Experiment No 7

Aim: To create a web application using Django framework to demonstrate user login and registration.

Theory:

Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks.
When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

Django is based on **MVT (Model-View-Template)** architecture. MVT is a software design pattern for developing a web application.

MVT Structure has the following three parts –
Model: The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres).

View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.

Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

Project Structure :

A Django Project when initialized contains basic files by default such as manage.py, view.py, etc. A simple project structure is enough to create a single-page application. Here are the major files and their explanations. Inside the website folder (project folder) there will be the following files

manage.py- This file is used to interact with your project via the command line(start the server, sync the database… etc). For getting the full list of commands that can be executed by manage.py type this code in the command window

python manage.py help

folder (website) – This folder contains all the packages of your project. Initially, it contains four files –

- **_init_.py –** It is a python package. It is invoked when the package or a module in the package is imported. We usually use this to execute package initialization code, for example for the initialization of package-level data.
- **settings.py –** As the name indicates it contains all the website settings. In this file, we register any applications we create, the location of our static files, database configuration details, etc.
- **urls.py –** In this file, we store all links of the project and functions to call. · **wsgi.py –** This file is used in deploying the project in WSGI. It is used to help your Django application communicate with the webserver.

Installation of Django

Use command**:** pip install Django

Wait for the django to be downloaded and installed at the same time. Then After that type "python -m django version" and hit enter to check if django is installed and what version of django is.

python -m django version

Creating the App

After setting django we will now create the web app for the web server. First create a new folder named " RegistrationAndLogin", then cd to a newly created folder, then type "**django-admin startproject website**" and hit enter. A new folder will be created on the directory named 'website'

Running The Server

After creating a project, cd to the newly created directory, then type "manage.py runserver" and hit enter to start the server running. The "manage.py" is a command of django-admin that utilize the administrative tasks of python web framework.

Type '127.0.0.1:8000' in the url browser to view the server. When there is code changes in the server just (ctrl + C) to command prompt to stop the server from running, then start again to avoid errors.

Creating The Website

This time will now create the web app to display the web models. First locate the directory of the app via command prompt cd, then type "manage.py startapp web" and hit enter. A new directory will be create inside the app named "web".

Setting up The URL

This time will now create a url address to connect the app from the server. First Go to website directory, then open urls via Python IDLE's or any text editor. Then import "include" module beside the url module and import additional module to make a redirect url to your site "from . import views". After that copy/paste the code below inside the urlpatterns.

1. url(r'^$', views.index_redirect, name='index_redirect'),
2. url(r'^web/', include('web.urls')),

It will be look like this:

1. **from django.urls import include,re_path**
2. **from** django.contrib **import** admin
3. **from** . **import** views

.

5. urlpatterns = [
6. re_path(r'^$', views.index_redirect, name='index_redirect'),
7. re_path(r'^web/', include('web.urls')),
8. re_path(r'^admin/', admin.site.urls),
9.]

Then after that create a view that will catch the redirect url. To do that create a file "views.py" then copy/paste the code below and save it as "views.py".

1. **from** django.shortcuts **import** redirect
2. **from** . **import** views
3.
4. **def** index_redirect(request):
5. **return** redirect('/web/')

Creating The Path For The Pages

Now that we set the connect we will now create a path for the web pages. All you have to do first is to go to web directory, then copy/paste the code below and save it inside "web" directory named 'urls.py' The file name must be urls.py or else there will be an error in the code.

```
1. from django.urls import include,re_path
2. from . import views
3.
4. urlpatterns = [
5. re_path(r'^$', views.index, name='index'),
6. re_path(r'^login/$', views.login, name='login'),
7. re_path(r'^home/$', views.home, name='home'),
8. ]
```

Creating A Static Folder

This time we will create a directory that store an external file. First go to the web directory then create a directory called "static", after that create a sub directory called "web". You'll notice that it is the same as your main app directory name, to assure the absolute link. This is where you import the css, js, etc directory.

