COURSE MATERIAL REPOSITORY SYSTEM

By

PISINI.LALITHYA

(Regd. No. 23VV1A1242)

Batch 11

Malavathu Dharani

Yelleti Bhavanikumar

Venna Surendra Reddy

Under the supervision of

MRS.MADHUMITA CHANDA

Department of Information Technology JNTU-GV CEV

Vizianagaram-533003

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJADA VIZIANAGARAM

VIZIANAGARAM – 533003

ANDHRA PRADESH

JNTU-GURAJADA VIZIANAGARAM COLLEGE

OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Regd.No : 23VV1A1242

CERTIFICATE

This is to certify that this is a bonafide record of practical work done by Ms.PISINI LALITHYA of II B.Tech II Semester Class in DJANGO FRAMEWORK and Innovation Lab during the year 2024-25.

No.of Tasks Completed and Certified:

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

INDEX-DANGO FRAME WORK

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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 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**

Understanding Django and its Libraries

PYTHON LIBRARIES

1. Python Collections - Container Datatypes:

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

2. Key Types:

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

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

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

   iv. **Dictionary**: Unordered, key-value pairs, fast lookups.

3. **Common Use:** Data manipulation, storing and accessing collections of data in web apps (like user data or API responses).

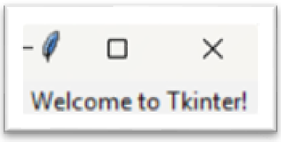
2. Tinker

1. **Purpose**: Python's standard library for creating graphical user interfaces (GUIs).
2. **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.

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:


Welcome to Tkinter!

3.Requests - HTTP Requests:

1. **Purpose**: Simplifies HTTP requests to interact with web APIs.

2. Key Features:

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

   b) Handle request parameters, headers, and cookies.

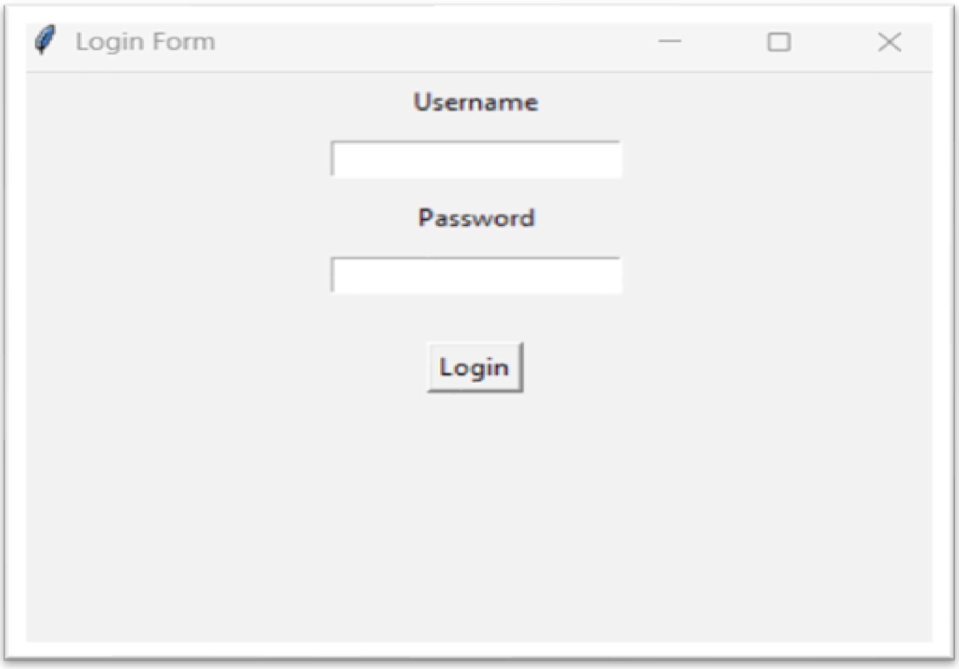
   c) Simple error handling and response handling.

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

Code:

```
    from tkinter import Tk, Label, Entry, Button
    def login():
        username = username_entry.get()
        password = password_entry.get()
        print(f"Username: {username}, Password: {password}")  # Placeholder for real login logic
    # Create main window
    root = Tk()
    root.title("Login Form")
    root.geometry("300x200")  # Set size of the window
    # Username Label and Entry
    Label(root, text="Username", font=('Arial', 10, 'bold')).pack(pady=(10, 0))
    username_entry = Entry(root, width=30)
    username_entry.pack(pady=(5, 10))
    # Password Label and Entry
    Label(root, text="Password", font=('Arial', 10, 'bold')).pack()
    password_entry = Entry(root, show="*", width=30)
    password_entry.pack(pady=(5, 10))
    # Login Button
    Button(root, text="Login", width=10, command=login).pack(pady=10)
    # Run the application
root.mainloop()
```

Output:



4. Scrapy:

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

2. 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.

3. **Common Use**: Web crawling and scraping projects that require high performance.

5.BeautifulSoup4 - Web Scraping:

1. **Purpose:** Parses HTML and XML documents to extract data.

2. Key Features:

   i. Easy navigation and searching within HTML.
   ii. Supports different parsers like html.parser, lxml, and html5lib.
   iii. **Common Use**: Extract data from websites for analysis, e.g., for building data-driven application

Code:

```
import requests
from bs4 import BeautifulSoup
def scrape_quotes():
    base_url = "http://quotes.toscrape.com"
    next_page = "/"
    while next_page:
        response = requests.get(base_url + next_page)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, "html.parser")
            quotes = soup.find_all("span", class_="text")
            authors = soup.find_all("small", class_="author")
            for quote, author in zip(quotes, authors):
                print(f'"{quote.text}" - {author.text}\n')
            next_btn = soup.find("li", class_="next")
            next_page = next_btn.a["href"] if next_btn else None
        else:
            print(f"Failed to fetch webpage. Status code: {response.status_code}")
            break
# Run the scraper
scrape_quotes()
```

Output:

```
(myenv) C:\Users\Lenovo>python -u "c:\Users\Lenovo\import requests.py"

"“The world as we have created it is a process of our thinking. It cannot be changed
    without changing our thinking.”" - Albert Einstein

"“It is our choices, Harry, that show what we truly are, far more than our abilities.”" -
    J.K. Rowling

"“There are only two ways to live your life. One is as though nothing is a miracle. The
    other is as though everything is a miracle.”" - Albert Einstein

"“The person, be it gentleman or lady, who has not pleasure in a good novel, must be
    intolerably stupid.”" - Jane Austen

"“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous
    than absolutely boring.”" - Marilyn Monroe

"“Try not to become a man of success. Rather become a man of value.”" - Albert Einstein

"“It is better to be hated for what you are than to be loved for what you are not.”" -
    André Gide

"“I have not failed. I've just found 10,000 ways that won't work.”" - Thomas A. Edison

"“A woman is like a tea bag; you never know how strong it is until it's in hot water.”" -
    Eleanor Roosevelt

"“A day without sunshine is like, you know, night.”" - Steve Martin
```

6. Zappa:

1. **Purpose**: Deploy Python web applications to AWS Lambda and API Gateway.

2. Key Features:

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

   ii. Manages serverless architecture and deployment configurations.

3. **Common Use:** Build scalable, serverless web apps without maintaining servers.

7.Dash:

1. **Purpose:** Web application framework for building interactive data visualization applications.

2. Key Features:

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

   ii. Integrates seamlessly with data science libraries (e.g., Pandas, Plotly).

3.Common Use: Building dashboards and data-driven web applications.

Turbo Gears

1. **Purpose:** Full-stack web framework built on top of WSGI.

2. Key Features:

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

   ii. Focus on rapid development and scalability.

3.Common Use: Develop scalable, enterprise-level web applications.

8.CherryPy:

1.Purpose: Minimalistic web framework for building web applications.

2.Key Features:

   i. Provides a simple and fast HTTP server.

   ii. Handles routing, cookies, sessions, and file uploads.

3.Common Use: Building web applications with a lightweight framework.

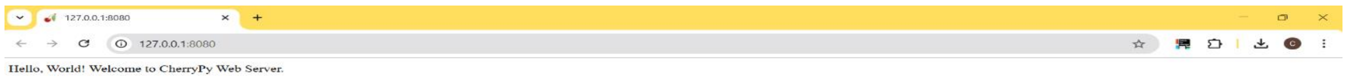
Code:

```
import cherrypy
class HelloWorld:
    @cherrypy.expose  # Exposes this method as a web page
    def index(self):
        return "Hello, World! Welcome to CherryPy Web Server."
# Configure and start the CherryPy server
if __name__ == "__main__":
    cherrypy.quickstart(HelloWorld(), "/", config={
        "global": {
            "server.socket_host": "127.0.0.1",  # Localhost
            "server.socket_port": 8080,         # Port number
        }
    })
```

Output:

```
        (myenv) C:\Users\Lenovo>python -u "c:\Users\Lenovo\import requests.py"
        [10/Apr/2025:01:34:09] ENGINE Listening for SIGTERM.
        [10/Apr/2025:01:34:09] ENGINE Bus STARTING
        [10/Apr/2025:01:34:09] ENGINE Started monitor thread 'Autoreloader'.
        [10/Apr/2025:01:34:09] ENGINE Serving on http://127.0.0.1:8080
[10/Apr/2025:01:34:09] ENGINE Bus STARTED
```

After run the server :-


127.0.0.1:8080

Hello, World! Welcome to CherryPy Web Server.

9.Flask:

1. **Purpose:** Lightweight micro-framework for building web applications.

2. Key Features:

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

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

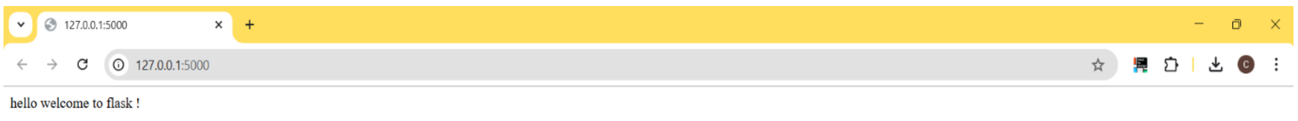
Code:

```
from flask import Flask
app = Flask(__name__)
@app.route('/', methods=['GET'])
def hellouser():
    return "Hello, welcome to Flask!"
if __name__ == '__main__':
    app.run(debug=True)
```

```
        (myenv) C:\Users\Lenovo> * Serving Flask app 'import requests'
         * 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: 134-121-940
```

Output:

After run the server:



127.0.0.1:5000

hello welcome to flask !

10.Web2Py:

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

2.Key Features:

i. Includes a web-based IDE for development.

ii. Built-in ticketing system and database integration.

3.Common Use: Enterprise web applications with minimal setup.

11.Bottle:

1.Purpose: Simple and lightweight WSGI micro-framework.

2.Key Features:

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

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

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

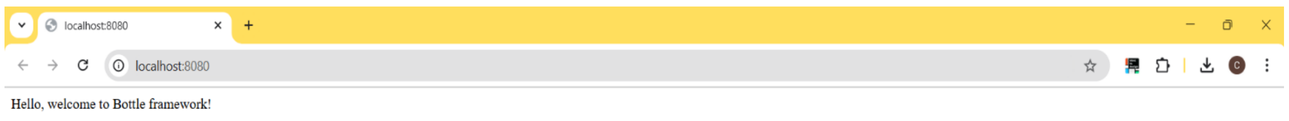
Code:

```
from bottle import Bottle, run
app = Bottle()
@app.route('/')
def home():
    return "Hello, welcome to Bottle framework!"
if __name__ == '__main__':
    run(app, host='localhost', port=8080, debug=True)
```

output:

```
        (myenv) C:\Users\Lenovo>python -u "c:\Users\Lenovo\import requests.py"
        Bottle v0.13.2 server starting up (using WSGIRefServer())...
        Listening on http://localhost:8080/
Hit Ctrl-C to quit.
```

After run the server



Hello, welcome to Bottle framework!

