. Integration with Other Tools
8. User Interviews ,Surveys & Questionnaires, Observation ,Empathy Maps

Why Use Mixpanel?

- Helps **product managers, marketers, and developers** understand user behavior.
- Improves **customer retention** by identifying and fixing pain points.
- Enables **data-driven decision-making** for product growth.

Output:

Great work! Insights are one step away.

You are ready to connect your data to Mixpanel

✓ Your planned events Edit	
Edited by Sushmitha shiny Killada on Mar 30, 2025	
Event	Property
🔗 Sign up	django,python
🔗 Start course	django
🔗 Finish course	django ,2 weeks
🔗 Finish module	987654,3 weeks,45

2. Define – Identifying the Problem

Definition: In this stage, designers analyze the data collected during the Empathy phase to clearly define the problem statement. A well-defined problem ensures that the solutions are focused, relevant, and impactful.

Objective: Clearly define the problem based on user research.

Goal: Create a clear problem statement to guide solution development.

Methods Used:

Affinity Diagrams – Group common pain points from research.

Point-of-View Statements (POV) – Define the user's needs and challenges.

Customer Journey Mapping – Map the entire user experience.

Define AI-Tools:

MakeMyPersona:

Description and Overview

MakeMyPersona is a free buyer persona creation tool developed by HubSpot. It helps businesses and marketers create detailed customer personas based on research and insights, enabling them to better understand their target audience and tailor marketing strategies accordingly.

Key Features of MakeMyPersona

1. Step-by-Step Persona Creation
2. Customizable Persona Profiles
3. User-Friendly Interface
4. Downloadable Persona Reports
5. Integration with HubSpot CRM
6. Analyze User Data ,Identify User Needs

Why Use MakeMyPersona?

- Helps businesses create **well-defined customer profiles** for better marketing.
- Saves time compared to manual persona research.
- Improves content marketing, advertising, and sales targeting.

Output:

Make My Persona Overview

Color Scheme Save Download/Export

 Change Avatar Name Sushmitha shiny Job Title sales manager	Preferred Method of Communication Enter text here	Tools They Need to Do Their Job <ul style="list-style-type: none"> • Project Management
Age Under 18 years Highest Level of Education Less than a high school diploma Social Networks Industry Technology Organization Size Self-employed	Job Responsibilities management	Their Job Is Measured By team productivity
	Reports to director	Goals or Objectives Enter text here
	They Gain Information By Enter text here	Biggest Challenges <ul style="list-style-type: none"> • Communication • Problem Solving & Decision Making • Project Management & Disorganization • Collaboration & Creativity • Change Management

Add New Section +

HubSpot TOOLS

3.Ideate – Brainstorming Solutions

Definition: Ideation is the stage where designers generate multiple ideas for solving the defined problem. The goal is to explore as many creative and innovative solutions as possible.

Objective: Generate multiple creative solutions to solve the defined problem.

Goal: Explore a variety of potential solutions before selecting the best one.

Methods Used:

- Brainstorming Sessions – Generate as many ideas as possible.
- Mind Mapping – Visually connect different ideas.
- SCAMPER Technique – Modify existing ideas using Substitute, Combine, Adapt, Modify, Put to another use, Eliminate, Reverse.

Ideate AI-Tools:

SessionLab Library:

Description and Overview

The SessionLab Library is a comprehensive collection of facilitation methods, workshop activities, and training exercises designed to help facilitators, trainers, and team leaders create effective and engaging sessions. It is part of SessionLab, a platform that provides tools for planning and structuring workshops, meetings, and training sessions.

Key Features of the SessionLab Library

1. Extensive Collection of Facilitation Methods
2. Search and Filtering Options
3. User-Contributed and Expert-Reviewed Methods
4. Integration with Session Planning Tools
5. Free and Premium Access
6. Mind Mapping ,Substitute ,Combine , Adapt ,Modify ,Brainstorming, Eliminate ,Role Playing

Why Use the SessionLab Library?

