This project is my full-stack task management system built using an NX monorepo. It provides secure role-based access control (RBAC) with JWT authentication, allowing different types of users (Owner, Admin, Viewer) to manage projects, teams, and tasks according to their permissions.

Setup Instructions

1. Clone the repository

```
git clone <your-repo-url>
cd securetask
```

2. Install dependencies

What I had to install

```
npm install @nestjs/jwt passport-jwt
npm install passport @nestjs/passport
npm install bcrypt
npm install --save-dev @types/bcrypt
npm install @nestjs/config
npm install nodemailer
npm install class-validator class-transformer
```

3. Environment variables

Create .env files inside apps/api and apps/dashboard.

apps/api/.env

```
JWT_SECRET=your-secret-key
DB_HOST=localhost
DB_PORT=5432
DB_USER=your-db-user
DB_PASS=your-db-password
DB_NAME=securetasks
```

apps/dashboard/.env

```
API_URL=http://localhost:3000
```

4. Run backend : npx nx serve api

Backend available at: http://localhost:3000

5. Run frontend: npx nx serve dashboard
Frontend available at: http://localhost:4200

Architecture Overview

api: Implements authentication, access control, and CRUD APIs for tasks and projects.
Dashboard: Angular + TailwindCSS frontend for authentication and task management.
data: Shared interfaces for consistency across backend and frontend.
auth: Contains reusable RBAC guards/decorators.

Roles
Owner: Full access across projects and tasks.
Admin: Manage tasks, users, and teams within their organization.
Viewer: Read-only access to tasks and projects.

Guards & Decorators
@Roles('Admin'): Restricts endpoint access by role.
@OrgScoped(): Ensures users can only access data within their organization.
JWT strategy integrated into all secured endpoints.

Data Models
User: email, password (hashed), role, organization.
Organization: supports two-level hierarchy.
Project: contains teams and tasks.
Task: title, subject, body, linked to a team/project.
Audit Logs: records actions for Owners/Admins to review.

Sample API Requests & Responses
Authentication

POST /auth/login

```
{ "email": "user@example.com", "password": "password123" }
```

Response:

```
{ "access_token": "JWT_TOKEN" }
```

Projects

POST /projects

```
{ "project_name": "MyProject", "num_teams": 2 }
```

Response:

```
{ "id": 1, "project_name": "MyProject", "num_teams": 2 }
```

Tasks

POST /tasks

```
{ "title": "Frontend Fix", "subject": "UI Bug", "body": "Fix navbar issue",
"team_id": 6 }
```

Response:

```
{ "id": 12, "title": "Frontend Fix", "subject": "UI Bug", "team_id": 6 }
```

Frontend Features

User login with JWT authentication.

Dashboard showing user info, role, and projects.

Project creation and management (teams, members, roles).

Task management within teams:

- Create, edit, delete tasks.
- Task listing scoped to team/project.

Responsive UI built with TailwindCSS.

Testing

Backend: Jest tests for RBAC guards, JWT strategy, and CRUD endpoints.

Frontend: Jest/Karma component tests and state management checks.

Future Enhancements

Security: refresh tokens, CSRF protection, RBAC caching.

Scalability: audit log persistence in DB, background job handling.

UI/UX: drag-and-drop task reordering, dark/light mode toggle, task completion charts.

Notifications: email or in-app alerts via nodemailer integration.

Better Data links.