**DEPARTMENT OF INFORMATION TECHNOLOGY**

**JNTU-GURAJADA VIZIANAGARAM**

**COLLEGE OF ENGINEERING**

**VIZIANAGARAM ANDHRA PRADESH**

***A logo with a flower and cogwheel

AI-generated content may be incorrect.***

***DJANGO FRAME WORK LAB***

**YELAMANCHILI NAVEEN (24VV5A1260)**

SHAIK ASMA (23VV1A1247)

TATAPUDI MADHU SARVANI (23VV1A1251)

BODAPATI BINDU SRI (24VV5A1268)

UNDER GUIDANCE OF

**MRS.MADHUMITA CHANDA**

**DEPARTMENT OF INFORMATION**

**TECHNOLOGY**

****

**JNTU-GURAJADA VIZIANAGARAM**

**COLLEGE OF ENGINEERING VIZIANAGARAM (A)**

**VIZIANAGARAM**

**Regd.No : 23VV1A1260**

**CERTIFICATE**

**This is to certify that this is a bonafide record of practical work done by Mr .yelamanchili naveen of IInd B.Tech IInd Semester Class in DJANGO FRAMEWORK 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 . |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Outcomes** | | **Program Outcomes (POs)** | | | | | | | | | | | | | | | |
| **PO 1** | **PO 2** | **PO 3** | **PO 4** | **PO 5** | **PO 6** | **PO 7** | **PO 8** | **PO 9** | **PO 10** | **PO 11** | **PO 12** | **PSO 1** | **PSO 2** | **PSO 3** |
|  | **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** |

**CO-PO Mapping**Mapping of Course Outcomes (COs) with Program Outcomes (POs)

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DANGO FRAME WORK  INDEX | | | | | |
| **S.NO** | **DATE** | **CONCEPT** | **PAGE**  **NO.** | **MARKS** | **SIGNATURE** |
| 1 | 13-12-2024 | **Understanding Django and Its Libraries** | 05-16 |  |  |
| 2 | 20-12-2024 | **Introduction to Django Framework** | 17-18 |  |  |
| 3 | 27-12-2024 | **Step-by-Step Guide to Installing Django** | 19-20 |  |  |
| 4 | 03-01-2025 | Linking Views and URL Configurations | 21-22 |  |  |
| 5 | 24-01-2025 | **Exploring Django Views** | 23-28 |  |  |
| 6 | 24-01-2025 | **Setting Up App-Level URLs** | 29-31 |  |  |
| 7 | 31-01-2025 | **Working with Templates in Django** | 31-55 |  |  |
| 8 | 21-02-2025 | **Database Integration and Configuration-MYSQL** | 56-58 |  |  |
| 9 | 21-02-2025 | **Handling Forms in Django** | 59-61 |  |  |
| 10 | 21-02-2025 | **Defining and Using Models** | 62-65 |  |  |
| 11 | 07-03-2025 | **Migrations: Synchronizing Models with the Database** | 66-68 |  |  |
| 12 | 27-03-2025 | **Deploying Django Applications on Cloud Platforms** | 69-70 |  |  |
| 13 | 04-04-2025 | Frontend developer certificate | 70-71 |  |  |

 **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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : Y.NAVEEN

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 20-12-2024

DATE: Signature of Faculty

**PYTHON LIBRARIES**

**1. Python Collections - Container Datatypes:**

1. **Purpose**: Provides specialized container datatypes that support efficient handling of data.
2. **Key Types**:
   1. **List**: Ordered, mutable, allows duplicates.
   2. **Tuple**: Ordered, immutable, allows duplicates.
   3. **Set**: Unordered, no duplicates, fast membership testing.
   4. **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).

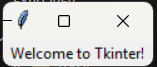
**2. Tkinter - GUI Applications:**

1. **Purpose**: Python’s standard library for creating graphical user interfaces (GUIs).
2. **Key Features**:
   1. Widgets: Buttons, labels, text boxes, etc.
   2. Event handling: Respond to user interactions like clicks or key presses.
   3. Simple layout management.
3. **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:**



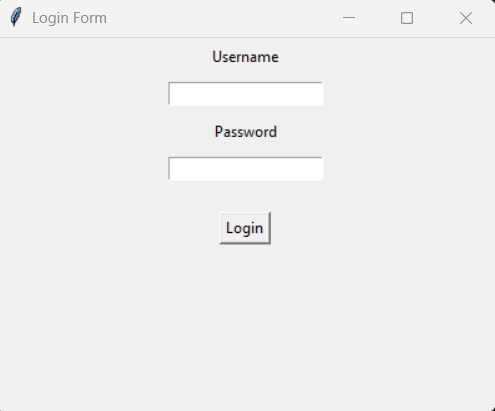
**3. Requests - HTTP Requests:**

1. **Purpose**: Simplifies HTTP requests to interact with web APIs.
2. **Key Features**:
   1. Send GET, POST, PUT, DELETE requests easily.
   2. Handle request parameters, headers, and cookies.
   3. 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**:
   1. Fast, extensible, and asynchronous web scraping.
   2. Supports handling requests, data extraction, and storing results.
   3. 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**:
   1. Easy navigation and searching within HTML.
   2. Supports different parsers like html.parser, lxml, and html5lib.
3. **Common Use**: Extract data from websites for analysis, e.g., for building data-driven applications.

**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**:
   1. Supports frameworks like Flask and Django for serverless deployments.
   2. 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**:
   1. Built on top of Flask, React, and Plotly.
   2. Integrates seamlessly with data science libraries (e.g., Pandas, Plotly).
3. **Common Use**: Building dashboards and data-driven web applications.

**8. TurboGears:**

1. **Purpose:** Full-stack web framework built on top of WSGI.
2. **Key Features:** 
   1. Modular: Mix and match components like SQLAlchemy, Genshi, and others.
   2. Focus on rapid development and scalability.
3. **Common Use:** Develop scalable, enterprise-level web applications.

**9. CherryPy:**

1. **Purpose**: Minimalistic web framework for building web applications.
2. **Key Features**:
   1. Provides a simple and fast HTTP server.
   2. 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 :-



**10. 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)*** |

**Output:**

|  |
| --- |
| ***(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*** |

After run the server:



**11. Web2Py:**

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

**12. Bottle:**

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

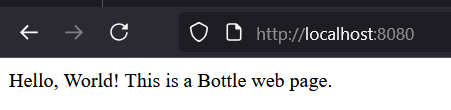
**Code:**

|  |
| --- |
| ***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:



**13. Falcon:**

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

**14. CubicWeb:**

1. **Purpose**: Web application framework based on an entity-relation model.
2. **Key Features**:
   1. Uses a highly modular architecture for development.
   2. Focus on building web apps with rich data models.
3. **Common Use**: Semantic web applications or data-driven web apps.

**15. Quixote:**

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

**16. Pyramid:**

1. **Purpose**: Full-stack web framework that can scale from simple to complex applications.
2. **Key Features**:
   1. Highly flexible with support for routing, templating, authentication, and authorization.
   2. 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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

4. Class : II B-Tech II Semester

5. Academic Year : 2024-25

6. Name of Experiment : Introduction to Django Frame Work

7. Date of Experiment : 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 andObservations | 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 |  |

8. Date of Submission of Report : 27-12-2024

DATE: Signature of Faculty

**Django: A Web Framework for Python**

1. 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:**

1. Model (M) – Handles database interactions (e.g., User, Booking).
2. View (V) – Manages business logic and connects models to templates.
3. Template (T) – Renders HTML pages dynamically.

**Example MVT Folder Structure in Django**

A screenshot of a computer program

AI-generated content may be incorrect.



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 03-01-2025

**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

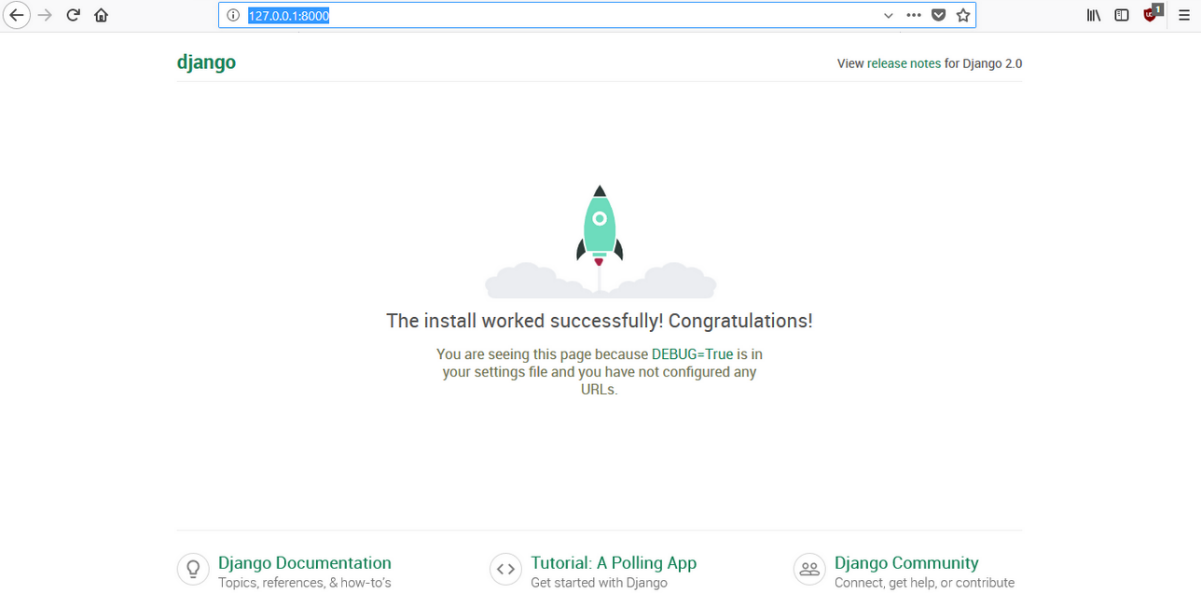
Step-6 : Start new\_project django-admin startproject project\_name

django-admin startapp app\_name

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

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

**OUTPUT:**





**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 24-01-2025

**DATE:** signature of faculty

**Connecting Views and URLs:**

**Set Up URLs:**

**Project-level URL Configuration:**

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('', include('myapp1.urls')), # Include app URLs

]

**App-level URL Configuration**

from django.urls import path

from .views import home

urlpatterns = [

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

]

**Create a Sample View:**

from django.http import HttpResponse

def home(request):

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

**Run Migrations:**

python manage.py migrate

**Run the Server and Test:**

python manage.py runserver



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 24-01-2025

**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.

**Code:**

|  |
| --- |
| ***from django.shortcuts import render, redirect***  ***from django.contrib.auth import login, authenticate, logout***  ***from .forms import SignupForm, LoginForm***  ***def signup\_view(request):***  ***# If the request method is POST, it means the form was submitted***  ***if request.method == 'POST':***  ***form = SignupForm(request.POST) # Bind data to the form***  ***if form.is\_valid(): # Check if the form is valid***  ***user = form.save()***  ***login(request, user) # Log the user in***  ***return redirect('dashboard')***  ***else:***  ***form = SignupForm()***  ***return render(request, 'core/signup.html', {'form': form}) # Render the signup page with form***  ***def login\_view(request):***  ***# If the request method is POST, it means the form was submitted***  ***if request.method == 'POST':***  ***form = LoginForm(data=request.POST) # Bind POST data to login form***  ***if form.is\_valid():***  ***user = form.get\_user() # Get the authenticated user***  ***login(request, user) # Log the user in***  ***return redirect('dashboard')***  ***else:***  ***form = LoginForm() # If GET request, show empty login form***  ***return render(request, 'core/login.html', {'form': form})***  ***def logout\_view(request):***  ***logout(request) # Log the user out***  ***return redirect('login')*** |

The above views connects the login page ,logout page and singning of the mentee and mentor.

