3. Design

[SafeSurfing]
URL 안전 검사 어플리케이션

Student No.	22213487
Name	권기범
E-mail	gkdnld99@yu.ac.kr

[Revision history]

Revision date	Version #	Description	Author
03/20/2025	1.00	First Draft	권기범

= Contents =

1. Introduction

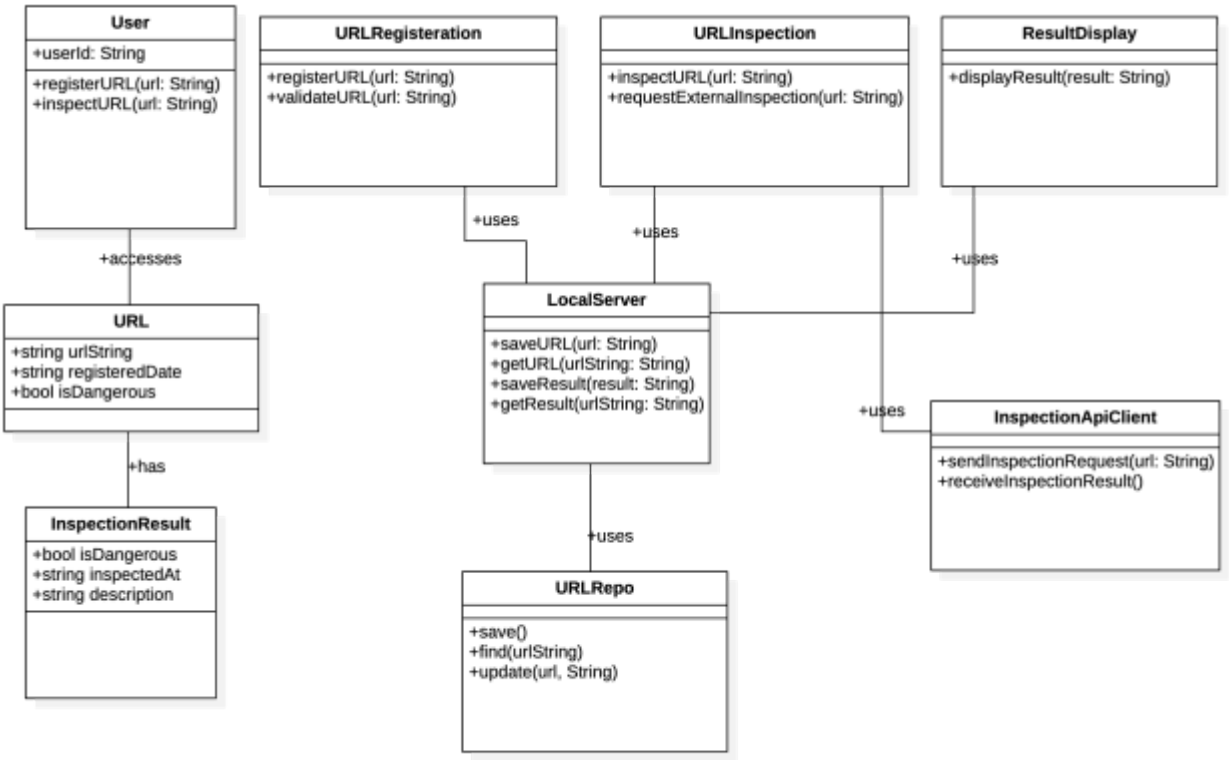
1) Summary

SafeSurfing은 사용자가 입력한 URL의 안전성을 검사하여 위험한 사이트로부터 사용자를 보호하는 모바일 어플리케이션이다. 본 문서는 SafeSurfing의 설계 전반에 관한 내용을 담고 있으며, 주요 기능과 시스템 구성, 사용자 인터페이스 설계, 데이터 흐름 및 상호작용 방식을 포함한다.
주요 페이지는 URL 입력과 검사를 담당하는 메인 페이지, 저장된 URL을 관리하는 URL 관리 페이지, 그리고 사용자 환경설정을 제공하는 설정 페이지로 구성되어 있다. 시스템은 로컬 데이터베이스 역할을 하는 서버 클래스를 포함하며, 외부 API와 연동하여 URL 검사를 수행한다. 또한 모바일 사용 환경에 적합한 UI 설계와 원활한 사용자 경험을 목표로 한다.

2) Important points of Design

- 모듈화된 시스템 구성:
  URL 입력, 검사, 저장, 설정 기능을 명확히 분리하여 유지보수와 확장성을 고려하였다.
- 로컬 데이터베이스 역할 서버 클래스 도입:
  실제 서버 없이 로컬 환경에서 URL 데이터 관리가 가능하도록 설계하였다.
- 외부 API 연동:
  URL 검사 기능은 신뢰성 있는 외부 API를 통해 수행하며, 비동기 통신으로 사용자 경험을 향상시켰다.
- 모바일 최적화 UI:
  하단 내비게이션 바와 반응형 레이아웃을 적용하여 스마트폰 환경에서 직관적이고 편리한 조작이 가능하도록 하였다.
- 피드백 제공 메커니즘:
  검사 결과는 모달 창과 알림 메시지로 사용자에게 명확히 전달되도록 설계하였다.

