DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)
VIZIANAGARAM ANDHRA PRADESH

DJANGO FRAME WORK LAB

UNIVERSITY TIMETABLE GENERATOR

DONE BY

BATCH-2

D.LAVANYA	**23VV1A1211**
G. NAGA LAXM	23VV1A1216
B. HEMA SAGAR	23VV1A1206
M. VIJAY KUMAR	23VV1A1228

UNDER GUIDANCE OF

MRS.MADHUMITA CHANDA

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Regd.No:23VV1A1211

CERTIFICATE

This is to certify that this is a bonafide record of practical work done by Mr/Mrs: D. LAVANYA of IInd B.Tech, IInd Semester Class in Django Framework Lab during the year 2024-25.

No.of Tasks Completed and Certified:13

Lecture In Charge **Head Of The Department**

Date:

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Website:www.jntugvcev.edu.in

Subject Name: DJANGO FRAMEWORK **Regulation: R23**

Subject Code: R2322BSH01 **Academic Year: 2025**

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

INDEX

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Understanding Django and Its Libraries
7. Date of Experiment : 13-12-2024
8. Date of Submission of Report : 20-12-2024

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Python Libraries:

1.Python Collections - Container Datatypes:

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

Key Types:

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

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

2.Tkinter – GUI Applications:

a) Purpose:

Python's standard library for Creating graphical user interfaces (GUIS).

b) key features:

widgets: Buttons, labels, text boxes, etc.
Event handling: Respond to user interactions like clicks on keypresses.
Simple layout management.

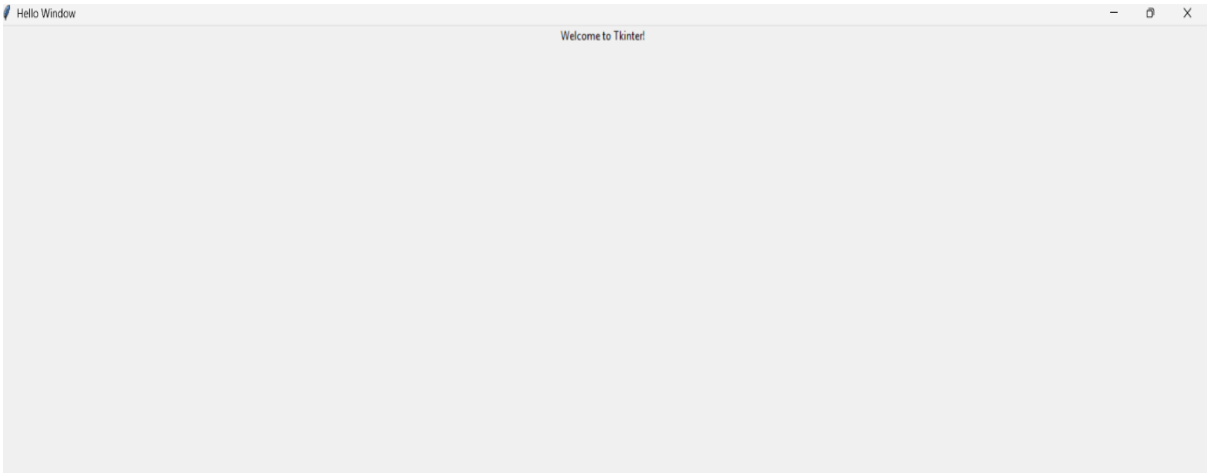
c) Common use:

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

Code:

```
#pip install tkinter
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. main loop ()
```

Output:



2. Requests-HTTP Requests:

a) Purpose: Simplifies HTTP requests to interact with web APIS.

b) key-features:

1.Send GET, POST, PUT, DELETE requests easily.
2.Handle requests parameters, headers, and Cookies.
3.Simple error handling and responsible handling.

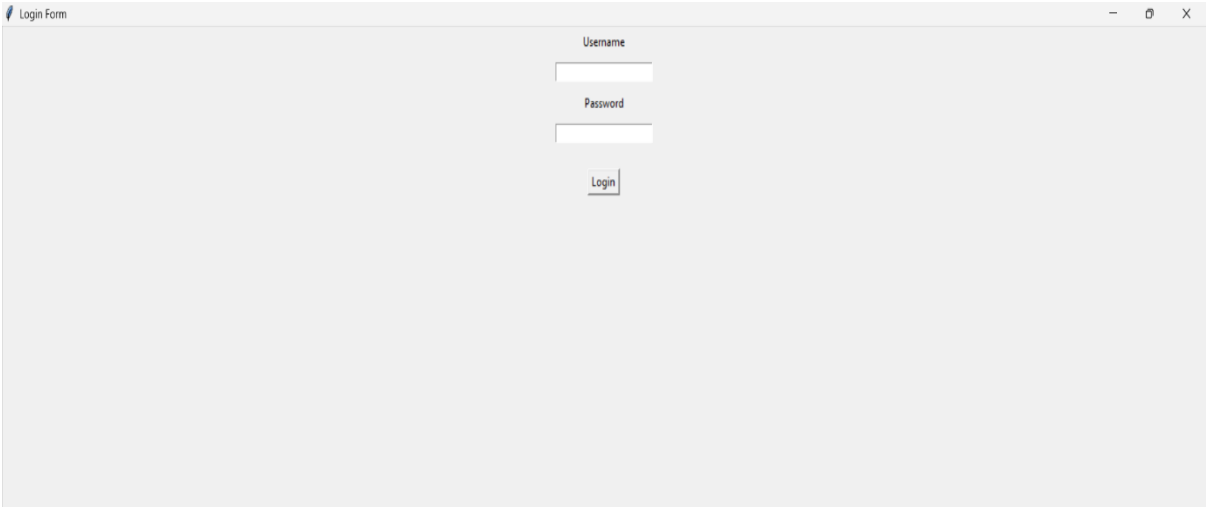
c) **Common use:** Interact with REST APIS, download content from the web.

Code:

```
(pady=5)
#pip install tk
import tkinter as tk
from tkinter import message box
# Function to validate login
def validate_login():
    username = username_entry.get()
    password = password_entry.get()
# Example credentials
    if username == "user" and password == "password":
        messagebox.showinfo("Login Success", "Login Successful!")
    else:
messagebox.showerror("Login Failed", "Invalid username or
password")
  # Create the main window
root = tk.Tk()
root.title("Login Form")
root.geometry("1080x720")
  # Create username and password labels and entry widgets
username_label = tk.Label(root, text="Username")
username_label.pack
username_entry = tk.Entry(root)
username_entry.pack(pady=5)
```

```
password_label = tk.Label(root, text="Password")

password_label.pack(pady=5)

password_entry = tk.Entry(root, show="*")  # 'show' hides the password characters

password_entry.pack(pady=5)

    # Create the login button

login_button = tk.Button(root, text="Login", command=validate_login)

login_button.pack(pady=20)

    # Run the Tkinter event loop

root.mainloop()
```

Output:



3. Beautiful Soup 4 - Web Scraping:

a) Purpose: parses HTML and XML documents to extract data.

b) key features:

1. Easy navigation and searching within HTML
2. Supports different parses like html, parser, lxml, and html5lib.

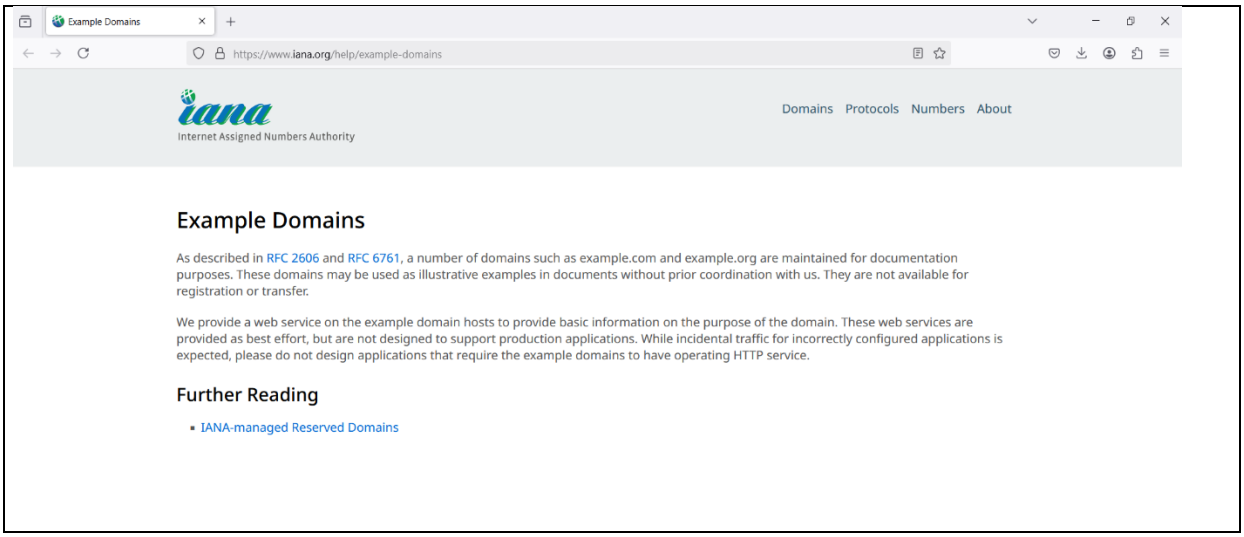
c) Common use: Extract data from websites for analysis, e.g., for building data-driven applications.

Code:

```
#pip install bs4

#pip install beautifulsoup4

import requests

from bs4 import BeautifulSoup

# The URL of the website to scrape

url = 'https://example.com'  # Replace with the website you want to
scrape

    # Send a GET request to fetch the raw HTML content

response = requests.get(url)

  # Parse the raw HTML content with BeautifulSoup

soup = BeautifulSoup(response.text, 'html.parser')

  # Find the title of the webpage

title = soup.title.string

print(f"Title of the page: {title}")

  # Find all headings (e.g., <h1>, <h2>, <h3>, etc.)

headings = soup.find_all(['h1', 'h2', 'h3'])

for heading in headings:

  print(heading.text.strip())  # Print the heading text

  # Find all links (<a> tags) on the page
```

```
links = soup.find_all('a', href=True)

for link in links:

   print(f"Link: {link['href']}"
```

Output:



4.Scrapy:

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

b. Key Features:

i. Fast, extensible, and asynchronous web scraping.

ii. Supports handling requests, data extraction, and storing results.

iii. Built-in handling for logging, retries, and sessions.

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

5. Cherry py:

a) Purpose: Minimalistic web framework for building web applications.

b) key features:

1. provides a simple and flask HTTP server.
2. Handles routing, cookies, sessions and file uploads.

c) **Common Use:** Building web applications with a light weight framework.

Code:

```
# pip install cherrypy
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:

6. Flask:

a) **Purpose:** Lightweight micro-framework for building web applications.

b) **key features:**

1.simple to learn and use, but highly extensible.

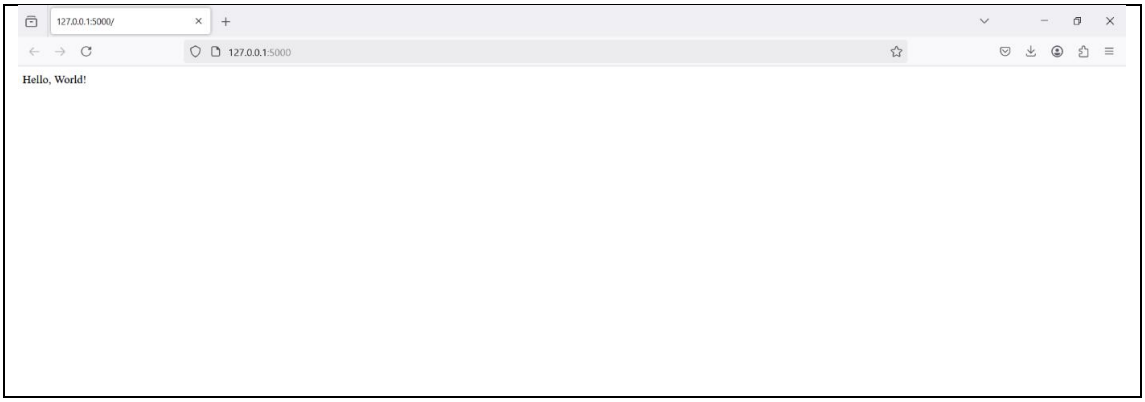
2.suppose extensions for database integration. form handling authentication

c) **Common Use:** small to medium web applications, Apls, or microservices.

Code:

```
#pip install flask
   from flask import Flask
 # Create a Flask application instance
app = Flask(__name__)
     # Define a route for the root URL ("/")
@app.route('/')
def hello_world():
   return 'Hello, World!'
   # Run the Flask application
if __name__ == '__main__':
   app.run(debug=True)
```

Output:



7.Zappa:

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

b. Key Features:

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

   ii. Manages serverless architecture and deployment configurations.

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

8.Dash:

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

b. Key Features:

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

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

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

9.Turbo Gears:

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

b) **Key Features:**

   i. Modular: Mix and match components like SQLAlchemy, Genshi, and others.

   ii. Focus on rapid development and scalability.

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

10. Bottle:

a) Purpose: Simple and light weight WSGI micro-- frame work.

b) key features:

1.single-file framework, minimalistic, and fast.

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

c) **Common Use:** Small web applications, Prototypes.

Code:

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

Output:

11.Web2Py:

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

b. Key Features:

i. Includes a web-based IDE for development.

ii. Built-in ticketing system and database integration.

c. **Common Use:** Enterprise web applications with minimal setup.

12.CubicWeb:

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

b. **Key Features:**

   i. Uses a highly modular architecture for development.

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

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

13.Quixote:

a. **Purpose:** A web framework designed for simplicity and scalability.

b. Key Features:

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

   ii. Easily extensible, with minimalistic core.

c. **Common Use:** Scalable and customizable web applications.

14.Pyramid:

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

b. Key Features:

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

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

c. **Common Use:** Building large, enterprise-grade web applications

and REST APIs.

SUMMARY:

a) **Flask, Django, Pyramid**: Popular web frameworks, each offering flexibility and scalability.

b) **Scrapy, BeautifulSoup4:** Specialized for web scraping and data extraction.

c) **Requests, Zappa, Dash:** Tools for making HTTP requests, serverless apps, and interactive data visualizations.

d) **Tkinter, Bottle, Cherry Py:** Libraries for building lightweight desktop and web applications.

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD Email: hod. it@intugvcev.edu.in

1. Name of the Laboratory : Django Framework Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
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 EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Django: A Web Framework for Python

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

- ***Key Features of Django:***

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

Django's MVT Architecture:

Model (M) – Handles database interactions (e.g., User,

Booking). View (V) – Manages business logic and

connects models to templates. Template (T) – Renders

HTML pages dynamically.

Example MVT Folder Structure in Django

```
myproject/          # Project Directory
├── myproject/      # Project Settings Directory
│   ├── __init__.py
│   ├── settings.py  # Project settings
│   ├── urls.py      # URL routing
│   ├── wsgi.py      # WSGI entry point
│   └── asgi.py      # ASGI entry point
│
├── myapp1/         # Django App Directory
│   ├── migrations/  # Database migrations
│   ├── __init__.py
│   ├── admin.py     # Admin panel configurations
│   ├── apps.py      # App configuration
│   ├── models.py    # Models (Database structure)
│   ├── views.py     # Business logic
│   ├── urls.py      # App-specific URLs
│   └── templates/   # Template folder
│       └── home.html # HTML file for rendering UI
│
├── manage.py       # Django command-line utility
```

Django Definition:

Django is a high-level, open-source Python web framework that facilitates rapid development of secure and maintainable websites, focusing on backend development and providing a robust set of tools and libraries.

Key Concepts in Django:

1. Project and App:

a) Project: A project is a collection of configurations and apps. It's the highest-level structure in Django.

b) App: An app is a web application that does something specific. A project can contain multiple apps, like a blog app, a forum app, etc.

2. MTV Architecture:

Django follows an architectural pattern called the MTV (Model-Template-View) pattern. Let's break it down:

a) Model:

1.Represents your data and the database structure.

2.It's like a blueprint for your data.

b) Template:

1.Determines how the user interface looks.

2.It's like a template for your web pages.

c) View:

1.Handles the logic of your application and controls what data is displayed in the template.

2.It's like the brain of your application.

2.OBJECTIVES:

Objective:

Create a tool that generates a timetable for students and faculty based on the courses offered and faculty availability.

Features:

1. Allow students and faculty to view personalized timetables.

2.Admin can input courses, timings, and faculty assignments.

3. Automatically resolve timetable conflicts.

4.Frameworks: Flask or Django for backend, Tkinter for GUI.

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Step-by-Step Guide to Installation Django
7. Date of Experiment : 27-12-2024
8. Date of Submission of Report : 03-01-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

4.1 Django installation steps:

Step-1: Checking the installation & version of Python

