****

**Spo-iler 프로젝트 개요**

**조장: 피승빈**

**조원: 박서하**

**1. 요구사항 정의서**본 프로젝트의 요구사항은 다음과 같이 분류됩니다:

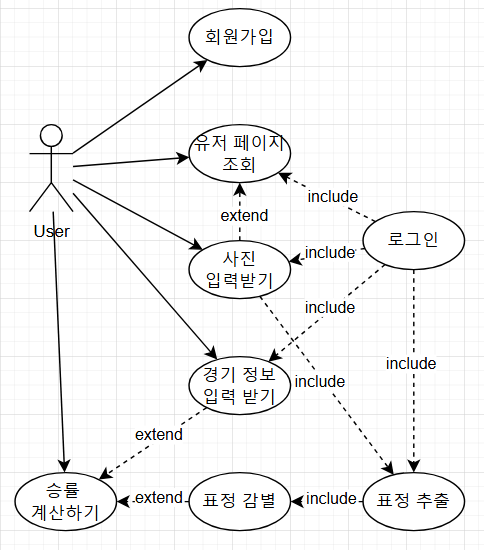
**1-1. 사용자 기능**- 회원가입: 사용자 정보를 입력받아 계정을 생성  
- 로그인/로그아웃: 사용자가 보안 인증을 통해 시스템에 접근 및 종료

**1-2. 관리 기능**- 권한 관리: 사용자 역할에 따라 접근 가능한 리소스 제한  
- 관리자 페이지: 모든 사용자 정보 및 활동 로그 관리

**1-3. 보안 기능**- JWT 기반 인증: 토큰을 통한 세션 관리  
- 데이터 암호화: 사용자 비밀번호 저장 시 암호화

**1-4. 경기 분석 기능**- 입력받은 경기의 현황과 경기 관계자의 사진을 입력  
- 경기 관계자의 표정에서 감정을 추출  
- 추출된 감정을 통해 경기의 흐름을 예측, 승리확률을 표시  
- 사용자가 분석을 의뢰한 사진들을 profile 페이지에 표시

**2. UseCase 다이어그램**아래는 주요 UseCase 다이어그램입니다.



**액터**:  
- User

**기능**:  
- 회원가입, 로그인/로그아웃  
- 유저 페이지 조회(지금까지 분석을 진행하였던 사진 로그 저장)  
- 경기 정보 및 사진의 입력  
- 경기 정보 및 표정의 감별을 통해 승률을 계산

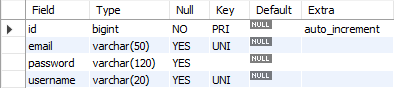
**3. 테이블 구조도 (ERD)**아래는 데이터베이스 테이블 구조를 나타냅니다:  
(해당 개발은 안정성을 위해 NoSQL 쿼리를 사용하여 작성되었습니다. 아래 도표는 Hubernate를 통해 생성되는 테이블과 칼럼의 목록입니다. )

|  |  |  |
| --- | --- | --- |
| 테이블 이름 | 컬럼 | 설명 |
| Role | id, name(역할명) | 사용자 역할 정보 |
| User | id, username(이름), email(이메일),password( 비밀번호), | 사용자 정보 |
| User\_Role | 사용자 ID, 역할 ID | 다대다 관계 관리 |

