INPICK

AI 기반 여행 일정 추천 플랫폼

기관

코드랩아카데미

팀명

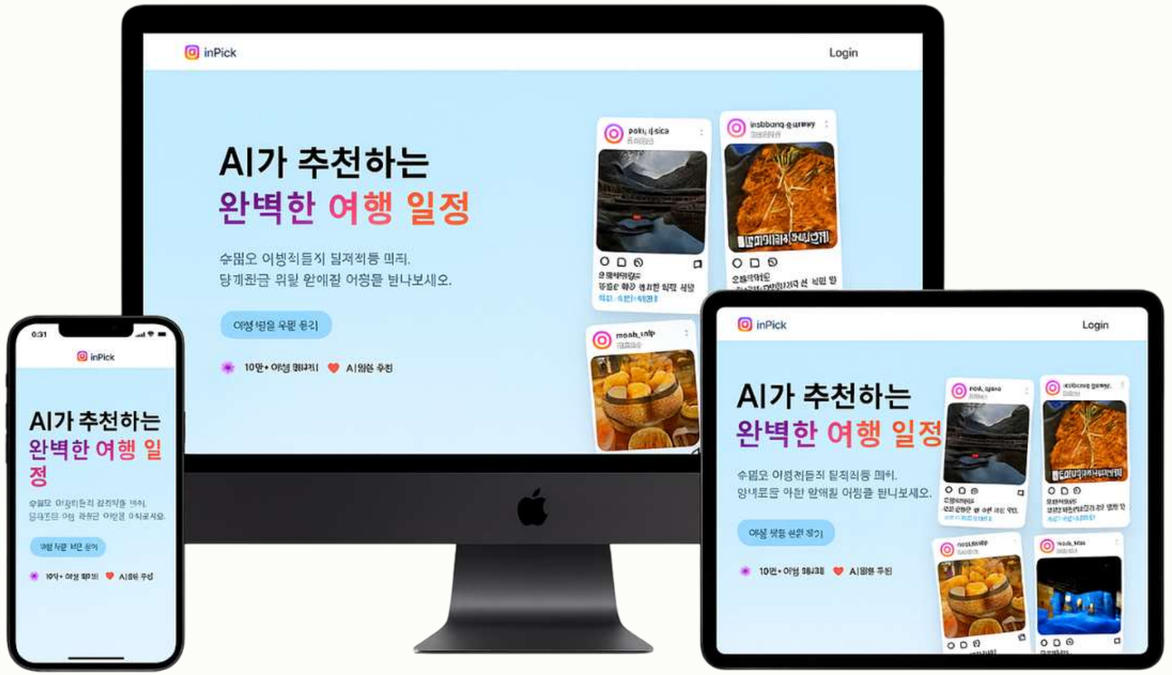
맹글러

일시

2025.11.04

CONTENTS

SITE SCREENS



SiteAdress : https://inpick.aicc-project.com

Backend : https://github.com/AICC6-Maenggler-3rd/BACKEND

Frontend : https://github.com/AICC6-Maenggler-3rd/FRONTEND

1. 프로젝트 개요

Inpick

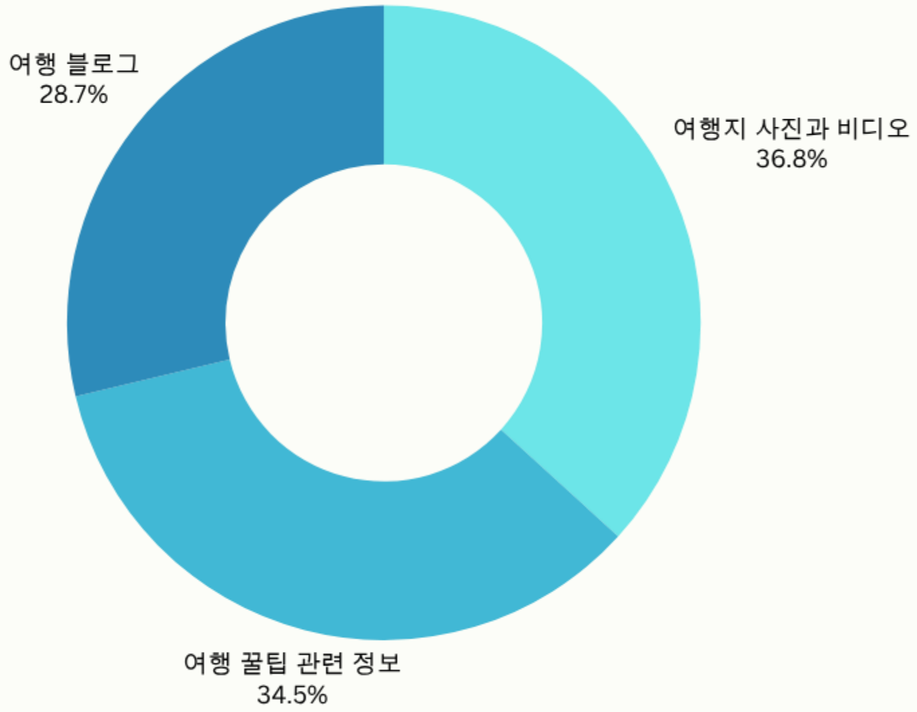
01

프로젝트 개요

< 시장 분석 >

“한국인의 여행지 선택은 이제 ‘인스타그램’에서 시작된다”

- 한국인 67.1%가 여행 계획 시 SNS(인스타그램, 유튜브 등) 참고
- 여행지 사진·영상 콘텐츠가 선택에 영향을 준다는 응답 36.8%
- 여행 동기 = ‘시각적 영감(비주얼 Inspiration)’ → 감정적 공유 욕구와 연결
- 기존 포털 검색 기반에서 **시각 경험 중심의 선택**으로 이동



출처: 한국경제 (2025-05-15) 《2030 세대 67.1% “여행지 SNS 참고” 조사결과》

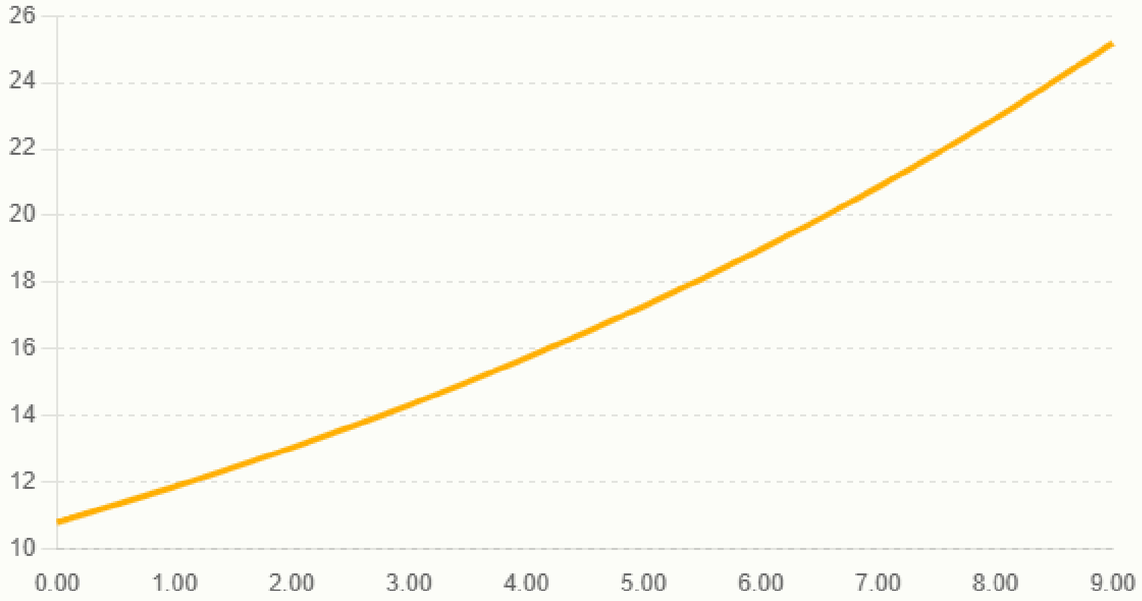
01

프로젝트 개요

< 시장 분석 >

“온라인 여행 플랫폼 성장과 AI 여행 추천 수요 확대”

- KR 한국 온라인 여행 시장 규모 (2024): 약 10.8 억 달러
- 2025–2033 년 CAGR 9.86 % 성장 예상
- 앱 기반 여행 예약 및 플래닝 앱 이용 비율 매년 증가



출처: IMARC Group (2024) 《SOUTH KOREA ONLINE TRAVEL MARKET REPORT》

01

페르소나

예상 시나리오

최근 SNS를 통해 아름다운 여행지 사진과 영상을 자주 보게 되는 **김여행 씨(27세, 직장인)**는, 가끔 마음에 드는 장소를 저장해두지만 막상 여행을 계획하려고 하면 “가고 싶은 장소는 몇 군데 안 되는데, 그 곳만 가기엔 여행을 가기 부담스러워” 포기하는 경우가 많다.

특히 SNS에서 본 장소는 특정 장소의 한 장소만 나와, 실제 여행 일정을 짜기 어렵다. 김여행 씨처럼 **‘가보고 싶은 곳은 많지만, 일정 짜기가 부담스러운 사람들’**에게 INPICK은 새로운 해결책이 된다.

