Zelestra

Here's a proposed **revised version of the assessment** tailored specifically for a **Senior Web Developer** role, based on the original test you used for the mid-level Web and Mobile App Developer role.

Evaluation Test for the Role of Senior Web Developer

Overview

You are required to develop a robust, scalable **web application** that demonstrates your ability to design clean architecture, integrate external data sources, manage authentication, and ensure high performance and maintainability.

The application should:

- Fetch and visualize renewable energy project data from a public API.
- Support authentication and role-based access control (RBAC).
- Demonstrate best practices in frontend state management, testing, and clean code principles.
- Include modular backend logic if applying as a full-stack candidate.

Task Requirements

1. Authentication & Access Control

- Implement secure **email/password authentication** using Auth0/Firebase/Cognito.
- Add **role-based access control (RBAC)**: separate admin and user roles.
- Maintain secure session handling and token management.

2. Frontend Development

- Use **React.js** (Next.js optional) with **TypeScript**.
- Implement responsive UI using **Tailwind CSS / Material UI**.
- Create **modular components** and demonstrate code reusability.

3. API Integration

- Fetch data from a public API (e.g., NREL API or Open Energy Data).
- Implement pagination or infinite scrolling.
- Display structured project details with clear hierarchy.

4. Advanced Features

- Implement:
    - **Search and filters** (by energy type, location, status).
    - **Sorting** (e.g., by capacity, year).
    - **Charts** using Chart.js or Recharts to show capacity trends.

5. Backend (Optional but Strongly Preferred)

- Build a simple REST API using Node.js (Express) or Python (FastAPI/Django).
- Expose a `/projects` endpoint that fetches and transforms public API data.
- Use environment variables and proper config management.

6. Performance & Security

- Apply **lazy loading** and **memorization** where needed.
- Use **JWT/token-based auth** securely.
- Demonstrate **error boundaries** and **API retries**.

7. Testing & Documentation

- Write **unit and integration tests** using Jest/React Testing Library.
- Provide a well-structured **README**:
    - Project overview
    - Tech stack
    - Setup instructions
    - Architecture diagram (optional but preferred)
- Host on **GitHub** with frequent commits.

Zelestra

Evaluation Criteria

Category	Weight	Description
Frontend Architecture	30%	Modularity, state management, responsive design
API Integration & Data Flow	20%	Efficient fetching, error handling, caching
Authentication & Security	15%	Secure login, RBAC, session persistence
Performance Optimization	10%	Lazy loading, API call reduction, responsiveness
Code Quality & Testing	15%	Clean code, folder structure, unit/integration tests, best practices
Documentation & Git Hygiene	10%	README quality, commit history, usage instructions