

System Modeling and Design Document

Stroke Spoiler: 뇌졸중 예방 시스템

Team 4

Sunwoo Kwon, Minjun Kim,
Giwon Jang, Buyeon Hwang

2025.11.18.

Phase 1 – System Modeling

1.1. System Summary

뇌졸중 예방 시스템은 **4개의 주요 계층**으로 구성되어 있습니다:

- **User Layer (사용자 계층)**: 환자, 보호자, 주치의, 관리자
- **Data Layer (데이터 계층)**: 건강 데이터, 예측 결과, 이력 정보
- **Analysis Layer (분석 계층)**: AI 예측, What-if 분석, 추세 예측 등
- **Service Layer (서비스 계층)**: 인증, 알림, 데이터 검증 등

1.2. Class Diagram & Sequence Diagram

다이어그램 사진은 **Appendix A** 참조해주시기 바랍니다.

1.3. System Modeling Details

클래스 다이어그램에 표현된 클래스들 및 그 속성/메서드, 클래스간 관계에 대한 상세한 설명입니다.

1.3.1. User 관련 클래스

User (Abstract Class)

설명: 모든 사용자 유형의 기본 클래스

속성 (Attributes)

속성명	타입	접근제어	설명
userId	String	private	사용자 고유 ID
email	String	private	이메일 주소 (로그인용)
password	String	private	암호화된 비밀번호
name	String	private	사용자 이름
phoneNumber	String	private	전화번호
createdAt	DateTime	private	계정 생성 일시

메서드 (Methods)

메서드명	반환타입	파라미터	설명

login()	boolean	-	로그인 수행
logout()	void	-	로그아웃 수행
updateProfile()	void	-	프로필 정보 업데이트

관계 (Relationships)

- Patient, Guardian, Physician, Administrator의 부모 클래스 (상속)

Patient (개인 사용자)

설명: 뇌졸중 예방을 위해 시스템을 사용하는 환자

상속: User 클래스 상속

속성 (Attributes)

속성명	타입	접근제어	설명
age	int	private	나이
gender	Gender	private	성별 (MALE/FEMALE/OTHER)
medicalHistory	List	private	병력 정보

메서드 (Methods)

메서드명	반환타입	파라미터	설명
enterHealthData()	void	data: HealthData	건강 데이터 입력
viewRiskPrediction()	RiskPrediction	-	위험도 예측 결과 조회
shareDataWithGuardian()	void	guardian: Guardian	보호자와 데이터 공유
shareDataWithPhysician()	void	physician: Physician	주치의와 데이터 공유
runWhatIfAnalysis()	AnalysisResult	scenario: Scenario	What-if 분석 실행

관계 (Relationships)

- 1 Patient --- 0..* HealthData: 여러 건강 데이터 기록 보유
- 1 Patient --- 1 HealthDataHistory: 건강 데이터 이력 보유
- 1 Patient --- 0..* RiskPrediction: 여러 위험도 예측 결과 수신
- 1 Patient --- 1 PredictionHistory: 예측 결과 이력 보유
- 0..* Patient --- 0..* Guardian: 여러 보호자가 모니터링
- 0..* Patient --- 0..* Physician: 여러 주치의가 진료

- 1 Patient --- 0..* SharingPermission: 공유 권한 부여
- 1 Patient --- 0..* WhatIfAnalysis: What-if 분석 수행
- 1 Patient --- 0..1 PercentileRanking: 순위 비교 정보 보유
- 1 Patient --- 0..* TrendForecasting: 추세 예측 정보 보유

책임 (Responsibilities)

- 자신의 건강 데이터를 입력하고 관리
- 위험도 예측 결과 확인 및 개선 방안 모색
- 보호자 및 주치의와 데이터 공유 권한 관리
- 개인 맞춤형 분석 기능 활용

참조 요구사항: HD001, HD004, RP002, WA001, MP001, MP002, MP003, MP004

Guardian (보호자)

설명: 환자의 건강 상태를 모니터링하는 가족/보호자

상속: User 클래스 상속

속성 (Attributes):

속성명	타입	접근제어	설명
relationship	String	private	환자와의 관계 (부모, 자녀, 배우자 등)

메서드 (Methods):

메서드명	반환타입	파라미터	설명
viewPatientStatus()	HealthStatus	patient: Patient	환자의 건강 상태 조회
receiveAlert()	void	alert: Notification	위험 알림 수신

관계 (Relationships):

- 0..* Guardian --- 0..* Patient: 여러 환자 모니터링 가능
- 1 Guardian --- 0..* SharingPermission: 공유 권한 수신
- 1 Guardian --- 0..* Notification: 알림 수신

책임 (Responsibilities):

- 환자의 건강 상태를 원격으로 모니터링
- 고위험 상황 발생 시 즉시 알림 수신
- 환자가 부여한 권한 범위 내에서 정보 접근

참조 요구사항: MP002, MP005, MP006, MP007, MP008

Physician (주치의)

설명: 환자의 건강을 전문적으로 관리하는 의료 전문가

상속: User 클래스 상속

속성 (Attributes)

속성명	타입	접근제어	설명
licenseNumber	String	private	의사 면허 번호
specialty	String	private	전문 분야 (신경과, 심장내과 등)

메서드 (Methods)

메서드명	반환타입	파라미터	설명
viewPatientPanel()	List	-	담당 환자 목록 조회
sortPatientsByRisk()	List	-	위험도 순으로 환자 정렬
writeManagementNote()	void	patient: Patient, note: String	관리 메모 작성

관계 (Relationships)

- 0..* Physician --- 0..* Patient: 여러 환자 진료
- 1 Physician --- 1 PatientPanel: 환자 패널 관리
- 1 Physician --- 0..* SharingPermission: 공유 권한 수신
- 1 Physician --- 0..* ManagementNote: 관리 메모 작성

책임 (Responsibilities)

- 담당 환자들의 건강 상태를 통합 모니터링
- 고위험 환자를 우선적으로 식별하고 관리
- 각 환자에 대한 전문적인 관리 메모 및 처방 기록
- 환자에게 의료 전문가 관점의 권고사항 제공

참조 요구사항: MP004, MP005, MP012, MP013, MP014, MP015, MP016, MP017

5. Administrator (시스템 관리자)

설명: 시스템 운영 및 콘텐츠를 관리하는 관리자

상속: User 클래스 상속

메서드 (Methods)

메서드명	반환타입	파라미터	설명
manageContent()	void	-	교육 자료 및 가이드 관리
configureAlertPolicy()	void	-	알림 정책 설정
manageUsers()	void	-	사용자 계정 관리

책임 (Responsibilities)

- 시스템 전반의 안정성 유지
- 교육 콘텐츠 및 가이드 업데이트
- 알림 정책 및 규칙 관리
- 사용자 계정 및 권한 관리

Gender (Enum)

설명: 사용자 성별을 나타내는 열거형

값 (Values)

- MALE : 남성
- FEMALE : 여성

1.3.2. Health Data 관련 클래스**HealthData (건강 데이터)**

설명: 환자가 입력하는 건강 지표 정보

속성 (Attributes)

