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

DJANGO FRAMEWORK

YERRA TEJASWINI

(Regd. No. 23VV1A1264)

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

CERTIFICATE

Regd.No : 23VV1A1264

This is to certify that this is a bonafide record of practical work done by YERRA. TEJASWINI of IInd B.Tech IInd Semester Class in DJANGO FRAMEWORK Lab during the year 2024-25.

No.of Tasks Completed and Certified: 13

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

Date:

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Website:www.jntugvcev.edu.in

Subject Name: DJANGO FRAMEWORK Subject Code:R232212SE01

Year: 2025 Regulation: R23

COURSE OUTCOMES

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

CO-PO Mapping

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

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

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

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

Signature of the Course Instructor

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

PYTHON LIBRARIES

1. Python Collections - Container Datatypes:

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

Key Types:

a. **List**: Ordered, mutable, allows duplicates.
b. **Tuple**: Ordered, immutable, allows duplicates.
c. **Set**: Unordered, no duplicates, fast membership testing.
d. **Dictionary**: Unordered, key-value pairs, fast lookups.

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

2. Tkinter - GUI Applications:

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

Key Features:

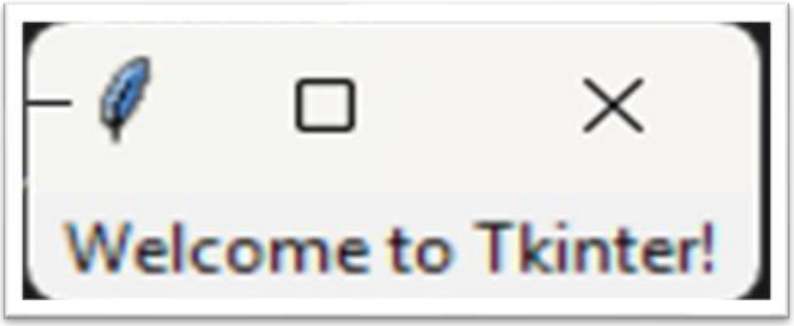
a. Widgets: Buttons, labels, text boxes, etc.
b. Event handling: Respond to user interactions like clicks or key presses.
c. Simple layout management.

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

CODE:

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

Output:



3. Requests - HTTP Requests:

Purpose: Simplifies HTTP requests to interact with web APIs.

Key Features:

a. Send GET, POST, PUT, DELETE requests easily.
b. Handle request parameters, headers, and cookies.
c. Simple error handling and response handling.

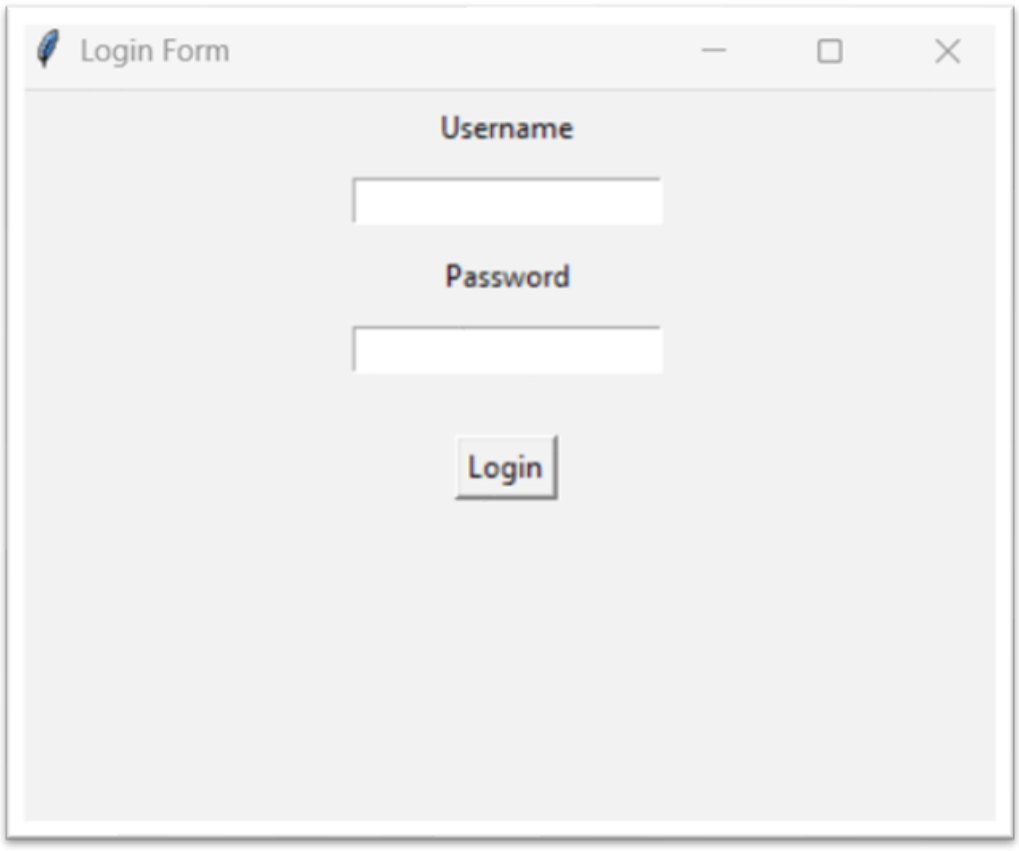
Common Use: Interact with REST APIs, download content from the web.

CODE:

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

```
root = tk.Tk()root.title("Login Form")
root.geometry("400x300")  # Adjusted to a more reasonable size
# Create username and password labels and entry widgets
username_label = tk.Label(root, text="Username")
username_label.pack(pady=5)
username_entry = tk.Entry(root)
username_entry.pack(pady=5)
password_label = tk.Label(root, text="Password")
password_label.pack(pady=5)
password_entry = tk.Entry(root, show="*")  # 'show' hides the password characters
password_entry.pack(pady=5)
# Create the login button
```

Output:



4. Scrapy:

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

Key Features:

a. Fast, extensible, and asynchronous web scraping.
b. Supports handling requests, data extraction, and storing results.
c. Built-in handling for logging, retries, and sessions.

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

5. BeautifulSoup4 - Web Scraping:

Purpose: Parses HTML and XML documents to extract data.

Key Features:

a. Easy navigation and searching within HTML.
b. Supports different parsers like html.parser, lxml, and html5lib.

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

CODE:

```
import requests
from bs4 import BeautifulSoup
# The URL of the website to scrape
url = 'https://example.com'  # Replace with the actual website URL
# Set user-agent headers to avoid getting blocked
headers = {
   'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36'}
try:
   # Send a GET request
   response = requests.get(url, headers=headers, timeout=10)
   response.raise_for_status()  # Raise an error for HTTP errors (e.g., 404, 500)
   # Parse the HTML content with BeautifulSoup
   soup = BeautifulSoup(response.text, 'html.parser')
```

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

Output:

6. Zappa:

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

Key Features:

a. Supports frameworks like Flask and Django for serverless deployments.
b. Manages serverless architecture and deployment configurations.

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

7. Dash:

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

Key Features:

a. Built on top of Flask, React, and Plotly.
b. Integrates seamlessly with data science libraries (e.g., Pandas, Plotly).

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

8. TurboGears:

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

Key Features:

a. Modular: Mix and match components like SQLAlchemy, Genshi, and others.
b. Focus on rapid development and scalability.

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

9. CherryPy:

Purpose: Minimalistic web framework for building web applications.

Key Features:

Provides a simple and fast HTTP server.

Handles routing, cookies, sessions, and file uploads.

Common Use: Building web applications with a lightweight framework.

CODE:

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

Output:

After run the server :-

10. Flask:

Purpose: Lightweight micro-framework for building web applications.

Key Features:

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

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

CODE:

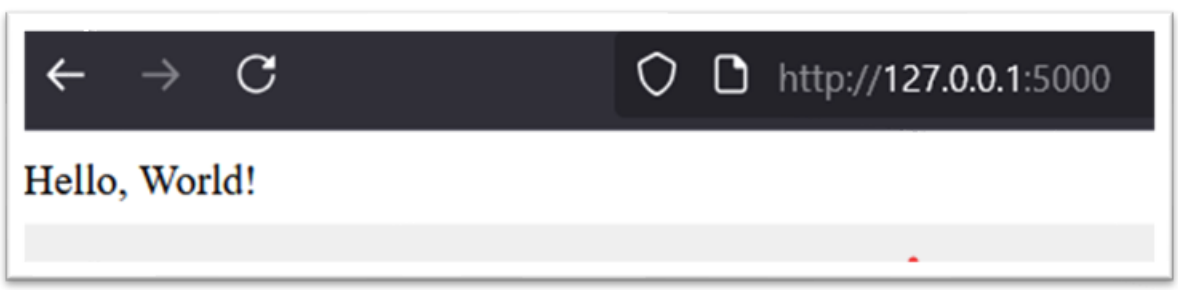
```
From flask import Flask
app = Flask(__name__)
@app.route('/')
def hello(): return "Hello, World!'
if __name__ =='__main__':
    app.run(debug=True)
```

Output:

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

C:\Users\yerra\OneDrive\Desktop\python\Basic\New folder\Books>
```

After run the server:



11. Web2Py:

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

Key Features:

a. Includes a web-based IDE for development.
b. Built-in ticketing system and database integration.

Common Use: Enterprise web applications with minimal setup.

12. Bottle:

Purpose: Simple and lightweight WSGI micro-framework.

Key Features:

a. Single-file framework, minimalistic, and fast.
b. No dependencies, supports routing, templates, and form handling.

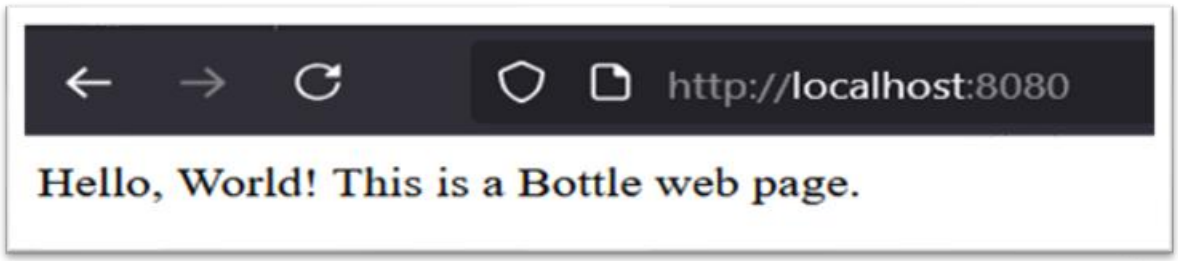
Common Use: Small web applications, APIs, and prototypes.

CODE:

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

Output:

After run the server:



13. Falcon:

Purpose: High-performance framework for building APIs.

Key Features:

a. Focuses on speed and minimalism.
b. Supports RESTful API development and is optimized for large-scale deployments.

Common Use: Building fast, high-performance APIs.

14. CubicWeb:

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

Key Features:

a. Uses a highly modular architecture for development.
b. Focus on building web apps with rich data models.

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

15. Quixote:

Purpose: A web framework designed for simplicity and scalability.

Key Features:

a. Full support for Python's object-oriented programming.
b. Easily extensible, with minimalistic core.

Common Use: Scalable and customizable web applications.

16. Pyramid:

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

Key Features:

1. Highly flexible with support for routing, templating, authentication, and authorization.
2. Allows for small and large applications, with fine-grained control.

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

SUMMARY:

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

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

Django: A Web Framework For Python

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

key features:

1. **MVT Architecture:**

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

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

   View: Manages business logic and processes user requests.

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

2. **Built-in Admin Interface**

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

3. **ORM (Object-Relational Mapping)**

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

Security Features :

Django comes with built-in security features, including:

1. CSRF Protection (Cross-Site Request Forgery)
2. XSS Protection (Cross-Site Scripting)
3. SQL Injection Prevention
4. Secure Authentication System
5. Clickjacking Protection

4. **Scalability & Performance :**

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

5. **URL Routing:**

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

6. **Templating Engine**:

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

7. **Form Handling:**

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

Why Use Django?

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

Conclusion :

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

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

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

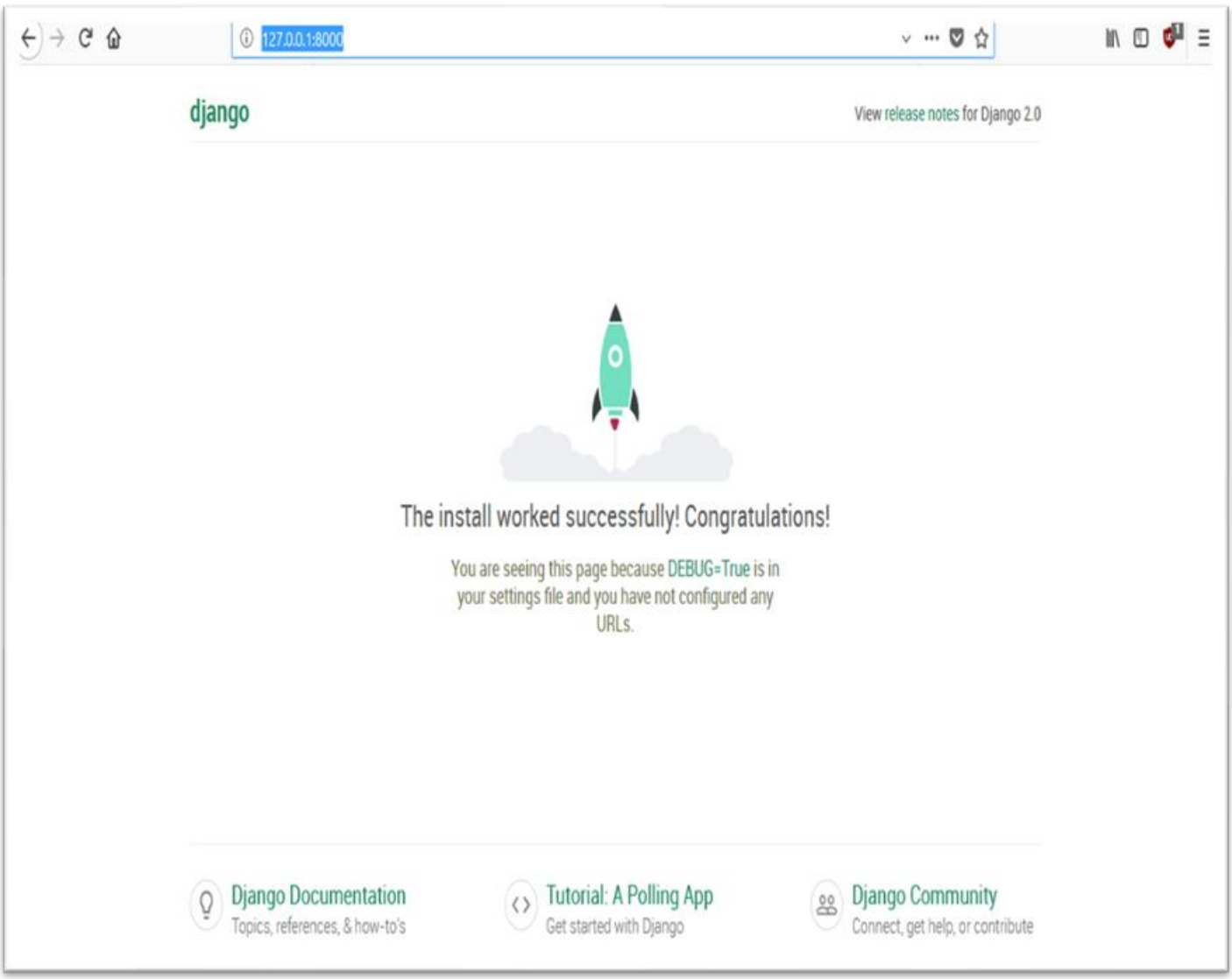
```
django-admin startproject project_name
django-admin startapp app_name
```

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

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

LINKING VIEW's AND URL's:

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

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

Exploring Django Views

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.contrib import messages
from django.http import HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from .models import Assignment, Submission, Feedback
from .forms import AssignmentForm, SubmissionForm, FeedbackForm
import os
from django.core.mail import send_mail
from django.conf import settings
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate, logout
from django.contrib import messages
from .forms import RegisterForm, LoginForm
from django.shortcuts import render
from django.urls import reverse
import time
import cv2
import face_recognition
import numpy as nps
from django.conf import settings
from .forms import EmailForm
# Register View
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate, logout
from django.contrib import messages
```

```
# Register View
def register_view(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)  # Log the user in after registration
            messages.success(request, 'Registration successful!')
            return redirect('login')  # Redirect to home page after registration
        else:
            messages.error(request, 'Registration failed. Please correct the errors below.')
    else:
        form = RegisterForm()
    return render(request, 'register.html', {'form': form})
# Login View
def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)  # Log the user in
                messages.success(request, f'Welcome back, {username}!')
                return redirect('home')  # Redirect to home page after login
            else:
                messages.error(request, 'Invalid username or password.')
    else:
        form = LoginForm()
```

```
    return render(request, 'login.html', {'form': form})
# Logout View
def logout_view(request):
    logout(request)
    messages.success(request, 'You have been logged out.')
    return redirect('home')  # Redirect to home page after logout
# Student view: Submit assignment
def submit_assignment(request, assignment_id):
    assignment = get_object_or_404(Assignment, id=assignment_id)
    if request.method == 'POST':
        form = SubmissionForm(request.POST, request.FILES)
        if form.is_valid():
            submission = form.save(commit=False)
            submission.assignment = assignment  # Link submission to the assignment
            submission.save()
            return redirect('home')
    else:
        form = SubmissionForm()
    messages.success(request, 'Your assignment has been submitted successfully!')
    return render(request, 'submit_assignment.html', {'form': form, 'assignment':
assignment})
# Teacher view: View submissions for a specific assignment
def view_submissions(request, assignment_id):
    assignment = get_object_or_404(Assignment, id=assignment_id)
    submissions = Submission.objects.filter(assignment=assignment)
    return render(request, 'view_submissions.html', {'submissions': submissions,
'assignment': assignment})
def viewst_submissi(request, assignment_id):
    assignment = get_object_or_404(Assignment, id=assignment_id)
    submissions = Submission.objects.filter(assignment=assignment)
```

```
  if student_id:
      submissions = submissions.filter(student_id__icontains=student_id)
    context = {
      'assignment': assignment,
      'submissions': submissions, }
    return render(request, 'viewst_submissi.html', context)
# Teacher view: Leave feedback for a submission
def leave_feedback(request, submission_id):
    submission = Submission.objects.get(id=submission_id)
    if request.method == 'POST':
      form = FeedbackForm(request.POST)
      if form.is_valid():
        feedback = form.save(commit=False)
        feedback.submission = submission
        feedback.save()
        return redirect('staff')  # Redirect to a success page
    else:
      form = FeedbackForm()
    return render(request, 'leave_feedback.html', {'submission': submission, 'form': form})
# Admin view: Create assignments
def create_assignment(request):
    if request.method == 'POST':
      form = AssignmentForm(request.POST)
      if form.is_valid():
        form.save()
        return redirect('assignment_list')
    else:
      form = AssignmentForm()
    return render(request, 'create_assignment.html', {'form': form})
```

```
# List all assignments
def assignment_list(request):
    assignments = Assignment.objects.all()
    return render(request, 'assignment_list.html', {'assignments': assignments})
def assignmentst_list(request):
    assignments = Assignment.objects.all()
    return render(request, 'assignmentst_list.html', {'assignments': assignments})
def view_feedback(request, submission_id):
    # Fetch the submission or return a 404 error if not found
    submission = get_object_or_404(Submission, id=submission_id)
    # Fetch the first feedback associated with the submission (if any)
    feedback = Feedback.objects.filter(submission=submission).first()
    # Render the template with the submission and feedback
    return render(request, 'view_feedback.html', {
        'submission': submission,
        'feedback': feedback
    })
def delete_submission(request, submission_id):
    submission = get_object_or_404(Submission, id=submission_id)
    # Delete the file from the filesystem
    if submission.file:
        file_path = os.path.join(settings.MEDIA_ROOT, submission.file.name)
        if os.path.exists(file_path):
            os.remove(file_path)
    # Delete the submission from the database
    submission.delete()
    messages.success(request, 'Submission deleted successfully.')
    return redirect('assignment_list')
def delete_assignment(request, assignment_id):
    assignment = get_object_or_404(Assignment, id=assignment_id)
```

```
    # Delete associated submissions and their files
    for submission in assignment.submission_set.all():
        if submission.file:
            file_path = os.path.join(settings.MEDIA_ROOT, submission.file.name)
            if os.path.exists(file_path):
                os.remove(file_path)
    # Delete the assignment
    assignment.delete()
    # Add a success message
    messages.success(request, 'Assignment deleted successfully.')
    return redirect('assignment_list')
def staff(request):
    assignments = Assignment.objects.all()  # Fetch all assignments
    total_assignments = Assignment.objects.count()  # Total assignments
    total_submissions = Submission.objects.count()  # Total submissions
    completed_submissions = Submission.objects.filter().count()  # Completed submissions
    pending_feedback = Feedback.objects.filter().count()  # Pending feedback
    # Combine all data into a single context dictionary
    context = {
        'assignments': assignments,
          'assignment': assignments.first(),  # Add assignments to the context
        'total_assignments': total_assignments,
        'total_submissions': total_submissions,
        'completed_submissions': completed_submissions,
        'pending_feedback': pending_feedback,
    }
    return render(request, 'staff.html', context)
def assignment_detail(request, assignment_id):
 assignments = Assignment.objects.all()  # Fetch all assignments
     # Debugging: Check if assignments are fetched
```

```
 context = {'assignments': assignments,  # Pass assignments to the template}
 return render(request, 'assignment_detail.html', context)
def home(request):
    assignments = Assignment.objects.all()
    return render(request, 'home.html', {'assignments': assignments})
def assignment_de(request, assignment_id):
    # Fetch the assignment or return a 404 error if not found
    assignment = get_object_or_404(Assignment, id=assignment_id)
    return render(request, 'assignment_de.html', {'assignment': assignment})
def send_email(request):
    if request.method == 'POST':
        form = EmailForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            email = form.cleaned_data['email']
            subject = form.cleaned_data['subject']
            message = form.cleaned_data['message']
            from_email = 'teja123@gmail.com'
            recipient_email = 'teja123@gmail.com'
            # Create the email message
            email_message = f"Feedback from {name} ({email}):\n\n{message}"
            send_mail(subject, email_message, from_email, [recipient_email])
            return HttpResponse('Feedback sent successfully!', status=200)
    else:
        form = EmailForm()
    return render(request, 'send_email.html', {'form': form})
KNOWN_IMAGE_PATH = os.path.join(settings.BASE_DIR, 'kelly', 'teju.jpg')
known_image = face_recognition.load_image_file(KNOWN_IMAGE_PATH)
known_face_encoding = face_recognition.face_encodings(known_image)[0]
known_face_encodings = [known_face_encoding]
```

```
known_face_names = ["Authorized Person"]
def capture_frame(timeout=10):
    Captures frames from the webcam for a specified duration.
    Returns the last captured frame or None if no frames are captured.
    video_capture = cv2.VideoCapture(0,cv2.CAP_DSHOW)
    if not video_capture.isOpened():
        print("Error: Could not open webcam.")
        return None
    start_time = time.time()
    frames = []
    while time.time() - start_time < timeout:
        ret, frame = video_capture.read()
        if ret:
            frames.append(frame)
        time.sleep(0.1)
    video_capture.release()
    if not frames:
        print("Error: No frames captured.")
        return None
    return frames[-1]
def detect_motion(prev_frame, curr_frame, threshold=5000):
    Detects motion between two frames.
    Returns True if motion is detected, otherwise False.
    diff = cv2.absdiff(prev_frame, curr_frame)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
```

```
        if cv2.contourArea(contour) > threshold:
            return True return False
def recognize_face(live_feed_duration=10, motion_threshold=5000):
    Recognizes faces in the live feed and checks if it matches the known face.
    Only grants access if a live face with motion is detected.
    video_capture = cv2.VideoCapture(0,cv2.CAP_DSHOW)
    start_time = time.time()
    prev_frame = None
    while time.time() - start_time < live_feed_duration:
        ret, frame = video_capture.read()
        if not ret:
            continue
        small_frame = cv2.resize(frame, (0, 0), fx=0.50, fy=0.50)
        rgb_small_frame = small_frame[:, :, ::-1]
        if prev_frame is not None and detect_motion(prev_frame, small_frame,
motion_threshold):
            face_locations = face_recognition.face_locations(rgb_small_frame)
            face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
            for face_encoding in face_encodings:
                matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
                if True in matches:
                    video_capture.release()
                    return True
        prev_frame = small_frame
    video_capture.release()
    return False
def index(request):
    Renders the home page.
    return render(request, 'index.html')
```

```
def check_face(request):
    Checks if the captured face matches the known face.
    Redirects to the success or failure page based on the result.
    print("Checking face...")
    try:
        # Perform face recognition
        if recognize_face():
            print("Face recognized. Redirecting to success page.")
            return redirect(reverse('staff'))
        else:
            print("Face not recognized. Redirecting to failure page.")
            return redirect(reverse('failure'))
    except Exception as e:
        print(f"Error during face recognition: {e}")
        return redirect(reverse('failure'))
def failure(request):
    Renders the failure page.
    return render(request, 'failure.html')
def about(request):
    Renders the failure page.
    return render(request, 'about.html')
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

Setting up App-Level URLs

1.App's.py

CODE:

```
from django.apps import AppConfig
class KellyConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'kelly'
```

2. Url's.py

CODE:

```
from django.contrib import admin
from django.urls import path
from django.urls import path, include
from kelly import views
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', views.home, name='home'),
    path('staff/', views.staff, name='staff'),
    path('', views.register_view, name='register'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),
```

```
path('assignment_list', views.assignment_list, name='assignment_list'),

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

    path('assignment/create/', views.create_assignment, name='create_assignment'),

    path('assignment/<int:assignment_id>/', views.assignment_detail,
name='assignment_detail'),

    path('assignment/<int:assignment_id>/de', views.assignment_de,
name='assignment_de'),

    path('assignment/<int:assignment_id>/submit/', views.submit_assignment,
name='submit_assignment'),

    path('assignment/<int:assignment_id>/submissions/', views.view_submissions,
name='view_submissions'),

    path('assignment/<int:assignment_id>/submissi/', views.viewst_submissi,
name='viewst_submissi'),

    path('submission/<int:submission_id>/feedback/', views.leave_feedback,
name='leave_feedback'),

    path('submission/<int:submission_id>/view_feedback/', views.view_feedback,
name='view_feedback'),

    path('submission/<int:submission_id>/delete/', views.delete_submission,
name='delete_submission'),

    path('assignment/<int:assignment_id>/delete/', views.delete_assignment,
name='delete_assignment'),

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

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

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

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

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

]

if settings.DEBUG:

    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

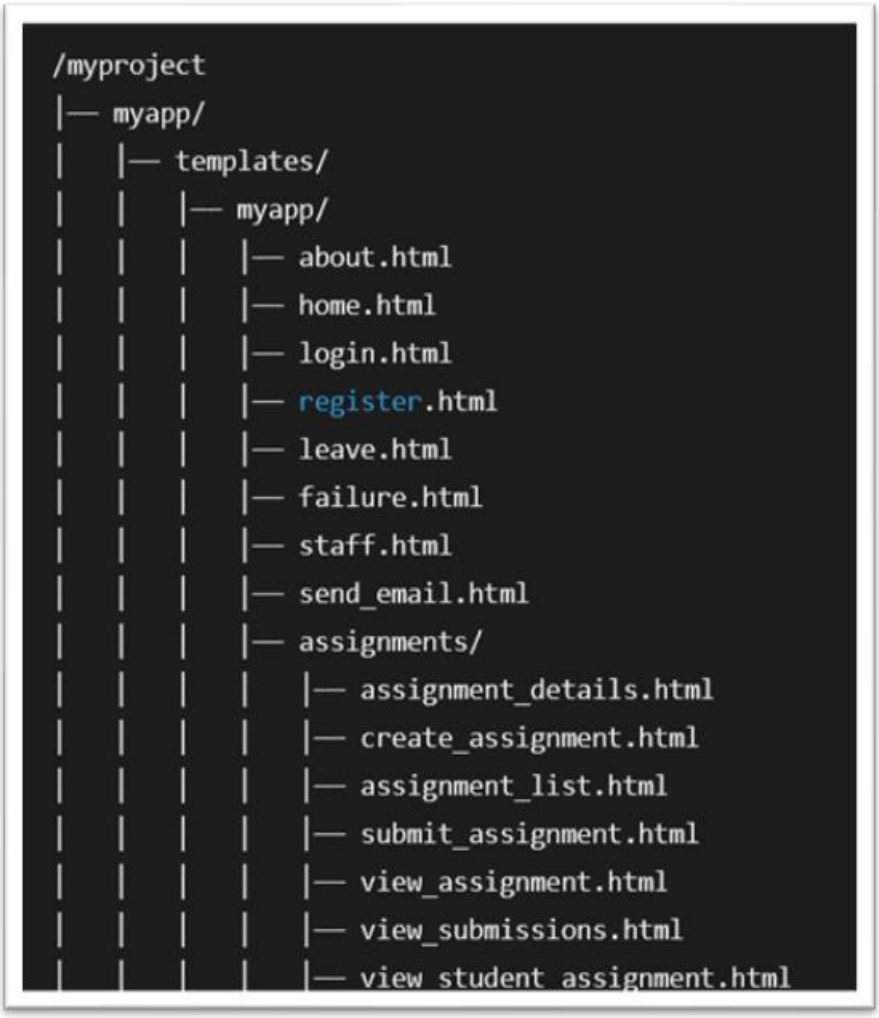
Signature of Faculty:

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.



