**[소프트웨어 공학 Team Project Final Report]**

**Soccer Highlights Film Service**

**Team : ‘문신김홍조’**

**Final Report**

**Software Engineering 2022-2**

**Prof. Seok-Won Lee**



|  |  |
| --- | --- |
| 팀원1 | 201821039 조완식 |
| 팀원2 | 201821190 김지수 |
| 팀원3 | 201820770 문민수 |
| 팀원4 | 201820805 홍성표 |
| 팀원 5 | 201723280 신효원 |

# [Table of contents]

[[Table of contents] 2](#_Toc121094690)

[[Table of Tables] 4](#_Toc121094691)

[[Table of Figures] 7](#_Toc121094692)

[1. Introduction 9](#_Toc121094693)

[1.1. Motivation and Background 9](#_Toc121094694)

[Why you choose such a topic for your project? 9](#_Toc121094695)

[Why it is important to work on this topic? 9](#_Toc121094696)

[1.2. Project Problem Statement 9](#_Toc121094697)

[What problem you are going to solve in your project? 10](#_Toc121094698)

[What will be the impact if you can solve the problem through your project? 10](#_Toc121094699)

[1.3. Scope and Objective 10](#_Toc121094700)

[What is “in the scope” and “out of scope” of your project? 10](#_Toc121094701)

[What is the objective of your project? 11](#_Toc121094702)

[1.4. Proposed Idea 11](#_Toc121094703)

[1.5. Main Features of the System-To-Be 11](#_Toc121094704)

[1.6. Technical approaches 12](#_Toc121094705)

[2. Requirements 13](#_Toc121094706)

[2.1. User Requirements 13](#_Toc121094707)

[2.2. System Requirements 16](#_Toc121094708)

[2.2.1. Functional Requirements 16](#_Toc121094709)

[2.2.2. Non-Functional Requirements 20](#_Toc121094710)

[2.3. Domain Requirements 21](#_Toc121094711)

[2.4. Use-Case Model 21](#_Toc121094712)

[2.4.1. Use-Case Model 22](#_Toc121094713)

[2.4.2. 회원 가입 22](#_Toc121094714)

[2.4.3. 하이라이트 영상 검색 24](#_Toc121094715)

[2.4.4. 하이라이트 영상 시청 25](#_Toc121094716)

[2.4.5. 로그인 27](#_Toc121094717)

[2.4.5. 댓글 작성 29](#_Toc121094718)

[2.4.6. 영상 평가 30](#_Toc121094719)

[2.4.7. 하이라이트 개인 저장 32](#_Toc121094720)

[2.5. Supplementary Requirements 34](#_Toc121094721)

[2.5.1. Interface requirements 34](#_Toc121094722)

[2.5.2. Physical requirements 34](#_Toc121094723)

[2.5.3. Design requirements 34](#_Toc121094724)

[2.5.4. Implementation requirements 34](#_Toc121094725)

[2.6. Requirements Analysis 35](#_Toc121094726)

[2.6.1. overall analysis domain model diagram 35](#_Toc121094727)

[2.6.2 Briefly describe each analysis class 36](#_Toc121094728)

[2.7. Validation Criteria (Metrics & Measure) 37](#_Toc121094729)

[2.7.1 [use-case 1] 회원가입 37](#_Toc121094730)

[2.7.2 [use-case 2] 영상 검색 37](#_Toc121094731)

[2.7.3 [use-case 3] 영상 시청 38](#_Toc121094732)

[2.7.4 [use-case 4] 로그인 38](#_Toc121094733)

[2.7.5 [use-case 5] 댓글 작성 38](#_Toc121094734)

[2.7.6 [use-case 6] 영상 평가 38](#_Toc121094735)

[2.7.7 [use-case 7] 하이라이트 개인 저장 38](#_Toc121094736)

[2.8.8 [use-case 8] 경기 일정 검색 38](#_Toc121094737)

[3. Design Efforts 40](#_Toc121094738)

[3.1. Architecture Design 40](#_Toc121094739)

[3.1.1. System Architecture 40](#_Toc121094740)

[3.1.1. System Description 40](#_Toc121094741)

[3.2. Class Diagram 42](#_Toc121094742)

[3.2.1. Class Diagram 42](#_Toc121094743)

[3.2.2. Class Specification 42](#_Toc121094744)

[3.3. Use-Case Realization 46](#_Toc121094745)

[3.3.1. Whole design sequence diagram 47](#_Toc121094746)

[3.3.2 Use Case 1 Realization. 회원가입 및 로그인 47](#_Toc121094747)

[3.3.3 Use Case 2 Realization. 영상 검색 49](#_Toc121094748)

[3.3.4. Use Case 3 Realization. 영상 시청 50](#_Toc121094749)

[3.3.5. Use Case 4 Realization. 댓글 작성 52](#_Toc121094750)

[3.3.6. Use Case 5 Realization. 영상 평가 53](#_Toc121094751)

[3.3.7. Use Case 6 Realization. 개인 저장 55](#_Toc121094752)

[3.3.8. Use Case 7 Realization. 경기 일정 검색 56](#_Toc121094753)

[4. Implementation 58](#_Toc121094754)

[4.1. Highlight Generator 58](#_Toc121094755)

[4.1.1. Overview 58](#_Toc121094756)

[4.1.2. Dataset 59](#_Toc121094757)

[4.1.3. Audio 60](#_Toc121094758)

[4.1.3. Video 61](#_Toc121094759)

[4.1.5. Action Spotting 61](#_Toc121094760)

[4.1.6. Highlight Generating 61](#_Toc121094761)

[4.2. BackEnd 62](#_Toc121094762)

[4.2.1. Rest API 62](#_Toc121094763)

[4.2.2. Database 63](#_Toc121094764)

[4.2.3. Controller 70](#_Toc121094765)

[4.3. FrontEnd 73](#_Toc121094766)

[4.3.1. overview 74](#_Toc121094767)

[4.3.2. UI 74](#_Toc121094768)

[5. Testing 83](#_Toc121094769)

[5.1. 회원가입 83](#_Toc121094770)

