JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJADA-VIZIZANAGARAM

VIZIANAGARAM – 5330003

ANDHRA PRADESH

STUDENT FEEDBACK AND SURVEY SYSTEM

By

Yamini desula

(Regd. No. 23VV1A1213)

Batch no: 4

B.Akhil babu

L.Charan kumar

N.Jyothsna

Under the supervision of

MRS.MADHUMITA CHANDA

Department of Information Technology
JNTU-GV CEV

Vizianagaram-533003

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

Regd.No : 23VV1A1213

CERTIFICATE

This is to certify that this is a bonafide record of practical work done by Ms. YAMINI DESULA of II B.Tech II 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 .

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

DANGO FRAME WORK INDEX

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
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**

LIBRARIES

PYTHON LIBRARIES

1. Python Collections – C ontainer Datatypes:

- **Purpose:** Provides specialized container datatypes that support efficient handling of data**.**
- *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.
- **Common Use**: Data manipulation, storing and accessing collections of data in web apps (like user data or API responses).

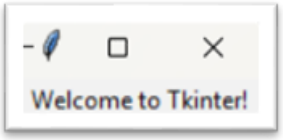
2. Tinker

- **Purpose**: Python's standard library for creating graphical user interfaces (GUIs).
- **Keyfeatures:**
  i. Widgets: Buttons, labels, text boxes, etc.
  ii. Event handling: Respond to user interactions like clicks or key presses.
  iii. 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:



3. Requests - HTTP Requests:

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

2. *Key Features:*

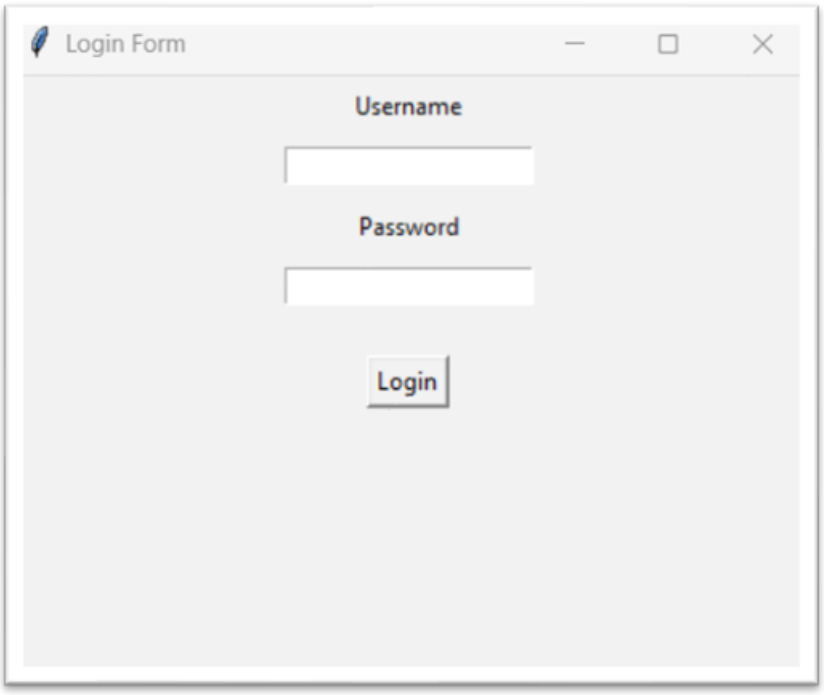
   i. Send GET, POST, PUT, DELETE requests easily.
   ii. Handle request parameters, headers, and cookies.
   iii. Simple error handling and response handling.

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:

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

5. BeautifulSoup4 - Web Scraping:

- **Purpose**: Parses HTML and XML documents to extract data.
- *Key Features:*
  i. Easy navigation and searching within HTML.
  ii. Supports different parsers like html.parser, lxml, and html5lib.
- **Common Use**: Extract data from websites for analysis, e.g., for building data-driven applications

Code:

```
import requests
from bs4 import BeautifulSoup
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:

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

Key Features:

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

ii. Manages serverless architecture and deployment configurations.

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

7. Dash:

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

Key Features:

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

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

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

8. CherryPy:

Purpose: Minimalistic web framework for building web applications.

Key Features:

i. Provides a simple and fast HTTP server.

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

Common Use: Building web applications with a lightweight framework.

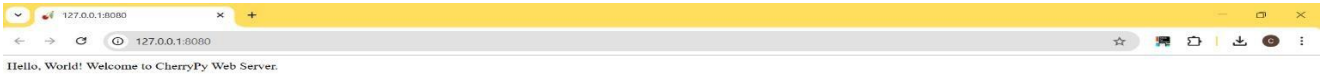
Code:

```
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 :-



9. Flask:

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

Code:

```
from flask import Flask
app = Flask(__name__)
@app.route('/', 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:

10. Web2Py:

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

Key Features:

i. Includes a web-based IDE for development.

ii. Built-in ticketing system and database integration.

Common Use: Enterprise web applications with minimal setup.

11. Bottle:

Purpose: Simple and lightweight WSGI micro-framework.

Key Features:

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

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

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 runn the server

12. Falcon:

Purpose: High-performance framework for building APIs.

Key Features:

i. Focuses on speed and minimalism.

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

Common Use: Building fast, high-performance APIs.

13. CubicWeb:

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

Key Features:

i. Uses a highly modular architecture for development.

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

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

14. Quixote:

Purpose: A web framework designed for simplicity and scalability.

Key Features:

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

ii. Easily extensible, with minimalistic core.

Common Use: Scalable and customizable web applications.

15. Pyramid:

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

Key Features:

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

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

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : introduction 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**

Django: A Web Framework for Python

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. Excellent Documentation

This is one of the main reasons to start learning Django. If we compare Django with other open source technologies, it offers the best documentation in the market.Better documentation of any technology is like a very well-established library for any developer. There, he can search for any function desired with ease with the time involving in the searching purpose only.

2. Python Web-framework

Python is also one of the main reasons people started learning Django. It is that one tool which can solve all your problems and in any kind of operation out there, we can use it. It's very simple and easy to use. All these features are inside Python. In fact, Python is currently the most popular language in the market. It is because of these 2 main features.

3. SEO Optimised

This is a special feature of Django due to which it has edge over others. SEO is Search Engine Optimization as from the name it means that adding your website to the search engine such that it appears in the top results. As we know that the search engines do use some algorithms which sometimes doesn't cooperate much with the web-developer. Since we are creating our website in the human understandable form and they have to add it in the URL form on the server so that its best recognized by the search engine.

4. High Scalability

A lot of MNCs on a worldwide scale uses Django and it gets implement there without any defects or errors. It is the best example of Django being scalable.

5. Versatile in Nature

Django is very versatile in its own Django way. The logical project structure and MVT architecture of Django sometimes seem very limiting. But, that's just opposite because by giving us the files it is providing us with a solid foundation which can then be used to make whichever application we want to create.

6. Offers High Security

Django is super secure. To prove the feature, you can always take examples of lots of websites which are worldwide and posses huge traffic.Django is secure because it covers the loopholes by default which were once left open for the backend developer to complete. Although while using Django you may not feel it but those expert backend developers can tell the quality and security of the work done by Django.

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

```
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
```

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : step by step guide to install 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 INSTALL 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

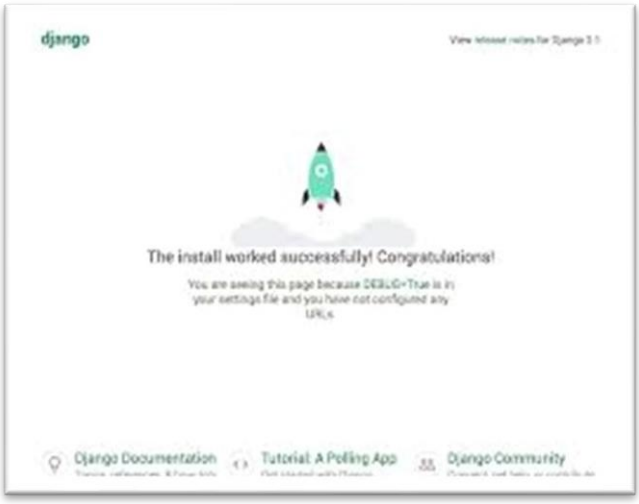
```
Django-admin startproject mysite
```

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



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

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 :

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : exploring Django views
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**

VIEWS:

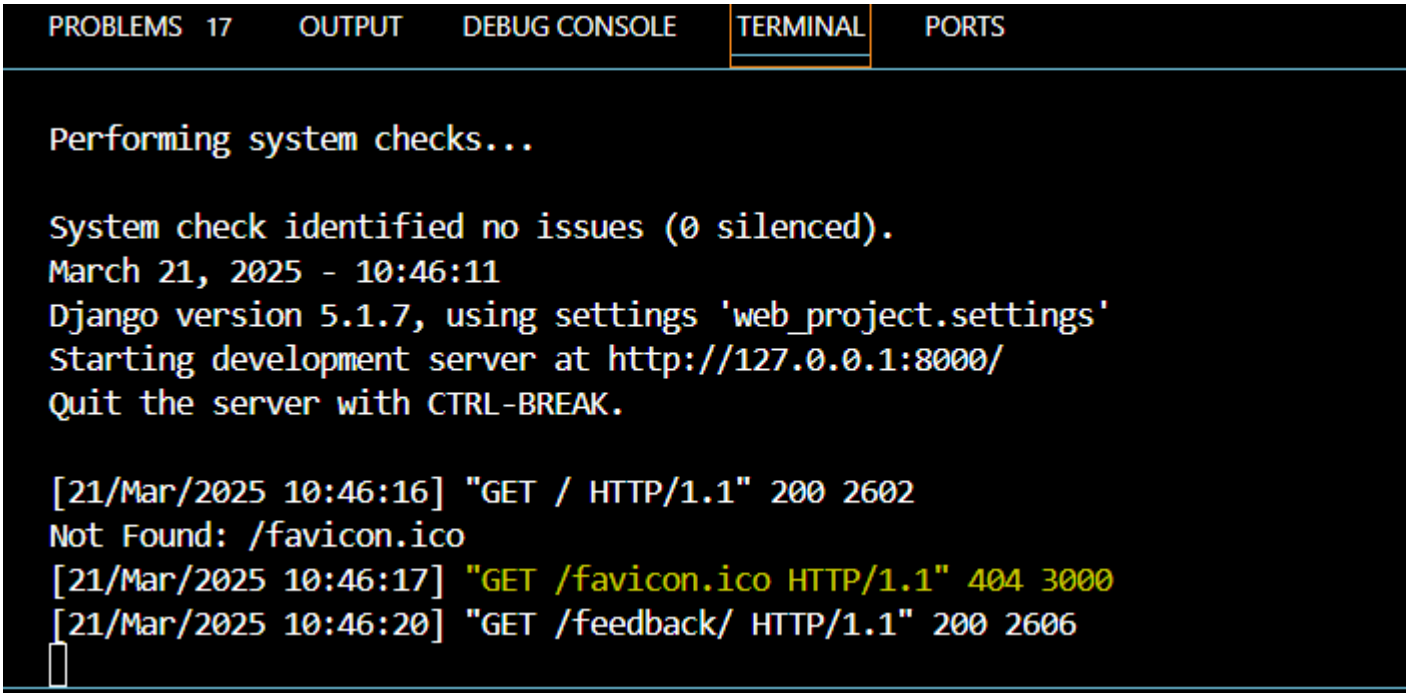
Web_app/views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import Feedback  # Import the Feedback model

# Homepage View
def homepage(request):
    return render(request, 'homepage.html')

# Feedback Form View
def feedback_form(request):
    return render(request, 'feedback.html')

# Handle Feedback Submission
def submit_feedback(request):
if request.method == "POST":
    faculty = request.POST.get("faculty")
    rating = request.POST.get("rating")
    comment = request.POST.get("comment")

    # Save to database
    Feedback.objects.create(faculty=faculty, rating=rating, comment=comment)

    return HttpResponse("Feedback Submitted Successfully!")

  return redirect("feedback_form")
```