12.Falcon:

1.Purpose: High-performance framework for building APIs.

2.Key Features:

i. Focuses on speed and minimalism.

ii. Supports RESTful API development and is optimized for large-scale deployments.

3.Common Use: Building fast, high-performance APIs.

13.CubicWeb:

1.Purpose: Web application framework based on an entity-relation model.

2.Key Features:

i. Uses a highly modular architecture for development.

ii. Focus on building web apps with rich data models.

3.Common Use: Semantic web applications or data-driven web apps.

14.Quixote:

1.Purpose: A web framework designed for simplicity and scalability.

2.Key Features:

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

ii. Easily extensible, with minimalistic core.

3.Common Use: Scalable and customizable web applications.

15.Pyramid:

1.Purpose: Full-stack web framework can scale from simple complex application.

2.Key Features:

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

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

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

SUMMARY:

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

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

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

4.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 **Email: hod.it@intugvcev.edu.in**

1. Name of the Laboratory	: Django FrameWork lab
2. Name of the Student	: Pisini.Lalithya
3. Roll No	: 23VV1A1242
4. Class	: II B.TECH II SEMESTER
5. Academic Year	: 2024-25
6. Name of Experiment	: Introduction to Django Framework
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 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**

Introduction to Django Framework

Django: Django is a high-level Python web framework that allows developers to build secure, scalable, and maintainable web applications quickly and efficiently. It follows the Model-View-Template (MVT) architectural pattern.

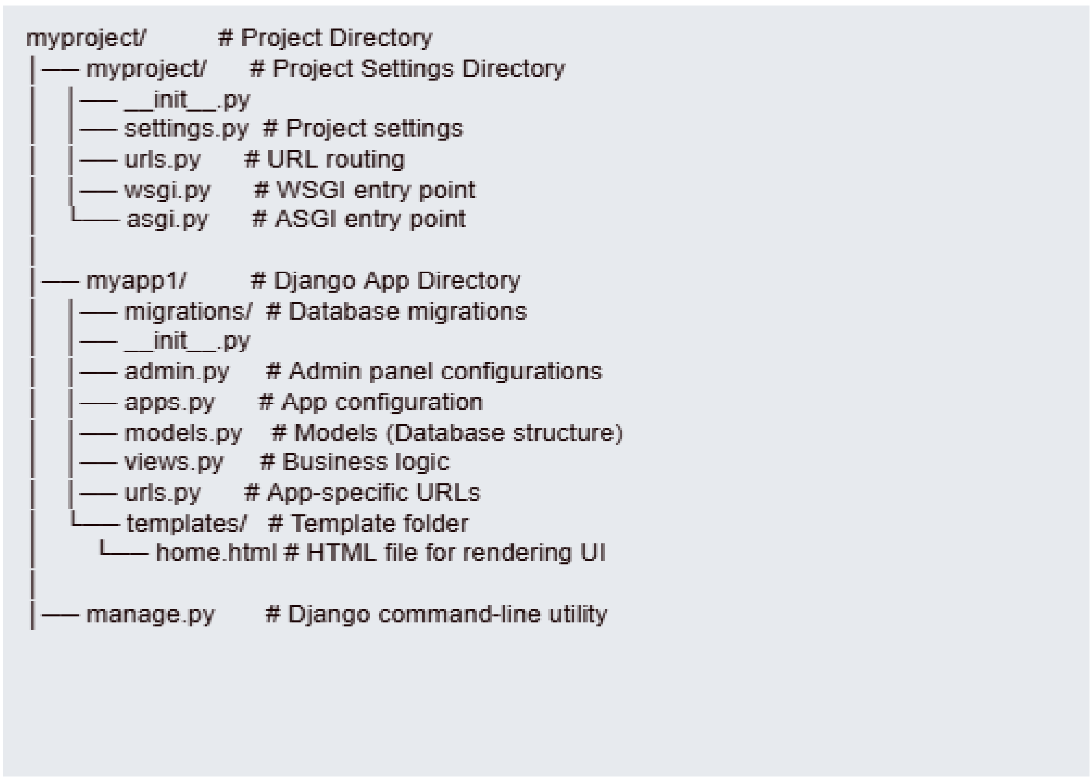
Key Features of Django:

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.

Example MVT Folder Structure in Django



```
myproject/         # Project Directory
|── myproject/      # Project Settings Directory
|    |── __init__.py
|    |── settings.py  # Project settings
|    |── urls.py      # URL routing
|    |── wsgi.py      # WSGI entry point
|    └── asgi.py      # ASGI entry point
|
|── myapp1/         # Django App Directory
|    |── migrations/  # Database migrations
|    |── __init__.py
|    |── admin.py     # Admin panel configurations
|    |── apps.py      # App configuration
|    |── models.py    # Models (Database structure)
|    |── views.py     # Business logic
|    |── urls.py      # App-specific URLs
|    └── templates/   # Template folder
|        └── home.html # HTML file for rendering UI
|
|── manage.py       # Django command-line utility
```

Templates:

Django templates are text files that use the Django template language to separate the design of a website from the underlying code. They can generate HTML,XML,CSV, and other text-based formats**.**

How templates work:

1. **Variables:**
   Replace values into the template when it's evaluated
2. **Tags:**
   Control the logic of the template, such as loops, database queries, and access to other template tags
3. **Filters:**

   Transform the values of variables and tag arguments

4. **Context:**
   A dictionary-like structure that stores the data to be displayed in the template

Benefits of Django templates

1. **Clean, maintainable code:**

   Keeps HTML business logic separate from the Python code

2. **Follows DRY design principle:**
   Avoids repetition while designing an application
3. **Can generate complete web pages:**
   Database queries and other data processing tasks are handled by views and models

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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	:	27-12-2024
8. Date of Submission of Report	:	03-01-2025

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**

Step by Step Guide to Installing Django

Django requires python, so first make sure python and pip (python’s package manager) are installed.

1.insatll python

Check if python is already installed:

```
Python3  --version
```

If it’s not installed, you can install it with

```
Sudo apt update

Sudo apt install python3 python3-pip
```

2. install virtual environment

```
Sudo pip3 install virtualenv
```

3.create a virtual environment

```
mkdir myproject

cd myproject

python -m env venv
```

for virtual environment activation

```
.\env\Scripts\activate
```

4. install Django

```
Pip install django
```

5. verify the installation

```
Django-admin –version
```

6. create a new Django project

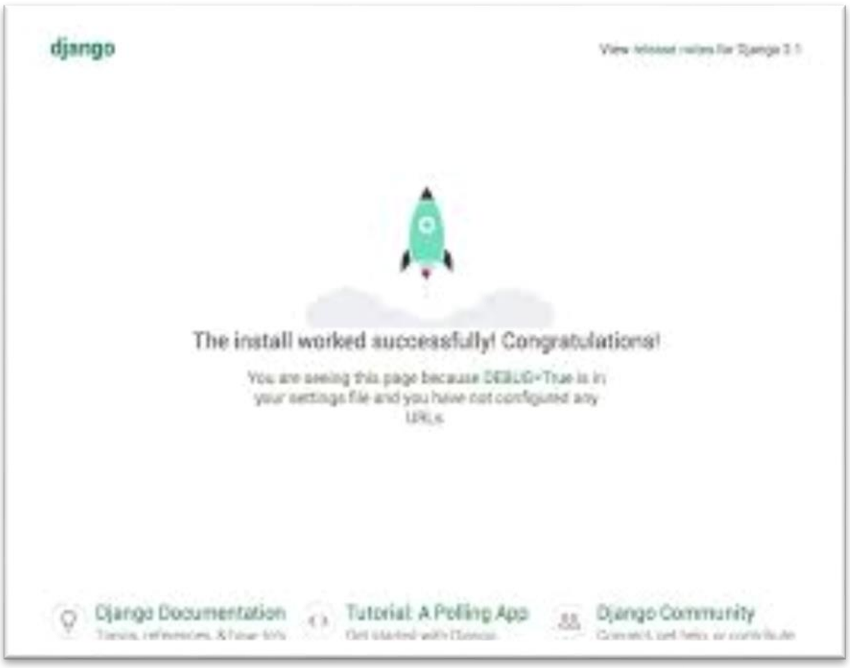
```
Django-admin startproject mysite
```

7. run development server

```
Python manage.py runserver
```

Example:

1. Python -m venv env
2. env\scripts\activate
3. pip install Django
4. django-admin satartproject myproject
5. Cd myproject
6. django-admin satartapp myapp
7. python manage.py runserver


django
View release notes for Django 3.1

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

In my project there is one file it is views.py. in this file insert the below code

```
from django.http import HttpResponse

def helloView(request):
   return HttpResponse("Hello world")
```

The function *helloView()* will be called, each time someone opens the webpage.

The function returns a *HttpResponse*, this is text that is shown in your web browser

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Linking Views and URL Configurations
7. Date of Experiment	:	03-01-2025
8. Date of Submission of Report	:	24-01-2025

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**

Linking Views and URL Configurations

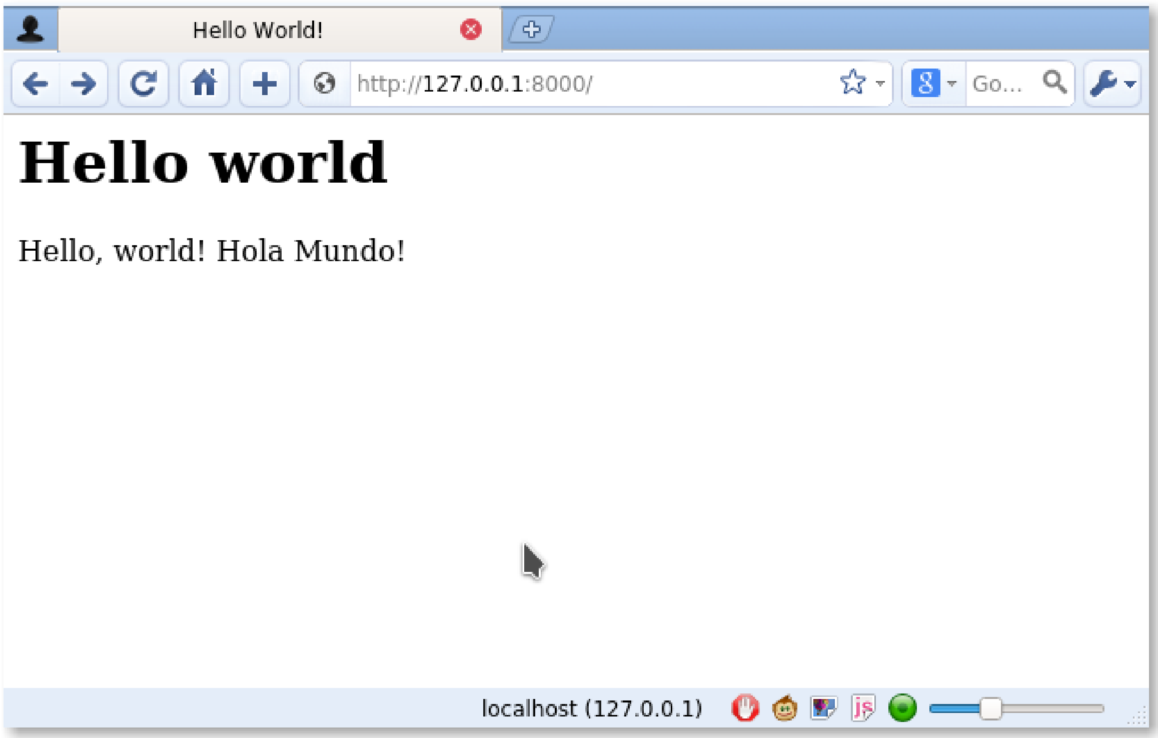
myapp/urls.py:

```
from django.contrib import admin
from django.urls import path
from .views import helloView

urlpatterns = [
    path('', helloView, name='hello')
]
```

This file maps all of the paths to the Python functions. In this case the webpage path maps to the *helloView* function.

