

1. Create project - RegistraionLogin and Create app – myapp in it

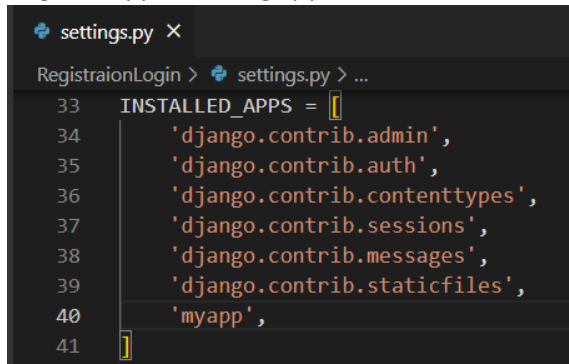
```
\my notes\Django>django-admin startproject RegistraionLogin

\my notes\Django>cd RegistraionLogin

\my notes\Django\RegistraionLogin>python manage.py startapp myapp

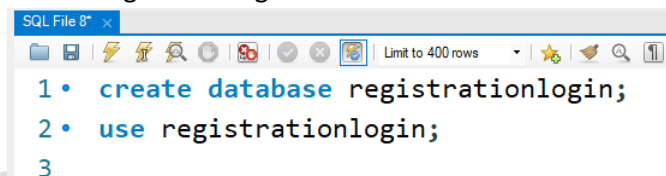
\my notes\Django\RegistraionLogin>
```

2. Register app in settings.py



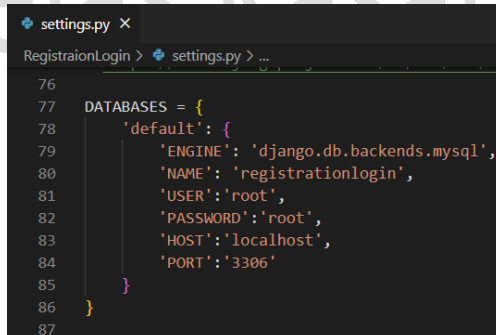
```
settings.py X
RegistraionLogin > settings.py > ...
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp',
41 ]
```

3. Create RegistraionLogin database in workbench



```
SQL File 8* X
Limit to 400 rows
1 • create database registrationlogin;
2 • use registrationlogin;
3
```

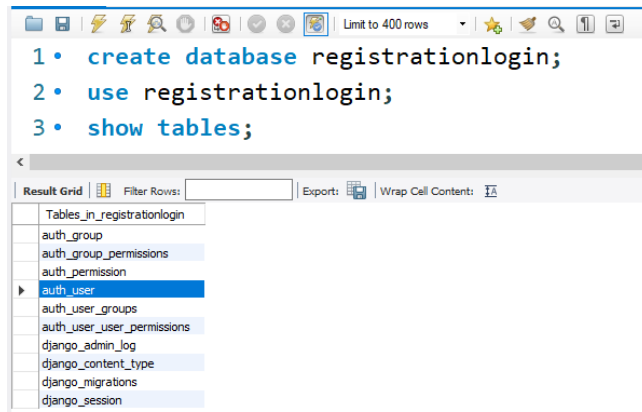
4. In settings.py change database details into mysql



```
settings.py X
RegistraionLogin > settings.py > ...
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'registrationlogin',
81         'USER': 'root',
82         'PASSWORD': 'root',
83         'HOST': 'localhost',
84         'PORT': '3306'
85     }
86 }
87
```

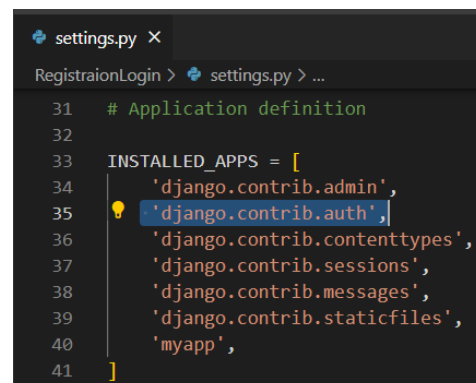
5. makemigrations
6. migrate
7. check tables in workbench

Where auth is name of application and user is name of model present inside auth application



```
1 • create database registrationlogin;
2 • use registrationlogin;
3 • show tables;
```

Tables_in_registrationlogin
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session



```
settings.py X
RegistraionLogin > settings.py > ...
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp',
41 ]
```

8. let's describe auth\_user table to get its structure

4 • desc auth\_user;