|  |
| --- |
| ***from django.conf import settings  #  Import settings***  ***from django.shortcuts import render, redirect, get\_object\_or\_404***  ***from django.contrib.auth.decorators import login\_required***  ***from django.contrib.auth import get\_user\_model  #  Get the actual user model***  ***from .models import Profile, MentorshipRequest, Message, MentorshipSession,Notification***  ***from .forms import ProfileForm, MentorshipRequestForm, MessageForm, MentorshipSessionForm***  ***from django.contrib import messages***  ***from django.http import JsonResponse***  ***User = get\_user\_model()  #  Now we have the actual CustomUser model***  ***@login\_required***  ***def dashboard\_view(request):***  ***print(f"User: {request.user}, is\_staff: {request.user.is\_staff}")  # Debugging user role***    ***if request.user.is\_staff:  # Mentor Dashboard***  ***pending\_requests = MentorshipRequest.objects.filter(mentor=request.user, status="pending").select\_related('mentee', 'mentor')***  ***mentoring\_sessions = MentorshipSession.objects.filter(mentor=request.user).select\_related('mentee', 'mentor').order\_by('scheduled\_time')***  ***upcoming\_sessions = mentoring\_sessions  # Mentors don't need this***  ***else:  # Mentee Dashboard***  ***pending\_requests = None  # Mentees don't see pending requests***  ***upcoming\_sessions = MentorshipSession.objects.filter(mentee=request.user).select\_related('mentee', 'mentor').order\_by('scheduled\_time')***  ***mentoring\_sessions = None  # Mentees don’t mentor others***  ***# Fetch unread notifications for the user***  ***unseen\_notifications = Notification.objects.filter(user=request.user, is\_read=False).order\_by('-created\_at')***  ***# Debugging: Print query results in the terminal***  ***print(f"Pending Requests: {pending\_requests}")***  ***print(f"Upcoming Sessions: {upcoming\_sessions}")***  ***print(f"Mentoring Sessions: {mentoring\_sessions}")***  ***print(f"Unseen Notifications: {unseen\_notifications}")***  ***return render(request, 'dashboard/dashboard.html', {***  ***'pending\_requests': pending\_requests,***  ***'upcoming\_sessions': upcoming\_sessions,***  ***'mentoring\_sessions': mentoring\_sessions,***  ***'unseen\_notifications': unseen\_notifications,  # Pass unseen notifications to template***  ***})***    ***@login\_required***  ***def profile\_view(request):***  ***profile, created = Profile.objects.get\_or\_create(user=request.user)***  ***if request.method == 'POST':***  ***form = ProfileForm(request.POST, request.FILES, instance=profile)***  ***if form.is\_valid():***  ***form.save()***  ***return redirect('dashboard')***  ***else:***  ***form = ProfileForm(instance=profile)***  ***return render(request, 'dashboard/profile.html', {'form': form})***  ***@login\_required***  ***def send\_request\_view(request):***  ***if request.method == "POST":***  ***form = MentorshipRequestForm(request.POST)***  ***if form.is\_valid():***  ***mentorship\_request = form.save(commit=False)***  ***mentorship\_request.mentee = request.user***  ***mentorship\_request.save()***  ***messages.success(request, )***  ***return redirect('dashboard')***  ***else:***  ***form = MentorshipRequestForm()***  ***form.fields['mentor'].queryset = User.objects.filter(is\_staff=False).exclude(id=request.user.id)***  ***return render(request, 'dashboard/mentorship\_request.html', {'form': form})***  ***@login\_required***  ***def accept\_request\_view(request, request\_id):***  ***mentorship\_request = get\_object\_or\_404(MentorshipRequest, id=request\_id, mentor=request.user)***  ***if mentorship\_request.status == "pending":***  ***mentorship\_request.status = "accepted"***  ***mentorship\_request.save()***  ***# Send notification to mentee***  ***send\_notification(***  ***mentorship\_request.mentee,***  ***f"Your mentorship request to {mentorship\_request.mentor.username} has been accepted! 🎉"***  ***)***  ***messages.success(request, "You have accepted the mentorship request.")***  ***return redirect('dashboard')***  ***@login\_required***  ***def decline\_request\_view(request, request\_id):***  ***mentorship\_request = get\_object\_or\_404(MentorshipRequest, id=request\_id, mentor=request.user)***  ***if mentorship\_request.status == "pending":***  ***mentorship\_request.status = "declined"***  ***mentorship\_request.save()***  ***messages.warning(request, "You have declined the mentorship request.")***  ***return redirect('dashboard')***  ***@login\_required***  ***def chat\_view(request, user\_id):***  ***other\_user = get\_object\_or\_404(User, id=user\_id)***  ***messages = Message.objects.filter(***  ***sender\_\_in=[request.user, other\_user],***  ***receiver\_\_in=[request.user, other\_user]***  ***).order\_by('timestamp')***  ***if request.method == "POST":***  ***form = MessageForm(request.POST)***  ***if form.is\_valid():***  ***message = form.save(commit=False)***  ***message.sender = request.user***  ***message.receiver = other\_user***  ***message.save()***  ***return redirect('chat', user\_id=other\_user.id)***  ***else:***  ***form = MessageForm()***  ***return render(request, 'dashboard/chat.html', {'messages': messages, 'form': form, 'other\_user': other\_user})***  ***@login\_required***  ***def schedule\_session\_view(request, request\_id):***  ***mentorship\_request = get\_object\_or\_404(MentorshipRequest, id=request\_id, mentor=request.user, status='accepted')***  ***if request.method == "POST":***  ***form = MentorshipSessionForm(request.POST)***  ***if form.is\_valid():***  ***session = form.save(commit=False)***  ***session.mentorship\_request = mentorship\_request***  ***session.mentor = request.user***  ***session.mentee = mentorship\_request.mentee***  ***session.save()***  ***# Send notification to mentee***  ***send\_notification(session.mentee, f"Your mentorship session with {session.mentor.username} is scheduled for {session.scheduled\_time}.")***  ***messages.success(request, "Mentorship session scheduled successfully!")***  ***return redirect('dashboard')***  ***else:***  ***form = MentorshipSessionForm()***  ***return render(request, 'dashboard/schedule\_session.html', {'form': form, 'mentorship\_request': mentorship\_request})***    ***def send\_notification(user, message):***  ***Notification.objects.create(user=user, message=message)***  ***@login\_required***  ***def notifications\_view(request):***  ***notifications = Notification.objects.filter(user=request.user).order\_by('-created\_at')***  ***return render(request, 'dashboard/notifications.html', {'notifications': notifications})***    ***@login\_required***  ***def unread\_notifications\_count(request):***  ***unread\_count = request.user.notifications.filter(is\_read=False).count()***  ***return JsonResponse({'unread\_count': unread\_count})*** |

In Django, the urls.py file is responsible for mapping URLs to views. It acts as the router of your application, directing user requests to the correct function in views.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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 31-01-2025

**DATE:** signature of faculty

**App URLs.py:**

**Creating urls.py in a Django App:**

Each Django app should have its own urls.py file to define app-specific routes. Steps to Create urls.py in a Django App

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

**The URLs.py in the core app:**

**Code :**

|  |
| --- |
| ***from django.urls import path***  ***from .views import signup\_view, login\_view, logout\_view***  ***urlpatterns = [***  ***path('signup/', signup\_view, name='signup'),***  ***path('login/', login\_view, name='login'),***  ***path('logout/', logout\_view, name='logout'),***  ***]*** |

**The URLs.py in the dashboard app:**

**Code :**

|  |
| --- |
| ***from django.urls import path***  ***from .views import (***  ***dashboard\_view,***  ***profile\_view,***  ***send\_request\_view,***  ***accept\_request\_view,***  ***decline\_request\_view,  # Corrected duplicate import***  ***chat\_view,***  ***schedule\_session\_view,***  ***notifications\_view,***  ***unread\_notifications\_count***  ***)***  ***urlpatterns = [***  ***path('', dashboard\_view, name='dashboard'),***  ***path('profile/', profile\_view, name='profile'),***  ***path('request/', send\_request\_view, name='send\_request'),***  ***path('request/accept/<int:request\_id>/', accept\_request\_view, name='accept\_request'),***  ***path('request/decline/<int:request\_id>/', decline\_request\_view, name='decline\_request'),***  ***path('chat/<int:user\_id>/', chat\_view, name='chat'),***  ***path('request/schedule/<int:request\_id>/', schedule\_session\_view, name='schedule\_session'),  # ✅ Updated path***  ***path('notifications/', notifications\_view, name='notifications'),***  ***path('notifications/unread\_count/', unread\_notifications\_count, name='unread\_notifications\_count'),***  ***]*** |

**Connecting App urls.py to Project urls.py:**

To use the app's URLs, include them in the project-level urls.py (myproject/urls.py)

**Code :**

|  |
| --- |
| ***from django.contrib import admin***  ***from django.urls import path, include***  ***from django.shortcuts import redirect***  ***urlpatterns = [***  ***path('admin/', admin.site.urls),***  ***path('', include('core.urls')),***  ***path('', lambda request: redirect('dashboard/', permanent=True)),***  ***path('dashboard/', include('dashboard.urls')),  # Include core app 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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 17-02-2025

**DATE:** signature of faculty**Template in Django :**

A template in Django is an HTML file that can include dynamic data using Django's template language.  
Templates are stored in the templates/ directory inside a Django app**.**

**The Folder Structure** **should is like:**

Mentorship\_platform/

│── core/

│ ├── templates/

│ │ ├── base.html # Base template

│ │ ├── dashboard/

│ │ │ ├── dashboard.html

│ │ │ ├── profile.html

│ │ │ ├── chat.html

│ │ │ ├── notifications.html

│ │ │ ├── mentorship\_request.html

│ │ │ ├── schedule\_session.html

│ ├── views.py

│ ├── models.py

│ ├── urls.py

│── manage.py

**Templates in the Peer Mentorship Platform**

**1. Base Template: base.html (Located in core/templates/core/)**

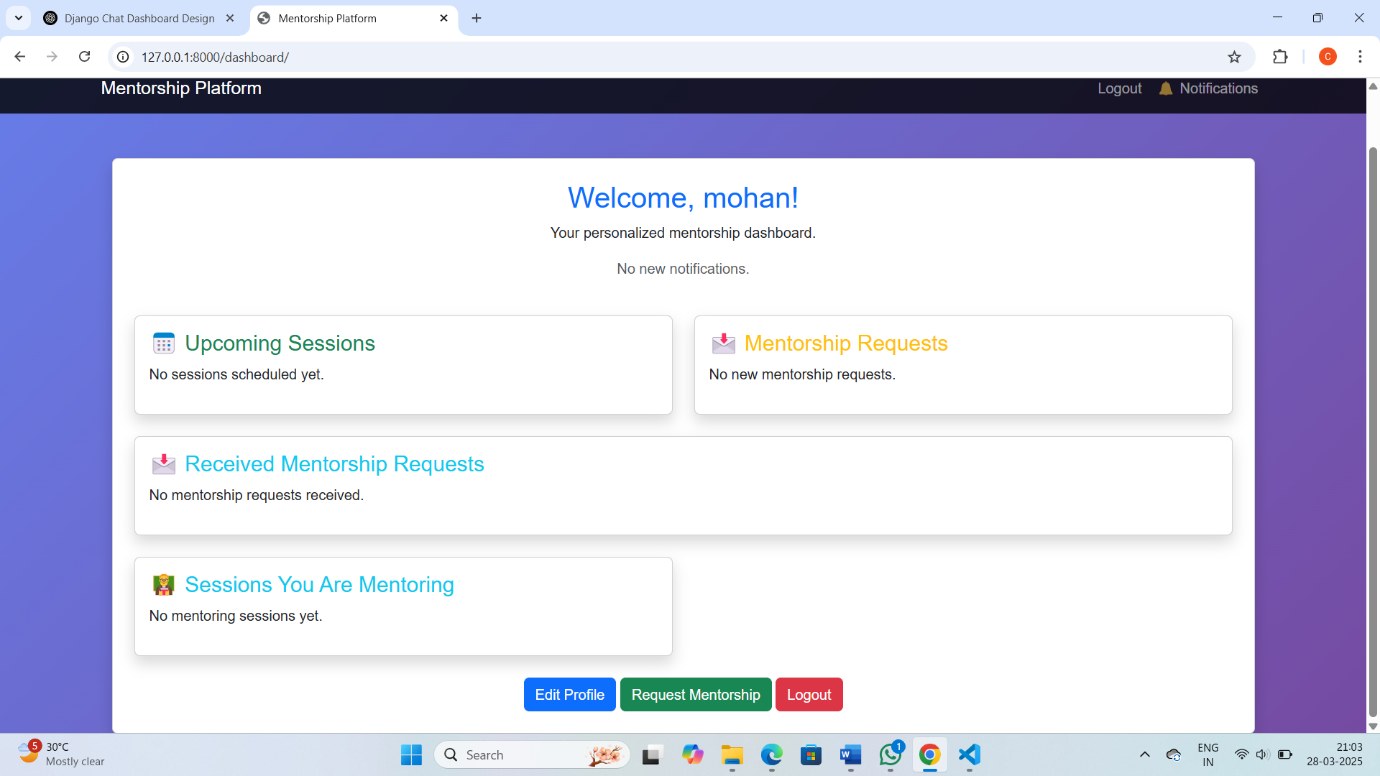
1. This is the main layout file that all other templates extend.
2. It includes Bootstrap for styling and defines a navigation bar.
3. The {% block content %}{% endblock %} section is where other templates insert their specific content.
4. It contains JavaScript functions for:
   1. Fetching unread notifications and updating the notification count dynamically.
   2. Checking for new notifications every 10 seconds.

