

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



**DJANGO FRAME WORK LAB
COURSE SYLLABUS REPOSITORY**

DONE BY

MOHAMMAD ARIF 23VV1A1233

**UNDER GUIDANCE OF
Mrs. MADUMITHA CHANDA**

**DEPARTMENT OF
INFORMATION TECHNOLOGY**



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

Regd. No : 23VV1A1233

CERTIFICATE

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

No. of Task Completed and Certified: 13

Lecture In-Charge:

Date:

Head of the Department



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

Website: www.jntugvcev.edu.in

Subject Name: DESIGN THINKING & INNOVATION

Regulation: R23

Subject Code: R2322BSH01

Academic Year: 2025

COURSE OUTCOMES

NBA Subject Code	Course Outcomes	
	CO1	Define concepts of design thinking.
	CO2	Explain fundamentals of design thinking & innovation.
	CO3	Apply design thinking to solve problems in various sectors.
	CO4	Analyze to work in multidisciplinary environment.
	CO5	Evaluate the value of creativity. Formulate specific real-time problem statements.
	CO6	

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	2	2	1	1	1	1	1	1	2	1	1	2	2	1	1
	CO2	3	3	2	1	2	2	1	2	2	2	1	2	3	2	2
	CO3	3	3	3	2	3	2	2	2	3	3	2	3	3	3	3
	CO4	2	2	2	2	2	1	2	1	3	2	2	2	3	3	2
	CO5	2	3	3	2	3	2	1	2	2	2	2	3	3	3	3
	CO6	3	3	3	3	3	2	2	2	3	2	2	3	3	3	3

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

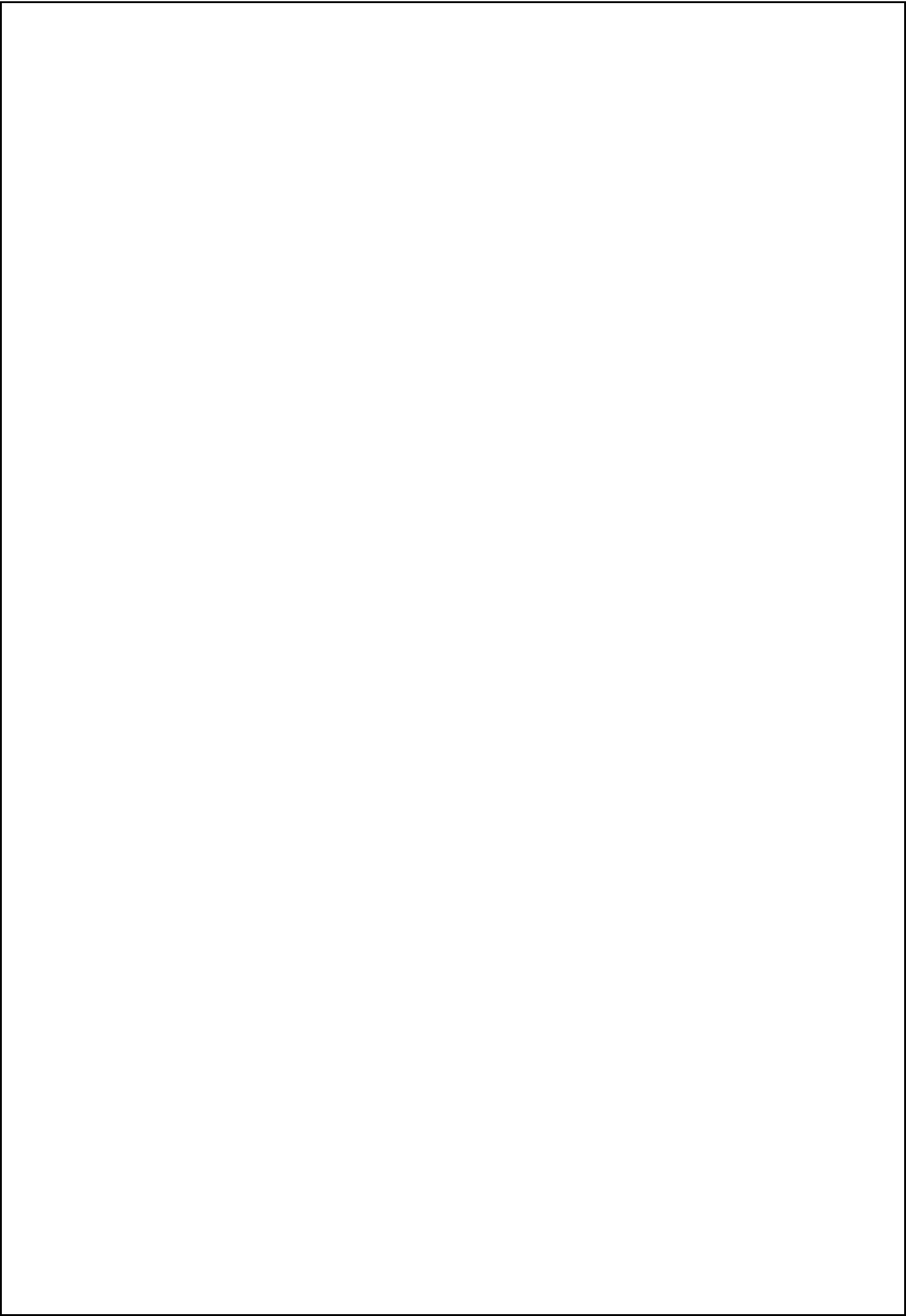
1:3 Slight (Low) 2: Moderate (Medium) 3: Substantial (High) If there is no correlation, put “-”

Signature of the course instructor

Table of contents:

SNO	DATE	TABLE OF CONCENTS	PAGENO	MARKS	SIGNATURE
1	13-12-2024	Understanding Django and its Libraries	1-12		
2	20-12-2024	Introduction to Django Framework	13-14		
3	27—12-2024	Step-by-Step Guide to Installation of Django	15-16		
4	03-01-2025	Linking Views and URL Configurations	17-18		
5	24-01-2025	Exploring Django Views	19-24		
6	24-01-2025	Setting Up App-level URLS	25-27		
7	31-1-2025	Working with Templates in Django	28-77		
8	17-02-2025	Database Integration and Configuration SQLITE	78-80		
9	21-02-2025	Heading Forms in Django	81-83		
10	21-02-2025	Defining and Using Models	84-87		
11	07-03-2025	Migration Sync with the Database	88-89		
12	27-03-2025	Deploying Django Application on cloud platforms	90-91		
13	04-04-2025	Front End Wed Developer Certification	92-93		

Date:**Signature of Faculty**





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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|--|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Understanding Django and its Libraries |
| 7. Date of Experiment | : 13-12-2024 |
| 8. Date of Submission of Report | : 20-12-2024 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Python Libraries

Python libraries are pre-written code that provides a set of functionalities, making it easier to perform specific tasks. They are reusable, well-tested, and widely adopted, saving developers time and effort.

Tkinter - GUI:

- **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.

implement the code we need to install the tinker library:

```
pip install tk

from tkinter import Tk, Label

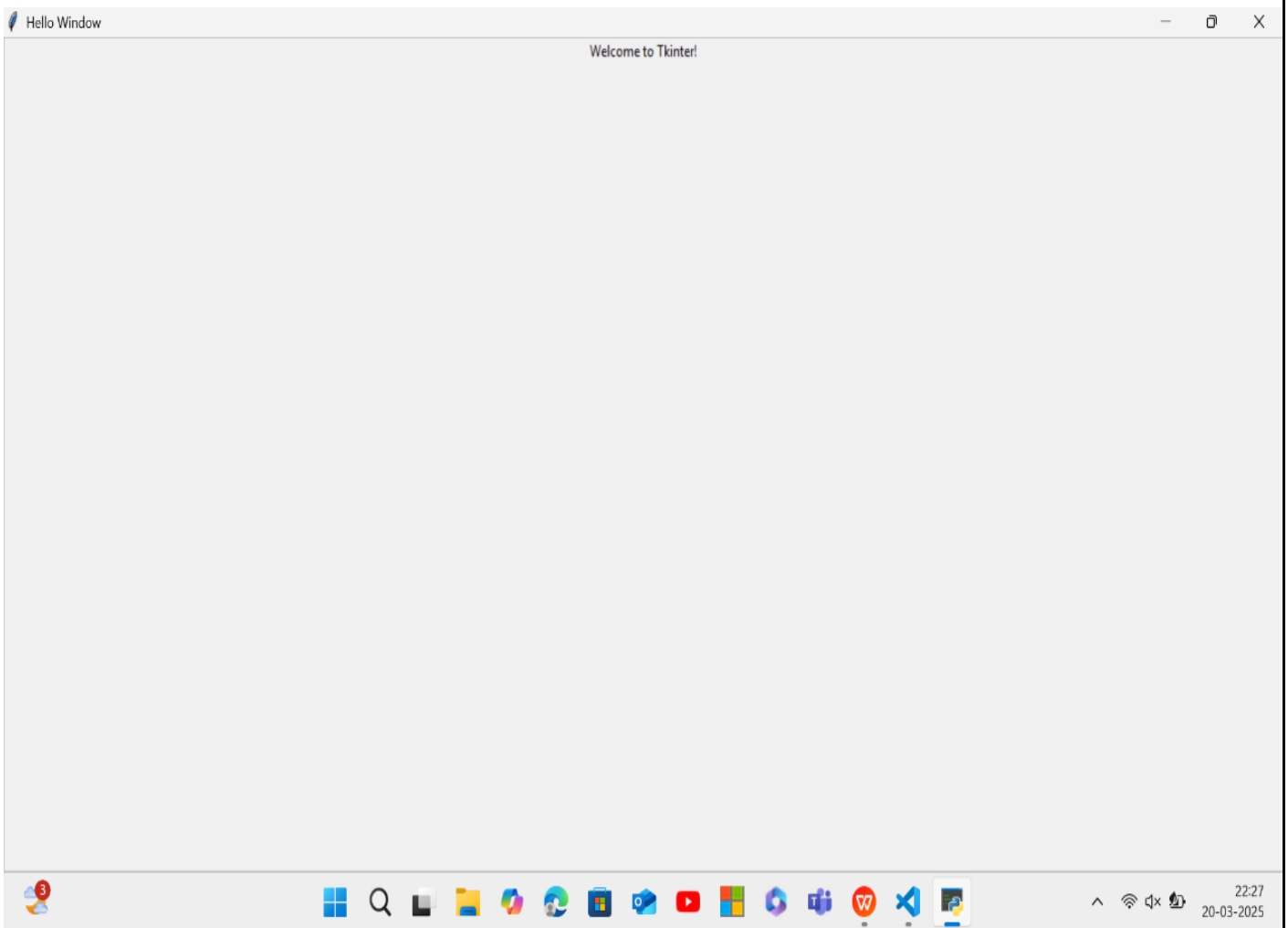
root = Tk()

root.title("Hello Window")

Label(root, text="Welcome to Tkinter!").pack()

root.mainloop()
```

Output:



Requests - HTTP Requests

- **Purpose:** Simplifies HTTP requests to interact with web APIs.
- **Key Features:**
 - Send GET, POST, PUT, DELETE requests easily.
 - Handle request parameters, headers, and cookies.
 - Simple error handling and response handling.
- **Common Use:** Interact with REST APIs, download content from the web.

