걸즈밴드인포(가칭) **Flutter** 모바일 앱 개발 기획서

Part 1: 애플리케이션 기반 및 디자인 시스템

본 문서는 걸즈밴드 팬덤 커뮤니티 애플리케이션 '걸즈밴드인포'의 Flutter 클라이언트 개발을 위한 종합 기획서입니다. 이 기획서는 AI 코드 생성 에이전트가 명확하고 일관된 결과물을 도출할 수 있도록, 애플리케이션의 디자인 철학, 아키텍처, 기술 스택, 그리고 모든 기능의 상세 사양을 정의하는 것을 목표로 합니다. 본 기획서에서 명시하는 모든 기능은 추후 제공될 REST API를 기반으로 구현되는 것을 전제로 하며, 기존 기획서에 명시된 백엔드 아키텍처 및 GraphQL API 사양은 고려하지 않습니다.[1]

1.1. 핵심 디자인 철학**: "**데이터 기반 미니멀리즘 **(Data-Driven Minimalism)"**

본 애플리케이션의 시각적 및 상호작용 정체성은 금융 앱 토스(Toss)의 극도로 정제된 기능적 미학과 항공 정보 앱 Flighty의 정보 밀도 높은 명료함을 결합하여 정의됩니다. 이 모든 디자인 결정은 Apple의 Human Interface Guidelines (HIG)를 최우선 원칙으로 준수하여 iOS 사용자에게 최적의 경험을 제공하는 것을 목표로 합니다.[2]

1.1.1. 디자인 가이드 원칙

- 무엇보다 명확성 **(Clarity Above All):** 모든 화면은 핵심 정보를 모호함 없이 전달해야 합니다. 텍스트는 높은 가독성을 확보하고, 아이콘은 보편적으로 이해 가능해야 하며, 정보의 위계질서는 시각적으로 즉시 인지될 수 있도록 설계합니다.[3] 불필요한 장식 요소를 배제하고 본질에 집중하여 사용자의 인지 부하를 최소화합니다.[6]
- 콘텐츠에 대한 존중 **(Deference to Content):** UI 요소(크롬)는 사용자가 생성한 게시물,

장소 정보, 공식 정보와 같은 핵심 콘텐츠를 돋보이게 하는 보조적 역할을 수행하며, 콘텐츠와 경쟁하지 않습니다.[3] 이를 위해 반투명 효과나 블러(blur)를 적용한 '리퀴드 글래스(Liquid Glass)' 스타일을 적극 활용하여 UI가 콘텐츠 위에 자연스럽게 떠 있는 듯한 깊이감을 부여합니다.[9]

- 정돈된 정보 밀도 **(Information Density without Clutter):** Flighty 앱의 디자인에서 영감을 받아, 복잡한 데이터를 한눈에 파악할 수 있는 형태로 제공합니다.[11] 정보는 잘 구조화된 카드, 리스트, 타임라인 형식을 통해 맥락에 맞게 풍부하게 표현되지만, 결코 사용자를 압도하지 않도록 세심하게 설계됩니다. 충분한 여백과 명확한 시각적 구분선을 사용하여 정보의 가독성을 극대화합니다.[13]
- 직관적인 탐색 **(Effortless Navigation):** 사용자의 행동 흐름은 예측 가능하고 간결해야 합니다. 주요 기능은 대부분의 사용자가 한 손으로 기기를 파지했을 때 엄지손가락이 자연스럽게 닿는 'Thumb Zone' 내에 배치하여 접근성을 높입니다.[6] 앱의 핵심 기능으로의 진입점은 화면 하단에 고정된 탭 바(Tab Bar)를 통해 제공되며, 이는 iOS의 표준적이고 효율적인 내비게이션 패턴입니다.[15]

1.1.2. 디자인 영감의 통합**: Toss**와 **Flighty**의 조화

토스(Toss)의 미니멀리즘은 한 화면에 하나의 명확한 과업을 제시하며 정보의 양을 극단적으로 줄이는 방식으로 명확성을 달성합니다.[13] 반면, Flighty는 방대한 양의 정보를 고도로 구조화된 카드와 타임라인으로 조직화하여 복잡한 데이터를 한눈에 이해할 수 있도록 만듭니다.[12]