Register.html

CODE:

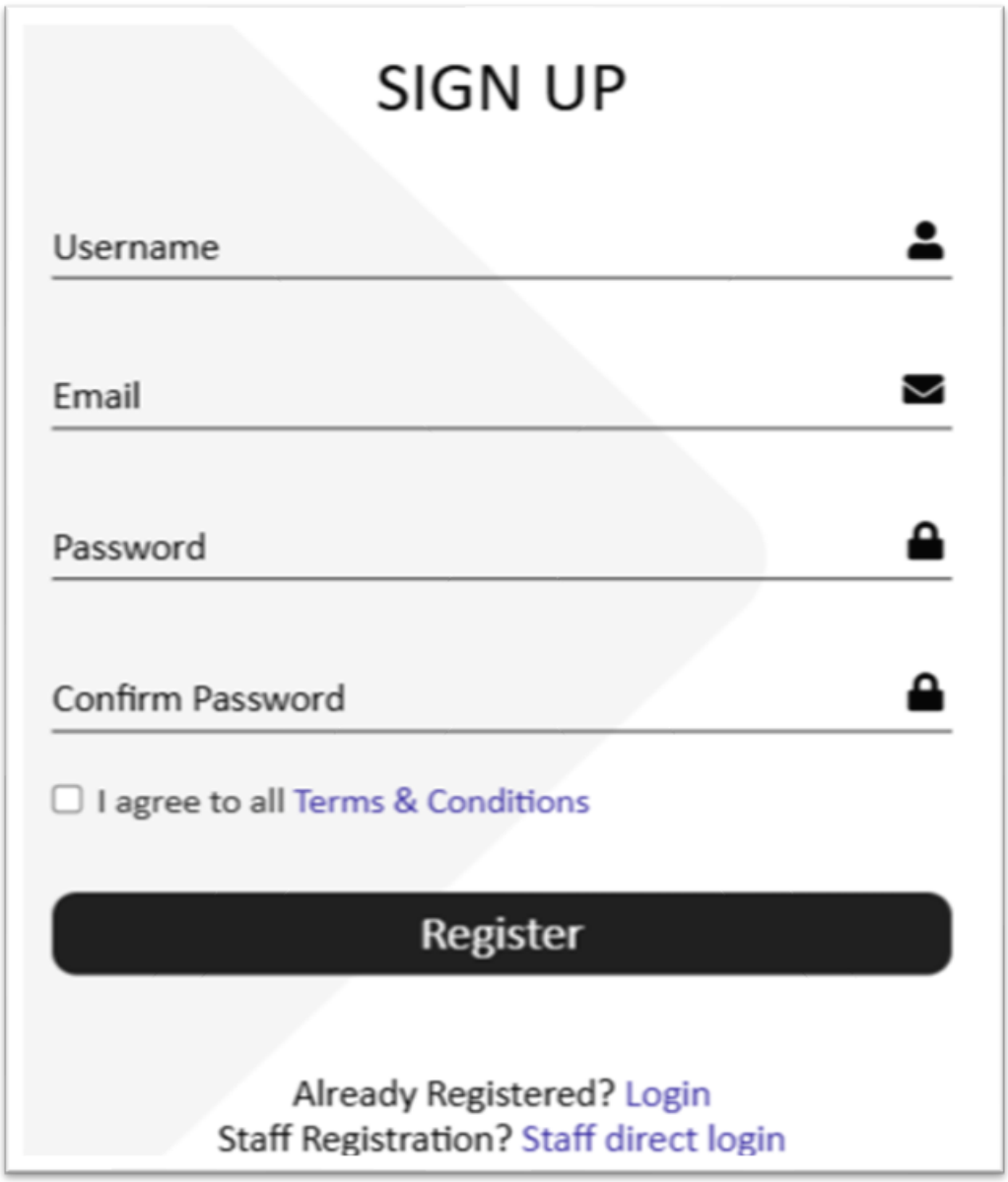
```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Register</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'style.css' %}">
</head><body><div id="main-container">
      <div class="form-container">
         <div class="signup-form">
            <div class="title">SIGN UP</div>
            <form method="post" action="{% url 'register' %}">
               {% csrf_token %}<div class="field">
      <input type="text" name="username" id="username" placeholder=" " required>
                  <label for="username">Username</label>
                  <span class="fa fa-user"></span>
               </div><div class="field">
                  <input type="email" name="email" id="email" placeholder=" "
required>
                  <label for="email">Email</label>
                  <span class="fa fa-envelope"></span>
               </div><div class="field">
     <input type="password" name="password1" id="password1" placeholder=" "
required>   <label for="password1">Password</label>
                  <label for="password2">Confirm Password</label>
                  <span class="fa fa-lock"></span></div><section>
```

```
                <span class="fa fa-lock"></span></div><div class="field">
    <input type="password" name="password2" id="password2" placeholder=" "
required>
                <label for="agree"><input type="checkbox" id="agree">I agree to
all <a href="#">Terms & Conditions</a></label>
                </section><button type="submit" class="signup-btn">Register</button>
            </form> <div class="bottom">
                <span>Already Registered? <a href="{% url 'login' %}"
class="login-switch" onclick="swapPos(this)">Login</a></span>
                <span>Staff Registration? <a href="{% url 'index' %}" class="login-
switch" onclick="swapPos(this)">Staff direct login</a></span>
            </div> </div></div></div></div>
    <script src="{% static 'script.js' %}"></script></body></html>
```

Description:

This is a Django-based user registration template with containing input fields for username, email, password, and password confirmation. It includes a terms and conditions checkbox, a submit button, and links for user login and staff registration. The template also references external CSS (style.css) and JavaScript (script.js) for styling and functionality.

Output:


SIGN UP
Username
Email
Password
Confirm Password
I agree to all Terms & Conditions
Register
Already Registered? Login
Staff Registration? Staff direct login

Login.html

CODE:

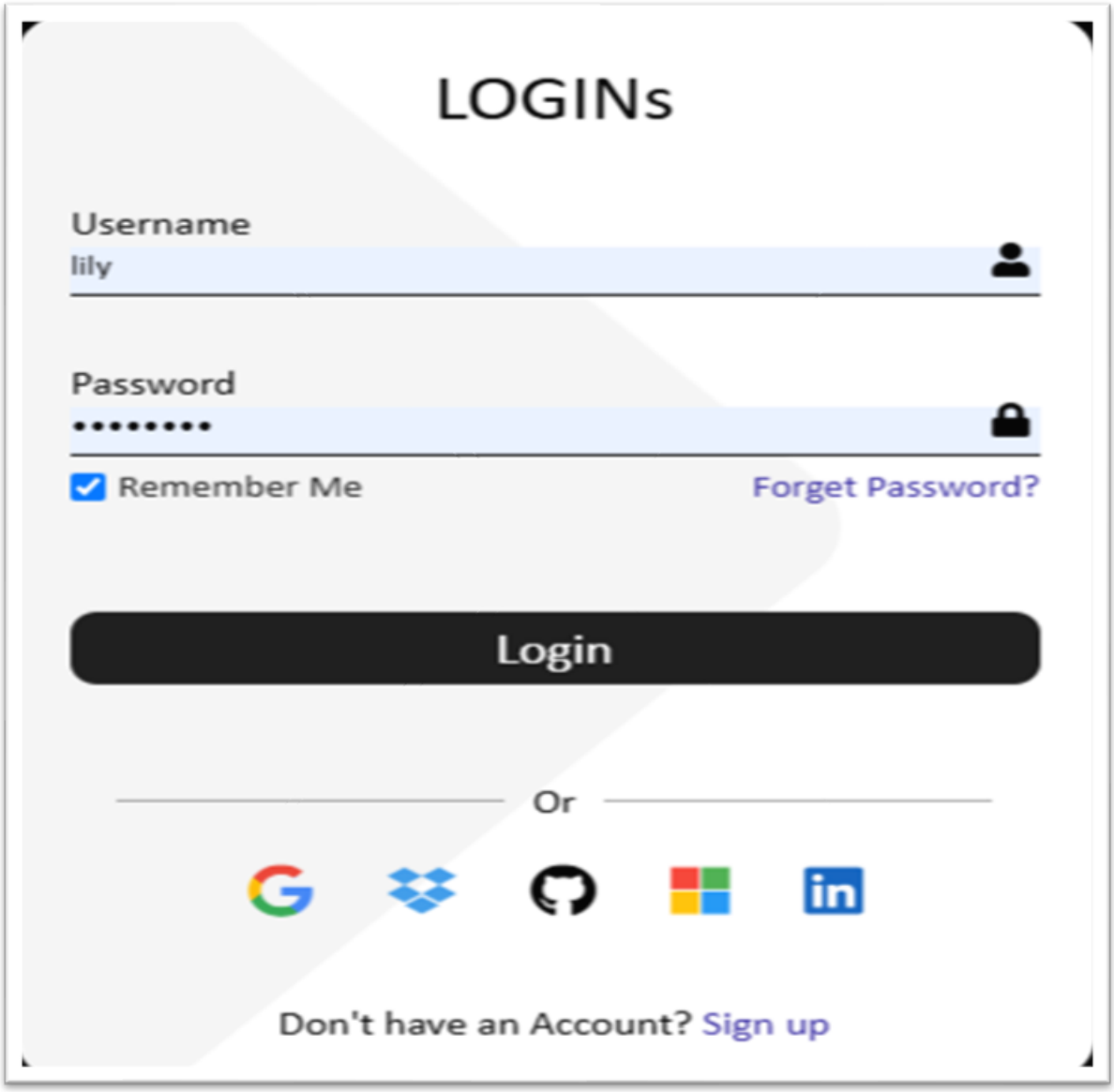
```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'style.css' %}">
</head><body>
   <div id="main-container">
      <div class="form-container">
         <div class="login-form">
            <div class="title">LOGIN</div>
            <form method="post" action="{% url 'login' %}">
               {% csrf_token %}
               <div class="field">
        <input type="text" name="username" id="username" placeholder=" " required>
                  <label for="username">Username</label>
                  <span class="fa fa-user"></span>
               </div> <div class="field">
        <input type="password" name="password" id="password" placeholder=" "
required>
```

```
 <label for="password">Password</label>
 <span class="fa fa-lock"></span>
</div><section>
                <label for="remember"><input type="checkbox"
id="remember">Remember Me</label>
                <a href="#">Forget Password?</a>
            </section>
            <button type="submit" class="login-btn">Login</button>
          </form>
          <div class="bottom">
            <div class="other">
              <div class="separator">Or</div>
              <div class="alternatives">
                <a href="#"><span class="fab fa-google"></span></a>
                <a href="#"><span class="fab fa-dropbox"></span></a>
                <a href="#"><span class="fab fa-github"></span></a>
                <a href="#"><span class="fab fa-microsoft"></span></a>
                <a href="#"><span class="fab fa-linkedin"></span></a>
              </div>
            </div>
            <div>Don't have an Account? <a href="{% url 'register' %}"
class="signup-switch" onclick="swapPos(this)">Sign up</a></div>
          </div>
        </div>
      </div>
    </div>
    <script src="{% static 'script.js' %}"></script>
</body>
</html>
```

Description:

This is a Django-based login form template with allowing users to log in using a username and password. It includes a "Remember Me" checkbox, a "Forgot Password" link, and social login options for Google, Dropbox, GitHub, Microsoft, and LinkedIn. The template also provides a sign-up link for new users and references external CSS (style.css) and JavaScript (script.js).

Output:



home.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Home - Assignment Portal</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'homestyle.css' %}">
</head>
<body>
   <!-- Navigation Bar -->
   <header>
      <nav>
         <div class="logo">
            <a href="{% url 'home' %}">Assignment Portal</a>
         </div>
         <ul class="nav-links">
            <li><a href="{% url 'home' %}">Home</a></li>
            <li><a href="{% url 'assignmentst_list' %}">Assignments</a></li>

            <!-- Dropdown for Assignment Details -->
            <li class="dropdown">
               <a href="#" class="dropbtn">Assignment Details <i class="fas fa-caret-
down"></i></a>
               <div class="dropdown-content">
```

```
                    <a href="{% url 'assignment_de' assignment.id %}">Assignment {{
assignment.id }}: {{ assignment.title }}</a>
                {% endfor %}
            </div>
        </li>
        <li><a href="{% url 'send_email' %}">FeedbackUs</a></li>
        <li><a href="{% url 'about' %}">Aboutus</a></li>
        <li><a href="{% url 'register' %}">Logout</a></li>
      </ul>
    </nav>
  </header>
  <!-- Hero Section -->
  <section class="hero">
    <div class="hero-content">
      <h1>Welcome to the Assignment Portal</h1>
      <p>Manage your assignments, submit your work, and get feedback from
teachers—all in one place.</p>
      <a href="{% url 'assignmentst_list' %}" class="cta-btn">Get Started</a>
    </div>
  </section>
  <!-- Assignment List Section -->
  <section class="assignment-list" style="background-color: #fff; padding: 30px;">
    <h2 style="font-size: 28px; color: #343a40;">Assignments Submissions</h2>
    <ul>
      {% for assignment in assignments %}
        <li style="background-color: #f8f9fa; margin-bottom: 10px; border-radius:
8px;">
          <a href="{% url 'viewst_submissi' assignment.id %}">
            <span class="assignment-id" style="font-weight: 600; color:
#17a2b8;">Assignment {{ assignment.id }}</span>
            <span class="assignment-title" style="font-size: 16px; color:
```

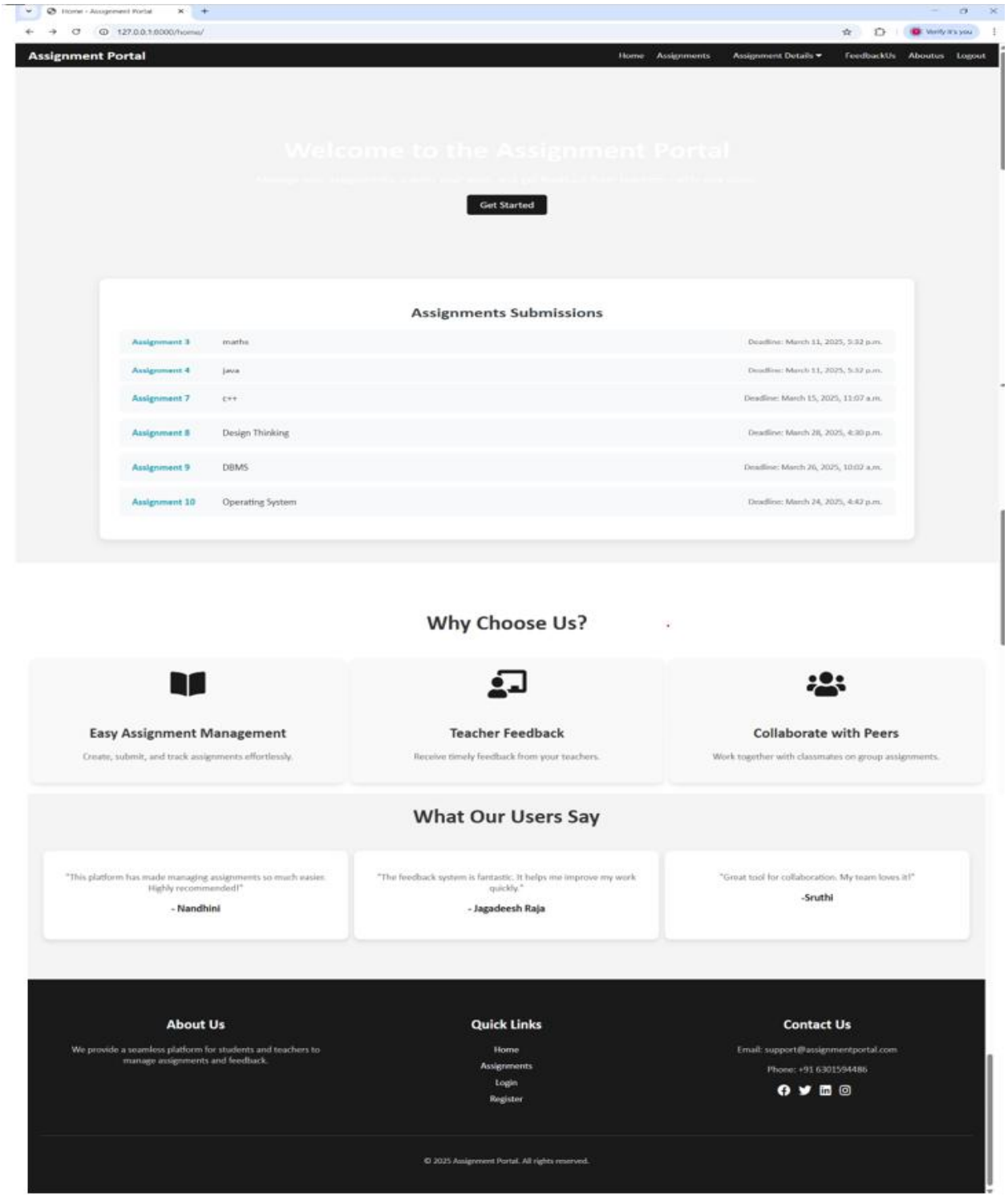
```
<span class="assignment-deadline" style="font-size: 14px; color: #666;">Deadline: {{
assignment.deadline }}</span>
                </a>
            </li>
        {% endfor %}
    </ul>
</section>
<!-- Features Section -->
<section class="features">
    <h2>Why Choose Us?</h2>
    <div class="feature-cards">
        <div class="card">
            <i class="fas fa-book-open"></i>
            <h3>Easy Assignment Management</h3>
            <p>Create, submit, and track assignments effortlessly.</p>
        </div>
        <div class="card">
            <i class="fas fa-chalkboard-teacher"></i>
            <h3>Teacher Feedback</h3>
            <p>Receive timely feedback from your teachers.</p>
        </div>
        <div class="card">
            <i class="fas fa-users"></i>
            <h3>Collaborate with Peers</h3>
            <p>Work together with classmates on group assignments.</p>
        </div></div></section>
<!-- Testimonials Section -->
<section class="testimonials">
    <h2>What Our Users Say</h2>
< div class="testimonial-cards">  <div class="testimonial">
```

