App Store

Google Play

사진찍고 포인트받자!

찍고머니

Frontend Developer 이동훈

찍고머니

프로젝트 개요

- 멀티버스 플랫폼 '제프월드' 연동 서비스
- '찍고머니'는 사용자의 사진 촬영 등의 일상 활동을 통해 포인트를 적립받는 리워드 기반 모바일 애플리케이션입니다.
- Flutter 기반의 모바일 앱과 Next.js 기반의 어드민 페이지로 구성
- 거래 불가한 NFT를 발급하여 사용자에게 거부감 없는 Web3 환경 제공
- 보상 기반 UX 설계를 통해 사용자의 참여도를 유도하며, 앱 내 다양한 미션과 광고 연계 시스템을 통해 사용자 경험과 수익화를 동시에 고려한 서비스입니다.

개발 일정

- 2024.12.01 ~ 2025.3.28, 3월 20일 1차 정식 배포

기술 스택

- Mobile App : Flutter
- Admin Page: Next.js
- ETC : Firebase, Android Studio, Xcode, Figma, Notion

담당 역할

- 1. 모바일 프론트엔드 전담 개발
  - **Flutter** 기반의 전체 앱 구조 설계와 핵심 기능 화면 개발을 주도하였습니다.
  - 사용자 인터랙션이 많은 메인 화면에 대해 화면 전환 시 라이프사이클을 고려하여 렌더링 최적화 및 불필요한 **rebuild** 최소화.
- 2. 상태 관리 및 아키텍처 설계
  - **Riverpod**과 **GetX**를 혼합한 하이브리드 상태 관리 시스템을 도입하여 의존성 주입과 반응형 **UI** 구현을 분리하고 효율적으로 처리.
  - **MVVM** 아키텍처 패턴을 기반으로 앱 구조를 모듈화하여 데이터, 비즈니스 로직, **UI**를 명확히 분리하고 유지보수성과 테스트 편의성을 확보했습니다.

찍고머니

MVVM 폴더 구조

```
lib/
├── model/          # 데이터 모델과 비즈니스 로직
│   ├── dio.dart    # 네트워크 통신 관련
│   ├── apis.dart   # API 엔드포인트 정의
│   ├── auth_service.dart  # 인증 관련 서비스
│   └── utils.dart  # 유틸리티 함수들
│
├── viewModel/      # 상태 관리 및 비즈니스 로직
│   ├── auth/       # 인증 관련 뷰모델
│   ├── home/       # 홈 화면 관련 뷰모델
│   └── providers/  # Riverpod 프로바이더들
│
├── view/           # UI 컴포넌트
│   ├── login/      # 로그인 관련 화면
│   ├── main/       # 메인 화면 컴포넌트
│   └── common/     # 공통 UI 컴포넌트
│
├── main.dart       # 앱 진입점
└── theme.dart      # 테마 정의
```

화면 라이프사이클 관리

```dart
class _MainPageState extends State<MainPage> {
  // 네비게이터 키로 각 탭의 상태 유지
  final List<GlobalKey<NavigatorState>> _navigatorKeys = [
    GlobalKey<NavigatorState>(),
    GlobalKey<NavigatorState>(),
    GlobalKey<NavigatorState>(),
    GlobalKey<NavigatorState>(),
  ];

  // 탭 전환 시 페이지 상태 관리
  void _setPages(int index, double? latitude, double? longitude) {
    setState(() {
      switch (index) {
        case 0:
          pages = [const Homepage(), const SizedBox(), const SizedBox(), const SizedBox()];
          break;
        // ... 다른 탭들
      }
      selectedIndex = index;
    });
  }

  // 탭 클릭 시 상태 관리
  void _onTap(int index, double? latitude, double? longitude) {
    if (index == selectedIndex) {
      _fetchUserInfo();
      _navigatorKeys[index].currentState?.popUntil((route) => route.isFirst);
    } else {
      if (index != 1) {
        _fetchUserInfo();
      }
      _setPages(index, latitude, longitude);
    }
  }
}
```