속성명	타입	접근제어	설명
dataId	String	private	데이터 고유 ID
patientId	String	private	환자 ID
recordedAt	DateTime	private	기록 일시
bloodPressureSystolic	int	private	수축기 혈압 (mmHg)
bloodPressureDiastolic	int	private	이완기 혈압 (mmHg)
bloodSugar	double	private	혈당 수치 (mg/dL)

weight	double	private	체중 (kg)
height	double	private	신장 (cm)
bmi	double	private	체질량지수
smokingStatus	SmokingStatus	private	흡연 상태
alcoholConsumption	AlcoholLevel	private	음주 수준
exerciseFrequency	int	private	주간 운동 횟수

메서드 (Methods)

메서드명	반환타입	파라미터	설명
validate()	boolean	-	데이터 유효성 검증
calculateBMI()	double	-	BMI 자동 계산

관계 (Relationships)

- 1 Patient --- 0..* HealthData
- 1 HealthDataHistory o--- 0..* HealthData (집합 관계)
- AIEngine ..> HealthData (의존 관계)
- DataValidationService ..> HealthData (의존 관계)

책임 (Responsibilities)

- 환자의 건강 지표를 정확하게 저장
- 입력된 데이터의 유효성 검증
- BMI 등 파생 지표 자동 계산

유효성 규칙

- 혈압: 0-300 mmHg
- 혈당: 0-600 mg/dL
- BMI: weight(kg) / (height(m))²

참조 요구사항: HD001, HD002, HD005

SmokingStatus (Enum)

설명: 흡연 상태를 나타내는 열거형

값 (Values)

- NEVER : 비흡연자
- FORMER : 과거 흡연자
- CURRENT : 현재 흡연자

AlcoholLevel (Enum)

설명: 음주 수준을 나타내는 열거형

값 (Values)

- NONE: 음주 안 함
- OCCASIONAL: 가끔 음주
- MODERATE: 적당한 음주
- HEAVY: 과음

HealthDataHistory (건강 데이터 이력)

설명: 환자의 건강 데이터 시계열 이력 관리

속성 (Attributes)

속성명	타입	접근제어	설명
historyId	String	private	이력 고유 ID
patientId	String	private	환자 ID
dataRecords	List	private	건강 데이터 목록

메서드 (Methods)

메서드명	반환타입	파라미터	설명
addRecord()	void	data: HealthData	새 데이터 추가
getRecordsByPeriod()	List	start: Date, end: Date	기간별 데이터 조회
visualizeTimeSeries()	Chart	-	시계열 그래프 생성

관계 (Relationships)

- 1 Patient --- 1 HealthDataHistory
- 1 HealthDataHistory o--- 0..* HealthData

책임 (Responsibilities)

- 환자의 모든 건강 데이터를 시간순으로 관리

- 특정 기간의 데이터 조회 및 필터링
- 시계열 데이터 시각화

참조 요구사항: HD004, HD005, HD006, HD007

1.3.3. Risk Prediction 관련 클래스

RiskPrediction (위험도 예측)

설명: AI 모델이 생성한 뇌졸중 위험도 예측 결과

속성 (Attributes)

속성명	타입	접근제어	설명
predictionId	String	private	예측 고유 ID
patientId	String	private	환자 ID
predictedAt	DateTime	private	예측 수행 일시
riskProbability	double	private	위험 확률 (0.0 ~ 1.0)
riskLevel	RiskLevel	private	위험도 레벨
topRiskFactors	List	private	주요 위험 요인 상위 3개
featureImportance	Map	private	각 특성의 중요도
recommendations	List	private	개선 권고사항

메서드 (Methods)

메서드명	반환타입	파라미터	설명
classifyRiskLevel()	RiskLevel	-	확률 기반 레벨 분류
identifyTopRiskFactors()	List	n: int	상위 n개 위험 요인 추출

분류 규칙

- LOW : riskProbability < 0.3
- MEDIUM : $0.3 \leq \text{riskProbability} < 0.7$
- HIGH : $\text{riskProbability} \geq 0.7$

관계 (Relationships)

- 1 Patient --- 0..* RiskPrediction
- 1 RiskPrediction --- 1 RiskLevel

- AIEngine ..> RiskPrediction (생성)
- 1 PredictionHistory o--- 0..* RiskPrediction

책임 (Responsibilities):

- AI 예측 결과를 구조화하여 저장
- 위험도 레벨 자동 분류
- 주요 위험 요인 식별 및 권고사항 제공

참조 요구사항: RP001, RP003, RP004, RP005, RP006, RP007, RP008, RP009, RP010

RiskLevel (Enum)

설명: 위험도 레벨을 나타내는 열거형

값 (Values)

- LOW: 저위험 (< 30%)
- MEDIUM: 중위험 (30-70%)
- HIGH: 고위험 ($\geq 70\%$)

색상 코딩

- LOW: 녹색 (Green)
- MEDIUM: 노란색 (Yellow)
- HIGH: 빨간색 (Red)

AIEngine (AI 예측 엔진)

설명: XGBoost 기반 뇌졸중 위험 예측 머신러닝 모델

속성 (Attributes)

속성명	타입	접근제어	설명
modelVersion	String	private	모델 버전
modelPath	String	private	모델 파일 경로

메서드 (Methods)

메서드명	반환타입	파라미터	설명
loadModel()	void	-	XGBoost 모델 로드
preprocessData()	ProcessedData	data: HealthData	데이터 전처리

<code>predict()</code>	double	<code>data: ProcessedData</code>	위험도 예측 수행
<code>calculateFeatureImportance()</code>	Map	<code>data: ProcessedData</code>	특성 중요도 계산
<code>calculateSHAPValues()</code>	Map	<code>data: ProcessedData</code>	SHAP 값 계산

전처리 과정

1. 결측값 처리 (평균/중앙값 대체)
2. 범주형 변수 인코딩 (성별, 흡연 상태 등)
3. 수치형 변수 정규화
4. 모델 입력 형식 변환

관계 (Relationships)

- AIEngine ..> HealthData (사용)
- AIEngine ..> RiskPrediction (생성)
- AIEngine ..> WhatIfAnalysis (지원)
- AIEngine ..> RiskFactorAttribution (계산)

책임 (Responsibilities)

- 건강 데이터 기반 뇌졸중 위험도 예측
- 특성 중요도 및 SHAP 값 계산
- What-if 분석 시 시나리오별 예측 수행

참조 요구사항: RP002, RP003, RP009

PredictionHistory (예측 이력)

설명: 환자의 위험도 예측 이력 관리

속성 (Attributes)

속성명	타입	접근제어	설명
<code>patientId</code>	String	private	환자 ID
<code>predictions</code>	List	private	예측 결과 목록

메서드 (Methods)

메서드명	반환타입	파라미터	설명
<code>addPrediction()</code>	void	<code>prediction: RiskPrediction</code>	새 예측 추가

getRiskTrend()	TrendData	-	위험도 추세 분석
compareWithPrevious()	ComparisonResult	-	이전 예측과 비교