김여행

- 나이 : 27세
- 직업 : 마케팅 회사 직원
- 거지주 : 서울
- 여행 빈도 : 3~4번
- 여행 스타일 : 짧고 알찬 일정, 새로운 경험 선호

01

사용자 여정

SNS 흥미

김여행 씨는 SNS에서 본 여행지 사진에 흥미를 느낌

가고 싶은 장소만으로는 일정 짜기가 번거로움

일정 짜기 어려움

INPICK 도움

INPICK이 주변 장소를 자동 추천하여 일정 구성 지원

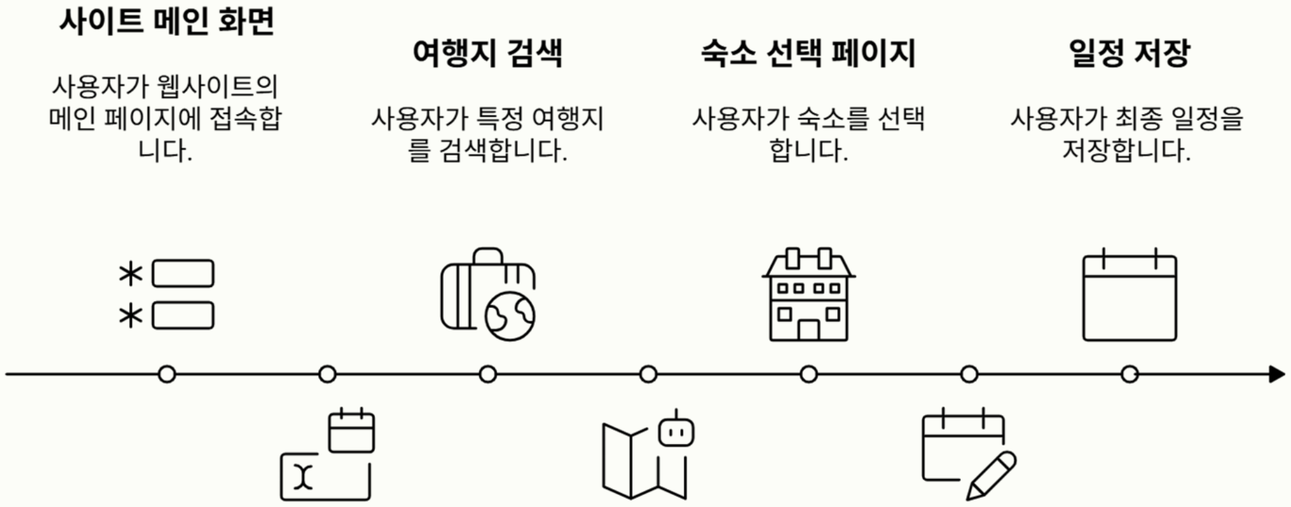
AI가 기간·테마·동행자 기반 맞춤형 일정을 생성

AI 맞춤형 일정

개인화 강화

여행 후 기록이 다음 추천에 반영되어 개인화 강화

여행 일정 생성 프로세스



사이트 메인 화면

사용자가 웹사이트의 메인 페이지에 접속합니다.

여행지 검색

사용자가 특정 여행지를 검색합니다.

숙소 선택 페이지

사용자가 숙소를 선택합니다.

일정 저장

사용자가 최종 일정을 저장합니다.

일정 생성 페이지

사용자가 키워드를 입력하여 새 일정을 시작합니다.

AI 맞춤형 일정 생성

AI가 사용자의 선호도에 따라 일정을 생성합니다.

일정 확인 및 수정

사용자가 일정을 검토하고 필요한 변경을 합니다.