[5.2. 로그인 83](#_Toc121094771)

[5.3. 댓글 작성 84](#_Toc121094772)

[5.4. 댓글 삭제 85](#_Toc121094773)

[5.5. 유저 별 댓글 리스트 호출 86](#_Toc121094774)

[5.6. 북마크 저장 87](#_Toc121094775)

[5.7. 북마크 리스트 호출 88](#_Toc121094776)

[5.9. 영상 평가 88](#_Toc121094777)

[5.10. 경기 일정 89](#_Toc121094778)

[6. Appendix : Contribution 91](#_Toc121094779)

[6.1. Document 91](#_Toc121094780)

[6.2. Project 92](#_Toc121094781)

[Reference. 95](#_Toc121094782)

# [Table of Tables]

[Table 1 - Main Features 12](#_Toc121094783)

[Table 2 : User Requirements 1 13](#_Toc121094784)

[Table 3 : User Requirements 2 13](#_Toc121094785)

[Table 4: User Requirements 3 13](#_Toc121094786)

[Table 5: User Requirements 4 13](#_Toc121094787)

[Table 6: User Requirements 5 14](#_Toc121094788)

[Table 7: User Requirements 6 14](#_Toc121094789)

[Table 8: User Requirements 7 14](#_Toc121094790)

[Table 9: User Requirements 8 15](#_Toc121094791)

[Table 10: User Requirements 9 15](#_Toc121094792)

[Table 11: User Requirements 10 15](#_Toc121094793)

[Table 12: User Requirements 11 15](#_Toc121094794)

[Table 13: User Requirements 12 15](#_Toc121094795)

[Table 14: User Requirements 13 16](#_Toc121094796)

[Table 15: User Requirements 14 16](#_Toc121094797)

[Table 16: User Requirements 15 16](#_Toc121094798)

[Table 17: Functional Requirements 1 16](#_Toc121094799)

[Table 18: Functional Requirements 2 17](#_Toc121094800)

[Table 19: Functional Requirements 3 17](#_Toc121094801)

[Table 20: Functional Requirements 4 17](#_Toc121094802)

[Table 21: Functional Requirements 5 17](#_Toc121094803)

[Table 22: Functional Requirements 6 18](#_Toc121094804)

[Table 23: Functional Requirements 7 18](#_Toc121094805)

[Table 24: Functional Requirements 8 18](#_Toc121094806)

[Table 25: Functional Requirements 9 18](#_Toc121094807)

[Table 26: Functional Requirements 10 19](#_Toc121094808)

[Table 27: Functional Requirements 11 19](#_Toc121094809)

[Table 28: Functional Requirements 12 19](#_Toc121094810)

[Table 29: Functional Requirements 13 19](#_Toc121094811)

[Table 30: Functional Requirements 14 20](#_Toc121094812)

[Table 31: Functional Requirements 15 20](#_Toc121094813)

[Table 32: Functional Requirements 16 20](#_Toc121094814)

[Table 33: Non-Functional Requirements 1 20](#_Toc121094815)

[Table 34: Non-Functional Requirements 2 21](#_Toc121094816)

[Table 35: Non-Functional Requirements 3 21](#_Toc121094817)

[Table 36: Domain Requirements 1 21](#_Toc121094818)

[Table 37: Use-case 1 23](#_Toc121094819)

[Table 38: Use-case 2 25](#_Toc121094820)

[Table 39: Use-case 3 27](#_Toc121094821)

[Table 40: Use-case 4 29](#_Toc121094822)

[Table 41: Use-case 5 30](#_Toc121094823)

[Table 42: Use-case 6 32](#_Toc121094824)

[Table 43: Use-case 7 33](#_Toc121094825)

[Table 44 : User class 36](#_Toc121094826)

[Table 45 : HomePage class 36](#_Toc121094827)

[Table 46 : BookMarkPage class 36](#_Toc121094828)

[Table 47 : PlayHighlightPage class 37](#_Toc121094829)

[Table 48 : AI class 37](#_Toc121094830)

[Table 49 : Class Specification 1 43](#_Toc121094831)

[Table 50 : Class Specification 2 43](#_Toc121094832)

[Table 51 : Class Specification 3 43](#_Toc121094833)

[Table 52 : Class Specification 4 44](#_Toc121094834)

[Table 53 : Class Specification 5 44](#_Toc121094835)

[Table 54 : Class Specification 6 44](#_Toc121094836)

[Table 55 : Class Specification 7 45](#_Toc121094837)

[Table 56 : Class Specification 8 45](#_Toc121094838)

[Table 57 : Class Specification 9 45](#_Toc121094839)

[Table 58 : Class Specification 10 45](#_Toc121094840)

[Table 59 : Class Specification 11 46](#_Toc121094841)

[Table 60 : Class Specification 12 46](#_Toc121094842)

[Table 61 : Classes of highlight video 60](#_Toc121094843)

[Table 62 : Rest API 63](#_Toc121094844)

[Table 63 : User table with Sequelize 65](#_Toc121094845)

[Table 64 : Video table with Sequelize 66](#_Toc121094846)

[Table 65 : Comment table with Sequelize 67](#_Toc121094847)

[Table 66 : Eval table with Sequelize 68](#_Toc121094848)

[Table 67 : Bookmark table with Sequelize 68](#_Toc121094849)

[Table 68 : Schedule table with Sequelize 69](#_Toc121094850)

[Table 69 : Testing for sign up 83](#_Toc121094851)

[Table 70 : Testing for sign in 84](#_Toc121094852)

[Table 71 : Testing for add comment 85](#_Toc121094853)

[Table 72 : Testing for delete comment 86](#_Toc121094854)

[Table 73 : Testing for call the list of comments by user 87](#_Toc121094855)

[Table 74 : Testing for bookmark 88](#_Toc121094856)

[Table 75 : Testing for call the list of bookmark 88](#_Toc121094857)

[Table 76 : Testing for evaluation of video 89](#_Toc121094858)

[Table 77 : Testing for call the specific schedule 90](#_Toc121094859)

# [Table of Figures]

[Figure 1: Use-Case Model 22](#_Toc121094860)

