Code structure

LMS Project Folder

File name: Asgi.py

Code: """

ASGI config for LMS\_Project project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/

"""

import os

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'LMS\_Project.settings')

application = get\_asgi\_application()

Settings.py

"""

Django settings for LMS\_Project project.

Generated by 'django-admin startproject' using Django 4.2.17.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/4.2/ref/settings/

"""

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-l9gc+b\*2pm\*knper-$bj!yre=)83g65(e+0e=@)bs7ax@i=!^z'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'users',

'courses',

]

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',

]

ROOT\_URLCONF = 'LMS\_Project.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'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',

],

},

},

]

WSGI\_APPLICATION = 'LMS\_Project.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC\_URL = 'static/'

# Default primary key field type

# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

AUTH\_USER\_MODEL = 'users.CustomUser'

LOGIN\_REDIRECT\_URL = '/redirect/' # Redirects users to the admin panel after login

LOGOUT\_REDIRECT\_URL = '/users/login/' # Redirects users to the login page after logout

Urls.py

"""

URL configuration for LMS\_Project project.

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/4.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path, include

from users.views import home, role\_based\_redirect # Import necessary views

urlpatterns = [

path('', home, name='home'), # Default root path

path('admin/', admin.site.urls), # Admin panel

path('users/', include('users.urls')), # Include users app URLs

path('redirect/', role\_based\_redirect, name='role\_based\_redirect'), # Direct redirect path

]

wsgi.py

"""

WSGI config for LMS\_Project project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'LMS\_Project.settings')

application = get\_wsgi\_application()

Users/Migration

0001\_intial.py

# Generated by Django 4.2.17 on 2024-12-19 10:21

import django.contrib.auth.models

import django.contrib.auth.validators

from django.db import migrations, models

import django.utils.timezone

class Migration(migrations.Migration):

initial = True

dependencies = [

('auth', '0012\_alter\_user\_first\_name\_max\_length'),

]

operations = [

migrations.CreateModel(

name='CustomUser',

fields=[

('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('password', models.CharField(max\_length=128, verbose\_name='password')),

('last\_login', models.DateTimeField(blank=True, null=True, verbose\_name='last login')),

('is\_superuser', models.BooleanField(default=False, help\_text='Designates that this user has all permissions without explicitly assigning them.', verbose\_name='superuser status')),

('username', models.CharField(error\_messages={'unique': 'A user with that username already exists.'}, help\_text='Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.', max\_length=150, unique=True, validators=[django.contrib.auth.validators.UnicodeUsernameValidator()], verbose\_name='username')),

('first\_name', models.CharField(blank=True, max\_length=150, verbose\_name='first name')),

('last\_name', models.CharField(blank=True, max\_length=150, verbose\_name='last name')),

('email', models.EmailField(blank=True, max\_length=254, verbose\_name='email address')),

('is\_staff', models.BooleanField(default=False, help\_text='Designates whether the user can log into this admin site.', verbose\_name='staff status')),

('is\_active', models.BooleanField(default=True, help\_text='Designates whether this user should be treated as active. Unselect this instead of deleting accounts.', verbose\_name='active')),

('date\_joined', models.DateTimeField(default=django.utils.timezone.now, verbose\_name='date joined')),

('role', models.CharField(choices=[('admin', 'Admin'), ('instructor', 'Instructor'), ('student', 'Student')], max\_length=20)),

('groups', models.ManyToManyField(blank=True, help\_text='The groups this user belongs to.', related\_name='customuser\_set', to='auth.group')),

('user\_permissions', models.ManyToManyField(blank=True, help\_text='Specific permissions for this user.', related\_name='customuser\_permissions\_set', to='auth.permission')),

],

options={

'verbose\_name': 'user',

'verbose\_name\_plural': 'users',

'abstract': False,

},

managers=[

('objects', django.contrib.auth.models.UserManager()),

],

),

]

0002\_alter\_customuser\_role.py

# Generated by Django 4.2.17 on 2024-12-19 13:42

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

('users', '0001\_initial'),

]

operations = [

migrations.AlterField(

model\_name='customuser',

name='role',

field=models.CharField(choices=[('superadmin', 'SuperAdmin'), ('admin', 'Admin'), ('instructor', 'Instructor'), ('learner', 'Learner')], default='learner', max\_length=20),

),

]

Templates/Users

Login.html

<h2>Login</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Login</button>

</form>

register.html

<h2>Register</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Register</button>

</form>

Templates/users/superadmin

dashboard.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SuperAdmin Dashboard</title>

</head>

<body>

<h1>Welcome to the SuperAdmin Dashboard</h1>

<h2>System Overview</h2>

<ul>

<li>Total Users: {{ total\_users }}</li>

<li>Total Active Courses: {{ total\_courses }}</li>

</ul>

</body>

</html>

Users

admin.py

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from .models import CustomUser

@admin.register(CustomUser)

class CustomUserAdmin(UserAdmin):