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
password	varchar(128)	NO		NULL	
last_login	datetime(6)	YES		NULL	
is_superuser	tinyint(1)	NO		NULL	
username	varchar(150)	NO	UNI	NULL	
first_name	varchar(150)	NO		NULL	
last_name	varchar(150)	NO		NULL	
email	varchar(254)	NO		NULL	
is_staff	tinyint(1)	NO		NULL	
is_active	tinyint(1)	NO		NULL	
date_joined	datetime(6)	NO		NULL	

9. create templates folder, register it in settings.py

```
settings.py X
RegistraionLogin > settings.py > ...
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends
58         'DIRS': [BASE_DIR/'templates'],
59         'APP_DIRS': True,
```

10. create myapp folder in templates and create register.html file

```
▼ REGISTRAIONLOGIN
  > myapp
  > RegistraionLogin
  ▼ templates\myapp
    <> register.html
    ⚙ manage.py
```

11. code for register.html

```
<body>
  <table align="center" border="1" cellpadding="5" cellspacing="5">
    <form method="POST">
      {% csrf_token %}
      <thead>
        <tr>
          <th colspan="2">Registration Form</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td><label for="username">UserName</label></td>
          <td><input type="text" id="username" name="username" value=""></td>
        </tr>
        <tr>
          <td><label for="password">Password</label></td>
          <td><input type="password" id="password" name="password" value=""></td>
        </tr>
        <tr>
          <td><label for="password2">Confirm Password</label></td>
          <td><input type="password" id="password2" name="password2" value=""></td>
        </tr>
        <tr>
          <td><input type="reset"></td>
          <td><input type="submit"></td>
        </tr>
      </tbody>
    </form>
  </table>
</body>
```

12. create view to show registration form

```
views.py X
myapp > views.py > ...
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5 def register_user(request):
6     return render(request, 'myapp/register.html')
7
```

13. create url pattern for above view at application level (create myapp\_urls.py in myapp folder)

```
myapp_urls.py X
myapp > myapp_urls.py > ...
1 from django.urls import path
2 from myapp import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6 ]
7
```

14. create url for myapp -> myapp\_urls.py in urls.py (at project level)

```
urls.py X
RegistraionLogin > urls.py > ...
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('myapp/', include('myapp.myapp_urls')),
23 ]
24
```

15. runserver

16. check output



Registration Form	
UserName	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>

17. create superuser

```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\RegistraionLogin>python manage.py createsuperuser
Username (leave blank to use 'admin'): tejas
Email address: tejas@gmail.com
Password:
Password (again):
Superuser created successfully.
```

18. check superuser details in auth\_user table

```
5 • select * from auth_user;
```

Result Grid										
Filter Rows:										
Edit: Export/Import: Wrap Cell Content:										
id	password	last_login	is_superuser	username	first_name	last_name	email	is_staff	is_active	date_joined
1	pbkdf2_sha256\$600000\$OLLuzQsPY0ntVDVqQ...	NULL	1	tejas	tejas	tejas@gmail.com	tejas@gmail.com	1	1	2024-01-22 09:31:40.159219
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

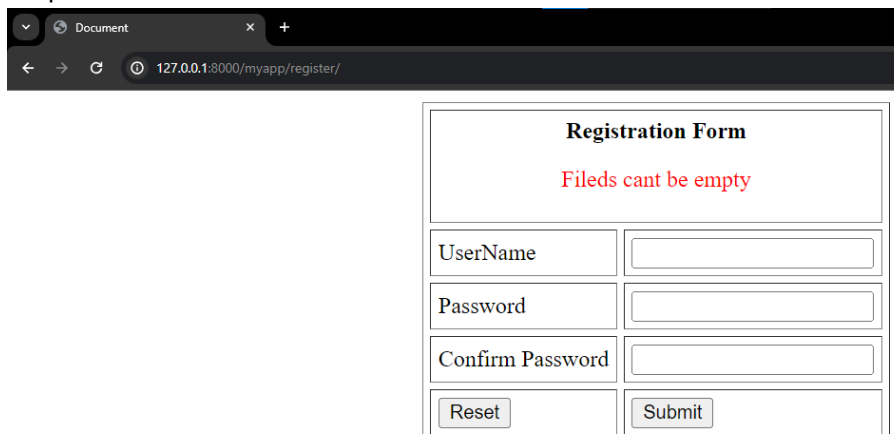
19. collect user data from registration form and insert into table – auth\_user (update view logic)

```
views.py ×
myapp > views.py > ...
4 from django.contrib.auth.models import User
5 # Create your views here.
6 def register_user(request):
7     data={}
8     if request.method=="POST":
9         uname=request.POST['username']
10        upass=request.POST['password']
11        uconf_pass=request.POST['password2']
12        #implementing validation
13        if (uname==' ' or upass =='' or uconf_pass ==''):
14            data['error_msg']='Fields cant be empty'
15            return render(request,'myapp/register.html',context=data)
16        elif(upass!=uconf_pass):
17            data['error_msg']='Password and confirm password does not matched'
18            return render(request,'myapp/register.html',context=data)
19        elif(User.objects.filter(username=uname).exists()):
20            data['error_msg']=uname + ' already exist'
21            return render(request,'myapp/register.html',context=data)
22        else:
23            user=User.objects.create(username=uname)
24            #here username and password are column names present inside auth_user table
25            user.set_password(upass) #encrypting password
26            user.save() #saving data into table
27            return HttpResponse("Registration done")
28        return render(request,'myapp/register.html')
```