- Saves time by providing ready-to-use workshop exercises.
- Improves session engagement and effectiveness with well-structured facilitation methods.

- Helps both new and experienced facilitators design better training sessions, meetings, and workshops.

MindMeister:

Description and Overview

MindMeister is a cloud-based mind mapping tool that allows users to visually organize ideas, collaborate in real-time, and streamline brainstorming sessions. It is widely used for project planning, note-taking, idea management, and team collaboration.

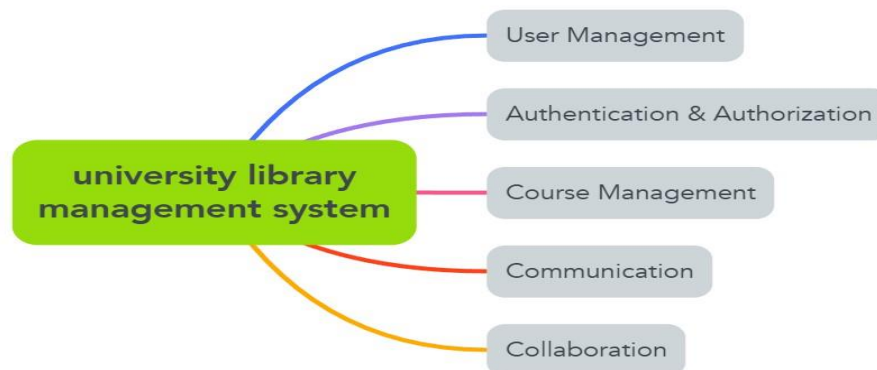
Key Features of MindMeister

1. Intuitive Mind Mapping
2. Real-Time Collaboration
3. Cloud-Based Accessibility
4. Task & Project Management
5. Presentation Mode
6. Integration with Other Tools

Why Use MindMeister?

- Enhances creativity and productivity with structured brainstorming.
- Simplifies complex concepts through visual organization.
- Supports team collaboration with real-time editing and sharing.

output



4. Prototype – Creating a Model

Definition: Prototyping involves developing low-cost, simplified models of the solution to test its functionality and usability.

Objective: Build a basic version of the solution for testing.

Goal: Create a low-fidelity or high-fidelity prototype to visualize and refine idea **Methods**

Used:

Wireframes – Sketch out the layout and functionality.

Clickable Prototypes – Use tools like Figma, Adobe XD, Sketch.

Mockups – More detailed visual representations.

Prototype AI-Tools:

Boords:

Description and Overview

Boords is a storyboarding software designed for filmmakers, animators, and creative teams to easily create, edit, and share storyboards. It simplifies the pre-production process by providing an intuitive, drag-and-drop interface, collaboration tools, and automatic animatics generation.

