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

DJANGO FRAMEWORK LAB

ONLINE EXAM REGISTRATION SYSTEM

BY

CH.SAI RUPINI 23VV1A1210

UNDER GUIDANCE OF
MRS.MADHUMITA CHANDA
DEPARTMENTOF INFORMATION TECHNOLOGY
JNTU-GV-CV,VIZIANAGRAM

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

Regd.No : 23VV1A1210

CERTIFICATE

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

No.of Tasks Completed and Certified:

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

Date:

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

Website:www.jntugvcev.edu.in

Subject Name: DJANGO FRAMEWORK Subject Code: R232212SE01

Academic 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)														
		PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
	CO1	3	1	3	1	3	1	1	1	2	3	2	1	3	3	2
	CO2	3	2	3	1	3	1	1	1	2	2	2	2	3	3	3
	CO3	2	3	3	3	3	2	2	2	2	3	3	3	3	3	3

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

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

Signature of the Course Instructor

Table of Contents:

Date: Signature:

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Django libraries
7. Date of Experiment : 13-12-2024
8. Date of Submission of Report : 20-12-2024

Sno	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:**

UNDERSTANDING DJANGO LIBRARIES

1. Python Collections - Container Datatypes:

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

Key Types:

1. List: Ordered, mutable, allows duplicates.
2. Tuple: Ordered, immutable, allows duplicates.
3. Set: Unordered, no duplicates, fast membership testing.
4. Dictionary: Unordered, key-value pairs, fast lookups.

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:

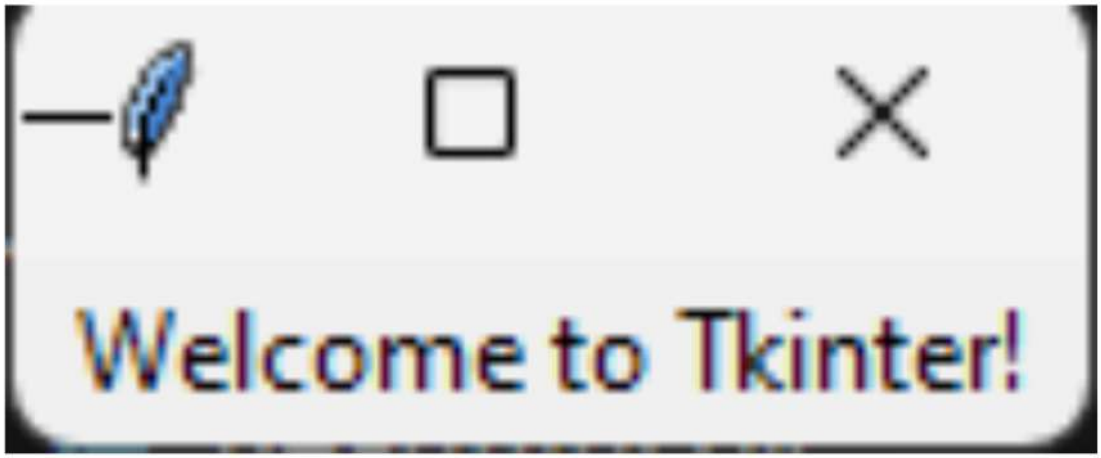
1. Widgets: Buttons, labels, text boxes, etc.
2. Event handling: Respond to user interactions like clicks or key presses.
3. Simple layout management.

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:

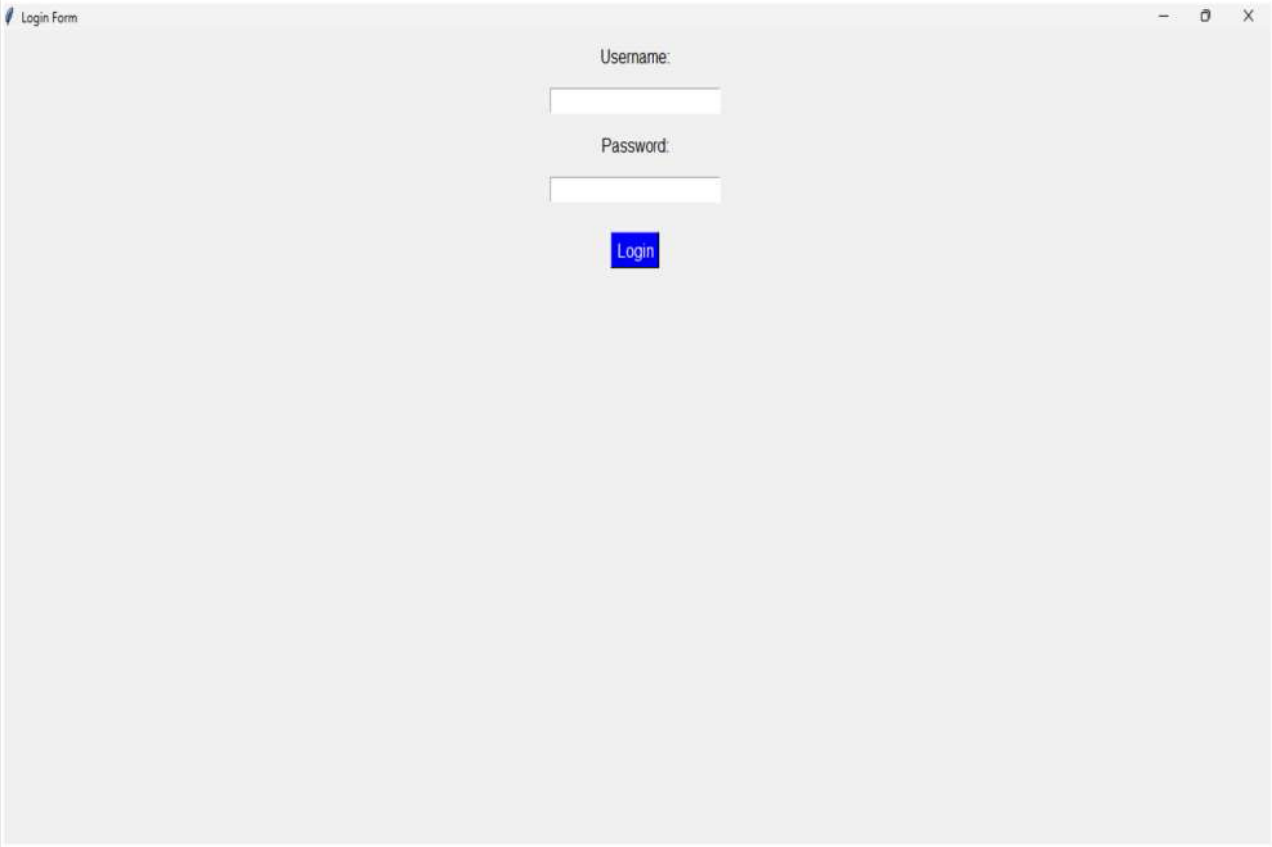
1. Send GET, POST, PUT, DELETE requests easily.
2. Handle request parameters, headers, and cookies.
3. Simple error handling and response handling.

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

Code:

```
from tkinter import Tk, Label, Entry, Button

def login():
    username = username_entry.get()
    password = password_entry.get()
    print(f"Username: {username}, Password: {password}")  # Placeholder for real login logic

# Create main window
root = Tk()
root.title("Login Form")
root.geometry("300x200")  # Set size of the window

# Username Label and Entry
Label(root, text="Username", font=('Arial', 10, 'bold')).pack(pady=(10, 0))
username_entry = Entry(root, width=30)
username_entry.pack(pady=(5, 10))

# Password Label and Entry
Label(root, text="Password", font=('Arial', 10, 'bold')).pack()
password_entry = Entry(root, show="*", width=30)
password_entry.pack(pady=(5, 10))

# Login Button
Button(root, text="Login", width=10, command=login).pack(pady=10)

# Run the application
root.mainloop()
```

Output:



4.CherryPy:

Purpose: Minimalistic web framework for building web applications.

Key Features:

1. Provides a simple and fast HTTP server.
2. Handles routing, cookies, sessions, and file uploads.

Common Use: Building web applications with a lightweight framework.

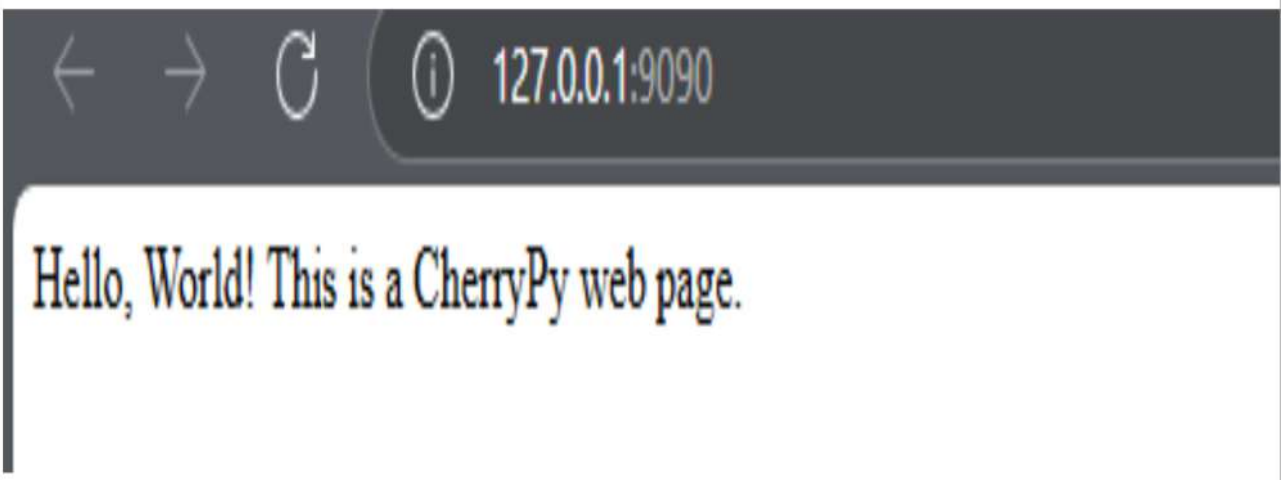
Code:

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

Output:

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

After running the server:



5.Flask:

Purpose: Lightweight micro-framework for building web applications.

Key Features:

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

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

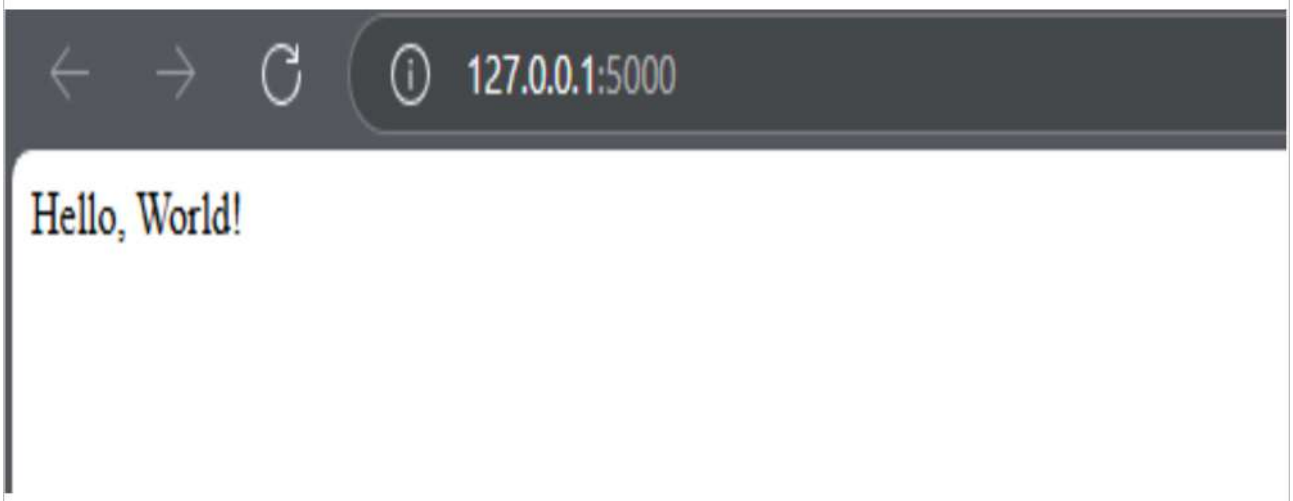
Code:

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

Output:

```
(myenv) C:\Users\Lenovo> * Serving Flask app 'import requests'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 134-121-940
```

After running the server:



6.Bottle:

Purpose: Simple and lightweight WSGI micro-framework.

Key Features:

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

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

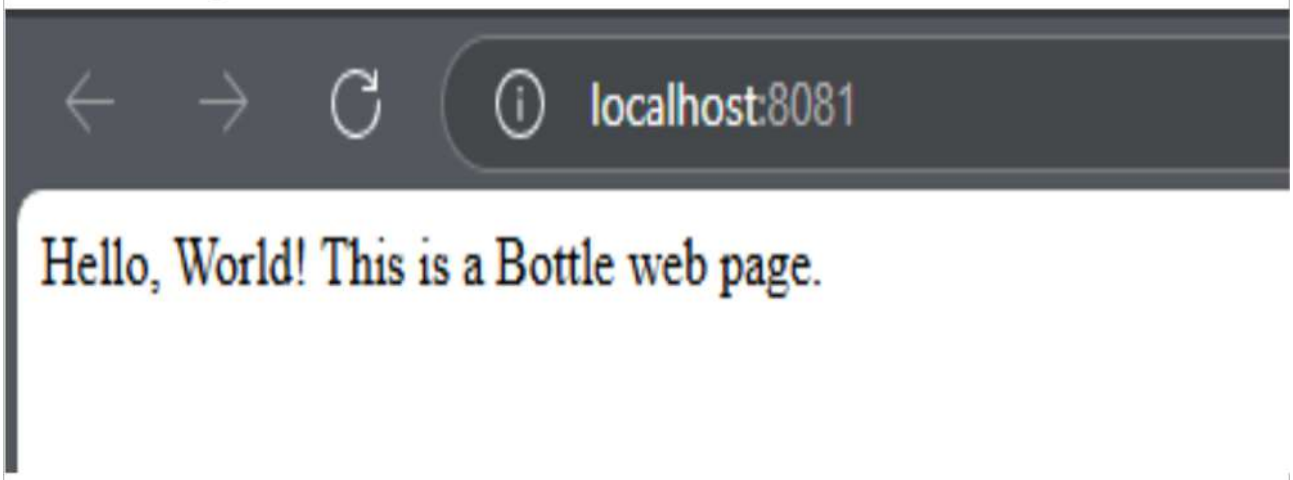
Code:

```
# Import the necessary components from the Bottle framework
from bottle import Bottle, run
# Create an instance of the Bottle application
app = Bottle()
# Define a route for the root URL ('/')
@app.route('/')
def home():
    # This function returns a simple welcome message when the root URL is accessed
    return "Hello, welcome to Bottle framework!"

# Entry point of the application
if __name__ == '__main__':# Run the Bottle app on localhost at port 8080 with debugging
    run(app, host='localhost', port=8080, debug=True)
```

Output:

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

After running the server:



7.BeautifulSoup4 - Web Scraping:

Purpose: Parses HTML and XML documents to extract data.

Key Features:

1. Easy navigation and searching within HTML.
2. 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 required modules
import requests  # For sending HTTP requests
from bs4 import BeautifulSoup  # For parsing HTML content

# Define a function to scrape quotes from the website
def scrape_quotes():
    base_url = "http://quotes.toscrape.com"  # Base URL of the quotes website
    next_page = "/"  # Starting with the homepage

    # Loop through all pages until there's no next page
    while next_page:
        # Send GET request to the current page
        response = requests.get(base_url + next_page)

        # Check if the request was successful
        if response.status_code == 200:
            # Parse the page content using BeautifulSoup
            soup = BeautifulSoup(response.text, "html.parser")

            # Find all quote texts on the page
            quotes = soup.find_all("span", class_="text")
            # Find all author names corresponding to the quotes
            authors = soup.find_all("small", class_="author")

            # Loop through quotes and authors simultaneously and print them
            for quote, author in zip(quotes, authors):
                print(f'"{quote.text}" - {author.text}\n')

            # Find the "Next" button to navigate to the next page
            next_btn = soup.find("li", class_="next")
            # Get the href link if the "Next" button exists, otherwise stop looping
            next_page = next_btn.a["href"] if next_btn else None
        else:
            # If the request failed, print the status code and exit the loop
            print(f"Failed to fetch webpage. Status code: {response.status_code}")
            break

scrape_quotes()
```

Output:

