I want to make ProjectManagementSystem using django. This project contains this feature. In short, here are two kinds of user. Teacher who can (edit, modify, delete, approve, suggest idea) of any project. And student who can (register, login, submit their project as pdf/rar/zip, chat with admin to get suggest idea, search project by name/short name, download other student’s project). Some advance feature along with some common feature( authentication ) also be there.

Can you assist me on that? I create Backend\_Management\_App for manage the backend. Can you generate the models.py

from django.db import models

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):

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

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

class Project(models.Model):

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

short\_name = models.CharField(max\_length=50, unique=True)

description = models.TextField()

file = models.FileField(upload\_to='projects')

teacher = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='projects', limit\_choices\_to={'is\_teacher': True})

students = models.ManyToManyField(User, related\_name='student\_projects', blank=True)

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

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

class Idea(models.Model):

project = models.ForeignKey(Project, on\_delete=models.CASCADE, related\_name='ideas')

teacher = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='ideas')

description = models.TextField()

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

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

class Message(models.Model):

sender = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='sent\_messages')

receiver = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='received\_messages')

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

body = models.TextField()

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

class Comment(models.Model):

project = models.ForeignKey(Project, on\_delete=models.CASCADE, related\_name='comments')

author = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='comments')

text = models.TextField()

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

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

This models.py file includes the following models:

* User: This model extends Django's built-in AbstractUser model to add the is\_teacher and is\_student boolean fields. It will be used for authentication and authorization purposes.
* Project: This model represents a project that can be submitted by students and managed by teachers. It includes fields for the project name, short name, description, uploaded file, teacher (who created the project), students (who can access the project), and timestamps.
* Idea: This model represents an idea that a teacher can suggest for a project. It includes fields for the related project, teacher, description, and timestamps.
* Message: This model represents a message sent between users (e.g., a student asking a teacher for an idea suggestion). It includes fields for the sender, receiver, subject, body, and timestamp.
* Comment: This model represents a comment made on a project. It includes fields for the related project, author, text, and timestamps.

I hope this helps! Let me know if you have any questions or need further assistance.

Views.py code are here

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

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

from django.contrib import messages

from django.http import HttpResponse

from django.core.exceptions import PermissionDenied

from .models import Project, Idea, Message, Comment

from .forms import ProjectForm, IdeaForm, MessageForm, CommentForm

def index(request):

projects = Project.objects.all()

context = {'projects': projects}

return render(request, 'index.html', context)

@login\_required

def submit\_project(request):

if request.method == 'POST':

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

if form.is\_valid():

project = form.save(commit=False)

project.teacher = request.user

project.save()

form.save\_m2m()

messages.success(request, 'Your project has been submitted.')

return redirect('index')

else:

form = ProjectForm()

context = {'form': form}

return render(request, 'submit\_project.html', context)

@login\_required

def edit\_project(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name, teacher=request.user)

if request.method == 'POST':

form = ProjectForm(request.POST, request.FILES, instance=project)

if form.is\_valid():

form.save()

messages.success(request, 'Your project has been updated.')

return redirect('project\_detail', short\_name=project.short\_name)

else:

form = ProjectForm(instance=project)

context = {'form': form, 'project': project}

return render(request, 'edit\_project.html', context)

@login\_required

def delete\_project(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name, teacher=request.user)

if request.method == 'POST':

project.delete()

messages.success(request, 'Your project has been deleted.')

return redirect('index')

context = {'project': project}

return render(request, 'delete\_project.html', context)

@login\_required

def approve\_project(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name, teacher=request.user)

if project.students.count() == 0:

raise PermissionDenied('This project has no students.')

project.approved = True

project.save()

messages.success(request, 'The project has been approved.')

return redirect('project\_detail', short\_name=project.short\_name)

@login\_required

def suggest\_idea(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name)

if request.method == 'POST':

form = IdeaForm(request.POST)

if form.is\_valid():

idea = form.save(commit=False)

idea.project = project

idea.teacher = request.user

idea.save()

messages.success(request, 'Your idea has been submitted.')

