**Aim:** Create a web application to display “Hello world “ (with django)

**Steps:**

Install python :

**> sudo apt-get update**

**> sudo apt install python3.8**

Install pip in python:

**> sudo apt install python3-pip**

Set virtual environment:

**> sudo apt install python3-virtualenv**

**> virtualenv env**

**> sudo apt install .env/bin/activate**

**Step 1:**

Install Django Make sure you have Python and pip (the Python package manager)

installed on your system. Then, open a terminal and run the following command to install

Django:

**> pip install django** (use command line prompt)

**Step 2:**

Create a new Django project using the following

command:

(to create django admin, sudo apt install python3-django)

**> django-admin startproject helloworld** (use command line prompt)

This creates a new directory “hellloworld” with basic django structure

**Step 3:**

Create a Django App Inside the project directory using the

following command:

**> cd helloworld** (use command line prompt)

**> python manage.py startapp myapp**

This will create a new directory called “myapp" with the basic structure of a Django app.

**Step 4:**

Define a View Open the "views.py" file inside the "myapp" directory and define a

simple view that returns "Hello, World!" as the response:

(open Visual studio)

(in VS code, open helloworld project folder)

(in that open views.py file)

**from django.http import HttpResponse**

**def hello(request):**

**return HttpResponse("Hello, World!")**

**Step 5:**

Define a URL Create a "urls.py" file inside the "myapp" directory if it doesn’t

already exist, and define a URL pattern that maps to the view you defined in the previous

**from django.urls import path**

**from . import views**

**urlpatterns = [**

**path(‘hello/', views.hello, name='hello'),**

**]**

**Step 6:**

Include the App URLs in Project URLs Open the “urls.py” file in the project

directory and include the app's URLs by adding the following code:

**from django.urls import path, include**

**urlpatterns = [**

**path(‘myapp/', include('myapp.urls')),**

**]**

**Step 7:**

Run the Development Server Start the Django development server by running the

following command in the project directory:

**> python manage.py runserver** (in command line prompt)

**Step 8:**

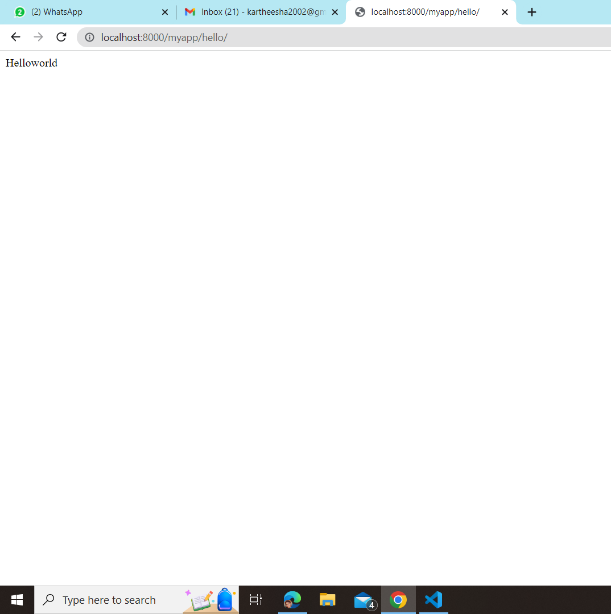
Test the "Hello World" Application Open a web browser and go to

**"localhost:8000/myapp/hello/"**

You should see "Hello, World!" displayed in your browser,

indicating that your "Hello World" application using Django is working correctly.

**Output:**



**Aim:**Create a User registration web application to store the login name ,password, mail id,contact number,abd address in the database.

**Steps:**

**Step 1:**

Activate virtual environment:

**> source env/bin/activate**

Create a Django project and app:

Assuming you have Django installed, open a terminal and run the following commands:

**> django-admin startproject registrationproject**

**> cd myproject**

**> python manage.py startapp registration**

This will create a Django project named "registrationproject" and an app named "registration".

**Step 2:**

In the models.py file of the "registration" app, define the User model with the fields as follows:

**from django.db import models**

**class User(models.Model):**

**login\_name = models.CharField(max\_length=255)**

**password = models.CharField(max\_length=255)**

**mailid = models.EmailField()**

**contact\_num = models.CharField(max\_length=10)**

**address = models.TextField()**

**Step 3:**

In the forms.py file of the "registration" app, create a registration form using Django's built-in

ModelForm:

**from django import forms**

**from .models import User**

**class UserRegistrationForm(forms.ModelForm):**

**class Meta:**

**model = User**

**fields = ['login\_name', 'password', 'mailid', 'contact\_num', 'address']**

**widgets = {**

**'password': forms.PasswordInput()**

**}**

**Step 4:**

Create a registration view:

In the views.py file of the "registration" app, create a view for user registration:

Code:

**from django.http import HttpResponse**

**from django.shortcuts import render, redirect**

**from .forms import UserRegistrationForm**

**def register(request):**

**if request.method == 'POST':**

**form = UserRegistrationForm(request.POST)**

**if form.is\_valid():**

**form.save()**

**return redirect('display') # replace 'home' with the name of your home page URL**

**else:**

**form = UserRegistrationForm()**

**return render(request, 'registration/register.html', {'form': form})**

**def display(request):**

**return HttpResponse("done")**

**Step 5:**

Create a registration template:

Create a register.html file in a folder named registration inside the templates folder of

your app folder. This will be the registration page template where users can input their registration

information.

**<form method="post">**

**{% csrf\_token %}**

**{{ form.as\_p }}**

**<button type="submit">Register</button>**

**</form>**

(if we use register.html in render on above views.py, then no need to create registration folder under

templates)

**Step 6:**

In the urls.py file of the "registration" app, add the following URL pattern:

**from django.urls import path**

**#from .views import register**

**from . import views**

**urlpatterns = [**

**path('register/', views.register, name='register'),**

**path('display/', views.display, name='display'),**

**]**

Include the app URLs in the project URLs:

In the urls.py file of your Django project, include the app URLs by adding the following line to the

urlpatterns list:

**from django.contrib import admin**

**from django.urls import path, include**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('registration/', include('registration.urls')),**

**]**

**Step 7:**

make migrations:

**> python manage.py makemigrations**

**> python manege.py migrate**

**Step 8:**

Open the settings.py file of your Django project, located at myproject/settings.py.

In the INSTALLED\_APPS setting, add the app containing the User model, in this case,

"registration", to the list of installed apps. Make sure to include the full path to the app, relative to

the project's root directory. For example:

**INSTALLED\_APPS = [**

**# other apps**

**'registration',**

**]**

**Step 9:**

Run the Django development server:

Finally, run the Django development server using the following command:

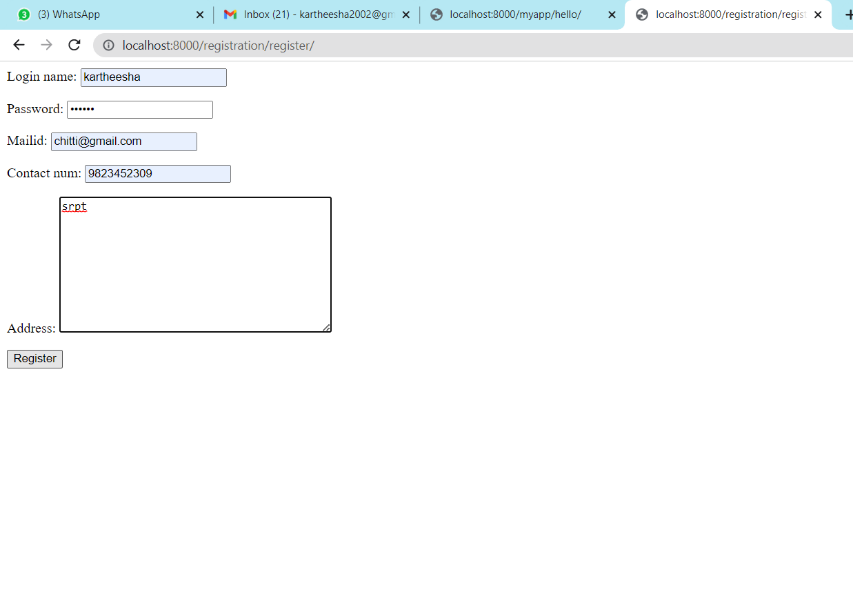
**> python manage.py runserver**

You should now be able to access the registration page at

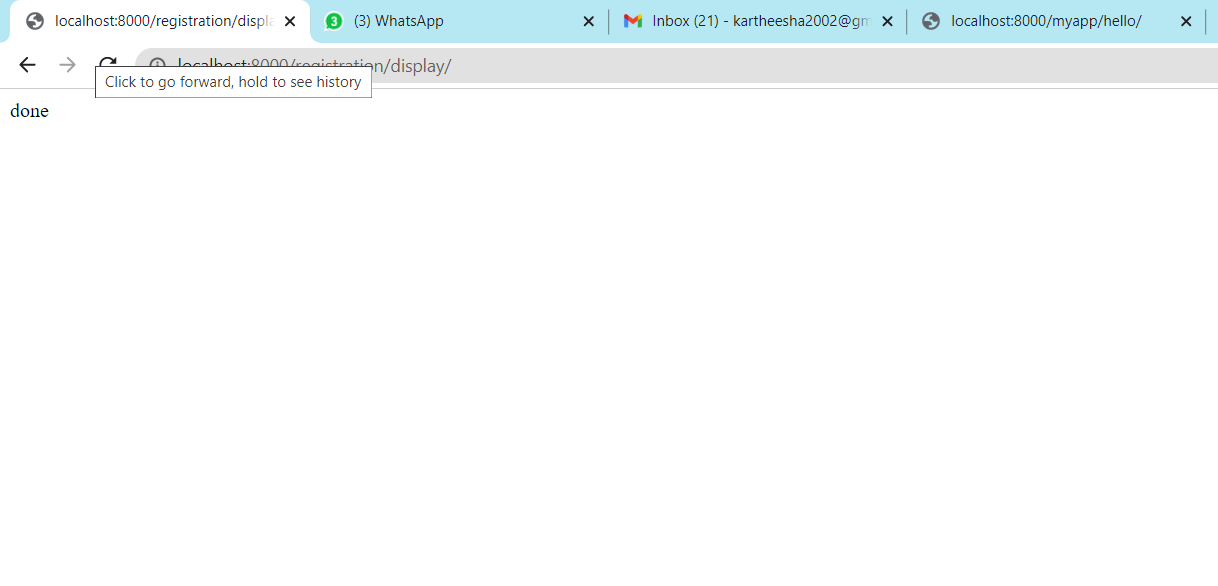
**localhost:8000/registrationproject/register/**

and register a user with the specified information.

**Output:**

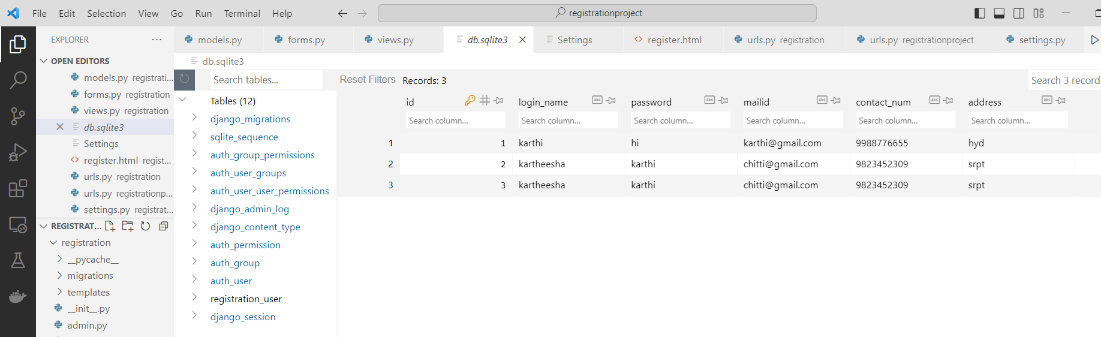


Click on register



after successful registration form execution, go to db.sqlite3 in VS code. There we can observe data in

registeration\_user



**Aim:**

Implement web application with Django to validate User, where the user submits the

login name and password to the server. The name and password are checked against the

data already available in Database and if the data matches, a successful login page is

retuned, which also contain Logout option to exit the page. Otherwise, a failure message is

shown to the user.

**Steps:**

**Step 1:**

Activate virtual environment:

**> source env/bin/activate**

**Step 2:**

Start a new Django project:

**> django-admin startproject user\_validation**

**> cd user\_validation**

**Step 3:**

Create a new Django app:

**> python manage.py startapp loginform**

**Step 4:**

Define the User model

In the loginform/models.py file, define a basic User model with name and password fields:

Code:

**from django.db import models**

**# Create your models here.**

**class User(models.Model):**

**name=models.CharField(max\_length=100)**

**password=models.CharField(max\_length=100)**

**Step 5:**

Create the login view

In the loginform/views.py file, define a view function to handle the login logic:

Code:

**from django.shortcuts import render, redirect**

**from .models import User**

**def login(request):**

**if request.method == 'POST':**

**username = request.POST['username']**

**password = request.POST['password']**

**try:**

**#return render(request,'index.html',{'error\_message':password})**

**user =User.objects.get(username=username, password=password)**

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

**except User.DoesNotExist:**

**return render(request, 'index.html', {'error\_message': 'Invalid Login'})**

**return render(request, 'index.html')**

**def logout(request):**

**return render(request,'logout.html')**

**Step 6:**

Create the following templates in the authentication/templates/login directory:

In the templates folder, create a folder named login and create the following files with the following