```
Python  - -version
```

Step-2: Installation of Pip

```
Python -m pip install -U pip
```

Step-3: Creation of Virtual Environment

```
Python -m venv timetable
```

Step-4: Activate the Virtual environment

```
timetable\Scripts\activate
```

Step-5: Installation of Django in Virtual environment

```
pip install Django
```

Step-6: Create a folder to store all the projects

```
mkdir _ proj_ folder _name
```

Step-7: Start a new Django project

```
Django -admin start project Project _name
```

Step-8: Start app

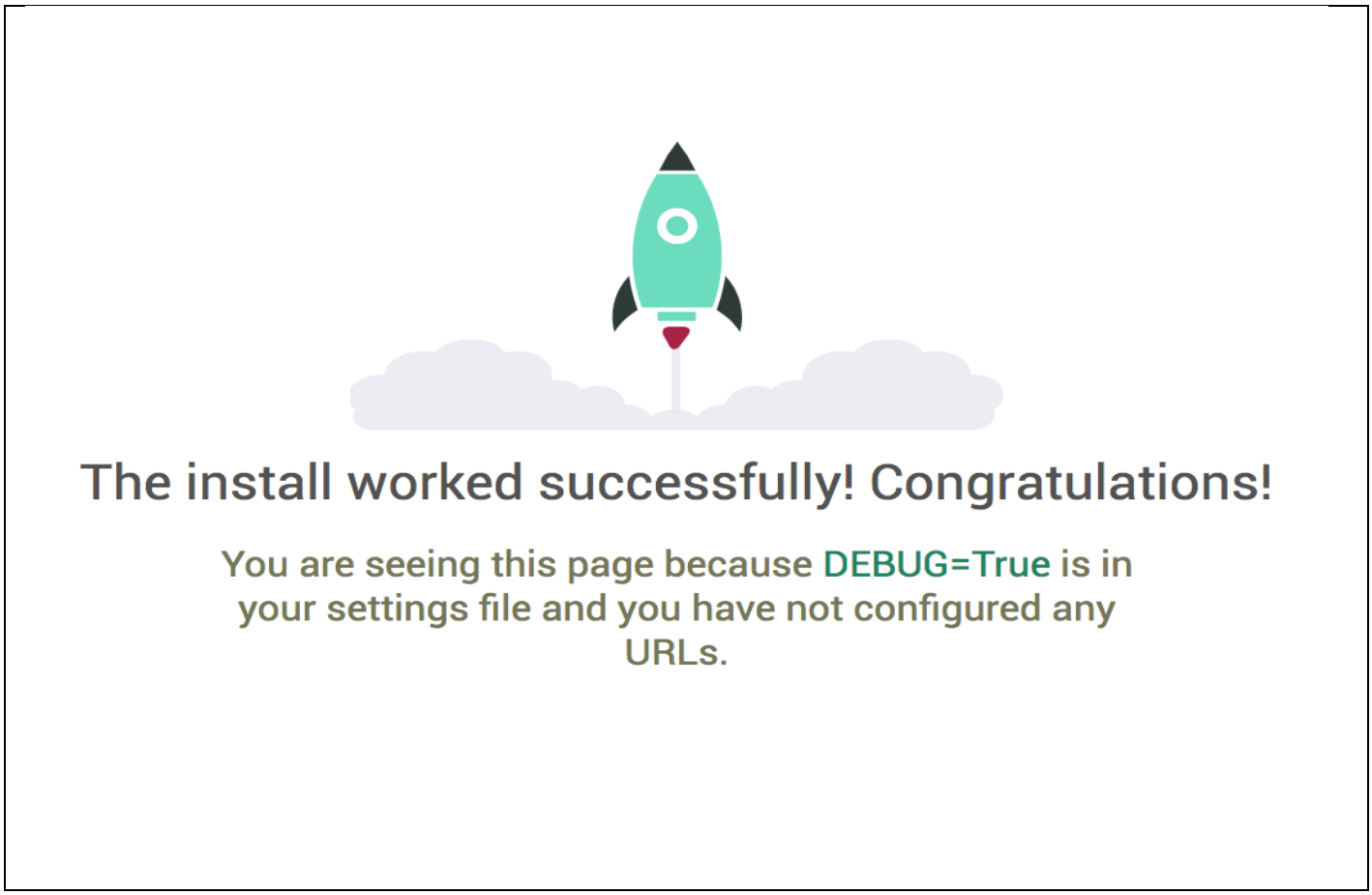
```
Django-admin start app app _name
```

Step-9: Run the server

python manage.py run server

Step-10: Open the browser and check the homepage of Django

OUTPUT:



4.2 COMMANDS:

C:\Users\lavan>python --version

Python 3.13.1

C:\Users\lavan>pip –version

pip 24.3.1 from C:\Program Files\Python313\Lib\site-packages\pip (python 3.13)

C:\Users\lavan>python -m venv timetable

C:\Users\lavan>timetable\Scripts\activate

(timetable) C:\Users\lavan>pip install Django

Successfully installed asgiref-3.8.1 django-5.1.4 sqlparse-0.5.3 tzdata-2024.2

(timetable) C:\Users\91720>mkdir Django_ projects

(timetable) C:\Users\91720>cd Django_ projects

(timetable) C:\Users\91720\Django_Projects>django-admin start project Hello _world

(timetable) C:\Users\91720\Django_Projects>django-admin start app hello world _app

(timetable) C:\Users\91720\Django_Projects>cd Hello _World

(timetable) C:\Users\91720\Django_Projects\Hello_World>python manage.py runserver

Django version 5.1.4, using settings 'Hello _World .settings'

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

Quit the server with CTRL-BREAK.

To Come out from the ENVS - Ctrl + C

To Come back – Cd ..

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Linking Views and URL Configrations
7. Date of Experiment : 03-01-2024
8. Date of Submission of Report : 24-01-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Linking Views and URLs:

Set Up URLs:

Project-level URL Configuration:

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

App-level URL Configuration from

```
django.urls import
path from .views
import home
urlpatterns = [
    path('', home, name='home'),
]
```

Create a Sample View:

```
from django.http import
HttpResponse def
home(request):
    return HttpResponse("<h1>Welcome to My Django App!</h1>")
```

Run Migrations:

```
python manage.py migrate
```

Run the Server and Test:

```
python manage.py runserver
```

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD Email: hod. it@intugvcev.edu.in

1. Name of the Laboratory : Django Framework Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
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 : 24-01-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Project views code:

Views.py:

In Django, views.py is the file where you define functions or classes that handle requests and return responses. Views act as the logic layer of a Django web application, controlling how data is processed and which HTML templates are displayed.

