Software Requirements Specification (SRS)

Project Title: LifeLog A Personal Journal Web Application

Version: 1.0
Authors: LifeLog Group
Date: 20/10/2025

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to describe the functionality and behavior of **LifeLog**, a personal journal web application that enables users to record, view, and manage their daily thoughts, habits, moods, and tasks.
This document serves as a guide for the developers, project supervisors, and testers to ensure all system components meet the desired specifications and user needs.

1.2 Scope

LifeLog is a lightweight, browser-based journaling application built using **HTML, CSS, and JavaScript**. It allows users to:

- Create, view, and delete journal entries.
- Track habits, moods, and daily to-do tasks.
- Review weekly progress.

All user data is stored in **Browser Local Storage**, enabling offline use.
Future versions may integrate **PHP** and **database storage** for authentication, cloud backup, and **AI-based enhancements** (summarization, tone analysis, and smart suggestions).

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
UI	User Interface
UX	User Experience
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SRS	Software Requirements Specification
Local Storage	Browser storage for persisting client-side data
AI	Artificial Intelligence

1.4 References

- IEEE 830-1998 & IEEE 29148-2011 SRS Standards
- MDN Web Docs: HTML, CSS, JavaScript
- Git and GitHub Documentation
- Visual Studio Code User Guide

1.5 Overview

This document presents a detailed outline of the LifeLog project, including its purpose, scope, functionality, and technical requirements. It also provides diagrams and use cases illustrating the flow of user interaction within the system.

2. Overall Description

2.1 Product Perspective

LifeLog is a standalone, front-end web application that functions entirely on the client side. It requires no external server for its core features and relies solely on browser Local Storage for saving and retrieving data.
All application modules (Journal, Habits, Moods, To-Do, and Weekly Progress) are integrated within a unified dashboard.

2.2 Product Functions

The system provides the following functions:

- Add, view, and delete journal entries.
- Track daily habits using checkboxes.
- Log daily moods and optional notes.
- Manage daily to-do tasks.
- View weekly progress reports.
- Maintain responsive and user-friendly UI across devices.

2.3 User Characteristics

- Users have basic knowledge of computer operation and web navigation.
- No programming or database knowledge required.
- Suitable for personal users such as students, professionals, and anyone wanting to track their daily life activities.

2.4 Constraints

- Limited to single-user operation (no login system).
- Data storage restricted to browser Local Storage.
- Works only in modern browsers supporting HTML5 and Local Storage.

2.5 Assumptions and Dependencies

- Users have a modern browser (Chrome, Firefox, Edge, Safari).

- Application performance depends on Local Storage availability.
- Internet connection required only for the initial page load.

3. Specific Requirements

3.1 Functional Requirements

ID	Requirement	Description
FR1	Journal Entry	-Users shall be able to create, view, and delete journal entries.
FR2	Habit Tracker	- Users shall be able to check and uncheck daily habits.
FR3	Mood Tracker	- Users shall log their mood and add optional notes.
FR4	To-Do List	- Users shall add, mark, or remove tasks.
FR5	Weekly Summary	-The system shall summarize activities and display weekly progress.
FR6	Local Data Storage	-All entries shall be stored and retrieved using browser Local Storage.
FR7	Responsive UI	-Interface shall adapt to desktop and mobile devices.

3.2 Non-Functional Requirements

Category	Requirement Description
Performance	-The app must load within 2 seconds on a standard browser.
Usability	-UI must be clean, intuitive, and easy to navigate.
Reliability	-Data must persist after page reload or browser closure.
Security	-All user data remains private and stored locally.
Portability	-Must work across all modern browsers.
Maintainability	-Source code must be modular, readable, and Git-managed.
Scalability	-Should support future backend and AI integration.

3.3 External Interface Requirements

Type	Requirement
User Interface	-Accessible through any standard web browser. Uses responsive layout and clean design.
Hardware Interface	-No special hardware required beyond a device with a web browser.
Software Interface	-Runs locally using HTML/CSS/JS, requires no external dependencies.
Communications Interface	-Optional for future backend version using HTTP/HTTPS.

4. System Features

4.1 Journal Entry Management

Users can record personal reflections, view previous entries, and delete outdated logs.

4.2 Habit Tracker

Allows users to create and monitor daily habits using simple checkboxes.

4.3 Mood Tracker

Lets users log daily emotions and attach short notes to describe their mood.

4.4 Daily To-Do List

Provides a checklist for daily tasks, helping users stay organized and productive.

4.5 Weekly Progress Tracker

Summarizes the user’s habits, moods, and to-do completions over the week.

4.6 Responsive User Interface

Automatically adjusts layout for optimal viewing on mobile, tablet, and desktop devices.

5. Use Case Scenarios

Use Case 1: Add Journal Entry

Actor: User
Precondition: Application is open in a browser.
Main Flow:

1. User selects “Add New Entry.”
2. User writes content and saves.
3. System stores data in Local Storage.
   Exception:

- E1: Empty input → System prompts the user to enter text.

Use Case 2: Manage To-Do List

Actor: User
Main Flow:

1. User adds a new task.
2. Marks completed tasks.
3. Deletes unnecessary tasks.
   Exception:

- E1: Task input left empty → System displays warning.

Use Case 3: Track Mood

Actor: User
Main Flow:

1. User selects mood from available options.
2. Optionally adds a short note.
3. Data is saved automatically.

6. System Models

6.1 Use Case Diagram

(Diagram to be included showing interactions among User, Journal, Habit, Mood, and To-Do modules.)

6.2 Data Flow Diagram (DFD)

Level 0 DFD:

- User → [Input Data] → LifeLog App → [Store/Retrieve] → Local Storage.

6.3 Mockups

Sample UI mockups will display the Journal Dashboard, Habit Tracker, and Mood Tracker sections.

7. Appendix

Term	Definition
Entry	-A single journal record created by the user.
Tracker	-A system module used for monitoring daily activities or moods.
Local Storage	-Client-side storage provided by browsers.
Progress Summary	-A weekly overview showing user's recorded activities.

Revision History

Date	Version	Description	Author
20/10/2025	1.0	Initial Draft LifeLog	Group