🐞 Full-Stack Developer Assignment
Bug Reporting System

🎯 Objective

Design and implement a **full-stack bug reporting system** where users can:

- Create projects.
- Report and track issues.
- Manage issue status and assignments.
- Interact through a frontend UI.

This assignment evaluates **backend (Django REST Framework)** + **frontend (React or similar)** + **integration skills**.

Part 1: Backend (Django REST Framework)

Models

1. **User** (use Django's built-in).
2. **Project**: name, description, created_at.
3. **Issue**:
    - title, description, status (`open`, `in_progress`, `closed`), priority (`low`, `medium`, `high`, `critical`),
    - created_at, updated_at,
    - project (FK),
    - reporter (FK to User),
    - assignee (FK to User, nullable).

4. **Comment**: content, created_at, issue (FK), author (FK to User).

API Endpoints

- **Auth**: JWT login & register.
- **Projects**:
    - `POST /projects/` (create project)
    - `GET /projects/` (list projects)
- **Issues**:
    - `POST /projects/<id>/issues/` (create issue under project)
    - `GET /projects/<id>/issues/` (list issues)
    - `PATCH /issues/<id>/` (update status/assignee)
- **Comments**:
    - `POST /issues/<id>/comments/`
    - `GET /issues/<id>/comments/`

Requirements

- Only authenticated users can create/update issues or comments.
- Only a reporter or assignee can update an issue.
- Optimize queries with `select_related` / `prefetch_related`.

Part 2: Frontend (React / Next.js)

Pages

1. **Login Page**
    - Login with JWT (store token).
    - Redirect to dashboard.
2. **Dashboard** (Projects)
    - Show list of projects (API: GET `/projects/`).
    - Click project → navigate to Issue List page.

3. **Project Issues**
    - Show issues in a project (GET `/projects/<id>/issues/`).
    - Create a new issue (POST).
    - Update issue status or assignee (PATCH).
    - Filter issues by status & priority.
4. **Issue Detail**
    - Show issue details + comments.
    - Add a comment (POST).

Part 3: Integration

- Frontend must consume the backend APIs.
- Handle JWT authentication in API requests.
- Show loading, error, and empty states.

Bonus Features (Optional)

- Search issues by keyword.
- Pagination or infinite scroll for issues.
- Swagger/OpenAPI docs for backend.
- Dockerized setup (backend + frontend).
- Unit tests for backend or frontend.

Submission Guidelines

- Push code to **GitHub/GitLab**.
- Backend & frontend should be in separate folders (`/backend`, `/frontend`).
- Include a **README** with setup instructions for both.
- (Bonus) Deploy frontend (Netlify/Vercel) + backend (Heroku/Render).

Evaluation Rubric (Out of 100)

Section	Points	Notes
Backend		
Models & Relationships	15	Schema design
API Endpoints	20	CRUD, nesting
Auth & Permissions	15	JWT + RBAC
Query Optimizations	5	Efficient DB usage
Frontend		
Authentication Flow	10	JWT login
Projects Dashboard	10	Projects list UI
Issue Management	15	CRUD integration
Filtering & Sorting	5	Query params
Integration & Extras		
Full Integration	5	Frontend ↔ Backend
Bonus Features	+10	Docs, Tests, Docker, Architecture
Total	100 +10	