[Figure 2: domain model diagram 35](#_Toc121094861)

[Figure 3 : System Architecture 40](#_Toc121094862)

[Figure 4 : Class Diagram 42](#_Toc121094863)

[Figure 5 : Whole Design Sequence Diagram 47](#_Toc121094864)

[Figure 6 : Use case1 Interaction Diagram 47](#_Toc121094865)

[Figure 7 : Use case1 DCD 48](#_Toc121094866)

[Figure 8 : Use case2 Interaction Diagram 49](#_Toc121094867)

[Figure 9 : Use case2 DCD 50](#_Toc121094868)

[Figure 10 : Use case3 Interaction Diagram 50](#_Toc121094869)

[Figure 11 : Use case3 DCD 51](#_Toc121094870)

[Figure 12 : Use case4 Interaction Diagram 52](#_Toc121094871)

[Figure 13 : Use case4 DCD 52](#_Toc121094872)

[Figure 14 : Use case5 Interaction Diagram 53](#_Toc121094873)

[Figure 15 : Use case5 DCD 54](#_Toc121094874)

[Figure 16 : Use case6 Interaction Diagram 55](#_Toc121094875)

[Figure 17 : Use case6 DCD 56](#_Toc121094876)

[Figure 18 : Use case7 Interaction Diagram 56](#_Toc121094877)

[Figure 19 : Use case7 DCD 57](#_Toc121094878)

[Figure 20 : Highlight Generator AI Architecture 58](#_Toc121094879)

[Figure 21 : Labels of game 59](#_Toc121094880)

[Figure 22 : Database diagram 63](#_Toc121094881)

[Figure 23. UI – 로그인 74](#_Toc121094882)

[Figure 24. UI – 회원가입 75](#_Toc121094883)

[Figure 25. UI – 홈페이지 76](#_Toc121094884)

[Figure 26. UI - 영상 재생 페이지 77](#_Toc121094885)

[Figure 27. UI - 댓글 작성 78](#_Toc121094886)

[Figure 28. UI - 일정 페이지 79](#_Toc121094887)

[Figure 29. UI - 검색 페이지 80](#_Toc121094888)

[Figure 30. UI - 내 정보 페이지(북마크) 81](#_Toc121094889)

[Figure 31. UI - 내 정보 페이지(댓글) 81](#_Toc121094890)

# 1. Introduction

## 1.1. Motivation and Background

### *Why you choose such a topic for your project?*

전 세계에서 매일 수많은 스포츠 경기가 중계되고, 그에 따라 매일 수 천만 명의 사람들이 다양한 종류의 스포츠 영상을 시청한다. 일반적으로 스포츠 영상은 흥미롭고 중요한 부분과 다소 지루하거나 중요하지 않은 부분으로 구성된 매우 긴 길이의 영상이다. 대부분의 시청자들은 가능하다면 영상의 흥미롭고 중요한 부분만을 보는 것을 선호한다. 따라서, 현재 대부분의 스포츠 중계 회사 및 방송국은 경기가 끝난 뒤 해당 경기의 주요 부분만을 편집한 하이라이트 영상을 제공한다.

하지만 현재는 해당 중계사나 방송국들이 직접 경기의 주요 장면을 수작업으로 편집하여 하이라이트 영상을 제작하여 제공한다. 이는 시간이 많이 요구되는 작업이다. 경기의 주요 장면을 빠르고 쉽게 찾아보고 싶은 사용자로서, 경기가 끝났음에도 하이라이트 영상이 편집 과정에 있어 올라오지 않아 기다려야 하는 경험은 매우 불편하고 불쾌한 경험이다. 따라서 빠른 속도로 제작되어 제공할 수 있는 자동 하이라이트 영상 제공 서비스가 있으면 이러한 불편함을 해소할 수 있다고 보고, 해당 주제를 선정하였다.

### *Why it is important to work on this topic?*

인공지능의 도입을 통해 자동적으로 빠르게 하이라이트 영상을 제공할 수 있게 된다면, 제작자 입장에서 현재 이루어지는 고비용의 수작업을 대체할 수 있게 되며 소비자 입장에서도 하이라이트 영상을 오래 기다려야 하는 불편한 경험을 해소할 수 있게 된다. 따라서 해당 하이라이트 영상 제작 분야에서의 사람의 수작업을 대체할 수 있는 인공지능 기술의 도입은 매우 중요한 격동점이라고 할 수 있다.

## 1.2. Project Problem Statement

### *What problem you are going to solve in your project?*

하이라이트 영상은 다양한 스포츠 경기의 내용중 중요하거나 흥미로운 부분을 모아 시청자들에게 짧은 시간안에 경기의 주요 내용을 얻을 수 있는 기회를 제공한다. 이러한 하이라이트 영상은 경기가 끝난 후 각 방송사 또는 제작자들에 의해 제작되는데 영상 편집자는 전체의 게임을 관람하고 일정 판단 기준에 따라 하이라이트라고 생각되는 부분을 정한 뒤, 그 부분만을 편집한다. 이는 시간적으로, 인적으로 비용이 굉장히 많이 발생하는 방법이다. 또한 편집자의 역량에 따라 하이라이트 결과물이 판이하게 달라진다는 단점 또한 존재한다.

따라서 이러한 문제들을 해결하고자 본 프로젝트에서는 비용이 많이 필요한 수동적인 스포츠 하이라이트 영상 제작 및 제공 문제를 AI 기술을 통해 자동으로 빠르게, 일정한 품질의 수준으로 생성하여 제공하는 서비스를 만들어 하이라이트 영상 제작에 대한 비용을 줄이고자 한다.

### *What will be the impact if you can solve the problem through your project?*

본 서비스가 성공적으로 구현 및 제공된다면, 모든 스포츠사 및 방송 / 중계사에서 진행하는 고비용의 수동 하이라이트 제작 작업을 완전히 대체하여 사용자들에게 빠르고 일정한 수준의 품질의 하이라이트 필름을 제공할 수 있게 될 것이다.

