Ex-14
Fetch user details from server using REST API and show in profile menu 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 and pip install djangorestframework**

Step6: create a project

- **django-admin startproject program14**

Step7: create a text file

- **pip freeze > requirements.txt**

Step 8: move to project folder

- **cd program14**

Step 9: create an application

- **django-admin startapp api**
- **django-admin startapp base**

Step 10: open project inside sub folder open api

- And create two files **Urls.py, Serializers.py**

Add the below code

- **Views.py**

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

# Create your views here.
from rest_framework.response import Response
from rest_framework.decorators import api_view
from base.models import User
from .serializers import UserSerializer
import json
def home(request):
return render(request,'home.html')

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

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

@api_view(['POST'])
def getuser(request):
req_email=request.data.get('email')
Users=User.objects.filter(email=req_email)
serializer = UserSerializer(Users,many=True)
return Response(serializer.data)

@api_view(['POST'])
def adduser(request):
serializer = UserSerializer(data=request.data)
if serializer.is_valid():
serializer.save()
return Response(serializer.data)
```

- **Urls.py**

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

urlpatterns = [
path('',views.home),
path('register/',views.register,name="register"),
```

```
6. path('profile/',views.profile,name="profile"),
7. path('add/', views.adduser,name="add"),
8. path('get/',views.getuser,name="get")
9. ]
```

- **Serializers.py**

```
1. from rest_framework import serializers
2. from base.models import User
3. class UserSerializer(serializers.ModelSerializer):
4. class Meta:
5. model = User
6. fields = '__all__'
```

Step 11: open sub folder program14
Add the below code

- **Urls.py**

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

3. urlpatterns = [
4. path('admin/', admin.site.urls),
5. path('',include('api.urls'))
6. ]
```

- **Settings.py**

```
1. import os
2. installed apps => 'rest_framework',
3.     'base',
4. templates =>DIRS =>os.path.join(BASE_DIR,'templates')
```

Step 12: open sub folder base
Add the below code

- **Models.py**

```
1. from django.db import models

2. # Create your models here.
3. class User(models.Model):
4. userid = models.IntegerField()
5. username = models.CharField(max_length=200)
6. occupation = models.CharField(max_length=200)
7. email = models.CharField(max_length=200)
8. created_date = models.DateTimeField(auto_now_add=True)
```

Again open inside subfolder **migrate**
Create one file

- **000_initial.py**

Add the below code

```
from django.db import migrations, models


class Migration(migrations.Migration):

initial = True

dependencies = [
]

operations = [
migrations.CreateModel(
name='User',
fields=[
    ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
    ('userid', models.IntegerField()),
    ('username', models.CharField(max_length=200)),
    ('occupation', models.CharField(max_length=200)),
    ('email', models.CharField(max_length=200)),
    ('created_date', models.DateTimeField(auto_now_add=True)),
],
),
]
```

Step 13: create and open sub folder templates
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>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<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>
<!-- main menu bar -->
<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="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
<li class="nav-item">
    a. <a class="nav-link active" aria-current="page" href="#">Home</a>
</li>
<li class="nav-item">
    a. <a class="nav-link" href="/register">REGISTER</a>
</li>
<li class="nav-item">
    a. <a class="nav-link" href="/profile">PROFILE</a>
</li>
</ul>
</div>
</div>
</nav>
</body>
</html>
```

- **Register.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Register</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<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>
<!-- registration form -->
<div id="ucard" class="card shadow-sm w-50 mt-5 p-2 mx-auto ">
<form id="addUserForm">
<div data-mdb-input-init class="form-outline mb-4">
<input type="text" id="userid" class="form-control" />
<label class="form-label" for="form2Example1">USER ID</label>
</div>
<div data-mdb-input-init class="form-outline mb-4">
<input type="text" id="name" class="form-control" />
<label class="form-label" for="form2Example2">USER NAME</label>
</div>
<div data-mdb-input-init class="form-outline mb-4">
<input type="text" id="occ" class="form-control" />
<label class="form-label" for="form2Example3">OCCUPATION</label>
</div>
<div data-mdb-input-init class="form-outline mb-4">
<input type="text" id="email" class="form-control" />
<label class="form-label" for="form2Example4">EMAIL</label>
</div>
<button  type="submit" data-mdb-button-init data-mdb-ripple-init class="btn btn-primary btn-block mb-4">SAVE</button>
</form>
</div>
<div class="w-50 mt-5 p-2 mx-auto ">
<h4 id="success" style="display:none">User Added Successfully</h4>
<a href="/" data-mdb-button-init data-mdb-ripple-init class="btn btn-primary btn-block mb-4">BACK</a>
</div>
</body>
<script>
$(document).ready(function() {
$('#addUserForm').submit(()=>{
event.preventDefault();
var userid = $("#userid").val();
var name=$("#name").val();
var occupation=$("#occ").val();
var email=$("#email").val();
//save data to django db
$.ajax({
url:'{% url "add" %}',
type:"POST",
contentType:"application/json",
dataType:"json",
data: JSON.stringify({
userid:userid,
username: name,
occupation: occupation,
```