Finally restart the server by clicking the green button on your project page.
Open the url in your web browser, and you'll see *Hello world* in the web page.

In web browser :



Hello world

Hello, world! Hola Mundo!

Problem statement:

Course Material Repository System

Objective: Develop a centralized platform where professors and students can upload, share, and access course materials.

Features:

Professors can upload lecture notes, slides, and assignments.

Students can browse, download, or upload materials.

Admin can moderate content uploads for appropriateness.

Categories and search functionality for easy access.

Frameworks: Django, SQLite, or PostgreSQL.

Design Thinking Stages:

Empathize: Interview faculty and students about their material-sharing practices.

Ideate: Brainstorm features like search filters, rating systems, and a content management system.

Prototype: Develop a system with uploading and searching capabilities.

Test: Get feedback on content organization and ease of use.

Create a App:

Before creating an app, make sure you have a Django project already set up. you can create one by following these steps:

- **Install Django:**

  ```
  pip install Django
  ```

- **Create a Django project:**

  ```
  django-admin startproject myproject
  cd myproject
  ```

- **Create app:**
  Once you're inside your Django project folder, you can create a new app by running the following command:
  ```
  python manage.py startapp myapp
  ```

project directory structure:

project1/

```
|
 ├── manage.py
└── app1/
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    ├── views.py
   └── migrations/
```

Explanation of the Structure:

project1/: This is the root directory for your Django project.

manage.py: The command-line utility for Django.

project1/: This folder contains project-level settings and configurations.

settings.py: Contains all the settings for the Django project (databases, static files, middleware, etc.).

urls.py: The root URL configurations that can include URLs for the main project and apps.

wsgi.py and asgi.py: Entry points for running your Django application.

app1/: This is the app that you created in your Django project.

admin.py: Register your models to the Django admin interface.

apps.py: Contains the app configuration.

models.py: Define the database models.

views.py: Handles the logic for the views of the app.

app1/urls.py:

the urls.py file in an app plays a crucial role in routing HTTP requests to the appropriate view functions or class-based views that handle the requests. It essentially maps URL patterns (the web addresses users will access) to specific views in your Django app.

```
# core/urls.py

from django.urls import path

from . import views

urlpatterns = [


path('', views.home, name='home'),

   path('login/', views.login, name='login'),

   path('signup/', views.signup, name='signup'),

   path('dashboard/', views.dashboard, name='dashboard'),

   path('course/', views.course, name='course'),

   path('admindashboard/', views.admin_dashboard, name='admin_dashboard'),

   path('studentdashboard/', views.student_dashboard, name='student_dashboard'),

   path('teacherdashboard/', views.teacher_dashboard, name='teacher_dashboard'),

]
```

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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	:	24-01-2025

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**

Exploring Django Views

Project1/setting.py:

Register the app in **settings.py**, Once the app is created, you need to add it to your project's **INSTALLED_APP**S in the settings.py file.

```
import os

from pathlib import Path


# Build paths inside the project like this: BASE_DIR / 'subdir'.

BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/


# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = 'django-insecure-&_@7*h3)bch+2#u12jt@xra7m$kem0v2rj0*mmc)mza741%@mi'


# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True


ALLOWED_HOSTS = []


# Application definition


INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app1',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'project1.urls'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'online_course_material_repository',
        'USER':'root',
        'PASSWORD':'lalithya',
        'HOST':'127.0.0.1',
        'PORT':'3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
    }
}}
```

```
WSGI_APPLICATION = 'project1.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases


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


# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/


LANGUAGE_CODE = 'en-us'


TIME_ZONE = 'UTC'


USE_I18N = True


USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.1/howto/static-files/


STATIC_URL = 'static/'


# Default primary key field type
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field


DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Project1/urls.py:

The URL configuration of a Django project determines how URLs are mapped to the views and which part of your application should handle each request.

```
from django.contrib import admin

from django.urls import path, include

urlpatterns = [

   path('admin/', admin.site.urls),

   path('', include('app1.urls')),  # Including the core app's URLs

]
```

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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 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**

Setting Up App-Level URLs

app1/views.py:

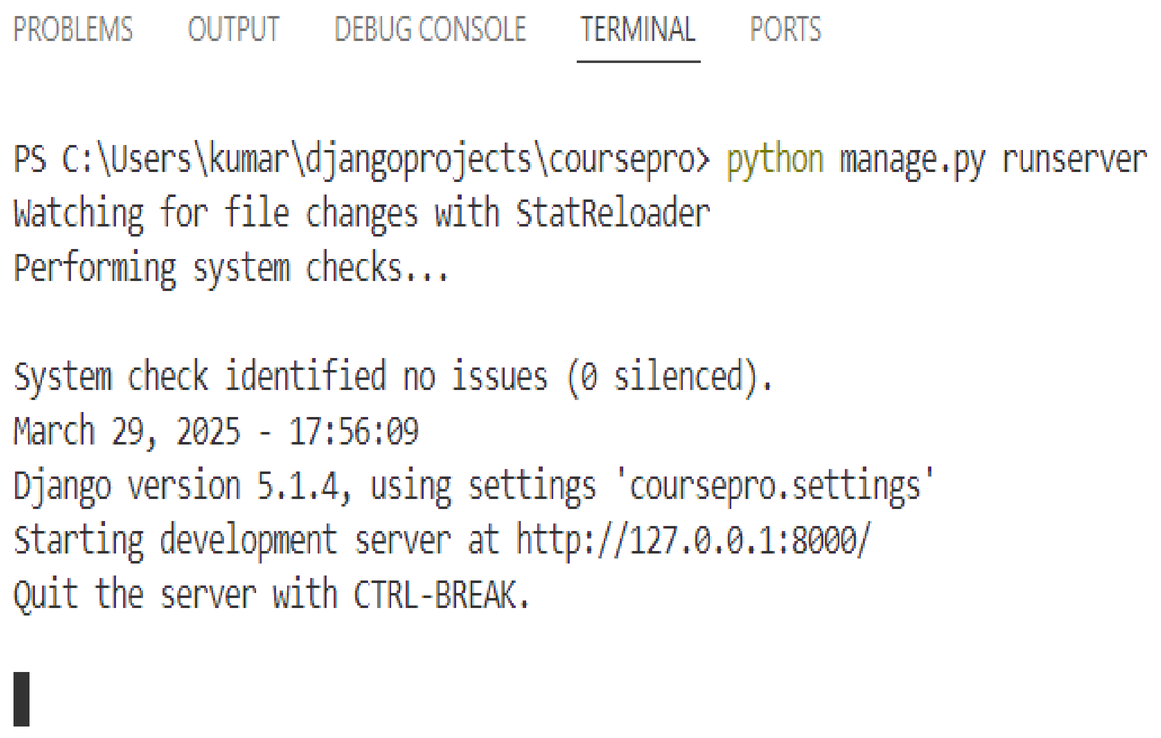
views are the central component that handle the business logic of your web application. It can interact with the **models** to fetch, create, update, or delete data from the database.

```
# core/views.py
from django.shortcuts import render
def home(request):
    return render(request, 'home.html')
def login(request):
    return render(request, 'login.html')
def signup(request):
    return render(request, 'signup.html')
def dashboard(request):
    return render(request, 'dashboard.html')
def course(request):
    return render(request, 'course.html')
def admin_dashboard(request):
    return render(request, 'admindash.html')
def student_dashboard(request):
    return render(request, 'student.html')
def teacher_dashboard(request):
    return render(request, 'teachdash.html')
```

Run the server:

Using the following command we can run the server

Python manage.py runserver



```
PS C:\Users\kumar\djangoprojects\coursepro> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 29, 2025 - 17:56:09
Django version 5.1.4, using settings 'coursepro.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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	:	31-02-2025
8. Date of Submission of Report	:	17-02-2025

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**

Working with Templates in Django

templates/home.html:

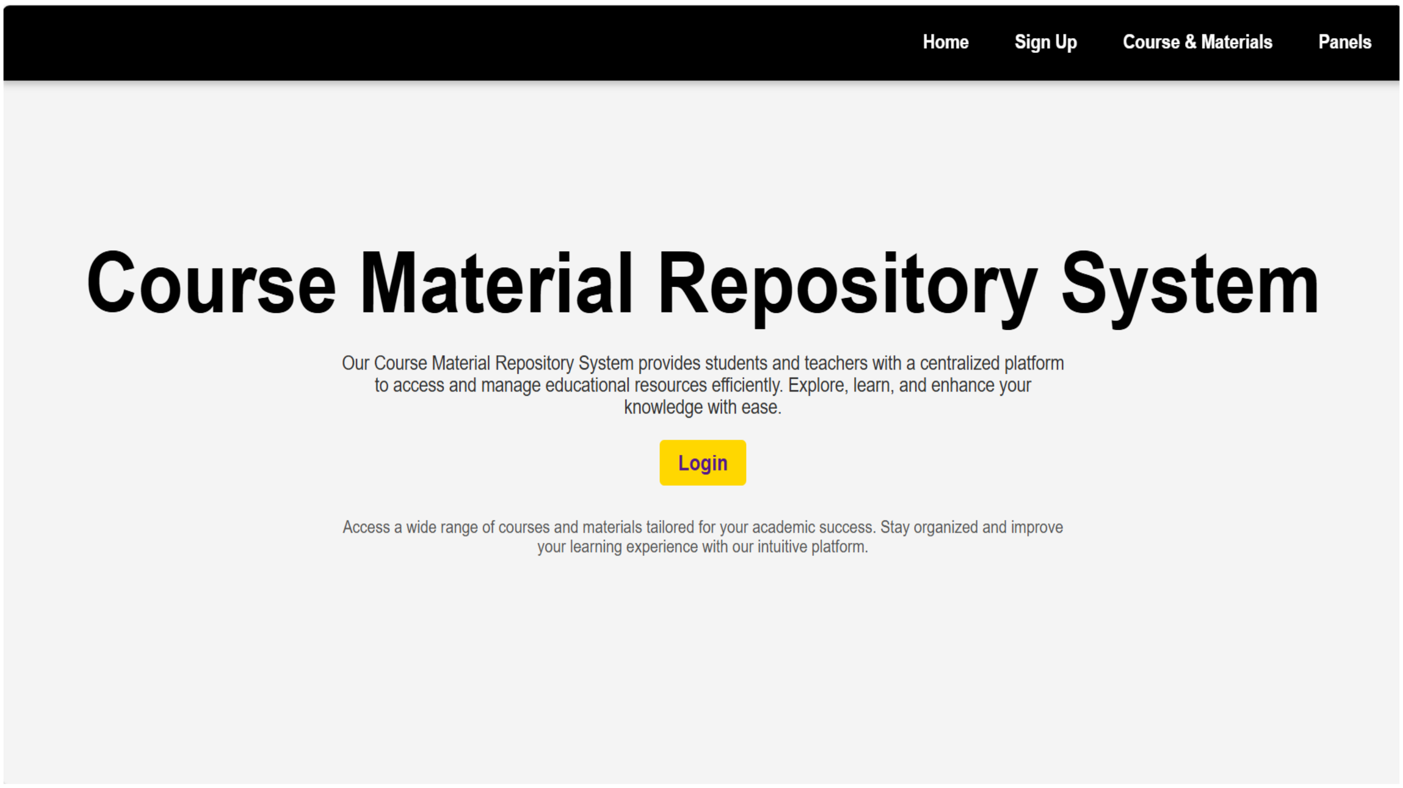
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Course Material Repository</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: Arial, sans-serif;
            list-style: none;
            text-decoration: none;
        }
        body {
            background: #f4f4f4;
            padding: 20px;
            position: relative;
            display: flex;
            flex-direction: column;
            align-items: center;
            height: 100vh;
            justify-content: center;
        }
        .navbar {
            display: flex;
            justify-content: flex-end;
```

