Google Chrome 확장 프로그램 개발 기획서 초안

1. 프로젝트 개요 **(Project Overview)**

1.1 프로젝트 목표

사용자의 웹 브라우징 경험을 [구체적인 목표: 보안 관리 자동화 및 생산성 증대]하는 것을 목표로 합니다.

- 최종 목표**:** 로그인, 회원가입을 했을 경우 자동으로 해당 확장 프로그램에 기록, 관리하여 사용자의 디지털 보안 습관을 개선합니다.
- 사용자 가치**:** 회원가입을 진행한 사이트 관리, 마지막으로 해당 사이트 로그인 날짜 기록, 마지막 비밀번호 변경 이력 추적을 통해 주기적 비밀변호 변경을 유도하여 보안 위험을 줄입니다.

1.2 확장 프로그램 이름 **(**가칭**)**

계정 기록장 (Account Ledger)
(대안: Pass Tracker, Site Guard)

1.3 핵심 기능 **(Core Features)**

No.	기능 분류	세부 기능 및 역할
1	메인 기능 (회원가입 감지 및 기록)	크롬에서 회원가입이 발생했다고 판단되면 (URL 분석 및 DOM 이벤트 감지) 해당 사이트를 추가하여 관리합니다. (사이트 도메인, 가입 시기 등) 본인이 어떠한 사이트에 회원가입이 되어 있는지 기록하기 위한 용도입니다.
2	메인 기능 (로그인 감지 및 갱신)	로그인이 발생했다고 판단되면 해당 사이트에 대한 최근 로그인 시간을 수정합니다.
3	메인 기능 (비밀번호 변경 감지 및 갱신)	비밀번호를 변경했다고 판단되면 해당 사이트에 대한 최근 비밀번호 변경

		일자를 수정합니다.
4	메인 기능 (비밀번호 변경 주기 알림)	비밀번호가 일정 주기 (주기 수정 가능)를 넘어서면 팝업 UI 또는 데스크톱 알림을 통해 경고 표시나 알람을 제공합니다.
5	부가 기능 (데이터 가져오기)	**Google**에서 제공하는 비밀번호 관리 데이터를 활용해 해당 데이터를 업로드 하여 해당 사이트들을 관리한다**. (CSV Import)**
6	부가 기능 **(**수동 관리**)**	자동 감지에서 누락된 사이트를 직접 **URL**을 입력하여 등록하고**,** 기존 사이트의 로그인**/**비밀번호 변경 시간을 수동으로 업데이트 **(**현재 시간 클릭 또는 직접 입력**)** 및 목록에서 사이트를 삭제하는 기능을 제공합니다**.**

1.4 대상 사용자 및 환경

- 대상**:** 웹 사용자 (다수의 웹사이트 계정을 관리하는 모든 사용자)
- 필수 환경**:** Google Chrome 브라우저 (최신 버전 권장)

2. 확장 프로그램 명세 **(Extension Specification)**

2.1 기술 스택

데이터의 영속적인 저장과 동기화가 필수적이므로, localStorage 대신 **Firebase Firestore**를 사용하는 것을 전제로 기술 스택을 구성했습니다.

구성 요소	기술**/**언어	역할
프론트엔드	HTML5, CSS3 (**Tailwind CSS**)	팝업(Popup) 및 옵션 페이지의 직관적이고

		반응성이 좋은 UI 구성
로직/데이터	**JavaScript (ES6+)**	핵심 기능 (감지 로직), 이벤트 처리, Chrome API 통신
백엔드/DB	**Firebase Firestore**	사용자 계정 데이터 (사이트 목록, 각 날짜)의 영구적 저장 및 동기화 (다수 기기 지원)
UI 프레임워크	**Vanilla JS**	기능 복잡도를 고려했을 때, 초기에는 Vanilla JS로 구현하여 가볍게 유지

2.2 크롬 확장 프로그램 구조

핵심 기능 감지 및 데이터 관리를 위해 content_script와 background.js 간의 메시지 통신 및 강력한 호스트 권한이 필요합니다.

