Coding Assignment: Build a Task Management API

Objective:
Design and implement a simple RESTful API for a task management system. The goal is to evaluate your problem-solving skills, code quality, and understanding of software development best practices.

Assignment Overview:
You are tasked with building a backend service for managing tasks. The service should support the following functionalities:

1. **Create a Task:** Add a new task with details such as title, description, due date, and status (e.g., Pending, In Progress, Completed).
2. **Retrieve Tasks:** Fetch a list of all tasks or filter tasks based on their status.
3. **Update a Task:** Modify task details, including changing its status.
4. **Delete a Task:** Remove a task from the system.

Requirements:

- Use a programming language of your choice (e.g., Python, Java, JavaScript/Node.js).
- Use a framework like Flask, Django, Spring Boot, or Express.js to set up the API.
- Data storage can be in-memory (using a dictionary/array) or with a database (e.g., SQLite, MongoDB).

API Specifications:

Method	Endpoint	Description	Parameters/Body
POST	`/tasks`	Create a new task	JSON: `{ "title": "Task1", "description": "Details", "due_date": "YYYY-MM-DD", "status": "Pending" }`

GET	`/tasks`	Retrieve all tasks or filter by status	Query params: `?status=Pending`
PUT	`/tasks/{id}`	Update a task's details	JSON: `{ "title": "Updated Title", "status": "Completed" }`
DELETE	`/tasks/{id}`	Delete a task by its ID	None

Expectations:

- Code should be clean, well-commented, and structured.
- Use proper HTTP status codes and error handling.
- Include basic validation for inputs (e.g., title is required, status must be valid).
- Implement pagination for fetching tasks (optional but encouraged).
- Include a README file with instructions on how to set up and run the project.

Submission Guidelines:

- Upload your code to a GitHub repository or submit it as a zip file.
- Ensure your README includes setup instructions and any assumptions made.

Evaluation Criteria:

- Functionality: Does the API perform the required operations?
- Code Quality: Is the code readable, maintainable, and modular?
- Error Handling: Are errors handled gracefully and appropriately?
- Bonus: Use of advanced features (e.g., authentication, testing, or deployment scripts).