2. 프로젝트 구조

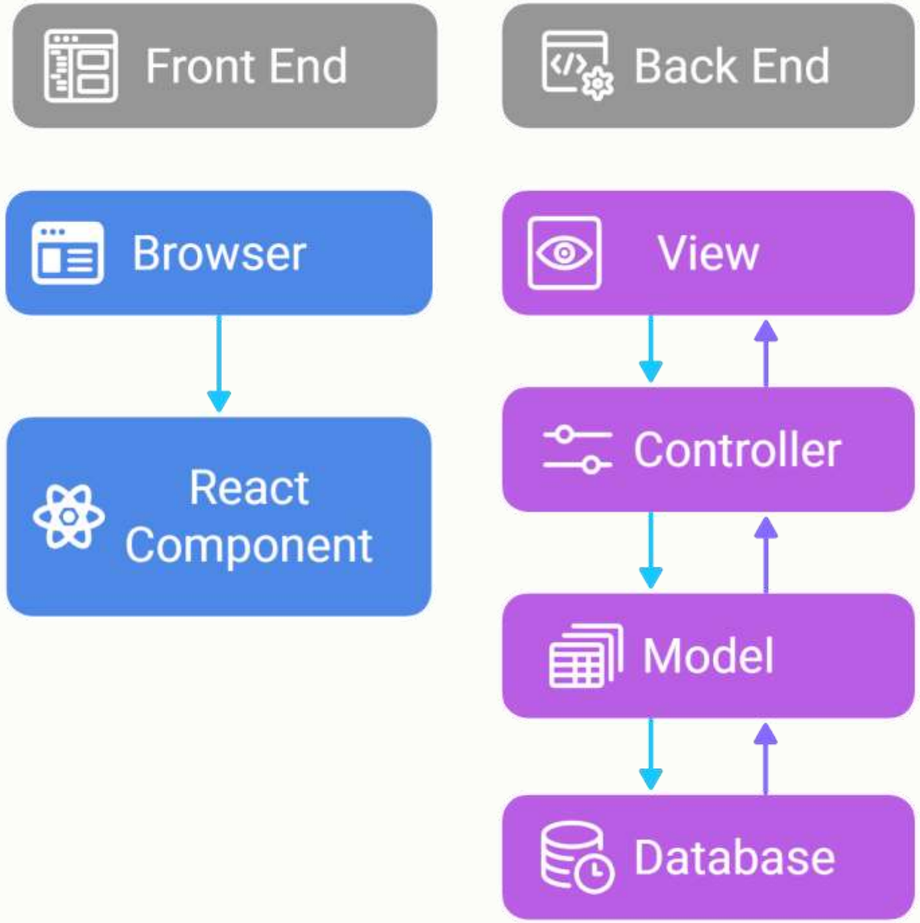
Inpick

02

프로젝트 구조

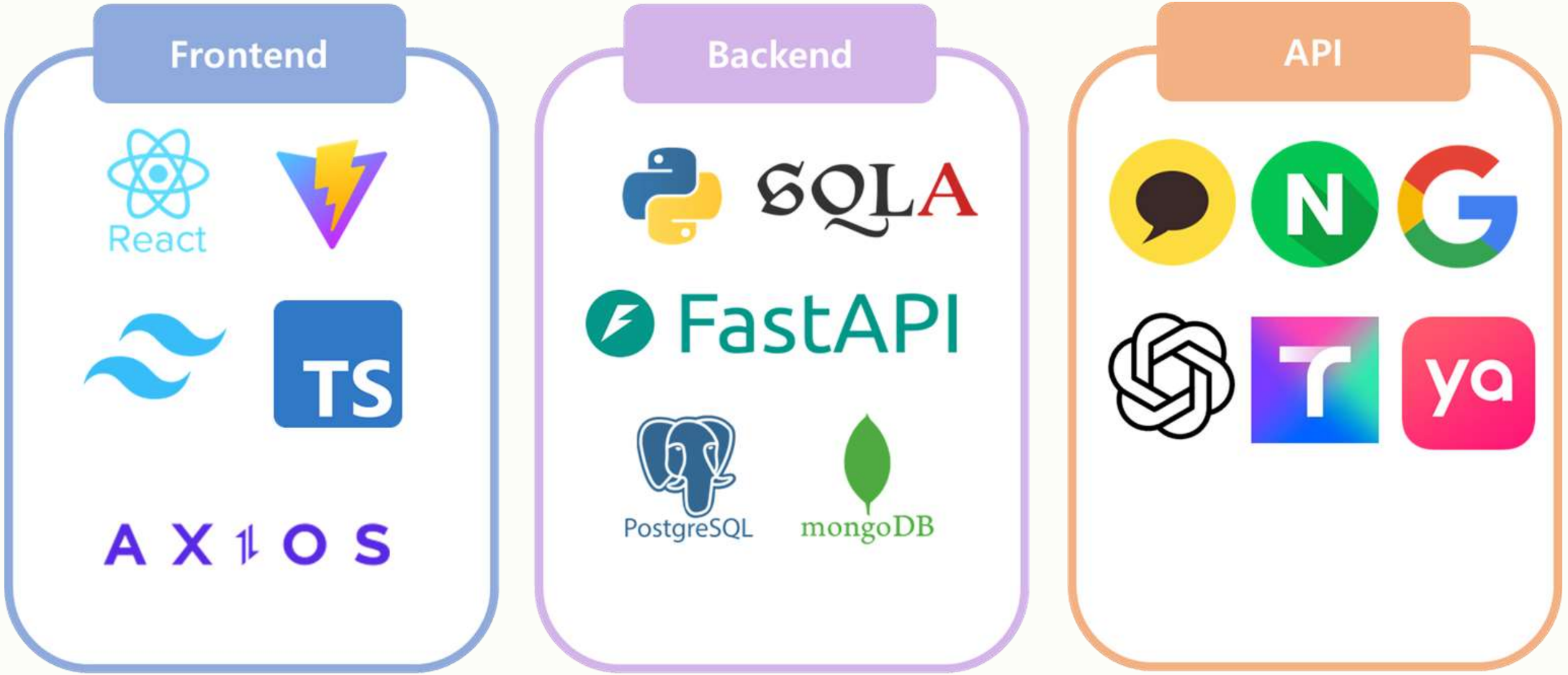
Front And Back End Structure

- 프론트엔드와 백엔드의 분리
- REST API를 통해 데이터 통신
- 레이어드 아키텍처 패턴 적용



02

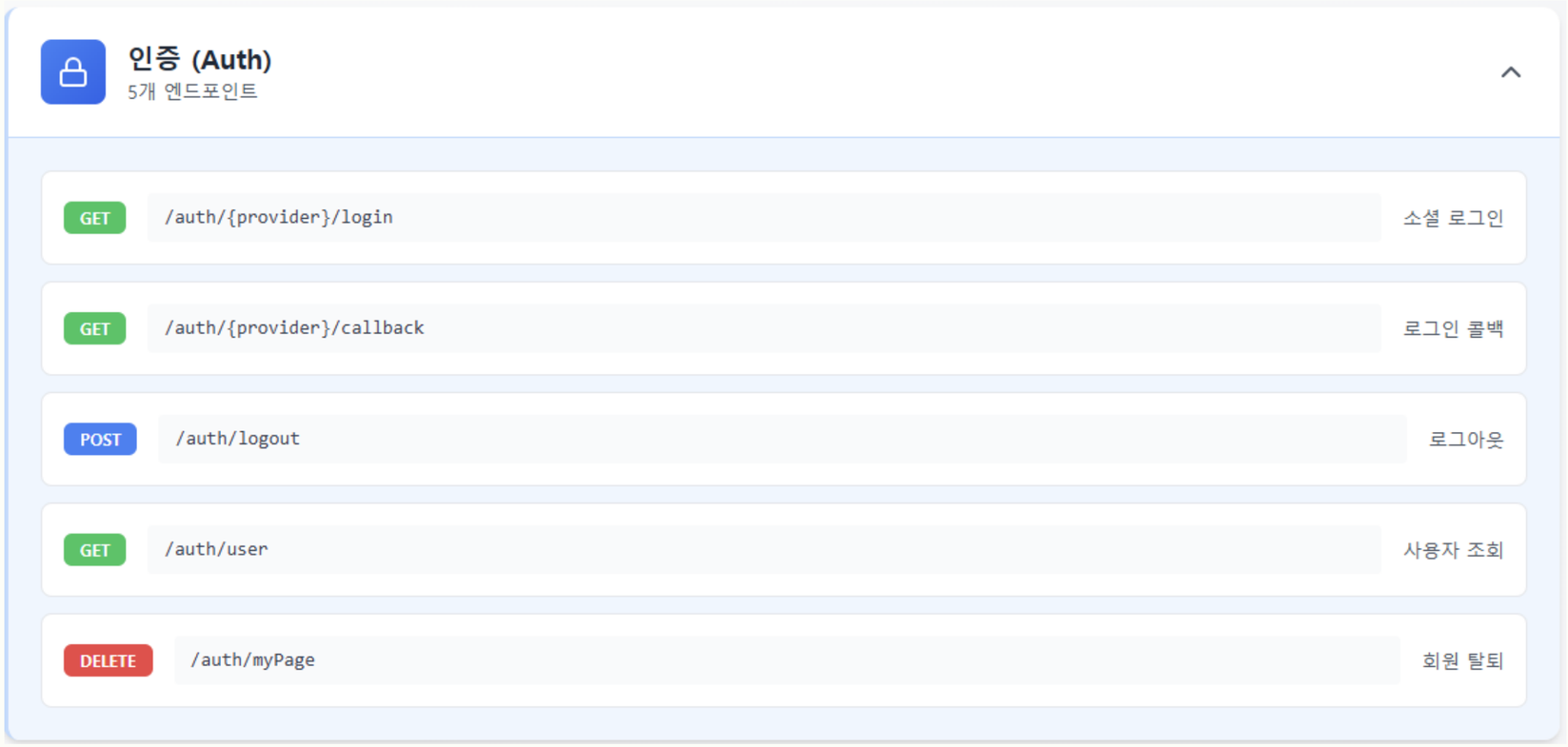
프로젝트 구조


Frontend
React
TS
AXIOS
Backend
SQLA
FastAPI
PostgreSQL
mongoDB
API
N
G
ya

02

프로젝트 구조

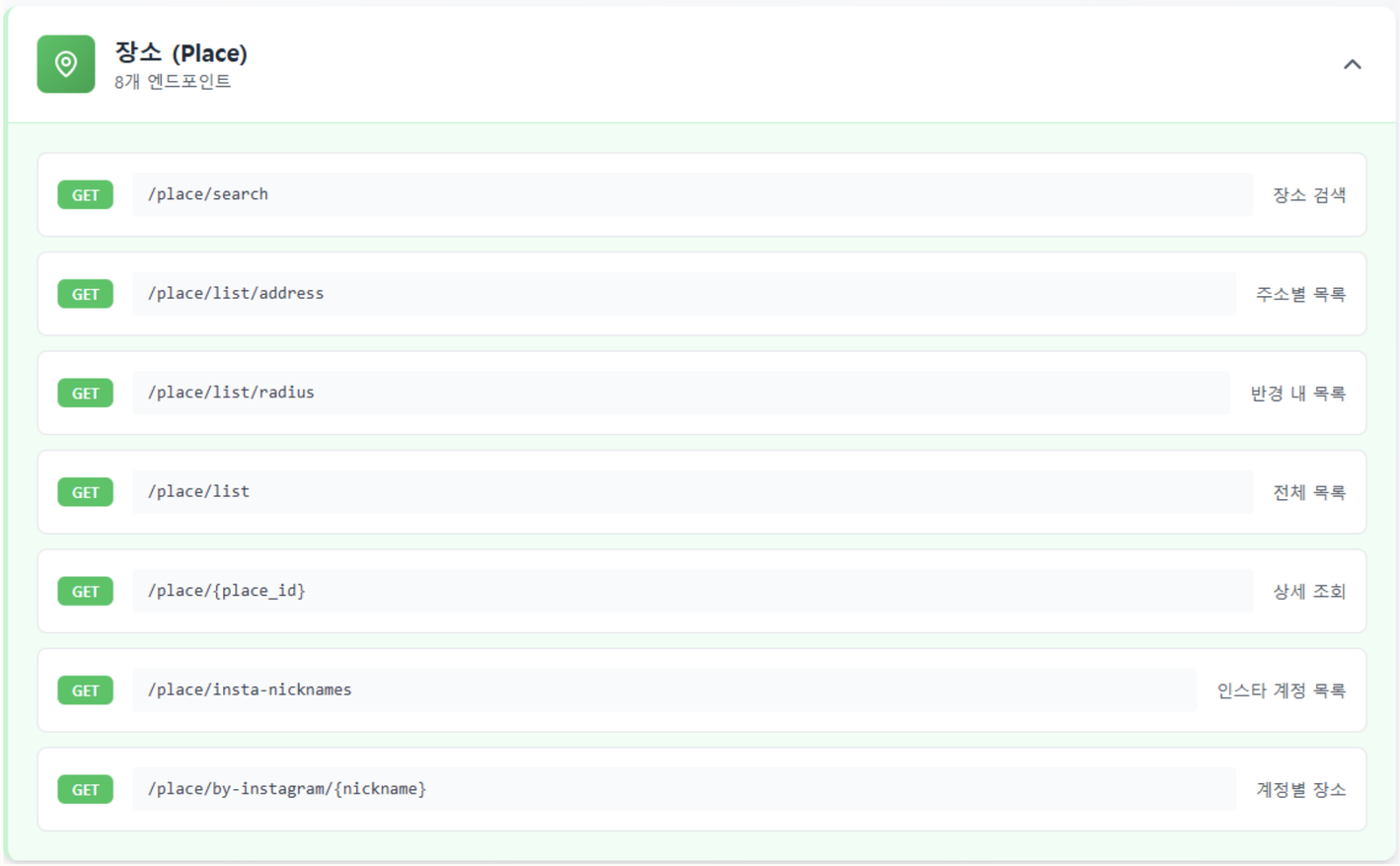
API 명세서 - 인증



02

프로젝트 구조

