Exp-13
Design web application with different menu items using Django

Step1:Check Python version

- **python –version**

Step2: Install virtual environment

- **py –m pip install virtualenv**

Step3: create an virtual environment

- **py –m venv second**

Step4: activate the environment

- **.\second\scripts\activate**

Step5: install django package

- **pip install Django**

Step6: create a project

- **django-admin startproject program13**

Step 7: move to project folder

- **cd program13**

Step 8: open project **program13** inside create **templates** folder.
Templates folder inside create **home.html, contactUs.html and dashboard.html**
Add the below code

- **home.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</head>
<body>
    <h1 class="text-center">Welcome to CMRIT</h1>
    <nav class="navbar navbar-expand-lg bg-body-tertiary">
        <div class="container-fluid">
          <a class="navbar-brand" href="#">CMRIT</a>
```

```
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
              <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="#">Home</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{% url 'dashboard' %}">Dashboard</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{% url 'contactUs' %}">ContactUs</a>
              </li>
            </ul>
          </div>
        </div>
      </nav>

</body>
</html>
```

- **contactUs.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ContactUs</title>
</head>
<body>
    <h1>CMR Institue of Technology,Medchal, Hyderabad.</h1>
    <a href="{% url 'home' %}">Back to home</a>
</body>
</html>
```

- **dashboard.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
</head>
<body>
    <h1>Welcome to CMRIT Dashboard</h1>
    <a href="{% url 'home' %}">Back to home</a>
</body>
</html>
```

Step 9: open project **program13** subfolder Write below code

- **urls.py**

```
from django.contrib import admin
from django.urls import path
from . import index

urlpatterns = [
#path('admin/', admin.site.urls),
path('',index.home,name='home'),
path('dashboard/', index.dashboard,name='dashboard'),
path('contactUs/', index.contactUs,name='contactUs'),
]
```

- **settings.py**

1. import os
2. templates =>DIRS =>os.path.join(BASE_DIR,'templates')

- **index.py**

```
from django.http import HttpResponse
from django.shortcuts import render

def home(request):
    return render(request,'home.html')

def dashboard(request):
    return render(request,'dashboard.html')

def contactUs(request):
    return render(request,'contactUs.html')
```

Step 10: run the server

- **py manage.py runserver**

Step 11: open browser, run http://127.0.0.1:8000/

Output:-

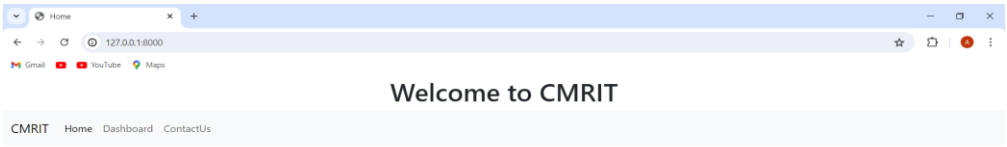

Fig.1.Home page with different menu items

VIVA QUESTIONS:

1.What is purpose of Django apps ?

Ans: Django apps are modular components within a Django project, designed to perform a specific function or set of functions. Each Django app encapsulates a set of related features or behaviors, such as handling user authentication, managing blog posts, or processing payments. The purpose of structuring a Django project into multiple apps is to promote modularity, reusability, and maintainability of the codebase. Here are the key purposes and benefits of using Django apps: Modularity, Reusability, Maintainability, Scalability.

2.How does Django manage static files like CSS, JavaScript and images ?

Ans: **Handle Static Files:**

Static files in your project, like stylesheets, JavaScripts, and images, are not handled automatically by Django when `DEBUG = False`.

When `DEBUG = True`, this worked fine, all we had to do was to put them in the `static` folder of the application.

When `DEBUG = False`, static files have to be collected and put in a specified folder before we can use it.

Collect Static Files:

To collect all necessary static files for your project, start by specifying a `STATIC_ROOT` property in the `settings.py` file.

This specifies a folder where you want to collect your static files.

You can call the folder whatever you like, we will call it `productionfiles`

3.Difference between Django and react ?

Ans: Django
Type: Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the Model-View-Template (MVT) architectural pattern.
Language: Python.
Scope: Django is a full-stack web framework that provides built-in solutions for backend development, including ORM (Object-Relational Mapping), authentication, routing, templating, and more. It's designed to help developers build complex, database-driven websites quickly.
Use Cases: Django is used for developing the server-side (backend) of web applications. It's well-suited for applications that require a robust database system, complex backend logic, administrative interfaces, and other backend functionalities.
Execution Environment: Django applications run on a web server and serve dynamic web pages or RESTful APIs that interact with a database based on client requests.

React
Type: React is an open-source JavaScript library for building user interfaces, maintained by Facebook and the community. It follows the component-based architecture.
Language: JavaScript (or TypeScript, if preferred).
Scope: React is primarily concerned with the view layer (frontend) of web applications. It allows developers to create reusable UI components and manage the state of those components efficiently.
Use Cases: React is used for developing the client-side (frontend) of web applications, particularly single-page applications (SPAs) where dynamic content needs to be updated without reloading the page. It's well-suited for applications that require a rich, interactive user experience.
Execution Environment: React applications run in the browser (client-side). They can be served by any backend server,

but they execute and render in the web browser.

4.How can you deploy a Django application in a production server?

Ans: Deploying a Django application to a production server involves several key steps to ensure that the application is secure, efficient, and accessible to users. Here's a general overview of the process:

1.Prepare Your Application for Production

Settings: Separate your development settings from production settings. This includes database configurations, secret keys, debug settings (set DEBUG = False in production), and allowed hosts (ALLOWED_HOSTS setting).

Static and Media Files: Configure settings for serving static files (STATIC_ROOT, STATIC_URL) and media files (MEDIA_ROOT, MEDIA_URL).

Database: Migrate your application to use a production-level database like PostgreSQL, MySQL, or another database supported by Django.

Dependencies: Ensure all dependencies are correctly specified in a requirements.txt file or using a package manager like Pipenv.

5.What is content delivery Network (CDN)?

Ans: A Content Delivery Network (CDN) is a globally distributed network of proxy servers and their data centers. The goal of a CDN is to provide high availability and high performance by spatially distributing the service relative to end-users. Essentially, a CDN serves content to users with high availability and high speed by caching content in multiple locations around the world and serving the content from the location closest to each user.