**Code:**

|  |
| --- |
| ***<!DOCTYPE html>***  ***<html lang="en">***  ***<head>***  ***<meta charset="UTF-8">***  ***<meta name="viewport" content="width=device-width, initial-scale=1">***  ***<title>Mentorship Platform</title>***  ***<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">***  ***<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">***  ***<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">***  ***<style>***  ***body {***  ***background: linear-gradient(135deg, #667eea, #764ba2);***  ***color: white;***  ***font-family: Arial, sans-serif;***  ***}***  ***.navbar {***  ***background: rgba(0, 0, 0, 0.8);***  ***}***  ***.container {***  ***margin-top: 50px;***  ***}***  ***.btn-custom {***  ***background-color: #ff8c00;***  ***color: white;***  ***transition: 0.3s;***  ***}***  ***.btn-custom:hover {***  ***background-color: #ff6a00;***  ***}***  ***</style>***  ***</head>***  ***<body>***  ***<nav class="navbar navbar-expand-lg navbar-dark">***  ***<div class="container">***  ***<a class="navbar-brand" href="#">Mentorship Platform</a>***  ***<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">***  ***<span class="navbar-toggler-icon"></span>***  ***</button>***  ***<div class="collapse navbar-collapse" id="navbarNav">***  ***<ul class="navbar-nav ms-auto">***  ***{% if user.is\_authenticated %}***  ***<li class="nav-item"><a class="nav-link" href="{% url 'logout' %}">Logout</a></li>***  ***<li class="nav-item">***  ***<a class="nav-link" href="{% url 'notifications' %}">🔔 Notifications  <span id="notif-count" class="badge bg-danger">0</span></a>***  ***</li>***  ***{% else %}***  ***<li class="nav-item"><a class="nav-link" href="{% url 'login' %}">Login</a></li>***  ***<li class="nav-item"><a class="nav-link" href="{% url 'signup' %}">Sign Up</a></li>***  ***{% endif %}***  ***</ul>***  ***</div>***  ***</div>***  ***</nav>***  ***<div class="container">***  ***{% block content %}{% endblock %}***  ***</div>***  ***<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>***  ***<script>***  ***function checkNotifications() {***  ***fetch("{% url 'dashboard' %}") // ✅ Fetch the dashboard page every 10 seconds***  ***.then(response => response.text())***  ***.then(html => {***  ***let parser = new DOMParser();***  ***let doc = parser.parseFromString(html, 'text/html');***  ***let newNotifications = doc.querySelector('.alert-info');***  ***let currentNotifications = document.querySelector('.alert-info');***  ***if (newNotifications && (!currentNotifications || newNotifications.innerHTML !== currentNotifications.innerHTML)) {***  ***if (currentNotifications) {***  ***currentNotifications.innerHTML = newNotifications.innerHTML;***  ***} else {***  ***document.body.insertAdjacentHTML('afterbegin', newNotifications.outerHTML);***  ***}***  ***}***  ***});***  ***}***  ***// ✅ Check for notifications every 10 seconds***  ***setInterval(checkNotifications, 10000);***  ***</script>***  ***<script>***  ***function fetchNotifications() {***  ***fetch("{% url 'unread\_notifications\_count' %}")***  ***.then(response => response.json())***  ***.then(data => {***  ***let notifCount = document.getElementById("notif-count");***  ***if (data.unread\_count > 0) {***  ***notifCount.textContent = data.unread\_count;***  ***notifCount.style.display = "inline";***  ***} else {***  ***notifCount.style.display = "none";***  ***}***  ***});***  ***}***    ***// Call the function every 10 seconds***  ***setInterval(fetchNotifications, 10000);***    ***// Fetch notifications on page load***  ***fetchNotifications();***  ***</script>***  ***</body>***  ***</html>*** |

In this base template is the main layout file that all other templates extend.

**Output:**



**Django Admin Page:**

The **Django Admin page** is a built-in interface provided by Django to manage database models efficiently. It allows superusers (admins) to add, edit, delete, and view database records without writing SQL queries.

**Purpose of the Admin Page**

1. Provides a **Graphical User Interface (GUI)** for managing database models.
2. Allows authorized users (superusers) to **create, modify, and delete** records.
3. Simplifies managing **users, mentorship requests, sessions, and messages** in the Peer Mentorship Platform.
4. Supports **filtering, searching, and sorting** data efficiently.

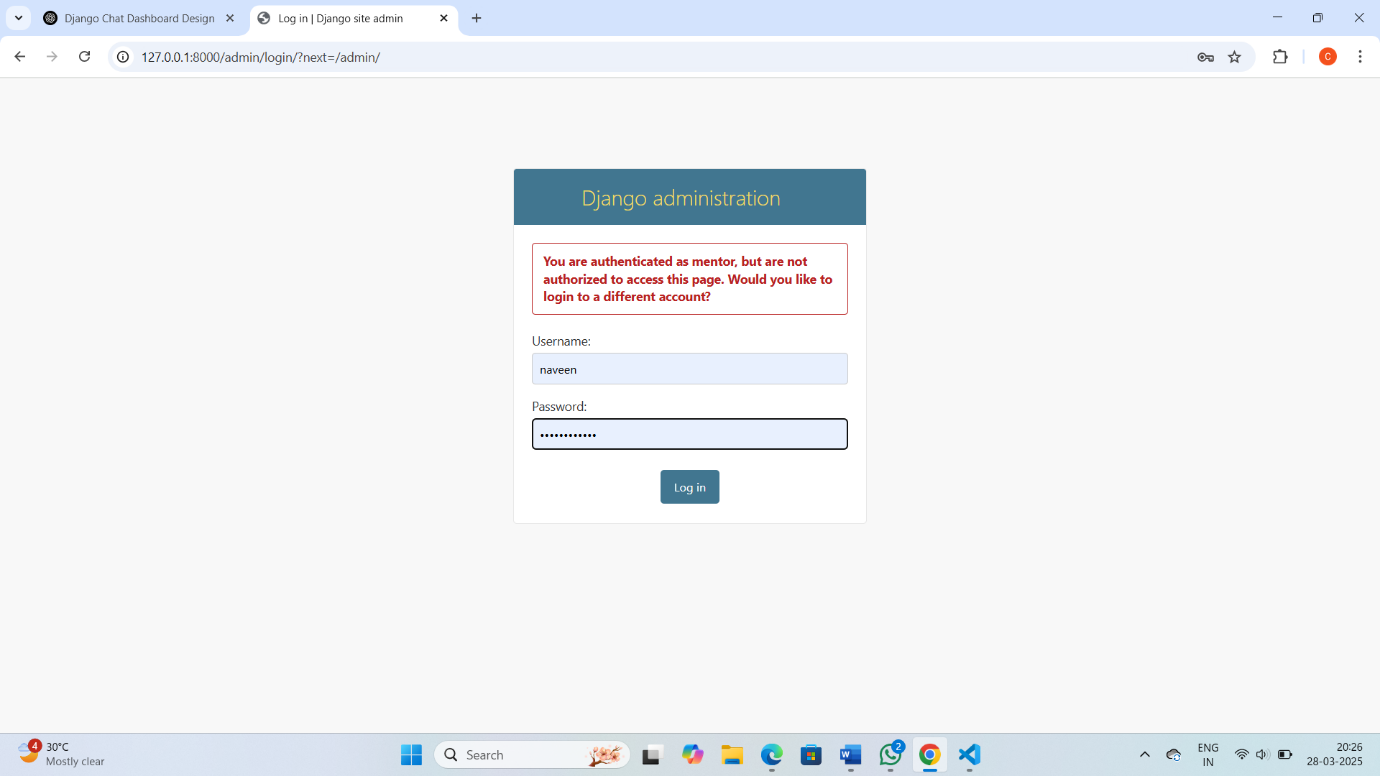
**Enabling the Admin Page**

Django includes the admin interface by default. To use it:

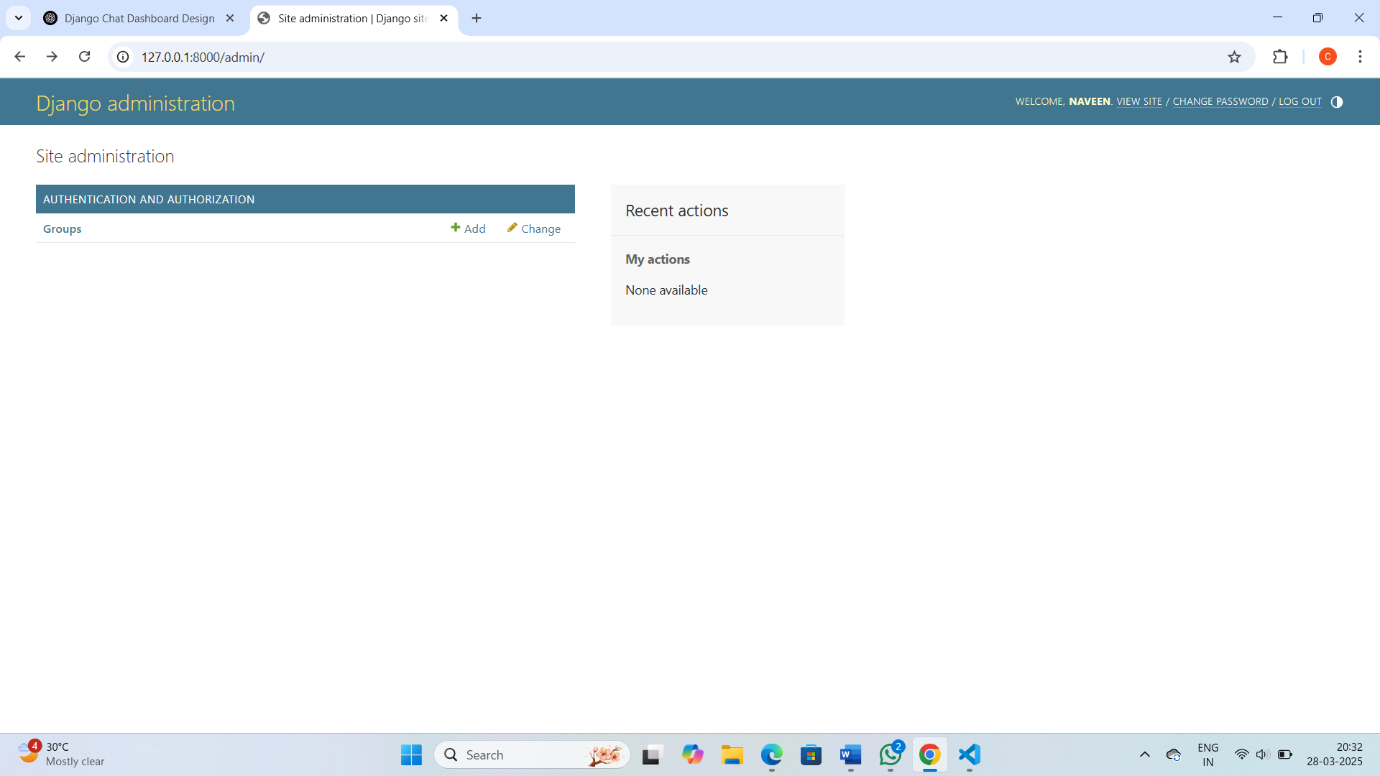
1. Ensure that 'django.contrib.admin' is present in the INSTALLED\_APPS list inside settings.py.
2. Run **migrations** to apply the necessary database changes:

python manage.py migrate

**Log in page for admin:**



**Interface for admin page:**



**2. Login Template (login.html)**

1. Provides a user-friendly interface for authentication.
2. Contains a login form where users enter their credentials.
3. Uses Django’s built-in authentication system (django.contrib.auth).
4. **Login Form:** Contains fields for username and password.
5. **Messages Display:** If login fails, error messages are shown using {% if messages %}.

