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

DJANGO FRAMEWORK LAB

UNIVERSITY LIBRARY MANAGEMENT SYSTEM
DONE BY

M.Bhavana	**(24VV5A1271)**
K.Sushmitha shiny	(23VV1A1220)
K. Sai	(23VV1A1218)
N. Tarun	(23VV1A1237)

UNDER GUIDANCE OF
MRS.MADHUMITA CHANDA

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM
COLLEGE OF ENGINEERING VIZIANAGARAM (A)
VIZIANAGARAM

Regd. No: 24VV5A1271

CERTIFICATE

This is to certify that this is a Bonafide record of practical work done by MS. M.Bhavana of IInd B-Tech IInd Semester Class in Django Framework Lab during the year 2024-25.
No.of Tasks Completed and Certified: 13

Lecture In-Charge **Head of The Department**

Date:

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

Website:www.jntugvcev.edu.in

Subject Name: DJANGO FRAMEWORK **Subject Code:R232212SE01**

Year: 2025 **Regulation: R23**

COURSE OUTCOMES

NBA Subject Code	Course Outcomes	
	CO1	Design and build static as well as dynamic web pages and interactive web-based applications .
	CO2	Web development using Django framework.
	CO3	Analyze and create functional website in Django and deploy Django Web Application on Cloud .

CO-PO Mapping

Mapping of Course Outcomes (COs) with Program Outcomes (POs)

Course Outcomes		Program Outcomes (POs)														
		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
	CO1	3	1	3	1	3	1	1	1	2	3	2	1	3	3	2
	CO2	3	2	3	1	3	1	1	1	2	2	2	2	3	3	3
	CO3	2	3	3	3	3	2	2	2	2	3	3	3	3	3	3

Enter correlation levels 1,2 and 3 as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High) If there is no correlation, put "-"

Signature of the Course Instructor :

Date: **signature:**

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Understanding Django and its Libraries
7. Date of Experiment :13-12-2024
8.Date of Submission of Report :20-12-2024

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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

Definition:

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.

.

Key Features of Django:

1. Fast Development – Comes with built-in features like authentication, database management, and an admin panel.
2. Scalability – Suitable for small projects to enterprise-level applications.
3. Security – Protects against common security threats (SQL Injection, CSRF, XSS, etc.).
4. ORM (Object-Relational Mapper) – Allows database interaction using Python instead of SQL.
5. Built-in Admin Panel – Auto-generates an admin interface for managing data.
6. Reusable App-Developers can create modular and reusable components.

Django's MVT Architecture:

Model (M) – Handles database interactions (e.g., User, Booking).

View (V) – Manages business logic and connects models to templates.

Template (T) – Renders HTML pages dynamically.

Why Use Django?

i. **Fast Development** – Reduces development time.

ii. **Scalability** – Handles high traffic efficiently.

iii. **Security** – Protects against common web threats.

iv. **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.

PYTHON LIBRARIES:

1. Python Collections - Container Datatypes:

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

Key Types:

a. **List**: Ordered, mutable, allows duplicates.

b. **Tuple**: Ordered, immutable, allows duplicates.

c. **Set**: Unordered, no duplicates, fast membership testing.

d. **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:

a. Widgets: Buttons, labels, text boxes, etc.

b. Event handling: Respond to user interactions like clicks or key presses.

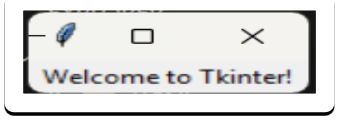
c. Simple layout management.

Common Use: Build desktop applications or tools for local interaction with a web app backend.

CODE:

```
from tkinter import Tk, Label
# Create a window
root = Tk()
root.title("Hello Window")
# Add a label to display text
Label(root, text="Welcome to Tkinter!").pack()
   # Run the application
root.mainloop()
```

Output:



3. Requests - HTTP Requests:

Purpose: Simplifies HTTP requests to interact with web APIs.

Key Features:

a. Send GET, POST, PUT, DELETE requests easily.

b. Handle request parameters, headers, and cookies.

c. Simple error handling and response handling.

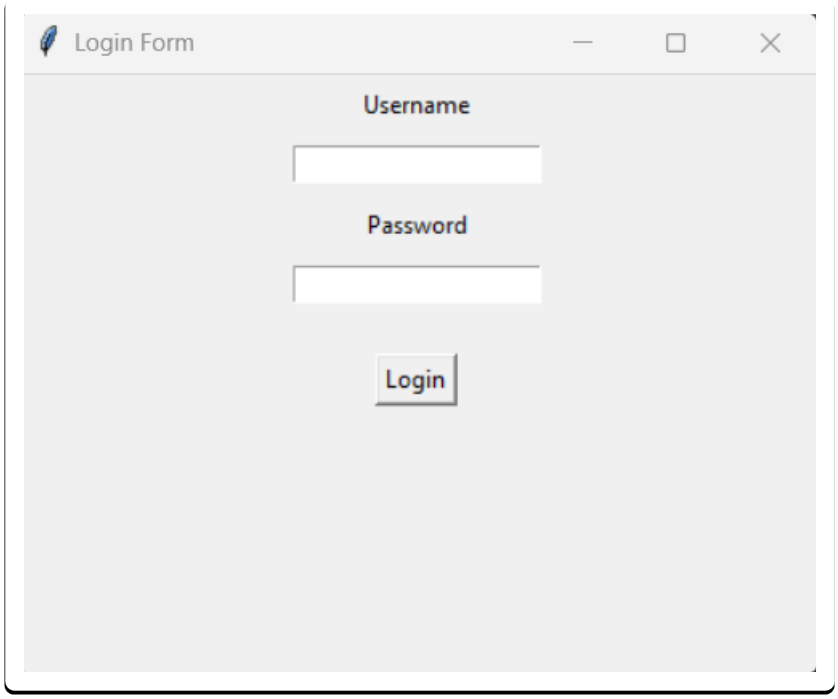
Common Use: Interact with REST APIs, download content from the web.

CODE:

```
# Install tkinter (not needed for standard Python installations)
# pip install tk  # This is not necessary for Tkinter, as it comes with Python by default
import tkinter as tk
from tkinter import messagebox
# Function to validate login
def validate_login():
    username = username_entry.get()
    password = password_entry.get()
    # Example credentials
    if username == "user" and password == "password":
        messagebox.showinfo("Login Success", "Login Successful!")
    else:
        messagebox.showerror("Login Failed", "Invalid username or password")
# Create the main window
root = tk.Tk()root.title("Login Form")
root.geometry("400x300")  # Adjusted to a more reasonable size
```

```
# Create username and password labels and entry widgets

username_label = tk.Label(root, text="Username")
username_label.pack(pady=5)
username_entry = tk.Entry(root)
username_entry.pack(pady=5)
password_label = tk.Label(root, text="Password")
password_label.pack(pady=5)
password_entry = tk.Entry(root, show="*")  # 'show' hides the password characters
password_entry.pack(pady=5)
# Create the login button
login_button = tk.Button(root, text="Login", command=validate_login)
login_button.pack(pady=20)
# Run the Tkinter event loop
root.mainloop()
```

Output:



4. Scrapy:

Purpose: An open-source web crawling framework for large-scale web scraping.

Key Features:

a. Fast, extensible, and asynchronous web scraping.

b. Supports handling requests, data extraction, and storing results.

c. 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:

a. Easy navigation and searching within HTML.

b. 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:

```
import requests

from bs4 import BeautifulSoup

# The URL of the website to scrape

url = 'https://example.com'  # Replace with the actual website URL

# Set user-agent headers to avoid getting blocked

headers = {

    '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'}

try:

    # Send a GET request

    response = requests.get(url, headers=headers, timeout=10)

    response.raise_for_status()  # Raise an error for HTTP errors (e.g., 404, 500)

    # Parse the HTML content with BeautifulSoup
# Extract and print the page title

 title = soup.title.string if soup.title else "No title found"

 print(f"Title of the page: {title}\n")
```

```
# Extract and print the page title
title = soup.title.string if soup.title else "No title found"
print(f"Title of the page: {title}\n")
# Extract and print all headings (h1, h2, h3)
headings = soup.find_all(['h1', 'h2', 'h3'])
if headings:
    print("Headings:")
for heading in headings:
    print(f" - {heading.name}: {heading.text.strip()}")
    else:
        print("No headings found.")
# Extract and print all links
links = soup.find_all('a', href=True)
    if links:
        print("\nLinks:")
        for link in links:
            print(f" - {link['href']}")
    else:
        print("No links found.")
except requests.exceptions.RequestException as e:
    print(f"Error fetching the webpage: {e}")
```

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:

a. Supports frameworks like Flask and Django for serverless deployments.

b. 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:

a. Built on top of Flask, React, and Plotly.

b. 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:

a. Modular: Mix and match components like SQLAlchemy, Genshi, and others.

b. 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:

```
import cherrypy
class HelloWorld:
    @cherrypy.expose
    def index(self):
        return "Hello, World! This is a CherryPy web page."
if __name__ == '__main__':
    # Server Configuration (Custom Port & Logging)
    cherrypy.config.update({
        'server.socket_host': '127.0.0.1',  # Bind to localhost
        'server.socket_port': 9090,         # Change port (default is 8080)
        'log.error_file': 'cherrypy_error.log',  # Log errors
        'log.access_file': 'cherrypy_access.log'})
    print("Server is running on http://127.0.0.1:9090/")
    try:cherrypy.quickstart(HelloWorld())
    except Exception as e:
        print(f"Error starting CherryPy: {e}")
```

Output: After run the server:-

```
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"
```

10. Flask:

Purpose: Lightweight micro-framework for building web applications.

