Code & File Structure

**File Structure**

LMS\_Project/

├── LMS\_Project/ # Django configuration files

│ ├── \_\_init\_\_.py

│ ├── settings.py # Global settings

│ ├── urls.py # Project-level URL routing

│ ├── asgi.py

│ ├── wsgi.py

├── Project Documentation/ # Documentation and SRS

│ ├── SRS/

│ │ ├── Software Requirement Specification v1.0.docx

│ │ ├── Software Requirement Specification v1.2.docx

│ ├── Work Statements/

│ │ ├── LMS Project Diary 1.docx

│ └── Project Documents/

│ ├── Chat Prompts/

│ │ ├── Chat History/

│ │ │ ├── Chat Logs (various dates).docx

│ ├── Code structure notes

├── courses/ # Course-related functionalities

│ ├── models.py # Course models

│ ├── views.py # Views for course-related operations

│ ├── templates/ # Course templates (e.g., SCORM playback)

│ └── forms.py # Forms for course content upload

├── users/ # User-related functionalities

│ ├── models.py # Custom user model

│ ├── views.py # User authentication and dashboards

│ └── templates/ # User templates

├── Scorm/ # SCORM-related functionalities

│ ├── database/ # Mock and seed data

│ │ ├── mocks/

│ │ ├── seeders/

│ ├── app/ # SCORM-related logic

│ │ ├── Controllers/

│ │ ├── Models/

│ └── resources/

│ ├── views/ # Blade templates for SCORM

├── README.md # Project overview

└── manage.py # Django management script

1. LMS\_Project
   1. /Users/harikrishnan/LMS\_Project/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()

* 1. /Users/harikrishnan/LMS\_Project/LMS\_Project/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

import os

# 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', # Custom users app

'courses', # Courses app

]

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': [BASE\_DIR / "templates"], # Directory for custom 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',

],

},

},

]

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': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

# Override database settings for CI environment

if os.getenv('GITHUB\_ACTIONS'):

DATABASES['default'] = {

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

'NAME': ':memory:', # Use in-memory SQLite database for CI pipelines

}

# Password validation

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

STATICFILES\_DIRS = [BASE\_DIR / 'static']

# Media files (SCORM content and other uploaded files)

MEDIA\_URL = '/course\_content/'

MEDIA\_ROOT = BASE\_DIR / 'course\_content'

# Default primary key field type

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

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

# Custom user model

AUTH\_USER\_MODEL = 'users.CustomUser'

# Login and logout settings

LOGIN\_REDIRECT\_URL = '/redirect/' # Redirects users to the role-based redirect view after login

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

LOGIN\_URL = '/users/login/' # Ensures @login\_required redirects to the correct login page

# Session engine (ensure session table is created)

SESSION\_ENGINE = 'django.contrib.sessions.backends.db'

# Logging for Debugging

# Enable logging to monitor errors in production and testing

LOGGING = {

'version': 1,

'disable\_existing\_loggers': False,

'handlers': {

'console': {

'class': 'logging.StreamHandler',

},

},

'root': {

'handlers': ['console'],

'level': 'DEBUG' if DEBUG else 'ERROR',

},

}

# Ensure static directory exists and create placeholder files if necessary (Development Only)

if DEBUG:

STATICFILES\_INIT = [

('favicon.ico', b''),

('apple-touch-icon.png', b''),

('apple-touch-icon-precomposed.png', b''),

]

if not os.path.exists(BASE\_DIR / 'static'):

os.makedirs(BASE\_DIR / 'static')

for file\_name, content in STATICFILES\_INIT:

file\_path = BASE\_DIR / 'static' / file\_name

if not file\_path.exists():

with open(file\_path, 'wb') as f:

f.write(content)

* 1. /Users/harikrishnan/LMS\_Project/LMS\_Project/urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

from users.views import role\_based\_redirect, home

urlpatterns = [

path('', home, name='home'),

path('admin/', admin.site.urls),

path('users/', include('users.urls')),

path('courses/', include('courses.urls')),

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

]

# Serve media files in development

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

# Serve static files in development

if settings.DEBUG:

# Safely get the first STATICFILES\_DIRS entry if it exists

static\_root = settings.STATICFILES\_DIRS[0] if settings.STATICFILES\_DIRS else None

if static\_root:

urlpatterns += static(settings.STATIC\_URL, document\_root=static\_root)

* 1. **/Users/harikrishnan/LMS\_Project/LMS\_Project/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()

1. Courses
   1. /Users/harikrishnan/LMS\_Project/courses/\_\_init\_\_.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()

* 1. /Users/harikrishnan/LMS\_Project/courses/admin.py

from django.contrib import admin

from .models import Course

@admin.register(Course)

class CourseAdmin(admin.ModelAdmin):

# Display these fields in the list view

list\_display = ('title', 'branch', 'instructor', 'content\_type', 'is\_active', 'created\_at')

# Filters for branch, content type, and active status

list\_filter = ('branch', 'content\_type', 'is\_active')

# Searchable fields

search\_fields = ('title', 'description')

# Enables multi-select for enrolled learners

filter\_horizontal = ('enrolled\_users',)

# Organizing fields into sections

fieldsets = (

(None, {

'fields': ('title', 'description', 'is\_active', 'branch', 'instructor', 'enrolled\_users')

}),

('Content Details', {

'fields': ('content\_type', 'content\_file'),

'description': "Upload course content and specify the content type (SCORM, Video, or Document)."

}),

('Timestamps', {

'fields': ('created\_at', 'updated\_at'),

}),

)

# Ensure timestamps are read-only

readonly\_fields = ('created\_at', 'updated\_at')

* 1. /Users/harikrishnan/LMS\_Project/courses/apps.py

from django.apps import AppConfig

class CoursesConfig(AppConfig):

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

name = 'courses'

* 1. /Users/harikrishnan/LMS\_Project/courses/forms.py

from django import forms

from .models import Course

class CourseForm(forms.ModelForm):

class Meta:

model = Course

fields = ['title', 'description', 'branch', 'instructor', 'content\_type', 'content\_file']

* 1. /Users/harikrishnan/LMS\_Project/courses/models.py

from django.db import models

from users.models import Branch, CustomUser

def content\_file\_path(instance, filename):

if isinstance(instance, Course):

return f"course\_content/{instance.id}/{filename}"

elif isinstance(instance, SCORM):

return f"course\_content/scorm/{instance.course.id}/{filename}"

class Course(models.Model):

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

description = models.TextField()

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

branch = models.ForeignKey(

Branch,

on\_delete=models.CASCADE,

related\_name="courses",

null=True,

blank=True,

help\_text="The branch this course belongs to."

)

instructor = models.ForeignKey(

CustomUser,

on\_delete=models.SET\_NULL,

null=True,

blank=True,

related\_name="instructor\_courses",

limit\_choices\_to={"role": "instructor"},

help\_text="The instructor assigned to this course."

)

enrolled\_users = models.ManyToManyField(

CustomUser,

related\_name="enrolled\_courses",

blank=True,

limit\_choices\_to={"role": "learner"},

help\_text="The learners enrolled in this course."

)

content\_type = models.CharField(

max\_length=50,

choices=[

('SCORM', 'SCORM'),

('Video', 'Video'),

('Document', 'Document'),

],

null=True,

blank=True,

help\_text="The type of content for this course."

)

content\_file = models.FileField(

upload\_to=content\_file\_path,

null=True,

blank=True,

help\_text="The file associated with this course (SCORM, Video, or Document)."

)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

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

return f"Course: {self.title} (Branch: {self.branch})"

class SCORM(models.Model):

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

course = models.ForeignKey(

Course,

on\_delete=models.CASCADE,

related\_name="scorms",

help\_text="The course this SCORM package belongs to."

)

launch\_url = models.CharField(max\_length=255)

version = models.CharField(max\_length=50, default="SCORM 1.2")

runtime\_tracking = models.JSONField(

default=dict,

blank=True,

help\_text="Tracks learner progress, scores, and completion status."

)

time\_spent = models.DecimalField(

max\_digits=6, # Allow larger values for total time

decimal\_places=2,

default=0.0,

help\_text="Total time spent by the learner in hours."

