

# Login system in Python Django

March 14, 2022 by Bijay Kumar



In this [Python Django Tutorial](#), we will learn to **Create a Login system in Python Django**. And we'll also examples related to this. These are the following topics that we are going to discuss in this tutorial.

- Login System in Python Django
- How to create Django built-in login system
- How to create Django customize login system
- How to create Django multiple model login system

Table of Contents



## Login System in Python Django

In this section, we'll learn what does login system means in python Django.

users from accessing resources on a website. There are three parts to the login system:

1. **Signup:** Signing up is the process of creating a new account. If you want to use a portal or application for the first time, you must first sign up.
2. **Login:** A login is a collection of credentials that is used to verify a user's identity. It denotes that the user has been identified and authenticated in order to gain access to the website. The most common combination is a username and password.
3. **Logout:** To log out of a website, you must first log in. When you log out, you're telling the website that you'd like to discontinue your login session.

Read: [Python Django set timezone](#)

## How to create django built-in login system

In this section, we'll learn how to use Django's built-in feature to develop a login page of the login system. I'll guide you through it step by step using an example.

### 1. Create Project

The command to create a new project is as follow.

---

```
django-admin startproject project_name
```

**Example:** Here I create a new project name as LoginSystem.

**Command:**

```
django-admin startproject LoginSystem
```

```
PS E:\LoginSystem> django-admin startproject LoginSystem
PS E:\LoginSystem> █
```

Start Project

## 2. Check Necessary Settings

Before creating a login system, make sure you have

**django.contrib.auth** in your **INSTALLED APPS** and that your authentication middleware is configured properly in the **MIDDLEWARE** settings.

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

An application that is already included

### 3. Make Migration

Migration is a means of applying changes that we have made to a model, into the database schema. To create all tables according to the schema defined in the migration file, we run the following command.

```
python manage.py migrate
```

### 4. Define Project URLs

A web address is referred to as a URL. Every time you visit a website, you can see the URL in your browser's address bar. Every

application will know what to show a user and view that user.

Open your's project's urls.py file and specify the particular URLs.

```
from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', auth_views.LoginView.as_view(), name='login'),
    path('login/', include('home.urls')),
]
```

Import the django.contrib.auth.views module and add the URLconf for Login Page and Home Page to your urls.py file.

Read [Python list append Django](#)

## 5. Login Template

Basically, Django Templates are used to generate dynamic HTML web pages that are visible to the end-user. A template in Django is written in HTML, CSS, and Javascript in a .html file.

Create a registration folder in your templates folder and a

login.html page inside it.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>LoginSystem </title>
</head>
<body>
  <form method="post">
    {% csrf_token %}
    {{form.as_p}}
    <input type="submit" value="Login" />

  </form>

</body>
</html>
```

## 6. Template Configuration

To configure the template system, we must specify several items in the settings.py file. The name of our template directory is templates. Django Templates searches each of the INSTALLED APPS subdirectories by default for a templates subdirectory.

```
3     'BACKEND': 'django.template.backends.django.DjangoTemplates',
3     'DIRS': [os.path.join(BASE_DIR, 'templates')],
1     'APP_DIRS': True,
2     'OPTIONS': {
3         'context_processors': [
4             'django.template.context_processors.debug',
5             'django.template.context_processors.request',
5             'django.contrib.auth.context_processors.auth',
7             'django.contrib.messages.context_processors.messages',
3         ],
3     },
3 },
1 ]
```

Template Configuration

## 7. Redirect to Home Page

We need to redirect to the next page, which is the home page, after a successful login. So, let's learn to create the app first

### Command to create an app:

```
python manage.py startapp app_name
```

Here I create an app with the name Home.

```
PS E:\LoginSystem\LoginSystem> python manage.py startapp home
```

Create App

Now, we to include the app in your project, add your app name to the INSTALLED APPS list in settings.py of the project.

```
]

```

Next, define the URL in the app's `urls.py` file.

```
from django.urls import path, include
from home import views

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

After this, define the view of the app.

```
from django.shortcuts import render, HttpResponse

# Create your views here.

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

Next, create a `home.html` file inside the template folder of the Home app.

```
<!doctype html>
<html lang="en">
```



```

<meta charset= utf - 8 >
<meta name="viewport" content="width=device-width, initial-
scale=1">

<!-- Bootstrap CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/boots
trap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgplJLm9NaoOYz1ztcQTWFspD3yD65Vohhp
uuCOMLASjC" crossorigin="anonymous">

<title>PythonGuides!</title>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">PythonGuides</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 class="nav-link active" aria-current="page"
href="#">Home</a>
        </li>
      </ul>

```

```

placeholder= Search aria-label= Search >
    <button class="btn btn-outline-success"
type="submit">Search</button>
    </form>
</div>
</div>
</nav>
<div class="container my-3">
    <h1 style="text-align:center;">Welcome to Python Guides</h1>
</div>

<div class="card">
    <div class="card-body">
        <h3 style="text-align:center;">Thanks for landing on this page
to know more about PythonGuides.com.</h3>
        <br>
        <h6>I welcome you to the website and hopefully, you got to
learn something in Python. I started this website to share my finding
and learnings in Python with you.</h6>
        <h6>To keep things simple, I am trying to write a lot of articles
on Python. Feel free to give your valuable comments and also share
the articles if you are liking and hoping it will be helpful to someone.
</h6>
        <br>
        <h2 style="text-align:center;">Also, Subscribe to Our YouTube
Channel for FREE Python Video Tutorials.</h2>
    </div>
<!-- Optional JavaScript; choose one of the two! -->

<!-- Option 1: Bootstrap Bundle with Popper -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstra

```

```
n/tWtlaxVXM crossorigin = anonymous ></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->
<!--
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWIC2Al4u+LWgxfKTRlcfuOJTzR+EQDz/bglDoEyl4H0zUFOQKbrJOEcQF" crossorigin="anonymous"></script>
-->
</body>
</html>
```

We now require a redirect to the next page. The following code can be added to settings.py to set the next page:

```
LOGIN_REDIRECT_URL = next_page
```

In my example, I want to redirect to the home page after successful login.