```
57. email:email
58. }),
59. success: function(data) {
        a. var responseData=data;
        b. console.log(responseData)
        c. if(responseData!="")
        d. {
        e. $("#success").show()
        f. }
        g. },
        h. error: function(data ){
        i. console.log(data);
        j. }
        k. });
60. });
61. });
62. </script>
63. </html>
```

- **Profile.html**

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <title>Profile</title>
7. <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
   integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
   crossorigin="anonymous">
8. <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
   YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
   crossorigin="anonymous"></script>
9. <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
10. </head>
11. <body>
12. <!-- Search Bar -->
13. <div class="w-50 mt-5 p-2 mx-auto ">
14. <form id="findUserForm" class="d-flex" role="search">
15. <input id="email" class="form-control me-2" type="search" placeholder="Search Email" aria-
    label="Search">
16. <button class="btn btn-outline-success" type="submit">Search</button>
17. </form>
18. </div>
19. <!-- profile card template -->
20. <div id="ucard" style="display:none" class="card shadow-sm w-50 mt-5 p-2 mx-auto ">
21. <img class="avatar d-block mx-auto" width="200px" height="200px" src="https://cdn-icons-
```

```
    png.freepik.com/512/5733/5733290.png" alt="Icon" />
<h2 id="greet" class="text-center">Hi, </h2>
<table class="table table-bordered mt-3">
<tr>
    a. <th>USER ID</th>
    b. <td id="uid"></td>
</tr>
<tr>
    a. <th>USERNAME</th>
    b. <td id="uname"></td>
</tr>
<tr>
    a. <th>EMAIL</th>
    b. <td id="umail"></td>
</tr>
<tr>
    a. <th>OCCUPATION</th>
    b. <td id="uoccu"></td>
</tr>
</table>
</div>
<div class="w-50 mt-5 p-2 mx-auto ">
<h4 id="err" style="display:none">User Not Found</h4>
<a href="/" data-mdb-button-init data-mdb-ripple-init class="btn btn-primary btn-block mb-4">BACK</a>
</div>
</body>
<script>
$(document).ready(function() {
$('#findUserForm').submit(()=>{
event.preventDefault();
$("#ucard").hide();
var email=$("#email").val();
//find user on django db
$.ajax({
url:'{% url "get" %}',
type:"POST",
contentType:"application/json",
dataType:"json",
data: JSON.stringify({
email:email
}),
success: function(data) {
    a. var responseData=data;
    b. console.log(responseData)
    c. if(responseData.length!=0)
    d. {
    e. $("#ucard").show();
```

```
            f. $("#greet").text(responseData[0].username);
            g. $("#uid").text(responseData[0].userid);
            h. $("#uname").text(responseData[0].username);
            i. $("#umail").text(responseData[0].email);
            j. $("#uoccu").text(responseData[0].occupation);
            k. }
55. else{
            a. $("#err").show();
            b. }
            c. },
            d. error: function(data ){
            e. console.log(data);
            f. }
            g. });
56. });
57. });
58. </script>
59. </html>
```

