

Full-Stack Mini Project Assignment

Objective

Build a simple full-stack web application with role-based authentication. This assignment evaluates your problem-solving skills, ability to integrate frontend and backend, and deployment capabilities.

Deliverables

- GitHub repository with clear README.md
- Working deployed frontend and backend URLs (publicly reachable)

What to Build

Create a web application with role-based signup/login and role-specific dashboards.

Core Requirements (Mandatory)

1. Authentication with Roles

- Signup page with role selection (User or Admin)
- Login page
- Secure password storage (bcrypt or similar)
- JWT or session-based authentication

2. Dashboard

- After login, redirect to dashboard
- Dashboard displays a header showing: "Welcome, [Name] (User)" or "Welcome, [Name] (Admin)"
- User dashboard and Admin dashboard can be the same page with different header text
- Protected route (accessible only when logged in)

3. Deployment

- Deploy frontend (Vercel/Netlify)
- Deploy backend (Render/Railway/Vercel Serverless)
- Include `.env.example` file

Technical Stack

Backend

- Node.js with Express
- Database: PostgreSQL with Prisma (Supabase/Neon free tier) OR MongoDB (MongoDB Atlas free tier)
- Password hashing: bcrypt
- Auth: JWT or session-based

Minimum Endpoints:

```
POST    /auth/signup
POST    /auth/login
GET     /auth/me  (returns current user info)
```

Frontend

- Next.js with TypeScript
- Any UI library: Shad Cn, TailwindCSS
- Form handling library (optional): react-hook-form

Required Pages:

- Signup (with role selection dropdown)
- Login
- Dashboard (protected route)

Optional Enhancements

Implement these if you finish early or want to showcase additional skills:

- CRUD operations (create/edit/delete items specific to each user)
- Item listing with search and pagination
- Backend and frontend tests (jest, React Testing Library)
- Data tables with filtering
- Logout functionality
- Form validation (zod/joi)
- Different UI for Admin vs User dashboards

Evaluation Criteria

Note: You retain full ownership of your work and may include it in your portfolio.

Category	Description	Weight
Functionality	Auth flow and dashboard work correctly	30%
Problem Solving	Code organization and approach	25%
Deployment	Working deployed URLs	20%
Code Quality	Clean, readable code	15%
Documentation	Clear README with setup instructions	10%