Assessment: Full Stack Engineer

Objective

To assess the candidate's ability to develop a full-stack application tailored to compliance workflows. The task involves creating a simplified KYC (Know Your Customer) system that includes key features like user registration, ID verification, and basic reporting.

Task Overview

Build a small web application where users can:

1. Register and log in with role-based access (e.g., Admin, User).
2. Submit KYC information (e.g., name, email, uploaded ID document).
3. Admins can:
    - View a list of users and their KYC statuses.
    - Approve or reject KYC submissions.
4. Display a dashboard with basic KPIs (e.g., number of approved/rejected submissions).

Requirements

Frontend (Vue 3 (Preferred), React)

1. Authentication:
    - Implement client-side validation for forms
2. KYC Submission:
    - A form for users to input their details and upload an ID document.
    - Display feedback on the form's submission status (pending, approved, rejected).
3. Admin Dashboard:
    - A table to display users' details and KYC statuses.
    - Approve/Reject buttons for each user with corresponding status updates.
    - A KPI section summarizing:
        - Total users.
        - Approved, rejected, and pending KYCs.

Backend (Node.js)

1. Authentication and Authorization:
    - Use JWT for secure authentication.
    - Implement role-based access control (e.g., middleware to restrict admin functionality).
2. KYC Management:
    - Endpoints for submitting and retrieving user KYC data.
    - Endpoints for updating KYC statuses by admin users.
3. Database:
    - Use a relational database (e.g., PostgreSQL/MySQL) or a NoSQL database (e.g., MongoDB).
    - Store user credentials, roles, and KYC details securely.
4. File Storage:
    - Save uploaded ID documents using local storage or cloud storage (optional).

Evaluation Criteria

Technical Skills

- Frontend:
    - Effective use of Vue 3 features (composition API, reactivity, routing).
    - Clean and modular component design.
    - Implementation of client-side validation and state management.
- Backend:
    - Secure implementation of APIs (e.g., avoiding SQL injection, secure authentication).
    - Proper use of Node.js best practices (e.g., error handling, async programming).
- Database:
    - Logical database schema design for compliance-related data.
    - Efficient queries for fetching and updating user details.

Delivery and Design

- Clear and maintainable code.
- Comprehensive documentation (e.g., README explaining setup and usage).
- Scalability considerations (e.g., handling large user bases).
- Ability to meet deadlines and deliver a functional solution.

Bonus Points

- Integration of unit tests for critical components.
- Deployment of the application (e.g., using Heroku, AWS, or Vercel).
- Use of TypeScript for type safety.

Deliverables

1. Source Code: A GitHub repository with well-structured commits.
2. README:
    - Overview of the project.
    - Setup instructions for both frontend and backend.
    - Any assumptions or trade-offs made during development.
3. Deployed Application (optional): A live demo link (if deployed).

Timeframe

- Estimated Completion Time: 16-20 hours (to be submitted within 3 days of receiving the assessment).