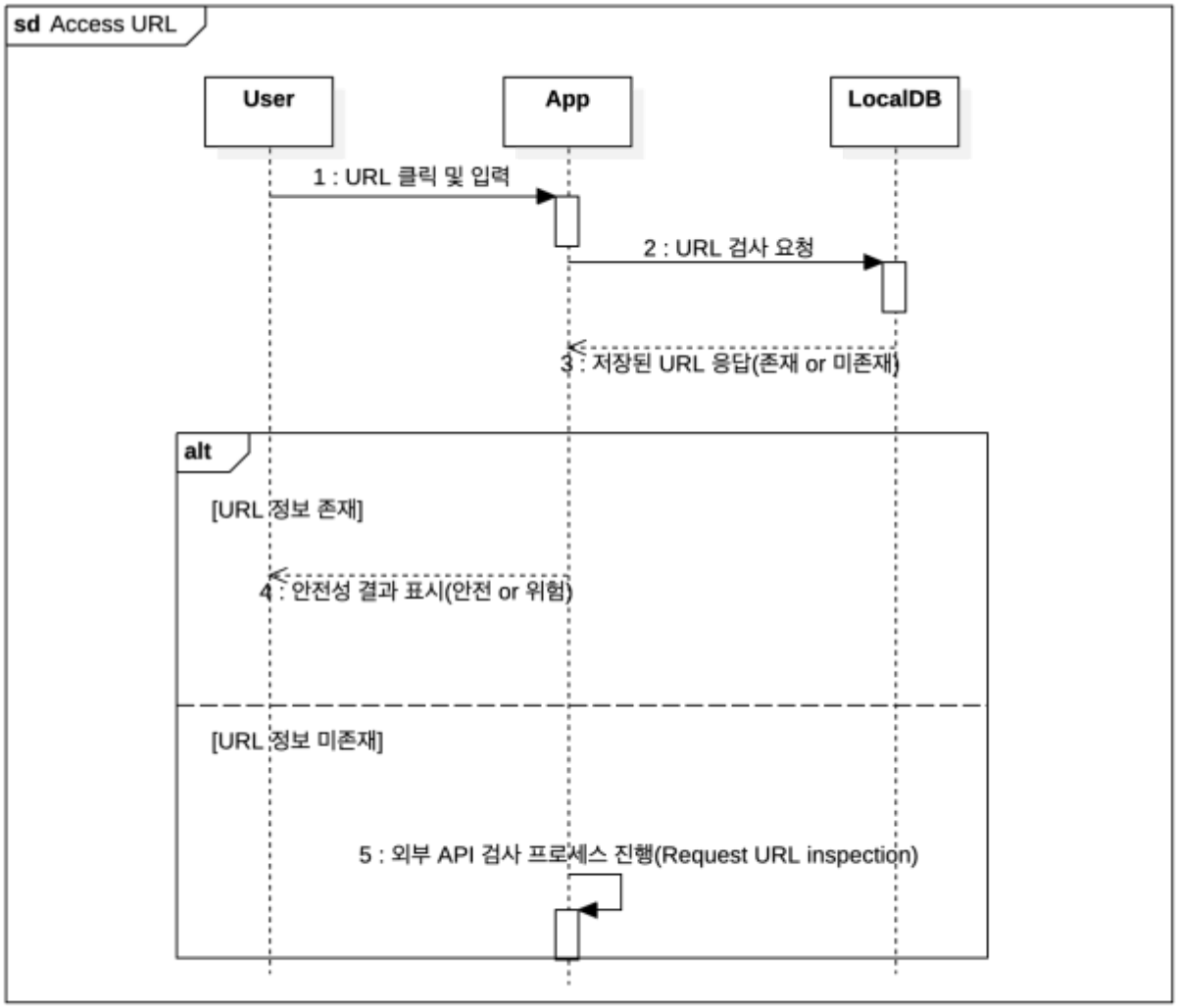
2. Class diagram



클래스명	설명
User	시스템을 사용하는 주체. URL 등록/검사 요청을 한다.
URL	검사 또는 등록의 대상이 되는 웹 주소, 등록 날짜, 위험 여부 등의 정보를 가진다.
InspectionResult	특정 URL에 대한 검사 결과(위험 여부, 검사 일시, 결과 설명 등)를 저장한다.
LocalServer	로컬 데이터베이스를 관리하고, URL 및 검사 결과의 저장/조회 역할을 한다.
URLRegistration	URL 등록 요청을 처리하며, 중복 확인과 유효성 검증을 담당한다.
URLInspection	URL 검사 요청을 처리하고, 필요시 외부 API에 검사 요청을 전달한다.
ResultDisplay	검사 결과를 사용자에게 제공하고, 결과를 저장한다.
InspectionAPIClient	외부 보안 API와 통신하여 검사 결과를 받아오고, 결과 포맷 변환 및 오류 처리를 한다.
URLRepository	로컬 DB에 URL과 검사 결과를 저장, 조회, 업데이트하는 역할을 한다.