```
from django.shortcuts import render


def home(request):
    return render(request,"homepage.html")


def word(request):
    return render(request,"loginpage.html")


def hello(request):
    return render(request,"sign in page.html")


def admin(request):
    return render(request,"adminpage.html")


def forgot(request):
    return render(request,"reset.html")


def timetable(request):
    return render(request,"generate timetable.html")
```

```
def logout(request):
    return render(request,"logout.html")

def home2(request):
    return render(request,"homepage2.html")
```

In Django, the urls.py file is responsible for mapping URLs to views. It acts as the router of your application, directing user requests to the correct function in views.py.

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD Email: hod. it@intugvcev.edu.in

1. Name of the Laboratory : Django Framework Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
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 EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

App Urls.Py :

Creating urls.py in a Django App:

Each Django app should have its own urls.py file to define app-

specific routes. Steps to Create urls.py in a Django App

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

Code:

```
from django.urls import path

from . import views


urlpatterns=[

  path('',views.home),

  path('adminpage.html/',views.admin),

  path('loginpage.html/',views.word),

  path('sign in page.html/',views.hello),

  path('reset.html/',views.forgot),

  path('generate timetable.html/',views.timetable),

  path('logout.html/',views.logout),

  path('homepage2.html/',views.home2),

]
```

Connecting App urls.py to Project urls.py:

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

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

DEPARTMENT OF INFORMATION TECHNOLOGY

JNTU-GURAJADA VIZIANAGARAM

COLLEGE OF ENGINEERING VIZIANAGARAM (A)

VIZIANAGARAM

Dr.Ch. Bindu Madhuri

Asst. Professor & HOD Email: hod. it@intugvcev.edu.in

1. Name of the Laboratory : Django Framework Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Working with Templates in Django
7. Date of Experiment : 31-01-2025
8. Date of Submission of Report : 17-02-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Templates:

a) Templates are the third and most important part of Django's MVT Structure. A template in Django is basically written in HTML, CSS, and Java script in a .html file.
b) Django framework efficiently handles and generates dynamic HTML web pages that are visible to the end-user. Django mainly functions with a backend so in order to provide a frontend and provide a layout to our website, we use templates.
c) There are two methods of adding the template to our website depending on our needs. We can use a single template directory which will be spread over the entire project.
d) For each app of our project, we can create a different template directory.

TEMPLATES CODE:

```
TEMPLATES = [
  {
    'BACKEND':
'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(BASE_DIR,'tableapp','templates')],
    'APP_DIRS': True,
    'OPTIONS': {
      'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
      ],
    },
  },
]
```

Using Django Templates:

- Illustration of How to use templates in Django using an Example Project.
- Templates not only show static data but also the data from different databases connected to the application through a context dictionary.
- Consider a project named My Project having an app named My Project App.

Homepage:

- If you want to provide an introductory page or a central hub for users, then a homepage is useful.
- Reason: For users who aren't logged in yet, the homepage can show an overview of the system (e.g., an explanation of the tool and login/sign-up options).
- Functionality: A simple page with basic info about the system, login links, and possibly links to resources like FAQs or guides.
- Alternatives:
  Single Landing Page: You can combine the homepage and login page into a single landing page that prompts users to either log in or sign up (if required).

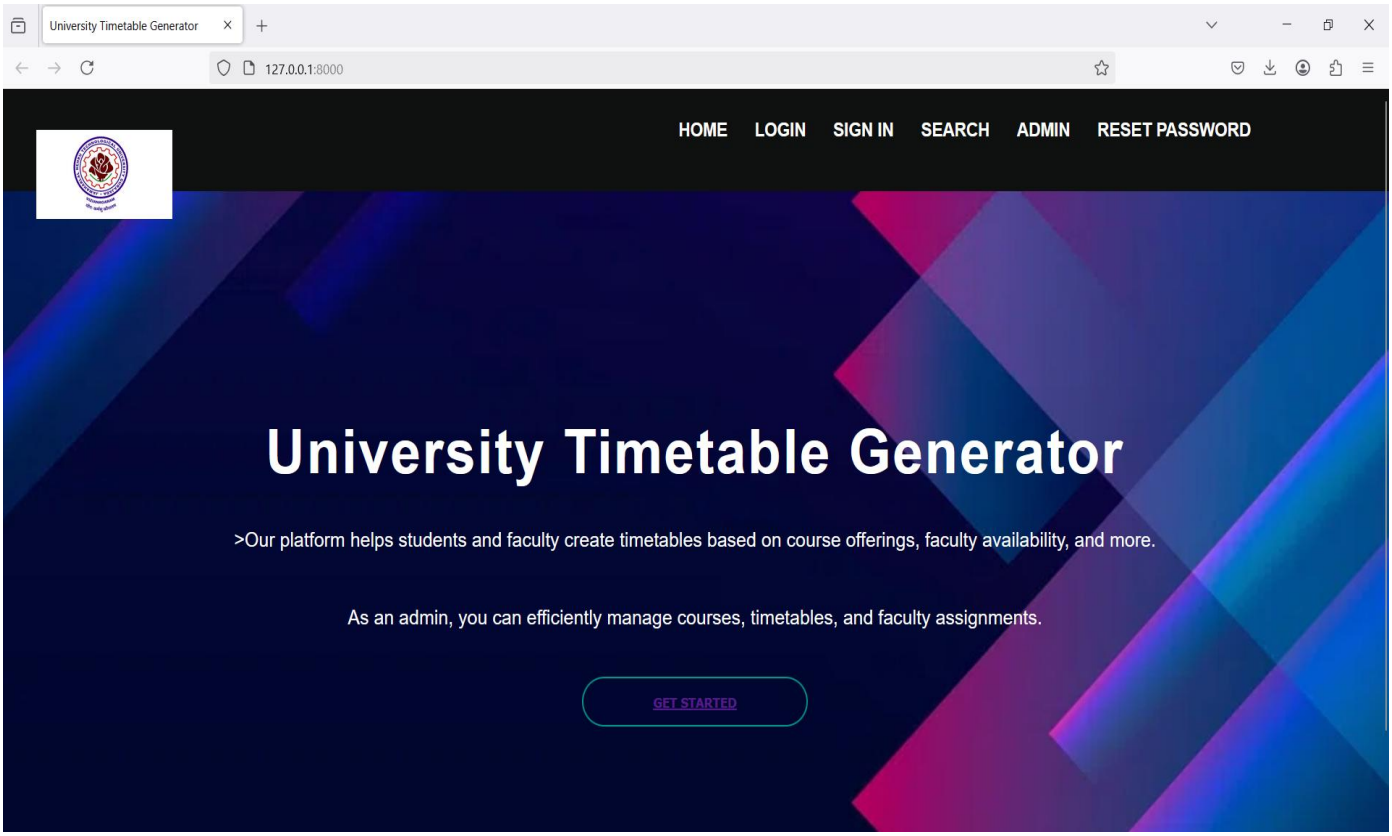
Code:

```
<!DOCTYPE html>
{% load static %}
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>University Timetable Generator</title>
    <link rel="stylesheet" href="{% static 'homepage.css' %}">
  </head>
```