Key Features:

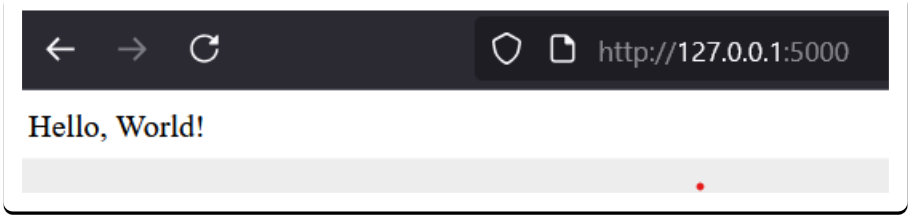
a. Simple to learn and use, but highly extensible.
b. Supports extensions for database integration, form handling, authentication, etc.

Common Use: Small to medium web applications, APIs, or microservices.

CODE:

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

Output: After run the server



11. Web2Py:

Purpose: Full-stack framework for rapid web application development.

Key Features:

a. Includes a web-based IDE for development.
b. 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:

a. Single-file framework, minimalistic, and fast.

b. No dependencies, supports routing, templates, and form handling.

Common Use: Small web applications, APIs, and prototypes.

CODE:

```
from bottle import Bottle, run
# Create a Bottle application instance
app = Bottle()
@app.route('/')
def index():
    return "Hello, World! This is a Bottle web page."
# Run the Bottle application
if __name__ == '__main__':
    run(app, host='localhost', port=8080, debug=True)
```

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:

a. Focuses on speed and minimalism.

b. 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:

a. Uses a highly modular architecture for development.

b. 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:

a. Full support for Python's object-oriented programming.

b. Easily extensible, with minimalistic core.

c. **Common Use**: Scalable and customizable web applications.

16. Pyramid:

Purpose: Full-stack web framework that can scale from simple to complex applications.

Key Features:

1. Highly flexible with support for routing, templating, authentication, and authorization.

2. Allows for small and large applications, with fine-grained control.

Common Use: Building large, enterprise-grade web applications and REST APIs.

SUMMARY:

a) **Flask**, **Django**, **Pyramid**: Popular web frameworks, each offering flexibility and scalability.

b) **Scrapy**, **BeautifulSoup4**: Specialized for web scraping and data extraction.

c) **Requests**, **Zappa**, **Dash**: Tools for making HTTP requests, serverless apps, and interactive data visualizations.

d) **Tkinter**, **Bottle**, **CherryPy**: Libraries for building lightweight desktop and web applications.

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Introduction to Django Framework in Project
7. Date of Experiment :20-12-2024
8.Date of Submission of Report :27-12-2024

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation 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:

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:
a.Enhance the learning experience for students.
b.Provide a structured way to manage courses and materials.
c.Automate administrative tasks like enrollment, grading, and reporting.
d.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:

a. CSS3 – For styling and layout design.

b. JavaScript – For adding interactivity and enhancing user experience..

c. HTML5 – For structuring the web pages.

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Step-by-step Guide to installing Django
7. Date of Experiment :20-12-2024
8.Date of Submission of Report :27-12-2024

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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 INSTALLATION:

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

```
djangoadmin startproject project_name

django-admin startapp app_name
```

Step-7 : Run the server

```
python manage.py runserver
```

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

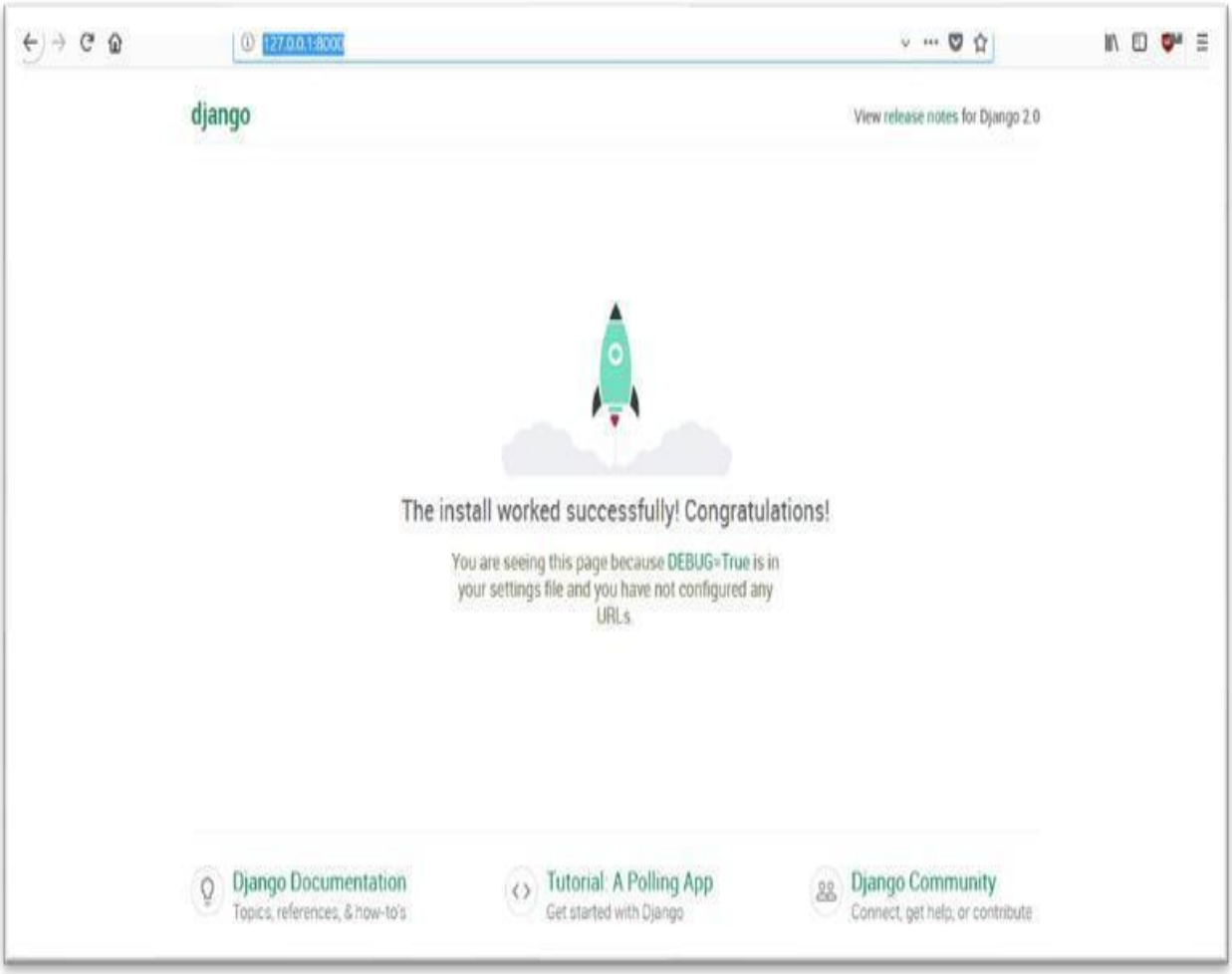
Successfully installed asgiref-3.8.1 django-5.1.4 sqlparse-0.5.3 tzdata-2024.2
(pythondjango) C:\Users\vccl>mkdir Django_Projects

(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:


127.0.0.1:8000
django
View release notes for Django 2.0
The install worked successfully! Congratulations!
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Linking views and URLconfigurations
7. Date of Experiment :27-12-2024
8.Date of Submission of Report :03-01-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

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'
```

Run Migrations:

```
python manage.py migrate
```

Run the Server and Test:

```
python manage.py runserver
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :setting up APP-LEVEL URLs
7. Date of Experiment :24-01-2025
8.Date of Submission of Report :31-01-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation 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:

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"),
]
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Exploring Django Views
7. Date of Experiment :24-01-2025
8.Date of Submission of Report :31-01-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation 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:

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. 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.

Code:

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

```
user.save()
            return render(request, "change_password.html", {'alert': True})
        else:
            return render(request, "change_password.html", {'currpasswrong': True})
 return render(request, "change_password.html", {'alert': True})
        else:
            return render(request, "change_password.html", {'currpasswrong': 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 HttpResponse("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
HttpResponse("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("/")
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :working With Templates in Django
7. Date of Experiment :31-01-2025
8.Date of Submission of Report :17-02-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation 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:

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.

1.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.

2.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.

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

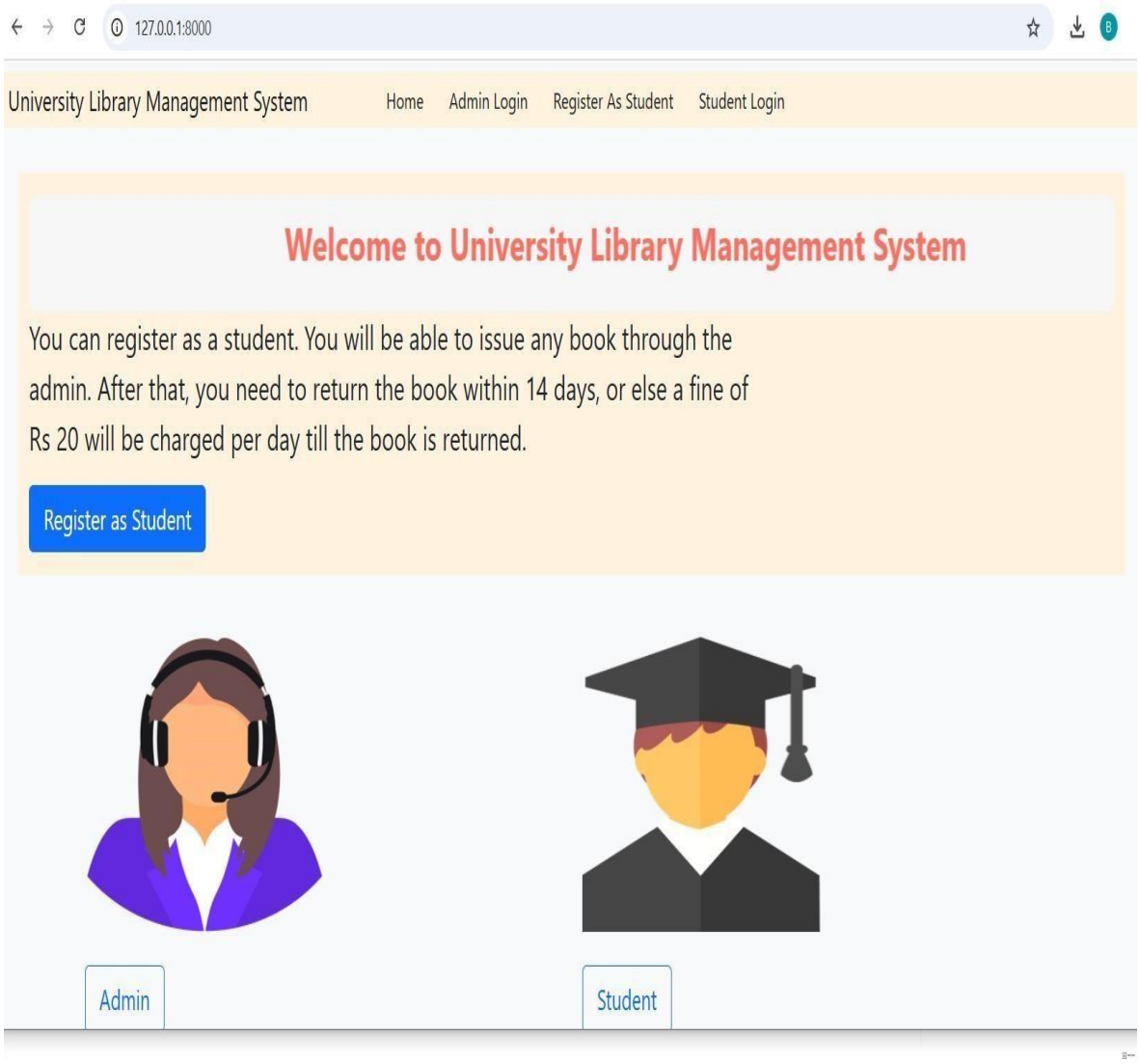
Index.html:

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

{% load static %}

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

{% block css %}

<style>
   /* Styling the marquee */     .marquee-container {  width: 100%;overflow: hidden;
   .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>
```

```
<div class="container">

 <div class="row">

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

 <img src="{% static 'admin.png' %}" width="50%" height="50%
alt=""><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">
<img src="{% static 'students.png' %}" width="50%" height="50%"
alt=""><br><br>

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

   </div>

  </div>

  </div>

</div>

{% endblock %}
```

Output:



Role of index.html:

This is the **homepage** of your University Library Management System. It:

1. **Shows a welcome message** that scrolls across the screen (like a marquee).
2. **Explains how the system works** (like book issuing, fines, etc.).
3. Has a **button to register as a student**.
4. Has **buttons for Admin and Student login**, each with an image.
5. Uses some **CSS styling** to make the page look nice.
6. **Extends a base template** so the layout is consistent with other pages.
7. Loads **static images** (like admin and student icons).

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;
textalign: 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;

   }
```

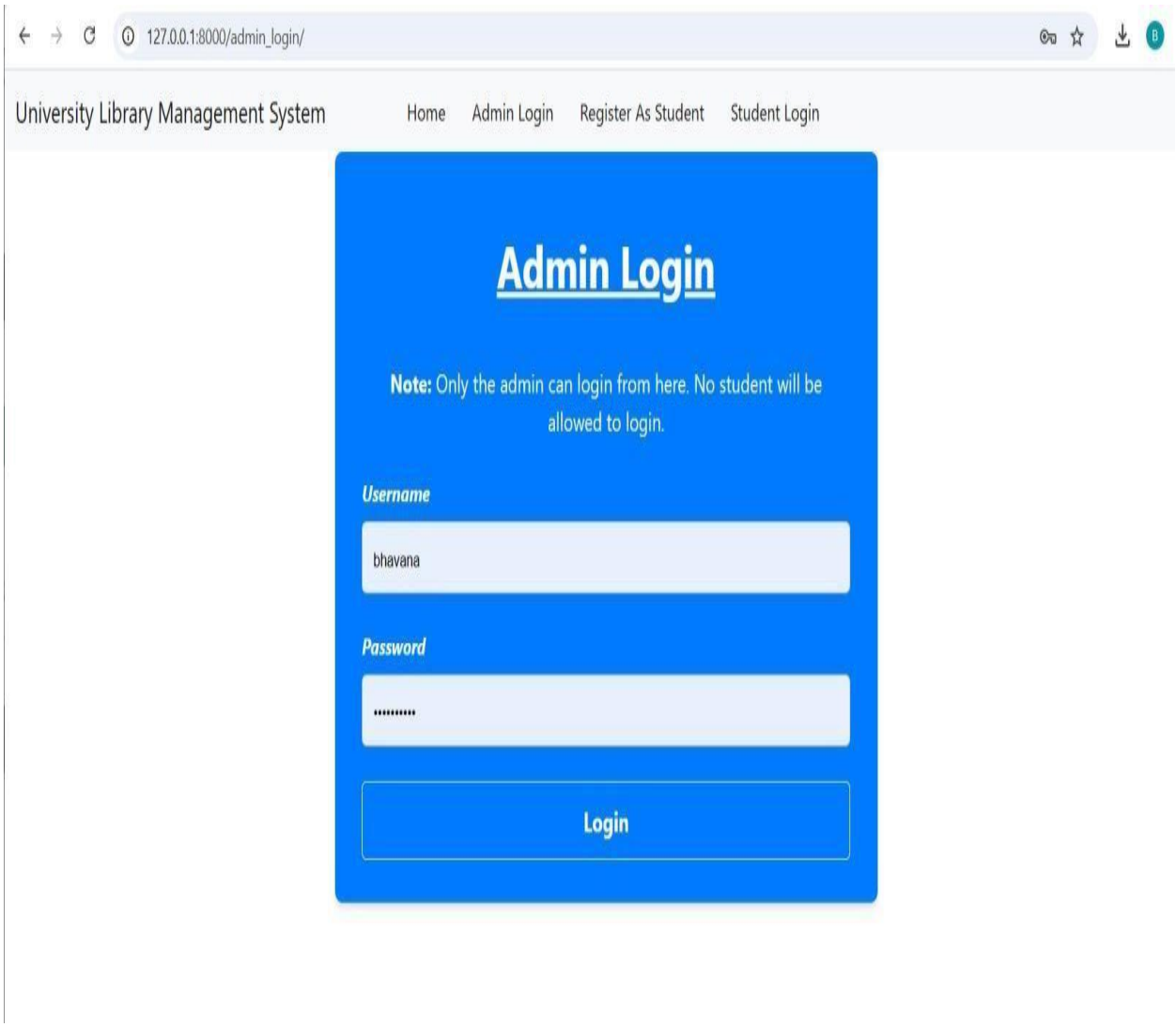
```
    }    /* Form fields styling */
.formcontrol {         border-
radius: 5px;         box-shadow:
none;         border: 1px solid
#ccc;         padding: 12px;
fontsize: 1rem;         transition:
border-color 0.3s ease;
 } padding:12px;fontsize:1re;
transition:  border-color 0.3s
ease;  }   .form-control:focus {
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 %}
 <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
mt4">Login</button>

    </form>

</div>

{% endblock %}

{% block js %}

<script>

    {% if alert %}     alert("Invalid

Username or Password.") document.location = "/admin_login"

    {% endif %}
</script>   {%endblock%}
```

Output:



Role of admin_login.html:

This page is the **login screen for the admin** of your Library Management System.

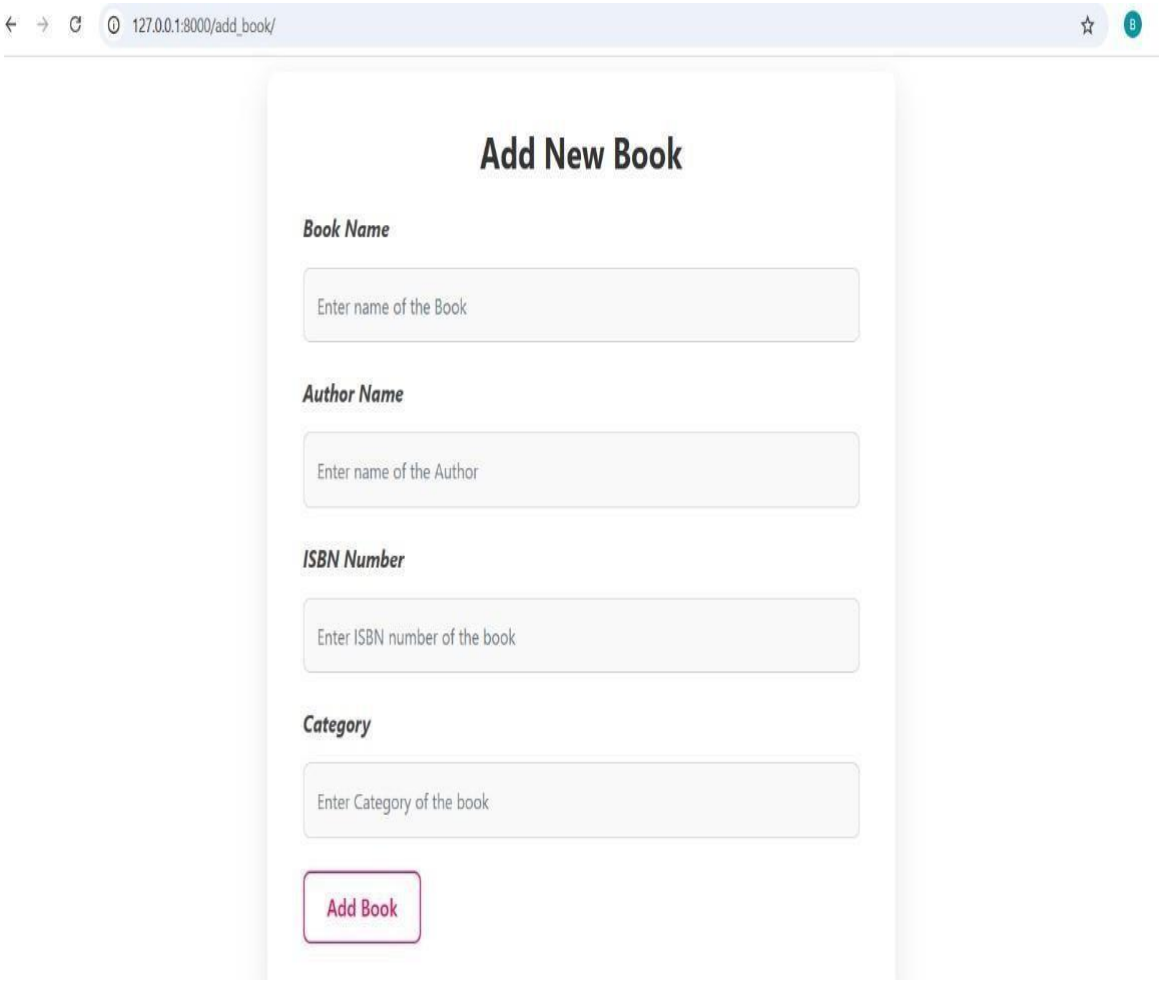
1. **Extends a base layout** (basic.html) for a consistent look.
2. **Shows a form** where the admin can enter their **username** and **password**.
3. Has a **note** telling users that **only admins are allowed** to log in here.
4. **Handles login errors** by showing a red warning if login fails.

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
mt4">AddBook</button>
</div>
</form>
{% endblock %}
{% block js %}
<script>
   {% if alert %}     alert("Book is added successfully.")

document.location =  "/view_books"

   {% endif %}
</script>
{% endblock %}
```