관계 (Relationships)

- 1 Patient --- 1 PredictionHistory
- 1 PredictionHistory o--- 0..* RiskPrediction

책임 (Responsibilities)

- 모든 예측 결과를 시간순으로 저장
- 위험도 변화 추세 분석
- 이전 예측과의 비교를 통한 개선/악화 감지

참조 요구사항: RP007, RP013, RP014, RP015

1.3.4. Analysis Features (차별화 기능)

WhatIfAnalysis (개선 시뮬레이션)

설명: 건강 지표 개선 시 예상 위험도를 시뮬레이션

속성 (Attributes)

속성명	타입	접근제어	설명
analysisId	String	private	분석 고유 ID
patientId	String	private	환자 ID
currentRisk	double	private	현재 위험도
scenarios	List	private	시나리오 목록

메서드 (Methods)

메서드명	반환타입	파라미터	설명
createScenario()	Scenario	changes: Map	새 시나리오 생성
compareScenarios()	ComparisonResult	-	시나리오 비교 분석
recommendBestScenario()	Scenario	-	최적 시나리오 추천

사용 예시

시나리오 1: 혈당을 120으로 낮춤 → 위험도 65% → 50% (15% 감소)

시나리오 2: 금연 + 운동 증가 → 위험도 65% → 45% (20% 감소) ★ 추천

관계 (Relationships)

- 1 Patient --- 0..* WhatIfAnalysis
- 1 WhatIfAnalysis o--- 1..* Scenario
- AIEngine ..> WhatIfAnalysis

책임 (Responsibilities)

- 가상의 건강 지표 개선 시나리오 생성
- 각 시나리오별 예상 위험도 계산
- 여러 시나리오 비교 및 최적 방안 제안

참조 요구사항: WA001, WA002, WA003, WA004

Scenario (시나리오)

설명: What-if 분석의 개별 시나리오

속성 (Attributes)

속성명	타입	접근제어	설명
scenarioid	String	private	시나리오 ID
name	String	private	시나리오 이름
modifiedData	HealthData	private	개선된 건강 데이터
projectedRisk	double	private	예상 위험도
riskReduction	double	private	위험도 감소량

메서드 (Methods)

메서드명	반환타입	파라미터	설명
calculateProjectedRisk()	double	-	예상 위험도 계산

관계 (Relationships)

- 1 WhatIfAnalysis o--- 1..* Scenario

책임 (Responsibilities)

- 특정 개선 시나리오의 데이터 및 결과 저장
- 현재 위험도 대비 감소량 계산

PercentileRanking (집단 내 순위 비교)

설명: 동일 조건 집단 내에서 사용자의 위치를 백분위로 표시

속성 (Attributes)

속성명	타입	접근제어	설명
rankingId	String	private	순위 정보 ID
patientId	String	private	환자 ID
cohortDefinition	CohortDefinition	private	비교 집단 정의
percentileRank	double	private	백분위 순위 (0-100)
totalInCohort	int	private	집단 내 총 인원

메서드 (Methods)

메서드명	반환타입	파라미터	설명
identifyCohort()	CohortDefinition	patient: Patient	비교 집단 식별
calculatePercentile()	double	-	백분위 계산
compareWithAverage()	ComparisonResult	-	집단 평균과 비교

사용 예시

"당신은 같은 연령대(50대) 남성 중 상위 30%의 위험도입니다"

"같은 조건(고혈압+당뇨) 환자 중 하위 20% → 관리를 잘하고 있습니다"

관계 (Relationships)

- 1 Patient --- 0..1 PercentileRanking
- 1 PercentileRanking --- 1 CohortDefinition

책임 (Responsibilities)

- 사용자와 유사한 조건의 집단 식별
- 집단 내 백분위 순위 계산
- 집단 평균 대비 위치 비교

참조 요구사항: PR001, PR002, PR003, PR004

CohortDefinition (집단 정의)

설명: 비교 집단의 조건 정의

속성 (Attributes)

속성명	타입	접근제어	설명
ageRange	Range	private	연령대 범위
gender	Gender	private	성별
conditions	List	private	질환 조건 (고혈압, 당뇨 등)

메서드 (Methods)

메서드명	반환타입	파라미터	설명
matchPatient()	boolean	patient: Patient	환자가 집단에 속하는지 확인

관계 (Relationships)

- 1 PercentileRanking --- 1 CohortDefinition

책임 (Responsibilities)

- 비교 집단의 조건을 명확히 정의
- 환자가 해당 집단에 속하는지 판별

TrendForecasting (추세 예측)

설명: 시계열 분석을 통한 미래 위험도 예측

속성 (Attributes)

속성명	타입	접근제어	설명
forecastId	String	private	예측 ID
patientId	String	private	환자 ID
historicalData	List	private	과거 데이터 (최소 3회)
forecastPeriod	int	private	예측 기간 (개월)
futureRiskPredictions	List	private	미래 예측 목록

메서드 (Methods)

메서드명	반환타입	파라미터	설명
analyzeTimeSeries()	TrendData	-	시계열 추세 분석
predictFutureRisk()	List	months: int	미래 위험도 예측 (1~6개월)
detectHighRiskEntry()	boolean	-	고위험군 진입 감지

사용 예시

"현재 추세대로라면 2개월 후 고위험군 진입 가능성 70%"

"최근 3개월 개선 추세 유지 시 6개월 후 저위험군 진입 예상"

관계 (Relationships)

- 1 Patient --- 0..* TrendForecasting
- 1 TrendForecasting o--- 0..* FuturePrediction

책임 (Responsibilities)

- 과거 데이터 추세 분석 (증가/감소/안정)
- 시계열 모델(ARIMA/Prophet)을 통한 미래 예측
- 고위험군 진입 조기 경고

참조 요구사항: TF001, TF002, TF003

FuturePrediction (미래 예측)

설명: 특정 미래 시점의 위험도 예측

속성 (Attributes)

속성명	타입	접근제어	설명
targetDate	Date	private	목표 날짜
projectedRisk	double	private	예상 위험도
confidenceInterval	Range	private	신뢰구간

관계 (Relationships)

- 1 TrendForecasting o--- 0..* FuturePrediction

책임 (Responsibilities)

- 특정 미래 시점의 위험도 저장
- 예측의 신뢰도 구간 제공

RiskFactorAttribution (위험 요인 기여도 분석)

설명: 각 건강 지표가 전체 위험도에 기여하는 정도를 분석

속성 (Attributes)

속성명	타입	접근제어	설명
attributionId	String	private	분석 ID
patientId	String	private	환자 ID
factorContributions	Map	private	요인별 기여도 (%)

메서드 (Methods)

메서드명	반환타입	파라미터	설명
calculateContributions()	Map	-	SHAP/Feature Importance 기반 계산
identifyTopFactors()	List	n: int	상위 n개 요인 추출
generatePriorityRecommendations()	List	-	우선 개선 권고

