동덕여자대학교 족보 공유 플랫폼 구축 기획서

1. 프로젝트 개요 (Project Overview)

1.1. 프로젝트 목표

본 프로젝트는 동덕여자대학교 재학생만을 위한 족보(시험 자료) 공유 및 학술 커뮤니티 플랫폼을 구축하는 것을 목표로 한다. 학교 이메일 인증을 기반으로 안전하고 신뢰할 수 있는 자료 공유 환경을 제공하고, 족보 자료의 업로드 및 다운로드를 통해 학생들 간의 학습 및 정보 교류를 활성화하여 학업 성취도 향상에 기여하고자 한다.

1.2. 주요 기능 요약

카테고리	주요 기능	상세 설명
사용자 관리	학교 이메일 인증 기반 회원가입/로그인 및 재학생 전용 접근 통제	@dongduk.ac.kr 도메인 이메일 인증을 통과한 재학생만 회원가입 및 서비스 이용
자료 공유	과목별/과별 족보 게시판 및 PDF 업로드/다운로드	과(학과)단위 게시판을 통해 족보 자료 분류, PDF 업로드 및 다운로드 및 게시글 CRUD 기능 제공
경제 시스템	포인트 시스템	자료 업로드 시 포인트 적립, 자료 다운로드 시 포인트 차감으로 자료 공유 유도

2. 요구사항 분석 (Requirements Analysis)

2.1. 사용자 인증 및 접근 통제

ID	요구사항	상세 내용	우선순위
R-AUTH-001	학교 이메일 인증 기반 회원가입	학번@dongduk.ac.kr 형식의 이메일로 인증 메일을 발송하고, 인증 완료 후 비밀번호를 설정하여 회원가입을 완료한다.	High
R-AUTH-002	로그인 기능	인증된 이메일(ID)과 비밀번호를 사용하여 로그인한다.	High
R-AUTH-003	재학생 전용 서비스	학교 이메일 인증을 통과한 사용자만 플랫폼을 이용할 수 있도록 접근을 통제한다.	High

2.2. 족보 공유 시스템

ID	요구사항	상세 내용	우선순위
R-SHARE-002	PDF 파일 업로드	사용자는 PDF 형식의 파일을 게시글에 첨부하여 업로드할 수 있다.	High
R-SHARE-004	자료 검색 및 필터링	과목명, 교수명, 학과 등으로 자료를 검색하고 필터링할 수 있는 기능을 제공한다.	Medium

2.3. 포인트 및 경제 시스템

ID	요구사항	상세 내용	우선순위
R-POINT-001	자료 업로드 포인트 지급	족보 자료를 업로드할 때 일정량의 포인트를 지급한다. (예: 100P)	High
R-POINT-002	자료 다운로드 포인트 차감	다른 사용자의 족보 자료를 다운로드할 때 일정량의 포인트를 차감한다. (예: 50P)	High

3. 시스템 아키텍처 및 Tech Spec 설계 (System Architecture & Tech Spec)

3.1. AWS 기반 시스템 아키텍처 (AWS-based System Architecture)

구성 요소	AWS 서비스	역할 및 설명
프론트엔드 호스팅	**Amazon S3 & Amazon CloudFront**	정적 웹사이트 호스팅 및 CDN(Content Delivery Network)을 통한 빠른 콘텐츠 전송.
백엔드/API 서버	**AWS Elastic Beanstalk**	애플리케이션 배포 및 관리를 자동화하고, 트래픽 증가에 따른 서버 자동 확장(Auto Scaling)을 지원.
데이터베이스	**Amazon RDS (PostgreSQL)**	사용자 정보, 게시글, 포인트 내역 등 핵심 관계형 데이터 저장. 높은 안정성과 자동 백업 기능 제공.
파일 스토리지	**Amazon S3**	사용자가 업로드하는 PDF 족보 파일 저장. 무제한에 가까운 확장성과 99.9%의 내구성을 보장.
사용자 인증	**Spring Security + JWT(자체 발급) + 이메일 인증(토큰)**	자체 인증 서버 구축
네트워크/보안 (추후 구현)	**Amazon VPC, Security Group, AWS WAF**	격리된 네트워크 환경 구축 및 외부 위협으로부터 시스템 보호.
DNS 관리	**Amazon Route 53**	도메인 등록 및 안정적인 DNS 서비스 제공.