Output:



Role of add_book.html:

This page is used by the **admin to add a new book** to the library database.

Extends admin_navbar.html so the admin sees the normal navigation layout.

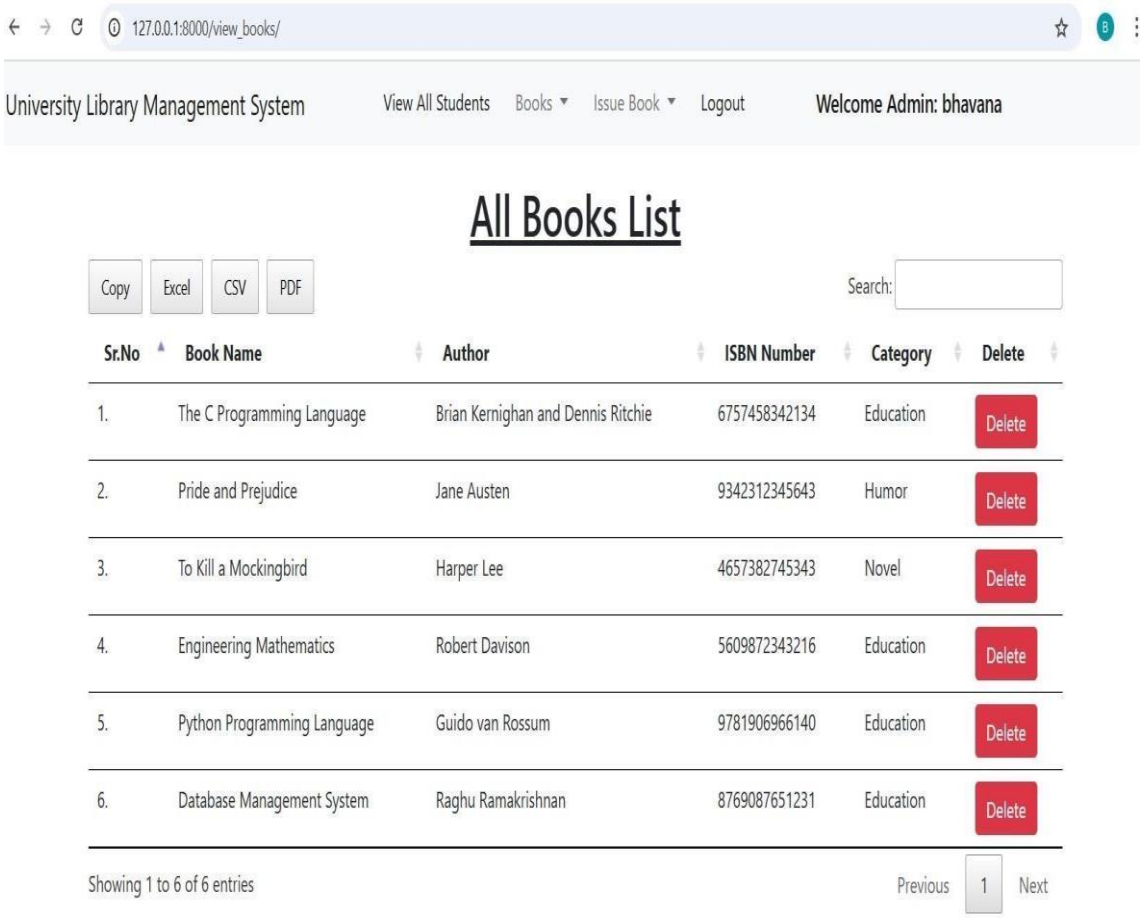
1. Displays a **form** for the admin to enter:
2. Uses Django's csrf_token to keep the form secure.
3. Has a **Submit button** labeled "Add Book".
4. If the book is successfully added, it shows a **popup message** and redirects the admin to the **View Books** page.

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:



Role of view books.html:

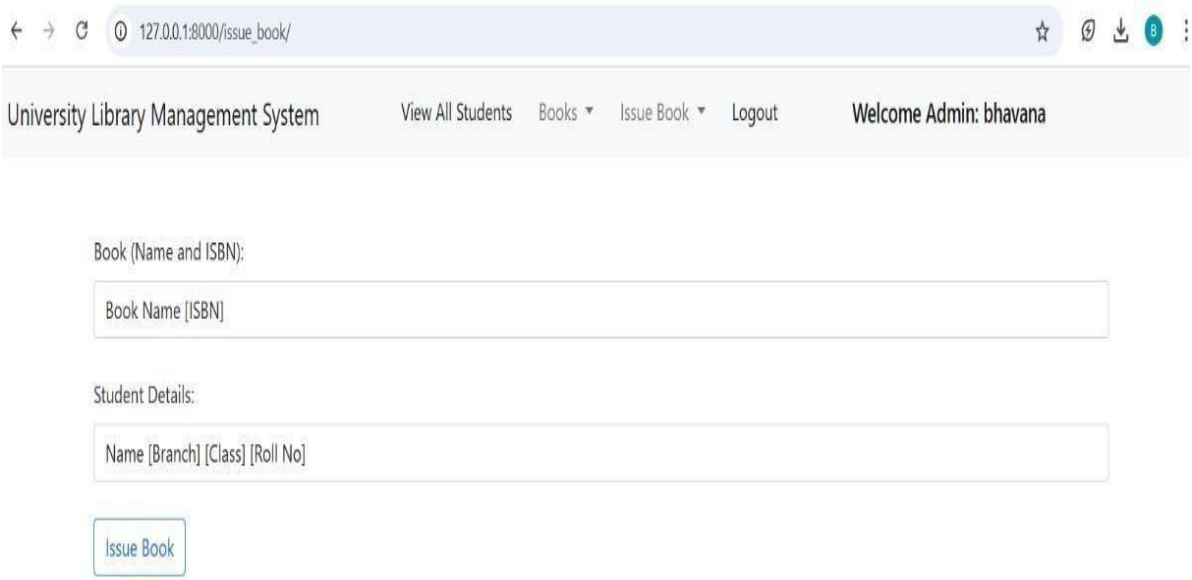
This page shows the list of all books currently in the library — visible to the admin.

1. Extends admin_navbar.html to keep the admin layout consistent.
2. Displays a table with the following info for each book:
    a. Serial Number (Sr.No)
    b. Book Name
    c. Author
    d. ISBN Number
    e. Category
3. Adds a Delete button for each book that:
    a. Sends the admin to /delete_book/<book.id>/
    b. Asks for confirmation before deleting
4. Uses Django's {% for book in books %} loop to dynamically list all books passed from the backend.