사용 예시

고혈압: 35%

흡연: 25%

혈당: 20%

나이: 15%

기타: 5%

→ "고혈압 개선을 최우선으로 권장합니다"

관계 (Relationships)

- 1 Patient --- 0..* RiskFactorAttribution
- AIEngine ..> RiskFactorAttribution

책임 (Responsibilities)

- SHAP 값 또는 Feature Importance 계산
- 위험 요인별 기여도 정량화
- 개선 우선순위 결정 지원

참조 요구사항: RF001, RF002, RF003, RF004

1.3.5. Collaborative Monitoring 관련 클래스

SharingPermission (공유 권한)

설명: 환자가 보호자/주치의에게 부여하는 데이터 공유 권한

속성 (Attributes)

속성명	타입	접근제어	설명
permissionId	String	private	권한 ID
patientId	String	private	환자 ID
sharedWithId	String	private	공유 대상 ID
sharedWithType	UserType	private	대상 유형 (Guardian/Physician)
permissionLevel	PermissionLevel	private	권한 레벨
grantedAt	DateTime	private	권한 부여 일시
isActive	boolean	private	활성화 여부

메서드 (Methods)

메서드명	반환타입	파라미터	설명
grant()	void	-	권한 부여
revoke()	void	-	권한 취소
updatePermissionLevel()	void	level: PermissionLevel	권한 레벨 변경

관계 (Relationships)

- 1 Patient --- 0..* SharingPermission
- 1 Guardian --- 0..* SharingPermission
- 1 Physician --- 0..* SharingPermission
- 1 SharingPermission --- 1 PermissionLevel

책임 (Responsibilities)

- 환자가 부여한 데이터 공유 권한 관리
- 권한 레벨에 따른 접근 제어
- 권한 부여/취소 이력 관리

참조 요구사항: MP001, MP002, MP003, MP004

UserType (Enum)

설명: 공유 대상 유형

값 (Values)

- GUARDIAN: 보호자
- PHYSICIAN: 주치의

PermissionLevel (Enum)

설명: 데이터 공유 권한 레벨

값 (Values)

- FULL_ACCESS: 모든 데이터 및 상세 정보 접근 가능
- SUMMARY_ONLY: 요약 정보만 접근 가능
- ALERT_ONLY: 고위험 알림만 수신

25. Notification (알림)

설명: 시스템에서 발송하는 알림

속성 (Attributes)

속성명	타입	접근제어	설명
notificationId	String	private	알림 ID
recipientId	String	private	수신자 ID
type	NotificationType	private	알림 유형
message	String	private	알림 메시지
sentAt	DateTime	private	발송 일시
isRead	boolean	private	읽음 여부

메서드 (Methods)

메서드명	반환타입	파라미터	설명
send()	void	-	알림 발송
markAsRead()	void	-	읽음 처리

관계 (Relationships)

- 1 User --- 0..* Notification
- NotificationService ...> Notification

책임 (Responsibilities)

- 알림 메시지 구조화 및 저장

- 발송 이력 관리
- 읽음/안 읽음 상태 추적

참조 요구사항: MP008, MP009, MP010, MP019, MP020

NotificationType (Enum)

설명: 알림 유형

값 (Values)

- HIGH_RISK_ALERT: 고위험 알림
- DATA_REMINDER: 데이터 입력 리마인더
- TREND_WARNING: 추세 경고
- GENERAL_INFO: 일반 정보

PatientPanel (환자 패널)

설명: 주치의가 담당 환자들을 관리하는 패널

속성 (Attributes)

속성명	타입	접근제어	설명
physicianId	String	private	주치의 ID
patients	List	private	담당 환자 목록

메서드 (Methods)

메서드명	반환타입	파라미터	설명
addPatient()	void	patient: Patient	환자 추가
removePatient()	void	patient: Patient	환자 제거
sortByRisk()	List	-	위험도 순 정렬
filterByRiskLevel()	List	level: RiskLevel	위험도 레벨로 필터링
searchPatient()	List	query: String	환자 검색

정렬 기준

1. 위험도 (High → Medium → Low)
2. 동일 위험도 내에서 최근 데이터 입력 시간 역순

관계 (Relationships)

- 1 Physician --- 1 PatientPanel
- 1 PatientPanel o--- 0..* Patient

책임 (Responsibilities)

- 주치의의 담당 환자 목록 관리
- 위험도 기반 우선순위 정렬
- 고위험 환자 특별 표시

참조 요구사항: MP012, MP013, MP014, MP015

ManagementNote (관리 메모)

설명: 주치의가 작성하는 환자 관리 메모

속성 (Attributes)

속성명	타입	접근제어	설명
noteId	String	private	메모 ID
physicianId	String	private	주치의 ID
patientId	String	private	환자 ID
content	String	private	메모 내용
createdAt	DateTime	private	작성 일시
nextAppointment	Date	private	다음 검진 예정일 (선택)

메서드 (Methods)

메서드명	반환타입	파라미터	설명
create()	void	-	메모 생성
update()	void	content: String	메모 수정

관계 (Relationships)

- 1 Physician --- 0..* ManagementNote
- 1 Patient --- 0..* ManagementNote

책임 (Responsibilities)

- 전문가의 관리 메모 및 처방 정보 저장
- 다음 검진 일정 관리

- 한자와의 공유 기능 지원

참조 요구사항: MP016, MP017, MP018

1.3.6. System Services

AuthenticationService (인증 서비스)

설명: 사용자 인증 및 계정 관리 서비스

메서드 (Methods)

메서드명	반환타입	파라미터	설명
authenticate()	User	email: String, password: String	로그인 인증
register()	boolean	user: User	회원가입
verifyPhysician()	boolean	license: String	주치의 면허 인증
resetPassword()	void	email: String	비밀번호 재설정

관계 (Relationships)

- AuthenticationService ..> User

책임 (Responsibilities)

- 사용자 로그인/로그아웃 처리
- 새 사용자 등록 및 계정 생성
- 주치의 계정의 의료 면허 검증
- 비밀번호 재설정 지원

DataValidationService (데이터 검증 서비스)

설명: 건강 데이터 유효성 검증 서비스

메서드 (Methods)

메서드명	반환타입	파라미터	설명
validateHealthData()	ValidationResult	data: HealthData	건강 데이터 검증
checkRequiredFields()	boolean	data: Object	필수 필드 확인
validateRange()	boolean	value: double, min: double, max: double	범위 검증

검증 규칙

- 필수 필드: 성별, 나이, 혈압, 혈당, BMI
- 혈압 범위: 0-300 mmHg
- 혈당 범위: 0-600 mg/dL
- BMI 일관성: weight / (height)² 계산 결과와 일치

관계 (Relationships)

- DataValidationService ..> HealthData

책임 (Responsibilities)

- 입력된 건강 데이터의 유효성 검증
- 필수 항목 누락 확인
- 데이터 타입 및 범위 검증

참조 요구사항: HD002, HD005

NotificationService (알림 서비스)