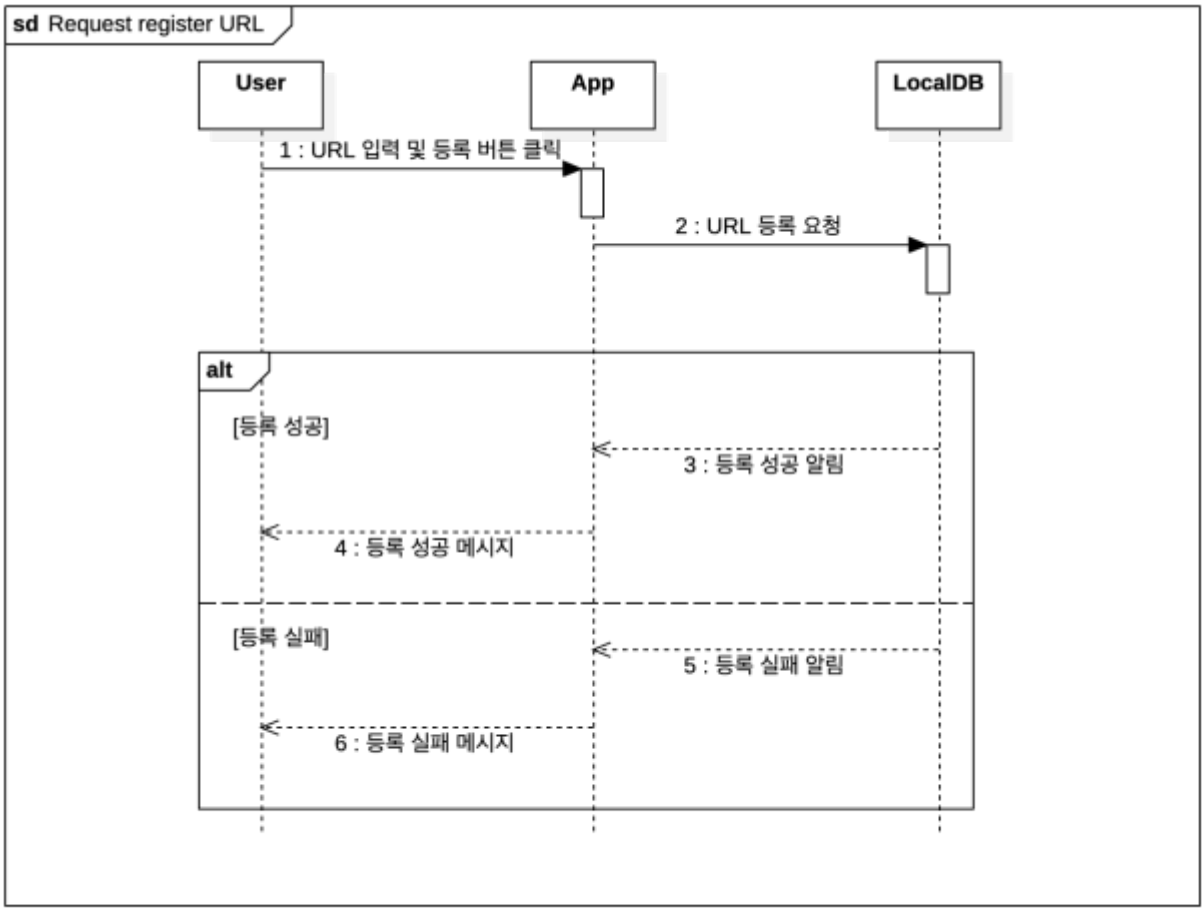
3. Sequence diagram

1) Access URL



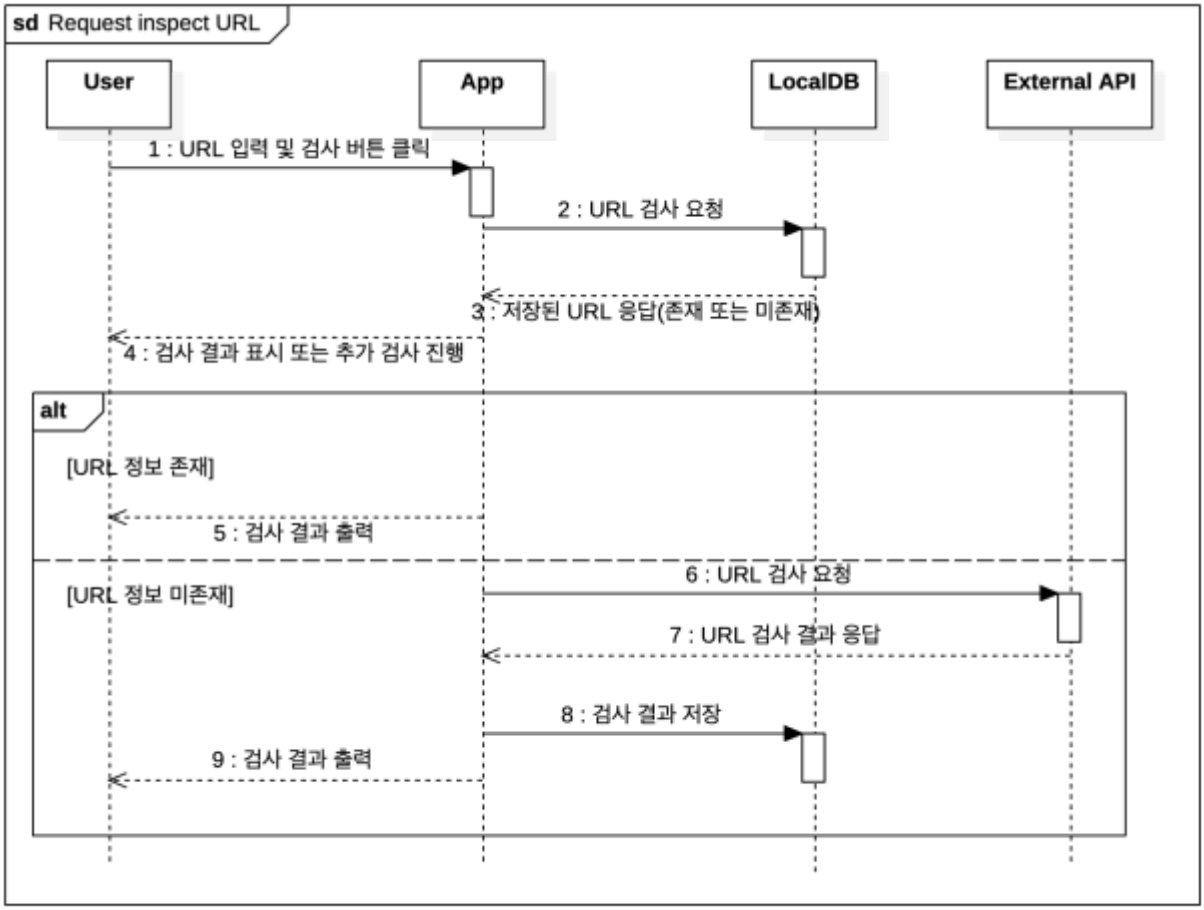
위 시퀀스 다이어그램은 사용자가 원하는 웹사이트(URL)에 접근할 때의 흐름을 보여준다. 사용자가 URL을 클릭하거나 입력하면, 애플리케이션(App)은 먼저 로컬 데이터베이스(LocalDB)에 해당 URL의 검사 이력이 있는지 확인 요청을 보낸다. LocalDB는 URL이 이미 저장되어 있는지(즉, 과거에 검사된 적이 있는지) 여부를 응답한다. 만약 검사 이력이 존재하면, 앱은 해당 URL의 안전성 결과(예: 안전/위험)를 사용자에게 즉시 안내한다. 만약 검사 이력이 없다면, 앱은 외부 API를 통한 추가 검사 프로세스를 진행한다. 이 과정은 사용자가 웹사이트에 접근하기 전에 위험 여부를 빠르게 확인하고, 위험한 경우 접근을 차단하거나 경고하는 데 목적이 있다.