파일/요소	설명	역할
manifest.json	확장 프로그램의 메타데이터 및 설정 파일	**storage, alarms, notifications, tabs, scripting, host_permissions** 권한 정의 및 스크립트 등록
background.js	백그라운드 스크립트 (Service Worker)	주기적 비밀번호 만료 검사 **(**알람 기능**)**, Content Script와 팝업으로부터 받은 메시지를 처리하고 Firestore에 데이터를 저장/업데이트하는 역할
popup.html/js	팝업 UI (아이콘 클릭 시 열림)	현재 페이지의 상태 표시, 즉각적인 수동 기록 버튼, 빠른 사이트 검색/확인
content_script.js	콘텐츠 스크립트	웹페이지 **DOM**에 접근하여 URL 패턴 변화 감지, 폼 제출 이벤트 리스너 부착,

		감지된 이벤트를 background.js로 전달
options.html/js	옵션 페이지 (필수)	전체 계정 목록 확인 및 관리, 비밀번호 변경 주기 설정, 수동 데이터 수정 및 삭제, **CSV** 파일 가져오기 **UI**, 수동 사이트 등록 및 시간 업데이트 **UI**

3. 핵심 기능 상세 구현 방안

3.1 기능 **A:** 계정 이벤트 감지 및 기록 **(**회원가입**,** 로그인**,** 비밀번호 변경**)**

- 목표**:** 사용자 행동을 최대한 덜 침해하는 방식으로 핵심 계정 이벤트를 정확하게 탐지하고, 해당 사이트 정보를 Firestore에 영구적으로 기록/업데이트합니다.
- 구현 방법**:**
  1. **content_script.js** 역할 **(**이벤트 감지**):**
     - **URL** 패턴 모니터링**:** navigation.onDOMContentLoaded 시점에 현재 URL을 검사하여 /signup, /join, /login, /change-password, /reset-password 등 일반적인 이벤트 관련 URL 패턴이 포함되어 있는지 확인합니다.
     - 폼 제출 리스너 부착**:** 페이지 로드 시, <form> 태그 중 비밀번호 필드 (type="password")를 포함하는 폼에 submit 이벤트 리스너를 부착합니다. 제출 성공(페이지 전환 또는 성공 메시지) 후, 추정되는 이벤트 유형을 메시지로 전송합니다.
     - 메시지 전송**:** 감지된 이벤트와 현재 도메인 정보를 담아 chrome.runtime.sendMessage({ type: 'EVENT_DETECTED', action: 'SIGNUP'/'LOGIN'/'PASS_CHANGE', domain: '...' })을 background.js로 보냅니다.
  2. **background.js** 역할 **(**데이터 관리**):**
     - content_script로부터 메시지를 수신합니다.
     - 데이터 처리**:** Firestore의 사용자 계정 컬렉션(/artifacts/{appId}/users/{userId}/accounts)에 접근하여 해당 domain의 문서를 찾습니다.
     - 업데이트**:**
       - SIGNUP 시: 새 문서를 생성하고 signUpDate에 현재 시각 기록.
       - LOGIN 시: 기존 문서의 lastLoginDate에 현재 시각 업데이트.
       - PASS_CHANGE 시: 기존 문서의 lastPasswordChangeDate에 현재 시각 업데이트.
- 데이터 처리**: Firebase Firestore**를 통해 사용자별 계정 데이터 구조를 관리합니다.
  - **Firestore** 데이터 구조 **(**예시**):**

    // accounts/{domain} 문서

    {

```
  "domain": "example.com",
  "signUpDate": "2023-10-25T10:00:00Z",
  "lastLoginDate": "2024-05-15T15:30:00Z",
  "lastPasswordChangeDate": "2024-01-01T00:00:00Z",
  "isWarning": false
}
```