```
LOGIN_REDIRECT_URL = 'home/'
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.  
BASE_DIR = Path(__file__).resolve().parent.parent  
LOGIN_REDIRECT_URL = 'home/'
```

#### Login Settings

If no login redirect URL is specified, the default URL of `/accounts/profile/` will be used, resulting in a **TemplateDoesNotExist** problem.

## 8. Create Super User

Till now, we have not created any users for this. So, we are going to create a user. So see, how we can do that.

To create a superuser run the following command:

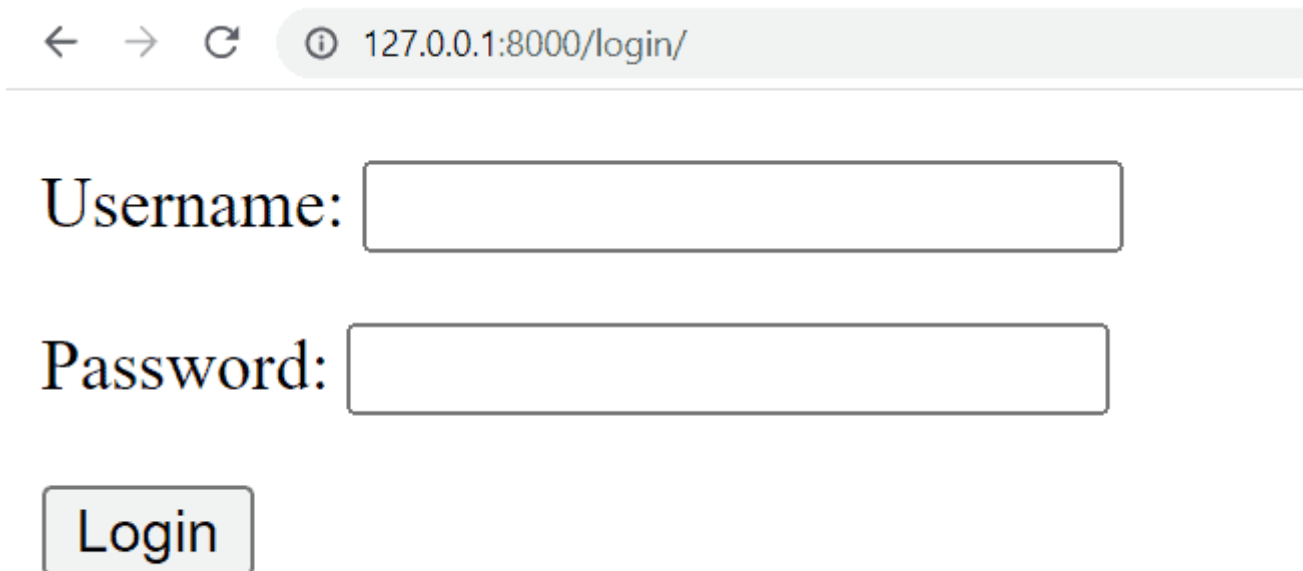
```
python manage.py createsuperuser
```

- Then enter the username of your choice, if you want. Otherwise, it takes the default system name.
- Then enter the email address and press enter. You can leave it blank also.
- Then, in front of the Password field, type the password and hit enter. To keep it safe, enter a strong password.
- Then again enter the same Password for confirmation.

In this section, we'll learn how to run the server in Django. To start the server, use the following command in the terminal.

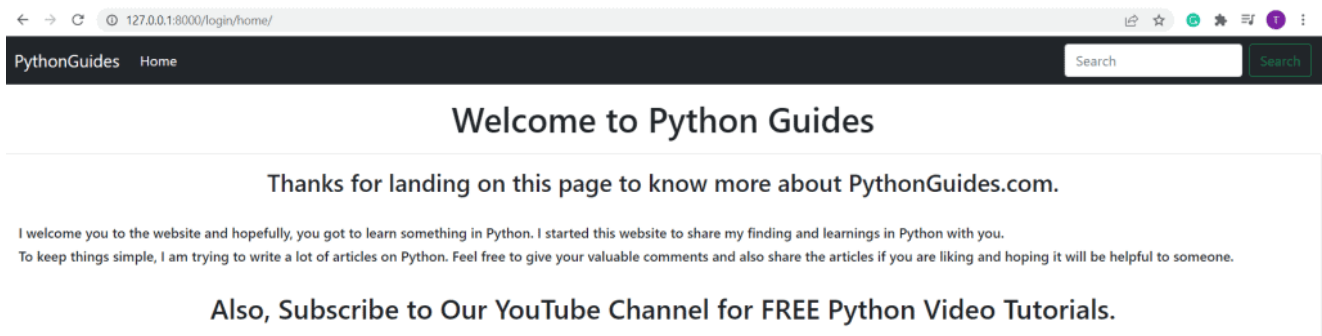
```
python manage.py runserver
```

The output is as below:



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:8000/login/`. The page contains a login form with two input fields: "Username:" and "Password:". Below these fields is a button labeled "Login".

Now, enter the username and password and click on the login button. After successful login, you will redirect to the home page.



A screenshot of the PythonGuides.com home page. The browser address bar shows `127.0.0.1:8000/login/home/`. The page has a dark header with "PythonGuides" and "Home" links, and a search bar. The main content area has a heading "Welcome to Python Guides", a paragraph "Thanks for landing on this page to know more about PythonGuides.com.", a smaller paragraph about the website's purpose, and a call to action: "Also, Subscribe to Our YouTube Channel for FREE Python Video Tutorials."

# How to create Django customize login system

In this section, we'll learn to create a signup page, login page, and logout page without using Django's in-built feature. I'll guide you through it step by step using an example.

## 1. Basic Steps

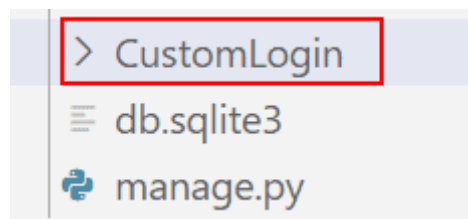
Before we can create signup, login, or logout page, we must first go through the following procedures.

**Create Project:** First and foremost, we must create a Django project.

The command to create a Django project is:

```
django-admin startproject project_name
```

So, I build a project called **CustomLogin**.

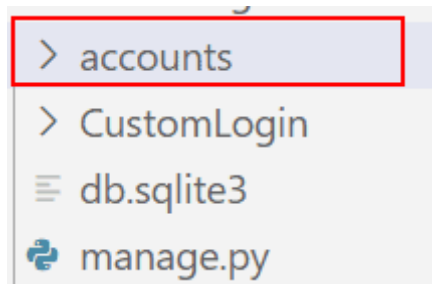


Create Project

create a Django app. The command to create a Django app is:

```
python manage.py startapp app_name
```

So, I build an app called **accounts**.



Create App

**Install App:** After we've created the Django app, we'll need to install a Django app. To do so, go to your project directory and open the settings.py file. Now, in INSTALLED\_APPS, type the name of your application.

```
# Application definition
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'accounts',  
]
```

So, under your app directory, make a templates folder. Go to the settings.py file and define the DIRS path in templates.

```
import os

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

Define Templates

## 2. Define URLs File

According to Django, all resources should be mapped using urls.py files.

**Project URLs File**: By default, Django includes a urls.py file in the project. This file has a pre-defined path to the admin app. And we define the path of the newly created urls.py file under the app.



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

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('accounts/', include('accounts.urls'))  
]
```

**App URLs File:** Firstly, we have to create a file with the name `urls.py` in the app directory. Now, we define the path of the different views created under the `views.py` file of the app.

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

```
urlpatterns = [  
    path("register", views.register, name="register"),  
    path("login_user", views.login_user, name="login_user"),  
    path("logout_user", views.logout_user, name="logout_user"),  
    path("home", views.home, name="home")  
]
```

### 3. SIGNUP

Now, we'll learn to create register views. For this, we have to create a function `register` in the **`views.py`** file of the app directory.

```
def register(request):
    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']

        if password == confirm_password:
            if User.objects.filter(username=username).exists():
                messages.info(request, 'Username is already taken')
                return redirect(register)
            elif User.objects.filter(email=email).exists():
                messages.info(request, 'Email is already taken')
                return redirect(register)
            else:
                user = User.objects.create_user(username=username,
password=password,
                                                email=email, first_name=first_name,
last_name=last_name)
                user.save()

                return redirect('login_user')

        else:
            messages.info(request, 'Both passwords are not matching')
            return redirect(register)
```

- We begin by defining the **register function**, which makes use of the request object. Django makes an HttpRequest when a page is requested. In this case, we'll utilize the **HTTP method Post** to submit data to a server in order to create or update a resource.
- To the server, we send **first\_name**, **last\_name**, **username**, **email**, **password**, and **confirm\_password**.
- Then we use various **object filters** to check if the password is the same as the confirm password if the email address already exists, and print the message accordingly, as well as redirect the page based on the filter response.

Now, we'll see the **registration.html**:

The resgistration.html file specifies the appearance of the signup web page.

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
```

```

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootst
trap.min.css"
        integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFA
W/dAiS6JXm" crossorigin="anonymous">

    <title>SIGNUP</title>
</head>

<body>
    <form action="/accounts/register" method="POST"
class="form-horizontal">
        {% csrf_token%}
        <div class="mx-auto" style="width: 400px;">

            <div class="col-xs-8 col-xs-offset-4">
                <h2 style="text-align:center;">SIGN UP</h2>
            </div>
            <hr />
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control"
name="username" id="username" placeholder="Enter Username"
                Required>
            </div>
            <div class="form-group">
                <label for="fname">First Name</label>
                <input type="text" class="form-control"
name="first_name" id="first_name" placeholder="Enter First Name"

```

```

<div class= form-group >
    <label for = "lname">Last Name</label>
    <input type="text" class="form-control"
name="last_name" id="last_name" placeholder="Enter Last Name"
    Required>
</div>
<div class="form-group">
    <label for="email">Email address</label>
    <input type="email" class="form-control"
name="email" id="email" placeholder="Enter email" Required>
</div>
<div class="form-group">
    <label for="password">Password</label>
    <input type="password" class="form-control"
name="password" id="password" placeholder="Enter Password"
    Required>
</div>
<div class="form-group">
    <label for="confirm_password">Confirm
Password</label>
    <input type="password" class="form-control"
name="confirm_password" id="confirm_password"
    placeholder="Confirm Your Password"
    Required>
</div>
<button type="submit" class="btn btn-
primary">SIGNUP</button>
    <a class="btn btn-dark" href="login_user"
role="button">LOGIN</a>
</div>
</form>
<div class="form-group">

```

```

        {{message}}
    </div>
{% endfor %}
</div>

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/p
opper.min.js"
        integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskv
Xusvfa0b4Q"
        crossorigin="anonymous"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstra
p.min.js"
        integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+7
6PVCmYl"
        crossorigin="anonymous"></script>
</body>

</html>

```

---

`{% csrf_token %}`. By using this tag, we can avoid CSRF attacks and assure the security of post requests from a user to the server.