```
<body>
    <div class="header-txt">
    </div>
    <div class="container">
        <div class="navbar">
            <img src="https://education.sakshi.com/sites/default/files/images/2024/02/23/jntu-gv-1708690466.jpg" class="logo">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="loginpage.html">Login</a></li>
                <li><a href="sign in page.html">Sign in</a></li>
                <li><a href="adminpage.html">Admin</a></li>
                <li><a href="reset.html">Reset password</a></li>
                <li><a href="generate timetable.html">timetable</a></li>
                <li><a href="logout.html">logout</a></li>
            </ul>
        </div>
        <div class="content">
            <h1>University Timetable Generator</h1>
            <p>>Our platform helps students and faculty create timetables based on course offerings, faculty availability, and more.</p>
             <p>As an admin, you can efficiently manage courses, timetables, and faculty assignments.</p>
             <div>
                <button type="button"><a href="generate timetable.html"><span></span>Get Started</a></button>
             </div>
        </div>
    </div>
```

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

Output:



Login Page:

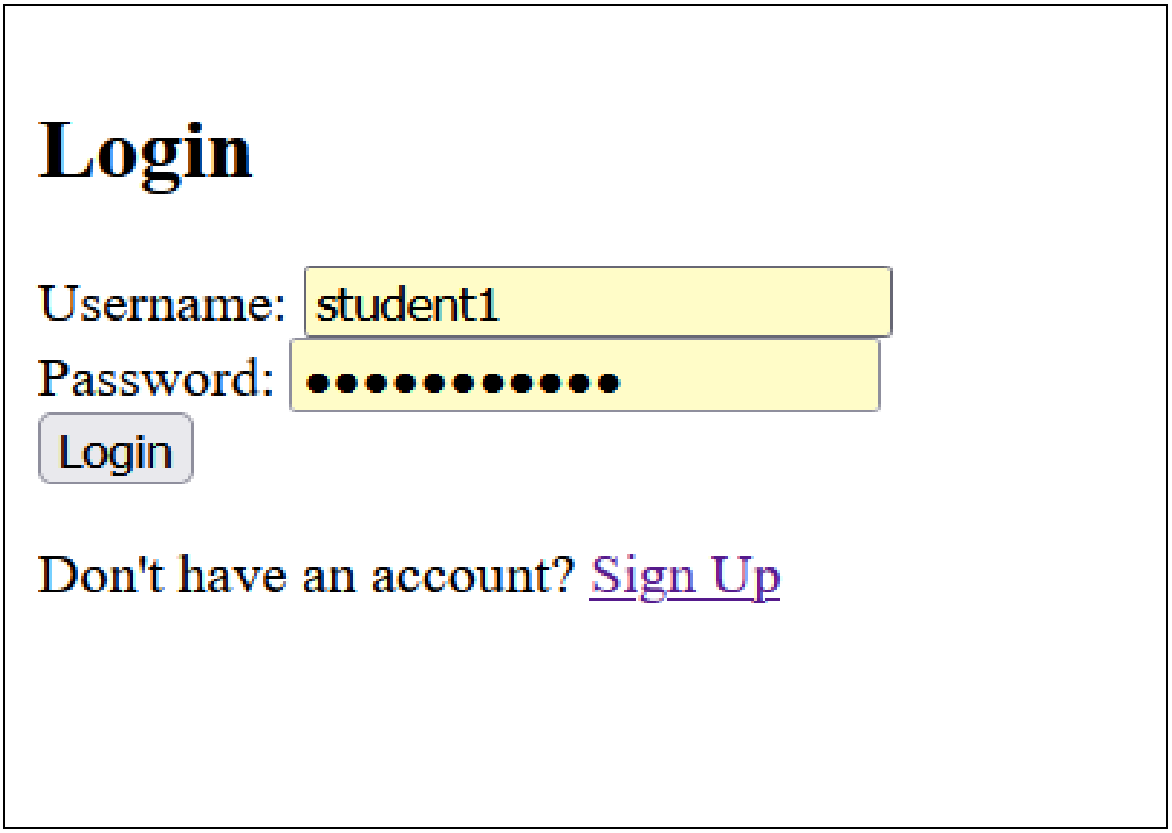
- **Required:** Yes, especially for projects where data is sensitive or personalized.
- **Reason:**
  Users (students, faculty, admins) will need to log in to access their personalized timetables and roles.
- **Functionality:**
  Authentication (e.g., email/password or single sign-on) to ensure only authorized users access specific functionalities.

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login - University Timetable Generator</title>
  <link rel="stylesheet" href="loginpage.css">
</head>
<body>
  <div class="login-container">
    <h2>Login</h2>
    <form action="/login" method="POST">
      <div class="input-group">
        <label for="username">Username:</label>
```

```
            <input type="text" id="username" name="username" required>
        </div>
        <div class="input-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"
required>
        </div>
        <button type="submit">Login</button>
        <p>Don't have an account? <a href="sign in page.html">sign
in</a></p>
      </form>
   </div>
</body>
</html>
```

Output:



Sign-Up Page:

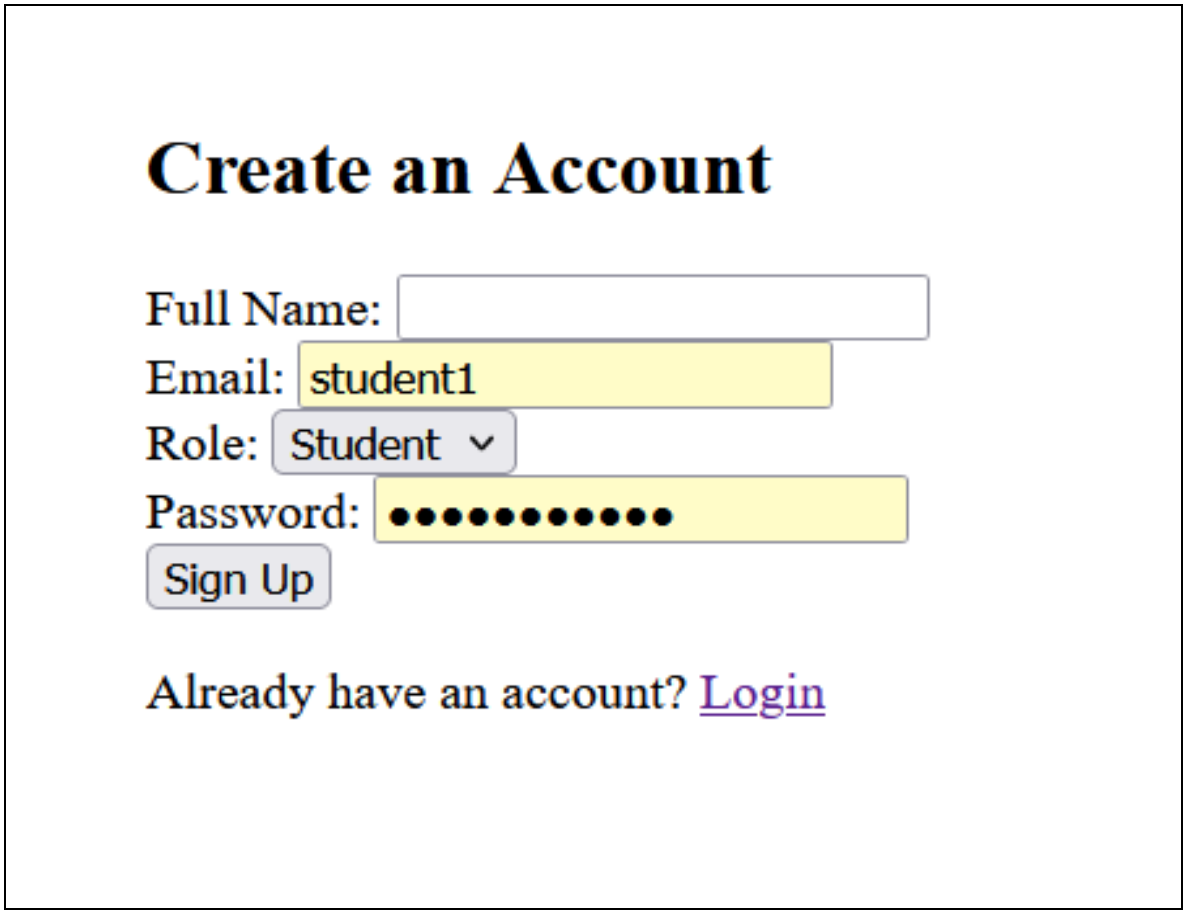
- This depends on whether you want to allow users to self-register or if registration is handled by an admin.
- **Reason:** If the university or admin handles user creation (e.g., students and faculty are registered automatically), a sign-up page might not be necessary.
- **Functionality:** This page allows users to create new accounts, typically with a form asking for basic details like name, email, role (student/faculty), etc. Admins could add users directly, bypassing the need for sign-up.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sign in - University Timetable Generator</title>
  <link rel="stylesheet" href="sign in page.css">
</head>
<body>
  <div class="signin-container">
    <h2>Create an Account</h2>
    <form action="/sign in" method="POST">
      <div class="input-group">
        <label for="name">Full Name:</label>
        <input type="text" id="name" name="name" required>
      </div>
      <div class="input-group">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
      </div>
      <div class="input-group">
        <label for="role">Role:</label>
        <select id="role" name="role">
          <option value="student">Student</option>
          <option value="faculty">Faculty</option>
        </select>
      </div>
```