```
<p>"This platform has made managing assignments so much easier. Highly recommended!"</p>
            <h4>- Nandhini</h4> </div><div class="testimonial">
            <p>"The feedback system is fantastic. It helps me improve my work quickly."</p>
            <h4>- Jagadeesh Raja </h4>
        </div><div class="testimonial">
            <p>"Great tool for collaboration. My team loves it!"</p>
            <h4>-Sruthi</h4>
        </div></div></section><!-- Footer --><footer>
    <div class="footer-content">
        <div class="footer-section">
            <h3>About Us</h3>
            <p>We provide a seamless platform for students and teachers to manage assignments and feedback.</p>
        </div> <div class="footer-section">
            <h3>Quick Links</h3>
            <ul> <li><a href="{% url 'home' %}">Home</a></li>
                <li><a href="{% url 'assignment_list' %}">Assignments</a></li>
                <li><a href="{% url 'login' %}">Login</a></li>
                <li><a href="{% url 'register' %}">Register</a></li> </ul></div>
        <div class="footer-section"><h3>Contact Us</h3>
            <p>Email: support@assignmentportal.com</p>
            <p>Phone: +91 6301594486</p>
            <div class="social-icons">
                <a href="#"><i class="fab fa-facebook"></i></a>
                <a href="#"><i class="fab fa-twitter"></i></a>
                <a href="#"><i class="fab fa-linkedin"></i></a>
                <a href="#"><i class="fab fa-instagram"></i></a>
            </div></div></div><div class="footer-bottom">
        <p>&copy; 2025 Assignment Portal. All rights reserved.</p>
    </div></footer></body></html>
```

Description:

This is a web-based Assignment Portal homepage designed for students and teachers to manage, submit, and track assignments. It features a user-friendly interface with assignment submission details, deadlines, and sections highlighting platform benefits such as Easy Assignment Management, Teacher Feedback, and Student Collaboration. The page also includes user testimonials, quick navigation links, and contact details for support.

Output:



assignmentst_list.html

CODE:

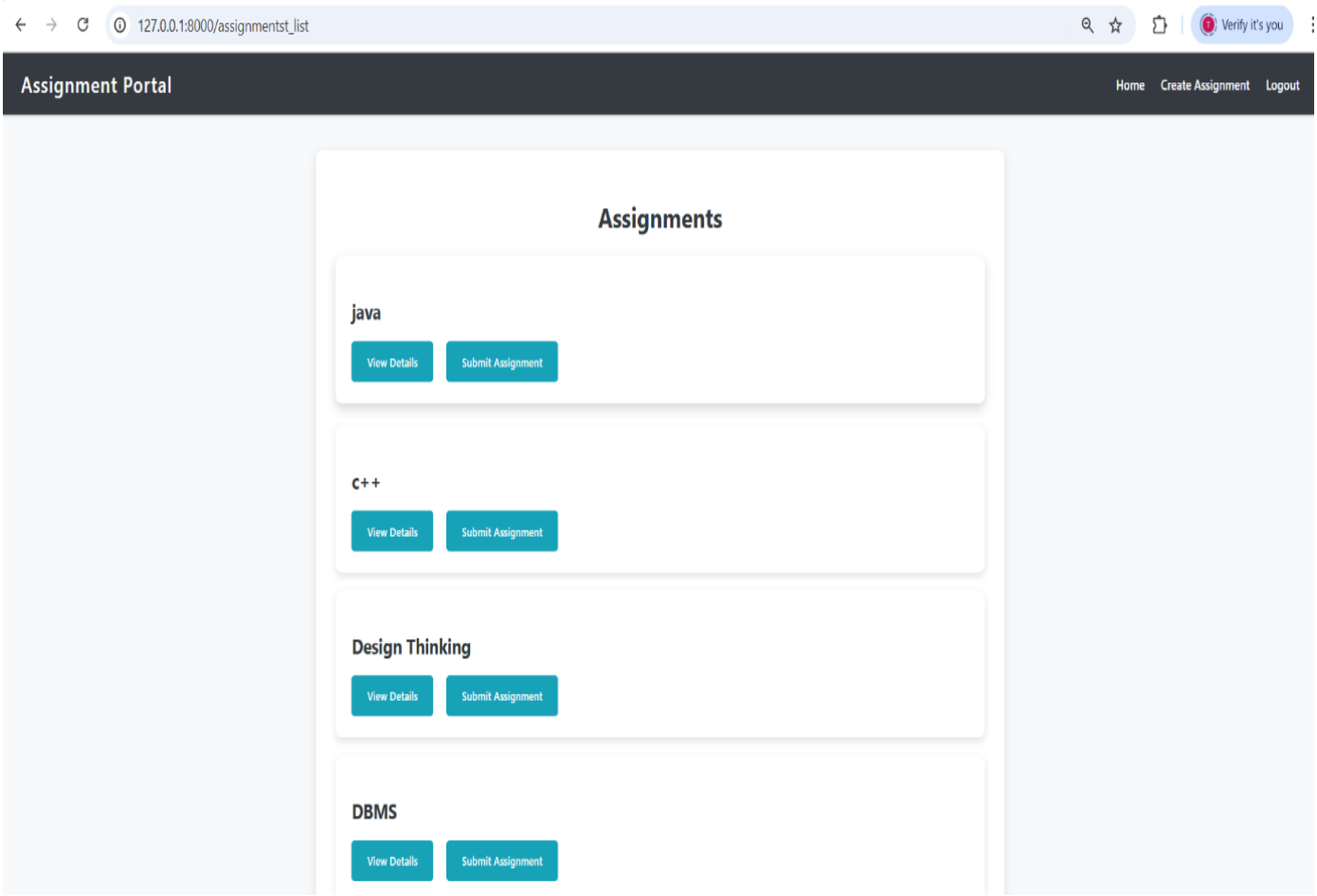
```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignments</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="{% static 'list.css' %}">
</head>
<body>
    <!-- Navigation Bar -->
    <nav>
        <div class="logo">
            <a href="{% url 'assignment_list' %}">Assignment Portal</a>
        </div>
        <ul class="nav-links">
            <li><a href="{% url 'assignment_list' %}">Home</a></li>
            <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
            <li><a href="{% url 'logout' %}">Logout</a></li>
        </ul>
    </nav>
    <!-- Assignments List Section -->
    <div id="main-container">
        <h1>Assignments</h1>
```

```
<ul class="assignments-list">
        {% for assignment in assignments %}
            <li class="assignment-item">
                <h2>{{ assignment.title }}</h2>
                <div class="action-buttons"> <a href="{% url 'assignment_de'
assignment.id %}" class="btn">View Details</a>
<a href="{% url 'submit_assignment' assignment.id %}" class="btn">Submit
Assignment</a>
    {% if submissions %} {% for submission in submissions %}
<a href="{% url 'view_feedback' submission.id %}" class="btn">View Feedback</a>
                {% endfor %}
                {% endif %}
            </div></li>
        {% endfor %}
    </ul></div>
  {% if messages %}
  <div class="messages">
    {% for message in messages %}
        <div class="alert alert-{{ message.tags }}">
            {{ message }}
        </div>
    {% endfor %}
  </div>{% endif %}<br><br><br><br>
  <!-- Footer -->
  <footer><div class="footer-content">
        <p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'home'
%}" style="color: blue; text-decoration: underline; font-weight: bold;">Home</a></p>
    </div>
  </footer></body></html>
```

Description:

This assignments list template displays available assignments dynamically using a loop. It includes a navigation bar with links for home, creating assignments, and logout. Each assignment entry has buttons for viewing details, submitting assignments, and accessing Results. Additionally, the page supports displaying messages and has a footer with copyright information and a home link. The template uses CSS for styling (list.css) and incorporates Django template tags for dynamic content rendering.

Output:



Assignment_de.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignment Detail - {{ assignment.title }}</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="{% static 'detail.css' %}">
</head>
<body>
    <!-- Navigation Bar -->
    <nav>
        <div class="logo">
            <a href="{% url 'assignmentst_list' %}">Assignment Portal</a>
        </div>
        <ul class="nav-links">
            <li><a href="{% url 'home' %}">Home</a></li>
            <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
            <li><a href="{% url 'logout' %}">Logout</a></li>
        </ul>
    </nav>
<br><br><br>
    <!-- Assignment Detail Section -->
    <div id="main-container">
```

```
<div class="assignment-detail">
        <p><strong>Description:</strong> {{ assignment.description }}</p>
        <p><strong>Deadline:</strong> {{ assignment.deadline }}</p>
        <p><strong>Created At:</strong> {{ assignment.created_at }}</p>
        <p><strong>Updated At:</strong> {{ assignment.updated_at }}</p>
      </div>
   </div>
<br><br><br><br><br>
   <!-- Footer -->
   <footer>
      <div class="footer-content">
         <p>&copy; 2025 Assignment Portal. All rights reserved. <a href="{% url 'home' %}"
            style="color: blue; text-decoration: underline; font-weight: bold;">Home</a></p>
      </div>
   </footer>
</body> </html>
```

Description:

This assignment detail page provides specific information about a selected assignment, including its title, description, deadline, creation date, and last update timestamp. It is part of an Assignment Portal where students can view, manage, and submit their assignments. The page follows a clean and minimalistic design with easy navigation links for home, creating assignments, and logging out.

Output:

Submit_assignment.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Submit Assignment - {{ assignment.title }}</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="{% static 'submit.css' %}">
</head
><body>
    <!-- Navigation Bar -->
    <nav>
```

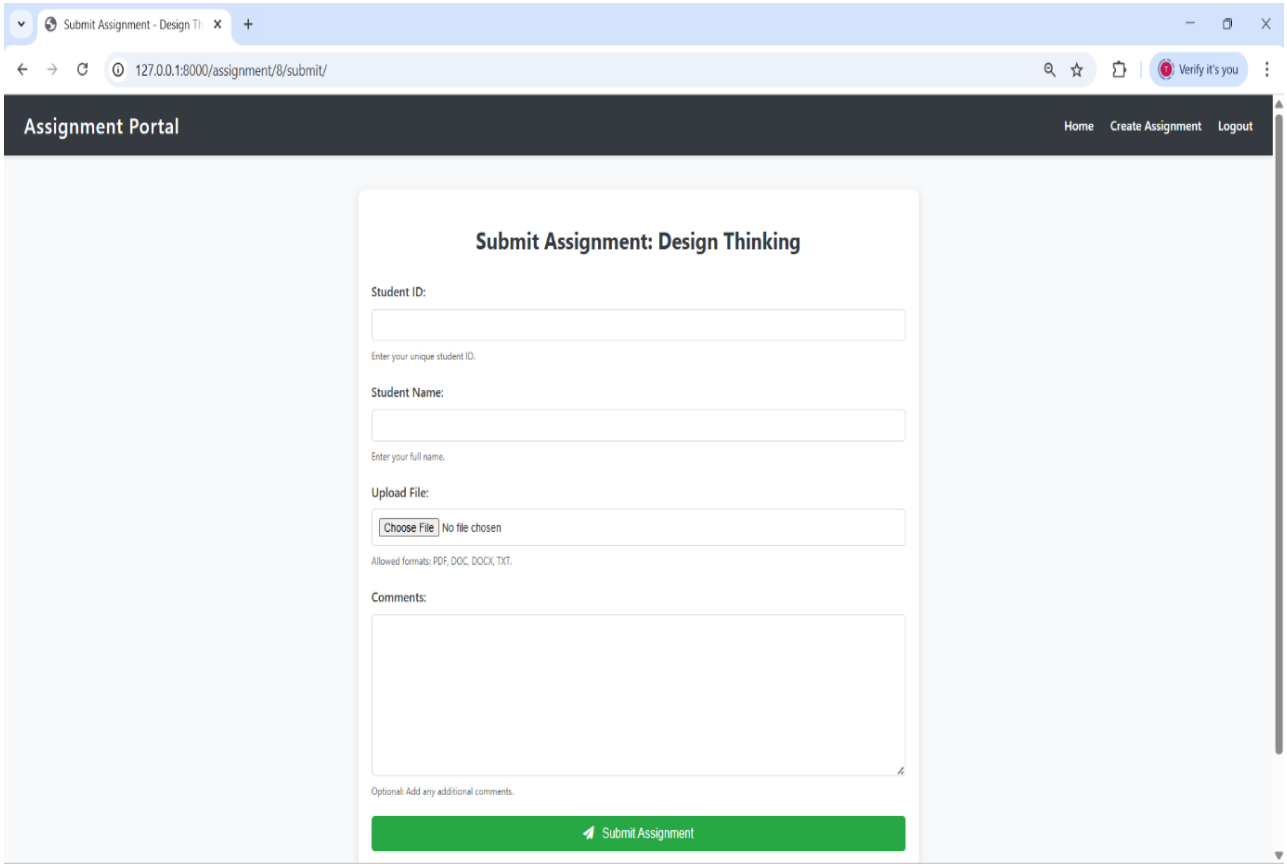
```
<div class="logo">
        <a href="{% url 'assignment_list' %}">Assignment Portal</a></div>
    <ul class="nav-links">
        <li><a href="{% url 'assignment_list' %}">Home</a></li>
        <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
        <li><a href="{% url 'logout' %}">Logout</a></li> </ul></nav>
  <!-- Submission Form -->
  <div id="main-container">
    <h1>Submit Assignment: {{ assignment.title }}</h1>
    <form method="post" enctype="multipart/form-data" class="submission-form">
      {% csrf_token %}
      <!-- Display form errors -->
      {% if form.errors %}
        <div class="form-errors">
          <p>Please correct the errors below:</p><ul>
            {% for field, errors in form.errors.items %}
              {% for error in errors %}
                <li>{{ error }}</li>
              {% endfor %}
            {% endfor %}</ul> </div>
      {% endif %}
      <!-- Student ID -->
      <div class="form-group">
        <label for="id_student_id">Student ID:</label>
        {{ form.student_id }}
        <small class="form-help-text">Enter your unique student ID.</small></div>
      <!-- Student Name -->
      <div class="form-group">
        <label for="id_student_name">Student Name:</label>
        {{ form.student_name }}
```

```
<small class="form-help-text">Enter your full name.</small></div>
        <!-- File Upload -->
        <div class="form-group">
            <label for="id_file">Upload File:</label>
            {{ form.file }}
            <small class="form-help-text">Allowed formats: PDF, DOC, DOCX,
TXT.</small></div>
        <!-- Comments -->
        <div class="form-group">
            <label for="id_comments">Comments:</label>
            {{ form.comments }}
            <small class="form-help-text">Optional: Add any additional
comments.</small> </div>
        <!-- Submit Button -->
        <div class="form-group">
            <button type="submit" class="btn submit-button">
                <i class="fas fa-paper-plane"></i> Submit Assignment
            </button>
        </div> </form></div>
    <!-- Footer -->
    <footer><div class="footer-content">
        <p>&copy; 2025 Assignment Portal. All rights reserved.</p> </div>
      </footer> </body> </html>
```

Description:

"Assignment Portal" webpage where students can submit their assignments. The form includes fields for Student ID, Student Name, file upload (PDF, DOC, DOCX, TXT), and optional comments. A green "Submit Assignment" button is provided at the bottom.

Output:



About.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formers Section with Hover Effect</title>
    <link rel="stylesheet" href="{% static 'styles.css' %}">
</head><body><!-- Formers Section -->
    <section class="formers-section">
        <h2>Our Formers</h2>
```

```
<div class="formers-container">
        <!-- Former 1 -->
        <div class="former-card">
          <div class="image-container">
 <img src="{% static 'Main Photo.jpg' %}" alt="Former 1" class="default-image"></div><h3>Tejaswini Yerra</h3>
          <p>Position:Devolper</p>
        </div> </section><!-- JavaScript for Hover Effect -->
   <script src="{% static 'scripts.js' %}"></script>
   <p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'home' %}" style="color: blue; text-decoration: underline; font-weight: bold;">Home</a>
   </p></body></html>
```

Description:

This about.html template says about the position and name of the person. Due to the design are completely designed by this particular developer .