- We also, use the Django template tag `{% for %}` to display the message notification.

Now, run the server and view the signup web page.

Username

First Name

Last Name

Email address

Password

Confirm Password

Signup Page

## 5. LOGIN

Now, we'll learn to create `login_user` views. For this, we have to create a function **`login_user`** in the **`views.py`** file of the app



```
from django.contrib import messages
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth

def login_user(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username,
password=password)

        if user is not None:
            auth.login(request, user)
            return redirect('home')
        else:
            messages.info(request, 'Invalid Username or Password')
            return redirect('login_user')

    else:
        return render(request, 'login.html')
```

- To begin, we'll create the login\_user function, which uses the POST request object.
- The username and password are then sent to the server.

username and password.

- If the user is authenticated, go to the home page; if not, go to the login page and print the message Invalid Username or Password.

Now, we'll see the **login.html**:

The login.html file specifies the appearance of the login web page.

```
<!doctype html>
<html lang="en">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/boots
trap.min.css"
  integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFA
W/dAiS6JXm" crossorigin="anonymous">

  <title>LOGIN</title>
</head>
```

```

<form action = login_user method = post class = form -
horizontal">
    {% csrf_token %}
    <div class = "col-xs-8 col-xs-offset-4">
        <h2 style = "text-align:center;">LOGIN</h2>
    </div>

    <div class = "form-group">
        <label for = "username">Username</label>
        <input type = "text" class = "form-control"
name = "username" id = "username" placeholder = "Enter Username">
    </div>
    <div class = "form-group">
        <label for = "password">Password</label>
        <input type = "password" class = "form-control"
name = "password" id = "password" placeholder = "Enter Password">
    </div>
    <button type = "submit" class = "btn btn -
primary">LOGIN</button>
    <br />
    <div class = "form-group">
        {% for message in messages %}
        <div class = "alert alert-danger" role = "alert">
            {{ message }}
        </div>
        {% endfor %}
    </div>
</form>

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src = "https://code.jquery.com/jquery-3.2.1.slim.min.js"

```

---

F93hXpG5KkN

```

        crossorigin = "anonymous" > </script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/p
opper.min.js"
        integrity = "sha384-
ApNbgh9B+ Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskv
Xusvfa0b4Q"
        crossorigin = "anonymous" > </script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstra
p.min.js"
        integrity = "sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+7
6PVCmYl"
        crossorigin = "anonymous" > </script>
</body>

</html>

```

- To display the message notification, we use the Django template tag **{%for%}**.

We redirect you to the login page after successful signup, or you can move to the login page directly if you are already a user.

Username

Password

Login Page

## 5. HOME

Now, we'll learn to create home views. For this, we have to create a function **home** in the **views.py** file of the app directory.

```
from django.contrib import messages
from django.shortcuts import render, redirect
```

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

- To begin, we'll create the home function, with request object.
- The home function redirect to home.html.

Now, we'll see the **home.html**:

The home.html file specifies the appearance of the home web page.

```

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/boots
trap.min.css"
  integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFA
W/dAiS6JXm" crossorigin="anonymous">
  <title>Login System</title>
</head>

<body>

  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
  integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
  crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/p
opper.min.js"
  integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskv
Xusvfa0b4Q"
  crossorigin="anonymous"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstra
p.min.js"

```

## 6PVCmYl

```

crossorigin="anonymous"></script>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="home">Login System</a>
  <button class="navbar-toggler" type="button" data-
toggle="collapse" data-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 active">
        <a class="nav-link" href="home">HOME<span class="sr-
only">(current)</span></a>
      </li>
      {% if user.is_authenticated %}
      <li class="nav-item">
        <a class="nav-link" href="#">Hi, {{user.first_name}}</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="logout_user">LOGOUT</a>
      </li>
      {% else %}
      <li class="nav-item">
        <a class="nav-link" href="register">SIGN UP</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="login_user">LOGIN</a>
      </li>
      {% endif %}

```

```
</nav>

<div class="container my-3">
  <h1 style="text-align:center;">Welcome to Python Guides</h1>
</div>

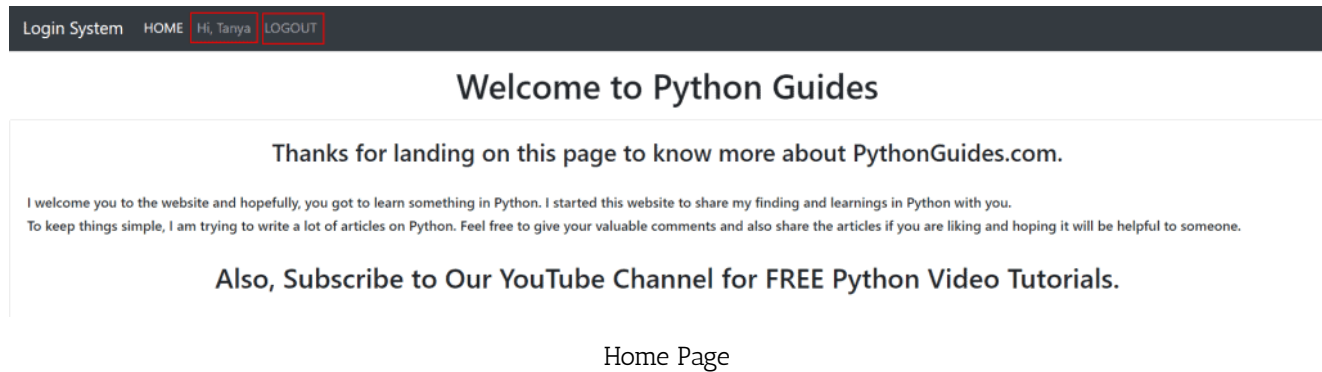
<div class="card">
  <div class="card-body">
    <h3 style="text-align:center;">Thanks for landing on this page to
know more about PythonGuides.com.</h3>
    <br>
    <h6>I welcome you to the website and hopefully, you got to
learn something in Python. I started this website to
    share my finding and learnings in Python with you.</h6>
    <h6>To keep things simple, I am trying to write a lot of articles
on Python. Feel free to give your valuable
    comments and also share the articles if you are liking and
hoping it will be helpful to someone.</h6>
    <br>
    <h2 style="text-align:center;">Also, Subscribe to Our YouTube
Channel for FREE Python Video Tutorials.</h2>
  </div>
</body>

</html>
```

