Assignment: Ticket Dashboard

Objective

Build a mini project management dashboard (Trello/Atlassian style) with email-based authentication, project & ticket management, super-user controls, and notifications.

Base Requirements

Step 1. Authentication

- Email-based OTP login (no password required).
- After successful login → access to ticket dashboard page.

Step 2. Projects & Tickets

- Ticket Dashboard lists **all projects**.
- If no projects → allow creating a project.
- Each project can have multiple **tickets** with description.
    - whoever moves ticket, it should instantly reflect for other users when they are viewing this dashboard
- **Super-user toggle (**More info refer figma - Figma **)**:
    - **ON** → display who created/updated tickets.
    - **OFF** → hide user info.
    - Toggling **ON** requires entering a password.

Step 3. Notifications & Updates

- **Activity feed**: All ticket updates are shown in notifications for active users instantly.
- **Email notifications**:
  - If any team member has already visited and remains offline later, then send updates via email.
  - UI notifications only for active users only.

Step 4. Backend Design (NestJS/Node.js/Flask/FastAPI/Golang)

- Use a database which best suits it.
- Must include:
  - At least **one design pattern** (e.g., Strategy for notifications, Factory for ticket creation etc)

Step 5. Frontend Design (React/Next.js + Redux/Zustand) (Figma - Figma)

- **Minimal but structured UI**:
  - Project list page
  - Project detail page with ticket list
  - Super-user toggle with password prompt
  - Notifications icon
- **Styling:** Basic clean UI; no heavy animations required.

---

6. Deliverables

- GitHub repo with `backend/` and `frontend/`.
- `README.md` explaining:

- Database/Design decisions
- (Optional) Basic Design patterns & architecture

Evaluation Criteria

1. **Backend logic & design**
    - Super-admin toggle implementation
    - Notifications handling
    - Database usage like SQL/NoSQL - justification
        i. Why did you preferred NoSQL over SQL - Viceversa
2. **Frontend architecture & state management**
    - Conditional rendering based on toggle
3. **Code quality**
    - Proper Naming convention, clean, modular, reusable code
    - Proper folder structure & patterns
    - Deployment - Make sure deploy end to end from frontend to backend

---

Note

1. **Frontend preferred**
    - Libraries/Frameworks - **Typescript** with React/Nextjs
    - Css - Tailwind css, or Scss or plain css
    - Any state management preferred like Redux, Zustand, or MobX
    - Maintaining output pixel perfect
2. **Backend preferred**
    - JS/Python/Golang stack **-** NestJS/Nodejs/Flask/FastAPI
    - If you are using JS stack - then prefer **Typescript** must use it to avoid Javascript.
3. **Avoid Javascript**, Must use **Typescript** if JS stack chosen in frontend/backend.
4. Clean code, design patterns, maintaining code efficient and scalable

5. Try atlassian or trello for better understanding of ticketing platform usage
6. Send the github link and deployed links and other details before the deadline, which is communicated via email (cc: hqneurs@gmail.com & team@cognitoinnovations.com)
7. The Github link should be public.