Run the server:

Run server:

Python manage.py runserver



Click on the link: http://127.0.0.1:8000/

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
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 : 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**

Setting up app level urls

Web_app/

├── migrations/

├── _init_.py

├── admin.py

├── apps.py

├── models.py

├── tests.py

├── views.py

├── urls.py (you need to create this manually)

1. Install Django (if not already installed)

Ensure you have Django installed in your environment. If not, install it using:

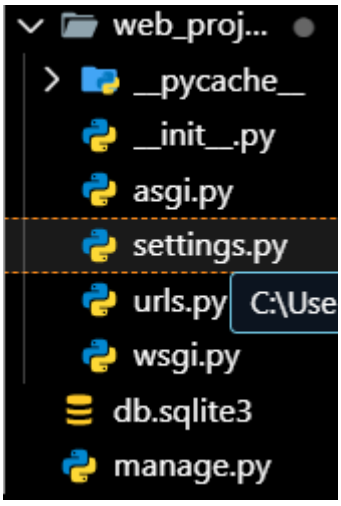
pip install django

2. Create a Django Project

A Django project is a collection of apps and configurations. Create a new project using:

django-admin startproject web_project

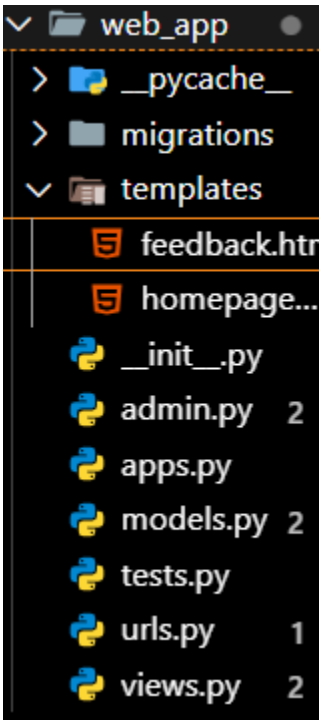
cd web_project



3. Create a Django App

A Django app is a modular component of a project. Inside the project folder, run:

python manage.py startapp web_app



web_app/urls.py

Create URLs for the App

Inside web_app/, create a new file urls.py and add:

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

urlpatterns = [
    path('', views.homepage, name='homepage'),
    path('feedback/', views.feedback_form, name='feedback_form'),
    path('submit_feedback/', views.submit_feedback, name='submit_feedback'),
]
```

Web_app/settings.py

```
    # 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'
```

Web_project/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('web_app.urls')),  # Include app-level 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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : linking views and ULR configuration
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**

Linking URLs and View Configuration in Django

In Django, linking URLs and views allows you to control how web requests are handled. The process involves:

1. Creating a View: A function or class in views.py that receives an HTTP request and returns a response. For example, a simple function-based view can return an HttpResponse with plain text or render an HTML template.
2. Defining URL Patterns: In urls.py, you define URL routes using Django's path() function, mapping a specific URL path to a view. This configuration tells Django which view to call for a given URL.
3. Including App URLs in the Project: For modularity, each Django app can have its own urls.py, which is included in the project's main urls.py using the include() function. This allows scalable and organized routing.

This setup creates a clear and maintainable structure for handling requests and serving responses in a Django web application.

