Competency Assessment: Full-Stack Web Developer

📝 Task Overview

This task is designed to evaluate your practical technical competency as a **Full-Stack Web Developer**.
Based on the job description for the internship, you will demonstrate your ability to design, implement, and organize frontend and backend codebases using modern web technologies with scalability and maintainability in mind.

> ⚠️ **You are free to utilize AI tools**, but you must not blindly copy and paste AI-generated content without understanding or customizing it.
> Submissions that show no personal insight or logical decision-making will be automatically disqualified.

📌 Task:

🧩 “모두의 권리” – Service Matching Platform Feature Module Development

“모두의 권리” is a real-world service matching platform similar to Korea’s 숨고 **(Soomgo)**.
Users can sign in, request quotations from service providers, manage consultations, and edit their personal profile.

You are required to implement the **core client-facing modules** and the **associated backend logic** based on the provided Figma design.

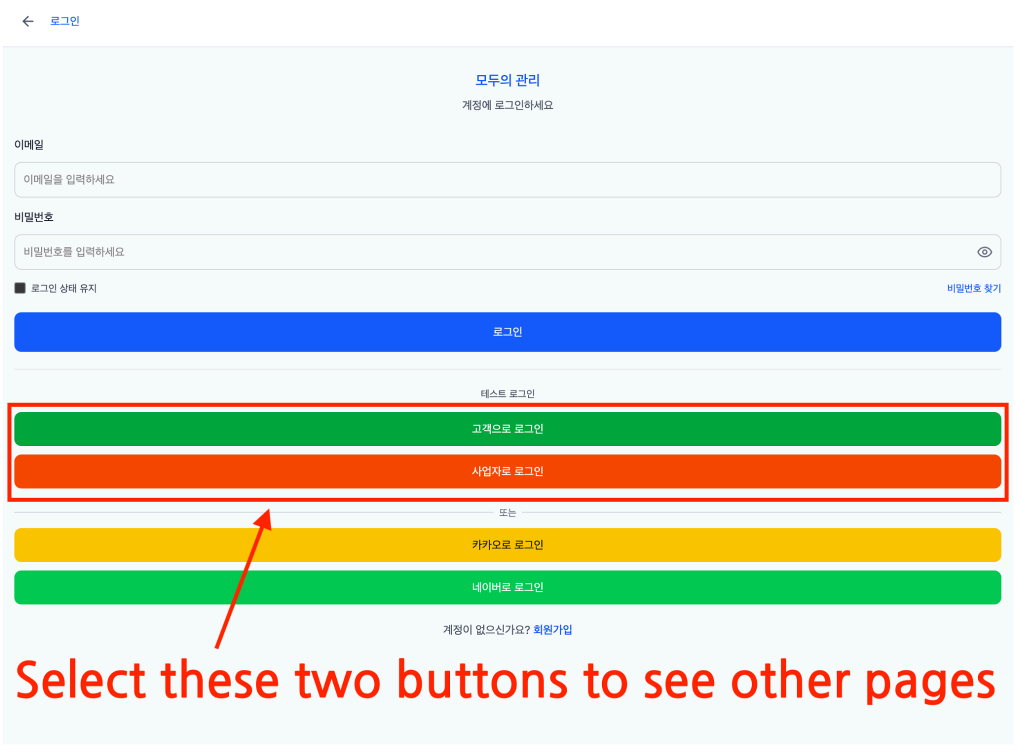
🎯 Objective

- Build a real login system and main user pages.
- Showcase frontend architecture, API integration, and scalable backend logic.
- Deliver a complete working demo that can be deployed and tested.

🟦 1. Full Stack Web Development (PC + Mobile)

✅ Site example

👉 [Link]



📄 Required Pages to Implement

- **Login Page** (must include actual login functionality)
- **Home**
- 받은 견적 **(Received Quotations)**
- 상담내역 **(Consultation History)**
- 마이페이지 **(My Page)**

🔐 Login Functionality Requirements

The login flow must be **fully functional and secure**.
A simple mock button that redirects to another page without authentication logic is **not allowed**.

Your implementation must:

- Accept and validate user credentials (e.g., email & password)
- Send authentication requests to the backend API
- Store the access token securely (e.g., in `localStorage` or `cookie`)

- Handle login success and failure UI cases
- Redirect to protected pages after login
- Optionally restrict unauthorized access to protected routes

---

⚙️ Frontend Architecture Guidelines

- Do **not** build your pages using plain HTML or hardcoded layouts.
- This task emphasizes **scalable frontend architecture and code organization**.

✅ Please use **modern component and state management practices**, such as:

- Atomic Design Pattern (e.g., atoms, molecules, organisms)
- MVC-inspired logic separation
- Folder-by-feature or domain-based structure
- Apply best practices in **Zustand** for state management
- Use **TanStack Query** for efficient server communication and caching
- Feel free to use **mock data** where necessary

Your architectural decisions should be clearly explained in the README.

🛠️ Backend Guidelines (NestJS + PostgreSQL)

You are expected to build a backend using **NestJS** and **PostgreSQL**.
You may choose between **Prisma** or **TypeORM** for your ORM.

At minimum, the backend must support:

- User login authentication (with token issuing)
- Basic user info API
- Mock endpoints for:
  - Quotation list
  - Consultation history

💡 Swagger documentation is optional but encouraged.

📂 Submission Format

You must submit all of the following:

1. **GitHub Repository** with:
   - All source code (frontend & backend)

    - Organized project structure

2. **README.md** that includes:
    - Setup & run instructions
    - Explanation of your technical decisions and folder/component architecture
    - Challenges faced and potential improvements

3. **Working Demo Video** with **your voice narration** explaining the flow and structure

4. **Live Deployment URL**
   (Frontend: Vercel, Netlify / Backend: Render, Railway, etc.)

🧪 Evaluation Criteria

Category	Details
✅ Code Quality	Readability, modularity, reusability, and maintainability
✅ Functionality	Correct and complete implementation of required features
✅ Scalable Architecture	Clean frontend structure, backend domain modeling, folder organization
✅ Tooling & Modern Stack	Effective use of Zustand, TanStack Query, Prisma/TypeORM, etc.
✅ Authentication Logic	Fully functional login flow with secure token handling
⚠️ AI Usage	Permitted, but must include your own logic and insight — no blind copy-paste