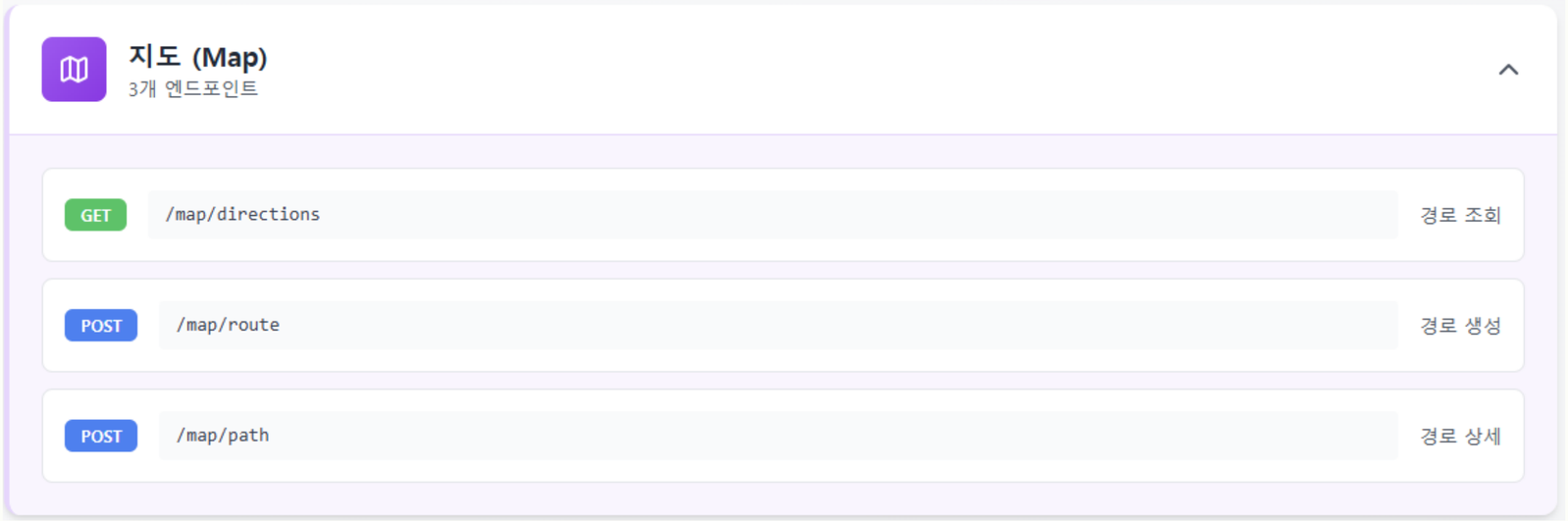
API 명세서 - 장소



02

프로젝트 구조

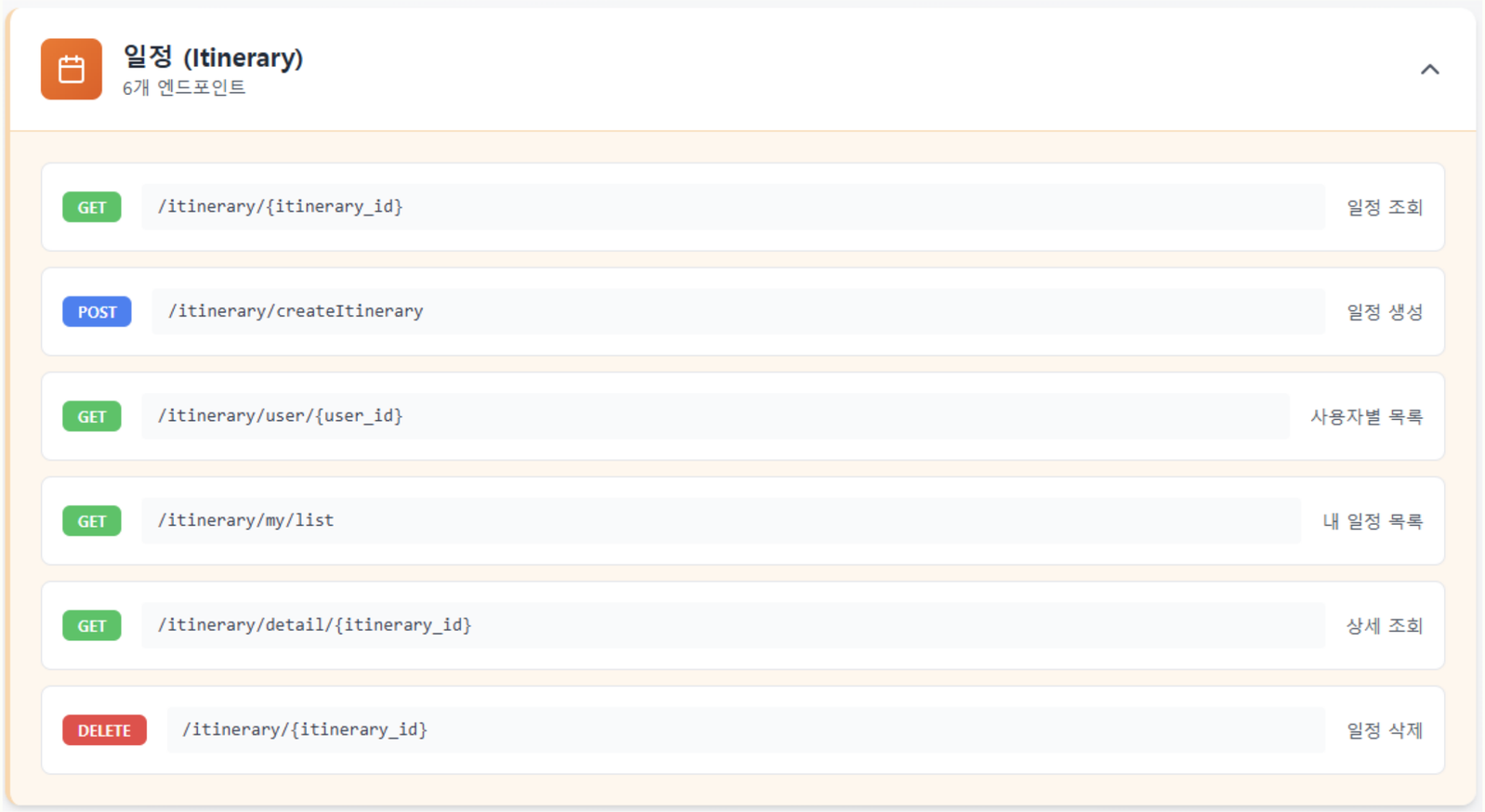
API 명세서 - 지도



02

프로젝트 구조

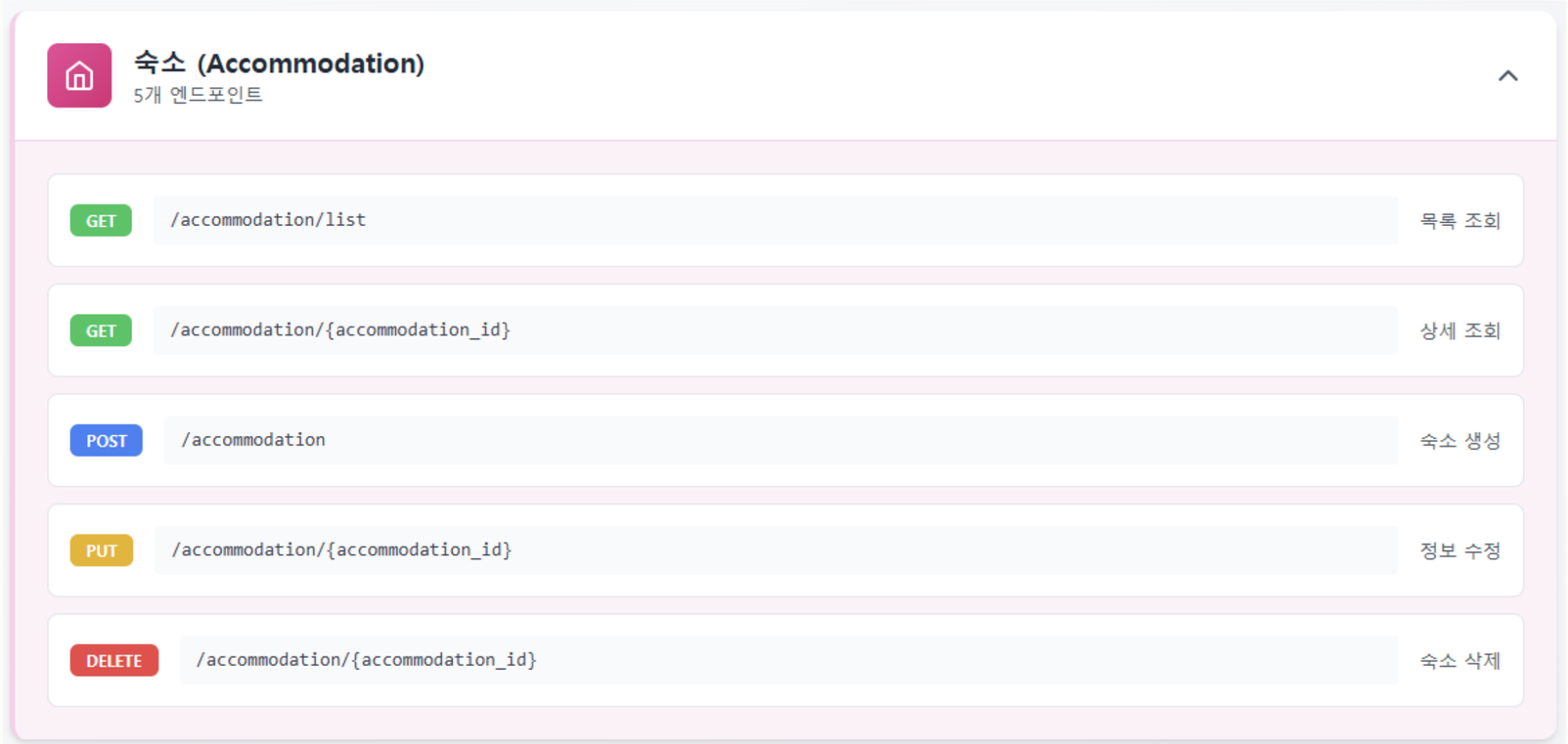
API 명세서 - 일정



02

프로젝트 구조

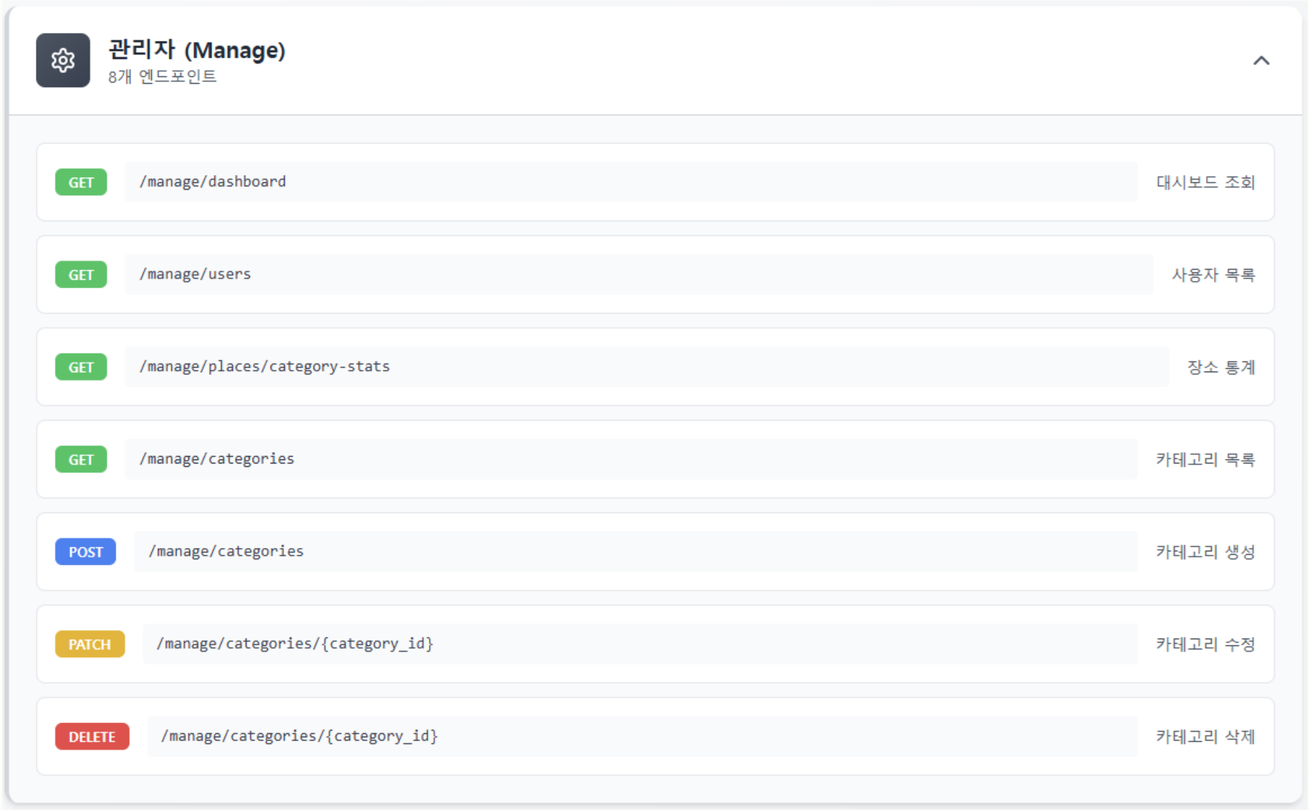
API 명세서 - 숙소



02

프로젝트 구조

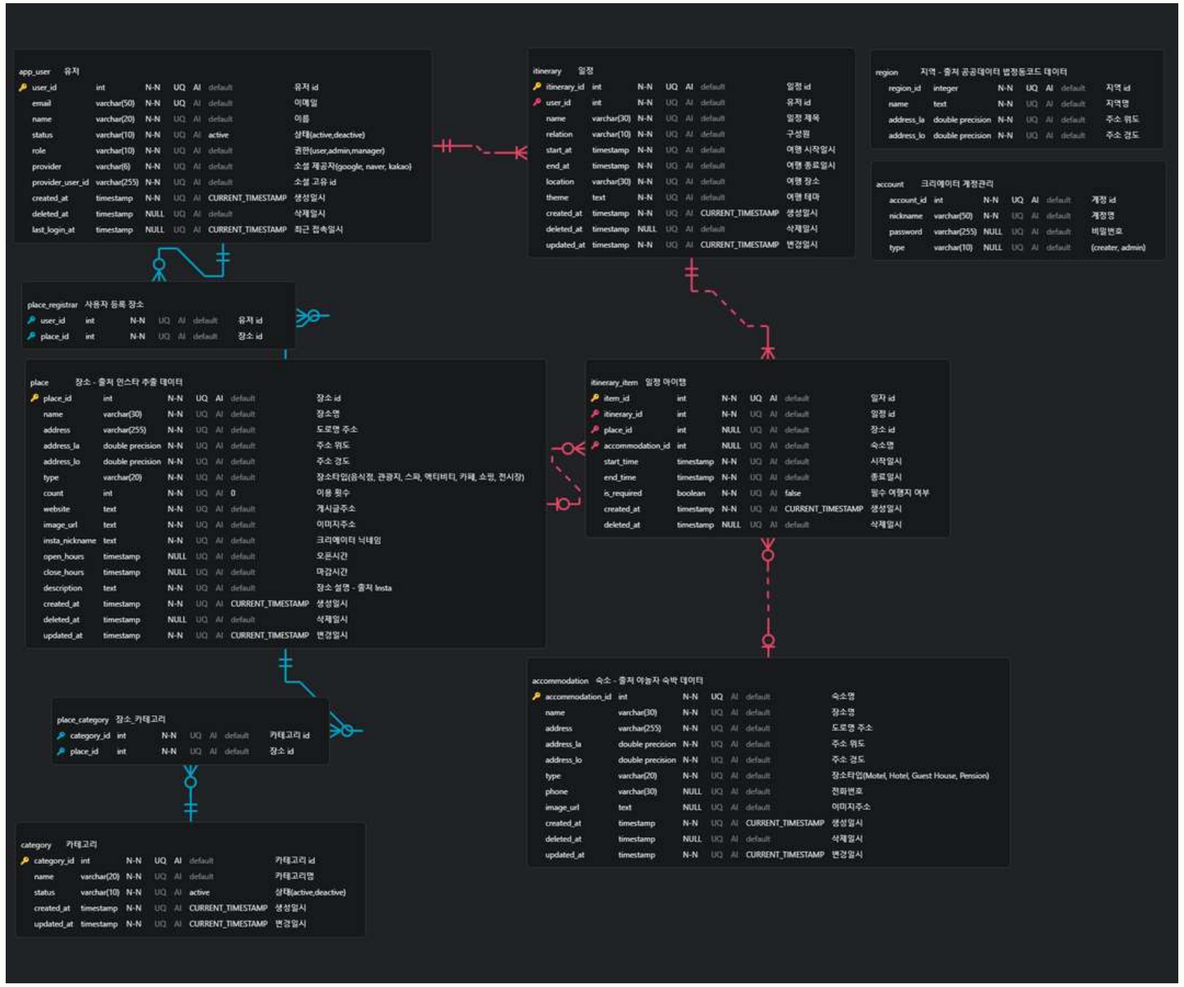
API 명세서 - 관리자



02

프로젝트 구조

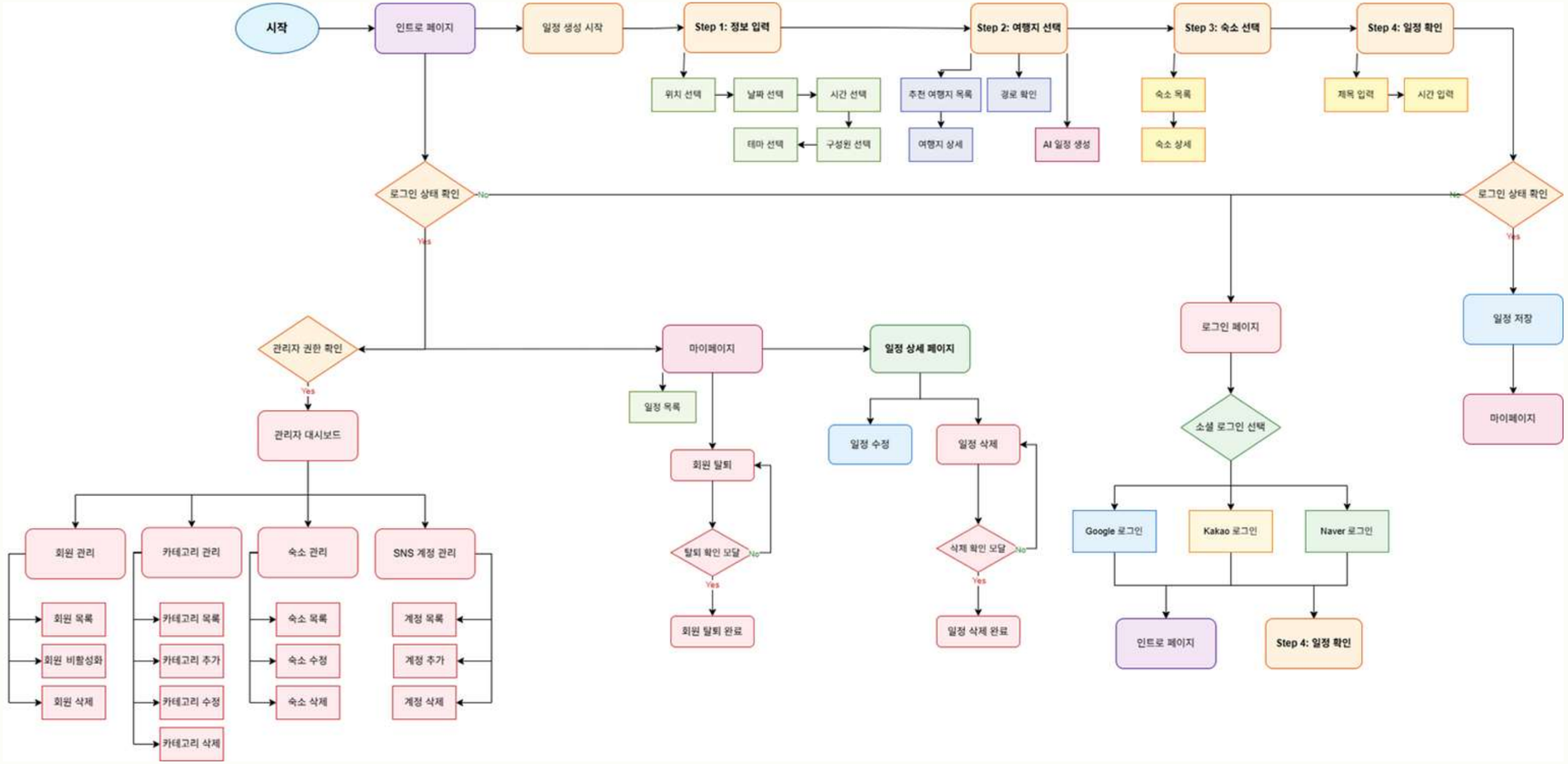
ERD



02

프로젝트 구조

Flow Chart



3. 팀 구성 및 역할

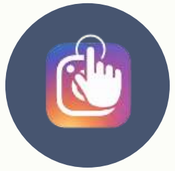
Inpick

03

팀 구성 및 역할

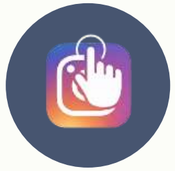
공통 작업

Backend / Frontend / 데이터 전처리



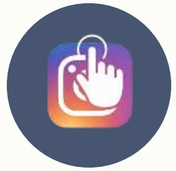
김준영 | 팀장

- 팀 프로젝트 총괄
- DB 설계 및 구현
- AI 추천 알고리즘 (ALS)



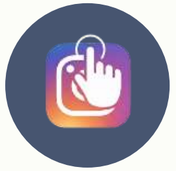
홍원섭 | 팀원

- 서비스 배포
- AI 추천 알고리즘(시퀀스)



장한결 | 팀원

- 기획 및 산출물
- 데이터 수집
- AI 추천 알고리즘(Content Based)



오민주 | 팀원

- 기획 및 산출물
- 데이터 수집
- AI 추천 알고리즘(Content Based)