```
from django.shortcuts import render, redirect

from django.http import HttpResponse

from .models import Feedback  # Import the Feedback model


# Homepage View
def homepage(request):
    return render(request, 'homepage.html')


# Feedback Form View
def feedback_form(request):
    return render(request, 'feedback.html')


# Handle Feedback Submission
def submit_feedback(request):
    if request.method == "POST":
        faculty = request.POST.get("faculty")
        rating = request.POST.get("rating")
        comment = request.POST.get("comment")
```

```
        # Save to database
        Feedback.objects.create(faculty=faculty, rating=rating, comment=comment)

        return HttpResponse("Feedback Submitted Successfully!")

    return redirect("feedback_form")
```

output:

python manage.py runserver

Click on the link: http://127.0.0.1:8000/

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : working with templates in django
7. Date of Experiment : 31-01-2025
8. Date of Submission of Report : 17-02-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY 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	

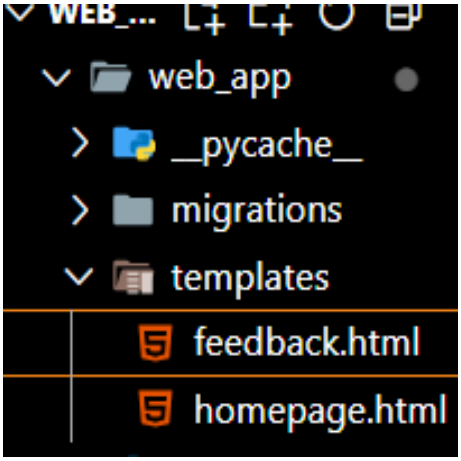
DATE: **Signature of Faculty**

TEMPLATES:

Templates in Django are HTML files that display dynamic content. They separate the frontend (UI) from the backend logic, following the MVT (Model-View-Template) architecture.

Where to Store Templates?

By default, Django looks for templates in a folder named **templates/** inside your app.



Web_app/templates/homepage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Feedback System</title>
</head>
<style>
    body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

/* Navigation Bar */
nav {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background: #333;
    padding: 15px 30px;
}

nav .logo {
    color: white;
    font-size: 22px;
   font-weight: bold; }
```

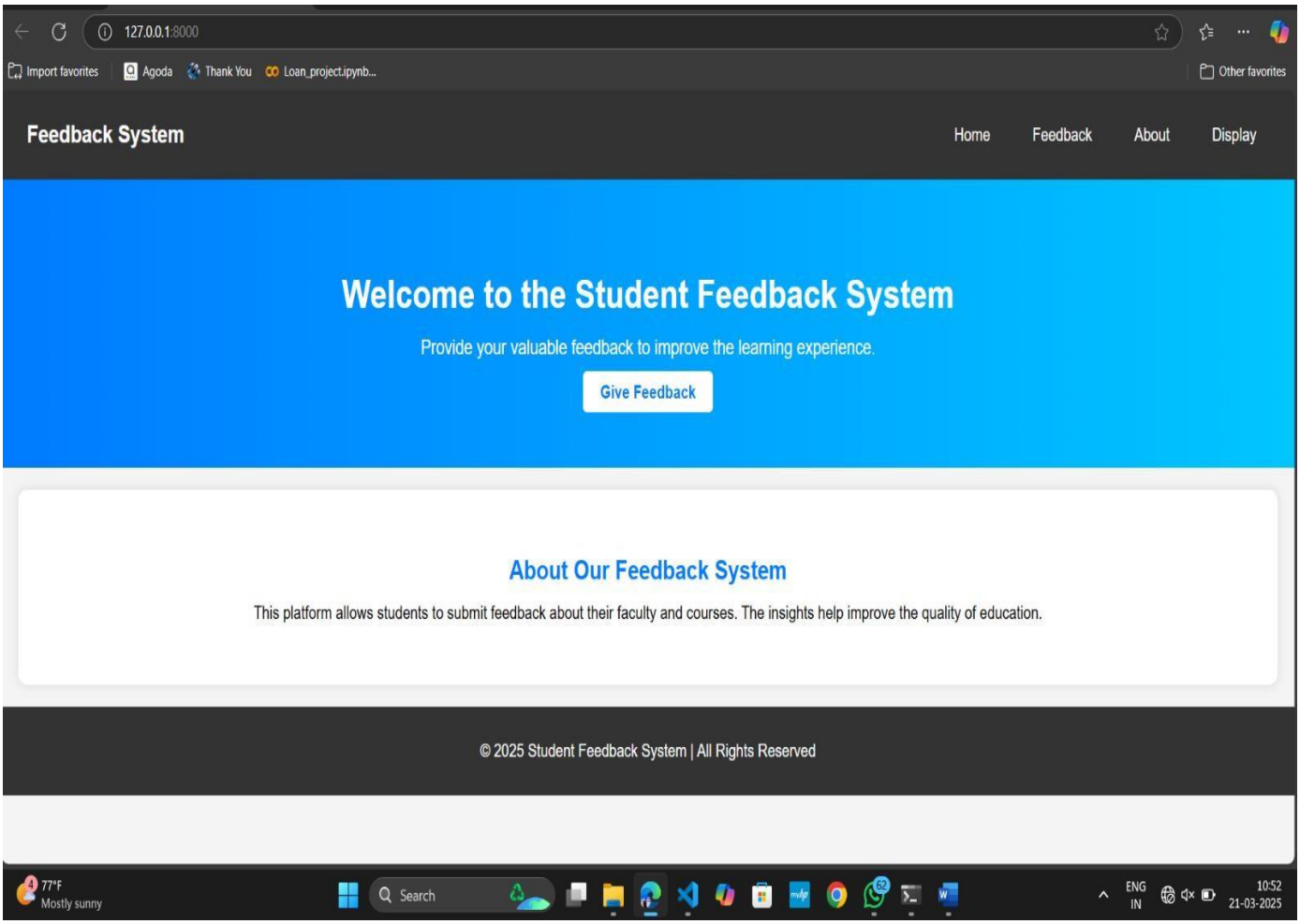
```
nav ul {
    list-style: none;
    display: flex;
    gap: 20px;
}

nav ul li {
    display: inline;
}

nav ul li a {
    color: white;
    text-decoration: none;
    font-size: 16px;
    padding: 8px 15px;
    transition: 0.3s;
}

nav ul li a:hover {
    background: #ff6b6b;
    border-radius: 5px;
}

/* Header Section */
header {
    text-align: center;
    padding: 60px;
    background: linear-gradient(to right, #007bff, #00c6ff);
    color: white;
}
```

```
header h1 {
    font-size: 36px;
    margin-bottom: 10px;
}

header p {
    font-size: 18px;
    margin-bottom: 20px;
}

header .btn {
    background: white;
    color: #007bff;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
}

header .btn:hover {
    background: #ff6b6b;
    color: white;
}

/* About Section */
.about {
    text-align: center;
```

```
    padding: 40px;
    background: white;
    margin: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

.about h2 {
    color: #007bff;
    margin-bottom: 10px;
}

/* Footer */
footer {
    text-align: center;
    padding: 15px;
    background: #333;
    color: white;
    margin-top: 20px;
}
</style>
<body>
    <nav>
        <div class="logo">Feedback System</div>
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/feedback">Feedback</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Display</a></li>
        </ul>
```

```
</nav>

  <header>
    <h1>Welcome to the Student Feedback System</h1>
    <p>Provide your valuable feedback to improve the learning experience.</p>
    <a href="/feedback" class="btn">Give Feedback</a>
  </header>

  <section class="about">
    <h2>About Our Feedback System</h2>
    <p>This platform allows students to submit feedback about their faculty and courses. The insights help improve the quality of education.</p>
  </section>

  <footer>
    <p>&copy; 2025 Student Feedback System | All Rights Reserved</p>
  </footer>
</body>
</html>
```