)

completion\_status = models.CharField(

max\_length=20,

choices=[

('not\_started', 'Not Started'),

('in\_progress', 'In Progress'),

('completed', 'Completed'),

],

default='not\_started',

help\_text="The current completion status of the SCORM package."

)

score = models.IntegerField(

null=True,

blank=True,

help\_text="The score returned from the SCORM package (if applicable)."

)

last\_accessed = models.DateTimeField(

auto\_now=True,

help\_text="The timestamp for the last access of the SCORM package."

)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class Meta:

unique\_together = ('course', 'title')

verbose\_name = "SCORM Package"

verbose\_name\_plural = "SCORM Packages"

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

return f"{self.title} ({self.version})"

* 1. /Users/harikrishnan/LMS\_Project/courses/tests.py

from django.test import TestCase

# Create your tests here.

* 1. /Users/harikrishnan/LMS\_Project/courses/urls.py

from django.urls import path

from . import views

urlpatterns = [

path('', views.course\_list, name='course\_list'),

path('upload/', views.upload\_content, name='upload\_content'),

path('scorm/play/<int:course\_id>/', views.play\_scorm, name='play\_scorm'), # Updated to align with play\_scorm

path('scorm/play/view/<int:course\_id>/', views.scorm\_playback, name='scorm\_playback'),

path('scorm/runtime/update/<int:course\_id>/', views.scorm\_runtime\_update, name='scorm\_runtime\_update'),

]

* 1. /Users/harikrishnan/LMS\_Project/courses/views.py

from django.shortcuts import render, redirect, get\_object\_or\_404

from .models import Course, SCORM

from .forms import CourseForm

from django.http import JsonResponse, HttpResponse

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

from django.views.decorators.csrf import csrf\_exempt

import logging

import json

# Set up logging

logger = logging.getLogger(\_\_name\_\_)

@login\_required

def course\_list(request):

courses = Course.objects.all()

return render(request, 'courses/list/course\_list.html', {'courses': courses})

@login\_required

def upload\_content(request):

if request.method == "POST":

form = CourseForm(request.POST, request.FILES)

if form.is\_valid():

uploaded\_file = request.FILES['content\_file']

allowed\_types = ['application/zip', 'video/mp4', 'application/pdf']

if uploaded\_file.content\_type not in allowed\_types:

return JsonResponse({"error": "Unsupported file type"}, status=400)

form.save()

return redirect('course\_list')

else:

form = CourseForm()

return render(request, 'courses/upload/upload\_content.html', {'form': form})

@login\_required

def scorm\_playback(request, course\_id):

try:

scorm\_package = get\_object\_or\_404(SCORM, course\_id=course\_id)

logger.info(f"SCORM Package Found: {scorm\_package.title}, ID: {scorm\_package.id}")

return render(request, 'courses/scorm/playback.html', {

'scorm\_title': scorm\_package.title,

'launch\_url': scorm\_package.launch\_url,

'course\_id': course\_id

})

except Exception as e:

logger.error(f"Error during SCORM playback: {str(e)}")

return HttpResponse("An unexpected error occurred.", status=500)

@login\_required

def play\_scorm(request, course\_id):

"""

Retrieve SCORM metadata and provide a launch URL for playback.

"""

try:

scorm = get\_object\_or\_404(SCORM, course\_id=course\_id)

if request.GET.get('format') == 'json':

return JsonResponse({

'title': scorm.title,

'description': scorm.course.description,

'launch\_url': scorm.launch\_url,

})

return render(request, 'courses/play\_scorm.html', {'scorm': scorm})

except Exception as e:

logger.error(f"Error in play\_scorm: {str(e)}")

return JsonResponse({"error": "SCORM playback failed"}, status=500)

@login\_required

@csrf\_exempt

def scorm\_runtime\_update(request, course\_id):

if request.method == "POST":

try:

data = json.loads(request.body)

scorm\_package = get\_object\_or\_404(SCORM, course\_id=course\_id)

progress = data.get('progress', 0)

score = data.get('score')

completion\_status = data.get('completion\_status', 'not\_started')

scorm\_package.time\_spent += progress

scorm\_package.score = score or scorm\_package.score

scorm\_package.completion\_status = completion\_status

scorm\_package.last\_accessed = scorm\_package.updated\_at

scorm\_package.save()

logger.info(f"SCORM Runtime Updated: Course ID: {course\_id}, Data: {data}")

return JsonResponse({"status": "success"})

except Exception as e:

logger.error(f"Runtime tracking error: {str(e)}")

return JsonResponse({"error": "Unexpected error"}, status=500)

