Full-Stack Developer Candidate Challenge (React + Django)

Objective

Build a **collaborative task management web app** with **authentication**, **real-time updates**, and a **dashboard**. This will test your skills in **API design, frontend state management, authentication, deployment considerations, and code clarity**.

Requirements

Backend (Django + Django REST Framework)

- **User Authentication**
  - Implement registration & login with JWT-based authentication (using `djangorestframework-simplejwt` or equivalent).
  - Each user should only see and manage their own tasks.
- **Task Model**
  - Fields:
    - `id` (auto-generated)
    - `title` (string, required)
    - `description` (text, optional)
    - `status` (choices: `pending`, `in-progress`, `completed`)
    - `priority` (choices: `low`, `medium`, `high`)
    - `due_date` (date, optional)
    - `created_at` (timestamp)
    - `updated_at` (timestamp)
    - `owner` (linked to the authenticated user)
- **Endpoints**
  - `POST /auth/register/` → Register a user
  - `POST /auth/login/` → Login & receive JWT tokens
  - `GET /tasks/` → List authenticated user's tasks (with filtering & search by status/priority/due date)
  - `POST /tasks/` → Create a task
  - `PUT /tasks/{id}/` → Update a task

    - `DELETE /tasks/{id}/` → Delete a task
    - **BONUS:** Add a WebSocket (via Django Channels) or Server-Sent Events endpoint to push real-time task updates.

Frontend (React)

- **Authentication Flow**
    - Register & login pages
    - Store JWT tokens securely (in memory or HttpOnly cookie)
    - Redirect users to login if not authenticated
- **Task Management**
    - Show all tasks in a table or Kanban-style board
    - Allow creating, editing, and deleting tasks
    - Filter & search tasks by status, priority, and due date
    - Display overdue tasks clearly (e.g., red highlight)
- **Dashboard**
    - Display statistics:
        - Number of tasks per status (pending, in-progress, completed)
        - Tasks due today / this week
        - A small chart (e.g., Pie/Bar chart using Chart.js/Recharts)
- **Real-Time Updates**
    - If you implement WebSockets/Server-Sent Events, the frontend should update automatically when a task changes (without page refresh).

Bonus (Optional, but impressive)

- Deploy the project (e.g., backend on Heroku/Render and frontend on Netlify/Vercel).
- Add role-based permissions (e.g., admin can see all tasks, normal users see only their own).
- Add drag-and-drop for task status updates (e.g., move a card from "Pending" → "In Progress").

Submission

1. Push code to a **public GitHub repository** (separate `/frontend` and `/backend` folders).
2. Include a detailed `README.md` with:
    - Setup instructions

    - API documentation (endpoints, request/response examples)
    - How to run frontend & backend together
    - Any deployment link (if done)
3. Record a **5-minute video** explaining:
    - Your architecture choices
    - API design & authentication
    - A walkthrough of the app features
    - (If done) deployment link demo
4. Submit:
    - GitHub repository link
    - Video link (YouTube unlisted, Loom, or Drive)

Evaluation Criteria

- **Technical Depth:** Authentication, filtering, state management, API design
- **Code Quality:** Structure, readability, and use of best practices
- **Frontend Skills:** React hooks, state management, clean UI, optional real-time handling
- **Backend Skills:** Secure authentication, efficient query filtering, scalable API design
- **Bonus Efforts:** Deployment, real-time updates, role-based permissions, UX polish