To implement the code, we need to install the tkinter library:

```
pip install tk
```


Code:

root.mainloop()

import tkinter as tk

from tkinter import messagebox

def validate_login():

if username_entry.get() == "user" and password_entry.get() == "password":

messagebox.showinfo("Success", "Login Successful!")

else:

messagebox.showerror("Failed", "Invalid username or password")

root = tk.Tk()

root.title("Login Form")

root.geometry("400x300")

tk.Label(root, text="Username").pack(pady=5)

username_entry = tk.Entry(root)

username_entry.pack(pady=5)

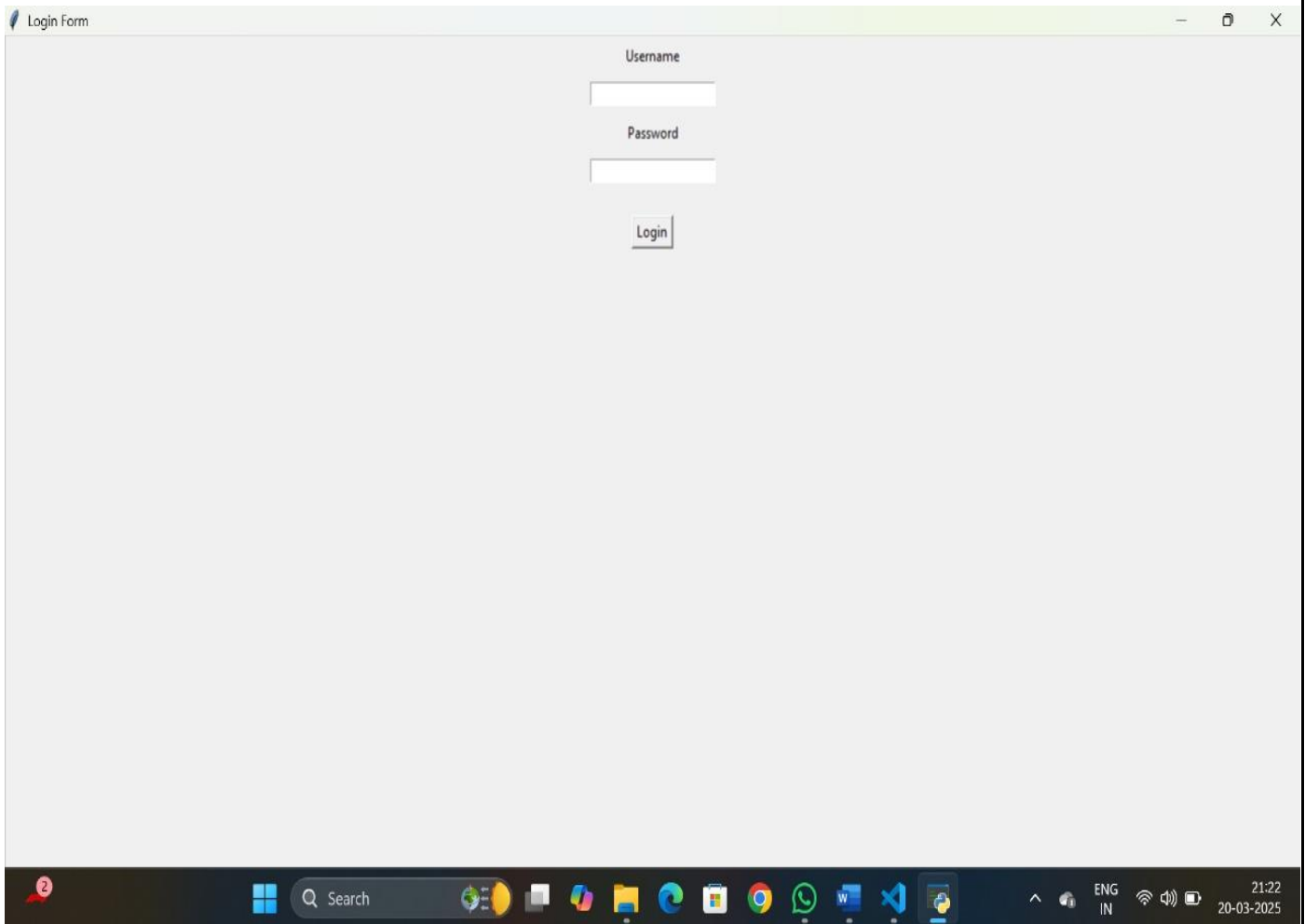
tk.Label(root, text="Password").pack(pady=5)

password_entry = tk.Entry(root, show="*")

password_entry.pack(pady=5)

tk.Button(root, text="Login", command=validate_login).pack(pady=20)

Output:



BeautifulSoup4 - Web Scraping

Purpose: Parses HTML and XML documents to extract data.

Key Features:

- Easy navigation and searching within HTML.
- Supports different parsers like HTML. Parser, and html5lib.

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

To implement the code, we need to install the beautifulsoup4 library:

```
pip install bs4
```

(or)

```
pip install beautifulsoup4
```

Code:

```
import requests

from bs4 import BeautifulSoup

# URL of the website to scrape
url = 'https://example.com' # Replace with the actual URL

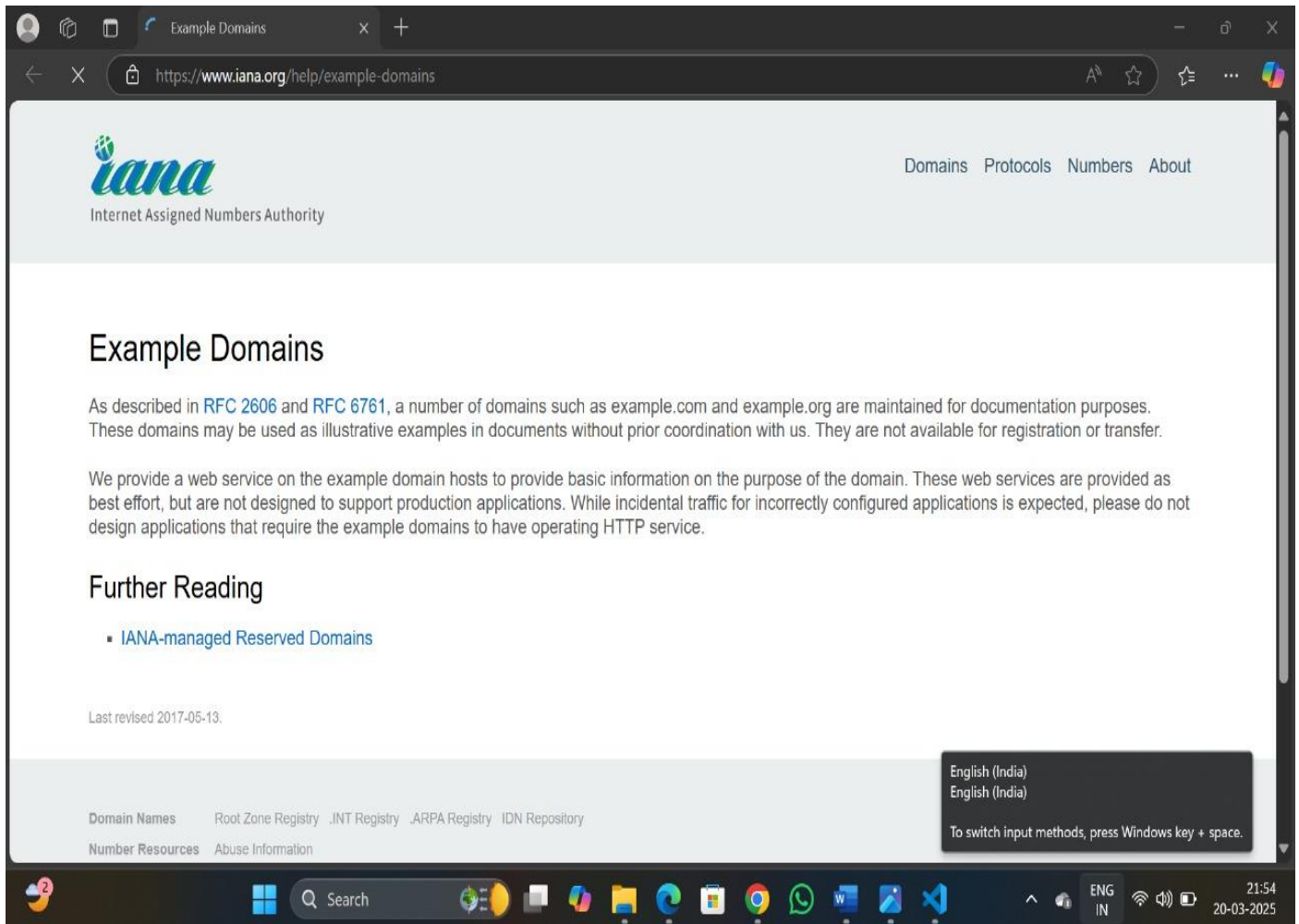
# Send a GET request to fetch the raw HTML content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Get the page title
title = soup.title.string if soup.title else "No title found"
print(f"Title of the page: {title}")

# Find all headings (h1, h2, h3)
headings = soup.find_all(['h1', 'h2', 'h3'])
print("\nHeadings:")
for heading in headings:
    print(heading.get_text(strip=True))

# Find all hyperlinks on the page
links = soup.find_all('a', href=True)
print("\nLinks:")
for link in links:
    print(f"Link: {link['href']}")
```

Output:



CherryPy

- **Purpose:** Minimalistic web framework for building web applications.
- **Key Features:**
 - Provides a simple and fast HTTP server.
 - Handles routing, cookies, sessions, and file uploads.
- **Common Use:** Building web applications with a lightweight framework.

To implement the code, we need to install the cherry py library:

```
pip install cherypy
```

Code:

```
import cherrypy
```

```
class HelloWorld(object):
```

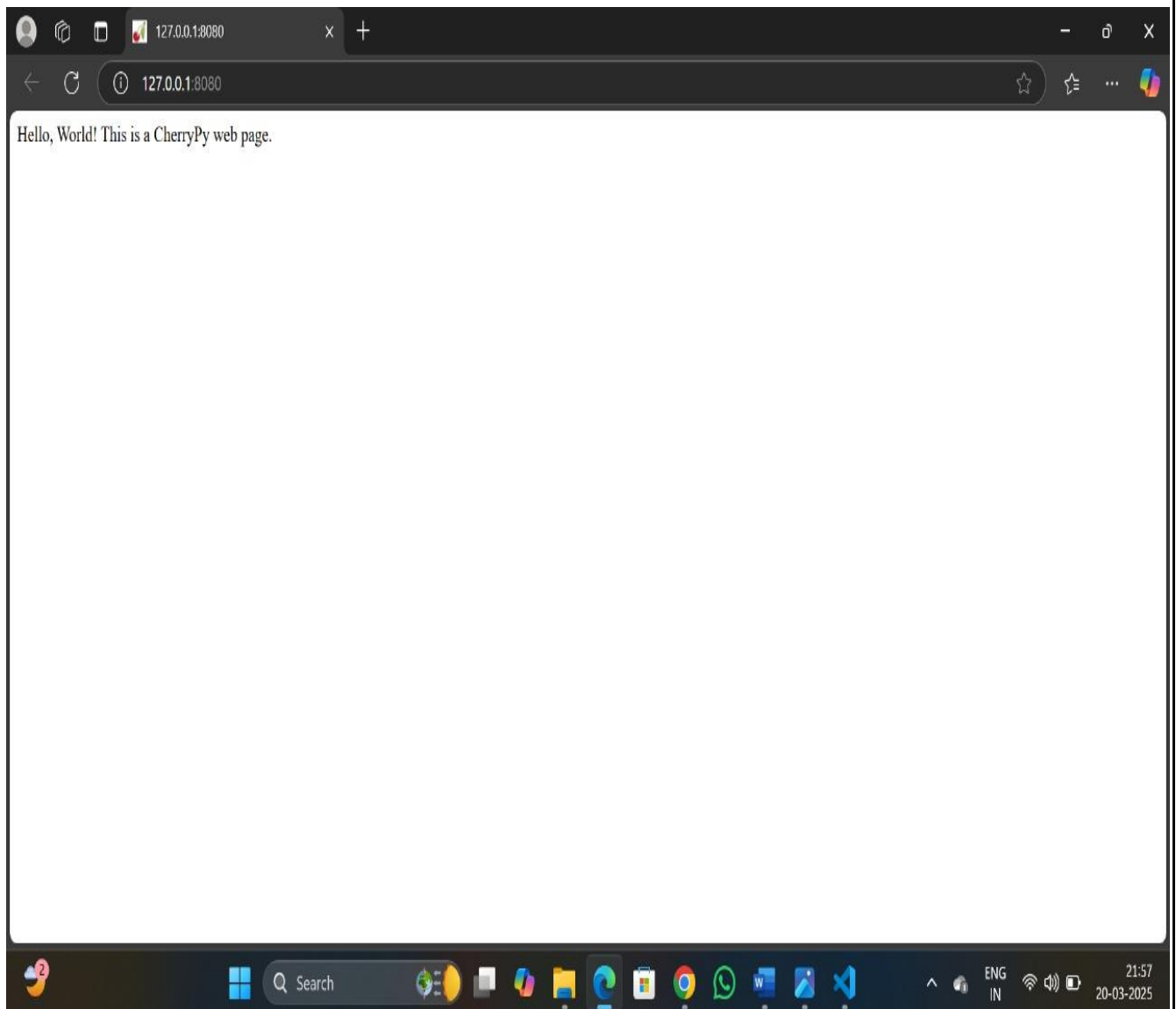
```
    @cherrypy.expose
```

```
    def index(self):
```

```
        return "Hello, World! This is a CherryPy web page."
```

```
if __name__ == '__main__':
```

```
    cherrypy.quickstart(HelloWorld())
```

