TASK – 3

TASK 3: Project Management Tool

Build a collaborative tool similar to Trello or Asana.

Users should be able to:

● Create group projects

● Assign tasks

● Comment and communicate within tasks

Full stack with auth system, project boards, task cards.

Backend to manage users, projects, tasks, comments.

Bonus: Add notifications and real-time updates using WebSockets.

FRONTEND : INDEX.HTML

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Project Management Dashboard</title>


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

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">


<style>

* { margin:0; padding:0; box-sizing:border-box; font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; }
```

```
body { background:#ffffff; color:#1e293b; }  /* White background */
a { text-decoration:none; }

/* ---------- Navbar ---------- */
.navbar {
  display:flex; justify-content:space-between; align-items:center;
  padding:1rem 2rem; background:#ffe4e6;   /* Light pink navbar */
  box-shadow:0 4px 10px rgba(0,0,0,0.05); color:#1e293b;
}
.navbar .logo { font-size:1.6rem; font-weight:bold; display:flex; align-items:center; gap:10px; }
.navbar ul { display:flex; gap:2rem; list-style:none; }
.navbar ul li a { color:#1e293b; display:flex; gap:6px; transition: all 0.3s ease; font-weight:500; }
.navbar ul li a:hover { color:#ec4899; transform:scale(1.05); }  /* Pink hover */

/* ---------- Header ---------- */
header { text-align:center; padding:3rem 1rem; }
header h1 { font-size:2.2rem; color:#1e293b; }
header span { color:#ec4899; font-weight:bold; }
header p { color:#475569; margin-top:0.5rem; font-size:1.05rem; }

/* ---------- Container ---------- */
.container {
  display:grid;
  grid-template-columns: repeat(3, 1fr);
  gap:1.5rem;
  padding:2rem; max-width:1200px; margin:auto;
  min-height:50vh;
```

```
}

/* ---------- Card Styles ---------- */
.card {
  border-radius:16px; padding:2rem;
  min-height:180px; display:flex; flex-direction:column; justify-content:center; text-align:center;
  color:#1e293b; position:relative; overflow:hidden;
  transition: all 0.3s ease; cursor:pointer;
  box-shadow:0 4px 12px rgba(0,0,0,0.05);
  background:#ffe4e6;   /* Light pink cards */
}
.card i { font-size:2.3rem; margin-bottom:1rem; transition: transform 0.3s; color:#ec4899; }
.card h3 { font-size:1.2rem; margin-bottom:0.4rem; }
.card p { font-size:0.9rem; opacity:0.95; }
.card:hover { transform:translateY(-6px); background:#ffc1d0; }  /* Slightly darker pink on hover */
.card:hover i { transform: scale(1.15); }

/* ---------- Footer ---------- */
footer { background:#ffe4e6; color:#1e293b; text-align:center; padding:1.5rem; margin-top:2rem; font-size:0.95rem; }
footer i { color:#ec4899; }
</style>
</head>
<body>

<!-- Navbar -->
<nav class="navbar">
  <div class="logo"><i class="fa-solid fa-diagram-project"></i> ProjectBoard</div>
```

```
  <ul>
    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
    <li><a href="#"><i class="fa fa-user"></i> Profile</a></li>
    <li><a href="#"><i class="fa fa-cog"></i> Settings</a></li>
  </ul>
</nav>

<!-- Header -->
<header>
  <h1>Welcome to <span>Project Management Dashboard</span></h1>
  <p>Organize and track your projects efficiently</p>
</header>

<!-- Dashboard Cards (3x3 neat grid) -->
<div class="container">
  <a href="{% url 'users' %}"><div class="card"><i class="fa fa-users"></i><h3>Users</h3><p>Manage team members</p></div></a>
  <a href="{% url 'projects' %}"><div class="card"><i class="fa fa-diagram-project"></i><h3>Projects</h3><p>View and manage projects</p></div></a>
  <a href="{% url 'tasks' %}"><div class="card"><i class="fa fa-tasks"></i><h3>Tasks</h3><p>Assign and track tasks</p></div></a>
  <a href="{% url 'comments' %}"><div class="card"><i class="fa fa-comments"></i><h3>Comments</h3><p>Team discussions</p></div></a>
  <a href="{% url 'notifications' %}"><div class="card"><i class="fa fa-bell"></i><h3>Notifications</h3><p>Get project updates</p></div></a>
  <a href="{% url 'activities' %}"><div class="card"><i class="fa fa-clock"></i><h3>Activities</h3><p>Track recent actions</p></div></a>
</div>

<!-- Footer -->
<footer>
```

```
  <p>© 2025 ProjectBoard | Built with <i class="fa fa-heart"></i> by Nathiya</p>
</footer>

<script src="{% static 'js/main.js' %}"></script>
</body>
</html>
```

FRONTEND_URLS.PY

```
# frontend/frontend_urls.py
from django.urls import path
from . import frontend_views as views

urlpatterns = [
    path("", views.index, name="index"),  # Home / Dashboard
    path("users/", views.users, name="users"),
    path("projects/", views.projects, name="projects"),
    path("tasks/", views.tasks, name="tasks"),
    path("comments/", views.comments, name="comments"),
    path("notifications/", views.notifications, name="notifications"),
    path("activities/", views.activities, name="activities"),
]
```

FRONTEND_VIEWS.PY

```
# frontend/frontend_views.py
from django.shortcuts import render

def index(request):
    return render(request, "index.html")  # Dashboard / Home
```

