Kn9ght

The below is the folder structure of my virtual classrooom project using django framework

Virtual\_learning (project folder)-|

|-------- |-accounts-|

| |-Templates-|

|-------- |-courses-|

| |-.py files

|-------- |-virtual\_learning (project folder)-|

|----------------|-urls.py-|

|------- --------|-settings.py-|

|-------- |-templates-|

|-------- |-static-|

Virtual\_learning>templates>assignment.html

{% extends 'base.html' %}

{% block content %}

<h2>Upload Assignment for {{ course.title }}</h2>

<form method="post" enctype="multipart/form-data">{% csrf\_token %}

{{ form.as\_p }}

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

</form>

{% endblock %}

Virtual\_learning>templates>base.html

{% load static %}

<!DOCTYPE html>

<html>

<head>

    <title>Virtual Learning</title>

    <link rel="stylesheet" href="{% static 'css/style.css' %}">

</head>

<body>

<header>

    <nav class="navbar">

        <div class="logo">

            <a href="{% url 'home' %}">Virtual Learning</a>

        </div>

        <div class="nav-links">

            {% comment %} {% if user.is\_authenticated %}

                <span class="welcome">Welcome, {{ user.username }}!</span> {% endcomment %}

                <a href="{% url 'dashboard' %}">Dashboard</a>

                <a href="{% url 'about' %}">About us</a>

                {% if user.is\_authenticated %}

                <a href="{% url 'profile' %}">Profile</a>

                <a href="{% url 'logout' %}" class="logout">Logout</a>

                {% else %}

                <a href="{% url 'login' %}" class="logout">Login or Sign up</a>

                {% endif %}

            {% comment %} {% else %}

                <a href="{% url 'login' %}">Login</a>

                <a href="{% url 'register' %}">Signup</a>

            {% endif %} {% endcomment %}

            <!-- Dark mode toggle -->

            <button id="darkModeToggle">🌙</button>

        </div>

    </nav>

</header>

<main>

    {% block content %}{% endblock %}

</main>

<script>

    const toggle = document.getElementById('darkModeToggle');

    const body = document.body;

    // Check for saved mode in localStorage

    if(localStorage.getItem('darkMode') === 'enabled'){

        body.classList.add('dark-mode');

    }

    toggle.addEventListener('click', () => {

        body.classList.toggle('dark-mode');

        if(body.classList.contains('dark-mode')){

            localStorage.setItem('darkMode','enabled');

        } else {

            localStorage.setItem('darkMode','disabled');

        }

    });

</script>

</body>

</html>

Virtual\_learning>templates>chat.html

{% extends 'base.html' %}

{% block content %}

<h2>Course Chat</h2>

<div id="chat-log" style="border:1px solid #ccc; padding:10px; height:300px; overflow-y:scroll;"></div>

<input id="chat-message-input" type="text" size="100">

<button id="chat-message-send">Send</button>

<script>

const courseId = "{{ course.id }}";

const chatSocket = new WebSocket(

    'ws://' + window.location.host +

    '/ws/chat/' + courseId + '/'

);

chatSocket.onmessage = function(e) {

    const data = JSON.parse(e.data);

    const chatLog = document.getElementById('chat-log');

    chatLog.innerHTML += '<b>' + data.username + ':</b> ' + data.message + '<br>';

    chatLog.scrollTop = chatLog.scrollHeight;

};

document.getElementById('chat-message-send').onclick = function() {

    const input = document.getElementById('chat-message-input');

    chatSocket.send(JSON.stringify({

        'message': input.value

    }));

    input.value = '';

};

</script>

{% endblock %}

Virtual\_learning>templates>course\_detail.html

<h2>{{ course.title }}</h2>

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

<p><b>Teacher:</b> {{ course.teacher.username }}</p>

{% if user in course.students.all %}

  <p>You are enrolled in this course ✅</p>

{% else %}

  <form method="post" action="{% url 'enroll\_course' course.id %}">

    {% csrf\_token %}

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

  </form>

{% endif %}

<a href="{% url 'course\_list' %}">Back to Courses</a>

Virtual\_learning>templates>course\_list.html

<h2>All Courses</h2>

<ul>

  {% for course in courses %}

    <li>

      <a href="{% url 'course\_detail' course.id %}">{{ course.title }}</a>

      (Teacher: {{ course.teacher.username }})