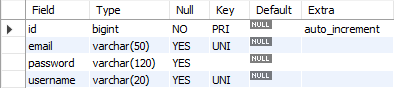
**3-1. Roles table**

****

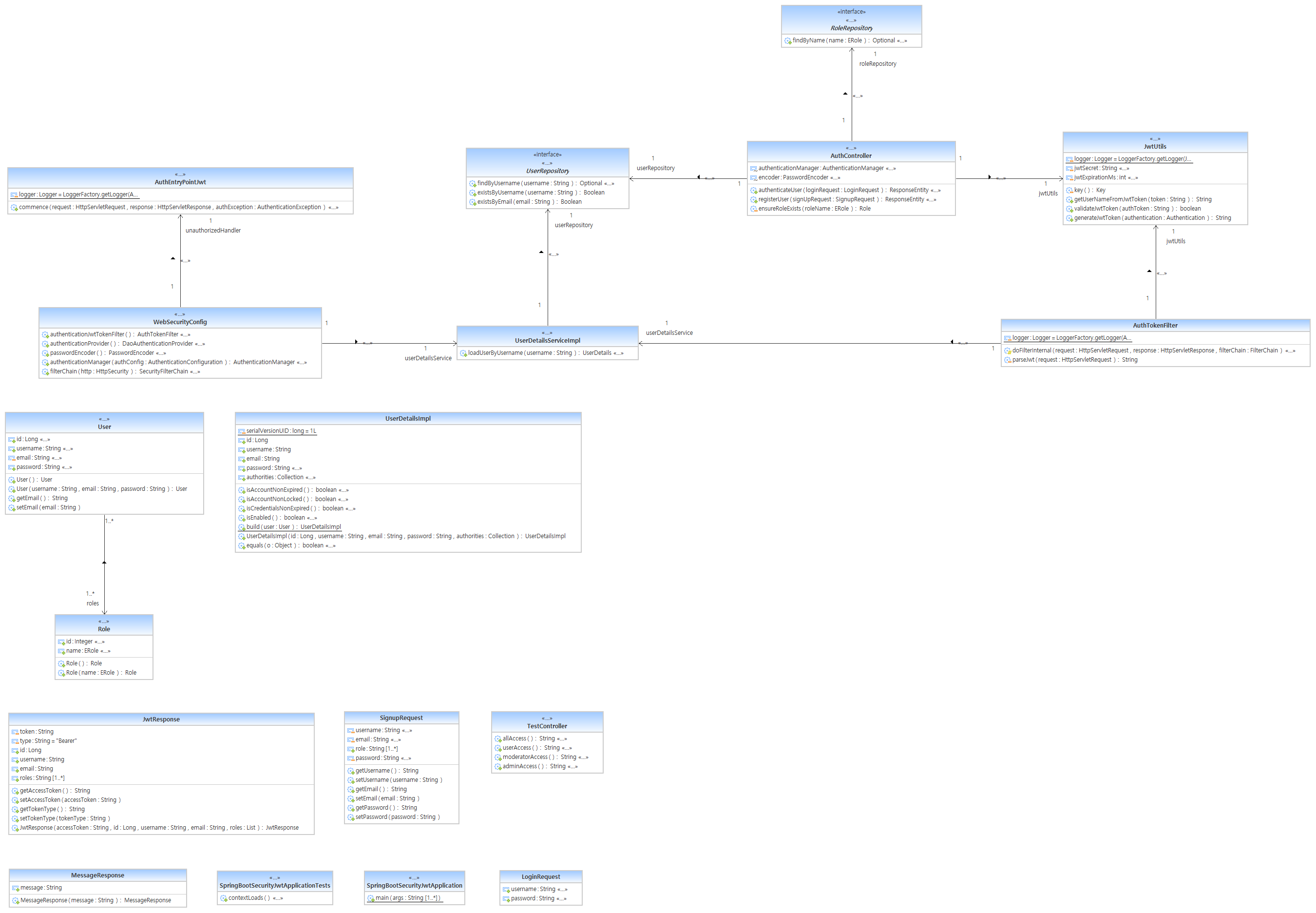
**3-2. User table**

****

**3-3. User\_Role table**



**4. 클래스 다이어그램**프로젝트의 주요 클래스와 이들의 관계를 다음과 같이 나타낼 수 있습니다:



**5. 사용된 라이브러리 및 프레임워크**본 프로젝트에서 사용된 주요 라이브러리와 프레임워크는 다음과 같습니다:

**5-1. 프론트엔드**- Vue.js: 사용자 인터페이스를 구축하기 위한 프레임워크로, 코드의 재사용성을 높임.- Vue Router: 애플리케이션의 클라이언트 사이드 라우팅을 관리.- Vuex: 상태 관리를 중앙에서 관리하여 컴포넌트 간 상태 동기화를 간소화.- Axios: HTTP 클라이언트로, 백엔드 API와의 통신을 처리.

