**오픈소스SW 기초 2분반 프로젝트 제안서TodoList**

9조 박승민, 박재성, 송원석, 조현준

https://github.com/seungminbluebox/TodoList

**< 목 차 >**

**제1장 프로젝트 개요**

1 프로젝트명3

2 프로젝트 목표3

3 주요기능 요약3

4 기대효과3

**제2장 요구사항 명세서**

1 기능 요구사항4

2.1.1 데이터관리4

2.1.2. 할 일 관리4

2 비기능 요구사항5

**제3장 유저 시나리오**

1 유저 시나리오5

2 유저 시나리오 효과6

**제4장 위험요소 및 평가방법**

1 프로젝트 개발 위험요소6

2 위험요소 평가방법6

**제5장 아키텍쳐 구조**

1 UML 다이어그램7

2 화면 및 메뉴 구성8

3 기술 스택10

**제6장 개발전략**

1 역할분담10

2 개발 일정10

**제7장 참고문헌 및 용어설명**

1 참고문헌11

2 용어설명11

**1. 프로젝트 개요**

**1.1. 프로젝트명**

브라우저 기반 TodoList 웹 애플리케이션 개발

**1.2. 프로젝트 목표**

본 프로젝트는 순수 웹 기술인 HTML, CSS, JavaScript와 브라우저 내장 저장소인 localStorage를 활용하여, 별도의 서버나 회원가입 없이 사용자가 자신의 브라우저에서 할 일 목록을 관리할 수 있는 웹 애플리케이션을 개발하는 것을 목표로 합니다. 이를 통해 사용자는 빠르고 직관적인 환경에서 데이터를 영구적으로 보관하고 관리하는 경험을 할 수 있습니다.

**1.3. 주요 기능 요약**

* **데이터 영속성**: localStorage를 활용하여 사용자가 브라우저를 닫거나 컴퓨터를 재시작해도 데이터가 유지되도록 구현
* **할 일(Todo) 관리**: 할 일 추가, 내용 수정, 삭제 및 완료 여부 토글(Toggle) 기능, 카테고리 설정 및 정렬 기능, 우선순위 표시, 할 일 개수 표시
* **즉각적인 반응성**: 서버와의 통신이 없어 모든 데이터 처리가 즉각적으로 이루어지는 빠른 사용자 경험 제공
* **반응형 UI**: 데스크톱, 태블릿, 모바일 등 다양한 디바이스에서 최적화된 화면 제공
* **캘린더 기능(Optional)**: 캘린더 형식으로 할 일을 조회 (선택 사항)

**1.4. 기대 효과**

* **순수 JavaScript(Vanilla JS) 역량 강화**: 외부 라이브러리 의존 없이 JavaScript의 핵심 기능(DOM 조작, 이벤트 처리, 데이터 관리)에 대한 깊은 이해와 구현 능력 습득
* **웹 브라우저 저장소(Web Storage) 이해**: localStorage의 작동 원리를 이해하고, 클라이언트 사이드에서 데이터를 영속적으로 관리하는 능력 배양
* **포트폴리오 활용**: 웹 개발의 기본기를 다질 수 있는 실용적이고 완성도 높은 결과물 확보

**2. 요구사항 명세**

**2.1. 기능 요구사항 (Functional Requirements)**

**2.1.1. 데이터 관리**

* **REQ-101**: 사용자의 할 일 목록 데이터는 웹 브라우저의 localStorage에 저장되어야 한다.
* **REQ-102**: 브라우저를 종료하거나 컴퓨터를 재시작해도 이전에 저장된 할 일 목록이 그대로 유지되어야 한다.
* **REQ-103**: 데이터는 문자열 형태로 변환(JSON 직렬화)되어 localStorage에 저장되고, 조회 시에는 다시 객체 형태로 변환(JSON 파싱)되어 사용되어야 한다.