2) Request register URL



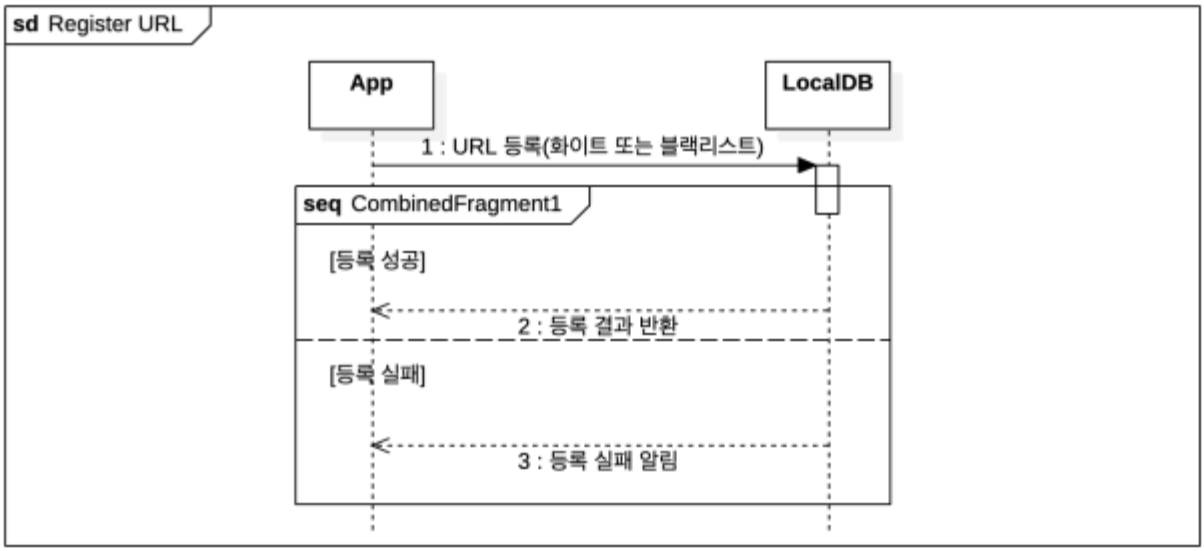
위 시퀀스 다이어그램은 사용자가 새로운 URL을 등록하는 과정을 나타낸다. 사용자는 애플리케이션의 입력창에 URL을 입력하고, 등록 버튼을 클릭한다. 앱은 입력된 URL을 로컬 데이터베이스에 저장(등록) 요청을 보낸다. LocalDB는 등록 시도 결과(성공 또는 실패)를 앱에 응답한다. 앱은 이 결과를 사용자에게 안내하며, 등록 성공 시에는 성공 메시지, 실패 시에는 실패 사유(예: 이미 등록된 URL, DB 오류 등)를 알려준다. 이 흐름은 사용자가 신뢰하는 URL을 직접 등록해 관리하거나, 위험 URL을 수동으로 차단하고자 할 때 활용된다.