```
Title of the page: Example Domain

Headings:
Example Domain

Links:
Link: https://www.iana.org/domains/example
```

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Introduction to django framework
7. Date of Experiment : 20-12-2024
8. Date of Submission of Report : 27-12-2024

Sno	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:**

INTRODUCTION TO DJANGO FRAMEWORK

Django is a high-level, open-source web framework written in Python that enables rapid development of secure and maintainable websites. It was created with the goal of simplifying the process of building complex, database-driven web applications. Django follows the **Model-View-Template (MVT)** architectural pattern, which helps in separating the logic of the application, the user interface, and data management.

Key Features of Django

1. **Rapid Development:** Django emphasizes reusability and "don't repeat yourself" (DRY) principles, allowing developers to build web applications quickly and efficiently.
2. **Secure:** It includes built-in protections against common security threats like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
3. **Scalable:** Django is designed to handle high-traffic websites and can scale easily as the project grows.
4. **Versatile:** It supports everything from simple content management systems to social networks and scientific computing platforms.
5. **Built-in Admin Interface:** One of Django's standout features is its automatic admin interface, which is generated from the project's models and provides a powerful way to manage data.

Django Architecture (MVT Pattern)

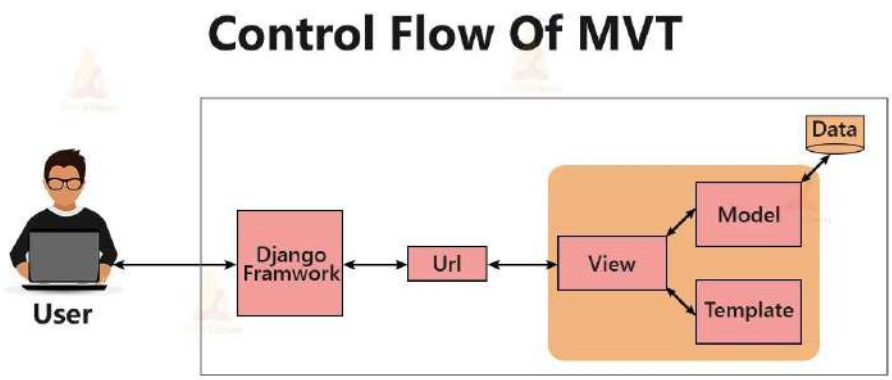
1. **Model:** Defines the structure of the database. Each model maps to a single table in the database.
2. **View:** Contains the logic that processes user requests and returns appropriate responses.
3. **Template:** Handles the presentation layer. Templates define how the data is presented to users using HTML.

Advantages of Django (Brief Overview)

1. **Rapid Development:** Django's built-in tools and clear structure allow developers to build applications quickly with less code.
2. **Security:** It provides protection against common security threats like SQL injection, XSS, and CSRF by default.
3. **Scalability:** Django can handle high-traffic websites and scales well for both small and large projects.
4. **Versatility:** Suitable for a wide range of applications, including e-commerce, social media, CMS, and scientific platforms.

Typical Workflow in Django

1. The user sends a request via the browser.
2. The URL dispatcher routes the request to the appropriate view.
3. The view interacts with the model if necessary and renders a response using templates.
4. The final HTML is returned to the user.



Why Use Django?

1. It speeds up development with less code.
2. It comes with a lot of built-in functionalities.
3. It has strong community support and comprehensive documentation.

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Step by Step Guide to install Django
7. Date of Experiment : 27-12-2024
8. Date of Submission of Report : 03-01-2025

Sno	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 PROCESS

Step 1: Install Prerequisites

Before starting, make sure you have:

1. Python (3.8 or later) installed. Check by running: python –version
2. Visual Studio Code (VS Code)
3. Python Extension for VS Code (Install from the Extensions Marketplace)

Step 2: Create a Virtual Environment

A virtual environment keeps dependencies isolated for your Django project.

On Windows:

Open the Terminal (Ctrl + ~).
Run:
python -m venv .venvActivate it :
.venv\Scripts\activate

Step-3:- Install pip

Install pip in your system using the command:
sudo apt install python3-pip

Step 4: Install Django

With the virtual environment activated, install Django by running: pip install django

Check if Django is installed: django-admin –version

Step 5: Create a Django Project

Run the following command to create a Django project: django-admin startproject myproject .

(The dot (.) ensures that the project is created in the current folder.)

Step 6: Configure VS Code for Django Development

1. In VS Code, open Command Palette (Ctrl + Shift + P).
2. Search for and select Python: Select Interpreter.
3. Choose the Python interpreter inside the virtual environment (.venv).

Step 7: Run the Django Development Server

Start the development server to check if everything works:

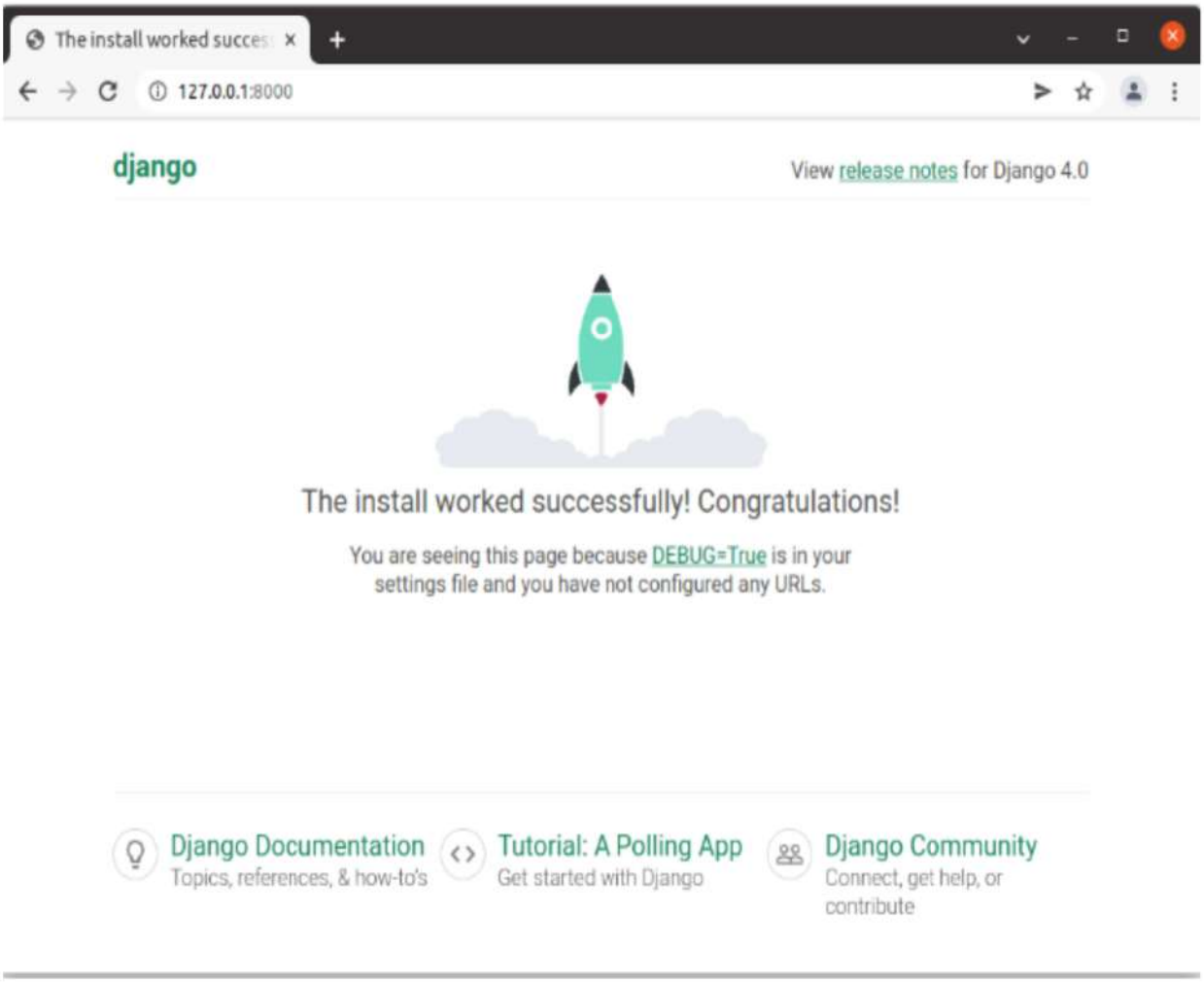
python manage.py runserver

You should see output like:

Starting development server at http://127.0.0.1:8000/

Open the link in your browser to see the default Django welcome page.

Output:



DJANGO PROJECT CREATION

Step 1: Create a New Django Project

Open your terminal or command prompt and run the following command:

django-admin startproject myproject

Replace myproject with your desired project name.

Step 2: Navigate to the Project Directory

Move into the newly created project folder:

cd myproject

Step 3: Run the Development Server

Start the Django development server to verify that the project was created successfully:

python manage.py runserver

Open your browser and go to http://127.0.0.1:8000/ to see the default Django welcome page.

Your Django project is now created and ready for further development! 🚀

DJANGO APP CREATION:

Step 1: Navigate to Your Django Project Directory
Make sure you are inside your Django project folder: cd myproject

Step 2: Create a New Django App
Run the following command: python manage.py startapp myapp.
Replace myapp with your desired app name.
Step 3: Register the App in settings.py
Open myproject/settings.py and add your app to the INSTALLED_APPS list:

```
INSTALLED_APPS = [
# Default Django apps 'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles',
# Your app'myapp',
]
```

Your Django app is now created and registered.

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Linking views and URLs configurations
7. Date of Experiment : 03-01-2025
8. Date of Submission of Report : 24-01-2025

Sno	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:**

CONNECTING VIEWS AND URLS

Connection of Django views and URLs to display **"Hello, World!"** by mapping a URL pattern to a view function that returns the message as an HTTP response.

URL Creation:

In Django, URLs define how different pages and views are accessed in a web application. The URL dispatcher maps URLs to views.

The urls.py file is where you specify patterns to route different URLs to their appropriate view.

For example, to get hello world page the url code would be:

App-level URL Configuration

```
#Importing path

from django.urls import path

from hello import views urlpatterns = [

path("", views.home, name="home")# inking urls to home function of views
```

Project-level URL Configuration:

```
#import admin and path
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp1.urls')),  # Include app URLs
]
```

View Creation:

- A view in Django is a function or class that handles HTTP requests and returns a response.
- Views act as the logic layer of a Django application.
- For example , to get hello world page the views would be:

```
#Importing Http response

from django.http import HttpResponse

        def home(request):# function for displaying hello world

            return HttpResponse("<h1><b>Helloworld</b></h1>")
```

Output:

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Exploring Django views
7. Date of Experiment : 24-01-2025
8. Date of Submission of Report : 31-01-2025

Sno	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 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:

```
# Import necessary Django utilities and models
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseForbidden
from django.db.models import Avg

# Import forms and models
from .forms import (
    StudentRegistrationForm, TeacherRegistrationForm, AdminRegistrationForm,
    LoginForm, StudentCreationForm, StudyMaterialForm,
    EditProfileForm, StudentEditForm, TeacherEditForm
)
from .models import (
    CustomUser, Student, Teacher, Admin, Exam, Question, Result, StudyMaterial
)

# Landing page
def index(request):
    return render(request, 'index.html')

# -------------------- Registration Views --------------------

# Handles student registration
def student_register(request):
    if request.method == 'POST':
        form = StudentRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Student registered successfully. Please log in.")
            return redirect('student_login')
        else:
            messages.error(request, "Registration failed. Please check the form fields.")
    else:
        form = StudentRegistrationForm()
    return render(request, 'student_register.html', {'form': form})

# Handles teacher registration
def teacher_register(request):
    if request.method == 'POST':
```

```
        form = TeacherRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Teacher registered successfully. Please log in.")
            return redirect('teacher_login')
        else:
            messages.error(request, "Registration failed. Please check the form fields.")
    else:
        form = TeacherRegistrationForm()
    return render(request, 'teacher_register.html', {'form': form})

# Handles admin registration
def admin_register(request):
    if request.method == 'POST':
        form = AdminRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Admin registered successfully. Please log in.")
            return redirect('admin_login')
        else:
            messages.error(request, "Registration failed. Please check the form fields.")
    else:
        form = AdminRegistrationForm()
    return render(request, 'admin_register.html', {'form': form})

# ---------- Login / Logout ----------

# Handles login based on user type (student, teacher, admin)
def user_login(request, user_type):
    if request.method == 'POST':
        form = LoginForm(request, data=request.POST)
        if form.is_valid():
            user = authenticate(
                request,
                username=form.cleaned_data['username'],
                password=form.cleaned_data['password']
            )
            if user is not None and user.user_type == user_type:
                login(request, user)
                # Redirect to the respective dashboard
                if user_type == 'student':
                    return redirect('student_dashboard')
                elif user_type == 'teacher':
                    return redirect('teacher_dashboard')
                elif user_type == 'admin':
                    return redirect('admin_dashboard')
            else:
                messages.error(request, "Invalid credentials for this user type.")
    else:
        form = LoginForm()
```

```
    return render(request, f'{user_type}_login.html', {'form': form})

# Logs out the current user
def user_logout(request):
    logout(request)
    return redirect('index')

# ---------- Student Views ----------

# Student dashboard with statistics and progress
@login_required
def student_dashboard(request):
    user = request.user
    total_materials = StudyMaterial.objects.count()
    student_results = Result.objects.filter(student=user, score__isnull=False)
    completed_exams_count = student_results.count()
    passed_exams_count = student_results.filter(score__gte=50).count()
    failed_exams_count = completed_exams_count - passed_exams_count
    avg_score = student_results.aggregate(average_score=Avg('score'))['average_score']

    context = {
        'total_materials': total_materials,
        'completed_exams_count': completed_exams_count,
        'passed_exams_count': passed_exams_count,
        'failed_exams_count': failed_exams_count,
        'avg_score': round(avg_score, 2) if avg_score else 0,
    }

    return render(request, 'student_dashboard.html', context)

# Displays available exams to students
def view_exams(request):
    exams = Exam.objects.all()
    return render(request, 'view_exams.html', {'exams': exams})

# Allows student to take an exam and submit answers
@login_required
def take_exam(request, exam_id):
    exam = get_object_or_404(Exam, pk=exam_id)
    questions = exam.questions.all()
    if request.method == 'POST':
        score = 0
        for question in questions:
            selected = request.POST.get(str(question.id))
            if selected == question.correct_option:
                score += 1
        percentage = int(score / len(questions) * 100)
        Result.objects.create(student=request.user, exam=exam, score=percentage)
        return redirect('student_view_results')
    return render(request, 'take_exam.html', {'exam': exam, 'questions': questions})
```

```
# Displays results of exams taken by the student
@login_required
def view_results(request):
    results = Result.objects.filter(student=request.user)
    return render(request, 'view_results.html', {'results': results})

# Allows students to view study materials
@login_required
def view_study_materials(request):
    materials = StudyMaterial.objects.all()
    return render(request, 'view_materials.html', {'materials': materials})

# Allows students to edit their profile
@login_required
def edit_student_profile(request):
    user = request.user
    if request.method == 'POST':
        form = EditProfileForm(request.POST, instance=user)
        if form.is_valid():
            form.save()
            return redirect('student_dashboard')
    else:
        form = EditProfileForm(instance=user)
    return render(request, 'edit_profile.html', {'form': form})

# ---------- Teacher Views ----------

# Teacher dashboard with exam and student statistics
def teacher_dashboard(request):
    exam_count = Exam.objects.count()
    student_count = Student.objects.count()
    return render(request, 'teacher_dashboard.html', {
        'exam_count': exam_count,
        'student_count': student_count
    })

# Allows teachers to create exams and questions dynamically
@login_required
def create_exam(request):
    if request.method == 'POST':
        title = request.POST.get('title')
        description = request.POST.get('description')
        question_count = int(request.POST.get('question_count', 0))

        exam = Exam.objects.create(
            title=title,
            description=description,
            created_by=request.user
        )
```

```
        # Save each question to the exam
        for i in range(question_count):
            Question.objects.create(
                exam=exam,
                text=request.POST.get(f'question_{i}'),
                option_a=request.POST.get(f'option_a_{i}'),
                option_b=request.POST.get(f'option_b_{i}'),
                option_c=request.POST.get(f'option_c_{i}'),
                option_d=request.POST.get(f'option_d_{i}'),
                correct_option=request.POST.get(f'correct_option_{i}').upper()
            )

        return redirect('teacher_dashboard')

    return render(request, 'create_exam.html')

# Allows teachers to view students and their exam results
@login_required
def view_students_and_results(request):
    if request.user.user_type != 'teacher':
        return redirect('login')  # Optional: You could use HttpResponseForbidden
    students = Student.objects.all()
    results = Result.objects.select_related('student', 'exam')
    return render(request, 'view_students.html', {
        'students': students,
        'results': results,
    })

# Teachers can upload study materials
def upload_material(request):
    if request.method == 'POST':
        form = StudyMaterialForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            messages.success(request, "Material uploaded successfully!")
    else:
        form = StudyMaterialForm()
    return render(request, 'upload_material.html', {'form': form})

# -------------------- Admin Views --------------------

# Admin dashboard with total counts
def admin_dashboard(request):
    student_count = Student.objects.count()
    teacher_count = Teacher.objects.count()
    return render(request, 'admin_dashboard.html', {
        'student_count': student_count,
        'teacher_count': teacher_count
    })
```

```
# Admin management views
def manage_students(request):
    students = Student.objects.all()
    return render(request, 'manage_students.html', {'students': students})

def manage_teachers(request):
    teachers = Teacher.objects.all()
    return render(request, 'manage_teachers.html', {'teachers': teachers})

# Admin can edit student profiles
def edit_student(request, student_id):
    student = get_object_or_404(Student, id=student_id)
    user = student.user
    if request.method == 'POST':
        form = StudentEditForm(request.POST, instance=student, user_instance=user)
        if form.is_valid():
            form.save()
            return redirect('manage_students')
    else:
        form = StudentEditForm(instance=student, user_instance=user)
    return render(request, 'edit_student.html', {'form': form, 'student': student})

# Admin can edit teacher profiles
def edit_teacher(request, teacher_id):
    teacher = get_object_or_404(Teacher, id=teacher_id)
    user = teacher.user
    if request.method == 'POST':
        form = TeacherEditForm(request.POST, instance=teacher, user_instance=user)
        if form.is_valid():
            form.save()
            return redirect('manage_teachers')
    else:
        form = TeacherEditForm(instance=teacher, user_instance=user)
    return render(request, 'edit_teacher.html', {'form': form, 'teacher': teacher})
```

Importance of views:

1. Define the core logic of how your web pages behave.
2. Act as a bridge between the models (data) and templates (HTML).
3. Control what data is shown and how it's processed.

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Setting Up App-Level URLs
7. Date of Experiment : 24-01-2025
8. Date of Submission of Report : 31-01-2025

Sno	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 URLs

Django URLs are configurations that map specific web addresses (URLs) to their corresponding view functions or classes.
Project-level urls.py connects your main project to the URLs defined in your app.
App-level urls.py to define local routing.

App Urls.py:

```
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static
from .views import (
    index, student_register, teacher_register, admin_register,
    user_login, user_logout, student_dashboard, teacher_dashboard, admin_dashboard,
    view_exams, take_exam, view_results, create_exam, view_students_and_results,
    upload_material, view_study_materials, edit_student_profile,
    manage_students, manage_teachers, edit_student, edit_teacher
)

urlpatterns = [
    # 🌐 Homepage
    path('', index, name='index'),

    # 👥 Registration URLs
    path('register/student/', student_register, name='student_register'),
    path('register/teacher/', teacher_register, name='teacher_register'),
    path('register/admin/', admin_register, name='admin_register'),

    # 🔐 Login URLs (Role-based using lambda)
                path('login/student/', lambda request: user_login(request, 'student'),
name='student_login'),
                path('login/teacher/', lambda request: user_login(request, 'teacher'),
name='teacher_login'),
    path('login/admin/', lambda request: user_login(request, 'admin'), name='admin_login'),

    # 🔓 Logout
    path('logout/', user_logout, name='logout'),

    # 🖥️ Student Dashboard and Features
    path('dashboard/student/', student_dashboard, name='student_dashboard'),
    path('student/view-exams/', view_exams, name='student_view_exams'),
    path('student/take-exam/<int:exam_id>/', take_exam, name='student_take_exam_detail'),
    path('student/view-materials/', view_study_materials, name='view_materials'),
    path('student/edit-profile/', edit_student_profile, name='edit_student_profile'),
    path('student/view-results/', view_results, name='student_view_results'),

    # 👨‍🏫 Teacher Dashboard and Features
    path('dashboard/teacher/', teacher_dashboard, name='teacher_dashboard'),
```

```
    path('teacher/create-exam/', create_exam, name='create_exam'),
    path('teacher/view-students/', view_students_and_results, name='view_students_results'),
    path('teacher/upload-material/', upload_material, name='upload_material'),

    # 👨‍💼 Admin Dashboard and Management
    path('dashboard/admin/', admin_dashboard, name='admin_dashboard'),
    path('manage-students/', manage_students, name='manage_students'),
    path('manage-students/<int:student_id>/edit/', edit_student, name='edit_student'),
    path('manage-teachers/', manage_teachers, name='manage_teachers'),
    path('manage-teachers/<int:teacher_id>/edit/', edit_teacher, name='edit_teacher'),
]

# 🖼️ Serving media files during development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Project Urls.py:(Connecting App urls.py)

```
#import admin and path
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp1.urls')),  # Include app URLs
]
```

Importance of URLs:

1. Control how users navigate your website.
2. Route incoming requests to the correct views.
3. Help organize and manage your web application's structure.

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Working with templates in Django
7. Date of Experiment : 31-01-2025
8. Date of Submission of Report : 17-02-2025

Sno	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

A template in Django is an HTML file that contains static content and dynamic placeholders using Django's template language.

Templates allow you to separate HTML design from Python logic, making web development more organized.

Creation of templates:

Step 1: Open onlineexamproject/settings.py
Add import os at the top.

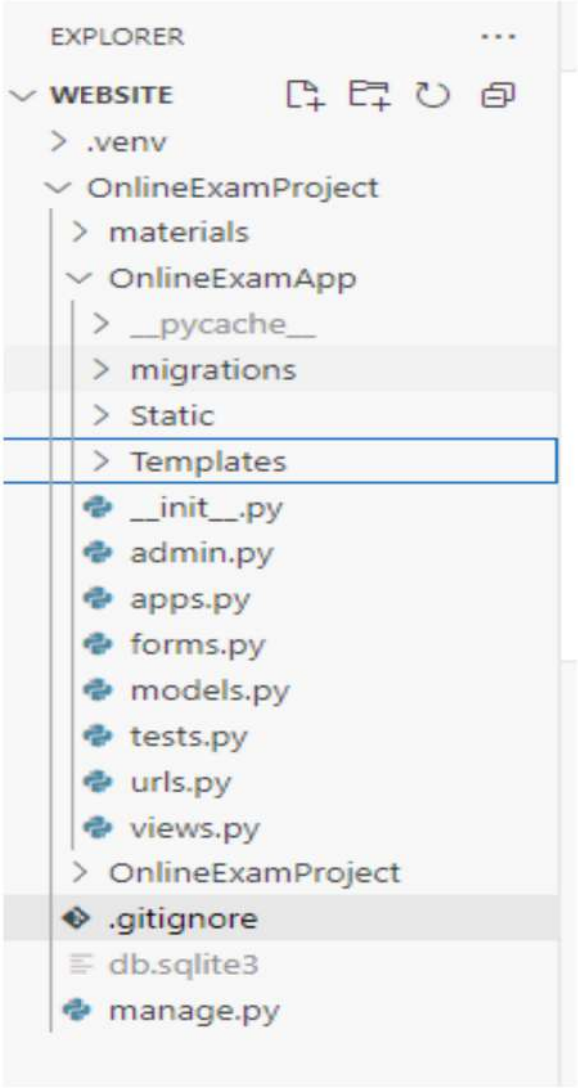
In the TEMPLATES setting:

Set DIRS to [os.path.join(BASE_DIR, 'templates')]

Set APP_DIRS = True to allow Django to find templates inside app folders.