**5-2. 백엔드**- Spring Boot: 애플리케이션 개발을 간소화하고 빠르게 설정할 수 있는 프레임워크.- Spring Security: 인증 및 권한 부여를 관리하여 높은 수준의 보안을 제공.- Hibernate (JPA): 데이터베이스와 객체 간의 매핑을 제공하여 DB 작업을 간소화.- JWT (JSON Web Token): 클라이언트-서버 간의 안전한 인증을 위해 사용.

**6. 프론트 및 백엔드의 프레임워크 선택 이유**각 프레임워크를 사용한 이유는 다음과 같습니다:

**6-1. 프론트엔드**- Vue.js: 경량의 SPA(Single Page Application)를 빠르고 간편하게 개발 가능  
- Vue Router: 페이지 리로드 없이 라우팅이 가능하여 사용자 경험(UX)을 향상.  
- Vuex: 애플리케이션 상태 관리를 단순화, 복잡한 컴포넌트 간 데이터 흐름 문제 해결.

**6-2. 백엔드**- Spring Boot: 간편한 설정으로 빠르고 재생산이 쉬운 코드 작성- Spring Security: 인증 및 권한 부여를 표준화된 방식으로 처리하여, 취약점을 방지.- Hibernate (JPA): NoSQL 구조를 채택하여 구조 안정성 및 배포용이성 확보- JWT: 세션리스(stateless) 인증을 구현하여 서버의 부하를 줄이고 확장성 향상

**7. face-recognition 모델링 및 관련 api의 사용과 관련하여**- 이번 프로젝트에서 얼굴 인식 모델링 구축은 face-api.js 패키지를 사용  
- 해당 모델에서 제공하는 메서드(matchDimensions, TinyFaceDetectorOptions)등을 활용하여 메서드 호출 등을 용이화  
- 이미지 탐지 및 감정분석에 사용하기 위한 모델은 자체적으로 학습시킨 파일을 적용  
*🡪 좋은 성능을 지닌 GPU가 있는 개발환경에서 학습을 진행한 이후, 모델을 TenserFlow.js 형식으로 변환한 후, 요소의 가중치 정보를 shard1 파일에, vue 및 타 프레임워크와 통신하기 위한 모듈을 json으로 분리하여 저장하는 방식으로 결합을 진행하였습니다.*

**7-1. 얼굴 탐지 모듈  
(tiny\_face\_model-shard1 & tiny\_face\_detector\_model-weights\_manifest.json)**- TenserFlow.js를 사용하여 사람 얼굴의 윤곽을 탐지  
- weights\_manifest.json파일은 탐지된 얼굴의 윤곽을 json파일의 형태로 변환하여 vue와 통신  
- CNN을 사용하여 얼굴을 탐지하며, Depthwise Separable Convolution을 사용해 경량화  
- 이미지의 밝기조절 및 회전 등을 통해 다양한 조명 조건과 각도에도 대응이 가능하도록 학습  
- 실시간 분석이 가능하도록 경량화된 MobileNet 기반의 구조를 채택

**7-2. 표정 인식 모듈  
(face\_expression\_model-shard1 & face\_expression\_model-weights\_manifest.json)**- TenserFlow.js를 사용하여 탐지된 얼굴이 짓고 있는 표정을 탐지  
-아래 링크에서 제공하는 한국인 감정인식을 위한 이미지를 학습시켜 진행  
(https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=82)  
-neutral, sad, surprised, happy, angry, disgusted, scared의 7가지 레이블을 사용  
-WebGL을 활용하여 브라우저 기반 GPU 가속을 사용