* 1. /Users/harikrishnan/LMS\_Project/courses/migrations/\_\_init\_\_.py

Blank

* 1. /Users/harikrishnan/LMS\_Project/courses/migrations/0001\_initial.py

# Generated by Django 4.2.17 on 2024-12-23 06:05

import courses.models

from django.conf import settings

from django.db import migrations, models

import django.db.models.deletion

class Migration(migrations.Migration):

initial = True

dependencies = [

('users', '0002\_alter\_customuser\_options\_and\_more'),

migrations.swappable\_dependency(settings.AUTH\_USER\_MODEL),

]

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)),

('content\_type', models.CharField(

blank=True,

choices=[('SCORM', 'SCORM'), ('Video', 'Video'), ('Document', 'Document')],

help\_text='The type of content for this course.',

max\_length=50,

null=True

)),

('content\_file', models.FileField(

blank=True,

help\_text='The file associated with this course (SCORM, Video, or Document).',

null=True,

upload\_to=courses.models.content\_file\_path

)),

('created\_at', models.DateTimeField(auto\_now\_add=True)),

('updated\_at', models.DateTimeField(auto\_now=True)),

('branch', models.ForeignKey(

blank=True,

help\_text='The branch this course belongs to.',

null=True,

on\_delete=django.db.models.deletion.CASCADE,

related\_name='courses',

to='users.branch'

)),

('enrolled\_users', models.ManyToManyField(

blank=True,

help\_text='The learners enrolled in this course.',

limit\_choices\_to={'role': 'learner'},

related\_name='enrolled\_courses',

to=settings.AUTH\_USER\_MODEL

)),

('instructor', models.ForeignKey(

blank=True,

help\_text='The instructor assigned to this course.',

limit\_choices\_to={'role': 'instructor'},

null=True,

on\_delete=django.db.models.deletion.SET\_NULL,

related\_name='instructor\_courses',

to=settings.AUTH\_USER\_MODEL

)),

],

),

migrations.CreateModel(

name='SCORM',

fields=[

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

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

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

('version', models.CharField(default='SCORM 1.2', max\_length=50)),

('runtime\_tracking', models.JSONField(

blank=True,

default=dict,

help\_text='Tracks learner progress, scores, and completion status.'

)),

('time\_spent', models.DecimalField(

decimal\_places=2,

default=0.0,

help\_text='Total time spent by the learner in hours.',

max\_digits=6

)),

('completion\_status', models.CharField(

choices=[

('not\_started', 'Not Started'),

('in\_progress', 'In Progress'),

('completed', 'Completed')

],

default='not\_started',

help\_text='The current completion status of the SCORM package.',

max\_length=20

)),

('score', models.IntegerField(

blank=True,

help\_text='The score returned from the SCORM package (if applicable).',

null=True

)),

('last\_accessed', models.DateTimeField(

auto\_now=True,

help\_text='The timestamp for the last access of the SCORM package.'

)),

('created\_at', models.DateTimeField(auto\_now\_add=True)),

('updated\_at', models.DateTimeField(auto\_now=True)),

('course', models.ForeignKey(

help\_text='The course this SCORM package belongs to.',

on\_delete=django.db.models.deletion.CASCADE,

related\_name='scorms',

to='courses.course'

)),

],

options={

'verbose\_name': 'SCORM Package',

'verbose\_name\_plural': 'SCORM Packages',

'unique\_together': {('course', 'title')},

},

),

]

* 1. /Users/harikrishnan/LMS\_Project/courses/templates/courses/list/course\_list.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Course List</title>

</head>

<body>

<h1>Courses</h1>

<ul>

{% for course in courses %}

<li>{{ course.title }} - Branch: {{ course.branch.name }}</li>

{% endfor %}

</ul>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/courses/templates/courses/scorm/playback.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>{{ scorm\_title }}</title>