## 1.3. Scope and Objective

### *What is “in the scope” and “out of scope” of your project?*

본 프로젝트를 통해 구현하고자 하는 기능의 범위는 ‘스포츠 하이라이트 영상 제작’ 및 그에 따른 ‘영상 제공 서비스 구현’이다. 따라서 AI를 활용한 스포츠 하이라이트 영상 제작 및 웹 페이지를 통한 영상 제공, 검색, 그리고 간단한 게시판 및 커뮤니티 기능만을 구현의 범위로 제한한다.

여타 방송사나 스포츠 채널 등 현존하는 하이라이트 영상 제공을 포함한 비슷한 기능을 제공하는 서비스들에서 일반적으로 추가적인 기능으로 제공되는 다른 영상 추천 알고리즘 구현 및 구체적인 스포츠 수치적 통계 제공 등의 기능은 현 프로젝트의 구현 범위에서는 벗어나는 내용이다.

### *What is the objective of your project?*

AI 기술을 활용한 스포츠 하이라이트 영상 제작 및 웹 페이지를 통한 영상 제공 서비스의 구현.

## 1.4. Proposed Idea

따라서 본 연구에서는 영상 정보와 음성 정보를 통합하여 분석한 인공지능 모델을 제시하고, 구현한다. 훈련 데이터를 통해 미리 학습시킨 본 모델을 통해 매 스포츠 경기가 끝나면 하이라이트 필름을 제작하여 데이터베이스 서버에 저장한다. 웹 페이지의 구현을 통해, 해당 영상들을 적절하게 배치하여 제공한다.

## 1.5. Main Features of the System-To-Be

따라서 본 프로젝트에서는 영상 정보와 음성 정보를 통합하여 분석하는 인공지능 모델을 제시하고, 구현한다. 훈련 데이터를 통해 미리 학습시킨 본 모델을 통해 매 스포츠 경기가 끝나면 하이라이트 필름을 제작하여 데이터베이스 서버에 저장한다. 웹 페이지의 구현을 통해, 해당 영상들을 적절하게 배치하여 제공한다.

|  |  |
| --- | --- |
| Bullet points on the main features | Bullet points on essential high-level requirements |
| * ⋅ 경기가 끝나면 원본 영상을 API를 통해 서버로 받아온다. * ⋅ 사전 학습된 모델을 통해 원본 영상으로부터 하이라이트 영상을 제작한다. * ⋅ 데이터베이스 서버에 해당 하이라이트 영상을 저장한다. * ⋅ 웹 페이지를 통해 하이라이트 영상을 제공한다. * ⋅ 추가적으로 웹 페이지를 통해 유저간 커뮤니티 기능을 제공한다. | ⋅ 하이라이트 영상은 최대한 빠른 속도로 제공되어야 한다.   * ⋅ 하이라이트 영상은 일정한 품질을 가지고 제작되어야 한다. * ⋅ 따라서 시스템이 항상 이러한 요구사항을 충족시킬 수 있도록 모델을 설계하고 구현해야 한다. |

Table 1 - Main Features

## 1.6. Technical approaches

영상 제작 AI 모델은 영상의 음성 데이터로부터GMM(Gaussian Mixture Model)을 통해 특성을 추출한다. 영상 데이터로부터는 R-CNN 또는 masked R-CNN등의 방법을 통해 특성 추출을 진행한다. 그 후 적절한 multi-modal fusion network를 구축한다.

또한 웹 페이지 서비스를 위하여 Node.js (express)를 통해 API를 구현하고, MySQL로 DB를 관리한다. HTML, CSS, Javascript 프레임워크인 React를 이용하여 웹 페이지를 구현한다.

# 2. Requirements

## 2.1. User Requirements

|  |  |
| --- | --- |
| No. | USER\_RS\_01 |
| Title | 유저는 아이디와 비밀번호를 통해 계정을 만들 수 있어야 한다. |
| Description | 유저는 아이디와 비밀번호를 통해 해당 웹 페이지 계정을 만들 수 있다 |

Table 2 : User Requirements 1

|  |  |
| --- | --- |
| No. | USER\_RS\_02 |
| Title | 유저는 아이디와 비밀번호를 통해 로그인할 수 있어야 한다. |
| Description | 유저는 고유한 아이디와 비밀번호를 통해 웹 페이지에 로그인할 수 있다. |

Table 3 : User Requirements 2

|  |  |
| --- | --- |
| No. | USER\_RS\_03 |
| Title | 유저는 영상 목록에서 경기를 선택해 해당 경기에 대한 영상 재생 페이지로 들어갈 수 있어야 한다. |
| Description | 영상 목록은 제목으로 경기의 날짜, 리그 정보, 팀 정보 등을 나타내어 유저가 어떤 경기인지 알 수 있어야 한다. 또한 썸네일 등 이미지 정보 또한 표시되어 유저가 선택하기 쉽도록 한다. |

Table 4: User Requirements 3

|  |  |
| --- | --- |
| No. | USER\_RS\_04 |
| Title | 유저는 영상 목록에서 팀 이름이나 날짜로 영상을 검색할 수 있어야 한다. |
| Description | 유저는 본인이 보고싶은 경기의 하이라이트만 검색해서 볼 수 있어야 한다. |

Table 5: User Requirements 4

|  |  |
| --- | --- |
| No. | USER\_RS\_05 |
| Title | 유저는 영상 재생 페이지에서 해당 경기 영상의 하이라이트 영상을 볼 수 있어야 한다. |
| Description | 유저는 본 서비스의 목적에 따라 주요 장면이라고 생각되는 하이라이트만으로 구성되어 편집된 영상을 볼 수 있어야 한다. |

Table 6: User Requirements 5

|  |  |
| --- | --- |
| No. | USER\_RS\_06 |
| Title | 유저는 영상 재생 페이지에서 해당 경기의 원본 영상의 링크를 제공받을 수 있어야 한다. |
| Description | 유저는 메인 페이지에서 제공되는 하이라이트 영상 말고도 해당 영상의 원본 경기 전체 영상의 링크를 제공받아 원본 전체 경기를 볼 수 있어야 한다. |