return redirect('project\_detail', short\_name=project.short\_name)

else:

form = IdeaForm()

context = {'form': form, 'project': project}

return render(request, 'suggest\_idea.html', context)

@login\_required

def send\_message(request, username):

recipient = get\_object\_or\_404(User, username=username)

if request.method == 'POST':

form = MessageForm(request.POST)

if form.is\_valid():

message = form.save(commit=False)

message.sender = request.user

message.receiver = recipient

message.save()

messages.success(request, 'Your message has been sent.')

return redirect('inbox')

else:

form = MessageForm()

context = {'form': form 'project': project}

return render(request, ' send\_message.html', context)

@login\_required

def inbox(request):

messages = Message.objects.filter(receiver=request.user)

context = {'messages': messages}

return render(request, 'inbox.html', context)

@login\_required

def sent(request):

messages = Message.objects.filter(sender=request.user)

context = {'messages': messages}

return render(request, 'sent.html', context)

@login\_required

def project\_detail(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name)

if not project.approved and project.teacher != request.user and project.students.filter(id=request.user.id).count() == 0:

raise PermissionDenied('This project has not been approved yet.')

if request.method == 'POST':

form = CommentForm(request.POST)

if form.is\_valid():

comment = form.save(commit=False)

comment.project = project

comment.author = request.user

comment.save()

messages.success(request, 'Your comment has been posted.')

return redirect('project\_detail', short\_name=project.short\_name)

else:

form = CommentForm()

comments = Comment.objects.filter(project=project)

context = {'project': project, 'form': form, 'comments': comments}

return render(request, 'project\_detail.html', context)

@login\_required

def search\_project(request):

query = request.GET.get('q')

if query:

projects = Project.objects.filter(title\_\_icontains=query)

else:

projects = Project.objects.all()

context = {'projects': projects, 'query': query}

return render(request, 'search\_project.html', context)

@login\_required

def download\_project(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name)

if not project.approved and project.teacher != request.user and project.students.filter(id=request.user.id).count() == 0:

raise PermissionDenied('This project has not been approved yet.')

file\_path = project.file.path

with open(file\_path, 'rb') as fh:

response = HttpResponse(fh.read(), content\_type='application/force-download')

response['Content-Disposition'] = 'attachment; filename=' + os.path.basename(file\_path)

return response

raise Http404

@login\_required

def register\_project(request, short\_name):

project = get\_object\_or\_404(Project, short\_name=short\_name)

if project.students.filter(id=request.user.id).count() == 1:

messages.warning(request, 'You have already registered for this project.')

return redirect('project\_detail', short\_name=project.short\_name)

if request.method == 'POST':

project.students.add(request.user)

messages.success(request, 'You have successfully registered for this project.')

return redirect('project\_detail', short\_name=project.short\_name)

context = {'project': project}

return render(request, 'register\_project.html', context)

@login\_required

def create\_project(request):

    if request.method == 'POST':

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

        if form.is\_valid():

            project = form.save(commit=False)

            project.teacher = request.user

            project.save()

            messages.success(request, 'Your project has been created.')

            return redirect('project\_detail', short\_name=project.short\_name)

    else:

        form = ProjectForm()

    context = {'form': form}

    return render(request, 'create\_project.html', context)

forms.py code

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from .models import Project, Message, Comment

class ProjectForm(forms.ModelForm):

class Meta:

model = Project

fields = ['title', 'short\_name', 'description', 'file', 'teacher']

widgets = {

'description': forms.Textarea(attrs={'rows': 5}),

}

class UserRegisterForm(UserCreationForm):

email = forms.EmailField(required=True)

class Meta:

model = User

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

class MessageForm(forms.ModelForm):

class Meta:

model = Message

fields = ['receiver', 'subject', 'body']

widgets = {

'body': forms.Textarea(attrs={'rows': 5}),

}

class CommentForm(forms.ModelForm):

class Meta:

model = Comment

fields = ['text']

widgets = {

'text': forms.Textarea(attrs={'rows': 3}),

}