3) Request inspect URL



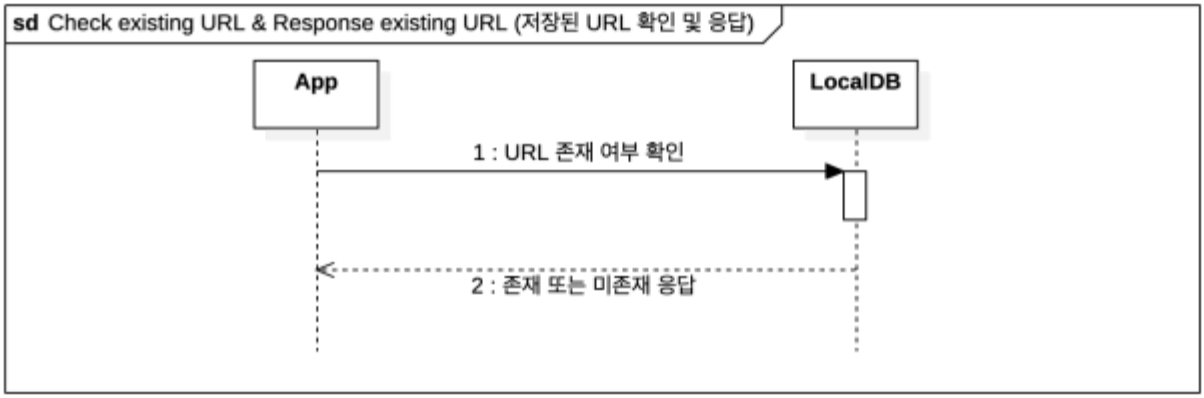
위 시퀀스 다이어그램은 사용자가 URL의 안전성을 검사하고자 할 때의 과정을 보여준다. 사용자가 검사 버튼을 클릭하면, 앱은 우선 로컬 데이터베이스에 해당 URL의 검사 이력이 있는지 확인한다. LocalDB가 검사 결과를 반환하면, 앱은 즉시 결과를 사용자에게 보여준다. 만약 검사 이력이 없다면, 앱은 외부 보안 API에 URL 검사 요청을 보내고, API로부터 검사 결과를 받은 뒤, 이 결과를 로컬 데이터베이스에 저장하고 사용자에게 안내한다. 이 과정은 실시간으로 URL의 위험 여부를 확인하고, 검사 결과를 기록해 추후 빠른 응답이 가능하도록 한다.