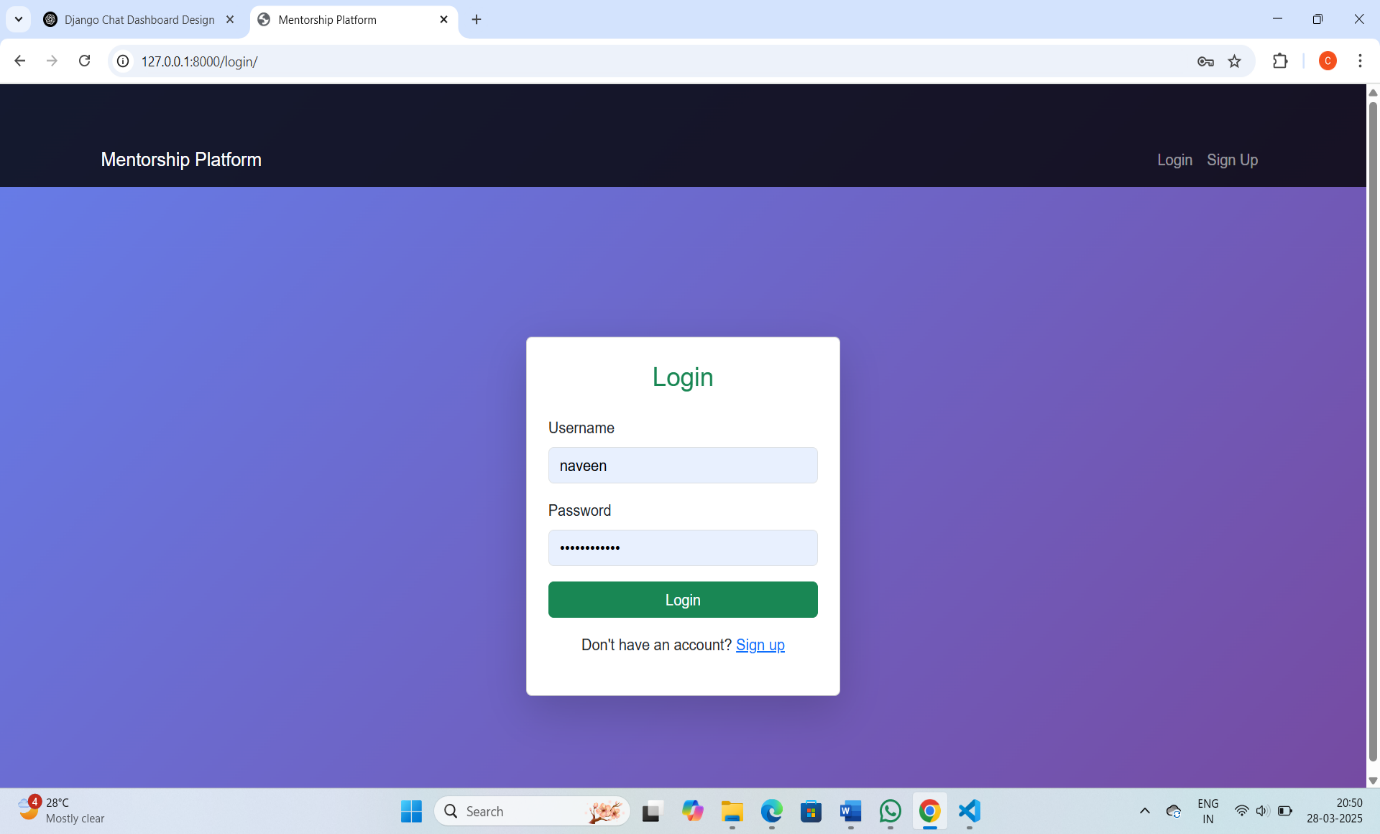
**Code :**

|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="d-flex justify-content-center align-items-center" style="height: 80vh;">***  ***<div class="card p-4 text-dark shadow-lg" style="width: 350px;">***  ***<h3 class="text-center text-success animate\_\_animated animate\_\_fadeInDown">Login</h3>***  ***<form method="POST" class="mt-3">***  ***{% csrf\_token %}***  ***<div class="mb-3">***  ***<label class="form-label">Username</label>***  ***{{ form.username }}***  ***</div>***  ***<div class="mb-3">***  ***<label class="form-label">Password</label>***  ***{{ form.password }}***  ***</div>***  ***<button type="submit" class="btn btn-success w-100">Login</button>***  ***</form>***  ***<p class="text-center mt-3">Don't have an account?***  ***<a href="{% url 'signup' %}" class="text-primary">Sign up</a>***  ***</p>***  ***</div>***  ***</div>***  ***{% endblock %}*** |

**How It Works:**

1. The form submits data to Django’s login view.
2. If credentials are valid, users are redirected to the dashboard; otherwise, an error message is shown.

**OUTPUT:**



**3. Signup Template (signup.html)**

1. Allows new users to register.
2. Captures essential user details like username, email, and password.
3. Uses Django's UserCreationForm for easy handling of user authentication.

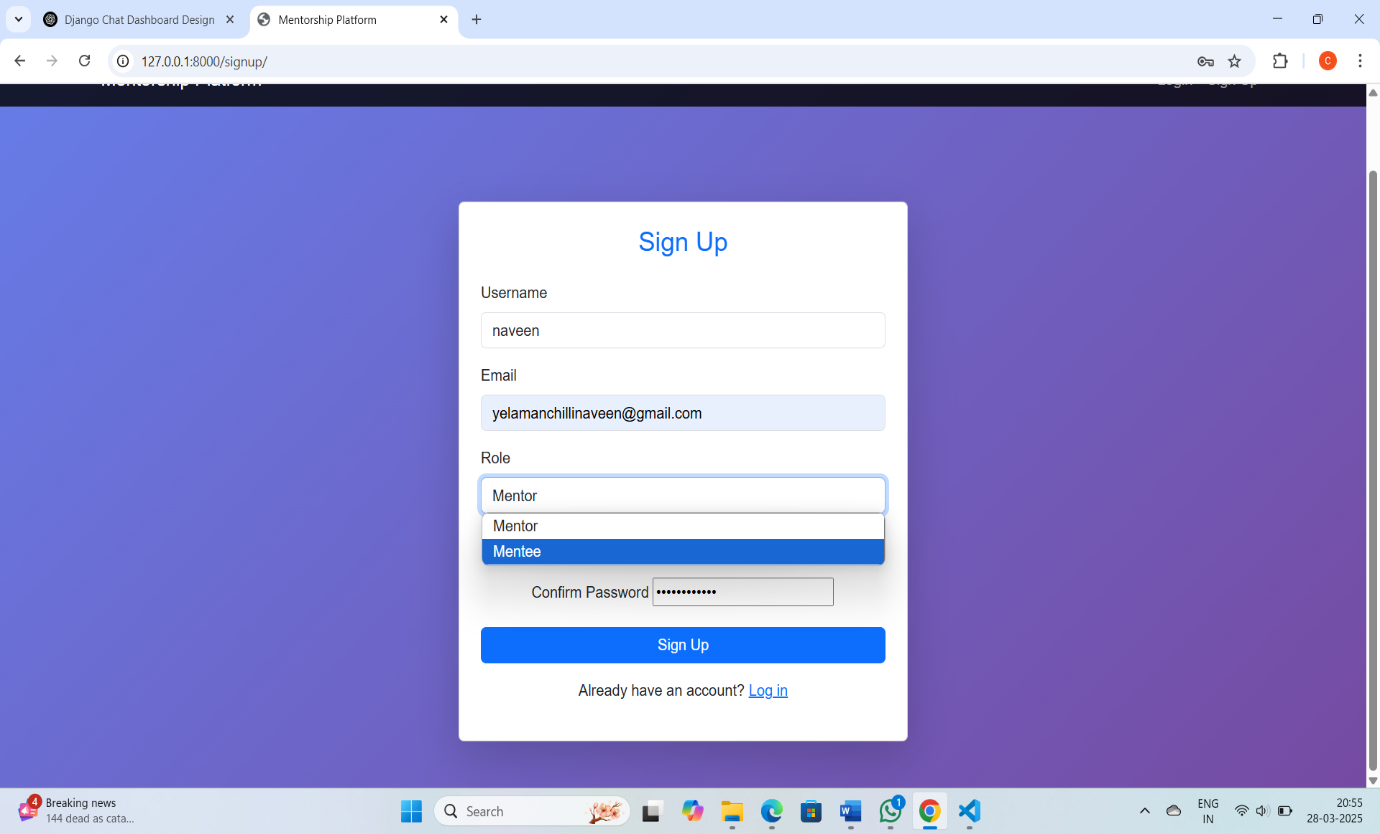
|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="d-flex justify-content-center align-items-center" style="height: 90vh;">***  ***<div class="card p-4 text-dark shadow-lg" style="width: 500px;">***  ***<h3 class="text-center text-primary animate\_\_animated animate\_\_fadeInDown">Sign Up</h3>***  ***<form method="POST" class="mt-3">***  ***{% csrf\_token %}***  ***<div class="mb-3">***  ***<label class="form-label">Username</label>***  ***{{ form.username }}***  ***</div>***  ***<div class="mb-3">***  ***<label class="form-label">Email</label>***  ***{{ form.email }}***  ***</div>***  ***<div class="mb-3">***  ***<label class="form-label">Role</label>***  ***{{ form.role }}***  ***</div>***    ***<!-- Password and Confirm Password Fields in Separate Rows, Centered -->***  ***<div class="d-flex flex-column align-items-center">***  ***<div class="mb-3 w-75">***  ***<label class="form-label">Password</label>***  ***{{ form.password1 }}***  ***</div>***  ***<div class="mb-3 w-75">***  ***<label class="form-label">Confirm Password</label>***  ***{{ form.password2 }}***  ***</div>***  ***</div>***  ***<button type="submit" class="btn btn-primary w-100">Sign Up</button>***  ***</form>***  ***<p class="text-center mt-3">Already have an account?***  ***<a href="{% url 'login' %}" class="text-primary">Log in</a>***  ***</p>***  ***</div>***  ***</div>***  ***{% endblock %}*** |

1. **Registration Form:** Includes fields for username, email, password, and password confirmation.
2. **Validation Messages:** Ensures strong passwords and prevents duplicate usernames.

