프로젝트 제안서

[Project Proposal] School Mate: 교사·학생·학부모를 잇는 스마트 학사 관리 플랫폼

1. 프로젝트 목표 (Why Backend Only?)

> **"화면 껍데기보다 데이터의 무결성과 복잡한 비즈니스 로직에 집중합니다."**

본 프로젝트는 프론트엔드 개발 리소스를 배제하고, 학교 시스템의 핵심인 **권한 관리(RBAC)**, **학사 데이터 무결성**, **복잡한 통계 처리** 등 백엔드 본질에 100% 집중하기 위해 기획 되었습니다.

- **Backend Intensive (기술적 깊이 심화)**
  - 단순 게시판을 넘어, **수강신청(동시성 제어)**, **성적 산출(복잡한 연산)**, **출결 관리(상태 제어)** 등 학교 현장의 심도 있는 로직을 코드에 녹여냅니다.
  - *AOP(Aspect Oriented Programming)**를 적극 도입하여 로깅, 권한 체크, 성능 측정 등 횡단 관심사를 분리하고 코드의 응집도를 높입니다.
- **Hybrid Data Access Strategy (이원화 전략)**
  - **Command (CUD):** 데이터의 정합성이 중요한 학사 정보 생성/수정/삭제는 **Spring Data JPA**를 사용하여 객체 지향적으로 설계합니다.
  - **Query (Read):** 성적 석차 산출, 기간 별 출석 통계 등 복잡한 조회가 필요한 영역은 **MyBatis**를 사용하여 SQL을 직접 튜닝합니다.
- **Validation & Documentation (검증과 문서화)**
  - 화면이 없는 Headless 환경이므로 **Swagger**를 통해 API 명세가 곧 테스트 도구가 되도록 "살아있는 문서"를 구축합니다.

---

2. 개발 환경 및 기술 스택

구분	기술 스택	선정 이유 및 활용 전략
Language	Java 17+	Record Class 등 최신 문법 활용
Framework	Spring Boot 3.x	Jakarta EE 기반의 최신 생태계 및 AOP 모듈 활용

Security	Spring Security + JWT	교사/학생/학부모 3단계 권한 분리 및 세션/토큰 관리
Database	MySQL (or MariaDB)	관계형 데이터 설계를 통한 데이터 무결성 보장
ORM (Write)	**Spring Data JPA**	도메인 모델 중심 설계, Dirty Checking 활용
Mapper (Read)	**MyBatis**	동적 쿼리를 활용한 성적/출결 통계 및 대시보드 최적화
Docs	Swagger (SpringDoc)	UI 없이 API 테스트가 가능한 환경 구축
Test	Postman	User Flow(사용자 흐름) 시나리오 테스트
Collab	GitHub	Git Flow 전략 및 Pull Request 코드 리뷰

3. 전체 시스템 구조 (Architecture)

> **"View 없이 오직 Data(JSON)로 소통하며, AOP로 관점을 분리한 구조"**

본 프로젝트는 계층화 아키텍처(Layered Architecture)를 따르되, 모든 팀원이 **Controller → Service → Repository** 전 과정을 담당하는 Vertical Slicing 방식을 채택합니다.

코드 스니펫

```
graph TD
    User[API Client / Postman] --> |HTTP Request| Swagger[Swagger UI]
    Swagger --> |JSON Data| Controller[API Controller Layer]

    subgraph Backend System
        direction TB
        Controller --> |DTO| Service[Service Layer]

        subgraph "AOP Aspect Layer"
        LogAspect[Logging Aspect] -.-> Service
        AuthAspect[Role Check Aspect] -.-> Service
        PerfAspect[Performance Monitor] -.-> Service
        end

        Service --> |Domain Entity| Repository[Data Access Layer]

        subgraph "Dual Data Access Strategy" [cite: 35]
        direction LR
```

```
        JPA[JPA Repositories] --> |Save/Update/Delete| DB[(MySQL Databas
e)]
        MyBatis[MyBatis Mappers] --> |Complex Select/Stats| DB
        end
    end
```

- **API Controller:** 요청 검증(Validation) 및 응답 포맷 통일.
- **Service Layer:** @Transactional 기반 비즈니스 로직, **AOP**를 적용하여 핵심 로직과 부가 기능(로깅, 권한) 분리
- **Repository Layer:** JPA(CUD용)와 MyBatis(조회용)의 명확한 역할 분리

---

4. 팀원 역할 분담 (R&R)

> **"5명 전원, Controller부터 DAO까지 책임지는 도메인 전문가"**

1. Member 1 (Team Leader): 공통 모듈 & 사용자 관리 (Auth & Infra)

- **Core Logic: 인증/인가 및 AOP 설계**
  - **Spring Security & JWT:** 교사(TEACHER), 학생(STUDENT), 학부모(PARENT)의 Multi-Role 인증 시스템 구현.
  - **AOP Framework 구축:** 모든 API 요청/응답 로그를 남기는 `LoggingAspect`와 실행 시간을 측정하는 `PerformanceAspect`를 구현하여 팀원들이 어노테이션 하나로 적용할 수 있게 지원합니다.
- **Support:** GlobalExceptionHandler를 통한 에러 응답 표준화