**2.1.2. 할 일(Todo) 관리**

* **REQ-201 (생성)**: 사용자는 새로운 할 일을 텍스트로 입력하여 목록에 생성할 수 있어야 한다.
* **REQ-202 (조회)**: 사용자는 자신이 추가한 모든 할 일 목록을 화면에서 조회할 수 있어야 한다.
* **REQ-203 (수정)**: 사용자는 기존에 작성된 할 일의 내용을 수정할 수 있어야 한다.
* **REQ-204 (삭제)**: 사용자는 원하는 할 일을 목록에서 영구적으로 삭제할 수 있어야 한다. (선택 삭제, 일괄 삭제 기능 등을 고려할 수 있음)
* **REQ-205 (상태 변경)**: 사용자는 각 할 일의 완료/미완료 상태를 변경(토글)할 수 있어야 하며, 완료된 항목은 시각적으로 구분(예: 취소선)되어야 한다.
* **REQ-206 (우선순위 표시)**: 중요도에 따라 높음/중간/낮음 등으로 표시하거나 색깔을 다르게 하여 가장 중요한 일을 한눈에 파악할 수 있게 한다.
* **REQ-207 (카테고리/태그 지정)**: '업무', '개인', '쇼핑', '긴급' 등 할 일에 태그나 카테고리를 부여하여 쉽게 분류할 수 있게 한다.
* **REQ-208 (카테고리/태그 관리)**: 사용자는 생성된 태그나 카테고리를 수정, 삭제, 추가할 수 있어야 한다.
* **REQ-209 (할 일 개수 표시)**: 당일 기준 '완료한 일의 개수/해야 할 일의 개수'를 표시하여 사용자로 하여금 전체적인 진행도를 직관적으로 확인할 수 있게 한다.
* **REQ-210 (필터링 기능)**: 사용자는 완료한 일, 수행해야 하는 일 등을 기준으로 목록을 필터링하여 최상단에 노출되게 할 수 있어야 한다.
* **REQ-211 (정렬 기능)**: 사용자는 할 일 목록을 작성일, 마감일, 우선순위, 완료 상태를 기준으로 정렬할 수 있어야 한다.

**2.2. 비기능 요구사항 (Non-functional Requirements)**

* **NFR-01 (사용성)**: UI는 직관적이고 미니멀한 디자인을 채택하여 누구나 쉽게 사용할 수 있어야 한다.
  + **피드백 제공**: 할 일 추가, 수정, 삭제 시 사용자에게 성공 또는 오류를 알리는 명확한 시각적/텍스트 피드백을 즉시 제공해야 한다.
  + **접근성**: 마우스뿐만 아니라 키보드만으로도 할 일 생성, 완료 토글, 삭제 등 모든 주요 기능을 사용할 수 있도록 웹 접근성을 기본적으로 고려해야 한다.
* **NFR-02 (성능)**: 페이지 로딩 및 모든 데이터 처리(CRUD) 작업은 네트워크 지연 없이 즉각적으로(1초 이내) 이루어져야 한다.
* **NFR-03 (호환성)**: 최신 버전의 Chrome, Firefox, Safari, Edge 브라우저에서 모든 기능이 정상적으로 동작해야 한다.

**3. 유저 시나리오**

**3.1. 유저 시나리오 (User Scenario)**

회사원 김지원 씨는 웹 브라우저를 통해 나중에 해야 할 일을 한눈에 보고 정리하기 위해 TodoList를 사용합니다.

* '저녁에 장보기'와 같이 입력창에 할 일을 적으면 화면에 입력한 할 일이 바로 떠서 쉽게 볼 수 있습니다.
* 일주일 후에 만나기로 한 약속 날짜가 바뀌더라도 수정 버튼을 통해 이미 입력한 할 일도 수정할 수 있습니다.
* TodoList에 입력한 약속이 이미 지났다면 버튼을 통해 완료/미완료 상태로 변경할 수 있으며 완료한 내용은 취소선으로 바로 구분할 수 있습니다.
* 아예 필요 없는 내용은 삭제를 통해 완전히 없앨 수도 있습니다.
* 회사에서 처리하는 내용과 집에 와서 처리하는 내용 등을 구분하기 위해 각각의 태그를 설정하여 쉽게 분류하여 볼 수 있습니다.
* 또한 당장 오늘 처리해야 하는 급한 일과 나중에 장기적으로 처리하는 일을 구분하기 위해 우선순위를 설정합니다.
* 지금까지 완료한 내용과 아직 해야 할 미완료된 내용의 개수를 통해 할 일들의 전체적인 진행도 또한 확인이 가능합니다.
* 갑자기 브라우저가 꺼지거나 컴퓨터를 재시작하더라도 localStorage로 인해 입력한 리스트가 사라지지 않게 합니다.

