Software Requirements Specification (SRS)

Project: Todo App

Version: 1.0

Date: August 07, 2025

# 1. Introduction

## 1.1 Purpose

To help users manage tasks efficiently.

## 1.2 Scope

- Task management

## 1.3 Intended Audience

- Students

- Professionals

- General users

# 2. System Overview

A simple and intuitive to-do list app designed to help users manage tasks efficiently.

# 3. Functional Requirements

## 3.1 Task Creation [0]

**Description:**

Users can create new tasks with a title, description, and optional due date and reminder.

**Priority:**

High

**Subtasks:**

- Allow users to input a title for the task

- Enable users to add a description for the task

- Provide an option to set a due date for the task

- Implement a reminder feature for tasks with optional date and time settings

## 3.2 Task Editing [1]

**Description:**

Users can modify existing tasks, updating their title, description, due date, reminder, or status.

**Priority:**

High

**Subtasks:**

- Allow users to change the task title.

- Enable users to modify the task description.

- Provide a mechanism for users to update the due date of a task.

- Implement functionality for setting and modifying task reminders.

- Allow users to change the status of a task (e.g., Pending, In Progress, Completed).

## 3.3 Task Deletion [2]

**Description:**

Users can remove tasks from their list.

**Priority:**

High

**Subtasks:**

- Implement a delete button for each task.

- Display a confirmation dialog before deleting a task.

- Remove the task from the local storage/database upon confirmation.

## 3.4 Task Scheduling [3]

**Description:**

Users can assign due dates and set reminders for tasks.

**Priority:**

High

**Subtasks:**

- Implement due date input field for each task.

- Allow users to set custom reminders for tasks.

- Display reminders to users based on set due dates and reminder times.

## 3.5 Task Categorization [4]

**Description:**

Users can categorize tasks using tags or labels.

**Priority:**

High

**Subtasks:**

- Implement a mechanism for users to add tags/labels to tasks.

- Allow users to search and filter tasks by tags/labels.

- Display categorized tasks visually (e.g., color-coded labels).

## 3.6 Task Display [5]

**Description:**

The app provides a visual interface to display tasks, including a calendar view and a list view.

**Priority:**

High

**Subtasks:**

- Implement a calendar view to display tasks based on due dates.

- Implement a list view to display tasks chronologically or by priority.

- Allow users to filter tasks by date, priority, or status.

## 3.7 Task Completion [6]

**Description:**

Users can mark tasks as complete, indicating they have been finished.

**Priority:**

High

**Subtasks:**

- Implement a UI element to mark a task as complete.

- Persist the completion status of a task in the database.

- Update the UI to visually reflect the completion status of a task.

## 3.8 Data Synchronization [7]

**Description:**

The app synchronizes task data across multiple devices using cloud storage.

**Priority:**

High

**Subtasks:**

- Implement real-time data syncing using WebSockets.

- Store task data securely in a cloud database (e.g., AWS, Firebase).

- Ensure data integrity and conflict resolution during synchronization.

- Provide offline capabilities for task management.

-

## 3.9 User Authentication [8]

**Description:**

Users can log in securely to access their tasks and account information.

**Priority:**

High

**Subtasks:**

- Implement secure password hashing

- Provide login and registration functionality

- Handle user session management

- Implement password recovery mechanism

## 3.10 User Registration [9]

**Description:**

The app supports a user registration process, allowing new users to create accounts.

**Priority:**

High

**Subtasks:**

- Implement user registration form with required fields (username, email, password)

- Validate user input for email format and password strength

- Store user data securely in a database

- Generate a unique user ID upon registration

- Send a confirmation email to the registered user

## 3.11 Data Security [10]

**Description:**

The app maintains user data securely and protects it from unauthorized access.

**Priority:**

High

**Subtasks:**

- Implement secure password hashing using bcrypt or a similar algorithm.

- Store user data in an encrypted database.

- Offer multi-factor authentication (MFA) for added security.

- Regularly perform security audits and penetration testing.

## 3.12 Reminder Notifications [11]

**Description:**

The app provides notifications to remind users of upcoming deadlines.

**Priority:**

High

**Subtasks:**

- Implement a system for scheduling notifications based on task deadlines.

- Integrate with a notification service (e.g., push notifications, email, in-app alerts).

- Allow users to customize notification preferences (e.g., frequency, method).

- Display notification history for users to track past reminders.

## 3.13 User Interface Responsiveness [12]

**Description:**

The app is designed to be responsive and user-friendly on various devices and screen sizes.

**Priority:**

High

**Subtasks:**

- Ensure smooth scrolling and responsiveness on mobile devices.

- Optimize layout for different screen widths (landscape and portrait)

- Implement adaptive font sizes based on screen size.

- Validate responsiveness across various browsers and operating systems.

# 4. Non-Functional Requirements

**sync**: Tasks should sync across devices using cloud storage.

**concurrency**: The system should support at least 10,000 users simultaneously.

**responsiveness**: The app should be responsive and mobile-friendly.