20. update register.html to show error

```
<body>
<table align="center" border="1" cellpadding="5" cellspacing="5">
<form method="POST">
    {% csrf_token %}
<thead>
<tr>
<th colspan="2">
    Registration Form
    {% if error_msg %}
    <p style="color: red; font-weight: lighter;">{{error_msg}}</p>
    {% endif %}
</th>
</tr>
</thead>
```

21. output



Registration Form	
Fields cant be empty	
UserName	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>

## 22. login functionality

- a. create login.html in templates > myapp folder

```
login.html X
templates > myapp > login.html
8  <body>
9      <table align="center" border="1" cellpadding="5" cellspacing="5">
10         <form method="POST">
11             {% csrf_token %}
12         <thead>
13             <tr>
14                 <th colspan="2">
15                     Login Form
16                     {% if error_msg %}
17                     <p style="color: red; font-weight: lighter;">{{error_msg}}</p>
18                     {% endif %}
19                 </th>
20             </tr>
21         </thead>
22         <tbody>
23             <tr>
24                 <td><label for="username">UserName</label></td>
25                 <td><input type="text" id="username" name="username" value=""></td>
26             </tr>
27             <tr>
28                 <td><label for="password">Password</label></td>
29                 <td><input type="password" id="password" name="password" value=""></td>
30             </tr>
31             <tr>
32                 <td><input type="reset"></td>
33                 <td><input type="submit"></td>
34             </tr>
35         </tbody>
36     </form>
37 </table>
38 </body>
```

- b. create view to view login form

```
33 def login_user(request):
34     return render(request, 'myapp/login.html')
35
```

- c. create url for above view (in application level – myapp\_urls.py)

```
myapp_urls.py X
myapp > myapp_urls.py > ...
1  from django.urls import path
2  from myapp import views
3
4  urlpatterns = [
5      path('register/', views.register_user),
6      path('login/', views.login_user),
7  ]
8
```

- d. logic to perform login operation

```
33 def login_user(request):
34     data={}
35     if request.method=="POST":
36         uname=request.POST['username']
37         upass=request.POST['password']
38         #implementing validation
39         if (uname==' ' or upass =='' ):
40             data['error_msg']='Fields cant be empty'
41             return render(request,'myapp/login.html',context=data)
42         elif(not User.objects.filter(username=uname).exists()):
43             data['error_msg']=uname + ' user is not registered'
44             return render(request,'myapp/login.html',context=data)
45         else:
46             #from django.contrib.auth import authenticate
47             user=authenticate(username=uname,password=upass)
48             print(user)
49             if user is not None:
50                 return redirect('/myapp/home')
51             else:
52                 data['error_msg']='Wrong Password'
53                 return render(request,'myapp/login.html',context=data)
54     return render(request,'myapp/login.html')
```

There are 2 important line

- 1 – authenticate() function – accepts username and password and return User object if user exist with given credentials otherwise None(line 47 above)
- 2- on successful login, we are redirecting to home page. So we need to create home.html file, view to display home.html and finally url for view.