</head>

<body>

<h1>{{ scorm\_title }}</h1>

<iframe src="{{ launch\_url }}" width="100%" height="600" frameborder="0"></iframe>

<script>

// Example of runtime data tracking (replace with SCORM API data)

const runtimeData = {

progress: 50, // Replace with actual SCORM data

score: 85,

completion\_status: "in\_progress"

};

// Dynamically include the course\_id in the runtime tracking endpoint

fetch(`/courses/scorm/runtime/update/{{ course\_id }}/`, {

method: "POST",

headers: {

"Content-Type": "application/json",

"X-CSRFToken": "{{ csrf\_token }}",

},

body: JSON.stringify(runtimeData)

}).then(response => response.json())

.then(data => console.log("Runtime Update Response:", data))

.catch(error => console.error('Error:', error));

</script>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/courses/templates/courses/upload/upload\_content.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Upload Course Content</title>

</head>

<body>

<h1>Upload Course Content</h1>

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

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

</form>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/courses/templates/courses/play\_scorm.html

<!DOCTYPE html>

<html>

<head>

<title>{{ scorm.title }}</title>

</head>

<body>

<h1>{{ scorm.title }}</h1>

<p>{{ scorm.description }}</p>

<iframe src="{{ scorm.launch\_url }}" width="100%" height="600px" allowfullscreen></iframe>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/courses/course\_tests/\_\_init\_\_.py

Blank

* 1. /Users/harikrishnan/LMS\_Project/courses/course\_tests/test\_scorm\_playback.py

from django.test import TestCase

from courses.models import SCORM, Course

class SCORMPlaybackTestCase(TestCase):

def setUp(self):

# Create a test course and SCORM module

self.course = Course.objects.create(title="Test Course", description="Test Description")

self.scorm = SCORM.objects.create(

title="Test SCORM",

course=self.course,

launch\_url="/course\_content/scorm/1/test/index.html"

)

def test\_scorm\_launch\_url(self):

# Verify the SCORM launch URL response

response = self.client.get(self.scorm.launch\_url)

self.assertEqual(response.status\_code, 404) # Adjust to 200 if the URL is valid and served

* 1. /Users/harikrishnan/LMS\_Project/courses/course\_tests/test\_scorm\_runtime.py

from django.test import TestCase

from courses.models import SCORM, Course

from users.models import CustomUser

class SCORMRuntimeTestCase(TestCase):

def setUp(self):

# Create a learner

self.learner = CustomUser.objects.create\_user(username='Care\_Learner1', password='password', role='Learner')

# Create a course and SCORM module

self.course = Course.objects.create(title="Your Personal Development", description="SCORM Content")

self.scorm = SCORM.objects.create(

title="SCORM Module",

course=self.course,

launch\_url="/course\_content/scorm/1/1648213263\_your-personal-development-in-care/scormcontent/index.html"

)

def test\_scorm\_runtime\_tracking(self):

# Simulate SCORM runtime update

self.client.login(username='Care\_Learner1', password='password')

response = self.client.post(f"/courses/scorm/runtime/update/{self.course.id}/", data={

"progress": 50,

"status": "in\_progress",

"start\_time": "2024-12-23T12:00:00Z",

"end\_time": "2024-12-23T12:30:00Z"

}, content\_type="application/json")

self.assertEqual(response.status\_code, 200)

self.assertEqual(response.json().get("status"), "success")

1. Users
   1. /Users/harikrishnan/LMS\_Project/templates/users/login.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form method="POST" action="/users/login/">

{% csrf\_token %}

<label for="username">Username:</label>

<input type="text" name="username" id="username" required><br>

<label for="password">Password:</label>

<input type="password" name="password" id="password" required><br>

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

</form>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/migrations/\_\_init\_\_.py

Blank

* 1. /Users/harikrishnan/LMS\_Project/users/migrations/0001\_initial.py

# Generated by Django 4.2.17 on 2024-12-21 17:04

import django.contrib.auth.models

import django.contrib.auth.validators

from django.db import migrations, models

import django.db.models.deletion

import django.utils.timezone

class Migration(migrations.Migration):

initial = True

dependencies = [

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

]

operations = [

migrations.CreateModel(

name='Branch',

fields=[

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

('name', models.CharField(max\_length=255, unique=True)),

('created\_at', models.DateTimeField(auto\_now\_add=True)),

('updated\_at', models.DateTimeField(auto\_now=True)),

],

),

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

('language', models.CharField(default='en', help\_text='Preferred language of the user.', max\_length=10)),

