Software Design Specification(SDS)

201811223 조찬형

202213524 오택호

목차

4. 아키텍처 설계
   4.1. 시스템 아키텍처 다이어그램
   4.2. 클라이언트 구조
   4.3. 서버 구조
   4.4. 데이터베이스 구조

5. 상세 설계
   5.1. 모듈별 설계
   5.2. 클래스 설계
   5.3. 데이터 흐름
   5.4. API 인터페이스 설계

1. 소개 (Introduction)

1.1. 목적 (Purpose)

이 문서의 목적은 리그 오브 레전드 유저를 위한 AI 기반 승패 예측 및 피드백 시스템의 요구사항을 정의하는 것이다. 해당 프로그램은 라이엇 API 와 외부 게임 데이터 사이트(OP.GG 등)를 활용하여 유저의 전적을 분석하고 라인전 및 후반 성과 바탕의 승패 예측, 피드백 제공, 밴픽 및 조합에 따른 아이템 추천, 룬 추천을 제공한다. 이 시스템은 신규 및 초보 유저의 게임 이해도 증진, 중급 유저의 실력 향상을 목표로 한다.

1.2. 범위 (Scope)

이 시스템은 전적 데이터와 현재 플레이중인 게임 참가자들 데이터를 기반으로 우리 팀과 상대팀을 분석하고 플레이 보조 프로그램이다.

우선사용자가 직접 현재 티어(Tier)와 플레이하고 싶은 라인(Lane)을 입력해야 하며, 시스템은 티어나 라인을 자동으로 인식하지 않습니다.사용자의 숙련도(Riot API) + 서버 측 메타데이터(티어×라인의 승률/픽률/밴률/상대전적)를 기반으로,랭크 픽 단계에서 챔피언 추천(Pick)과 밴(Ban) 제안을 제공합니다.

게임이 시작되면 프로그램에서 API를 이용해 유저 데이터를 가져오고 승패 예측 및 분석을 시작한다. 입력된 라인에 맞춰 우선적으로 분석을 시작하고, 내 라인에 영향을 줄 수 있는 정글을, 마지막으로 탑, 미드, 바텀 순으로 분석을 해 피드백과 승패 예측 결과를 제공한다.

10명의 챔피언에 대한 정보가 들어오면 해당 챔피언들의 역할 군을 확인 후 자신의 라인과 챔피언에 맞춰 아이템 방향성을 제시한다.

해당 소프트웨어는 모바일에서 동작하며 실력 향상을 원하는 일반 유저들이 손쉽게 분석과 피드백을 받을수 있도록 설계되었다.

라이엇에서 제공하는 익명모드를 사용하는 유저의 경우 API를 통해서도 정보를 가져올 수 없기에 분석과 피드백 제공에 어려움이 있다.

1.3. 용어 정의 및 약어 (Definitions, Acronyms, and Abbreviations)

롤	리그 오브 레전드의 약어
DB	데이터베이스
API	라이엇 API
밴픽	챔피언 밴과 챔피언 픽의 약어
오브젝트	리그 오브 레전드의 에픽 몬스터(바론, 드래곤,...)
챔피언	리그 오브 레전드 게임 내에서 내가 플레이 중인 캐릭터
티어	리그 오브 레전드 챔피언들의 성능을 등수로 나타낸것
숙련도	리그 오브 레전드에서 제공하는 점수 시스템으로 해당 챔피언을 얼마나 플레이 했는지 알려줌
피드백	게임 라인업 분석에서 얻은 구조화 된 결과.
라인/포지션	팀에서 영웅의 고정 위치 또는 역할, 예 : 탑 (상단), 정글 (정글), 미드 (중간), 원딜 (ADC), 서폿 (지원)와 같은 역할을 나타냅니다.
익명모드	라이엇에서 제공하는 모드로 실시간 게임에서 자신의 닉네임을 가려 정보를 알 수 없게함

1.4. 참조 문서 (References)

https://developer.riotgames.com/apis

https://www.op.gg

1.5. 문서 개요 (Overview)

본 문서는 승률 예측 및 피드백 시스템의 설계 방안을 정의한다. SRS에 정의된 요구사항을 충족하기 위해 설계, 데이터 흐름, 인터페이스 설계 등을 기술한다.

2. 시스템 개요(System Overview)

2.1. 제품 관점 (Product Perspective)

본 시스템은 리그 오브 레전드(LoL) 전적 데이터와 라이엇 API를 활용하여 사용자의 경기 데이터를 분석하고, 승률 예측 및 피드백을 제공하는 모바일 애플리케이션이다.
클라이언트는 Android 환경에서 동작하며, 서버는 FastAPI 기반으로 구축되어 클라이언트의 요청을 처리하고 머신러닝 모델을 통해 결과를 반환한다.

2.1.1. 설계 방법 (Design Method)

- 아키텍쳐 : 클라이언트 - 서버 구조
- 클라이언트 : MVVM + Repository 패턴
- 서버 : Python(FastAPI)기반 REST API 설계, ML 모델 연동

2.1.2. 사용자 인터페이스 (User Interfaces)

- Android 모바일 앱(UI: Jetpack Compose 기반)
- 주요 화면 : API키 입력, 로그인, 챔피언 픽밴, 피드백, 아이템 방향성

2.1.3. 하드웨어 인터페이스 (Hardware Interfaces)

- 모바일 디바이스(Android 14 이상)
- 서버 환경 : AWS EC2 기반 CPU/GPU 리소스

2.1.4. 소프트웨어 인터페이스 (Software Interfaces)

- 클라이언트 ↔ 서버 통신 : REST API(JSON 기반)
- 외부 API : Riot Games API
- 데이터베이스 : MySQL 기반 RDBMS

2.2. 제품 기능 (Product Functions)

본 시스템은 리그 오브 레전드 플레이어의 경기 데이터를 분석하여, 챔피언 선택 단계부터 경기 후 피드백 제공까지 전 과정을 지원한다.(아이언 티어~다이아몬드 티어)
주요 기능은 다음과 같다

- 라이엇 아이디를 통한 경기 조회
- 숙련도 데이터 수집
- 숙련도, 티어, 라인 기반 밴픽 추천
- 전적 분석을 통한 승률 예측
- 전적 분석을 통한 피드백 제공
- 챔피언 역할을 바탕으로 아이템 추천

2.3. 사용자 특성 (User Characteristics)

본 시스템은 리그 오브 레전드를 플레이하는 다이아몬드 티어 이하의 일반 유저를 대상으로 하며, 게임 실력 향상 및 전략적 플레이 지원을 목적으로 한다.

사용 가능 대상은 아이언 티어부터 다이아몬드 티어의 유저들이며, 특히 골드 티어 이하의 유저들이 주요 사용자층으로 설정된다.

- 대상 : LoL 플레이어(아이언 티어~다이아몬드 티어)
- 플랫폼: Android 스마트폰 사용자
- 요구 기술 수준 : 기본적인 모바일 앱 사용 능력
- 사용 목적 : 게임 이해도 증진, 실력 향상, 효율적인 밴픽

2.4. 제약 사항 (Constraints)

본 시스템은 라이엇에서 제공하는 API를 사용하기 때문에 API 호출 횟수 제한이나 응답 지연이 발생할 수 있습니다. 또한 라이엇의 데이터 사용 정책을 준수해야 합니다.