Key Features of Boords

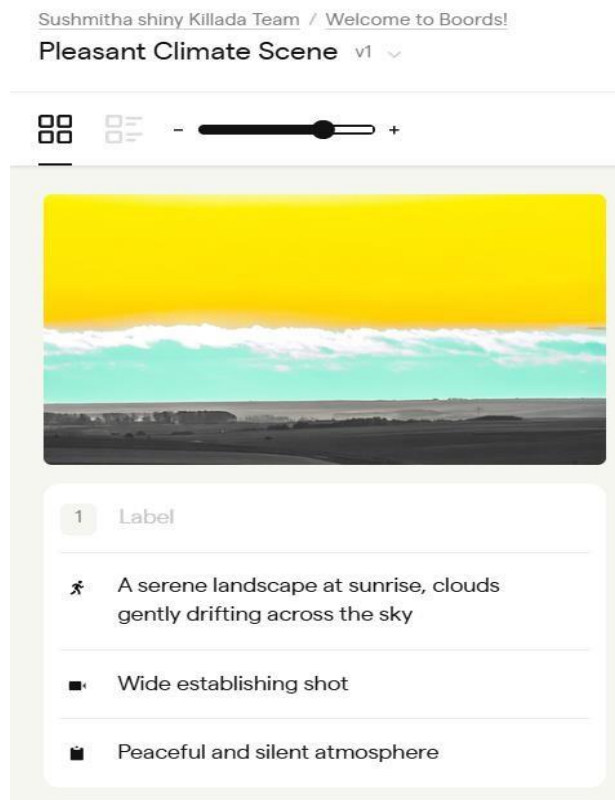
1. Easy-to-Use Storyboarding Tool

2. AI-Powered Storyboard Generator
3. Real-Time Collaboration
4. Automatic Animatics Creation
5. Script & Shot List Integration
6. Export & Sharing Options
7. Paper Prototypes ,Click-through Prototypes ,Physical Models ,Role-Playing

Why Use Boords?

- Speeds up the storyboarding and planning process.
- Enhances collaboration among directors, animators, and teams.
- Helps with visualizing ideas before full-scale production.

Output:



Figma:

Description and Overview

Figma is a cloud-based design and prototyping tool used for UI/UX design, wireframing, and collaboration. It allows designers, developers, and teams to create, test, and share digital designs in real time. Unlike traditional design tools, Figma operates entirely in the browser, making it highly accessible and ideal for remote collaboration.