**index.html:**

**<html>**

**<head>**

**<title>Login</title>**

**</head>**

**<body>**

**<h1>Login</h1>**

**<form method="post">**

**{% csrf\_token %}**

**<p>Username: <input type="text" name="username"></p>**

**<p>Password: <input type="password" name="password"></p>**

**<p><input type="submit" value="Login"></p>**

**</form>**

**{% if error\_message %}**

**<p style="color: red">{{ error\_message }}</p>**

**{% endif %}**

**</body>**

**</html>**

**logout.html:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Logout</title>**

**</head>**

**<body>**

**<h1>Logout Successful</h1>**

**<p>You have been logged out.</p>**

**</body>**

**</html>**

**Success.html:**

**<html>**

**<head>**

**<title>Success</title>**

**</head>**

**<body>**

**<h1>Login Successful</h1>**

**<a href="{% url 'logout' %}">Logout</a>**

**</body>**

**</html>**

**Step 8:**

Define URLs

In the authentication app create a file named urls.py:

**from django.urls import path**

**from . import views**

**urlpatterns = [**

**path('login/', views.login, name='login'),**

**path('logout/', views.logout, name='logout'),**

**]**

In the user\_validation /urls.py file, define the URLs for the login and logout views:

**from django.contrib import admin**

**from django.urls import path, include**

**from authentication import views**

**urlpatterns = [**

**path('loginform/',include('loginform.urls')),**

**]**

**Step 9:**

Add ‘loginform’ app name in settings.py file of user\_validation folder

**INSTALLED\_APPS=[**

**‘loginform’,**

**]**

**Step 10:**

Create a data.py file in user\_validation and insert the following code:

**import sqlite3**

**conn = sqlite3.connect('db.sqlite3')**

**cursor = conn.cursor()**

**cursor.execute('''INSERT INTO loginform\_user VALUES (1, 'karthi', 'kar')''')**

**cursor.execute('''INSERT INTO loginform\_user VALUES (2, 'nani', 'csd')''')**

**# Display data inserted**

**print("Data Inserted in the table: ")**

**data=cursor.execute('''SELECT \* FROM loginform\_user''')**

**for row in data:**

**print(row)**

**conn.commit()**

**conn.close()**

**Step 10:**

Apply the migrations :

**> python manage.py makemigrations**

**> python manage.py migrate**

**Step 11:**

Finally, run the Django development server by running the following command:

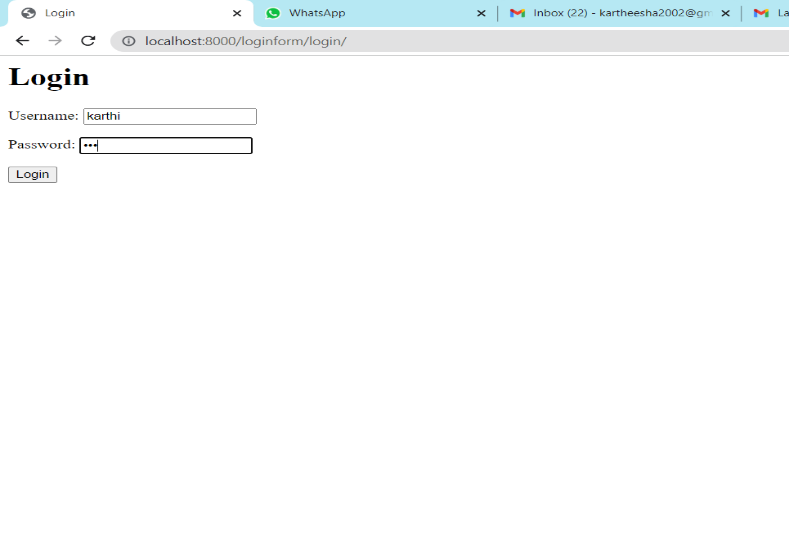
**> python manage.py runserver**

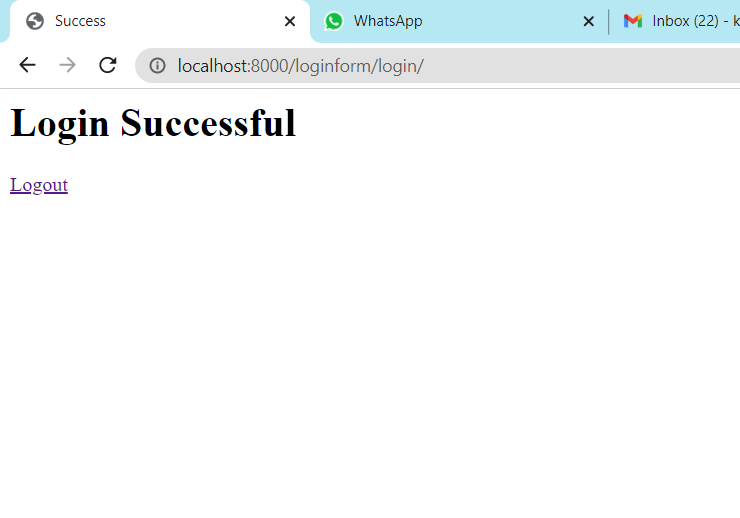
**Step 12:**

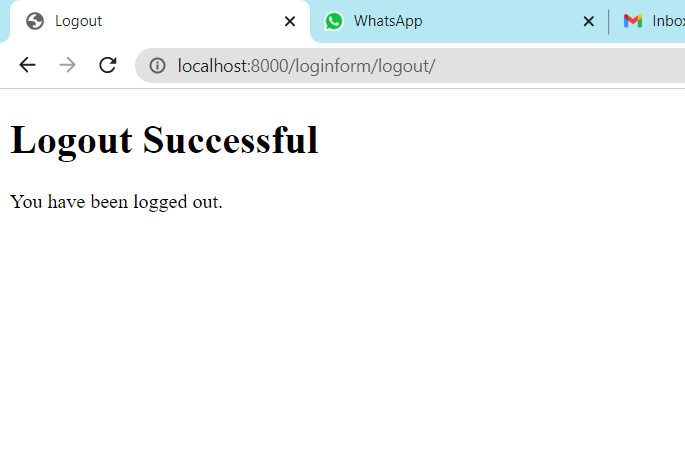
Open web browser and go to

**http://localhost:8000/loginform/login/**

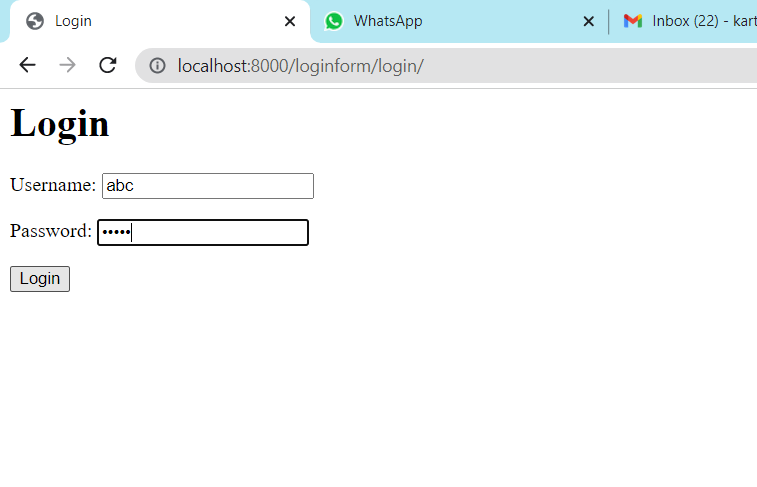
**Output:(Successful login)**

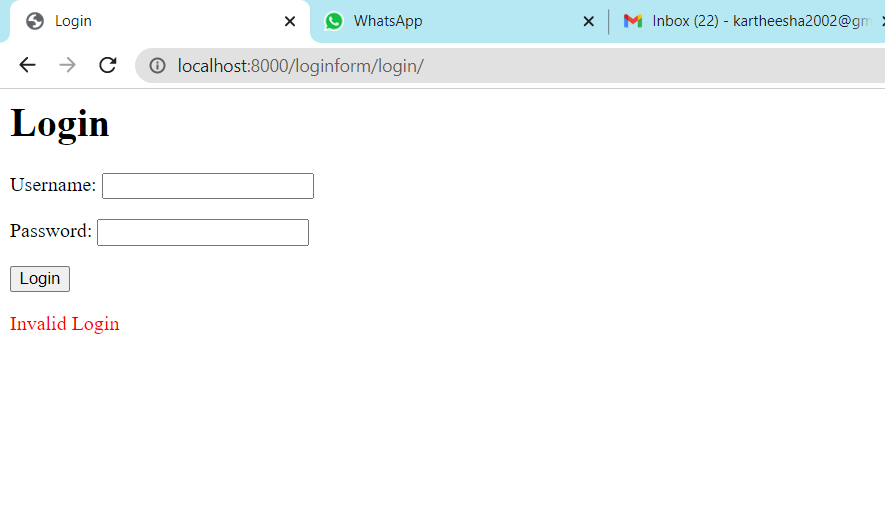






**Output:(invalid user)**





**AIM**: Implement web application with Django that lists all cookies stored in the browser on clicking "List Cookies" button. Add cookies if necessary.