2.5. 가정 및 의존성 (Assumptions and Dependencies)

본 시스템은 사용자가 라이엇 계정을 가지고 있다는 것을 가정하고 설계되었습니다. 시스템은 라이엇 API에 의존하며 응답 속도는 상황에 따라 달라질 수 있습니다. 본 소프트웨어는 지속적인 인터넷 연결이 가능한 상태에서의 실행을 전제로 합니다.

3. 설계 고려사항 (Design Considerations)

3.1. 운영 환경 (Operating Environment)

3.1.1. 클라이언트 환경

본 시스템은 Android 14 이상을 지원하는 모바일 기기에서 동작하도록 설계되었다. 사용자는 일반적인 스마트폰 환경을 갖춘 기기를 사용하면 되며 안정적인 인터넷 연결이 필요하다. 애플리케이션은 MVVM + Repository 패턴을 사용한다.

주요 라이브러리는 다음과 같다

- Coroutines/Flow: 비동기 처리
- Lifecycle/ViewModel KTX: 상태 관리
- Navigation-Compose: 화면 전환
- Retrofit2 + OkHttp3 (또는 Ktor Client): 서버 통신
- Moshi/Gson: JSON 처리

3.1.2. 서버 환경

본 시스템의 서버는 FastAPI 기반으로 구축되며, 클라이언트로부터 전송받은 전적 데이터를 처리하고 머신러닝 모델을 통해 승률 예측과 피드백을 생성한다. 서버는 AWS EC2 인스턴스 환경에서 운영된다.

주요 소프트웨어 구성 요소는 다음과 같다:

- Python 3.10 이상: 서버 로직 및 ML 모델 연동
- FastAPI: REST API 서버 구축
- Uvicorn: ASGI 서버 실행
- Pandas / Numpy: 데이터 가공 및 분석
- Scikit-learn / TensorFlow / PyTorch: 머신러닝 모델 학습 및 추론

3.1.3. 외부 시스템

본 시스템은 승률 예측 및 피드백 제공을 위해 다음 외부 시스템과 상호작용한다.

- 라이엇 API (Riot Games API)
  - 목적: 사용자 계정 정보, 챔피언 숙련도, 최근 경기 전적 등 데이터를 수집
  - 접속 방법: HTTPS 기반 RESTful API
  - 데이터 포맷: JSON
  - 제약 사항: API 요청 제한(Rate Limit) 존재, 유효한 API 키 필요
  - 사용 범위: 클라이언트에서 직접 호출하여 사용자 전적을 수집, 이후 서버에 전송
- OP.GG 등 외부 게임 데이터 사이트
  - 목적: 메타 데이터 수집(티어별 승률, 픽률, 밴률 등)
  - 접속 방법: 웹 API 또는 크롤링
  - 데이터 포맷: HTML / JSON 등 제공 형식에 따라 다름
  - 제약 사항: 사이트 접근 정책 준수 필요, 과도한 요청 시 차단 가능
  - 사용 범위: 서버에서 참조용 데이터 수집, 모델 학습 및 승률 예측에 활용

- 클라이언트 ↔ 서버 통신

- 목적: 클라이언트에서 수집한 전적 데이터를 서버로 전달하고, 서버에서 생성한 피드백 및 예측 결과를 클라이언트로 전송
- 접속 방법: HTTP 기반 REST API
- 데이터 포맷: JSON
- 제약 사항: 안정적인 네트워크 연결 필요

3.1.4. 제약 조건

- 플랫폼 제약
  본 애플리케이션은 Android 13 이상에서 동작하도록 설계되어 있으며, iOS 등 다른 플랫폼에서는 지원되지 않는다.

- 기기 제약
  일반 스마트폰 환경(CPU, RAM, 저장 용량 등)을 기준으로 설계되며, 저사양 기기에서는 성능 저하가 발생할 수 있다.

- 네트워크 제약
  전적 분석 및 피드백 제공을 위해 인터넷 연결이 필요하다.
  라이엇 API 호출 시 Rate Limit가 존재하므로, 호출 횟수 제한을 고려해야 한다.

3.2. 아키텍처 설계 (Architectural Design)

3.2.1. 구조

본 시스템은 클라이언트-서버 구조로 설계되었으며, 클라이언트와 서버는 REST API를 통해 JSON 데이터를 주고받는다.
클라이언트는 MVVM + Repository 패턴을 사용하여 UI와 데이터 처리를 분리한다.
서버는 Python(FastAPI) 기반으로 구축되며, 머신러닝 모델을 이용한 승률 예측 및 피드백 생성, 그리고 아이템 추천을 담당한다.

3.2.2. 클라이언트

- 유저로부터 API 키와 idTag를 입력받음
- Riot API와 직접 통신하여 숙련도 및 전적 데이터를 수집
- 수집한 데이터를 가공하여 서버에 전송
- 서버로부터 전달받은 피드백 데이터를 UI에 표시
- 챔피언 역할에 기반하여 아이템 추천

3.2.3. 서버

- 서버 API 키를 사용하여 티어별 랜덤 플레이어 전적 수집 및 모델 생성

- 클라이언트로부터 전달받은 데이터를 사용하여 승률 예측 모델 실행
- 승률 예측 및 피드백 생성, 추천 아이템 생성
- 결과를 클라이언트에 반환

3.3. 사용자 인터페이스 (User Interface)

본 시스템은 Android 스마트폰 환경에서 동작하며, Jetpack Compose 기반으로 UI를 구현한다. 사용자는 직관적인 화면 전환과 시각적 피드백을 통해 전적 분석, 피드백 확인, 밴픽 추천, 아이템 방향성 제시 기능을 이용할 수 있다.

- 주요 화면 구성
  - API 키 입력 화면: 사용자가 라이엇 API 키와 아이디를 입력하는 화면으로, 필수 입력값 유효성 검사 및 오류 메시지 제공.
  - 챔피언 픽/밴 화면: 사용자의 라인과 티어에 따라 추천 챔피언과 밴을 제공.
  - 피드백 화면: 내 라인과 타 라인 피드백 정보를 확인할 수 있으며, 선택한 라인에 대한 분석 내용이 실시간으로 업데이트됨.
  - 아이템 추천 화면: 챔피언 역할을 기반으로 추천 아이템 제시.
- 입력 방식
  - 화면 터치, 버튼 선택, 텍스트 입력을 통한 사용자 명령 수집.
  - API 키 입력, idTag 등 필수 입력값에 대해 유효성 검사 수행.
- 출력 방식
  - JSON 데이터를 기반으로 화면 요소를 동적으로 업데이트.
  - 피드백 및 추천 결과는 텍스트와 그래픽 요소로 사용자에게 표시.
- 사용자 상호작용
  - 화면 간 네비게이션: Jetpack Navigation 사용, 자연스러운 화면 전환 지원.
  - 상태 표시: 데이터 로딩, 분석 진행 상황, 오류 발생 시 안내 메시지 제공.
  - 피드백 선택: 사용자가 타 라인의 피드백을 선택하면 로컬에서 데이터를 가져와 즉시 화면 업데이트.

