**SRS**

**Table of Contents**

1. **Introduction**
   1. Purpose
   2. Scope
   3. Definitions and Abbreviations
   4. Overview
2. **General Description**
   1. Product Perspective
   2. Product Function
   3. Hardware Interface
   4. Software Interface
   5. Communication Interface
   6. General Constraints
   7. Technologies used
3. **Specific Requirements**
   1. Functional Requirements
   2. Non-Functional Requirements
4. **Introduction**
   1. **Purpose**

The purpose of this document is to present a detailed description of our project. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it is intended for both the user and the developers of the system.

* 1. **Scope**

The To-Do List is a simple web-based application that allows users to create, edit, mark as completed, and delete tasks. The application provides an interactive interface for task management and stores tasks using local storage to maintain persistence.

* 1. **Abbreviations and Definition**
* HTML: Hypertext Markup Language
* CSS: Cascading Style Sheets
* JS: JavaScript
* LocalStorage: Web storage API for persisting data on the client side

**1.4 Overview**

This document outlines the functional and non-functional requirements of the To-Do List project, describing its structure, user characteristics, constraints, and technology stack.

1. **Overall Description**
   1. **Product Perspective**

**2.1.1 Product function**

The To-Do List application provides the following functions:

1. Users can add new tasks.
2. Users can edit existing tasks.
3. Users can mark tasks as completed.
4. Users can delete tasks.
5. The application maintains task data using local storage.

* + 1. **Hardware Interface**

**Client side:**

Processor: Pentium P2 onwards

RAM: 256Mb or more

Hard disk: 128Mb or more

* + 1. **Software Interface**

**Client side:**

Operating system: Any OS system compatible

Browser: Any browser compatible with IE 5.0 or onwards

* + 1. **Communication Interface**

HTTP, HTTPS, TCP/IP, LocalStorage

* 1. **General Constraints**
* 24/7 hours available.
  1. **Technologies used**

Frontend: HTML, CSS, JavaScript , FontAwesome(for icons)

Backend: LocalStorage (Client-Side Storage)

1. **Specific Requirements**
   1. **Functional –requirements**
2. Add Task: Users can input a task and add it to the list.
3. Edit Task: Users can modify an existing task.
4. Mark Task as Completed: Users can mark a task as completed, which visually differentiates it.
5. Delete Task: Users can remove a task from the list.
6. Data Persistence: The application stores tasks in LocalStorage for persistence across sessions.
   1. **Non-Functional Requirements**
7. Availability: The application should be accessible 24/7 as long as the browser supports JavaScript and LocalStorage.
8. Security: Data is stored on the client side, making it vulnerable to local storage manipulation.
9. Reliability: The system should function properly across different browsers and screen sizes.
10. Maintainability: The system should be easy to modify and expand with additional features in the future.
11. Top of Form

.