설명: 다양한 알림을 발송하고 관리하는 서비스

속성 (Attributes)

속성명	타입	접근제어	설명
alertPolicy	AlertPolicy	private	알림 정책

메서드 (Methods)

메서드명	반환타입	파라미터	설명
sendHighRiskAlert()	void	patient: Patient, prediction: RiskPrediction	고위험 알림 전송
sendDataReminder()	void	patient: Patient	데이터 입력 리마인더
configureAlertPolicy()	void	policy: AlertPolicy	알림 정책 설정
getNotificationHistory()	List	userId: String	알림 이력 조회

관계 (Relationships)

- NotificationService ..> Notification
- NotificationService ..> Patient
- NotificationService ..> Guardian
- NotificationService ..> Physician

- 1 NotificationService --- 1 AlertPolicy

책임 (Responsibilities)

- 고위험 상황 발생 시 즉시 알림 전송
- 정기적인 데이터 입력 리마인더 발송
- 알림 정책 및 빈도 관리
- 다중 채널 알림 지원 (푸시, SMS, 이메일)

참조 요구사항: HD003, HD012, MP008, MP009, MP010, MP011

AlertPolicy (알림 정책)

설명: 알림 발송 정책 설정

속성 (Attributes)

속성명	타입	접근제어	설명
highRiskThreshold	double	private	고위험 판정 기준 (기본: 0.7)
alertFrequency	String	private	알림 빈도 (즉시/일일 요약 등)
enabledChannels	List	private	활성화된 알림 채널

관계 (Relationships)

- 1 NotificationService --- 1 AlertPolicy

책임 (Responsibilities)

- 고위험 판정 기준 설정
- 알림 빈도 및 방식 정의
- 알림 채널 관리 (푸시/SMS/이메일)

ReportGenerator (보고서 생성 서비스)

설명: 개인 및 주치의용 보고서 생성 서비스

메서드 (Methods)

메서드명	반환타입	파라미터	설명
generatePersonalReport()	Report	patient: Patient	개인 건강 리포트 생성
generatePhysicianReport()	Report	physician: Physician	주치의 환자 관리 리포트
exportData()	File	patient: Patient, format: String	데이터 내보내기 (CSV/PDF)

책임 (Responsibilities)

- 환자의 건강 상태 종합 보고서 생성
- 주치의의 환자 관리 현황 리포트
- 데이터 내보내기 (CSV, PDF 등)

1.4. Class Diagram Summary

주요 패키지 구조

Java 언어 기준으로 표현한 패키지 구조입니다. 실제로는 Python으로 구현 예정입니다.

```

com.stroke.prevention
  └── user
    ├── User (abstract)
    ├── Patient
    ├── Guardian
    ├── Physician
    └── Administrator
  └── healthdata
    ├── HealthData
    ├── HealthDataHistory
    ├── SmokingStatus (enum)
    └── AlcoholLevel (enum)
  └── prediction
    ├── RiskPrediction
    ├── RiskLevel (enum)
    ├── AIEngine
    └── PredictionHistory
  └── analysis
    ├── WhatIfAnalysis
    ├── Scenario
    ├── PercentileRanking
    ├── CohortDefinition
    ├── TrendForecasting
    ├── FuturePrediction
    └── RiskFactorAttribution
  └── collaboration
    ├── SharingPermission
    ├── PermissionLevel (enum)
    ├── UserType (enum)
    ├── Notification
    ├── NotificationType (enum)
    ├── PatientPanel
    └── ManagementNote
  └── service
    ├── AuthenticationService
    ├── DataValidationService
    ├── NotificationService
    ├── AlertPolicy
    └── ReportGenerator

```

핵심 관계 매트릭스

From	Relationship	To	Multiplicity
Patient	records	HealthData	1 to 0..*
Patient	has	HealthDataHistory	1 to 1
Patient	receives	RiskPrediction	1 to 0..*
AIEngine	creates	RiskPrediction	- (dependency)
Patient	performs	WhatIfAnalysis	1 to 0..*
WhatIfAnalysis	contains	Scenario	1 to 1..*
Patient	monitored by	Guardian	0..* to 0..*
Patient	treated by	Physician	0..* to 0..*
Physician	manages	PatientPanel	1 to 1
PatientPanel	monitors	Patient	1 to 0..*

설계 패턴 적용

- Strategy Pattern:** AIEngine - 다양한 예측 알고리즘 (XGBoost, 시계열 모델 등)
- Observer Pattern:** NotificationService - 위험도 변화 시 자동 알림
- Factory Pattern:** User 계정 생성 (Patient, Guardian, Physician 등)
- Composite Pattern:** HealthDataHistory - 여러 HealthData 집합 관리
- Singleton Pattern:** AuthenticationService, NotificationService

데이터 흐름 요약

[Patient] → [HealthData] → [AIEngine] → [RiskPrediction] → [NotificationService]

↓

[WhatIfAnalysis]
 [PercentileRanking]
 [TrendForecasting]
 [RiskFactorAttribution]

1.5. 요구사항 추적 매트릭스

클래스명	관련 요구사항
HealthData	HD001, HD002, HD003, HD005
HealthDataHistory	HD004, HD005, HD006, HD007
RiskPrediction	RP001~RP015
AIEngine	RP002, RP003, RP009
WhatIfAnalysis	WA001~WA004
PercentileRanking	PR001~PR004
TrendForecasting	TF001~TF003
RiskFactorAttribution	RF001~RF004
SharingPermission	MP001~MP004
PatientPanel	MP012~MP015
ManagementNote	MP016~MP018
NotificationService	MP008~MP011, HD003, HD012

Phase 2 – Architecture Design

실제 구현을 위한 프론트엔드/백엔드 각 파트에 대한 아키텍처 설계입니다.

2.1. Frontend Architecture (Flutter)

2.1.1. Frontend Package Diagram

[Appendix B. Frontend Package Diagram](#)을 참조해주세요.

2.1.2. 아키텍처 설명

Feature-First Architecture + MVVM Pattern 기반의 Clean Architecture 적용 예정



그림 1 프론트엔드(Flutter) 코드베이스 구조

Feature-First 아키텍처

시스템의 기능(Feature)을 중심으로 폴더를 나누어 독립적으로 관리함.

클린 아키텍처

각 피처는 Data, Domain, Presentation의 독립적인 3계층 구조를 가지도록 함.