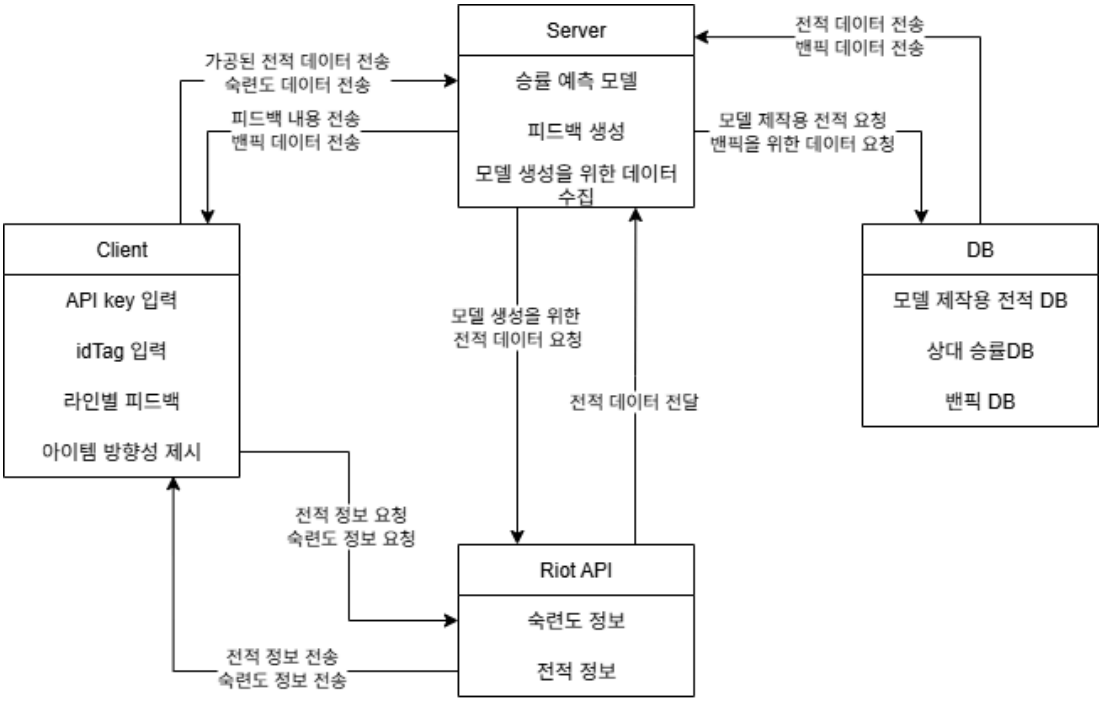
3.4. 성능 고려사항 (Performance Considerations)

본 시스템은 리그 오브 레전드 전적 데이터를 분석하고 피드백을 제공하는 것을 주요 기능으로 하며, 사용자에게 지연 없이 정보를 전달하기 위해 성능 측면에서 다음과 같은 고려사항을 적용한다.

- 응답 시간
  - 사용자가 라인을 선택하고 티어 정보를 입력하면 추천 결과를 5초 이내에 제공하도록 설계한다.
  - 내 라인과 정글 포함 분석은 3분 이내, 전체 라인 분석 및 피드백 제공은 10분 이내 완료를 목표로 한다.
  - 아이템 추천은 피드백 화면에 들어서면 바로 제공 받고, 추가사항은 라인 분석이 완료될때마다 업데이트 된다.
- 데이터 처리량
  - 서버는 다수의 클라이언트 요청을 동시에 처리할 수 있도록 설계하며, 각 요청에 대한 피드백 생성 및 승률 예측 모델 연산이 병목 없이 수행될 수 있어야 한다.
- 사용자 경험
  - 분석 진행 중 로딩 상태를 표시하여 사용자가 기다리는 동안 진행 상황을 확인할 수 있도록 한다.
  - 서버로부터 피드백 데이터를 수신하는 시간 동안 UI가 멈추지 않도록 비동기 처리(Coroutines/Flow)를 적용한다.

4. 아키텍처 설계 (Architectural Design)

4.1. 시스템 아키텍처 다이어그램



본 시스템은 클라이언트-서버 구조로 설계되어 있으며, 클라이언트 앱은 Android 환경에서 동작하고, 서버는 FastAPI 기반으로 운영된다.

- 클라이언트는 API 키와 유저 입력 정보를 바탕으로 Riot Games API와 통신하여 숙련도 데이터 및 전적 데이터를 수집
- 서버는 수신한 데이터를 기반으로 밴픽 추천 모듈, 승률 예측 모델과 피드백 모듈을 통해 결과 생성
- 클라이언트는 서버로부터 전달받은 피드백 및 추천 정보를 UI에 표시

4.2. 클라이언트 구조

본 시스템의 클라이언트는 Android 환경에서 동작하며, 사용자 인터페이스와 데이터 처리, 서버 통신 기능을 담당한다. 클라이언트 구조는 MVVM + Repository 패턴을 기반으로 설계되어 모듈 간 책임을 명확히 분리한다.

- 주요 구성
    - View (UI Layer) : Jetpack Compose 기반 화면 구성. 사용자가 입력한 API 키, Riot ID, 티어, 라인 등의 정보를 수집하고, 서버로부터 전달받은 피드백과 아이템 추천 데이터를 표시한다.

- ViewModel (Presentation Layer) : UI와 데이터 처리 사이의 중계 역할 수행. Coroutine/Flow를 사용하여 서버와 비동기 통신 및 UI 업데이트를 처리한다.

- Repository (Data Layer) : 서버 통신 및 로컬 데이터 관리 담당. Retrofit2 + OkHttp3를 통한 REST API 호출, JSON 데이터 파싱(Moshi/Gson)

- 주요 기능
  - API 키 및 idTag 입력 : 유저가 입력한 API 키를 사용해 idTag의 숙련도 데이터를 수집한다.
  - ~~밴픽 추천 : 유저에게 티어와 라인을 입력받고 해당 정보와 숙련도 데이터를 사용하여 밴픽을 추천해준다.~~
  - 피드백 제공 : idTag의 유저가 게임중인 경우 해당 게임에 참가한 유저들의 전적을 수집, 가공하여 서버로 전송하고, 서버로 부터 피드백 내용을 받아 유저에게 전달한다.
  - 아이템 추천 : 챔피언 역할을 바탕으로 아이템을 추천한다.