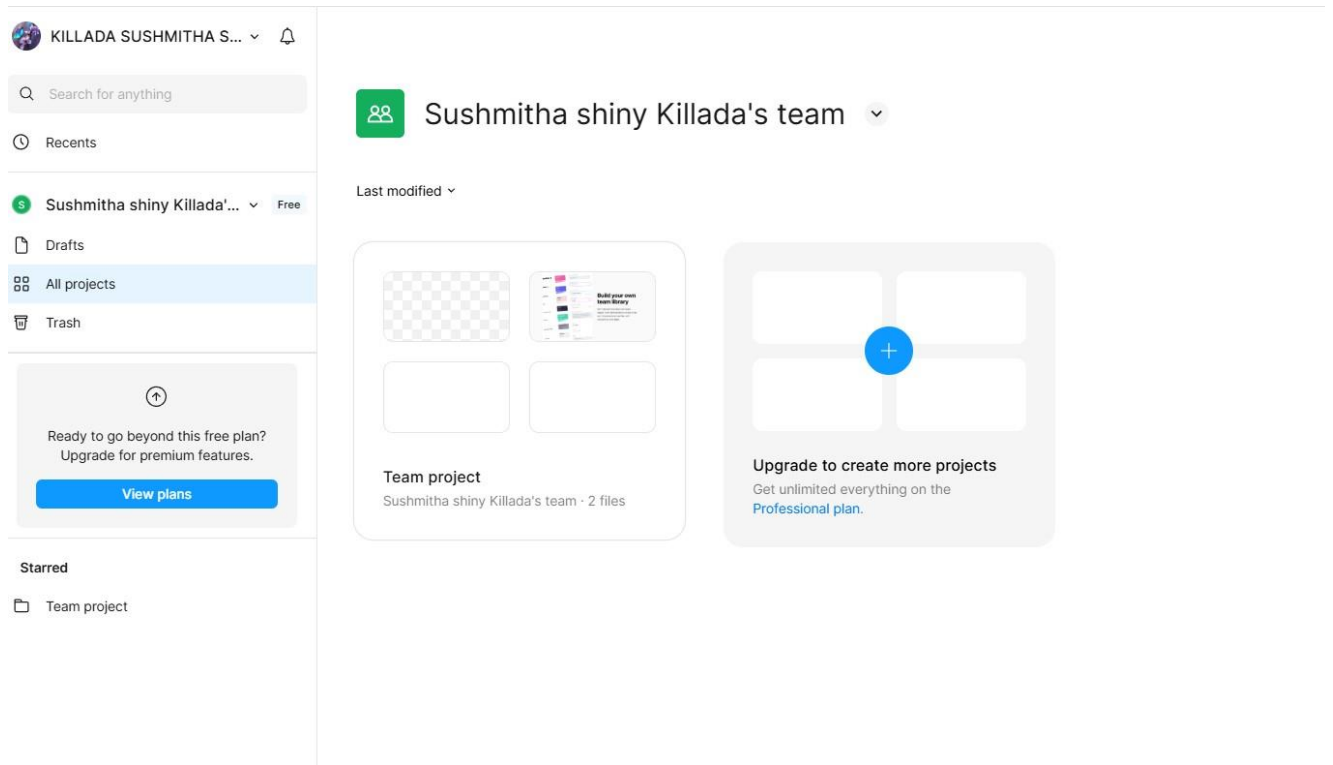
Key Features of Figma

1. Cloud-Based Design Tool
2. Real-Time Collaboration
3. Advanced UI/UX Design Capabilities
4. Prototyping & User Testing
5. Design System & Component Libraries

Why Use Figma?

- **Collaborative:** Ideal for teams working remotely or in real-time.
- **Powerful & Flexible:** Combines design, prototyping, and feedback in one tool.
- **Browser-Based:** No installation required, making it accessible anywhere.

Output



5.Test – Evaluating the Solution

Definition: Testing involves gathering feedback on the prototype and refining the solution based on real user experiences.

Objective: Gather user feedback to refine and improve the solution.

Goal: Identify areas for improvement and ensure usability.

Methods Used:

Usability Testing – Observe how real users interact with the prototype.

A/B Testing – Compare different versions of the interface.

Feedback Surveys – Collect user opinions on functionality and design.

Test AI-Tools:

Maze:

Description and Overview

Maze is a remote user testing and research platform that enables designers, product managers, and marketers to collect real-time user insights through interactive testing. It helps teams validate prototypes, conduct usability tests, and gather actionable feedback to improve digital products before development.