3.2 기능 B: 비밀번호 변경 주기 알림 및 경고 표시

- 목표: 사용자가 설정한 비밀번호 변경 주기를 초과한 사이트에 대해 선제적으로 알림을 제공합니다.
- 구현 방법:
  1. 주기 설정 **(Options Page):** options.html/js에서 사용자가 원하는 변경 주기 (예: 90일, 180일)를 입력하고, 이 값을 chrome.storage.sync에 저장합니다.
  2. 알람 등록 **(background.js):** background.js에서 chrome.alarms.create('check_password_expiry', { periodInMinutes: 60 * 24 });를 사용하여 매일 한 번씩 알람이 울리도록 등록합니다.
  3. 알람 처리 **(background.js):**
     - chrome.alarms.onAlarm.addListener에서 알람을 감지합니다.
     - Firestore에서 사용자의 모든 계정 데이터를 불러옵니다.
     - 각 계정의 lastPasswordChangeDate와 현재 시각을 비교하여 저장된 주기 (예: 90일)를 초과했는지 확인합니다.
     - 경고 트리거: 주기를 초과한 계정이 있다면, 해당 계정 문서의 isWarning 필드를 true로 업데이트하고, 데스크톱 알림(chrome.notifications.create)을 생성하여 사용자에게 알립니다.
     - **UI** 경고: 팝업 UI가 열릴 때 background.js로부터 경고 목록을 받아 아이콘 뱃지나 팝업 내에 경고 메시지를 표시합니다.
- 필요 권한:
  - storage (주기 설정 저장)
  - alarms (주기적 검사 실행)
  - notifications (데스크톱 알림 생성)

3.3 기능 C: 비밀번호 데이터 CSV 가져오기 (Import from CSV)

- 목표: 사용자가 기존에 Chrome에 저장했던 계정 목록을 빠르고 쉽게 확장 프로그램의 관리 목록에 추가하여 초기 설정의 부담을 줄입니다.
- 구현 방법:
  1. 업로드 **UI (Options Page):** options.html에 파일 업로드(input type="file") 필드 및 버튼을 구현합니다.
  2. **CSV** 파싱 및 도메인 추출 **(options.js):**
     - 사용자가 CSV 파일을 업로드하면, File API (FileReader)를 사용하여 파일 내용을 읽습니다.
     - 제공된 CSV 형식(헤더: name,url,username,password,note)에서 **url** 컬럼의 값을 파싱합니다.

    - 각 url에서 http://, https://, android:// 및 경로 정보를 제거하고, **유효한 도메인(hostname)**만 추출하여 고유 도메인 목록을 생성합니다. (예: https://www.dataq.or.kr/www/join/main.do -> dataq.or.kr)
  3. **Firestore** 기록**:**
    - 추출된 각 고유 도메인에 대해, background.js의 Firestore 로직을 통해 새 문서를 생성합니다.
    - 이 기능은 회원가입**/**로그인 시점을 알 수 없으므로, signUpDate, lastLoginDate, lastPasswordChangeDate 필드는 **null** 또는 **"Imported"**와 같은 초기값으로 설정하여 추후 실제 이벤트가 감지될 때 업데이트되도록 합니다.
    - 대신, 데이터 가져오기 시점을 기록하는 importedAt 필드를 추가하여 관리합니다.
- **Firestore** 데이터 구조 **(CSV Import** 시**):**

```
// accounts/{domain} 문서 (CSV import 시)
{
  "domain": "dataq.or.kr",
  "signUpDate": null,
  "lastLoginDate": null,
  "lastPasswordChangeDate": null,
  "importedAt": "2025-11-12T10:00:00Z", // Import 시점 기록
  "isWarning": false
}
```