- 주요 라이브러리
  - UI/아키텍처 : Jetpack Compose, MVVM, Repository Pattern
  - 비동기 처리 : Kotlin Coroutines
  - 네트워크 처리 : Retrofit2 + OkHttp3

- 데이터 흐름
  1) 사용자 정보 입력
     a) 출발점 : 유저
     b) 처리 : API 키와 idTag 입력 → 로컬 저장
     c) 도착점 : 앱 내부 저장소
  2) 숙련도 데이터 수집
     a) 출발점**:** 클라이언트
     b) 처리**:** 입력된 API 키와 idTag를 사용해 Riot API 호출 → 숙련도 데이터 수집
     c) 도착점**:** 클라이언트 내부 저장
  3) 밴픽 요청
     a) 출발점 : 클라이언트
     b) 처리 : 라인과 티어 데이터를 입력받고 숙련도 데이터와 함께 서버로 전송
     c) 도착점 : 서버

4) 밴픽 결과 수신
    a) 출발점 : 서버
    b) 처리 : 전송받은 데이터로 밴픽 생성 후 클라이언트로 전송
    c) 도착점 : 클라이언트(밴픽 화면)
5) 게임 상태 확인
    a) 출발점 : 클라이언트
    b) 처리 : API를 통해 idTag의 유저가 게임중인지 확인
    c) 도착점 : 클라이언트(유저 라인 배치)
6) 전적 데이터 수집 및 전송
    a) 출발점 : 클라이언트
    b) 처리 : 유저 라인 배치에 맞게 API를 통해 전적 수집 및 가공 후 서버로 전송
    c) 도착점 : 서버
7) 승률 예측 및 피드백
    a) 출발점 : 서버
    b) 처리 : 가공된 데이터를 기반으로 승률 예측 및 피드백 생성 후 클라이언트로 전송
    c) 도착점 : 클라이언트(피드백 화면)
8) 아이템 추천 요청
    a) 출발점 : 클라이언트
    b) 처리 : 사용자 챔피언 역할과 적 챔피언 5명 역할 서버로 전송
    c) 도착점 : 서버
9) 아이템 추천 수신
    a) 출발점: 서버
    b) 처리: 전송받은 데이터로 아이템 추천 생성 후 클라이언트로 전송
    c) 도착점: 클라이언트(아이템 추천 화면)

4.3. 서버 구조 (Server Structure)

본 시스템의 서버는 **FastAPI** 기반으로 구축되며, 클라이언트로부터 전달받은 전적 데이터를 처리하고, 머신러닝 모델을 활용하여 승률 예측과 피드백을 생성하는 역할을 담당한다. 서버는 AWS EC2 인스턴스 환경에서 동작하며, 데이터베이스 및 외부 Riot API와 연동된다.

- 주요 구성
    - **API** 계층 **(FastAPI)**
        - 클라이언트 요청 수신 및 응답 반환
        - RESTful API 형식으로 설계
    - 서비스 계층 **(**비즈니스 로직**)**
        - ~~밴픽 추천~~, 승률 예측, 피드백 생성 로직 수행
        - 비동기 처리(Await/Async) 기반으로 병렬 처리 지원
        - 아이템 추천 로직 수행
    - 데이터 계층 **(Database &** 모델 저장소**)**

        - MySQL 기반 데이터베이스 운영(전적, 밴픽, 챔피언 데이터 저장)
        - Joblib로 직렬화된 머신러닝 모델 관리
    - 외부 연동
        - Riot Games API 호출을 통한 전적 데이터 수집
        - OP.GG 등 외부 메타데이터 활용

- 주요 기능
    - ~~밴픽 추천 모듈~~
        - ~~클라이언트가 전송한 티어, 라인, 숙련도 데이터를 기반으로 밴픽 전략을 산출한다~~.
    - 승률 예측 모듈
        - 가공된 전적 데이터를 머신러닝 모델에 입력하여 팀별 승률을 계산한다.
    - 피드백 생성 모듈
        - 승률 예측 결과를 기반으로 피드백을 생성해 클라이언트로 전달한다.
    - 아이템 추천 모듈
        - 사용자 챔피언 역할과 적 챔피언 역할을 기반으로 아이템 추천을 생성해 클라이언트로 전달한다.
    - 데이터 관리 모듈
        - Riot API 호출 결과, 전적 데이터, 모델 결과 등을 데이터베이스에 저장 및 조회한다.
- 사용 기술 스택
    - 백엔드 프레임워크 : FastAPI
    - 비동기 처리 : Python asyncio, HTTPX
    - 데이터베이스 : MySQL
    - 머신러닝 모델 관리 : scikit-learn, joblib
    - 배포 환경 : AWS EC2 (Ubuntu), Uvicorn

4.4. 데이터베이스 구조

본 시스템은 유저 전적 데이터, 밴픽 데이터, 챔피언 상대 승률 데이터를 효율적으로 관리하기 위해 **MySQL** 기반 **RDBMS**를 사용하며, 데이터 접근은 서버를 통해 이루어진다. 데이터베이스 구조는 다음과 같은 주요 테이블로 구성된다.

- 전적 DB
    - 승률 예측 모델 제작에 필요한 전적 데이터를 저장
    - 각 티어별 전적을 따로 관리
    - 주요 칼럼

        - match_id(PK) : 매치 고유 식별자
        - teamposition : 해당 매치에서의 역할
        - my_champion : 내가 플레이한 챔피언
        - enemy_champion : 상대가 플레이한 챔피언
        - win : 승패 여부
        - 그 외 경기 데이터
- 상대 승률 DB
    - 밴픽 추천 및 승률 예측 모델이 필요한 상대 승률 데이터를 저장
    - 주요 칼럼
        - hero_name : 내가 플레이한 챔피언
        - counter_name : 상대가 플레이한 챔피언
        - position : 해당 매치에서의 라인
        - tier : 해당 매치에서의 티어
        - winrate : 상대 승률
- 밴픽 DB
    - 챔피언별 밴률, 픽률 데이터를 저장
    - OP.GG에서 데이터를 가져옴
    - 주요 칼럼
        - name : 챔피언 이름
        - postition : 챔피언 라인
        - tier : 기준 티어
        - positionWinRate : 해당 라인 승률
        - positionBanRate : 해당 라인 밴률
        - positionPickRate : 해당 라인 픽률
- 아이템 DB
    - 아이템 추천 모델에 필요한 데이터 저장
    - 챔피언 역할과 상대팀 챔피언 역할, 해당 게임에서 구매한 아이템 저장
    - 주요 칼럼
        - my_role: 내 챔피언 역할
        - my_items: 해당 게임에서 구매한 아이템
        - enemy_roles: 상대팀 챔피언 역할
        - win: 승패 여부