3.2. 기술 스택 (Technology Specification)

구분	기술 스택	상세 설명
프론트엔드	**React / TypeScript**	사용자 인터페이스 구축. Next.js를 활용하여 SEO 및 성능 최적화(SSR/SSG)를 고려.
백엔드	**Java (Spring Boot)**	안정적이고 강력한 API 서버 구축. Java 기반의 엔터프라이즈급 구조로 높은 안정성과 확장성 확보.
데이터베이스	**PostgreSQL + Redis**	안정성과 데이터 무결성이 중요한 서비스에 적합. 지리 정보, JSONB 등 다양한 데이터 타입 지원.
파일 처리	**PDF.js (프론트엔드)**	PDF 미리보기 기능 구현.

3.3. 인프라 설계 근거

프론트엔드 호스팅**: S3 + CloudFront**

문제 상황
프론트엔드는 React 기반의 정적 웹 애플리케이션으로, 별도의 서버를 두고 운영할 경우 인프라 관리 부담이 커지고 트래픽 변동에 따른 비용 예측이 어렵다는 문제가 있었다.

해결 방향
서버 로직이 필요 없는 정적 리소스는 서버와 분리하여 배포하고, 사용자 위치와 관계없이 빠르게 제공할 수 있는 구조가 필요했다.

기술 선택 이유
이에 따라 S3를 정적 웹 호스팅 용도로 사용하고, CloudFront CDN을 적용하여 전 세계 엣지 로케이션을 통한 빠른 콘텐츠 전송과 비용 효율적인 운영을 가능하게 했다.

백엔드**/API** 서버**: Elastic Beanstalk**

문제 상황
Spring Boot 기반 API 서버를 직접 EC2에 배포할 경우, 배포·롤백·로그 관리·오토스케일링 등을 모두 수동으로 관리해야 하는 부담이 있었다.

해결 방향
애플리케이션 개발에 집중하면서도 트래픽 증가에 유연하게 대응할 수 있는 관리형 배포 환경이 필요했다.

기술 선택 이유
Elastic Beanstalk를 사용하여 애플리케이션 배포와 운영을 자동화하고, Auto Scaling 및 헬스체크 기능을 통해 안정적인 API 서버 운영이 가능하도록 설계하였다.

데이터베이스**: RDS (PostgreSQL)**

문제 상황
사용자 정보, 게시글, 포인트 내역 등 핵심 데이터는 정합성과 트랜잭션 처리가 중요하며, 장애 발생 시 데이터 손실 위험을 최소화해야 했다.

해결 방향
안정적인 관계형 데이터베이스와 자동 백업, 장애 대응 기능을 제공하는 관리형 DB 서비스가 필요했다.

기술 선택 이유
Amazon RDS(PostgreSQL)를 사용하여 데이터 무결성을 보장하고, 백업·복구·모니터링을 자동화함으로써 DB 운영 부담을 줄였다.

파일 스토리지**: S3 (PDF** 업로드**/**보관**)**

문제 상황
PDF 족보 파일은 용량이 크고 장기간 보관되어야 하며, 다운로드 트래픽이 API 서버 성능에 영향을 줄 수 있었다.

해결 방향
파일 저장과 전송을 애플리케이션 서버와 분리하여, 서버 부하를 최소화하고 확장성을 확보할 필요가 있었다.

기술 선택 이유
무제한에 가까운 확장성과 높은 내구성을 제공하는 Amazon S3를 파일 저장소로 사용하여, 안정적인 파일 관리와 효율적인 트래픽 처리를 가능하게 했다.

인증/인가**: Spring Security + JWT +** 이메일 인증 **(**확장 설계**)**

문제 상황
사용자 수 증가와 서버 확장 시, 세션 기반 인증은 서버 간 세션 공유 문제가 발생할 수 있으며 확장성이 떨어진다는 한계가 있다.

해결 방향
서버 확장에 유리하고 프론트엔드·백엔드 분리 구조에 적합한 인증 방식이 필요했다.

기술 선택 및 적용 범위
JWT 기반 인증과 Spring Security를 활용한 구조를 목표로 설계하였으나,
본 프로젝트에서는 개발 기간과 범위를 고려하여 인증 기능을 단순화하여 구현하고,
JWT 및 이메일 인증은 추후 확장 가능한 구조로 설계 단계에서 반영하였다.

