**Project Name**

**To-Do List Application with Task Prioritization**

**Software Requirements Specification**

**Course Code :** INT219 & INT220

**Course Name:**1)FRONT END WEB DEVELOPER

2) SERVER SIDE SCRIPTING

**Student Name:**Salunkhe Aditya Sudhakar

**Student Registration Number:**12301138

Prepared for

Continuous Assessment 3

Spring 2025

**Table of Contents**

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.5 Non-Functional Requirements 3

3.5.1 Performance 3

3.5.2 Reliability 3

3.5.3 Availability 3

3.5.4 Security 3

3.5.5 Maintainability 3

3.5.6 Portability 3

3.7 Design Constraints 3

3.9 Other Requirements 3

4. Analysis Models 4

4.1 Data Flow Diagrams (DFD) 4

5. Github link…………………………………………………………………………………………………….5

**6. DEPLOYED LINK………………………………………………………………………………………………...6**

7. CLIENT APPROVAL PROOF……………..…………………………………………………………………….7

**8. CLIENT LOCATION PROOF……………………………………………………………………………………8**

**9. TRANSACTION ID PROOF……………………………………………………………………………………...9**

**10. EMAIL ACKNOWLEDGEMENT…………………………………………………………………………….10**

**11. GST No…………………………………………………………………………………………………………...11**

A. Appendices

A.1 Appendix 1

A.2 Appendix 2

# 1. Introduction

## This Software Requirements Specification (SRS) document defines the requirements for the development of a To-Do List Application with Task Prioritization. The document serves as a formal agreement between the development team and stakeholders by detailing the functionality, constraints, and performance criteria of the system. The goal of this application is to provide users with a simple yet powerful tool to manage their tasks efficiently, helping them stay organized and meet their deadlines effectively. This document follows the IEEE standard format to ensure clarity, completeness, and traceability of requirements throughout the development lifecycle.

## 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to detail the requirements for the development of a "To-Do List Application with Task Prioritization". It is intended for software developers, testers, project managers, and end users. This document outlines the functional and non-functional requirements to ensure the application meets user needs for organizing, prioritizing, and tracking tasks effectively..

## 1.2 Scope

## The software product to be developed is a To-Do List Application with Task Prioritization.

## The application allows users to:

## Add, edit, delete, and mark tasks as complete.

## Assign priority levels to tasks (e.g., High, Medium, Low).

## Set deadlines and receive timely reminders.

## View tasks in an organized and filtered manner.

## This application does not include advanced features like collaboration, calendar sync, or AI-based recommendations in its initial version.

## It will benefit users who require a simple, intuitive tool to stay organized and productive by helping them focus on important tasks and manage their time better.

## 1.3 Definitions, Acronyms, and Abbreviations

SRS: Software Requirements Specification

UI : User Interface

CRUD : Create, Read, Update, Delete

Task : A unit of work defined by the user

Priority : Level of importance assigned to a task (e.g., High, Medium, Low)

## 1.4 References

 IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

 Course materials from [Insert Course Name and Instructor Name]

 Online documentation on UX principles for task management tools.

## 1.5 Overview

This subsection should:

(1) Describe what the rest of the SRS contains

(2) Explain how the SRS is organized.

# 2. General Description

## This section provides an overview of the system and the general factors that affect the product and its requirements. It gives context to the specific requirements outlined in later sections and helps establish a clear understanding of the product's environment, purpose, and limitations.

## 2.1 Product Perspective

## The To-Do List Application with Task Prioritization is a standalone software product designed to assist users in efficiently managing their daily tasks. The system is designed to function on both desktop and mobile platforms and does not depend on any pre-existing system. It follows a user-centered design to ensure a smooth and intuitive experience.

## 2.2 Product Functions

## The key functions of the application include:

## Creating and managing tasks

## Assigning priority levels (High, Medium, Low)

## Setting deadlines and due dates

## Sending reminders and notifications

## Marking tasks as complete or pending

## Categorizing tasks into projects or lists

## Filtering and sorting tasks by date, priority, or status

## 2.3 User Characteristics

## The target users of this system are individuals or professionals who want to manage their time and tasks effectively. Users may have basic to intermediate technical knowledge. The interface is designed to be user-friendly and does not require prior experience with similar applications.

## 2.4 General Constraints

## The application must be responsive and accessible across various devices (mobile, tablet, desktop).

## Development will be constrained by the technologies chosen (e.g., React, Node.js, MongoDB).

## The system must be designed with scalability in mind for potential future enhancements.

## 2.5 Assumptions and Dependencies

# It is assumed that users will have access to a stable internet connection for syncing tasks and receiving notifications.

# The application depends on third-party services (e.g., Firebase or email APIs) for push notifications and authentication.

# It is assumed that users will have modern web browsers or updated mobile OS versions.

# 3. Specific Requirements

## This section details the software requirements necessary to design, build, test, and deploy the To-Do List Application with Task Prioritization. Each requirement is presented in a structured and traceable manner to ensure clarity and completeness.