5. 상세 설계

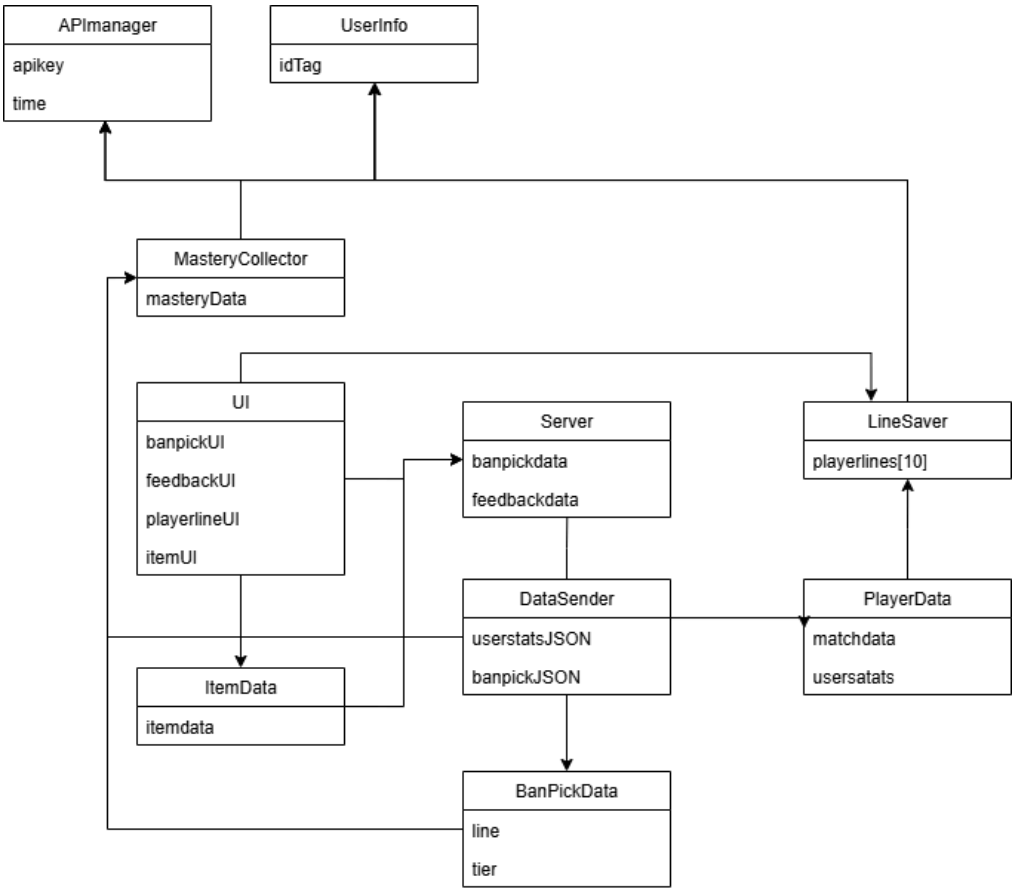
5.1. 모듈별 설계

- API 키 관리 모듈(클라이언트)
    - 목적 : 유저가 입력한 API 키 관리 모듈(Req-1.1, Req-1.2.1, Req-1.2.2)
    - 입력 : Riot API 키(혹은 기존 저장된 키), 유저의 idTag
    - 산출물 : API 키가 저장된 파일, API 키 유효성 확인 결과
- 유저 idTag 관리 모듈(클라이언트)

    - 목적 : 유저 idTag 확인 및 밴픽 화면 연결(Req-1.3)
    - 입력 : Riot API 키(혹은 기존 저장된 키), 유저의 idTag
    - 산출물 : idTag 유효성 확인 결과, 밴픽 화면
- 숙련도 데이터 수집 모듈(클라이언트)
    - 목적 : idTag 유저의 숙련도 파일 수집(Req-1.4)
    - 입력 : Riot API 키(혹은 기존 저장된 키), 유저의 idTag
    - 산출물 : 숙련도 데이터 파일
- ~~밴픽 데이터 전송 모듈(클라이언트)~~
    - ~~목적 : 유저의 숙련도, 라인, 티어 정보를 서버로 전송(Req-2.1)~~
    - ~~입력: 유저의 라인, 티어, 숙련도 데이터 파일~~
    - 산출물 : ~~서버로 전송할 숙련도, 라인, 티어를 포함한 데이터~~ 밴픽 데이터
- ~~밴픽 추천 모듈(서버)~~
    - ~~목적 : 전송된 데이터를 기반으로 밴픽 추천(Req-2.1)~~
    - ~~입력 : 숙련도, 라인, 티어가 입력된 데이터~~
    - ~~산출물 : 추천 밴픽 3종류(추천 챔피언 3개, 챔피언 하나당 추천 밴 3개)~~
- 밴픽 추천 UI 모듈(클라이언트)
    - 목적 : 사용자의 입력에 맞게 밴픽 추천
    - 입력 : 사용자 라인과 티어
    - 산출물 : 추천 밴픽 3종류
- 라인 저장 모듈(클라이언트)
    - 목적 : 플레이어 분석을 위한 라인 입력(Req-3.1)
    - 입력 : 10명의 플레이어의 라인
    - 산출물 : 저장된 10명의 플레이어 라인
- 플레이어 데이터 가공 모듈(클라이언트)
    - 목적 : 각 플레이어 전적 분석을 통해 서버로 전송할 데이터 생성(Req-3.1)
    - 입력 : 플레이어의 전적 데이터
    - 산출물 : 서버로 전송할 가공된 JSON 데이터(10명의 데이터가 순서대로 제공됨)
- 플레이어 데이터 전송 모듈(클라이언트)
    - 목적 : 전적 분석 데이터를 서버로 전송(Req-3.1, Req-4.1)
    - 입력 : JSON 데이터
    - 산출물 : 서버로 전송되는 JSON 데이터(생성 즉시 전송)
- 플레이어 피드백 생성 모듈(서버)
    - 목적 : 전적 분석 데이터를 기반으로 피드백 생성(Req-4.1, Req-4.2)
    - 입력 : 전적 분석 데이터
    - 산출물 : 클라이언트로 전송할 피드백 데이터
- 플레이어 피드백 UI 모듈(클라이언트)
    - 목적 : 피드백 데이터를 UI로 출력(Req-4.1, Req-4.2)
    - 입력 : 피드백 데이터
    - 산출물 : 피드백 UI(클라이언트 UI에 순차적으로 반영)

- 선택한 플레이어 피드백 UI 모듈(클라이언트)
    - 목적 : 선택한 플레이어 피드백 UI로 출력(Req-4.3)
    - 입력 : 선택한 플레이어 idTag 클릭
    - 산출물 : 타 플레이어 피드백 UI
- 아이템 추천 모듈(클라이언트)
    - 목적 :챔피언 역할 따른 아이템 추천(Req-4.4)
    - 입력 : 사용자 챔피언 역할과 상대팀 챔피언 역할 5개
    - 산출물 : 역할에 따른 아이템