캐시/성능 개선**: Redis (**확장 고려**)**

문제 상황
반복 조회되는 데이터가 증가할 경우, DB 부하와 응답 속도 저하가 발생할 수 있다.

해결 방향
자주 사용되는 데이터를 메모리 기반 캐시로 분리하여 성능을 개선할 필요가 있었다.

기술 선택 이유
Redis는 빠른 응답 속도와 TTL 기반 데이터 관리가 가능해 캐시 및 인증 보조 용도로 적합하므로, 서비스 확장 단계에서 적용할 수 있도록 설계에 반영하였다.

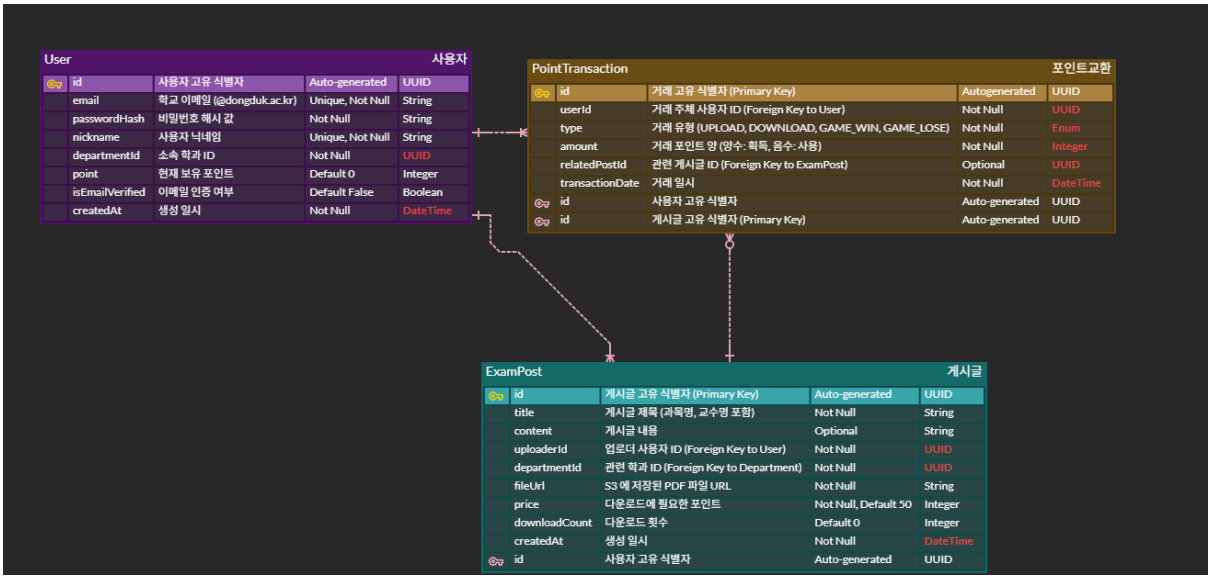
DNS 관리**: Route 53**

문제 상황
도메인, CDN, 서버를 각각 관리할 경우 설정 복잡도와 운영 실수가 발생할 수 있다.

해결 방향
AWS 인프라와 자연스럽게 연동되는 통합 DNS 관리가 필요했다.

기술 선택 이유
Route 53을 사용하여 도메인과 DNS를 일관되게 관리하고, 향후 환경 분리 및 확장에 대비하였다.

4. ERD 및 Domain Class 정의 (ERD & Domain Class Definition)



4.1. 개체-관계 다이어그램 (Entity-Relationship Diagram, ERD)

- **User (사용자)**: 플랫폼을 이용하는 동덕여대 재학생 개체.
  - 관계: User는 Department에 속하며 (N:1), 여러 개의 ExamPost, FreePost, Comment, PointTransaction을 생성한다 (1:N).
- **Department (학과)**: 족보 게시판의 분류 기준이 되는 학과 개체.
  - 관계: 하나의 Department는 여러 명의 User와 여러 개의 ExamPost를 가진다 (1:N).
- **ExamPost (족보 게시글)**: 사용자가 업로드하는 족보 자료 게시글 개체.
  - 관계: User가 작성하며 (N:1), 여러 개의 Comment를 가질 수 있다 (1:N). PointTransaction과 연관되어 다운로드/업로드 기록을 남긴다 (1:N).
- **Comment (댓글)**: ExamPost 또는 FreePost에 달리는 댓글 개체.
  - 관계: User가 작성하며 (N:1), ExamPost 또는 FreePost에 종속된다 (N:1).
- **PointTransaction (포인트 거래)**: 포인트의 획득 및 사용 내역을 기록하는 개체.
  - 관계: User와 ExamPost에 종속되어 거래의 주체와 대상을 명확히 한다 (N:1).

