

# Full-Stack Practical Exam

## Intern Staff Developer (Hybrid)

Thank you for applying for the Intern Staff Developer (Hybrid) role. This practical exam is designed to evaluate how you think, design, and deliver a real-world system. There is no single correct solution. We value clarity, technical decision-making, and practical execution over perfection.

### Project Overview

Build a Mini Multi-Tenant Property Listing Platform. This is a simplified version of a real-world system used for listing, managing, and viewing properties with different user roles.

### Required Tech Stack

**Backend (choose one):** NestJS (preferred) or Node.js with Express.js (must justify choice)

**Frontend (choose one):** Next.js (preferred) or React.js

**State Management:** Redux Toolkit, Zustand, or TanStack Query (justify choice)

**Database:** PostgreSQL or MongoDB

**Deployment:** Frontend (Vercel/Netlify), Backend (Railway/Render/Fly.io), Cloud Database

### User Roles & Permissions

**Admin:** View all properties, disable any property, view basic system metrics

**Property Owner:** Create properties, upload images, edit draft properties, publish properties

**Regular User:** View published properties, save favorites, contact property owners

### Property Rules

Each property must include title, description, location, price, multiple images, and status (draft, published, archived). Published properties cannot be edited and must be validated before publishing. Soft deletes are required.

### Backend Requirements

- JWT-based authentication and role-based access control
- Pagination and filtering (location, price range, status)
- Soft deletes using `deletedAt`
- Transactional logic when publishing properties
- Environment-based configuration (dev vs prod)
- Proper error handling and HTTP status codes

### Frontend Requirements

- Authentication pages (login/register)
- Public property listing page (server-side rendered)

- Property detail page
- User, Owner, and Admin dashboards (client-side rendered)
- Persist authentication across refresh
- Favorites synced across tabs
- At least one optimistic UI update
- Protected routes and proper loading/error states

## **Image Handling**

Images must work in production, be validated by type and size, and be stored using cloud storage or external URLs (with justification).

## **Deployment**

The application must be fully deployed and accessible online. You must provide live frontend and backend URLs.

## **Technical Decision Document**

Include a short write-up answering: why you chose your backend framework, your state management approach, how access control is enforced, the hardest technical challenge faced, and what would break first at scale.

## **Submission Requirements**

- GitHub repository with clean commit history
- Deployed frontend URL
- Deployed backend API URL
- API documentation (Postman collection or Swagger)
- Technical decision document (README is acceptable)

## **Time Limit**

You have 7 days to complete this exam. Do not over-engineer. Incomplete but well-reasoned solutions are acceptable.