이 두 가지 상반되어 보이는 접근 방식의 공통점은 정보의 '양'이 아니라 정보를 전달하는 '효율성'에 있습니다. 본 앱은 이 두 스타일을 조화롭게 결합합니다. 즉, 토스와 같이 시각적으로 단순하고 정돈된 컴포넌트(카드, 버튼, 여백)를 '그릇'으로 삼고, 그 안에 Flighty처럼 풍부하고 맥락적인 데이터를 '내용물'로 담아낼 것입니다. 예를 들어, '장소 정보 카드'는 시각적으로는 단순한 사각형이지만, 그 안에는 장소 이름, 나의 방문 횟수, 현재 위치로부터의 거리 등 핵심 정보가 일목요연하게 표시됩니다. 이러한 "데이터 기반 미니멀리즘"은 앱의 모든 시각적 요소와 상호작용을 관통하는 핵심 철학이 될 것입니다.

1.1.3. 시각적 정체성 시스템 **(Visual Identity System)**

일관된 사용자 경험을 보장하기 위해 다음과 같은 시각적 정체성 시스템을 정의합니다.

표 **1:** 색상 팔레트 **(Color Palette)**

구분	Light Mode	Dark Mode	설명
Primary (주요)	#FFFFFF (배경)	#000000 (배경)	앱의 기본 배경색
Secondary (보조)	#F2F2F7 (카드 배경)	#1C1C1E (카드 배경)	카드, 그룹화된 목록 등 부가 요소의 배경색
Accent (강조)	#007AFF (Blue)	#0A84FF (Blue)	버튼, 링크, 활성화된 탭 등 상호작용 요소
Primary Text (주요 텍스트)	#000000	#FFFFFF	제목, 본문 등 주요 텍스트
Secondary Text (보조 텍스트)	rgba(60, 60, 67, 0.6)	rgba(235, 235, 245, 0.6)	부가 설명, 캡션 등 보조 텍스트
Success (성공)	#34C759 (Green)	#30D158 (Green)	인증 성공, 완료 등 긍정적 피드백
Error (오류)	#FF3B30 (Red)	#FF453A (Red)	인증 실패, 유효성 검사 실패 등 부정적 피드백

모든 색상 조합은 WCAG AA 레벨 이상의 명도 대비를 충족하도록 설계되었습니다.[7]

표 **2:** 타이포그래피 스케일 **(Typography Scale)**

요소	글꼴	크기 (pt)	굵기	색상
Large Title	SF Pro (시스템)	34	Bold	Primary Text
Title 1	SF Pro (시스템)	28	Bold	Primary Text
Title 2	SF Pro	22	Bold	Primary Text

	(시스템)			
Headline	SF Pro (시스템)	17	Semi-bold	Primary Text
Body	SF Pro (시스템)	17	Regular	Primary Text
Callout	SF Pro (시스템)	16	Regular	Primary Text
Subhead	SF Pro (시스템)	15	Regular	Secondary Text
Footnote	SF Pro (시스템)	13	Regular	Secondary Text
Caption	SF Pro (시스템)	12	Regular	Secondary Text

Flutter에서는 cupertino_icons 패키지와 시스템 폰트를 활용하여 iOS 네이티브 타이포그래피를 구현합니다.[15]

표 **3:** 아이코노그래피 **(Iconography)**

구분	설명	구현
아이콘 세트	Apple의 SF Symbols를 기본으로 사용합니다.	flutter_sf_symbols 또는 유사한 패키지를 통해 일관된 아이콘을 제공합니다.
스타일	선(line) 스타일을 기본으로 하며, 채워진(filled) 스타일은 활성화 상태를 나타낼 때 사용합니다.	아이콘은 명확하고, 단순하며, 보편적으로 인지 가능해야 합니다.[3]

사용 예시	공유, 편집, 설정, 검색 등 표준적인 기능에는 시스템 관례를 따르는 아이콘을 사용합니다.	CupertinoIcons 라이브러리를 적극 활용하여 iOS 사용자에게 익숙한 경험을 제공합니다.

1.2. 프론트엔드 아키텍처 및 기술 스택

애플리케이션의 확장성, 테스트 용이성, 개발 생산성을 극대화하기 위해 다음과 같은 기술적 청사진을 제시합니다.