```
gap: 20px;
        background: black;
        padding: 15px;
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
        padding-right: 20px;
    }
    .navbar a {
        color: white;
        text-decoration: none;
        font-size: 18px;
        font-weight: bold;
        padding: 10px 15px;
        border-radius: 5px;
        transition: background 0.3s;
    }
    .navbar a:hover {
        background: gold;
        color: black;
    }
    .header {
        text-align: center;
        font-size: 80px;
        font-weight: bold;
    }
    .description {
        margin-top: 20px;
```

```
            text-align: center;
            font-size: 18px;
            max-width: 800px;
            color: #333;
        }
        .login-button {
            margin-top: 20px;
            background: gold;
            color: black;
            font-size: 20px;
            font-weight: bold;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            transition: background 0.3s;
        }
        .login-button:hover {
            background: darkgoldenrod;
        }
        .extra-content {
            margin-top: 30px;
            text-align: center;
            font-size: 16px;
            max-width: 800px;
            color: #555;
        }
    </style>
</head>
<body>
```

```
  <div class="navbar">
    <a href="home.html">Home</a>
    <a href="signup.html">Sign Up</a>
    <a href="course.html">Course & Materials</a>
    <a href="dashboard.html">Panels</a>
  </div>
  <div class="header">Course Material Repository System</div>
  <p class="description">Our Course Material Repository System provides students and teachers with a centralized platform to access and manage educational resources efficiently. Explore, learn, and enhance your knowledge with ease.</p>
  <button class="login-button">
    <a href="login.html">login</a>
  </button>
  <p class="extra-content">Access a wide range of courses and materials tailored for your academic success. Stay organized and improve your learning experience with our intuitive platform.</p>
</body>
</html>
```

Output:



Course Material Repository System

Our Course Material Repository System provides students and teachers with a centralized platform to access and manage educational resources efficiently. Explore, learn, and enhance your knowledge with ease.

Access a wide range of courses and materials tailored for your academic success. Stay organized and improve your learning experience with our intuitive platform.

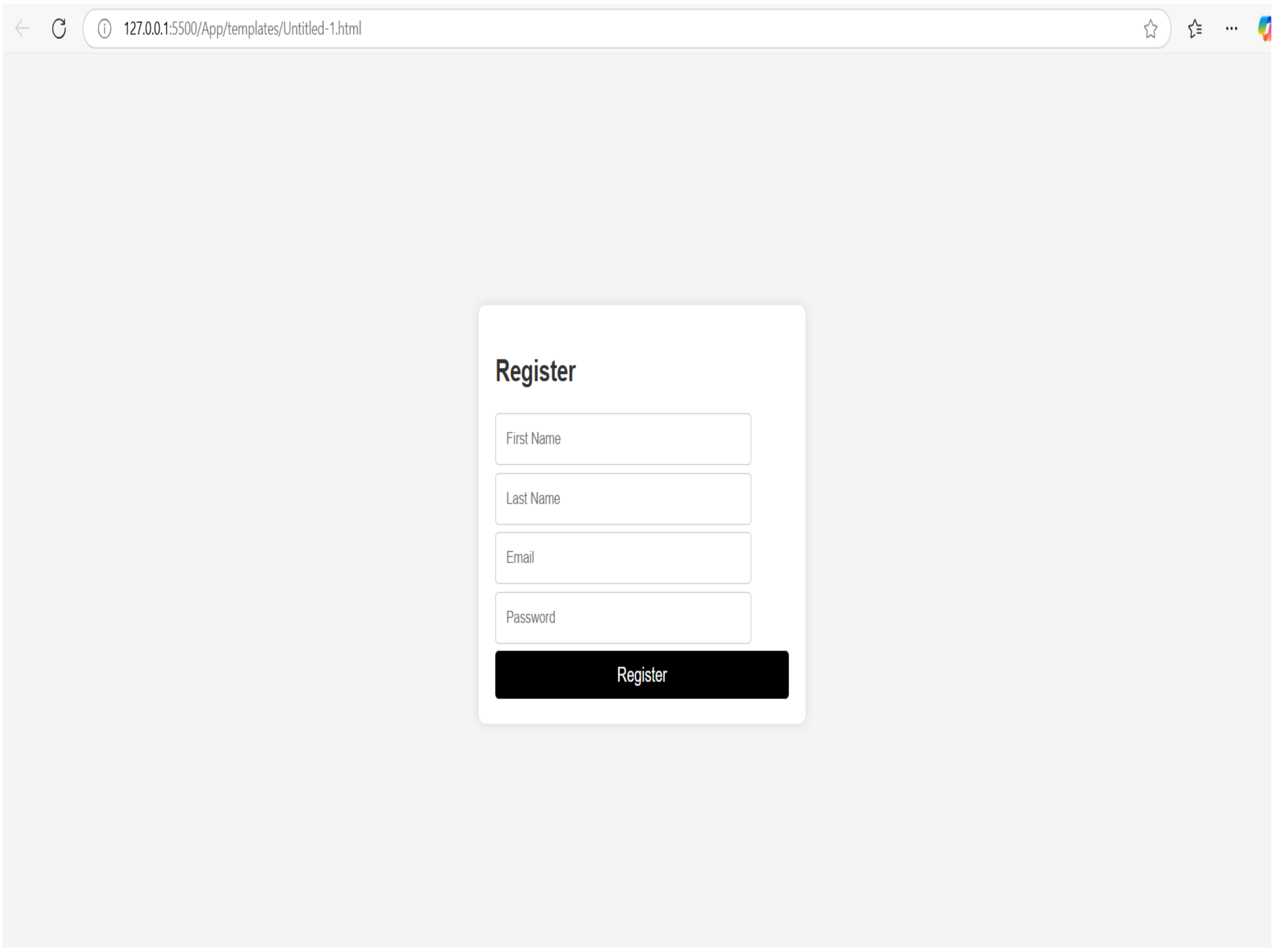
templates/signup:

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
```

```
}
.registration-form {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 350px;
}
.registration-form h2 {
    margin-bottom: 20px;
    font-size: 24px;
    color: #333;
}
.registration-form input[type="text"],
.registration-form input[type="email"],
.registration-form input[type="password"] {
    width: 100%;
    padding: 12px;
    margin: 6px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
    display: block;
}
.registration-form input[type="submit"] {
    width: 100%;
    padding: 10px;
    background-color: #000000;
    border: none;
    border-radius: 4px;
    color: #fff;
```

```
          font-size: 16px;
          cursor: pointer;
       }
       .registration-form input[type="submit"]:hover {
          background-color: #333333;
       }
   </style>
</head>
<body>
   <div class="registration-form">
      <h2>Register</h2>
       <form action="/submit_registration" method="post">
          <input type="text" name="firstname" placeholder="First Name" required style="width: 80%;">
          <input type="text" name="lastname" placeholder="Last Name" required style="width: 80%;">
          <input type="email" name="email" placeholder="Email" required style="width: 80%;">
             <input type="password" name="password" placeholder="Password" required style="width:
80%;">
          <input type="submit" value="Register">
       </form>
   </div>
</body>
</html>
```

Output:



templates/login.html:

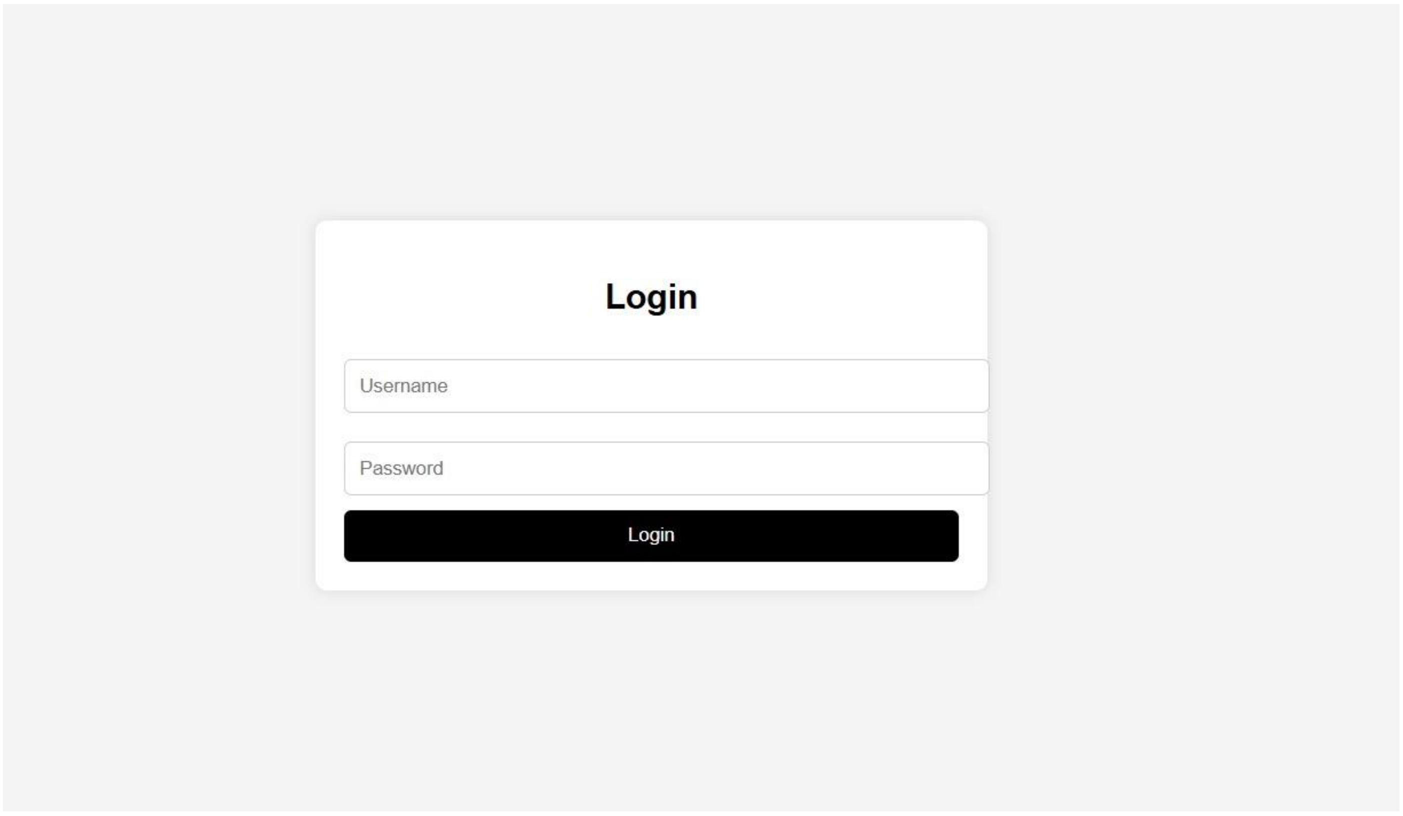
```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Login Page</title>

  <style>

    body {
```

```
    text-decoration: none;
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}
.login-container {
    background: white;
    padding: 20px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    border-radius: 5px;
    width: 400px;
    text-align: center;
}
input {
    width: 90%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    width: 100%;
    padding: 10px;
    background-color: #000000;
    color: white;
    border: none;
    border-radius: 5px;
```

```
          cursor: pointer;
        }
        button:hover {
          background-color: #333333;
        }
        .error {
          color: red;
          font-size: 14px;
        }
        .registration-link {
          margin-top: 20px;
          font-size: small;
        }
        .registration-link a {
          color: #007bff;
          text-decoration: none;
        }
        .registration-link a:hover {
          color: #000000;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login</h2>
        <input type="text" id="username" placeholder="Username">
        <input type="password" id="password" placeholder="Password">
        <p id="error" class="error"></p>
        <button onclick="validateLogin()">Login</button>
        <div class="registration-link">
```

```
            <a href="signup.html">Don't have an account? Register here</a>
        </div>
    </div>

    <script>
        function validateLogin() {
            var username = document.getElementById('username').value;
            var password = document.getElementById('password').value;
            var error = document.getElementById('error');

            if (username === "admin" && password === "password123") {
                alert("Login successful!");
            } else {
                error.textContent = "Invalid username or password";
            }
        }
    </script>
</body>
</html>
```