- e. Creating home.html in templates > myapp folder

```
<> home.html X
templates > myapp > <> home.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 | <title>Document</title>
5 </head>
6 <body>
7 | <h1>Welcome to home</h1>
8 </body>
9 </html>|
```

- f. Creating view to display home.html

```
55
56 def home(request):
57     return render(request,'myapp/home.html')
58
```

- g. url for above view

```
myapp_urls.py X
myapp > myapp_urls.py > ...
1 from django.urls import path
2 from myapp import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6     path('login/', views.login_user),
7     path('home/', views.home),
8 ]
9
```

#### h. rnsrver and check for output

Document x +  
127.0.0.1:8000/myapp/login/

**Login Form**  
Filed cant be empty

UserName

Password

Reset Submit

Document x +  
127.0.0.1:8000/myapp/login/

**Login Form**  
asfasf user is not registered

UserName

Password

Reset Submit

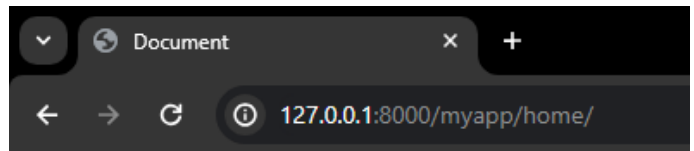
Document x +  
127.0.0.1:8000/myapp/login/

**Login Form**  
Wrong Password

UserName

Password

Reset Submit



#### 23. Session

- Session is piece of code that store on server side.
- There is built function which is responsible to start sessin – login() function
  - This login function is present inside django.contrib.auth package
    - From django.contrib.auth import login
    - Use : login(request, user\_object)
    - This login function return an id of logged in (authenticated) user (step 25)
  - On successful login, this login function stores id of authenticated user inside django\_session table

3. **show tables;**

Result Grid | Filter Rows: | Ex

Tables_in_registrationlogin
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
<b>django_session</b>

1.

6. **select \* from django\_session;**

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

session_key	session_data	expire_date
NULL	NULL	NULL

2.

c. Similarly there is logout function which is responsible for stopping the session

i. This logout function is present inside django.contrib.auth package

24. Session : Starting session using login() and checking data inside djanog\_session table

a. Import

```
#importing authentic function to login user
from django.contrib.auth import authenticate,login,logout
```

i.

b. Use login function in user\_login() function

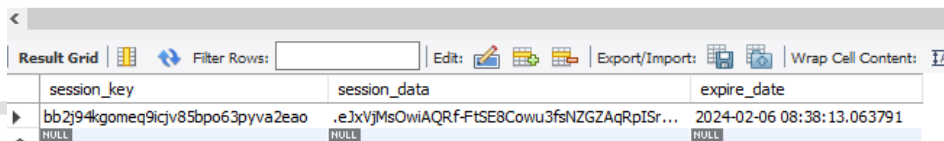
```
views.py
myapp > views.py > ...
46 #from django.contrib.auth import authenticate
47 user=authenticate(username=username,password=upass)
48 print(user)
49 if user is not None:
50     login(request,user)
51     return redirect('/myapp/home')
52 else:
53     data['error_msg']='Wrong Password'
54     return render(request,'myapp/login.html',context=data)
55 return render(request,'myapp/login.html')
```

i.

c. Do login from our login page (http://127.0.0.1:8000/myapp/login/)

d. Check data in djanog\_session table

```
6. select * from django_session;
```



The screenshot shows a database query result for the 'django\_session' table. The table has three columns: 'session\_key', 'session\_data', and 'expire\_date'. One row is displayed with the following values: 'bb2j94kgomeq9icjv85bpo63pyva2eao' for session\_key, 'eJxVjMsOwiAQRf-FtSE8Cowu3fsNZGZAqRpISr...' for session\_data, and '2024-02-06 08:38:13.063791' for expire\_date. The first two columns have a 'NULL' value in the first column and a 'NULL' value in the second column.

session_key	session_data	expire_date
bb2j94kgomeq9icjv85bpo63pyva2eao	eJxVjMsOwiAQRf-FtSE8Cowu3fsNZGZAqRpISr...	2024-02-06 08:38:13.063791

25. Session : Accessing id of logged in user after starting a session

a. Here we are going to see 2 important things :

i. How to get id of logged in user and (request.user.id)

ii. How to check user is authenticated (request.user.is\_authenticated)

b. Go to home() function and add following code

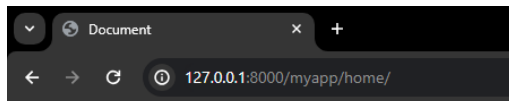
```
57 def home(request):
58     data = {}
59     #checking whether user is authenticated user
60     user_authenticated=request.user.is_authenticated
61     print(user_authenticated)
62     #if authenticated, then show home else go to login page
63     if(user_authenticated):
64         user_id = request.user.id
65         user=User.objects.get(id=user_id)
66         # print(user_id)
67         data['user_data'] = user.username
68         return render(request,'myapp/home.html',context=data)
69     else:
70         #control came here because user is not logged in
71         # so you can redirect to login page
72         data['user_data'] = "User"
73         return render(request,'myapp/home.html',context=data)
```

