Task Manager(SRS)

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Project Title

Multi-Role Task Management System with Real-Time Collaboration

---

1. Introduction

1.1 Purpose

This document defines the functional and non-functional requirements for a **multi-role, project-based task management system** designed for SaaS-style deployment within organizations.
The system enforces **role-based access control (RBAC)**, supports **real-time collaboration**, and maintains **auditability** of task activity.

1.2 Scope

The system enables organizations to:

- Manage projects and tasks
- Assign responsibilities through hierarchical roles
- Track task lifecycle with history
- Share files at project and task level
- Collaborate in real time (chat, updates)
- Enforce strict access control without exposing sensitive data

The system is **desktop-first**, **production-grade**, and **scalable**.

---

2. Overall Description

2.1 Product Perspective

The application follows a **client-server architecture**:

- Frontend: React (component-based UI)
- Backend: Node.js + Express
- Database: MySQL
- Authentication: JWT (Access + Refresh)
- Communication: REST APIs + WebSockets

2.2 User Classes and Characteristics

Role	Description
Manager	Highest authority, manages projects and roles
Project Manager	Manages one project and its members
Member	Executes tasks

Constraints:

- A user has **exactly one global role**
- Roles are **global**, not per project
- Only Managers can promote roles
- No Viewer/Admin roles (intentional security choice)

2.3 Operating Environment

- OS: Linux (Ubuntu)
- Browser: Modern desktop browsers
- Backend runtime: Node.js (LTS)
- Database server: MySQL

2.4 Design Constraints

- JWT-based authentication required
- Role-based authorization enforced at API level

- MySQL must use prepared statements
- Local file storage for development
- WebSockets scoped per project
- No anonymous or public access

---

3. System Features & Functional Requirements

3.1 Authentication & Authorization

Description

The system authenticates users using email and password and authorizes actions using JWT and RBAC.

Functional Requirements

- Users shall log in using email and password
- System shall issue access and refresh tokens
- Refresh tokens shall be stored in database and HTTP-only cookies
- Access tokens shall expire after configurable duration
- System shall revoke refresh tokens on logout
- System shall enforce role-based permissions

3.2 User Management

Functional Requirements

- Managers shall promote Members to Project Managers
- Users shall have exactly one global role
- Users shall not self-promote or downgrade roles

3.3 Project Management

Functional Requirements

- Managers shall create projects
- Managers and Project Managers shall manage project members
- Project ownership shall be transferable by Managers

- Members shall not leave projects voluntarily
- A user may belong to only one project (except Managers)

3.4 Task Management

Functional Requirements

- Managers and Project Managers shall create tasks
- Each task shall be assigned to exactly one Member
- Assigned Member shall update task status
- Managers and Project Managers shall:
    - Change priority
    - Delete tasks
- All project members shall view all tasks

3.5 Task Status & Priority

Functional Requirements

- Task statuses shall be fixed initially
- Task priorities shall be fixed enums
- Status and priority definitions shall be configurable in future versions

---

3.6 Task History (Audit Trail)

Functional Requirements

- System shall log all task state changes
- History shall include:
    - Action type
    - Old and new values
    - User performing the action
    - Timestamp
- History shall be immutable

3.7 File Management

Functional Requirements

- Files shall be uploaded at:
    - Project level
    - Task level
- Any project member shall upload or delete files
- File metadata shall be stored in database
- Files shall be stored locally (dev)

3.8 Real-Time Collaboration

Functional Requirements

- System shall provide real-time project chat
- Task updates shall propagate in real time
- File uploads shall notify project members
- WebSocket connections shall be scoped per project

3.9 Comments System

Functional Requirements

- Users shall comment on tasks
- Comments shall be visible to all project members
- Comments shall be timestamped and attributed

---

4. Non-Functional Requirements

4.1 Security

- Passwords must be hashed (bcrypt)
- No sensitive data in client storage
- JWT secrets must be environment-based
- SQL injection prevention mandatory

4.2 Performance

- API responses < 300ms under normal load
- WebSocket updates < 200ms latency

- Connection pooling required

4.3 Scalability

- Stateless backend
- Modular services
- Cloud-ready architecture

4.4 Reliability

- Graceful error handling
- Centralized logging
- Consistent API responses

4.5 Maintainability

- Modular codebase
- Clear separation of concerns
- Strict linting and formatting

---

5. External Interface Requirements

5.1 User Interface

- Desktop-first responsive design
- Kanban board for tasks
- Dedicated task detail pages
- Role-aware UI controls

5.2 APIs

- REST APIs for CRUD
- WebSocket APIs for real-time features
- Versioned endpoints (`/api/v1`)

5.3 Database

- MySQL with normalized schema
- Foreign key constraints enforced
- Audit tables preserved

---

6. Future Enhancements

- Configurable task statuses
- Notifications system
- Analytics dashboard
- Multi-tenant SaaS onboarding
- Cloud file storage
- Organization-level settings

---

7. Approval & Versioning

Version	Description
v1.0	Initial production-grade SRS

Eumns

Below are finalized, production-grade enums for Task Status and Task Priority, designed to be:

- **Clear for users**
- **Strict for backend validation**
- **Extensible for future configurability**
- **Suitable for Kanban + audit logs**

These will now be treated as LOCKED v1 enums in the SRS.

✅ TASK STATUS ENUM (v1)

Enum Name

`task_status`

Values (Ordered Workflow)

Status	Meaning	Who can set
`backlog`	Task exists but not scheduled	Manager, Project Manager
`todo`	Ready to be worked on	Manager, Project Manager
`in_progress`	Actively being worked on	Assigned Member

`blocked`	Cannot proceed due to dependency	Assigned Member
`in_review`	Work completed, awaiting review	Assigned Member
`done`	Task completed and approved	Manager, Project Manager

Key Design Decisions

- Explicit `backlog` → supports planning
- `in_review` → enterprise workflow signal
- Assigned member cannot mark `done`
- Prevents silent task completion

SQL Definition

```
ENUM(
  'backlog',
  'todo',
  'in_progress',
  'blocked',
  'in_review',
  'done'
)
```

✅ TASK PRIORITY ENUM (v1)

Enum Name

`task_priority`

Values

Priority	Meaning
`low`	No urgency
`medium`	Normal work
`high`	Time-sensitive
`critical`	Immediate attention required

SQL Definition

```
ENUM(
  'low',
  'medium',
  'high',
  'critical'
)
```

✅ BACKEND ENFORCEMENT RULES (MANDATORY)

These are not optional.

Status Transitions (v1 Rules)

From	Allowed To
backlog	todo
todo	in_progress
in_progress	blocked, in_review
blocked	in_progress
in_review	done, in_progress
done	❌ (immutable)

➡ **Enforced in backend service layer**
➡ **Logged in `task_history`**

Role Constraints

- **Assigned Member**
  - **Can move between:**

        - `todo → in_progress → blocked/in_review`
- **Manager / Project Manager**
    - **Can move task to:**
        - `done`
        - `backlog`
        - **Override any state if required**

✅ DATABASE COLUMN DEFINITIONS

status ENUM(

'backlog',

'todo',

'in_progress',

'blocked',

'in_review',

'done'

) NOT NULL DEFAULT 'backlog',

priority ENUM(

'low',

'medium',

'high',

'critical'

) NOT NULL DEFAULT 'medium'

✅ FRONTEND USAGE (IMPORTANT)

Kanban Columns (Initial)

Backlog | To Do | In Progress | Blocked | In Review | Done

UI Rules

- **Drag-and-drop restricted by role**
- **Invalid transitions blocked client-side AND server-side**
- **Status labels centralized in constants file**