**How It Works:**

1. Submits user data to the signup view.
2. After successful registration, redirects users to the login page.

**OUTPUT:**



**4. Dashboard Template (dashboard.html)**

1. Serves as the home page after login.
2. Displays an overview of mentorship activities.
3. Shows pending mentorship requests, upcoming sessions, and notifications.

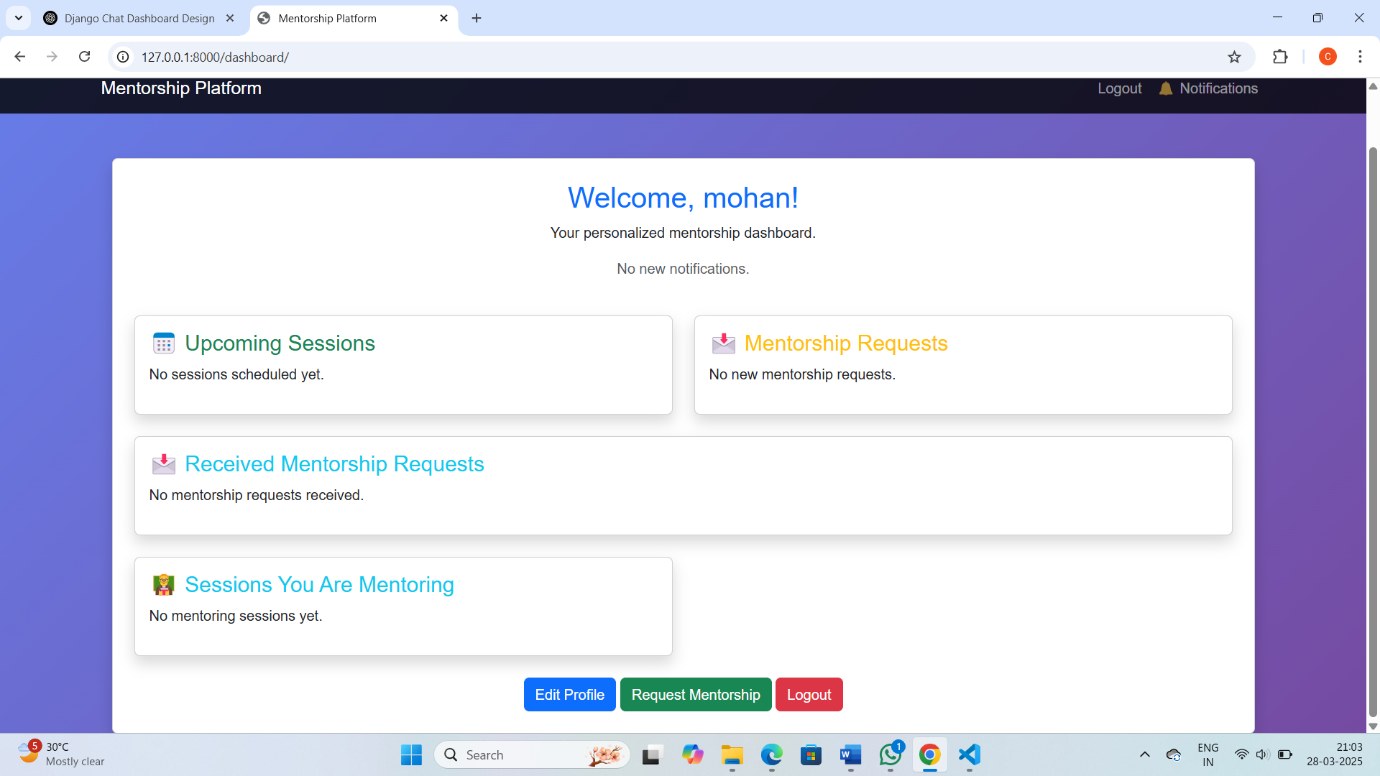
|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="container mt-4">***  ***<div class="card shadow p-4">***  ***<h2 class="text-center text-primary">Welcome, {{ user.username }}!</h2>***  ***<p class="text-center">Your personalized mentorship dashboard.</p>***  ***<!-- Notifications Section -->***  ***{% if unseen\_notifications %}***  ***<div class="alert alert-info">***  ***<strong>🔔 Notifications:</strong>***  ***<ul>***  ***{% for notification in unseen\_notifications %}***  ***<li>{{ notification.message }}</li>***  ***{% endfor %}***  ***</ul>***  ***</div>***  ***{% else %}***  ***<p class="text-center text-muted">No new notifications.</p>***  ***{% endif %}***  ***<div class="row mt-4">***  ***<!-- Upcoming Sessions -->***  ***<div class="col-md-6">***  ***<div class="card shadow p-3">***  ***<h4 class="text-success">📅 Upcoming Sessions</h4>***  ***{% if upcoming\_sessions %}***  ***<ul class="list-group">***  ***{% for session in upcoming\_sessions %}***  ***<li class="list-group-item">***  ***<strong>Mentee:</strong> {{ session.mentee.username }} <br>***  ***<strong>Date & Time:</strong> {{ session.scheduled\_time }} <br>***  ***<strong>Notes:</strong> {{ session.notes|default:"No notes" }}***  ***</li>***  ***{% endfor %}***  ***</ul>***  ***{% else %}***  ***<p>No sessions scheduled yet.</p>***  ***{% endif %}***  ***</div>***  ***</div>***  ***<!-- Mentorship Requests -->***  ***<div class="col-md-6">***  ***<div class="card shadow p-3">***  ***<h4 class="text-warning">📩 Mentorship Requests</h4>***  ***{% if pending\_requests %}***  ***<p>You have {{ pending\_requests.count }} new mentorship requests!</p>***  ***<ul class="list-group">***  ***{% for request in pending\_requests %}***  ***<li class="list-group-item">***  ***<strong>From:</strong> {{ request.mentee.username }}***  ***<br>***  ***<p>Request ID: {{ request.id }}</p>***  ***<p>Status: {{ request.status }}</p>***  ***<a href="{% url 'accept\_request' request.id %}" class="btn btn-success btn-sm">✅ Accept</a>***  ***<a href="{% url 'decline\_request' request.id %}" class="btn btn-danger btn-sm">❌ Decline</a>***  ***<a href="{% url 'schedule\_session' request.id %}" class="btn btn-primary btn-sm">📅 Schedule Session</a>***  ***</li>***  ***{% endfor %}***  ***</ul>***  ***{% else %}***  ***<p>No new mentorship requests.</p>***  ***{% endif %}***  ***</div>***  ***</div>***  ***</div>***  ***<!-- Received Mentorship Requests -->***  ***<div class="row mt-4">***  ***<div class="col-md-12">***  ***<div class="card shadow p-3">***  ***<h4 class="text-info">📩 Received Mentorship Requests</h4>***  ***{% if received\_requests %}***  ***<table class="table table-bordered table-striped">***  ***<thead class="table-dark">***  ***<tr>***  ***<th>Mentee</th>***  ***<th>Message</th>***  ***<th>Actions</th>***  ***</tr>***  ***</thead>***  ***<tbody>***  ***{% for request in received\_requests %}***  ***<tr>***  ***<td>{{ request.mentee.username }}</td>***  ***<td>{{ request.message }}</td>***  ***<td>***  ***<a href="{% url 'accept\_request' request.id %}" class="btn btn-success btn-sm">Accept</a>***  ***<a href="{% url 'decline\_request' request.id %}" class="btn btn-danger btn-sm">Reject</a>***  ***<a href="{% url 'chat' user\_id=request.mentee.id %}" class="btn btn-info btn-sm">Chat</a>***  ***<a href="{% url 'schedule\_session' request.id %}" class="btn btn-warning btn-sm">Schedule Session</a>***  ***</td>***  ***</tr>***  ***{% endfor %}***  ***</tbody>***  ***</table>***  ***{% else %}***  ***<p>No mentorship requests received.</p>***  ***{% endif %}***  ***</div>***  ***</div>***  ***</div>***  ***<!-- Sessions You Are Mentoring -->***  ***<div class="row mt-4">***  ***<div class="col-md-6">***  ***<div class="card shadow p-3">***  ***<h4 class="text-info">👨‍🏫 Sessions You Are Mentoring</h4>***  ***{% if mentoring\_sessions %}***  ***<ul class="list-group">***  ***{% for session in mentoring\_sessions %}***  ***<li class="list-group-item">***  ***<strong>Topic:</strong> {{ session.topic }} <br>***  ***<strong>Date & Time:</strong> {{ session.scheduled\_time }} <br>***  ***<strong>Mentee:</strong> {{ session.mentee.username }} <br>***  ***<strong>Notes:</strong> {{ session.notes|default:"No notes" }}***  ***</li>***  ***{% endfor %}***  ***</ul>***  ***{% else %}***  ***<p>No mentoring sessions yet.</p>***  ***{% endif %}***  ***</div>***  ***</div>***  ***</div>***  ***<!-- Action Buttons -->***  ***<div class="text-center mt-4">***  ***<a href="{% url 'profile' %}" class="btn btn-primary">Edit Profile</a>***  ***<a href="{% url 'send\_request' %}" class="btn btn-success">Request Mentorship</a>***  ***<a href="{% url 'logout' %}" class="btn btn-danger">Logout</a>***  ***</div>***  ***</div>***  ***</div>***  ***{% endblock %}*** |

1. Mentorship Requests Section: Displays requests with buttons to accept, decline, or schedule.
2. Upcoming Sessions: Lists scheduled mentorship sessions with time and date.
3. Profile Navigation: Links to edit the profile, request mentorship, and log out.

**How It Works:**

1. Fetches mentorship data using Django’s ORM (MentorshipRequest.objects.filter()).
2. Uses {% for request in mentorship\_requests %} to dynamically list requests

