I am assuming:

* You have install Django
* Postgres and PG Admin are installed

Models

* Projects – will have Project model
* Users – will have CustomUser model – which will inherit from Djangos User mod
* Inbox – will have Message model

Relationships

* CustomUser will inherit from User
* Each Project will belong to one User – so Project will have a user id as a foreign key.
* Each Message will have a sender and a receiver which will both be users. So Message will have a sender\_id and a receiver\_id – which will be user IDs as foreign keys for each of these.

**Phase 1: Setting Up the Django Environment**

**1. Start by creating the project and apps**

Assuming you already have Django installed, create your project and apps as follows:

# Create the Django project

django-admin startproject assignment .

# Create the apps

python manage.py startapp projects

python manage.py startapp users

python manage.py startapp inbox

At this point, your folder structure should look like this:

assignment/

\_\_init\_\_.py

asgi.py

settings.py

urls.py

wsgi.py

projects/

\_\_init\_\_.py

admin.py

apps.py

models.py

views.py

...

users/

...

inbox/

...

manage.py

**2. Register the apps in settings.py**

Open settings.py and add your apps to the INSTALLED\_APPS list:

INSTALLED\_APPS = [

...

'projects',

'users',

'inbox',

]

**3. Set up PostgreSQL as the database**

In settings.py, update the DATABASES configuration to use PostgreSQL:

DATABASES = {

'default': {

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

'NAME': 'projects\_db',

'USER': 'staff', # Replace with your PostgreSQL user

'PASSWORD': '', # Add the password if required

'HOST': 'localhost',

'PORT': '5432',

}

}

**4. Migrate the default database tables**

Run the following commands to initialise the database with Django's default tables:

python manage.py makemigrations

python manage.py migrate

This will create the necessary tables for user authentication, sessions, and other built-in features.

**Phase 2: Designing the Database Schema**

**Plan your models:**

* **Projects App**:
  + Model for storing project details: name, description, start and end dates, stakeholders, and status.
* **Users App**:
  + Extends the default Django user model to include additional fields (e.g., phone, address).
* **Inbox App**:
  + Models for messages: sender, recipient, subject, body, and timestamp.

**1. Define Models in Each App**

**projects/models.py**

from django.db import models

class Project(models.Model):

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

description = models.TextField()

start\_date = models.DateField()

end\_date = models.DateField()

stakeholders = models.TextField() # Can be a JSON field for multiple stakeholders

status = models.CharField(

max\_length=20,

choices=[('Pending', 'Pending'), ('Ongoing', 'Ongoing'), ('Completed', 'Completed')],

default='Pending'

)

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

return self.name

**users/models.py**

from django.contrib.auth.models import AbstractUser

from django.db import models

class CustomUser(AbstractUser):

phone = models.CharField(max\_length=15, blank=True, null=True)

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

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

return self.username

You also need to tell Django to use this custom user model. Update settings.py:

AUTH\_USER\_MODEL = 'users.CustomUser'

**inbox/models.py**

from django.db import models

from users.models import CustomUser

class Message(models.Model):

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

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

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

body = models.TextField()

timestamp = models.DateTimeField(auto\_now\_add=True)

archived = models.BooleanField(default=False)

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

return f'{self.subject} - {self.sender} to {self.recipient}'

**2. Make and Apply Migrations**

Run the following commands to create migration files and apply them to the database:

python manage.py makemigrations

python manage.py migrate

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WE FINISHED HERE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Phase 3: Building Core Backend Functionalities**

**1. Register Models in the Admin Site**

To manage data easily, register your models in the admin panel.

**Projects/admin.py**

from django.contrib import admin

from .models import Project

admin.site.register(Project)

**users/admin.py**

from django.contrib import admin

from .models import CustomUser

admin.site.register(CustomUser)

**inbox/admin.py**

from django.contrib import admin

from .models import Message

admin.site.register(Message)

**create a superuser with this command**

python manage.py createsuperuser

keep the user name and password as admin – so it’s easy to remember,

Now, run the development server:

python manage.py runserver

Visit <http://127.0.0.1:8000/admin>, and log in using the superuser credentials.

A screenshot of a login form

AI-generated content may be incorrect.

**Phase 4: Crafting the Frontend**

**Set Up Bootstrap**: Install django-bootstrap4:

1. pip install django-bootstrap4

Add it to INSTALLED\_APPS: (settings.py)

INSTALLED\_APPS += ['bootstrap4']

1. **Create HTML Templates**:
   * Create a templates folder in your project root.
   * Inside each app, create specific templates (e.g., projects/templates/projects/project\_list.html).

Pop a bit of text in each html page – so we can test.

1. **Set Up URLs**: Create urls.py in each app (e.g., projects/urls.py) and link to views. For example:

**projects/urls.py**

from django.urls import path

from . import views

urlpatterns = [

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

]

Link these URLs to the main urls.py in the webapp\_project folder.

**webapp\_project/urls.py**

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

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

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

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

]

1. **Write Views**: For example, a simple view for listing projects:

**projects/views.py**

from django.shortcuts import render

from .models import Project

def project\_list(request):

projects = Project.objects.all()

return render(request, 'projects/project\_list.html', {'projects': projects})

**For the next few classes….**

1. Add forms and authentication for users.
2. Implement CRUD functionality for projects and messages.
3. Test everything thoroughly.