**3.2. 유저 시나리오 효과**

유저 시나리오는 유저의 일상 속 TodoList 사용으로 일어나는 일반적인 상황과 갑자기 브라우저가 꺼지거나 약속 날짜가 바뀌는 것과 같은 예측 불가능한 상황까지 고려하여 작성됩니다. 이를 통해 '문제인식'을 하여 서비스화할 점, 개선점을 파악하는 데 유용한 효과를 낼 수 있습니다.

**4. 위험 요소 및 평가 방법**

**4.1. 프로젝트 개발 위험 요소**

* **데이터 손실**: localStorage관리 도중 데이터가 손실될 수 있음.
* **기능 충돌**: 여러 개의 기능들이 상호 다른 작업을 시도하다가 서로 충돌하여 프로젝트에 오류가 발생할 수 있음.
* **요구사항 변동**: 프로젝트 개발 도중 요구사항이 변동되어 기능의 추가 또는 삭제 등이 필요할 수 있음.
* **버그(BUG)**: 알 수 없는 다양한 이유들로 프로젝트에 오류, 즉 버그가 발생할 수 있음.

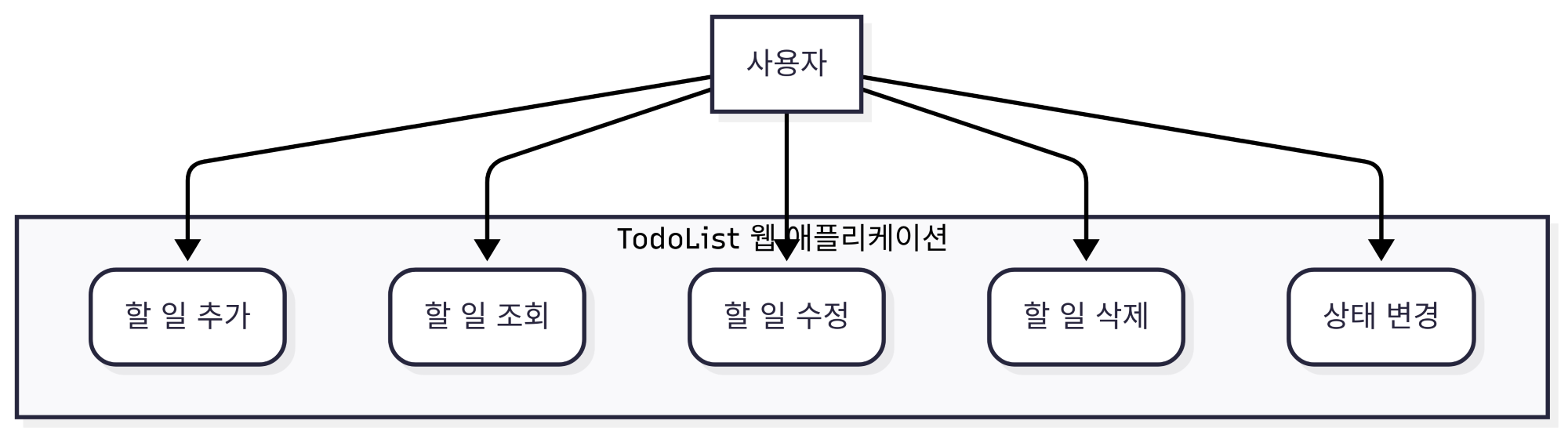
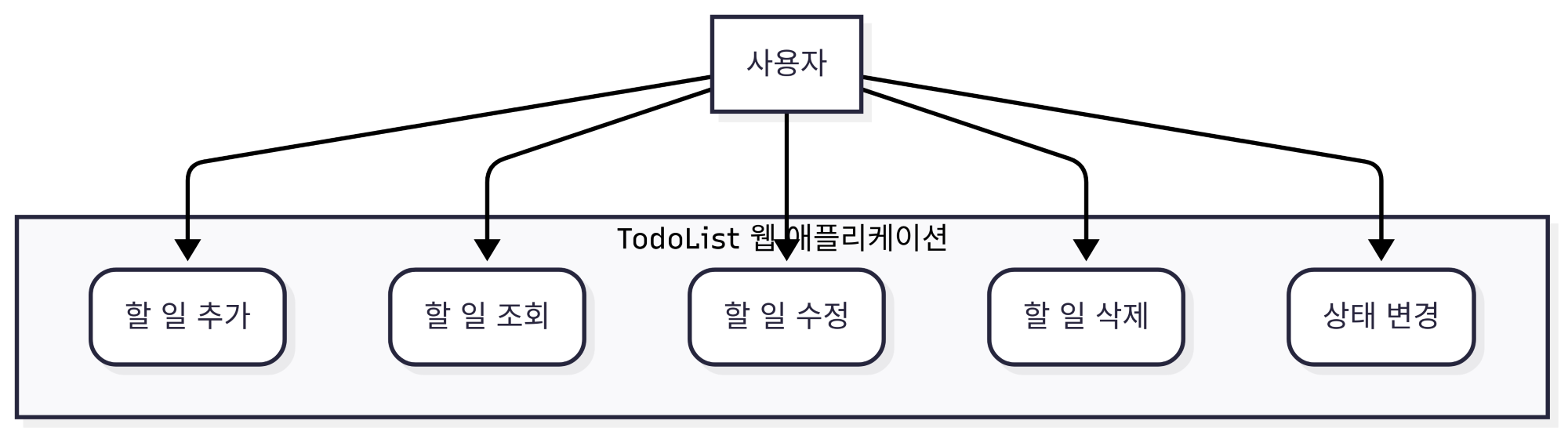
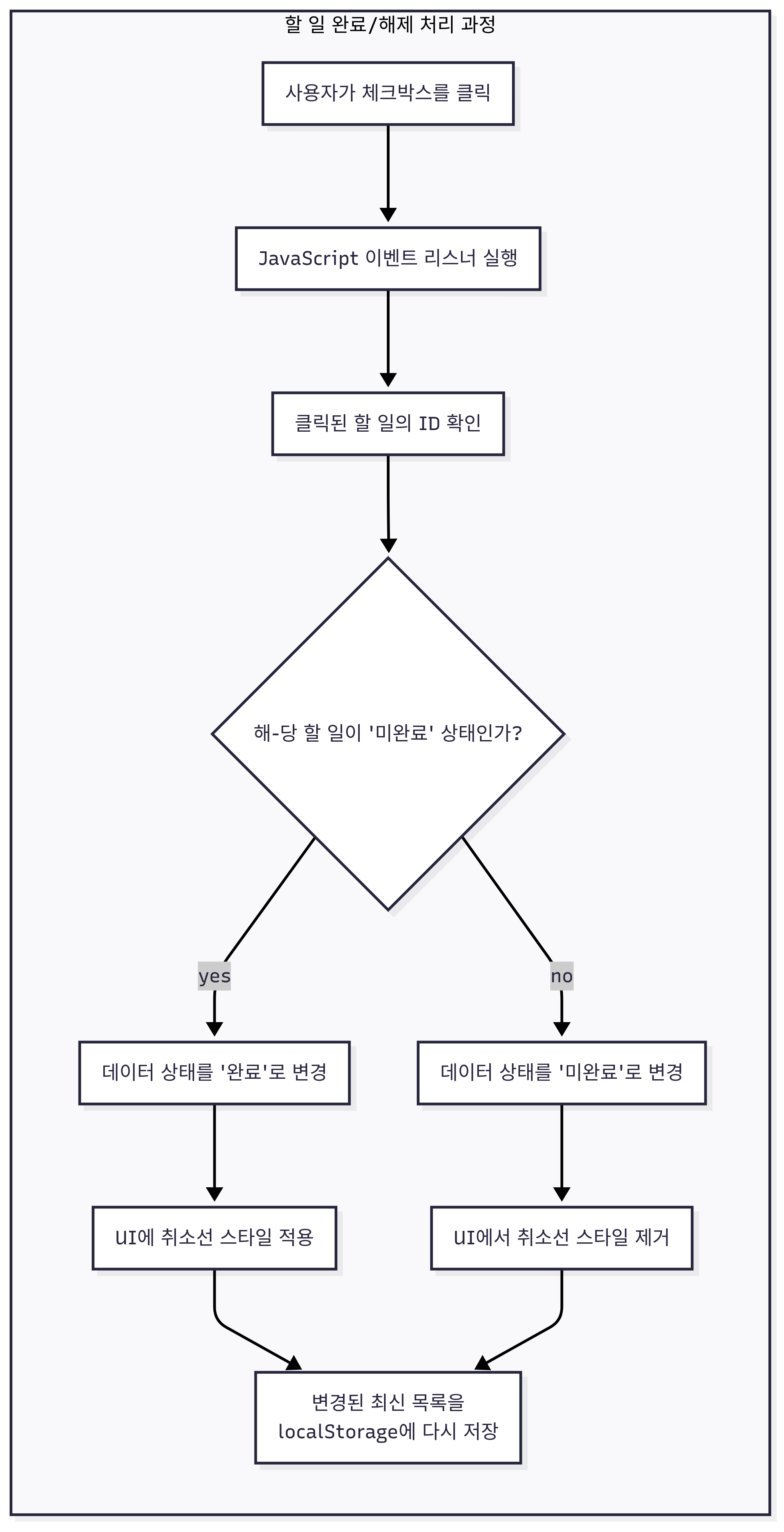
**4.2. 위험 요소 평가 방법**