**STEPS:**

**Step 1:**

Activate virtual environment:

**> source env/bin/activate**

**Step 2:**

Start a new Django project:

**> django-admin startproject cookies\_app**

**> cd cookies\_app**

**Step 3:**

Create a new Django app:

**> python manage.py startapp cookies**

**Step 4:**

Create a view that returns a response with all the cookies in the browser:

In the cookies app, create a file named views.py and add the following code:

**from django.shortcuts import render**

**from django.http import HttpResponse**

**def show\_cookies(request):**

**cookies = request.COOKIES**

**return render(request, 'cookies/show\_cookies.html', {'cookies': cookies})**

**def setcookie(request):**

**response = HttpResponse("Cookie Set")**

**response.set\_cookie("example\_cookie", "example\_value")**

**return response**

**Step 5:**

Add a URL pattern that maps to the show\_cookies view:

In the cookies app, create a file named urls.py and add the following code:

Code:

**from django.urls import path**

**from . import views**

**urlpatterns = [**

**path('show\_cookies/', views.show\_cookies, name='show\_cookies'),**

**path('setcookie/', views.setcookie, name='setcookie'),**

**]**

**Step 6:**

Update the cookies\_app project's URL configuration to include the cookies app's URL

patterns:

In the cookies\_app project, open the urls.py file and add the following code:

**from django.urls import path, include**

**urlpatterns = [ path('cookies/', include('cookies.urls')), ]**

**Step 7:**

Create the template for the show\_cookies view in the templates folder:

In the templates folder, create a folder named cookies and create a file named show\_cookies.html

with the following code:

**<html>**

**<head>**

**<title>List Cookies</title>**

**</head>**

**<body>**

**<h1>List Cookies</h1>**

**<table>**

**{% for key, value in cookies.items %}**

**<tr>**

**<td>{{ key }}</td>**

**<td>{{ value }}</td>**

**</tr>**

**{% endfor %}**

**</table>**

**</body>**

**</html>**

**Step 8:**

Add ‘cookies’ app name in settings.py file of cookies\_app folder

**INSTALLED\_APPS=[ ‘cookies’, ]**

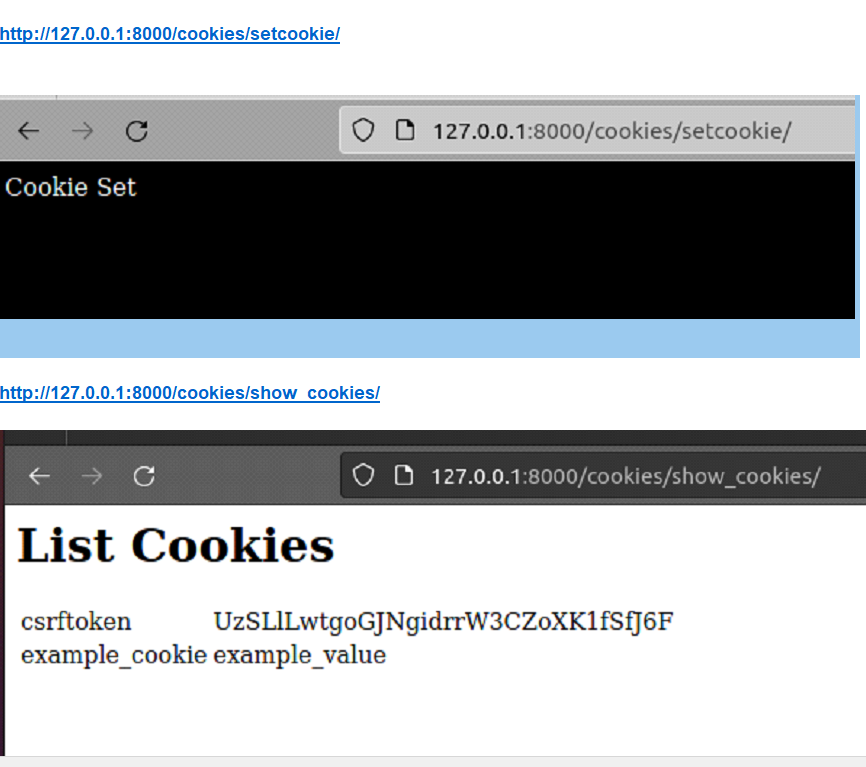
**Step 9:**

Finally, run the Django development server by running the following command:

> **python manage.py runserv**

**Step 10:**

Open web browser and go to



**Aim:**Implement web application with Django to track the user which counts the number of

times the user visited the site.

**Steps:**

**1**.Activate virtual environment:

**source env/bin/activate**

**2**. Start a new Django project:

**django-admin startproject visit\_counter**

**cd visit\_counter**

**3**. Create a new Django app:

**python manage.py startapp counter**

**4.**Configure Database and adding ’counter’

Open the visit\_counter/settings.py file and update the DATABASES section with your database settings.