    </li>

  {% endfor %}

</ul>

{% if user.is\_authenticated %}

  <a href="{% url 'create\_course' %}">Create New Course</a>

{% endif %}

Virtual\_learning>templates>create\_course.html

<h2>Create a Course</h2>

<form method="post">

  {% csrf\_token %}

  {{ form.as\_p }}

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

</form>

<a href="{% url 'course\_list' %}">Back</a>

Virtual\_learning>templates>home.html

{% extends 'base.html' %}

{% block content %}

  {% if user.is\_authenticated %}

  <span class="welcome"><h1>Welcome, {{ user.username }}!</h1></span>

  {% else %}

  <h1>Welcome to my professional platform for virtual classroom</h1>

  <h5>To join us</h5> <a href="{% url 'login' %}">Login</a> or <a href="{% url 'register' %}">Signup</a>

  {% endif %}

  <div>

    <button><a href="{% url 'course\_list' %}">Browse Courses</a></button>

  </div>

{% endblock %}

Virtually\_learning>templates>logged\_out.html

{% extends 'base.html' %}

{% block content %}

<h2>You have successfully logged out.</h2>

<a href="{% url 'login' %}">Login again</a>

{% endblock %}

Virtual\_learning>templates>teacher\_dashboard.html

{% extends 'base.html' %}

{% block content %}

<h2>Teacher Dashboard</h2>

<a href="{% url 'create\_course' %}">Create New Course</a>

<ul>

{% for course in courses %}

    <li class="course-card">{{ course.title }} - <a href="{% url 'course\_detail' course.pk %}">View</a></li>

{% empty %}

    <li>No courses created yet.</li>

{% endfor %}

</ul>

{% endblock %}

settings.py

"""

Django settings for virtual\_learning project.

Generated by 'django-admin startproject' using Django 3.2.25.

For more information on this file, see

https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.2/ref/settings/

"""

import os

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/3.2/howto/deployment/checklist/

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

SECRET\_KEY = 'django-insecure-cn$u@fe$m-8w(cj!@xwz0z&unet$pbf3qq07aic3+t^d5^kytd'

# 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',

    'channels',

    'accounts',

    '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 = 'virtual\_learning.urls'

TEMPLATES = [

    {

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

        'DIRS': [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',

            ],

        },

    },

]

WSGI\_APPLICATION = 'virtual\_learning.wsgi.application'

ASGI\_APPLICATION = 'virtual\_learning.asgi.application'

# Database

# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {

    'default': {

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

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

    }

}

# Password validation

# https://docs.djangoproject.com/en/3.2/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/3.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC\_URL = '/static/'

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

# Default primary key field type

# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

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

LOGIN\_REDIRECT\_URL = 'dashboard'

LOGOUT\_REDIRECT\_URL = 'login'

# AUTH\_USER\_MODEL = 'accounts.User'

CHANNEL\_LAYERS = {

    "default": {

        "BACKEND": "channels\_redis.core.RedisChannelLayer",

        "CONFIG": {

            "hosts": [("127.0.0.1", 6379)],

        },

    },

}

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

Virtual\_learning>urls.py

from django.contrib import admin

from django.urls import path, include

from django.views.generic import TemplateView

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', TemplateView.as\_view(template\_name='home.html'), name='home'),

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

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

]

from django.conf import settings

from django.conf.urls.static import static

if settings.DEBUG:

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

virtual\_learning>courses>views.py

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

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

from .models import Course, Assignment

from .forms import AssignmentForm

from accounts.models import User

@login\_required

def dashboard\_view(request):

    # Redirect based on user type

    if request.user.is\_teacher:

        return redirect('teacher\_dashboard')

    return render(request, 'dashboard.html')

@login\_required

def course\_list(request):

    courses = Course.objects.all()

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

@login\_required

def course\_detail(request, pk):

    course = get\_object\_or\_404(Course, pk=pk)

    assignments = Assignment.objects.filter(course=course)

    return render(request, 'course\_detail.html', {'course': course, 'assignments': assignments})

@login\_required

def upload\_assignment(request, pk):

    course = get\_object\_or\_404(Course, pk=pk)

    if request.method == 'POST':