```
        <div class="input-group">
          <label for="password">Password:</label>
          <input type="password" id="password" name="password" required>
        </div>
        <button type="submit">Sign in</button>
        <p>Already have an account? <a href="loginpage.html">Login</a></p>
      </form>
    </div>
</body>
</html>
```

Output:


Create an Account
Full Name:
Email: student1
Role: Student
Password:
Sign Up
Already have an account? Login

Admin Page

- **Required:** Yes, essential for the management of courses, faculty assignments and conflict resolution.

- **Reason:** The admin will need a page to enter courses, assign faculty, manage conflicts, and potentially view system usage statistics.

- **Functionality:** Admins can add/edit courses, assign faculty, resolve conflicts in the timetable, and monitor student enrollments.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Dashboard - University Timetable Generator</title>
  <link rel="stylesheet" href="adminpage.css">
</head>
<body>
  <header>
    <div class="admin-header">
      <h1>Admin Dashboard</h1>
      <nav>
        <a href="view-timetable.html">View Timetable</a>
        <a href="add-course.html">Add New Course</a>
        <a href="logout.html">Logout</a>
      </nav>
```

```
    </div>
  </header>


  <main>
    <div class="admin-container">
      <h2>Manage Courses</h2>
      <section class="course-management">
        <h3>Add New Course</h3>
        <form action="/add-course" method="POST">
          <div class="input-group">
            <label for="course-name">Course Name:</label>
            <input type="text" id="course-name" name="course-name"
required>
          </div>
          <div class="input-group">
            <label for="course-time">Course Time:</label>
            <input type="text" id="course-time" name="course-time" required>
          </div>
          <div class="input-group">
            <label for="course-faculty">Assign Faculty:</label>
```

```
        <select id="course-faculty" name="course-faculty">
          <option value="faculty1">Faculty 1</option>
          <option value="faculty2">Faculty 2</option>
          <option value="faculty3">Faculty 3</option>
          <!-- Add more faculty options dynamically -->
        </select>
      </div>
      <button type="submit">Add Course</button>
    </form>
  </section>
  <section class="course-list">
    <h3>Existing Courses</h3>
    <table>
      <thead>
        <tr>
          <th>Course Name</th>
          <th>Course Time</th>
          <th>Assigned Faculty</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>Math 101</td>
          <td>9:00 AM - 11:00 AM</td>
          <td>Faculty 1</td>
          <td><button>Edit</button><button>Delete</button></td>
```

```
            </tr>
            <tr>
              <td>Physics 201</td>
              <td>11:00 AM - 1:00 PM</td>
              <td>Faculty 2</td>
              <td><button>Edit</button><button>Delete</button></td>
            </tr>
              <!-- Add dynamic rows for other courses -->
          </tbody>
        </table>
      </section>
    </div>
  </main>
</body>
</html>
```

Output:

Admin Dashboard

View Timetable Add New Course Logout

Manage Courses

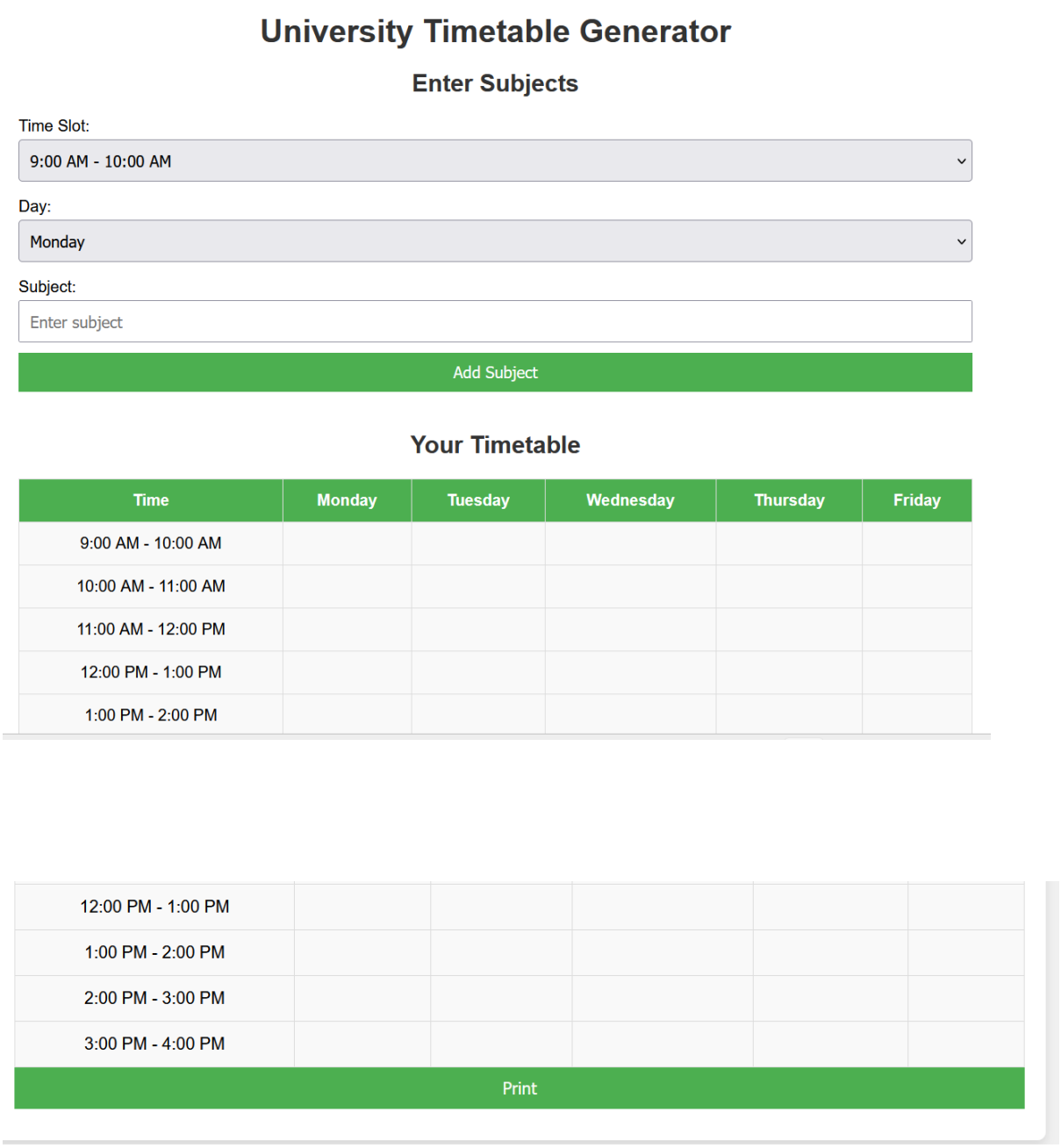
Add New Course

Course Name:
Course Time:
Assign Faculty: Faculty 1
Add Course

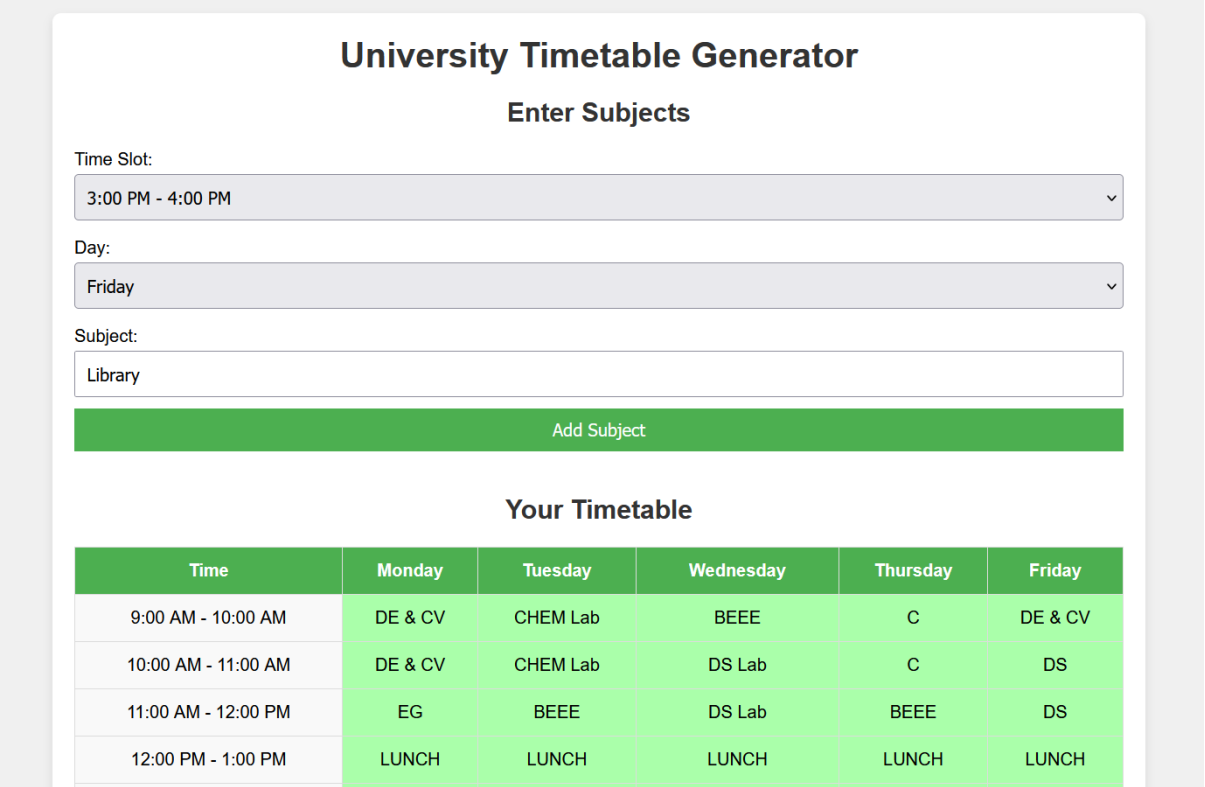
Existing Courses

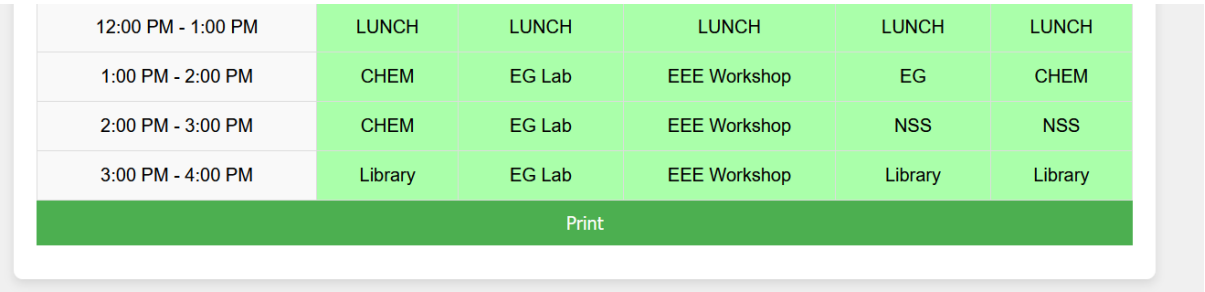
Course Name	Course Time	Assigned Faculty	Actions
Math 101	9:00 AM - 11:00 AM	Faculty 1	Edit Delete
Physics 201	11:00 AM - 1:00 PM	Faculty 2	Edit Delete

OUTPUT:


University Timetable Generator
Enter Subjects
Time Slot:
9:00 AM - 10:00 AM
Day:
Monday
Subject:
Enter subject
Add Subject
Your Timetable
Time
Monday
Tuesday
Wednesday
Thursday
Friday
9:00 AM - 10:00 AM
10:00 AM - 11:00 AM
11:00 AM - 12:00 PM
12:00 PM - 1:00 PM
1:00 PM - 2:00 PM
12:00 PM - 1:00 PM
1:00 PM - 2:00 PM
2:00 PM - 3:00 PM
3:00 PM - 4:00 PM
Print

FINAL 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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Database Integration and Configuration- SQL LITE
7. Date of Experiment : 17-02-2025
8. Date of Submission of Report : 21-02-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Database in Django:

To add a database to your Django project, you need to configure Django to connect to a specific database backend. Django supports several databases, such as SQLite (default), PostgreSQL, MySQL, and Oracle. Below are the key steps to configure a database for your Django project.

1. Install Database Backend

First, depending on which database you want to use, you'll need to install the appropriate database driver. Here's how to install the database drivers for the most common backends.

For PostgreSQL:

Code:

pip install psycopg2

For MySQL:

Code:

pip install mysqlclient

For SQLite (default database):

SQLite is included by default with Django, so no installation is necessary if you're using SQLite.

2. Configure the Database in settings.py

Open your Django project's settings.py file and locate the DATABASES setting. This is where you configure your database connection.

Settings.Py:

```
"""
Django settings for tableproject project.

Generated by 'django-admin startproject' using Django 5.1.5.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-=4j+*iuibbp2j383)gg9v(1+s1mb%su29-i@fq$623eowx@4by'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tableapp',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'tableproject.urls'

TEMPLATES = [
    {
```

```
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'tableapp','templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'tableproject.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
  {
    'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
  },
]


# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.1/howto/static-files/
```

```
STATIC_URL = 'static/'

STATICFILES_DIRS=[BASE_DIR / 'static']

STATIC_ROOT= 'staticfiles'


# Default primary key field type

# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field


DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Default (SQLite) Configuration:

If you're using SQLite (which is the default database), the configuration will look like this:

Code:

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

PostgreSQL Configuration:

If you're using PostgreSQL, update the DATABASES setting like this:

Code:

```
DATABASES = {
   'default': {
      'ENGINE': ' django. db. Backends. postgresql ',
      'NAME': 'your_ d b _name', # The name of your database
      'USER': 'your_ d b _user', # Your database user
      'PASSWORD': 'your_ db _password', # Your database user's password
      'HOST': 'localhost', # If your database is on the same machine, use 'localhost'
      'PORT': '5432', # Default PostgreSQL port
   }
}
```

MySQL Configuration:

For MySQL, configure your DATABASES setting as follows:

Code:

```
DATABASES = {
   'default': {
      'ENGINE': 'django. db .backends .mysql',
      'NAME': 'your_ d b _name',
      'USER': 'your_ d b _user',
      'PASSWORD': 'your_ d b _password',
      'HOST': 'localhost',
      'PORT': '3306',  # Default MySQL port
   }
}
```

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Handling Forms in Django
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 21-02-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

What is forms.py in Django:

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

Why Use forms.py:

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

Types of Forms in Django:

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

```
# myapp1/forms.py
from django import forms
from .models import Room,
    Booking from datetime
    import time

    class
        RoomForm(forms.ModelF
        orm): class Meta:
       model = Room
       fields = ['name', 'capacity', 'teacher', 'is_active']  # Include the fields you need

    class
        BookingForm(forms.ModelF
        orm): class Meta:
       model = Booking
       fields = ['room', 'date', 'time_slot', 'booked_by', 'user']

   time_slot = forms.TimeField(required=True)  # Make it mandatory to catch errors
```

What is models.py in Django:

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

Why Use Django Models:

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

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Defining and Using Models
7. Date of Experiment : 21-02-2025
8. Date of Submission of Report : 07-03-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

How to Apply Models:

Create the Model in models.py:

- Write your models inside the models.py file. **MODELS.PY:**

```
from django.db import models
from django.contrib.auth.models
    import User from django.core.mail
    import send_mail from django.conf
    import settings
class Room(models.Model):
    name =
        models.CharField(max_length=100
        ) capacity = models.IntegerField()
         teacher = models.ForeignKey(User, on_delete=models.CASCADE, related_name="rooms",
    null=True, blank=True)
    is_active =
        models.BooleanField(default=True)
        def _str_(self):
       return self.name
class Booking(models.Model):
    room = models.ForeignKey(Room,
        on_delete=models.CASCADE) date =
        models.DateField()
    time_slot = models.TimeField()  # Single time field for
        booking booked_by = models.CharField(max_length=100)
    user = models.ForeignKey(User,
        on_delete=models.CASCADE) email =
```

