# Software Requirements Specification for Task Management System

**Version 1.1**

Prepared for: Flutter Development Project

Date: August 8, 2025

## 1. Introduction

### 1.1. Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the Task Management System, Version 1.1. This document is intended to provide a detailed overview of the system, its features, and its operational capabilities to guide the project's development, design, and testing phases.

### 1.2. Document Conventions

This document follows the IEEE Std 830-1998 standard for SRS documentation. Functional requirements are uniquely identified with the prefix REQ- for traceability.

### 1.3. Intended Audience and Reading Suggestions

This document is intended for:

* **Developers:** To understand what to build.
* **Project Managers:** To manage the project scope and plan.
* **QA/Testers:** To create test cases.
* **Stakeholders:** To understand the capabilities of the final product.

It is recommended to read the document sequentially, starting with the overall description and then moving to the specific system features.

### 1.4. Product Scope

The Task Management System is a comprehensive mobile application designed to help individuals and teams organize, track, and collaborate on tasks and projects. The system will provide robust features for task management, role-based access control within projects, real-time notifications, and advanced capabilities such as calendar integration, file attachments, and reporting. The primary goal is to enhance productivity and streamline workflow management for its users.

### 1.5. References

No external documents are referenced in this SRS.

## 2. Overall Description

### 2.1. Product Perspective

The product is a new, self-contained system consisting of a mobile client application built with Flutter (for iOS and Android) and a dedicated server-side backend application. It is designed from the ground up to serve the needs of users requiring a modern and collaborative task management solution.

### 2.2. Product Functions

The major functions of the Task Management System are:

* Secure user registration, authentication, and profile management.
* Comprehensive creation, updating, deletion, and tracking of tasks and subtasks.
* Project-based collaboration with role-based access control (Leader, Worker, Viewer).
* Powerful tools to search, sort, and filter tasks.
* Real-time notifications for critical events and deadlines.
* Advanced features including calendar views, file attachments, analytics dashboards, and UI theming.

### 2.3. User Classes and Characteristics

* **Project Leader:** This user has full administrative control over a project. They can create/edit the project, manage members and their roles, and perform all task-related operations. This role is typically for project managers or team leads.
* **Worker:** This user is a standard team member. They can be assigned tasks, update the status of their assigned tasks, comment, and view all project tasks. They have no administrative privileges.
* **Viewer:** This user has read-only access to a project. They can view tasks and comments but cannot make any changes. This role is suitable for stakeholders or clients who need to monitor progress.

### 2.4. Operating Environment

* **Frontend (Client):** The mobile application will run on:
  + iOS (Version 13.0 and newer)
  + Android (Version 6.0 - Marshmallow and newer)
* **Backend (Server):** The backend will be developed to run on a standard cloud hosting environment (e.g., AWS, Google Cloud, Azure) with a Node.js/Python/Java runtime and a corresponding database (e.g., PostgreSQL, MongoDB).

### 2.5. Design and Implementation Constraints

* The mobile application **must** be developed using the Flutter framework.
* The backend **must** expose a RESTful API for communication with the client.
* User authentication **must** be handled using JSON Web Tokens (JWT).
* All communication between client and server **must** be encrypted using HTTPS.

### 2.6. User Documentation

The following user documentation will be provided:

* In-app Onboarding Guide
* Online Help Center / FAQ

### 2.7. Assumptions and Dependencies

* Users must have a stable internet connection for real-time collaboration and data synchronization.
* The notification system is dependent on third-party services like Firebase Cloud Messaging (FCM) or Apple Push Notification Service (APNS).

## 3. External Interface Requirements

### 3.1. User Interfaces

The application will feature a modern, clean, and intuitive user interface designed for mobile platforms. Key screens will include:

* Authentication (Login, Register, Forgot Password)
* Dashboard (Task summary)
* Project List & Detail View
* Task List & Detail View
* Project Member Management Screen
* Calendar View
* Analytics Dashboard
* User Profile & Settings

### 3.2. Hardware Interfaces

The application will interface with standard mobile device hardware, including:

* Touchscreen for user input.
* Network interface (Wi-Fi, Cellular) for communication with the backend.

### 3.3. Software Interfaces

* **Backend API:** The Flutter application will communicate with the backend via a RESTful API over HTTPS.
* **Push Notification Services:** The backend will interface with Firebase Cloud Messaging (FCM) and Apple Push Notification Service (APNS) to send notifications to mobile clients.

### 3.4. Communications Interfaces

* **Protocol:** All client-server communication will use the HTTPS protocol to ensure data encryption and security.

## 4. System Features

### 4.1. User Management & Authentication

* **4.1.1. Description:** Provides users with secure access to the system and management of their personal information.
* **4.1.2. Functional Requirements:**
  + **REQ-001:** The system shall allow a user to register with a unique email and a password.
  + **REQ-002:** The system shall validate user credentials and grant access upon successful login.
  + **REQ-003:** The system shall provide a secure logout mechanism.
  + **REQ-004:** The system shall allow users to view and update their name and profile picture.
  + **REQ-005:** The system shall provide a mechanism for users to reset their forgotten password via email.

