

프론트 기술 스택 선정 ing

- 모바일웹앱(PWA)으로 기존 서비스 제공
- 여기에 오버레이 기능을 추가하려고 하니 !!
네이티브 앱이 필요 → 네이티브 앱과 기존 PWA를 연결해서 구현하는 것을 생각

1. (기존) PWA을 활용한 모바일 웹앱 | react + pwa

기술 스택 : React + PWA

- PWA : 설치 없이 웹앱처럼 동작하며, **오프라인 지원**과 **푸시 알림** 등의 기능 제공
- 그러나 PWA 단독으로는 네이티브 플랫폼에 접근하기 어려움, **오버레이 UI** 기능 지원 X

한계

- 브라우저 범위 내에서만 동작, 화면 캡처와 같은 네이티브 API 호출에는 제약
- 네이티브와의 통합이 필요하면 **웹앱-네이티브 브릿지** 구축

2. 네이티브 앱 | Kotlin / react-native / Flutter

- 네이티브 앱 개발 시 오버레이 UI와 화면 캡처 기능 구현 가능

3. PWA을 활용한 모바일 웹앱 | 크롬 익스텐션

기술 스택 : React + PWA

- 크롬 익스텐션 : PWA를 보완하기 위해 특정 기능(오버레이 버튼, 화면 캡처) 제공
- 단, 브라우저 환경에서만 동작하며, 네이티브 앱만큼의 자유도는 없다.

한계

- 크롬 브라우저에서만 동작
- 브라우저 외부의 네이티브 기능(오버레이, 전체화면 캡처 등)을 구현할 수 없음

3주 안에 개발 가능한 방향

▼ 네이티브앱 vs 네이티브앱 + pwa | GPT 피셜

3주라는 짧은 개발 기간을 고려한다면 **최대한 효율적으로 빠르게 개발할 수 있는 방향**을 선택해야 합니다. 기획단계에서의 목표와 리소스를 바탕으로 최적의 선택을 도출해 보겠습니다.

1. PWA를 사용하지 않고 네이티브 앱으로만 개발 (추천)

• 이유:

- 오버레이 버튼, 화면 캡처 같은 디바이스 제어 기능은 네이티브 앱에서 구현이 더 간단하고 안정적입니다.
- PWA와 네이티브 앱을 통합하려면 추가적인 연결 작업(브릿지 구축)이 필요해 시간과 복잡도가 증가할 수 있습니다.
- Android 기반 개발(Kotlin 또는 Flutter)로 기능을 집중하면, 목표에 맞춰 빠르게 구현 가능.

• 필수 개발 요소:

1. 오버레이 버튼 구현:

- Android의 `WindowManager` 와 `SYSTEM_ALERT_WINDOW` 권한을 사용해 오버레이 버튼 생성.
- 버튼 UI와 동작(단일 클릭/더블 클릭) 정의.

2. 화면 캡처 기능:

- Android의 `MediaProjection` API를 사용해 화면 캡처 구현.
- 캡처된 이미지를 해당 웹 서버(PWA와 관계없이)로 전송.

3. 데이터 전송 로직:

- REST API 또는 WebSocket을 사용해 캡처 이미지와 URL을 서버로 전송.
- **장점:**
 - 빠른 개발 가능 (3주 내 구현 가능).
 - 디바이스 제어에 특화된 Android 네이티브 기능을 활용해 안정적 실행.
 - PWA 통합 없이 간소화된 코드베이스.
- **단점:**
 - Android 외의 플랫폼(iOS, 웹 사용자 등)에서는 동일한 경험을 제공하기 어려움.

2. PWA와 네이티브 앱을 통합 (시간이 더 걸릴 가능성 있음)

- **이유:**
 - 기존 PWA 서비스가 이미 완전히 구축되어 있고, 이를 활용하지 않으면 개발 시간이 더 길어질 경우에 적합.
 - 네이티브 앱에서 디바이스 제어(오버레이, 캡처)만 추가하고 나머지 기능은 PWA에서 처리 가능.
 - 하지만 PWA와 네이티브 앱 간 데이터 통신(예: Intent, Deep Link)을 구현해야 하기 때문에 시간과 리소스가 더 필요.

- **필수 개발 요소:**

1. **오버레이 버튼과 캡처 기능 (네이티브 앱):**

- 1번과 동일한 방식으로 구현.

2. **PWA와의 연결:**

- 네이티브 앱에서 캡처 데이터를 브릿지(Deep Link 또는 WebView)로 PWA에 전달.
- PWA는 데이터를 수신하고 추가 처리를 수행.

3. **PWA 서버 업데이트:**

- 기존 PWA가 캡처 데이터를 처리하고 저장할 수 있도록 API와 UI를 업데이트.
 - **장점:**
 - 기존 PWA의 기능을 재활용해 데이터 처리와 사용자 관리를 간소화.
 - 향후 Android 외 다른 플랫폼(iOS, 웹)으로 확장하기에 유리.
 - **단점:**
 - PWA와 네이티브 앱 간 통신 로직 개발로 인해 시간이 더 필요.
 - 브릿지 연결 작업이 추가적인 복잡도를 가져옴.
-

결정 기준

네이티브 앱만 사용할 경우 (추천):

- 빠르게 실행 가능한 오버레이와 캡처 기능만 필요하고, 웹/PWA와의 연결이 반드시 필요하지 않다면 이 방향이 가장 적합합니다.

PWA와 통합할 경우:

- 이미 구축된 PWA를 꼭 활용해야 하고, 캡처 데이터나 URL이 PWA 서버에서 처리되어야 하는 경우.

현실적인 실행 계획 (3주 안)

1. 기술 스택 결정:

- **Kotlin (Android 네이티브)** 사용 권장.
 - Flutter나 React Native도 가능하지만, Android 디바이스 제어에서 Kotlin이 더 직관적이고 안정적.

2. 주요 기능 개발 단계:

1. 오버레이 버튼 개발:

- Android `WindowManager` 를 사용해 항상 띠 있는 버튼 구현.
- 클릭/더블 클릭 이벤트 정의 및 동작 테스트.

2. 화면 캡처 구현:

- `MediaProjection` API로 캡처 기능 개발.

- 캡처 시 이미지 파일 저장 또는 메모리 내 데이터 생성.

3. 데이터 전송:

- 캡처된 이미지와 현재 URL을 REST API로 웹 서버로 전송.
- 간단한 백엔드 구축(Node.js, Flask 등으로 처리).

3. QA 및 배포:

- 모든 기능이 안정적으로 작동하는지 테스트.
 - Google Play 스토어에 배포.
-

결론

- **최우선 추천:** 네이티브 앱만으로 개발.
 - 시간, 복잡성, 안정성을 모두 고려했을 때 가장 적합.
- PWA 통합은 기존 서비스 활용이 반드시 필요하거나 Android 외 플랫폼 확장을 염두에 둘 때 선택.