

# TASK MANAGER API

**Author:** Keves

**Version:** 1.0

**Date:** 2025-11-26

## INTRODUCTION

### Purpose

The **Task Manager API** is a backend service designed to allow clients to Create, Read, Update, and Delete (CRUD) task records. Its purpose is to provide a reliable and structured interface for task management that can be integrated with web or mobile applications.

### Intended Audience and Use

- Audience: Developers, testers, and technical reviewers who will use or integrate the API.
- Use: Clients will interact with the API using HTTP requests to perform CRUD operations on tasks.

### Product Scope

- **Included:**

- CRUD operations on tasks (title, description, status, due\_date)
- Data persistence in SQL database (PostgreSQL/MySQL)
- User authentication

- **Excluded:**

- Frontend UI
- Notifications or scheduling beyond basic due dates
- Project-level task grouping beyond single tasks

### Definitions, Acronyms, Abbreviations

- API: Application Programming Interface
- CRUD: Create, Read, Update, Delete
- REST: Representational State Transfer

- DB: Database

## References

- Node.js v20+
- PostgreSQL / MySQL
- GitHub repository for version control

# OVERALL DESCRIPTION

## Product Perspective

This API is a standalone backend service intended to serve frontend applications or other services requiring task management functionality.

## Product Functions

- Create a new task
- Retrieve all tasks or a single task by ID
- Update a task's details
- Delete a task
- User registration and authentication

## User Classes and Characteristics

*This section describes user types interacting with the system. The design class structure is detailed in the design artifacts section.*

- **API Client / Developer:** Integrates API with applications; uses JSON for requests and responses.
- **Registered User:** Can perform CRUD operations only on their own tasks.

## Operating Environment / Dependencies / Constraints

- Node.js runtime
- SQL database available (PostgreSQL/MySQL)
- Internet access for GitHub/optional deployment

## Assumptions and Dependencies

- Database is operational and accessible
- Clients send valid JSON requests
- No high-load or enterprise-scale performance is required

# SYSTEM FEATURES AND REQUIREMENTS

## Functional Requirements

ID	REQUIREMENT	DESCRIPTION
FR-001	Create Task	POST /tasks with required "title" creates a new task; returns <b>201 Created</b> with task JSON including ID.
FR-001-E	Create Task - Invalid	Missing "title" > <b>400 Bad Request</b> with error message.
FR-002	Read All Tasks	GET /tasks returns <b>200 OK</b> with JSON array of all tasks.
FR-003	Read Single Task	GET /tasks/{id} returns <b>200 OK</b> with task JSON; invalid ID > <b>404 Not Found</b> .
FR-004	Update Task	PUT /tasks/{id} updates fields; returns <b>200 OK</b> with updated JSON. Invalid ID > <b>404</b> , invalid input > <b>400</b> .
FR-005	Delete Task	DELETE /tasks/{id} deletes task; returns <b>204 No Content</b> . Invalid ID > <b>404</b> .
FR-006	Authentication/Authorization	CRUD operations require valid token; invalid/missing > <b>401 Unauthorized</b> .

## External Interface Requirements

- API Endpoints: /tasks, /tasks/:id with standard HTTP methods (GET, POST, PUT, DELETE)
- Database: PostgreSQL/MySQL; schema defined in **Appendices**
- Configuration: Environment variables for database connection, auth secret

## Non-Functional Requirements

- **Performance:** Average response under 200ms
- **Reliability/Data Integrity:** Tasks must be persistently stored; no data loss under normal operation
- **Security:** Inputs sanitized; auth required if implemented
- **Maintainability:** Modular, documented code; version-controlled via GitHub
- **Scalability:** Database/server can be upgraded if usage grows

## **Other Requirements / Supplemental Sections**

### **Database Requirements**

- SQL relational database
- Tables: Tasks (id, title, description, status, due\_date, user\_id), Users (id, username, password hash, etc.)

### **Development Tools / Environment Constraints**

- Node.js v24+
- Git/GitHub for version control
- Postman or similar for testing endpoints