Step 2: Inside your app onlineexam, create a templates/ folder.

Now your directory structure should look like:



Index.html:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack - Landing Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<style>
    html {
    scroll-behavior: smooth;
}

    body {
    font-family: sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f7f7f7;
}

.div.container {
    max-width: 1200px;
    margin: 0 auto;
    position: relative;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px;
    background-color: #fff;
    position: fixed; /* Keeps it fixed at the top */
    top: 0;
    left: 0;
    width: 100%;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Adds shadow for better visibility */
    z-index: 1000; /* Ensures navbar stays on top */
}

/* Prevent content from being hidden behind the fixed navbar */
body {
    padding-top: 80px; /* Adjust based on your navbar height */
```

```
}


.logo {
    font-size: 2em;
    font-weight: bold;
    color: #8856c6;
}

nav ul {
    list-style: none;
    display: flex;
}

nav ul li {
    margin-right: 20px;
}

nav ul li a {
    text-decoration: none;
    color: #333;
}

.auth-buttons button {
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    margin-left: 10px;
    cursor: pointer;
}

.auth-buttons .login {
    background-color: #fff;
    color: #8856c6;
    border: 1px solid #8856c6;
    padding: 10px 20px;
    cursor: pointer;
    transition: all 0.3s ease-in-out;
}


.auth-buttons .register {
    background-color: #8856c6;
    color: #fff;
    padding: 10px 20px;
    cursor: pointer;
```

```
    transition: all 0.3s ease-in-out;
}



.hero {
    text-align: center;
    padding: 80px 20px;
}

.hero h1 {
    font-size: 2.5em;
    color: #333;
    margin-bottom: 20px;
}

.cards {
    display: flex;
    justify-content: center;
    gap: 20px;
    padding: 40px 20px;
}

.card {
    background-color: #e6c9f5;
    padding: 30px;
    border-radius: 10px;
    text-align: center;
    width: 300px;
}

.card h2 {
    color: #8856c6;
    margin-bottom: 10px;
}

.card ul {
    list-style: none;
    padding: 0;
    margin-bottom: 20px;
}

.card ul li {
    margin-bottom: 5px;
}
```

```
.card button {
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    margin: 5px;
    cursor: pointer;
}

.card .login {
    background-color: #fff;
    color: #8856c6;
    border: 1px solid #8856c6;
}

.card .register {
    background-color: #8856c6;
    color: #fff;
}

.teacher-card {
    background-color: #c9f5d1;
}

.admin-card {
    background-color: #f5c9c9;
}

.learn-more {
    text-align: center;
    padding: 40px 20px;
}

.learn-btn {
    padding: 15px 30px;
    border: none;
    border-radius: 5px;
    background-color: #8856c6;
    color: #fff;
    font-size: 1.2em;
    cursor: pointer;
}

.manage-exams {
    display: flex;
    justify-content: center; /* Center horizontally */
    align-items: center; /* Center vertically */
```

```
    height: 100vh; /* Full viewport height */
    padding: 0 15%; /* Adds space on the left and right */
}

.manage-content {
    text-align: center; /* Center text inside */
    background: #c9adeb;
    padding: 20px 40px; /* Extra padding */
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    max-width: 800px; /* Restrict width to avoid stretching */
    width: 100%; /* Ensures responsiveness */
}

.manage-exams h2 {
    color: #8856c6;
    margin-bottom: 20px;
}

.manage-icons {
    display: flex;
    justify-content: center;
    gap: 30px;
}

.manage-icon {
    text-align: center;
}

.manage-icon img {
    width: 50px;
    height: 50px;
    margin-bottom: 10px;
}

.about-examtrack {
    padding: 40px 20px;
}

.about-content {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
}

.about-content h2 {
```

```
    color: #8856c6;
    margin-bottom: 20px;
    width: 100%;
}
.about-content img{
    width: 80px; /* Smaller width */
    height: 80px; /* Smaller height */
    margin-bottom: 10px; /* Slightly reduce bottom margin */
    display: inline-block; /* Ensures it behaves like an inline element */
    object-fit: contain; /* Ensures proper scaling */
}

.trusted-student {
    width: 100%; /* Ensure full width */
    display: flex;
    flex-direction: row; /* Align items in a row */
    gap: 20px; /* Space between blocks */
    flex-wrap: nowrap; /* Prevent wrapping */
    justify-content: space-between; /* Spread blocks evenly */
    align-items: stretch; /* Make all blocks the same height */
}

.trusted, .student-focused, .comprehensive {
    background-color: #f2b7f5;
    padding: 20px;
    border-radius: 10px;
    display: flex;
    align-items: center;
    flex-direction: row;
    gap: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    width: 30%; /* Ensure equal width */
    min-width: 250px; /* Prevent blocks from shrinking too much */
}

/* Ensure images do not break alignment */
.trusted img, .student-focused img, .comprehensive img {
    width: 40px;
    height: 40px;
    object-fit: contain;
}


.trusted h4, .student-focused h4, .comprehensive h4 {
    color: #8856c6;
    margin-bottom: 5px;
```

```
}

trusted p, .student-focused p, .comprehensive p {
    line-height: 1.6;
}

.key-features {
    text-align: center;
    padding: 40px 20px;
}

.features-cards {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 20px;
    margin-top: 30px;
}

.feature-card {
    background-color: #e6c9f5;
    padding: 30px;
    border-radius: 10px;
    width: 300px;
    box-sizing: border-box;
}

.feature-card img {
    width: 50px; /* Smaller width */
    height: 50px; /* Smaller height */
    margin-bottom: 10px; /* Slightly reduce bottom margin */
    display: inline-block; /* Ensures it behaves like an inline element */
    object-fit: contain; /* Ensures proper scaling */
}


.contact-us {
    padding: 40px 20px;
    text-align: center;
}

.contact-us h2 {
    color: #8856c6;
    margin-bottom: 30px;
}
```

```
.contact-content {
    display: flex;
    justify-content: space-between;
    gap: 20px;
}

.get-in-touch, .our-information {
    background-color: #e6c9f5;
    padding: 30px;
    border-radius: 10px;
    width: 48%;
    box-sizing: border-box;
}

.get-in-touch h3, .our-information h3 {
    color: #8856c6;
    margin-bottom: 20px;
}

.get-in-touch label {
    display: block;
    margin-top: 15px;
    text-align: left;
}

.get-in-touch input[type="email"], .get-in-touch textarea {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
}

.get-in-touch textarea {
    height: 150px;
}

.get-in-touch button {
    background-color: #8856c6;
    color: #fff;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    margin-top: 20px;
    cursor: pointer;
```

```
}

.our-information h4 {
    color: #8856c6;
    margin-bottom: 10px;
}

.our-information p {
    line-height: 1.6;
}

.social-icons {
    display: flex;
    gap: 10px;
    margin-top: 15px;
}

.social-icons img {
    width: 30px;
    height: 30px;
}

.main-footer {
    background-color: #e6c9f5;
    padding: 40px 20px;
    color: #333;
}

.footer-content {
    display: flex;
    justify-content: space-between;
}

.footer-logo h2 {
    color: #8856c6;
    margin-bottom: 10px;
}

.quick-links, .legal {
    text-align: left;
}

.quick-links h3, .legal h3 {
    color: #8856c6;
    margin-bottom: 20px;
}
```

```
.quick-links ul, .legal ul {
    list-style: none;
    padding: 0;
}

.quick-links li, .legal li {
    margin-bottom: 10px;
}

.quick-links a, .legal a {
    text-decoration: none;
    color: #333;
}

.copyright {
    text-align: center;
    margin-top: 20px;
}

hr {
    border: none;
    border-top: 1px solid #ccc;
    margin-top: 30px;
}
</style>
<body>
    <div class="container">
        <header>
            <div class="logo">ExamTrack</div>
            <nav>
                <ul>
                    <li><a href="#home">Home</a></li>
                    <li><a href="#about">About Us</a></li>
                    <li><a href="#features">Key Features</a></li>
                    <li><a href="#contact">Contact</a></li>
                </ul>
            </nav>




        </header>

        <section id="home" class="hero">
```

```
            <h1>The modern way to manage and register for your academic examinations</h1>
        </section>

        <section class="cards">
            <div class="card student-card">
                <h2>For Students</h2>
                    <p>Register for exams, track schedules, and manage your academic calendar.</p>
                <ul>
                    <li>Easy Exam Registration</li>
                    <li>Exam Schedule Tracking</li>
                </ul>
                            <button class="login"><a href="{% url 'student_login' %}">Login</a></button>
                        <button class="register"><a href="{% url 'student_register' %}">Register</a></button>
            </div>


            <div class="card teacher-card">
                <h2>For Teachers</h2>
                     <p>Create and manage exams, track student registrations, and set schedules.</p>
                <ul>
                    <li>Create & Manage Exams</li>
                    <li>Track Student Registration</li>
                </ul>
                            <button class="login"><a href="{% url 'teacher_login' %}">Login</a></button>
                        <button class="register"><a href="{% url 'teacher_register' %}">Register</a></button>
            </div>

            <div class="card admin-card">
                <h2>For Administrators</h2>
                <p>Oversee all exams, manage users, and control system settings.</p>
                <ul>
                    <li>User Management</li>
                    <li>System Configuration</li>
                </ul>
                             <button class="login"><a href="{% url 'admin_login' %}">Login</a></button>
                         <button class="register"><a href="{% url 'admin_register' %}">Register</a></button>
            </div>
```

```
    </section>

    <section class="learn-more">
        <button class="learn-btn"><a href="#about">Learn More</a></button>
    </section>

    <section class="manage-exams">
        <div class="manage-content">
            <h2>Manage Your Exams with Ease</h2>
            <div class="manage-icons">
                <div class="manage-icon">
                        <img src="{% static 'Images/Calendar.png' %}" alt="Schedule
Exams">
                    <p>Schedule Exams</p>
                </div>
                <div class="manage-icon">
                    <img src="{% static 'Images/Register.png' %}" alt="Register Easily">
                    <p>Register Easily</p>
                </div>
                <div class="manage-icon">
                            <img src="{% static 'Images/Notification.png'%}" alt="Get
Notifications">
                    <p>Get Notifications</p>
                </div>
            </div>
        </div>
    </section>

    <section id="about" class="about-examtrack">

        <div class="about-content">
            <h2>About ExamTrack</h2>
            <div class="trusted-student">
                <div class="trusted">
                        <img src="{% static 'Images/University.jpg'%}" alt="Trusted by
Universities">
                    <h4>Trusted by Universities</h4>
                      <p>Leading educational institutions rely on ExamTrack for managing
their examination schedules and student registrations.</p>
                </div>
                <div class="student-focused">
                    <img src="{% static 'Images/Students.jpeg'%}" alt="Student-Focused">
                    <h4>Student-Focused</h4>
                        <p>Our platform is designed with students in mind, making the
registration process simple, intuitive, and stress-free.</p>
                </div>
```

```
                <div class="comprehensive">
                        <img src="{% static 'Images/Solution.png'%}" alt="Comprehensive
Solution">
                    <h4>Comprehensive Solution</h4>
                      <p>From scheduling to notification reminders, ExamTrack provides a
complete solution for exam management.</p>
                </div>
            </div>
        </div>
    </section>

    <section id="features" class="key-features">

        <h2>Key Features</h2>
        <div class="features-cards">
            <div class="feature-card">
                <img src="{% static 'Images/Calendar.png' %}" alt="Feature 1">
                <h3>Easy Scheduling</h3>
                    <p>Teachers can easily create and manage exams, set registration
deadlines, and allocate resources efficiently.</p>
            </div>
            <div class="feature-card">
                <img src="{% static 'Images/Students.jpeg'%}" alt="Feature 2">
                <h3>Student Portal</h3>
                   <p>Students can view available exams, register with a few clicks, and
receive confirmation and reminders.</p>
            </div>
            <div class="feature-card">
                <img src="{% static 'Images/Admin.png'%}" alt="Feature 3">
                <h3>Admin Dashboard</h3>
                      <p>Administrators have a comprehensive overview of all exams,
registrations, and user management capabilities.</p>
            </div>
            <div class="feature-card">
                <img src="{% static 'Images/Exam.png'%}" alt="Feature 4">
                <h3>Exam Analytics</h3>
                    <p>Get insights into registration patterns, popular exam times, and
resource utilization.</p>
            </div>
            <div class="feature-card">
                <img src="{% static 'Images/Notification.png'%}" alt="Feature 5">
                <h3>Smart Notifications</h3>
                 <p>Automated reminders for upcoming exams, registration deadlines, and
important updates.</p>
            </div>
            <div class="feature-card">
```

```
                <img src="{% static 'Images/Check.jpeg'%}" alt="Feature 6">
                <h3>Role-Based Access</h3>
                        <p>Secure role-based permissions for students, teachers, and administrators.</p>
            </div>
        </div>
    </section>

    <section id="contact" class="contact-us">
        <h2>Contact Us</h2>
        <div class="contact-content">
            <div class="get-in-touch">
                <h3>Get In Touch</h3>
                <form>
                    <label for="email">Email</label>
                                        <input type="email" id="email" name="email" placeholder="your.email@example.com">
                    <label for="message">Message</label>
                      <textarea id="message" name="message" placeholder="How can we help you?"></textarea>
                    <button type="submit">Send Message</button>
                </form>
            </div>
            <div class="our-information">
                <h3>Our Information</h3>
                <div class="support-team">
                    <h4>Support Team</h4>
                        <p>Our dedicated team is here to help with any questions about ExamTrack.</p>
                </div>
                <div class="working-hours">
                    <h4>Working Hours</h4>
                    <p>Monday to Friday: 9AM - 5PM<br>Weekend: Closed</p>
                </div>
                <div class="follow-us">
                    <h4>Follow Us</h4>
                    <div class="social-icons">
                        <a href="#"><img src="facebook.png" alt="Facebook"></a>
                        <a href="#"><img src="twitter.png" alt="Twitter"></a>
                        <a href="#"><img src="instagram.png" alt="Instagram"></a>
                        <a href="#"><img src="other-social.png" alt="Other Social"></a>
                    </div>
                </div>
            </div>
        </div>
    </section>
```

```
        <footer class="main-footer">
            <div class="footer-content">
                <div class="footer-logo">
                    <h2>ExamTrack</h2>
                                <p>The modern way to manage and register for academic examinations.</p>
                </div>
                <div class="quick-links">
                    <h3>Quick Links</h3>
                    <ul>
                        <li><a href="#about">About Us</a></li>
                        <li><a href="#features">Features</a></li>
                        <li><a href="#contact">Contact</a></li>
                        <li><a href="#" class="login">Log In</a></li>
                        <li><a href="#" class="register">Register</a></li>
                    </ul>

                </div>
                <div class="legal">
                    <h3>Legal</h3>
                    <ul>
                        <li><a href="#">Terms of Service</a></li>
                        <li><a href="#">Privacy Policy</a></li>
                        <li><a href="#">Cookie Policy</a></li>
                    </ul>
                </div>
            </div>
            <hr>
            <p class="copyright">© 2023 ExamTrack. All rights reserved.</p>
        </footer>

    </div>
</body>
</html>
```

Description:

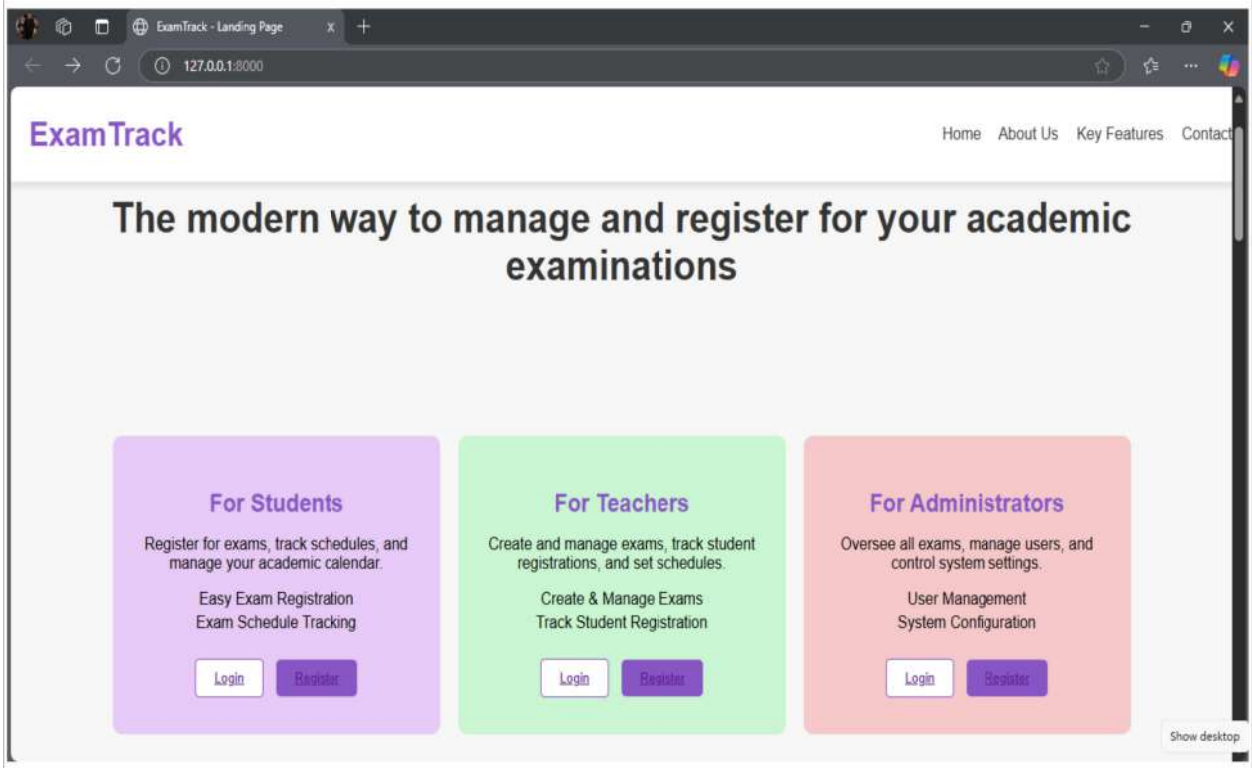
Purpose:

1. Serves as the landing page of the Online Exam Registration System.
2. Provides an introduction to the platform with details about its features and benefits.
3. Acts as a navigation hub to access login, registration, and other pages.

Key Features:

1. Welcome Message: Brief introduction about the platform.
2. Navigation Bar: Links to login, registration, contact, and help sections.
3. Call-to-Action Buttons: "Register Now" and "Login" buttons for quick access.
4. Footer: Includes contact details, FAQs, and social media links

Output:



Student_login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Student Login</title>
    <style>
    /* Import Google Font */
@import                                    url('https://fonts.googleapis.com/css2?
family=Poppins:wght@300;400;600&display=swap');

* {
    margin: 0;
    padding: 0;
```

```
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

/* Background Styling */
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background: linear-gradient(135deg, #4a90e2, #9856f0);
    color: #fff;
}

/* Header */
header {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 40px;
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(5px);
    border-radius: 8px;
    margin-bottom: 30px;
}

.header-logo {
    font-size: 1.8em;
    font-weight: bold;
    color: white;
}

.header-links a {
    text-decoration: none;
    color: white;
    font-weight: bold;
    margin-left: 20px;
    transition: 0.3s;
}

.header-links a:hover {
    color: #ffdb4d;
}
```

```
/* Main Content */
.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 90%;
    max-width: 1000px;
    gap: 20px;
}

/* Student Info Section */
.student-login {
    flex: 1;
    padding: 40px;
    background: rgba(255, 255, 255, 0.2);
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
}

.student-login h2 {
    color: #ffdb4d;
    margin-bottom: 15px;
}

.student-login p {
    color: #f3f3f3;
    line-height: 1.5;
}

.student-login ul {
    list-style: none;
    padding: 0;
    margin-top: 15px;
}

.student-login li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
}

.student-login li::before {
    content: '✔';
    position: absolute;
    left: 0;
```

```
    color: #ffdb4d;
    font-weight: bold;
}

/* Login Form */
.login-form {
    flex: 1;
    padding: 40px;
    background: white;
    border-radius: 12px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
    display: flex;
    flex-direction: column;
    align-items: center;
}

/* Avatar */
.login-form .avatar {
    width: 90px;
    height: 90px;
        background:   url('https://cdn-icons-png.flaticon.com/512/3135/3135715.png')   no-
repeat center;
    background-size: cover;
    border-radius: 50%;
    margin-bottom: 15px;
}

/* Messages */
.alert {
    width: 100%;
    padding: 12px;
    margin-bottom: 10px;
    border-radius: 5px;
    text-align: center;
    font-size: 14px;
}

.alert-success {
    background-color: #d4edda;
    color: #155724;
}

.alert-error {
    background-color: #f8d7da;
    color: #721c24;
}
```

```
/* Form Fields */
.login-form input {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 2px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
    transition: all 0.3s ease;
}

.login-form input:focus {
    border-color: #4a90e2;
    outline: none;
}

/* Submit Button */
button {
    width: 100%;
    padding: 12px;
    background: #4a90e2;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s;
}

button:hover {
    background: #3a78c2;
}

/* Links */
.login-form a {
    text-decoration: none;
    color: #4a90e2;
    margin-top: 10px;
    transition: 0.3s;
}

.login-form a:hover {
    color: #3a78c2;
}
```

```
/* Responsive Design */
@media (max-width: 768px) {
    .main-content {
        flex-direction: column;
        align-items: center;
    }

    .student-login, .login-form {
        width: 90%;
    }
}

</style>
</head>
<body>
    <header>
        <div class="header-logo">ExamTrack</div>
        <div class="header-links">
            <a href="{% url 'teacher_login' %}">Teacher Login</a>
            <a href="{% url 'admin_login' %}">Admin Login</a>
        </div>
    </header>

    <div class="main-content">
        <div class="student-login">
            <h2>Student Login</h2>
             <p>Access your ExamTrack student account to view available exams, register
for tests, and check your exam schedule.</p>
            <p><strong>As a student, you can:</strong></p>
            <ul>
                <li>Browse available exams</li>
                <li>Register for upcoming exams</li>
                <li>View your exam schedule</li>
            </ul>
        </div>

        <div class="login-form">
            <div class="avatar"></div>

            <!-- Display messages -->
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-{{ message.tags }}">
                        {{ message }}
                    </div>
                {% endfor %}
```

```
            {% endif %}

            <!-- Login Form -->
            <form method="POST" action="{% url 'student_login' %}">
                {% csrf_token %}
                {{ form.as_p }}
                <button type="submit">Login</button>
            </form>


            <a href="#">Forgot your password?</a>
                    <a href="{% url 'student_register' %}">Don't have an account? <strong>Register</strong></a>
        </div>
    </div>
</body>
</html>
```