Table 7: User Requirements 6

|  |  |
| --- | --- |
| No. | USER\_RS\_07 |
| Title | 유저는 각 하이라이트 영상 마다 추천/비 추천의 형식으로 평가를 내릴 수 있다. |
| Description | 유저는 각 경기마다 자동으로 생성된 하이라이트 영상에 대해서 중요한 순간이라고 생각되는 부분이 포함되지 않았거나, 하이라이트에 포함될 만하지 않은 장면이 포함됐거나, 부자연스럽게 영상이 끊기는 등 자동 생성 모델에 대한 평가를 내릴 수 있다. |

Table 8: User Requirements 7

|  |  |
| --- | --- |
| No. | USER\_RS\_08 |
| Title | 유저는 각 하이라이트 영상 마다 댓글을 작성할 수 있다. |
| Description | 유저는 각 경기 영상 하단에 위치한 댓글 창에 자신의 의견을 표현할 수 있는 댓글을 등록할 수 있다. |

Table 9: User Requirements 8

|  |  |
| --- | --- |
| No. | USER\_RS\_09 |
| Title | 유저는 각 하이라이트 영상 마다 작성한 댓글을 수정 및 삭제가 가능하다. |
| Description | 유저는 각 경기 영상 하단에 위치한 댓글 창에 자신이 작성한 댓글을 수정 및 삭제가 가능해야 한다. |

Table 10: User Requirements 9

|  |  |
| --- | --- |
| No. | USER\_RS\_10 |
| Title | 유저는 하이라이트 영상을 하이라이트 저장 리스트 페이지에서 볼 수 있다. |
| Description | 유저별 개인 하이라이트 저장 리스트에서 저장한 하이라이트의 목록을 볼 수 있고, 리스트의 하이라이트 정보는 메인 페이지에서 제공하는 하이라이트 정보와 동일하다. |

Table 11: User Requirements 10

|  |  |
| --- | --- |
| No. | USER\_RS\_11 |
| Title | 유저는 개인 하이라이트 저장 리스트 페이지에서 찾고 싶은 영상을 검색할 수 있다. |
| Description | 하이라이트 저장 리스트에 저장한 영상 목록들이 많을 경우 검색을 통해서 원하는 하이라이트를 찾을 수 있다. |

Table 12: User Requirements 11

|  |  |
| --- | --- |
| No. | USER\_RS\_12 |
| Title | 유저는 개인 하이라이트 저장 리스트에서 저장된 하이라이트 영상을 선택해 해당 경기에 대한 영상 재생 페이지로 접속할 수 있어야 한다. |
| Description | 유저는 저장된 하이라이트 중에서 원하는 영상을 시청할 수 있다. |

Table 13: User Requirements 12

|  |  |
| --- | --- |
| No. | USER\_RS\_13 |
| Title | 유저는 저장된 개인 하이라이트 영상을 삭제할 수 있다. |
| Description | 유저는 하이라이트를 개인 하이라이트 영상 리스트에 저장할 수 있고, 개인 하이라이트 저장 리스트에서 삭제할 수 있다. |

Table 14: User Requirements 13

|  |  |
| --- | --- |
| No. | USER\_RS\_14 |
| Title | 유저는 경기 일정을 확인 할 수 있다. |
| Description | 유저는 경기 일정 페이지에서 경기 일정을 확인 할 수 있다. |

Table 15: User Requirements 14

|  |  |
| --- | --- |
| No. | USER\_RS\_15 |
| Title | 유저는 경기 일정 페이지에서 팀 이름 및 날짜를 통해 경기 일정을 검색 할 수 있다. |
| Description | 유저는 경기 일정 페이지에서 검색을 통해 특정 팀이나 특정 날짜의 일정을 볼 수 있다.. |

Table 16: User Requirements 15

## 2.2. System Requirements

### 2.2.1. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_01 | Related Requirements | USER\_RS\_01 |
| Title | 시스템은 유저마다 아이디가 중복되지 않도록 해야 한다. | | |
| Description | 시스템은 유저가 아이디 생성 시 아이디가 중복되지 않는 지 확인한다. 아이디가 중복된다면 유저는 계정 생성에 실패한다. | | |

Table 17: Functional Requirements 1

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_02 | Related Requirements | USER\_RS\_01 |
| Title | 시스템은 유저 계정 생성 시 아이디와 비밀번호의 최대 길이를 제한해야 한다. | | |
| Description | 시스템은 유저가 아이디 생성 시 아이디와 비밀번호의 길이가 최대 15가 넘지 않도록 한다 | | |

Table 18: Functional Requirements 2

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_03 | Related Requirements | USER\_RS\_02 |
| Title | 시스템은 유저가 비밀번호 입력 시 비밀번호 값이 보이지 않게 해야 한다. | | |
| Description | 시스템은 보안을 위해 비밀번호를 숨기기 기능으로 표현한다. | | |

Table 19: Functional Requirements 3

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_04 | Related Requirements | USER\_RS\_05 |
| Title | 시스템은 자동으로 축구 경기들에 대해 하이라이트 영상을 제작하고 제공해야 한다. | | |
| Description | 시스템은 축구 경기가 끝나면 그 원본 영상을 바탕으로 AI 모델을 통해 하이라이트 영상을 제작한다. | | |

Table 20: Functional Requirements 4

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_05 | Related Requirements | USER\_RS\_03 |
| Title | 시스템은 제작한 하이라이트 영상들의 목록을 제공해야 한다. | | |
| Description | 영상들의 목록에서 각 영상들은 경기 날짜와 경기한 팀, 리그 정보나 썸네일 등의 형식으로 구분되도록 표기된다. | | |

Table 21: Functional Requirements 5

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_06 | Related Requirements | USER\_RS\_04 |
| Title | 시스템은 제작한 하이라이트 영상들의 목록에서 검색 기능을 제공해야 한다. | | |
| Description | 유저들은 본인이 관심있는 경기만 검색해서 보고 싶으므로 영상 목록에서 필터링 등의 기능이 제공되어 검색될 수 있어야 한다. | | |