1.2.1. 아키텍처 패턴**: MVVM (Model-View-ViewModel) + Repository Pattern**

Flutter 공식 문서에서 권장하는 관심사 분리(Separation of Concerns) 원칙에 따라 계층형 아키텍처를 채택합니다.[19]

- **View (**뷰**):** Flutter 위젯으로 구성되며, 오직 ViewModel의 상태를 기반으로 UI를 렌더링하는 책임만 가집니다. 뷰 레이어에는 최소한의 로직만 포함됩니다.[19]
- **ViewModel (**뷰모델**):** UI 상태를 관리하고, 데이터와 커맨드(함수)를 뷰에 노출합니다. 데이터의 요청 및 가공을 위해 Repository와 상호작용합니다.
- **Model (**모델**):** 순수 Dart 객체(Plain Old Dart Object)로, User, Place와 같은 데이터 구조를 표현합니다.
- **Repository (**리포지토리**):** 데이터에 대한 단일 진실 공급원(Single Source of Truth) 역할을 합니다. 데이터 소스를 추상화하여, 네트워크(API Service) 또는 로컬 캐시로부터 데이터를 가져오는 로직을 캡슐화합니다.

1.2.2. 기술 스택 및 도입 근거

- 상태 관리 **(State Management): flutter_riverpod**
  - 도입 근거**:** Riverpod는 컴파일 타임에 안전하고 유연한 최신 상태 관리 솔루션입니다. 위젯 트리로부터 상태를 분리하여 테스트 용이성과 확장성을 크게 향상시킵니다.[20] BLoC 패턴에 비해 보일러플레이트 코드가 현저히 적고 학습 곡선이 완만하여 개발 속도를 높일 수 있습니다.[21] Provider 기반 시스템은 전역적인 사용자 인증 상태부터 화면 단위의 일시적인 상태까지 다양한 범위의 상태를 효과적으로 관리하는 데 이상적입니다.

- 네트워크 통신 **(Network Communication): dio**
  - 도입 근거**:** dio는 Flutter 생태계에서 검증된 강력하고 성숙한 HTTP 클라이언트입니다. 특히 인터셉터(Interceptor) 기능은 모든 API 요청에 인증 토큰을 자동으로 삽입하는 등 공통 로직을 처리하는 데 필수적입니다. 또한, 강력한 오류 처리, 요청 취소, 전역 설정 등 프로덕션 레벨 애플리케이션에 필요한 고급 기능을 포괄적으로 지원합니다.[22]
- 지도 **(Mapping): flutter_map**
  - 도입 근거**:** 핵심 기능인 '성지 방문 인증'은 지도 표시를 요구합니다. google_maps_flutter는 훌륭한 옵션이지만, 네이티브 플랫폼 뷰에 의존하기 때문에 다수의 마커를 렌더링할 때 성능 저하가 발생할 수 있습니다.[25] flutter_map은 순수 Dart로 작성된 오픈소스 대안으로, 대규모 데이터셋에서 더 나은 성능을 제공하고, 높은 수준의 커스터마이징이 가능하며, 특정 벤더에 종속되지 않는 장점이 있습니다.[25] 따라서 수많은 '성지' 위치를 성능 저하 없이 표시하는 데 가장 적합한 선택입니다.
- 디바이스 권한 및 위치 정보**: permission_handler & geolocator**
  - 도입 근거**:** 각각 런타임 권한(특히 위치 정보 접근) 처리와 디바이스의 GPS 좌표를 가져오는 데 사용되는 산업 표준 패키지입니다. 안정적이고 지속적으로 유지보수되고 있어 신뢰성이 높습니다.

1.3. 전역 컴포넌트 및 내비게이션 설계

디자인 철학에 부합하는 일관된 사용자 경험을 제공하기 위해, 앱 전반의 내비게이션 구조와 재사용 가능한 UI 컴포넌트 라이브러리를 정의합니다.

1.3.1. 내비게이션 설계 **(Navigation Schema)**

- 루트 내비게이터 **(Root Navigator):** MaterialApp 위젯으로 앱의 최상위 내비게이션 스택을 구성합니다.
- 메인 내비게이션 **(Main Navigation):** Scaffold 위젯과 하단 CupertinoTabBar를 사용하여 4개의 주요 탭을 구성합니다. 이는 앱의 핵심 기능을 반영합니다.
  1. 홈**/**지도 **(Home/Map):** 기본 화면. 성지 및 라이브 장소를 지도 위에 표시합니다.
  2. 정보 **(Info):** 공식 정보 허브.
  3. 커뮤니티 **(Community):** 팬 커뮤니티 게시판.
  4. 프로필 **(Profile):** 사용자 프로필 및 방문 기록.
- 라우팅 **(Routing):** 중앙화된 라우팅 솔루션(예: GoRouter)을 사용하여 화면 간 이동, 파라미터 전달, 딥링킹(Deep Linking)을 체계적으로 관리합니다.