Output:

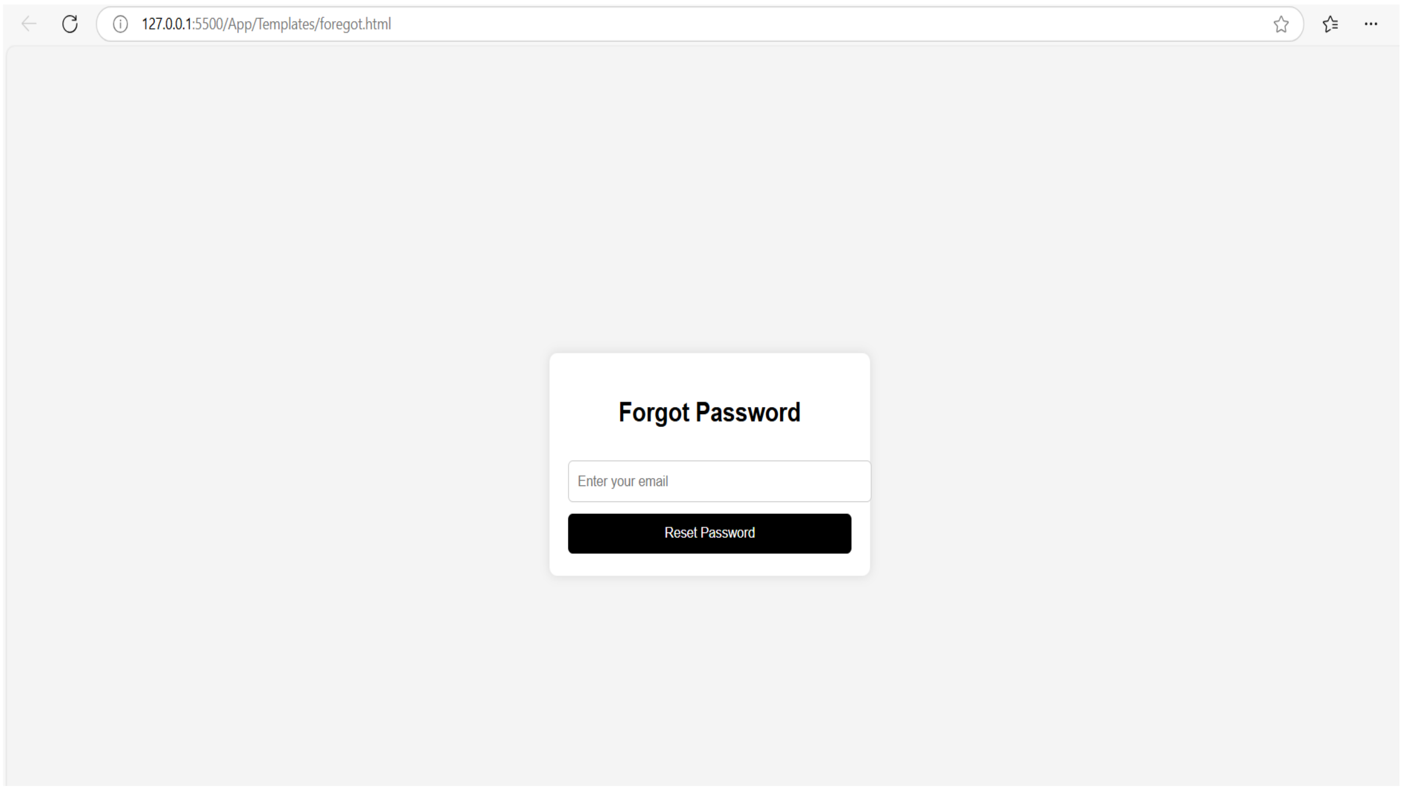


templates/forget.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Forgot Password</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      background-color: #f4f4f4;

    }

    .forgot-password-container {
        background: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        text-align: center;
    }
    input {
        width: 100%;
        padding: 10px;
        margin: 10px 0;
        border: 1px solid #ccc;
        border-radius: 5px;
    }
    button {
        width: 100%;
        padding: 10px;
        background: #000000;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }
    button:hover {
        background: #333333;
    }
</style>
<script>
    function resetPassword(event) {
        event.preventDefault();
        let email = document.getElementById("email").value;
```

```
            if (email === "") {
                alert("Please enter your email address.");
            } else {
                alert("Password reset link has been sent to your email.");
            }
        }
    </script>
</head>
<body>
    <div class="forgot-password-container">
        <h2>Forgot Password</h2>
        <form onsubmit="resetPassword(event)">
            <input type="email" id="email" placeholder="Enter your email" required>
            <button type="submit">Reset Password</button>
        </form>
    </div>
</body>
</html>
```

Output:

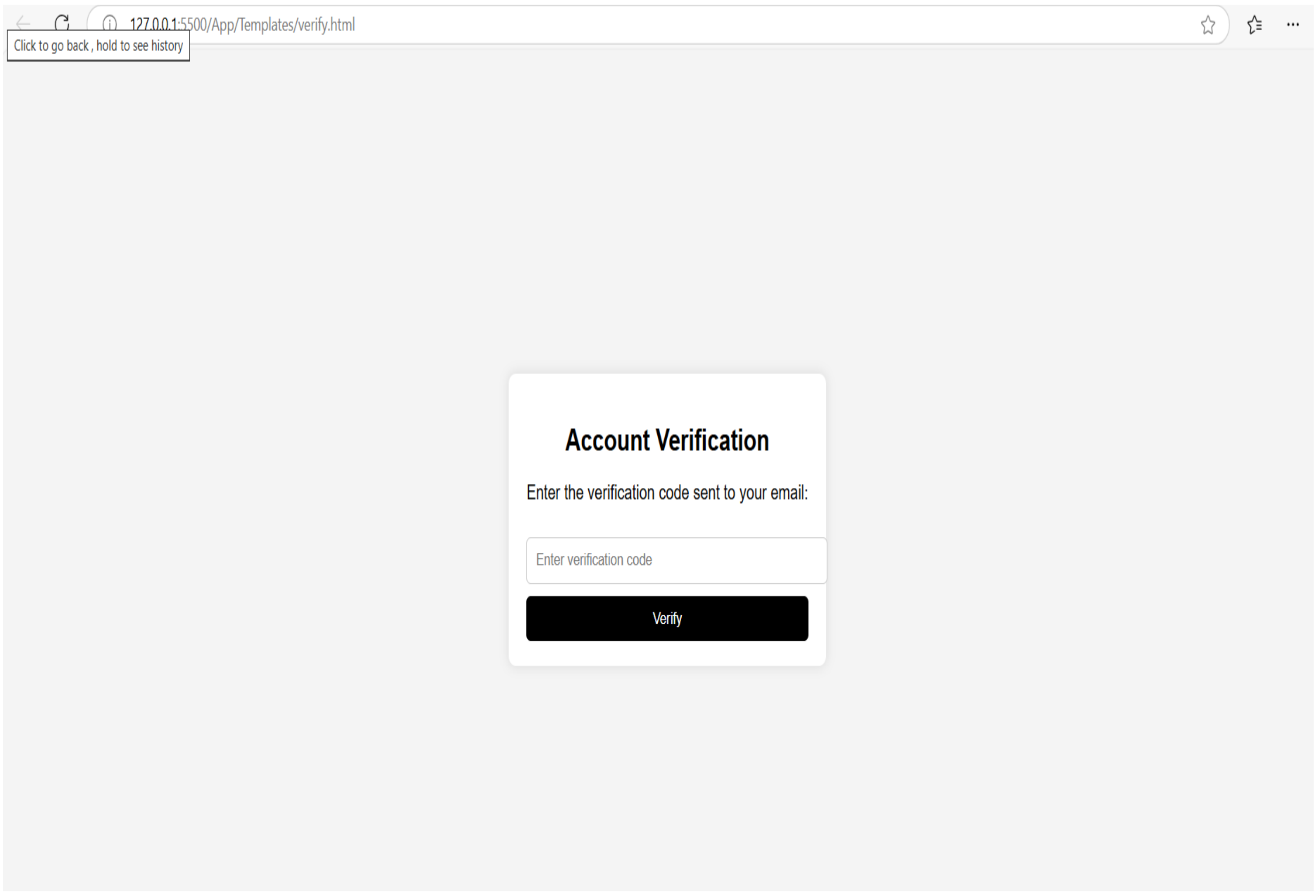

127.0.0.1:5500/App/Templates/foregot.html
Forgot Password
Enter your email
Reset Password

templates/verify.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Account Verification</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f4;
    }
    .verification-container {
      background: white;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
      text-align: center;
    }
    input {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
```

```
        button {
            width: 100%;
            padding: 10px;
            background: #000000;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background: #333333;
        }
    </style>
    <script>
        function verifyAccount(event) {
            event.preventDefault();
            let code = document.getElementById("verification-code").value;

            if (code === "") {
                alert("Please enter the verification code sent to your email.");
            } else {
                alert("Account verified successfully!");
            }
        }
    </script>
</head>
<body>
    <div class="verification-container">
        <h2>Account Verification</h2>
        <p>Enter the verification code sent to your email:</p>
        <form onsubmit="verifyAccount(event)">
          <input type="text" id="verification-code" placeholder="Enter verification code" required>
          <button type="submit">Verify</button>
        </form>
    </div>
</body>
</html>
```

Output:


127.0.0.1:5500/App/Templates/verify.html
Click to go back , hold to see history
Account Verification
Enter the verification code sent to your email:
Enter verification code
Verify

templates/dashboard.html:

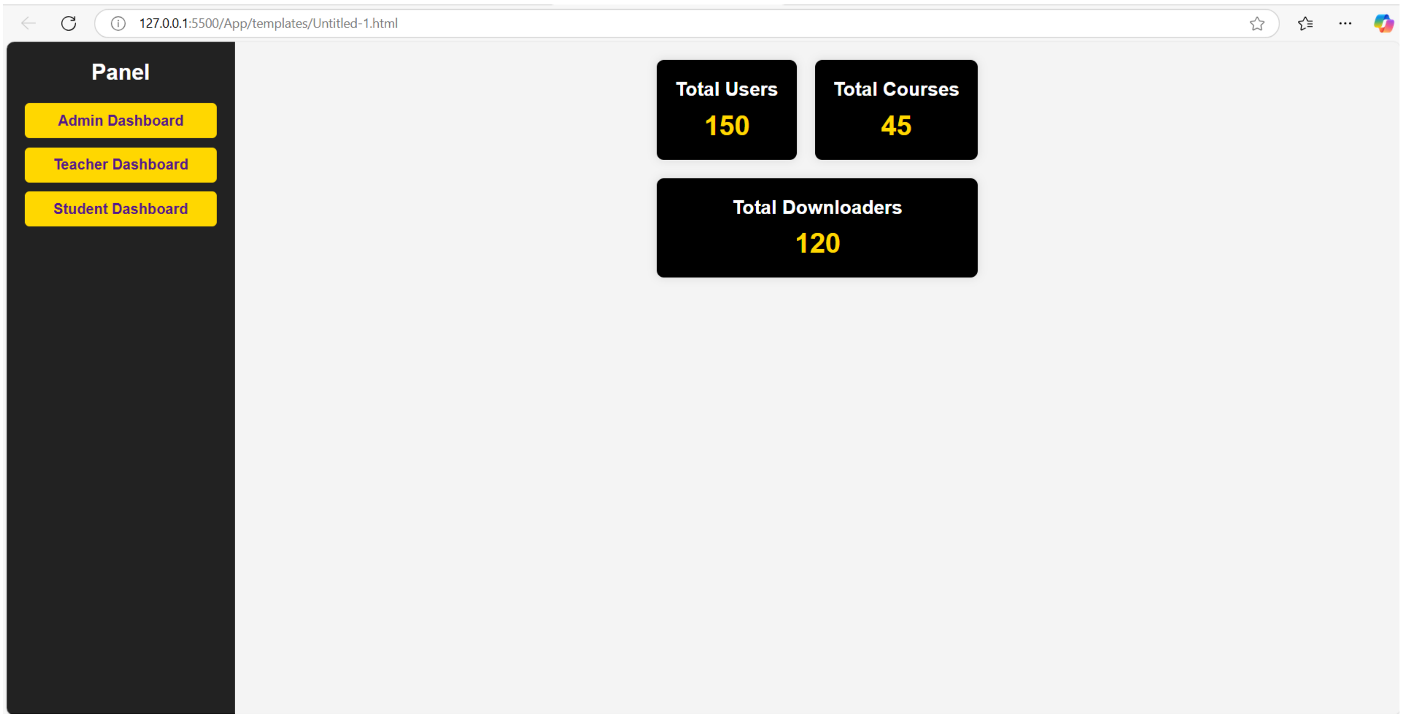
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Dashboard</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: Arial, sans-serif;
    }
    body {
      display: flex;
      min-height: 100vh;
      background: #f4f4f4;
    }
    .sidebar {
      width: 250px;
      background: #222;
      color: white;
      padding: 20px;
      position: fixed;
      height: 100%;
    }
    .sidebar h2 {
      text-align: center;
      margin-bottom: 20px;
```

```
}
.sidebar ul {
   list-style: none;
}
.sidebar ul li {
   padding: 10px;
   cursor: pointer;
   border-radius: 5px;
   margin-bottom: 10px;
   background: gold;
   text-align: center;
   color: black;
   font-weight: bold;
}
.sidebar ul li:hover {
   background: #daa520;
}
.main-content {
   margin-left: 250px;
   flex: 1;
   padding: 20px;
}
.stats {
   display: grid;
   grid-template-areas:
      "users courses"
      "downloaders downloaders";
   gap: 20px;
   justify-content: center;
   align-items: center;
```

```
        }
        .stat-card {
            background: black;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            font-size: 18px;
            color: white;
            text-align: center;
            font-weight: bold;
        }
        .stat-card p {
            font-size: 30px;
            font-weight: bold;
            margin-top: 10px;
            color: gold;
        }
        .stat-card:nth-child(1) { grid-area: users; }
        .stat-card:nth-child(2) { grid-area: courses; }
        .stat-card:nth-child(3) { grid-area: downloaders; }
    </style>
</head>
<body>
    <div class="sidebar">
        <h2>Panel</h2>
        <ul>
            <ul>
                <ul>
                    <li><a href="admindash.html" style="text-decoration: none;">Admin
Dashboard</a></li>
```

```
                        <li><a href="teachdesh.html" style="text-decoration: none;">Teacher Dashboard</a></li>

                        <li><a href="studentdash.html" style="text-decoration: none;">Student Dashboard</a></li>

            </ul>


          </ul>


       </ul>

    </div>

    <div class="main-content">

       <div class="stats">

          <div class="stat-card">

             <h3>Total Users</h3>

             <p>150</p>

          </div>

          <div class="stat-card">

             <h3>Total Courses</h3>

             <p>45</p>

          </div>

          <div class="stat-card">

             <h3>Total Downloaders</h3>

             <p>120</p>

          </div>

       </div>

    </div>

</body>
```

Output:



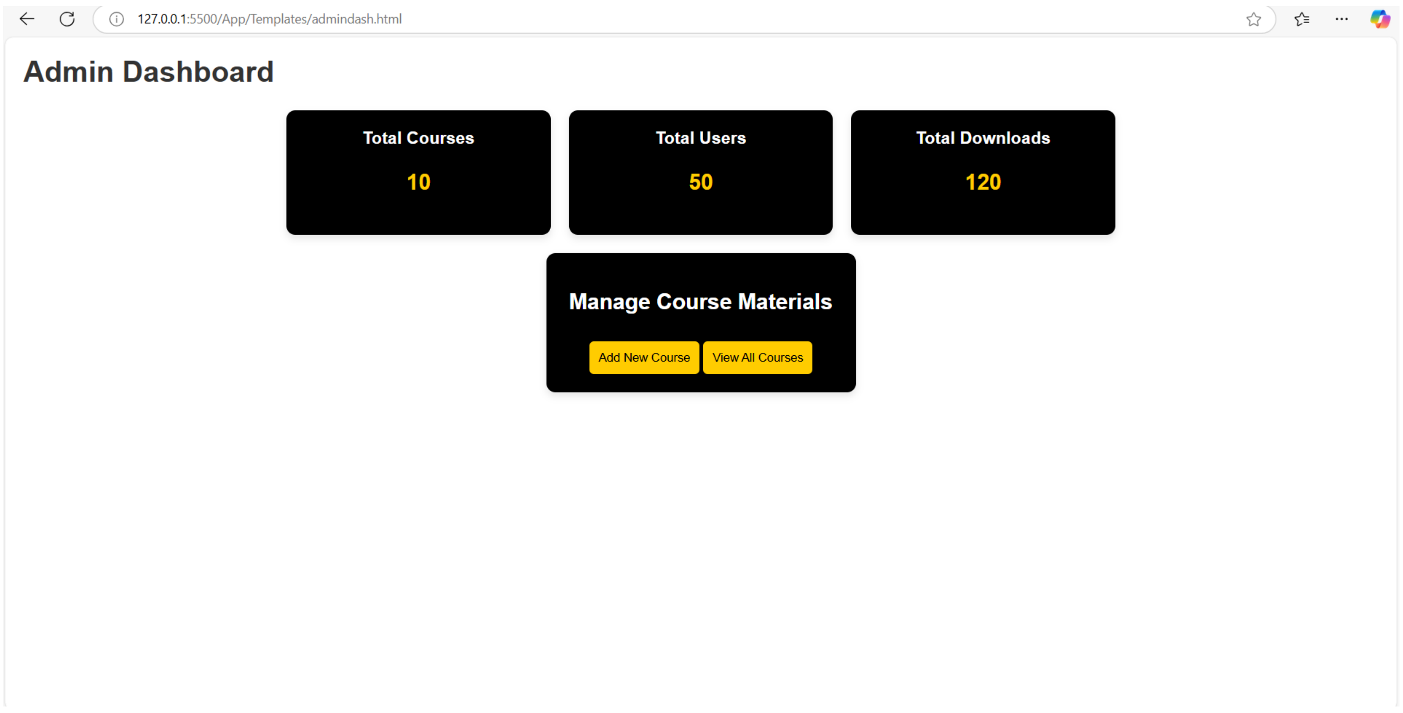
templates/admindash.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Admin Dashboard</title>
 <link rel="stylesheet" href="styles.css">
 <style>
  body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   background-color: #fff; /* Background changed to white */
   display: flex;
   flex-direction: column;
   align-items: center;
   height: 100vh;
  position: relative;
}
h1 {
  color: #333;
  font-size: 32px;
  position: absolute;
  top: 20px;
  left: 20px;
  margin: 0;
}
.dashboard-cards {
  display: flex;
  gap: 20px;
  margin-top: 80px;
}
.card {
  background: #000;
  color: white;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  width: 250px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.card h3 {
  margin: 0;
}
.card .count {
  font-size: 24px;
  font-weight: bold;
```

```
    color: #ffcc00;
   }
   .course-management {
    margin-top: 20px;
    background: #000;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
    width: 300px;
    color: #fff;
   }
   .course-management button {
    margin-top: 10px;
    background-color: #ffcc00;
    color: #000;
    border: none;
    padding: 10px;
    border-radius: 5px;
    cursor: pointer;
   }
  </style>
</head>
<body>
  <h1>Admin Dashboard</h1>
  <div class="dashboard-cards">
   <div class="card">
    <h3>Total Courses</h3>
    <p class="count">10</p>
   </div>
```

```
    <div class="card">
      <h3>Total Users</h3>
      <p class="count">50</p>
    </div>
    <div class="card">
      <h3>Total Downloads</h3>
      <p class="count">120</p>
    </div>
  </div>
  <div class="course-management">
    <h2>Manage Course Materials</h2>
    <button>Add New Course</button>
    <button>View All Courses</button>
  </div>
</body>
</html>
```

Output:



127.0.0.1:5500/App/Templates/admindash.html

Admin Dashboard

Total Courses
10

Total Users
50

Total Downloads
120

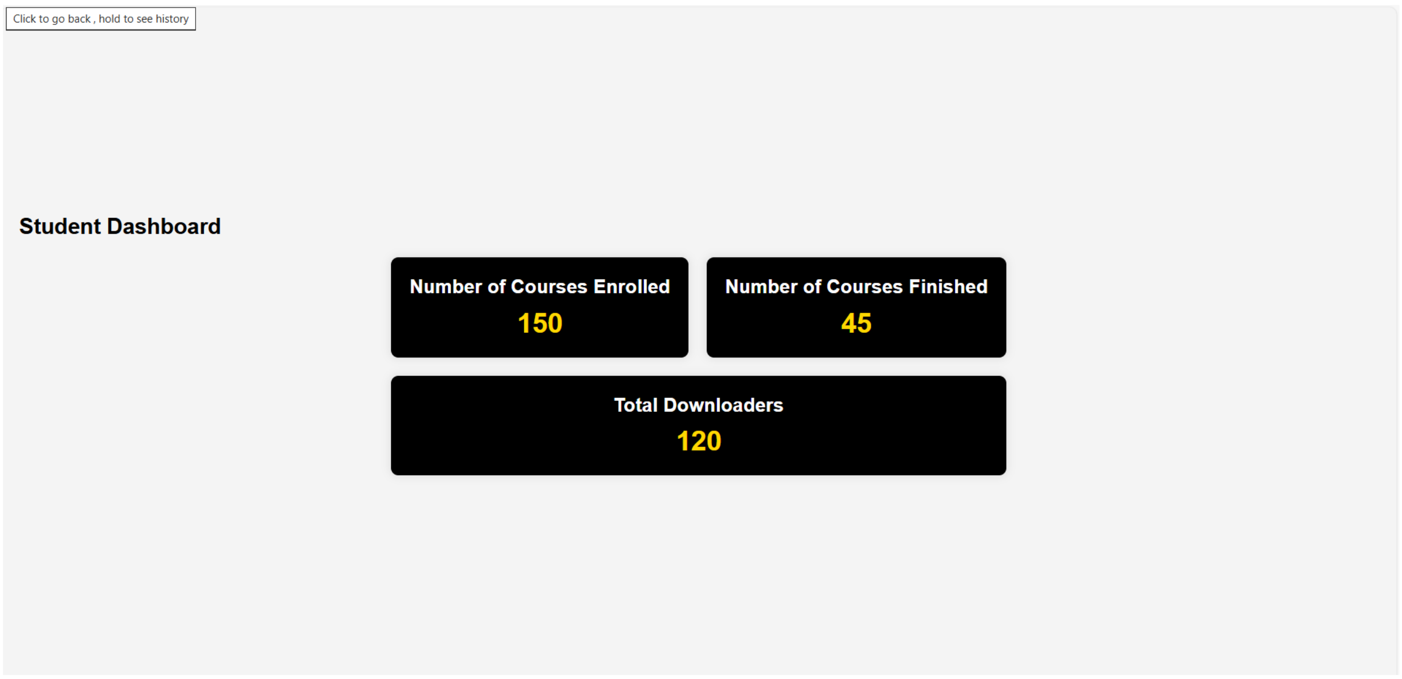
Manage Course Materials

Templates/studentdash.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title>Student Dashboard</title>