## 3.1 External Interface Requirements

## 3.1.1 User Interfaces

## The system will feature a clean and intuitive interface.

## Users can add, edit, and delete tasks through forms.

## Drag-and-drop functionality will be provided for reordering tasks by priority.

## Users will have access to filtering and sorting options.

## 3.1.2 Hardware Interfaces

## The application is designed for general use and does not require specialized hardware.

## Runs on any device (desktop, tablet, mobile) with a modern web browser.

## 3.1.3 Software Interfaces

## Backend APIs built using Node.js/Express.

## Frontend developed in React.

## Database interface with MongoDB.

## Third-party APIs for email notifications and reminders (e.g., Firebase, SendGrid).

## 3.1.4 Communications Interfaces

## HTTPS protocol for secure communication between client and server.

## JSON format for data exchange.

## WebSockets (optional) for real-time updates.

**3.2 Functional Requirements**

**3.2.1 Task Management**

**3.2.1.1 Introduction**

This feature allows users to manage their tasks effectively.

**3.2.1.2 Inputs**

* Task title, description, deadline, priority level.

**3.2.1.3 Processing**

* Create, update, delete, view, and reorder tasks.

**3.2.1.4 Outputs**

* Display updated task list and changes in real-time.

**3.2.1.5 Error Handling**

* Input validation errors, server errors handled with appropriate user alerts.

**3.2.2 Reminders & Notifications**

**3.2.2.1 Introduction**

Allows users to receive reminders before task deadlines.

**3.2.2.2 Inputs**

* Task deadline, notification preferences.

**3.2.2.3 Processing**

* System schedules and sends notifications at user-specified times.

**3.2.2.4 Outputs**

* Push/email notification.

**3.2.2.5 Error Handling**

* Handles failures in notification services gracefully.

…

## 3.5 Non-Functional Requirements

## 3.5.1 Performance

## 95% of user requests should be processed in under 1 second.

## The application should support at least 100 concurrent users.

## 3.5.2 Reliability

## The system should experience less than 0.5% error rate in task operations.

## It should recover automatically from minor crashes or connectivity losses.

## 3.5.3 Availability

## The application must be accessible 24/7 except during planned maintenance.

## 3.5.4 Security

## Passwords are hashed using a secure algorithm.

## Users cannot access data of other users (authorization enforced).

## Input fields are protected against XSS, CSRF, and SQL/NoSQL injection.

## 3.5.5 Maintainability

## Code must be modular with separation of concerns.

## Documentation and inline comments will be maintained to ease future updates.

## 3.5.6 Portability

## The application should work seamlessly across Chrome, Firefox, Edge, and Safari.

## It must run on both Windows and macOS without platform-specific dependencies.

## 3.7 Design Constraints

## Technological Stack: The backend will be implemented using PHP, with an appropriate framework (such as Laravel or Symfony) to ensure proper routing, data management, and security.

## Frontend: The frontend will be developed using standard HTML, CSS, and JavaScript, along with any JavaScript libraries or frameworks such as jQuery or Bootstrap for responsive design and enhanced user interaction.

## Security: The application must follow security best practices such as input validation, CSRF protection, and password encryption (using PHP’s password\_hash()).

## Performance: The application must be optimized for performance, ensuring that tasks load in under 3 seconds, even with a larger number of tasks.

## Compatibility: The application must be compatible across modern browsers, including Chrome, Firefox, Safari, and Edge.

## 3.9 Other Requirements

# User Authentication: The application should include a login system that allows users to create an account, log in securely, and manage their tasks. Sessions and cookies will be used for maintaining user login states.

# Task Reminders: Users will be able to set reminders for tasks, with notifications sent via email or through the application interface.

# Data Persistence: All tasks, their priorities, deadlines, and user information will be stored in a MySQL database, which the PHP backend will manage.

# Backup and Restore: Users should be able to export their tasks and import them back in case of data loss.

# Internationalization: The application should initially support English but will be structured to support future translations (e.g., adding more language support).

# 4. Analysis Models

## Use Case Diagrams:

## Use case diagrams will describe the interaction between users and the system. For instance, the "Add Task" use case would show how users can input tasks, set deadlines, and assign priorities.

## Entity-Relationship Diagram (ERD):

## The ERD will define how tasks, users, and reminders are stored in the MySQL database. The Task entity will have attributes such as task\_id, task\_name, priority, deadline, etc., and will relate to the User entity and Reminder entity.

## Data Flow Diagram (DFD):

## Level 0 DFD: This diagram will show the core processes such as "Add Task," "View Task List," and "Set Reminder."

## Level 1 DFD: This will break down the processes like adding a task into more detail, including user input, database operations, and UI updates.

# A. Appendices

**A.1 Appendix 1: Wireframe for To-Do List Application**

* A screenshot of a login form

  AI-generated content may be incorrect.
* A screenshot of a login form

  AI-generated content may be incorrect.
* A screenshot of a computer

  AI-generated content may be incorrect.
* A screenshot of a computer

  AI-generated content may be incorrect.
* A screenshot of a computer

  AI-generated content may be incorrect.

**A.2 Appendix 2: Initial Client Feedback**

* **Sample .env Structure:**
* GOOGLE\_CLIENT\_ID=your\_google\_client\_id
* GOOGLE\_CLIENT\_SECRET=your\_google\_client\_secret
* MONGO\_URI=your\_mongodb\_connection\_string
* SESSION\_SECRET=your\_session\_secret
* **Deployment Guide for Render or Vercel:**
* Push code to a GitHub repository.
* Create a new Web Service on <https://render.com>.
* Connect your GitHub repo.
* Add environment variables as per .env structure.
* Select Node environment and auto deploy.
* Alternatively, use Vercel CLI for frontend if separated.