Task Overview

You are required to build a simple internal **Ticketing System** where employees can submit support tickets, and admins can view and manage them.

The system must include:

- Authentication (only registered users can submit tickets)
- Role-based access (Admin / User)
- Ticket submission (with file attachments)
- Ticket list with filters, search, and pagination
- Ticket details page
- Integration with **Tawk.to** or **Chatwoot** to provide real-time support chat
- Clean API structure using Django
- Frontend interface using **your choice**: React OR Vanilla JS

The goal of this task is to evaluate your understanding of backend architecture, API design, integration logic, UI implementation, and state management.

Features to Build

1. Authentication

- Users must log in before submitting or viewing tickets
- Admin panel access restricted to Admin role only
- Use Django authentication (JWT or session-based allowed)

User Roles:

- **Admin**:
  - View all tickets
  - Filter, search, paginate
  - Change ticket status
- **User**:
  - Create tickets only
  - View only their own tickets

2. Ticket Management

Each ticket contains:

Field	Type
id	integer
title	string
description	string
category	enum (Technical / Financial / Product)
status	enum (New / Under Review / Resolved)
attachment	file (optional)
createdBy	user
createdAt	datetime

3. User Ticket Submission Page

Users should be able to:

- Enter ticket title
- Choose category
- Write description
- Upload a file
- Submit the ticket

---

4. Ticket List (Frontend)

For **Admin**:

- View all tickets
- Filter by:
  - Category
  - Status
- Search by title + username
- Pagination (client-side or server-side – your choice)

For **User**:

- See only **their** tickets
- Same filters/search but only for their items

5. Ticket Details Page

Shows:

- Ticket info
- Attachment preview (if image/PDF)
- Status history (optional but good to have)
- Admin can update ticket status

---

6. Real-Time Support Chat Integration

Integrate **Tawk.to OR Chatwoot** widget.

Requirements:

- Use the default widget (no custom chat UI)
- Visible on all pages
- Correct installation + working session

Frontend Requirements

You are allowed to use:

Option A – React

- Any free template
- Axios or Fetch
- React Router
- Simple state management (Context API or local state)

Option B – Vanilla JS

- Any free HTML/CSS/Bootstrap template
- Fetch API
- SPA or multi-page – your choice

The candidate **must provide the template link** inside the submission.

Backend (Django) Requirements

Models

- User
- Ticket

API Endpoints for example (at minimum)

Method	Endpoint	Description
POST	/api/auth/login	Log in
GET	/api/tickets	List tickets (filtered by role)
POST	/api/tickets	Create ticket
GET	/api/tickets/:id	Ticket details

PATCH	/api/tickets/:id/status	Update status (Admin only)

File Upload

Use Cloudinary storage.

Deliverables

1. GitHub Repository

Containing:

- Django backend folder
- React/Vanilla frontend folder
- README with setup instructions

2. Environment Setup Instructions

README must include:

- How to run backend
- How to run frontend
- How to create superuser
- How to test APIs
- Where the design template came from
- How to test chat integration

3. Submission

- GitHub repo link
- Live demo is optional but extra credit

Evaluation Criteria

Ability to design clean APIs

Correct role-based access control

Quality of frontend integration

Search, filters, and pagination working correctly

Functional ticket creation

Proper Tawk.to / Chatwoot integration

Clean architecture + readable code

Clear GitHub structure & documentation