2. Member 2: 학사 일정 & 수강 신청 (Concurrency Specialist)

- **Core Logic: 동시성 제어 및 스케줄링**
  - **수강 신청 시스템:** 인기 강좌(방과 후 학교 등) 신청 시 발생하는 트래픽을 처리하기 위해 비관적 락(Pessimistic Lock) 또는 낙관적 락을 적용하여 정원 초과를 방지합니다.
  - **시간표 관리:** JPA로 강좌 정보를 관리하고, MyBatis로 학급 별/교사 별 주간 시간표 격자 데이터를 효율적으로 조회합니다.

3. Member 3: 출결 & 생활 기록부 (State Machine Logic)

- **Core Logic: 상태 관리 및 프로세스**

- **출결 프로세스:** 출석(ATTEND) → 지각(LATE) → 조퇴(EARLY) → 결석(ABSENT) 상태를 Enum으로 관리하고, 나이스(NEIS) 시스템을 모방한 출결 마감 로직을 구현합니다.
- **체험 학습 신청(결재):** 학생 신청 → 학부모 승인 → 담임 교사 최종 승인으로 이어지는 3단계 결재 프로세스를 구현합니다.

4. Member 4: 소통 & 알림장 (File & Hierarchy)

- **Core Logic: 계층형 데이터 및 파일 처리**
  - **가정통신문/알림장:** PDF, HWP 등 첨부파일 업로드 기능을 `MultipartFile`로 구현하고 로컬/외부 저장소에 저장합니다.
  - **Q&A 게시판:** 비밀 글 기능(작성자와 교사만 열람)과 계층형 댓글(답글) 구조를 설계하여 학부모 상담 기능을 시스템 화 합니다.

5. Member 5: 성적 관리 & 통계 (Data Analysis & MyBatis)

- **Core Logic: 복잡한 연산 및 리포팅**
  - **성적 산출:** 지필 평가 + 수행 평가 반영 비율에 따른 최종 점수 계산 로직을 도메인 서비스에 구현합니다.
  - **MyBatis 통계:** 학급별 평균, 과목별 석차 등급 산출, 성적 추이 그래프 용 데이터를 SQL의 `Window Function(RANK, DENSE_RANK)`과 `GROUP BY`를 활용해 추출합니다.

---

5. 개발 프로세스 및 검증 전략

> **"UI가 없기에 더욱 엄격한 API 검증 프로세스"**

1. **Swagger = Live Documentation**
   - `@Operation`, `@Schema` 어노테이션을 꼼꼼히 작성하여 프론트엔드 없이도 API의 입력/출력 예시를 명확히 보여줍니다.
2. **Postman Scenario Test**
   - 단위 테스트를 넘어 **User Flow**를 검증합니다.
   - *Scenario 예시:* [Step 1] 교사 로그인 → [Step 2] 수학 시험 점수 입력 → [Step 3] 학생 로그인 → [Step 4] 내 성적 및 석차 조회
3. **AOP Validation**

- 핵심 비즈니스 로직에 AOP가 정상적으로 적용되었는지(로그 적재, 권한 없는 접근 차단 등) 확인하는 테스트 코드를 작성합니다.

---

6. 프로젝트 일정 (Timeline) - 4주 완성

- **1주차: 설계 및 환경 구축 (Foundation)**
  - ERD 설계 (JPA Entity 및 MyBatis Table 매핑 전략 수립).
  - Spring Boot 초기 세팅 및 AOP 공통 모듈(Log, Auth) 구현.
- **2주차: 핵심 도메인 구현 (Core Logic)**
  - 회원가입/로그인, 수강신청(Lock 구현), 성적 입력 CRUD 완료.
  - Member별 Vertical Slicing 개발 진행.
- **3주차: 심화 로직 및 연동 (Advanced & Integration)**
  - MyBatis를 활용한 통계 쿼리 작성 및 최적화.
  - Swagger를 통한 팀원 간 API 연동 테스트.
- **4주차: 안정화 및 문서화 (Polishing)**
  - Postman 시나리오 테스트 수행 및 버그 수정.
  - 최종 발표 자료 준비 및 리드미(README) 작성.

---

7. 마무리

"School Mate, 데이터로 학교를 연결하다."

우리는 이번 프로젝트를 통해 단순한 CRUD를 넘어, **JPA와 MyBatis의 장점을 적재적소에 활용**하고 **AOP로 시스템의 관점을 분리**하는 아키텍처를 경험할 것입니다. 화려한 UI 대신 견고한 서버 사이드 로직으로 무장한 백엔드 개발자로 성장하겠습니다.