AI_Librarian

Tech Report

2403110284 정윤우

프로젝트 개요

2403110284 정윤우

프로젝트명 : 사진 기반 책장 도서 정리 앱(AI Librarian)

- 개발 기간 : 2025.03.01~2025.06.26
- 팀 구성 : 정윤우, 안형욱, 이국정, 이예은
- 기술스택 :
  - 어플리케이션 : Android Studio(Java)
  - 서버 : Flask Server
  - Database : OracleDB
  - OCR : PaddleOCR, EasyOCR, Korean-Light-OCR
  - 객체인식 : YOLOv8
  - API : Naver Book Search API
- 프로젝트 목표 : 사진 한 장으로 사용자의 보유도서의 제목을 추출하고, 기존에 사용자가 보유하고 있는 도서를 등록 책장에서 검색 및 보유도서 목록을 csv파일로 내려받는 어플리케이션 서비스 구현.

역할 소개

- 정윤우(팀장) : 전체 구조 조율 및 통합
- 안형욱 : 공간배치 모듈
- 이국정 : 객체탐지 모듈, OCR, 검색모듈(부분)
- 책 검색 모듈, OCR

담당 모듈

프로젝트에서 저는 기능 제어 모듈 개발 및 Android 어플리케이션 통합을 담당했습니다.

- 기능 제어 모듈 : 모듈로 역할을 구분했지만 결국 개발 단계에 들어갔을때 각 모듈별 기능을 연결 후 동작하는 Flask 기반 Server 및 연동 Oracle DB를 구현하였습니다.
- Android App : Flask 기반의 백엔드 서버와 연동하여 사용자가 책장 사진을 업로드하고, 책 정보를 확인/수정/검색 하는 기능을 사용하는 UI/UX를 구현했습니다.
- 팀장으로서 프로젝트의 전체적인 진행상황을 관리하고 최종 시스템 통합을 담당했습니다.

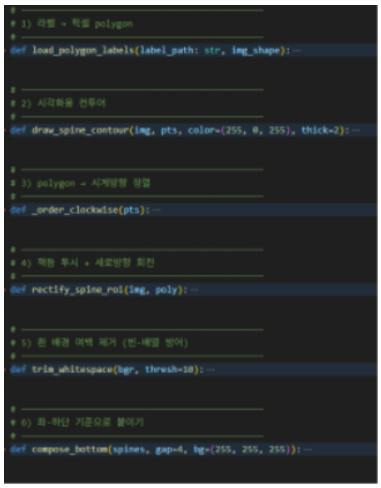
개발 상세 및 구현내용

2403110284 정윤우

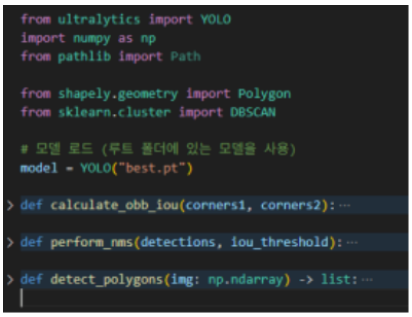
구현내용 : Flask Server

- Flask 서버에 /ocr, /rearrange, /shelf-upload, /bookshelves, /books/update, /search-books, /upload-booklist 등 다양한 API 엔드포인트를 구현하여 각 기능(책 제목 추출, 재배치, 책장 업로드, 도서 조회/수정/검색, CSV 업로드)을 지원했습니다.
- 초기 구현단계에서는 객체인식을 담당한 이국정 학생이 개발한 모델성능이 올라오기까지 객체인식 바운딩 좌표값을 label로 적어둔 csv파일을 같이 매칭시켜 서버와 앱간에 원활한 기능테스트가 이루어지도록 했으며 추후 모델의 성능이 완성된 뒤 그 모델로 교체하였습니다. 처음에 서버를 설계할때 매칭된 바운딩좌표를 가지고있는 파일이 없다면 서버에 탑재된 YOLO모델을 자동으로 사용하게 설계했기때문에 모델성능이 개선된 뒤에도 서버에서 모델교체이후에 객체탐지과정에서 수정할 사항이 없도록 개발계획을 새웠습니다.
- 재배치과정에서 객체인식단계에서 인식한 객체들을 책등별로 자른뒤 각 객체를 회전하여 수직인 상태로 변환 후 재배치하는 로직에서 재배치 이전에 잘라진 객체값들을 처음에는 Server의 Root폴더 안쪽에 ./static/spines/ 하위로 저장하였고 추후 각 이미지를 List에 담아 메모리에 저장하는 과정을 거쳤으며 추후에 DB에 이미지 자체를 저장하여 동일사진에 대한 변환에 대해서 각 기능을 전체적용을 하지 않고 일부 적용할 수 있게 개선할 예정입니다.

server.py



spine_processor.py



detection.py

개발 상세 및 구현내용

2403110284 정윤우

구현내용 : Android App (Java)

- 사용자가 AI Librarian의 기능을 모바일 환경에서 사용할 수 있도록 구현한 애플리케이션입니다. Flask 서버와 연동하여 책장 사진 업로드(책장모양 선택 및 각 칸 사진 등록), 책 제목 인식(등록된 책장속 도서들의 제목 인식), 도서 정보 조회/수정 기능을 구현했습니다.