Step 14:migrate the changes

- **py manage.py makemigrations**
- **py manage.py migrate**

Step 15: run the server

- **py manage.py runserver**

Step 16: open browser, run http://127.0.0.1:8000/

Output:-

Fig.1. Home Page

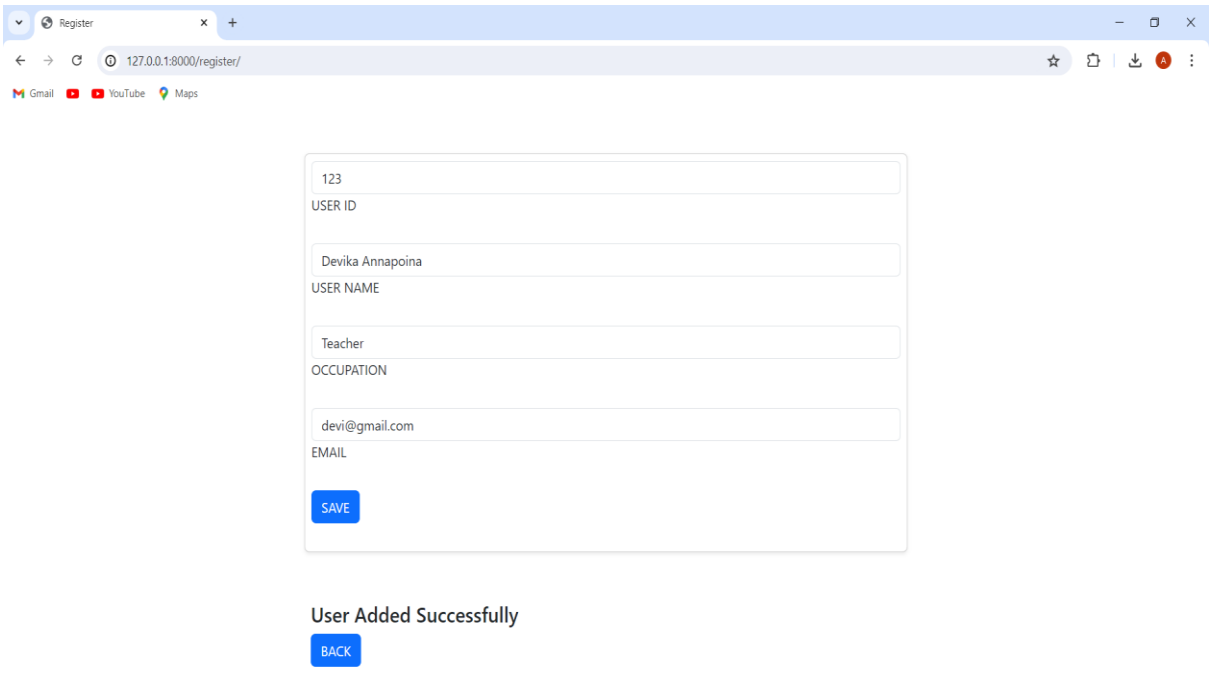

Fig.2.Register Page data added inside api

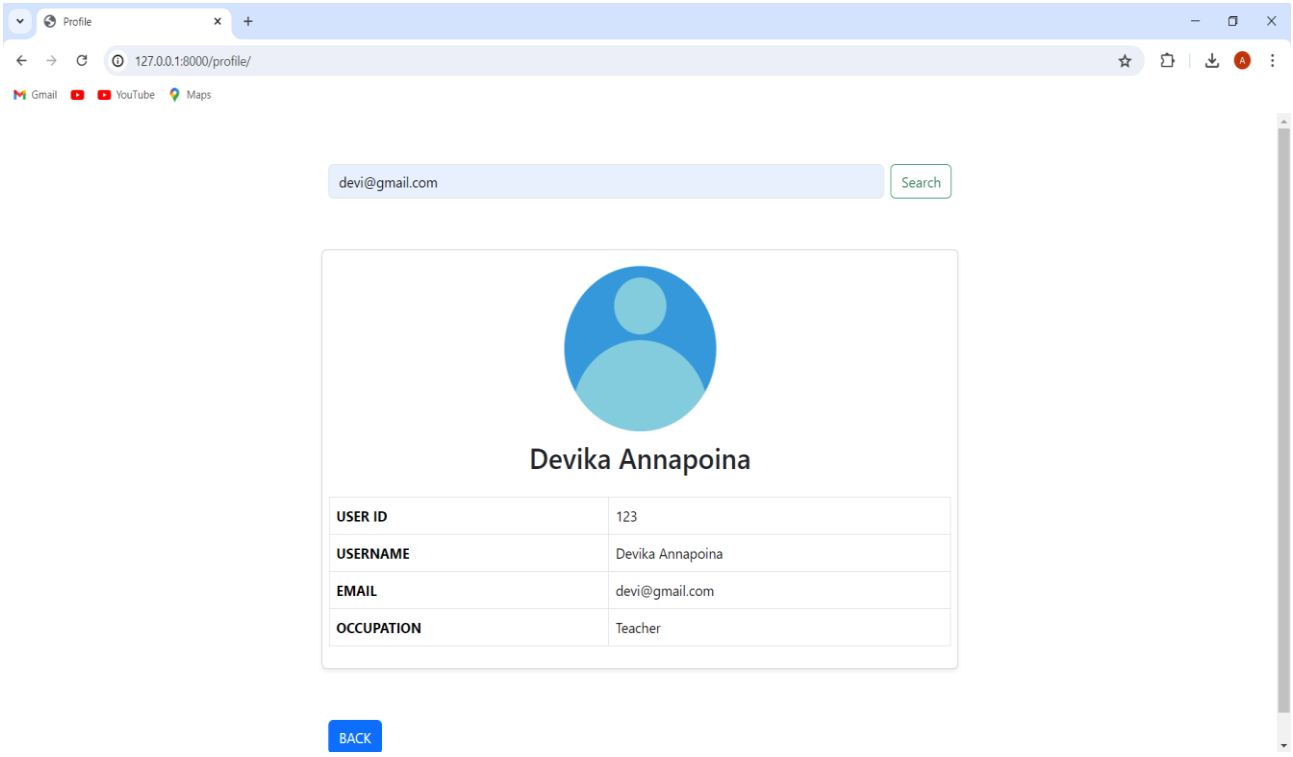

Fig.3.Fetch data from rest api (profile page)

VIVA QUESTIONS:

1.Why should we use Django ?

Ans:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It's designed to help developers take applications from concept to completion as quickly as possible. Here are several reasons why you might choose to use Django for your web development projects:

1.Rapid Development
2.Batteries Included
3.Highly Scalable
4.Security
5.Versatile
6.Vibrant Community
7.Good for SEO
8.Mature Software
9.Excellent Documentation
10.Python

2.Explain the concept of middleware in Django

Ans: Middleware in Django is a framework of hooks into Django's request/response processing. It's a lightweight, low-level plugin system for globally altering Django's input or output. Each middleware component is responsible for doing some specific function. For example, Django includes middleware to handle sessions, CSRF protection, authentication, and more.

Middleware components are executed during the request and response phases of a webpage and can modify the request before it reaches the view or the response before it's sent to the client. They are processed in order for requests, and in reverse order for responses, forming a layer around the view function.

Key Functions of Middleware:

Request Processing: Before a request reaches the view, middleware can process the request for various purposes like security checks, request modifications, user authentication, session management, etc. Response Processing: After a view has processed the request, middleware can modify the response before it's sent to the client. This is useful for tasks like setting cookies, caching, content compression, etc.