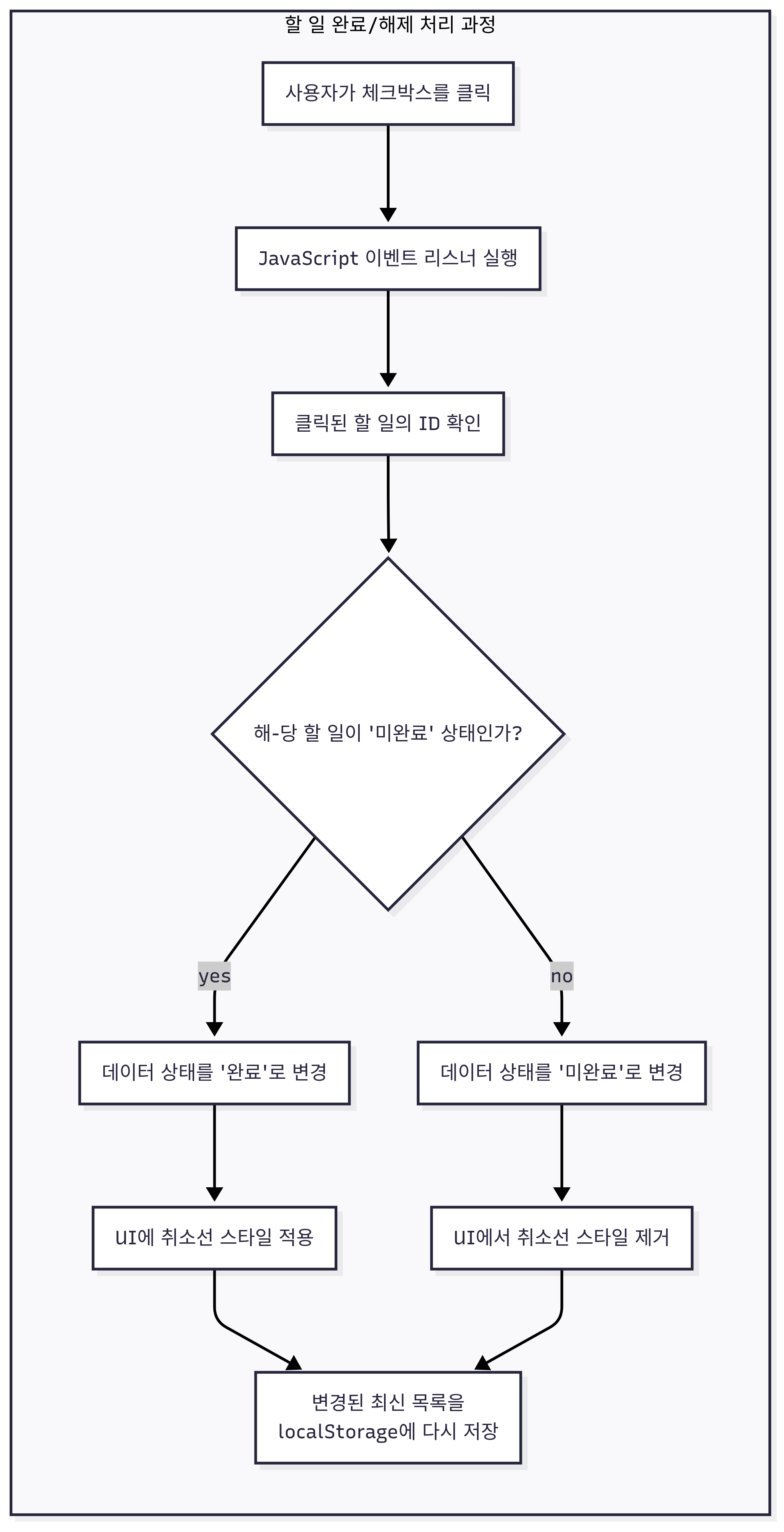
* **데이터 백업**: 데이터의 백업본을 지속적으로 저장하여 데이터 손실의 타격을 최소화. (예: 특정 시점의 localStorage데이터를 텍스트 파일로 내보내기)
* **다양한 환경에서의 테스트**: 다양한 환경(ex. 저장된 데이터 크기의 극대화, 여러 브라우저)에서의 테스트를 통해 프로젝트의 오류를 조기에 확인 및 해결.
* **GitHub를 이용한 피드백**: 개발자들끼리의 지속적인 피드백과 코드 리뷰를 통해 변경사항 및 오류에 빠르게 반응.

