Backend developer interview exercise

Requirements

- Must be written in Typescript running on NodeJS 20+ runtime environment.
- The use of a framework is required.
    - **Fastify (preferred)**, Koa, Express
- The use of an ORM is required.
    - **Sequelize (preferred)**, TypeORM, Prisma, Mikro
- The following relational databases are allowed:
    - **MySQL (preferred)**, SQLite, MariaDB, Postgres
- Code needs to be linted, eslint with standardjs is preferred.
- Ensure the code is well-documented, follows best practices, and is scalable.
- Implement basic error handling and validation.
- Include unit tests for *ONLY critical* functionalities.
- *Optional: Docker can be used to set up the project's dependencies.*

Task Management

This assessment focuses on key aspects such as authentication, task management, filtering and tagging. It should demonstrate good code organisation, security, and scalability practices. The forethought made in how the assessment can be later developed to introduce new features.

Tasks

Task 1: User Authentication

Implement user authentication with the following requirements:

1. Users should be able to sign up with a unique email and password.
2. Passwords should be securely hashed and stored in the database.
3. Users should be able to log in with their email and password.
4. Implement token-based authentication for securing API endpoints.

Task 2: Task Management

Implement endpoints to manage tasks with the following features:

1. Create a new task with attributes such as title, description, due date, and priority.
2. Retrieve a list of all tasks assigned to the authenticated user.
3. Retrieve details of a specific task.
4. Update the details of an existing task.
5. Delete a task.
6. Assign a task to another user.
7. Update the status of a task (e.g., open, in progress, completed).

Task 3: Tagging Functionality

Implement the ability to tag tasks with labels with the following requirements:

1. Users should be able to tag a task with a set of labels.
2. The labels should be stored in a separate table, created on the fly

Task 4: Task Filtering and Sorting

Implement endpoints to filter and sort tasks with the following requirements:

1. Allow users to filter tasks based on status (e.g., open, in progress, completed).
2. Allow users to sort tasks based on due date or priority.

Submission:

1. Provide the source code in a version-controlled repository (e.g., GitHub).
2. Include a README file with instructions on how to set up and run the application.
3. Include a Postman or Insomnia collection with the APIs.
4. Include any additional documentation or comments that would be helpful.

Evaluation Criteria:

1. Correct implementation of all functionality.
2. Code organisation, readability, and adherence to best practices.
3. Scalability and performance considerations.
4. Secure handling of user data and authentication tokens.
5. The forethought made to additional functionalities that can be developed at a later date
6. Quality and coverage of unit tests.