[레이어 및 구성요소]

- **Data Layer:** 외부 데이터(API, DB, Cache) 접근
 - **Data Sources:** API/DB/Storage 등에 대한 직접 접근
 - **Models:** API 요청/응답에 대한 DTO(Data Transfer Objects)를 모델링
 - **Repository Implementation:** Domain 레이어의 Repository에서 정의된 인터페이스를 implement
- **Domain Layer:** 비즈니스 로직 및 데이터 처리
 - **Repository Interface:** Data 레이어 추상화
 - **Entities:** 비즈니스 데이터 엔티티 모델링
 - **Use Cases:** 실제 비즈니스 로직을 처리
- **Presentation Layer:** UI 렌더링 및 사용자 인터랙션 처리
 - **View:** UI 렌더링
 - **ViewModel:** MVVM 패턴에 따라 비즈니스 로직 및 상태 관리
 - **State:** 뷰에 대한 상태를 정의

[핵심 원칙]

1. **의존성 규칙** – 의존성은 항상 내부 레이어(Domain)을 향해야 함. 외부 레이어는 내부 레이어를 알 수 있지만, 내부 레이어는 외부 레이어를 알 수 없어야 함. 즉, Data Layer는 Domain의 인터페이스를 구현하고, Presentation Layer는 Domain의 인터페이스만 사용해야 함
2. **관심사의 분리** – 각 레이어 및 클래스는 하나의 명확한 책임만을 가져야 함
3. **의존성 역전 원칙** – Repository 패턴을 통해 데이터 접근을 추상화하여 도메인 레이어는 구체적인 구현을 모르게 하고 실제 구현체는 런타임에서 의존성 주입(Dependency Injection)함.

MVVM 패턴

Presentation 레이어에서, Model, View, ViewModel 세 부분으로 나눈 패턴을 적용함.

[구성 요소]

- **Model:** 데이터 구조를 클래스로 모델링
- **View:** 비즈니스 로직 없이 UI만 담당
- **ViewModel:** 비즈니스 로직, 상태 관리

[핵심 규칙]

- 단방향 데이터 흐름: View는 ViewModel의 상태를 구독하기만 하고, 직접 수정하지 않음. -> 모든 상태 변경은 ViewModel을 통해서만 발생
- View는 어떠한 비즈니스 로직도 포함하지 않음. API 호출/데이터 변환/유효성 검증 등은 ViewModel 및 Repository에서 차리해야 함. 또한 ViewModel에도 UI 의존성이 존재하면 안 됨

2.1.3. 아키텍처 선정 이유

전반적으로 Flutter 개발 생태계에서 해당 패턴을 채택한 프로젝트가 많아 레퍼런스가 풍부하다는 점, 또한 Flutter 프레임워크 측에서 공식적으로 해당 패턴을 모범 개발 패턴으로서 소개하고 있으며 이 패턴을 기준으로 개발자 문서에서 설명하고 있으므로 해당 패턴이 사실상 Flutter 개발의 검증된 표준이라고 볼 수 있음. 따라서 이 패턴을 채택하여 효율적인 개발을 진행할 수 있도록 하고자 함.

Feature-First 아키텍처

- 새 기능 추가 시 기존 코드에 영향을 최소화할 수 있고, 기능별로 독립적인 확장이 가능하여 확장성이 높음
- 한 기능과 관련된 코드가 한 곳에 모여 있으므로 코드 관리가 편리하고 코드 간 참조 관계를 이해하기 쉬움
- 팀원이 서로 다른 피처에서 동시에 작업할 수 있으며, merge 시 conflict를 감소시킬 수 있음
- 버그 수정이나 변경이 있을 시 영향 범위를 명확히 파악할 수 있으며, 한 피처의 변경이 다른 피처에 영향을 주지 않음
- 피처 단위로 독립적인 테스팅이 가능하며, 모킹 구현도 간단함

클린 아키텍처

- 테스트 용이성** – 각 레이어를 독립적으로 테스트할 수 있으며, 인터페이스 추상화를 통한 의존성 역전 및 런타임 의존성 주입(DI) 과정 덕분에 실제 구현체 대신 외부 의존성을 모킹하여 주입하는 방식으로 비즈니스 로직만 검증할 수 있으며, UI나 데이터 소스 없이도 핵심 로직만을 테스트할 수 있음
- 확장성 및 유지보수성** – 한 레이어의 변경이 다른 레이어에 발생시키는 영향을 최소화할 수 있으므로 확장성이 높고, 엄격하게 계층화된 일관적인 코드베이스 구조를 확립하여 효율적인 팀원 간 협업이 가능해지며 유지보수도 용이함

MVVM 패턴

- 테스트 용이성** – 클린 아키텍처의 특성과 마찬가지로, ViewModel은 UI에 독립적이므로 View 로직 없이 비즈니스 로직만 격리하여 테스트가 가능함
- 명확한 관심사 분리** – 클린 아키텍처와 마찬가지로, Presentation 레이어를 구성하는 각 레이어(View, ViewModel, Model)의 책임이 명확하게 구분되어 코드 가독성과 유지보수성이 향상됨
- 반응형 UI 구현 용이** – ViewModel 상태 변경 시 View가 자동으로 갱신되므로 사용자 상호작용에 따른 UI 변화 구현이 간편함

- **상태 관리/예외 처리 단순화** – 복잡한 UI 상태를 ViewModel에서 체계적으로 관리할 수 있으며, 예외 처리 또한 ViewModel에서 중앙화하여 관리할 수 있음.

2.2. Backend Architecture (FastAPI)

2.2.1. Backend Package Diagram

Appendix B. Backend Package Diagram을 참조해주세요.

2.2.2. 아키텍처 설명

FastAPI 개발 표준인 Layered Architecture + Repository Pattern을 채택함.



그림 2 백엔드(FastAPI) 코드베이스 구조

계층별 역할

- **API Layer (Router)**: HTTP 요청 라우팅 및 검증, 응답 직렬화
- **Service Layer**: 비즈니스 로직 구현 및 트랜잭션 관리
- **Repository Layer**: DB CRUD 작업 캡슐화

2.2.3. 아키텍처 선정 이유

- 명확한 관심사 분리 – 각 계층의 역할이 명확하여 유지보수성 향상
- 데이터 접근 로직 추상화로 테스트 과정 용이 및 유연성 확보
- FastAPI의 DI 시스템을 활용하여 결합도 감소 가능

2.3. 내부 시스템 통합

2.3.1. 통신 방식 설계

- RESTful API 방식을 통한 Frontend-Backend 통신
- JSON 기반의 데이터 교환 (이를 위한 프론트, 백 단에서의 직렬화/역직렬화 로직 구현 필요)

2.3.2. API 문서화 및 테스팅

- FastAPI의 OpenAPI Specification 자동 생성 기능 및 Swagger UI와의 연동을 통해 API 규격 명확화

2.3.3. DB 통합

- DB는 PostgreSQL 사용 – 복잡한 데이터 및 시계열 데이터 처리 위함

2.3.4. AI 및 통계 분석 모듈 통합

개요

뇌졸중 위험도 예측을 위한 머신러닝 모델을 FastAPI 백엔드에 통합하여 실시간 위험도 분석 기능을 제공합니다. XGBoost 기반 분류 모델을 사용하며, 불균형 데이터 처리 및 특성 중요도 분석 기능을 포함하도록 미리 구현 해 둔 상태입니다.