Getting user data in html



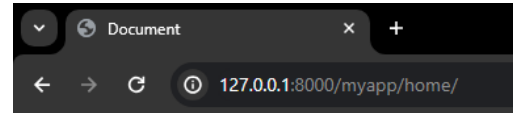
c. Code for index.html

```
home.html X
templates > myapp > home.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Document</title>
5 </head>
6 <body>
7   <h1>Welome {{user_data}}</h1>
8 </body>
9 </html>
```



**Welcome User**

Before login



**Welcome tejas**

After Login

26. Providing logout function;ity

a. Create user\_logout() in view

```
views.py X
myapp > views.py > ...
74
75 def user_logout(request):
76     logout(request)
77     return render(request, 'myapp/home.html', {'user_data': "User"})
78
79
```

b. Create url for same in myapp\_urls.py

```
myapp_urls.py X
myapp > myapp_urls.py > ...
1 from django.urls import path
2 from myapp import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6     path('login/', views.login_user),
7     path('home/', views.home),
8     path('logout/', views.user_logout),
9 ]
10
```

27. Providing Navbar in home.html with login, logout and register functionality (links)

```
<> home.html X
templates > myapp > <> home.html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Document</title>
5      <style>
6          header {
7              width: 100%;
8              height: 50px;
9              background-color: orange;
10             position: fixed;
11             display: flex;
12             align-items: center;
13             justify-content: space-around;
14         }
15         header * {
16             display: inline;
17         }
18         header li {
19             margin: 20px;
20         }
21         header li a {
22             color: blue;
23             text-decoration: none;
24         }
25     </style>
26 </head>
27 <body>
28     <h1>Welome {{user_data}}</h1>
29     <header>
30         <nav>
31             <ul>
32                 <li> <a href="/myapp/home">Home</a> </li>
33                 <li> <a href="/myapp/home">Product</a> </li>
34
35                 {% if user.is_authenticated %}
36                 <li> <a href="/myapp/logout">Logout</a> </li>
37                 {% else %}
38                 <li> <a href="/myapp/login">Login</a> </li>
39                 <li> <a href="/myapp/register">Register</a> </li>
40                 {% endif %}
41             </ul>
42         </nav>
43     </header>
44 </body>
45 </html>
```

## 28. Proving login option on registration page

```
register.html x
templates > myapp > register.html > html
36      <tr>
37          <td><input type="reset"></td>
38          <td><input type="submit"></td>
39      </tr>
40      <tr>
41          <td colspan="2">
42              <p align="center">Alredy User? Click <a href="/myapp/login">here</a> to Login</p>
43          </td>
44      </tr>
45  </tbody>
46 </form>
```



Registration Form	
UserName	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>
Alredy User? Click <a href="#">here</a> to Login	

## 29. Proving register option on login page

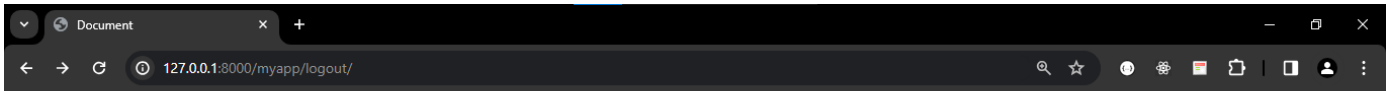
```
login.html x
templates > myapp > login.html
31      <tr>
32          <td><input type="reset"></td>
33          <td><input type="submit"></td>
34      </tr>
35      <tr>
36          <td colspan="2">
37              <p align="center">New User? Click <a href="/myapp/register">here</a> to Register</p>
38          </td>
39      </tr>
40  </tbody>
41 </form>
42 </table>
```



Login Form	
UserName	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>
New User? Click <a href="#">here</a> to Register	

### 30. Final output

#### a. Home page – Before Login



## Welome User

[Home](#) [Product](#) [Login](#) [Register](#)

#### b. Home page – After Login



## Welome tejas

[Home](#) [Product](#) [Logout](#)

### 31. DONE!!!!!!!!!!!!!!!!!!!!!! – Registration, Login, Session

Tejas Kasare - 8459859415