TECHLINT

Sr Back End Developer Practical Test

Web-Based IP Address Management System

Objective:

Design and implement a web-based IP address management solution that allows authenticated users to:

- Record an IP address with a label and an optional comment.
- Perform CRUD operations on IP records based on user roles.
- Maintain a secure and tamper-proof audit log.

Functional Requirements:

Expected Architecture:

1. You will create a Micro Service that is composed of three Services
    a. **Gateway Service , Auth Service, IP Management Service**
2. Implement your preferred Design Pattern

Authentication & Authorization:

1. Users must log in to the system and receive an authentication token (JWT).
2. The system should support automatic token renewal to prevent session expiration issues.

IP Address Management :

1. Users can add an IP address (IPv4 or IPv6) along with a label and an optional comment.
2. Users can modify the label of an existing IP address.
3. Regular users can only modify IP addresses they added but cannot delete them.
4. Super-admins can modify or delete any IP address.
5. All users can view all recorded IP addresses.

Audit Logging:

1. Maintain an audit log that records all changes made by users, including login/logout events.
2. The audit log should track changes on an IP address:
   - Within a user session.
   - Over the entire lifetime of the IP address.
3. The audit log should track changes made by a user:
   - Within a session.
   - Over the user's lifetime.
4. The audit log must not be deletable by any user, including privileged ones.
5. Create an Audit log dashboard
   a. This only accessible by Users with super admin role

Tech Stack & Architecture Guidelines :

1. You are free to use your preferred tech stack. However, we recommend the following:
   a. Backend: **PHP (Lumen, Laravel, Symfony) / Node.js (Express, Nest) / Python (Flask, Django, FastAPI).**
   b. Frontend: **React, Angular, or Vue.js.**
2. The Front-End must communicate with the **gateway microservice only to request data from the backend (All service communication must be orchestrated via the gateway)**.
3. Follow best practices when setting up your front-end architecture.
4. All Services must have **independent Databases**, you can use **RDB or NoSQL**.
5. All microservices should be **completely independent, potentially unaware of each other**. However, data integrity must be maintained across the system. Service communication should be achieved primarily via internal API Request.

Submission Guidelines:

- Submit your work to a **publicly accessible Git repository**.
- A repository with **only one commit is not acceptable**—your development process matters just as much as the final outcome.
- Include a **README with clear installation and usage instructions**.
- **Dockerize** the application setup (Its expected all Services including FE will live within its own container).
- **Bonus:** Include some level of code testing.
- **Bonus:** Ensure the **UI/UX** of your application is well designed and intuitive

Timeframe:

You have **25 days** from receiving this test to submit your work.
Good Luck, and if you have any questions, feel free to email info@techlint.com anytime!