Output:


127.0.0.1:8000
Import favorites
Agoda
Thank You
Loan_project.ipynb...
Other favorites
Feedback System
Home
Feedback
About
Display
Welcome to the Student Feedback System
Provide your valuable feedback to improve the learning experience.
Give Feedback
About Our Feedback System
This platform allows students to submit feedback about their faculty and courses. The insights help improve the quality of education.
© 2025 Student Feedback System | All Rights Reserved
77°F
Mostly sunny
Search
ENG
IN
10:52
21-03-2025

Web_app/templates/feedback.html

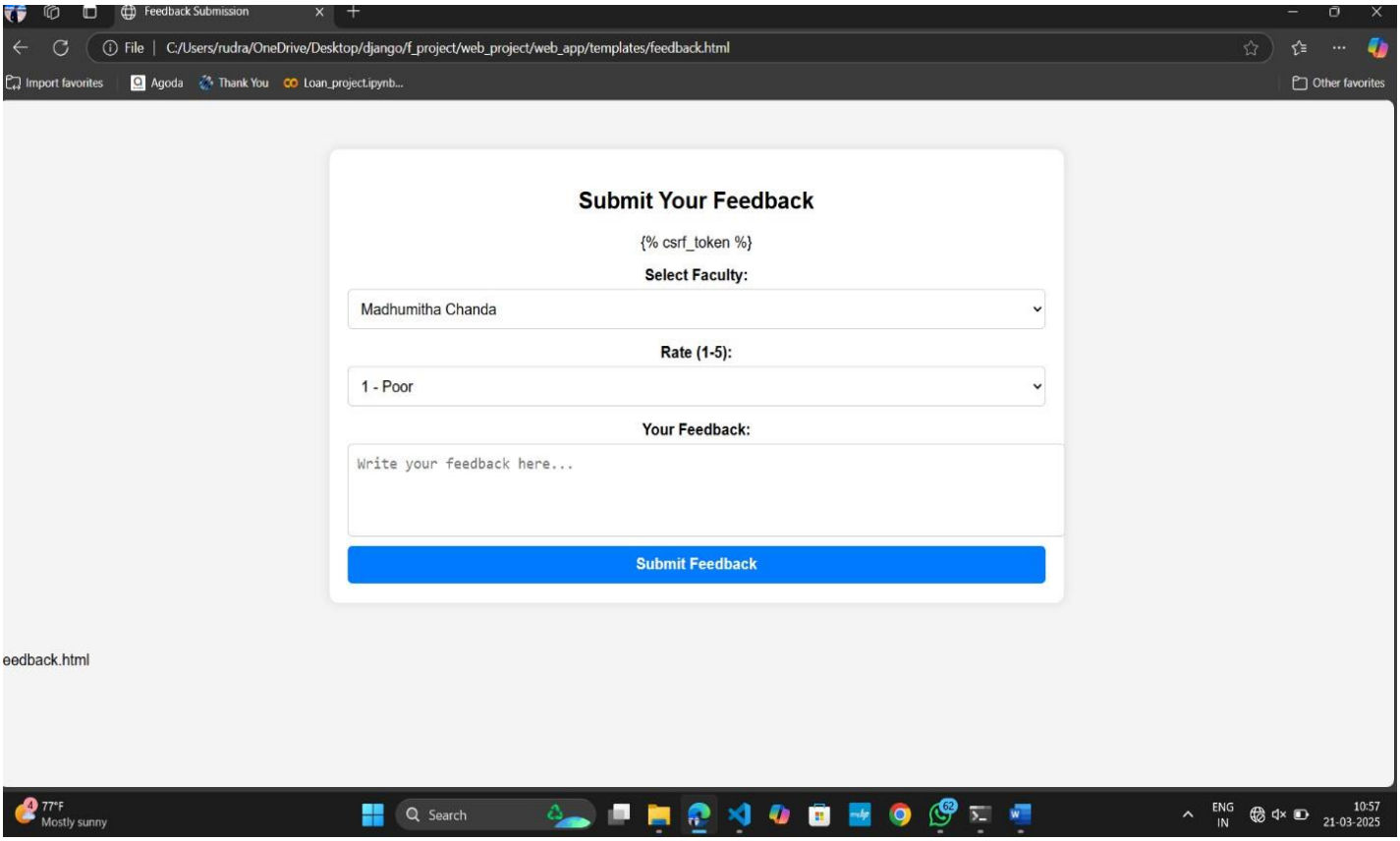
```
<!-- templates/feedback.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Feedback Submission</title>
</head>
<style>
    body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
/* Container */
.container {
    width: 50%;
    margin: 50px auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
/* Form Elements */
label {
    display: block;
    margin-top: 15px;
```

```
    font-weight: bold;
}

select, textarea, button {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}

textarea {
    resize: none;
}

/* Submit Button */
button {
    background: #007bff;
    color: white;
    cursor: pointer;
    font-weight: bold;
    border: none;
    transition: 0.3s;
}
button:hover {
    background: #0056b3;
}
```

```
</style>
<body>
  <div class="container">
    <h2>Submit Your Feedback</h2>
    <form method="POST" action="/submit_feedback/">
      {% csrf_token %}
      <!-- Faculty Selection -->
      <label for="faculty">Select Faculty:</label>
      <select id="faculty" name="faculty">
        <option value="Madhumitha Chanda">Madhumitha Chanda</option>
        <option value="Roje Spandana">Roje Spandana</option>
        <option value="Venkateswarulu">Venkateswarulu</option>
        <option value="Paparao">Paparao</option>
        <option value="Bindumadhuri">Bindumadhuri</option>
      </select>

      <!-- Rating Selection -->
      <label for="rating">Rate (1-5):</label>
      <select id="rating" name="rating">
        <option value="1">1 - Poor</option>
        <option value="2">2 - Fair</option>
        <option value="3">3 - Good</option>
        <option value="4">4 - Very Good</option>
        <option value="5">5 - Excellent</option>
      </select>

      <!-- Comment Section -->
      <label for="comment">Your Feedback:</label>
      <textarea id="comment" name="comment" rows="4" placeholder="Write your feedback
```

```
here..."></textarea>

        <!-- Submit Button -->
        <button type="submit">Submit Feedback</button>
      </form>
    </div>
 </body>
</html>
```

