📘 Software Requirements Specification (SRS) – Task Manager API

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the **Task Manager API**, which allows authenticated users to manage personal tasks securely using a RESTful interface. The API supports user authentication, task creation, updating, filtering, and deletion.

1.2 Scope

This project provides:

- User registration and login
- Secure authentication using JWT
- CRUD operations for tasks
- Filtering and organizing tasks
- Persistent storage using SQServer

The system will run as a backend API only (no UI), designed for learning and demonstration.

1.3 Definitions

- **API** — Application Programming Interface
- **JWT** — JSON Web Token
- **CRUD** — Create, Read, Update, Delete
- **Task** — A user-created item containing information like title and status

---

2. System Overview

The **Task Manager API** allows users to:

- Register and log in to receive a JWT token
- Manage personal tasks securely
- Filter tasks by category or status
- Modify or delete tasks individually

The API enforces strict authentication:
Every task belongs to one specific user, and only the owner can access or modify it.

---

3. Functional Requirements

3.1 User Management

FR-1: User Registration

- The system shall allow users to create an account using:
  - Username
  - Email
  - Password (hashed)

FR-2: User Login

- The system shall authenticate users via email + password.
- The system shall return a **JWT token** upon successful login.

FR-3: JWT Authentication

- The system shall protect all task-related endpoints using JWT.
- Only authenticated users with a valid token can access task resources.

3.2 Task Management

FR-4: Create Task

- Authenticated users can create a task with:
  - Title (required)
  - Description (optional)
  - Category (optional: work, personal, etc.)
  - Status (default: pending)

FR-5: View All Tasks

- The system shall return all tasks belonging to the authenticated user only.

FR-6: Filter Tasks

The system shall allow filtering tasks by:

- Status (completed, pending)
- Category (e.g., work, personal)

FR-7: View Single Task

- The system shall allow retrieving a task by its ID (only if owned by the user).

FR-8: Update Task

- The user can update:
  - Title
  - Description
  - Category
  - Status (completed/pending)

FR-9: Delete Task

- Authenticated users can delete a task by ID (only if owned by them).

---

4. Non-Functional Requirements

4.1 Security Requirements

- Passwords must be hashed before storage.
- JWT tokens must expire after a configured duration.
- Only task owners may access their tasks.

4.2 Performance

- API must respond within 500ms under normal load.
- Database operations must be optimized for small-scale usage.

4.3 Reliability

- The API shall not lose data after restart (SQLite persistence).
- Input validation must be enforced on all endpoints.

4.4 Maintainability

- Code must follow RESTful standards.
- Project structure must follow Flask blueprints or modular organization.

---

5. Database Requirements (For ERD)

Entities

1) User

Field	Type	Description
user_id (PK)	INTEGER	Unique user ID
username	TEXT	Username
email	TEXT	Unique email
password_hash	TEXT	Hashed password

---

2) Task

Field	Type	Description
task_id (PK)	INTEGER	Unique task ID
user_id (FK)	INTEGER	References User

Field	Type	Description
title	TEXT	Task title
description	TEXT	Task description
category	TEXT	(work, personal, etc.)
status	TEXT	pending/completed
created_at	DATETIME	Creation date
updated_at	DATETIME	Last update date

ERD Relationship

- **One User → Many Tasks**
- user_id in Task references user_id in User

Perfect for ERD drawing.

---

6. API Endpoints

6.1 Authentication

Method	Endpoint	Description
POST	`/auth/register`	Register new user
POST	`/auth/login`	Login & return JWT

6.2 Task Endpoints (Protected by JWT)

Method	Endpoint	Description
POST	`/tasks`	Create new task
GET	`/tasks`	Get all user tasks
GET	`/tasks?status=&category=`	Filter tasks
GET	`/tasks/<id>`	Get task by ID
PUT	`/tasks/<id>`	Update task
DELETE	`/tasks/<id>`	Delete task

---

7. Assumptions

- System serves one user base; no admin panel.
- Each user sees only their own tasks.
- Category values are free text unless predefined later.

---

8. Future Enhancements

- Task priority
- Subtasks
- Deadlines & reminders
- Email notifications
- Teams & shared tasks