Table 22: Functional Requirements 6

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_07 | Related Requirements | USER\_RS\_05,  USER\_RS\_06 |
| Title | 시스템은 홈 페이지의 하이라이트 영상 목록에서의 각 영상 마다 영상 재생 페이지로 연결할 수 있도록 한다. | | |
| Description | 각 경기마다 영상 재생 페이지에서는 하이라이트 영상, 영상 평가, 경기 원본 영상 링크, 댓글 기능이 제공된다. | | |

Table 23: Functional Requirements 7

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_08 | Related Requirements | USER\_RS\_07 |
| Title | 시스템은 영상 평가 시, 횟수를 제한하여 제공해야 한다. | | |
| Description | 하나의 영상에 대한 평가, 즉 추천/비추천 기능은 정확한 평가를 위해 유저 계정 당 1회로 제한하여 제공되어야 한다. | | |

Table 24: Functional Requirements 8

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_09 | Related Requirements | USER\_RS\_09 |
| Title | 시스템은 댓글 수정 및 삭제의 권한은 유저 본인에게만 제공해야 한다. | | |
| Description | 댓글 수정 및 삭제는 유저 자신의 계정이 작성한 댓글에 대해서만 해당 작업이 이루어져야 한다. | | |

Table 25: Functional Requirements 9

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_10 | Related Requirements | USER\_RS\_09 |
| Title | 시스템은 메인 페이지에서 북마크한 하이라이트 리스트를 하이라이트 영상 목록에 보여준다. | | |
| Description | 홈 페이지에서 제공되는 하이라이트 목록과 동일한 형식으로 개인 하이라이트 저장 리스트 페이지에서 제공한다. | | |

Table 26: Functional Requirements 10

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_11 | Related Requirements | USER\_RS\_09 |
| Title | 시스템은 저장된 하이라이트 영상을 한 페이지 당 9개의 하이라이트 씩 페이지 별로 나타낸다. | | |
| Description | 저장된 하이라이트를 한 페이지 당 9개로 나타내고, 저장되는 하이라이트의 개수는 제한이 없다. | | |

Table 27: Functional Requirements 11

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_12 | Related Requirements | USER\_RS\_10 |
| Title | 시스템은 유저가 저장한 개인 저장 하이라이트 목록에서 검색할 수 있는 기능을 제공한다. | | |
| Description | 유저들은 본인이 관심있는 경기만 검색해서 보고 싶으므로 영상 목록에서 필터링 기능이 제공되어 검색될 수 있어야 한다. | | |

Table 28: Functional Requirements 12

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_13 | Related Requirements | USER\_RS\_11 |
| Title | 시스템은 하이라이트 개인 저장 페이지에서 목록의 각 영상 마다 영상 재생 페이지로 연결할 수 있도록 한다. | | |
| Description | 유저가 저장된 하이라이트 리스트 중에 하나를 선택하면, 하이라이트 재생 페이지로 연결되어 영상을 시청할 수 있도록 한다. | | |

Table 29: Functional Requirements 13

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_14 | Related Requirements | USER\_RS\_12 |
| Title | 시스템은 유저가 저장된 하이라이트 영상의 저장을 해제하면 개인 하이라이트 영상 리스트에서 삭제하도록 한다. | | |
| Description | 저장 취소 시 하이라이트 영상 삭제 시점은 새로 고침 했을 경우이다. 저장 취소를 한 뒤, 다시 저장하고 싶은 경우를 대비해서 저장 취소 시, 저장 아이콘 표시로만 삭제 되었음을 표시한다. | | |

Table 30: Functional Requirements 14

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_15 | Related Requirements | USER\_RS\_14 |
| Title | 시스템은 현재 시간을 기준으로 끝이 난 경기는 일정 페이지에서 표시되지 않도록 해야 한다. | | |
| Description | 새로 고침을 한 시각을 기준으로 진행중이거나 진행 예정인 경기를 제외한 끝이 난 경기는 일정에서 삭제해 보여지지 않도록 한다. | | |

Table 31: Functional Requirements 15

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_RS\_16 | Related Requirements | USER\_RS\_15 |
| Title | 시스템은 경기 일정 목록에서 검색 할 수 있는 기능을 제공한다. | | |
| Description | 유저는 검색할 축구팀을 입력하여 원하는 축구팀의 일정을 확인 할 수 있어야 한다. | | |

Table 32: Functional Requirements 16

### 2.2.2. Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_NFR\_1 | Related Requirements | USER\_RS\_03,  USER\_RS\_10,  USER\_RS\_14, SYS\_RS\_05,  SYS\_RS\_07,  SYS\_RS\_10,  SYS\_RS\_13, |
| Title | 각 인터페이스(페이지)간 전환은 1초 내에 이루어져야 한다. | | |
| Description | 유저의 사용자 경험을 위하 각 페이지 간 전환은 빨리 이루어져야 한다. | | |

Table 33: Non-Functional Requirements 1

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_NFR\_2 | Related Requirements |  |
| Title | 웹 페이지 서비스는 브라우저에 상관 없이 동작하여야 한다 | | |
| Description | 다양한 환경에서 작동할 수 있도록 접속한 인터넷 브라우저에 상관 없이 서비스가 동작하여야 한다. | | |

Table 34: Non-Functional Requirements 2

|  |  |  |  |
| --- | --- | --- | --- |
| No. | SYS\_NFR\_3 | Related Requirements | USER\_RS\_05, SYS\_RS\_04 |
| Title | 제작된 하이라이트 영상은 사용자가 재생 요청을 했을 때로부터 5초 내에 재생 시작되어야 한다. | | |
| Description | 유저의 사용자 경험을 위해 영상 재생 시작은 최대한 빠른 시간 내에 이루어져야 한다. | | |

Table 35: Non-Functional Requirements 3

## 2.3. Domain Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DM\_RS\_01 | Related Requirements | USER\_RS\_05, SYS\_RS\_04 |
| Title | 시스템은 하이라이트 영상 제작 AI 모델을 제공해야 한다. | | |
| Description | SoccerNet V2 데이터를 통해 사전 훈련된 Transformer-based DL 모델을 구축하고, 서버에 모델을 저장해 서비스에 제공하여 사용한다. | | |