Output:



Templates/display.html:

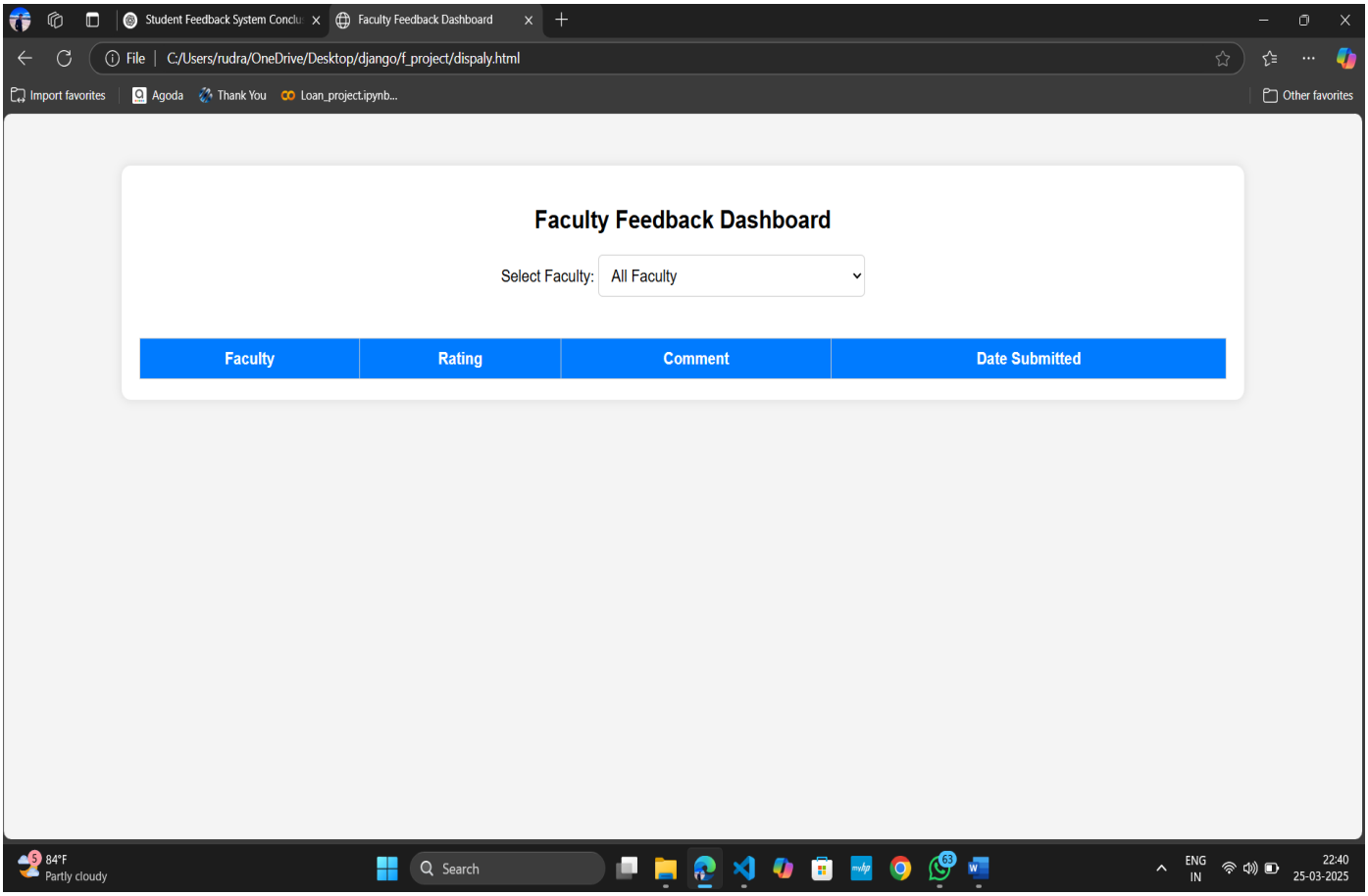
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Faculty Feedback Dashboard</title>
    <link rel="stylesheet" href="display.css">
</head>
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
/* Container */
.container     {
    width: 80%;
    margin: 50px auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
/* Faculty Selection */
select {
    padding: 10px;
    font-size: 16px;
    margin-bottom: 20px;
```