네트워크 및 최적화

찍고머니

- 네트워크 통신 최적화
  - Retrofit과 Dio를 활용하여 API 통신의 타입 안정성과 예외 처리 일관성을 강화.
  - 토큰 갱신 로직에서 중복 요청 방지 로직을 직접 구현하여, 요청 횟수를 기존 평균 5회 → 2회로 감소시키며 서버 부하를 약 60% 줄였습니다.
- 사용자 경험 향상을 위한 기능 구현
  - Firebase Analytics로 사용자 행동을 추적, 활용도 높은 미션 기능을 고도화.
  - Firebase Messaging을 통해 푸시 및 인앱 알림 기능을 구현하고, 타겟 맞춤형 메시지 전달이 가능하도록 구성하였습니다.
  - AdMob과 연동하여 광고 기반 수익화 구조도 설계.
- 기타 기능 및 SDK 통합
  - 네이버 지도 SDK 연동으로 위치 기반 미션 기능 제공
  - easy_localization 패키지를 사용하여 다국어 지원이 가능한 글로벌 환경 대응 구조를 준비했습니다.

Retrofit API 인터페이스

```dart
@RestApi()
abstract class MissionAPI {
  factory MissionAPI(Dio dio) = _MissionAPI;

  @GET("/missions")
  Future<ApiResponse<List<Mission>>> getMissions(
    @Query("page") int page,
    @Query("size") int size,
    @Query("category") String category,
  );

  @POST("/missions/{missionId}/complete")
  Future<ApiResponse<MissionComplete>> completeMission(
    @Path("missionId") int missionId,
    @Body() CompleteMissionRequest request,
  );

  @GET("/missions/{missionId}")
  Future<ApiResponse<MissionDetail>> getMissionDetail(
    @Path("missionId") int missionId,
  );
}

// 응답 모델
@JsonSerializable()
class ApiResponse<T> {
  final T data;
  final String message;
  final int statusCode;

  ApiResponse({
    required this.data,
    required this.message,
    required this.statusCode,
  });

  factory ApiResponse.fromJson(Map<String, dynamic> json) =>
      _$ApiResponseFromJson(json);
}
```