Description:

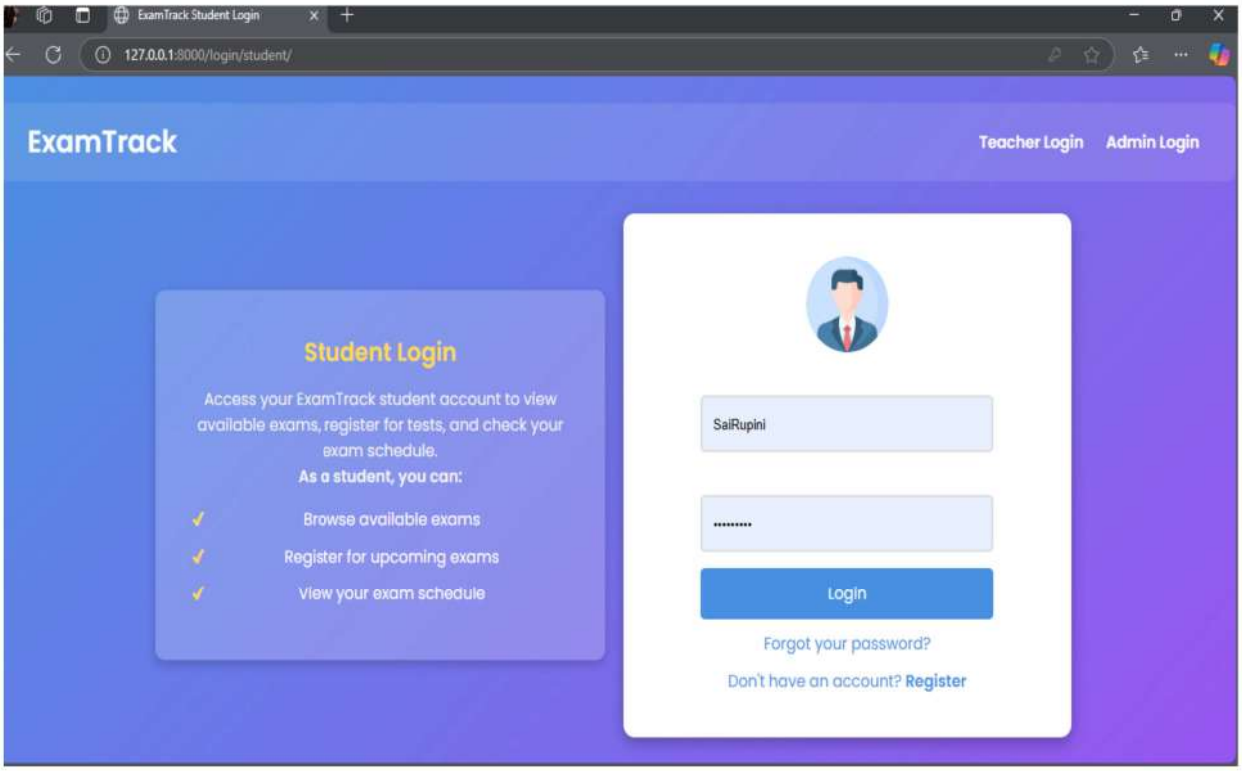
Purpose:

1. A dedicated login page for students.
2. Makes student login efficient and esay to use.

Key Features:

1. Username & Password Fields for student authentication.
2. Redirect to Student Dashboard upon successful login.

Output:



Student_Register.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Student Registration</title>

    <style>
        body {
            font-family: sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            background: linear-gradient(135deg, #E6F7FF, #F8F8FF);
        }

        header {
            width: 100%;
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 40px;
    box-sizing: border-box;
}

.header-logo {
    font-size: 1.5em;
    font-weight: bold;
    color: #4A5568;
}

.header-links {
    display: flex;
}

.header-links a {
    text-decoration: none;
    color: #4A5568;
    margin-left: 20px;
}

.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 80%;
    max-width: 1200px;
    margin-top: 50px;
}

.student-registration {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.student-registration h2 {
    color: #3182CE;
    margin-bottom: 20px;
}

.student-registration p {
```

```
    color: #4A5568;
    line-height: 1.6;
}

.student-registration ul {
    list-style: none;
    padding: 0;
    margin-top: 20px;
}

.student-registration li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
    color: #4A5568;
}

.student-registration li::before {
    content: '✓';
    position: absolute;
    left: 0;
    color: #3182CE;
}

.registration-form {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.registration-form h3 {
    color: #4A5568;
    margin-bottom: 10px;
}

.registration-form p {
    color: #718096;
    font-size: 0.9em;
    margin-bottom: 20px;
}

.registration-form label {
    display: block;
    font-weight: bold;
```

```
    color: #4A5568;
    margin-bottom: 5px;
}

.registration-form input {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #CBD5E0;
    border-radius: 5px;
    font-size: 1em;
}

.registration-form input[type="submit"] {
    width: 100%;
    padding: 12px;
    background-color: #3182CE;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.registration-form input[type="submit"]:hover {
    background-color: #2C5282;
}

.message-container {
    margin-bottom: 15px;
}

.alert {
    padding: 10px;
    border-radius: 5px;
    text-align: center;
}

.alert-success {
    background-color: #D4EDDA;
    color: #155724;
}

.alert-error {
    background-color: #F8D7DA;
    color: #721C24;
```

```
        }
    </style>
</head>
<body>
    <header>
        <div class="header-logo">ExamTrack</div>
        <div class="header-links">
            <a href="{% url 'teacher_register' %}">Teacher Portal</a>
            <a href="{% url 'admin_register' %}">Admin Portal</a>
        </div>
    </header>

    <div class="main-content">
        <div class="student-registration">
            <h2>Student Registration</h2>
                <p>Join ExamTrack to register for exams, track your schedule, and receive important notifications for upcoming tests.</p>
            <p><strong>As a student, you can:</strong></p>
            <ul>
                <li>Register for available exams</li>
                <li>Track your exam schedule</li>
                <li>View exam details and locations</li>
            </ul>
        </div>

        <div class="registration-form">
            <h3>Create Student Account</h3>
            <p>Fill in the details below to register</p>

            <!-- ☑ Display Django Messages -->
            <div class="message-container">
                {% if messages %}
                    {% for message in messages %}
                        <div class="alert alert-{{ message.tags }}">{{ message }}</div>
                    {% endfor %}
                {% endif %}
            </div>

            <!-- ☑ Django Form -->
            <form method="POST" action="{% url 'student_register' %}">
                {% csrf_token %}

                {{ form.non_field_errors }}  <!-- Displays form validation errors -->

                <label for="username">Username:</label>
                        <input type="text" name="username" id="username" required
```

```
value="{{ form.username.value|default_if_none:'' }}">

            <label for="email">Email:</label>
                        <input type="email" name="email" id="email" required
value="{{ form.email.value|default_if_none:'' }}">

            <label for="password1">Password:</label>
            <input type="password" name="password1" id="password1" required>

            <label for="password2">Confirm Password:</label>
            <input type="password" name="password2" id="password2" required>

            <input type="submit" value="Register">
        </form>


    </div>
</div>
<div class="message-container">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">{{ message }}</div>
        {% endfor %}
    {% endif %}
</div>

<!-- ☑ Display form validation errors -->
{% if form.errors %}
    <div class="alert alert-danger">
        <ul>
            {% for field, errors in form.errors.items %}
                {% for error in errors %}
                    <li>{{ field|capfirst }}: {{ error }}</li>
                {% endfor %}
            {% endfor %}
        </ul>
    </div>
{% endif %}
</body>
</html>
```

Description:

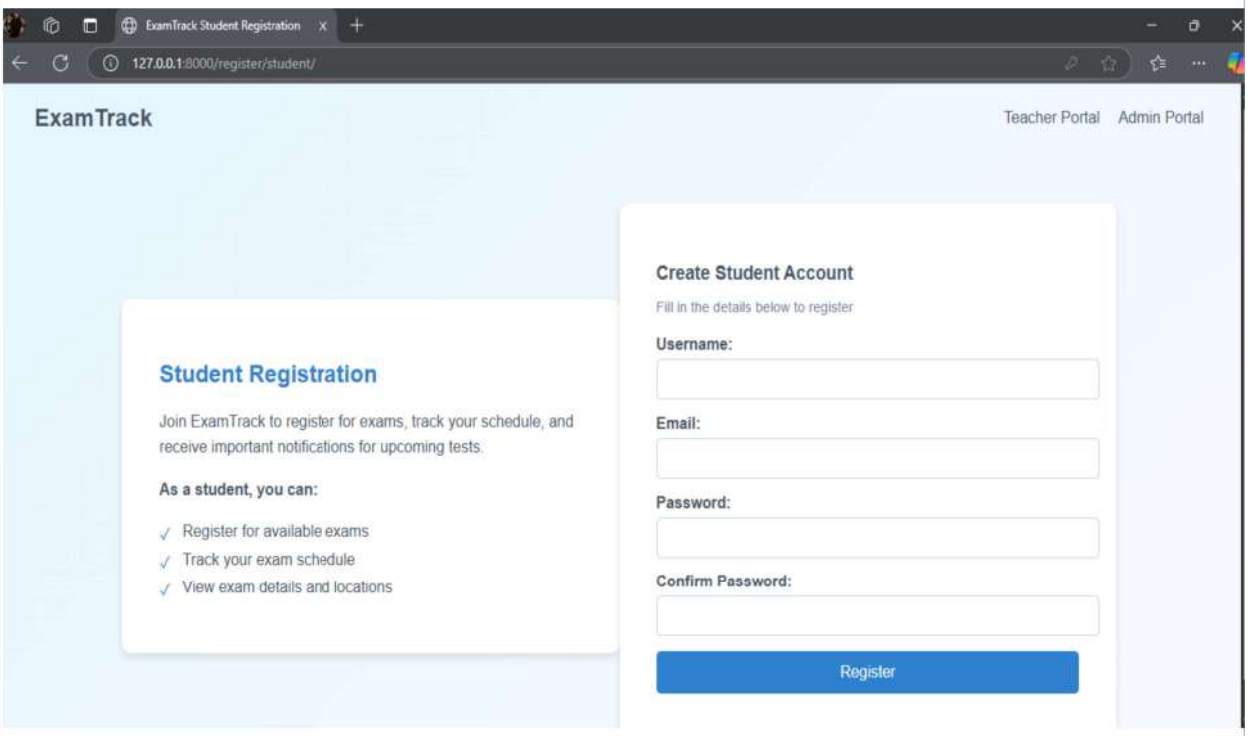
Purpose:

Allows students to register and create an account for exam registration.

Form Fields:

1. Full Name
2. Email
3. Student ID
4. Course/Department
5. Password & Confirm Password
6. Submit Button: Stores student details in the database.
7. Success Message: Displays confirmation upon successful registration.

Output:



Student_Dashboard.html:

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <title>Student Dashboard</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        body {
            margin: 0;
            font-family: 'Segoe UI', sans-serif;
            background-color: #f4f4f4;
        }

        .sidebar {
            height: 100vh;
            width: 230px;
            position: fixed;
            background-color: #2c3e50;
            padding-top: 20px;
        }

        .sidebar a {
            padding: 12px 20px;
            display: block;
            color: #ecf0f1;
            text-decoration: none;
        }

        .sidebar a:hover {
            background-color: #34495e;
        }

        .main {
            margin-left: 240px;
            padding: 30px;
        }

        .card {
            background-color: white;
            border-radius: 8px;
            padding: 20px;
            box-shadow: 0px 4px 10px rgba(0,0,0,0.05);
            margin-bottom: 20px;
        }
```

```
        .card h5 {
            font-size: 18px;
            color: #555;
        }

        .card p {
            font-size: 28px;
            margin: 10px 0 0;
            font-weight: bold;
            color: #2c3e50;
        }

        .welcome {
            font-size: 24px;
            margin-bottom: 25px;
            color: #2c3e50;
        }

        .motivation {
            background-color: #e0ffe0;
            border-left: 6px solid #28a745;
            padding: 15px;
            font-size: 16px;
            margin-top: 25px;
            border-radius: 5px;
            color: #2c662d;
        }

        .chart-card {
            background-color: #ffffff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 10px rgba(0,0,0,0.05);
            margin-top: 20px;
            max-width: 500px;
        }
    </style>
</head>
<body>

<!-- Sidebar -->
<div class="sidebar">
    <h2 style="color:white; text-align:center;">Student</h2>
    <a href="{% url 'student_dashboard' %}">🏠 Dashboard</a>
    <a href="{% url 'student_view_exams' %}">📝 View Exams</a>
    <a href="{% url 'student_view_results' %}">📊 View Results</a>
```

```
    <a href="{% url 'view_materials' %}">📚 View Materials</a>
    <a href="{% url 'edit_student_profile' %}">⚙️ Edit Profile</a>
    <a href="{% url 'logout' %}">📄 Logout</a>
</div>

<!-- Main Content -->
<div class="main">
    <div class="welcome">Hello, {{ user.first_name|default:user.username }} 👋</div>

    <div class="card">
        <h5>📚 Study Materials Available</h5>
        <p>{{ total_materials }}</p>
    </div>

    <div class="card">
        <h5>☑️ Completed Exams</h5>
        <p>{{ completed_exams_count }}</p>
    </div>

    <div class="chart-card">
        <h5 style="margin-bottom: 20px;">📊 Your Progress</h5>
        <canvas id="progressChart"
                data-passed="{{ passed_exams_count }}"
                data-failed="{{ failed_exams_count }}"
                width="400" height="400"></canvas>
    </div>

    <div class="motivation">
        🌟 Keep going, {{ user.first_name|default:user.username }}! You're making great
progress. Stay consistent and aim high!
    </div>
</div>

<!-- Chart Script -->
<script>
    const passed = parseInt(document.getElementById("progressChart").getAttribute("data-
passed"));
     const failed = parseInt(document.getElementById("progressChart").getAttribute("data-
failed"));

    const ctx = document.getElementById("progressChart").getContext("2d");

    new Chart(ctx, {
        type: 'pie',
        data: {
            labels: ['Passed', 'Failed'],
```

```
            datasets: [{
                label: 'Exam Results',
                data: [passed, failed],
                backgroundColor: ['#2ecc71', '#e74c3c'],
                borderWidth: 1
            }]
        },
        options: {
            responsive: true,
            plugins: {
                legend: {
                    position: 'bottom',
                }
            }
        }
    });
</script>

</body>
</html>
```

Description:

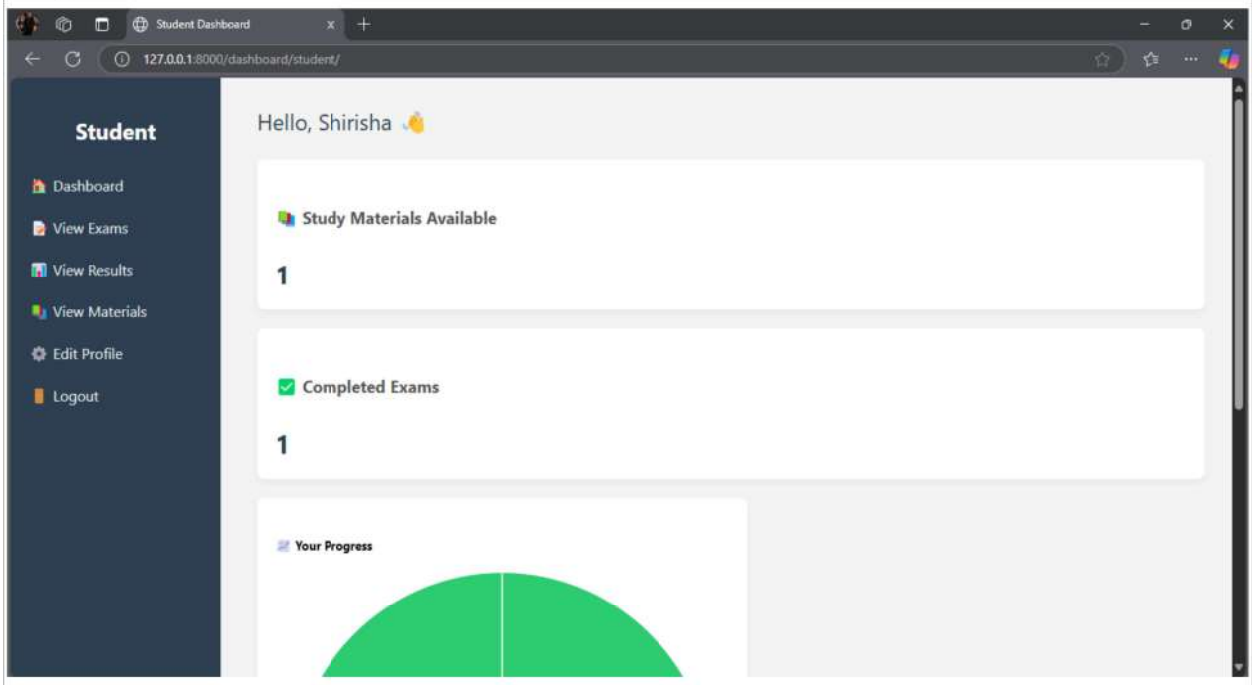
Purpose:

A personalized student portal where they can view and manage their exam registrations.