위험도 예측 모델 아키텍처

- 알고리즘: XGBoost
- 클래스 불균형 처리: SMOTE (Synthetic Minority Over-sampling Technique)
- 하이퍼파라미터 최적화: GridSearchCV
- 평가 지표: Recall, PR-AUC

모델 로딩 전략

- FastAPI 백엔드 애플리케이션 시작 시 모델을 메모리에 로드 (Singleton 패턴)
- joblib을 사용한 모델 직렬화/역직렬화 적용

2.4. 외부 시스템 연계

2.4.1. Firebase Authentication

개요

Firebase Authentication을 사용하여 사용자 인증 및 계정 관리를 처리합니다.

통합 아키텍처

Frontend (Flutter)

- firebase_auth 패키지를 통한 인증 처리
- 이메일/비밀번호 기반 또는 소셜 로그인 기반 인증 지원
- 인증 토큰(ID Token) 자동 관리

Backend (FastAPI)

- Firebase Admin SDK를 통한 토큰 검증
- 커스텀 클레임을 활용한 역할 기반 접근 제어
- 사용자 역할(Patient, Guardian, Physician, Administrator) 관리

인증 흐름

[Flutter App] → Firebase Auth → [ID Token] → [FastAPI Backend] → Token 검증 → 사용자 정보 반환

1. 회원가입

- Flutter에서 Firebase Auth로 계정 생성
- 백엔드로 추가 사용자 정보(나이, 성별, 역할 등) 전송
- PostgreSQL에 사용자 프로필 저장
- 역할에 따른 Custom Claims 설정

2. 로그인

- Flutter에서 Firebase Auth 로그인
- ID Token 획득 및 로컬 저장
- 이후 모든 API 요청 시 Authorization Header에 포함

3. 토큰 검증

- FastAPI의 get_current_user() dependency에서 토큰 검증

- Firebase Admin SDK를 통한 서명 검증
- 토큰에서 uid 추출 후 DB 사용자 정보 조회

2.4.2. Firebase Cloud Messaging (FCM)

개요

FCM을 활용하여 고위험 알림, 데이터 입력 리마인더, 추세 경고 등을 실시간으로 푸시 알림으로 전송합니다.

통합 아키텍처

Frontend (Flutter)

- firebase_messaging 패키지를 통한 FCM 설정
- FCM Token 획득 및 백엔드로 전송
- 포그라운드/백그라운드 알림 처리
- 알림 탭 시 해당 화면으로 내비게이션

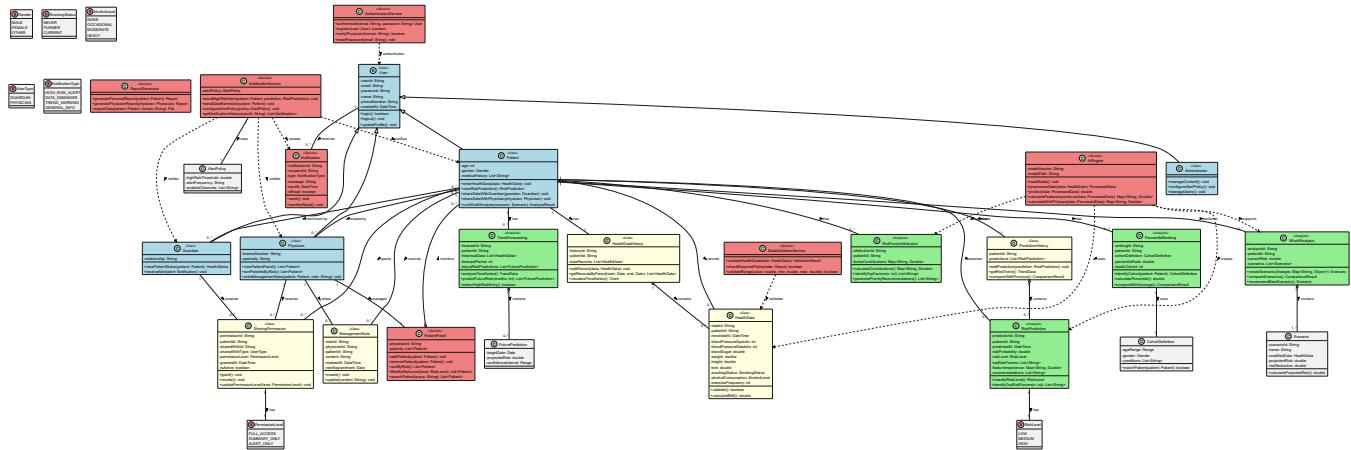
Backend (FastAPI)

- Firebase Admin SDK를 통한 알림 전송
- 사용자별 FCM Token 관리 (PostgreSQL 저장)
- NotificationService를 통한 중앙화된 알림 발송

Appendix A. Class Diagram & Sequence Diagram

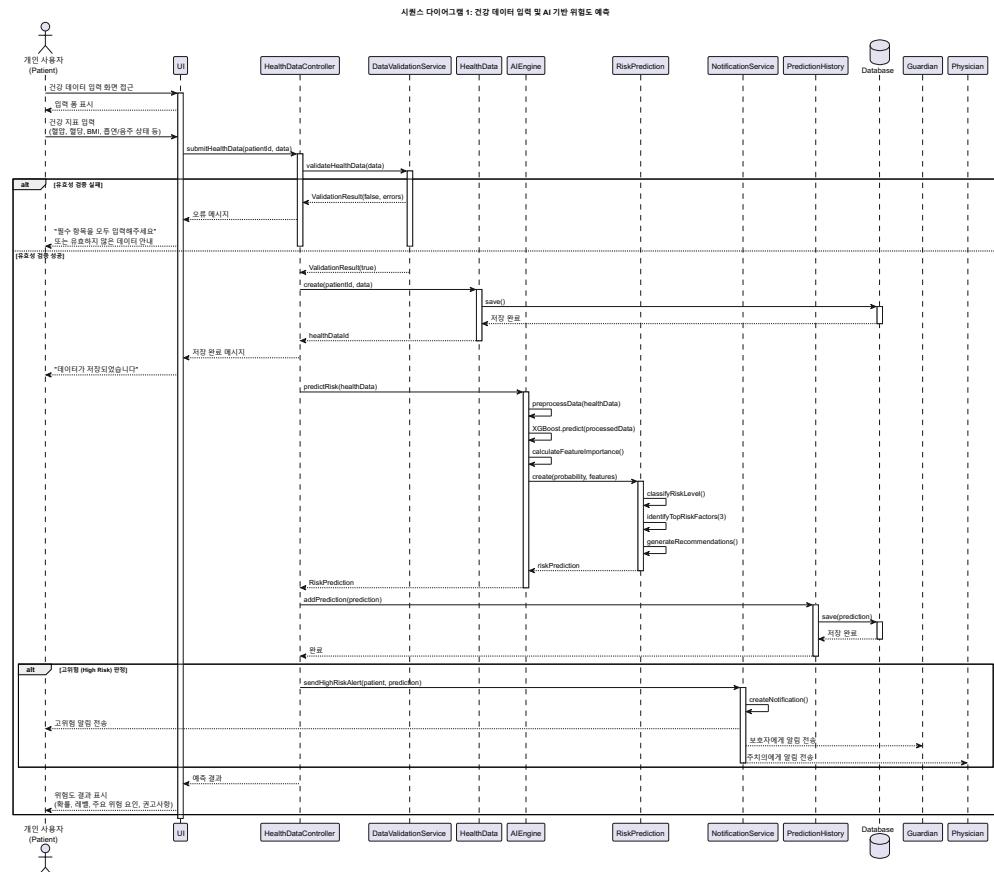
[Class Diagram]

본 팀에서 구상한 시스템을 기반으로 설계한 클래스 다이어그램입니다.