1.3.2. 컴포넌트 라이브러리 (Widgets)

자주 사용되는 UI 요소를 재사용 가능한 위젯으로 추상화하여 개발 효율성과 디자인 일관성을 높입니다.

- **AppScaffold:** 일관된 배경색과 안전 영역(Safe Area) 처리를 포함하는 표준 화면 래퍼(wrapper)입니다.
- **ThemedAppBar:** 스크롤 시 큰 타이틀이 작은 타이틀로 자연스럽게 축소되는 네이티브 iOS 동작을 모방한 커스텀 앱 바(App Bar)입니다.[15]
- **InfoCard:** 그림자, 모서리 둥글기, 콘텐츠 슬롯을 설정할 수 있는 다목적 카드 위젯입니다. Toss의 UI에서 영감을 받아 대부분의 목록 항목을 구성하는 기본 빌딩 블록으로 사용됩니다.[27]
- **PrimaryButton & SecondaryButton:** 주요 액션과 보조 액션을 위한 표준화된 스타일(패딩, 폰트, 색상)을 가진 버튼입니다.
- **LoadingIndicator & EmptyStateView:** 데이터 로딩, 오류 발생, 데이터 없음 상태를 사용자 친화적으로 처리하기 위한 표준화된 뷰를 제공하여 UX를 향상시킵니다.[6]

Part 2: 기능별 상세 구현 청사진

이 파트에서는 AI 에이전트가 직접 참조하여 코드를 생성할 수 있도록 각 기능의 화면 단위 상세 명세를 제공합니다.

2.1. 온보딩, 인증 및 사용자 프로필

- 사용자 플로우: 앱 최초 실행 → 온보딩 → 로그인/회원가입 → 프로필 화면
- 화면 명세:
  - 스플래시 화면 **(Splash Screen):** 앱 로고를 잠시 노출합니다.
  - 온보딩 화면 **(Onboarding Screens):** 3-4개의 슬라이드를 통해 앱의 핵심 가치(발견, 방문, 연결)를 시각적으로 전달합니다. 사용자가 원할 경우 건너뛸 수 있는 옵션을 반드시 제공해야 합니다.[6]
  - 로그인/회원가입 화면 **(Login/Sign-Up Screen):** 이메일/비밀번호 입력 필드와 소셜 로그인 버튼(예: Apple, Google)으로 구성된 미니멀한 폼을 제공합니다.
  - 프로필 화면 **(Profile Screen):** 사용자 정보(닉네임, 프로필 이미지)와 활동 요약(총 방문 횟수, 작성 게시글 수)을 표시합니다. 상단에는 주요 정보를 담은 헤더 카드를

배치하고, 그 아래로 '나의 방문 기록', '설정' 등 내비게이션 옵션 목록을 제공합니다.
    - 나의 방문 기록 화면 **(My Visit History Screen):** 사용자가 인증한 모든 방문 기록을 시간순으로 보여주는 스크롤 가능한 목록입니다. 각 항목은 InfoCard 컴포넌트를 사용하여 장소 이름, 마지막 방문일, 해당 장소의 총 방문 횟수를 표시합니다.

2.2. 성지 및 라이브 방문 인증

지도 기능과 게임화된 체크인 경험을 결합한 이 앱의 핵심 상호작용 기능입니다.