Output:

Send_email.html

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Send Feedback</title>
   <style>
      /* General Styles */
      body {
         font-family: 'Arial', sans-serif;
         background-color: #f4f4f9;
         margin: 0;
         padding: 0;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
      }
      .feedback-container {
         background: white;
         padding: 2rem;
         border-radius: 10px;
         box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
         width: 100%;
         max-width: 400px;
         text-align: center;}
      h1
```

```
{ color: #333;
        margin-bottom: 1.5rem;
    }
    /* Form Styles */
    form {
        display: flex;
        flex-direction: column; }
    form p {
        margin-bottom: 1rem; }
    input, textarea {
        width: 100%;
        padding: 0.75rem;
        margin-top: 0.5rem;
        border: 1px solid #ddd;
        border-radius: 5px;
        font-size: 1rem;}
    input:focus, textarea:focus {
        border-color: #007bff;
        outline: none;
        box-shadow: 0 0 5px rgba(0, 123, 255, 0.5)}
    button {
        background-color: #007bff;
        color: white;
        padding: 0.75rem;
        border: none;
        border-radius: 5px;
      font-size: 1rem;
        cursor: pointer;
        transition: background-color 0.3s ease;}
    button:hover {
```

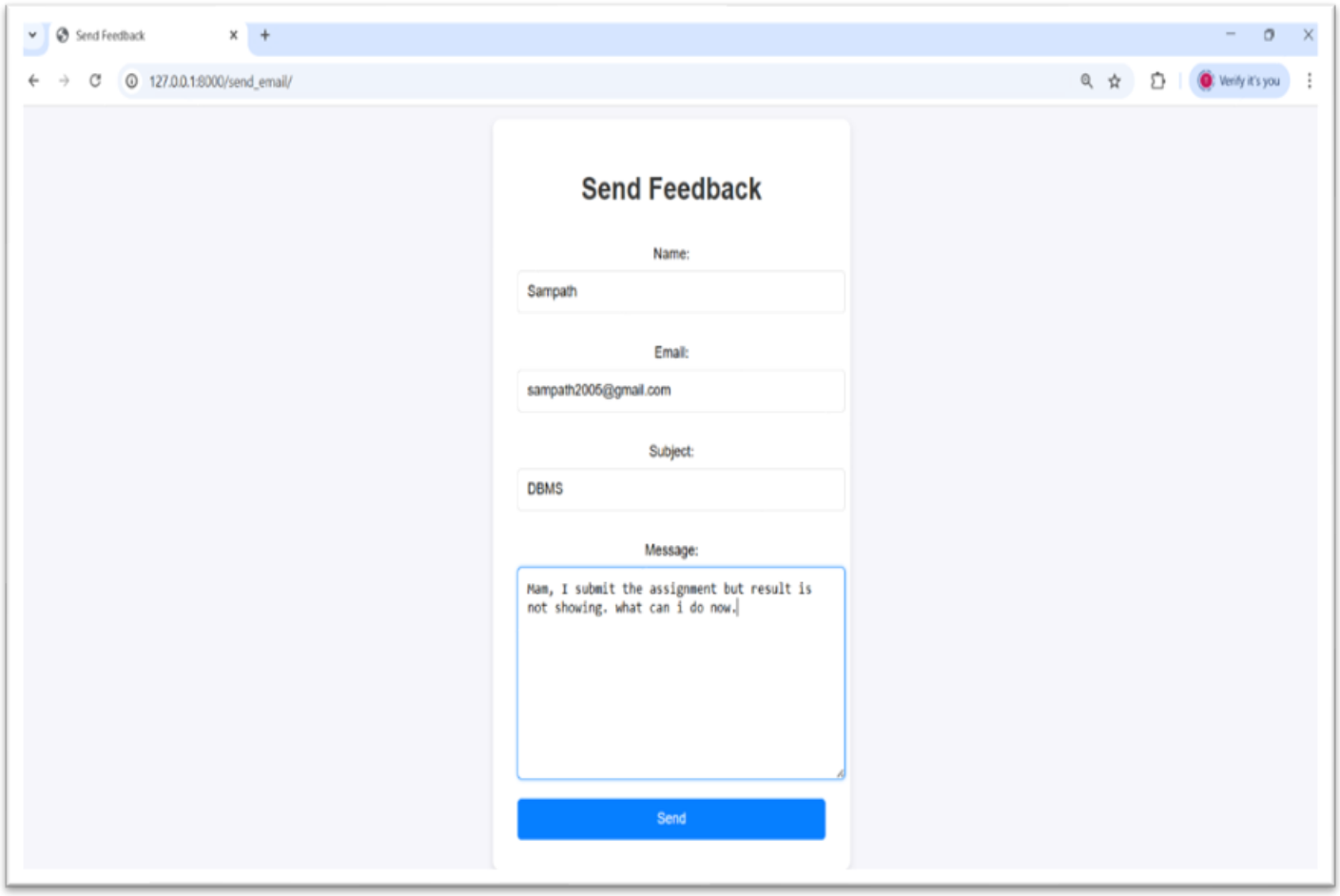
```
        background-color: #0056b3;}
    /* Success Message */
    .success-message {
        color: #28a745;
        margin-top: 1rem;
        display: none;}
    /* Home Link */
    .home-link {
        display: none;
        margin-top: 1rem;}
    .home-link a {
        color: black;
        text-decoration:underline; }
  </style>
 </head>
<body>
  <div class="feedback-container">
    <h1>Send Feedback</h1>
    <form method="post" id="feedbackForm" action="#">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Send</button>
    </form><div class="success-message" id="successMessage">
        Feedback sent successfully!
    </div><div class="home-link" id="homeLink">
        <a href="{% url 'home' %}">Back to Home</a>
    </div> </div>
<script>// JavaScript for form submission feedback
document.getElementById('feedbackForm').addEventListener('submit', function(event)
{event.preventDefault(); // Prevent the form from submitting the traditional way
```

```
        // Simulate form submission success
        setTimeout(function() {
            document.getElementById('feedbackForm').style.display = 'none';
            document.getElementById('successMessage').style.display = 'block';
            document.getElementById('homeLink').style.display = 'block';
        }, 1000);
        // Submit the form programmatically
        fetch(event.target.action, {
            method: 'POST',
            body: new FormData(event.target),
            headers: {
         'X-CSRFToken': document.querySelector('[name=csrfmiddlewaretoken]').value}
})
        .then(response => {
            if (response.ok) {
                console.log('Feedback sent successfully!');
            } else {
                console.error('Failed to send feedback.'); }})
        .catch(error => {
            console.error('Error:', error);  });
    </script>
  </body>
 </html>
```

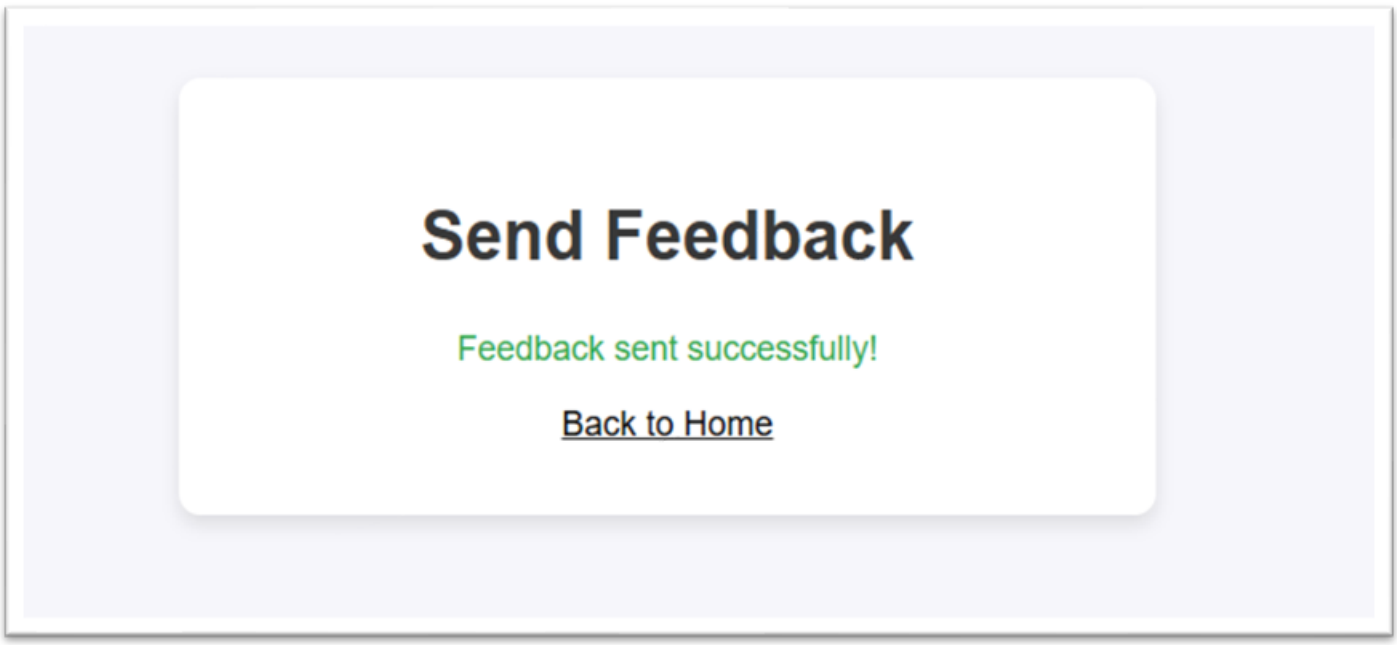
Description:

"Send Feedback" form where a user named Sampath is submitting a query regarding a DBMS assignment. The message states that the assignment was submitted, but the result is not showing, and they are seeking guidance. A blue "Send" button is available at the bottom.

Output:



After sending the feedback. This is the Result



Index.html

CODE *transition: background-color 0.3s ease; }*

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Face Recognition</title>
   <style>
      { margin: 0;
         padding: 0;
         box-sizing: border-box;  }
      body {font-family: Arial, sans-serif;
         background-color: #f4f4f4;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
         flex-direction: column;}
      h1 {
         text-decoration: underline;
         font-size: 2.5rem;
         color: #db2323;
         margin-bottom: 20px;}
      a {text-decoration: none;color: #fff;
         background-color: #007BFF;
         padding: 10px 20px;
         border-radius: 5px;
         font-size: 1.2rem;
```

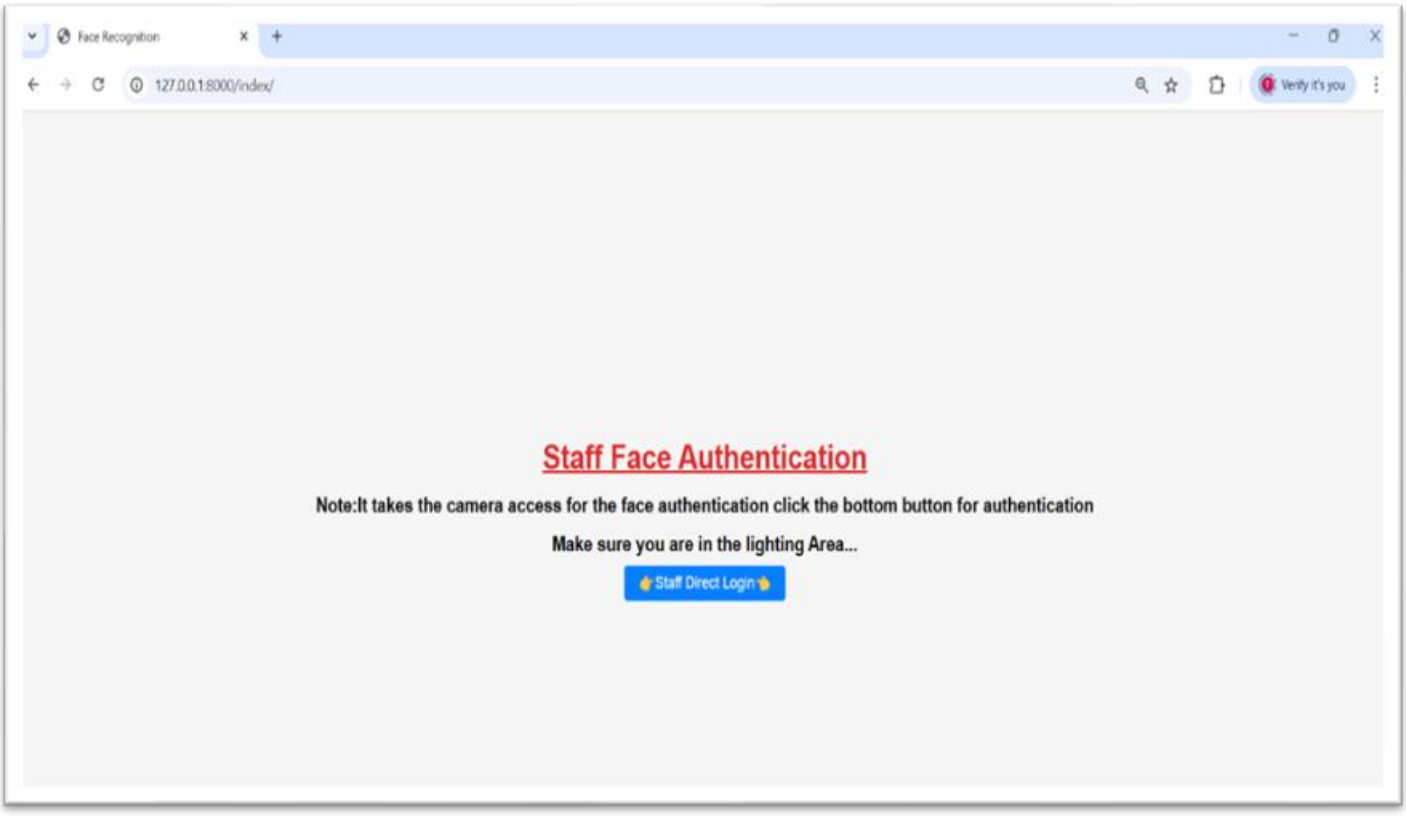
```
        transition: background-color 0.3s ease; }
    a:hover { background-color: #0056b3;  }
    /* Additional Styling for Interactive Elements */
    .container {text-align: center;  }
    .message { margin-top: 20px;font-size: 1.1rem;
        color: #555   }</style></head><body>
<div class="container">
    <h1>Staff Face Authentication</h1>
    <h2>Note:It takes the camera access for the face authentication click the bottom button for authentication </h2><br>
    <h2>Make sure you are in the lighting Area...</h2><br>
    <a href="{% url 'check_face' %}" id="checkFaceLink">
        👉Staff Direct Login 👈</a>
    <p class="message" id="message"></p> </div><script>
    // JavaScript for interactivity
    document.getElementById('checkFaceLink').addEventListener('click',function(event) {  event.preventDefault(); // Prevent the default link behavior
        document.getElementById('message').textContent = 'Checking face... Please wait.';// Redirect to the check_face view after a short delay
        setTimeout(function() {
            window.location.href = "{% url 'check_face' %}";
        }, 1000); // 1 second delay});
</script></body></html>
```

Description:

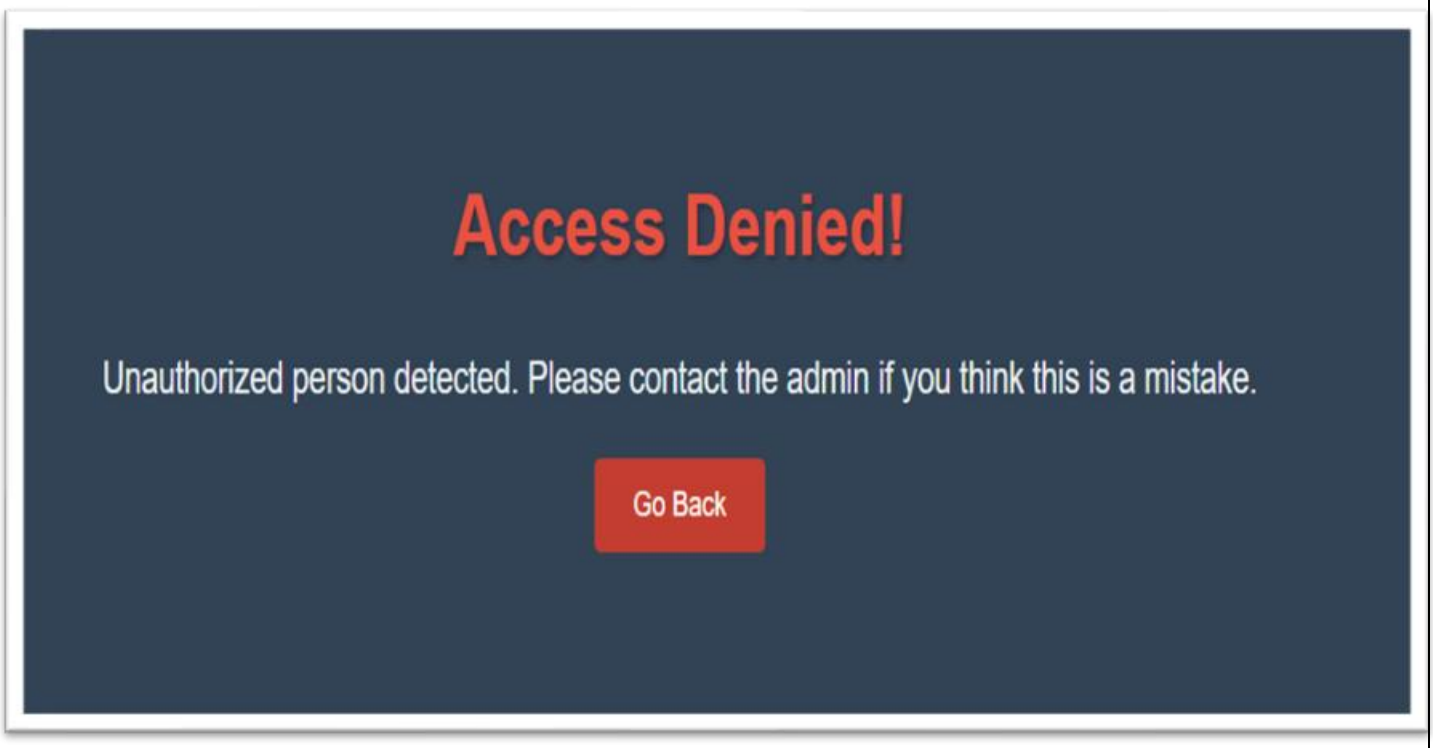
"Staff Face Authentication" page, which allows staff members to log in using facial recognition. It requires camera access for authentication and advises users to be in a well-lit area. A blue "Staff Direct Login" button is provided for initiating the process.

