**Django Auth User Signup and Login**

**Introduction**

**User** **authentication** and **authorization** are vital components of web applications. **Django**, a popular Python web framework, provides *robust* functionality for implementing user *login* and *signup* features.

**In this blog post,** we will explore how to create a user login and signup system using Django, and HTML forms. We will cover the necessary steps and provide valuable resources to help you successfully implement this functionality in your Django projects.

**Create a simple application**

**Prerequisites**

**Before diving into this tutorial,** ensure that you have the following prerequisites in place:

* Basic understanding of **Python** and **Django Framework**.
* **Python** (version *3.6* or above) and **pip** installed on your system.
* Familiarity with the Django framework and its **concepts**.

**Steps**

**1. Setting Up the Django Project**

**To start** building our simple application, we need to set up a new **Django**project. We’ll cover the installation process and project initialization.

**To get started,** follow these steps to set up the Django framework:

**Install**Django Framework using *pip*:

pip install django

**Create**a new Django **project**:

django-admin startproject myproject

**Create**a new Django **app**within your project:

cd myproject  
python manage.py startapp myapp

**2. Configure Django**

Open your Django project’s settings.py file and write the following code :

INSTALLED\_APPS = [  
 # ...  
 # ..  
 # .  
 # 👇 1. Add this line  
 'myapp',  
]  
  
TEMPLATES = [  
 {  
 # 👇 2. Add this line   
 'DIRS': ['templates'],  
   
 },  
]

**3. Add the URLs**

**In this,** For accessing our application myapp urls we have to add the following line to the myproject/urls.py file.

**Open**the urls.py from inside of myproject folder and write the following code:

from django.contrib import admin  
from django.urls import path, include # 👈 1. Add this line  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 # 👇 2. Add the app url on this  
 path('', include('myapp.urls'))  
]

**URL Configuration of views:**

**Create** a **new**file urls.py inside the myapp folder and write the below code:

from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('', views.index, name='home'),  
 path('login/', views.user\_login, name='login'),  
 path('signup/', views.user\_signup, name='signup'),  
 path('logout/', views.user\_logout, name='logout'),  
]

1. urlpatterns: This is a list that holds the URL patterns for the application. Each URL pattern is defined as an element in the list.
2. path('', views.index, name='home'): This line defines the URL pattern for the homepage of the application. The empty string '' represents the root URL. The views.index specifies that the index function in the views module will handle the logic for rendering the homepage. The name='home' is an optional parameter that assigns a name to the URL pattern, which can be used to refer to this URL pattern in other parts of the code.
3. path('login/', views.user\_login, name='login'): This line defines the URL pattern for the login page. The 'login/' string represents the URL path /login/. The views.user\_login specifies that the user\_login function in the views module will handle the logic for rendering the login page. The name='login' assigns a name to this URL pattern.
4. path('signup/', views.user\_signup, name='signup'): This line defines the URL pattern for the signup page. The 'signup/' string represents the URL path /signup/. The views.user\_signup specifies that the user\_signup function in the views module will handle the logic for rendering the signup page. The name='signup' assigns a name to this URL pattern.
5. path('logout/', views.user\_logout, name='logout'): This line defines the URL pattern for the logout page. The 'logout/' string represents the URL path /logout/. The views.user\_logout specifies that the user\_logout function in the views module will handle the logic for rendering the logout page. The name='logout' assigns a name to this URL pattern.

These URL patterns determine the mapping between the URLs entered by the users and the corresponding views that should be rendered. In this case, the views module contains functions like index, user\_login, user\_signup, and user\_logout, which handle the rendering of the respective pages.