- 사용자 플로우**:** 지도 탭 진입 → 지도 위의 핀 선택 → 장소 상세 정보 확인 → 실제 장소로 이동 → '방문 인증하기' 버튼 탭 → 앱이 위치 검증 → 성공/실패 피드백 표시
- 화면 명세**:**
    - 지도 뷰 **(Map View -** 홈 탭**):**
        - 레이아웃**:** flutter_map 위젯을 전체 화면으로 사용하여 모든 Place 엔티티(성지 및 공연장)를 커스텀 스타일 마커로 표시합니다. 플로팅 액션 버튼(Floating Action Button)이나 하단 시트(Bottom Sheet)를 통해 지도 뷰와 목록 뷰를 전환할 수 있습니다.
        - 상태 관리**:** 서버로부터 모든 장소 목록을 가져와 표시합니다. 디바이스의 현재 위치를 지속적으로 수신하여 사용자의 위치를 지도 위에 표시합니다.
        - 상호작용**:** 마커를 탭하면 해당 장소의 장소 상세 화면으로 이동합니다.
    - 목록 뷰 **(List View):**
        - 레이아웃**:** 각 Place를 나타내는 InfoCard 위젯으로 구성된 수직 스크롤 ListView입니다. 거리순, 이름순 정렬 및 장소 유형(성지/공연장)에 따른 필터링 옵션을 제공합니다.
        - 상태 관리**:** 지도 뷰와 동일한 장소 목록 데이터를 사용합니다.
    - 장소 상세 화면 **(Location Detail Screen):**
        - 레이아웃**:** 상단에 장소의 대표 이미지를 크게 배치하고, 그 아래로 장소 설명, 주소, 정확한 위치를 보여주는 작은 지도 뷰, 그리고 해당 장소에 대한 사용자의 개인 방문 기록("이 장소에 3번 방문했어요")을 담은 카드를 순차적으로 표시합니다. 화면 하단에는 '방문 인증하기' PrimaryButton이 눈에 띄게 배치됩니다.
    - 방문 인증 플로우 **(**모달**):**
        - 상호작용**:** '방문 인증하기' 버튼을 탭하면 인증 절차가 시작됩니다.
        - **1**단계 **(**권한 확인**):** 위치 정보 접근 권한을 확인합니다. 권한이 없는 경우, 권한이 필요한 이유를 설명하는 시스템 다이얼로그를 표시합니다.
        - **2**단계 **(**위치 정보 수집**):** 현재 GPS 좌표를 수집합니다. 이 과정 동안 로딩 인디케이터를 표시합니다.
        - **3**단계 **(API** 호출**):** 수집된 좌표를 백엔드 서버로 전송하여 유효성을 검증합니다.
        - **4**단계 **(**피드백**):** 인증 성공 시, 축하 애니메이션과 햅틱 피드백을 포함한 성공 모달을 표시합니다. 실패 시, "장소로부터 너무 멉니다."와 같이 실패 사유를 명확히 설명하는 모달을 표시합니다.

2.2.1. 위치 위변조 방지를 위한 클라이언트의 역할

기존 백엔드 기획서는 GPS 스푸핑(spoofing) 방지를 위해 모의 위치 앱 탐지, 이동 패턴 분석 등 다층적 보안 전략을 제시하고 있습니다.[1] 이러한 보안 체계가 효과적으로 작동하려면 클라이언트 앱의 적극적인 역할이 필수적입니다. 백엔드 서버는 클라이언트 기기에서 모의 위치 기능이 활성화되었는지 직접 확인할 수 없기 때문입니다.

따라서, '방문 인증하기' 기능은 단순히 위도와 경도 좌표만 서버로 전송하는 것을 넘어, '위치 증명(Proof of Location)' 데이터를 수집하여 전송해야 합니다. 이 데이터 페이로드에는 다음과 같은 정보가 포함되어야 합니다.

1. 위도 및 경도 **(latitude, longitude)**
2. 위치 정확도 **(accuracy)**: GPS 신호의 정확도 미터 값.
3. 타임스탬프 **(timestamp)**: 위치 정보가 수집된 정확한 시각.
4. 모의 위치 여부 **(isMocked)**: geolocator 패키지 등을 통해 확인한 모의 위치(mock location) 사용 여부 플래그.
5. 기타 디바이스 정보: 개발자 모드 활성화 여부 등 추가적인 신뢰도 판단 지표.

이처럼 풍부한 메타데이터를 포함한 페이로드를 서버로 전송함으로써, 서버는 더 정교한 위변조 탐지 로직을 수행할 수 있습니다. 이는 서비스의 핵심 가치인 '인증'의 신뢰도를 보장하기 위한 매우 중요한 클라이언트 측 구현 요구사항입니다.

2.3. 정보 허브

공식적으로 제공되는 콘텐츠를 사용자가 쉽게 탐색하고 발견할 수 있도록 설계된 읽기 전용 기능입니다.

- 사용자 플로우**:** 정보 탭 진입 → 카테고리 탐색 → 항목 선택 → 상세 페이지 조회
- 화면 명세**:**
  - 정보 허브 대시보드 **(Info Hub Dashboard):** '캐릭터', '밴드', '음반 정보', '이벤트' 등 주요 카테고리를 그리드나 리스트 형태로 보여줍니다.
  - 목록 화면 **(List Screens):** 각 카테고리에 해당하는 항목 목록을 보여줍니다. 예를 들어 '캐릭터 목록' 화면은 각 캐릭터의 이미지와 이름을 담은 InfoCard의 ListView 또는 GridView로 구성됩니다.
  - 상세 화면 **(Detail Screens):** 선택된 항목의 상세 정보를 표시합니다. 예를 들어 '캐릭터 상세' 화면은 큰 이미지, 인물 소개 텍스트, 그리고 관련된 다른 정보(소속 밴드, 참여 앨범 등)로 연결되는 링크를 포함합니다.