If the unauthorized person was detected. It advises the user to contact the admin if they believe this is a mistake. A red "Go Back" button is provided for navigation.

Output:



If its fail recognition:----



Staff.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Staff Home - Assignment Portal</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'staff.css' %}">
</head><body>
      <!-- Navigation Bar -->
   <nav><div class="logo">
         <a href="{% url 'assignment_list' %}">Staff Portal</a></div>
      <ul class="nav-links">
         <li><a href="{% url 'staff' %}">Home</a></li>
         <li><a href="{% url 'create_assignment' %}">Create Assignments</a></li>
         {% if assignment and assignment.id %}
            <li><a href="{% url 'assignment_detail' assignment.id %}">View
Details</a></li>
         {% else %}
            <li><a href="#" class="disabled">View Details (No Assignment)</a></li>
         {% endif %}
         <li><a href="{% url 'logout' %}">Logout</a></li>
      </ul> </nav><!-- Dashboard Section -->
   <section class="dashboard">
      <h1>Welcome, {{ request.user.username }}</h1>
```

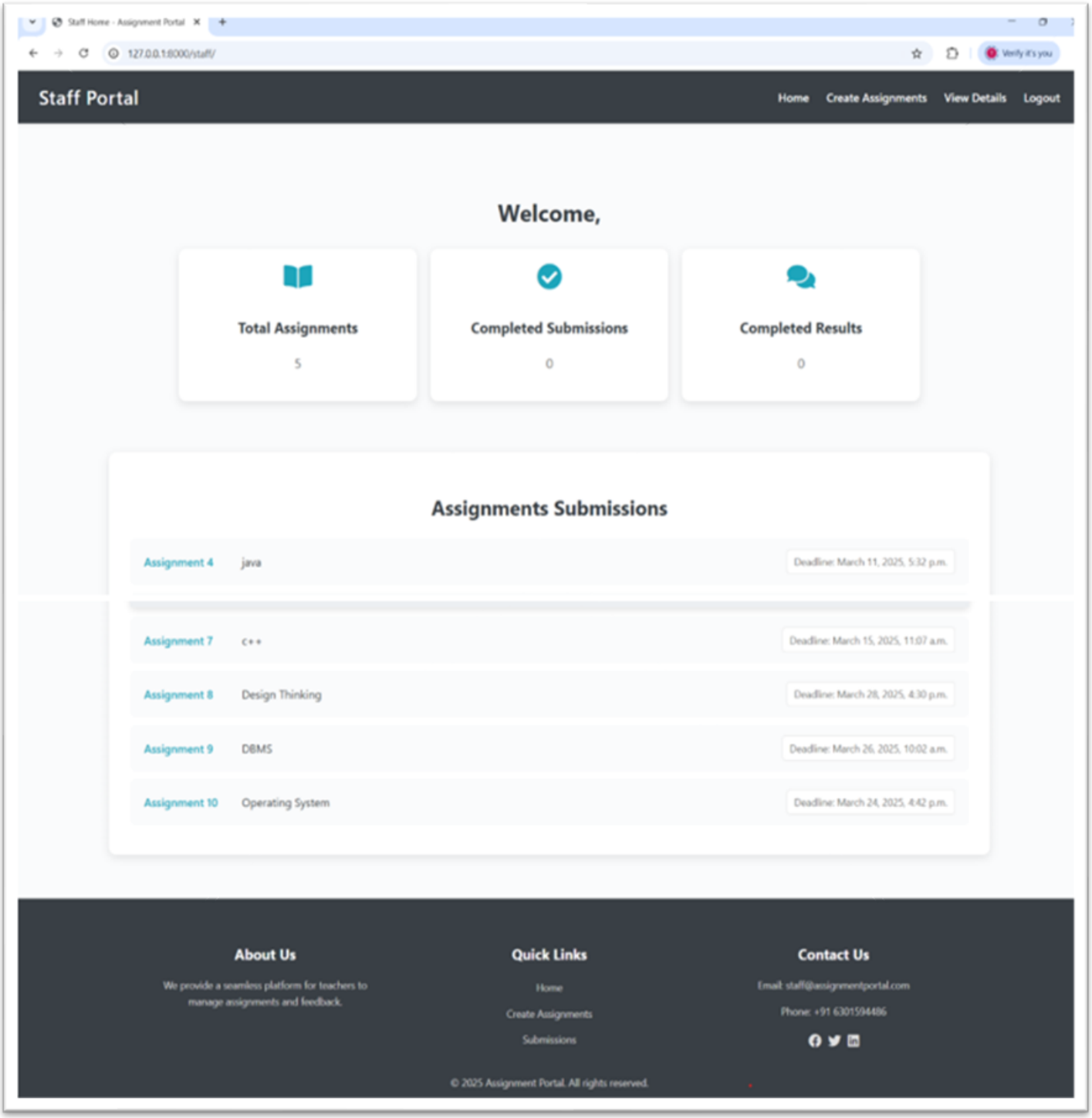
```
        <div class="dashboard-cards">
            <div class="card">
                <i class="fas fa-book-open"></i>
                <h3>Total Assignments</h3>
                <p>{{ total_assignments }}</p>
            </div>
            <div class="card">
                <i class="fas fa-check-circle"></i>
                <h3>Completed Submissions</h3>
                <p>{{ total_submissions }}</p>
            </div>
            <div class="card">
                <i class="fas fa-comments"></i>
                <h3>Completed Results</h3>
                <p>{{ pending_feedback }}</p>
            </div>
        </div>
    </section>
    <!-- Assignment List Section -->
    <section class="assignment-list">
        <h2>Assignments Submissions</h2>
        <ul>
            {% for assignment in assignments %}
                <li>
                    <a href="{% url 'view_submissions' assignment.id %}">
                        <span class="assignment-id">Assignment {{ assignment.id }}</span>
                        <span class="assignment-title">{{ assignment.title }}</span>
                    <span class="assignment-deadline">Deadline: {{ assignment.deadline
}}</span>
                    </a> </li>
```

```
        {% endfor %}
      </ul>
    </section>
    <!-- Footer -->
    <footer>
      <div class="footer-content">
        <div class="footer-section">
          <h3>About Us</h3>
<p>We provide a seamless platform for teachers to manage assignments and
feedback.</p>
        </div><div class="footer-section">
          <h3>Quick Links</h3>
          <ul><li><a href="{% url 'assignment_list' %}">Home</a></li>
            <li><a href="{% url 'create_assignment' %}">Create
Assignments</a></li>
            <li><a href="#">Submissions</a></li>
          </ul></div>
        <div class="footer-section">
          <h3>Contact Us</h3>
          <p>Email: staff@assignmentportal.com</p>
          <p>Phone: +91 6301594486</p>
          <div class="social-icons">
            <a href="#"><i class="fab fa-facebook"></i></a>
            <a href="#"><i class="fab fa-twitter"></i></a>
            <a href="#"><i class="fab fa-linkedin"></i></a>
          </div> </div> </div>
      <div class="footer-bottom">
        <p>&copy; 2025 Assignment Portal. All rights reserved.</p>
      </div>  </footer> </body> </html>
```

Description:

"Staff Portal" dashboard for managing assignments. It displays the total assignments, completed submissions, and completed results, along with a list of assignments and their deadlines. Navigation options include creating assignments, viewing details, and logging out. This portal are design for staff.

Output:



Create_assignment.html

CODE:

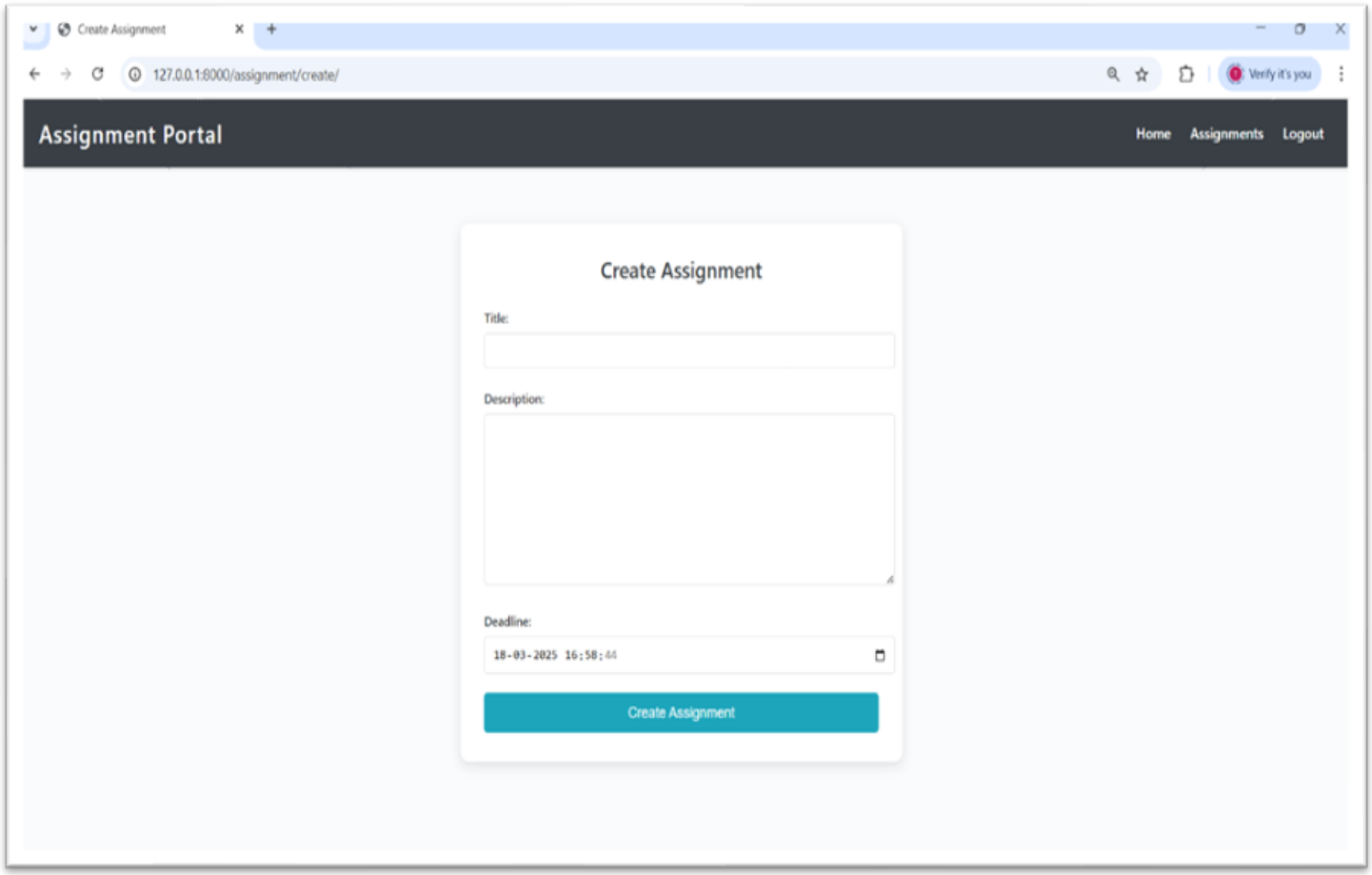
```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Create Assignment</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'create.css' %}">
</head><body>
   <!-- Navigation Bar -->
   <nav> <div class="logo">
         <a href="{% url 'home' %}">Assignment Portal</a>
      </div>  <ul class="nav-links">
         <li><a href="{% url 'home' %}">Home</a></li>
         <li><a href="{% url 'assignment_list' %}">Assignments</a></li>
         <li><a href="{% url 'logout' %}">Logout</a></li>
      </ul></nav>
   <!-- Main Container -->
   <div id="main-container">
      <div class="form-container">
         <div class="create-assignment-form">
            <div class="title"> Create Assignment</div>
            <form method="post">
               {% csrf_token %}
               <div class="field">
                  {{ form.title.label_tag }}
                  {{ form.title }}
```

```
</div><div class="field">
                {{ form.description.label_tag }}
                {{ form.description }}
            </div>
            <div class="field">
                {{ form.deadline.label_tag }}
                {{ form.deadline }}
            </div>
            <button type="submit" class="create-btn">Create Assignment</button>
        </form></div></div></div>
    <!-- Footer -->
    <footer>
        <div class="footer-content">
            <p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'staff'
%}" style="color: blue; text-decoration: underline; font-weight: bold;">Home</a>
            </p></div>
    </footer> </footer> </html>
```

Description:

Create the submission web page for the students and teachers due to both the teacher and student can access the webpage efficiently. The teacher can add the questions on it.

Output:



Assignment_details.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>All Assignments</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="{% static 'detail.css' %}">
</head><body> <!-- Navigation Bar -->
```