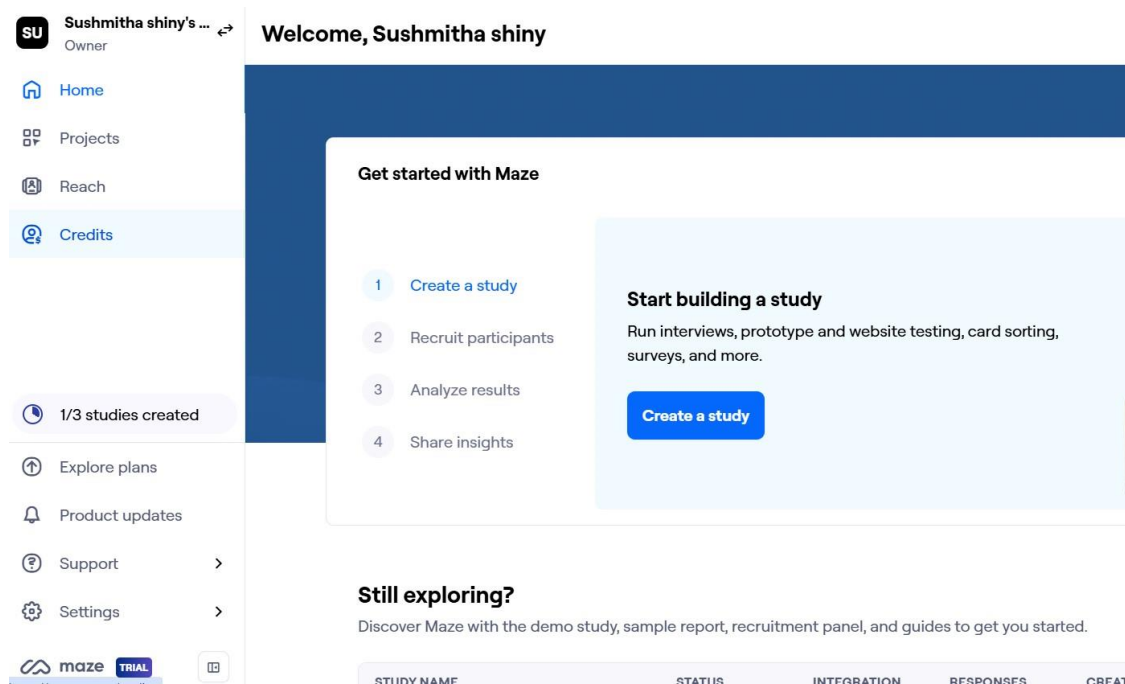
Key Features of Maze

1. Prototype Testing
2. User Surveys & Feedback Collection
3. A/B Testing & Concept Validation
4. Real-Time Analytics & Heatmaps
5. No-Code Testing Platform
6. Integration with Design & Collaboration Tools
7. Automated Reports & Insights

Why Use Maze?

- Speeds up product validation by allowing quick remote testing.
- Enhances user experience by identifying usability issues early.
- Helps teams make data-driven decisions before product development.

Output:



6.Implement (Evolve) – Deploying & Improving

Definition: In the final stage, the tested solution is launched and continuously improved based on ongoing feedback and performance tracking.

Objective: Finalize the solution, launch it, and continuously improve it.

Goal: Deploy the system and iterate based on real-world usage.

Methods Used:

- Beta Testing – Test with a small group before full launch.
- Continuous Integration & Deployment (CI/CD) – Release updates smoothly.
- User Feedback Loop – Regularly collect feedback and improve.

Key features:

- Finalizing the Solution
- Launching in a Real Environment
- Monitoring Performance
- Collecting User Feedback
- Iterating & Scaling

Design Thinking Analysis for a University Library Management System (ULMS):

1. Empathy – Understanding the User

Before developing the ULMS, it was important to understand the challenges faced by students, faculty, and librarians.

- **Students** may struggle to find and borrow books efficiently.
- **Faculty** may require access to research materials and digital libraries.

Empathy Output in ULMS:

- User pain points documented (e.g., difficulty locating books, managing late returns).
- Clear understanding of expectations (e.g., fast book search, online renewal).
- Personas and usage scenarios guiding system design.

2. Define – Identifying the Problem

After gathering insights, the next step was to define the core problems ULMS would solve.

- Analyzing common library management challenges.
- Identifying essential features (e.g., book search, automated fines, digital resources).
- Creating a clear problem statement.

Define Output in ULMS:

- A problem statement outlining the purpose of ULMS.
- A list of key system features (e.g., book catalog, loan tracking, fine management).

3. Ideate – Brainstorming Solutions

With a clear problem definition, various solutions were explored for ULMS.

- Implementing an advanced book search with filters.
- Automating due date reminders and fine calculations.
- Enabling e-book access for remote learning.

Ideate Output in ULMS:

- A list of potential technical solutions.
- Wireframes or sketches of the user interface.
- Defined technology stack (Django, PostgreSQL, Redis, etc.).

4. Prototype – Creating a Model

A working version of ULMS was developed to test its functionality.

- A **minimal viable product (MVP)** was deployed locally.
- Users could log in, search for books, and issue/return them.
- Basic reports and user dashboards were implemented.

Prototype Output in ULMS:

- A functional prototype with limited features. -Django models, views, and templates set up.
- A connected and tested database.

5. Test – Evaluating the Solution

After creating the prototype, user feedback was collected and improvements were made.

- **If book search was slow**, optimized database queries.
- **If students faced login issues**, refined authentication.
- **If navigation was unclear**, adjusted UI design.

Test Output in ULMS:

- Feedback from students, faculty, and librarians.
- A refined system with bug fixes and improvements. - A checklist of final changes before full deployment.

6. Implement (Evolve) – Deploying & Improving

Once testing was complete, the ULMS was deployed and continuously improved.