4.2. 핵심 도메인 클래스 정의 (Core Domain Class Definition)

4.2.1. User Class

속성 (Attribute)	타입 (Type)	설명	제약 조건
id	UUID	사용자 고유 식별자 (Primary Key)	Auto-generated
email	String	학교 이메일 (@dongduk.ac.kr)	Unique, Not Null
passwordHash	String	비밀번호 해시 값	Not Null
nickname	String	사용자 닉네임	Unique, Not Null

departmentId	UUID	소속 학과 ID (Foreign Key to Department)	Not Null
point	Integer	현재 보유 포인트	Default 0
isEmailVerified	Boolean	이메일 인증 여부	Default False
createdAt	DateTime	생성 일시	Not Null

4.2.2. ExamPost Class

속성 **(Attribute)**	타입 **(Type)**	설명	제약 조건
id	UUID	게시글 고유 식별자 (Primary Key)	Auto-generated
title	String	게시글 제목 (과목명, 교수명 포함)	Not Null
content	String	게시글 내용	Optional
uploaderId	UUID	업로더 사용자 ID (Foreign Key to User)	Not Null
departmentId	UUID	관련 학과 ID (Foreign Key to Department)	Not Null
fileUrl	String	S3에 저장된 PDF 파일 URL	Not Null
price	Integer	다운로드에 필요한 포인트	Not Null, Default 50
downloadCount	Integer	다운로드 횟수	Default 0
createdAt	DateTime	생성 일시	Not Null

4.2.3. PointTransaction Class

속성 **(Attribute)**	타입 **(Type)**	설명	제약 조건
id	UUID	거래 고유 식별자 (Primary Key)	Auto-generated
userId	UUID	거래 주체 사용자 ID (Foreign Key to User)	Not Null
type	Enum	거래 유형 (UPLOAD, DOWNLOAD, GAME_WIN, GAME_LOSE)	Not Null
amount	Integer	거래 포인트 양 (양수: 획득, 음수: 사용)	Not Null
relatedPostId	UUID	관련 게시글 ID (Foreign Key to ExamPost)	Optional
transactionDate	DateTime	거래 일시	Not Null

5. 칸반보드 (Kanban Board)

5.1. 칸반보드 (Kanban Board) 주요 항목

본 프로젝트의 개발 관리를 위해 사용할 칸반보드의 주요 에픽(Epic) 및 태스크(Task) 목록은 다음과 같다. 각 태스크는 **To Do**, **In Progress**, **Review**, **Done**의 상태를 거치게 된다.

에픽 **(Epic)**	주요 태스크 **(Task)**	담당 **(Role)**	예상 소요 시간 **(Story Points)**
E1: Core Infrastructure & Setup	AWS VPC, S3, RDS 설정 및 보안 그룹 구성	DevOps	5
	Next.js/NestJS 프로젝트 초기 설정 및 배포 자동화 (CI/CD)	Backend/DevOps	8
	DB 스키마 초기 마이그레이션 (User, Department 테이블)	Backend	3
E2: User & Authentication	학교 이메일 인증 (SMTP 연동 및 토큰 발급) API 구현	Backend	8
	회원가입/로그인/비밀번호 찾기 UI/UX 구현	Frontend	5
	사용자 정보 관리 (닉네임, 포인트 조회) API/UI 구현	Full-stack	5
E3: Exam Sharing System	과별 게시판 목록 및 상세 조회 API 구현	Backend	5
	PDF 파일 S3 업로드 UI 구현	Full-stack	8
	PDF 미리보기 기능 (PDF.js 연동) 구현	Frontend	5
	자료 다운로드 및 포인트 차감 로직 구현	Backend	8
E4: Point System	포인트 거래 내역 (PointTransaction) 기록 API 구현	Backend	5

6. 기획서 통합 및 다이어그램 (Integration & Diagram)

6.1. 시스템 아키텍처 다이어그램