|  |
| --- |
| **settings.py:**  INSTALLED\_APPS = [  …      "rest\_framework",      "rest\_framework\_simplejwt",      "taskApi",  ]  REST\_FRAMEWORK = {      'DEFAULT\_AUTHENTICATION\_CLASSES': (          'rest\_framework\_simplejwt.authentication.JWTAuthentication',      )  }  TEMPLATES = [      {          …          "DIRS": ["templates"],          …          },      },  ]  DATABASES = {      "default": {          'ENGINE': 'django.db.backends.mysql',          'NAME': 'task\_db',          'USER': 'root',          'PASSWORD': 'root',          'HOST': 'localhost',          'PORT': '3306',      }  }  STATIC\_URL = "static/"  STATICFILES\_DIRS = [ BASE\_DIR / 'static' ] |

|  |
| --- |
| **models.py:**  from django.db import models  from django.contrib.auth.models import User  # Create your models here.  class UserProfile(models.Model):      user = models.OneToOneField(User, on\_delete=models.CASCADE, related\_name='profile')      def \_\_str\_\_(self):          return self.user.username  class Task(models.Model):      status\_choices=[("Pending", "pending"), ("Completed", "completed")]      title=models.CharField(max\_length=100, blank=False, null=False)      description=models.TextField()      due\_date=models.DateField()      status=models.CharField(max\_length=50, choices=status\_choices,default="pending")      user=models.ForeignKey(User, on\_delete=models.CASCADE)      def \_\_str\_\_(self):          return self.title |

|  |
| --- |
| **serializers.py:**  from rest\_framework import serializers  from .models import Task  from django.contrib.auth.models import User  class RegisterSerializer(serializers.ModelSerializer):      class Meta:          model = User          fields = ('id', 'username', 'password')          extra\_kwargs = {              'password': {'write\_only': True},          }      def create(self, validated\_data):          user = User.objects.create\_user(\*\*validated\_data)          return user  class TaskSerializer(serializers.ModelSerializer):      class Meta:          model=Task          fields="\_\_all\_\_"          read\_only\_fields=["user"] |

|  |
| --- |
| **tokens.py:**  from rest\_framework\_simplejwt.tokens import RefreshToken  def get\_tokens\_for\_user(user):      refresh = RefreshToken.for\_user(user)      refresh['username'] = user.username      refresh['is\_superuser'] = user.is\_superuser      return {          'refresh': str(refresh),          'access': str(refresh.access\_token),      } |

|  |
| --- |
| **views.py:**  from django.shortcuts import render, redirect, get\_object\_or\_404  from django.contrib.auth import authenticate, login, logout  from django.contrib.auth.models import User  from rest\_framework import status, generics  from rest\_framework.response import Response  from rest\_framework.decorators import api\_view, permission\_classes  from rest\_framework.permissions import AllowAny, IsAuthenticated  from rest\_framework.views import APIView  from rest\_framework.generics import (      ListAPIView, RetrieveAPIView, CreateAPIView, UpdateAPIView, DestroyAPIView  )  from .models import Task  from .serializers import RegisterSerializer, TaskSerializer  from .tokens import get\_tokens\_for\_user  # Create your views here.  class RegisterView(generics.CreateAPIView):      queryset = User.objects.all()      serializer\_class = RegisterSerializer      permission\_classes = [AllowAny]    class CustomLoginView(APIView):      permission\_classes = []      def post(self, request):          username = request.data.get('username')          password = request.data.get('password')          user = authenticate(username=username, password=password)          if user is not None:              tokens = get\_tokens\_for\_user(user)              return Response({                  'refresh': tokens['refresh'],                  'access': tokens['access'],                  'username': user.username,              })          else:              return Response({'error': 'Invalid credentials'}, status=status.HTTP\_401\_UNAUTHORIZED)    class TaskListAPIView(ListAPIView):      serializer\_class = TaskSerializer      permission\_classes = [IsAuthenticated]      def get\_queryset(self):          return Task.objects.filter(user=self.request.user)  class TaskRetrieveAPIView(RetrieveAPIView):      serializer\_class = TaskSerializer      def get\_queryset(self):          return Task.objects.filter(user=self.request.user)  class TaskCreateAPIView(CreateAPIView):      serializer\_class = TaskSerializer      permission\_classes = [IsAuthenticated]      def perform\_create(self, serializer):          serializer.save(user=self.request.user)  class TaskUpdateAPIView(UpdateAPIView):      serializer\_class = TaskSerializer      permission\_classes = [IsAuthenticated]      def get\_queryset(self):          return Task.objects.filter(user=self.request.user)  class TaskDestroyAPIView(DestroyAPIView):      serializer\_class = TaskSerializer      permission\_classes = [IsAuthenticated]      def get\_queryset(self):          return Task.objects.filter(user=self.request.user)    def register\_template(request):      if request.method == "POST":          username = request.POST['username']          password = request.POST['password']          User.objects.create\_user(username=username, password=password)          return redirect('login')      return render(request, 'register.html')  def login\_template(request):      if request.method == "POST":          username = request.POST['username']          password = request.POST['password']          user = authenticate(username=username, password=password)          if user:              login(request, user)              return redirect('task-list-template')          return render(request, 'login.html', {'error': 'Invalid credentials'})      return render(request, 'login.html')  def logout\_view(request):      logout(request)      return redirect('login')  @permission\_classes([IsAuthenticated])  def task\_list\_view(request):      # tasks = Task.objects.filter(user=request.user)      return render(request, 'tasks/task\_list.html')  @permission\_classes([IsAuthenticated])  def task\_create\_view(request):      if request.method == 'POST':          title = request.POST.get('title')          description = request.POST.get('description')          Task.objects.create(user=request.user, title=title, description=description)          return redirect('task-list-template')      return render(request, 'tasks/task\_form.html')  @permission\_classes([IsAuthenticated])  def task\_update\_view(request, pk):      task = get\_object\_or\_404(Task, pk=pk, user=request.user)      if request.method == 'POST':          task.title = request.POST.get('title')          task.description = request.POST.get('description')          task.save()          return redirect('task-list-template')      return render(request, 'tasks/task\_form.html', {'task': task})  @permission\_classes([IsAuthenticated])  def task\_delete\_view(request, pk):      task = get\_object\_or\_404(Task, pk=pk, user=request.user)      if request.method == 'POST':          task.delete()          return redirect('task-list-template')      return render(request, 'tasks/task\_confirm\_delete.html', {'task': task}) |

|  |
| --- |
| **backend/urls.py:**  from django.contrib import admin  from django.urls import path, include  from taskApi import views  urlpatterns = [      path("admin/", admin.site.urls),      path("task-api/", include("taskApi.urls"))  ] |

|  |
| --- |
| **taskApi/urls.py**  from . import views  from rest\_framework\_simplejwt.views import (  TokenObtainPairView,  TokenRefreshView  )  from .views import \*  urlpatterns = [      path('token/', TokenObtainPairView.as\_view()),      path('token/refresh/', TokenRefreshView.as\_view()),       # API endpoints for authentication      path('cregister/', RegisterView.as\_view(), name='api-register'),      path('custom-login/', CustomLoginView.as\_view(), name='api-login'),      # API endpoints for Task CRUD      path('api/tasks/', TaskListAPIView.as\_view(), name='api-task-list'),      path('api/tasks/<int:pk>/', TaskRetrieveAPIView.as\_view(), name='api-task-detail'),      path('api/tasks/create/', TaskCreateAPIView.as\_view(), name='api-task-create'),      path('api/tasks/<int:pk>/update/', TaskUpdateAPIView.as\_view(), name='api-task-update'),      path('api/tasks/<int:pk>/delete/', TaskDestroyAPIView.as\_view(), name='api-task-delete'),      # Template URLs for task management      path('c-register/', register\_template, name='register'),      path('login/', login\_template, name='login'),      path('tasks/', task\_list\_view, name='task-list-template'),      path('tasks/create/', task\_create\_view, name='task-create-template'),      path('tasks/<int:pk>/edit/', task\_update\_view, name='task-update-template'),      path('tasks/<int:pk>/delete/', task\_delete\_view, name='task-delete-template'),      path('logout/', views.logout\_view, name='logout'),  ] |

|  |
| --- |
| **base.html:**  {% extends 'base.html' %}  {% block title %}Register{% endblock %}  {% block content %}  <h2>Register</h2>  <form id="registerForm">    <label>Username:</label>    <input type="text" id="username" required><br><br>    <label>Password:</label>    <input type="password" id="password" required><br><br>    <button type="submit">Register</button>  </form>  <p id="error-message" style="color: red;"></p>  <a href="{% url 'login' %}">Login</a>  <script>  const registerForm = document.getElementById('registerForm');  const errorMessage = document.getElementById('error-message');  registerForm.addEventListener('submit', async function(e) {      e.preventDefault();        const username = document.getElementById('username').value.trim();      const password = document.getElementById('password').value.trim();      const data = { username, password };      try {          const res = await fetch('/task-api/cregister/', {              method: 'POST',              headers: { 'Content-Type': 'application/json' },              body: JSON.stringify(data)          });          const result = await res.json();          if (res.ok) {              alert("Registration successful! You can now log in.");              window.location.href = "/task-api/login/";          } else {              // Handle errors              if (result.username) {                  errorMessage.textContent = result.username.join(", ");              } else if (result.password) {                  errorMessage.textContent = result.password.join(", ");              } else if (result.detail) {                  errorMessage.textContent = result.detail;              } else {                  errorMessage.textContent = "Registration failed. Try again.";              }          }      } catch (err) {          console.error(err);          errorMessage.textContent = "An error occurred. Try again.";      }  });  </script>  {% endblock %} |

|  |
| --- |
| **register.html:**  {% extends 'base.html' %}  {% block title %}Register{% endblock %}  {% block content %}  <h2>Register</h2>  <form id="registerForm">    <label>Username:</label>    <input type="text" id="username" required><br><br>    <label>Password:</label>    <input type="password" id="password" required><br><br>    <button type="submit">Register</button>  </form>  <p id="error-message" style="color: red;"></p>  <a href="{% url 'login' %}">Login</a>  <script>  const registerForm = document.getElementById('registerForm');  const errorMessage = document.getElementById('error-message');  registerForm.addEventListener('submit', async function(e) {      e.preventDefault();        const username = document.getElementById('username').value.trim();      const password = document.getElementById('password').value.trim();      const data = { username, password };      try {          const res = await fetch('/task-api/cregister/', {              method: 'POST',              headers: { 'Content-Type': 'application/json' },              body: JSON.stringify(data)          });          const result = await res.json();          if (res.ok) {              alert("Registration successful! You can now log in.");              window.location.href = "/task-api/login/";          } else {              // Handle errors              if (result.username) {                  errorMessage.textContent = result.username.join(", ");              } else if (result.password) {                  errorMessage.textContent = result.password.join(", ");              } else if (result.detail) {                  errorMessage.textContent = result.detail;              } else {                  errorMessage.textContent = "Registration failed. Try again.";              }          }      } catch (err) {          console.error(err);          errorMessage.textContent = "An error occurred. Try again.";      }  });  </script>  {% endblock %} |

|  |
| --- |
| **login.html:**  {% extends 'base.html' %}  {% block title %}Login{% endblock %}  {% block content %}  <h2>Login</h2>  <form id="loginForm">    {% csrf\_token %}    <label>Username:</label>    <input type="text" id="username" required><br><br>    <label>Password:</label>    <input type="password" id="password" required><br><br>  <div id="error-message" style="color: red;"></div>    <button type="submit">Login</button>  </form>  <script>  document.getElementById("loginForm").addEventListener("submit", async function (e) {      e.preventDefault();      const username = document.getElementById("username").value.trim();      const password = document.getElementById("password").value.trim();      try {          const response = await fetch("/task-api/custom-login/", {              method: "POST",              headers: {                  "Content-Type": "application/json",              },              body: JSON.stringify({ username, password }),          });          const data = await response.json();          if (response.ok) {              // Save JWT tokens and username in localStorage              localStorage.setItem("access", data.access);              localStorage.setItem("refresh", data.refresh);              localStorage.setItem("username", data.username);              alert("Login successful!");              window.location.href = "/task-api/tasks/"; // Your frontend page or API test route          } else {              document.getElementById("error-message").innerText = data.error || "Invalid login credentials";          }      } catch (error) {          document.getElementById("error-message").innerText = "Something went wrong. Try again.";      }  });  </script>  {% endblock %} |

|  |
| --- |
| **tasks/task\_form.html**  {% extends "base.html" %}  {% block content %}  <h2>{% if task %}Edit Task{% else %}Create Task{% endif %}</h2>  <form id="taskForm">      <label for="title">Title:</label>      <input type="text" id="title" name="title" value="{{ task.title|default:'' }}" required>      <br><br>      <label for="description">Description:</label>      <textarea id="description" name="description" required>{{ task.description|default:'' }}</textarea>      <br><br>      <label for="due\_date">Due Date:</label>      <input type="date" id="due\_date" name="due\_date" value="{{ task.due\_date|date:'Y-m-d' }}">      <br><br>      <label for="status">Status:</label>      <select id="status" name="status">          <option value="Pending" {% if task.status == "Pending" %}selected{% endif %}>Pending</option>          <option value="Completed" {% if task.status == "Completed" %}selected{% endif %}>Completed</option>      </select>      <br><br>      <button type="submit">Save</button>  </form>  <script>  document.getElementById("taskForm").addEventListener("submit", async function(e) {      e.preventDefault();      const token = localStorage.getItem("access");      const data = {          title: document.getElementById("title").value,          description: document.getElementById("description").value,          due\_date: document.getElementById("due\_date").value,          status: document.getElementById("status").value      };      const response = await fetch("/task-api/api/tasks/create/", {          method: "POST",          headers: {              "Content-Type": "application/json",              "Authorization": `Bearer ${token}`          },          body: JSON.stringify(data)      });      if (response.ok) {          alert("Task saved!");          window.location.href = "/task-api/tasks/";      } else {          alert("Error saving task");      }  });  </script>  {% endblock %} |