Output:

Flask

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

To implement the code, we need to install the flask library:

```
pip install flask
```

Code:

```
from flask import Flask
```

```
# Initialize the Flask application
```

```
app = Flask(__name__)
```

```
# Define a route for the root URL "/"
```

```
@app.route('/')
```

```
def greet():
```

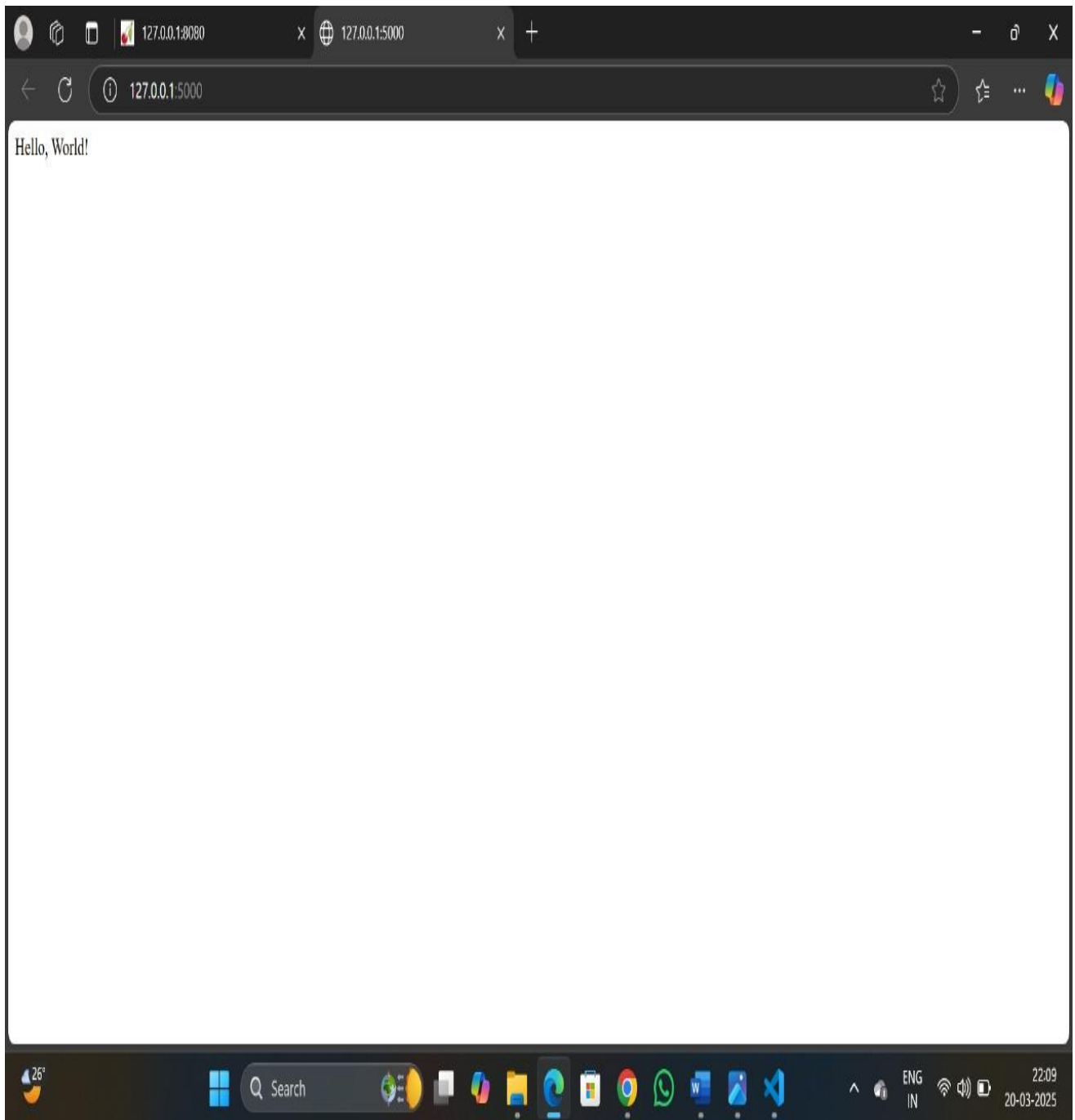
```
    return 'Hello, World!'
```

```
# Run the app in debug mode
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

Output:



Bottle

Purpose: Simple and lightweight WSGI micro-framework.

- **Key Features:**
 - Single-file framework, minimalistic, and fast.
 - No dependencies, supports routing, templates, and form handling.
- **Common Use:** Small web applications, APIs, and prototypes.

To implement the code, we need to install the bottle library:

```
pip install bottle
```

Code:

```
from bottle import route, run  
@route('/')  
def index():  
    return "Hello, World! This is a Bottle web page."  
run(host='localhost', port=8080)
```

Output:



Hello, World!
This is a Bottle web page.

Summary:

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



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Introduction to Django Framework
7. Date of Experiment : 20-12-2024
8. Date of Submission of Report : 27-12-2024

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Django: A Web Framework for Python

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

Key Features of Django:

- Fast Development – Comes with built-in features like authentication, database management, and an admin panel.
- Scalability – Suitable for small projects to enterprise-level applications.
- Security – Protects against common security threats (SQL Injection, CSRF, XSS, etc.).
- ORM (Object-Relational Mapper) – Allows database interaction using Python instead of SQL.
- Built-in Admin Panel – Auto-generates an admin interface for managing data.
- Reusable App-Developers can create modular and reusable components.

Django's MVT Architecture:

- Model (M): Handles all interactions with the database. Each model maps to a database table.
- View (V): Contains the business logic of your application. It processes requests, fetches data from the model, and passes it to the template.
- Template (T): HTML files that are dynamically rendered using Django Template Language (DTL). These files receive data from views and present it to the user.

```
├── hello_dj
│   ├── app1
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── forms.py
│   │   ├── migrations
│   │   │   └── __init__.py
│   │   ├── models.py
│   │   ├── templates
│   │   │   └── app1
│   │   ├── tests.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── hello_dj
│   │   ├── __init__.py
│   │   ├── asgi.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   └── wsgi.py
│   ├── manage.py
│   ├── static
│   └── templates
8 directories, 15 files
```



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: it@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Step by Step Guide to Installation of Django
7. Date of Experiment : 27-12-2024
8. Date of Submission of Report : 03-01-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Steps to Create Django Project:

Step-1: Checking the installation & version of Python

```
python --version
```

Step-2: Create a folder to store the project

Step-3: Create virtual environment

```
python -m venv
```

Step-4: Installation of Django

```
pip install Django
```

Step-5: Start new project

```
django-admin startproject myproject
```

Step-6: Start a new app

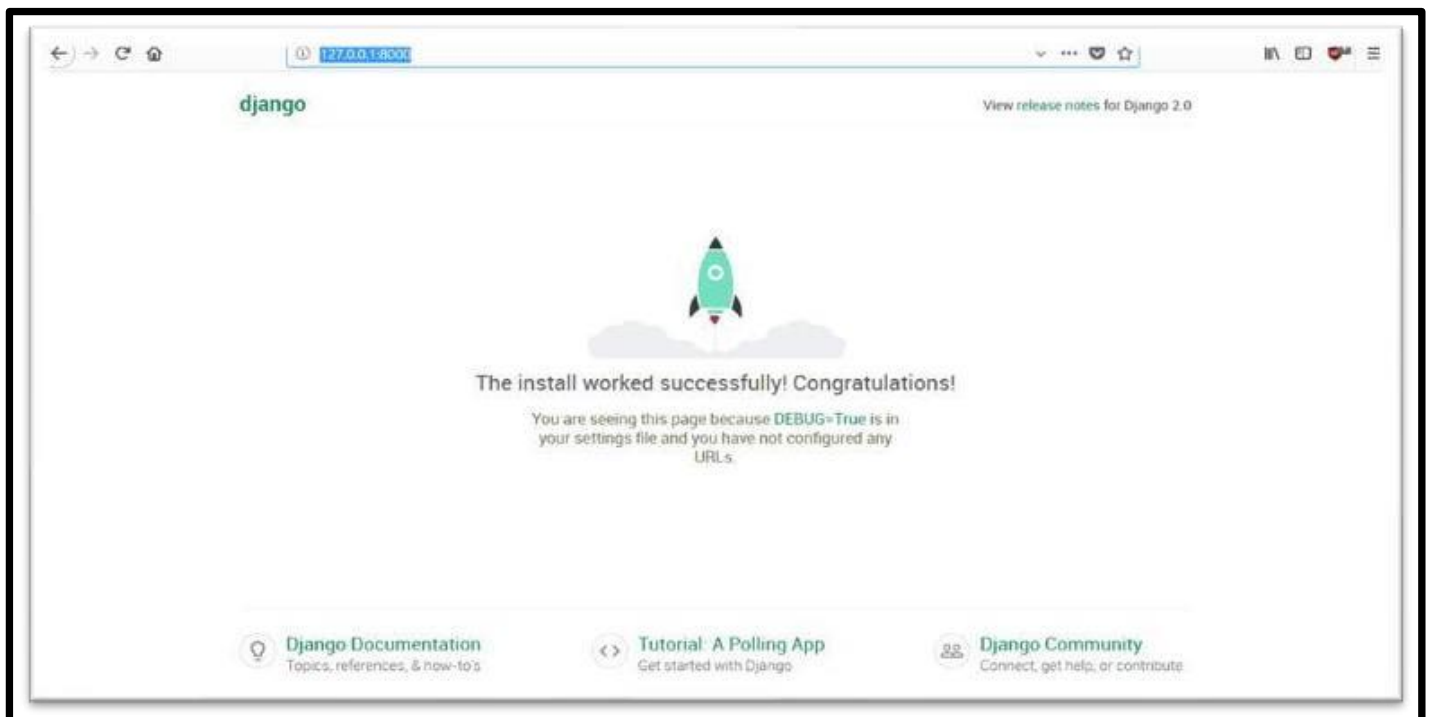
```
python manage.py startapp myapp
```

Step-7: Run the server

```
python manage.py runserver
```

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

OUTPUT:



=



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|--------------------------|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Linking Views and Urls |
| 7. Date of Experiment | : 03-01-2025 |
| 8. Date of Submission of Report | : 24-01-2025 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Connecting Views and URLs:

Set Up URLs:

Project-level URLs:

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('myapp1.urls')), # Include URLs from your app
]
```

App-level URLs;

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

```
urlpatterns = [
    path("", home, name='home'),
]
```

Create a Sample View:

```
from django.http import HttpResponse
def home(request):
    return HttpResponse('<h1>Welcome to my project</h1>')
```

Run Migrations:

```
python manage.py migrate
```

Run the Server and Test:

```
python manage.py runserver
```



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|--------------------------|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Exploring Django views |
| 7. Date of Experiment | : 24-01-2025 |
| 8. Date of Submission of Report | : 31-01-2025 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

views.py :

views.py in this project handles the logic for displaying pages (like home, profile, settings, syllabus), managing user authentication, saving user preferences, and filtering syllabi. It connects the frontend templates with the backend database to process and respond to user actions.