```
def users(request):
    return render(request, "users.html")  # Users list or profile

def projects(request):
    return render(request, "projects.html")  # Project boards

def tasks(request):
    return render(request, "tasks.html")  # Task cards

def comments(request):
    return render(request, "comments.html")  # Task comments

def notifications(request):
    return render(request, "notifications.html")  # Notifications page

def activities(request):
    return render(request, "activities.html")  # Activity log
```

BACKEND : DJANGO (MODELS.PY)

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# ---------------- USER ----------------
class User(AbstractUser):
    bio = models.TextField(blank=True, null=True)
    profile_pic = models.ImageField(upload_to='profile_pics/', blank=True, null=True)

    def __str__(self):
        return self.username
```

```
# ---------------- PROJECT ----------------
class Project(models.Model):
    name = models.CharField(max_length=200)
    description = models.TextField(blank=True, null=True)
    members = models.ManyToManyField(User, related_name='projects')
    created_at = models.DateTimeField(auto_now_add=True)
    due_date = models.DateField(blank=True, null=True)

    def __str__(self):
        return self.name

# ---------------- TASK ----------------
class Task(models.Model):
    STATUS_CHOICES = [
        ('TODO', 'To Do'),
        ('IN_PROGRESS', 'In Progress'),
        ('DONE', 'Done'),
    ]

    PRIORITY_CHOICES = [
        ('LOW', 'Low'),
        ('MEDIUM', 'Medium'),
        ('HIGH', 'High'),
    ]

    project = models.ForeignKey(Project, on_delete=models.CASCADE, related_name='tasks')
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True, null=True)
```

```
    assigned_to = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, blank=True, related_name='tasks')
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='TODO')
    priority = models.CharField(max_length=10, choices=PRIORITY_CHOICES, default='MEDIUM')
    due_date = models.DateField(blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

# ---------- COMMENT ----------
class Comment(models.Model):
    task = models.ForeignKey(Task, on_delete=models.CASCADE, related_name='comments')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user.username}: {self.content[:20]}"

# ---------- NOTIFICATION ----------
class Notification(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='notifications')
    content = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)
    read = models.BooleanField(default=False)

    def __str__(self):
```

```
        return f"Notification for {self.user.username}"


# ---------------- ACTIVITY ----------------
class Activity(models.Model):
    project = models.ForeignKey(Project, on_delete=models.CASCADE,
related_name='activities')
    content = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)


    def __str__(self):
        return f"{self.project.name} - {self.content[:20]}"
```

ADMIN.PY

```
from django.contrib import admin
from .models import User, Project, Task, Comment, Notification, Activity

admin.site.register(User)
admin.site.register(Project)
admin.site.register(Task)
admin.site.register(Comment)
admin.site.register(Notification)
admin.site.register(Activity)
```

URLS.PY

```
from django.urls import path
from . import views

urlpatterns = [
    # USERS
```

```
    path('users/', views.get_users),
    path('users/add/', views.post_user),

    # PROJECTS
    path('projects/', views.get_projects),
    path('projects/add/', views.post_project),

    # TASKS
    path('tasks/', views.get_tasks),
    path('tasks/add/', views.post_task),

    # COMMENTS
    path('comments/', views.get_comments),
    path('comments/add/', views.post_comment),

    # NOTIFICATIONS
    path('notifications/', views.get_notifications),
    path('notifications/add/', views.post_notification),

    # ACTIVITIES
    path('activities/', views.get_activities),
    path('activities/add/', views.post_activity),
]
```

VIEWS.PY

```
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from django.shortcuts import get_object_or_404
```

```
from .models import User, Project, Task, Comment, Notification, Activity
from .serializers import (
    UserSerializer, ProjectSerializer, TaskSerializer,
    CommentSerializer, NotificationSerializer, ActivitySerializer
)

# ---------------- USERS ----------------
@api_view(['GET'])
def get_users(request):
    users = User.objects.all()
    serializer = UserSerializer(users, many=True)
    return Response(serializer.data)

@api_view(['POST'])
def post_user(request):
    serializer = UserSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

# ---------------- PROJECTS ----------------
@api_view(['GET'])
def get_projects(request):
    projects = Project.objects.all()
    serializer = ProjectSerializer(projects, many=True)
    return Response(serializer.data)

@api_view(['POST'])
```

```
def post_project(request):
    serializer = ProjectSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

# ---------------- TASKS ----------------
@api_view(['GET'])
def get_tasks(request):
    tasks = Task.objects.all()
    serializer = TaskSerializer(tasks, many=True)
    return Response(serializer.data)

@api_view(['POST'])
def post_task(request):
    serializer = TaskSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

# ---------------- COMMENTS ----------------
@api_view(['GET'])
def get_comments(request):
    comments = Comment.objects.all()
    serializer = CommentSerializer(comments, many=True)
    return Response(serializer.data)
```

```
@api_view(['POST'])
def post_comment(request):
    serializer = CommentSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

# ---------- NOTIFICATIONS ----------
@api_view(['GET'])
def get_notifications(request):
    notifications = Notification.objects.all()
    serializer = NotificationSerializer(notifications, many=True)
    return Response(serializer.data)

@api_view(['POST'])
def post_notification(request):
    serializer = NotificationSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

# ---------- ACTIVITIES ----------
@api_view(['GET'])
def get_activities(request):
    activities = Activity.objects.all()
    serializer = ActivitySerializer(activities, many=True)
    return Response(serializer.data)
```

```
@api_view(['POST'])
def post_activity(request):
    serializer = ActivitySerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

SERIALIZERS.PY

```
from rest_framework import serializers
from .models import User, Project, Task, Comment, Notification, Activity


class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = "__all__"


class ProjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Project
        fields = "__all__"


class TaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        fields = "__all__"


class CommentSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = Comment
        fields = "__all__"


class NotificationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Notification
        fields = "__all__"


class ActivitySerializer(serializers.ModelSerializer):
    class Meta:
        model = Activity
        fields = "__all__"
```