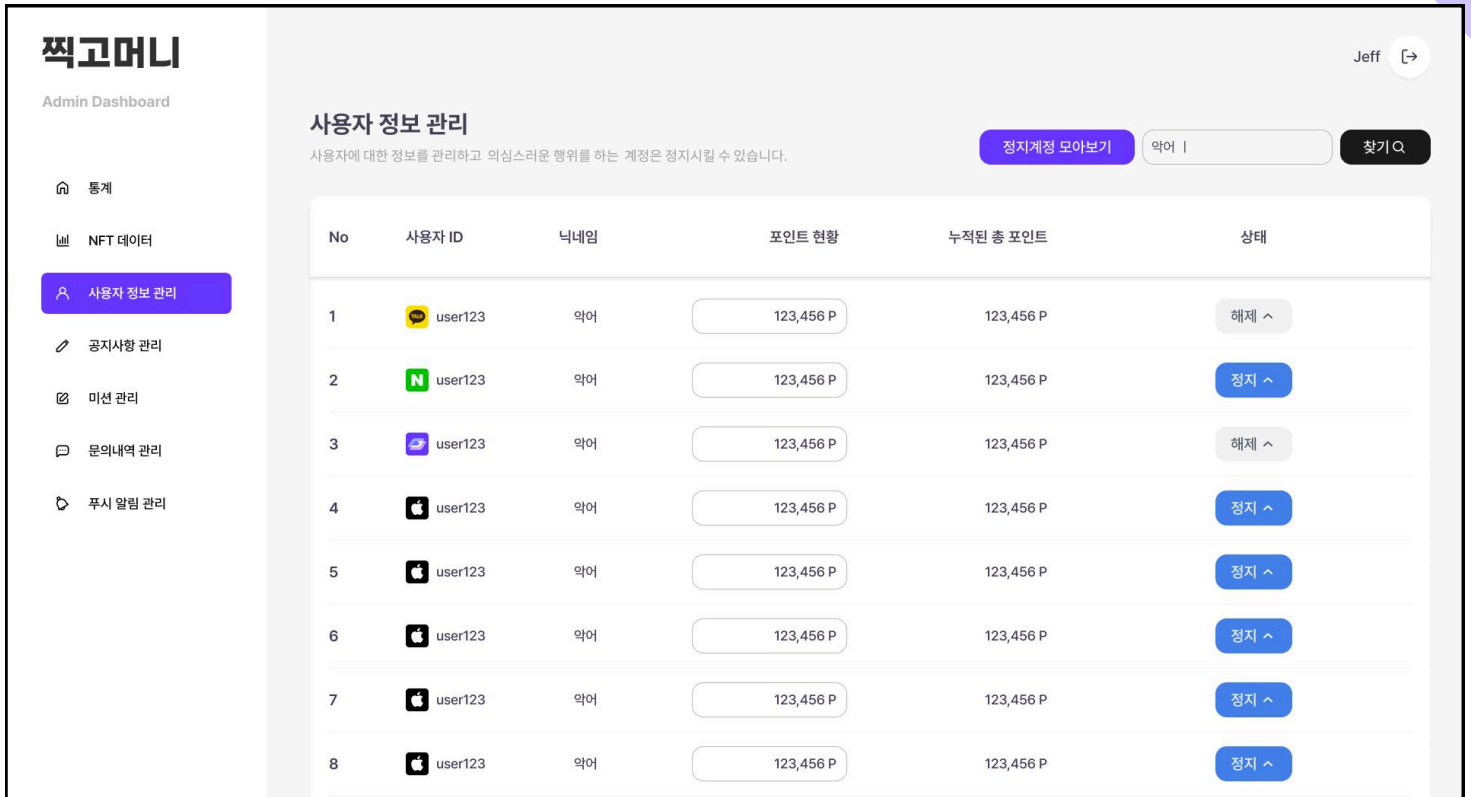
Dio를 활용한 Intorceptor 처리

```dart
class DioProvider {
  var logger = Logger(
    printer: PrettyPrinter(
      colors: true,
      printEmojis: true,
    ),
  );

  Dio baseDio({bool login = false}) {
    if (dotenv.env["BASE_API_URL"] == null) {
      throw Exception("BASE_API_URL is not set in .env file");
    } else {
      Dio dio = Dio();
      dio.options.baseUrl = dotenv.env["BASE_API_URL"]!;
      dio.interceptors.add(newInterceptorsWrapper(login: login));
      return dio;
    }
  }

  // 인터셉터로 토큰 및 에러 처리 최적화
  InterceptorsWrapper newInterceptorsWrapper({bool login = false}) {
    return InterceptorsWrapper(
      onRequest: (options, handler) async {
        if (!login) {
          final token = await AuthService().getAccessToken();
          if (token != null) {
            options.headers['Authorization'] = "Bearer $token";
          }
        }
        return handler.next(options);
      },
      onError: (DioException e, handler) async {
        // 토큰 만료 시 자동 갱신
        if (e.response?.statusCode == 401) {
          var req = await ReissueApi().reissue();
          if (req != null) {
            try {
              AuthService().saveKeys(req.accessToken, req.refreshToken);
              e.requestOptions.headers['Authorization'] = "Bearer ${req.accessToken}";
              return handler.resolve(await Dio().fetch(e.requestOptions));
            } catch (err) {
              return handler.reject(e);
            }
          }
        }
        return handler.next(e);
      },
    );
  }
}
```

찍고머니

어드민 페이지 개발

- 1. Next.js 기반 어드민 페이지 구축
  - 앱 소개, 이용 약관 등 핵심 정적 콘텐츠 페이지를 SSG(Static Site Generation) 방식으로 프리렌더링하여 초기 로딩 속도와 SEO 최적화 효과를 동시에 달성하였습니다.
- 2. 운영 효율성 개선
  - 어드민 페이지를 통해 앱 내 콘텐츠 관리 및 통계 확인이 가능하도록 구성하였으며, 콘텐츠 관리의 직관성을 고려한 UI/UX 설계에도 기여하였습니다.
  - SSG와 SSR을 필요에 따라 분리하여 적용하며 페이지 특성에 맞는 렌더링 전략을 설계하였습니다.



찍고머니

결과 및 기여

- 1. 성과 요약
  - 앱 출시 후 정식 서비스 론칭 성공
  - 앱 최적화 및 구조 개선을 통해 성능 및 사용자 이탈률 개선
  - 앱 구조 내 상태 관리, 아키텍처 설계, 네트워크 구조 등 프론트엔드 전반을 주도적으로 설계 및 구현
- 2. 기여도 및 성장 포인트
  - 프론트엔드 개발뿐 아니라 **API** 설계, 상태 관리, 릴리즈 후 디버깅까지 엔드 투 엔드(**End-to-End**) 개발 경험을 수행
  - 디자이너, 백엔드, 기획자와의 원활한 커뮤니케이션을 통해 협업 능력 강화
  - 사용자 데이터 기반 기능 개선 과정을 통해 서비스의 품질과 사용자 만족도 모두를 고려한 개발 의식 강화