Assignment Project

By GoKapture

July 22, 2024

Project Overview

Create a RESTful API for a task management system. The API should support the following functionalities:

- User Registration and Authentication
- CRUD operations for tasks (Create, Read, Update, Delete)
- Task assignment to users
- Filtering and searching tasks

Requirements

1. User Registration and Authentication

- Users should be able to register with a username and password.
- Passwords should be securely hashed.
- Users should be able to log in and receive a JWT token for authenticated requests.

2. Task Management

- Tasks should have the following fields: `id`, `title`, `description`, `status` (e.g., Todo, In Progress, Done), `priority`, `due_date`, `created_at`, `updated_at`, and `user_id` (to indicate the user the task is assigned to).
- Users should be able to create, read, update, and delete their tasks.
- Tasks should be assigned to users.

3. Filtering and Searching

- Implement filtering options for tasks based on `status`, `priority`, and `due_date`.
- Implement a search functionality to search tasks by `title` or `description`.

4. Dockerization

- Create a Dockerfile to containerize the application.
- Provide a `docker-compose.yml` file to set up the application along with any necessary services (e.g., database).
- Use of Dockerfile will be considered for additional points.

5. Database

- Use a relational database (e.g., PostgreSQL, MySQL) to store user and task data.
- Use an ORM (e.g., SQLAlchemy for Python, GORM for GoLang, TypeORM for TypeScript) for database operations.

Bonus Points

- Implement role-based access control (e.g., Admin and User roles).
- Implement pagination for task lists.
- Write unit and integration tests for the API endpoints.
- Set up a CI/CD pipeline (using GitHub Actions, Travis CI, etc.) to run tests and deploy the application.

Example Endpoints

1. User Registration

```
POST /api/register
{
    "username": "user1",
    "password": "password123"
}
```

2. User Login

```
POST /api/login
{
    "username": "user1",
    "password": "password123"
}
```

3. Create Task

```
POST /api/tasks
{
    "title": "New Task",
    "description": "Task description",
    "status": "Todo",
    "priority": "High",
    "due_date": "2024-07-31"
}
```

4. Get Tasks

```
GET /api/tasks
```

5. Update Task

```
PUT /api/tasks/{taskId}
{
    "title": "Updated Task",
    "description": "Updated description",
    "status": "In Progress",
    "priority": "Medium"
}
```

6. Delete Task

`DELETE /api/tasks/{taskId}`

Submission Guidelines

Candidates should submit:

- Source code in a GitHub repository.
- Instructions to run the application locally.
- API documentation (e.g., Swagger or Postman collection).
- A brief write-up on their approach and any assumptions made.