Full Stack Coding Exam

Build an IAM-Style Access Control System

- **Difficulty**: Moderate
- **Type**: Full-stack (React + Express + SQLite in-memory)

Objective

Create a simplified Identity and Access Management (IAM) system where:

- Users are assigned to groups
- Groups have roles
- Roles define fine-grained permissions CRUD on modules
- Users inherit permissions **only via group memberships**
- A React frontend enables full **CRUD** for all IAM entities including **Modules**

Technologies to Use

Backend – Node.js + Express

- **Express.js** for REST API structure
- **Middlewares**
- **Validations**
- **JWT Authentication**
- **Password Security**
- **Database**: **SQLite (in-memory)** for simplicity

Frontend – React.js

- **React Router** for page navigation
- **Redux Toolkit** for auth state and permissions
- **Axios** for communicating with the API
- **JWT Storage**: Redux store (or localStorage)
- **UI Framework**: Tailwind CSS

Project Scope & Features

Authentication

- `POST /register` – Create user
- `POST /login` – Authenticate and return JWT
- Secure all routes using JWT middleware

Users

- **Backend**: CRUD API (`GET/POST/PUT/DELETE`)
- **Frontend**:
  - Create/edit/delete users
  - View users and their group memberships

Groups

- **Backend**: CRUD API
- `POST /groups/:groupId/users` – Assign users to group
- **Frontend**:
  - Create/edit/delete groups
  - View and assign users to groups

Roles

- **Backend**: CRUD API
- `POST /groups/:groupId/roles` – Assign roles to groups
- **Frontend**:
  - Create/edit/delete roles
  - Assign roles to groups

Modules

- Modules define business areas like "Users", "Groups", "Assignments", etc.
- **Backend**: CRUD API for modules
- **Frontend**:
  - Create/edit/delete modules

Permissions

- Permissions represent actions on modules (`create`, `read`, `update`, `delete`)
- **Backend**: CRUD API
- `POST /roles/:roleId/permissions` – Assign permissions to a role

- **Frontend**:
  - Create/edit/delete permissions
  - Assign permissions to roles

Access Control

- `GET /me/permissions` – Fetch current user's inherited permissions
- `POST /simulate-action` – Test a user's ability to perform an action on a module
- Use `checkPermission(module, action)` middleware to protect routes

Frontend Pages

- `/login`: Login screen
- `/dashboard`: Show current permissions and simulate actions
- `/users`: Manage users (CRUD)
- `/groups`: Manage groups + assign users
- `/roles`: Manage roles + assign to groups
- `/modules`: Manage modules (CRUD)
- `/permissions`: Manage permissions + assign to roles

Constraints

- All permissions are inherited **through group membership only**
- No direct user-to-permission assignments
- Secure all protected routes with JWT
- Validate input on both **frontend and backend**