Key Features:

1. Welcome Message with student details.
2. Upcoming Exams Section: Displays registered exams with dates and times.
3. Exam Registration Button: Allows students to register for new exams.
4. Profile Settings: Students can update personal details.

Output:



Teacher_login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Teacher Login</title>
    <style>
    /* Import Google Font */
@import                                                    url('https://fonts.googleapis.com/css2?
family=Poppins:wght@300;400;600&display=swap');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

/* Background Styling */
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
```

```
    height: 100vh;
    background: linear-gradient(135deg, #3b5998, #8b9dc3);
    color: #fff;
}

/* Header */
header {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 40px;
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(5px);
    border-radius: 8px;
    margin-bottom: 30px;
}

.header-logo {
    font-size: 1.8em;
    font-weight: bold;
    color: white;
}

.header-links a {
    text-decoration: none;
    color: white;
    font-weight: bold;
    margin-left: 20px;
    transition: 0.3s;
}

.header-links a:hover {
    color: #ffdb4d;
}

/* Main Content */
.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 90%;
    max-width: 1000px;
    gap: 20px;
}
```

```
/* Teacher Info Section */
.teacher-login {
    flex: 1;
    padding: 40px;
    background: rgba(255, 255, 255, 0.2);
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
}

.teacher-login h2 {
    color: #ffdb4d;
    margin-bottom: 15px;
}

.teacher-login p {
    color: #f3f3f3;
    line-height: 1.5;
}

.teacher-login ul {
    list-style: none;
    padding: 0;
    margin-top: 15px;
}

.teacher-login li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
}

.teacher-login li::before {
    content: '✔';
    position: absolute;
    left: 0;
    color: #ffdb4d;
    font-weight: bold;
}

/* Login Form */
.login-form {
    flex: 1;
    padding: 40px;
    background: white;
    border-radius: 12px;
```

```
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
    display: flex;
    flex-direction: column;
    align-items: center;
}
.login-form input {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 2px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
    transition: all 0.3s ease;
}

.login-form input:focus {
    border-color: #4a90e2;
    outline: none;
}

/* Avatar */
.login-form .avatar {
    width: 90px;
    height: 90px;
     background:  url('https://cdn-icons-png.flaticon.com/512/1995/1995574.png')  no-repeat
center;
    background-size: cover;
    border-radius: 50%;
    margin-bottom: 15px;
}

/* Submit Button */
button {
    width: 100%;
    padding: 12px;
    background: #3b5998;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s;
}

button:hover {
    background: #2e4a7f;
```

```
}

/* Responsive Design */
@media (max-width: 768px) {
    .main-content {
        flex-direction: column;
        align-items: center;
    }

    .teacher-login, .login-form {
        width: 90%;
    }
}


</style>
</head>
<body>
    <header>
        <div class="header-logo">ExamTrack</div>
        <div class="header-links">
            <a href="{% url 'student_login' %}">Student Login</a>
            <a href="{% url 'admin_login' %}">Admin Login</a>
        </div>
    </header>

    <div class="main-content">
        <div class="teacher-login">
            <h2>Teacher Login</h2>
             <p>Access your ExamTrack teacher account to create and manage exams, track
student registrations, and set up exam schedules.</p>
            <p><strong>As a teacher, you can:</strong></p>
            <ul>
                <li>Create and schedule exams</li>
                <li>Track student registrations</li>
                <li>Manage exam resources</li>
            </ul>
        </div>

        <div class="login-form">
            <div class="avatar"></div>
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-{{ message.tags }}">
                        {{ message }}
                    </div>
```

```
            {% endfor %}
        {% endif %}

        <!-- Login Form -->
        <form method="POST" action="{% url 'teacher_login' %}">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit">Login</button>
        </form>


        <a href="#">Forgot your password?</a>
                <a href="{% url 'teacher_register' %}">Don't have an account?
<strong>Register</strong></a>
        </div>
    </div>

</body>
</html>
```

Description:

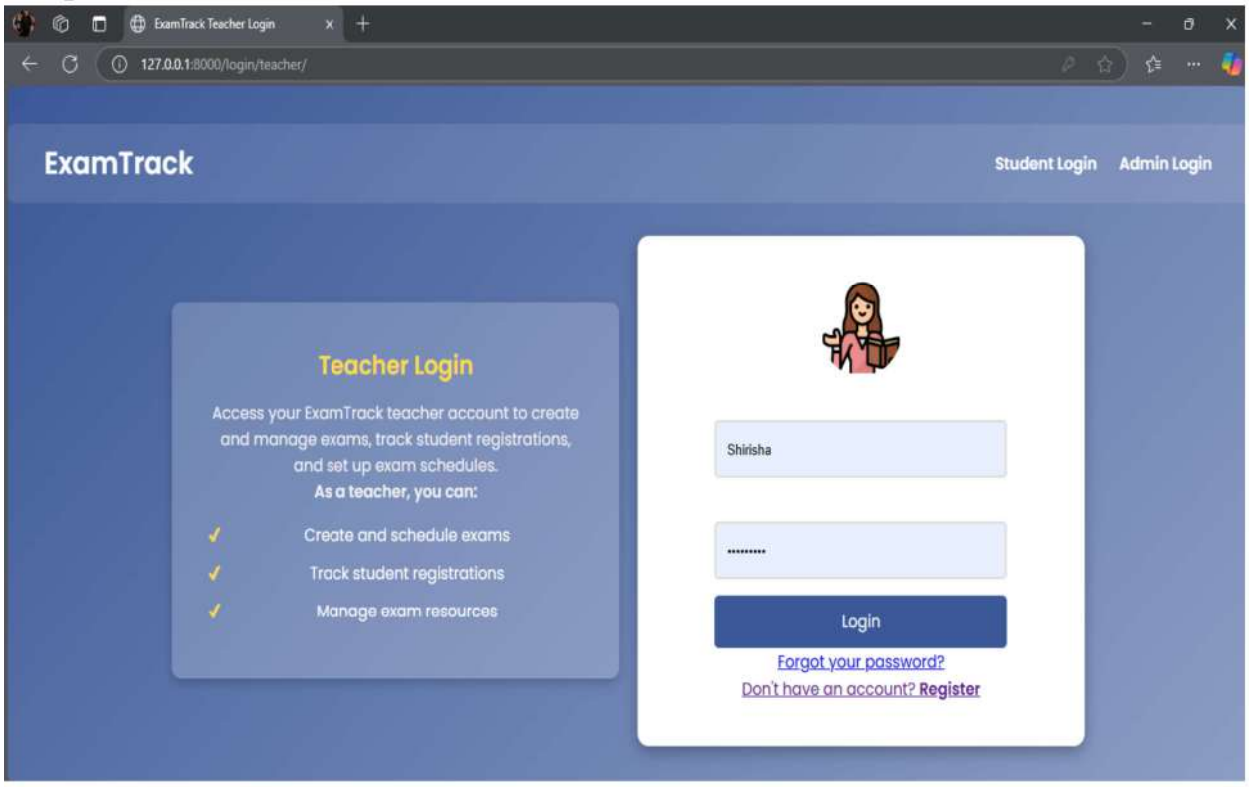
Purpose:

A dedicated login page for teachers to access their portal.

Key Features:

1. Username & Password Fields for teacher authentication.
2. Redirect to Teacher Dashboard upon successful login.
3. Option to Reset Password if needed.
4. Support Contact Information for assistance.

Output:



Teacher_register.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Teacher Registration</title>
    <style>
        body {
            font-family: sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            background: linear-gradient(135deg, #E6F7FF, #F8F8FF);
        }

        header {
            width: 100%;
            display: flex;
            justify-content: space-between;
            align-items: center;
```

```
    padding: 20px 40px;
    box-sizing: border-box;
}

.header-logo {
    font-size: 1.5em;
    font-weight: bold;
    color: #4A5568;
}

.header-links {
    display: flex;
}

.header-links a {
    text-decoration: none;
    color: #4A5568;
    margin-left: 20px;
}

.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 80%;
    max-width: 1200px;
    margin-top: 50px;
}

.teacher-registration {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.teacher-registration h2 {
    color: #3182CE;
    margin-bottom: 20px;
}

.teacher-registration p {
    color: #4A5568;
    line-height: 1.6;
}
```

```
.teacher-registration ul {
    list-style: none;
    padding: 0;
    margin-top: 20px;
}

.teacher-registration li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
    color: #4A5568;
}

.teacher-registration li::before {
    content: '✓';
    position: absolute;
    left: 0;
    color: #3182CE;
}

.registration-form {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
}

.registration-form h3 {
    color: #4A5568;
    margin-bottom: 10px;
}

.registration-form p {
    color: #718096;
    font-size: 0.9em;
    margin-bottom: 20px;
}

.registration-form label {
    color: #4A5568;
    margin-bottom: 5px;
}
```

```
        .registration-form input[type="text"],
        .registration-form input[type="email"],
        .registration-form input[type="password"] {
            width: calc(100% - 24px);
            padding: 12px;
            margin-bottom: 15px;
            border: 1px solid #CBD5E0;
            border-radius: 5px;
            box-sizing: border-box;
        }

        .registration-form .password-group {
            display: flex;
            justify-content: space-between;
        }

        .registration-form .password-group input[type="password"] {
            width: calc(50% - 8px);
        }

        .registration-form input[type="submit"] {
            width: 100%;
            padding: 12px;
            background-color: #3182CE;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }

        .registration-form input[type="submit"]:hover {
            background-color: #2C5282;
        }

        .registration-form .note {
            color: #718096;
            font-size: 0.8em;
            margin-top: 5px;
        }
    </style>
</head>
<body>
    <header>
        <div class="header-logo">ExamTrack</div>
```

```
        <div class="header-links">
            <a href="{% url 'student_register' %}">Student Portal</a>
            <a href="{% url 'admin_register' %}">Admin Portal</a>
        </div>
    </header>

    <div class="main-content">
        <div class="teacher-registration">
            <h2>Teacher Registration</h2>
              <p>Join ExamTrack as a teacher to create and manage exams, track student registrations, and organize your exam schedule.</p>
            <p><strong>As a teacher, you can:</strong></p>
            <ul>
                <li>Create and schedule exams</li>
                <li>Track student exam registrations</li>
                <li>Manage exam resources and materials</li>
            </ul>
        </div>

        <div class="registration-form">
            <h3>Create Teacher Account</h3>
            <p>Fill in the details below to register</p>
            <div class="message-container">
                {% if messages %}
                    {% for message in messages %}
                        <div class="alert alert-{{ message.tags }}">{{ message }}</div>
                    {% endfor %}
                {% endif %}
            </div>

            <!-- ☑ Django Form -->
            <form method="POST" action="{% url 'teacher_register' %}">
                {% csrf_token %}

                {{ form.non_field_errors }}  <!-- Displays form validation errors -->

                <label for="username">Username:</label>
                        <input type="text" name="username" id="username" required value="{{ form.username.value|default_if_none:'' }}">

                <label for="email">Email:</label>
                            <input type="email" name="email" id="email" required value="{{ form.email.value|default_if_none:'' }}">

                <label for="password1">Password:</label>
                <input type="password" name="password1" id="password1" required>
```

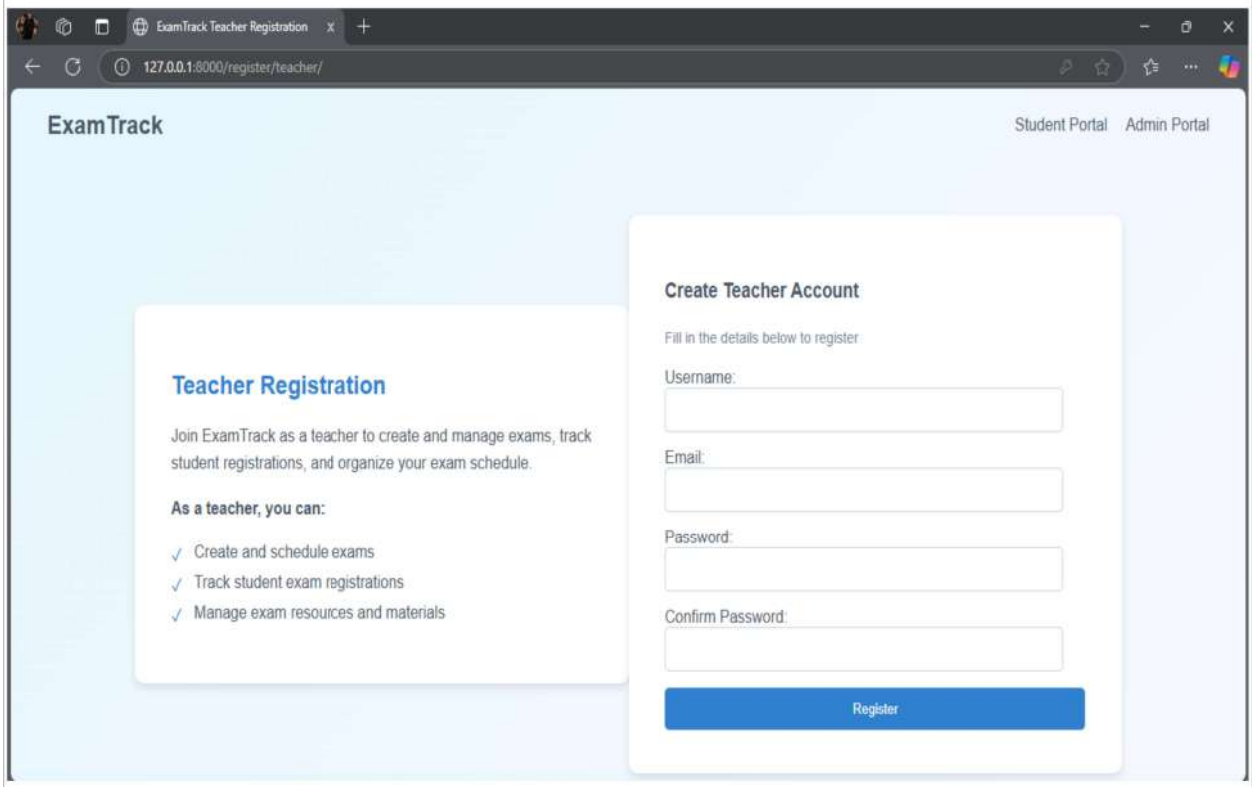
```
            <label for="password2">Confirm Password:</label>
            <input type="password" name="password2" id="password2" required>

            <input type="submit" value="Register">
        </form>


    </div>
</div>
<div class="message-container">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">{{ message }}</div>
        {% endfor %}
    {% endif %}
</div>

<!-- ☑ Display form validation errors -->
{% if form.errors %}
    <div class="alert alert-danger">
        <ul>
            {% for field, errors in form.errors.items %}
                {% for error in errors %}
                    <li>{{ field|capfirst }}: {{ error }}</li>
                {% endfor %}
            {% endfor %}
        </ul>
    </div>
{% endif %}
    </div>
</div>
</body>
</html>
```

Description:

Purpose:

1. Allows teachers to sign up and manage exams.
2. Form Fields:
    1. Full Name
    2. Email
    3. Password & Confirm Password

Output:



Teacher_dashboard.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Teacher Dashboard</title>
        <link   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            display: flex;
            height: 100vh;
            margin: 0;
```

```
        }
        .sidebar {
            width: 250px;
            background-color: #623596;
            padding: 20px;
            color: white;
        }
        .sidebar h3 {
            font-size: 24px;
            margin-bottom: 30px;
        }
        .sidebar a {
            color: white;
            display: block;
            padding: 10px 0;
            text-decoration: none;
        }
        .sidebar a:hover {
            background-color: #145c43;
            padding-left: 10px;
        }
        .main-content {
            flex-grow: 1;
            padding: 30px;
            background-color: #f8f9fa;
        }
        .card {
            border-radius: 12px;
        }
    </style>
</head>
<body>
    <div class="sidebar">
        <h3>Teacher Panel</h3>
        <a href="{% url 'create_exam' %}"> Create Exam</a>
        <a href="{% url 'view_students_results' %}"> View Results</a>
        <a href="{% url 'upload_material' %}"> Upload Materials</a>
        <a href="{% url 'logout' %}"> Logout</a>
    </div>

    <div class="main-content">
        <h2>Welcome, {{ user.username }}</h2>

        <div class="row mt-4">
            <div class="col-md-6">
                <div class="card text-white bg-primary">
```

```
                <div class="card-body">
                    <h5 class="card-title">📚 Exams Created</h5>
                    <p class="card-text display-6">{{ exam_count }}</p>
                </div>
            </div>
        </div>
        <div class="col-md-6">
            <div class="card text-white bg-success">
                <div class="card-body">
                    <h5 class="card-title">👨‍🎓 Total Students</h5>
                    <p class="card-text display-6">{{ student_count }}</p>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Description:

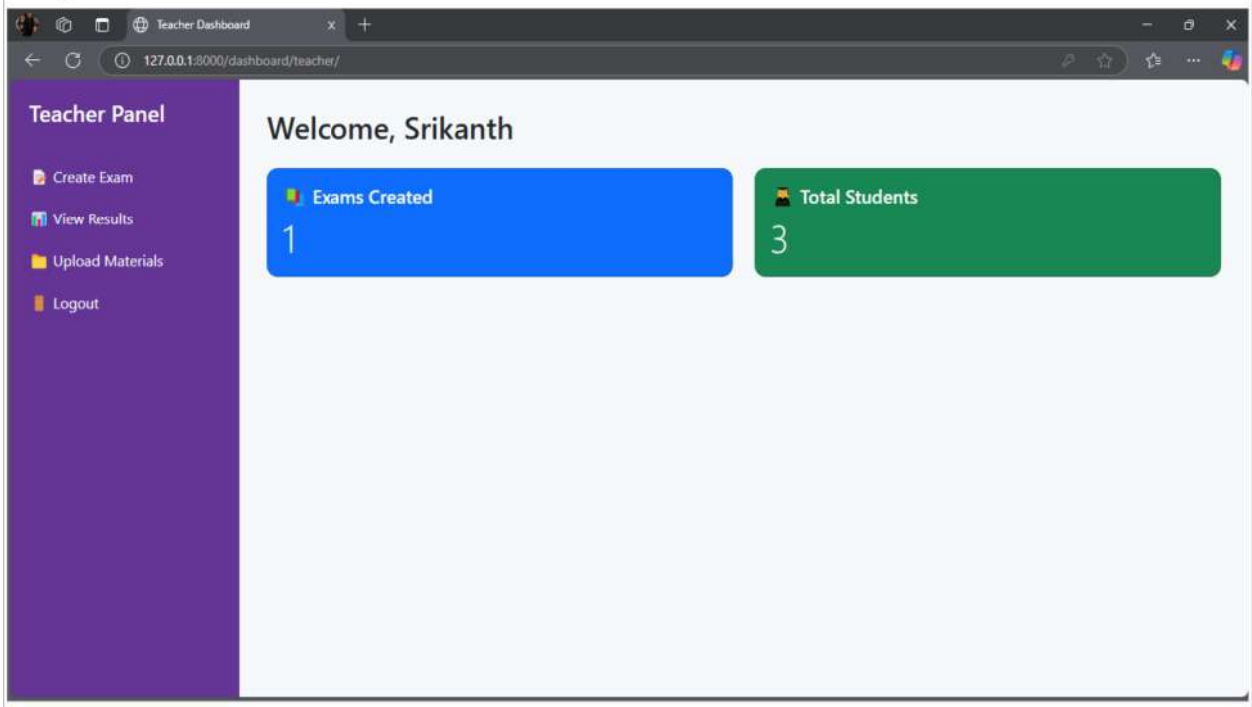
Purpose:

- A dedicated teacher portal for managing exams and students.

Key Features:

- Create & Manage Exams: Teachers can add/edit exam details.
- Student Exam List: View registered students for each exam.
- Grading & Feedback System: Assign grades and provide feedback.