- Here we, use `{%if%}` template tag with `is_authenticated` attribute, to verify whether the user is authorised or not.
- We print the user's first name on the nav bar if the user is authenticated. We use the tag `{{user.first_name}}` for this.



Let's have a look at home page, when user successfully login.



## 6. LOGOUT

Now, we'll learn to create `logout_user` views. For this, we have to create a function **`logout_user`** in the **`views.py`** file of the app directory.

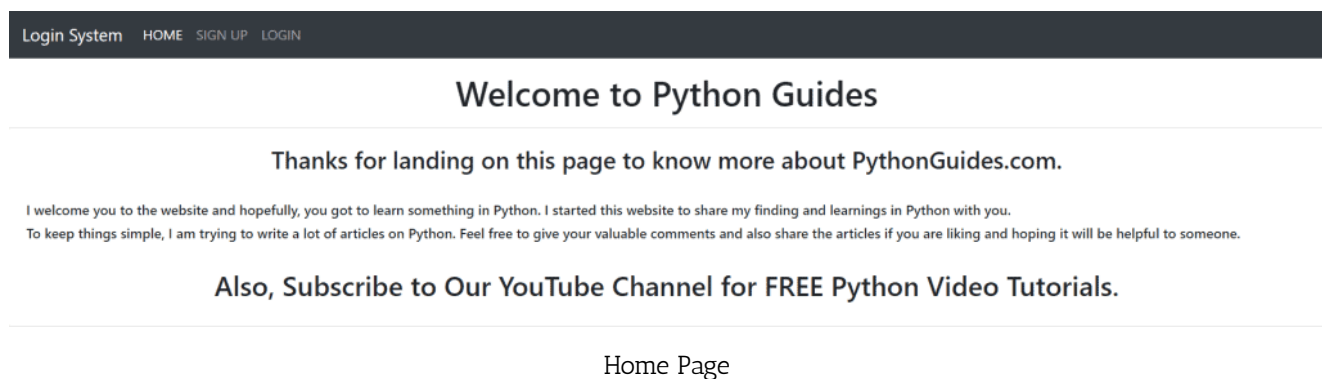
```
from django.contrib import messages
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
```

```
def logout_user(request):
    auth.logout(request)
    return redirect('home')
```

- To begin, we'll create the `logout_user` function, with request object.
- To log out the user we use the `logout()` function of `django.contrib.auth`.

When we click on the logout button, the `logout()` function is called. And, it completely cleaned out the session data for the current request.

Let's see how the home page looks when we click on the logout button.



## How to create Django multiple model login system

In this section, we'll learn how to use Django to develop a login system with multiple models. This login system consists of three modules: registration, login, and logout.

### Basic Steps

Before we begin learning how to develop the three modules of the login system described above, we must first complete some basic and necessary procedures.

We must first and foremost build a Django project. And, to make a Django project, use the following command:

```
django-admin startproject MultiLogin
```

- So, here I build a project named **MultiLogin**

### CREATE APP:

We'll need to develop a Django app after we've finished with the Django project. To make a Django app, use the following command:

```
python manage.py startapp accounts
```

- So, here I build an app named **accounts**.

### INSTALL APP:

We'll need to install a Django app after we've created it. To do so, open the **settings.py** file in your project directory. Now type the name of your application in **INSTALLED APPS**.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'accounts',  
]
```

INSTALL APP

## INSTALL / ACTIVATE TEMPLATES:

To store your HTML files, you'll need a template folder. Make a templates folder in your app directory. Define the **DIRS** path in templates in the **settings.py** file.

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    ],  
]
```

TEMPLATE

## Define URLs File

all the resources should be mapped using these files according to Django.

### PROJECT URLs FILE:

Django includes a **urls.py** file in the project (**MultiLogin**) by default. Add the following code in this file.

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('accounts.urls'))
]
```

- In this file, the admin app's path is pre-defined.
- Now, we have to specify the location of the urls.py file within the app (**accounts**).

### APP URLs FILE:

By default, an app doesn't consist any urls.py file. So firstly, we have to create a file with the name **urls.py** in the app directory (**accounts**).