### 4.2. Task Management

* **4.2.1. Description:** Core functionality for creating and managing individual tasks and subtasks.
* **4.2.2. Functional Requirements:**
  + **REQ-006:** The system shall allow users to create a task with a title, description, due date, and priority.
  + **REQ-007:** The system shall display tasks in a list view.
  + **REQ-008:** The system shall allow users to modify all editable fields of an existing task.
  + **REQ-009:** The system shall allow users to permanently delete a task.
  + **REQ-010:** The system shall allow users to change the status of a task (e.g., To-Do, In Progress, Completed).
  + **REQ-011:** The system shall allow a task to be broken down into a checklist of subtasks.

### 4.3. Collaboration and Role-Based Access

* **4.3.1. Description:** Enables team collaboration within projects using a defined set of user roles and permissions.
* **4.3.2. Functional Requirements:**
  + **REQ-012:** The system shall allow a user to create a project, which designates them as the Project Leader.
  + **REQ-013:** The Project Leader shall be able to invite other registered users to the project.
  + **REQ-014:** The Project Leader shall be able to assign a role (Worker, Viewer) to each project member.
  + **REQ-015:** The Project Leader shall be able to remove a member from a project.
  + **REQ-016:** The Project Leader shall be able to assign a task to any user with the 'Worker' role in the project.
  + **REQ-017:** All project members shall be able to add comments to tasks.

### 4.4. Search and Filtering

* **4.4.1. Description:** Provides tools for users to easily find and organize tasks.
* **4.4.2. Functional Requirements:**
  + **REQ-018:** The system shall provide a text search feature to find tasks by title or description.
  + **REQ-019:** The system shall allow users to filter the task list by status, priority, and due date range.
  + **REQ-020:** The system shall allow users to sort the task list by creation date, due date, and priority.

### 4.5. Notifications

* **4.5.1. Description:** Proactively informs users of important events and deadlines via push notifications.
* **4.5.2. Functional Requirements:**
  + **REQ-021:** The system shall send a push notification to a user 24 hours before a task's due date.
  + **REQ-022:** The system shall send a push notification when a task is assigned to a user.
  + **REQ-023:** The system shall send a push notification when a comment is added to a task the user is assigned to or has created.

### 4.6. Calendar Integration

* **4.6.1. Description:** Provides a visual representation of tasks and their deadlines over time.
* **4.6.2. Functional Requirements:**
  + **REQ-024:** The system shall display a monthly/weekly calendar view.
  + **REQ-025:** Tasks with due dates shall be displayed as events on the calendar.
  + **REQ-026:** Users shall be able to tap on a calendar event to view the corresponding task details.

### 4.7. File Attachments

* **4.7.1. Description:** Allows users to attach relevant files and documents to tasks.
* **4.7.2. Functional Requirements:**
  + **REQ-027:** The system shall allow users to upload one or more files to a task from their device.
  + **REQ-028:** The system shall display a list of attached files within the task details view.
  + **REQ-029:** The system shall allow users to download or view attached files.

### 4.8. Reporting & Analytics

* **4.8.1. Description:** Provides insights into project and user productivity through visual dashboards.
* **4.8.2. Functional Requirements:**
  + **REQ-030:** The system shall provide a dashboard view with charts visualizing project progress.
  + **REQ-031:** The dashboard shall display metrics such as tasks completed vs. pending, and tasks completed on time vs. overdue.

## 5. Other Nonfunctional Requirements

### 5.1. Performance Requirements

* API response times for all standard operations must be under 500ms.
* The mobile application UI must maintain a smooth performance of 60 frames per second during animations and scrolling.

### 5.2. Safety Requirements

* There are no specific safety requirements for this application.

### 5.3. Security Requirements

* User passwords must be hashed and salted before being stored in the database.
* The system must be protected against common web vulnerabilities (e.g., SQL Injection, Cross-Site Scripting).
* Role-based access control must be strictly enforced on the backend for all data modification requests.

### 5.4. Software Quality Attributes

* **Usability:** The application should be intuitive and easy to use for non-technical users, featuring a clean layout and clear navigation.
* **Reliability:** The application should be stable, with a target uptime of 99.9%, and handle network errors gracefully.
* **Maintainability:** The code should be well-structured, commented, and follow best practices to facilitate future updates and bug fixes.
* **Portability:** As a Flutter application, the codebase should be portable across iOS and Android platforms with minimal platform-specific code.

### 5.5. Business Rules

* A user who creates a project is automatically assigned the 'Leader' role for that project.
* Only a 'Leader' can assign tasks and manage project members.
* A task's due date cannot be in the past at the time of creation.

## 6. Other Requirements

This section is reserved for any requirements not covered elsewhere, such as internationalization, legal, or compliance requirements. None have been identified at this time.

## Appendix A: Glossary

* **API (Application Programming Interface):** A set of rules and protocols for building and interacting with software applications.
* **JWT (JSON Web Token):** A compact, URL-safe means of representing claims to be transferred between two parties. Used for authentication.
* **Project:** A workspace that contains a collection of related tasks and members.
* **Project Leader:** The user with full administrative rights over a project.
* **Task:** A single unit of work to be completed.
* **Viewer:** A user with read-only access to a project.
* **Worker:** A user who can be assigned tasks and can update their status.

## Appendix B: Analysis Models

This section is reserved for analysis models, such as Use Case diagrams, Class diagrams, or Data Flow diagrams, which will be developed during the design phase.