Output:



Admin_login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Administrator Login</title>
    <style>
     /* Import Google Font */
@import                                                    url('https://fonts.googleapis.com/css2?
family=Poppins:wght@300;400;600&display=swap');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

/* Background Styling */
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background: linear-gradient(135deg, #7f44a1, #a349b9);
    color: #fff;
}

/* Header */
header {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 40px;
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(5px);
    border-radius: 8px;
    margin-bottom: 30px;
}

.header-logo {
```

```
    font-size: 1.8em;
    font-weight: bold;
    color: white;
}

.header-links a {
    text-decoration: none;
    color: white;
    font-weight: bold;
    margin-left: 20px;
    transition: 0.3s;
}

.header-links a:hover {
    color: #ffdb4d;
}

/* Main Content */
.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 90%;
    max-width: 1000px;
    gap: 20px;
}

/* Admin Info Section */
.admin-login {
    flex: 1;
    padding: 40px;
    background: rgba(255, 255, 255, 0.2);
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
}

.admin-login h2 {
    color: #ffdb4d;
    margin-bottom: 15px;
}

.admin-login p {
    color: #f3f3f3;
    line-height: 1.5;
}
```

```
.admin-login ul {
    list-style: none;
    padding: 0;
    margin-top: 15px;
}

.admin-login li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
}

.admin-login li::before {
    content: '✔';
    position: absolute;
    left: 0;
    color: #ffdb4d;
    font-weight: bold;
}

/* Login Form */
.login-form {
    flex: 1;
    padding: 40px;
    background: white;
    border-radius: 12px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
    display: flex;
    flex-direction: column;
    align-items: center;
}
.login-form input {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 2px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
    transition: all 0.3s ease;
}

.login-form input:focus {
    border-color: #4a90e2;
    outline: none;
}
```

```
/* Avatar */
.login-form .avatar {
    width: 90px;
    height: 90px;
     background:  url('https://cdn-icons-png.flaticon.com/512/3135/3135715.png')  no-repeat
center;
    background-size: cover;
    border-radius: 50%;
    margin-bottom: 15px;
}

/* Submit Button */
button {
    width: 100%;
    padding: 12px;
    background: #7f44a1;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s;
}

button:hover {
    background: #7f44a1;
}

/* Responsive Design */
@media (max-width: 768px) {
    .main-content {
        flex-direction: column;
        align-items: center;
    }

    .admin-login, .login-form {
        width: 90%;
    }
}


    </style>
</head>
<body>
    <header>
```

```
        <div class="header-logo">ExamTrack</div>
        <div class="header-links">
            <a href="{% url 'student_login' %}">Student Login</a>
            <a href="{% url 'teacher_login' %}">Teacher Login</a>
        </div>
    </header>

    <div class="main-content">
        <div class="admin-login">
            <h2>Administrator Login</h2>
            <p>Access your ExamTrack admin dashboard to oversee all exams, manage users,
and control system settings.</p>
            <p><strong>As an administrator, you can:</strong></p>
            <ul>
                <li>Manage users and permissions</li>
                <li>Oversee all exams and registrations</li>
                <li>Configure system settings</li>
            </ul>
        </div>

        <div class="login-form">
            <div class="avatar admin-avatar"></div>
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-{{ message.tags }}">
                        {{ message }}
                    </div>
                {% endfor %}
            {% endif %}

            <!-- Login Form -->
            <form method="POST" action="{% url 'admin_login' %}">
                {% csrf_token %}
                {{ form.as_p }}
                <button type="submit">Login</button>
            </form>


            <a href="#">Forgot your password?</a>
                    <a href="{% url 'admin_register' %}">Don't have an account?
<strong>Register</strong></a>
        </div>
    </div>

</body>
</html>
```

Description:

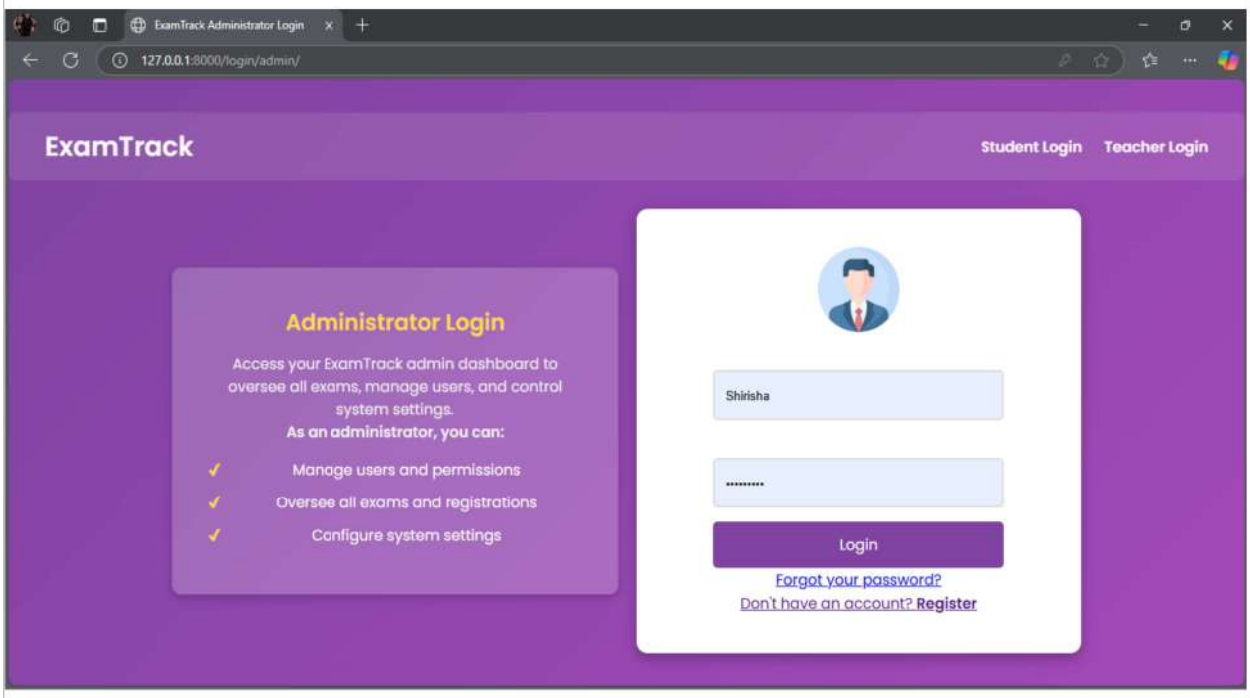
Purpose:

- A dedicated login page for administrators to manage the system.

Key Features:

- Admin Credentials Input Fields for authentication.
- Redirect to Admin Dashboard upon successful login.
- Security Features to prevent unauthorized access.

Output:



Admin_register.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ExamTrack Administrator Registration</title>
    <style>
        body {
            font-family: sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
```

```
    align-items: center;
    background: linear-gradient(135deg, #E6F7FF, #F8F8FF);
}

header {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 40px;
    box-sizing: border-box;
}

.header-logo {
    font-size: 1.5em;
    font-weight: bold;
    color: #4A5568;
}

.header-links {
    display: flex;
}

.header-links a {
    text-decoration: none;
    color: #4A5568;
    margin-left: 20px;
}

.main-content {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 80%;
    max-width: 1200px;
    margin-top: 50px;
}

.admin-registration {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

```
.admin-registration h2 {
    color: #3182CE;
    margin-bottom: 20px;
}

.admin-registration p {
    color: #4A5568;
    line-height: 1.6;
}

.admin-registration ul {
    list-style: none;
    padding: 0;
    margin-top: 20px;
}

.admin-registration li {
    position: relative;
    padding-left: 25px;
    margin-bottom: 10px;
    color: #4A5568;
}

.admin-registration li::before {
    content: '✓';
    position: absolute;
    left: 0;
    color: #3182CE;
}

.registration-form {
    flex: 1;
    padding: 40px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
}

.registration-form h3 {
    color: #4A5568;
    margin-bottom: 10px;
}

.registration-form p {
```

```
    color: #718096;
    font-size: 0.9em;
    margin-bottom: 20px;
}

.registration-form label {
    color: #4A5568;
    margin-bottom: 5px;
}

.registration-form input[type="text"],
.registration-form input[type="email"],
.registration-form input[type="password"] {
    width: calc(100% - 24px);
    padding: 12px;
    margin-bottom: 15px;
    border: 1px solid #CBD5E0;
    border-radius: 5px;
    box-sizing: border-box;
}

.registration-form .password-group {
    display: flex;
    justify-content: space-between;
}

.registration-form .password-group input[type="password"] {
    width: calc(50% - 8px);
}

.registration-form input[type="submit"] {
    width: 100%;
    padding: 12px;
    background-color: #3182CE;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.registration-form input[type="submit"]:hover {
    background-color: #2C5282;
}

.registration-form .note {
```

```
            color: #718096;
            font-size: 0.8em;
            margin-top: 5px;
        }
    </style>
</head>
<body>
    <header>
        <div class="header-logo">ExamTrack</div>
        <div class="header-links">
            <a href="{% url 'student_register' %}">Student Portal</a>
            <a href="{% url 'teacher_register' %}">Teacher Portal</a>
        </div>
    </header>

    <div class="main-content">
        <div class="admin-registration">
            <h2>Administrator Registration</h2>
             <p>Register as an ExamTrack administrator to manage the entire exam system, users, and platform settings.</p>
            <p><strong>As an administrator, you can:</strong></p>
            <ul>
                <li>Manage all users and permissions</li>
                <li>Configure system settings</li>
            </ul>
        </div>

        <div class="registration-form">
            <h3>Create Administrator Account</h3>
            <div class="message-container">
                {% if messages %}
                    {% for message in messages %}
                        <div class="alert alert-{{ message.tags }}">{{ message }}</div>
                    {% endfor %}
                {% endif %}
            </div>

            <!-- ☑ Django Form -->
            <form method="POST" action="{% url 'admin_register' %}">
                {% csrf_token %}

                {{ form.non_field_errors }}  <!-- Displays form validation errors -->

                <label for="username">Username:</label>
                        <input type="text" name="username" id="username" required value="{{ form.username.value|default_if_none:'' }}">
```

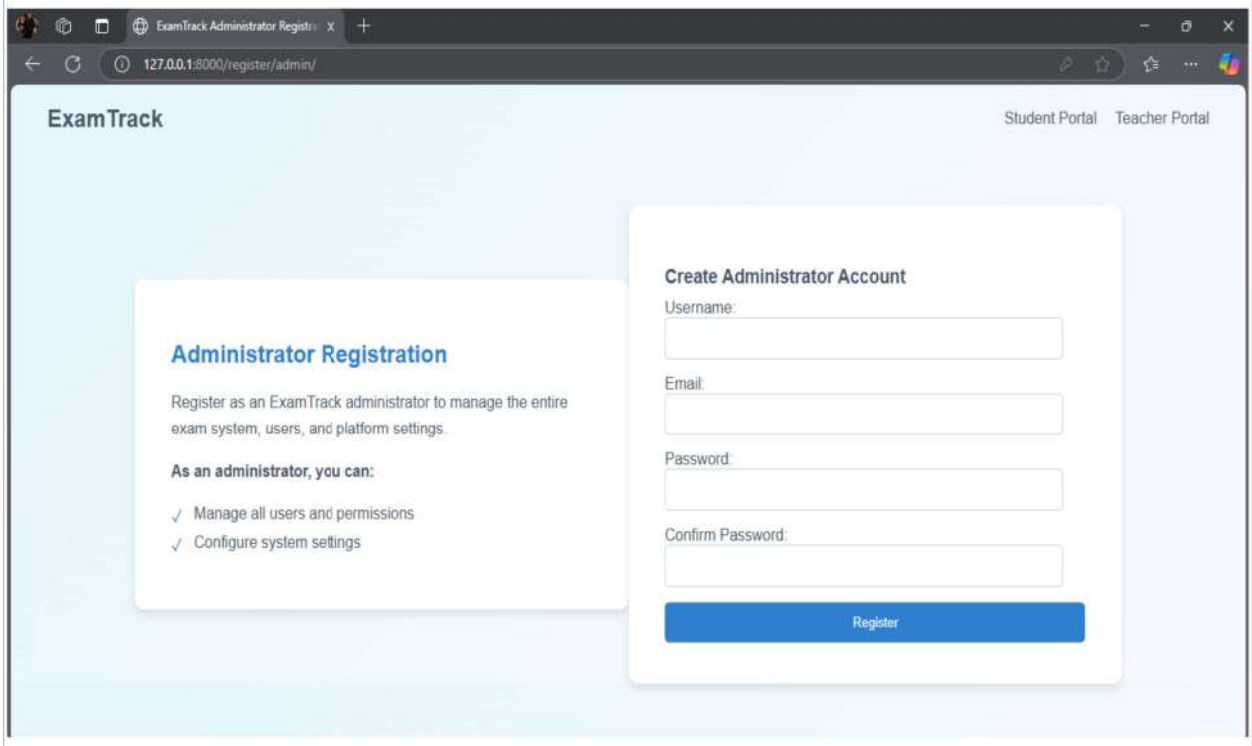
```
            <label for="email">Email:</label>
                        <input type="email" name="email" id="email" required
value="{{ form.email.value|default_if_none:'' }}">

            <label for="password1">Password:</label>
            <input type="password" name="password1" id="password1" required>

            <label for="password2">Confirm Password:</label>
            <input type="password" name="password2" id="password2" required>

            <input type="submit" value="Register">
        </form>

    </div>
</div>
<div class="message-container">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">{{ message }}</div>
        {% endfor %}
    {% endif %}
</div>

<!-- ☑ Display form validation errors -->
{% if form.errors %}
    <div class="alert alert-danger">
        <ul>
            {% for field, errors in form.errors.items %}
                {% for error in errors %}
                    <li>{{ field|capfirst }}: {{ error }}</li>
                {% endfor %}
            {% endfor %}
        </ul>
    </div>
{% endif %}
    </div>
</div>
</body>
</html>
```

Description:

Purpose:

1. Allows system administrators to create accounts for managing the platform.
2. Key Features:
3. Form Fields:
    1. Full Name
    2. Email
    3. Password & Confirm Password
4. Security Measures: Admins need a special access code to register.

Output:



Admin_Dashboard.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Admin Dashboard - ExamTrack</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    body {
      display: flex;
      background-color: #f4f6f9;
      min-height: 100vh;
    }

    /* Sidebar */
    .sidebar {
      width: 220px;
      background-color: #2c3e50;
      color: white;
      height: 100vh;
      padding: 20px 0;
      position: fixed;
    }

    .sidebar h2 {
      text-align: center;
      margin-bottom: 30px;
      font-size: 24px;
    }

    .sidebar a {
      display: block;
      color: white;
      text-decoration: none;
      padding: 12px 20px;
      transition: background 0.3s;
    }
```

```
.sidebar a:hover {
  background-color: #1abc9c;
}

/* Main content */
.main {
  margin-left: 220px;
  padding: 30px;
  flex: 1;
}

.dashboard-title {
  font-size: 28px;
  margin-bottom: 30px;
  color: #333;
}

.card-container {
  display: flex;
  gap: 20px;
  flex-wrap: wrap;
}

.card {
  flex: 1;
  min-width: 250px;
  background-color: #3498db;
  color: white;
  padding: 25px;
  border-radius: 12px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.1);
  transition: transform 0.2s ease;
}

.card.bg-success {
  background-color: #27ae60;
}

.card:hover {
  transform: translateY(-5px);
}

.card h5 {
  font-size: 18px;
  margin-bottom: 10px;
}
```

```
.card h2 {
  font-size: 36px;
  font-weight: bold;
}

.actions {
  margin-top: 40px;
}

.actions a {
  display: inline-block;
  text-decoration: none;
  background-color: #2c3e50;
  color: white;
  padding: 12px 20px;
  margin-right: 20px;
  border-radius: 8px;
  transition: background 0.3s;
}

.actions a:hover {
  background-color: #34495e;
}

@media (max-width: 768px) {
  body {
    flex-direction: column;
  }

  .sidebar {
    width: 100%;
    height: auto;
    position: relative;
  }

  .main {
    margin-left: 0;
    padding: 20px;
  }

  .card-container {
    flex-direction: column;
  }

  .actions a {
```

```
        display: block;
        margin: 10px 0;
      }
    }
  </style>
</head>
<body>

  <div class="sidebar">
    <h2>Admin Panel</h2>
    <a href="#">🏠Dashboard</a>
    <a href="{% url 'manage_students' %}">👩‍🎓Manage Students</a>
    <a href="{% url 'manage_teachers' %}">👨‍🏫Manage Teachers</a>
    <a href="{% url 'logout' %}">📄Logout</a>
  </div>

  <div class="main">
    <h2 class="dashboard-title">Welcome to Admin Dashboard</h2>

    <div class="card-container">
      <div class="card">
        <h5>Total Students</h5>
        <h2>{{ student_count }}</h2>
      </div>
      <div class="card bg-success">
        <h5>Total Teachers</h5>
        <h2>{{ teacher_count }}</h2>
      </div>
    </div>

    <div class="actions">
      <a href="{% url 'manage_students' %}">Manage Students</a>
      <a href="{% url 'manage_teachers' %}">Manage Teachers</a>
    </div>
  </div>

</body>
</html>
```

Description:

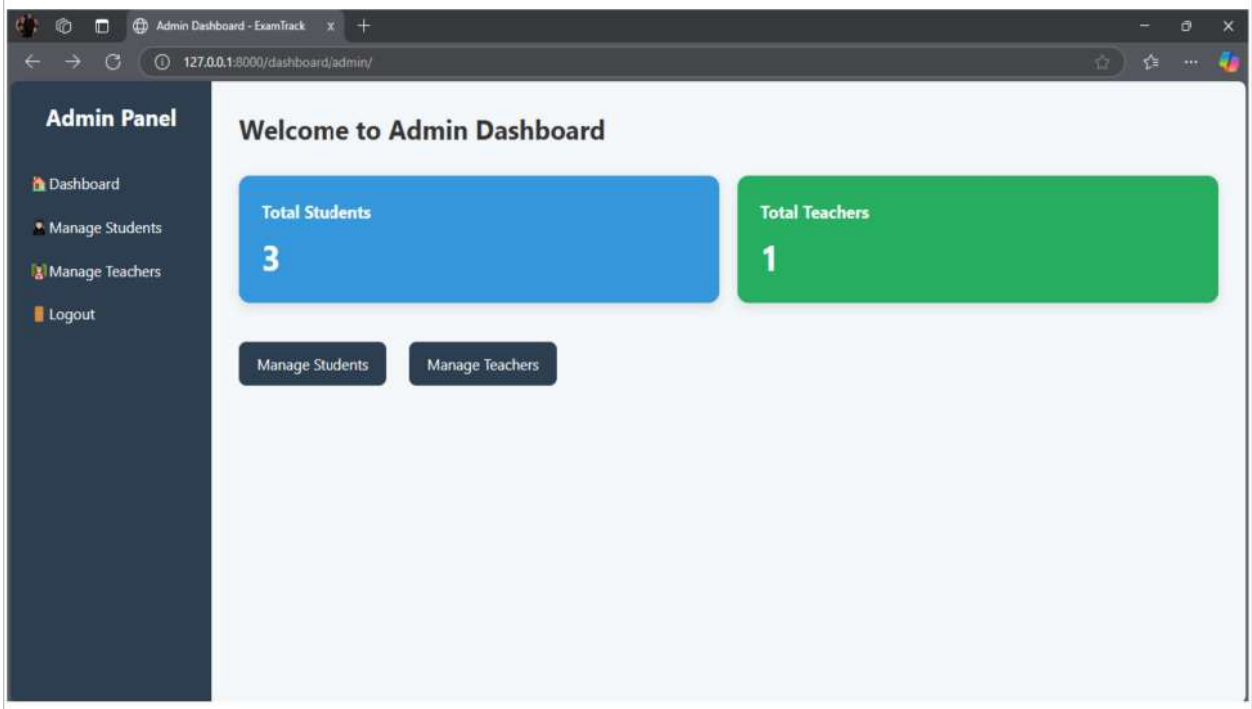
Purpose:

The control panel for administrators to manage users, exams, and overall system settings.