Table 36: Domain Requirements 1

## 2.4. Use-Case Model

### 2.4.1. Use-Case Model

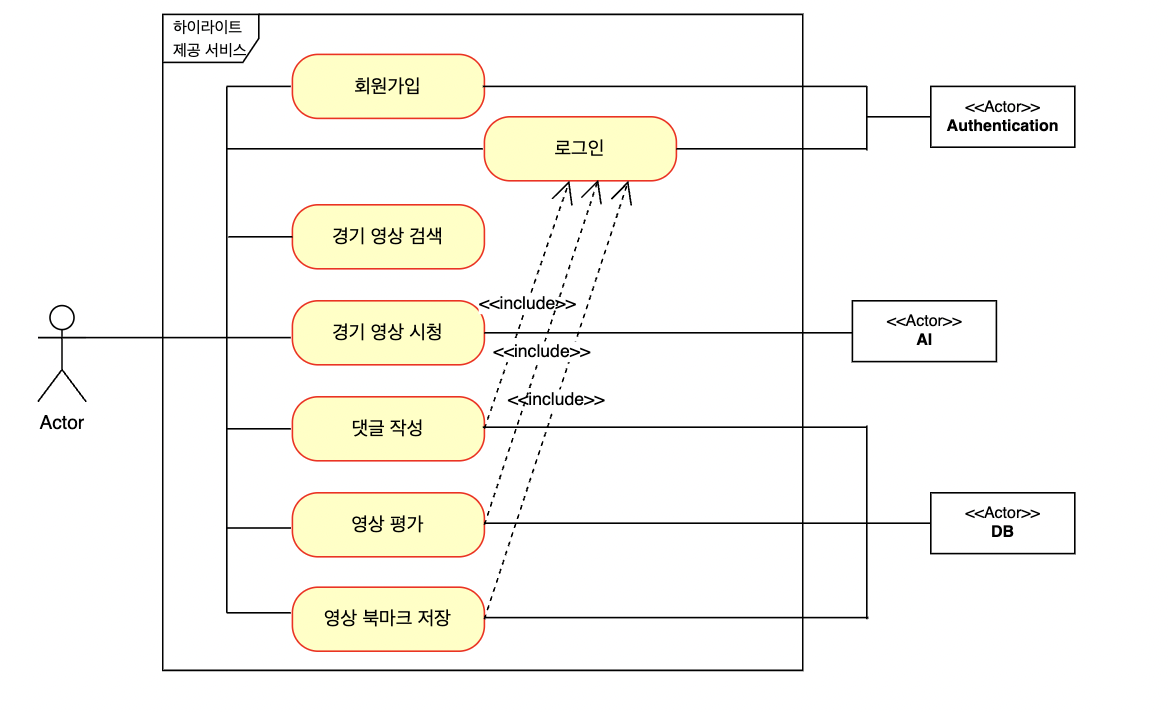


Figure 1: Use-Case Model

### 2.4.2. 회원 가입

|  |  |
| --- | --- |
| Use-Case | 1 |
| Case Name | 회원 가입 |
| Actors | User, System |
| Description | 유저마다 고유의 아이디 및 비밀번호를 통해 계정을 만들 수 있다. |
| Pre-conditions | 1. 유저는 길이가 15가 넘지 않는 아이디와 비밀번호를 입력해야 한다.  2. 유저의 아이디는 이미 존재하는 아이디와 겹치지 않아야 한다. |
| Post-conditions | 1. 유저가 가입하여 생성된 계정은 탈퇴할 수 있다. |
| Primary Flow | 1. 로그인 페이지에 접속한다.  2. 회원 가입 창을 보여준다.  3. 회원 가입할 아이디와 비밀번호를 입력한다.  4.Homepage에 접속한다 |
| Alternate Flow | 3a. 만약 유저가 입력한 아이디가 다른 유저와 중복된 경우  3a.1.아이디가 중복되었다는 메시지를 띄워준다. |
| Activity Diagram |  |

Table 37: Use-case 1

### 2.4.3. 하이라이트 영상 검색

|  |  |
| --- | --- |
| Use-Case | 2 |
| Case Name | 하이라이트 영상 검색 |
| Actors | User |
| Description | 자동으로 생성된 하이라이트 영상 목록 중 원하는 영상을 검색하여 필터링 할 수 있다. |
| Pre-conditions | 1. 유저는 홈페이지에 있어야 한다.  2. 유저는 팀 명, 경기 날짜 등으로 검색어를 넣어주어야 한다. |
| Post-conditions | 1.시스템은 유저가 입력한 검색어에 맞는 하이라이트 영상들만 필터링하여 목록을 재구성해 보여주어야 한다. |
| Primary Flow | 1. 유저는 홈페이지에 접속하면 만들어진 모든 경기의 하이라이트 영상 목록을 볼 수 있다.  2. 유저는 해당 홈페이지에서 팀 명, 경기 날짜를 통해 검색을 할 수 있다.  3. 시스템은 유저의 검색어에 맞게 필터링 된 하이라이트 영상들만으로 이루어진 목록을 재구성해 보여준다. |
| Alternate Flow | 2a.만약 유저가 검색한 검색 조건에 부합하는 영상이 존재하지 않을 경우,  2a.1.검색 조건에 맞는 영상이 존재하지 않는다는 메세지를 띄워준다. |
| Activity Diagram |  |