**5. 아키텍처 구조**

**5.1. UML 다이어그램**

**유스케이스 다이어그램**

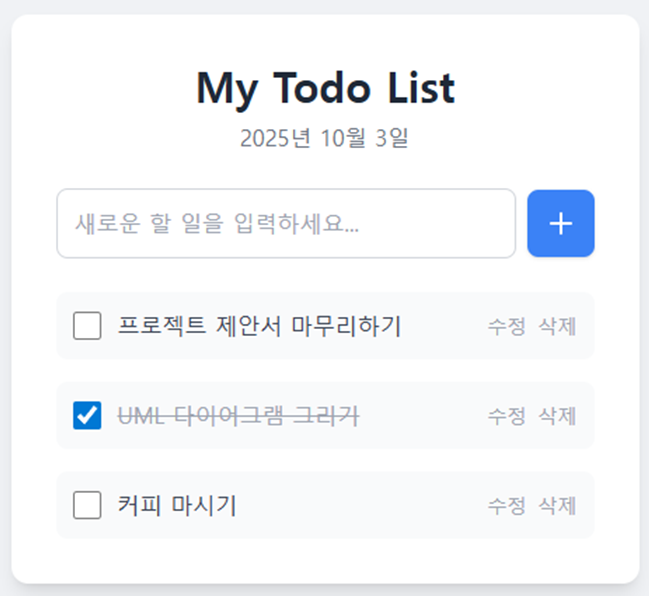
  
**활동 다이어그램 (예시: 할 일 완료/해제 처리 과정)**



**5.2. 화면 및 메뉴 구성**

복잡한 메뉴나 여러 페이지 없이, Single Page안에서 모든 것이 해결되도록 설계하는 것이 이 프로젝트의 핵심입니다.