```
    width: 100%;
    max-width: 300px;
    border-radius: 5px;
    border: 1px solid #ccc;
}

/* Table Styling */
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}
table, th, td {
    border: 1px solid #ccc;
}
th, td {
    padding: 10px;
    text-align: center;
}
th {
    background-color: #007bff;
    color: white;
}

/* Hover effect */
tbody tr:hover {
    background-color: #f1f1f1;
}
```

```
<body>
  <div class="container">
    <h2>Faculty Feedback Dashboard</h2>

    <!-- Faculty Selection -->
    <label for="faculty">Select Faculty:</label>
    <select id="faculty" onchange="loadFeedback()">
      <option value="all">All Faculty</option>
      <option value=" Mrs.Madhumitha Chanda">Madhumitha Chanda</option>
      <option value="Roje Spandana">Roje Spandana</option>
      <option value="Venkateswarulu">Venkateswarulu</option>
      <option value="Paparao">Paparao</option>
      <option value="Bindumadhuri">Bindumadhuri</option>
    </select>
    <table>
      <thead>
        <tr>
          <th>Faculty</th>
          <th>Rating</th>
          <th>Comment</th>
          <th>Date Submitted</th>
        </tr>
      </thead>
      <tbody id="feedbackTable">
        <!-- Feedback Data will be loaded here -->
      </tbody>
    </table>
  </div>
```

```
<script>
    function loadFeedback() {
        let selectedFaculty = document.getElementById("faculty").value;
        fetch('fetch_faculty_feedback.php?faculty=' + selectedFaculty)
            .then(response => response.text())
            .then(data => {
                document.getElementById("feedbackTable").innerHTML = data;
            })
            .catch(error => console.error('Error:', error));
    }

    window.onload = loadFeedback;
</script>
</body>
</html>
```

Output:


Student Feedback System Conclu
Faculty Feedback Dashboard
File | C:/Users/rudra/OneDrive/Desktop/django/f_project/dispaly.html
Import favorites
Agoda
Thank You
Loan_project.ipynb...
Other favorites
Faculty Feedback Dashboard
Select Faculty:
All Faculty
Faculty
Rating
Comment
Date Submitted
84°F
Partly cloudy
Search
ENG
IN
22:40
25-03-2025

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : database integration and configuration-MYSQL
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-MYSQL

Steps to connect MySQL to Django

Step 1: Create a new project and start the process to connect django with mysql.

django-admin startproject MyDB

Step 2: Move to the MyDB folder

cd MyDB

Step 3: Once you're logged into MySQL, you can create a new database. Replace your_database_name with the desired name for your database.

CREATE DATABASE your_database_name

Step 4: Update the settings.py

Open settings.py here inside the DATABASES variable configure MySQL database values, and add values of your database.

First, we have replaced the 'django.db.backends.sqlite3' to 'django.db.backends.mysql'. This is basically indicating we shift SQLite to MySQL database.

1. NAME: It indicates the name of the database we want to connect.
2. USER: The MYSQL username is the one who has access to the database and manages it.
3. PASSWORD: It is the password of the database.
4. HOST: It is indicated by "127.0.0.1" and "PORT" "3306" that the MySQL database is accessible at hostname "0.0.1" and on port "3306."

Step 5: Run the server.

python manage.py runserver

Step 6: Run the migration command

python manage.py makemigrations

python manage.py migrate

```
C:\WINDOWS\system32\cmd.
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rudra>mysql
ERROR 1045 (28000): Access denied for user 'ODBC'@'localhost' (using password: NO)

C:\Users\rudra>mysql -u root -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 44
Server version: 8.0.41 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| feedback           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.03 sec)

mysql>
```

```
mysql> use feedback;
Database changed
mysql> show tables
    -> \c
mysql> show tables;
+----------------------------+
| Tables_in_feedback         |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| web_app_feedback           |
| web_app_product            |
+----------------------------+
12 rows in set (0.01 sec)

mysql>
```

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
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**

Forms:

When one creates a Form class, the most important part is defining the fields of the form. Each field has custom validation logic, along with a few other hooks. This article revolves around various fields one can use in a form along with various features and techniques concerned with Django Forms.

Django Forms

Forms are used for taking input from the user in some manner and using that information for logical operations on databases. For example, Registering a user by taking input such as his name, email, password, etc.

Django maps the fields defined in Django forms into HTML input fields. Django handles three distinct parts of the work involved in forms:

i. Preparing and restructuring data to make it ready for rendering.

ii. Creating HTML forms for the data.

iii. Receiving and processing submitted forms and data from the client.

