PROTOTAL | UK

TECHNICAL CHALLENGE – TASK MANAGEMENT APP

OVERVIEW

In this technical challenge, you will be required to develop a simple task management application using Java, MongoDB, and the reactive programming model. The application will allow users to create, read, update, and delete tasks, which can contain nested sub-tasks.

USER STORY

As a user, I want to be able to store a list of tasks I need to complete. Each task will have a title and description, and could also contain sub-tasks., which themselves will have a title and description.

I want to be able to create, edit and delete my tasks. It would be ideal if I could also just delete sub-tasks without having to remove the entire task.

REQUIREMENTS

ENVIRONMENT

- Java 17 or higher
- MongoDB
- Spring Boot
- Reactive Mongo Template

SUGGESTED DOCUMENT STRUCTURE

```
{
    "_id": "ObjectId",
    "title": "Task Title",
    "description": "Task Description",
    "subTasks": [
        {
            "title": "Sub-Task Title 1",
            "description": "Sub-Task Description 1",
            "subTasks": [
                {
                    "title": "Nested Sub-Task Title",
                    "description": "Nested Sub-Task Description"
                }
            ]
        }
    ]
```

ENDPOINTS

Implement the following REST endpoints:

- **Create Task:** `POST /tasks`
- **Get Task by ID:** `GET /tasks/{id}`
- **Update Task:** `PUT /tasks/{id}`
- **Delete Task:** `DELETE /tasks/{id}`

IMPLEMENTATION DETAILS

- Use `ReactiveMongoTemplate` for interacting with the MongoDB database.
- Implement the endpoints in a reactive manner, using `mono` and `flux` from Project Reactor
- Code should be unit and integration tested

SUBMISSION

Please provide:

- Source code for the application, via GitHub
- Instructions for how to run the application, including information on setting up environment (MongoDB)
    - If possible, dockerise the application and provide a docker-compose file.

EVALUATION CRITERIA

- How complete and correct the implementation is.
- Code quality and use of best practice
- Proper usage of reactive programming principles
- Comprehensive unit and integration tests