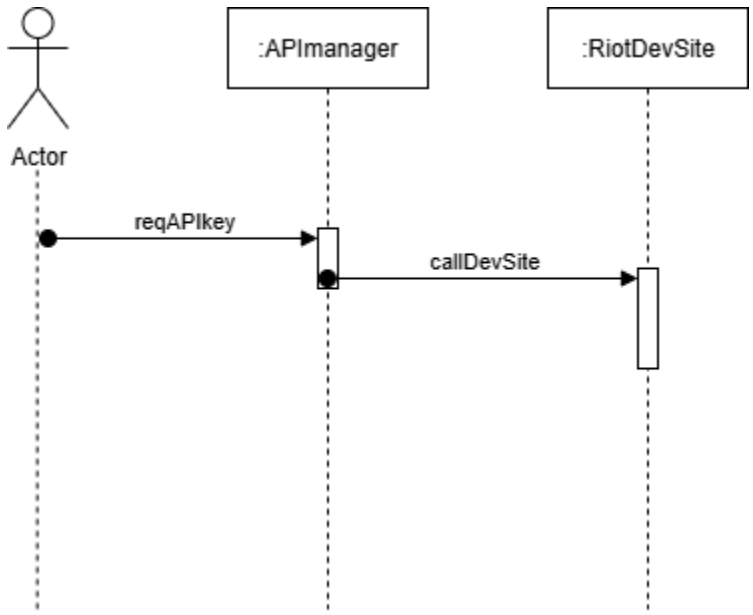
5.2. 클래스 설계



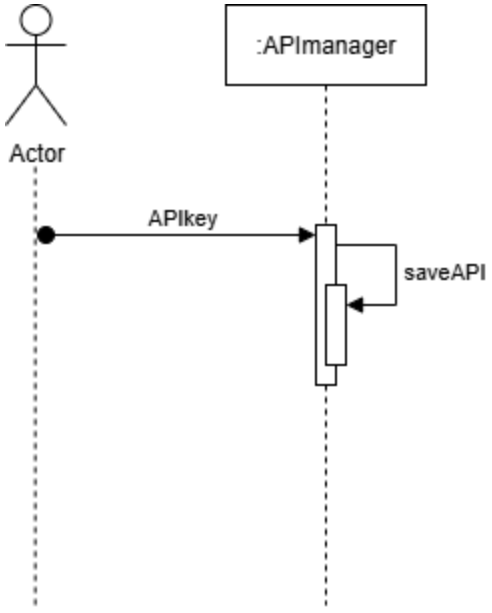
- APImanager
    - API 키를 관리
    - 만료시간 체크
- UserInfo
    - 유저idTag 관리
- MasteryCollector
    - 유저 숙련도 데이터 관리
- LineSaver
    - 매치에서 플레이어 라인 관리
- PlayerData
    - 서버로 전송할 전적 데이터 생성
- BanPickData

    - 서버로 전송한 밴픽 데이터 생성
- ItemData
    - 서버로부터 피드백 데이터 전달받아 아이템 방향성 정보 생성
- DataSender
    - 플레이어 전적 데이터 서버로 전송
    - 밴픽 데이터 서버로 전송
- Server
    - 플레이어 전적 데이터를 받아 피드백 생성
    - 밴픽 데이터를 받아 밴픽 추천 생성
- UI
    - 서버로부터 전송받은 피드백 데이터 출력
    - 서버로부터 전송받은 밴픽 데이터 출력
    - LineSaver로부터 라인 정보를 받아서 출력
    - ItemData로부터 아이템 방향성 정보를 받아서 출력

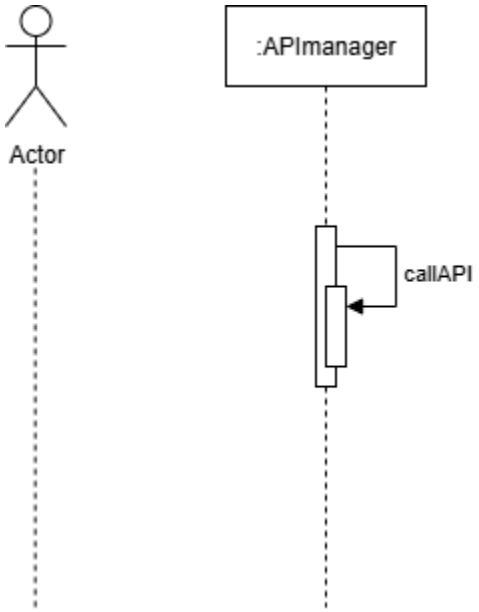
5.3. 데이터 흐름



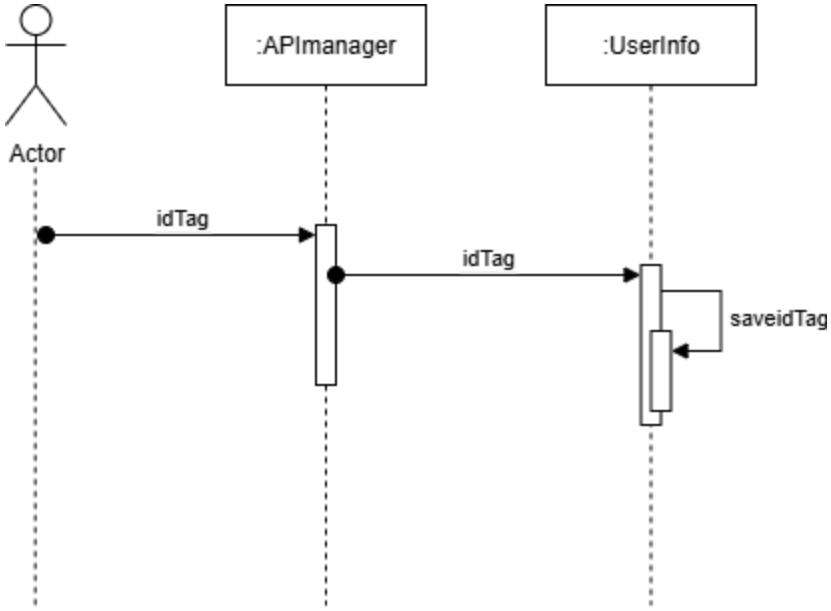
- Req-1.1 API 키 발급
    - 유저가 API 키 발급 버튼을 누르면 개발자 사이트로 이동



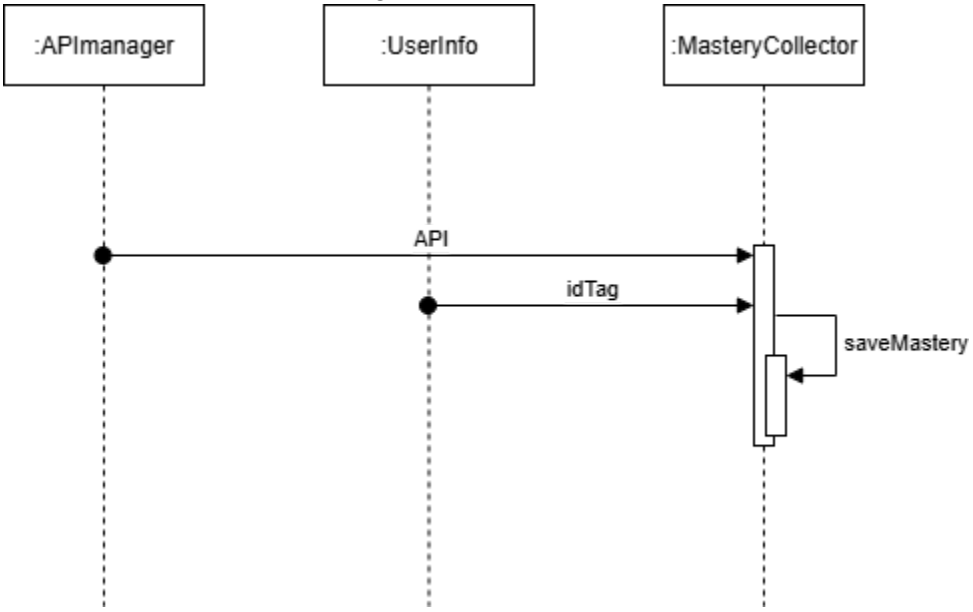
- Req-1.2.1 API 입력
  - 유저가 API 키를 입력하면 해당 키를 저장