* **컨셉**: 미니멀리즘, 직관성
* **단일 화면**: 사용자는 페이지를 이동할 필요 없이 한 곳에서 모든 기능을 사용할 수 있습니다.**불필요한 요소 제거**: 회원가입, 설정 메뉴, 카테고리 등 부가적인 기능을 모두 제외하고 오직 '할 일 관리'라는 본질에만 집중한 페이지 구성 (단, 추가된 카테고리/우선순위 기능은 직관적인 UI로 통합)



**1. 헤더 (Header) 영역**

* **역할**: 앱의 정체성을 보여주는 공간입니다.
* **구성 요소**:
  + **제목**: 'My Todo List', '오늘의 할 일' 등 앱의 이름이나 목적을 나타내는 큰 제목을 배치합니다.
  + **(선택) 날짜**: 오늘 날짜를 동적으로 표시해주면 사용자가 할 일을 관리하는 데 더 도움이 됩니다. (JavaScript의 new Date()객체를 활용)

**2. 입력 (Input) 영역**

* **역할**: 사용자가 새로운 할 일을 추가하는 공간입니다.
* **구성 요소**:
  + **입력창 (<input type="text">)**: 사용자가 할 일의 내용을 입력하는 곳입니다. placeholder속성을 이용해 "새로운 할 일을 입력하세요..."와 같은 안내 문구를 넣어주면 좋습니다.
  + **추가 버튼 (<button>)**: 입력된 내용을 목록에 추가하는 버튼입니다. '추가'라는 텍스트나 간단하게 '+' 아이콘을 사용할 수 있습니다. 사용자가 입력창에서 Enter 키를 눌렀을 때도 추가되도록 구현하면 편의성이 크게 향상됩니다.
  + **(추가) 우선순위/카테고리 선택**: 입력창 옆에 드롭다운 메뉴나 버튼 형태로 우선순위 및 카테고리를 선택할 수 있는 UI를 제공할 수 있습니다.

**3. 목록 (List) 영역**

* **역할**: 저장된 모든 할 일을 보여주고 관리하는 핵심 공간입니다.
* **구성 요소 (개별 할 일 항목 <li>기준)**:
  + **체크박스 (<input type="checkbox">)**: 할 일의 완료/미완료 상태를 표시하고 변경하는 토글(Toggle) 역할을 합니다.
    - 체크 시: 할 일 텍스트에 취소선(text-decoration: line-through;)을 그어 시각적으로 완료했음을 표시합니다.
  + **할 일 텍스트 (<span>또는 <label>)**: 할 일의 내용이 표시됩니다.
  + **(추가) 우선순위/카테고리 표시**: 작은 아이콘이나 색상, 텍스트 등으로 할 일의 우선순위와 카테고리를 명확히 표시합니다.
  + **수정 버튼**: 클릭 시 현재 텍스트를 입력창으로 바꿔 내용을 수정할 수 있게 합니다. (예: 텍스트 <span>을 숨기고, 내용이 담긴 <input>을 보여주는 방식)
  + **삭제 버튼**: 해당 할 일을 목록에서 완전히 제거합니다. 'X' 아이콘 버튼이 가장 직관적입니다.
  + **(상단) 필터링/정렬 컨트롤**: 완료/미완료 필터, 우선순위, 작성일 등으로 정렬할 수 있는 드롭다운이나 버튼 그룹이 목록 상단에 배치될 수 있습니다.
  + **(하단) 할 일 개수 표시**: 완료/미완료 할 일의 개수를 직관적으로 보여주는 UI를 배치합니다.

**5.3. 기술 스택**

* **HTML**: 기본적인 UI 구성에 사용
* **CSS**: 프로젝트의 지향점인 미니멀한 UI 디자인을 위해 사용 (Tailwind CSS와 같은 프레임워크 활용 고려)
* **Vanilla JS**: 날짜 표시와 같은 동적 서비스 구현 및 localStorage내의 사용자의 ToDo들을 생성(Create), 조회(Read), 수정(Update), 삭제(Delete)하기 위해 사용.

