Software Requirements Specification (SRS) for Task List

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional, non-functional, and technical requirements for the "Task List" project. This backend system will manage task-related operations using Flask, PostgreSQL, Celery, and Redis.

1.2 Document Overview

This document provides a comprehensive overview of the system's architecture, database structure, API endpoints, security, and performance expectations.

1.3 Intended Audience

- Developers
- Project Managers
- System Architects
- QA Engineers

1.4 Intended Use

This backend system will serve as the core of a task management application, handling user authentication, task creation, logging, and background task processing.

2. Overall Description

2.1 Product Perspective

The "Task List" backend is a standalone microservice that integrates with a frontend (if required) via REST APIs.

2.2 Product Functions

- User authentication and role-based access control (RBAC)
- Task creation, retrieval, updating, and deletion
- Background processing of scheduled tasks
- API rate limiting and caching

2.3 User Characteristics

- **Admin Users:** Manage all tasks and users
- **Regular Users:** Create, update, and track personal tasks

2.4 Constraints

- Must be containerized using Docker
- PostgreSQL as the primary database
- Redis for caching and Celery task queue

3. Specific Requirements

3.1 Functional Requirements

- **User Management:** JWT-based authentication, role management
- **Task Management:** CRUD operations on tasks
- **Task Logging:** Automatic logging of task status changes
- **Rate Limiting:** Prevent abuse of API calls
- **File Upload:** Upload CSV files for bulk task creation

3.2 Non-Functional Requirements

- **Performance:** API response time < 300ms
- **Scalability:** Support for horizontal scaling
- **Security:** JWT authentication, encrypted database connections
- **Availability:** 99.9% uptime with automated recovery

4. System Features

4.1 User Management

- Secure login and signup
- JWT authentication
- Role-based access control (Admin, User)

4.2 Task Management

- Create, edit, delete, and retrieve tasks
- Assign tasks to users
- Soft delete (mark inactive instead of hard deletion)

4.3 Task Logging

- Log task status changes in a separate table
- Auto-logging via Celery tasks

4.4 Security Features

- Rate limiting (200 requests per day, 50 per hour)
- Data encryption using PostgreSQL secure connections

4.5 Background Processing

- Celery worker to process background tasks
- Redis message queue for scalability

5. External Interfaces

5.1 API Endpoints

- **POST /upload-csv**: Bulk task upload
- **GET /tasks**: Fetch paginated task list
- **GET /tasks?date=YYYY-MM-DD**: Filtered task retrieval with caching
- **GET /task/<task_logger_id>**: Fetch single task details
- **POST /task**: Create a new task
- **PUT /task/<task_id>**: Update an existing task
- **DELETE /task/<task_id>**: Soft delete a task

5.2 Database Schema

- **User Table:** Stores user authentication details
- **TaskManager Table:** Stores task data
- **TaskLogger Table:** Logs task status updates

6. Performance Requirements

- Database queries optimized using indexes
- API response time within acceptable limits (<300ms)

7. Security & Compliance

- JWT authentication for all protected endpoints
- Input validation using Pydantic
- Database access secured via environment variables

8. Assumptions & Dependencies

- PostgreSQL database is hosted and available
- Redis and Celery worker are properly configured

- Docker is used for deployment