Creating The Views

The views contains the interface of the website. This is where you assign the html code for rendering it to django framework and contains a methods that call a specific functions. To do that first open the views.py, the copy/paste the code below.

```
2. from django.shortcuts import render, redirect, HttpResponseRedirect 3.
from .models import Member
4. # Create your views here.
5.
6. def index(request):
7. if request.method == 'POST':
8. member = Member(username=request.POST['username'],
password=request.POST['password'], firstname=request.POST['firstname'],
lastname=request.POST['lastname'])
9. member.save()
10. return redirect('/')
11. else:
12. return render(request, 'web/index.html')
13.
14. def login(request):
15. return render(request, 'web/login.html')
```

```
16. 16.
17. def home(request):
18. if request.method == 'POST':
19. if Member.objects.filter(username=request.POST['username'],
password=request.POST['password']).exists():
20. member = Member.objects.get(username=request.POST['username'],
password=request.POST['password'])
21. return render(request, 'web/home.html', {'member': member}) 22. else:
23. context = {'msg': 'Invalid username or password'}
24. return render(request, 'web/login.html', context)
```

Creating The Models

Now that we're done with the views we will then create a models. Models is module that will store the database information to django. To do that locate and go to web directory, then open the "models.py" after that copy/paste the code.

```
1. from django.db import models
2.
3. # Create your models here.
4.
5. class Member(models.Model):
6. firstname=models.CharField(max_length=30)
7. lastname=models.CharField(max_length=30)
8. username=models.CharField(max_length=30)
9. password=models.CharField(max_length=12)
10.
11. def__str__(self):
12. return self.firstname + " " + self.lastname
```

Registering The App To The Server

Now that we created the interface we will now then register the app to the server. To do that go to the website directory, then open "settings.py" via Python IDLE's or any text editor. Then copy/paste this script inside the INSTALLED_APP variables 'web'. It will be like this:

```
1. INSTALLED_APPS = [
2. 'web',
3. 'django.contrib.admin',
4. 'django.contrib.auth',
5. 'django.contrib.contenttypes',
6. 'django.contrib.sessions',
7. 'django.contrib.messages',
8. 'django.contrib.staticfiles',
9. ]
```

Creating The Mark up Language

Now we will create the html interface for the django framework. First go to web directory, then create a directory called "templates" and create a sub directory on it called web.

base.html

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8" name="viewport" content="width=device-width, initial
scale=1"/>
5. {% load static %}
6. <link rel="stylesheet" type="text/css" href="{% static 'web/css/bootstrap.css' %}" />
7. </head>
8. <body>
9. <nav class="navbar navbar-default">
10. <div class="container-fluid">
11. <a class="navbar-brand" href="give any link ">Experiment No 7</a> 12.
</div>
13. </nav>
14. <div class="col-md-3"></div>
15. <div class="col-md-6 well">
16. <h3 class="text-primary">Python - Simple Registration & Login Form</h3> 17. <hr
style="border-top:1px dotted #000;"/>
18. {% block body %}
19. {% endblock %}
20. </div>
21. </body>
22. </html>
```

Save it as "base.html" inside the web directory "sub directory of

templates". **index.html**

```
1. {% extends 'web/base.html' %}
2. {% block body %}
3. <form method="POST">
4. {% csrf_token %}
5. <div class="col-md-2">
6. <a href="{% url 'login' %}">Login</a>
7. </div>
8. <div class="col-md-8">
9. <div class="form-group">
10. <label for="username">Username</label>
11. <input type="text" name="username" class="form-control"
required="required">
12. </div>
13. <div class="form-group">
14. <label for="password">Password</label>
15. <input type="password" name="password" class="form-control"
required="required"/>
```

```
16. </div>
17. <div class="form-group">
18. <label for="firstname">Firstname</label>
19. <input type="text" name="firstname" class="form-control"
required="required"/>
20. </div>
21. <div class="form-group">
22. <label for="lastname">Lastname</label>

23. <input type="text" name="lastname" class="form-control"
required="required"/>
24. </div>
25. <br />
26. <div class="form-group">
27. <button type="submit" class="btn btn-primary form-control"><span
class="glyphicon glyphicon-save"></span> Submit</button>
28. </div>
29. </div>
30. </form>
31. {% endblock %}
```

Save it as "index.html" inside the web directory "sub directory of

templates". **login.html**

```
1. {% extends 'web/base.html' %}
2. {% block body %}
3. <form method="POST" action="{% url 'home' %}">
4. {% csrf_token %}
5. <div class="col-md-2">
6. <a href="{% url 'index' %}">Signup</a>
7. </div>
8. <div class="col-md-8">
9. <div class="form-group">
10. <label for="username">Username</label>
11. <input type="text" name="username" class="form-control"
required="required"/>
12. </div>
13. <div class="form-group">
14. <label for="password">Password</label>
15. <input type="password" name="password" class="form-control"
required="required"/>
16. </div>
17. <center><label class="text-danger">{{ msg }}</label></center> 18. <br
/>
```