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

        if form.is\_valid():

            assignment = form.save(commit=False)

            assignment.student = request.user

            assignment.course = course

            assignment.save()

            return redirect('course\_detail', pk=course.pk)

    else:

        form = AssignmentForm()

    return render(request, 'assignment\_upload.html', {'form': form, 'course': course})

# Teacher views

@login\_required

def teacher\_dashboard(request):

    if not request.user.is\_teacher:

        return redirect('dashboard')

    courses = Course.objects.filter(teacher=request.user)

    return render(request, 'teacher\_dashboard.html', {'courses': courses})

@login\_required

def create\_course(request):

    if not request.user.is\_teacher:

        return redirect('dashboard')

    if request.method == 'POST':

        title = request.POST.get('title')

        description = request.POST.get('description')

        video\_url = request.POST.get('video\_url')

        Course.objects.create(title=title, description=description, teacher=request.user, video\_url=video\_url)

        return redirect('teacher\_dashboard')

    return render(request, 'create\_course.html')

virtual\_learning>courses>urls.py

from django.urls import path

from .views import course\_list, course\_detail, upload\_assignment, teacher\_dashboard, create\_course

urlpatterns = [

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

    path('<int:pk>/', course\_detail, name='course\_detail'),

    path('<int:pk>/upload/', upload\_assignment, name='upload\_assignment'),

    path('teacher/', teacher\_dashboard, name='teacher\_dashboard'),

    path('teacher/create/', create\_course, name='create\_course'),

]

Virtual\_learning>courses>forms.py

from django import forms

from .models import Assignment

class AssignmentForm(forms.ModelForm):

    class Meta:

        model = Assignment

        fields = ['file']

from .models import Course

class CourseForm(forms.ModelForm):

    class Meta:

        model = Course

        fields = ["title", "description"]

virtual\_learning>courses>admin.py

from django.contrib import admin

from .models import Course, Assignment

class CourseAdmin(admin.ModelAdmin):

    list\_display = ('title', 'teacher')

    search\_fields = ('title', 'teacher\_\_username')

    list\_filter = ('teacher',)

class AssignmentAdmin(admin.ModelAdmin):

    list\_display = ('course', 'student', 'submitted\_at')

    search\_fields = ('course\_\_title', 'student\_\_username')

    list\_filter = ('course',)

admin.site.register(Course, CourseAdmin)

admin.site.register(Assignment, AssignmentAdmin)

virtual\_learning>courses>models.py

from django.db import models

from accounts.models import User

class Course(models.Model):

    title = models.CharField(max\_length=200)

    description = models.TextField()

    teacher = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name="courses\_taught")

    students = models.ManyToManyField(User, related\_name="courses\_enrolled", blank=True)

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

    video\_url = models.URLField(blank=True, null=True)

    def \_\_str\_\_(self):

        return self.title

class Assignment(models.Model):

    course = models.ForeignKey(Course, on\_delete=models.CASCADE)

    student = models.ForeignKey(User, on\_delete=models.CASCADE, limit\_choices\_to={'is\_student': True})

    file = models.FileField(upload\_to='assignments/')

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

virtual\_learning>accounts>views.py

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login as auth\_login, logout as auth\_logout

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

from django.contrib.auth.models import User

from django.contrib import messages

def register(request):

    if request.method == 'POST':

        username = request.POST['username']

        email = request.POST['email']

        password = request.POST['password']

        password2 = request.POST['password2']

        if password == password2:

            if User.objects.filter(email=email).exists():

                messages.info(request, "Email Already Used")

                return redirect('register')

            elif User.objects.filter(username=username).exists():

                messages.info(request, 'Username Already Used')

                return redirect('register')

            else:

                user = User.objects.create\_user(username=username, email=email, password=password)

                user.save()

                messages.success(request, "Account created successfully. Please log in.")

                return redirect('login')

        else:

            messages.info(request, "Passwords do not match")

            return redirect('register')

    return render(request, 'register.html')

def login\_view(request):

    if request.method == 'POST':

        username = request.POST['username']

        password = request.POST['password']

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

        if user is not None:

            auth\_login(request, user)

            return redirect('dashboard')   # 👈 redirect to dashboard

        else:

            messages.info(request, 'Invalid username or password')

            return redirect('login')

    return render(request, 'login.html')