('timezone', models.CharField(default='UTC', help\_text='Preferred timezone of the user.', max\_length=50)),

('branch', models.ForeignKey(blank=True, help\_text='The branch this user belongs to.', null=True, on\_delete=django.db.models.deletion.SET\_NULL, related\_name='users', to='users.branch')),

('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()),

],

),

]

* 1. /Users/harikrishnan/LMS\_Project/users/migrations/0002\_alter\_customuser\_options\_and\_more.py

# Generated by Django 4.2.17 on 2024-12-21 22:30

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

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

]

operations = [

migrations.AlterModelOptions(

name='customuser',

options={'ordering': ['-date\_joined'], 'verbose\_name': 'User', 'verbose\_name\_plural': 'Users'},

),

migrations.AlterField(

model\_name='customuser',

name='date\_joined',

field=models.DateTimeField(auto\_now\_add=True, help\_text='The date and time the user joined the system.'),

),

migrations.AlterField(

model\_name='customuser',

name='is\_active',

field=models.BooleanField(default=True, help\_text='Indicates whether this user account is active.'),

),

migrations.AlterField(

model\_name='customuser',

name='role',

field=models.CharField(choices=[('superadmin', 'SuperAdmin'), ('admin', 'Admin'), ('instructor', 'Instructor'), ('learner', 'Learner')], default='learner', help\_text='Role of the user (e.g., admin, instructor, learner).', max\_length=20),

),

]

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/dashboards/admin.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Admin Dashboard</title>

</head>

<body>

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

<h2>Branch Overview: {{ branch\_name }}</h2>

<ul>

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

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

</ul>

<h3>Branch Courses</h3>

<ul>

{% for course in branch\_courses %}

<li>{{ course.title }}</li>

{% endfor %}

</ul>

<h3>Quick Links</h3>

<ul>

<li><a href="/admin/">Admin Panel</a></li>

<li><a href="/courses/">Manage Courses</a></li>

<li><a href="/users/">Manage Users</a></li>

<li><a href="/users/logout/">Logout</a></li>

</ul>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/dashboards/instructor.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Instructor Dashboard</title>

</head>

<body>

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

<h2>Assigned Courses</h2>

<ul>

{% for course in assigned\_courses %}

<li>{{ course.title }} - {{ course.description }}</li>

{% endfor %}

</ul>

<h2>SCORM Progress for Learners</h2>

<ul>

{% for learner\_data in learner\_scorm\_progress %}

<li>

<strong>{{ learner\_data.learner.username }}</strong>

<ul>

{% for scorm in learner\_data.scorm\_progress %}

<li>

<a href="{{ scorm.launch\_url }}" target="\_blank"><strong>{{ scorm.title }}</strong></a>

<ul>

<li>Time Spent: {{ scorm.time\_spent }} hrs</li>

<li>Completion Status: {{ scorm.completion\_status }}</li>

<li>Score: {% if scorm.score %}{{ scorm.score }}{% else %}N/A{% endif %}</li>

</ul>

</li>

{% empty %}

<li>No SCORM progress available for this learner.</li>

{% endfor %}

</ul>

</li>

{% endfor %}

</ul>

<ul>

<li><a href="/users/logout/">Logout</a></li>

</ul>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/dashboards/learner.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Learner Dashboard</title>

</head>

<body>

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

<h2>Enrolled Courses</h2>

<ul>

{% for course in enrolled\_courses %}

<li>{{ course.title }} - {{ course.description }}</li>

{% endfor %}

</ul>

<h2>Your SCORM Courses</h2>

<ul>

{% for scorm in scorm\_content %}

<li>

<a href="{{ scorm.launch\_url }}" target="\_blank"><strong>{{ scorm.title }}</strong></a>