```
urlpatterns = [  
    path('register /', views.register, name='register'),  
    path('login /', views.login_user, name='login_user'),  
    path('logout /', views.logout_user, name='logout_user'),  
    path('home /', views.home, name='home'),  
    path('student_register /', views.student_register.as_view(),  
name='student_register'),  
    path('teacher_register /', views.teacher_register.as_view(),  
name='teacher_register')  
]
```

- Here, we define the path of the different views created under the **views.py** file of the app.

## Create Model

Basically, a table in your database is a Django model. And, by default, we got an empty SQLite database located in the root folder of your project (**MUTILogin**) when we built the project in Django. So, to make a new table, we'll need to make a new model.

Open the **models.py** file in the accounts folder. And by default, it's empty, so add the following code :

```
class User(AbstractUser):
    is_student = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)
    first_name = models.CharField(max_length=80)
    last_name = models.CharField(max_length=80)

class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)
    phone_number = models.CharField(max_length=10)
    class_name = models.CharField(max_length=100)

class Teacher(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)
    phone_number = models.CharField(max_length=10)
    department = models.CharField(max_length=30)
```

So, here we create three classes named as User, Student, and Teacher. So, let's discuss each class in detail:

- User Class:
  - Here, we create a custom user model and extend its functionality by using AbstractUser.
  - So, AbstractUser is a model having complete fields, similar as an abstract class so that you can easily inherit

### Student Class:

- Here, we create Student model class. And in this class we define one to one relationship between the fields of User class which we created above.
- Addition to it, we create a text field "**phone\_number**" and it will contain the phone number of the student having max length 10.
- Then, we create one more text field "**class\_name**" and it will contain the class of the student.
- Teacher Class:
  - Here, we create Teacher model class. And in this class we again define one to one relationship between the fields of User class which we created above.
  - Addition to it, we create a text field "**phone\_number**" and it will contain the phone number of the teacher having max length 10.
  - Then, we create one more text field "**departemnt**" and it will contain the department in which teacher is working.

## Register Model

Open the **admin.py** file in the accounts folder. And by default, it's empty, so add the following code :



```
from django.contrib.auth.models import User
```

```
# Register your models here.
```

```
admin.site.register(Student)
```

```
admin.site.register(Teacher)
```

- So, here we register above created two models i.e Student Model and Teacher Model with the admin interface.

## Create Form

A form is a page that contains multiple fields, or spaces to enter data by user. Create and Open the **forms.py** file in the accounts folder. By default, it's empty, so add the following code :

```
from django.contrib.auth.forms import UserCreationForm
from django.db import transaction
from django import forms
from .models import Student, Teacher, User
```

```
class StudentSignUpForm(UserCreationForm):
    first_name = forms.CharField(required=True)
    last_name = forms.CharField(required=True)
    phone_number = forms.CharField(required=True)
    class_name = forms.CharField(required=True)
```

```
class Meta(UserCreationForm.Meta):
    model = User
```

```
user = super().save(commit=False)
user.first_name = self.cleaned_data.get('first_name')
user.last_name = self.cleaned_data.get('last_name')
user.is_student = True
user.save()
student = Student.objects.create(user=user)
student.class_name = self.cleaned_data.get('class_name')
student.phone_number = self.cleaned_data.get('phone_number')
student.save()
return user
```

```
class TeacherSignUpForm(UserCreationForm):
    first_name = forms.CharField(required=True)
    last_name = forms.CharField(required=True)
    department = forms.CharField(required=True)
```

```
class Meta(UserCreationForm.Meta):
    model = User
```

```
@transaction.atomic
```

```
def data_save(self):
    user = super().save(commit=False)
    user.first_name = self.cleaned_data.get('first_name')
    user.last_name = self.cleaned_data.get('last_name')
    user.is_teacher = True
    user.save()
    teacher = teacher.objects.create(user=user)
    teacher.phone_number = self.cleaned_data.get('phone_number')
    teacher.department = self.cleaned_data.get('department')
    teacher.save()
    return user
```

---

and TeacherSignUpForm so, let's discuss each class in detail.

---

- StudentSignUpForm:
  - Django has an integrated user authentication system. So, here we are importing user authentication modules that allows us to create the student sign up.
  - And here, we use **UserCreationForm** that is used to create a new user for our application. By default, there are three fields in this form: **username**, **password**, and **confirm password**.
  - And addition to this, we create four text fields **first\_name**, **last\_name**, **phone\_number**, and **class\_name** respectively.
  - The we define Meta class as we need to add data about the model user.
  - Then we use the atomic transaction for student signup form, which is a sequence of one or more SQL operations that are treated as a unit. And the aim of the atomic transaction are to provide four properties commonly known as ACID.
- TeacherSignUpForm:
  - Here we are importing user authentication modules that allows us to create the teacher sign up.

new user for our application. By default, there are three fields in this form: **username**, **password**, and **confirm password**.

- Then, addition to this, we create three text fields **first\_name**, **last\_name**, and **department** respectively.
- Then we define Meta class as we need to add data about the user model.
- Then we use the atomic transaction for teacher signup form, as the aim of the atomic transaction are to provide four properties commonly known as ACID.

## Define Views File

Basically, Django views, like HTML documents, are Python functions that take http requests and return http responses. So, we can say that Django-based websites have a lot of views with various tasks and goals.

And, views are often stored in a file called **views.py** in the app's folder. By default, it's empty, so add the following code :

```
from email import message
from django.shortcuts import render, redirect
from django.contrib.auth import login, logout, authenticate
from django.contrib.auth.forms import AuthenticationForm
```

---

```
from .models import User
from .forms import StudentSignUpForm, TeacherSignUpForm

# Create your views here.

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

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

class student_register(CreateView):
    model = User
    form_class = StudentSignUpForm
    template_name = 'student_register.html'

    def form_valid(self, form):
        user = form.save()
        login(self.request, user)
        return redirect('/accounts/home')

class teacher_register(CreateView):
    model = User
    form_class = TeacherSignUpForm
    template_name = 'teacher_register.html'

    def form_valid(self, form):
        user = form.save()
        login(self.request, user)
        return redirect('/accounts/home')

def login_user(request):
```

```
if form.is_valid():
    username = form.cleaned_data.get('username')
    password = form.cleaned_data.get('password')
    user = authenticate(username=username,
password=password)
    if user is not None :
        login(request,user)
        return redirect('/accounts/home')
    else:
        messages.error(request,"Invalid username or password")
else:
    messages.error(request,"Invalid username or password")
return render(request, 'login.html',context =
{'form':AuthenticationForm()})