4) Register URL



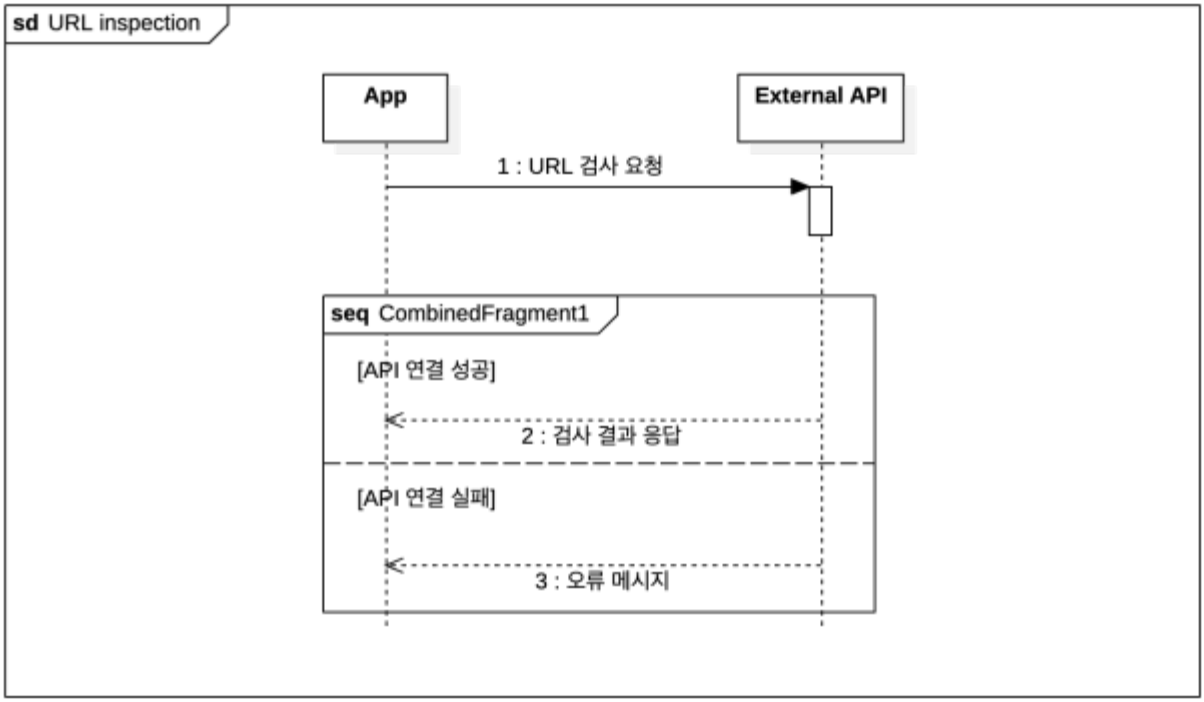
위 시퀀스 다이어그램은 시스템이 URL을 화이트리스트(안전) 또는 블랙리스트(위험)로 분류해 로컬 데이터베이스에 등록하는 내부 프로세스를 설명한다. 앱 또는 시스템은 검사 결과에 따라 URL을 분류하고, 해당 정보를 LocalDB에 저장 요청한다. LocalDB는 등록 성공 또는 실패 여부를 앱에 응답한다. 이 과정은 URL 검사 결과에 따라 자동으로 URL을 분류·저장하거나, 관리자가 수동으로 URL을 등록·차단할 때 사용된다. 이를 통해 반복적으로 검사할 필요 없이, 이미 분류된 URL에 대해 빠른 대응이 가능해진다.

5) Check existing URL & Response existing URL (저장된 URL 확인 및 응답)



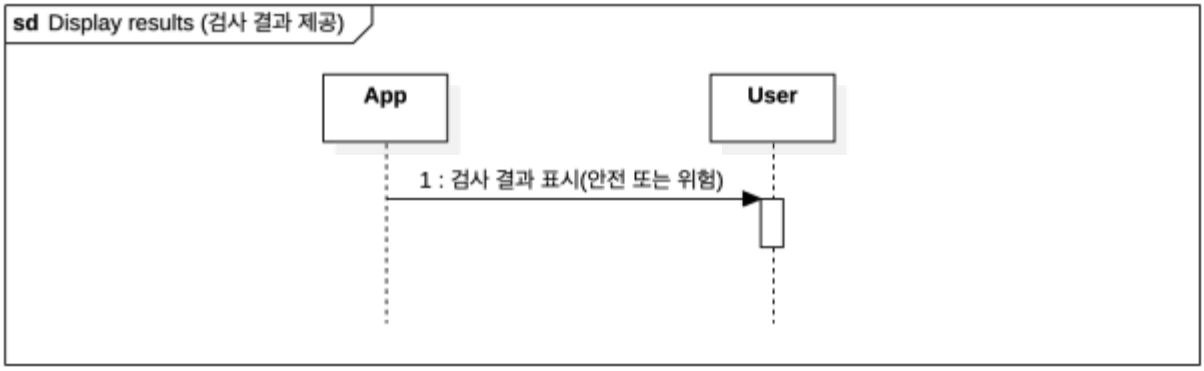
위 시퀀스 다이어그램은 앱이 로컬 데이터베이스에 URL의 존재 여부를 질의하고, LocalDB가 이에 대한 응답(존재/미존재)을 반환하는 과정을 보여준다. 이 과정은 URL 검사, 등록, 접근 등 모든 주요 프로세스에서 반복적으로 사용된다. 이미 검사된 URL의 경우 빠른 결과 제공이 가능하며, 정보가 없을 때만 외부 API 연동 등 추가 절차가 진행된다. 이로써 시스템의 효율성과 응답 속도를 높일 수 있다.