**OUTPUT:**



**5. Chat Template (chat.html)**

1. Enables real-time messaging between mentors and mentees.
2. Displays messages in a chat box with timestamps.

**Code:**

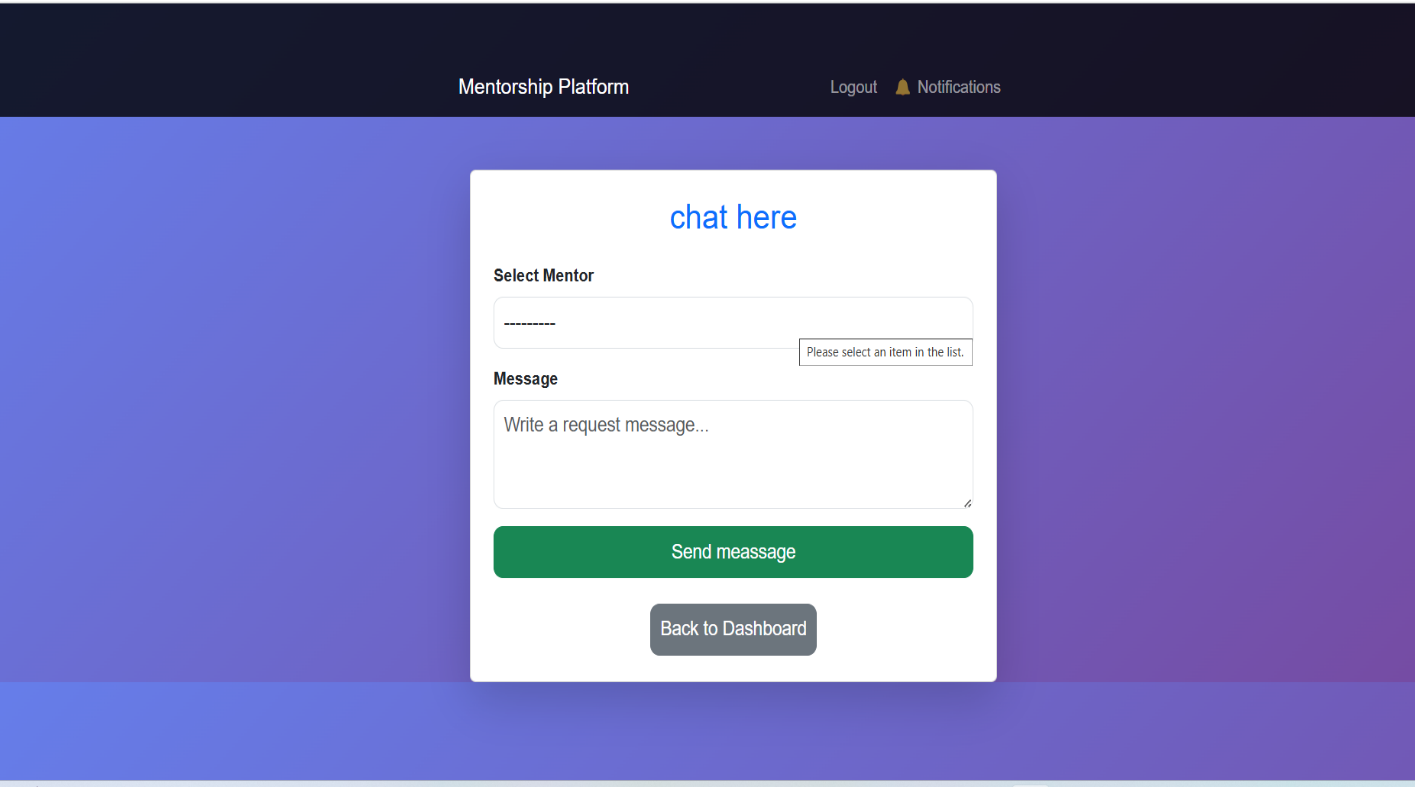
|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="container mt-4">***  ***<h2 class="text-primary">Chat with {{ other\_user.username }}</h2>***  ***<div class="card shadow p-3 chat-box" style="height: 400px; overflow-y: scroll;">***  ***{% for message in messages %}***    ***<div class="mb-2 {% if message.sender == request.user %} text-end {% else %} text-start {% endif %}">***  ***<strong>{{ message.sender.username }}</strong>: {{ message.content }}***  ***<br><small class="text-muted">{{ message.timestamp }}</small>***  ***</div>***  ***{% endfor %}***  ***</div>***  ***<form method="POST" class="mt-3">***  ***{% csrf\_token %}***  ***{{ form.as\_p }}***  ***<button type="submit" class="btn btn-primary">Send</button>***  ***</form>***  ***</div>***  ***{% endblock %}*** |

1. Message List: Shows past messages using {% for message in messages %}.
2. **Message Input Box:** Allows users to type and send new messages.
3. **WebSocket or AJAX Integration:** Ensures real-time chat updates without refreshing the page.

**How It Works:**

1. Fetches messages from the database.
2. Uses JavaScript (or Django Channels for WebSockets) to send and receive messages.

**Output :**



**6. Mentorship Request Template (mentorship\_request.html)**

**Functionality:**

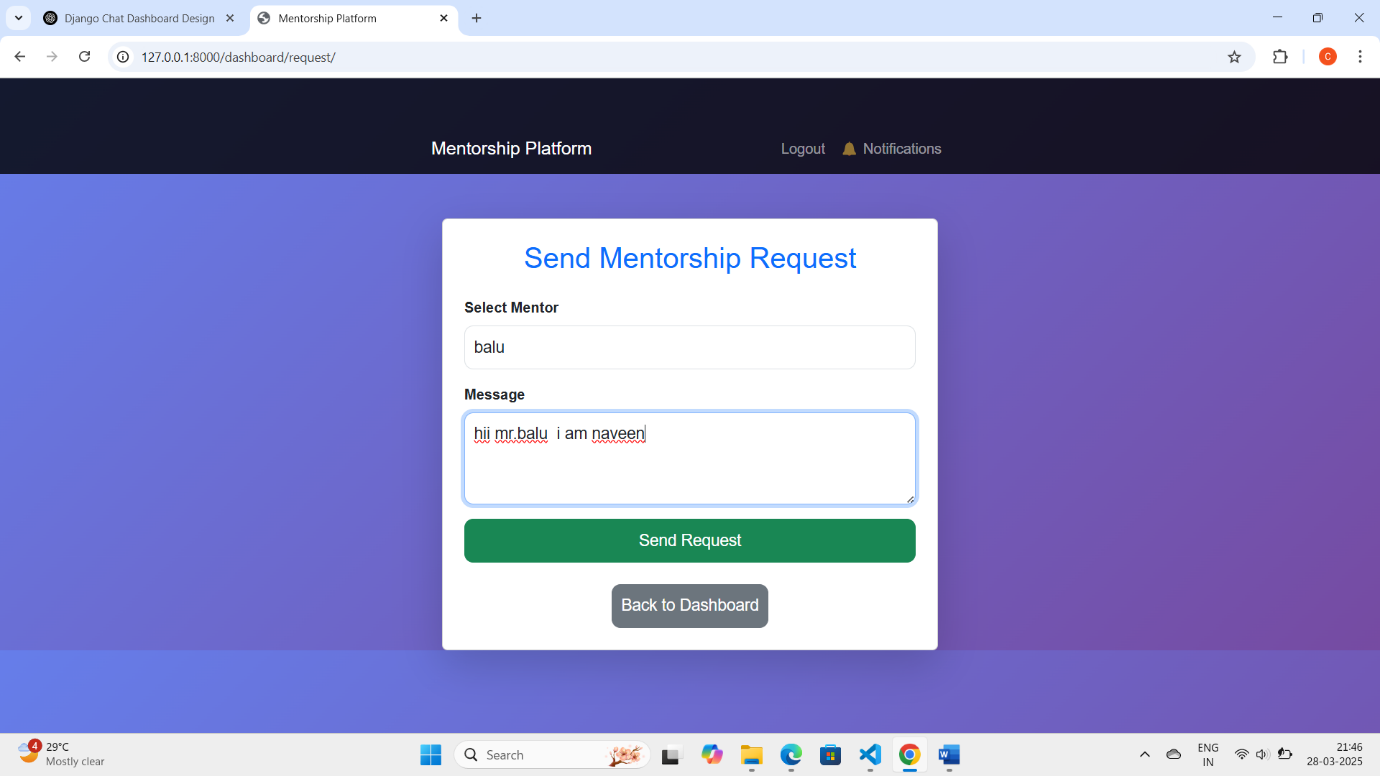
1. Displays all mentorship requests a user has received.
2. Allows users to accept, decline, or schedule a session.

**Code:**

|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="container mt-5">***  ***<div class="card shadow-lg p-4 rounded">***  ***<h2 class="text-center text-primary mb-4">Send Mentorship Request</h2>***  ***<form method="POST">***  ***{% csrf\_token %}***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Select Mentor</label>***  ***{{ form.mentor }}***  ***</div>***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Message</label>***  ***{{ form.message }}***  ***</div>***  ***<div class="d-grid">***  ***<button type="submit" class="btn btn-success btn-lg">Send Request</button>***  ***</div>***  ***</form>***  ***<div class="text-center mt-4">***  ***<a href="{% url 'dashboard' %}" class="btn btn-secondary btn-lg">Back to Dashboard</a>***  ***</div>***  ***</div>***  ***</div>***  ***<style>***  ***.container {***  ***max-width: 600px;***  ***}***  ***.form-control {***  ***font-size: 18px;***  ***padding: 10px;***  ***border-radius: 10px;***  ***}***  ***.btn-lg {***  ***font-size: 18px;***  ***padding: 10px;***  ***border-radius: 10px;***  ***}***  ***</style>***  ***{% endblock %}*** |

1. **Mentorship Request List:** Uses {% for request in mentorship\_requests %} to show all requests.
2. **Action Buttons:** Each request has buttons to accept, decline, or schedule.
3. **How It Works:**
4. Fetches pending mentorship requests using Django’s ORM.
5. Uses Django forms or JavaScript to submit responses.

**OUTPUT:**



**7. Profile Template (profile.html)**

1. Allows users to update their profile information.
2. Enables users to upload a profile picture.
3. Saves user bio, skills, and experience.

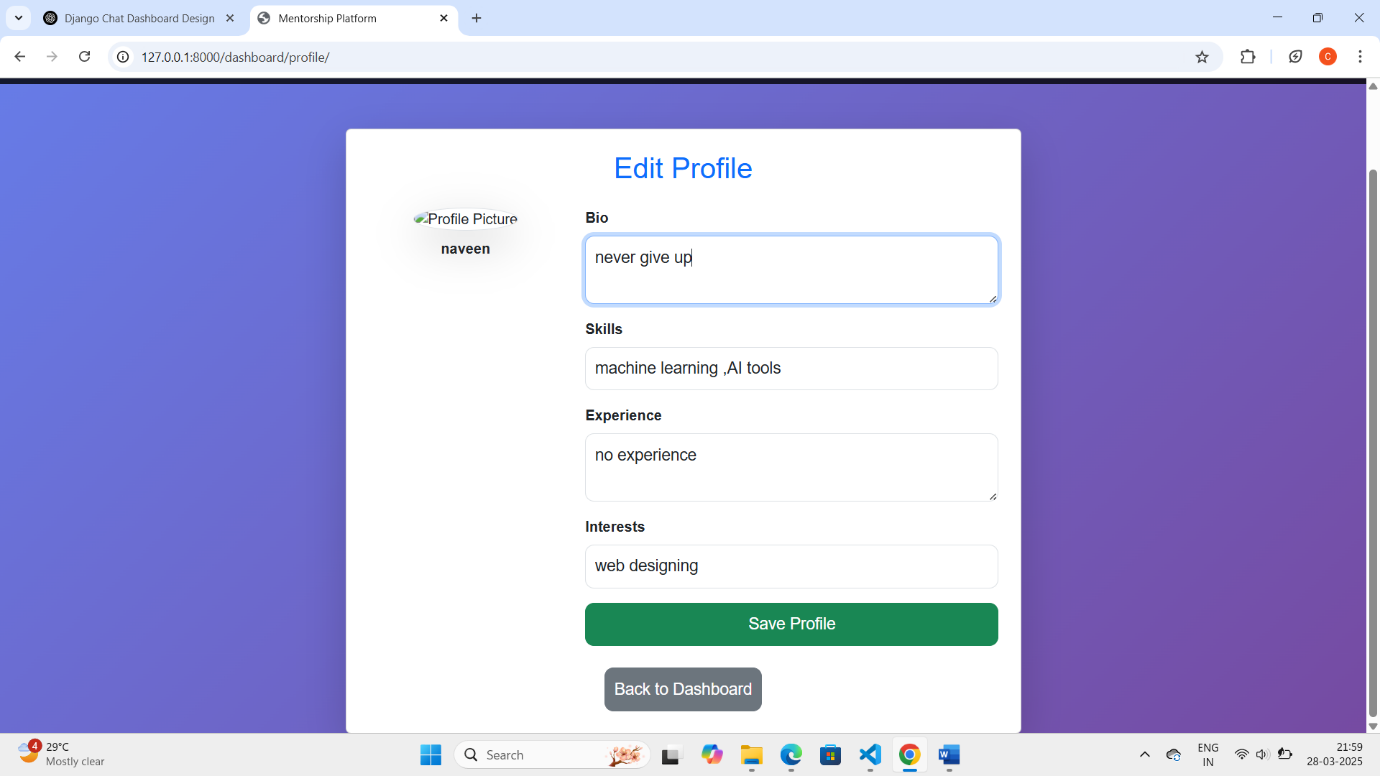
|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="container mt-5">***  ***<div class="card shadow-lg p-4 rounded">***  ***<h2 class="text-center text-primary mb-4">Edit Profile</h2>***  ***<div class="row">***  ***<!-- Profile Picture Section -->***  ***<div class="col-md-4 text-center">***  ***<label for="profile-picture-input">***  ***<img src="{{ user.profile.profile\_picture.url }}" alt="Profile Picture" class="rounded-circle shadow-lg border" width="150" height="150">***  ***</label>***  ***<br>***  ***<input type="file" id="profile-picture-input" name="profile\_picture" style="display: none;" onchange="previewImage(event)">***  ***<p class="mt-2"><strong>{{ user.username }}</strong></p>***  ***</div>***  ***<!-- Profile Form Section -->***  ***<div class="col-md-8">***  ***<form method="POST" enctype="multipart/form-data">***  ***{% csrf\_token %}***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Bio</label>***  ***<textarea class="form-control form-control-lg" name="bio" rows="2" placeholder="Write something about yourself...">{{ form.bio.value }}</textarea>***  ***</div>***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Skills</label>***  ***<input type="text" class="form-control form-control-lg" name="skills" value="{{ form.skills.value }}" placeholder="E.g., Python, Machine Learning">***  ***</div>***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Experience</label>***  ***<textarea class="form-control form-control-lg" name="experience" rows="2" placeholder="Describe your experience...">{{ form.experience.value }}</textarea>***  ***</div>***  ***<div class="form-group mb-3">***  ***<label class="form-label fw-bold">Interests</label>***  ***<input type="text" class="form-control form-control-lg" name="interests" value="{{ form.interests.value }}" placeholder="E.g., AI, Web Development">***  ***</div>***  ***<div class="d-grid">***  ***<button type="submit" class="btn btn-success btn-lg">Save Profile</button>***  ***</div>***  ***</form>***  ***</div>***  ***</div>***  ***<div class="text-center mt-4">***  ***<a href="{% url 'dashboard' %}" class="btn btn-secondary btn-lg">Back to Dashboard</a>***  ***</div>***  ***</div>***  ***</div>***  ***<!-- JavaScript for Profile Picture Preview -->***  ***<script>***  ***function previewImage(event) {***  ***var reader = new FileReader();***  ***reader.onload = function(){***  ***var output = document.querySelector("label[for='profile-picture-input'] img");***  ***output.src = reader.result;***  ***};***  ***reader.readAsDataURL(event.target.files[0]);***  ***}***  ***</script>***  ***<!-- CSS Styling -->***  ***<style>***  ***.container {***  ***max-width: 800px;***  ***}***  ***.form-control {***  ***font-size: 18px;***  ***padding: 10px;***  ***border-radius: 10px;***  ***}***  ***.btn-lg {***  ***font-size: 18px;***  ***padding: 10px;***  ***border-radius: 10px;***  ***}***  ***.rounded-circle {***  ***border: 3px solid #007bff;***  ***}***  ***</style>***  ***{% endblock %}*** |