- 주요 화면 구성 및 기능 흐름
  1. 메인화면
     - "갤러리 선택" 버튼을 통해 갤러리에서 사진을 선택 후 "책 제목 추출"버튼 선택후 "처리"버튼을 클릭하면 단일 사진에 대한 제목추출 결과가 출력됩니다.
     - 기존 사용자는 "보유 도서 검색"기능을 통해 등록된 도서가 어떤 책장의 몇 번 칸에 등록되어 있는지 확인 할 수 있습니다.
     - 등록된 책장을 선택한 후 각 책장 칸을 클릭하면 해당 칸에 등록된 도서 목록이 화면에 출력됩니다.
     - 신규 사용자는 "책장 등록"버튼을 클릭하여 새로운 책장 이름과 칸(행/열) 수를 설정하고, 각 칸에 해당하는 사진을 입력하여 업로드함으로써 책장을 등록할 수 있습니다.
     - 책장 등록 시, 입력된 사진 속 책의 제목을 확인할 수 있으며, 사용자는 인식된 도서 제목이 정확한지 확인하고 필요한 경우 수정하여 저장할 수 있습니다.
     - 이 과정에서 사용자 정보와 책장 데이터가 DB에 저장되어, 추후 재접속 시에도 서비스를 연속적으로 이용 가능합니다.

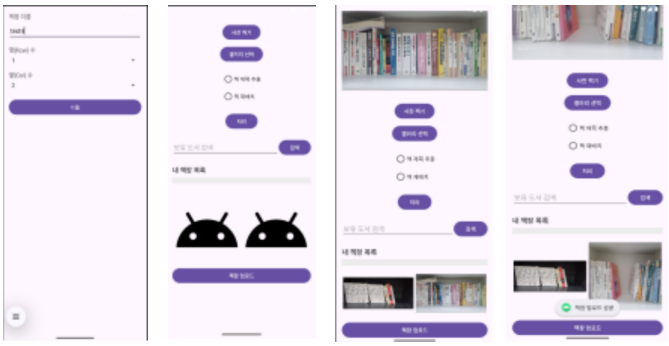
  2. 보유 도서 목록 관리
     - 기존에 등록한 도서 목록을 CSV파일로 다운로드 받을 수 있는 기능을 제공하여, 사용자의 도서 재고 관리를 용이하게 합니다.

  3. 서버와의 통신
     - 사용자의 모든 요청(사진 업로드, 기능 선택 등)은 HTTP POST/GET 방식으로 Flask 서버의 해당 기능의 API 엔드포인트로 전송됩니다.
     - 서버로부터 받은 처리 결과(책 정보, 검색결과, 수정결과)를 파싱하여 앱 화면에 시각적으로 표시함으로써 사용자에게 제공합니다.

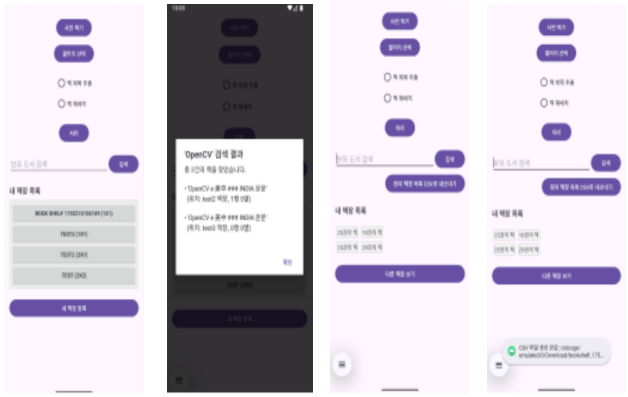
개발 상세 및 구현내용

2403110284 정윤우

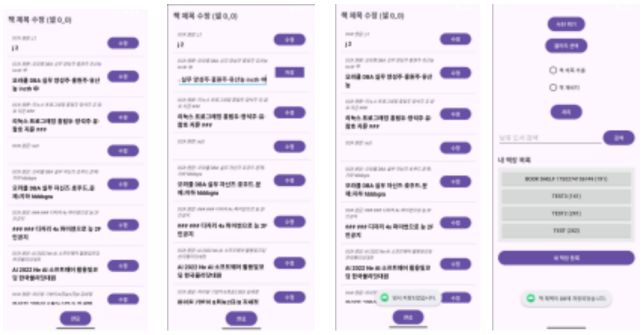
구현내용 : Android App (Java)



책장 등록



보유도서 검색, csv



등록 도서 수정

책 제목 추출 → 처리

프로젝트 진행과정 어려웠던점

2403110284 정윤우

팀장, 통합 담당자, 그리고 논문 집필자로서 여러 역할을 수행하며 프로젝트를 진행하는 과정에서 다음과 같은 어려움이 있었습니다.