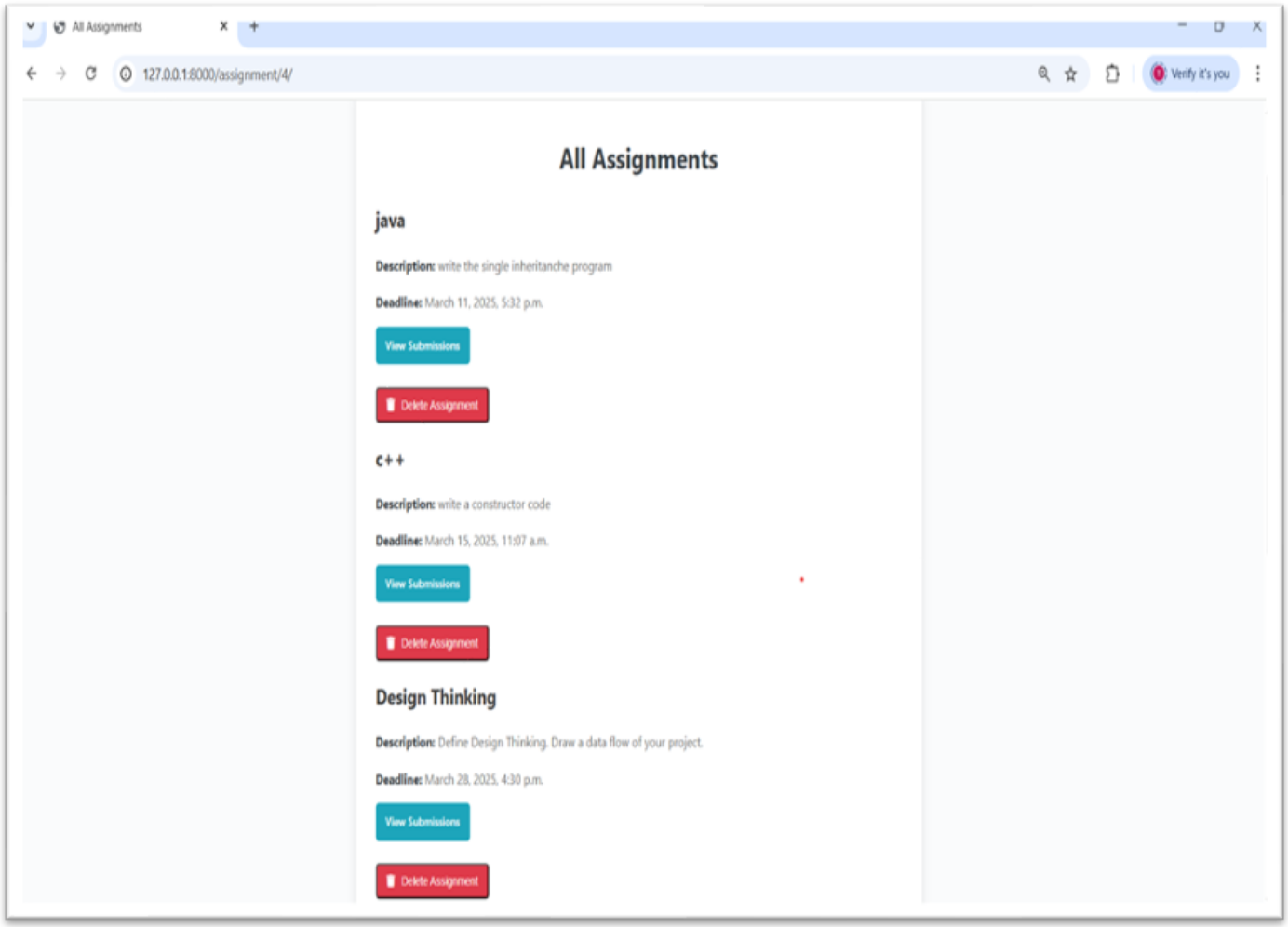
```
    <nav><div class="logo">
     </div><ul class="nav-links">
        <li><a href="{% url 'assignment_list' %}">Home</a></li>
        <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
        <li><a href="{% url 'logout' %}">Logout</a></li>
    </ul></na> <!-- Loop through all assignments -->
  <div id="main-container">
    <h1> All Assignments</h1>
    {% if assignments %}
      {% for assignment in assignments %}
        <div class="assignment-card">
          <h2>{{ assignment.title }}</h2>
          <div class="description">
            <p><strong>Description:</strong> {{ assignment.description }}</p>
          </div> <div class="deadline">
            <p><strong>Deadline:</strong> {{ assignment.deadline }}</p>
          </div> <!-- Buttons for Actions -->
          <div class="action-buttons">
 <a href="{% url 'view_submissions' assignment.id %}" class="btn">View
Submissions</a>
<form action="{% url 'delete_assignment' assignment.id %}" method="post"
class="delete-form"><br>{% csrf_token %}
                <button type="submit" class="btn delete-button" onclick="return
confirm('Are you sure you want to delete this assignment?');">
<i class="fas fa-trash"></i> Delete Assignment</button></form></div></div>
      {% endfor %}{% else %}<p>No assignments found.</p>
    {% endif %}
  </div><br><br><b><!-- Footer -->
  <footer><div class="footer-content">
    <p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'staff' %}"
```

Description:

"All Assignments" page, listing multiple assignments with their descriptions and deadlines. Each assignment has options to view submissions or delete the assignment.

- **View submissions**: view submissions are used to views the student assignment, after that teacher can be the result.
- **Delete the assignment:** Delete the assignment for after the deadline.

Output:



assignment_list.html

CODE:

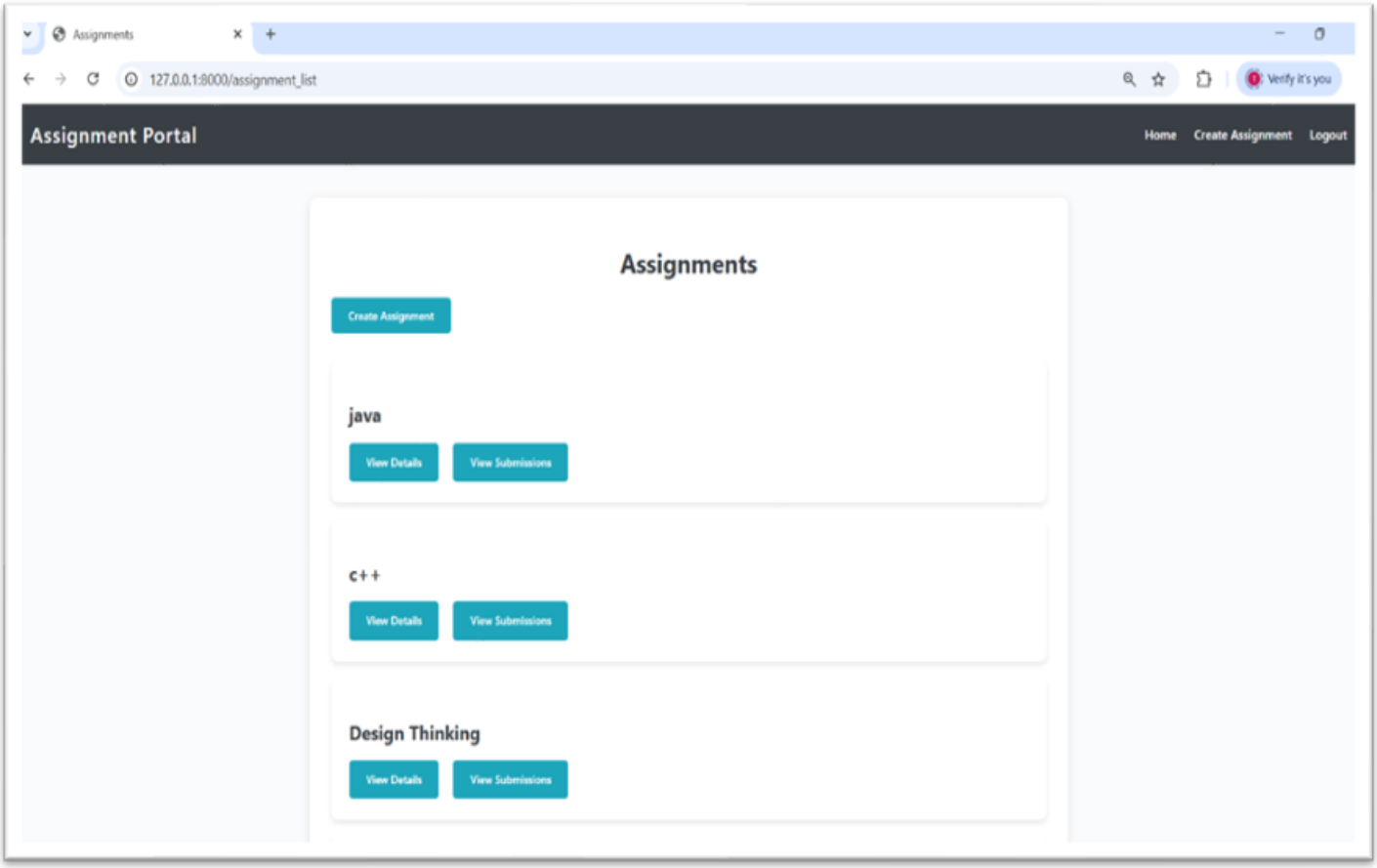
```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignments</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="{% static 'list.css' %}">
</head>
<body>
    <!-- Navigation Bar -->
    <nav>
        <div class="logo">
            <a href="{% url 'staff' %}">Assignment Portal</a>
        </div>
        <ul class="nav-links">
            <li><a href="{% url 'assignment_list' %}">Home</a></li>
            <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
            <li><a href="{% url 'logout' %}">Logout</a></li></ul></nav>
    <!-- Assignments List Section -->
    <div id="main-container">
        <h1>Assignments</h1>
        <a href="{% url 'create_assignment' %}" class="btn">Create
Assignment</a><br>
        <ul class="assignments-list"><br>
```

```
<li class="assignment-item">
<h2>{{ assignment.title }}</h2>
            <div class="action-buttons">
    <a href="{% url 'assignment_detail' assignment.id %}" class="btn">View Details</a>
<a href="{% url 'view_submissions' assignment.id %}" class="btn">View Submissions</a>
            </div> </li>
        {% endfor %}
    </ul> </div><!-- Footer -->
  <footer><div class="footer-content">
        <p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'staff' %}" style="color: blue; text-decoration: underline; font-weight: bold;">Home</a>
    </div>
  </footer>
</body></html>
```

Description:

This Assignment Portal are used to create the assignment by the staff. Here we can create the data directly by the use of create assignment button. And this shows all the Assignments what there presently.

Output:



viewst_submission.html

CODE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Submissions for {{ assignment.title }}
```

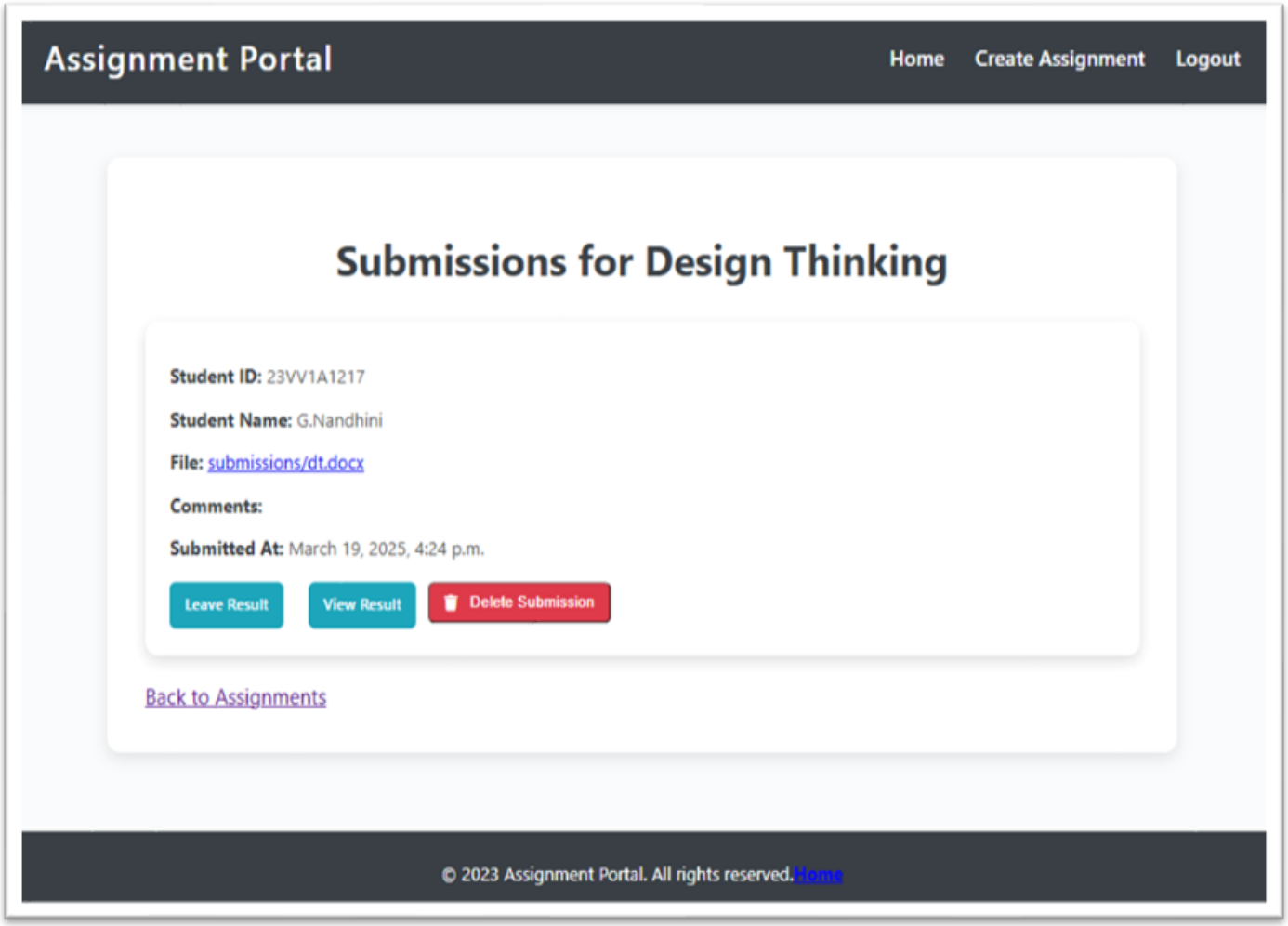
```
</title>
   <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{% static 'detail.css' %}">
</head><body>
<!-- Navigation Bar -->
   <nav><div class="logo">
         <a href="{% url 'assignment_list' %}">Assignment Portal</a>
      </div><ul class="nav-links">
         <li><a href="{% url 'assignment_list' %}">Home</a></li>
         <li><a href="{% url 'create_assignment' %}">Create Assignment</a></li>
         <li><a href="{% url 'logout' %}">Logout</a></li>
      </ul></nav><!-- Submissions List -->
   <div id="main-container">
      <h1>Submissions for {{ assignment.title }}</h1>
      <ul class="submissions-list">
         {% for submission in submissions %}
     <li class="submission-item">
<div class="submission-details"> <p><strong>Student ID:</strong> {{
submission.student_id }}</p>
    <p><strong>Student Name:</strong>
 {{ submission.student_name }}</p>
  <p><strong>File:</strong> <a href="{{ submission.file.url }}">
{{ submission.file.name }}</a></p>
<p><strong>Comments:</strong>
 {{ submission.comments }}</p>
       <p><strong>Submitted At:</strong> {{ submission.submitted_at }}</p>
</div><div class="submission-actions">
          <a href="{% url 'leave_feedback' submission.id %}" class="btn">Leave
Result</a>
```

```
<br>
<a href="{% url 'view_feedback' submission.id %}" class="btn">View Result</a>
<form action="{% url 'delete_submission' submission.id %}" method="post"
class="delete-form">
{% csrf_token %}
<button type="submit" class="btn delete-button" onclick="return confirm('Are you sure
you want to delete this submission?');">
<i class="fas fa-trash"></i> Delete Submission</button></form>  </div> </li>{%
endfor %}</ul>
<a href="{% url 'assignment_list' %}" class="back-link">Back to
Assignments</a></div>
   <!-- Footer --><footer><div class="footer-content">
<p>&copy; 2025 Assignment Portal. All rights reserved.<a href="{% url 'staff' %}"
style="color: blue; text-decoration: underline; font-weight: bold;">Home</a>
    </div>
</footer>
</body>
```

Description:

This Assignment portal are shows the students details .It includes the student's ID, name, file link, submission time, and options to leave a result, view the result, or delete the submission.

Output:



View_feedback.html

CODE:

```
<!-- view_feedback.html -->
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