6) URL inspection



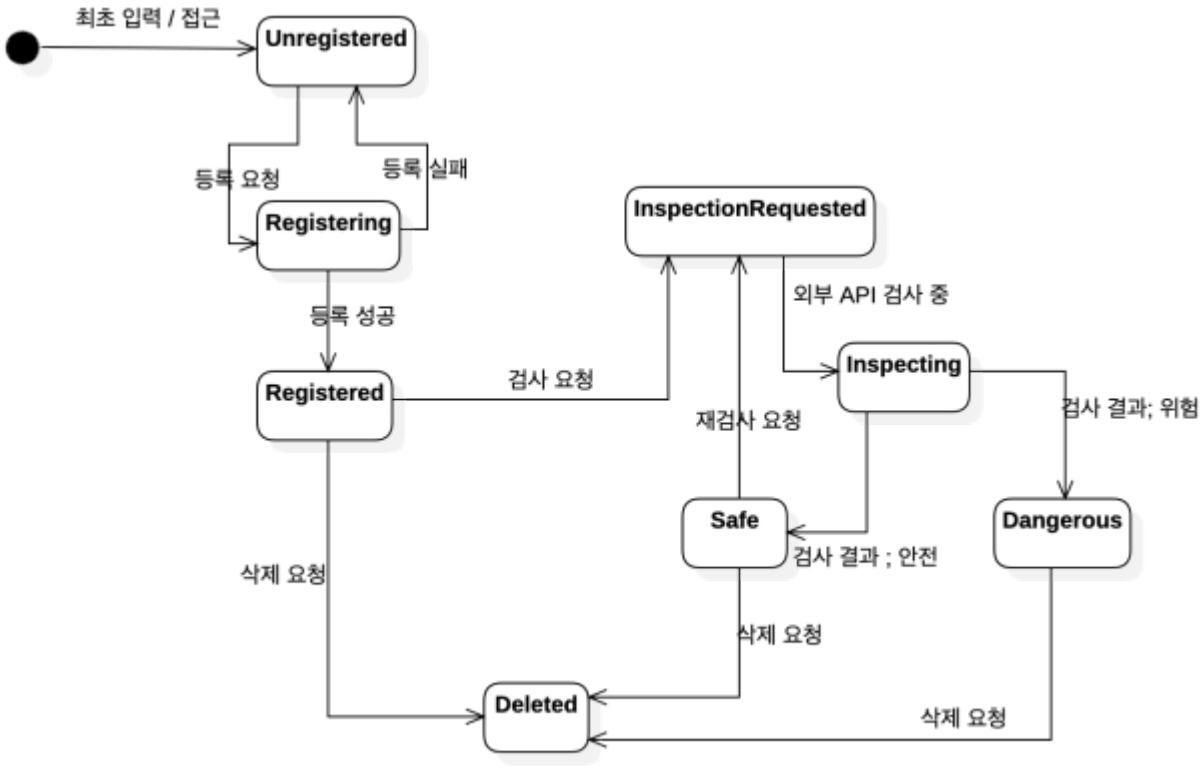
위 시퀀스 다이어그램은 앱이 외부 보안 API에 URL의 안전성 검사를 요청하는 과정을 나타낸다. 앱은 외부 API에 검사 요청 메시지를 보내고, API는 해당 URL을 검사한 뒤 결과(예: 안전/위험, 세부 설명 등)를 앱에 응답한다. 만약 API 연결에 실패하면 오류 메시지를 반환할 수 있다. 이 과정은 로컬 데이터베이스에 정보가 없는 새로운 URL에 대해 최신 보안 위협 정보를 실시간으로 반영하기 위해 필요하다. 외부 API의 결과는 이후 로컬DB에 저장되어, 동일 URL에 대한 반복 검사 시 빠른 응답이 가능하도록 한다.

7) Display results (검사 결과 제공)



위 시퀀스 다이어그램은 앱이 사용자에게 URL 검사 결과를 시각적으로 제공하는 과정을 설명한다. 앱은 검사 결과(안전/위험, 추가 정보 등)를 화면에 표시해 사용자가 즉시 확인할 수 있도록 한다. 만약 위험한 URL이라면 경고 메시지와 함께 접근 차단, 안전한 경우에는 정상 접근을 안내한다. 이 과정은 사용자 경험을 높이고, 실제 보안 사고를 사전에 예방하는 데 핵심적인 역할을 한다. 검사 결과는 필요에 따라 로컬 데이터베이스에 자동 저장되어, 추후 관리 및 재사용이 가능하다.

4. State Machine Diagram



상태명	설명
Unregistered (미등록)	URL이 시스템에 처음 입력되거나 접근된 상태입니다. 이 단계에서는 URL이 아직 등록되지 않았고, 검사 이력도 없습니다. 사용자가 URL을 등록하거나 검사 요청을 하면 다음 상태로 전이합니다.
Registered (등록됨)	URL이 로컬 데이터베이스에 등록된 상태입니다. 등록만 된 상태로, 아직 안전성 검사가 이루어지지 않았을 수 있습니다. 사용자가 검사 요청을 하면 InspectionRequested 상태로 이동합니다.
InspectionRequested (검사 요청됨)	URL의 안전성 검사가 요청된 상태입니다. 시스템은 로컬DB 또는 외부 API를 통해 URL의 위험 여부를 검사합니다. 검사 결과에 따라 InspectedSafe 또는 InspectedDangerous 상태로 전이합니다.
InspectedSafe (안전 판정)	검사 결과, 해당 URL이 안전하다고 판정된 상태입니다. 사용자는 안전하게 접근할 수 있습니다. 필요시 재검사 요청이 들어오면 InspectionRequested 상태로 다시 이동할 수 있습니다.
InspectedDangerous (위험 판정)	검사 결과, 해당 URL이 위험하다고 판정된 상태입니다. 사용자는 접근이 제한되거나 경고 메시지를 받게 됩니다. 필요시 재검사 요청이 들어오면 InspectionRequested 상태로 다시 이동할 수 있습니다.

