# To-Do List Project Documentation

## Introduction

This project is a To-Do List application divided into two sections:

1. Daily To-Do List

2. Weekly To-Do List

The project is built using HTML, CSS, and JavaScript while applying SOLID principles to ensure clean, maintainable, and scalable code.

## Project Goals

Create an organized and user-friendly task management system.

Allow users to add, edit, delete, and mark tasks as completed easily.

Store tasks permanently in the browser using Local Storage.

Follow clean coding practices with SOLID principles.

## Project Structure

### 1. Core Project Components

HTML File: Defines the structure and layout of the page.

CSS File: Handles the design and user experience improvements.

JavaScript File: Manages task operations and local storage handling.

### 2. To-Do List Sections

The tasks are categorized into two separate lists:

Daily To-Do List for tracking day-to-day tasks.

Weekly To-Do List for tracking tasks that need to be completed within a week.

## Implementation of SOLID Principles

### 1. Single Responsibility Principle (SRP)

StorageHandler: Handles interactions with Local Storage.

TaskManager: Manages task operations such as adding, deleting, and editing.

UIHandler: Updates the user interface and displays tasks.

### 2. Open/Closed Principle (OCP)

The code is open for extension but closed for modification.

New lists or features can be added without modifying the core logic.

### 3. Liskov Substitution Principle (LSP)

The Task class is designed to be extensible with additional properties (e.g., PriorityTask).

Any subclass can replace the base class without breaking the application.

### 4. Interface Segregation Principle (ISP)

The TaskManager class is designed to avoid unnecessary dependencies.

Each class serves a well-defined purpose, ensuring a modular and maintainable structure.

### 5. Dependency Inversion Principle (DIP)

The code depends on abstract classes rather than specific implementations.

TaskManager interacts with StorageHandler rather than directly using localStorage, making it easier to switch storage methods in the future.

## Local Storage Implementation

Local Storage is used instead of an external database for the following reasons:

Easy Accessibility: Data is stored directly in the browser.

Offline Availability: Users can manage tasks without an internet connection.

Fast Performance: Tasks are retrieved instantly without an API request.

🔹 Tasks are stored as follows:

Daily tasks are saved under dailyTasks in Local Storage.

Weekly tasks are saved under weeklyTasks in Local Storage.

## User Interface (UI) Design

The UI is designed with the following principles:

Two lists are displayed side by side using Flexbox.

A modern, visually appealing design with smooth animations.

Consistent colors and buttons for different actions (Edit, Delete, Complete, Save, Cancel).

## Conclusion

This project serves as a practical implementation of a daily and weekly task management system with persistent local storage, following Object-Oriented Programming (OOP) best practices and SOLID principles to ensure scalability and maintainability.

### Potential Future Enhancements:

Syncing tasks with user accounts via Firebase or a database.

Adding task categorization based on priority (Low, Medium, High).

Implementing a Dark Mode for improved user experience.

This project is built to provide a smooth user experience with well-structured and maintainable code!