Java Senior Developer Project

What is

This study aims to evaluate the proficiency of senior Java developers through abstract problem-solving exercises. Java, being a versatile and widely-used programming language, demands a comprehensive understanding of its core concepts, design patterns, and best practices.

The exercises presented in this assessment cover a spectrum of Java-related topics, including but not limited to:

1. Object-Oriented Programming (OOP): Candidates will be tasked with designing and implementing complex class hierarchies, utilizing principles such as encapsulation, inheritance, and polymorphism to achieve robust and maintainable code structures.

2. Concurrency and Multithreading: The assessment includes scenarios that require candidates to demonstrate their understanding of concurrent programming paradigms in Java, employing mechanisms such as threads, synchronization, and locks to manage shared resources effectively.

3. Data Structures: Participants should choose the correct and most efficient usage of Java data structures.

4. Exception Handling and Error Management: The assessment evaluates candidates' ability to handle exceptional conditions gracefully, employing try-catch blocks, custom exceptions, and appropriate error-handling strategies to ensure robustness and fault tolerance in Java applications.

5. Design Patterns and Architectural Principles: Candidates will be expected to apply various design patterns (e.g., Singleton, Factory, Observer) and architectural principles (e.g., MVC, SOLID) to design and implement scalable, maintainable, and extensible Java solutions.

6. Testing: Participants must write comprehensive unit tests using frameworks like JUnit.

7. Dockerization: Assess the developer's skills in containerizing the application using Docker, including writing Dockerfiles and building Docker images.

8. Database Integration: Check the developer's proficiency in designing and implementing database schemas.

9. API Design: Evaluate the clarity, consistency, and adherence to best practices in the API design.

10. Monitoring: use logs and also metrics to have a real-time monitored application

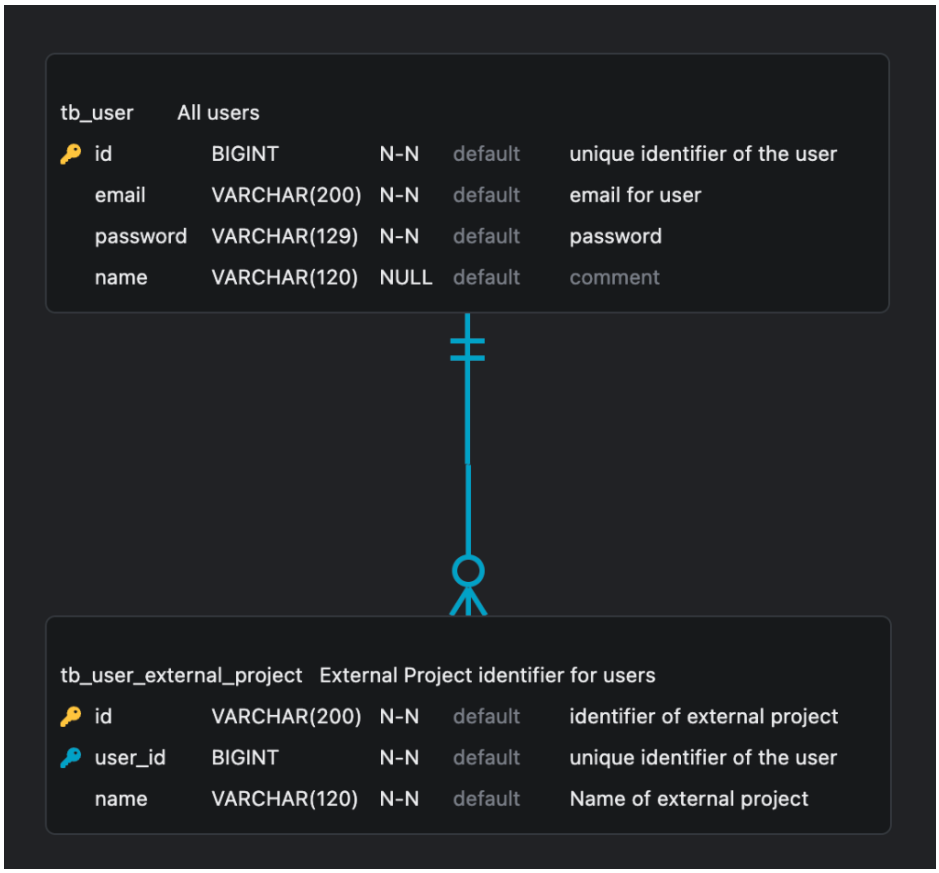
What needs to be done

Create a backend application using Spring Framework. The project can be easily generated using https://start.spring.io/. Java technology should be a minimum of 17 and a maximum of 21, and either Spring Web or Spring Reactive Web can be chosen.

The application should follow the RESTful API and best practices, including proper HTTP methods, error handling, and response formats (e.g., JSON).

ERM

Here is the entity model design, but feel free to change it. You can use either SQL or NoSQL databases. Do not be restricted to the types, column names or columns here, feel free to change them, create new columns.



```
CREATE TABLE tb_user
(
 id      BIGINT      NOT NULL COMMENT 'unique identifier of the user',
 email    VARCHAR(200) NOT NULL COMMENT 'email for user',
 password VARCHAR(129) NOT NULL COMMENT 'password',
 name     VARCHAR(120) NULL    ,
 PRIMARY KEY (id)
) COMMENT 'All users';
---
CREATE TABLE tb_user_external_project
(
 id     VARCHAR(200) NOT NULL COMMENT 'identifier of external project',
 user_id BIGINT      NOT NULL COMMENT 'unique identifier of the user',
 name    VARCHAR(120) NOT NULL COMMENT 'Name of external project',
 PRIMARY KEY (id, user_id)
) COMMENT 'External Project identifier for users';
```

Minimal Requirements

- Application should have at least basic auth
- Create a new user
- Retrieve user information
- Delete a user
- Add external project to a user
- Retrieve external projects from a user
- Write unit tests to ensure the correctness of the controller and service logic.
- Containerize project using docker

Optional requirements

- Update user information
- Configure logs
- Configure metrics
- Configure docker compose with database and service and all necessary ports to be tested