**import os**

**DATABASES = {**

**'default': {**

**'ENGINE': 'django.db.backends.sqlite3',**

**'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),**

**}**

**}**

**INSTALLED\_APPS=[**

**‘counter’,**

**]**

**5**.Run database migrations to create the necessary tables:

**python3 manage.py migrate**

**6.**Open the ‘counter/models.py’ file and define a model to track the user visits:

**from django.db import models**

**class Visitor(models.Model):**

**ip\_address = models.GenericIPAddressField(unique=True)**

**visit\_count = models.PositiveIntegerField(default=0)**

**def \_\_str\_\_(self):**

**return self.ip\_address**

**7**.Generate and apply the migrations for the new model:

**python3 manage.py makemigrations**

**python3 manage.py migrate**

**8**.Open the ‘counter/views.py’ file and define a view to handle the user visits:

**from django.shortcuts import render**

**from .models import Visitor**

**def home(request):**

**ip\_address = request.META.get('REMOTE\_ADDR')**

**visitor, created = Visitor.objects.get\_or\_create(ip\_address=ip\_address)**

**visitor.visit\_count += 1**

**visitor.save()**

**context = {**

**'visitor': visitor,**

**}**

**return render(request, 'counter/home.html', context)**

**9**.Create a ‘counter/templates/counter’ directory.

Inside the ‘counter/templates/counter’ directory, create a file called home.html and add the following content:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Visit Counter</title>**

**</head>**

**<body>**

**<h1>Welcome to the Visit Counter</h1>**

**<p>Your IP address: {{ visitor.ip\_address }}</p>**

**<p>Number of visits: {{ visitor.visit\_count }}</p>**

**</body>**

**</html>**

**10**.Open the ‘visit\_counter/urls.py’ file and update it to include the URL pattern for the home view:

**from django.contrib import admin**

**from django.urls import path**

**from counter.views import home**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('', home, name='home'),**

**]**

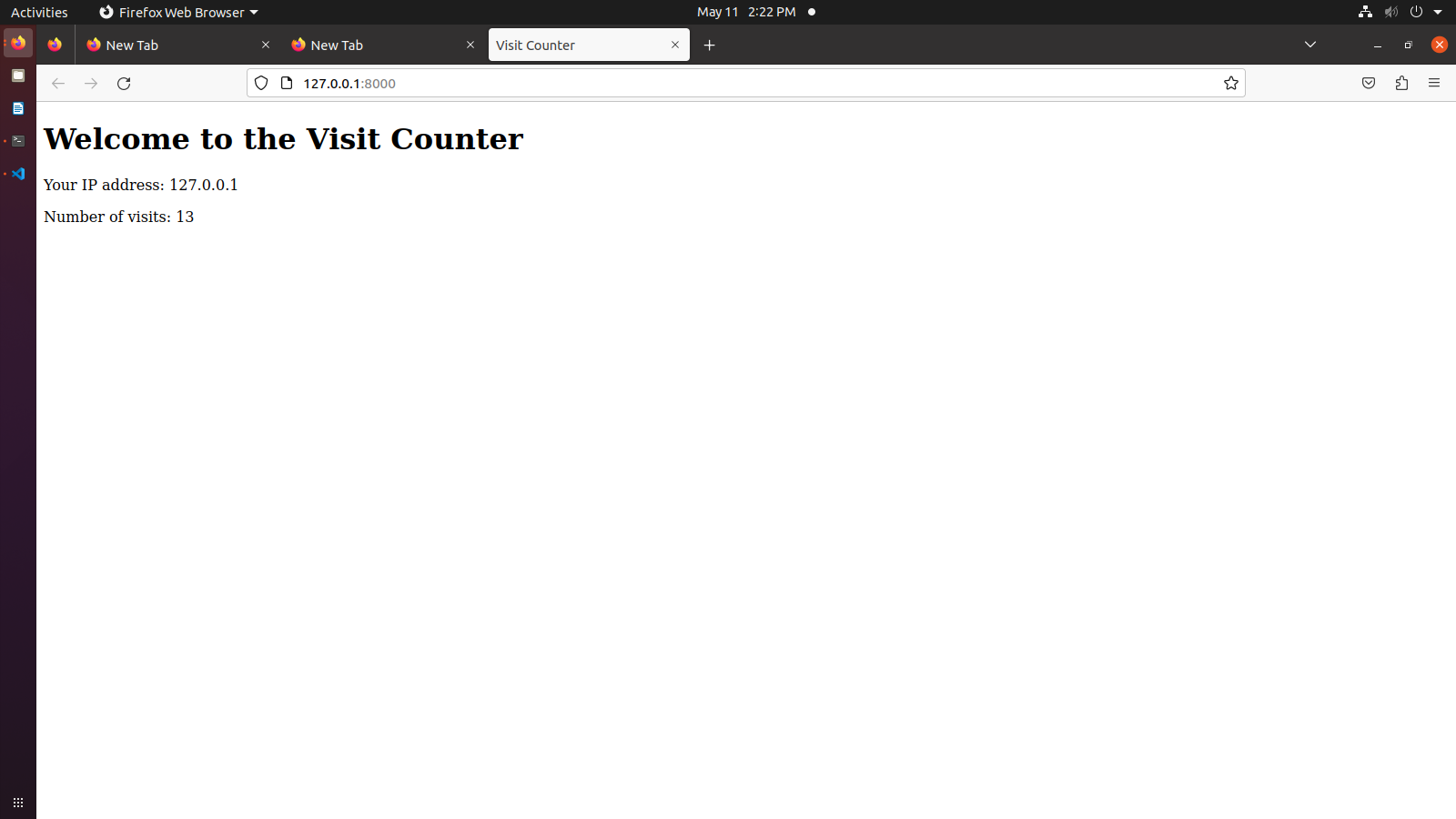
**11**.Run the Application:

**python manage.py runserver**

**12.**open browser and run:

**http://localhost:5000/**

**OUTPUT:**



**AIM:**Implement web application with Django to Create a model that saves user profile

(profile picture, resume in pdf, basic information).

**STEPS:**

1.Activate virtual environment:

**source env/bin/activate**

2. Start a new Django project:

**django-admin startproject profileproj**

**cd profileproj**

3. Create a new Django app:

**python manage.py startapp profileapp**

4.Open settings.py file in profileproj folder

(open Visual studio)

(in VS code, open profileproj project folder)

(in that open settings.py file)

**Add ‘profileapp’ to the ‘INSTALLED\_APPS’ list**

5.Define the profile model

(open the ‘profileapp/models.py’ file)

(Define a model called Profile with fields for basic information, profile picture, and resume. )

**from django.db import models**

**class Profile(models.Model):**

**name = models.CharField(max\_length=100)**

**email = models.EmailField()**

**profile\_picture = models.ImageField(upload\_to='profile\_pics')**

**resume = models.FileField(upload\_to='resumes')**

**created\_at = models.DateTimeField(auto\_now\_add=True)**

6.set up Database and migrations

Install Pillow

Pillow is a library working with images in Python.

**python -m pip install Pillow**

Create migration

**python manage.py makemigrations**

Run migration

**python manage.py migrate**

7.Create views

(open ‘profileapp/views.py’ file)

(Define views for creating and displaying profiles.)

**from django.shortcuts import render, redirect**

**from .forms import ProfileForm,Profile**

**def create\_profile(request):**

**if request.method == 'POST':**

**form = ProfileForm(request.POST, request.FILES)**

**if form.is\_valid():**

**form.save()**

**return redirect('profile\_ack’)**

**else:**

**form = ProfileForm()**

**return render(request, 'create\_profile.html', {'form': form})**

**def profile\_ack(request):**

**profiles = Profile.objects.all()**

**return render(request, 'profile\_ack.html', {'profiles': profiles})**

8. create ‘**forms.py’** in profileapp to handle the user input

**from django import forms**

**from .models import Profile**

**class ProfileForm(forms.ModelForm):**

**class Meta:**

**model = Profile**

**fields = ['name', 'email', 'profile\_picture', 'resume']**

9. Create the following templates in the profileapp/templates directory:

In the templates folder, create a file named **create\_profile.html**

**create\_profile.html:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Create Profile</title>**

**</head>**

**<body>**

**<h1>Create Profile</h1>**

**<form method="POST" enctype="multipart/form-data">**

**{% csrf\_token %}**

**{{ form.as\_p }}**

**<input type="submit" value="Save">**

**</form>**

**</body>**

**</html>**

**profile\_ack.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Profile List</title>**

**</head>**

**<body>**

**<h1>Profile Updated Succesfully</h1>**

**</body>**

**</html>**

10. Define URLs

In the profileapp create a file named urls.py:

**from django.urls import path**

**from profileapp import views**

**urlpatterns = [**

**path('create/', views.create\_profile, name='create\_profile'),**

**path('ack/', views.profile\_ack, name='profile\_ack'),**

**]**

In the profileproj/urls.py file, define the URLs :

**from django.contrib import admin**

**from django.urls import path, include**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('profileapp/',include('profileapp.urls')),**

**]**

11.Finally, run the Django development server by running the following command:

**python manage.py runserver**

12.Open web browser and go to:

[http://127.0.0.1:8000/profileapp/create/](http://localhost:8000/authentication/login/)

**OUTPUT:**