   <style>

      * {

         margin: 0;

         padding: 0;

         box-sizing: border-box;

         font-family: Arial, sans-serif;

      }

      body {

         display: flex;

         flex-direction: column;

         align-items: center;

         justify-content: center;

         min-height: 100vh;

         background: #f4f4f4;

         padding: 20px;

      }

      .header {

         font-size: 24px;

         font-weight: bold;

         margin-bottom: 20px;

         align-self: flex-start;

      }

      .stats {
            display: grid;
            grid-template-areas:
                "users courses"
                "downloaders downloaders";
            gap: 20px;
            justify-content: center;
            align-items: center;
        }
        .stat-card {
            background: black;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            font-size: 18px;
            color: white;
            text-align: center;
            font-weight: bold;
        }
        .stat-card p {
            font-size: 30px;
            font-weight: bold;
            margin-top: 10px;
            color: gold;
        }
        .stat-card:nth-child(1) { grid-area: users; }
        .stat-card:nth-child(2) { grid-area: courses; }
        .stat-card:nth-child(3) { grid-area: downloaders; }
    </style>
</head>
<body>
```

```
    <div class="header">Student Dashboard</div>

    <div class="stats">

        <div class="stat-card">

            <h3>Number of Courses Enrolled</h3>

            <p>150</p>

        </div>

        <div class="stat-card">

            <h3>Number of Courses Finished</h3>

            <p>45</p>

        </div>

        <div class="stat-card">

            <h3>Total Downloaders</h3>

            <p>120</p>

        </div>

    </div>

</body>

</html>
```

Output:



Student Dashboard

Number of Courses Enrolled
150

Number of Courses Finished
45

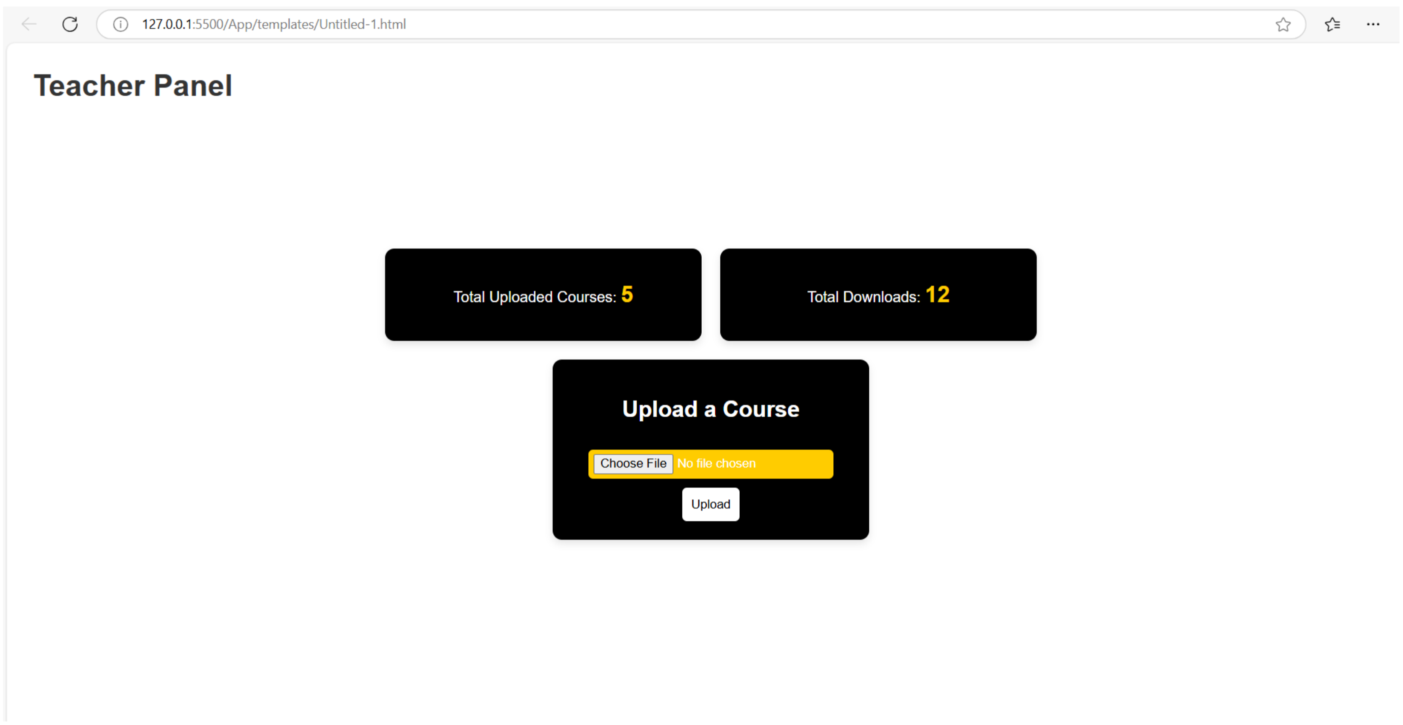
Total Downloaders
120

Templates/teachdash.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Teacher Dashboard</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f4f4f4;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
    }
    .container {
      background: white;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      text-align: center;
    }
    ul {
      list-style-type: none;
      padding: 0;
    }
    li {
      background: black;
```

```
            color: white;

            margin: 5px 0;

            padding: 10px;

            border-radius: 5px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Registered Teachers</h2>

        <ul>

            {% for teacher in teachers %}

                <li>{{ teacher.username }} - {{ teacher.email }}</li>

            {% empty %}

                <p>No teachers available.</p>

            {% endfor %}

        </ul>

    </div>

</body>

</html>
```

Output:



Teacher Panel

Total Uploaded Courses: 5

Total Downloads: 12

Upload a Course

Templates/course.html:

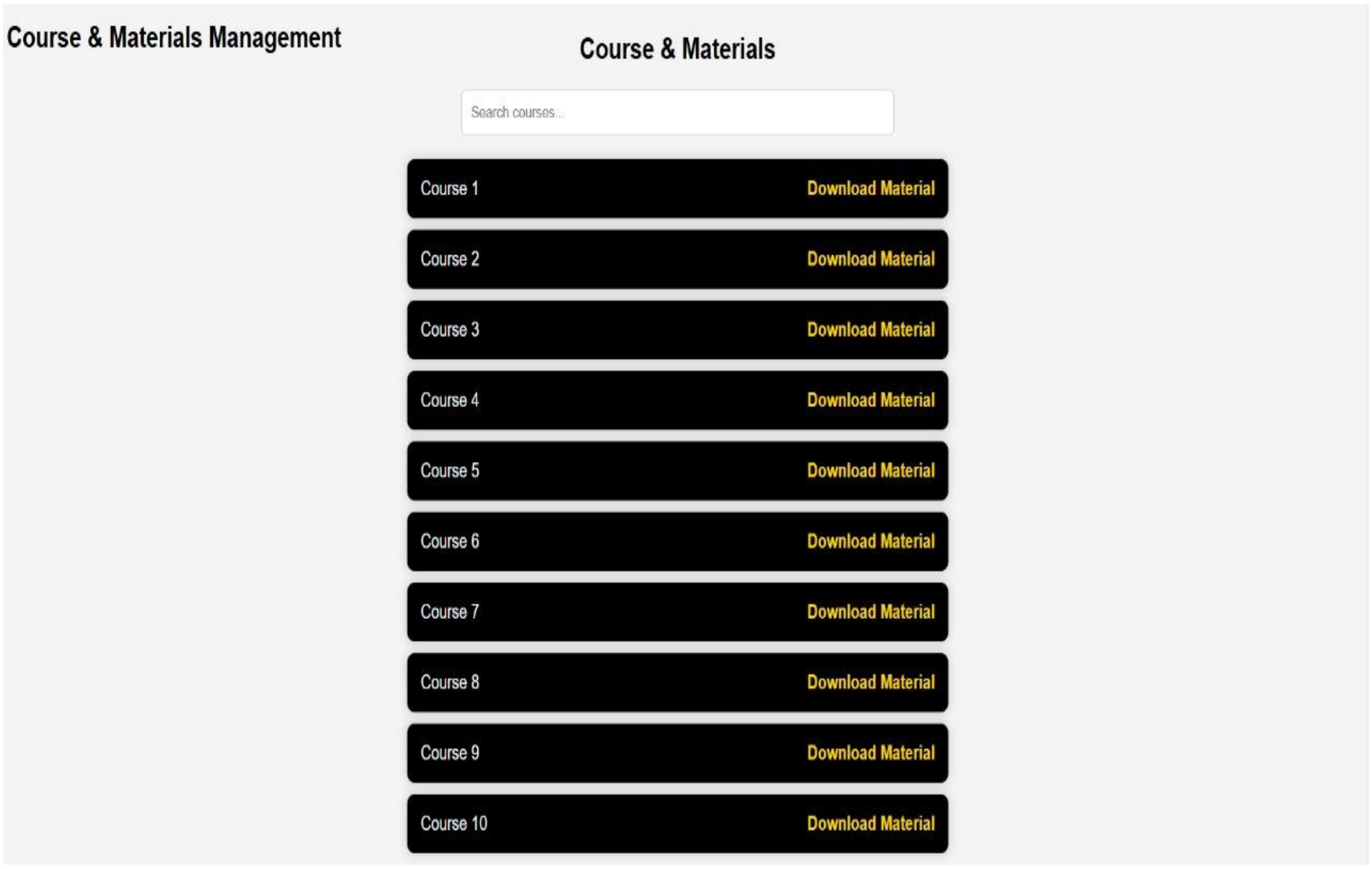
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Course & Materials</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: Arial, sans-serif;
    }
    body {
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
      background: #f4f4f4;
      padding: 20px;
      position: relative;
    }
    .header {
      position: absolute;
      top: 20px;
      left: 20px;
      font-size: 24px;
      font-weight: bold;
```

```
}
.course-container {
   width: 100%;
   max-width: 600px;
   text-align: center;
}
.course-container h2 {
   margin-bottom: 20px;
}
.search-box {
   margin-bottom: 20px;
   display: flex;
   justify-content: center;
}
.search-box input {
   width: 80%;
   padding: 10px;
   border: 1px solid #ccc;
   border-radius: 5px;
}
.course-list {
   display: flex;
   flex-direction: column;
   gap: 10px;
}
.course-item {
   background: black;
   color: white;
   padding: 15px;
   border-radius: 8px;
```