def logout_user(request):
    logout(request)
    return redirect('/accounts/home')
```

So, here we create six views named as **home**, **register**, **student\_register**, **teacher\_register**, **login\_user**, and **logout\_user**. So, let's discuss each views in detail:

- home:
  - In home view, we simple render to '**home.html**' template.
- register:
  - In register view, we simple render to '**register.html**' template.

---

Then, we create **teacher\_register** class based on generic view.

- Next, we use the **CreateView** as **StudentSignUpForm** is required on the page and a database insertion is required upon submission of a valid form.
- **teacher\_register**:
  - Then, we create **teacher\_register** class based on generic view.
  - Next, we use the **CreateView** as **TeacherSignUpForm** is required on the page and a database insertion is required upon submission of a valid form.
- **login\_user**:
  - We'll create the **login\_user** view, which uses the POST request object.
  - The username and password are then sent to the server.
  - The authenticate function is then used to verify the username and password.
  - If the user is authenticated, go to the home page; if not, go to the login page and print the message Invalid Username or Password.
- **logout\_user**:

- Next, the **logout()** function of `django.contrib.auth` is used to log out the user.
- On the successful logout user will returns you to your **'home page'**.

## Create Templates

We learned from the Django that the outcome should be in HTML and the HTML files are created in a template.

So create a templates folder inside the root folder (**MULTILOGIN**), and create five HTML files named as **'home.html'**, **'login.html'**, **'register.html'**, **'student\_register.html'**, and **'teacher\_regisiter.html'**.

So, let's discuss and create each HTML file:

- **home.html**:
  - The **home.html** file specifies the appearance of the home web page. And the following is the code of **home.html** file.
  - Here we, use **{%if%}** template tag with **is\_authenticated** attribute, to verify whether the user is authorised or not.



- We also add a logout button.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
</head>

<body>
  <h1>Welcome to PythonGuides</h1>

  {% if user.is_authenticated%}
  <h3>Hi {{user.get_username}}</h3>
  <a href="{% url 'logout_user' %}">Logout</a> <br>
  {% else %}
  <a href="{% url 'register' %}">Register</a> <br>
  <a href="{% url 'login_user' %}">Login</a>
  {% endif %}
</body>

</html>
```

- login.html

- Here, we add the link of **student\_register** and **teacher\_register**.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Login</title>
</head>

<body>

    <h2>Login</h2>

    <form action="{% url 'login_user' %}" method="POST" novalidate>
        {% csrf_token %}
        {{form.as_p}}

        <br>
        <input type="submit" value="Login" class="btn btn-block
btn - primary">
    </div>
</form>

</body>
```

- register.html:
  - The **register.html** file specifies the appearance of the register web page. And the following is the code of **register.html** file.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>User Registration</title>
</head>
<body>

  <section>
    <div class="container">
      <div class="row">
        <div class="col-md-6 mx-auto">
          <div class="card">
            <div class="card-header text-black">
              <h2>User Registration</h2>
            </div>
            <div class="card-body">
              <div class="row">
                <div class="col-lg-6">
                  <a type="button" class="btn btn-block" href="{% url
```

- student\_register.html:
  - The **student\_register.html** file specifies the appearance of the student register web page. And the following is the code of **student\_register.html** file.

52/62

```
<body>

    <h2> Student Registration Form</h2>
    <form action="{% url 'student_register' %}" method="POST"
novalidate>
        {% csrf_token %}
        {{form.as_p}}
        <input type="submit" value="Register" class="btn btn-block
btn - primary">
    </body>

</html>
```

- teacher\_register.html:
  - The **teacher\_register.html** file specifies the appearance of the teacher register web page. And the following is the code of **teacher\_register.html** file.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Teacher Registration</title>
</head>
```

```
<h2>Teacher Registration Form</h2>
<form action="{% url 'teacher_register' %}" method="POST"
novalidate>
    {% csrf_token %}
    {{form.as_p}}
    <input type="submit" value="Register" class="btn btn-block btn-
primary">
</body>

</html>
```

## Steps to execute Django Application

### **MAKE MIGRATIONS:**

If the models have been modified, this command prepares a makemigrations file for our new model. Run the following command in the terminal.

```
python manage.py makemigrations
```

### **MIGRATE:**

The migrate command executes the instructions given in the database's recent migrations file. Run the following command in the terminal.

## RUN SERVER:

To run the development server run the following command in the terminal.