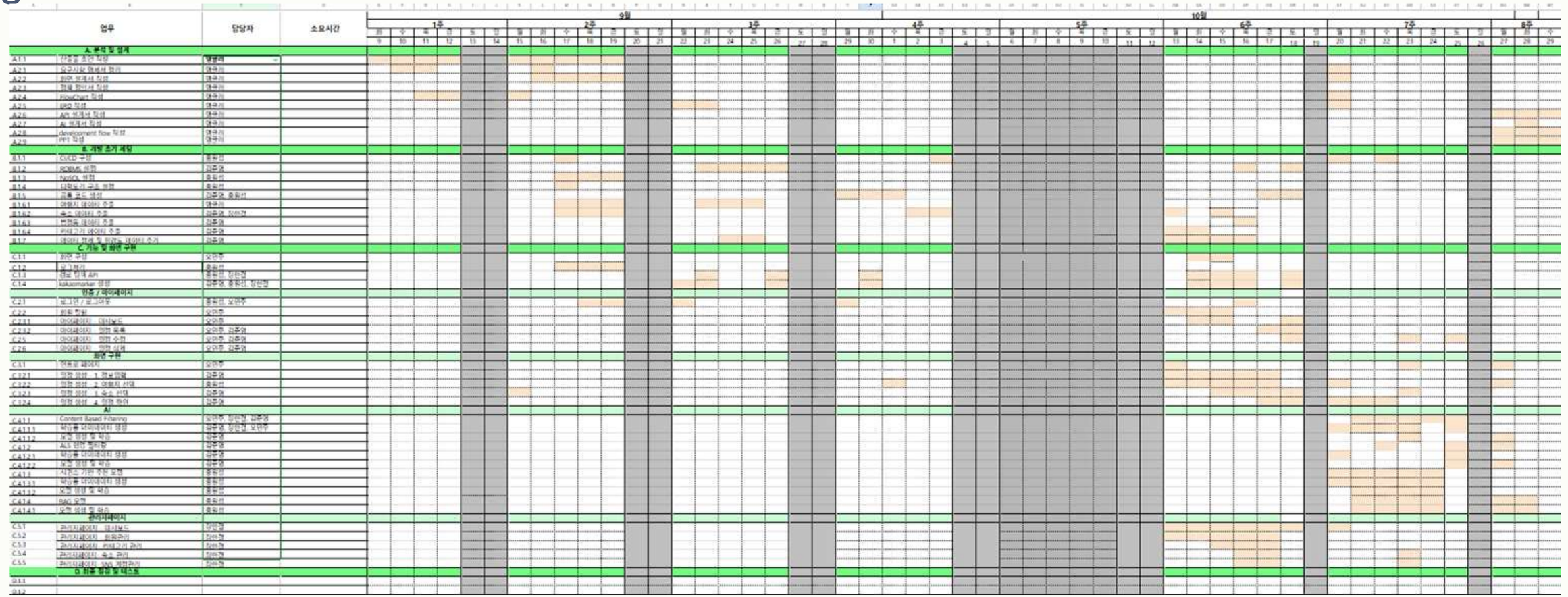
4. 수행절차 및 방법

Inpick

04

수행 절차 및 방법

WBS



04

수행 절차 및 방법

1단계 2단계 3단계 4단계 5단계

1단계

1단계

요구사항 분석 및 설계

> 주요 활동

- 사용자 유형(일반/관리자) 및 기능 정의
- DB 스키마 및 API 명세 설계
- 추천 알고리즘 구조 및 파이프라인 정의

> 주요 산출물

1. DB 스키마
2. API 명세서
3. 시스템 설계서

04

수행 절차 및 방법

1단계 **2단계** 3단계 4단계 5단계

2단계

백엔드 개발

> 주요 활동

- FAST API 기반 REST API 구현
- POSTGRESQL·MONGODB 비동기 연동 및 CRUD REPOSITORY
- DOCKERFILE 및 JENKINSFILE 작성

> 주요 산출물

1 REST API

2 배포 스크립트

04

수행 절차 및 방법

1단계 2단계 **3단계** 4단계 5단계

3단계

프론트엔드 개발

> 주요 활동

- REACT + VITE 개발 환경 구성
- KAKAOMAP SDK를 이용한 지도 기반 여행지 표시
- RADIX UI + TAILWINDCSS 디자인 시스템 구축
- 관리자용 페이지 구축

> 주요 산출물

1. REACT 앱
2. 디자인 시스템
3. 관리자 페이지

04

수행 절차 및 방법

1단계 2단계 3단계 **4단계** 5단계

4단계

통합 및 테스트

> 주요 활동

- SWAGGER UI 및 POSTMAN을 통한 API 검증
- 린팅(ESLINT, PRETTIER) 및 타입검사 통합

> 주요 산출물

1 테스트

2 API 검증

04

수행 절차 및 방법

1단계 2단계 3단계 4단계 **5단계**

5단계

요구사항 분석 및 설계

> 주요 활동

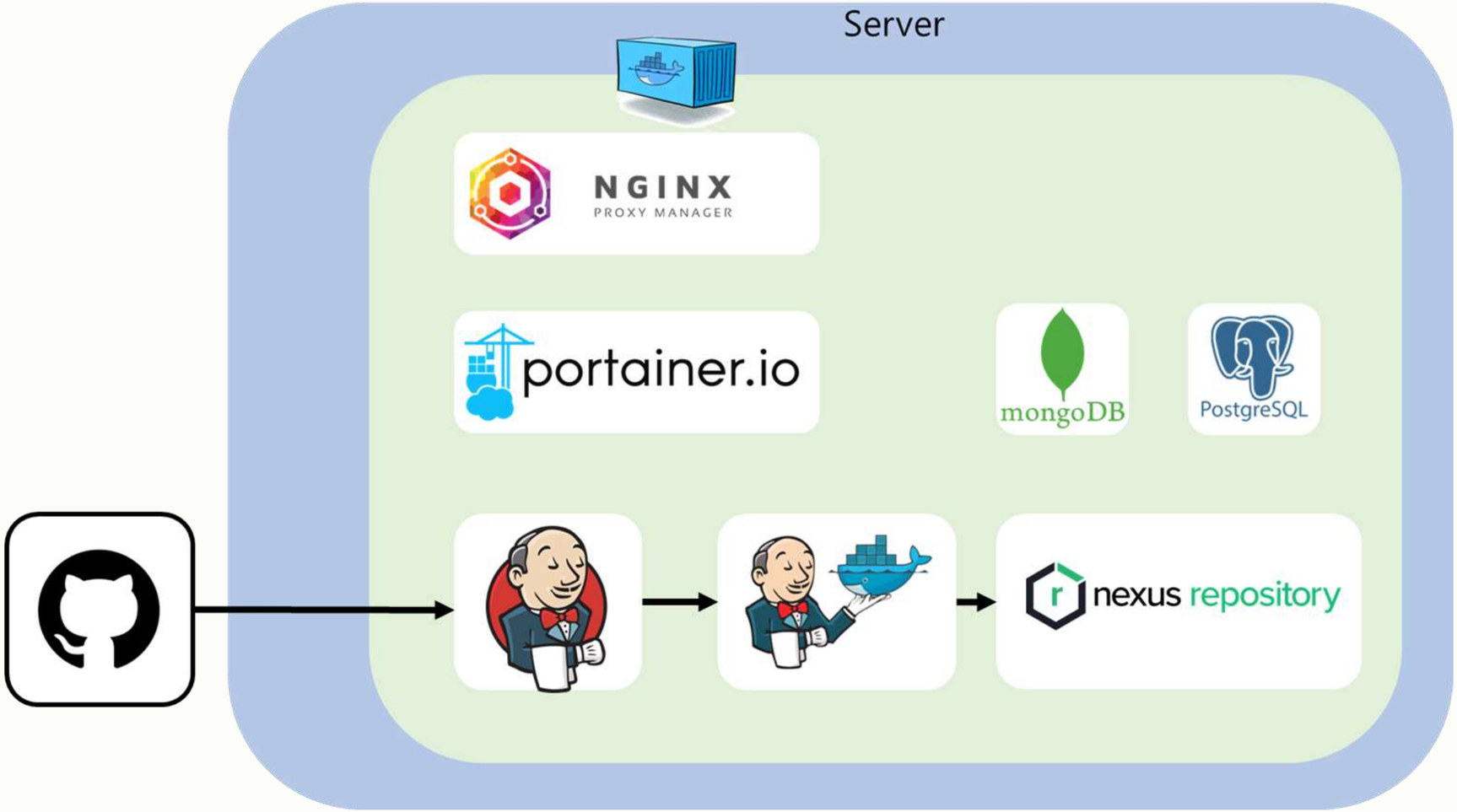
- JENKINS PIPELINE 기반 CI/CD
- PRODUCTION: inpick.alcc-project.com
- DOCKER NETWORK로 API 연동