<ul>

<li>Time Spent: {{ scorm.time\_spent }} hrs</li>

<li>Completion Status: {{ scorm.completion\_status }}</li>

<li>Score: {% if scorm.score %}{{ scorm.score }}{% else %}N/A{% endif %}</li>

</ul>

</li>

{% empty %}

<li>No SCORM content available for your courses.</li>

{% endfor %}

</ul>

<ul>

<li><a href="/users/logout/">Logout</a></li>

</ul>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/dashboards/superadmin.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>

<ul>

<li><a href="/users/logout/">Logout</a></li>

</ul>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/shared/login.html

<h2>Login</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

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

</form>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/shared/register.html

<h2>Register</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

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

</form>

* 1. /Users/harikrishnan/LMS\_Project/users/Templates/users/shared/user\_list.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User List</title>

</head>

<body>

<h1>User List</h1>

<table border="1">

<thead>

<tr>

<th>Username</th>

<th>Role</th>

<th>Branch</th>

</tr>

</thead>

<tbody>

{% for user in users %}

<tr>

<td>{{ user.username }}</td>

<td>{{ user.role }}</td>

<td>

{% if user.branch %}

{{ user.branch.name }}

{% else %}

No Branch

{% endif %}

</td>

</tr>

{% endfor %}

</tbody>

</table>

</body>

</html>

* 1. /Users/harikrishnan/LMS\_Project/users/\_\_init\_\_.py

Blank

* 1. /Users/harikrishnan/LMS\_Project/users/admin.py

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from .models import CustomUser, Branch

@admin.register(CustomUser)

class CustomUserAdmin(UserAdmin):

# Display these fields in the admin list view

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

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

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

fieldsets = UserAdmin.fieldsets + (

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

)

add\_fieldsets = UserAdmin.add\_fieldsets + (

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

)

# Make 'date\_joined' and 'last\_login' readonly to prevent editing

readonly\_fields = ('date\_joined', 'last\_login')

@admin.register(Branch)

class BranchAdmin(admin.ModelAdmin):

list\_display = ['name', 'created\_at', 'updated\_at'] # Display branch details

search\_fields = ['name'] # Add search functionality for branches

* 1. /Users/harikrishnan/LMS\_Project/users/apps.py

from django.apps import AppConfig

class UsersConfig(AppConfig):

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

name = 'users'

* 1. /Users/harikrishnan/LMS\_Project/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']

* 1. /Users/harikrishnan/LMS\_Project/users/models.py

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

from django.db import models

class Branch(models.Model):

name = models.CharField(max\_length=255, unique=True)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

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

return self.name

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

help\_text="Role of the user (e.g., admin, instructor, learner)."

)

branch = models.ForeignKey(

Branch,

on\_delete=models.SET\_NULL,

null=True,

blank=True,

related\_name='users',

help\_text="The branch this user belongs to."

)

language = models.CharField(

max\_length=10,

default='en',

help\_text="Preferred language of the user."

)

timezone = models.CharField(

max\_length=50,

default='UTC',

help\_text="Preferred timezone of the user."

)

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."

)

# Adding utility fields for better management

is\_active = models.BooleanField(

default=True,

help\_text="Indicates whether this user account is active."

)

date\_joined = models.DateTimeField(

auto\_now\_add=True,

help\_text="The date and time the user joined the system."

)

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

return f"{self.username} ({self.role})"

class Meta:

verbose\_name = "User"

verbose\_name\_plural = "Users"

ordering = ["-date\_joined"]

* 1. /Users/harikrishnan/LMS\_Project/users/tests.py

from django.test import TestCase

# Create your tests here.

* 1. /Users/harikrishnan/LMS\_Project/users/urls.py

from django.urls import path

from . import views

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

urlpatterns = [

# User authentication routes

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

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

path('logout/', auth\_views.LogoutView.as\_view(next\_page='/users/login/'), name='logout'),

# Role-based redirection

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

# User-related views

path('', views.user\_list, name='user\_list'),

# Dashboard views for each role

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

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

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

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

]

* 1. /Users/harikrishnan/LMS\_Project/users/views.py

from django.shortcuts import render, redirect

from .forms import CustomUserCreationForm

from django.http import HttpResponse, HttpResponseForbidden

from users.models import CustomUser, Branch

from courses.models import Course, SCORM

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

import logging

# Set up logging

logger = logging.getLogger(\_\_name\_\_)