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
mt4">IssueBook</button>

</div>  </form>

{% endblock %}

{% block js %}

<script>

 {% if alert %}     alert("Book Successfully

Issued.")     document.location =

"/issue_book"      {% endif %}   </script>


{% endblock %}
```

Output:



Role of issue_book.html:

This page lets the **admin issue a book to a student**.

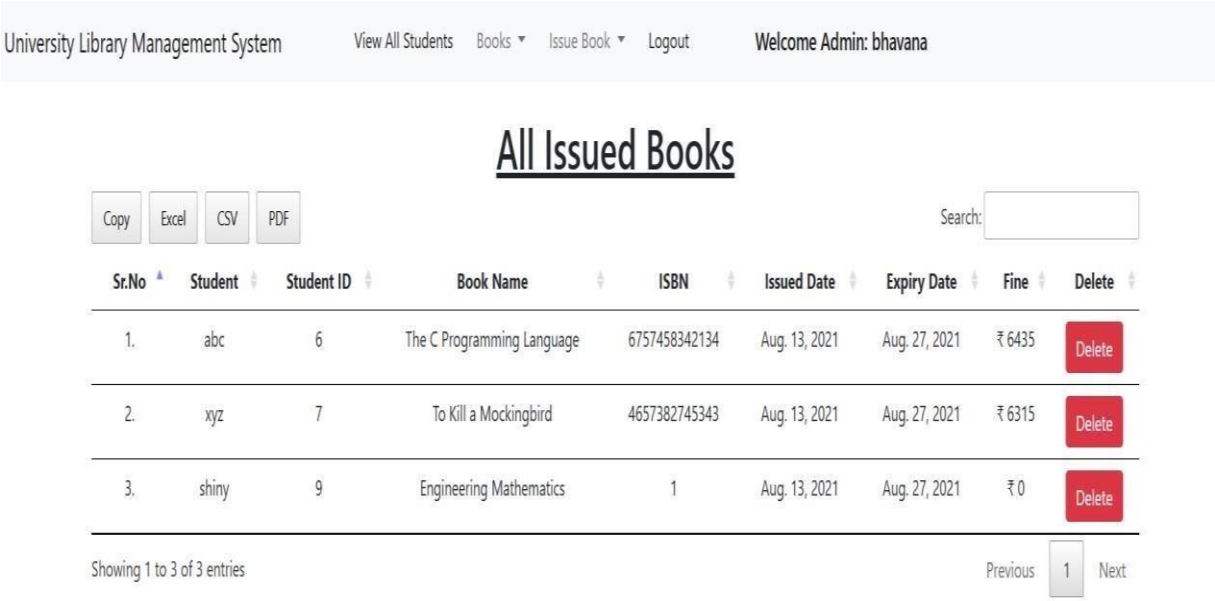
1. **Extends** the admin_navbar.html so the layout matches other admin pages.
2. Displays a **form**, most likely with:
    a. A dropdown to select a **student**
    b. A dropdown to select a **book**
    c. Possibly a date field (if included in the form

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:



Role of view_issued_book.html:

This page allows **students** to **view the list of books** they've been issued — along with return deadlines and any fines.

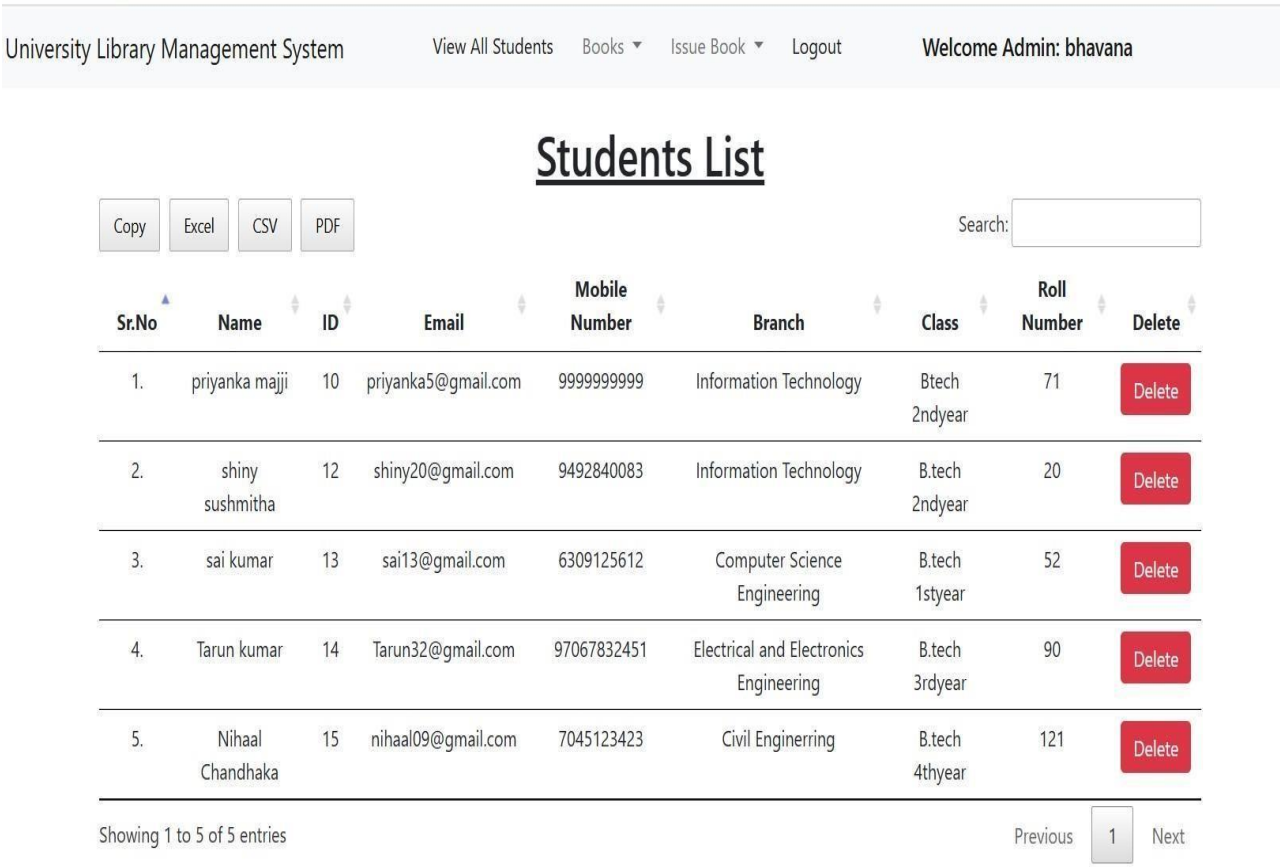
1. **Extends** student_navbar.html so the layout matches other student pages.
2. Displays a **table** showing:
    a. Serial Number
    b. Student ID
    c. Student Name
    d. Book Name
    e. Author
    f. Issued Date
    g. Expiry Date
    h. Fine (if any)

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 %}
```

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

  <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:



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

Role of view_students.html:

This page shows the **admin** a full list of **registered students** in a table format, with an option to **delete** any student.

For each student, the table displays:

a. Serial Number (Sr.No)
b. Full Name (student.user.get_full_name)
c. User ID
d. Email
e. Phone number
f. Branch (e.g., CSE, ECE)
g. Class (e.g., 3rd year, 2nd year)
h. Roll Number

Student_registration.html:

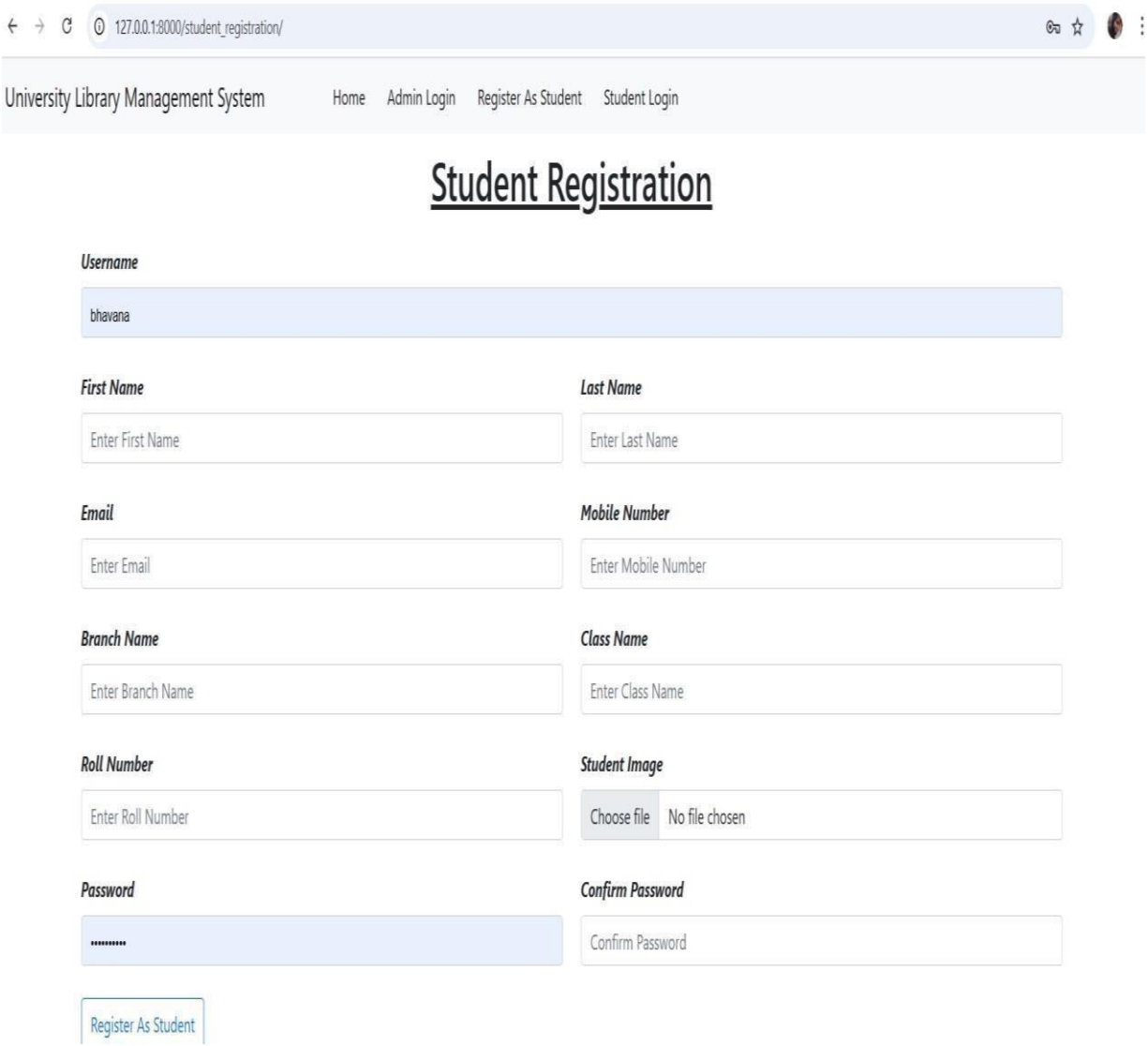
```
{% extends 'basic.html' %}

{% 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:



Role of Student Registartaion.html:

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_login.html:

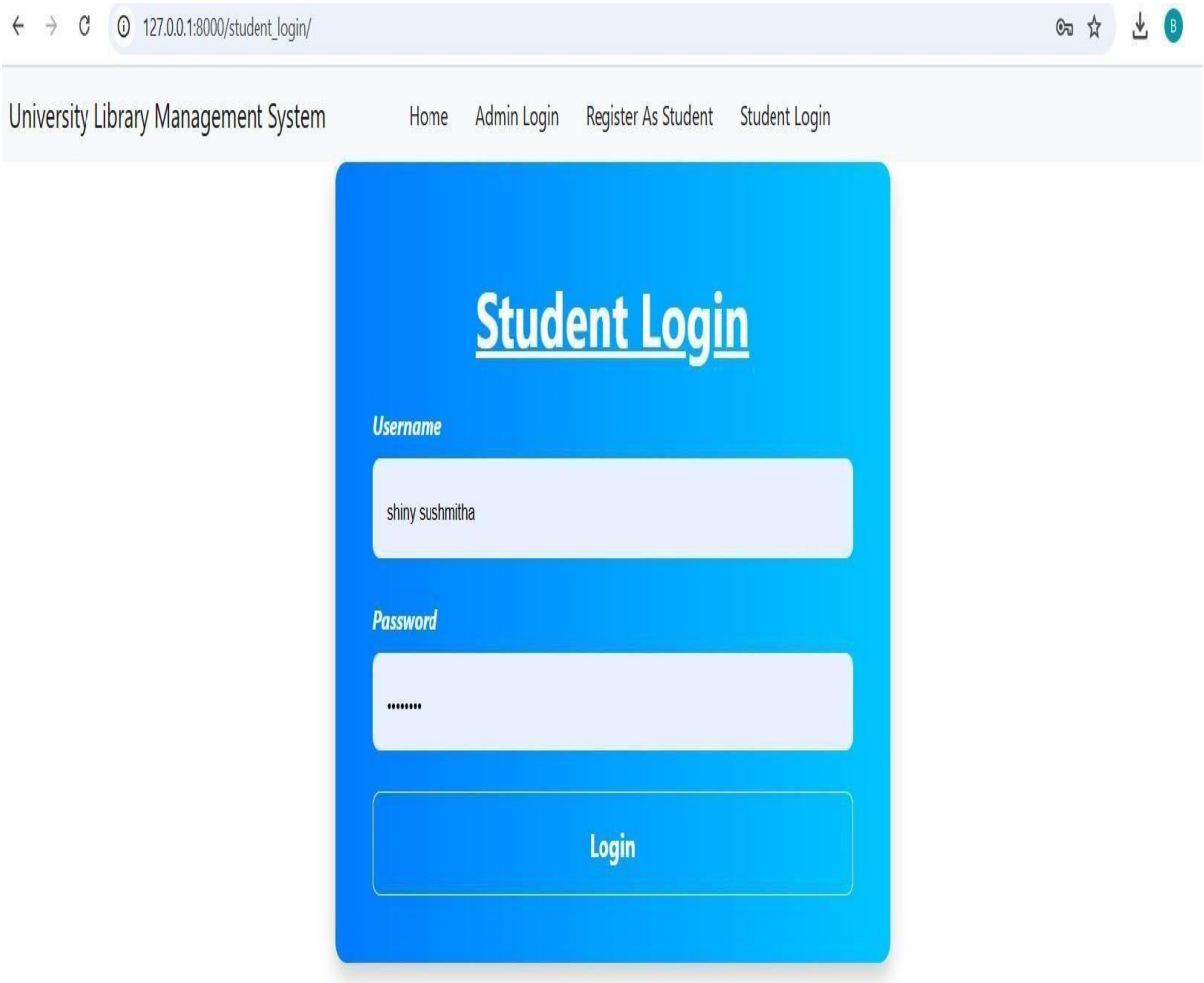
```
{% extends 'basic.html' %}