Code:

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import login, logout, authenticate
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .forms import CustomSignupForm, CustomLoginForm
from .models import CustomUser
from django.contrib.auth.models import User
from .models import Faculty
from .models import FacultyRequest
from django.contrib.auth.decorators import login_required
from django.views.decorators.csrf import csrf_exempt
from .forms import SyllabusForm
from .models import Syllabus
from django.http import JsonResponse

# Home Page View
def home(request):
    return render(request, 'home.html')

# Books Page View
def books(request):
    return render(request, 'books.html')

# Settings Page View
def settings(request):
    return render(request, 'settings.html')

# About Page View
def about(request):
    return render(request, 'about.html')

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

# Syllabus Page View
def syllabus(request):
    context = {
        "regulations": ["R17", "R20", "R23"],
        "departments": ["CSE", "IT", "ECE", "EEE"],
        "years": ["1st Year B.Tech", "2nd Year B.Tech", "3rd Year B.Tech", "4th Year B.Tech"],
    }
    return render(request, "syllabus.html", context)
```

```

# Login View
def login_view(request):
    if request.user.is_authenticated:
        return redirect('home')

    if request.method == 'POST':
        form = CustomLoginForm(request.POST)
        if form.is_valid():
            # Get the username and password from the form
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')

            # Authenticate the user
            user = authenticate(request, username=username, password=password)

            if user is not None:
                # If the user is authenticated, log them in
                login(request, user)
                return redirect('home') # Redirect to the homepage or dashboard after successful login
            else:
                # If authentication fails, add error
                form.add_error(None, 'Invalid username or password.')
        else:
            form = CustomLoginForm()

    return render(request, 'login.html', {'form': form})

# Signup View
def signup(request):
    if request.user.is_authenticated:
        return redirect('home')

    if request.method == 'POST':
        form = CustomSignupForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('home')
        else:
            form = CustomSignupForm()

    return render(request, 'signup.html', {'form': form})

# Profile Page View
@login_required
def profile(request):
    return render(request, 'profile.html')

# Add Syllabus (Faculty Only)
def add_syllabus(request):
    if request.user.role not in ["Faculty", "Admin", "Superadmin"]:
        messages.error(request, "You are not authorized to upload a syllabus.")
        return redirect("homee") # Redirect unauthorized users

    if request.method == "POST":

```

```

form = SyllabusForm(request.POST, request.FILES)
if form.is_valid():
    syllabus = form.save(commit=False)
    syllabus.uploaded_by = request.user
    syllabus.save()
    messages.success(request, "Syllabus uploaded successfully!")
    return redirect("add_syllabus")
else:
    messages.error(request, "Error uploading syllabus. Please check the form.")

else:
    form = SyllabusForm()

return render(request, "add_syllabus.html", {"form": form})

@login_required
def faculty_requests(request):
    if not request.user.is_admin():
        return redirect('home')

    pending_faculty = CustomUser.objects.filter(role="Faculty", is_active=False)
    return render(request, 'faculty_requests.html', {'pending_faculty': pending_faculty})

# Approve Faculty (Admin Only)
@login_required
def approve_faculty(request, faculty_id):
    if not request.user.is_admin():
        return redirect('home')

    faculty = get_object_or_404(CustomUser, id=faculty_id, role="Faculty", is_active=False)

    faculty.is_active = True
    faculty.save()

    messages.success(request, f"{faculty.username} has been approved as Faculty.")
    return redirect('faculty_requests')

# Reject Faculty (Admin Only) - NEW FUNCTION
@login_required
def reject_faculty(request, faculty_id):
    if not request.user.is_admin():
        return redirect('home')

    faculty = get_object_or_404(CustomUser, id=faculty_id, role="Faculty", is_active=False)

    # Prevent superadmin from being deleted
    if faculty.is_superuser:
        messages.error(request, "Superadmin cannot be rejected!")
        return redirect('faculty_requests')

    faculty.delete()

    messages.success(request, f"Faculty request from {faculty.username} has been rejected.")
    return redirect('faculty_requests')

```

```

# Upload Syllabus (Admin Only)
@login_required
def upload(request):
    if request.user.role != "Admin":
        return redirect('home')

    if request.method == 'POST' and request.FILES['syllabus']:
        syllabus_file = request.FILES['syllabus']
        # Handle file saving logic here, e.g., save to a model or file system

        messages.success(request, 'Syllabus uploaded successfully!')
        return redirect('upload')

    return render(request, 'upload.html')

# Logout View
def logout_view(request):
    logout(request)
    return redirect('login')

def superadmin_dashboard(request):
    if not request.user.is_superuser:
        return redirect('home') # Redirect non-superadmins to home

    users = CustomUser.objects.exclude(username=request.user.username) # Exclude self
    return render(request, 'superadmin_dashboard.html', {'users': users})

def manage_user_view(request, user_id):
    try:
        user = User.objects.get(id=user_id)
    except User.DoesNotExist:
        # Handle the case where the user does not exist
        user = None

    return render(request, 'manage_user.html', {'user': user})

def approve_faculty_view(request, faculty_id):
    # Make sure the block of code inside the function is indented properly
    faculty = get_object_or_404(Faculty, id=faculty_id)

    # Approve the faculty
    faculty.is_approved = True
    faculty.save()

    # Return a response (rendering a template in this case)
    return render(request, 'approve_faculty.html', {'faculty': faculty})

def reject_faculty(request, faculty_id):
    if not request.user.is_admin():
        return redirect('home')

    faculty = get_object_or_404(CustomUser, id=faculty_id)
    faculty.delete() # Remove faculty request
    messages.info(request, f"{faculty.username} has been rejected.")

```

```

return redirect('faculty_requests')

@login_required
def requests_view(request):
    if request.user.role not in ["Admin", "Superadmin"]: # Allow both roles
        return redirect("home")

    requests = FacultyRequest.objects.all() # Fetch actual requests
    return render(request, "requests.html", {"requests": requests})

def search_syllabus(request):
    print("✅ search_syllabus view called!") # Debugging

    regulation = request.GET.get("regulation", "")
    department = request.GET.get("department", "")
    year = request.GET.get("year", "")

    print(f"🔍 Received: regulation={regulation}, department={department}, year={year}")

    syllabi = Syllabus.objects.filter(
        regulation=regulation, department=department, year=year
    )

    if syllabi.exists():
        syllabus_data = [
            {
                "regulation": s.regulation,
                "department": s.department,
                "year": s.year,
                "preview_url": s.preview_url,
                "download_url": s.download_url
            }
            for s in syllabi
        ]
        print("✅ Returning JSON response with data!")
        return JsonResponse({"syllabi": syllabus_data}, safe=False)
    else:
        print("❌ No syllabus found!")
        return JsonResponse({"syllabi": []})

```



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Setting up App-level Urls
7. Date of Experiment : 24-01-2025
8. Date of Submission of Report : 31-01-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

urls.py:

urls.py defines all the URL routes in the project. It maps specific URLs (like /home/, /profile/, /syllabus/, etc.) to corresponding view functions, enabling navigation between different pages and features of the application.

Code:

```
from django.urls import path
from . import views
from .views import add_syllabus, search_syllabus

urlpatterns = [
    # Home Page
    path("", views.home, name='home'),
    path('home/', views.home, name='home'),

    # Books Page
    path('books/', views.books, name='books'),

    # Settings Page
    path('settings/', views.settings, name='settings'),

    # About Page
    path('about/', views.about, name='about'),

    # Feedback Page
    path('feedback/', views.feedback, name='feedback'),

    # Syllabus Page
    path('syllabus/', views.syllabus, name='syllabus'),

    # Login Page
    path('login/', views.login_view, name='login'),

    # Signup Page
    path('signup/', views.signup, name='signup'),

    # Profile Page (Login Required)
    path('profile/', views.profile, name='profile'),

    # Add Syllabus Page (Faculty Only)
    path('add_syllabus/', views.add_syllabus, name='add_syllabus'),

    # Faculty Requests Page (Admin Only)
    path('requests/', views.faculty_requests, name='faculty_requests'),

    path("add_syllabus/", views.add_syllabus, name="add_syllabus"),

    # Approve Faculty Page (Admin Only)
    path('approve_faculty/<int:faculty_id>/', views.approve_faculty, name='approve_faculty'),

    path('reject_faculty/<int:faculty_id>/', views.reject_faculty, name='reject_faculty'),
```

```
# Upload Syllabus Page (Admin Only)
path('upload/', views.upload, name='upload'),

# Logout
path('logout/', views.logout_view, name='logout'),

path("search-syllabus/", search_syllabus, name="search_syllabus"),

path('superadmin_dashboard/', views.superadmin_dashboard, name='superadmin_dashboard'),
path('manage_user/<int:user_id>', views.manage_user_view, name='manage_user'),
path('approve_faculty/<int:faculty_id>', views.approve_faculty_view, name='approve_faculty'),

]
```




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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: it@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Working with Templates in Django
7. Date of Experiment : 31-01-2025
8. Date of Submission of Report : 21-02-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Templates:

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

Where to Store Templates?

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

Folder Tree of the templates:

```
├── templates/ # HTML Templates
│   │   ├── base.html # Common Layout (Header, Sidebar, Footer)
│   │   ├── home.html # Home Page (Syllabus Filters, Most Visited)
│   │   ├── settings.html # Settings Page (Dark Mode, Preferences)
│   │   ├── about.html # About Us Page
│   │   ├── feedback.html # Feedback Page
│   │   ├── login.html # Login Page (Student & Faculty)
│   │   ├── signup.html # Signup Page (Student & Faculty)
│   │   ├── profile.html # Profile Page (Reapply for Faculty)
│   │   ├── upload.html # Upload Page (Admin & Faculty)
│   │   ├── requests.html # Faculty Approval Requests (For Superadmin)
│   │   ├── notifications.html # Admin Notification Page (For Faculty Requests)
│   │   └── admin_dashboard.html # Dashboard for Superadmin
```

Benefits:

- Separation of Concerns
- Maintainability
- Reusability
- Flexibility

home.html:

```
{% extends "base.html" %}

{% block title %}Home - Course Syllabus Repository{% endblock %}

{% block content %}
<style>
    /* Navigation Menu Styles */
.filter-container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    display: flex;
    gap: 15px;
    align-items: center;
    margin-bottom: 20px;
}

/* Dropdown & Button Styling */
.filter-container select,
.filter-container button {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
    color: black !important; /* Ensure text is visible */
    background: white; /* Ensure background contrast */
    appearance: none;
}

.filter-container button {
    background: #004080;
    color: white;
    cursor: pointer;
    transition: background 0.3s, transform 0.2s;
}

.filter-container button:hover {
    background: #0055aa;
    transform: scale(1.05);
}

/* Syllabus Display */
.syllabus-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 20px;
}
```

```

/* Syllabus Cards */
.syllabus-card {
  background: rgba(255, 255, 255, 0.9);
  backdrop-filter: blur(10px);
  padding: 15px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease-in-out;
}

.syllabus-card:hover {
  transform: scale(1.05);
}

/* Card Headings */
.syllabus-card h3 {
  margin: 0;
  font-size: 18px;
  color: #333; /* Darker text for better readability */
}

/* Button Container */
.syllabus-card .buttons {
  margin-top: 10px;
  display: flex;
  gap: 10px;
}

/* Card Buttons */
.syllabus-card button {
  padding: 8px 12px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 14px;
  font-weight: bold;
  transition: background 0.3s ease, transform 0.2s;
}

.syllabus-card button:hover {
  transform: scale(1.05);
}

/* Specific Button Colors */
.preview-btn {
  background: #ffaa00;
  color: black;
}

.download-btn {

```

```

background: #00aa00;
color: white;
}

</style>

<!-- Navigation Menu -->
<div class="filter-container">
  <select id="regulation">
    <option value="">Select Regulation</option>
    <option value="R17">R17</option>
    <option value="R20">R20</option>
    <option value="R23">R23</option>
  </select>
  <select id="department">
    <option value="">Select Department</option>
    <option value="CSE">CSE</option>
    <option value="IT">IT</option>
    <option value="ECE">ECE</option>
    <option value="EEE">EEE</option>
    <option value="MECH">MECH</option>
    <option value="CIVIL">CIVIL</option>
  </select>
  <select id="year">
    <option value="">Select Year</option>
    <option value="1st Year B.Tech">1st Year B.Tech</option>
    <option value="2nd Year B.Tech">2nd Year B.Tech</option>
    <option value="3rd Year B.Tech">3rd Year B.Tech</option>
    <option value="4th Year B.Tech">4th Year B.Tech</option>
  </select>
  <button onclick="filterSyllabus()">🔍 Search</button>
</div>