1. **Form Fields:** Includes fields for bio, skills, and experience.
2. **Image Upload Preview:** Uses JavaScript to preview profile pictures before uploading.

**How It Works:**

1. Uses Django’s UserProfile model to save and fetch data.
2. Updates the database when users submit change

**OUTPUT:**



**8. Schedule Session Template (schedule\_session.html)**

1. Allows users to set up mentorship sessions.
2. Users select date, time, and mentor.

|  |
| --- |
| • ***{% extends "base.html" %}***  ***•***  ***• {% block content %}***  ***• <div class="container mt-5">***  ***• <h2>Schedule a Mentorship Session</h2>***  ***• <form method="POST">***  ***• {% csrf\_token %}***  ***• {{ form.as\_p }}***  ***• <button type="submit" class="btn btn-success">Schedule Session</button>***  ***• </form>***  ***• </div>***  ***• {% endblock %}*** |

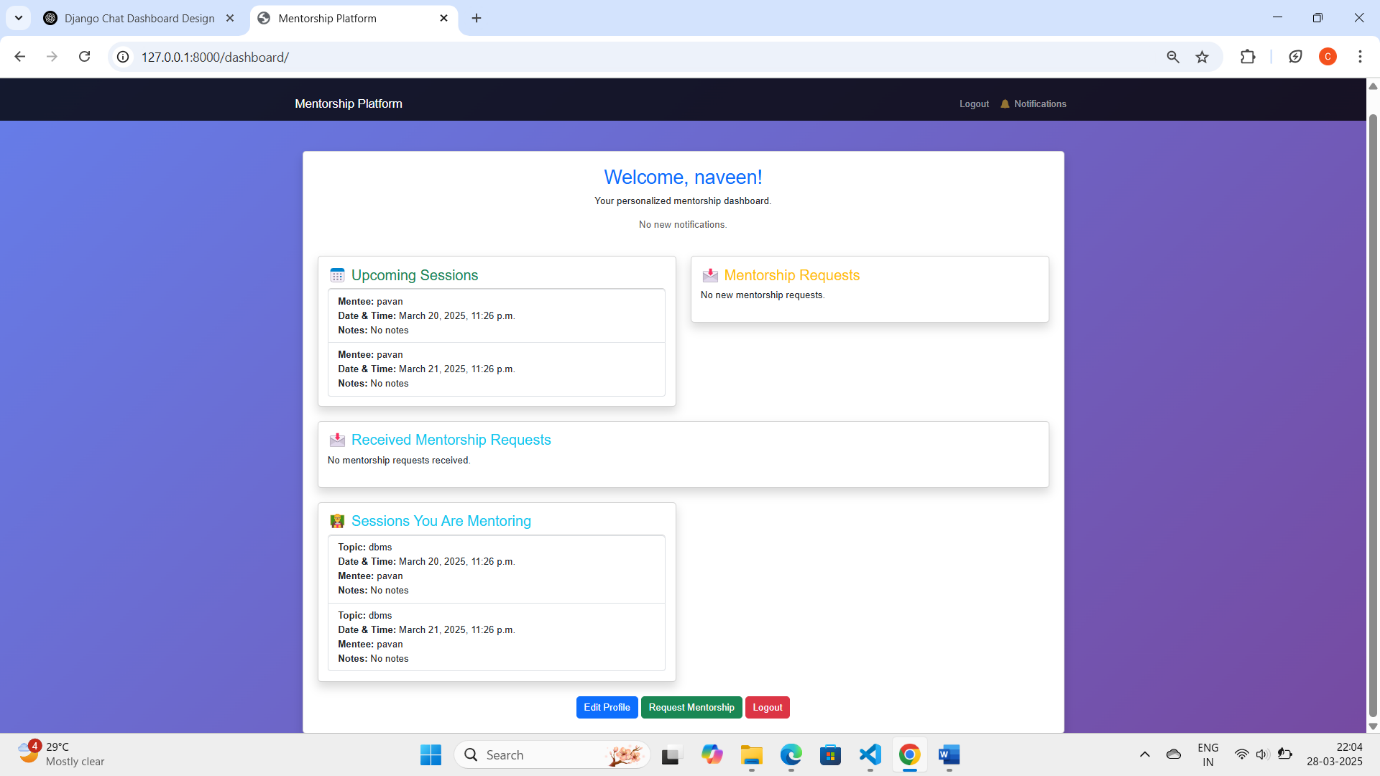
1. Date & Time Picker: Ensures valid scheduling.
2. Submit Button: Sends scheduling request to the database.

**How It Works:**

1. Uses Django forms to validate input.
2. Saves session details in the MentorshipSession model.

These templates form the foundation of the mentorship platform, ensuring smooth interaction between mentors and mentees. They dynamically fetch and display data, leveraging Django’s powerful backend capabilities.

**OUTPUT:**



**9.Notification Template (notifications.html)**

1. Displays real-time notifications related to mentorship requests, session updates, and messages.
2. Helps users stay updated on important actions without checking different sections manually.
3. Uses Django’s template engine to dynamically load notifications from the database.

|  |
| --- |
| ***{% extends 'core/base.html' %}***  ***{% block content %}***  ***<div class="container">***  ***<h2 class="text-center mt-4">Notifications 🔔</h2>***  ***<ul class="list-group mt-3">***  ***{% for notification in notifications %}***  ***<li class="list-group-item">***  ***{{ notification.message }}***  ***<span class="text-muted float-end">{{ notification.timestamp }}</span>***  ***</li>***  ***{% empty %}***  ***<li class="list-group-item">No new notifications.</li>***  ***{% endfor %}***  ***</ul>***  ***<a href="{% url 'dashboard' %}" class="btn btn-primary mt-3">Back to Dashboard</a>***  ***</div>***  ***{% endblock %}*** |

**Notification List:**

1. Fetches all notifications for the logged-in user.
2. Uses {% for notification in notifications %} to iterate and display them.

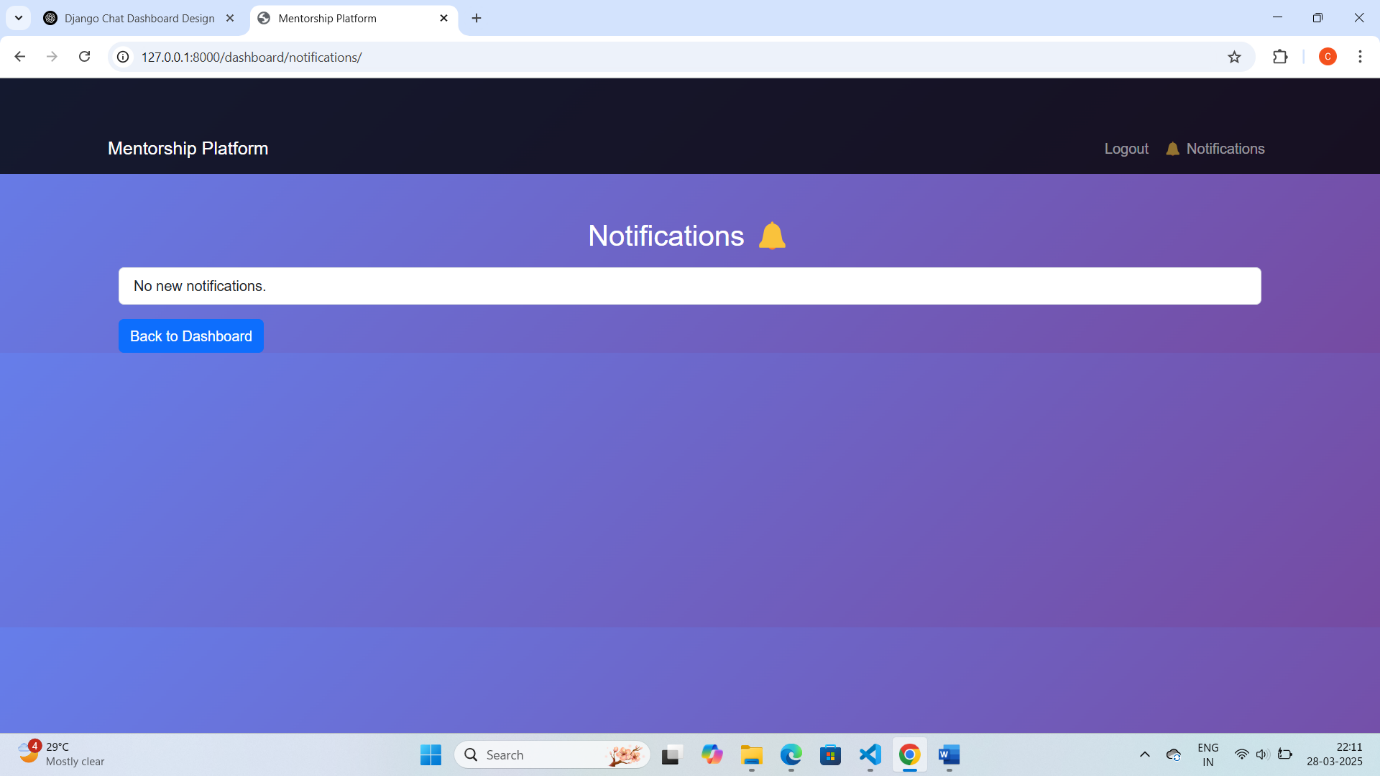
**Notification Types:**

1. New mentorship request.
2. Session confirmation or cancellation.
3. Message received in chat**.**

**How It Works:**

1. The Django view fetches unread notifications using Notification.objects.filter(user=request.user, is\_read=False).
2. The template loops through them and displays each notification with a timestamp.
3. JavaScript (AJAX) checks for new notifications periodically.
4. When a notification is clicked, it is marked as read and removed from the unread list.