```
    models.EmailField()
def __str__(self):
        return f"Booking for {self.room} on {self.date} at {self.time_slot} by
    {self.user.username}" def send_booking_email(self):
    subject = 'Room Booking Confirmation'
        message = f"Your room booking for {self.room.name} on {self.date} at {self.time_slot} is confirmed."
        email_from = settings.EMAIL_HOST_U
```

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Migrations: Syn with Database
7. Date of Experiment : 07-03-2025
8. Date of Submission of Report : 27-03-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Run Migrations to Create Database Tables:

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

A) python manage.py makemigrations

B) python manage.py migrate

Migrate the Database

Once the database is configured, you need to create the database tables based on the Django models. This is done using Django's migration system.

1. **Make Migrations**: This will create migration files based on the changes in your models.
   Code:

   python manage.py make migration

2. **Apply Migrations**: This will apply the migrations and create the database schema in the database you configured.
   Code:

   python manage.py migrate

Create a Superuser (Optional)

If you want to access the Django admin interface, you'll need to create a superuser account. Run the following command and follow the prompts:

Code:

python manage.py create superuser

Test the Database Connection

To test if your database connection is set up correctly, you can start the Django development server:

Code:

python manage.py runserver

Now, you should be able to access your project at http://127.0.0.1:8000 and use the Django admin at http://127.0.0.1:8000/admin to manage your data.

(Optional) Use Database-specific Features

Depending on the database backend you're using (PostgreSQL, MySQL, etc.), you may want to take advantage of specific features that are optimized for that database.

For example:

PostgreSQL has advanced features like JSON fields and full-text search.

MySQL is widely used in web applications, and you can configure various performance optimizations.

Refer to the Django documentation for database configuration for more advanced configurations and features.

(Optional) Database Backups & Migrations in Production

If you're deploying your Django application to production, it's important to:

Regularly **backup your database**.
Use Django's **database migrations** in a controlled deployment process (with tools like p g_ dump for PostgreSQL or my sql dump for MySQL).

Also, always ensure that your production environment uses strong credentials and a secure database connection (such as SSL for PostgreSQL).

Conclusion:

With these steps, you can configure your Django project to use a database of your choice, apply migrations to create the necessary database tables, and manage data through the Django admin interface. Let me know if you need further details on any of these steps!

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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Developing Django Application on Cloud Platforms
7. Date of Experiment : 27-03-2025
8. Date of Submission of Report : 04-04-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Setting Up a GitHub Repository for My University Timetable Generator Project

I started by creating a **GitHub repository** for my **University Timetable Generator** Django project. This involves setting up the project locally, initializing Git, pushing it to GitHub, and organizing the project structure.

1. Create a GitHub Repository

First, I created a repository on GitHub. Here's how:

1. Log in to GitHub.
2. Click the **+** icon on the top-right and select **New repository**.
3. Name the repository university-timetable-generator.
4. Make it **public** or **private**, depending on your choice.
5. Leave **Initialize this repository with a README** unchecked.
6. Click **Create repository**.

2. Initialize the Local Project with Git

1. Start a new Django project:
    a) In your terminal, navigate to your desired directory where you want to create the project, then run:
    b) django-admin startproject university_timetable
    c) cd university_timetable

2. Initialize Git:
    a) Run the following command to initialize Git in your project folder:
    b) git init

3. Create a .gitignore File:

a) Create a .gitignore file to exclude files that don't need to be tracked by Git (e.g., bytecode, virtual environment, database files):

b) touch .gitignore

Then, add the following contents to the .gitignore file:

```
*.pyc
*.pyo
*.pyd
__pycache__
db.sqlite3
venv/
.env
```

4. Commit the initial files:
   a) Add all the files to Git and make an initial commit:
   b) git add .
   c) git commit -m "Initial commit of University Timetable Generator"

3. Push the Project to GitHub

1. Link the local repository to GitHub:
   a) After creating the repository on GitHub, copy the repository URL (it will look like https://github.com/your-username/university-timetable-generator.git).
   b) Then, link your local project to GitHub:
   c) git remote add origin https://github.com/your-username/university-timetable-generator.git

2. Push to GitHub:
    a) Push the project to the main branch on GitHub:
    b) git branch -M main
    c) git push -u origin main

Your project is now on GitHub.

3. Organizing the Project Structure

Here is the basic structure I created for my project:

```
university_timetable/
├── university_timetable/      # Django project directory
│   ├── __init__.py
│   ├── settings.py          # Configuration file for the Django project
│   ├── urls.py              # URL routes for the project
│   ├── wsgi.py              # WSGI entry point for web servers like Gunicorn
│   ├── asgi.py              # ASGI entry point for async servers
├── timetable/               # Django app for the timetable
│   ├── migrations/
│   ├── __init__.py
│   ├── models.py            # Models for university timetable
│   ├── views.py             # Views for timetable-related requests
│   ├── urls.py              # URL routes for timetable app
│   ├── admin.py             # Admin configuration for models
│   ├── forms.py             # Forms for timetable data input
│   ├── templates/           # HTML templates for views
│   ├── static/              # Static files (CSS, JS, images)
├── manage.py                # Django's command-line utility
```

```
├── requirements.txt          # List of dependencies for the project
├── .gitignore                # Git ignore file to exclude unnecessary files
└── README.md                 # Project description and setup guide
```

5. Adding a README File

Next, I added a README.md file to explain the purpose of the project and how to set it up. Here's the code for the README.md:

 # University Timetable Generator

This Django-based web application generates a timetable for university courses. It allows administrators to assign professors, rooms, and time slots to courses and generates a timetable for students.

Features

- Add and manage courses, professors, rooms, and students.
- Automatically generate a timetable based on inputs.
- Prevent timetable conflicts (room or professor availability).
- Web-based interface for managing timetables.

Setup Instructions

Prerequisites

- Python 3.x
- Django 3.x+
- PostgreSQL (or any other preferred database)

Installation

1. Clone this repository:

   ```bash

   git clone https://github.com/your-username/university-timetable-generator.git

   cd university-timetable-generator

2. Create a virtual environment and activate it:
3. python3 -m venv venv
4. source venv/bin/activate  # On Windows, use `venv\Scripts\activate`
5. Install dependencies:
6. pip install -r requirements.txt
7. Set up the database and apply migrations:
8. python manage.py migrate
9. Create a superuser for the Django admin panel:
10. python manage.py createsuperuser
11. Run the development server:
12. python manage.py runserver
13. Open http://127.0.0.1:8000/admin in your browser to log in to the Django admin panel and start managing the timetable.

## 6. Push Changes to GitHub

Now that I've added the README and requirements.txt files, I committed and pushed the updates:

git add .

git commit -m "Added README.md and requirements.txt"

git push origin main
   ```

7. Continue Development and Push Updates

As I continue to develop the timetable generator, I can keep adding new features and commit the changes to GitHub regularly.

Conclusion:

I now have a well-structured **University Timetable Generator** project that is hosted on GitHub. It is set up for continuous development, and I can easily collaborate with others, or anyone can clone the project to contribute or run it locally.

GITHUB LINK:

https://github.com/lavanyadatti475/university-timetable-generator.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 Lab
2. Name of the Student : D. Lavanya
3. Roll No : 23VV1A1211
4. Class : II B.Tech, II Semester
5. Academic Year : 2024-25
6. Name of Experiment : Front End Web Developer Certification
7. Date of Experiment : 04-04-2025
8. Date of Submission of Report : 04-04-2025

S.NO	ABILITY AND ACTIVITY	WEIGHTAGE OF MARKS	DAY TO DAY EVALUTION SCORE
1	Aim Objective, Tools required	3	
2	Theory, Algorithm and Observations	3	
3	Implementation	3	
4	Schematic diagrams,Architecture, workflow,Flowchart	3	
5	Tidiness of his/her working area, proper maintenance of system during and after experiment.	3	
	Total Score	15	

Date: **Signature of faculty:**

Infosys®
Navigate your next

CERTIFICATE OF ACHIEVEMENT

The certificate is awarded to

Lavanya Datti

for successfully completing

Front End Web Developer Certification

on March 6, 2025

Infosys | Springboard

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