@login\_required

def dashboard(request):

    return render(request, 'dashboard.html')

@login\_required

def logout\_view(request):

    auth\_logout(request)

    return redirect('login')   # 👈 send them back to login

def about(request):

    return render(request, 'about.html')

from .forms import ProfileForm

from .models import Profile

@login\_required

def profile\_view(request):

    profile = request.user.profile

    return render(request, 'profile.html', {'profile': profile})

@login\_required

def edit\_profile(request):

    profile = request.user.profile

    if request.method == 'POST':

        form = ProfileForm(request.POST, request.FILES, instance=profile)

        if form.is\_valid():

            form.save()

            return redirect('profile')

    else:

        form = ProfileForm(instance=profile)

    return render(request, 'edit\_profile.html', {'form': form})

@login\_required

def profile(request):

    return render(request, 'accounts/profile.html', {"user": request.user})

virtual\_learning>accounts>urls.py

from django.urls import path

from . import views

urlpatterns = [

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

    path('login/', views.login\_view, name='login'),   # renamed

    path('logout/', views.logout\_view, name='logout'),  # renamed

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

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

    path('profile/', views.profile\_view, name='profile'),

    path('profile/edit/', views.edit\_profile, name='edit\_profile'),

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

    path('profile/edit/', views.edit\_profile, name='edit\_profile'),  # 👈 new

]

Virtual\_learning>accounts>models.py

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

from django.db import models

from django.db.models.signals import post\_save

from django.dispatch import receiver

class Profile(models.Model):

    user = models.OneToOneField(User, on\_delete=models.CASCADE)

    bio = models.TextField(blank=True, null=True)

    profile\_pic = models.ImageField(upload\_to='profile\_pics/', default='profile\_pics/default.png')

    role = models.CharField(max\_length=20, choices=[('student', 'Student'), ('teacher', 'Teacher')], default='student')

    def \_\_str\_\_(self):

        return f"{self.user.username} Profile"

@receiver(post\_save, sender=User)

def create\_profile(sender, instance, created, \*\*kwargs):

    if created:

        Profile.objects.create(user=instance)

@receiver(post\_save, sender=User)

def save\_profile(sender, instance, \*\*kwargs):

    instance.profile.save()

class User(AbstractUser):

    is\_student = models.BooleanField(default=False)

    is\_teacher = models.BooleanField(default=False)

    # Fix related\_name conflicts

    groups = models.ManyToManyField(

        Group,

        related\_name='custom\_user\_set',  # Change this to avoid clashes

        blank=True,

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

        verbose\_name='groups',

    )

    user\_permissions = models.ManyToManyField(

        Permission,

        related\_name='custom\_user\_set\_permissions',  # Change this

        blank=True,

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

        verbose\_name='user permissions',

    )

Virtual\_learning>accounts>forms.py

from django import forms

from django.contrib.auth.forms import UserCreationForm

from .models import User

class SignUpForm(UserCreationForm):

    class Meta:

        model = User

        fields = ['username', 'email', 'password1', 'password2', 'is\_student', 'is\_teacher']

from .models import Profile

class ProfileForm(forms.ModelForm):

    class Meta:

        model = Profile

        fields = ['bio', 'profile\_pic', 'role']

virtual\_learning>accounts>admin.py

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from .models import User

class CustomUserAdmin(UserAdmin):

    model = User

    list\_display = ('username', 'email', 'is\_student', 'is\_teacher', 'is\_staff')

    list\_filter = ('is\_student', 'is\_teacher', 'is\_staff', 'is\_superuser')

    fieldsets = (

        (None, {'fields': ('username', 'email', 'password')}),

        ('Permissions', {'fields': ('is\_active', 'is\_staff', 'is\_superuser', 'groups', 'user\_permissions')}),

        ('User Type', {'fields': ('is\_student', 'is\_teacher')}),

    )

    add\_fieldsets = (

        (None, {

            'classes': ('wide',),

            'fields': ('username', 'email', 'password1', 'password2', 'is\_student', 'is\_teacher', 'is\_staff', 'is\_superuser', 'groups', 'user\_permissions')}