Key Features:

1. User Management: View, approve, and manage student & teacher accounts.
2. Exam Schedule Management: Modify exam dates and times.
3. System Logs & Reports: Track user activities and generate reports.
4. Security Features: Manage platform security and user authentication.

Output:



Create_exam.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create Exam</title>
        <link    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            display: flex;
            margin: 0;
            font-family: 'Segoe UI', sans-serif;
        }
        .sidebar {
            width: 250px;
            background-color: #3f236b;
            color: white;
            padding: 30px 20px;
            min-height: 100vh;
        }
        .sidebar h3 {
            font-size: 26px;
            margin-bottom: 40px;
            font-weight: bold;
            text-align: center;
        }
        .sidebar a {
            color: white;
            display: block;
            padding: 12px 10px;
            text-decoration: none;
            border-radius: 6px;
            margin-bottom: 10px;
            transition: background 0.3s;
        }
        .sidebar a:hover {
            background-color: #5b349f;
        }
        .main-content {
            flex-grow: 1;
            padding: 40px;
            background-color: #f8f9fa;
        }
    </style>
```

```
</head>
<body>
<div class="sidebar">
    <h3>Teacher Panel</h3>
    <a href="{% url 'teacher_dashboard' %}"> Dashboard</a>
    <a href="{% url 'create_exam' %}"> Create Exam</a>
    <a href="{% url 'view_students_results' %}"> View Results</a>
    <a href="{% url 'upload_material' %}"> Upload Materials</a>
    <a href="{% url 'logout' %}"> Logout</a>
</div>
<div class="main-content">
    <h2 class="mb-4"> Create New Exam</h2>
    <form method="post" action="{% url 'create_exam' %}">
        {% csrf_token %}
        <div class="mb-3">
            <label class="form-label fw-bold">Exam Title</label>
            <input type="text" name="title" class="form-control" required>
        </div>
        <div class="mb-3">
            <label class="form-label fw-bold">Description</label>
            <textarea name="description" class="form-control" rows="3"></textarea>
        </div>
        <input type="hidden" id="question_count" name="question_count" value="0">
        <div id="questions-container"></div>
                        <button type="button" class="btn btn-secondary my-3"
onclick="addQuestion()"> Add Question</button>
        <button type="submit" class="btn btn-primary"> Create Exam</button>
    </form>
</div>
<script>
    let questionCount = 0;
    function addQuestion() {
        const container = document.getElementById('questions-container');
        const countInput = document.getElementById('question_count');
        const questionHTML = `
        <div class="card p-3 my-3 shadow-sm">
            <h5>Question ${questionCount + 1}</h5>
                <input type="text" name="question_${questionCount}" class="form-control
my-2" placeholder="Question Text" required>
                <input type="text" name="option_a_${questionCount}" class="form-control
my-1" placeholder="Option A" required>
                <input type="text" name="option_b_${questionCount}" class="form-control
my-1" placeholder="Option B" required>
                <input type="text" name="option_c_${questionCount}" class="form-control
my-1" placeholder="Option C" required>
                <input type="text" name="option_d_${questionCount}" class="form-control
```

```
my-1" placeholder="Option D" required>
            <input type="text" name="correct_option_${questionCount}" class="form-control
my-1" placeholder="Correct Option (A, B, C or D)" required pattern="[ABCDabcd]">
        </div>
        `;
        container.insertAdjacentHTML('beforeend', questionHTML);
        questionCount++;
        countInput.value = questionCount;
    }
</script>
</body>
</html>
```

Description:

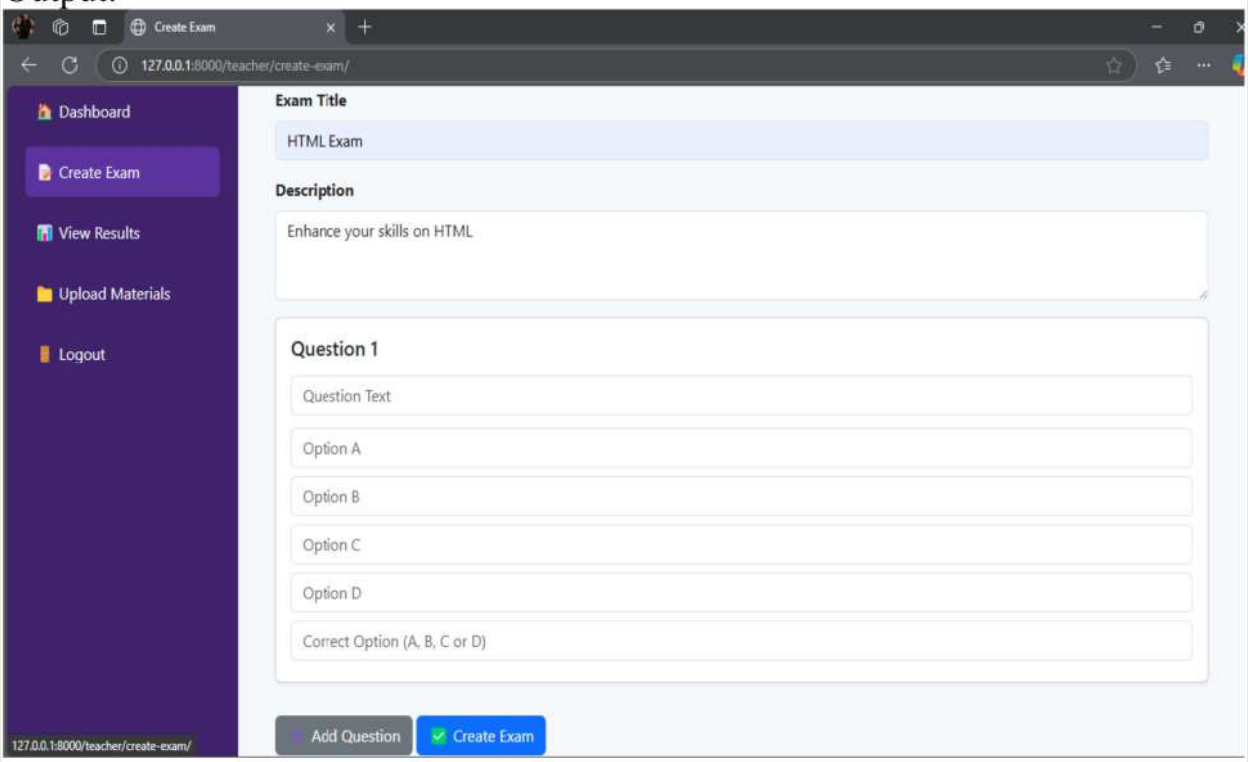
Purpose:

This page allows teachers to create a new exam with a title, description, and multiple-choice questions.

Key Features:

1. Dynamic question addition with options and correct answer.
2. Sidebar for easy navigation.
3. Clean, responsive Bootstrap layout

Output:



view_students.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Results</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
    }
    .sidebar {
      width: 230px;
      height: 100vh;
      background-color: #343a40;
      color: white;
      position: fixed;
      padding-top: 20px;
    }
    .sidebar a {
      display: block;
      color: white;
      padding: 12px 20px;
      text-decoration: none;
    }
    .sidebar a:hover {
      background-color: #495057;
    }
    .main {
      margin-left: 240px;
      padding: 20px;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    th, td {
      padding: 12px;
      text-align: left;
      border: 1px solid #dee2e6;
    }
    th {
      background-color: #28a745;
      color: white;
    }
```

```
  </style>
</head>
<body>
  <div class="sidebar">
    <h2 style="text-align:center;">Student</h2>
    <a href="{% url 'student_dashboard' %}"> Dashboard</a>
    <a href="{% url 'student_view_exams' %}"> View Exams</a>
    <a href="{% url 'student_view_results' %}"> View Results</a>
    <a href="{% url 'view_materials' %}"> View Materials</a>
    <a href="{% url 'edit_student_profile' %}"> Edit Profile</a>
  </div>
  <div class="main">
    <h2>My Results</h2>
    <table>
      <tr><th>Exam</th><th>Score</th><th>Date</th></tr>
      {% for result in results %}
      <tr>
        <td>{{ result.exam.title }}</td>
        <td>{{ result.score }}%</td>
        <td>{{ result.date_taken|date:"Y-m-d H:i" }}</td>
      </tr>
      {% endfor %}
    </table>
  </div>
</body>
</html>
```

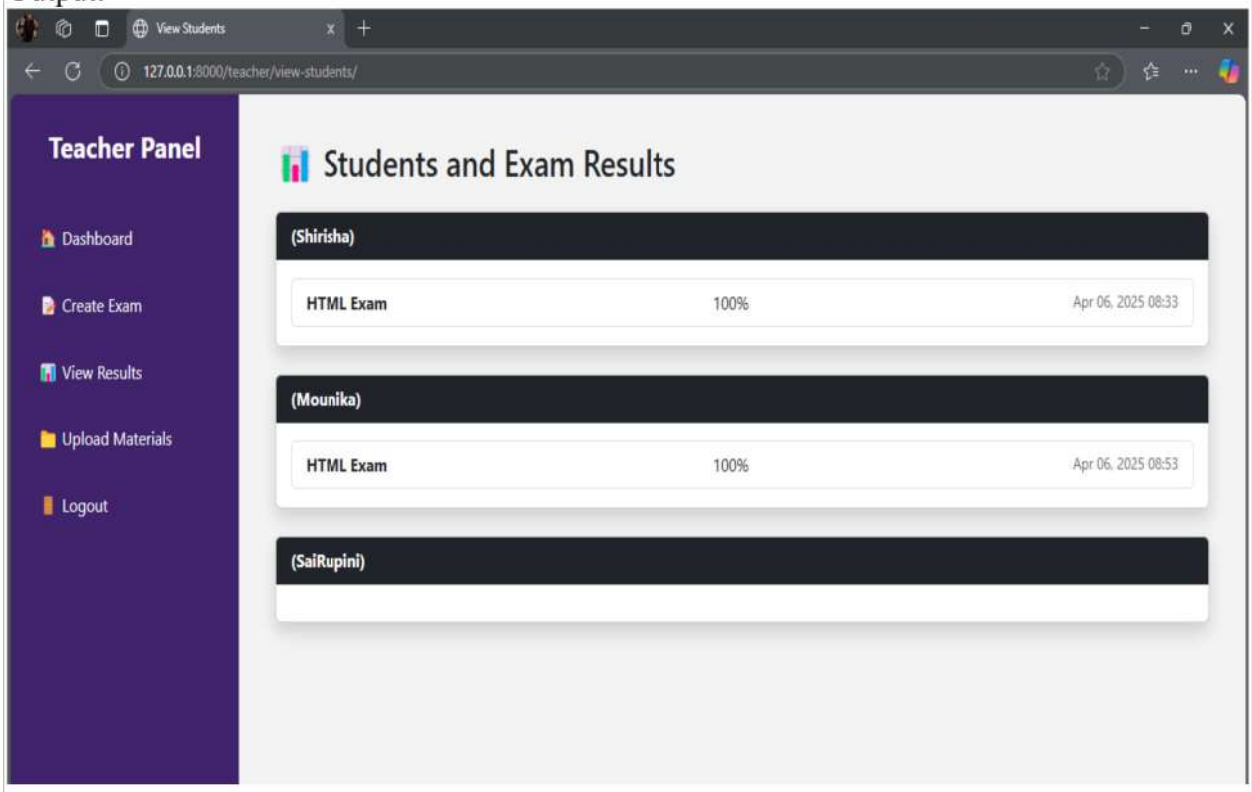
Description
Purpose:

Displays student exam results, including the exam title, score, and date taken.

Key Features:

1. Sidebar Navigation: Links to dashboard, exams, results, materials, and profile.
2. Compact Table Layout: Lists exam title, score, and date in a simple table.
3. Dynamic Content: Results are displayed using a Django loop for each student's data.
4. Responsive Design: Clean layout with clear typography and spacing for easy reading.

Output:



View_exams.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Available Exams</title>
    <style>
        body {
            font-family: 'Segoe UI', sans-serif;
            background-color: #f9f9f9;
            margin: 0;
        }
        .sidebar {
            height: 100vh;
            width: 220px;
            position: fixed;
            background-color: #343a40;
            padding-top: 20px;
            color: white;
        }
        .sidebar a {
            display: block;
            padding: 12px;
            color: white;
            text-decoration: none;
        }
        .sidebar a:hover {
            background-color: #495057;
        }
        .main {
            margin-left: 240px;
            padding: 30px;
        }
        table {
            width: 100%;
            background: white;
            border-collapse: collapse;
            box-shadow: 0 0 10px rgba(0,0,0,0.05);
        }
        th, td {
            padding: 14px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }
        h2 {
            margin-bottom: 20px;
```

```
        }
        a.take-btn {
            background-color: #28a745;
            color: white;
            padding: 6px 12px;
            text-decoration: none;
            border-radius: 4px;
        }
        a.take-btn:hover {
            background-color: #218838;
        }
    </style>
</head>
<body>
    <div class="sidebar">
        <h3 style="text-align:center;">Student</h3>
        <a href="{% url 'student_dashboard' %}">🏠 Dashboard</a>
        <a href="{% url 'student_view_exams' %}">📝 View Exams</a>
        <a href="{% url 'student_view_results' %}">📊 View Results</a>
        <a href="{% url 'view_materials' %}">📂 Materials</a>
        <a href="{% url 'edit_student_profile' %}">⚙️ Edit Profile</a>
    </div>

    <div class="main">
        <h2>Available Exams</h2>
        <table>
            <tr><th>Title</th><th>Description</th><th>Action</th></tr>
            {% for exam in exams %}
            <tr>
                <td>{{ exam.title }}</td>
                <td>{{ exam.description }}</td>
                    <td><a class="take-btn" href="{% url 'student_take_exam_detail' exam.id
%}">Take</a></td>
            </tr>
            {% endfor %}
        </table>
    </div>
</body>
</html>
```

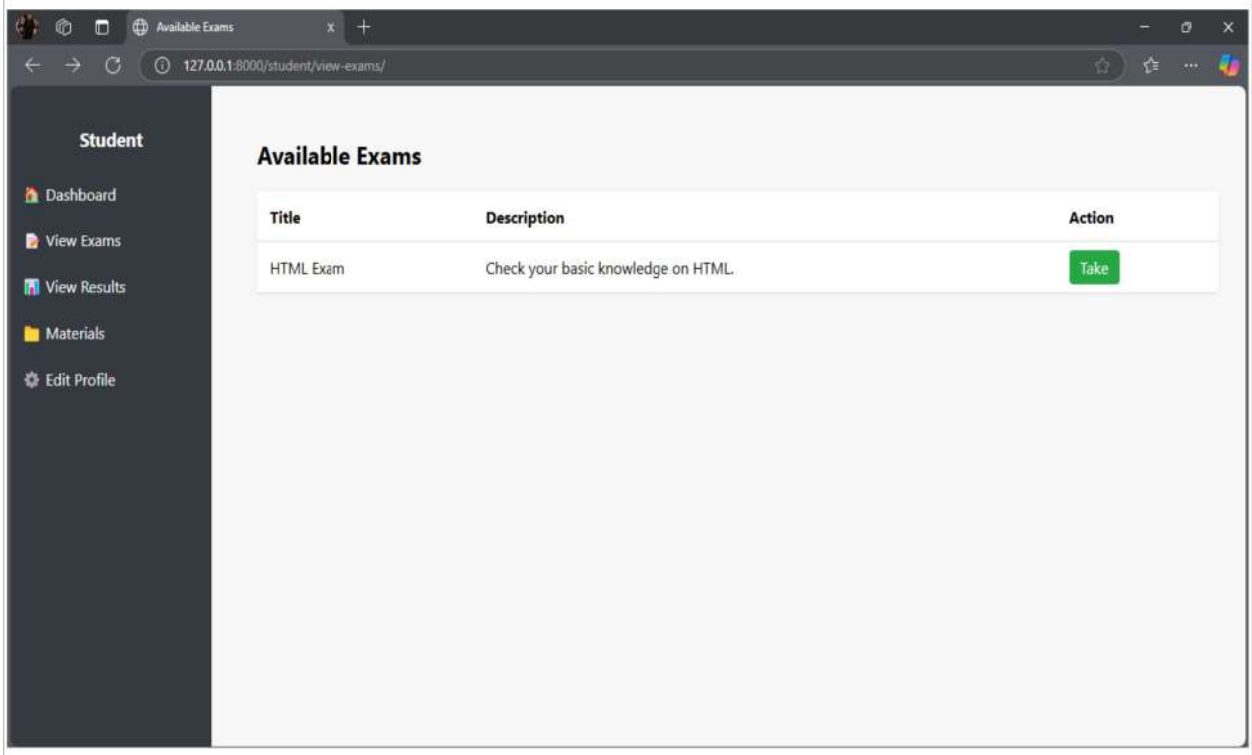
Description:

Purpose: Allows students to view and access exams available for them to take.

Key Features:

1. Sidebar Navigation: Quick links to dashboard, exams, results, materials, and profile.
2. Exam Table: Lists exam titles and descriptions with a "Take" button.
3. Clean Layout: Simple, user-friendly design with responsive styling.

Output:



take_exam.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>{{ exam.title }}</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
    }
    .sidebar {
      width: 230px;
      height: 100vh;
      background-color: #343a40;
      color: white;
      position: fixed;
      padding-top: 20px;
    }
    .sidebar a {
      display: block;
      color: white;
      padding: 12px 20px;
      text-decoration: none;
    }
    .sidebar a:hover {
      background-color: #495057;
    }
    .main {
      margin-left: 240px;
      padding: 20px;
    }
    form {
      margin-top: 20px;
    }
    ol {
      padding-left: 20px;
    }
    li {
      margin-bottom: 15px;
    }
    label {
      display: block;
      margin: 4px 0;
    }
    button {
```

```
      margin-top: 15px;
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border: none;
      border-radius: 4px;
    }
  </style>
</head>
<body>
  <div class="sidebar">
    <h2 style="text-align:center;">Student</h2>
    <a href="{% url 'student_dashboard' %}"> Dashboard</a>
    <a href="{% url 'student_view_exams' %}"> View Exams</a>
    <a href="{% url 'student_view_results' %}"> View Results</a>
    <a href="{% url 'view_materials' %}"> View Materials</a>
    <a href="{% url 'edit_student_profile' %}"> Edit Profile</a>
  </div>
  <div class="main">
    <h2>{{ exam.title }}</h2>
    <p>{{ exam.description }}</p>

    <form method="post">
      {% csrf_token %}
      <ol>
        {% for question in questions %}
        <li>
          <p>{{ question.text }}</p>
                <label><input type="radio" name="{{ question.id }}" value="A"> A.
{{ question.option_a }}</label>
                <label><input type="radio" name="{{ question.id }}" value="B"> B.
{{ question.option_b }}</label>
                <label><input type="radio" name="{{ question.id }}" value="C"> C.
{{ question.option_c }}</label>
                <label><input type="radio" name="{{ question.id }}" value="D"> D.
{{ question.option_d }}</label>
        </li>
        {% endfor %}
      </ol>
      <button type="submit">Submit</button>
    </form>
  </div>
</body>
</html>
```

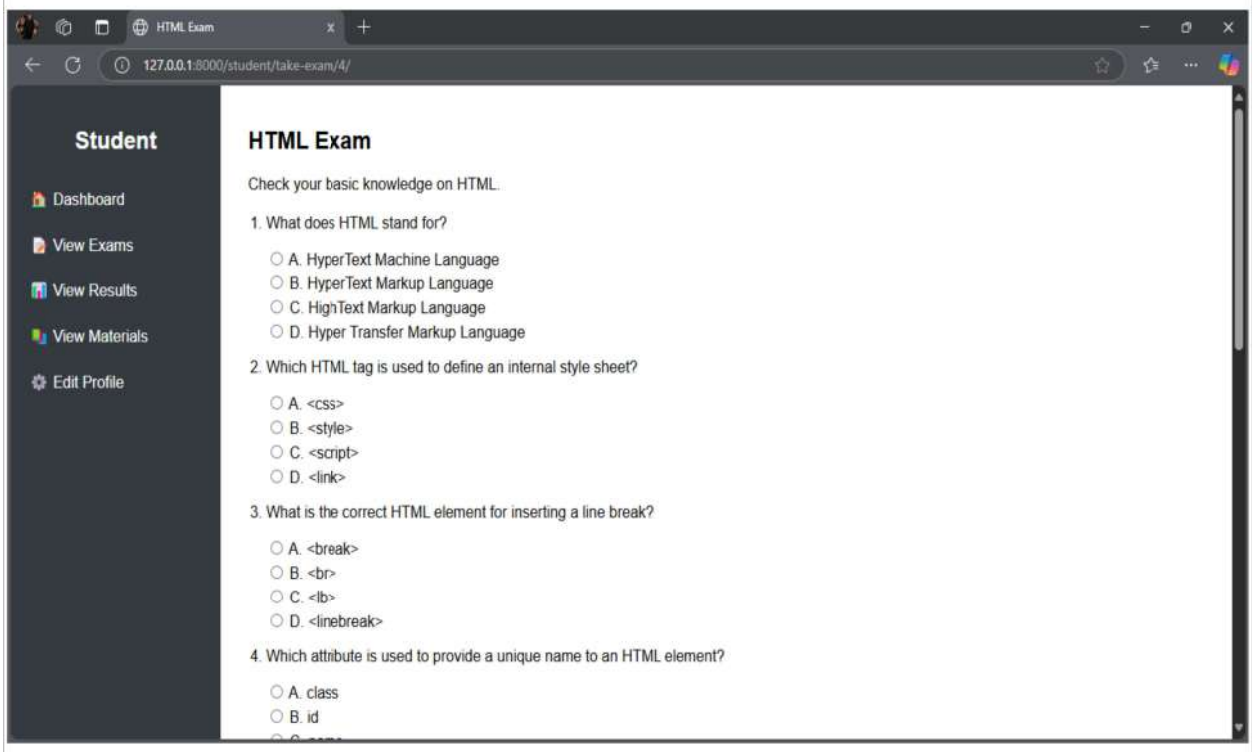
Description:

Purpose: Enables students to take a selected exam by answering multiple-choice questions.