**Output:**





**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

4. Class : II B-Tech II Semester

5. Academic Year : 2024-25

6. Name of Experiment : database integration and configuration-SOL LITE

7. Date of Experiment : 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 andObservations | 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 |  |

8. Date of Submission of Report : 21-02-2025

**DATE:** signature of faculty

Database Integration and Configuration-SQL LITE

Databasse 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

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:
   1. Search for "Environment Variables" in the Start menu.
   2. Click on "Environment Variables."
   3. In "System variables," select "Path" and click "Edit."

Add the folder path (e.g., C:\sqlite) and click OK.

**On macOS:**

sqlite3 –version

Step 3: 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',***  ***}***  ***}*** |

1. Step 4: Run Migrations to Create the Database
2. After setting up, run:
3. python manage.py migrate
4. Step 5: Open the Django DB Shell
5. Make sure your virtual environment is activated, then use the following command:
6. python manage.py db shell
7. Step 6: Common SQLite Commands
8. Once you're inside the SQLite shell, you can use the following commands:
9. List All Tables:
10. .tables
11. Step7: View Table Schema:
12. .schema table\_name;
13. Step8: Show All Data in a Table:
14. SELECT \* FROM table\_name;
15. Exit the SQLite Shell:
16. .exit



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

4. Class : II B-Tech II Semester

5. Academic Year : 2024-25

6. Name of Experiment : Handlling froms in django

7. Date of Experiment : 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 andObservations | 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 |  |

8. Date of Submission of Report : 21-02-2025

**DATE:** signature of faculty

**Handling 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:**

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

**CORE/FORMS.PY**

|  |
| --- |
| ***from django import forms***  ***from django.contrib.auth.forms import UserCreationForm, AuthenticationForm***  ***from .models import CustomUser***  ***class SignupForm(UserCreationForm):***  ***role = forms.ChoiceField(***  ***choices=CustomUser.ROLE\_CHOICES,***  ***widget=forms.Select(attrs={'class': 'form-control'})***  ***)***  ***class Meta:***  ***model = CustomUser***  ***fields = ['username', 'email', 'role', 'password1', 'password2']***  ***widgets = {***  ***'username': forms.TextInput(attrs={'class': 'form-control'}),***  ***'email': forms.EmailInput(attrs={'class': 'form-control'}),***  ***'password1': forms.PasswordInput(attrs={'class': 'form-control'}),***  ***'password2': forms.PasswordInput(attrs={'class': 'form-control'}),***  ***}***  ***class LoginForm(AuthenticationForm):***  ***username = forms.CharField(widget=forms.TextInput(attrs={'class': 'form-control'}))***  ***password = forms.CharField(widget=forms.PasswordInput(attrs={'class': 'form-control'}))*** |

**DASHBOARD/FORMS.PY**

|  |
| --- |
| ***from django import forms***  ***from .models import Profile, MentorshipRequest, Message, MentorshipSession***  ***class ProfileForm(forms.ModelForm):***  ***class Meta:***  ***model = Profile***  ***fields = ['bio', 'skills', 'experience', 'interests', 'profile\_picture']***  ***widgets = {***  ***'bio': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),***  ***'skills': forms.TextInput(attrs={'class': 'form-control'}),***  ***'experience': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),***  ***'interests': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),***  ***'profile\_picture': forms.ClearableFileInput(attrs={'class': 'form-control'}),***  ***}***  ***class MentorshipRequestForm(forms.ModelForm):***  ***class Meta:***  ***model = MentorshipRequest***  ***fields = ['mentor', 'message']***  ***widgets = {***  ***'mentor': forms.Select(attrs={'class': 'form-control'}),***  ***'message': forms.Textarea(attrs={'class': 'form-control', 'rows': 3, 'placeholder': 'Write a request message...'}),***  ***}***  ***class MessageForm(forms.ModelForm):***  ***class Meta:***  ***model = Message***  ***fields = ['content']***  ***widgets = {***  ***'content': forms.Textarea(attrs={'rows': 3, 'class': 'form-control', 'placeholder': 'Type a message...'})***  ***}***  ***class MentorshipSessionForm(forms.ModelForm):***  ***scheduled\_time = forms.DateTimeField(***  ***widget=forms.DateTimeInput(attrs={'type': 'datetime-local', 'class': 'form-control'}),***  ***input\_formats=['%Y-%m-%dT%H:%M']***  ***)***  ***class Meta:***  ***model = MentorshipSession***  ***fields = ['scheduled\_time', 'notes']***  ***widgets = {***  ***'notes': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),***  ***}*** |



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 07-03-2025

**DATE:** signature of faculty

**Defining and Using Models:**

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

***Apply Models:***

Create the Model in models.py:

Write your models inside the models.py file.

**CORE/MODELS.PY:**

|  |
| --- |
| ***from django.contrib.auth.models import AbstractUser***  ***from django.db import models***  ***# Custom User Model with Role***  ***class CustomUser(AbstractUser):***  ***ROLE\_CHOICES = (***  ***('mentor', 'Mentor'),***  ***('mentee', 'Mentee'),***  ***)***  ***role = models.CharField(max\_length=10, choices=ROLE\_CHOICES)***  ***bio = models.TextField(blank=True, null=True)***  ***profile\_picture = models.ImageField(upload\_to='profile\_pics/', blank=True, null=True)***  ***def \_\_str\_\_(self):***  ***return self.username*** |

**DASHBOARD/MODELS.PY**

|  |
| --- |
| ***from django.db import models***  ***from django.conf import settings***  ***class Profile(models.Model):***  ***user = models.OneToOneField(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)  # Corrected***  ***bio = models.TextField(blank=True, null=True)***  ***skills = models.CharField(max\_length=255, blank=True, null=True)***  ***experience = models.TextField(blank=True, null=True)***  ***interests = models.TextField(blank=True, null=True)***  ***profile\_picture = models.ImageField(upload\_to='profile\_pics/', default='default.jpg')***  ***def \_\_str\_\_(self):***  ***return self.user.username***  ***class MentorshipRequest(models.Model):***  ***STATUS\_CHOICES = [***  ***('pending', 'Pending'),***  ***('accepted', 'Accepted'),***  ***('declined', 'Declined'),***  ***]***  ***mentee = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='mentee\_requests')***  ***mentor = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='mentor\_requests')***  ***message = models.TextField()***  ***status = models.CharField(max\_length=10, choices=STATUS\_CHOICES, default='pending')***  ***created\_at = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"{self.mentee} → {self.mentor} ({self.status})"***  ***class Message(models.Model):***  ***sender = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='sent\_messages')***  ***receiver = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='received\_messages')***  ***content = models.TextField()***  ***timestamp = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"From {self.sender} to {self.receiver} at {self.timestamp}"***  ***class MentorshipSession(models.Model):***  ***mentorship\_request = models.OneToOneField('MentorshipRequest', on\_delete=models.CASCADE, null=True, blank=True)***  ***mentor = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='mentor\_sessions')***  ***mentee = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='mentee\_sessions')***  ***scheduled\_time = models.DateTimeField()***  ***topic = models.CharField(max\_length=255)***  ***notes = models.TextField(blank=True, null=True)***  ***created\_at = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"{self.topic} - {self.scheduled\_time.strftime('%Y-%m-%d %H:%M')} (Mentor: {self.mentor}, Mentee: {self.mentee})"***  ***class Notification(models.Model):***  ***user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='notifications')***  ***message = models.TextField()***  ***created\_at = models.DateTimeField(auto\_now\_add=True)***  ***is\_read = models.BooleanField(default=False)***  ***def \_\_str\_\_(self):***  ***return f"Notification for {self.user} - {self.message[:30]}"*** |



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

4. Class : II B-Tech II Semester

5. Academic Year : 2024-25

6. Name of Experiment : Defining and Using Models

7. Date of Experiment : 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 andObservations | 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 |  |

8. Date of Submission of Report : 27-03-2025

**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:

* 1. python manage.py makemigrations
  2. python manage.py migrate

**Register Models in admin.py**:

**Admin.py:**

|  |
| --- |
| ***from django.db import models***  ***from django.contrib.auth.models import User***  ***class Profile(models.Model):***  ***user = models.OneToOneField(User, on\_delete=models.CASCADE)***  ***bio = models.TextField(blank=True, null=True)***  ***profile\_picture = models.ImageField(upload\_to='profiles/', blank=True, null=True)***  ***location = models.CharField(max\_length=100, blank=True, null=True)***  ***def \_\_str\_\_(self):***  ***return f"{self.user.username}'s Profile"***  ***class MentorshipRequest(models.Model):***  ***mentee = models.ForeignKey(User, related\_name="mentee\_requests", on\_delete=models.CASCADE)***  ***mentor = models.ForeignKey(User, related\_name="mentor\_requests", on\_delete=models.CASCADE)***  ***status = models.CharField(max\_length=20, choices=[('pending', 'Pending'), ('accepted', 'Accepted'), ('declined', 'Declined')], default='pending')***  ***created\_at = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"{self.mentee.username} -> {self.mentor.username} ({self.status})"***  ***class MentorshipSession(models.Model):***  ***mentorship\_request = models.OneToOneField(MentorshipRequest, on\_delete=models.CASCADE)***  ***mentor = models.ForeignKey(User, related\_name="mentor\_sessions", on\_delete=models.CASCADE)***  ***mentee = models.ForeignKey(User, related\_name="mentee\_sessions", on\_delete=models.CASCADE)***  ***scheduled\_time = models.DateTimeField()***  ***notes = models.TextField(blank=True, null=True)***  ***def \_\_str\_\_(self):***  ***return f"Session: {self.mentor.username} -> {self.mentee.username} @ {self.scheduled\_time}"***  ***class Message(models.Model):***  ***sender = models.ForeignKey(User, related\_name="sent\_messages", on\_delete=models.CASCADE)***  ***receiver = models.ForeignKey(User, related\_name="received\_messages", on\_delete=models.CASCADE)***  ***content = models.TextField()***  ***timestamp = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"Message from {self.sender.username} to {self.receiver.username}"***  ***class Notification(models.Model):***  ***user = models.ForeignKey(User, on\_delete=models.CASCADE)***  ***message = models.TextField()***  ***is\_read = models.BooleanField(default=False)***  ***created\_at = models.DateTimeField(auto\_now\_add=True)***  ***def \_\_str\_\_(self):***  ***return f"Notification for {self.user.username}: {'Read' if self.is\_read else 'Unread'}"*** |



**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](mailto:it@intugvcev.edu.in)

1. Name of the Laboratory : Django Frame Work Lab

2. Name of the Student : y.naveen

3. Roll No : 23VV1A1260

4. Class : II B-Tech II Semester

5. Academic Year : 2024-25

6. Name of Experiment : deploying Django applications on cloud platform

7. Date of Experiment : 27-03-2025

|  |  |  |  |
| --- | --- | --- | --- |
| **S,NO** | **ABILITY AND ACTIVITY** | **WEIGHTAGE OF MARKS** | **DAY TO DAY EVALUTION SCORE** |
| 1 | Aim Objective, Tools required | 3 |  |
| 2 | Theory, Algorithm andObservations | 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 |  |

8. Date of Submission of Report : 04-04-2025

**DATE:** signature of facul

**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, DigitalOcean, Heroku, or PythonAnywhere.

**Features:**

1. **Scalability** – Handle more users without performance issues.
2. **Security** – Protect user data with SSL and secure databases.
3. **Global Accessibility** – Users can access your app from anywhere.
4. **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/naveen0121k/ peer mentorship platform.git](https://github.com/naveen@0121k/%20peer%20mentorship%20platform.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/naveen0121k /peer](https://github.com/naveen0121k%20/peer)mentorship platform.git

**Front End Wed Developer Certification:**

