**응급실 혼잡도 예측 웹서비스 개발**

**부산 지역 응급실 위치 및 혼잡도 예측 정보 제공 서비스**

1. **프로젝트 개요**
2. 목적

* 응급 상황 시, 사용자의 현재 위치와 가까운 응급실을 시각화하고, 응급실의 예측 혼잡도를 함께 제공함으로써 신속하고 효율적으로 응급실을 선택할 수 있는 서비스 제공

1. 핵심 기능

* 현재 위치와 가까운 응급실 찾기
* 머신 러닝으로 응급실 혼잡도 예측(1~3시간 후)
* 최적 경로 안내 및 소요시간 표시
* 응급실 상세 정보 제공

1. 기술 스택(상의 후 수정 필요)

* 백엔드: Java + Spring Boot
* 프론트엔드: React.js
* 데이터베이스: My sql
* AI/ML: python(streamlit)
* 외부 API: 카카오 맵, 공공데이터 포털(data.go.kr)
* IDE: IntelliJ IDEA

1. **시스템 아키텍처**
2. 전체 구조

프론트->데이터베이스->백엔드->AI 예측 서비스-> 외부 API



1. 주요 컴포넌트

* 응급실 검색 서비스: 위치 기반 병원(응급실) 찾기
* 혼잡도 예측 서비스: 머신 러닝 모델 활용 예측
* 지도 서비스: 카카오 맵 API 연동
* 데이터 수집 서비스: 공공데이터 자동 수집

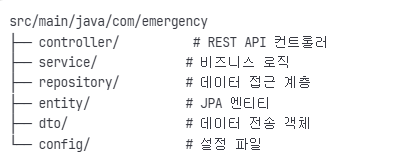
1. **데이터 수집 전략**
2. 필수 데이터

* 응급의료기관 정보(data.go.kr)
* 기관명, 주소, 연락처, 좌표
* 진료과목, 운영시간
* 응급실 이용 통계(data.go.kr)
* 시간대 별/ 요일 별 환자 수
* 평균 대기 시간, 중증도 별 분포
* 실시간 보조데이터
* 날씨 정보(기상청 API)
* 교통상황(카카오 모빌리티)

1. AI를 활용할 부분

* 공공데이터 API 연동 코드 생성
* 데이터 전처리 파이프라인 구축
* 크롤링 스크립트 작성

1. **SpringBoot 백엔드 구축**
2. 프로젝트 구조(config 패키지는 사용X)



1. 핵심 API 엔드포인트

* GET /api/hospitals/nearby: 주변 응급실 검색
* GET /api/hospitals/{id}/congestion: 혼잡도 예측
* GET /api/route/{hospitalId}: 경로 안내
* POST /api/feedback: 사용자 피드백

1. AI를 활용할 부분

* SpringBootStarter 코드 생성 요청
* JPA Entity 클래스 자동 생성
* REST API 컨트롤러 템플릿 생성
* 테스트 코드 자동작성

1. **예측 모델 구축**
2. 혼잡도 예측 알고리즘

* 입력 변수: 시간, 요일, 월, 날씨, 과거 이용률
* 모델: LSTM(시계열 예측) -> 상의 후 변경될 수도 있음
* 출력 형태: 여유/보통/혼잡/매우 혼잡(총 4단계)

1. Python Flask API 서버(Flask 안쓰고 streamlit으로 변경 함)



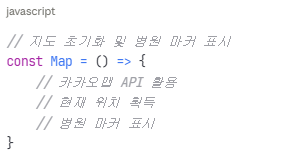
1. AI를 활용할 부분

* 시계열 예측 모델 코드 생성 요청
* 데이터 전처리 파이프라인 자동화
* 모델 성능평가 코드 작성

1. **React 프론트엔드 개발**
2. 주요 컴포넌트

* MapContainer: 카카오 맵 표시
* HospitalList: 응급실 목록
* HospitalDetail: 병원 상세 정보
* RouteGuide: 경로 안내