```
19. <div class="form-group">
20. <button class="btn btn-primary form-control" type=”submit”><span
class="glyphicon glyphicon-log-in"></span> Login</button>
21. </div>
22. </div>
23. </form>
24.
25.
26.
27.
28.
24.
25. {% endblock %}
```

Save it as "login.html" inside the web directory "sub directory of templates".

home.html

```
1. {% extends 'web/base.html' %}
2. {% block body %}
3. <h2>Welcome</h2>
4. {{ member.firstname }}
5. {% endblock %}
```

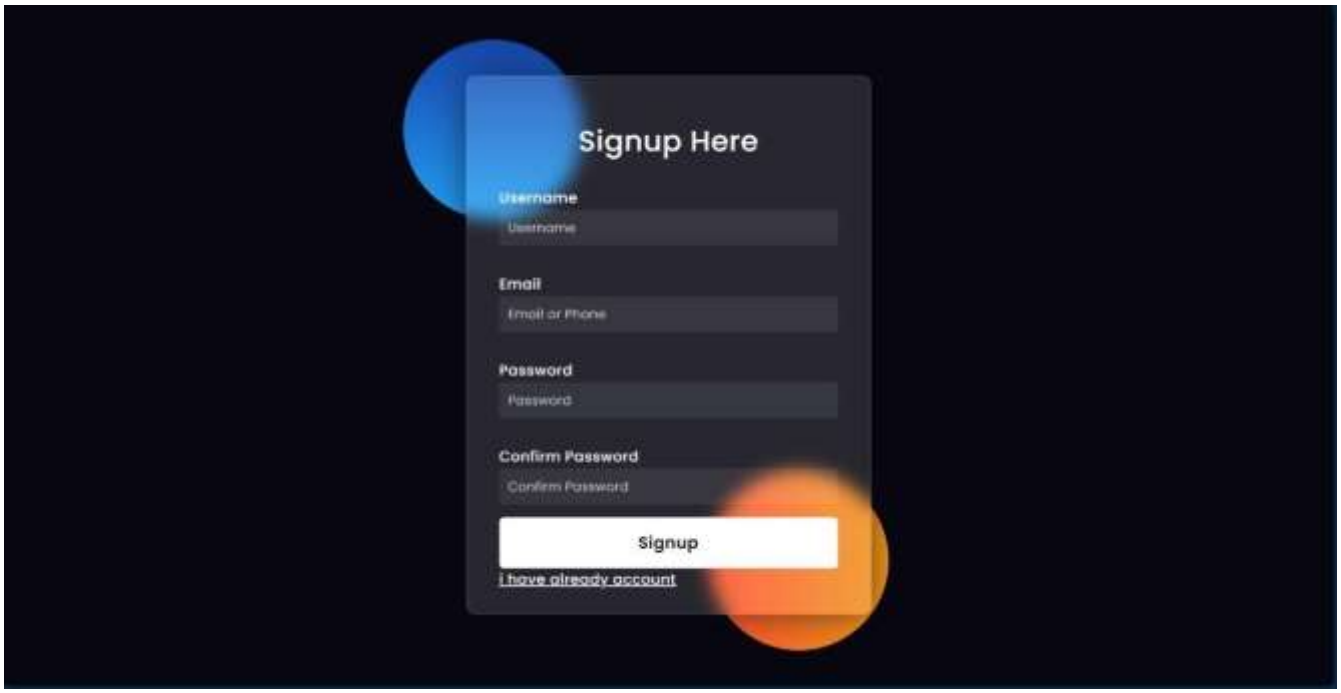
Save it as "home.html" inside the web directory "sub directory of

templates". **Migrating The App To The Server**

Now that we done in setting up all the necessary needed, we will now then make a migration and migrate the app to the server at the same time. To do that open the command prompt then cd to the "website" directory, then type "manage.py makemigrations" and hit enter. After that type again "manage.py migrate" then hit enter.

Now try to run the server again, using manage.py runserver and see if all things are done.

OUTPUT :


Signup Here
Username
Email
Email or Phone
Password
Password
Confirm Password
Confirm Password
Signup
I have already account


Login Here
Username
Enter Username
Password
Password
Log In
Create an account

Conclusion: the successful development of a web application using Django framework for user login and registration showcases its robustness and efficiency in handling authentication processes. This experiment underscores Django's capability in providing secure and user-friendlysolutions for web development, highlighting its relevance in modern application development paradigms.