{% 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: 1.1remtransition:

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{  margintop:
30px;   }
```

```
    .mb-4 { marginbottom: 30px;
    }   /* Responsive Design */
    @media (max-width: 767px) {
 .container {
padding: 20px;
        }
h1 {font-size: 2rem; /* Slightly smaller
titlemobile */
        }
 .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:



Role of Student Login.html:

This page lets students **log in** to your Library Management System using their:

a. This is a student login page.

b. Students enter their username and password.

c. They click "Login" to access their account.

d. It's part of a university library system.

e. The page is simple and focused on authentication.

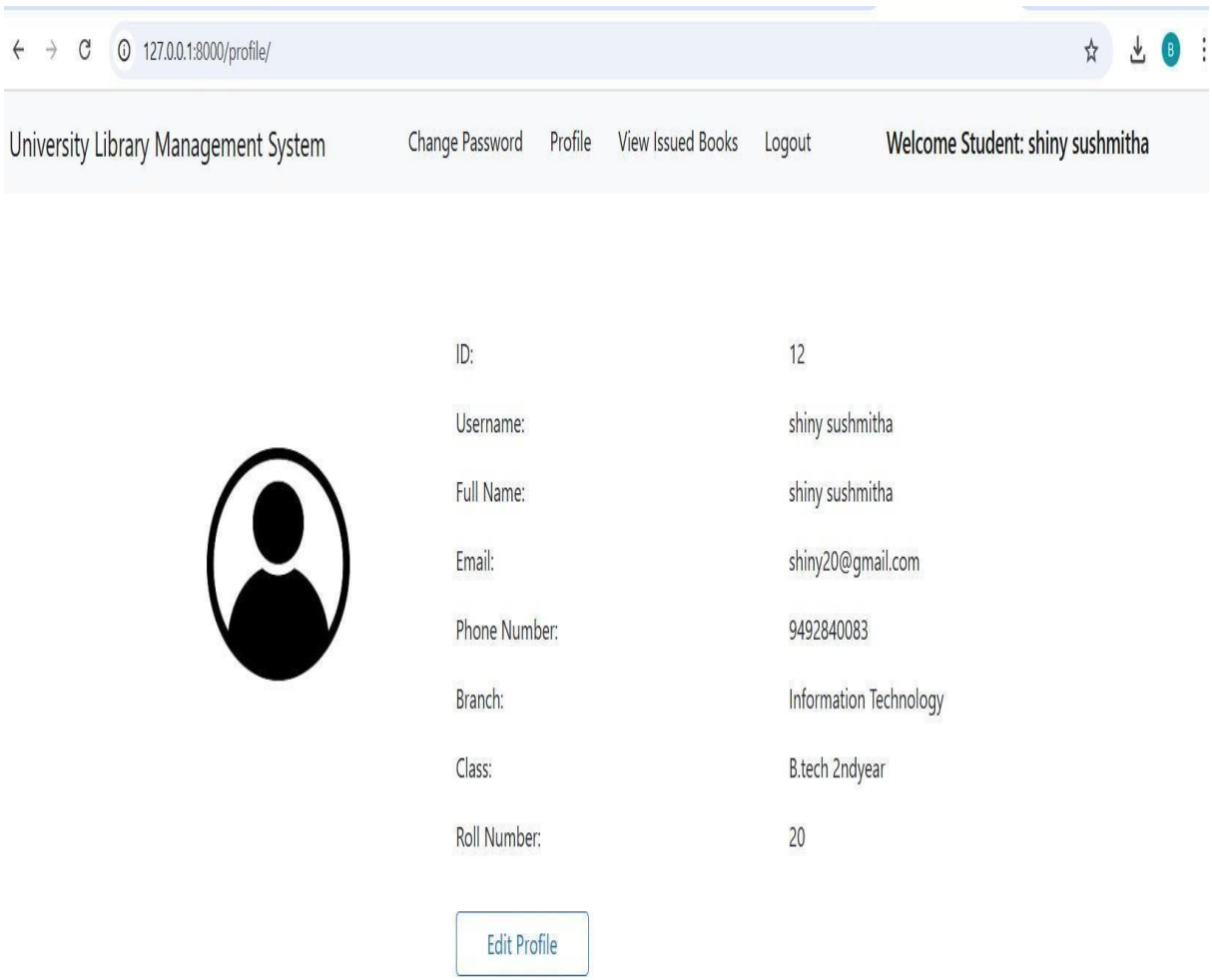
Profile.html:

```
{% 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;overf

low: 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">
<img src="{{user.student.image.url}}" alt="" width="310px"
height="270px">
                </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 class="col-md-6">
<p>{{user.email}}</p>
</div>
        </div>
       <div class="row">
     <div class="col-md-6">
            <label>Phone Number:</label>
     </div>
   <div class="col-md-6">
            <label>Phone Number:</label>
     </div>
   <div class="col-md-6">
       <p>{{user.student.phone}}</p>
   </div>
   </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">

 <label>Class:</label>
   </div>
  <div class="col-md-6">
  <p>{{user.student.classroom}}</p>
 </div>
 </div>
 <div class="col-md-6">
<label>Roll Number:</label>
</div>
<div class="col-md-6">
    <p>{{user.student.roll_no}}</p>
   </div>
  </div>
    </div>
  <a href="/edit_profile/" style="width: 9rem;" class="btn btn-outline-primary
mt3">Edit Profile</a>
  </div>
 </div>
</div>
</div>
{% endblock %}
```

Output:



Role of Profile.html:

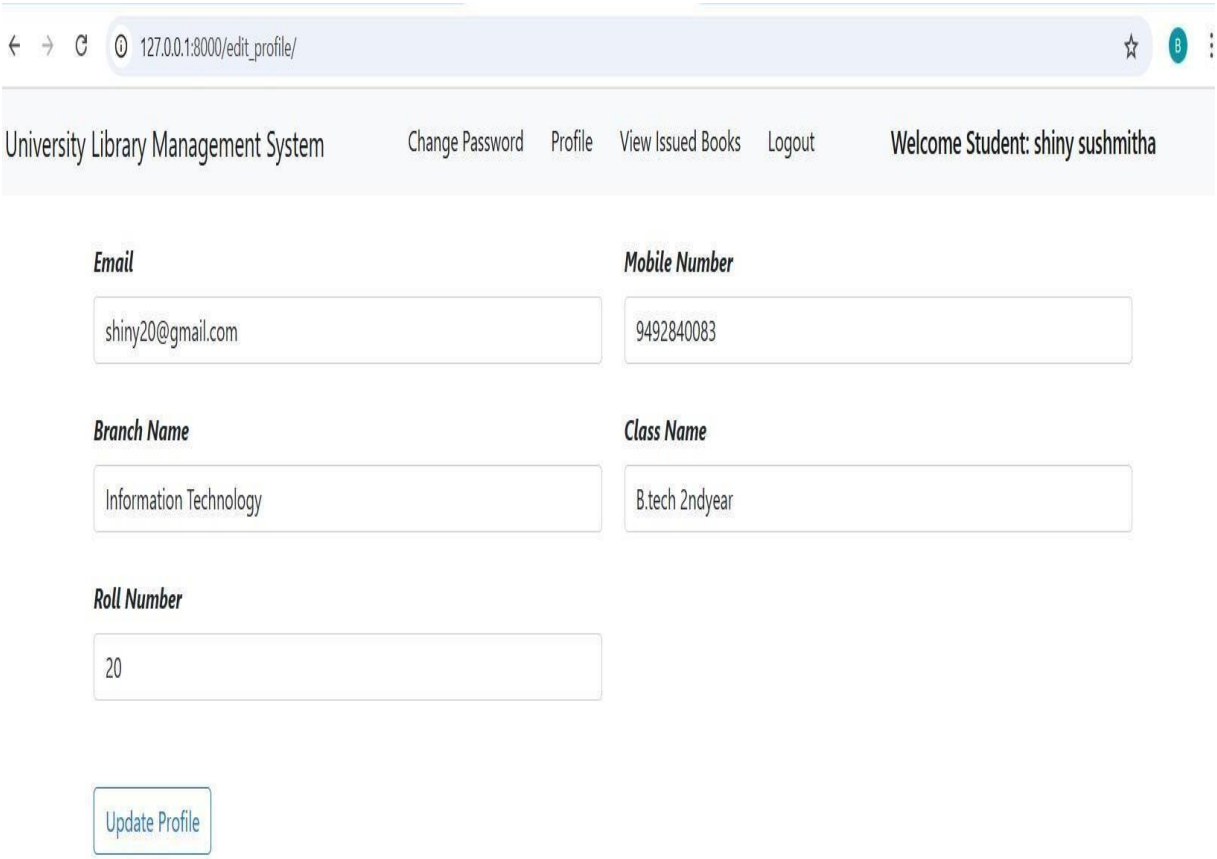
Edit profile button provided for updates. This is a student profile page.It displays the student's personal information like ID, name, email, and contact details.It also shows academic details like branch, class, and roll number.The student can edit their profile using the "Edit Profile" button.A profile picture is shown as a placeholder.The page is part of a University Library Management System.Navigation links include "Change Password" and "View Issued Books."The header shows the student's name, "shiny sushmitha."It provides a summary of the student's information.The page facilitates student profile management.

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:



Role of editprofile.html:

1. Students can change their contact and academic details.
2. They click "Update Profile" to save changes.
3. It's part of a library management system.
4. This is a student profile edit page.

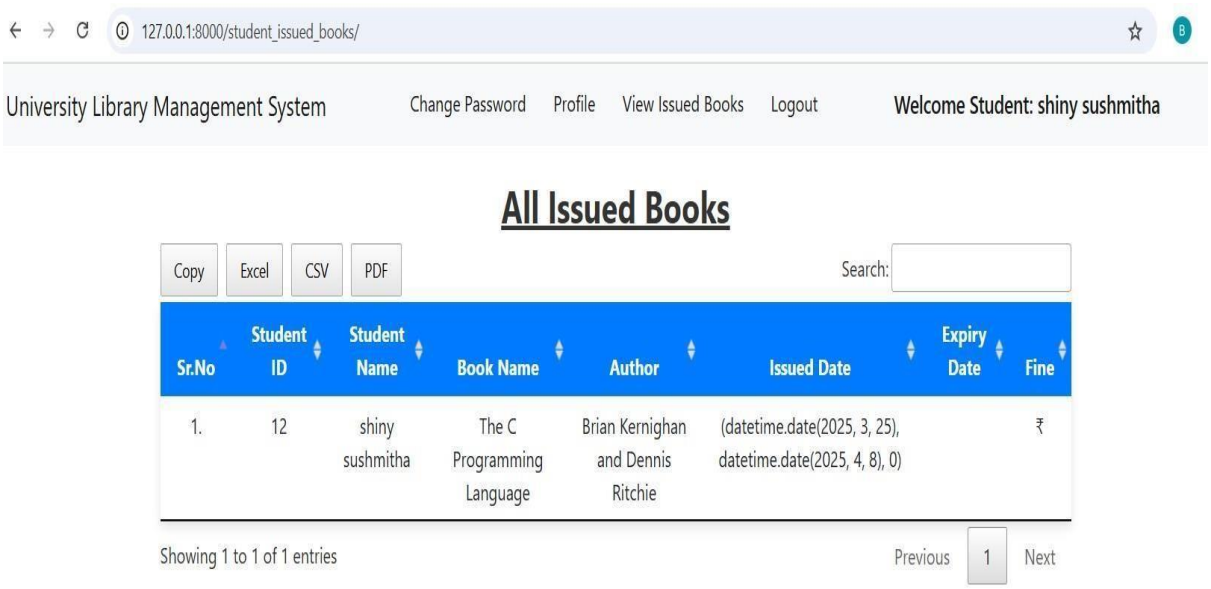
Student issued_books.html:

```
{% 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 {

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 %}

  <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:

.



Role of student issued books.html:

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.

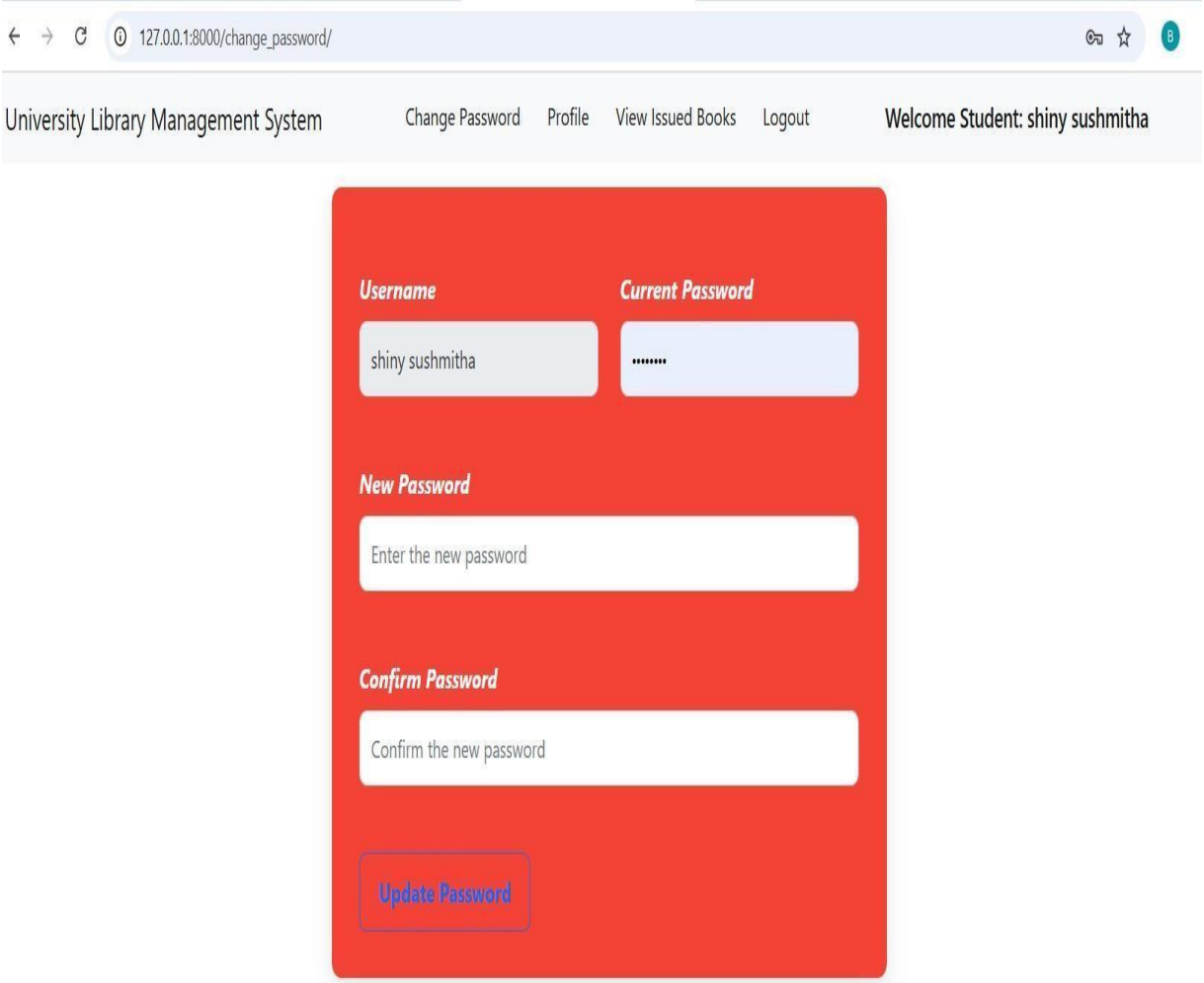
Change_password.html:

```
{% 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;

  }

<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>
   <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.");

/* 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>

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:



Role of change password.html:

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.

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Database Integration and Configuration
7. Date of Experiment :17-02-2025
8.Date of Submission of Report :21-02-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

Database:

A database is an organized collection of data that is stored and managed in a way that makes it easy to retrieve, update, and manage information. Databases are used to store data in a structured format, allowing efficient access, management, and modification.