<!-- Syllabus Display -->
<div id="syllabus-list" class="syllabus-container">
  <!-- Default: Show all available syllabi -->
  <div class="syllabus-card" data-reg="R17" data-dept="CSE" data-year="1st Year
B.Tech">
    <h3>R17 - CSE - 1st Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>
    </div>
  </div>
  <div class="syllabus-card" data-reg="R20" data-dept="IT" data-year="2nd Year
B.Tech">
    <h3>R20 - IT - 2nd Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>

```

```

    </div>
  </div>
  <div class="syllabus-card" data-reg="R23" data-dept="ECE" data-year="3rd Year
B.Tech">
    <h3>R23 - ECE - 3rd Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>
    </div>
  </div>
  <div class="syllabus-card" data-reg="R17" data-dept="EEE" data-year="4th Year
B.Tech">
    <h3>R17 - EEE - 4th Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>
    </div>
  </div>
  <div class="syllabus-card" data-reg="R20" data-dept="MECH" data-year="1st Year
B.Tech">
    <h3>R20 - MECH - 1st Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>
    </div>
  </div>
  <div class="syllabus-card" data-reg="R23" data-dept="CIVIL" data-year="2nd Year
B.Tech">
    <h3>R23 - CIVIL - 2nd Year</h3>
    <div class="buttons">
      <button class="preview-btn">👁 Preview</button>
      <button class="download-btn">↓ Download</button>
    </div>
  </div>
</div>

<script>
  function filterSyllabus() {
    const reg = document.getElementById("regulation").value;
    const dept = document.getElementById("department").value;
    const year = document.getElementById("year").value;

    if (!reg || !dept || !year) {
      alert("Please select all filters!");
      return;
    }

    fetch(`/search-
syllabus/?regulation=${encodeURIComponent(reg)}&department=${encodeURIComponent

```

```

(dept)}&year=${encodeURIComponent(year)}`)
    .then(response => {
        if (!response.ok) {
            throw new Error("Network response was not ok");
        }
        return response.json();
    })
    .then(data => {
        const syllabusList = document.getElementById("syllabus-list");
        syllabusList.innerHTML = ""; // Clear previous results

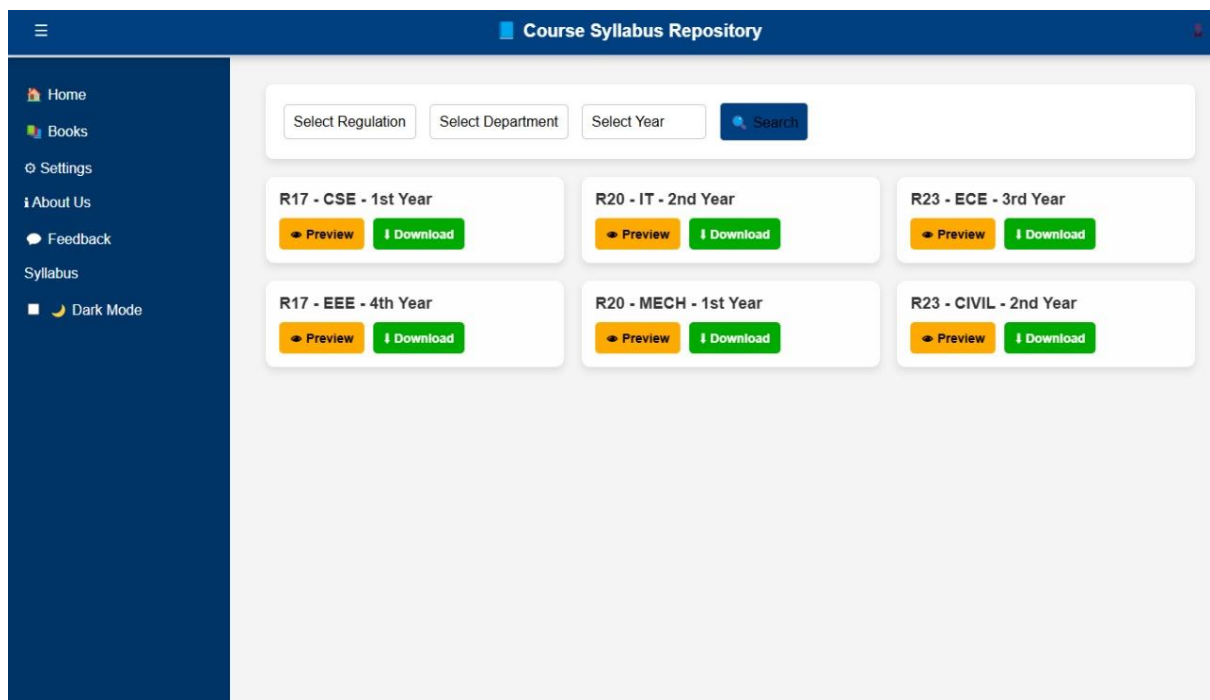
        if (data.syllabi && data.syllabi.length > 0) {
            data.syllabi.forEach(syllabus => {
                syllabusList.innerHTML += `
                    <div class="syllabus-card">
                        <h3>${syllabus.regulation} - ${syllabus.department} -
${syllabus.year}</h3>
                        <div class="buttons">
                            <button class="preview-btn"
onclick="window.open('${syllabus.preview_url}', '_blank')">👁 Preview</button>
                            <button class="download-btn"
onclick="window.location.href='${syllabus.download_url}'">↓ Download</button>
                        </div>
                    </div>`;
            });
        } else {
            syllabusList.innerHTML = `
                <p style="text-align: center; font-size: 18px; color: red;">
                    🚨 Syllabus not uploaded yet 🚨
                </p>
                <div class="missing-dialog">
                    <p>Want to request for missing syllabus?</p>
                    <button onclick="requestSyllabus(true)">✅ Yes</button>
                    <button onclick="requestSyllabus(false)">❌ No</button>
                </div>`;
        }
    })
    .catch(error => console.error("Error fetching syllabus:", error));
}

</script>

{% endblock %}

```

Output:



login.html:

```
{% extends 'base.html' %}

{% block title %}Login - Course Syllabus Repository{% endblock %}

{% block content %}
<style>
    .login-container {
        max-width: 400px;
        margin: auto;
        background: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        transition: background 0.3s, color 0.3s;
    }

    body.dark-mode .login-container {
        background: #1e1e1e;
        color: white;
    }

    h2 {
        text-align: center;
        color: #004080;
    }

    body.dark-mode h2 {
        color: #66b2ff;
    }

    .login-form {
        display: flex;
        flex-direction: column;
    }

    .login-form label {
        font-weight: bold;
        margin-top: 10px;
    }

    .login-form input,
    .login-form select {
        width: 100%;
        padding: 10px;
        margin-top: 5px;
        border: 1px solid #ccc;
        border-radius: 5px;
        transition: background 0.3s, color 0.3s;
    }
```

```

body.dark-mode .login-form input,
body.dark-mode .login-form select {
  background: #333;
  color: white;
  border: 1px solid #555;
}

.password-container {
  position: relative;
}

.password-container input {
  width: 100%;
  padding-right: 35px;
}

.toggle-password {
  position: absolute;
  right: 10px;
  top: 50%;
  transform: translateY(-50%);
  cursor: pointer;
  font-size: 18px;
  color: #555;
}

body.dark-mode .toggle-password {
  color: #bbb;
}

.login-btn {
  margin-top: 15px;
  padding: 10px;
  background: #004080;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
}

.login-btn:hover {
  background: #0055aa;
}

.extra-links {
  text-align: center;
  margin-top: 10px;
}

```

```

.extra-links a {
  text-decoration: none;
  color: #004080;
  font-weight: bold;
}

body.dark-mode .extra-links a {
  color: #66b2ff;
}

.popup {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
  text-align: center;
  display: none;
  width: 300px;
  z-index: 1000;
}

.popup button {
  margin-top: 10px;
  padding: 8px 15px;
  border: none;
  background: #004080;
  color: white;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
  margin: 5px;
}

.popup button:hover {
  background: #0055aa;
}

body.dark-mode .popup {
  background: #1e1e1e;
  color: white;
  box-shadow: 0 4px 10px rgba(255, 255, 255, 0.2);
}
</style>

<div class="login-container">

```

```

<h2>🔒 Login</h2>
{% if messages %}
  {% for message in messages %}
    <div class="popup" id="popup">
      <p>{{ message }}</p>
      <button onclick="closePopup()">OK</button>
    </div>
  {% endfor %}
{% endif %}

<form class="login-form" method="POST" action="{% url 'login' %}"
onsubmit="return validateLogin(event)">
  {% csrf_token %}
  <label for="role">Login As:</label>
  <select id="role" name="role">
    <option value="Student">Student</option>
    <option value="Faculty">Faculty</option>
    <option value="Admin">Admin</option>
  </select>

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

  <label for="password">Password:</label>
  <div class="password-container">
    <input type="password" id="password" name="password" required>
    <span class="toggle-password" onclick="togglePassword()">👁</span>
  </div>

  <button type="submit" class="login-btn">Login</button>
</form>

<div class="extra-links">
  <a href="/signup">Don't have an account? Sign up</a><br>
  <a href="/forgetpass">Forgot password?</a>
</div>
</div>

<script>
function togglePassword() {
  var passwordField = document.getElementById("password");
  passwordField.type = (passwordField.type === "password") ? "text" : "password";
}

function closePopup() {
  document.getElementById("popup").style.display = "none";
}

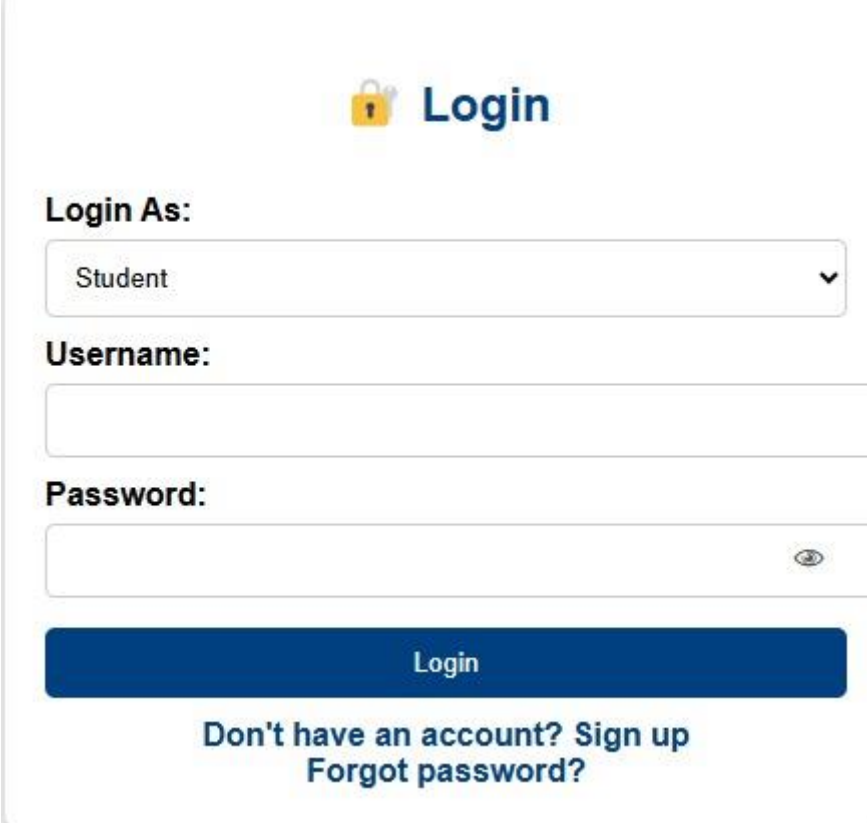
function validateLogin(event) {
  var username = document.getElementById("username").value;

```

```
var password = document.getElementById("password").value;

if (!username || !password) {
    alert("Please enter both username and password.");
    event.preventDefault();
    return false;
}
return true;
}
</script>
{% endblock %}
```

Output:



The image shows a login form with a light gray border. At the top center is a yellow padlock icon followed by the word "Login" in blue. Below this is the label "Login As:" followed by a dropdown menu showing "Student" with a downward arrow. Underneath is the label "Username:" followed by a text input field. Below that is the label "Password:" followed by a text input field with a small eye icon on the right. A large blue button with the text "Login" is positioned below the password field. At the bottom, the text "Don't have an account? Sign up" and "Forgot password?" are displayed in blue.

signup.html:

```
{% extends 'base.html' %}

{% block title %}Sign Up - Course Syllabus Repository{% endblock %}

{% block content %}
<style>
.signup-container {
    max-width: 400px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: background 0.3s, color 0.3s;
}

/* Dark Mode Support */
body.dark-mode .signup-container {
    background: #1e1e1e;
    color: white;
}

h2 {
    text-align: center;
    color: #004080;
}

body.dark-mode h2 {
    color: #66b2ff;
}

.signup-form {
    display: flex;
    flex-direction: column;
}

.signup-form label {
    font-weight: bold;
    margin-top: 10px;
}

.signup-form input,
.signup-form select {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: background 0.3s, color 0.3s;
}

body.dark-mode .signup-form input,
```

```

body.dark-mode .signup-form select {
  background: #333;
  color: white;
  border: 1px solid #555;
}

/* Password Field */
.password-container {
  position: relative;
}

.password-container input {
  width: 100%;
  padding-right: 35px;
}

.toggle-password {
  position: absolute;
  right: 10px;
  top: 50%;
  transform: translateY(-50%);
  cursor: pointer;
  font-size: 18px;
  color: #555;
}

body.dark-mode .toggle-password {
  color: #bbb;
}

.signup-btn {
  margin-top: 15px;
  padding: 10px;
  background: #004080;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
}

.signup-btn:hover {
  background: #0055aa;
}

.extra-links {
  text-align: center;
  margin-top: 10px;
}

.extra-links a {
  text-decoration: none;
  color: #004080;
  font-weight: bold;
}

```

```

body.dark-mode .extra-links a {
  color: #66b2ff;
}

/* Modal (Popup Box) */
.modal {
  display: none;
  position: fixed;
  z-index: 1000;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.5);
}

.modal-content {
  background: white;
  margin: 15% auto;
  padding: 20px;
  border-radius: 8px;
  width: 350px;
  text-align: center;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
}

body.dark-mode .modal-content {
  background: #1e1e1e;
  color: white;
}

.modal-buttons {
  margin-top: 15px;
  display: flex;
  justify-content: space-around;
}

.modal-buttons button {
  padding: 8px 12px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-weight: bold;
  transition: background 0.3s, transform 0.2s;
}

.confirm-btn {
  background: #28a745;
  color: white;
}

.cancel-btn {
  background: #dc3545;
  color: white;
}


```



```

    }

    .modal-buttons button:hover {
        transform: scale(1.05);
    }
</style>

<div class="signup-container">
    <h2>  Create an Account</h2>

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

    <form method="POST" action="{% url 'signup' %}">
        {% csrf_token %}


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

        <label for="email">Email:</label>
        <input type="email" name="email" required>

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

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

        <label for="role">Role:</label>
        <select name="role" required>
            <option value="Faculty">Faculty</option>
            {% if request.user.is_superuser %}
                <option value="Admin">Admin</option>
            {% endif %}
        </select>

        <button type="submit" class="signup-btn">  Sign Up</button>
    </form>

    <p>Already have an account? <a href="{% url 'login' %}">Login</a></p>
</div>

<!-- Custom Modal (Popup) -->
<div id="role-modal" class="modal">
    <div class="modal-content">
        <p id="modal-text">Faculty signups require admin approval. Do you want to send a
request?</p>
        <div class="modal-buttons">
            <button class="confirm-btn" onclick="submitForm()">Yes</button>

```

```

        <button class="cancel-btn" onclick="closeModal()">Cancel</button>
    </div>
</div>
</div>

<script>
    function togglePassword() {
        var passwordField = document.getElementById("password");
        passwordField.type = (passwordField.type === "password") ? "text" : "password";
    }

    function validateSignup(event) {
        event.preventDefault();
        var role = document.getElementById("role").value;

        if (role === "Faculty") {
            document.getElementById("modal-text").innerText = "Faculty signups require admin
approval. Do you want to send a request?";
            openModal();
        } else if (role === "Admin") {
            document.getElementById("modal-text").innerText = "Admin signups require super admin
approval. Do you want to send a request?";
            openModal();
        } else {
            submitForm();
        }
    }

    function openModal() {
        document.getElementById("role-modal").style.display = "block";
    }

S    function closeModal() {
        document.getElementById("role-modal").style.display = "none";
    }

    function submitForm() {
        document.getElementById("signup-form").submit();
    }
</script>
{% endblock %}

```

Output:



Create an Account

Username:	<input type="text"/>	Email:	<input type="text"/>
	<input type="text"/>	Password:	<input type="password"/>
	<input type="text"/>	Confirm Password:	<input type="password"/>
<input type="text"/>	Role:	<input type="text" value="Faculty"/>	<input type="button" value="Sign Up"/>

Already have an account? [Login](#)

profile.html:

```
{% extends 'base.html' %}

{% block title %}Profile - Course Syllabus Repository{% endblock %}

{% block content %}
{% if not user.is_authenticated %}
<script>
    setTimeout(function() {
        window.location.href = "/login";
    }, 2000);
</script>
<p>Redirecting to login...</p>
{% endif %}
<style>
    .profile-container {
        max-width: 500px;
        margin: auto;
        background: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        text-align: center;
        transition: background 0.3s, color 0.3s;
    }

    body.dark-mode .profile-container {
        background: #1e1e1e;
        color: white;
    }

    h2 {
        color: #004080;
    }

    body.dark-mode h2 {
        color: #66b2ff;
    }

    .profile-picture {
        width: 120px;
        height: 120px;
        border-radius: 50%;
        object-fit: cover;
        border: 3px solid #004080;
    }

    body.dark-mode .profile-picture {
        border-color: #66b2ff;
    }

    .profile-form {
```

```

margin-top: 20px;
display: flex;
flex-direction: column;
align-items: center;
}

.profile-form label {
  font-weight: bold;
  margin-top: 10px;
}

.profile-form input {
  width: 90%;
  padding: 10px;
  margin-top: 5px;
  border: 1px solid #ccc;
  border-radius: 5px;
  text-align: center;
  transition: background 0.3s, color 0.3s;
}

body.dark-mode .profile-form input {
  background: #333;
  color: white;
  border: 1px solid #555;
}

.edit-btn {
  margin-top: 15px;
  padding: 10px;
  background: #004080;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
}

.edit-btn:hover {
  background: #0055aa;
}

.save-btn {
  display: none;
  margin-top: 15px;
  padding: 10px;
  background: #008000;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
}

.save-btn:hover {

```

```

        background: #00aa00;
    }
</style>

<div class="profile-container">
    <h2>👤 My Profile</h2>

    <!-- Profile Picture -->
    
    <br>
    <input type="file" id="profile-upload" accept="image/*" style="display: none;"
onchange="updateProfilePicture()">
    <button class="edit-btn" onclick="document.getElementById('profile-upload').click();">📷
Change Photo</button>

    <!-- Profile Details -->
    <form class="profile-form">
        <label for="username">Username:</label>
        <input type="text" id="username" value="{{ user.username }}" disabled>

        <label for="email">Email:</label>
        <input type="email" id="email" value="{{ user.email }}" disabled>

        <label for="role">Role:</label>
        <input type="text" id="role" value="{{ user.role }}" disabled>

        <button type="button" class="edit-btn" onclick="enableEditing()">✎ Edit Profile</button>
        <button type="submit" class="save-btn" onclick="saveChanges(event)">💾 Save
Changes</button>
    </form>
</div>

<script>
    function enableEditing() {
        document.getElementById('username').disabled = false;
        document.getElementById('email').disabled = false;
        document.querySelector('.edit-btn').style.display = 'none';
        document.querySelector('.save-btn').style.display = 'block';
    }

    function saveChanges(event) {
        event.preventDefault();
        alert("Profile updated successfully! (Backend integration required)");
    }

    function updateProfilePicture() {
        var fileInput = document.getElementById('profile-upload');
        var profilePic = document.getElementById('profile-pic');


        if (fileInput.files && fileInput.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                profilePic.src = e.target.result;
            };
            reader.readAsDataURL(fileInput.files[0]);
        }
    }
</script>


```


```
};  
reader.readAsDataURL(fileInput.files[0]);  
}  
</script>
```

{% endblock %}

Output:

 **My Profile**


 Profile Picture

 Change Photo

Username:

Email:

Role:

 Edit Profile

settings.html:

```
{% extends "base.html" %}
```

```
{% block title %}Settings - Course Syllabus Repository{% endblock %}
```

```
{% block content %}
```

```
<style>
```

```
.settings-container {  
    max-width: 600px;  
    margin: 40px auto;  
    padding: 20px;  
    background: rgba(255, 255, 255, 0.8);  
    border-radius: 10px;  
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);  
    animation: fadeIn 0.8s ease-in-out;  
}
```

```
.settings-container h2 {  
    text-align: center;  
    color: #004080;  
}
```

```
.setting-option {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    margin: 15px 0;  
    font-size: 18px;  
}
```

```
.toggle-switch {  
    position: relative;  
    width: 50px;
```



```
        height: 24px;
    }
    .toggle-switch input {
        opacity: 0;
        width: 0;
        height: 0;
    }
    .slider {
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: #ccc;
        border-radius: 34px;
        transition: 0.4s;
    }
    .slider::before {
        content: "";
        position: absolute;
        height: 18px;
        width: 18px;
        left: 3px;
        bottom: 3px;
        background-color: white;
        border-radius: 50%;
        transition: 0.4s;
    }
    input:checked + .slider {
        background-color: #004080;
```

```

}
input:checked + .slider::before {
    transform: translateX(26px);
}
.save-btn {
    display: block;
    width: 100%;
    padding: 10px;
    margin-top: 20px;
    font-size: 18px;
    text-align: center;
    background-color: #004080;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s;
}
.save-btn:hover {
    background-color: #002d5a;
}
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

```

```

    }
</style>

<div class="settings-container">
    <h2> ⚙ Settings</h2>

    <div class="setting-option">
        <span>Dark Mode</span>
        <label class="toggle-switch">
            <input type="checkbox" id="darkModeToggle">
            <span class="slider"></span>
        </label>
    </div>

    <div class="setting-option">
        <span>Enable Notifications</span>
        <label class="toggle-switch">
            <input type="checkbox" id="notificationsToggle">
            <span class="slider"></span>
        </label>
    </div>

    <button class="save-btn">Save Settings</button>
</div>

<script>
    document.addEventListener("DOMContentLoaded", function() {
        let darkModeToggle = document.getElementById("darkModeToggle");

        // Load saved preference

```

```
if (localStorage.getItem("darkMode") === "enabled") {  
    document.body.classList.add("dark-mode");  
    darkModeToggle.checked = true;  
}  
  
// Toggle Dark Mode & Save Preference  
darkModeToggle.addEventListener("change", function() {  
    if (this.checked) {  
        document.body.classList.add("dark-mode");  
        localStorage.setItem("darkMode", "enabled");  
    } else {  
        document.body.classList.remove("dark-mode");  
        localStorage.setItem("darkMode", "disabled");  
    }  
});  
});  
</script>
```

```
{% endblock %}
```

Output:



upload.html:

```
{% extends 'base.html' %}
```

```
{% block title %}Upload Syllabus - Course Syllabus Repository{% endblock %}
```

```
{% block content %}
```

```
    {% if not user.is_authenticated or user.role != "Admin" %}
```

```
        <script>
```

```
            window.location.href = "/login";
```

```
        </script>
```

```
    {% endif %}
```

```
<style>
```

```
    .upload-container {
```

```
        max-width: 600px;
```

```
        margin: auto;
```

```
        background: white;
```

```
        padding: 20px;
```

```
        border-radius: 8px;
```

```
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
```

```
        text-align: center;
```

```
        transition: background 0.3s, color 0.3s;
```

```
    }
```

```
    body.dark-mode .upload-container {
```

```
        background: #1e1e1e;
```

```
        color: white;
```

```
    }
```

```
    h2 {
```

```
    color: #004080;
}
```

```
body.dark-mode h2 {
    color: #66b2ff;
}
```

```
.upload-form {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 10px;
}
```

```
.upload-form label {
    font-weight: bold;
}
```

```
.upload-form input, .upload-form select {
    width: 90%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: background 0.3s, color 0.3s;
}
```

```
body.dark-mode .upload-form input,
body.dark-mode .upload-form select {
    background: #333;
    color: white;
}
```