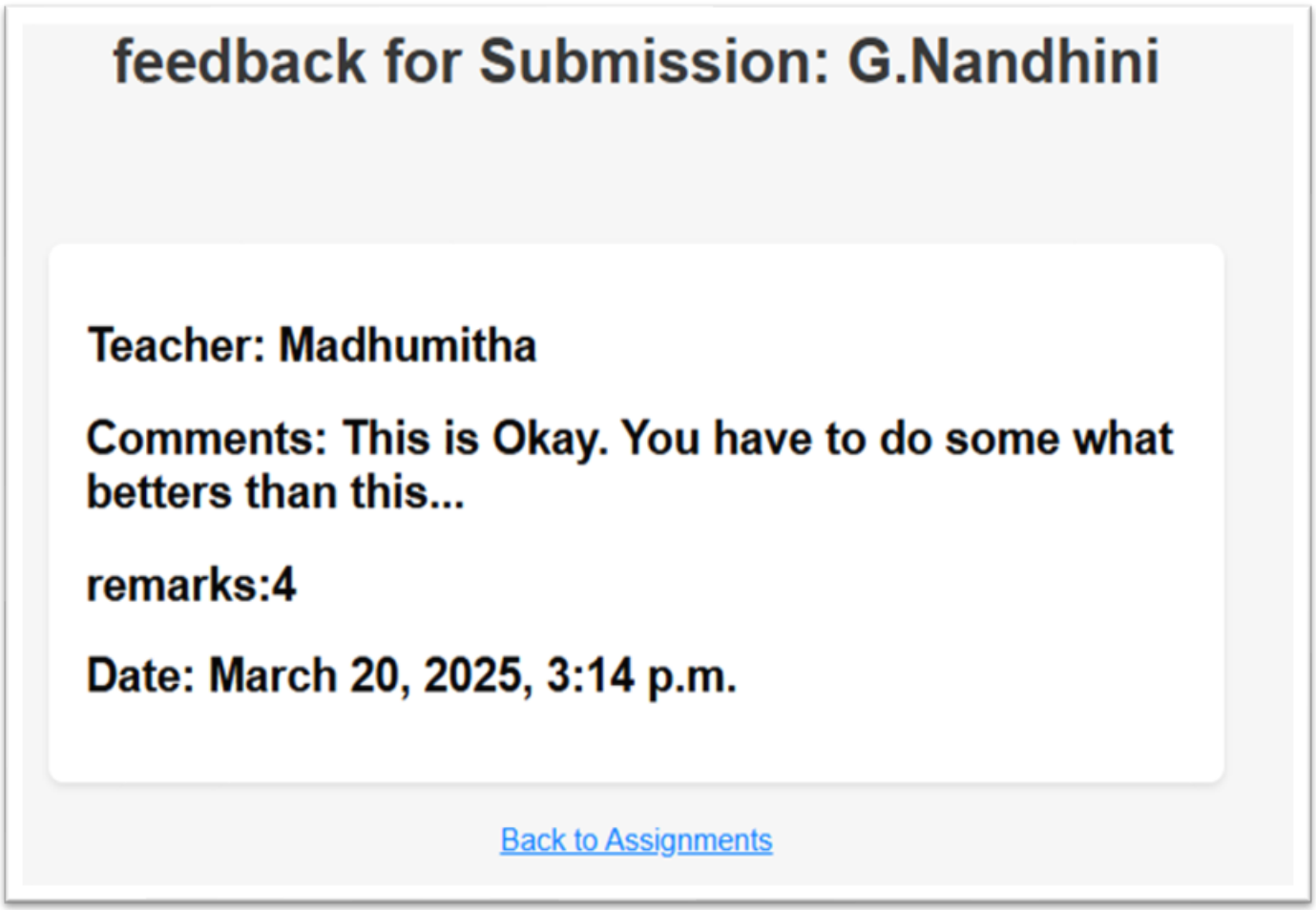
```
<title>Feedback for Submission: {{ submission.student_name }}</title>
  <style>body {
        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
       margin: 0;
        padding: 20px;}
     h1 { color: #333;
        text-align: center;}
    .feedback-container {
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        max-width: 600px;
        margin: 0 auto;}
    .feedback-container p {
        margin: 10px 0;
        font-size: 16px;
        color: #555;}
    .feedback-container p strong {
        color: #333;  }
    .no-feedback {
        text-align: center;
        color: #777;
        font-style: italic;}
    .back-link {
        display: block;
        text-align: center;
```

```
margin-top: 20px;
        color: #007bff;
        text-decoration: none;
        font-size: 16px;}
       back-link:hover {
        text-decoration: underline }
  </style></head><body>
<h1>feedback for Submission: {{ submission.student_name }}</h1>
<br><br><br><div class="feedback-container">
     {% if feedback %}<h2>Teacher: {{ feedback.teacher_name }}</p></h2>
        <h2>Comments: {{ feedback.comments }}<br></h2>
        <h2>remarks:{{ feedback.remarks }}<br></h2>
        <h2>Date:</strong> {{ feedback.created_at }}</p></h2>
     {% else %> <p class="no-feedback">No feedback has been submitted yet.</p>
     {% endif %}</div>
  <a href="{% url 'assignmentst_list' %}" class="back-link">Back to Assignments</a>
</body><html>
```

Description:

It displays feedback for a student's assignment submission. The teachers, provided comments for suggesting something do better . Teacher give results also.

Output:


feedback for Submission: G.Nandhini
Teacher: Madhumitha
Comments: This is Okay. You have to do some what betters than this...
remarks:4
Date: March 20, 2025, 3:14 p.m.
Back to Assignments

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

Forms.py

What is forms.py in Django:

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

Why Use forms.py:

- Simplifies form creation
- Handles input validation automatically
- Integrates with Django models
- Prevents security risks like SQL Injection & CSRF attacks

Types of Forms in Django:

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

Code:

```
from django import forms
from .models import Assignment, Submission, Feedback
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User
# Registration Form
class RegisterForm(UserCreationForm):
    email = forms.EmailField(required=True)
    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
def clean_password2(self):
        # Bypass password validation
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
```

```
if password1 and password2 and password1 != password2:
            raise forms.ValidationError("Passwords do not match.")
        return password2
# Login Form
class LoginForm(AuthenticationForm):
    class Meta:
        model = User
        fields = ['username', 'password']
class AssignmentForm(forms.ModelForm):
    class Meta:
        model = Assignment
        fields = ['title', 'description','deadline']
        widgets = {
            'deadline': forms.DateTimeInput(attrs={'type': 'datetime-local'}),  # Use datetime
picker
        }
class SubmissionForm(forms.ModelForm):
    class Meta:
        model = Submission
        fields = ['student_id','student_name', 'file', 'comments']
class FeedbackForm(forms.ModelForm):
    class Meta:
        model = Feedback
        fields = ['teacher_name', 'comments','remarks']
class EmailForm(forms.Form):
    name = forms.CharField(max_length=100)
    email = forms.EmailField()
    subject = forms.CharField(max_length=100)
    message = forms.CharField(widget=forms.Textarea)
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

MODELS.PY:

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 perform.

How to Apply Models:

Create the Model in models.py:

- Write your models inside the models.py file.

Code:

```
from django.db import models
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.utils import timezone
# Profile Model
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)
```

```
    def __str__(self):
        return self.user.username

# Signal to create a Profile instance when a User is created
@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

# Signal to save the Profile instance when the User is saved
@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()
class Assignment(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    deadline = models.DateTimeField(default=timezone.now)
    def __str__(self):
        return self.title

class Submission(models.Model):
    assignment = models.ForeignKey(Assignment, on_delete=models.CASCADE)
    student_name = models.CharField(max_length=100)
    student_id = models.CharField(max_length=50) # Student name instead of user
    file = models.FileField(upload_to='submissions/')
    comments = models.TextField(blank=True)
```

```
    submitted_at = models.DateTimeField(auto_now_add=True)
    is_completed = models.BooleanField(default=False)
    def __str__(self):
        return f"{self.student_name} - {self.assignment.title}"


class Feedback(models.Model):
    REMARKS_CHOICES = [
        ('5', '5 - Excellent'),
        ('4', '4 - Very Good'),
        ('3', '3 - Good'),
        ('2', '2 - Average'),
        ('1', '1 - Poor'),
 ]


    submission = models.ForeignKey(Submission, on_delete=models.CASCADE,
related_name='feedbacks')
    teacher_name = models.CharField(max_length=100)
    remarks = models.CharField(max_length=2, choices=REMARKS_CHOICES,
default='5')
    comments = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return f"Feedback by {self.teacher_name} for {self.submission.student_name}"
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

DATE :

Signature of Faculty:

Database in Django:

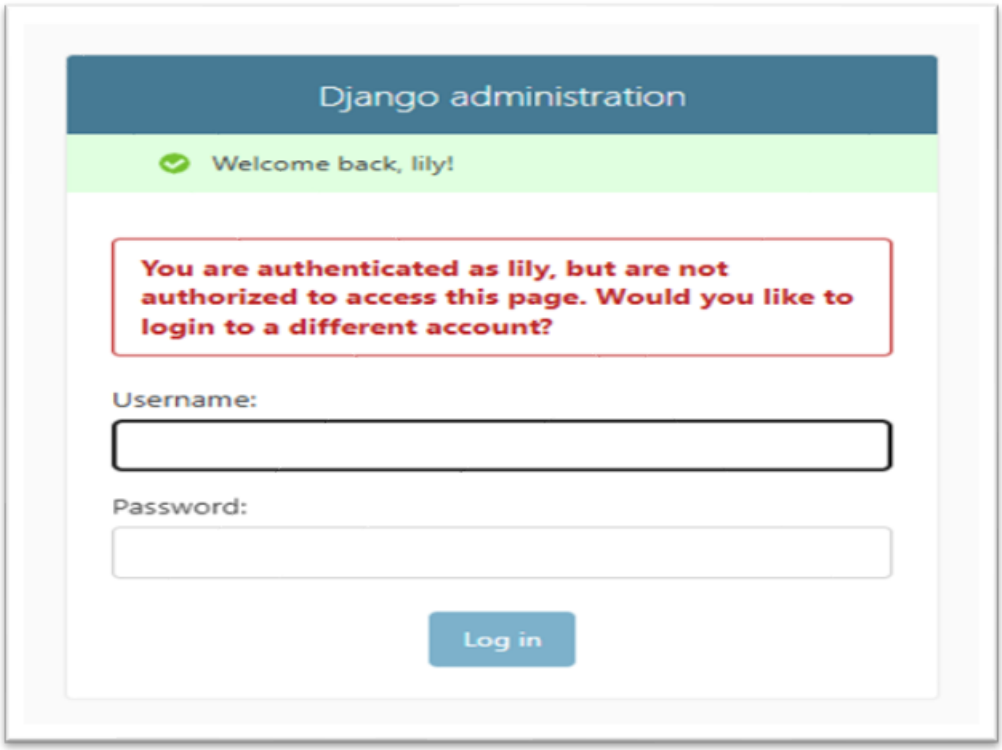
Django uses SQLite by default. The database is stored as a single file named db.sqlite3 in your project directory.

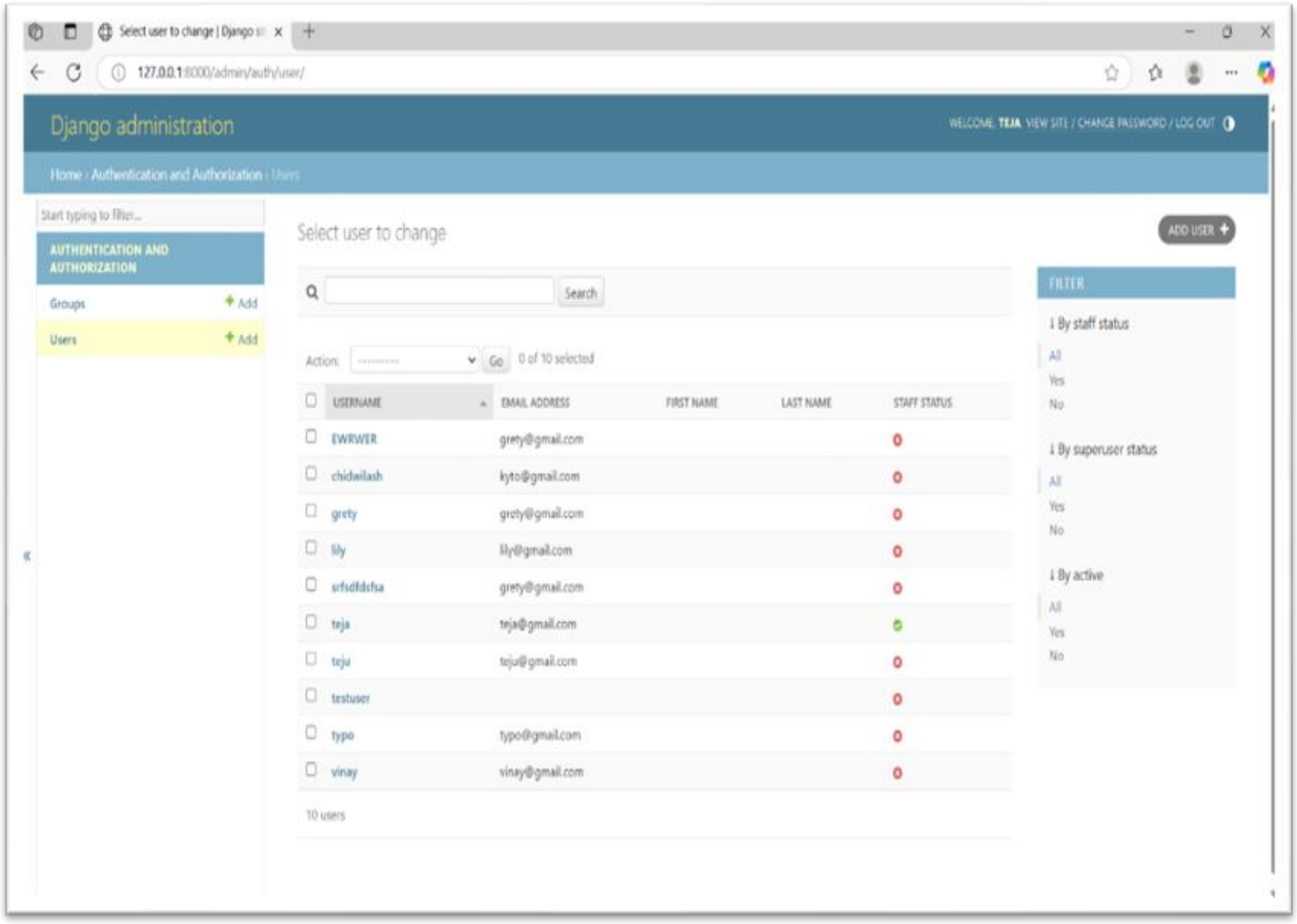
CODE:

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

How Databases store in Django:

- Django models define how data is structured.
- Migrations (python manage.py migrate) create tables in the database.





Run Migrations to Create Database Tables:

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

```
python manage.py makemigrations
python manage.py migrate
```

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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD
Email: hod. it@intugvcev.edu.in

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

S.No	Ability And Activity	Weightage Of Marks	Day To Day Evalution Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm 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	

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, DigitalOcean, 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/Tejaswiniy19/Assignment_Submission_Portal.git

CERTIFICATIONS:

Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Yerra Tejaswini

for successfully completing the course

CSS3

on January 28, 2025

Infosys | Springboard

Congratulations! You make us proud!

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

Issued on: Tuesday, January 28, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Yerra Tejaswini

for successfully completing the course

JavaScript

on January 29, 2025

Infosys | Springboard

Congratulations! You make us proud!

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

Issued on: Wednesday, January 29, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Infosys®
Navigate your next

CERTIFICATE OF ACHIEVEMENT

The certificate is awarded to

Yerra Tejaswini

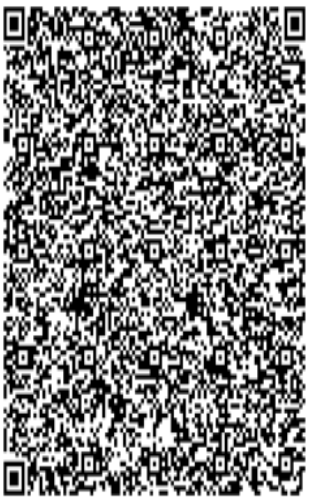
for successfully completing

Front End Web Developer Certification

on February 21, 2025

Infosys | Springboard

Congratulations! You make us proud!



Issued on: Friday, February 21, 2025
To verify, scan the QR code at https://verify.onwingspan.com

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