> 주요 산출물

1. CI/CD PIPELINE
2. PRODUCTION 서버
3. 시스템 설계서

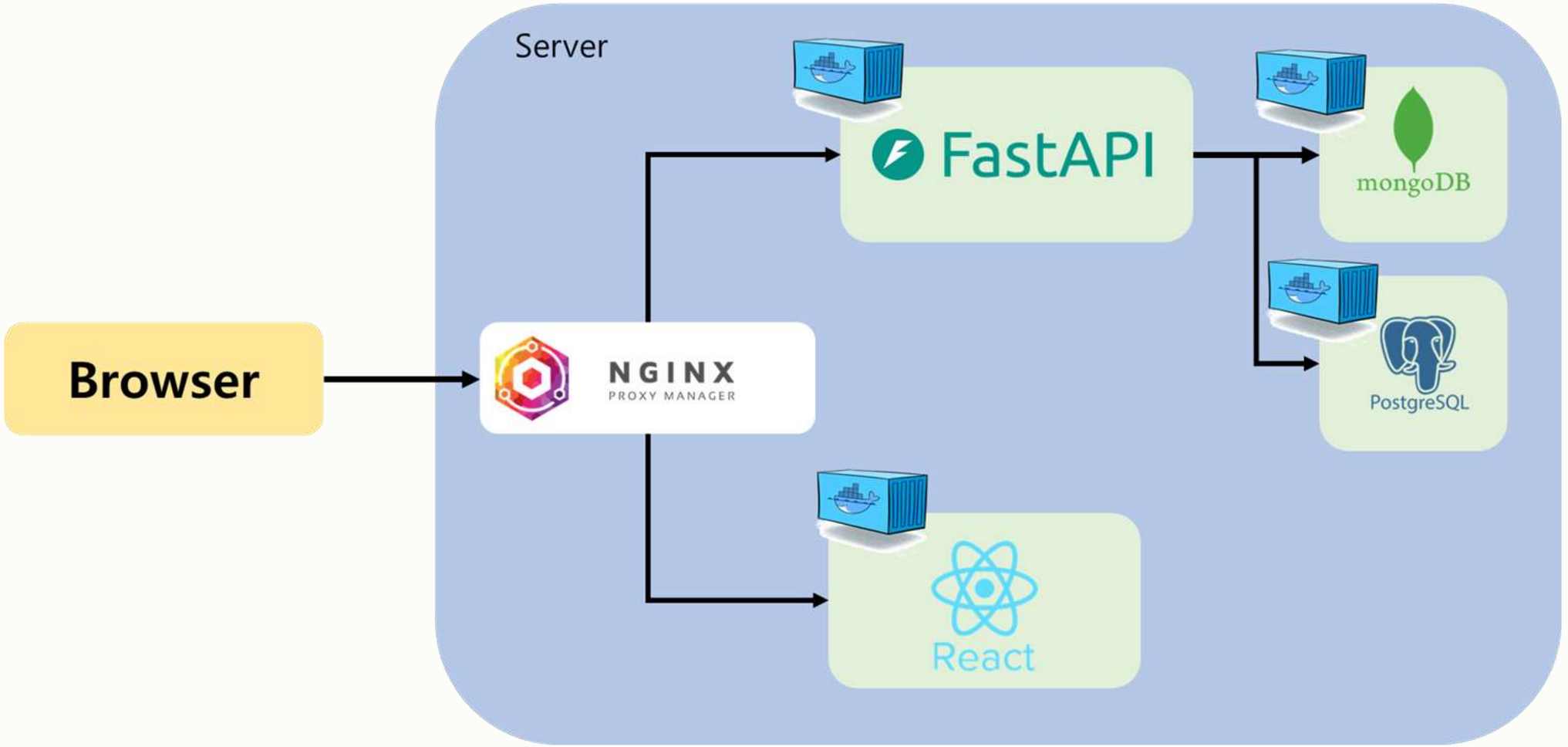
04

서버 배포


Server
NGINX
PROXY MANAGER
portainer.io
mongoDB
PostgreSQL
nexus repository

04

서버 구조


Server
FastAPI
mongoDB
PostgreSQL
Browser
NGINX
PROXY MANAGER
React

5. AI 모델 설계 및 동작 흐름

Inpick

05

AI 모델 설계 및 동작 흐름

< AI > 전체 프로세스 로직

1.사용자 입력 수집

- 지역 검색
- 날짜 지정
- 구성원 선택
- 테마 선택

2. AI 모델 사용

- 여행지 추천 :
- Content-Based
- Collaborative-Based
- 일정 생성 :
- 시퀀스 기반 필터링

3. 일정 생성

- 사용자 맞춤 일정 생성

4. 일정 수정

- 사용자 편의에 맞는 일정 수정 가능

05

AI 모델 설계 및 동작 흐름

< AI > Content-Based-Filtering

콘텐츠 기반 필터링(CBF) - 모델 생성 및 훈련 흐름

1. **DB 데이터 로드**
   데이테베이스에 저장된 여행지에 관련된 데이터를 읽습니다.
2. **데이터 전처리**
   카테고리 및 여행지에 적합한 구성원 조합을 문자열 리스트로 파싱합니다.
3. **특징 추출**
   TF-IDF 및 원-핫 인코딩을 사용하여 벡터화 시킨 후 특징을 추출합니다.
4. **특징 결합**
   추출된 특징을 결합하여 Sparse 행렬을 생성합니다.
5. **유사도 계산**
   특징 행렬을 기반으로 코사인 유사도 행렬을 계산합니다.
6. **평가**
   위치 필터링을 적용한 후 모델의 성능을 평가합니다.
7. **모델 저장**
   모델과 관련 구성 요소를 파일에 저장합니다.


```python
def evaluate(self, K: int = 10) -> dict:
    """
    전제: model, train_implicit, test_implicit 존재.
    처리: precision@K, map@K, ndcg@K, 옵션으로 auc@K.
    출력: {metric_name: float} 딕셔너리.
    상태 변화: 없음.
    예외: 내부 지표 계산 중 예외를 잡아 메시지 출력 후 해당 값 미기록 또는 키 다름(코드 그대로).
    """
    if self.model is None or self.train_implicit is None or self.test_implicit is None:
        raise RuntimeError("먼저 train()을 완료하세요.")
    results = {}
    try:
        results[f"precision@{K}"] = float(
            precision_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In precision_at_k: {e}")
    try:
        results[f"map@{K}"] = float(
            mean_average_precision_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In mean_average_precision_at_k: {e}")
    try:
        results[f"ndcg@{K}"] = float(
            ndcg_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In ndcg_at_k: {e}")
    if HAS_AUC:
        try:
            results[f"auc@{K}"] = float(AUC_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False))
        except TypeError:
            results["auc"] = float(AUC_at_k(self.model, self.train_implicit, self.test_implicit))
        except Exception as e:
            print(f"Error In AUC_at_k: {e}")
    print("📊 평가:", results)
    return results
```

평가 결과

- Precision@10: 0.6332 (±0.3150)
- Recall@10: 0.0316 (±0.0720)
- NDCG@10: 0.6154 (±0.3200)
- F1-Score: 0.0580

05

AI 모델 설계 및 동작 흐름

협업 필터링(ALS) - 모델 생성 및 훈련 흐름

1	**CSV 로드** 사용자-아이템 상호작용 데이터를 CSV 파일에서 읽습니다.
2	**매핑 생성** 사용자 및 아이템 ID를 인덱스에 매핑합니다.
3	**CSR 행렬 구성** 사용자-아이템 상호작용을 희소 행렬로 구성합니다.
4	**평점 이진화** 평점을 이진 피드백으로 변환합니다.
6	**BM25 가중치 적용** BM25 가중치를 적용하여 데이터에 가중치를 부여합니다.
7	**ALS 학습** ALS 알고리즘을 사용하여 모델을 훈련합니다.
8	**평가** 모델의 성능을 평가합니다.
9	**모델 저장** 훈련된 모델과 매핑을 파일에 저장합니다.

< AI > Collaborative Filtering

```python
def evaluate(self, K: int = 10) -> dict:
    """
    전제: model, train_implicit, test_implicit 존재.
    처리: precision@K, map@K, ndcg@K, 옵션으로 auc@K.
    출력: {metric_name: float} 딕셔너리.
    상태 변화: 없음.
    예외: 내부 지표 계산 중 예외를 잡아 메시지 출력 후 해당 값 미기록 또는 키 다름(코드 그대로).
    """
    if self.model is None or self.train_implicit is None or self.test_implicit is None:
        raise RuntimeError("먼저 train()을 완료하세요.")
    results = {}
    try:
        results[f"precision@{K}"] = float(
            precision_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In precision_at_k: {e}")
    try:
        results[f"map@{K}"] = float(
            mean_average_precision_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In mean_average_precision_at_k: {e}")
    try:
        results[f"ndcg@{K}"] = float(
            ndcg_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False)
        )
    except Exception as e:
        print(f"Error In ndcg_at_k: {e}")
    if HAS_AUC:
        try:
            results[f"auc@{K}"] = float(AUC_at_k(self.model, self.train_implicit, self.test_implicit, K=int(K), show_progress=False))
        except TypeError:
            results["auc"] = float(AUC_at_k(self.model, self.train_implicit, self.test_implicit))
        except Exception as e:
            print(f"Error In AUC_at_k: {e}")
    print("📊 평가 결과 (TOP-10):")
    print(f"  - Precision@{K}: {results[f'precision@{K}']:.4f}")
    print(f"  - MAP@{K}: {results[f'map@{K}']:.4f}")
    print(f"  - NDCG@{K}: {results[f'ndcg@{K}']:.4f}")
    print(f"  - AUC@{K}: {results[f'auc@{K}']:.4f}")
    return results
```

평가 결과

- Precision@10: 0.6001(±0.3124)
- MAP@10: 0.4723 (±0.2841)
- NDCG@10: 0.5932 (±0.2950)
- AUC@10: 0.8845

AI 모델 설계 및 동작 흐름

< AI > Sequence-Base(GRU)


```python
with torch.no_grad():
    vloss=0.0; n=0; recs=[]; mrrs=[]
    for (items, comp, cats_bt, distbin, hour, labels), _ in valid_loader:
        items, comp, cats_bt, distbin, hour, labels = \
            items.to(device), comp.to(device), cats_bt.to(device), distbin.to(device), hour.to(device),
            labels.to(device)
        logits = model(items, comp, cats_bt, distbin, hour)

        # 검증 손실 계산
        vloss += loss_fn(logits.reshape(-1, logits.size(-1)), labels.reshape(-1)).item(); n+=1

        # 마지막 타임스텝 예측 평가: Recall@10, MRR@10
        last_y = last_step_batch(labels)  # 각 시퀀스의 마지막 유효 정답 추출
        last_mask = last_y!=-100  # 유효한 샘플만 필터링

        if last_mask.any():
            # 각 시퀀스의 마지막 유효 타임스텝의 로짓 추출
            last_logits = logits[torch.arange(items.size(0)), (labels!=-100).sum(1)-1, :]
            # Recall@10과 MRR@10 계산
            r, m = recall_mrr_at_k(last_logits[last_mask], last_y[last_mask], k=10)
            recs.append(r); mrrs.append(m)

    va_loss = vloss/max(1,n)
    r10 = float(np.mean(recs)) if recs else 0.0  # 평균 Recall@10
    m10 = float(np.mean(mrrs)) if mrrs else 0.0  # 평균 MRR@10
```

🔷 평가 결과

[EP22] train_loss=2.7636 valid_loss=2.9952 recall@10=0.738 mrr@10=0.544

[EP23] train_loss=2.7507 valid_loss=2.9945 recall@10=0.740 mrr@10=0.545

[EP24] train_loss=2.7719 valid_loss=2.9942 recall@10=0.740 mrr@10=0.545

[EP25] train_loss=2.7605 valid_loss=2.9942 recall@10=0.740 mrr@10=0.545

best recall@10=0.740 at epoch 23

05

AI 모델 설계 및 동작 흐름

< AI > Sequence-Base(SAS)


```python
# ========== 학습 단계 ==========
model.train()
total_loss = 0
for seq, target in train_loader:
    seq, target = seq.to(device), target.to(device)
    optimizer.zero_grad()  # 그래디언트 초기화
    logits = model(seq)  # 순전파
    loss = criterion(logits, target)  # 손실 계산
    loss.backward()  # 역전파
    optimizer.step()  # 가중치 업데이트
    total_loss += loss.item()

# ========== 검증 단계 ==========
model.eval()
with torch.no_grad():  # 그래디언트 계산 비활성화 (메모리 절약)
    recall10_list = []
    for seq, target in val_loader:
        seq, target = seq.to(device), target.to(device)
        logits = model(seq)
        recall10 = recall_at_k(logits, target, k=10)
        recall10_list.append(recall10)
mean_recall10 = np.mean(recall10_list)

# 에포크별 학습 결과 출력
print(f"Epoch {epoch+1} ¦ Loss: {total_loss/len(train_loader):.4f} ¦ Recall@10: {mean_recall10:.4f}")
```