```
border: 1px solid #555;
}

.upload-btn {
margin-top: 15px;
padding: 10px;
background: #008000;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
transition: 0.3s;
}

.upload-btn:hover {
background: #00aa00;
}

/* Modal (Popup Box) */
.modal {
display: none;
position: fixed;
z-index: 1000;
left: 0;
top: 0;
width: 100%;
height: 100%;
background: rgba(0, 0, 0, 0.5);
}
```



```
.modal-content {  
  background: white;  
  margin: 15% auto;  
  padding: 20px;  
  border-radius: 8px;  
  width: 350px;  
  text-align: center;  
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);  
}
```

```
body.dark-mode .modal-content {  
  background: #1e1e1e;  
  color: white;  
}
```

```
.modal-buttons {  
  margin-top: 15px;  
  display: flex;  
  justify-content: space-around;  
}
```

```
.modal-buttons button {  
  padding: 8px 12px;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
  font-weight: bold;  
  transition: background 0.3s, transform 0.2s;  
}
```

```
.confirm-btn {  
    background: #28a745;  
    color: white;  
}
```


```
.cancel-btn {  
    background: #dc3545;  
    color: white;  
}
```

```
.modal-buttons button:hover {  
    transform: scale(1.05);  
}
```

</style>

<!-- Upload Form -->

<div class="upload-container">

<h2>  Upload Syllabus</h2>

<form method="POST" enctype="multipart/form-data" onsubmit="return
validateUploadForm(event)">

{% csrf_token %}

<label for="course-name">Course Name:</label>

<input type="text" id="course-name" name="course_name" required>

<label for="department">Department:</label>

<select id="department" name="department" required>

<option value="CSE">CSE</option>

<option value="IT">IT</option>

<option value="ECE">ECE</option>

<option value="EEE">EEE</option>

```

        <option value="MECH">MECH</option>
        <option value="CIVIL">CIVIL</option>
    </select>

    <label for="regulation">Regulation:</label>
    <select id="regulation" name="regulation" required>
        <option value="R17">R17</option>
        <option value="R20">R20</option>
        <option value="R23">R23</option>
    </select>

    <label for="syllabus-file">Upload Syllabus (PDF only):</label>
    <input type="file" id="syllabus-file" name="syllabus_file" accept=".pdf" required>

    <button type="submit" class="upload-btn">📁 Upload</button>
</form>
</div>

<!-- Modal (for feedback) -->
<div id="upload-modal" class="modal">
    <div class="modal-content">
        <p id="modal-text">Your syllabus has been uploaded successfully.</p>
        <div class="modal-buttons">
            <button class="confirm-btn" onclick="closeModal()">Ok</button>
        </div>
    </div>
</div>
</div>

<script>
    // Function to open modal

```

```
function openModal() {
    document.getElementById("upload-modal").style.display = "block";
}

// Function to close modal
function closeModal() {
    document.getElementById("upload-modal").style.display = "none";
}

// Form validation and submission
function validateUploadForm(event) {
    event.preventDefault();
    let form = event.target;

    // Here you can add form validation logic if necessary
    // For now, we simulate success and trigger modal display
    openModal();

    // After showing the modal, you would handle the actual backend upload
    // In a real scenario, you would submit the form here.
    // form.submit(); // Uncomment this line after adding backend upload functionality
}

</script>
{% endblock %}
```

Output:

feedback.html:

```
{% extends 'base.html' %}
```

```
{% block title %}Feedback - Course Syllabus Repository{% endblock %}
```

```
{% block content %}
```

```
<style>
```

```
.feedback-container {  
    max-width: 600px;  
    margin: auto;  
    background: white;  
    padding: 20px;  
    border-radius: 8px;  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);  
    transition: background 0.3s, color 0.3s;  
}
```

```
/* Dark Mode Styling */
```

```
body.dark-mode .feedback-container {  
    background: #1e1e1e;  
    color: white;  
}
```

```
h2 {  
    text-align: center;  
    color: #004080;  
}
```

```
body.dark-mode h2 {  
    color: #66b2ff;
```

```
}
```

```
.feedback-form {  
  display: flex;  
  flex-direction: column;  
}
```

```
.feedback-form label {  
  font-weight: bold;  
  margin-top: 10px;  
}
```

```
.feedback-form input,  
.feedback-form select,  
.feedback-form textarea {  
  width: 100%;  
  padding: 10px;  
  margin-top: 5px;  
  border: 1px solid #ccc;  
  border-radius: 5px;  
  transition: background 0.3s, color 0.3s;  
}
```

```
body.dark-mode .feedback-form input,  
body.dark-mode .feedback-form select,  
body.dark-mode .feedback-form textarea {  
  background: #333;  
  color: white;  
  border: 1px solid #555;  
}
```

```
.submit-btn {  
    margin-top: 15px;  
    padding: 10px;  
    background: #004080;  
    color: white;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
    transition: 0.3s;  
}
```

```
.submit-btn:hover {  
    background: #0055aa;  
}
```

```
.success-message {  
    text-align: center;  
    color: green;  
    font-weight: bold;  
    margin-top: 10px;  
    display: none;  
}
```

</style>

<div class="feedback-container">

<h2>🗨 Submit Your Feedback</h2>

<form class="feedback-form" onsubmit="submitFeedback(event)">

<label for="name">Name:</label>


```
<input type="text" id="name" name="name" required>
```

```
<label for="email">Email:</label>
```

```
<input type="email" id="email" name="email" required>
```

```
<label for="category">Category:</label>
```

```
<select id="category" name="category">
```

```
  <option value="Suggestion">Suggestion</option>
```

```
  <option value="Bug Report">Bug Report</option>
```

```
  <option value="General Inquiry">General Inquiry</option>
```

```
</select>
```

```
<label for="message">Message:</label>
```

```
<textarea id="message" name="message" rows="4" required></textarea>
```

```
<button type="submit" class="submit-btn">Submit Feedback</button>
```

```
</form>
```

```
<p class="success-message" id="successMessage">✅ Thank you for your  
feedback!</p>
```

```
</div>
```

```
<script>
```

```
function submitFeedback(event) {
```

```
  event.preventDefault();
```

```
  document.getElementById('successMessage').style.display = 'block';
```

```
  setTimeout(() => {
```

```
    document.getElementById('successMessage').style.display = 'none';
```


```
  }, 3000);
```

```
}
```

```
</script>
```

{% endblock %}

Output:



Submit Your Feedback

Name:

Email:

Category:

Suggestion

Message:

Submit Feedback

about.html:

```
{% extends 'base.html' %}

{% block title %}About Us - Course Syllabus Repository{% endblock %}

{% block content %}
<style>
    .about-container {
        max-width: 800px;
        margin: auto;
        background: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        transition: background 0.3s, color 0.3s;
    }

    /* Dark Mode Styling */
    body.dark-mode .about-container {
        background: #1e1e1e;
        color: white;
    }

    h2 {
        text-align: center;
        color: #004080;
    }

    body.dark-mode h2 {
        color: #66b2ff;
    }

    .about-section {
        margin-bottom: 20px;
    }

    .contact {
        text-align: center;
        font-weight: bold;
    }
</style>

<div class="about-container">
    <h2>About Us</h2>

    <div class="about-section">
        <p>
            Welcome to the **Course Syllabus Repository**, your one-stop platform for accessing and
            managing course syllabi.
            We aim to provide students and faculty with an organized and efficient way to find
            academic resources.
        </p>
    </div>
</div>
```

```

<div class="about-section">
  <h3>✦ Our Mission</h3>
  <p>
    Our mission is to simplify academic planning by offering easy access to updated syllabi.
    We are dedicated to improving student learning experiences and faculty efficiency.
  </p>
</div>

<div class="about-section">
  <h3>☎ Contact Us</h3>
  <p class="contact">✉ Email: support@syllabusrepo.com</p>
  <p class="contact">📍 Location: Your College, City, Country</p>
</div>
</div>
{% endblock %}

```

Output:

About Us

Welcome to the ****Course Syllabus Repository****, your one-stop platform for accessing and managing course syllabi. We aim to provide students and faculty with an organized and efficient way to find academic resources.

✦ Our Mission

Our mission is to simplify academic planning by offering easy access to updated syllabi. We are dedicated to improving student learning experiences and faculty efficiency.

☎ Contact Us

✉ Email: support@syllabusrepo.com

📍 Location: Your College, City, Country

requests.html:

```
{% extends 'base.html' %}

{% block title %}Requests - Course Syllabus Repository{% endblock %}

{% block content %}
    {% if not user.is_authenticated or user.role not in ["Admin", "Superadmin"] %}
        <script>
            window.location.href = "/login";
        </script>
    {% endif %}

    <style>
        .requests-container {
            max-width: 800px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            text-align: center;
            transition: background 0.3s, color 0.3s;
        }

        body.dark-mode .requests-container {
            background: #1e1e1e;
            color: white;
        }

        h2 {
            color: #004080;
        }

        body.dark-mode h2 {
            color: #66b2ff;
        }

        .requests-table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }

        .requests-table th, .requests-table td {
            padding: 10px;
            border: 1px solid #ddd;
            text-align: center;
        }

        body.dark-mode .requests-table th,
        body.dark-mode .requests-table td {
            border-color: #555;
        }
    </style>
</block>
```

```

.requests-table th {
  background: #004080;
  color: white;
}

body.dark-mode .requests-table th {
  background: #002a5c;
}


.action-btn {
  padding: 8px 12px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: 0.3s;
}

.approve-btn {
  background: #008000;
  color: white;
}

.approve-btn:hover {
  background: #00aa00;
}

.reject-btn {
  background: #cc0000;
  color: white;
}

.reject-btn:hover {
  background: #ff3333;
}
</style>

<div class="requests-container">
  <h2>  Pending Faculty Requests</h2>

  {% if requests %}
    <table class="requests-table">
      <thead>
        <tr>
          <th>Request ID</th>
          <th>User</th>
          <th>Course</th>
          <th>Department</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        {% for request in requests %}
          <tr>
            <td>{{ request.id }}</td>

```

```

        <td>{{ request.user.username }}</td>
        <td>{{ request.course.name }}</td>
        <td>{{ request.department }}</td>
        <td>
            <form method="POST" action="{% url 'approve_faculty' request.id %}"
style="display:inline;">
                {% csrf_token %}
                <button type="submit" class="action-btn approve-btn">✅ Approve</button>
            </form>
            <form method="POST" action="{% url 'reject_faculty' request.id %}"
style="display:inline;">
                {% csrf_token %}
                <button type="submit" class="action-btn reject-btn">❌ Reject</button>
            </form>
        </td>
    </tr>
    {% endfor %}
</tbody>
</table>
{% else %}
    <p>No pending requests.</p>
{% endif %}
</div>

{% endblock %}

```

Output:

superadmin_dashboard.html:

```
{% extends "base.html" %}

{% block title %}Superadmin Dashboard - Course Syllabus Repository{% endblock %}

{% block content %}
<style>
    .dashboard-container {
        margin-top: 20px;
    }

    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
    }

    th, td {
        padding: 10px;
        text-align: left;
        border: 1px solid #ddd;
    }

    th {
        background: #004080;
        color: white;
    }

    tr:hover {
        background: #f1f1f1;
    }

    .actions-btns {
        display: flex;
        gap: 10px;
    }

    .action-btn {
        padding: 8px 12px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        font-weight: bold;
        transition: background 0.3s, transform 0.2s;
    }