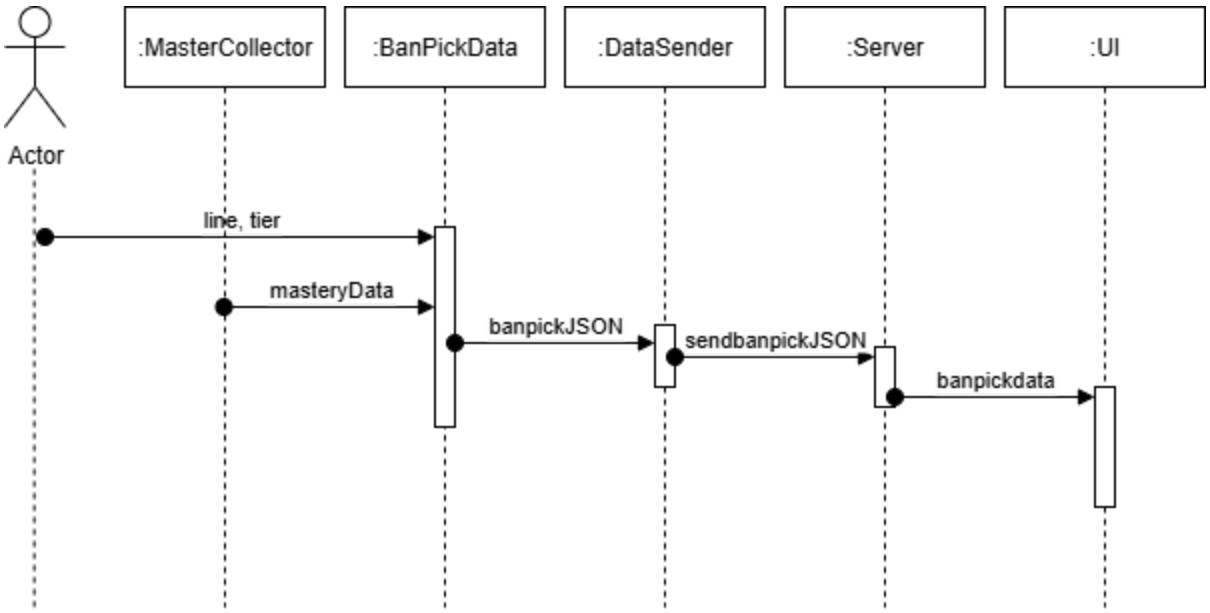
- Req-1.2.2 API 키 존재
  - 24시간 이내에 API 키를 발급받고 입력한 기록이 있으면 기존 키 불러옴



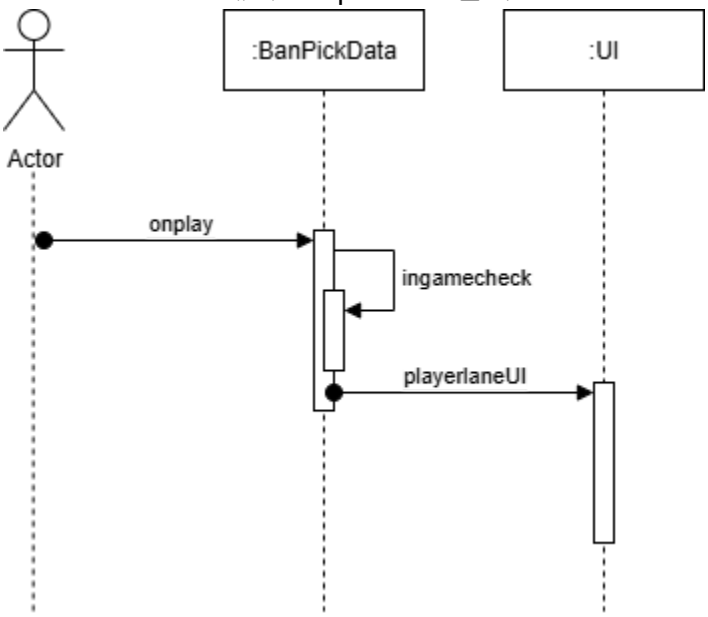
- Req-1.3 사용자 ID 입력
  - 사용자가 idTag를 입력하면 UserInfo에 저장



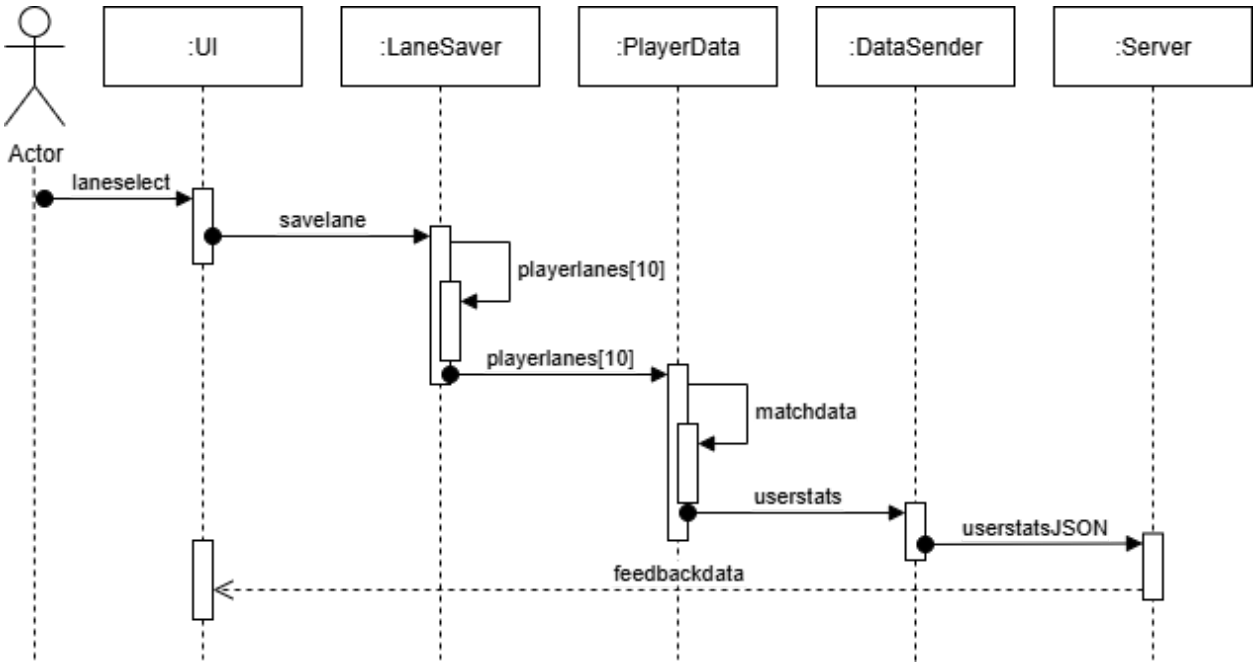
- Req-1.4 숙련도 데이터 수집
  - API 키와 idTag를 이용해 숙련도 데이터 저장