Key Features:

1. Sidebar Navigation: Easy access to other student pages.
2. Dynamic Form: Lists questions with options, using radio buttons for answers.
3. Clean Layout: Simple and clear structure for focused exam-taking.

Output:



view_results.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Results</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
    }
    .sidebar {
      width: 230px;
      height: 100vh;
      background-color: #343a40;
```

```
      color: white;
      position: fixed;
      padding-top: 20px;
    }
    .sidebar a {
      display: block;
      color: white;
      padding: 12px 20px;
      text-decoration: none;
    }
    .sidebar a:hover {
      background-color: #495057;
    }
    .main {
      margin-left: 240px;
      padding: 20px;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    th, td {
      padding: 12px;
      text-align: left;
      border: 1px solid #dee2e6;
    }
    th {
      background-color: #28a745;
      color: white;
    }
  </style>
</head>
<body>
  <div class="sidebar">
    <h2 style="text-align:center;">Student</h2>
    <a href="{% url 'student_dashboard' %}"> Dashboard</a>
    <a href="{% url 'student_view_exams' %}"> View Exams</a>
    <a href="{% url 'student_view_results' %}"> View Results</a>
    <a href="{% url 'view_materials' %}"> View Materials</a>
    <a href="{% url 'edit_student_profile' %}"> Edit Profile</a>
  </div>
  <div class="main">
    <h2>My Results</h2>
    <table>
      <tr><th>Exam</th><th>Score</th><th>Date</th></tr>
```

```
    {% for result in results %}
    <tr>
      <td>{{ result.exam.title }}</td>
      <td>{{ result.score }}%</td>
      <td>{{ result.date_taken|date:"Y-m-d H:i" }}</td>
    </tr>
    {% endfor %}
   </table>
  </div>
</body>
</html>
```

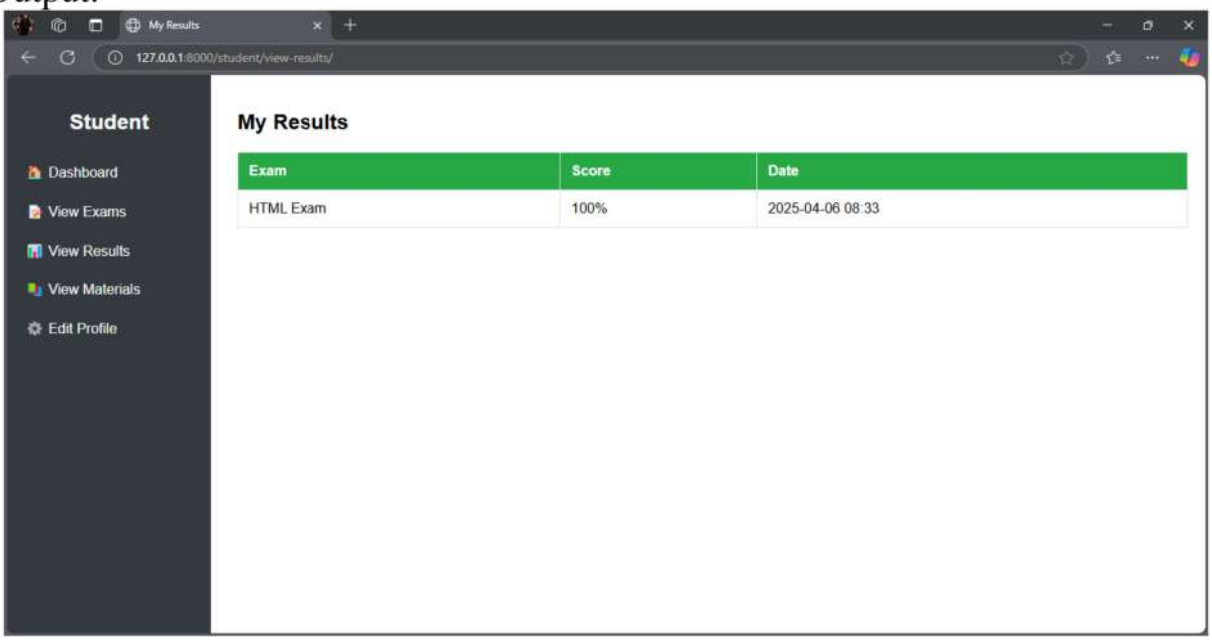
Description

Purpose: Displays student exam results, including the exam title, score, and date taken.

Key Features:

1. Compact Table Layout: Lists exam title, score, and date in a simple table.
2. Dynamic Content: Results are displayed using a Django loop for each student's data.

Output:



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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Database integration and configuration
7. Date of Experiment : 17-02-2025
8. Date of Submission of Report : 21-02-2025

Sno	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 INTEGRATION AND CONFIGURATION

Database Connectivity:

Database connectivity refers to the process of establishing a connection between an application (like a Django project) and a database system (like PostgreSQL, MySQL, or SQLite) so that the application can interact with the data stored in the database

Steps to connect database to django project:

Step-1: Install Database Driver

1. First, make sure you have the necessary database driver installed, depending on the type of database you are using.
2. For PostgreSQL: pip install psycopg2
3. For MySQL: pip install mysqlclien
4. For SQLite (default in Django, no installation required): SQLite is bundled with Python, so no need to install anything if you're using SQLite.

Step-2: Configure Database in settings.py

In your Django project's settings.py file, you need to define your database settings in the DATABASES dictionary.

Example for SQLite (default):

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

Step-3: Apply Migrations

After setting up the database, you need to create the database tables by running Django migrations.

First, run:

python manage.py makemigrations This creates the migration files.

Then apply the migrations with: python manage.py migrate

Step-4: Define Models

In Django, you define your database structure through models, which are Python classes that map to database tables.

Each model class corresponds to a table in the database, and the attributes of the model represent columns in the table.

Step-5: Run Migrations

Create Migrations: This generates migration files based on the changes in your models. python

manage.py makemigrations

Apply Migrations: This updates the database by applying the migration files and creating the

necessary tables in your database. python manage.py migrate

Step-6: Use Django Admin Interface

Create a Superuser: If you haven't already, create a superuser to access the admin interface. python manage.py createsuperuser

Register Models in Admin: To view and manage your models via the admin interface, register them in the admin.py file of your app.

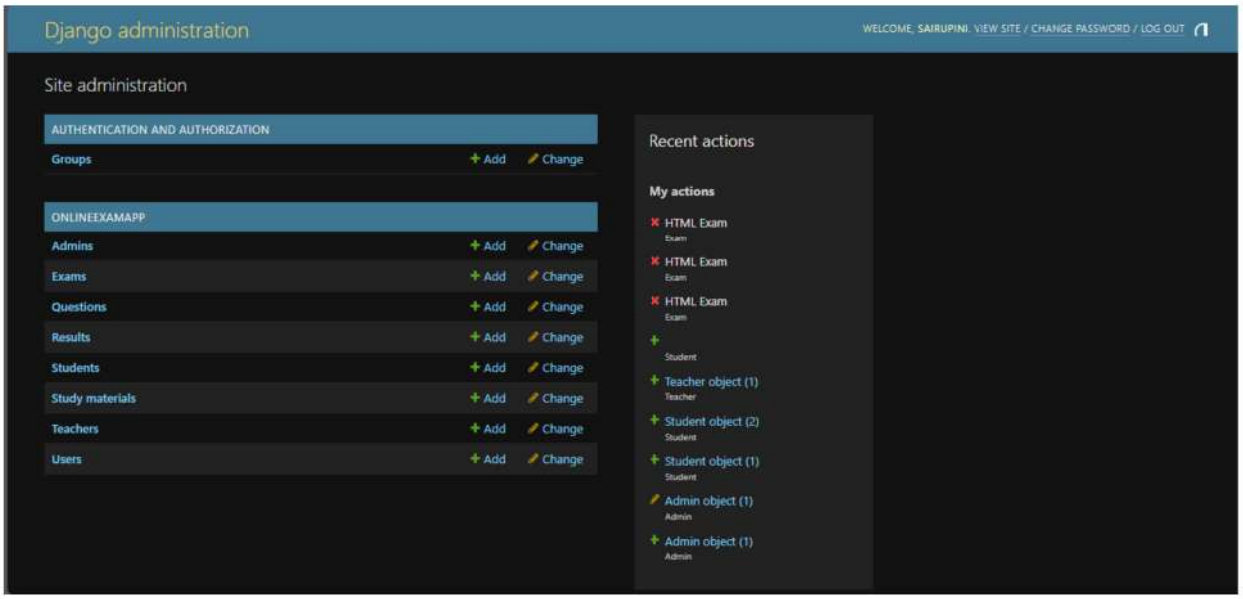
Access the Admin Interface: Start the development server and navigate to http://127.0.0.1:8000/admin/. Log in with the superuser credentials you created. You'll be able to manage your data here.

Step-7: Perform CRUD Operations

With your models and migrations in place, you can now perform CRUD operations (Create, Read, Update, Delete) on the data.

How databases are stored in django:

1. Django uses a relational database (SQLite by default) to store application data.
2. Data structures are defined using models in models.py (each model = table).
3. The makemigrations and migrate commands create and update database tables.
4. Data is stored in a file called db.sqlite3 when using SQLite (default setting).
5. Django's ORM (Object-Relational Mapper) allows you to interact with the database using Python code instead of SQL.



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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Forms in Django
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 07-03-2025

Sno	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 IN DJANGO

What are Forms?

Django forms handle user input and validation.

Forms are used in registration, login, and other data entry tasks.

Forms.py:

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from .models import CustomUser,Student,StudyMaterial,Teacher
from django.contrib.auth import get_user_model
class StudentRegistrationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password1', 'password2']
    def clean_email(self):  # Ensure email is unique
        email = self.cleaned_data.get('email')
        if CustomUser.objects.filter(email=email).exists():
            raise forms.ValidationError("A user with this email already exists.")
        return email
    def save(self, commit=True):
        user = super().save(commit=False)
        user.user_type = 'student'  # Assign user type
        if commit:
            user.save()
        return user


class TeacherRegistrationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password1', 'password2']

    def save(self, commit=True):
        user = super().save(commit=False)
        user.user_type = 'teacher'  # Assign user type
        if commit:
            user.save()
        return user


class AdminRegistrationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password1', 'password2']
```

```
    def save(self, commit=True):
        user = super().save(commit=False)
        user.user_type = 'admin'  # Assign user type
        if commit:
            user.save()
        return user


class LoginForm(AuthenticationForm):
       username   =   forms.CharField(widget=forms.TextInput(attrs={'class':   'form-control',
'placeholder': 'Username'}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={'class': 'form-control',
'placeholder': 'Password'}))

class StudentCreationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password1', 'password2']

    def save(self, commit=True):
        user = super().save(commit=False)
        user.user_type = 'student'
        user.is_active = True  # Ensure account is active
        user.set_password(self.cleaned_data["password1"])  # Properly hash password
        if commit:
            user.save()
            Student.objects.create(user=user)
        return user
class StudyMaterialForm(forms.ModelForm):
    class Meta:
        model = StudyMaterial
        fields = ['title', 'description', 'file']
class EditProfileForm(forms.ModelForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email']
CustomUser = get_user_model()

class StudentEditForm(forms.ModelForm):
    email = forms.EmailField()

    class Meta:
        model = Student
        fields = ['user']
```

```
    def __init__(self, *args, **kwargs):
        self.user_instance = kwargs.pop('user_instance', None)
        super().__init__(*args, **kwargs)
        if self.user_instance:
            self.fields['email'].initial = self.user_instance.email

    def save(self, commit=True):
        instance = super().save(commit=False)
        if self.user_instance:
            self.user_instance.email = self.cleaned_data['email']
            if commit:
                self.user_instance.save()
        if commit:
            instance.save()
        return instance

class TeacherEditForm(forms.ModelForm):
    email = forms.EmailField()

    class Meta:
        model = Teacher
        fields = ['user']

    def __init__(self, *args, **kwargs):
        self.user_instance = kwargs.pop('user_instance', None)
        super().__init__(*args, **kwargs)
        if self.user_instance:
            self.fields['email'].initial = self.user_instance.email

    def save(self, commit=True):
        instance = super().save(commit=False)
        if self.user_instance:
            self.user_instance.email = self.cleaned_data['email']
            if commit:
                self.user_instance.save()
        if commit:
            instance.save()
        return instance
```

Description:

CustomUserForm:

Extends UserCreationForm (built-in Django form for user registration).

StudentForm, TeacherForm, AdminForm:

Handles additional fields for each user type.

Forms in Views:

1. Creates a new user.
2. Based on role, it also creates a corresponding Student, Teacher, or Admin profile.
3. Redirects to Dashboard after Registration
4. Automatically logs in the user.

Forms in Templates:

1. Dynamically Shows/Hides Fields Based on Selected Role.
2. Uses Django form rendering ({{ user_form.as_p }}).

Types of Forms in Django:

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

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Models in Django
7. Date of Experiment : 07-03-2025
8. Date of Submission of Report : 27-03-2025

Sno	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 are Models?

Models define the database schema (structure).

Each model is a Python class that maps to a database table.

Django's ORM (Object-Relational Mapping) automatically converts models into database tables.

Models.py:

```
from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser(AbstractUser):
    USER_TYPE_CHOICES = (
        ('student', 'Student'),
        ('teacher', 'Teacher'),
        ('admin', 'Admin'),
    )
       user_type = models.CharField(max_length=10, choices=USER_TYPE_CHOICES,
default='student')
    email = models.EmailField(unique=True)

class Student(models.Model):
    user = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
    def __str__(self):
        return self.user.get_full_name() or self.user.username

    def get_results(self):
        return self.user.result_set.all()

class Teacher(models.Model):
    user = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
    def __str__(self):
        return self.user.username

class Admin(models.Model):
    user = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
class Exam(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    subject = models.CharField(max_length=100)
         created_by = models.ForeignKey(CustomUser, on_delete=models.CASCADE,
limit_choices_to={'user_type': 'teacher'})

    def __str__(self):
```

```
        return self.title

class Question(models.Model):
            exam = models.ForeignKey(Exam, on_delete=models.CASCADE,
related_name='questions')
    text = models.TextField()
    option_a = models.CharField(max_length=200)
    option_b = models.CharField(max_length=200)
    option_c = models.CharField(max_length=200)
    option_d = models.CharField(max_length=200)
     correct_option = models.CharField(max_length=1, choices=[('A', 'A'), ('B', 'B'), ('C',
'C'), ('D', 'D')]
    )
    def __str__(self):
        return self.text

class Result(models.Model):
          student = models.ForeignKey(CustomUser, on_delete=models.CASCADE,
limit_choices_to={'user_type': 'student'})
    exam = models.ForeignKey(Exam, on_delete=models.CASCADE)
    score = models.IntegerField()
    date_taken = models.DateTimeField(auto_now_add=True)

class StudyMaterial(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    file = models.FileField(upload_to='materials/')
    uploaded_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title
```

Description:

CustomUser Model

Extends Django's default User model to add a role field (student, teacher, or admin).

Student Model

One-to-one relationship with CustomUser (every student is a user).

Stores student_id and course.

Teacher Model

Similar to students but includes employee_id and department.

Admin Model

Stores additional admin-related data (admin_code).

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Migrations: Sync with database
7. Date of Experiment : 07-03-2025
8. Date of Submission of Report : 27-03-2025

Sno	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:**

MIGRATIONS

After creating your templates, forms, and especially models in Django, you need to sync your models with the database using Django's migration system.

What Are Migrations?

Migrations in Django are a way of applying changes made to models (like creating new fields or tables) to the actual database schema.
They allow you to update your database structure to match changes made to your models without having to manually write SQL. Essentially, migrations are Python files that contain the instructions to apply changes to the database, such as adding or modifying tables and fields.
They track and apply changes like:

1. Creating new models
2. Adding, removing, or modifying model fields
3. Changing relationships between models

Workflow after creating models:

Once you've written or updated your models.py file and corresponding templates and forms, follow these steps to sync the database:

1.Make migrations:

python manage.py makemigrations

This command tells Django to **look for changes** in your models and create new migration files in your app's `migrations/` directory. These files describe the database changes Django should make.

2.Migrate:

python manage.py migrate

This command applies the migration files to the actual database, creating or updating the tables.

What Happens:

1. Tables for your models are created in the SQLite database (or any DB you're using).
2. Django also applies any built-in migrations for apps like `auth`, `admin`, `sessions`, etc.

Migration Files:

Migration files are Python scripts located in your app's migrations/ folder. You don't need to modify these manually—they're auto-generated by Django.Each file has a name like 0001_initial.py, 0002_add_field.py, etc., and contains classes and operations Django uses to apply changes.

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Django on cloud platform
7. Date of Experiment : 27-03-2025
8. Date of Submission of Report : 04-04-2025

Sno	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 Applicaations on Cloud Platforms

Deployment:

Deployment is the process of making a Django web application live on the internet so users can access it. This involves hosting your app on a cloud server like AWS, Google Cloud, Digital Ocean, Heroku, or PythonAnywhere.

Cloning Project to Github:

1. Initialize Git in your project folder

   git init

   Starts version control in your local Django project

2. Add all files to Git staging area

   git add .

   Tells Git to track all files for the next commit

3. Commit the added files

   git commit -m "Initial commit"

   Saves a snapshot of your project with a message

4. Add the GitHub repository as the remote origin

   git remote add origin https://github.com/Sai-rupini/your-repo-name.git

   Connects your local project to the GitHub repo

5. Push your code to GitHub

   git branch -M main

   git push -u origin main

   Sets the main branch and uploads your code to GitHub

Github Link:

https://github.com/Sai-rupini/ExamTrack.git

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

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

1. Name of the Laboratory : Django Framework Laboratory
2. Name of the Student : Ch.Sai rupini
3. Roll No : 23VV1A1210
4. Class : II B.Tech II Semester
5. Academic Year : 2024-2025
6. Name of Experiment : Frontend Web developer Certification
7. Date of Experiment : 04-04-2025
8. Date of Submission of Report : 04-04-2025

Sno	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:**

Frontend Web Developer Certificate