```
from django import forms


# creating a form
class GeeksForm(forms.Form):
    title = forms.CharField()
    description = forms.CharField()
```

Render Django Forms

Django form fields have several built-in methods to ease the work of the developer but sometimes one needs to implement things manually for customizing User Interface(UI). A form comes with 3 in-built methods that can be used to render Django form fields.

i. {{ form.as_table }} will render them as table cells wrapped in <tr> tags

ii. {{ form.as_p }} will render them wrapped in <p> tags

iii. {{ form.as_ul }} will render them wrapped in <li> tags

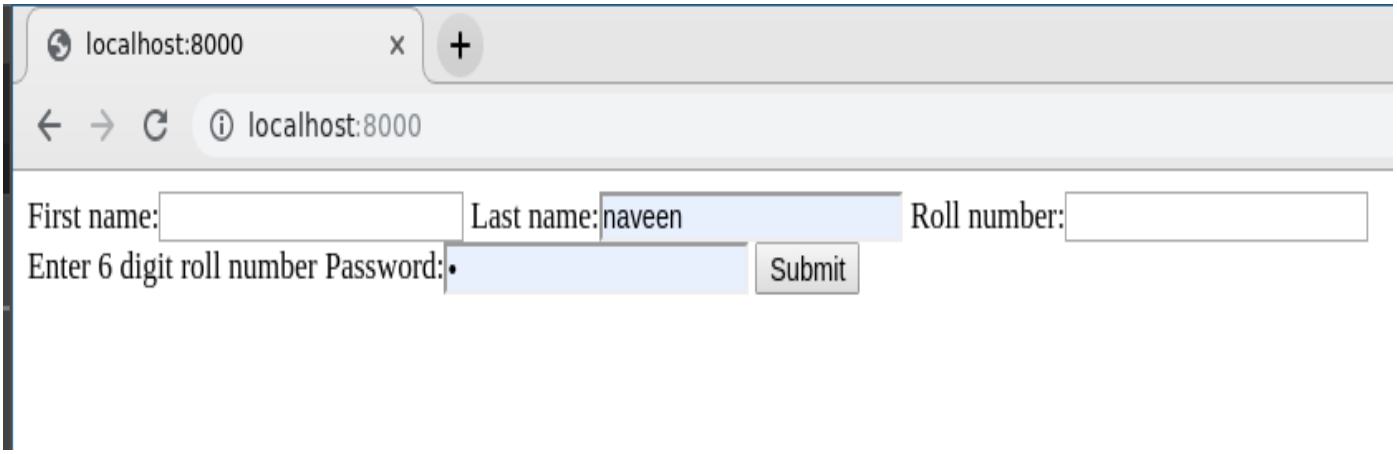
To render this form into a view, move to views.py and create a home_view as below.

```
from django.shortcuts import render
from .forms import InputForm


# Create your views here.
def home_view(request):
    context ={}
    context['form']= InputForm()
    return render(request, "home.html", context)
```

In view, one needs to just create an instance of the form class created above in forms.py. Now let's edit templates > home.html

```
<form action = "" method = "post">
    {% csrf_token %}
    {{form }}
    <input type="submit" value=Submit">
</form>
```

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

1. Name of the Laboratory	:	Django Frame work Lab
2. Name of the Student	:	Yamini Desula
3. Roll No	:	23VV1A1213
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**

What is models.py in Django:

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

Why Use Django Models:

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

How to Apply Models:

Create the Model in models.py:

Write your models inside the models.py file.

MODELS.PY:

```
from django.db import models


class Feedback(models.Model):
    faculty = models.CharField(max_length=100) rating
    = models.IntegerField()
    comment = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)


    def __str__(self):
        return f"{self.faculty} - {self.rating}" from
django.db import models


class Product(models.Model):
    name = models.CharField(max_length=255)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    created_at = models.DateTimeField(auto_now_add=True)
```

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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : migrations: synchronizing models with database
7. Date of Experiment : 07-03-2025
8. Date of Submission of Report : 27-03-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY 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**

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

admin.py

```
from django.contrib import admin
from .models import Feedback


admin.site.register(Feedback)
```

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'web_project.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__== '__main__':
    main()
```

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 Frame work Lab
2. Name of the Student	:	Yamini Desula
3. Roll No	:	23VV1A1213
4. Class	:	II B.Tech II semester
5. Academic Year	:	2024-25
6. Name of Experiment	:	deploying Django application on cloud platforms
7. Date of Experiment	:	27-03-2025
8. Date of Submission of Report	:	04-04-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY 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/Tejaswiniy19/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/yaminidesula/student_feedback_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 Frame work Lab
2. Name of the Student : Yamini Desula
3. Roll No : 23VV1A1213
4. Class : II B.Tech II semester
5. Academic Year : 2024-25
6. Name of Experiment : frontend developer 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 DEVELOPER CERTIFICATE: