싸멘틀(SSAmantle) 프로젝트 종합 문서

문서 정보

- **프로젝트명**: 싸멘틀 (SSAmantle)
- **버전**: 1.0
- **최종 수정일**: 2025-12-24
- **작성자**: Development Team

목차

---

1. 요구사항 명세

1.1 프로젝트 개요

1.1.1 프로젝트 목적

싸멘틀(SSAmantle)은 **SSAFY(싸피) + Semantle(의미 기반 단어 추론 게임)**을 결합한 단어 추론 게임 플랫폼입니다. 사용자는 매일 제시되는 정답 단어를 의미 유사도를 기반으로 추론하여 맞추는 게임을 즐길 수 있습니다.

1.1.2 핵심 가치

- **학습성**: 단어의 의미적 관계를 이해하며 어휘력 향상
- **경쟁성**: 일일 리더보드를 통한 실력 경쟁
- **지속성**: 연속 풀이 시스템(스트릭)을 통한 꾸준한 참여 유도
- **성취감**: 업적 시스템을 통한 자기 발전 추적

1.2 주요 기능 요구사항

1.2.1 사용자 관리

- **회원가입**: 이메일, 비밀번호, 닉네임으로 계정 생성
- **로그인**: JWT 토큰 기반 인증
- **내 정보 조회**: 사용자 기본 정보 확인
- **정보 수정**: 비밀번호/닉네임 변경
- **게임 통계 조회**: 총 플레이 횟수, 승률, 평균 시도 횟수 등

1.2.2 게임 플레이

- **단어 추측 제출**: 단어 입력 → 유사도 및 순위 확인
- **게임 포기**: 정답 공개 및 상위 유사 단어 확인
- **게임 상태 조회**: NOT_STARTED, IN_PROGRESS, SOLVED, GAVE_UP
- **정답 이력 조회**: 오늘/어제 정답 및 상위 100개 단어

1.2.3 리더보드

- **순위 조회**: 상위 50명 + 본인 순위
- **순위 결정 기준**: 시도 횟수 적을수록, 같으면 빨리 푼 사람 우선

1.2.4 업적 시스템

- **연속 풀이 업적**: 3일/7일/30일 연속
- **총 문제 업적**: 10개/50개/100개 해결
- **자동 부여**: 정답 맞출 때마다 조건 체크

1.2.5 일일 배치 작업

- **유지보수 모드**: 매일 자정 배치 작업 중 API 차단
- **최고 순위 갱신**: 전날 Top 50 사용자의 bestRank 업데이트
- **연속 풀이 초기화**: 오늘 풀지 않은 사용자의 nowCont = 0

1.3 비기능 요구사항

1.3.1 성능

- API 응답 시간: 평균 200ms 이하
- 동시 접속 사용자: 1,000명 이상
- Redis 캐시 히트율: 90% 이상

1.3.2 보안

- JWT 기반 인증 (Access Token: 1시간, Refresh Token: 7일)
- 비밀번호 BCrypt 암호화
- HTTPS 통신 강제

1.3.3 가용성

- 시스템 가동률(Uptime): 99.5% 이상
- 데이터베이스 일일 백업

1.4 기술 스택

분류	기술
언어	Java 17
프레임워크	Spring Boot 3.x

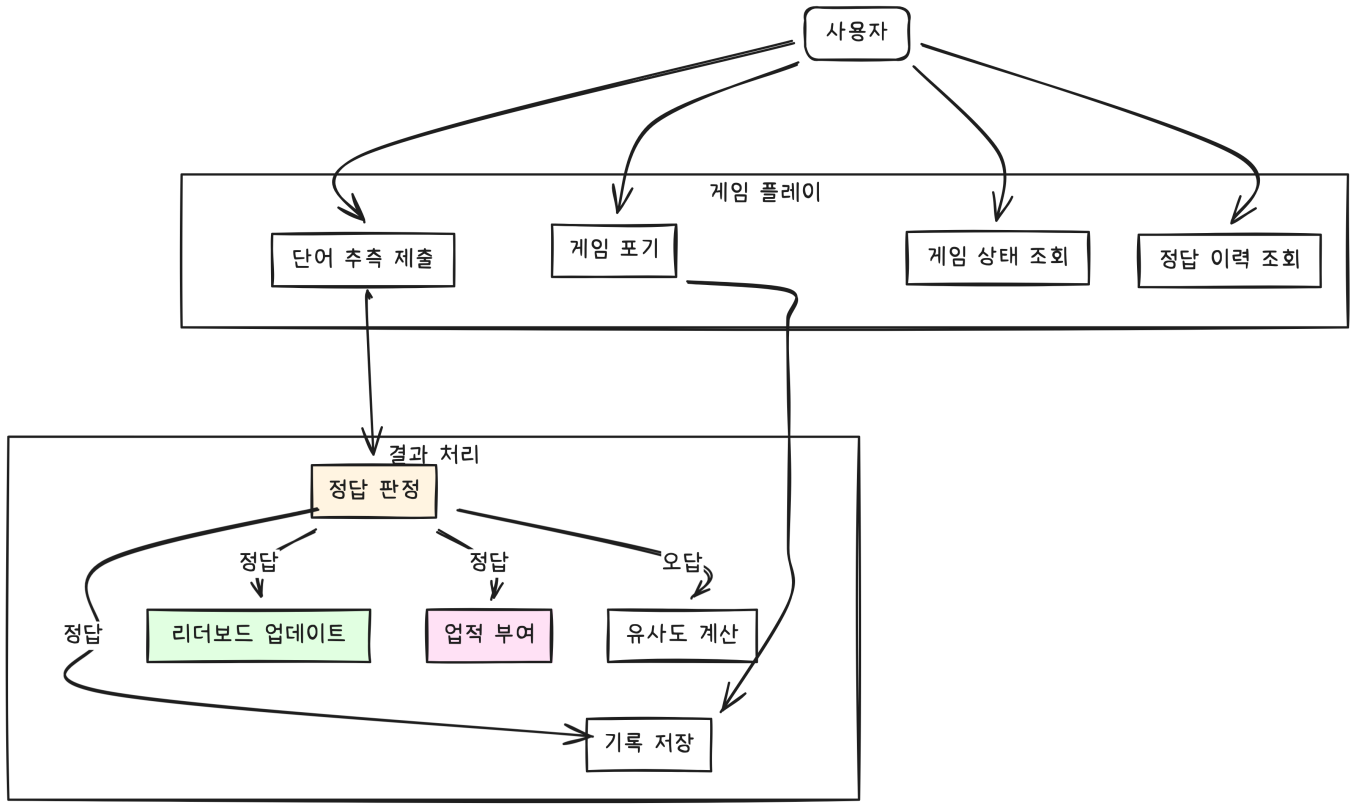
분류	기술
빌드 도구	Gradle (멀티모듈)
데이터베이스	MySQL 8.0
캐시	Redis 6.0+
ORM	MyBatis 3.x
보안	Spring Security, JWT
아키텍처	헥사고날 아키텍처, DDD

---

2. Use Case 다이어그램

2.1 전체 시스템 Use Case

싸멘틀 시스템의 전체 Use Case는 다음과 같습니다:



2.2 주요 액터

액터	설명
일반 사용자	게임 플레이, 회원 관리, 리더보드 조회 등
관리자	일일 배치 작업 실행, 시스템 관리
스케줄러	매일 자정 자동 배치 작업 트리거

2.3 Use Case 목록

인증 및 사용자 관리

- UC-001: 회원가입
- UC-002: 로그인
- UC-003: 토큰 갱신
- UC-004: 내 정보 조회
- UC-005: 사용자 정보 수정
- UC-006: 게임 통계 조회

게임 플레이

- UC-007: 단어 추측 제출
- UC-008: 게임 포기
- UC-009: 게임 상태 조회
- UC-010: 오늘 정답 이력 조회
- UC-011: 어제 정답 이력 조회

리더보드 및 업적

- UC-012: 리더보드 조회
- UC-013: 내 업적 조회

시스템 관리

- UC-014: 일일 배치 실행

2.4 Use Case 다이어그램 (상세)

2.4.1 전체 시스템 Use Case

```
graph TB
    subgraph "싸멘틀 시스템"
        UC1[회원가입]
        UC2[로그인]
        UC3[토큰 갱신]
        UC4[내 정보 조회]
        UC5[사용자 정보 수정]
        UC6[게임 통계 조회]

        UC7[단어 추측 제출]
        UC8[게임 포기]
        UC9[게임 상태 조회]
        UC10[오늘 정답 이력 조회]
        UC11[어제 정답 이력 조회]

        UC12[리더보드 조회]
        UC13[내 업적 조회]
        UC14[일일 배치 실행]
    end
```

```
    User([일반 사용자])
    Admin([관리자])
    System([스케줄러])

    User --> UC1
    User --> UC2
    User --> UC3
    User --> UC4
    User --> UC5
    User --> UC6
    User --> UC7
    User --> UC8
    User --> UC9
    User --> UC10
    User --> UC11
    User --> UC12
    User --> UC13

    Admin --> UC14
    System -.-> UC14

    UC7 -.includes.-> UC13

    style User fill:#e1f5ff
    style Admin fill:#ffe1e1
    style System fill:#f0f0f0
```

2.4.2 게임 플레이 Use Case (상세)

```
graph TB
    User([사용자])

    subgraph "게임 플레이"
        UC1[단어 추측 제출]
        UC2[게임 포기]
        UC3[게임 상태 조회]
        UC4[정답 이력 조회]
    end

    subgraph "결과 처리"
        UC5[정답 판정]
        UC6[유사도 계산]
        UC7[기록 저장]
        UC8[리더보드 업데이트]
        UC9[업적 부여]
    end

    User --> UC1
    User --> UC2
    User --> UC3
    User --> UC4
```

```
    UC1 --> UC5
    UC5 -->|정답| UC7
    UC5 -->|정답| UC8
    UC5 -->|정답| UC9
    UC5 -->|오답| UC6
    UC6 --> UC7
    UC2 --> UC7

    style User fill:#e1f5ff
    style UC5 fill:#fff4e1
    style UC8 fill:#e1ffe1
    style UC9 fill:#ffe1f5
```

2.5 주요 시퀀스 다이어그램

2.5.1 로그인 시퀀스

```
sequenceDiagram
    actor User as 사용자
    participant API as AuthController
    participant UC as SignInUseCase
    participant Service as SignInService
    participant Port as LoadUserByEmailPort
    participant Adapter as UserPersistenceAdapter
    participant DB as MySQL
    participant JWT as JwtProvider

    User->>API: POST /api/v1/auth/sign-in
{email, password}
    API->>UC: signIn(command)
    UC->>Service: execute(command)

    Service->>Port: loadByEmail(email)
    Port->>Adapter: loadByEmail(email)
    Adapter->>DB: SELECT * FROM users WHERE email=?
    DB-->>Adapter: UserEntity
    Adapter-->>Port: User
    Port-->>Service: User

    Service->>Service: 비밀번호 검증

    Service->>JWT: generateAccessToken(userId, email, role)
    JWT-->>Service: accessToken

    Service->>JWT: generateRefreshToken(userId)
    JWT-->>Service: refreshToken

    Service-->>UC: SignInResponse
    UC-->>API: SignInResponse
```

```
    API-->>User: 200 OK
{userId, email, nickname, accessToken, refreshToken}
```

2.5.2 단어 추측 제출 시퀀스 (정답인 경우)

```
sequenceDiagram
    actor User as 사용자
    participant API as GameController
    participant UC as SubmitGuessUseCase
    participant Service as SubmitGuessService
    participant Redis as GameRedisAdapter
    participant InferenceAPI as InferenceSimilarityAdapter
    participant RecordPort as SaveRecordPort
    participant UserPort as LoadUserPort
    participant LeaderboardPort as SaveLeaderboardPort
    participant AchievementUC as CheckAndGrantAchievementsUseCase
    participant DB as MySQL

    User->>API: POST /api/v1/games/guess
{word: "사과", failCount: 3}
    API->>UC: submitGuess(command)
    UC->>Service: execute(command)

    Service->>Redis: loadAnswerFromRedis(date)
    Redis-->>Service: "사과"

    Service->>Service: 정답 여부 판정
("사과" == "사과")

    alt 정답인 경우
        Service->>RecordPort: save(record)
failCount=3, solvedAt=now
        RecordPort->>DB: UPDATE records SET fail_count=3, solved_at=now

        Service->>UserPort: loadById(userId)
        UserPort->>DB: SELECT * FROM users WHERE id=?
        DB-->>UserPort: User
        UserPort-->>Service: User

        Service->>Service: user.solveProblem()
todaySolve=true, nowCont++

        Service->>UserPort: updateUser(user)
        UserPort->>DB: UPDATE users SET today_solve=true,
now_cont=nowCont+1

        Service->>LeaderboardPort: save(userId, score)
        LeaderboardPort->>Redis: ZADD ssamantle:leaderboard:2025-12-24
userId score

        Service->>AchievementUC: checkAndGrant(userId)
```

```
        AchievementUC-->>Service: newAchievements[]
    end

    Service-->>UC: SubmitGuessResponse
{isCorrect: true, answer: "사과", newAchievements}
    UC-->>API: SubmitGuessResponse
    API-->>User: 200 OK
{isCorrect: true, answer: "사과", failCount: 3, newAchievements}
```

2.5.3 단어 추측 제출 시퀀스 (오답인 경우)

```
sequenceDiagram
    actor User as 사용자
    participant API as GameController
    participant Service as SubmitGuessService
    participant Redis as GameRedisAdapter
    participant InferenceAPI as InferenceSimilarityAdapter
    participant RecordPort as SaveRecordPort
    participant DB as MySQL

    User->>API: POST /api/v1/games/guess
{word: "바나나", failCount: 1}
    API->>Service: execute(command)

    Service->>Redis: loadAnswerFromRedis(date)
    Redis-->>Service: "사과"

    Service->>Service: 정답 여부 판정
("바나나" != "사과")

    alt 오답인 경우
        Service->>Redis: loadWordFromTop1000("바나나", date)

        alt Top 1000에 있는 경우
            Redis-->>Service: WordSimilarity(85.42, 123)
        else Top 1000에 없는 경우
            Service->>InferenceAPI: calculateSimilarity("사과", "바나나")
            InferenceAPI-->>Service: SimilarityResponse(0.8542, 123)
            Service->>Service: convert to 0~100 scale
        end

        Service->>RecordPort: update(record)
failCount++
        RecordPort->>DB: UPDATE records SET fail_count=fail_count+1
    end

    Service-->>API: SubmitGuessResponse
{isCorrect: false, similarity: 85.42, rank: 123}
    API-->>User: 200 OK
{isCorrect: false, word: "바나나", similarity: 85.42, rank: 123}
```

2.5.4 리더보드 조회 시퀀스

```
sequenceDiagram
    actor User as 사용자
    participant API as LeaderboardController
    participant Service as GetLeaderboardService
    participant LeaderboardPort as LoadLeaderboardPort
    participant Redis as LeaderboardRedisAdapter
    participant UserPort as LoadUsersByIdsPort
    participant DB as MySQL

    User->>API: GET /api/v1/leaderboard?date=2025-12-24
    API->>Service: execute(command)

    Service->>LeaderboardPort: loadTopRankers(date, 50)
    LeaderboardPort->>Redis: ZRANGE ssamantle:leaderboard:2025-12-24
0 49 WITHSCORES
    Redis-->>LeaderboardPort: [(userId1, score1), (userId2, score2), ...]
    LeaderboardPort-->>Service: List

    Service->>LeaderboardPort: loadMyRank(userId, date)
    LeaderboardPort->>Redis: ZRANK ssamantle:leaderboard:2025-12-24 userId
    Redis-->>LeaderboardPort: rank (or null)
    LeaderboardPort->>Redis: ZSCORE ssamantle:leaderboard:2025-12-24
userId
    Redis-->>LeaderboardPort: score
    LeaderboardPort-->>Service: LeaderboardScore or null

    Service->>Service: 사용자 ID 목록 추출
[userId1, userId2, ..., myUserId]

    Service->>UserPort: loadByIds([userId1, userId2, ...])
    UserPort->>DB: SELECT * FROM users
WHERE id IN (?, ?, ...)
    DB-->>UserPort: List
    UserPort-->>Service: List

    Service->>Service: LeaderboardEntry 변환
(rank, nickname, failCount, solvedAt)

    Service-->>API: GetLeaderboardResponse
{topRankers[], myRank}
    API-->>User: 200 OK
{date, topRankers[], myRank}
```

2.5.5 일일 배치 작업 시퀀스

```
sequenceDiagram
    actor Scheduler as 스케줄러
    participant API as MaintenanceController
```

```
    participant Service as RunDailyMaintenanceBatchService
    participant MaintenancePort as MaintenanceRedisAdapter
    participant LeaderboardPort as LoadLeaderboardPort
    participant UserPort as LoadUserPort & UpdateUserPort
    participant Redis as Redis
    participant DB as MySQL

    Scheduler->>API: POST /api/v1/maintenance/daily-batch
(매일 00:00:00)
    API->>Service: execute(command)

    Service->>MaintenancePort: enableMaintenanceMode(60분)
    MaintenancePort->>Redis: SET ssamantle:maintenance:enabled true EX
3600
    Redis-->>MaintenancePort: OK

    Service->>LeaderboardPort: loadTopRankers(yesterday, 50)
    LeaderboardPort->>Redis: ZRANGE ssamantle:leaderboard:2025-12-23
0 49
    Redis-->>LeaderboardPort: [userId1, userId2, ...]
    LeaderboardPort-->>Service: List

    loop 상위 50명
        Service->>UserPort: loadById(userId)
        UserPort->>DB: SELECT * FROM users WHERE id=?
        DB-->>UserPort: User
        UserPort-->>Service: User

        Service->>Service: user.updateBestRank(rank)

        Service->>UserPort: updateUser(user)
        UserPort->>DB: UPDATE users
SET best_rank=? WHERE id=?
    end

    Service->>UserPort: resetTodaySolveForAllUsers()
    UserPort->>DB: UPDATE users
SET today_solve=false,
now_cont=CASE WHEN today_solve THEN now_cont ELSE 0 END
    DB-->>UserPort: affected rows

    Service->>MaintenancePort: disableMaintenanceMode()
    MaintenancePort->>Redis: DEL ssamantle:maintenance:enabled
    Redis-->>MaintenancePort: OK

    Service-->>API: BatchResponse
{success: true, updatedUsers: 50}
    API-->>Scheduler: 200 OK
```

---

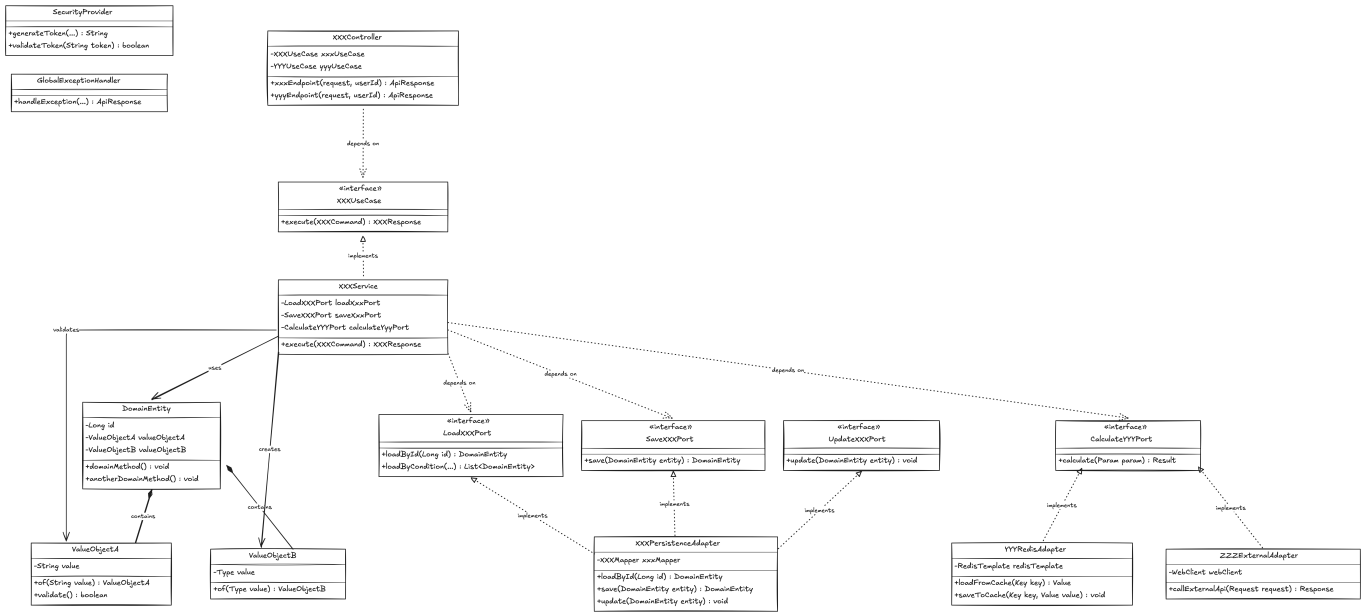
3. 클래스 다이어그램

3.1 헥사고날 아키텍처 - 전체 레이어 통합뷰

싸멘틀은 헥사고날 아키텍처(Ports & Adapters 패턴)를 기반으로 설계되었습니다.

3.2 클래스 다이어그램 이미지

3.2.1 도메인 및 애플리케이션 계층



3.3 주요 계층 및 패턴

계층	주요 클래스 패턴	책임
Inbound Adapter	XXXController	HTTP 요청/응답 처리, 인증 확인, DTO 변환
Inbound Port	XXXUseCase (interface)	애플리케이션 진입점 정의
Application Service	XXXService	Use Case 구현, 도메인 로직 오케스트레이션
Domain	Entity, ValueObject	비즈니스 규칙, 엔티티 행위, 값 객체 검증
Outbound Port	LoadXXXPort, SaveXXXPort (interface)	외부 시스템 추상화
Outbound Adapter (RDB)	XXXPersistenceAdapter	데이터베이스 영속성, 도메인 ↔ 엔티티 변환
Outbound Adapter (Redis)	XXXRedisAdapter	캐시 조회/저장, Top 1000, 리더보드 관리
Outbound Adapter (External)	XXXExternalAdapter	외부 API 호출 (유사도 계산)

3.4 의존성 규칙

```
bootstrap → adapter → application → domain
```

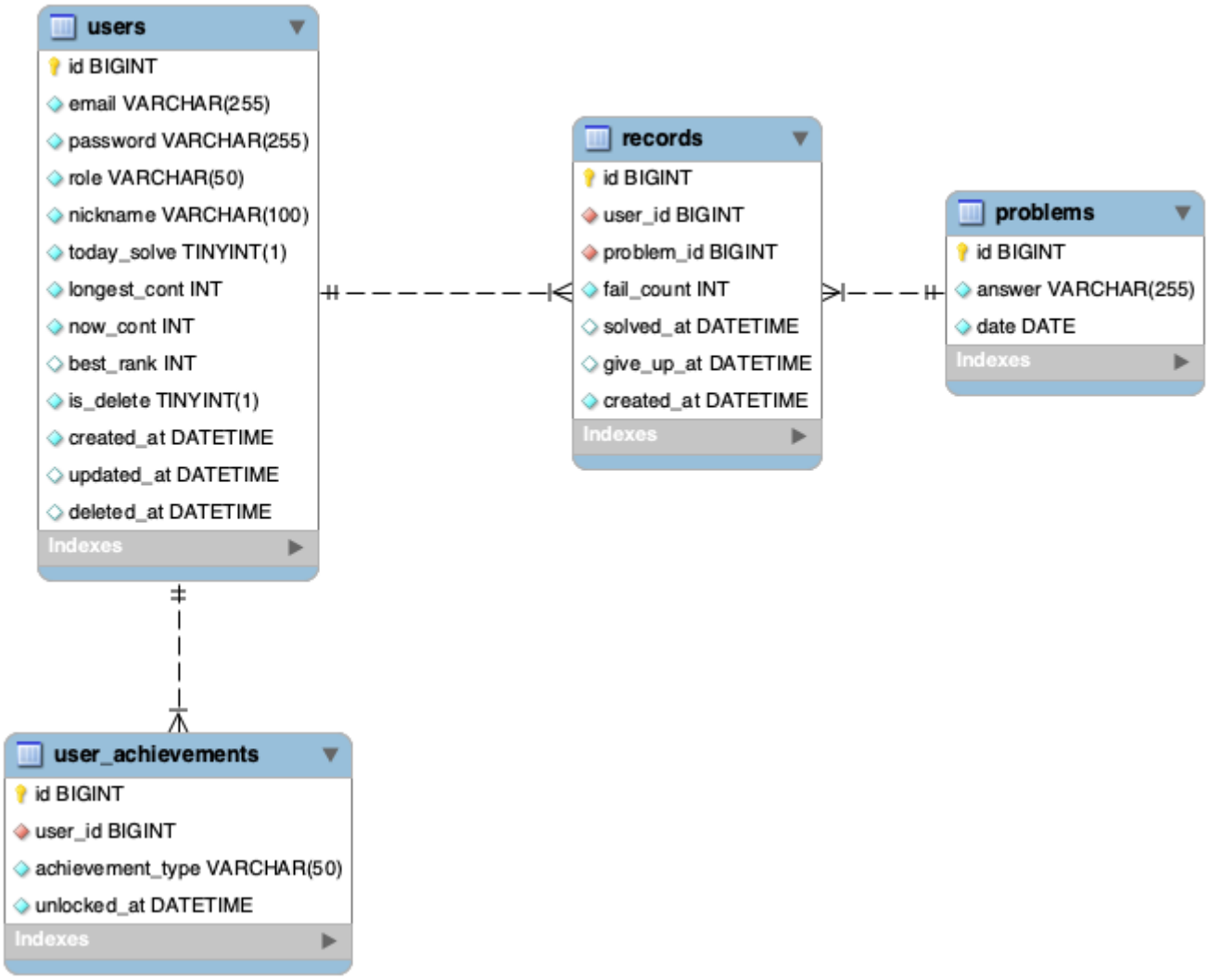
- **domain**: 의존성 없음 (순수 Java)
- **application**: domain만 의존
- **adapter**: application, domain 의존
- **bootstrap**: 모든 모듈 의존

---

4. ERD (Entity Relationship Diagram)

4.1 데이터베이스 설계

싸멘틀의 데이터베이스 ERD는 다음과 같습니다:



4.2 주요 테이블

4.2.1 users (사용자)

컬럼명	타입	제약조건	설명
id	BIGINT	PK, AUTO_INCREMENT	사용자 ID

컬럼명	타입	제약조건	설명
email	VARCHAR(255)	UNIQUE, NOT NULL	이메일
password	VARCHAR(255)	NOT NULL	암호화된 비밀번호
nickname	VARCHAR(20)	UNIQUE, NOT NULL	닉네임
role	VARCHAR(20)	NOT NULL, DEFAULT 'CUSTOMER'	역할
today_solve	BOOLEAN	NOT NULL, DEFAULT false	오늘 풀이 여부
longest_cont	INT	NOT NULL, DEFAULT 0	최장 연속 풀이 일수
now_cont	INT	NOT NULL, DEFAULT 0	현재 연속 풀이 일수
best_rank	INT	NULL	최고 순위
is_delete	BOOLEAN	NOT NULL, DEFAULT false	삭제 여부
created_at	DATETIME	NOT NULL	생성 시각
updated_at	DATETIME	NOT NULL	수정 시각

4.2.2 problems (문제)

컬럼명	타입	제약조건	설명
id	BIGINT	PK, AUTO_INCREMENT	문제 ID
answer	VARCHAR(100)	NOT NULL	정답 단어
date	DATE	UNIQUE, NOT NULL	문제 날짜

4.2.3 records (기록)

컬럼명	타입	제약조건	설명
id	BIGINT	PK, AUTO_INCREMENT	기록 ID
user_id	BIGINT	FK(users.id), NOT NULL	사용자 ID
problem_id	BIGINT	FK(problems.id), NOT NULL	문제 ID
fail_count	INT	NOT NULL, DEFAULT 0	실패 횟수
solved_at	DATETIME	NULL	해결 시각
give_up_at	DATETIME	NULL	포기 시각
created_at	DATETIME	NOT NULL	생성 시각

4.2.4 user_achievements (사용자 업적)

컬럼명	타입	제약조건	설명

컬럼명	타입	제약조건	설명
id	BIGINT	PK, AUTO_INCREMENT	업적 ID
user_id	BIGINT	FK(users.id), NOT NULL	사용자 ID
achievement_type	VARCHAR(50)	NOT NULL	업적 타입
unlocked_at	DATETIME	NOT NULL	획득 시각

4.3 주요 관계

- User (1) - (N) Record: 한 사용자는 여러 기록을 가질 수 있음
- Problem (1) - (N) Record: 한 문제는 여러 기록을 가질 수 있음
- User (1) - (N) UserAchievement: 한 사용자는 여러 업적을 획득할 수 있음

4.4 인덱스 전략

- users: email, nickname (UNIQUE)
- problems: date (UNIQUE)
- records: (user_id, problem_id) (UNIQUE), solved_at
- user_achievements: user_id, (user_id, achievement_type) (UNIQUE)

---

5. WBS & 간트 차트

5.1 Work Breakdown Structure (WBS)

1. 프로젝트 계획 (1주)

- 1.1 요구사항 분석
- 1.2 기술 스택 선정
- 1.3 아키텍처 설계
- 1.4 데이터베이스 설계

2. 개발 환경 구축 (1주)

- 2.1 멀티모듈 프로젝트 구조 설정
- 2.2 Spring Boot 초기 설정
- 2.3 데이터베이스 연결 설정
- 2.4 Redis 연동 설정
- 2.5 CI/CD 파이프라인 구축

3. 도메인 계층 개발 (1주)

- 3.1 User 도메인 모델
  - 3.1.1 User 엔티티
  - 3.1.2 Email, Password, Nickname 값 객체
  - 3.1.3 도메인 예외 정의
- 3.2 Problem 도메인 모델

- 3.3 Record 도메인 모델
- 3.4 Achievement 도메인 모델
- 3.5 게임 관련 값 객체 (WordSimilarity, LeaderboardScore)

4. 애플리케이션 계층 개발 (2주)

- 4.1 사용자 관리
  - 4.1.1 회원가입 UseCase & Service
  - 4.1.2 로그인 UseCase & Service
  - 4.1.3 정보 수정 UseCase & Service
  - 4.1.4 통계 조회 UseCase & Service
- 4.2 게임 플레이
  - 4.2.1 단어 추측 UseCase & Service
  - 4.2.2 게임 포기 UseCase & Service
  - 4.2.3 게임 상태 조회 UseCase & Service
  - 4.2.4 정답 이력 조회 UseCase & Service
- 4.3 리더보드 UseCase & Service
- 4.4 업적 시스템 UseCase & Service
- 4.5 일일 배치 UseCase & Service

5. 어댑터 계층 개발 (2주)

- 5.1 Inbound Adapter (HTTP)
  - 5.1.1 Controller 구현
  - 5.1.2 DTO 정의
  - 5.1.3 JWT 인증 필터
  - 5.1.4 전역 예외 처리
- 5.2 Outbound Adapter (RDB)
  - 5.2.1 MyBatis Mapper 작성
  - 5.2.2 Persistence Adapter 구현
  - 5.2.3 엔티티 매핑
- 5.3 Outbound Adapter (Redis)
  - 5.3.1 Game Redis Adapter
  - 5.3.2 Leaderboard Redis Adapter
  - 5.3.3 Maintenance Redis Adapter
- 5.4 Outbound Adapter (External API)
  - 5.4.1 추론 서버 연동 Adapter

6. 파이썬 추론 서버 개발 (1주)

- 6.1 Word2Vec 모델 학습
- 6.2 유사도 계산 API 구현
- 6.3 Top 1000 단어 사전 계산
- 6.4 Redis 데이터 저장 로직

7. 테스트 (1주)

- 7.1 단위 테스트 작성

- 7.2 통합 테스트 작성
- 7.3 E2E 테스트
- 7.4 부하 테스트

8. 배포 및 운영 (1주)

- 8.1 프로덕션 환경 구축
- 8.2 모니터링 설정
- 8.3 로그 수집 시스템 구축
- 8.4 백업 전략 수립

9. 문서화 (진행 중)

- 9.1 API 문서 작성
- 9.2 아키텍처 문서 작성
- 9.3 운영 가이드 작성

5.2 간트 차트

```
gantt
    title 싸멘틀 프로젝트 일정
    dateFormat  YYYY-MM-DD

    section 계획
    요구사항 분석            :a1, 2025-01-01, 3d
    기술 스택 선정           :a2, after a1, 2d
    아키텍처 설계            :a3, after a2, 2d

    section 환경 구축
    멀티모듈 설정            :b1, 2025-01-08, 2d
    Spring Boot 설정        :b2, after b1, 2d
    DB/Redis 연동           :b3, after b2, 2d
    CI/CD 구축              :b4, after b3, 1d

    section 도메인 개발
    User 도메인              :c1, 2025-01-15, 2d
    Problem/Record 도메인   :c2, after c1, 2d
    Achievement 도메인      :c3, after c2, 2d
    값 객체                 :c4, after c3, 1d

    section 애플리케이션
    사용자 관리             :d1, 2025-01-22, 4d
    게임 플레이             :d2, after d1, 4d
    리더보드/업적           :d3, after d2, 3d
    일일 배치               :d4, after d3, 2d

    section 어댑터
    HTTP Adapter           :e1, 2025-02-05, 4d
    RDB Adapter            :e2, after e1, 4d
    Redis Adapter          :e3, after e2, 3d
    External Adapter       :e4, after e3, 2d
```

```
    section 추론 서버
    모델 학습               :f1, 2025-02-19, 3d
    API 구현                 :f2, after f1, 2d
    Redis 연동               :f3, after f2, 2d

    section 테스트
    단위 테스트             :g1, 2025-02-26, 3d
    통합 테스트             :g2, after g1, 2d
    부하 테스트             :g3, after g2, 2d

    section 배포
    환경 구축               :h1, 2025-03-05, 2d
    모니터링 설정          :h2, after h1, 2d
    운영 준비               :h3, after h2, 2d
```

5.3 주요 마일스톤

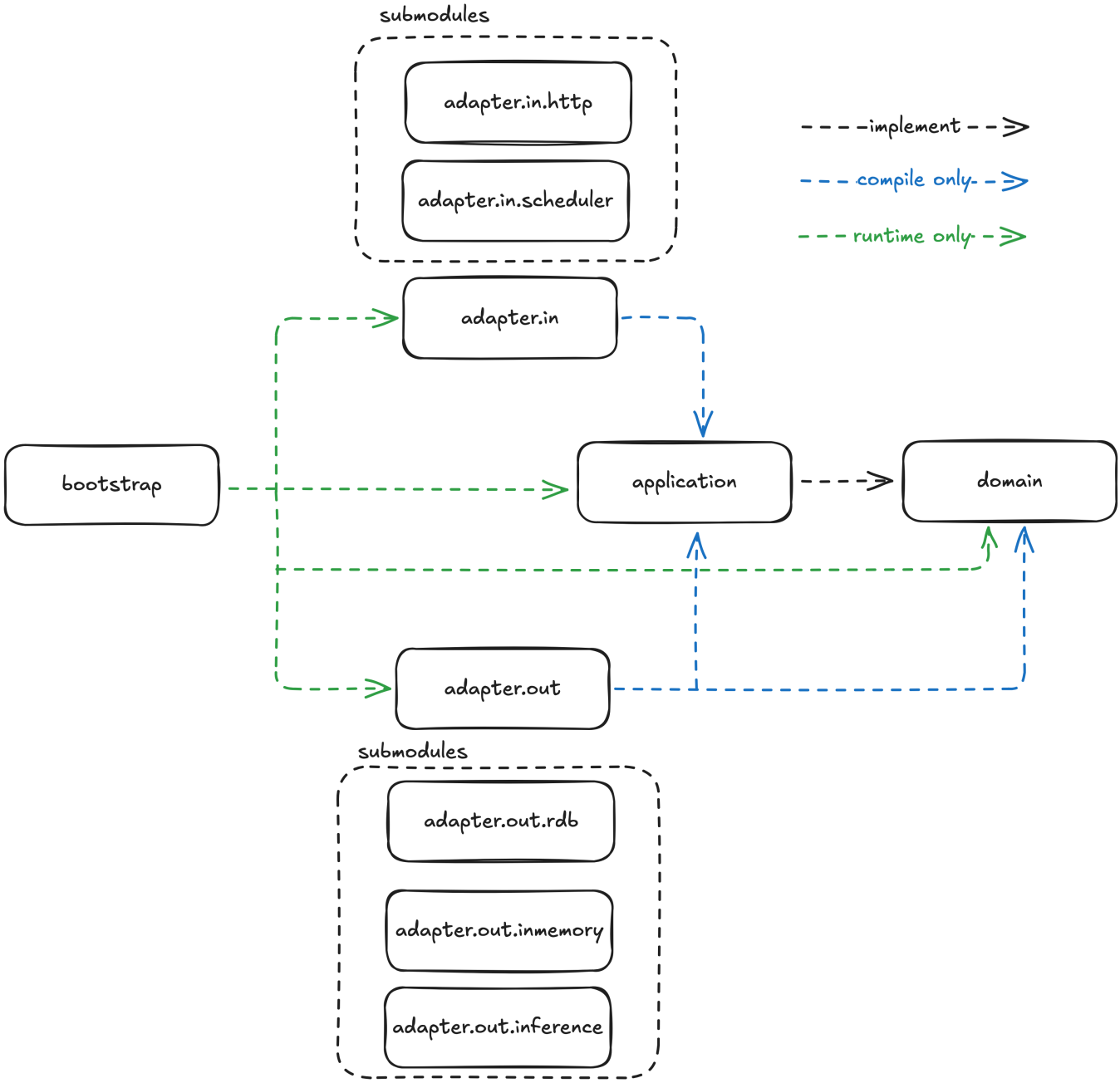
마일스톤	날짜	산출물
M1: 설계 완료	2025-01-14	요구사항 명세서, 아키텍처 설계서, ERD
M2: 도메인 개발 완료	2025-01-21	도메인 모델, 도메인 서비스
M3: 애플리케이션 개발 완료	2025-02-04	UseCase, Service, Port 인터페이스
M4: 어댑터 개발 완료	2025-02-18	Controller, Persistence Adapter, Redis Adapter
M5: 추론 서버 완료	2025-02-25	Word2Vec 모델, 유사도 API
M6: 테스트 완료	2025-03-04	테스트 리포트, 커버리지 보고서
M7: 배포 완료	2025-03-10	프로덕션 환경, 모니터링 대시보드

---

6. 모듈 구조

6.1 멀티모듈 프로젝트 구조

싸멘틀은 헥사고날 아키텍처를 기반으로 한 멀티모듈 프로젝트입니다.



6.2 모듈별 설명

6.2.1 domain 모듈

- **패키지**: `com.pigeon3.ssamantle.domain.model`
- **역할**: 핵심 비즈니스 로직, 도메인 모델
- **의존성**: 없음 (순수 Java)
- **주요 구성**:
  - Entity: User, Problem, Record, UserAchievement
  - Value Object: Email, Password, Nickname, WordSimilarity
  - Domain Service: 도메인 로직
  - Domain Exception: 도메인 예외

6.2.2 application 모듈

- **패키지**: `com.pigeon3.ssamantle.application`

- **역할**: Use Case 정의 및 구현
- **의존성**: domain
- **주요 구성**:
  - Inbound Port: XXXUseCase (interface)
  - Service: XXXService (UseCase 구현)
  - Outbound Port: LoadXXXPort, SaveXXXPort (interface)
  - Command/Response: DTO

6.2.3 adapter-in-http 모듈

- **패키지**: `com.pigeon3.ssamantle.adapter.in.http`
- **역할**: REST API 제공
- **의존성**: application, domain
- **주요 구성**:
  - Controller: HTTP 요청/응답 처리
  - DTO: Request/Response
  - Security: JWT 인증
  - Exception Handler: 전역 예외 처리

6.2.4 adapter-out-rdb 모듈

- **패키지**: `com.pigeon3.ssamantle.adapter.out.rdb`
- **역할**: MySQL 영속성
- **의존성**: application, domain
- **주요 구성**:
  - Persistence Adapter: Port 구현
  - MyBatis Mapper: SQL 매핑
  - Entity: RDB 엔티티
  - Mapper: 도메인 ↔ 엔티티 변환

6.2.5 adapter-out-in-memory 모듈

- **패키지**: `com.pigeon3.ssamantle.adapter.out.inmemory`
- **역할**: Redis 캐싱
- **의존성**: application, domain
- **주요 구성**:
  - Redis Adapter: 캐시 조회/저장
  - Leaderboard Adapter: 리더보드 관리
  - Maintenance Adapter: 유지보수 모드

6.2.6 adapter-out-inference 모듈

- **패키지**: `com.pigeon3.ssamantle.adapter.out.inference`
- **역할**: 추론 서버 연동
- **의존성**: application, domain
- **주요 구성**:
  - Inference Adapter: 유사도 계산 API 호출
  - WebClient: HTTP 통신

6.2.7 bootstrap 모듈

- **패키지**: `com.pigeon3.ssamantle.bootstrap`
- **역할**: 애플리케이션 시작점
- **의존성**: 모든 모듈
- **주요 구성**:
  - BootstrapApplication: Spring Boot Main
  - Configuration: 빈 설정

6.3 의존성 그래프

```
bootstrap
    ├── adapter-in-http
    │   ├── application
    │   │   └── domain
    │   └── domain
    ├── adapter-out-rdb
    │   ├── application
    │   │   └── domain
    │   └── domain
    ├── adapter-out-in-memory
    │   ├── application
    │   │   └── domain
    │   └── domain
    └── adapter-out-inference
        ├── application
        │   └── domain
        └── domain
```

6.4 빌드 설정 (Gradle)

```
// settings.gradle
rootProject.name = 'ssamatle-server'

include 'domain'
include 'application'
include 'adapter:in:http'
include 'adapter:out:rdb'
include 'adapter:out:in-memory'
include 'adapter:out:inference'
include 'bootstrap'
```

---

7. 화면 설계서

7.1 개요

(화면 설계는 프론트엔드 팀과 협의 후 작성 예정)

7.2 주요 화면 목록

7.2.1 인증 화면

- [ ] 로그인 화면
- [ ] 회원가입 화면

7.2.2 게임 화면

- [ ] 메인 게임 화면
- [ ] 게임 결과 화면 (정답 시)
- [ ] 게임 결과 화면 (포기 시)

7.2.3 정보 화면

- [ ] About 페이지 (게임 방법 설명)
- [ ] FAQ 페이지

7.2.4 사용자 화면

- [ ] 마이페이지
- [ ] 게임 통계 화면
- [ ] 업적 화면
- [ ] 정보 수정 화면

7.2.5 리더보드 화면

- [ ] 일일 리더보드
- [ ] 과거 리더보드 (날짜 선택)

7.2.6 정답 이력 화면

- [ ] 오늘 정답 이력
- [ ] 어제 정답 이력

7.3 화면 흐름도

(추후 작성)

7.4 와이어프레임

(추후 작성)

7.5 스타일 가이드

(추후 작성)

부록

A. 참고 문서

- API 문서
- 아키텍처 가이드
- 요구사항 정의서
- 클래스 다이어그램

B. 용어 정리

용어	설명
싸멘틀	SSAFY + Semantle의 합성어
Semantle	단어의 의미 유사도 기반 추론 게임
스트릭 (Streak)	연속 풀이 일수
Top 1000	정답과 가장 유사한 상위 1000개 단어
리더보드	일일 순위표
업적	특정 조건 달성 시 부여되는 성취
failCount	오답 시도 횟수
todaySolve	오늘 문제를 풀었는지 여부
nowCont	현재 연속 풀이 일수
longestCont	최장 연속 풀이 일수
bestRank	최고 순위

C. 변경 이력

버전	날짜	작성자	변경 내용
1.0	2025-12-24	Development Team	초안 작성

END OF DOCUMENT