Exception Handling: Middleware can catch exceptions raised during request processing and offer custom error handling or logging. View Alteration: Middleware can modify or replace the view that Django

executes. This could be used for purposes like content negotiation or providing fallback views under certain conditions.

3.What is the significance of proxy models?

Ans: Proxy models in Django are a subclass of a database model that allows you to change the behavior of the model it inherits from. They are used when you want to modify the behavior of a model, including the default ordering, the verbose name, the available managers, or the Python-level methods, without creating a new table in the database. This means that a proxy model shares the same underlying database table as the model it's proxying for.

Key Uses and Significance of Proxy Models:
1.Different Behaviors for the Same Data
2.Custom Managers and Model Methods
3.Admin Interface Customization
4.Maintaining a Single Source of Truth
5.No Migration Required

4. What is difference between Django Vs react and which is the best ?

Ans: The "best" choice depends on the specific needs of your project:

For Backend Development: If you need a robust, secure backend with lots of out-of-the-box functionalities like an admin interface, user authentication, and database management, Django is an excellent choice. It's particularly strong for applications where the backend logic and database design are complex.

For Frontend Development: If your project requires a dynamic, responsive user interface with lots of client-side interactivity, React is the way to go. It's optimized for building fast, interactive UIs where the page does not need to be reloaded.

5.What is RESTful API?

Ans: A RESTful API (Representational State Transfer Application Programming Interface) is an architectural style for designing networked applications. It relies on a stateless, client-server, cacheable communications protocol — in virtually all cases, the HTTP protocol. The idea of REST is to treat all server-side resources as objects that can be created, read, updated, or deleted (CRUD operations) using HTTP methods (verbs) such as GET, POST, PUT, DELETE, and PATCH.