Step 1: Check if SQLite3 is Already Installed

sqlite3 –version

On Windows:

1. Download the SQLite3 command-line tool from the official website:
   SQLite Downloads
2. Download the **"Precompiled Binaries for Windows"** (usually a ZIP file).
3. Extract the ZIP file and place sqlite3.exe in a folder (e.g., C:\sqlite).
4. Add the folder to your system's **PATH** environment variable:
   i. Search for "Environment Variables" in the Start menu.
   ii. Click on "Environment Variables."
   iii. In "System variables," select "Path" and click "Edit."
   iv. Add the folder path (e.g., C:\sqlite) and click OK.

On macOS: sqlite3 –version

Step 4: Using SQLite3 with Django

Since Django uses SQLite3 as the default database, you don't need to install any additional drivers. Just make sure your settings.py file has the following configuration:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Step 5: Run Migrations to Create the Database

i. After setting up, run:

ii. python manage.py migrate

1. **Step 1: Open the Django DB Shell**
2. Make sure your virtual environment is activated, then use the following command:
3. python manage.py db shell
4. **Step 2: Common SQLite Commands**
5. Once you're inside the SQLite shell, you can use the following commands:
6. List All Tables:
7. .tables
8. **Step3: View Table Schema:**
9. .schema table_name;
10. **Step4: Show All Data in a Table:**
11. SELECT * FROM table_name;
12. Exit the SQLite Shell:
13. .exit

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Handling Forms in Django
7. Date of Experiment :17-02-2025
8.Date of Submission of Report :21-02-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

What is forms.py in Django:

In Django, forms.py is used to handle user input efficiently and securely. It allows developers to create and manage forms without manually writing HTML and validation logic.

Why Use forms.py:

i. Simplifies form creation

ii. Handles input validation automatically

iii. Integrates with Django models

iv. Prevents security risks like SQL Injection & CSRF attacks

Types of Forms in Django:

1. Django Forms (forms.Form) – Used for manually creating forms
2. **Model Forms (forms.ModelForm)** – Used to create forms directly from a Django model

Forms.py:

```
from django import forms

from django.contrib.auth.models import User

from . import models

class IssueBookForm(forms.Form):

 isbn2 = forms.ModelChoiceField(queryset=models.Book.objects.all(),
empty_label="Book Name [ISBN]", to_field_name="isbn", label="Book (Name and
ISBN)"

 name2 = forms.ModelChoiceField(queryset=models.Student.objects.all(),
empty_label="Name [Branch] [Class] [Roll No]", to_field_name="user", label="Student
Details"

isbn2.widget.attrs.update({'class': 'form-control'})

 name2.widget.attrs.update({'class':'form-control'})
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Defining and using Models
7. Date of Experiment :21-02-2025
8.Date of Submission of Report :07-03-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation 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:

What is models.py in Django:

In Django, models.py is where you define the database structure using Python code. Django models act as a bridge between the database and the application, allowing you to create, read, update, and delete records easily.

Why Use Django Models:

No need to write raw SQL queries, Automatically creates tables in the database and

How to Apply Models:

Create the Model in models.py:

1. Write your models inside the models.py file.

Models.py:

```
from django.db import models

from django.contrib.auth.models import User

from datetime import timedelta, date

# Function to set expiry date 14 days from the issue date

def expiry():

    return date.today() + timedelta(days=14)

# Book model to store book details

class Book(models.Model):

    name = models.CharField(max_length=200)

    author = models.CharField(max_length=200)

    isbn = models.CharField(max_length=13, unique=True)  # ISBN as a string (with or without dashes)

    category = models.CharField(max_length=50)

    def __str__(self):
```

```
return f"{self.name} [{self.isbn}]"

# Student model to store student details

class Student(models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE)

    classroom = models.CharField(max_length=10)

    branch = models.CharField(max_length=10)

    roll_no = models.CharField(max_length=10, blank=True)  # You can make roll_no
optional

phone = models.CharField(max_length=10, blank=True)   # Optionally make phone
field blank

 image = models.ImageField(upload_to="students/", blank=True)  # Set upload path
for student images

 def __str__(self):

       return f"{self.user} [{self.branch}] [{self.classroom}] [{self.roll_no}]"

# IssuedBook model to store issued book details with foreign keys for student and book

class IssuedBook(models.Model):

    student = models.ForeignKey(Student, on_delete=models.CASCADE)  # ForeignKey
to Student model

    book = models.ForeignKey(Book, on_delete=models.CASCADE)  # ForeignKey to
Book model

    issued_date = models.DateField(auto_now_add=True)  # Set the issue date
automatically when created

    expiry_date = models.DateField(default=expiry)  # Set the expiry date to 14 days from
issue date

 def __str__(self):

        return f"{self.book.name} issued to {self.student.user.username} until
{self.expiry_date}"
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Migrations:syncwith the Database
7. Date of Experiment :07-03-2025
8.Date of Submission of Report :27-03-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

Migrations: Sync with the Database:

Run Migrations to Create Database Tables:

After defining your models, run the following commands to apply them to the database:

- python manage.py makemigrations
- python manage.py migrate

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Deploying Django Applications on Cloud Platforms
7. Date of Experiment :27-03-2025
8.Date of Submission of Report :04-04-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

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 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:

1.git init

2.Connect to GitHub:

git remote add origin

https://github.com/bhavana071/Bhavana

3.Add and commit changes: gitgit 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

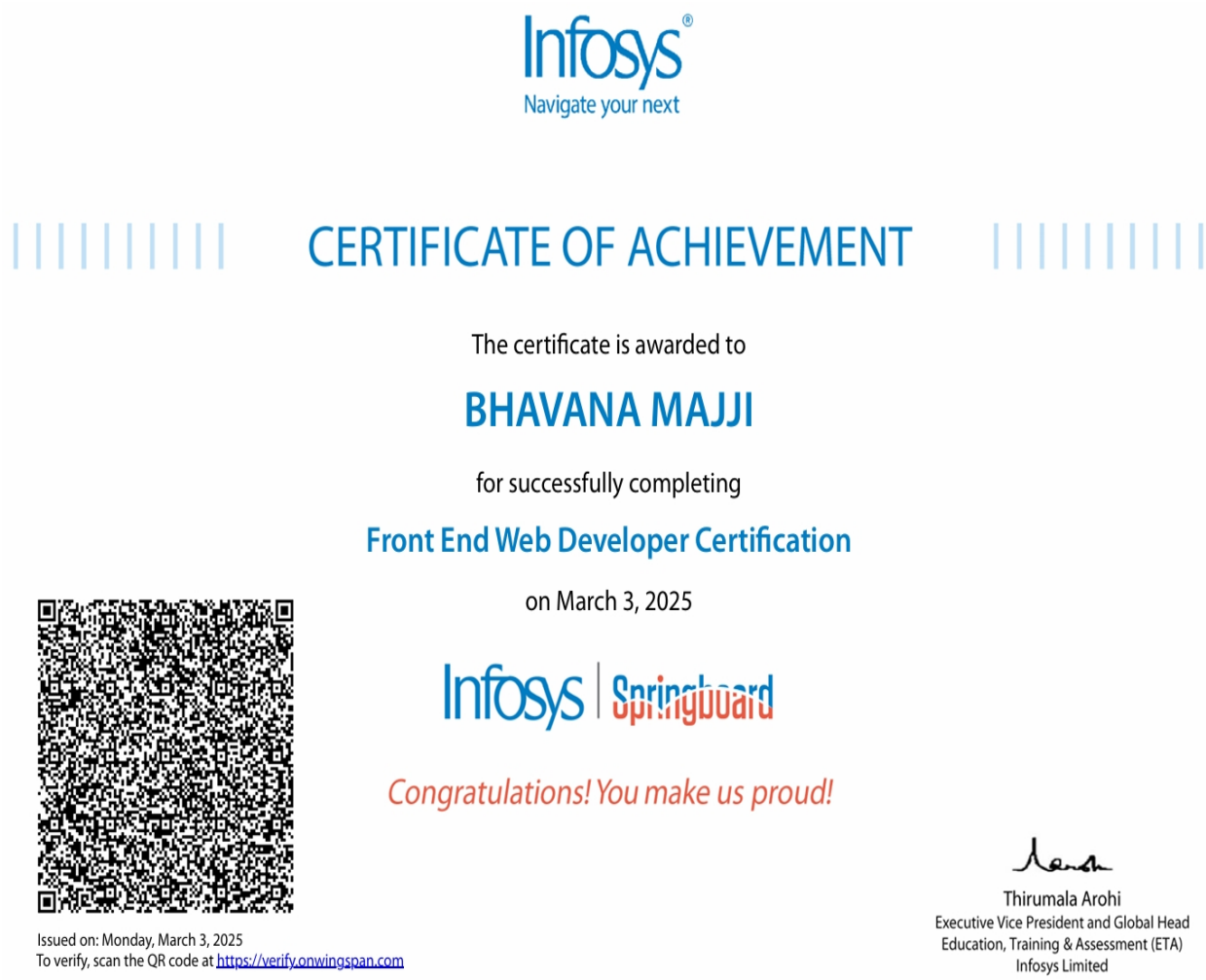
Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Emailhodit@intugvcev.edu.in

1. Name of the Laboratory :Django Framework Lab
2. Name of the Student : M. Bhavana
3. Roll No :24VV5A1271
4. Class : II B-Tech II Semester
5. Academic Year :2024-25
6. Name of Experiment :Front End Web Developer Certification
7. Date of Experiment :04-04-2025
8.Date of Submission of Report :04-04-2025

S. No	**Ability And Activity**	**Weightage Of Marks**	**Day To Day Evaluation 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:

Certification:



Infosys
Navigate your next

CERTIFICATE OF ACHIEVEMENT

The certificate is awarded to

BHAVANA MAJJI

for successfully completing

Front End Web Developer Certification

on March 3, 2025

Infosys | Springboard

Congratulations! You make us proud!

Thirumala Arohi
Executive Vice President and Global Head
Education, Training & Assessment (ETA)
Infosys Limited

Issued on: Monday, March 3, 2025
To verify, scan the QR code at https://verify.onwingspan.com