- Hosted on a live domain (e.g., **library.university.edu**).
- Integrated user authentication and email notifications.
- Regular updates based on user feedback.

Implementation Output in ULMS:

- A fully functional **live** ULMS accessible to users.
- Continuous improvements based on feedback.
- Performance monitoring and security enhancements.

Final Comparison Summary

Design Thinking Phase

Application in ULMS

Empathy – Understanding the user’s needs.

User research, interviews, identifying problems.

Define – Identifying the problem.

Writing a problem statement, defining key features.

Ideate – Brainstorming solutions.

Planning book catalog, search, fine management, and digital access.

Prototype – Creating a working model.

Developing the library system with core features.

Test – Evaluating the solution.

User testing, bug fixing, and performance testing.

Implement – Deploying & improving. Launching, monitoring, and enhancing the system.

Conclusion

The **University Library Management System** follows the **Design Thinking** process to ensure a **user-friendly, efficient, and scalable** solution. By **understanding users, defining key problems, brainstorming solutions, prototyping, testing, and continuously improving**, the system evolves to meet real-world library management needs effectively.

Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation of how data moves through a system, including its sources, processes, storage points, and destinations. It helps visualize the system's functional components and how they interact with each other. A DFD typically consists of:

- External Entities (Represented as squares or rectangles): Entities that interact with the system (e.g., users).
- Processes (Represented as ovals or circles): Actions that transform data (e.g., "User Authentication").
- Data Stores (Represented as parallel lines or open-ended rectangles): Places where data is stored (e.g., "User Table").
- Data Flows (Represented as arrows): The movement of data between entities, processes, and data stores.

Work flow:

1. Login/Registration Page

- Email & Password fields
- Role selection (**Student, Admin**)
- Authentication handled using Django's built-in authentication system

2. Search & Filter Books

- Search books by title, author, category
- Apply filters (availability, genre, edition)
- View real-time book availability status

3. Book Details Page

- Title, Author, ISBN
- Description & Genre
- Availability Status (Issued or Available)

4. Authentication System

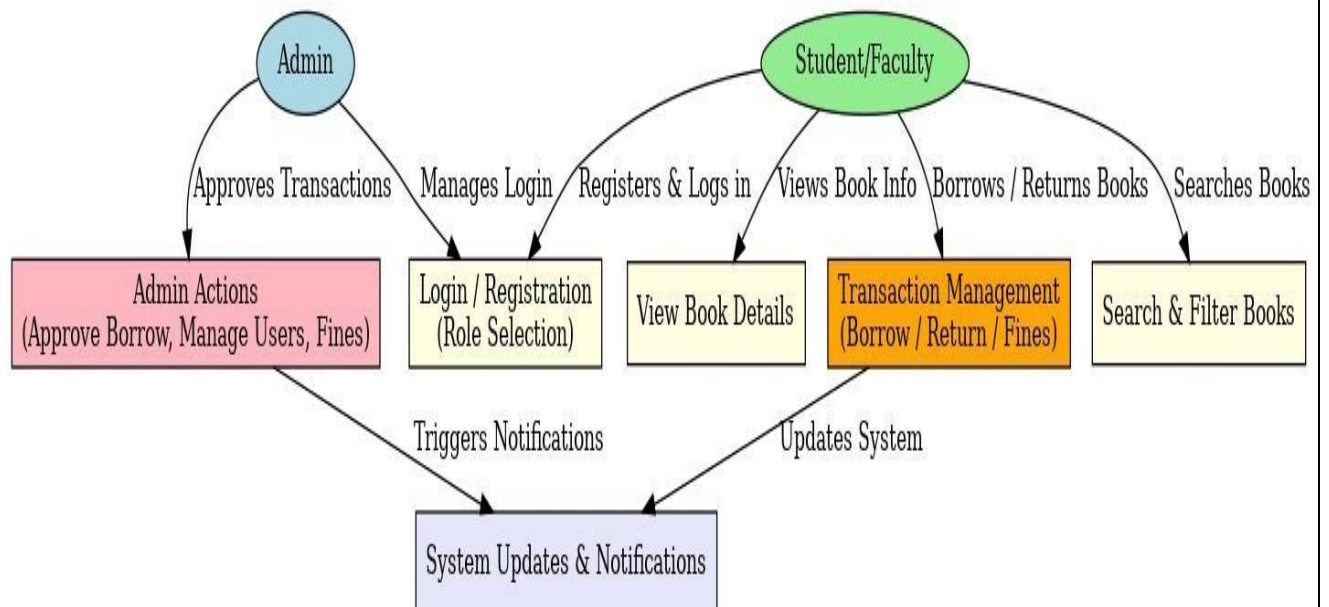
- **Admin** has access to book and student management.