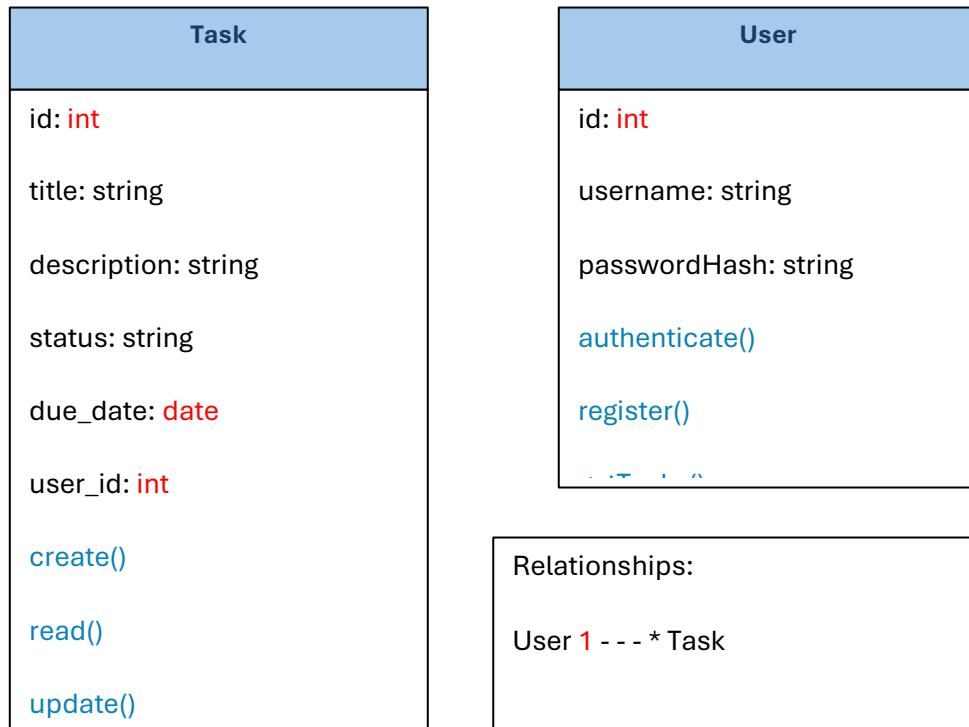
### **Limitations / Known Issues / Future Extensions**

- No frontend UI
- No advanced task filtering or pagination
- Future: add notifications, project grouping, and user roles

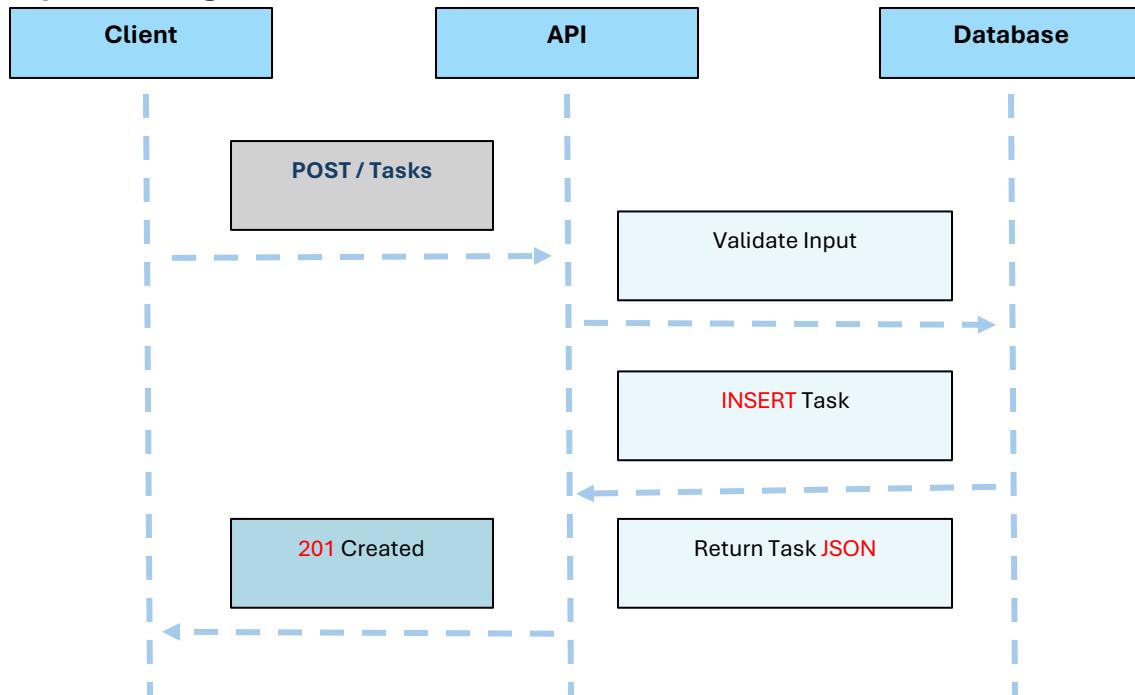
# Appendices

## Design Diagrams

### Class Diagram:



### Sequence Diagram:



## **Revision History**

<b>Version</b>	<b>Date</b>	<b>Author</b>	<b>Description</b>
1.0	11-26-2025	Keves	Initial draft.