2.4. 커뮤니티 포럼

팬들이 서로 소통하고 정보를 교류하는 앱의 사회적 중심 기능입니다.

- 사용자 플로우**:** 커뮤니티 탭 진입 → 게시글 목록 스크롤 → 게시글 선택하여 댓글 확인 → 새 게시글 작성
- 화면 명세**:**
  - 게시글 목록 화면 **(Post List Screen):** 커뮤니티 게시글을 최신순 또는 인기순으로 보여주는 무한 스크롤 목록입니다. 각 게시글은 제목, 작성자, 내용 일부, 그리고 참여도(좋아요 수, 댓글 수)를 보여주는 커스텀 카드로 표시됩니다.
  - 게시글 상세 화면 **(Post Detail Screen):** 게시글의 전체 내용과 그 아래로 시간순으로 정렬된 댓글 목록을 표시합니다. 화면 하단에는 새 댓글을 작성할 수 있는 입력 필드를 제공합니다.
  - 새 게시글 작성 화면 **(New Post Screen):** 제목과 여러 줄의 본문을 입력할 수 있는 간단한 폼과 '등록' 버튼으로 구성됩니다.

Part 3: 클라이언트 로직 및 데이터 관리

이 파트에서는 애플리케이션의 내부 동작에 대한 저수준 기술 명세를 제공합니다.

3.1. Riverpod를 활용한 상태 관리 전략

명확한 Provider 계층 구조를 정의하여 상태를 체계적으로 관리합니다.

- **Provider** 종류 및 역할**:**
  - **AuthRepositoryProvider (**싱글턴**):** 사용자의 인증 상태(로그인 여부, 사용자 객체, 인증 토큰)를 전역적으로 관리합니다. 앱의 생명주기와 동일하게 유지됩니다.
  - **PlacesRepositoryProvider (**싱글턴**):** 모든 성지 및 공연장 목록을 가져오고 캐시하는 역할을 담당합니다.
  - **Family Provider:** 특정 ID에 의존하는 데이터를 가져올 때 사용됩니다. 예: postDetailProvider(postId)는 특정 게시글의 상세 정보를 가져옵니다.
  - **StateNotifierProvider:** ViewModel 내에서 변경 가능한 UI 상태(예: 폼 입력 필드의 텍스트, 목록의 정렬 옵션)를 관리하는 데 사용됩니다. 뷰는 이 Provider를 구독하여 상태 변화에 따라 자동으로 다시 빌드됩니다.

3.2. API 통합 계층 및 계약 정의

Flutter 앱이 백엔드 REST API와 상호작용하는 방식을 명확히 정의합니다.

- **ApiService (Dio 기반):**
  - 모든 HTTP 통신을 담당하는 싱글턴 클래스입니다.
  - 기본 URL(Base URL)과 같은 공통 설정을 포함하여 초기화됩니다.
  - dio 인스턴스에 AuthInterceptor가 추가됩니다. 이 인터셉터는 AuthRepository로부터 인증 토큰을 읽어와, 로그인/회원가입을 제외한 모든 요청의 Authorization 헤더에 자동으로 추가하는 역할을 수행합니다.
- 표 4: REST API 계약 (REST API Contracts)
  이 표는 앱이 필요로 하는 모든 API 엔드포인트를 정의하며, AI 에이전트가 네트워크 계층을 생성하기 위한 명확한 가이드 역할을 합니다.