        ),

    )

    search\_fields = ('username', 'email')

    ordering = ('username',)

admin.site.register(User, CustomUserAdmin)

accounts>templates>dashboard.html

{% extends 'base.html' %}

{% block content %}

<h2>Dashboard</h2>

<p>Hello, {{ user.username }}</p>

{% if user.is\_teacher %}

    <p>You can create and manage courses.</p>

{% else %}

    <p>You can enroll and submit assignments.</p>

{% endif %}

{% endblock %}

Accounts>templates>edit\_profile.html

{% extends 'base.html' %}

{% block content %}

  <h2>Edit Profile</h2>

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

    {% csrf\_token %}

    {{ form.as\_p }}

    <button type="submit">Save Changes</button>

  </form>

  <a href="{% url 'profile' %}">Back to Profile</a>

{% endblock %}

Accounts>templates>login.html

{% extends 'base.html' %}

{% block content %}

  <h2>Login Now</h2>

  {% if messages %}

    {% for message in messages %}

      <p style="color: red;">{{ message }}</p>

    {% endfor %}

  {% endif %}

  <form method="POST" action="{% url 'login' %}">

    {% csrf\_token %}

    <p>Username</p>

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

    <p>Password</p>

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

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

  </form>

  <p>Don't have an account? <a href="{% url 'register' %}">Register here</a></p>

{% endblock %}

Accounts>templates>profile.html

{% extends 'base.html' %}

{% block content %}

  <h2>My Profile</h2>

  <p><strong>Username:</strong> {{ user.username }}</p>

  <p><strong>Email:</strong> {{ user.email }}</p>

  <p><strong>First Name:</strong> {{ user.first\_name }}</p>

  <p><strong>Last Name:</strong> {{ user.last\_name }}</p>

  {% if user.profile %}

    <p><strong>Bio:</strong> {{ user.profile.bio }}</p>

    {% if user.profile.profile\_pic %}

      <img src="{{ user.profile.profile\_pic.url }}" alt="Profile Picture" width="150">

    {% endif %}

  {% endif %}

  <br>

  <a href="{% url 'edit\_profile' %}">Edit Profile</a>

{% endblock %}

Accounts>templates>register,html

{% extends 'base.html' %}

{% block content %}

  <h2>Sign Up Below</h2>

  {% for messages in mesages %}

    <h5>{{mesages}}</h5>

  {% endfor %}

  <form method="post" action="register">

    {% csrf\_token %}

    <p>Username</p>

    <input type="text" name="username">

    <p>Email</p>

    <input type="text" name="email">

    <p>Password</p>

    <input type="text" name="password">

    <p>Repeat Password</p>

    <input type="text" name="password2"><br>

    <button type="submit">Sign Up</button>

  </form>

{% endblock %}

My codes above has all sort of errors give me a fully corrected code of the above project, and include the following:

1. The navigation bar that contains the menus of the project above should have the following responsive menu:
   1. Home
   2. Courses (for students)
   3. Create courses (for lecturers)
   4. Go live (for lecturers)
   5. About
   6. Blog
   7. Resources
   8. And a button that displays login when a user is logged in, and log out when a user is not logged in
2. Immediately a user uses my site through the link, the first thing I want the user to see is a large general welcome messages and below the message there will be a button named “Get started” (which is login required and leads to the login page)
3. Let the login and sign up page have a standard have a perfect user authentication feature, that will check if the username, teacher’s/student id and is the same (while creating an account), and the login form should require the username or id and the correct password. (error messages should be styled in red)
4. If user inputs correct login details, they should be led to the dashboard page with a greeting welcome user (the name of the user).
5. The lecturers should be able to upload resources for students, and students should be able to access and download the resources.
6. The blog page should be a page where the users will receive notifications, such as; lecture reminders with duration, assignments and e-tests, and there should also be a page for the lecturers to mark each tests and students too should be able to see their marks and performances in each courses.
7. There should be a live feature (go live) that will make the lecturers be seen live, and a notification should be sent to each students offering the course to notify them about the live class with a button at the end of the message the button name for the student will be attend class (only for live classes).
8. The lecturers should be able to see the names and numbers of the students watching the live class. (immediately a student minimizes the site he or she should be tagged offline, also if a student didn’t touch the mouse for at least 20 minutes he should be tagged offline)
9. Also there should be a comment section, which will display the name of the send, profile picture and comment.
10. There should be an editable page for the users to be able to edit and delete their profile
11. Lecturers should be able to delete courses after each class