    .promote-btn {
        background: #ffc107;
        color: white;
    }

    .remove-btn {
        background: #dc3545;
```



```

        color: white;
    }

    .action-btn:hover {
        transform: scale(1.05);
    }

    h2 {
        font-size: 24px;
        color: #333;
    }

    body.dark-mode h2 {
        color: #66b2ff;
    }
</style>

<h2>👑 Superadmin Dashboard</h2>

<div class="dashboard-container">
    <table>
        <tr>
            <th>Username</th>
            <th>Role</th>
            <th>Actions</th>
        </tr>
        {% for user in users %}
            <tr>
                <td>{{ user.username }}</td>
                <td>{{ user.role }}</td>
                <td>
                    <div class="actions-btns">
                        {% if user.role == "Faculty" %}
                            <form method="POST" action="{% url 'promote_user' user.id %}">
                                {% csrf_token %}
                                <button type="submit" class="action-btn promote-btn">🚀 Promote to
Admin</button>
                            </form>
                        {% endif %}
                            <form method="POST" action="{% url 'remove_user' user.id %}">
                                {% csrf_token %}
                                <button type="submit" class="action-btn remove-btn">✖ Remove</button>
                            </form>
                        </div>
                    </td>
                </tr>
            {% endfor %}
        </table>
    </div>
{% endblock %}

```

Output:

Superadmin Dashboard

Username	Role	Actions
----------	------	---------



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Database Integration
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 07-03-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Database Integration:

A **database** is an organized collection of data stored and managed in a structured format. It allows for efficient access, retrieval, updating, and management of information. In web development, databases are essential for storing user data, application states, and more.

SQLite3 Installation & Setup

Step 1: Check if SQLite3 is Already Installed. Open your terminal or command prompt and run:

```
sqlite3 --version
```

Step 2: Install SQLite3 (If Not Installed)

On Windows:

1. Download the SQLite3 command-line tool from the official SQLite Downloads page.
2. Scroll down to "**Precompiled Binaries for Windows**", and download the ZIP file.
3. Extract the ZIP file and place sqlite3.exe in a folder, e.g., C:\sqlite.
4. Add the folder to your system's **PATH** environment variable:
 - Search for "Environment Variables" in the Start menu.
 - Click on **Environment Variables**.
 - In **System Variables**, select Path and click **Edit**.
 - Click **New**, add the path to the folder (e.g., C:\sqlite), and click OK.

On macOS/Linux:

Most macOS and Linux distributions come with SQLite pre-installed. Verify with:

```
sqlite3 --version
```

If it's not installed, use a package manager like brew (macOS) or apt (Ubuntu):

```
# macOS
```

```
brew install sqlite
```

```
# Ubuntu/Linux
```

```
sudo apt-get install sqlite3
```

Step 3: Using SQLite3 with Django

Django uses SQLite3 as its **default database**, so you don't need to install any additional drivers. Just make sure your settings.py file has the default database configuration:

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

Step 4: Create the Database with Migrations

After configuring the settings, run the following command to apply the initial migrations and create the database:

```
python manage.py migrate
```

Interacting with the SQLite Database

Step 1: Open the Django DB Shell

Make sure your virtual environment is activated, then run:

```
python manage.py dbshell
```

Step 2: Common SQLite Commands

- **List All Tables:**

```
.tables
```

- **View Table Schema:**

```
.schema table_name;
```

- **Show All Data in a Table:**

```
SELECT * FROM table_name;
```

- **Exit the SQLite Shell:**

```
.exit
```



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Forms in Django
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 07-03-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

What is forms.py in Django?

In Django, forms.py is a Python file where you define **form classes** to handle user input efficiently and securely. It allows you to create web forms without manually writing raw HTML or custom validation logic.

Django provides two main ways to create forms:

- **forms.Form** – For custom or standalone forms.
- **forms.ModelForm** – For forms that interact directly with Django models.

Why Use forms.py?

Simplifies Form Creation

Easily build and customize forms using Python classes instead of writing repetitive HTML form code.

Automatic Input Validation

Django automatically validates user inputs (e.g., checking required fields, email formats, etc.) and gives useful error messages.

Integrates with Django Models


When using ModelForm, forms are tightly connected to Django models, enabling easy creation and updating of database records.

Prevents Security Risks

- Built-in protection against **SQL Injection**.
- Built-in support for **CSRF (Cross-Site Request Forgery)** protection with `{% csrf_token %}` in templates.

Code:

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User
from .models import CustomUser, Syllabus # Import CustomUser and Syllabus models
from django.core.exceptions import ValidationError
```

```
#  User Registration Form
class CustomUserCreationForm(UserCreationForm):
    ROLE_CHOICES = [
        ('Admin', 'Admin'),
        ('Faculty', 'Faculty'),
```

```

        ('Student', 'Student'),
    ]

    email = forms.EmailField(required=True)
    role = forms.ChoiceField(choices=ROLE_CHOICES, required=True)

    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password1', 'password2', 'role']

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

# ✅ User Login Form
class LoginForm(AuthenticationForm):
    username = forms.CharField(label="Username", widget=forms.TextInput(attrs={'class': 'form-control'}))
    password = forms.CharField(label="Password", widget=forms.PasswordInput(attrs={'class': 'form-control'}))

# ✅ Syllabus Upload Form
class SyllabusForm(forms.ModelForm):
    class Meta:
        model = Syllabus
        fields = ['course_name', 'department', 'regulation', 'syllabus_file'] # Corrected to match the model's field name

    def clean_syllabus_file(self):
        file = self.cleaned_data.get('syllabus_file')
        if file:
            if not file.name.endswith('.pdf'):
                raise ValidationError("Only PDF files are allowed.")
        return file

class CustomSignupForm(forms.ModelForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'password']

class CustomLoginForm(forms.Form):
    username = forms.CharField(max_length=100)
    password = forms.CharField(widget=forms.PasswordInput)

```




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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: it@intugvcev.edu.in

1. Name of the Laboratory : Django Frameworks Lab
2. Name of the Student : Mohammad Arif
3. Roll No : 23VV1A1233
4. Class : II B-Tech II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Defining Models in Django
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 07-03-2025

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

What is models.py in Django?

In Django, models.py is the file where you define your **database structure** using Python classes. Each class in this file represents a **database table**, and each attribute of the class represents a **field (column)** in that table.

Django models serve as a **bridge between the application and the database**, enabling you to perform CRUD operations—**Create, Read, Update, and Delete**—without writing raw SQL queries.

Why Use Django Models?

No Need for Raw SQL

You can interact with the database using Python code instead of manually writing SQL.

Automatic Table Creation

Django can automatically generate database tables based on your model definitions.

Built-in Validation and Relationships

Models can include field validation and support relationships like **OneToOne**, **ForeignKey**, and **ManyToMany**.

Seamless Integration with Django Admin

Once a model is created, it can be managed through the Django admin panel with minimal setup.

Code:

```
from django.contrib.auth.models import AbstractUser, Group, Permission
```

```
from django.db import models
```

```
from django.core.exceptions import ValidationError
```

```
#  Custom User Model
```

```
class CustomUser(AbstractUser):
```

```
    ROLE_CHOICES = [
```

```
        ('Superadmin', 'Superadmin'),
```

```
        ('Admin', 'Admin'),
```

```
        ('Faculty', 'Faculty'),
```

```

        ('Student', 'Student'),
        ('Pending Faculty', 'Pending Faculty'),
    ]

    role = models.CharField(max_length=20, choices=ROLE_CHOICES, default='Student')
    is_approved = models.BooleanField(default=False) # For faculty approval system
    is_active = models.BooleanField(default=True) # For activation control (relevant for
    faculty)

    def __str__(self):
        return f"{self.username} ({self.role})"

    # Custom methods for checking roles
    def is_superuser(self):
        return self.role == 'Superadmin'

    def is_admin(self):
        return self.role == 'Admin'

    def is_faculty(self):
        return self.role == 'Faculty'

# ✅ Syllabus Model
class Syllabus(models.Model):
    course_name = models.CharField(max_length=255)
    department = models.CharField(max_length=100)
    regulation = models.CharField(max_length=10)
    syllabus_file = models.FileField(upload_to='syllabi/') # The file path for the syllabus
    uploaded_by = models.ForeignKey(CustomUser, on_delete=models.CASCADE,
    limit_choices_to={'role': 'Faculty'})

```

```
    uploaded_at = models.DateTimeField(auto_now_add=True) # Automatically stores the
upload time
```

```
    description = models.TextField(blank=True, null=True) # Optional field to describe the
syllabus
```

```
    active = models.BooleanField(default=True) # Whether the syllabus is still active or used
```

```
def clean(self):
```

```
    # Custom validation for the file to be a PDF
```

```
    if self.syllabus_file:
```

```
        if not self.syllabus_file.name.endswith('.pdf'):
```

```
            raise ValidationError("Only PDF files are allowed.")
```

```
def __str__(self):
```

```
    return f"{self.course_name} - {self.department} - {self.regulation}
({self.uploaded_by.username})"
```

```
#  Faculty Model (moved outside Syllabus model)
```

```
class Faculty(models.Model):
```

```
    name = models.CharField(max_length=100)
```

```
    email = models.EmailField(unique=True)
```

```
    is_approved = models.BooleanField(default=False)
```

```
def __str__(self):
```

```
    return self.name
```

```
class FacultyRequest(models.Model):
```

```
    user = models.ForeignKey('CustomUser', on_delete=models.CASCADE) # Replace
'CustomUser' if using another model
```

```
    course = models.CharField(max_length=255)
```

```
    department = models.CharField(max_length=100)
```

```
    is_approved = models.BooleanField(default=False)
```

```
def __str__(self):
```

```
    return f"Request from {self.user.username} for {self.course}"
```



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|------------------------------|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Migrations Syn with Django |
| 7. Date of Experiment | : 07-03-2025 |
| 8. Date of Submission of Report | : 27-03-2025 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Run Migrations to Create Database Tables in Django

After defining or modifying your models in `models.py`, Django requires two main steps to apply those changes to the database: creating migrations and applying them.

Step 1: Create Migrations

```
python manage.py makemigrations
```

This command scans all installed apps for changes in model definitions. If any changes are detected, it generates migration files inside each app's `migrations/` directory. These migration files are Python scripts that contain instructions for modifying the database schema based on your models.

This step does **not** affect the database directly—it only prepares the instructions.

Step 2: Apply Migrations

```
python manage.py migrate
```

This command applies all unapplied migration files to the actual database. Django will create or alter the necessary tables and columns according to the migration instructions.

This step updates the database schema to match the current state of your model definitions

.



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|---|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Deploying Django Application on Cloud |
| 7. Date of Experiment | : 27-03-2025 |
| 8. Date of Submission of Report | : 04-04-2025 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Deploying Django Web Application on Cloud

What is Deployment?

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

Features:

- Scalability – Handle more users without performance issues.
- Security – Protect user data with SSL and secure databases.
- Global Accessibility – Users can access your app from anywhere.
- Continuous Deployment – Easily update your app with new features.

Step 1: Register on GitHub

- Go to GitHub and click Sign up.
- Enter your Username, Email, and Password.
- Complete the verification and click Create Account.
- Verify your email by clicking the link in your inbox.

Step 2: Push to GitHub

Initialize Git in your project

```
git init
```

Connect to GitHub

```
git remote add origin
```

```
https://github.com/haridammu/Classroom_Booking_system.git
```

Add and commit changes

```
git add .
```

```
git commit -m "Initial Commit: Login and Registration App"
```

Push to GitHub

```
git branch -M main
```

```
git push -u origin main
```

- Your code is now available on GitHub.

GITHUB LINK:

<https://github.com/livngcorpse/course.git>



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

Dr.Ch. Bindu Madhuri
Asst. Professor & HOD

Email: hodit@intugvcev.edu.in

- | | |
|---------------------------------|----------------------------------|
| 1. Name of the Laboratory | : Django Frameworks Lab |
| 2. Name of the Student | : Mohammad Arif |
| 3. Roll No | : 23VV1A1233 |
| 4. Class | : II B-Tech II Semester |
| 5. Academic Year | : 2024-25 |
| 6. Name of Experiment | : Frontend Developer Certificate |
| 7. Date of Experiment | : 04-04-2025 |
| 8. Date of Submission of Report | : 04-04-2025 |

S. No	Ability And Activity	Weightage Of Marks	Day To Day Evaluation Score
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams, Architecture, Workflow, Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date:

Signature of faculty

Frontend Developer Certificate:



CERTIFICATE OF ACHIEVEMENT



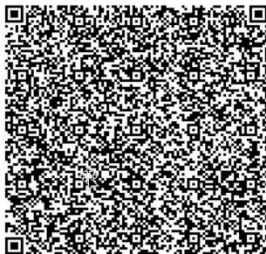
The certificate is awarded to

MOHAMMAD ARIF

for successfully completing

Front End Web Developer Certification

on January 31, 2025



Congratulations! You make us proud!

Issued on: Thursday, March 6, 2025
To verify, scan the QR code at <https://verify.onwingspan.com>

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

