**SECURITY IN DJANGO**

Django has several inbuilt security features like, when we use forms, - CSRF Token has to be used., it has prevented the cross site request forgery from happening.

**Authentication:**

This is where a user or client needs to identify himself so that our application know that he is the correct users, who can access our application.

Django will create a user model object for us and then our application can simply use them.

This is the table that will be created, when we use the migrate command, the auth\_user, is the table that is which will store all the user information and mapped to a user model object in Django Security.

A screenshot of a computer

Description automatically generated

The user has logged in and now, we want to know the Roles, what he has on this application.

**Authorization:**

Once the user logs, in we have user group and permissions to check for his role and the group which he belongs.

A screenshot of a computer program

Description automatically generated

These the table, which are mapped among with the auth\_user with auth\_group and auth\_permission.

The auth\_permission is a mapping table for these tables. The auth\_user\_user\_permission are relation tables between user or mapping tables user and user permission.

A user can have multiple groups, which that group can have multiple permissions.

For example, we can assign our application (Employee), to the customer group or admin group, and give permission to these groups.

.ie. The customer group can have only add and get the employee permissions.

The admin group can have all the above permissions, including delete.

**ADDING THE AUTH URLS**

from django.urls import path, include  
from fbvEmpApp import views  
  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('',views.getEmployees),  
 path('create/', views.createEmployees),  
 path('delete/<int:id>',views.deleteEmployees),  
 path('update/<int:id>',views.updateEmployees),  
 path('accounts/', include('django.contrib.auth.urls'))  
]

**CREATE THE LOGIN FORM**

<!DOCTYPE html>  
{% load static %}  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Employee</title>  
 <link rel="stylesheet" href="{% static 'css/index.css' %}" />  
</head>  
<body>  
 <div class="myDiv">  
 <h1> EMPLOYEES LOGIN PAGE</h1>  
 <div>  
 <img src="{%static 'images/image2.jpg'%}" />  
 </div>  
 <div>  
 <form method="POST">  
 {%csrf\_token%}  
 {{form.as\_p}}  
 <button type="submit" name="button"></button>  
 </form>  
 </div>  
 </div>  
</body>  
</html>

**SECURE THE VIEWS**

Import the below in the views.py

from django.contrib.auth.decorators import login\_required

Next add all the function with the login decorator.

@login\_required

**CREATE USERS**

Creating a superuser (admin) from the Command Prompt:

~\django\_projects\fbvEmployees git:[master]

py -m manage createsuperuser

Username (leave blank to use 'kamal'): admin

Email address:

Password: admin

Password (again):admin

The password is too similar to the username.

This password is too short. It must contain at least 8 characters.

This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

Based on this, we will create a user as below:

A screenshot of a computer

Description automatically generated

Hit Save button, which is at the bottom.

This will create the user and display as below:

A screenshot of a computer

Description automatically generated

Now it has one admin user created in command prompt and one user created in the admin form.

A group of people standing together

Description automatically generated

Enter the username and password created (kamal / Jazlyn@0910). That will take back to our index page.

A group of people standing together

Description automatically generated

**LOGOUT**

Add this in the settings.py

LOGOUT\_REDIRECT\_URL='/logout'

Create a View

def logout(request):  
 return render(request, 'fbvEmpApp/logout.html')

Create a logut.html file

<!DOCTYPE html>  
{% load static %}  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Employee</title>  
 <link rel="stylesheet" href="{% static 'css/index.css' %}" />  
</head>  
<body>  
 <div class="myDiv">  
 <h1> EMPLOYEES LOGOUT PAGE</h1>  
 <div>  
 <img src="{%static 'images/image2.jpg'%}" />  
 </div>  
 <div>  
 <h2>Logged out from Employee Application</h2>  
 </div>  
 </div>  
</body>  
</html>

Map in the URL

path('logout/', views.logout)

In the index.html add this

<button type="button" class="button button1" onclick="window.location.href='/logout';">  
 LOGOUT  
</button>

Now from the home if we logout, the output will be

A group of people standing together

Description automatically generated

AUTHORIZATION

Login to admin page and for the user <kamal>, please give the permissions which was required and save.

A screenshot of a computer

Description automatically generated

Now go to the views.py in the code and add the import

from django.contrib.auth.decorators import permission\_required

Add this on top of the view

@login\_required

*# SYNTAX FOR PERMISSION\_REQUIRED IS <appname>.delete\_<modelname>*@permission\_required('fbvEmpApp.delete\_employee')

A group of people standing in front of a screen

Description automatically generated

Once you select the Delete button, this will take back to the login screen, as this user (kamal) cannot delete any records.

A group of people standing together

Description automatically generated

IF there is another user, who has rights to delete (currently admin), then I will take back to the delete screen for the confirmation page before delete.