- **Students** can only view books and track their own issued books.
- Secure authentication using **Django's built-in authentication system**.

5. Transaction Management

- Borrow books (if available)
- Return books
- View fine details (if overdue) For Admin:
- Approve book borrowing
- Track due dates & impose fines
- Manage book returns

DATA FLOW DIAGRAM FOR UNIVERSITY LIBRARY MANAGEMENT SYSTEM



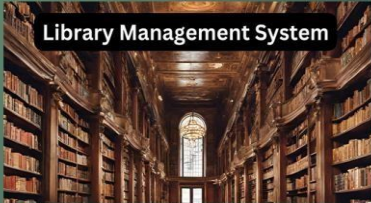
POWER POINT PRESENTATION FOR UNIVERSITY LIBRARY MANAGEMENT SYSTEM



OBJECTIVE:-

- This is a Library Management System developed using Django Framework. It helps in managing books, users, and issuing/returning of books efficiently.

Library Management System

A photograph showing a perspective view of a library's interior. Tall wooden bookshelves filled with books line both sides of a central aisle. A bright light source, possibly a window, is visible at the end of the aisle, creating a strong backlight effect.

TECHNOLOGY STACK(frame works)

- - Django (Python Framework)
- - SQLite (Database)
- - HTML/CSS (Frontend)
- - Bootstrap (Responsive Design)

KEY FEATURES

- - User Authentication (Admin & Student)
- - Book Management (Add, View, Issue, Return)
- - Issue and Return Tracking
- - Responsive User Interface
- - Contact and About Us Pages

SYSTEM FLOW

- - Admin:
 - - Add, View, Issue, and Return Books
 - - Manage Student Accounts
- - Student:
 - - View Available Books
 - - Check Issued Books
- - Authentication:
 - - Separate login for Admin and Students

DATABASE DESIGN

- - Models used:
 - - User (Admin/Student)
 - - Book (Details of books)
 - - IssuedBook (Track issued and returned books)

DESIGN COMPONENTS

- (frontend Design)
-
- 1.Login/Registration Page:
 - -Simple form with fields for email and password.
 - -Role selection (Student, Admin).
 - Role selection dropdown allows users to select either Student or Admin.
 - -Validation for input fields.
 - -UI Design: Clean, modern, and responsive
- 2.Search/filter books:
 - -Search bar with advanced filter
 - -Display the book details with availability status.

Design components

3. Book details Page:

- - Information about the book (author, title)
- Book Title and Author Information.
- Description Section for book details.
- Borrow & Return buttons for interaction.

4. Transaction management:

- -Borrowing, Return, books for users & over fine details

Additional features

• Additional features:

- -Notifications: Email reminders for overdue or d. reserved books available
- -mobile app: Simplified version (easy way) for quick book searches & reservations
- -Barcode scanning for a book
- -fine management : Automatic calculation of overdue

CONCLUSION

- This Library Management System simplifies the management of books and users. It ensures efficient tracking of book issuance and returns, enhancing library operations. Future enhancements could include:
 - - Adding notifications for book due dates
 - - Generating reports and statistics
 - - Integration with third-party APIs for enhanced functionality.

THANK YOU....