3.4 기능 **D:** 수동 관리 기능 **(**등록**/**수정**/**삭제**)**

- 목표**:** 자동 감지 실패 및 사용자 직접 관리가 필요한 경우, Options Page에서 모든 계정 정보를 직접 제어할 수 있는 기능을 제공합니다.
- 구현 방법**:**
  1. 수동 등록 **(Creation):**
    - Options Page에 **URL** 입력 필드와 '수동 등록' 버튼을 구현합니다.
    - 사용자가 URL을 입력하면, 유효성 검사 후 도메인을 추출하여 Firestore에 새로운 계정 문서를 추가합니다.
    - 초기 날짜 필드 (signUpDate, lastLoginDate, lastPasswordChangeDate)는 현재 시각으로 자동 설정됩니다.
  2. 시간 업데이트 **(Update - Two Ways):**
    - 현재 시간 클릭 방식**:** 계정 목록의 각 항목 옆에 '로그인 시간 업데이트' 및 '비밀번호 변경일 업데이트' 버튼을 제공합니다. 클릭 시, 해당 도메인의 Firestore 문서에서 지정된 필드 (lastLoginDate 또는 lastPasswordChangeDate)를 클라이언트의 현재 시각으로 즉시 업데이트합니다.
    - 직접 입력 방식**:** 계정 목록에서 날짜 필드 옆에 **날짜/시간 입력 폼(캘린더 또는 텍스트 입력)**을 제공하여, 사용자가 과거 또는 미래의 특정 시간을 직접 입력하고 '저장' 버튼을 눌러 해당 필드를 업데이트합니다.
  3. 삭제 **(Deletion):**
    - 계정 목록의 각 항목 옆에 '삭제' 버튼을 제공합니다.
    - 버튼 클릭 시 사용자에게 모달 형태의 확인 메시지를 표시합니다.

    - 사용자가 삭제를 확정하면, Firestore에서 해당 도메인의 문서를 deleteDoc을 사용하여 제거하고 목록을 새로 고칩니다.
- **Firestore 활용:** 모든 수동 작업은 options.js에서 직접 Firebase Firestore API를 호출하여 데이터를 조작합니다.

4. 개발 단계 및 일정 계획 **(Timeline)**

단계	기간 **(**예시**)**	주요 작업 내용	결과물**/**산출물
1단계**:** 환경 설정 및 프로토타입	1주차	manifest.json 권한 정의 및 Firebase 연동 설정, 인증 처리 로직 구현 **(signInWithCustomToken/Anonymously)**, 최소 기능 프로토타입 구현 (팝업 UI, Firestore CRUD 테스트)	기본 팝업 및 Firestore 연동 테스트 완료
2단계**:** 핵심 감지 및 백그라운드 구현	2-3주차	기능 A (감지 및 기록)의 **Content Script** 로직 구현 및 메시지 통신 안정화, 기능 B (주기적 알람)의 **background.js** 알람 및 검사 로직 구현	핵심 기능 80% 이상 구현 및 데이터 관리 테스트 완료
3단계**:** 디자인 및 옵션 페이지 구현	4주차	Tailwind CSS를 활용한 UI/UX 디자인 개선, 전체 계정 목록**, CSV Import UI,** 수동 관리 **UI (**등록**/**수정**/**삭제**)** 구현	베타 버전 출시 준비 완료
4단계**:** 출시	5주차	Chrome 웹 스토어 계정 등록 및 심사 요청, 마케팅 자료(스크린샷,	Chrome 웹 스토어 공식 출시

		설명) 준비, 사용자 피드백 반영	

5. 향후 계획 (Future Plans)

- **V2.0** 계획**:** 사용자 계정 정보의 백업 및 복구 기능 추가, 비밀번호 보안 수준 분석 기능 (단순 길이/복잡도 분석) 통합.
- 추가 기능**:** 특정 계정을 '비활성화' 또는 '탈퇴 완료'로 표시하여 관리 목록에서 제외하는 기능 추가.
- 유지보수**:** Chrome Manifest V3 API 변경 및 보안 요구사항에 따른 정기적인 코드 검토 및 패치.