**6. 개발 전략**

**6.1. 역할 분담**

* **기획 및 개발 (박승민)**: 프로젝트 기획 및 구성, 전체적인 개발 총괄
* **개발 (박재성)**: 전체적인 개발 총괄
* **QA (송원석)**: 프로젝트 테스트 및 버그 보고/해결
* **보고서 작성 및 발표 (조현준)**: 프로젝트 보고서 작성 및 발표

**6.2. 개발 일정**

* **1~2주차**: 프로젝트 기획 및 사용자 요구사항 구상
* **3~4주차**: 프로젝트 구성 확정 및 제안서 작성
* **5~6주차**: HTML을 이용, 페이지 구조 등의 전체적인 프로젝트의 UI 구성
* **7~8주차**: CSS를 이용, 프로젝트 UI의 디자인 구현
* **9~10주차**: JS를 이용, 동적인 서비스 구현 (데이터 관리 및 CRUD 기능, 카테고리, 우선순위, 필터링, 정렬 등 핵심 로직)
* **11~12주차**: 프로젝트 통합 테스트 및 버그 수정 (QA 역할 집중)
* **13~14주차**: 프로젝트 보고서 작성 및 서비스 배포 (GitHub Pages 등)

**7. 참고문헌 및 용어설명**

**7.1. 참고문헌**

본 프로젝트는 다음의 공식 기술 문서를 표준으로 참고하여 개발을 진행

1. **MDN Web Docs - HTML (HyperText Markup Language)**
   * 웹 페이지의 구조를 정의하기 위한 표준 마크업 언어의 공식 가이드
   * URL: https://developer.mozilla.org/ko/docs/Web/HTML
2. **MDN Web Docs - CSS (Cascading Style Sheets)**
   * HTML 문서의 표현(디자인, 레이아웃 등)을 기술하기 위한 스타일 시트 언어의 공식 가이드
   * URL: https://developer.mozilla.org/ko/docs/Web/CSS
3. **MDN Web Docs - JavaScript**
   * 웹 페이지의 동적 기능을 구현하기 위한 프로그래밍 언어의 공식 가이드
   * URL: https://developer.mozilla.org/ko/docs/Web/JavaScript
4. **MDN Web Docs - Web Storage API (localStorage)**
   * 브라우저가 키-값 쌍을 안전하게 저장할 수 있는 메커니즘을 제공하는 API 명세. 본 프로젝트의 핵심 데이터 저장 기술로 사용
   * URL: https://developer.mozilla.org/ko/docs/Web/API/Web\_Storage\_API

**7.2. 용어 설명**

* **localStorage(로컬 스토리지)**: 웹 브라우저에 데이터를 영구적으로 저장하는 기술. 사용자가 브라우저를 닫거나 컴퓨터를 재시작해도 데이터가 사라지지 않는 특징을 가짐
* **Vanilla JS (순수 자바스크립트)**: 외부 라이브러리나 프레임워크를 사용하지 않고, 웹 브라우저에 내장된 순수 JavaScript만을 사용하여 개발하는 방식
* **CRUD**: 대부분의 데이터 처리 소프트웨어가 가지는 기본적인 기능인 **Create(생성)**, **Read(조회)**, **Update(수정)**, **Delete(삭제)**를 묶어 일컫는 뜻.
* **UI (User Interface) / UX (User Experience)**: **UI**는 사용자가 마주하는 화면의 시각적인 디자인(레이아웃, 색상 등)을 의미. **UX**는 사용자가 이 서비스를 이용하면서 느끼는 전반적인 경험(편리함, 만족도 등)을 의미
* **JSON (JavaScript Object Notation)**: 데이터를 교환하는 데 사용되는 경량의 텍스트 기반 형식. JavaScript 객체 문법을 따르며, localStorage에 객체나 배열을 저장하기 위해 데이터를 문자열로 변환(직렬화)하거나 그 반대로 변환(파싱)하는 데 사용
* **Single Page Application (SPA)**: 단일 HTML 페이지에서 시작하여, 사용자의 행동에 따라 필요한 부분만 동적으로 업데이트하여 마치 여러 페이지를 이동하는 것처럼 느끼게 해주는 웹 애플리케이션 아키텍처