기능	엔드포인트	메소드	설명	요청 본문 (JSON)	성공 응답 (JSON)
인증	/auth/login	POST	이메일/비밀번호로 로그인	{ "email": "...", "password": "..." }	{ "token": "...", "user": {... } }
인증	/auth/register	POST	신규 사용자 회원가입	{ "email": "...", "password": "...", "username": "..." }	{ "token": "...", "user": {... } }
장소	/places	GET	모든 성지 및 공연장 목록 조회	null	``
장소	/places/{id}	GET	단일 장소 상세 정보 조회	null	{ "id": "...", "name": "...", "description": "...",

					"imageUrl": "...",... }
방문 인증	/visits/authenticate	POST	사용자 방문 인증 요청	{ "placeId": "...", "latitude":..., "longitude": ..., "accuracy": ..., "isMocked": ... }	{ "id": "...", "placeId": "...", "userId": "...", "visitCount":..., "lastVisitTimestamp": "..." }
방문 기록	/users/me/visits	GET	나의 전체 방문 기록 조회	null	``
커뮤니티	/posts?page={num}&size={num}	GET	커뮤니티 게시글 목록 조회 (페이지네이션)	null	{ "content": [{ "id": "...", "title": "...", "author": {... }, "commentCount":... }], "totalPages":..., "totalElements":... }
커뮤니티	/posts/{id}	GET	단일 게시글 및 댓글 조회	null	{ "id": "...", "title": "...", "content": "...", "author": {... }, "comments": [...] }
커뮤니티	/posts	POST	새 게시글	{ "title": "...", "content":	{ "id": "...", "title": "...",

			작성	"..." }	"content": "...",... }
커뮤니티	/posts/{id}/comments	POST	새 댓글 작성	{ "content": "..." }	{ "id": "...", "content": "...", "author": {... } }

3.3. 클라이언트 데이터 모델 및 영속성

- **Dart** 데이터 모델**:** API 계약 표에 정의된 각 JSON 객체에 대응하는 불변(immutable) Dart 클래스(User.dart, Place.dart, Post.dart 등)를 생성합니다. 이 클래스들은 API 응답을 쉽게 파싱할 수 있도록 fromJson 팩토리 생성자를 포함해야 합니다.
- 로컬 영속성 **(Local Persistence):**
  - **shared_preferences:** 사용자의 인증 토큰, 테마 설정(라이트/다크 모드) 등 간단한 키-값 데이터를 저장하는 데 사용됩니다.
  - 캐싱 전략 **(Caching Strategy):** Repository 계층에서 자주 변경되지 않는 데이터(예: 전체 장소 목록)에 대해 간단한 인-메모리 캐싱 전략을 구현하여 불필요한 네트워크 요청을 줄이고 앱 성능을 향상시킵니다. 향후 더 복잡한 오프라인 지원이 필요할 경우, Isar 또는 Drift와 같은 로컬 데이터베이스 도입을 고려할 수 있습니다.

결론

본 기획서는 '걸즈밴드인포' Flutter 모바일 애플리케이션 개발을 위한 포괄적인 청사진을 제시합니다. "데이터 기반 미니멀리즘"이라는 명확한 디자인 철학 아래, iOS 사용자에게 최적화된 미려하고 직관적인 사용자 경험을 제공하는 것을 목표로 합니다. 기술적으로는 Riverpod, Dio, flutter_map 등 현대적이고 검증된 기술 스택을 기반으로 한 MVVM 아키텍처를 채택하여, 유지보수성과 확장성이 뛰어난 고품질 애플리케이션을 구축할 기반을 마련했습니다.

특히, 각 기능에 대한 상세한 화면 명세와 명확하게 정의된 REST API 계약은 AI 코드 생성 에이전트가 모호함 없이 일관된 결과물을 생성하는 데 결정적인 역할을 할 것입니다. 또한, 서비스 신뢰도의 핵심인 위치 인증 기능에 대해 클라이언트 측의 적극적인 역할을 명시함으로써, 단순한 기능 구현을 넘어 서비스의 근본적인 가치를 지키기 위한 기술적

요구사항을 구체화했습니다.

본 기획서를 바탕으로 개발이 진행된다면, 기술적으로 견고하고 사용자에게 사랑받는 성공적인 팬덤 커뮤니티 플랫폼이 탄생할 것으로 기대합니다.