```
python manage.py runserver
```

## Output



← → ↻ ⓘ 127.0.0.1:8000/accounts/home/

# Welcome to PythonGuides

[Register](#)

[Login](#)

Home Page Multiple Model Login System

On home page we have two options Register and Login. If you are new user click on 'Register'. And if you are already register click on 'Login'.

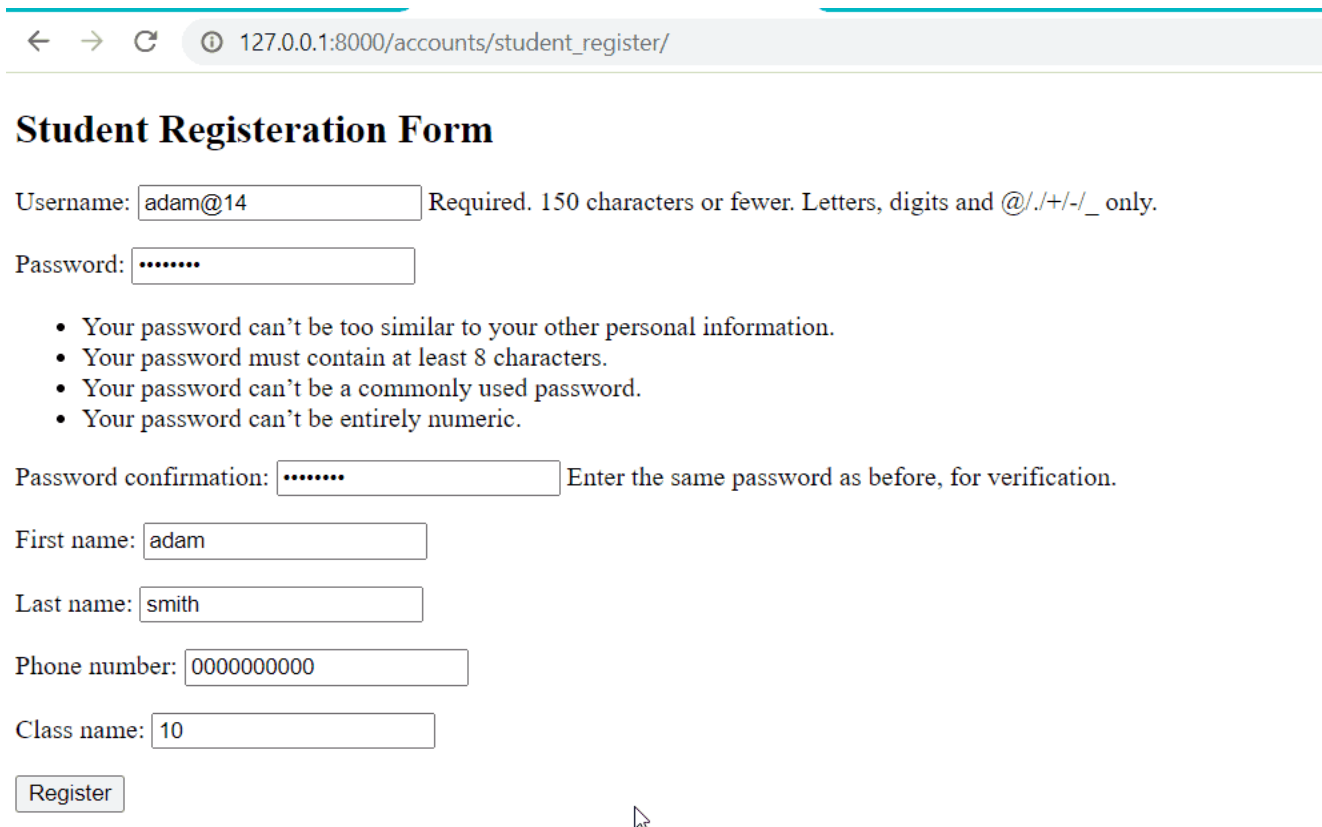
# User Registration

[I am a student](#)

[I am an teacher](#)

Register Page

When we click on Register, we get two options 'I am a student' and 'I am a teacher'. If you are a student click on student ones otherwise click on teacher ones.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/accounts/student\_register/". The page title is "Student Registration Form". The form contains the following fields and elements:

- Username:**  Required. 150 characters or fewer. Letters, digits and @/./+/\_ only.
- Password:**
- Password requirements:**
  - Your password can't be too similar to your other personal information.
  - Your password must contain at least 8 characters.
  - Your password can't be a commonly used password.
  - Your password can't be entirely numeric.
- Password confirmation:**  Enter the same password as before, for verification.
- First name:**
- Last name:**
- Phone number:**
- Class name:**
- Register** button

Student Registration

If you click on 'I am a Student' you will get page like this.



# Welcome to PythonGuides

Hi adam@14

[Logout](#)

Student Login

When you click on Register, page will look like that.



# Welcome to PythonGuides

[Register](#)

[Login](#)

Multi model Login System

When we click on Logout, we move to home page.

## Login

Username:

Password:

Login Page

When we click on Login, we will move to login page. And when you click on Login you will move again to Home Page with your username and Hi.



## User Registration

[I am a student](#)

[I am an teacher](#)

Register Page

If you click on I am a teacher.



Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:  Enter the same password as before, for verification.

First name:

Last name:

Department:

Teacher Register

You will move to the teacher\_register page where teacher get register.



# Welcome to PythonGuides

Hi eve@16

[Logout](#)

Home Page of Teacher

You will move to the Home page where you will get teacher username and Hi message. And If you click Logout button, you will

You may also like to read the following Django tutorials.

- [Python Change Django Version](#)
- [Python filter not in Django](#)
- [Python Django vs Pyramid](#)
- [Python Django length filter](#)
- [Get URL parameters in Django](#)

In this Python Django Tutorial, we discussed the Login System in Python Django. Also, we discuss the following list of topics.

- Login System in Python Django
- How to create Django built-in login system
- How to create Django customize login system
- How to create Django multiple model login system



## Bijay Kumar

Python is one of the most popular languages in the United States of America. I have been working with Python for a long time and I have expertise in working with various libraries on Tkinter, Pandas, NumPy, Turtle, Django, Matplotlib, Tensorflow, Scipy, Scikit-Learn, etc... I have experience in working with various clients in countries like United States, Canada, United Kingdom, Australia, New Zealand, etc. [Check out my profile](#).



- < [PyTorch MSELoss – Detailed Guide](#)
- > [Python TensorFlow Placeholder](#)

Search

## Follow us in Twitter & Facebook

Follow @pythonguides

1,441 followers



TSInfo Technologies

YouTube 999+



PythonGuides  
8.146 seguidores

Seguir Página

Assistir ao vídeo

## Recent Posts

[How to convert a dictionary into a string in Python](#)

[How to create web form in Python Django](#)

[How to build a contact form in Django using bootstrap](#)

[How to Convert a list to DataFrame in Python](#)

[How to find the sum of digits of a number in Python](#)