**5. Add the View Function**

**Open,**The views.py from myapp folder and write the below code for showing and redirecting to our templates:

from django.shortcuts import render, redirect  
from django.contrib.auth import authenticate, login, logout   
from .forms import SignupForm, LoginForm  
  
  
# Create your views here.  
# Home page  
def index(request):  
 return render(request, 'index.html')  
  
# signup page  
def user\_signup(request):  
 if request.method == 'POST':  
 form = SignupForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('login')  
 else:  
 form = SignupForm()  
 return render(request, 'signup.html', {'form': form})  
  
# login page  
def user\_login(request):  
 if request.method == 'POST':  
 form = LoginForm(request.POST)  
 if form.is\_valid():  
 username = form.cleaned\_data['username']  
 password = form.cleaned\_data['password']  
 user = authenticate(request, username=username, password=password)  
 if user:  
 login(request, user)   
 return redirect('home')  
 else:  
 form = LoginForm()  
 return render(request, 'login.html', {'form': form})  
  
# logout page  
def user\_logout(request):  
 logout(request)  
 return redirect('login')

The code snippet you provided represents a Django views.py file that includes various views for **user authentication** and account **management**. Here’s a breakdown of each function:

1. index(request): This view represents the home page of your application. It renders the 'index.html' template and returns it as a response.
2. user\_signup(request): This view handles the signup page. It checks if the request method is POST, which indicates a form submission. If so, it validates the submitted form data using UserCreationForm. If the form is valid, it saves the user and redirects them to the login page. If the request method is GET, it creates a new instance of UserCreationForm and renders the 'signup.html' template, passing the form as context.
3. user\_login(request): This view handles the login page. It checks if the request method is POST, indicating a form submission. It validates the submitted form data using LoginForm. If the form is valid, it retrieves the username and password from the cleaned data. It then authenticates the user using authenticate() and logs them in using login(). If the user is successfully authenticated, it redirects them to the home page. If the request method is GET, it creates a new instance of LoginForm and renders the 'login.html' template, passing the form as context.
4. user\_logout(request): This view handles the logout functionality. It calls the logout() function provided by Django to log out the user and redirects them to the login page.

The code also includes import statements for various Django modules and forms (render, redirect, authenticate, login, logout, UserCreationForm, and LoginForm). These imports are necessary for the proper functioning of the views.

**Overall,** this code demonstrates a basic implementation of user signup, login, and logout functionality using Django, along with the associated forms. You can integrate these views into your Django project to enable user authentication and account management.

**6. Forms**

**Create**a new file forms.py inside myapp folder and write the below code:

from django import forms   
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth.models import User  
  
class SignupForm(UserCreationForm):  
 class Meta:  
 model = User   
 fields = ['username', 'password1', 'password2']  
  
class LoginForm(forms.Form):  
 username = forms.CharField()  
 password = forms.CharField(widget=forms.PasswordInput)

The code snippet you provided represents a Django forms module that defines two forms: SignupForm and LoginForm. These forms are used for user signup and login functionality in a Django application.

SignupForm:

* Inherits from UserCreationForm, which is a built-in Django form specifically designed for user registration.
* The UserCreationForm provides fields for username, password1, and password2 (password confirmation).
* The Meta class specifies the model to be used, which is the default Django User model imported from django.contrib.auth.models.
* The fields attribute lists the fields that should be included in the form, namely 'username', 'password1', and 'password2'.

LoginForm:

* A standard Django form used for user login.
* It does not inherit from any specific Django form class.
* It defines two fields, ‘username’ and ‘password’, using the forms.CharField() method.
* The ‘password’ field is rendered as a password input field due to the widget=forms.PasswordInput argument.

These forms can be used within Django views to handle user registration and authentication processes. They provide a convenient way to generate HTML forms with appropriate fields and validation.

**7. Templates**

**Create** a new folder templates in myproject and create a new file index.html and write the below code:

{% if request.user.is\_authenticated %}  
 <p>{{ request.user.username }}</p>  
 <a href="{% url 'logout' %}">Logout</a>  
{% else %}  
 <a href="{% url 'login' %}">Login</a>  
 <a href="{% url 'signup' %}">Signup</a>  
{% endif %}  
  
<h1>Welcome!</h1>

**Create**a new login.html file and write the below code:

<h1>Login</h1>  
<form method="POST">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Login</button>  
 <a href="{% url 'signup' %}">Dont have Account Create</a>  
</form>

**Create** a new signup.html file and write the below code:

<h1>Signup</h1>  
<form method="POST">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Signup</button>  
 <a href="{% url 'login' %}">Already have account?</a>  
</form>

**8. Testing and Running**

**Now**that we have set up the basic structure of our application using Django Framework, it’s time to test and run the app. Follow these steps:

**Step 1: Open your Command-Line Interface:**

**Open**your command-line interface and navigate to the root directory of your Django project.

**To proceed,** please open the terminal within the myproject folder and execute the following command:

python manage.py makemigrations  
python manage.py migrate

**Step 2: Start the Server:**

**To start the server,** run the following command in your command-line interface:

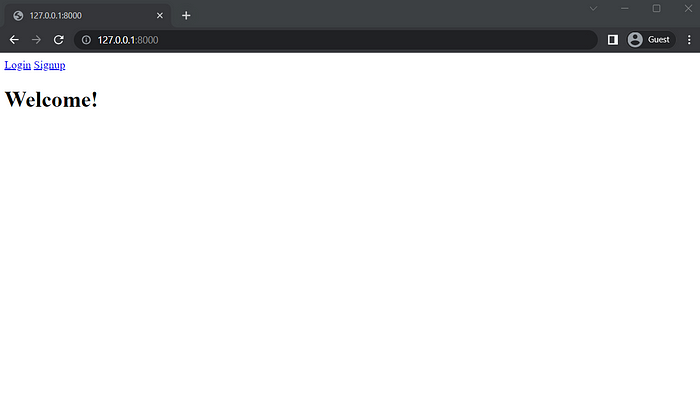
python manage.py runserver

This command will launch the Django development server.

**Step 3: Testing**

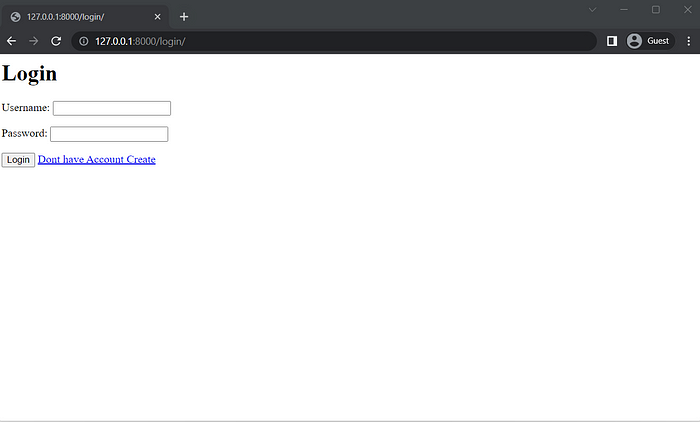
**After running** the server and accessing the project interface at [http://127.0.0.1:8000/](http://127.0.0.1:8000/api/hello) .

**Open** any browser and hit : [http://127.0.0.1:8000/](http://127.0.0.1:8000/api/hello) URL and you can the home page of our website looks like this:



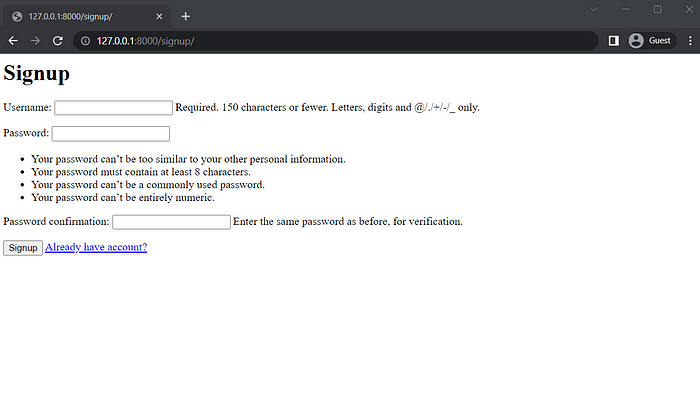
Home Page

**Login Page:**



login page

**Signup Page:**



**Conclusion**

**In this blog post,** We explored how to implement user login and signup functionality using Django, and HTML forms. We covered the necessary steps, from setting up a Django project to designing the templates and implementing the views and forms. By following the provided resources and references, you can further customize and enhance the user authentication process in your Django applications.

Remember, user authentication is a critical aspect of any web application, so it’s essential to prioritize security and follow best practices to protect user data. With Django and Bootstrap, you can create a seamless user experience while maintaining robust security measures.

**Happy coding!**