참고 자료

1. '걸즈밴드인포' 서비스 개발 기획서.pdf
2. Human Interface Guidelines | Apple Developer Documentation, 9월 30, 2025에 액세스, https://developer.apple.com/design/human-interface-guidelines
3. iOS App UI/UX Design Guidelines: You Must Follow in 2024 - Bitcot, 9월 30, 2025에 액세스, https://www.bitcot.com/ios-app-design-guidelines/
4. What is iOS Human Interface Guidelines? - Pangea.app, 9월 30, 2025에 액세스, https://pangea.app/glossary/ios-human-interface-guidelines
5. Insights from Apple's Human Interface Design Guidelines - Hashkaro Digitals, 9월 30, 2025에 액세스, https://hashkarodigitals.com/2023/08/18/insights-from-apples-human-interface-design-guidelines-2/
6. Mobile App Design: 8 UX Principles for Better Engagement - Celerart, 9월 30, 2025에 액세스, https://celerart.com/blog/mobile-app-design-8-ux-principles-for-better-engagement
7. The Role of Apple Human Interface Guidelines in Effective App Design - MoldStud, 9월 30, 2025에 액세스, https://moldstud.com/articles/p-the-role-of-apple-human-interface-guidelines
8. About UX: An Overview of iOS Human Interface Guidelines | by AxureBoutique - Medium, 9월 30, 2025에 액세스, https://axureboutique.medium.com/about-ux-an-overview-of-ios-human-interface-guidelines-4b255855b418
9. Get to know the new design system - WWDC25 - Videos - Apple Developer, 9월 30, 2025에 액세스, https://developer.apple.com/videos/play/wwdc2025/356/
10. Design - Apple Developer, 9월 30, 2025에 액세스, https://developer.apple.com/design/
11. Flighty – Live Flight Tracker on the App Store, 9월 30, 2025에 액세스, https://apps.apple.com/us/app/flighty-live-flight-tracker/id1358823008
12. Review: Why the Flighty Pro App is Still Our Favorite Travel Tool, 9월 30, 2025에 액세스, https://thriftytraveler.com/reviews/flighty-pro-app/
13. Designing dashboards that actually work | by Matheus Moura | Bootcamp - Medium, 9월 30, 2025에 액세스, https://medium.com/design-bootcamp/designing-dashboards-that-actually-work-55ddb467b1b9
14. 5 fundamental app design principles you need to follow - DECODE, 9월 30, 2025에 액세스, https://decode.agency/article/mobile-app-design-principles/
15. The iOS 17 Design Guidelines: An Illustrated Guide, 9월 30, 2025에 액세스, https://www.learnui.design/blog/ios-design-guidelines-templates.html

16. What is super app Toss & how it works? - IdeaUsher, 9월 30, 2025에 액세스, https://ideausher.com/blog/what-is-super-app-toss/
17. Flighty App Review - BoardingGroup.One, 9월 30, 2025에 액세스, https://boardinggroup.one/review/products/flighty-app-review/
18. iOS App Design Guidelines for 2025 - BairesDev, 9월 30, 2025에 액세스, https://www.bairesdev.com/blog/ios-design-guideline/
19. Guide to app architecture - Flutter Documentation, 9월 30, 2025에 액세스, https://docs.flutter.dev/app-architecture/guide
20. What's the Best State Management Library for Flutter in 2025? - Foresight Mobile, 9월 30, 2025에 액세스, https://foresightmobile.com/blog/whats-the-best-state-management-library-for-flutter
21. A Comprehensive Guide to Riverpod Vs. BLoC in Flutter - DhiWise, 9월 30, 2025에 액세스, https://www.dhiwise.com/post/flutter-insights-navigating-the-riverpod-vs-bloc-puzzle
22. Flutter Dio Tutorial: The Ultimate HTTP Client for Flutter Development - Mobisoft Infotech, 9월 30, 2025에 액세스, https://mobisoftinfotech.com/resources/blog/flutter-development/flutter-dio-tutorial-http-client
23. dio | Dart package - Pub.dev, 9월 30, 2025에 액세스, https://pub.dev/packages/dio
24. How to Use REST APIs in Flutter with Dio? - Technaureus Info Solutions, 9월 30, 2025에 액세스, https://www.technaureus.com/blog-detail/how-to-use-rest-api-in-flutter-with-dio
25. Thoughts on google_maps_flutter vs. flutter_map for Large Numbers of Markers - Reddit, 9월 30, 2025에 액세스, https://www.reddit.com/r/FlutterDev/comments/1i2ldzn/thoughts_on_google_maps_flutter_vs_flutter_map/
26. flutter_map | flutter_map Docs, 9월 30, 2025에 액세스, https://docs.fleaflet.dev/
27. bernadinkele/flutter-ui-card-collections - GitHub, 9월 30, 2025에 액세스, https://github.com/bernadinkele/flutter-ui-card-collections
28. Mobile App UX Principles | Think with Google, 9월 30, 2025에 액세스, https://www.thinkwithgoogle.com/_qs/documents/2081/Mobile_App_UX_Principles_3.pdf