Backend Engineer Technical Test: Python + FastAPI

Objective

Build a RESTful API to manage a simple task management system where users can create, update, and delete tasks. The API should follow best practices, be well-documented, and include test coverage.

Requirements

1. **Endpoints**
   - `POST /tasks/`: Create a new task.

Request Body:
json

```
{
    "title": "Task Title",
    "description": "Task Description",
    "priority": 1,
    "due_date": "2000-01-30T15:00:00"
}
```

Response:
json

```
{
    "id": 1,
    "title": "Task Title",
    "description": "Task Description",
    "priority": 1,
    "due_date": "2000-01-30T15:00:00",
    "completed": false
}
```

   - `GET /tasks/`: Retrieve all tasks.
     - **Query Parameters** (Optional):
       - `completed` (bool): Filter by completion status.
       - `priority` (int): Filter by task priority.

Response:
json

```
[
    {
        "id": 1,
        "title": "Task Title",
        "description": "Task Description",
        "priority": 1,
        "due_date": "2000-01-30T15:00:00",
        "completed": false
    }
]
```

  - `GET /tasks/{task_id}/`: Retrieve a specific task by ID.

Response:
json

```
{
    "id": 1,
    "title": "Task Title",
    "description": "Task Description",
    "priority": 1,
    "due_date": "2000-01-30T15:00:00",
    "completed": false
}
```

  - `PUT /tasks/{task_id}/`: Update an existing task.

Request Body (All fields optional):
json

```
{
    "title": "Updated Task Title",
    "description": "Updated Task Description",
    "priority": 2,
    "due_date": "2000-02-01T15:00:00",
    "completed": true
}
```

Response:
json

```
{
    "id": 1,
    "title": "Updated Task Title",
    "description": "Updated Task Description",
    "priority": 2,
    "due_date": "2000-02-01T15:00:00",
    "completed": true
}
```

   - `DELETE /tasks/{task_id}/`: Delete a task by ID.

Response:
json
```
{"message": "Task deleted successfully."}
```

2. **Database**
   - Use SQLite as the database.
   - Define a Task model with the following fields:
     - `id` (int): Primary key.
     - `title` (string): Task title (required).
     - `description` (string): Task description (optional).
     - `priority` (int): Task priority (1 = High, 2 = Medium, 3 = Low).
     - `due_date` (datetime): Due date for the task.
     - `completed` (bool): Completion status (default: False).
3. **Testing**
   - Write 3 unit tests for one endpoint using `pytest`.
   - In those three tests try to include edge cases, e.g., invalid input, non-existent task IDs, etc.
4. **Documentation**
   - Use FastAPI's auto-generated OpenAPI docs.
   - Add detailed docstrings for all endpoints.
5. **Code Quality**
   - Follow Python best practices (PEP 8).
   - Ensure code is modular and reusable.
6. **Bonus (Optional)**
   - Implement pagination for the `GET /tasks/` endpoint.
   - Add a search functionality to filter tasks by title or description.
   - Use Docker to containerize the application.

Submission Requirements

1. A GitHub repository containing:
    - Source code.
    - README file with:
        - Instructions on how to run the project.
        - Example API requests.
    - Instructions on running tests.
2. The application should be runnable locally inside a Docker container.

Evaluation Criteria

1. **Functionality**: Does the API meet the requirements and handle edge cases?
2. **Code Quality**: Is the code clean, modular, and well-documented?
3. **Testing**: Are there comprehensive unit tests?
4. **Documentation**: Is the API documentation clear and informative?

All the best, please feel free to contact us if you have any questions!

Email: jakub@sabermine.ai, mitch@sabermine.ai