1. 카카오 맵 연동



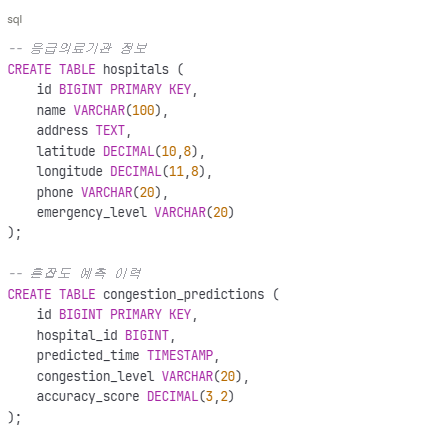
1. UI/UX 핵심 요소

* 응급상황을 고려한 간단한 인터페이스
* 큰 버튼, 명확한 색상 구분

1. **데이터베이스 설계**

**(아래 사진 참고용, 만들 때 수정 필요)**

1. 핵심 테이블



1. Redis 캐싱 전략 (강사님이랑 상이 후 안하기로 함)

* 응급실 검색 결과(5분 캐시)
* 예측 결과(1시간 캐시)
* 지도API 응답(30분 캐시) – 연결된 카카오 맵 API호출 비용 안들게 설정 필요

1. **개발 단계별 가이드**

**순서는 2. 시스템 아키텍처대로, 아래 내용은 참고**

1. 환경 설정

* IntelliJIDEA에서 SpringBoot 프로젝트 생성
* 필수 의존성 추가(JPA, Web, Redis 등)
* 데이터베이스 연결 설정
* Git 저장소 초기화

1. 백엔드 기본 구조

* Entity 클래스 작성(Hospital, Prediction)
* Repository 인터페이스 구현
* 기본 CRUD API 개발
* 공공 데이터 연동 서비스 구축

1. AI 모델 개발

* 데이터 분석 및 전처리
* LSTM 모델 훈련
* Flask API 서버 구축(streamlit으로)
* SpringBoot와 연동

1. 프론트엔드 개발

* React 프로젝트 초기화
* 카카오 맵 API 연동
* API 통신 로직 구현
* 반응형 UI 구현

1. 통합 및 테스트

* 전체 시스템 통합
* 기능 테스트 수행
* 성능 최적화
* 배포 준비(생략)

1. **AI 사용 시 참고할 프롬프트**

1) 코드 생성 프롬프트



2) 데이터 처리 프롬프트



3) 테스트 코드 프롬프트



4) 개발 효율성 극대화 방법

- 반복 작업 자동화: CRUD 코드, DTO 변환 로직

- 복잡한 알고리즘: 거리 계산, 데이터 분석 로직

- 테스트 코드: 단위 테스트, 통합 테스트

- 문서화: API 문서, 코드 주석

**10. 배포 및 운영(생략)**

1) 배포 환경

- 개발: 로컬 환경(H2 Database)

- 테스트: AWS EC2(Docker)

- 운영: AWS ECS + RDS

2) 모니터링 포인트

* API 응답시간(<2초)
* 예측 정확도(>80%)
* 시스템 가용성(>99%)
* 동시 사용자 처리(>1000명)

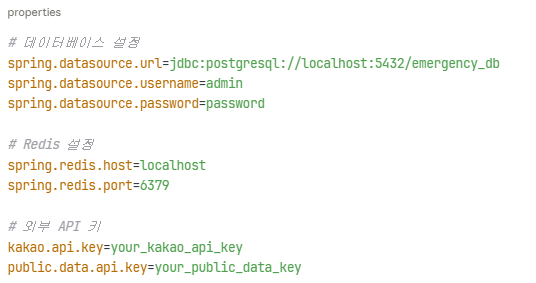
1. 유지보수 계획

* 주간 데이터 품질 체크
* 월간 모델 재학습
* 분기별 성능 최적화
* 사용자 피드백 기반 개선

**\*부록\***

<즉시 사용 가능한 코드 탬플릿>

1. SpringBoot Application.properties



1. 핵심 Entity 클래스



1. 거리 계산 유틸리티