# Display these fields in the admin list view

list\_display = ['username', 'email', 'role', 'is\_staff', 'is\_active']

list\_filter = ['is\_staff', 'is\_active', 'role'] # Add filters for quick access

# Add 'role' to the editable fields when creating or editing users

fieldsets = UserAdmin.fieldsets + (

('Custom Fields', {'fields': ('role',)}),

)

add\_fieldsets = UserAdmin.add\_fieldsets + (

('Custom Fields', {'fields': ('role',)}),

)

apps.py

from django.apps import AppConfig

class UsersConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'users'

forms.py

from django import forms

from django.contrib.auth.forms import UserCreationForm

from .models import CustomUser

class CustomUserCreationForm(UserCreationForm):

class Meta:

model = CustomUser

fields = ['username', 'email', 'role', 'password1', 'password2']

models.py

from django.contrib.auth.models import AbstractUser, Group, Permission

from django.db import models

class CustomUser(AbstractUser):

ROLE\_CHOICES = [

('superadmin', 'SuperAdmin'),

('admin', 'Admin'),

('instructor', 'Instructor'),

('learner', 'Learner'),

]

role = models.CharField(

max\_length=20,

choices=ROLE\_CHOICES,

default='learner' # Default role

)

# Fixing clashes by specifying related\_name

groups = models.ManyToManyField(

Group,

related\_name="customuser\_set",

blank=True,

help\_text="The groups this user belongs to.",

)

user\_permissions = models.ManyToManyField(

Permission,

related\_name="customuser\_permissions\_set",

blank=True,

help\_text="Specific permissions for this user.",

)

tests.py

from django.test import TestCase

# Create your tests here.

urls.py

from django.urls import path

from . import views

from django.contrib.auth import views as auth\_views

urlpatterns = [

path('register/', views.register, name='register'),

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'), # Login URL

path('redirect/', views.role\_based\_redirect, name='role\_based\_redirect'), # Redirection URL

path('dashboard/superadmin/', views.super\_admin\_dashboard, name='super\_admin\_dashboard'),

path('dashboard/admin/', views.admin\_dashboard, name='admin\_dashboard'),

path('dashboard/instructor/', views.instructor\_dashboard, name='instructor\_dashboard'),

path('dashboard/learner/', views.learner\_dashboard, name='learner\_dashboard'),

]

views.py

from django.shortcuts import render, redirect

from .forms import CustomUserCreationForm

from django.http import HttpResponse

from users.models import CustomUser # Import CustomUser model

# Replace 'courses.models' with the actual path of your Course model

from courses.models import Course

# Registration view

def register(request):

if request.method == 'POST':

form = CustomUserCreationForm(request.POST)

if form.is\_valid():

form.save()

return redirect('login') # Redirect to login page after registration

else:

form = CustomUserCreationForm()

return render(request, 'users/register.html', {'form': form})

# Dashboard views

def super\_admin\_dashboard(request):

total\_users = CustomUser.objects.count()

total\_courses = Course.objects.count() # Ensure Course model exists and is imported

return render(request, 'users/superadmin/dashboard.html', {

'total\_users': total\_users,

'total\_courses': total\_courses,

})

def admin\_dashboard(request):

return HttpResponse("Welcome to the Admin Dashboard")

def instructor\_dashboard(request):

return HttpResponse("Welcome to the Instructor Dashboard")

def learner\_dashboard(request):

return HttpResponse("Welcome to the Learner Dashboard")

def home(request):

return HttpResponse("Welcome to the LMS Home Page!")

# Role-based redirection

def role\_based\_redirect(request):

if not request.user.is\_authenticated:

return redirect('login')

if request.user.role == 'superadmin':

return redirect('super\_admin\_dashboard')

elif request.user.role == 'admin':

return redirect('admin\_dashboard')

elif request.user.role == 'instructor':

return redirect('instructor\_dashboard')

elif request.user.role == 'learner':

return redirect('learner\_dashboard')

else:

return redirect('login')

manage.py

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

"""Run administrative tasks."""

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'LMS\_Project.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Courses/migration

001\_initial.py

# Generated by Django 4.2.17 on 2024-12-19 15:41

from django.db import migrations, models

class Migration(migrations.Migration):

initial = True

dependencies = [

]

operations = [

migrations.CreateModel(

name='Course',

fields=[

('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('title', models.CharField(max\_length=255)),

('description', models.TextField()),

('is\_active', models.BooleanField(default=True)),

],

),

]

courses

admin.py

from django.contrib import admin

from .models import Course

# Register your models here.

@admin.register(Course)

class CourseAdmin(admin.ModelAdmin):

list\_display = ['title', 'is\_active']

apps.py

from django.apps import AppConfig

class CoursesConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'courses'

models.py

from django.db import models

# Create your models here.

class Course(models.Model):

title = models.CharField(max\_length=255)

description = models.TextField()

is\_active = models.BooleanField(default=True)

def \_\_str\_\_(self):

return self.title

tests.py

from django.test import TestCase

# Create your tests here.

views.py