5. Implementation requirements

1) Hardware requirements

CPU	2GHz 듀얼코어 이상(권장: 최신 ARM Cortex 또는 동급)
RAM	최소 4GB, 권장 6GB 이상
Storage	최소 320MB(앱 설치 및 데이터 저장), 여유 공간 확보 권장
네트워크	안정적인 Wi-Fi 또는 LTE/5G 인터넷 연결

2) Software requirements

Operating System	Android 9 이상(권장), Android 5.1 이상(최소), 또는 iOS 최신 버전
Implementation Language	[Front-end] React Native [Back-end] Flask
Local Database	Room
External API	Google Safe Browsing API, VirusTotal API 등 보안 URL 검사 API 사용

3) Nonfunctional requirements

SafeSurfing URL 안전 검사 애플리케이션의 비기능(Nonfunctional) 요구사항은 사용자 경험, 시스템의 신뢰성, 보안성, 그리고 확장성을 보장하기 위해 다음과 같이 정의된다. 본 시스템은 검사 요청에 대해 2초 이내의 빠른 응답 속도를 제공해야 하며, 다양한 Android 및 iOS 기기, 그리고 웹 브라우저 환경에서의 호환성을 갖추어야 한다. 데이터의 저장 및 전송 과정에서는 암호화와 인증을 적용하여 개인정보 및 검사 이력의 안전을 보장하고, 외부 API 키와 같은 민감 정보는 안전하게 관리되어야 한다. 또한, 시스템은 예기치 않은 오류나 장애 상황에서도 데이터 손실 없이 안정적으로 동작해야 하며, 24시간 365일 서비스 가용성을 유지해야 한다. 사용자 인터페이스는 직관적이고 접근성이 높아야 하며, 위험 URL 탐지 시 즉각적인 경고와 안내를 제공하여 사용자가 안전하게 웹을 탐색할 수 있도록 지원해야 한다. 마지막으로, 개인정보보호법 및 관련 규정(GDPR 등)을 준수하여 모든 사용자 데이터가 법적 기준에 따라 안전하게 처리되어야 하며, 시스템의 유지보수와 확장, 그리고 자동화된 업데이트가 용이하도록 설계되어야 한다.

6. Glossary

용어	정의/설명
ExternalAPI(외부 API)	Google Safe Browsing, VirusTotal 등 외부에서 제공하는 보안 검사 서비스의 프로그래밍 인터페이스. 외부 서버와 통신하여 URL의 안전성을 실시간으로 확인할 수 있다.
ORM(Object-Relational Mapping)	객체와 관계형 데이터베이스 간의 데이터를 자동으로 변환해주는 기술. Django, Room 등에서 사용된다.
암호화(Encryption)	데이터를 인가되지 않은 접근으로부터 보호하기 위해 특정 알고리즘을 사용해 내용을 변환하는 과정. 저장 및 전송 시 모두 적용될 수 있다.
인증(Authentication)	사용자의 신원이나 시스템 접근 권한을 확인하는 절차. 외부 API 사용 시 API Key를 통한 인증이 대표적이다.
데이터 무결성(Data Integrity)	데이터가 손상되거나 변조되지 않고 일관성과 정확성을 유지하는 특성. 시스템 장애나 오류 발생 시에도 데이터가 올바르게 보존되어야 한다.
접근성(Accessibility)	장애인 등 모든 사용자가 정보기술 서비스를 동등하게 이용할 수 있도록 보장하는 특성. UI 설계 시 고려해야 한다.
GDPR(General Data Protection Regulation)	유럽연합(EU)의 일반 개인정보 보호 규정. 개인정보 처리 및 보호에 관한 국제적 기준을 제시한다.
REST API	Representational State Transfer 아키텍처 스타일을 따르는 웹 기반 인터페이스. 클라이언트(React Native 등)와 서버(Flask, Django 등) 간 데이터 통신에 사용된다.
상태 다이어그램(State Diagram)	객체 또는 시스템의 상태 변화와 상태 간 전이(Transition)를 시각적으로 표현하는 UML 다이어그램의 한 종류.
시퀀스 다이어그램(Sequence Diagram)	시스템 내 여러 객체/컴포넌트 간의 메시지 흐름을 시간 순서대로 표현하는 UML 다이어그램.

마이그레이션(Migration)	데이터베이스 구조(스키마) 변경 시, 기존 데이터와 구조를 새로운 형태로 안전하게 이전하는 작업.
자동화된 업데이트(Automated Update)	사용자의 개입 없이 소프트웨어가 새로운 버전으로 자동으로 갱신되는 기능. 앱 스토어나 구글플레이를 통해 지원된다.

7. References

- Google Developers - Safe Browsing APIs
https://developers.google.com/safe-browsing/

- Flask 공식 문서
https://flask.palletsprojects.com/

- Android 개발자 문서 - Room Database
https://developer.android.com/training/data-storage/room

- React Native 공식 문서
https://reactnative.dev/