# Project Development Guidelines

## 1. Strong Testing (MUST)

- Always include unit tests, integration tests, and E2E tests.

- Use Jest + Supertest for backend APIs.

- Use React Testing Library + Cypress/Playwright for frontend.

- Maintain a separate test database (petrol\_pump\_test) to avoid corrupting production data.

- Run tests automatically before deployment (CI/CD pipeline ready).

## 2. Error Handling & Frontend Display

- Backend must always return structured errors: { 'status': 400, 'message': 'Invalid input' }

- Frontend must catch and display errors as user-friendly alerts (toast notifications, banners).

- No raw or technical errors should be shown to end users.

## 3. Clean & Scalable Code

- Use TypeScript everywhere (backend + frontend).

- Maintain modular architecture (NestJS modules, React components).

- Follow Prisma ORM best practices for DB schema & migrations.

- Keep code well-documented and linted (Prettier + ESLint).

## 4. Database Management

- Use PostgreSQL as main DB.

- Use Prisma for schema, migrations, and DB access.

- Ensure test DB and production DB remain separate.

- Keep schema optimized for financial operations (transactions, creditors, reports).

## 5. Performance & Scalability

- Use Redis caching for repeated queries (e.g., dashboard totals).

- Design APIs to be async/non-blocking.

- Keep system microservices-ready (so it can scale later).

## 6. Frontend Best Practices

- Use React + Next.js + TypeScript for performance.

- Build role-based dashboards (Admin vs Manager).

- Always connect UI forms with validation + error display.

- Ensure modern appearance (Material UI / Tailwind).

## 7. Automation & Deployment

- Use Docker for containerization.

- Setup CI/CD pipelines to run all tests before deployment.

- Deploy on cloud (AWS/GCP/Azure) with scalability in mind.

Assign code for all errors

All pages ,modules place well organized with simple and compleate names