파이널 프로젝트 #2

안드로이드 코틀린 프로젝트

김갑석 2025-07-28

Overview

1. 팀 내 역할 분담
2. 기획 및 설계
3. 프로젝트 설계
4. 개발 환경 구축
5. 기능 개발
6. 프로젝트 리뷰 및 리팩토링
7. 배포 준비 및 최종 마무리
8. 최종 발표

1. 팀 내 역할 분담

- 팀원 모두 기획 → 개발 → 발표 전 과정에 함께 참여
- 역할은 고정 없이 유연한 담당 분담 방식으로 운영
- 스크럼 마스터 : 일정 조율, GitHub 이슈 관리
- 예시 역할 담당:

  - 기획 및 화면 설계 담당 : 전체 팀원이 함께 기능 정의 및 화면 플로우 설계 진행

  - UI 개발 담당 : 주로 UI 위젯과 레이아웃 구현을 이끄는 역할

  - 로직 개발 담당 : 상태 관리, API 연동, 데이터 처리 등 핵심 로직 담당

  - 코드 구조 및 품질 관리 담당 : 폴더 구조 설계, 공통 컴포넌트 정리, 리뷰 주도

  - 문서 및 발표 준비 : 회의록, README, 발표 자료 작성 및 발표 리허설 주도

2. 기획 및 설계

- 주제 : 자율 주제
- 문서
  - 요구 사항 정의서
  - 기능 명세서
  - 화면 명세서
    - Figma : 시각적인 UI/UX 설계(디자인, 컴포넌트 배치)
    - 문서 : (Notion) : 동작/상세 기능 설명

2.1 요구 사항 정의서

- 목적 : 프로젝트에서 꼭 구현해야 할 기능과 범위를 명확하게 정의
- 예시)
  - 프로젝트 개요(앱 목적, 대상 사용자)
  - 주요 기능 목록
  - 비기능 요구 사항 (성능, 보안, 플랫폼)
  - 개발 범위 & 제외 범위
  - 일정/리스크

2.2 기능 명세서

- 목적 : 요구사항을 구체적인 기능 단위로 세분화하고 동작 규칙 정의
- 예시)
  - 각 기능별 상세 설명
    - 입력/출력 값
    - 동작 조건
    - 예외 처리
  - 화면 흐름과 기능 매핑
  - API 연동 내용(필요시)

2.3 화면 명세서

- 목적 : 실제 화면 설계를 통해 UI/UX를 정의 하고, 기능과 화면을 연결
- 도구 : Figma + Notion
  - Figma : 앱 전체 플로우 다이어그램
    - 화면별 UI 설계(컴포넌트 위치, 상태, 동작)
    - 화면과 기능 명세서 매핑
  - 화면 명세서(Notion) : 각화면별로
    - 화면 캡처
    - 기능 설명
    - 동작 규칙, 예외 처리
    - 해당 Figma 링크 삽입

3. 프로젝트 설계

- 기술 스택 및 아키텍처
  - MVVM + Feature-based
  - 레이어 구조

    UI(Jetpack Compose)

    ViewModel(상태관리, UI로직)

    Repository(데이터 관리) : 선택

    DataSource(Api/ Local DB)
- Di : Hilt (필요시)
- 네트워크 : Retrofit2 + OkHttp
- 로컬 DB : Room
- 비동기 : Kotlin Coroutines + Flow

4. 개발 환경 구축

- 개발 환경 설치 및 테스트

  - Android Studio 설치 및 에뮬레이터/ 실단말 테스트 환경준비

- GitHub 저장소 생성 및 연동
- Android 프로젝트 초기화 & 아키텍처 반영

  - MVVM + Feature-based

- .gitignore, build.gradle 등 기본 설정 파일 정비 및 의존성 추가
- 코드 스타일/ 기본 설정

5. 기능 개발

- UI 화면 구현

  - 화면 설계서(Figma) 기반으로 UI 개발 - 위젯 및 공통 컴포넌트

- 상태 관리 및 비즈니스 로직 분리
  - ViewModel + Stateflow 를 활영하여 상태(state)관리
  - UIState / UIEvent 패턴 적용
  - 비즈니스 로직 / 데이터 접근은 ViewModel에서 Repository 로 위임
- API 연동 및 비동기 처리
  - Retrofit2 + OkHttp 기반 API 연동
  - Coroutine + Flow 로 비동기 데이터 처리
  - 에러 / 예외 처리 공통화
    - 네트워크 예외, 인증 실패, 데이터 파싱 에러등 sealed class Result 로 관리
- 테스트 및 디버깅

  - 기능 단위 개발 후 시나리오 기반 테스트

  - 에러 및 예외 상황 점검 및 문제 해결 반복

6. 프로젝트 리뷰 및 리팩토링

- 코드 리뷰 및 공통 스타일 정리
  - 팀원 간 코드 공유를 통해 코드 흐름 및 작성 방식 통일
  - 불필요한 코드 제거 및 네이밍, 들여쓰기, 주석 등 코드 컨벤션 정리
- 중복 코드 제거 및 구조 개선
  - 중복된 로직을 함수/위젯으로 분리하여 재사용성 강화
  - 도메인별 폴더 구조 재정비 및 비즈니스 로직 정리
- 앱 성능 및 사용자 경험 개선
  - 불필요한 상태 변경 제거 및 렌더링 최적화
  - 로딩 인디케이터 처리 및 네트워크 지연 대응 추가
  - UI/UX 요소 정비 (버튼 반응성, 입력 폼 피드백 등)

7. 배포 준비 및 최종 마무리

- 앱 빌드
  - Android용 APK 파일 생성
  - 실제 디바이스 및 브라우저에서 최종 테스트 진행
- 결과물 정리 (스크린샷, 소개 영상 등)
  - 주요 화면 스크린샷 및 소개 영상 제작
  - 앱 사용 흐름 및 기능 시연 자료 구성
- 발표 자료 작성 및 리허설
  - 프로젝트 개요, 개발 흐름, 핵심 기능 위주로 발표 자료 구성
  - 팀원 간 역할 분담 및 발표 리허설 진행
- GitHub 저장소 정리
  - 최종 코드 정리, **README.md** 작성, 프로젝트 정보 명확히 표기

8. 최종 발표

- 팀별 발표 진행 (15~20분 내외)
  - 각 팀별로 프로젝트 진행 결과 발표
  - 앱 시연 중심으로 주요 기능 흐름을 설명
  - 핵심 구현 포인트 및 기술적 도전 과제 공유
- 프로젝트 회고
  - 프로젝트 진행 중 좋았던 점, 아쉬웠던 점, 배운 점 정리
  - 팀 협업에서 느낀 점 및 다음 프로젝트를 위한 개선 방향 도출
  - 기술적 성장 또는 협업 방식에 대한 팀원 개인 회고 포함