|  |
| --- |
| **tasks/task\_list.html:**  {% extends 'base.html' %}  {% block title %}My Tasks{% endblock %}  {% block extra\_css %}  <style>  .modal {      display: none; /\* Hidden by default \*/      position: fixed;      top: 0;      left: 0;      width: 100%;      height: 100%;      background-color: rgba(0,0,0,0.8); /\* Dark semi-transparent background \*/      z-index: 1000;      overflow-y: auto; /\* Scroll if content is too tall \*/      padding: 50px 20px; /\* Space around content \*/      box-sizing: border-box;  }  .modal-content {      background-color: #fff;      margin: auto;      padding: 20px;      border-radius: 10px;      max-width: 600px; /\* You can make this 100% if you want \*/      width: 100%;      box-shadow: 0 0 10px #000;      position: relative;  }  .close {      position: absolute;      top: 10px;      right: 15px;      font-size: 28px;      font-weight: bold;      cursor: pointer;  }  </style>  {% endblock %}  {% block content %}  <h2 id="greeting"></h2> <!-- Greeting will appear here -->  <h2>My Tasks</h2>  <div id="tasks"></div>  <!-- Edit Modal -->  <div id="editModal" class="modal" style="display:none;">      <div class="modal-content">          <span class="close" onclick="closeModal()">&times;</span>          <h3>Edit Task</h3>          <form id="editForm">              <input type="hidden" id="editId">              Title: <input type="text" id="editTitle" required><br><br>              Description: <textarea id="editDescription" rows="5" required></textarea><br><br>              Status:              <select id="editStatus">                  <option value="Completed">Completed</option>                  <option value="Pending">Pending</option>              </select><br><br>              <button type="submit">Save Changes</button>          </form>      </div>  </div>  <script>  const token = localStorage.getItem('access');  const username = localStorage.getItem('username');  const isSuperuser = localStorage.getItem('is\_superuser') === 'true';  const apiBase = "/task-api/api";  // Redirect if no token  if (!token) {      window.location.href = "/task-api/login/";  }  // Set greeting  document.getElementById('greeting').textContent = isSuperuser      ? "Hello Superuser"      : `Hello ${username || 'User'}`;  function handleAuthError(res) {      if (res.status === 401) {          alert("Session expired. Please log in again.");          localStorage.clear();          window.location.href = "/task-api/login/";          return true;      }      return false;  }  function loadTasks() {      fetch(`${apiBase}/tasks/`, {          headers: { 'Authorization': `Bearer ${token}` }      })      .then(res => {          if (handleAuthError(res)) return [];          return res.json();      })      .then(tasks => {          const container = document.getElementById('tasks');          container.innerHTML = '';          if (!tasks || tasks.length === 0) {              container.innerHTML = "<p>No tasks found.</p>";              return;          }          tasks.forEach(task => {              container.innerHTML += `                    <div class="col-md-3" style="border:1px solid #ccc;padding:10px;margin:10px; display:inline-block">                      <h3>${task.title}</h3>                      <p>${task.description}</p>                      <p><b>Status:</b> ${task.status}</p>                      <p><b>Due:</b> ${task.due\_date || 'N/A'}</p>                      <button onclick="openEditModal(${task.id}, '${task.title.replace(/'/g, "\\'")}', '${task.description.replace(/'/g, "\\'")}', '${task.status}')">Edit</button>                      <button onclick="deleteTask(${task.id})">Delete</button>                  </div>              `;          });      })      .catch(err => {          console.error("Error loading tasks:", err);          alert("Error loading tasks.");      });  }  function openEditModal(id, title, description, status) {      document.getElementById('editId').value = id;      document.getElementById('editTitle').value = title;      document.getElementById('editDescription').value = description;      document.getElementById('editStatus').value = status;      document.getElementById('editModal').style.display = 'block';  }  function closeModal() {      document.getElementById('editModal').style.display = 'none';  }  document.getElementById('editForm').addEventListener('submit', function(e) {      e.preventDefault();      const id = document.getElementById('editId').value;      const updatedTask = {          title: document.getElementById('editTitle').value,          description: document.getElementById('editDescription').value,          status: document.getElementById('editStatus').value      };      fetch(`${apiBase}/tasks/${id}/update/`, {          method: 'PATCH',          headers: {              'Content-Type': 'application/json',              'Authorization': `Bearer ${token}`          },          body: JSON.stringify(updatedTask)      })      .then(res => {          if (handleAuthError(res)) return;          if (res.ok) {              alert("Task updated");              closeModal();              loadTasks();          } else {              alert("Error updating task");          }      });  });  function deleteTask(id) {      if (!confirm("Delete this task?")) return;      fetch(`${apiBase}/tasks/${id}/delete/`, {          method: 'DELETE',          headers: { 'Authorization': `Bearer ${token}` }      })      .then(res => {          if (handleAuthError(res)) return;          if (res.ok) {              alert("Task deleted");              loadTasks();          } else {              alert("Error deleting task");          }      });  }  window.onclick = function(event) {      if (event.target === document.getElementById('editModal')) {          closeModal();      }  }  loadTasks();  </script>  {% endblock %} |