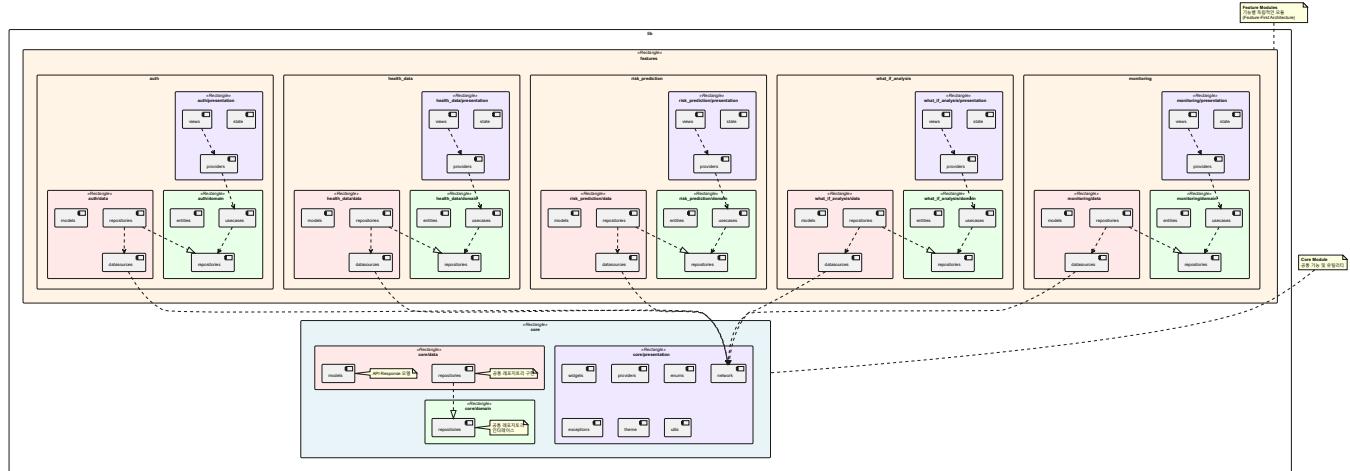
[Sequence Diagram]

시스템의 핵심이 되는 기능인 “건강 데이터 입력 및 AI 기반 위험도 예측”에 대한 시퀀스 다이어그램입니다.

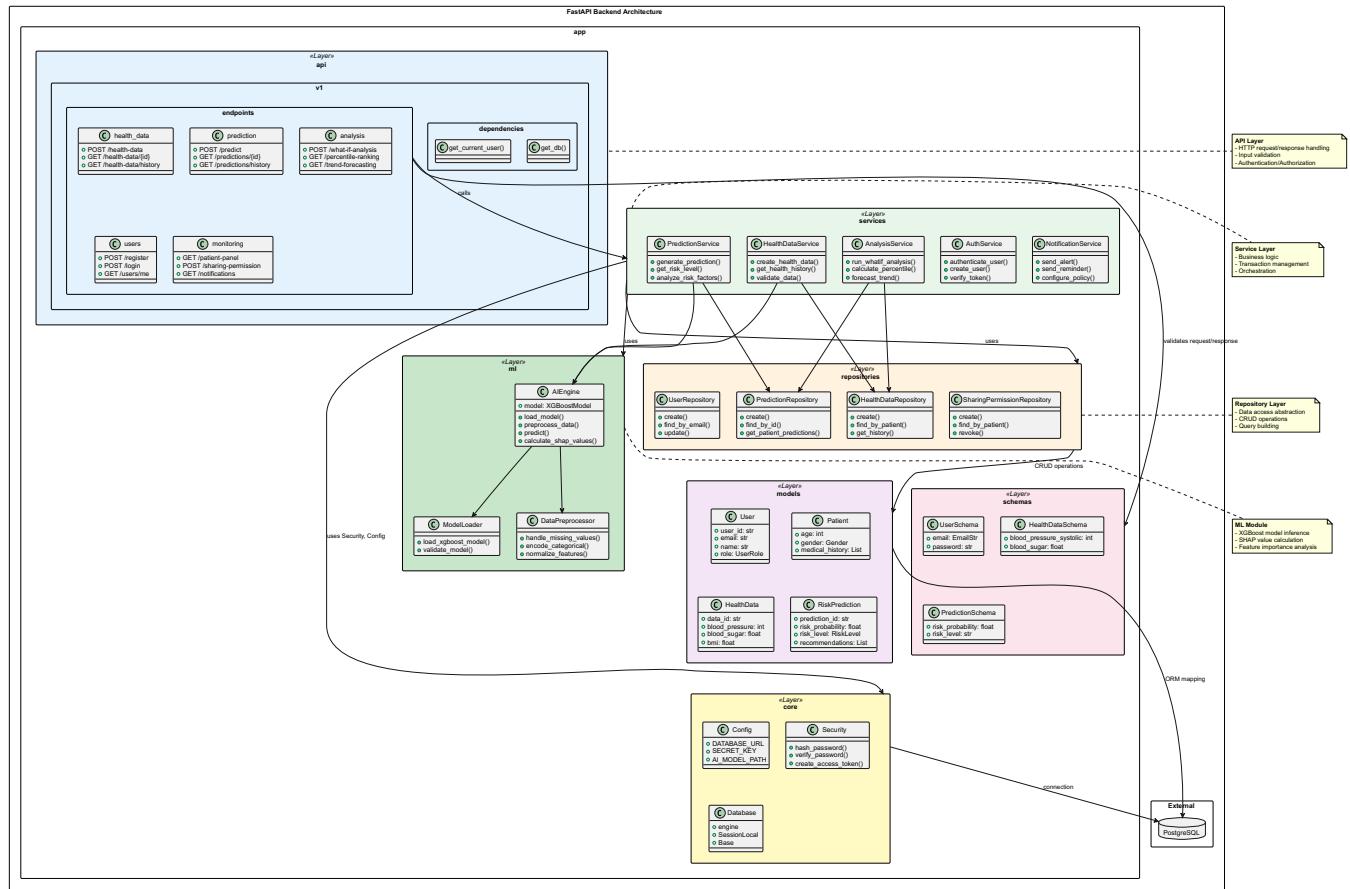


Appendix B. Package Diagram

[Frontend Package Diagram]



[Backend Package Diagram]



Appendix C. 기타 디아이어그램

[Frontend-Backend 통신 구조 디아이어그램]

프론트엔드와 백엔드 간 통합을 위해 나타낸 디아이어그램입니다.