평가 결과

Epoch 22 | Loss: 1.6478 | Recall@10: 0.6418

Epoch 23 | Loss: 1.6477 | Recall@10: 0.6531

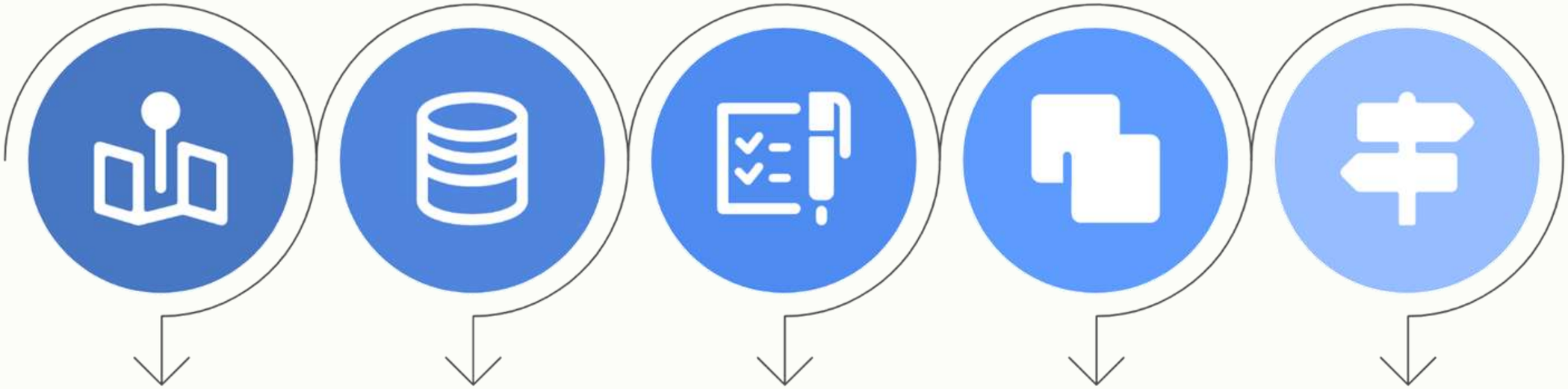
Epoch 24 | Loss: 1.6356 | Recall@10: 0.6480

Epoch 25 | Loss: 1.6488 | Recall@10: 0.6461

05

AI 모델 설계 및 동작 흐름

< AI RAG



장소 임베딩

여행 장소의 위치, 동행자, 테마, 소개, 이름을 임베딩합니다.

벡터DB 저장

임베딩된 여행 장소를 벡터 데이터베이스에 저장합니다.

쿼리 임베딩

사용자의 입력 쿼리를 임베딩합니다.

유사도 검색

쿼리 임베딩과 유사도 검색을 통해 관련 장소를 검색합니다.

장소 추천

검색된 장소 정보를 토대로 사용자에 맞는 장소 추천.

6. 수행 결과

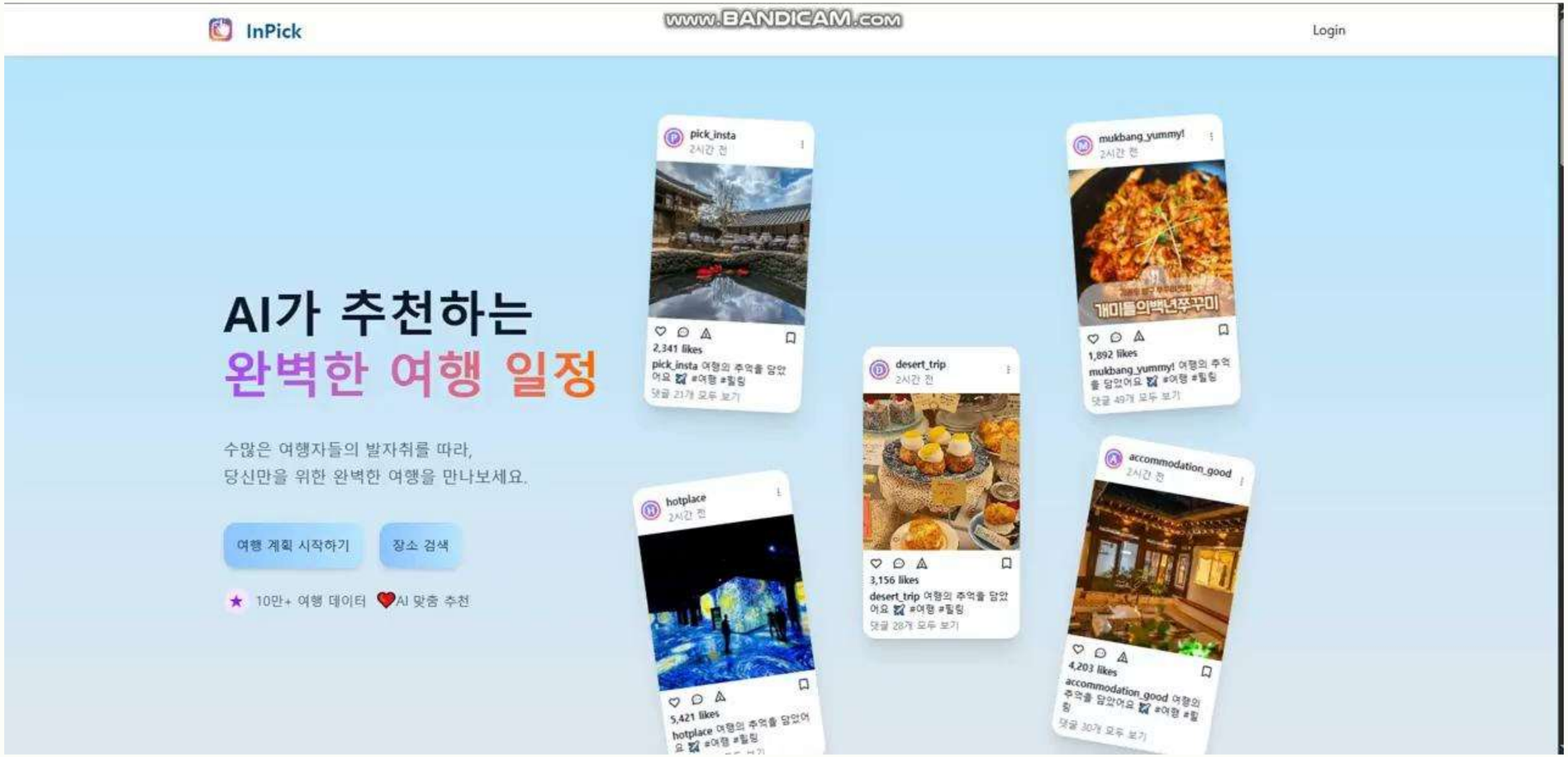
Inpick

06

수행 결과

< 링크 >

https://inpick.aicc-project.com/



07

자체 평가

김준영

팀장으로서 전체 일정과 개발 방향을 총괄하며 팀원 간의 협업을 조율했습니다. 데이터 구조의 일관성과 효율성을 확보하기 위해 데이터베이스를 직접 설계하고, 서비스 전반의 데이터 흐름을 관리했습니다. 또한 추천 시스템의 핵심인 콘텐츠 기반 필터링과 협업 필터링(ALS) 모델을 구현하여 사용자 맞춤형 여행지 추천 기능을 완성했습니다. 모델 성능 개선을 위해 다양한 하이퍼파라미터를 실험했지만 평가 점수가 만족스럽지 못한게 아쉽습니다.

프로젝트를 이끌며 문제 해결 능력을 강화했고, 팀 전체의 개발 방향을 조정하며 다양한 AI 모델 적용을 통해 역량을 향상시킬 수 있었던 프로젝트였습니다.

장한결

서비스 기획부터 데이터 수집, 추천 알고리즘 개발까지 전 과정을 맡으며 프로젝트의 흐름을 종합적으로 이해할 수 있었습니다.

콘텐츠 기반 추천 알고리즘을 구현하며 사용자 경험을 개선하는 기술적 접근의 재미를 느꼈습니다.

데이터 분석과 모델 설계를 반복하며 논리적 사고력과 문제 해결 능력이 한층 성장했다고 생각합니다.

홍원섭

이번 프로젝트를 통해 데이터를 활용하여 사용자 맞춤형 일정을 생성하는 과정을 직접 구현하며 데이터 기반 서비스 개발의 중요성을 체감했습니다.

GRU4Rec, SASRec, RAG 모델을 개발하며 추천 시스템과 생성형 AI의 동작 원리를 깊이 이해할 수 있었습니다.

특히 데이터 전처리와 모델 튜닝 과정에서 시행착오를 겪으며 문제 해결 능력이 향상되었습니다.

프론트엔드와 백엔드를 통합하는 과정에서 협업과 시스템 구조 설계의 중요성을 배웠습니다.

오민주

이번 프로젝트 초기 단계에서 기획과 산출물을 직접 작성하며 서비스의 전체 구조와 목표를 명확히 정의할 수 있었습니다.

인스타그램 데이터를 수집하고 정제하는 과정에서 데이터 품질의 중요성과 자동화의 필요성을 체감했습니다.

콘텐츠 기반 추천 알고리즘을 구현하며 사용자 특성에 맞는 개인화 로직을 설계하는 경험을 쌓았습니다.

기획부터 개발까지 전 과정을 경험하면서 데이터 중심 서비스 개발에 대한 통찰과 실무 감각을 키울 수 있었습니다.

08

산출물

문서(별첨)

- 요구사항 정의서
  - AI 설계서
- 개발 가이드
  - 프론트엔드
  - 백엔드
- 화면 설계서
- WBS
- ERD
- API 명세서
- FlowChart