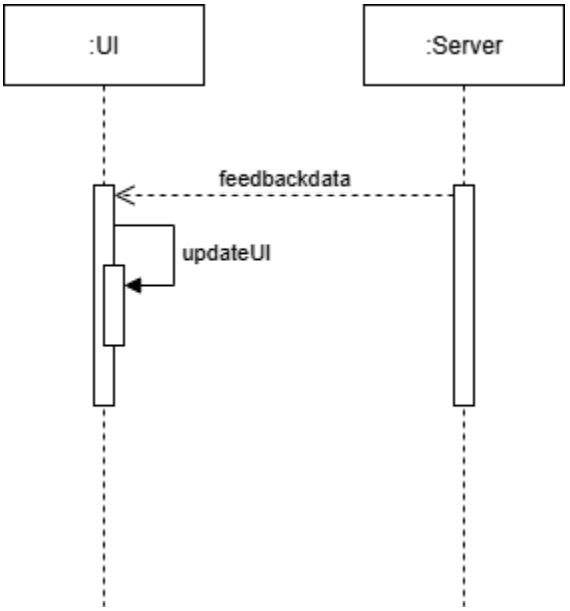
- Req-2.1 라인 티어 입력
  - 유저로부터 라인과 티어를 입력받음
  - BanPickData에서 masteryData 와 라인, 티어 정보 결합
  - DataSender에서 Server로 전송
  - Server에서 UI로 banpickdata 전송
  - UI에서 banpickdata 출력



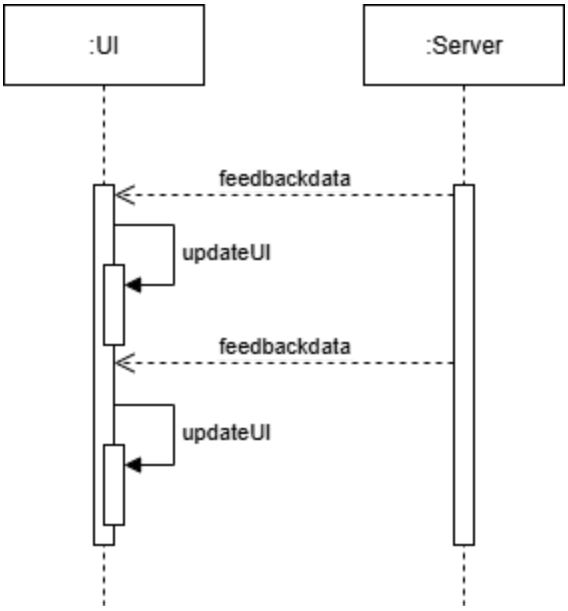
- Req-2.2 게임중 확인
  - 유저가 play버튼을 누름
  - BanPickData에서 api를 통해 게임중인지 확인
  - 게임중인 경우 playerlaneUI 출력



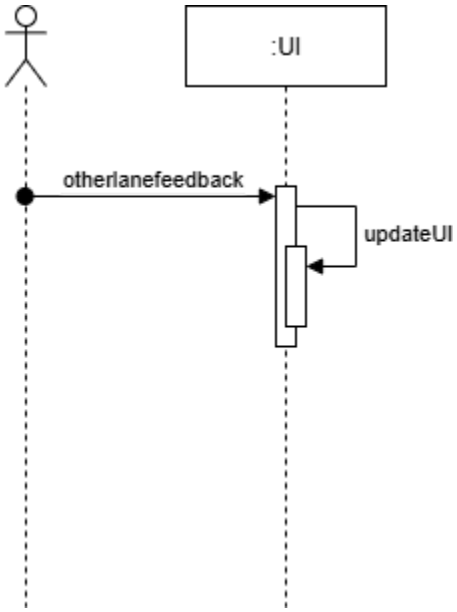
- Req-3.1 플레이어 라인 선택
  - 10명의 플레이어 라인 선택
  - LaneSaver에서 라인 저장
  - LaneSaver에서 라인 정보를 PlayerData로 전송
  - PlayerData에서 라인 정보를 바탕으로 매치를 수집하고 userstats 생성
  - userstats DataSender로 이동
  - DataSender에서 userstatsJSON으로 변환후 서버로 전송



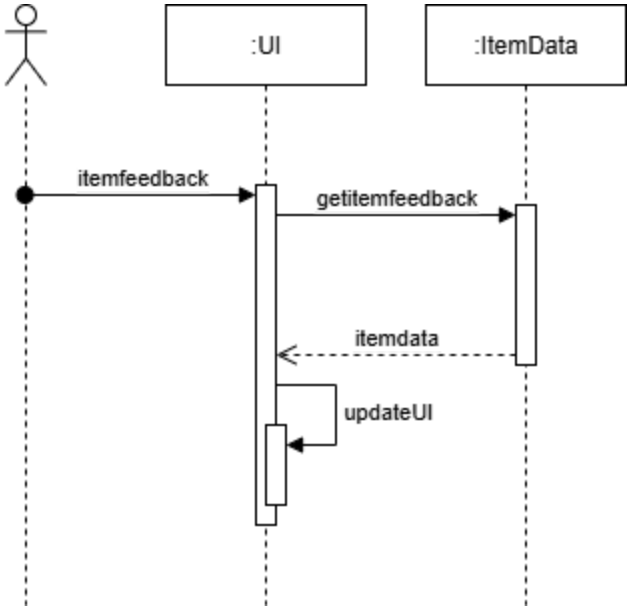
- Req-4.1 내 라인 피드백 제공
  - 서버로부터 피드백 데이터를 받아 UI에 출력



- Req-4.2
    - 서버로부터 지속해서 데이터를 받아 UI 업데이트



- Req-4.3 타 라인 피드백 제공
    - 유저가 화면에서 다른 라인을 클릭함
    - 해당 플레이어 피드백 화면으로 업데이트



- Req-4.4 아이템 방향성 제시
  - 유저가 아이템 버튼 클릭함
  - ItemData에 itemdata 요청
  - ItemData에서 itemdata 전달
  - itemUI로 업데이트

5.4. API 인터페이스 설계

- 피드백 전송 API
  - path : /analysis
  - method : POST
  - 목적 : 전적 분석 데이터를 기반으로 승률 예측 및 피드백 생성
- 밴픽 추천 API
  - path : /banpick
  - method : POST
  - 목적 : 유저의 라인, 티어, 숙련도 데이터를 기반으로 추천 밴픽 생성