# Registration view

def register(request):

if request.method == 'POST':

form = CustomUserCreationForm(request.POST)

if form.is\_valid():

form.save()

return redirect('login')

else:

form = CustomUserCreationForm()

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

# SuperAdmin dashboard view

@login\_required

def super\_admin\_dashboard(request):

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

logger.warning(f"Unauthorized access attempt by user {request.user.username} to SuperAdmin dashboard.")

return HttpResponseForbidden("Unauthorized")

total\_users = CustomUser.objects.count()

total\_courses = Course.objects.count()

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

'total\_users': total\_users,

'total\_courses': total\_courses,

})

# Admin dashboard view

@login\_required

def admin\_dashboard(request):

if request.user.role != 'admin':

logger.warning(f"Unauthorized access attempt by user {request.user.username} to Admin dashboard.")

return HttpResponseForbidden("Unauthorized")

admin\_branch = request.user.branch

if not admin\_branch:

logger.error(f"Admin {request.user.username} does not have an assigned branch.")

return HttpResponseForbidden("No branch assigned to this admin.")

# Correctly filter courses and users by branch

branch\_courses = Course.objects.filter(branch=admin\_branch)

total\_users = CustomUser.objects.filter(branch=admin\_branch).count()

total\_courses = branch\_courses.count()

return render(request, 'users/dashboards/admin.html', {

'branch\_name': admin\_branch.name,

'branch\_courses': branch\_courses,

'total\_users': total\_users,

'total\_courses': total\_courses,

})

# Instructor dashboard view

@login\_required

def instructor\_dashboard(request):

if request.user.role != 'instructor':

logger.warning(f"Unauthorized access attempt by user {request.user.username} to Instructor dashboard.")

return HttpResponseForbidden("Unauthorized")

# Fetch courses assigned to the instructor

assigned\_courses = Course.objects.filter(instructor=request.user)

# Fetch learners enrolled in these courses

learners = CustomUser.objects.filter(role='learner', enrolled\_courses\_\_in=assigned\_courses).distinct()

# Prepare SCORM progress for each learner

learner\_scorm\_progress = [

{

"learner": learner,

"scorm\_progress": SCORM.objects.filter(course\_\_in=assigned\_courses, course\_\_enrolled\_users=learner)

}

for learner in learners

]

return render(request, 'users/dashboards/instructor.html', {

'assigned\_courses': assigned\_courses,

'learner\_scorm\_progress': learner\_scorm\_progress,

})

# Learner dashboard view

@login\_required

def learner\_dashboard(request):

if request.user.role != 'learner':

logger.warning(f"Unauthorized access attempt by user {request.user.username} to Learner dashboard.")

return HttpResponseForbidden("Unauthorized")

# Fetch courses the learner is enrolled in

enrolled\_courses = Course.objects.filter(enrolled\_users=request.user)

# Fetch SCORM content for the enrolled courses

scorm\_content = SCORM.objects.filter(course\_\_in=enrolled\_courses)

return render(request, 'users/dashboards/learner.html', {

'enrolled\_courses': enrolled\_courses,

'scorm\_content': scorm\_content,

})

# Role-based redirection view

def role\_based\_redirect(request):

if not request.user.is\_authenticated:

return redirect('login')

role\_redirects = {

'superadmin': 'dashboard\_superadmin',

'admin': 'dashboard\_admin',

'instructor': 'dashboard\_instructor',

'learner': 'dashboard\_learner',

}

redirect\_url = role\_redirects.get(request.user.role, 'login')

logger.info(f"Redirecting user {request.user.username} to {redirect\_url} dashboard.")

return redirect(redirect\_url)

# Home view

def home(request):

# Placeholder home view for the LMS

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

# User list view

@login\_required

def user\_list(request):

users = CustomUser.objects.all()

return render(request, 'users/shared/user\_list.html', {'users': users})

1. SCORM

Structure of SCORM files

* 1. /Users/harikrishnan/LMS\_Project/course\_content/scorm/1
  2. /Users/harikrishnan/LMS\_Project/course\_content/scorm/1/1648213263\_your-personal-development-in-care/scormcontent/index.html
  3. /Users/harikrishnan/LMS\_Project/course\_content/scorm/2