1. 의사소통 부재로 인한 문제 심화
   - 초반에는 팀원들과 의견 차이가 발생했을 때 설득 과정의 어려움을 느껴 의사소통을 최소화하고 각자 담당한 역할 수행에 집중하는 팀 구조를 선택했습니다.
   - 하지만 결과적으로 의사소통이 필수적인 과제들이 후반부에 쌓이면서, 당면한 문제뿐 아니라 이전에 연관된 문제점들까지 프로젝트 마지막 단계에서 수면 위로 드러나게 되었습니다.
2. 비효율적인 문제 해결 방식과 시간관리 실패
   - 함께 모여 문제 해결을 진행하는 방식에 익숙하지 않아, 특정 문제점이 발견되었을 때 모듈 담당자에게 해결을 일임하는 방식으로 진행했습니다.
   - 이는 학기말 다른 과목들의 마감 기한과 겹치면서 통합 과정의 시간 관리 실패로 이어졌고, 최종적으로 팀원들에게 불편을 초래하는 결과를 낳았습니다.
3. 초기 설계의 한계 및 코드 구조 복잡성 증가
   - 처음 다루는 Android Studio Java 환경에서 개발하며 UI 호출 및 기능 구현 흐름을 MainActivity.java에 집중적으로 담는 방식으로 개발을 시작했습니다. 이로 인해 추후 기능별 분리의 필요성을 인지했음에도 불구하고, 기존 코드의 영향으로 주요 기능을 MainActivity에서 생성 및 호출하는 구조를 유지하게 되었습니다.
   - 모듈 담당자들과 협의한 데이터 구조에 변경이 필요했을 때, 처음부터 Data Class만 수정하고 호출 부위만 변경하도록 설계하지 못했습니다. 그 결과, 나중에 수정 시 민감도가 높아지고 초반 설계값을 고집하다가 결국 많은 코드를 수정해야 하는 상황이 발생했습니다.
   - 서버와 앱, DB를 연동하는 과정에서, 초기 설계 및 예상과 다르게 동작하거나 새로운 기능을 추가해야 할 때 기존 코드를 변경하기보다는 새로운 코드를 급하게 추가하여 앱 코드의 구조가 복잡해지는 문제가 있었습니다.

프로젝트 진행과정 느낀점

2403110284 정윤우

팀장 및 통합 담당자로서 기술적, 협업적 측면에서 다음과 같은 내용을 느꼈습니다.

1.통합 시스템 개발의 복합성과 협업의 중요성

- 이전까지는 하나의 에디터 환경 내에서 DB와 연결된 프로그램을 다루는 데 익숙했지만, 이번 프로젝트에서는 서버(Flask), 앱(Android), DB(Oracle)를 통합하고 각 기능의 로직을 연동하는 새로운 경험을 했습니다. 특히, 제가 처음부터 코드를 작성한 것이 아닌, 팀원들이 구현한 핵심 모듈들을 서버에 탑재하고 앱에 연동하며 그 결과를 데이터베이스에 적용하는 과정을 거쳤습니다. 이 과정에서 다른 팀원의 코드를 이해하고 구조를 파악하는 능력이 중요하다는 것을 배웠습니다. 이 과정에서 초반 래피드 프로토타입 작성 및 기획, 기능별 진행 방식, 데이터 교환 방식, 기능별 연결에 대해 참여했던 경험이 도움이 많이 됐습니다. 그래서 충분한 소통이 전체 시스템의 유기적인 결합에 필수적라는 점 또한 배웠습니다.

2.지속적인 의사소통의 필요성

- 프로젝트 초반, 의견 차이로 인한 설득의 어려움 때문에 의사소통을 최소화하고 각자 역할에 집중하는 방식을 택했습니다. 하지만 이는 결국 프로젝트 막바지에 이전에 연관된 문제점들까지 수면 위로 드러나게 만들었고 , 심지어 최초 계획한 기능 구현 목표를 줄이는 결과로 이어지기도 했습니다. 이러한 경험을 통해 의사소통이 아무리 힘들더라도 자주 소통하는 편이 장기적으로는 소통량을 줄이고 프로젝트 문제점이 커지는 것을 방지할 수 있다는 귀중한 교훈을 얻었습니다.

3.유연한 설계 변경 및 통합적 사고의 필요성

- Android Studio Java 환경에 대한 초기 경험 부족으로 MainActivity.java에 기능을 집중 구현하면서, 추후 기능 분리의 필요성을 인지했음에도 기존 코드의 제약으로 인해 구조 변경에 어려움을 겪었습니다. 또한, 모듈 간 데이터 구조 변경 시 Data Class만 수정하면 되도록 초기 설계하지 못하여 나중에 많은 코드를 수정해야 하는 비효율을 겪기도 했습니다. 이러한 경험들을 통해, 제가 짠 코드가 아니더라도 다른 팀원의 구현 방식과 아이디어에 대해 테스트 구조를 짜서 직접 적용해보고, 통합적인 관점에서 그 장단점을 논의하며 의견을 통합하는 과정의 중요성을 절실히 느꼈습니다. 초기 설계 단계부터 유연한 변경을 고려하고, 각 모듈의 통합성을 염두에 둔 협의가 필수적이라는 점을 배웠습니다.