```
          display: flex;
          justify-content: space-between;
          align-items: center;
          box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
       }
       .course-item a {
          text-decoration: none;
          color: gold;
          font-weight: bold;
       }
    </style>
    <script>
       function searchCourses() {
          let input = document.getElementById("searchInput").value.toLowerCase();
          let courses = document.getElementsByClassName("course-item");

          for (let i = 0; i < courses.length; i++) {
             let courseName = courses[i].getElementsByTagName("span")[0].innerText.toLowerCase();
             if (courseName.includes(input)) {
                courses[i].style.display = "flex";
             } else {
                courses[i].style.display = "none";
             }
          }
       }
    </script>
</head>
<body>
    <div class="header">Course & Materials Management</div>
    <div class="course-container">
```

```
      <h2>Course & Materials</h2>

      <div class="search-box">

         <input type="text" id="searchInput" placeholder="Search courses..."
onkeyup="searchCourses()">

      </div>

      <div class="course-list">

         <div class="course-item"><span>Course 1</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 2</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 3</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 4</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 5</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 6</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 7</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 8</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 9</span><a href="#">Download Material</a></div>

         <div class="course-item"><span>Course 10</span><a href="#">Download Material</a></div>

      </div>

   </div>

</body>

</html>
```

Output:



Course & Materials Management

Course & Materials

Search courses...

Course 1 Download Material

Course 2 Download Material

Course 3 Download Material

Course 4 Download Material

Course 5 Download Material

Course 6 Download Material

Course 7 Download Material

Course 8 Download Material

Course 9 Download Material

Course 10 Download Material

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Database Integration and Configuration SQL LITE
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 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**

Database Integration and Configuration SQL LITE

DATA BASE:

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: Ensure SQLite3 is Installed

SQLite3 comes pre-installed with Python, so you don’t need to install it separately. You can check if SQLite3 is available by running:

```
python -c "import sqlite3; print(sqlite3.sqlite_version)"
```

Step 2: Install SQLite3 (if not already installed)

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:
   - Search for "Environment Variables" in the Start menu.
   - Click on "Environment Variables."
   - In "System variables," select "Path" and click "Edit."
   - Add the folder path (e.g., C:\sqlite) and click OK.

Step 2: Configure the Database in settings.py

In Django, the settings.py file is the central configuration module for your project, containing essential settings required for its operation. It defines the project's database configuration, middleware, installed apps, and URL routing setup. Additionally, it manages settings for static and media files, localization (like language and time zone), and security, including ALLOWED_HOSTS and cookie options. The file also includes crucial details like the SECRET_KEY for cryptographic operations and the DEBUG flag, which helps differentiate between development and production environments. For larger projects, settings.py can be modularized into separate files to enhance organization and maintainability.

Django uses settings.py to define its database settings. Since you are using SQLite3, make sure your ENGINE: Specifies the database backend (sqlite3 in this case).

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'online_course_material_repository',
        'USER':'root',
        'PASSWORD':'lalithya',
        'HOST':'127.0.0.1',
'PORT':'3306',
       'OPTIONS': {
 'init_command':"SET sql_mode='STRICT_TRANS_TABLES'"
   }
}}
```

a) *ENGINE: Specifies the database backend (sqlite3 in this case).*
b) *NAME: The name of the SQLite database file (it will be stored as db.sqlite3 in the project's root directory).*

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Handling Forms in Django
7. Date of Experiment	:	21-02-2025
8. Date of Submission of Report	:	21-02-2025

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**

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:

1. Simplifies form creation
2. Handles input validation automatically
3. Integrates with Django models
4. Prevents security risks like SQL Injection & CSRF attacks

Types of Forms in Django:

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

```
# myapp1/forms.py
from django import forms
from .models import Room, Booking
from datetime import time

class RoomForm(forms.ModelForm):
    class Meta:
        model = Room
        fields = ['name', 'capacity', 'teacher', 'is_active']  # Include the fields you need

class BookingForm(forms.ModelForm):
    class Meta:
        model = Booking
        fields = ['room', 'date', 'time_slot', 'booked_by', 'user']

    time_slot = forms.TimeField(required=True)  # Make it mandatory to catch errors
```

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
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 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**

Defining and Using Models

Create Your Models in models.py

In Django,models.py is a file within an application where database models are defined using Python classes. Each model represents a table in the database, with class attributes corresponding to the table's columns. Django's Object-Relational Mapping (ORM) system translates these models into SQL queries, allowing developers to interact with the database using Python code rather than raw SQL. Fields in a model define the type and properties of the data stored in each column.

In Django, models define the database structure. Navigate to your App directory (your Django app) and open models.py. Define the models for your course repository system.

Model.py

- UserProfile extends Django's built-in AbstractUser model and adds a user_type field.
- Course contains a title, description, and a foreign key relationship to a teacher

Register Models in admin.py

The admin.py file in a Django application is where you register your models to appear in the Django admin interface. By adding your models to this file, you enable administrators to manage the application's data visually through the built-in admin dashboard. You typically use the admin.site.register() function to link your models, or you can create custom admin classes to specify how models should be displayed, filtered, or organized in the admin panel.

To manage your database through Django Admin, register the models in admin.py

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Migrations: Synchronizing Model with the Database
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 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**

Migrations: Synchronizing Model with the Database

Apply Migrations

In Django, migrations are a way to propagate changes made to your models (like adding a field, deleting a model, or changing constraints) into your database schema. They're an essential feature that allows your database structure to stay in sync with your code as it evolves.

Now, let’s create and apply database migrations

1.Generate Migration Files:

Run the following command:

```
python manage.py makemigrations
```

Django will generate migration files based on the models defined.

2.Apply Migrations to the Database:

```
python manage.py migrate
```

This will create the necessary tables in the db.sqlite3 file.

Create a Superuser

In Django, a superuser is a user account with administrative privileges. This account has full control over the Django project and can access the Django Admin interface to manage all models and data within the application.

Key Features of a Django Superuser:

1. Full permissions
2. User management
3. Data management

To access the Django admin panel, you need a superuser. Create one with:

```
python manage.py createsuperuser
```

Follow the prompts to enter a username, email, and password

Verify Database Connection

You can confirm that the tables were created successfully by running:

```
python manage.py dbshell
```

Then, type:

```
.tables
```

It should list your tables like App_userprofile and App_course.

Insert Sample Data

To manually add data, open the Django shell:

```
python manage.py shell
```

Then, enter:

```
from App.models import UserProfile, Course
teacher = UserProfile.objects.create(username="teacher1", user_type="teacher")
course = Course.objects.create(title="Django Basics", description="Learn Django from scratch", teacher=teacher)
```

Run the Server and Test

Start the Django server:

```
python manage.py runserver
```

Visit:

Home page: http://127.0.0.1:8000/

Admin panel: http://127.0.0.1:81000/admin/

You should see your database records displayed!

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	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Deploying Django Applications Models with the Database
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 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**

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: git init

2. Connect to GitHub:

git remote add origin
https://github.com/lalithya12/Assignment_Submission_Portal.git

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/haridammu/Course aterial repository system.git

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

Dr.Ch. Bindu Madhuri

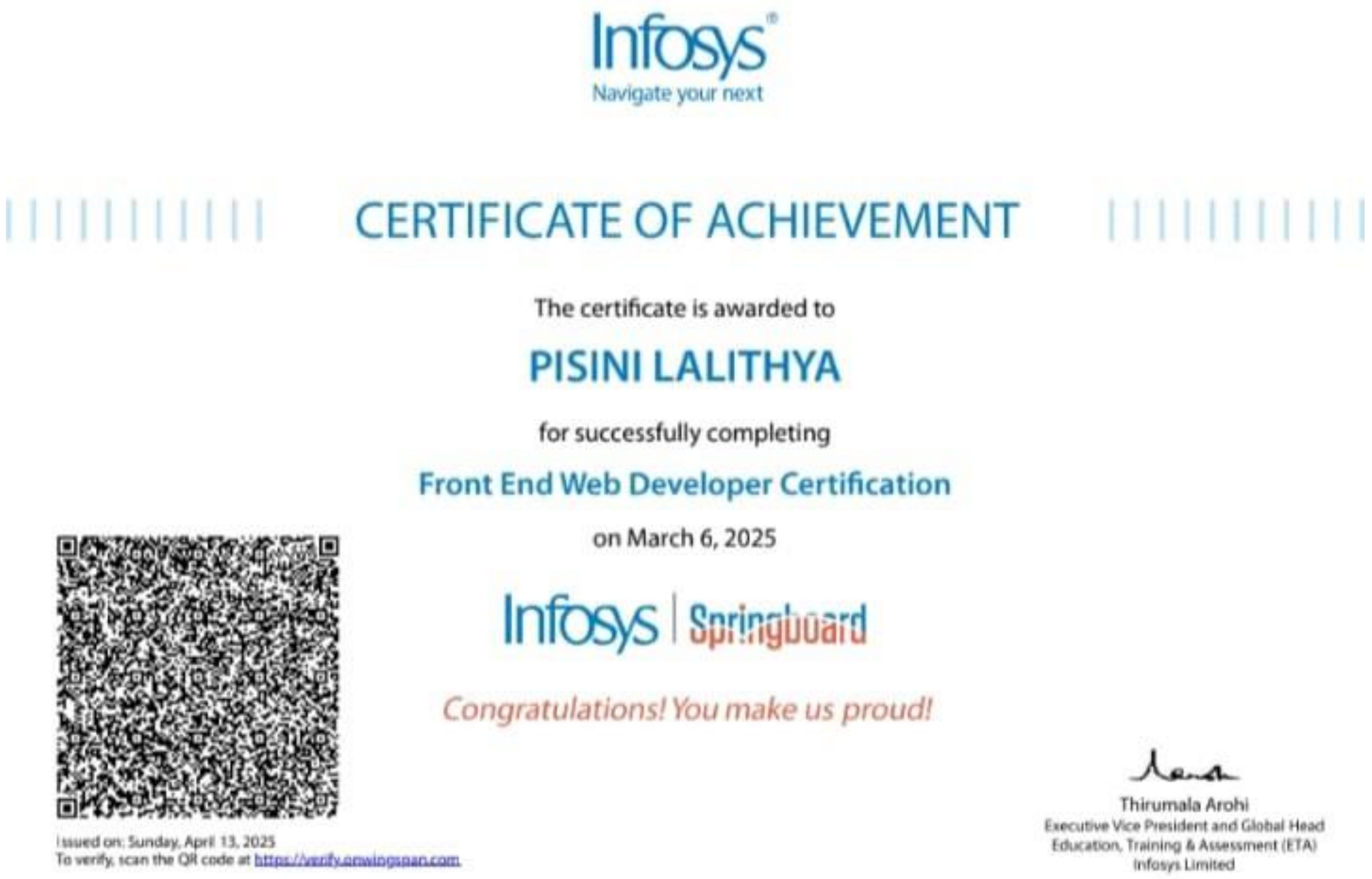
Asst. Professor & HOD **Email: hod.it@intugvcev.edu.in**

1. Name of the Laboratory	:	Django FrameWork lab
2. Name of the Student	:	Pisini.Lalithya
3. Roll No	:	23VV1A1242
4. Class	:	II B.TECH II SEMESTER
5. Academic Year	:	2024-25
6. Name of Experiment	:	Front End Certificate
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 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**

FrontEnd Certificate:



Infosys®
Navigate your next

CERTIFICATE OF ACHIEVEMENT

The certificate is awarded to

PISINI LALITHYA

for successfully completing

Front End Web Developer Certification

on March 6, 2025

Infosys | Springboard

Congratulations! You make us proud!

Issued on: Sunday, April 13, 2025
To verify, scan the QR code at https://verify.onwingspan.com

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