Table 38: Use-case 2

### 2.4.4. 하이라이트 영상 시청

|  |  |
| --- | --- |
| Use-Case | 3 |
| Case Name | 하이라이트 영상 시청 |
| Actors | User, Highlight Service |
| Description | 하이라이트 영상 목록 중 하나를 선택하여 하이라이트 영상을 시청할 수 있다. |
| Pre-conditions | 1. 유저는 하이라이트 영상 목록 중 보고싶은 하이라이트 영상을 선택해야 한다.  2. 시스템은 해당 경기에 대한 하이라이트 영상을 제작하여 제공해야 한다 |
| Post-conditions | 1.유저는 유저가 선택한 하이라이트 영상 재생 페이지로 접속할 수 있다. |
| Primary Flow | 1. 유저는 홈페이지, 개인 하이라이트 저장 페이지, 또는 검색 결과 페이지에서 하이라이트 영상 목록을 확인한다.  2. 유저는 하이라이트 영상 목록 중 보고싶은 영상을 선택한다.  3. 시스템은 해당 하이라이트 영상이 포함된 메인 페이지를 제공한다.  4. 유저는 해당 영상 재생 페이지에서 하이라이트 영상을 시청할 수 있다. |
| Alternate Flow | 4a. 유저는 해당 영상 재생 페이지에서 영상을 정지할 수 있다. |
| Activity Diagram |  |

Table 39: Use-case 3

### 2.4.5. 로그인

|  |  |
| --- | --- |
| Use-Case | 4 |
| Case Name | 로그인 |
| Actors | User, Authentication |
| Description | 유저는 아이디와 비밀번호를 통해 로그인 할 수 있다. |
| Pre-conditions | 1. 유저는 로그인 페이지에 있어야 한다.  2. 유저는 자신의 아이디 및 비밀번호를 입력해야 한다 |
| Post-conditions | 1.시스템은 유저가 입력한 아이디와 비밀번호가 유저 데이터 목록에 존재하는 지 확인하고 그에 대한 결과를 보여주어야 한다. |
| Primary Flow | 1.유저는 로그인 페이지에 접속한다.  2. 유저는 자신의 아이디와 비밀번호를 입력한다  3. 로그인 된 계정으로 홈페이지에 접속 할 수 있다. |
| Alternate Flow | 2a 만약 유저가 입력한 아이디와 비밀번호가 일치하지 않는 경우  2a.1. 아이디와 비밀번호가 일치하지 않는다는 메시지를 띄워준다. |
| Activity Diagram |  |

Table 40: Use-case 4

### 2.4.5. 댓글 작성

|  |  |
| --- | --- |
| Use-Case | 5 |
| Case Name | 댓글 작성 |
| Actors | User, DB |
| Description | 유저는 각 경기 영상 하단에 위치한 댓글 창에 자신의 의견을 표현할 수 있는 댓글을 등록할 수 있다. |
| Pre-conditions | 1. 유저는 회원가입을 통해 생성한 자신의 계정으로 로그인이 되어 있어야 한다.  2. 시스템은 유저가 댓글을 작성할 수 있는 댓글 창을 각 경기 영상 하단에 제공해야 한다. |
| Post-conditions | 1.유저가 작성한 댓글이 등록되어 댓글 목록에서 볼 수 있어야 한다. |
| Primary Flow | 1. 유저는 자신의 계정으로 로그인 한다.  2. 유저는 영상 하단에 위치한 댓글 창에 댓글을 입력한다.  3. 유저는 댓글 작성을 마치면 등록 버튼을 통해 댓글을 등록한다.  4. 유저는 댓글 목록에서 등록된 댓글들을 확인할 수 있다. |
| Alternate Flow | 1a. 만약 유저가 로그인을 하지 않았을 경우  1a.1. 댓글 등록 시, 로그인을 요구하는 메시지를 띄워준다. |
| Activity Diagram |  |

Table 41: Use-case 5

### 2.4.6. 영상 평가

|  |  |
| --- | --- |
| Use-Case | 6 |
| Case Name | 영상 평가 |
| Actors | User, DB |
| Description | 유저는 각 하이라이트 영상 마다 추천/비 추천의 형식으로 평가를 할 수 있다. |
| Pre-conditions | 1. 유저는 회원가입을 통해 생성한 자신의 계정으로 로그인이 되어 있어야 한다.  2. 시스템은 유저가 영상을 평가할 수 있는 추천/비 추천 버튼을 각 경기 영상 하단에 구성해야 한다. |
| Post-conditions | 1. 유저는 평가를 수정할 수 있다. |
| Primary Flow | 1. 유저는 자신의 계정으로 로그인 한다.  2. 유저는 영상 하단에 위치한 추천/비 추천 버튼을 통해 영상을 평가한다.  3. 유저는 영상 평가를 마치면 해당 버튼 클릭의 유무를 통해 자신의 평가 기록을 확인이 가능하다. |
| Alternate Flow | 1a. 만약 유저가 로그인을 하지 않았을 경우  1a.1. 영상 평가 버튼 클릭 시, 로그인을 요구하는 메시지를 띄워준다. |
| Activity Diagram |  |

Table 42: Use-case 6

### 2.4.7. 하이라이트 개인 저장

|  |  |
| --- | --- |
| Use-Case | 7 |
| Case Name | 하이라이트 개인 저장 |
| Actors | User, DB |
| Description | 유저는 저장하고 싶은 하이라이트를 홈페이지에서 저장하여, 하이라이트 개인 저장 페이지에서 저장 리스트를 모아 볼 수 있다. |
| Pre-conditions | 1. 유저는 회원가입을 통해 생성한 자신의 계정으로 로그인이 되어 있어야 한다.  2. 유저는 홈페이지에서 저장하고 싶은 하이라이트를 저장해야 하이라이트 개인 저장 페이지에서 볼 수 있다. |
| Post-conditions | 1. 유저는 저장한 하이라이트를 삭제할 수 있어야 한다.  2. 저장한 하이라이트를 선택하면 하이라이트 시청을 할 수 있다. |
| Primary Flow | 1. 유저는 자신의 계정으로 로그인 한다.  2. 유저는 홈페이지에서 원하는 하이라이트를 저장한다.  3. 유저는 하이라이트 개인 저장 페이지에 접속하여 보고 싶은 하이라이트 영상을 선택한다.  4. 유저는 메인 페이지로 연결되어 하이라이트 영상을 시청한다. |
| Alternate Flow | 2a. 만약 유저가 로그인을 하지 않았을 경우  2a.1 저장 버튼을 클릭 할 시, 로그인을 요구하는 메세지를 띄워준다.  3a. 만약 유저가 로그인을 하지 않았을 경우  3a.1하이라이트 개인 저장 페이지로 접근할 시, 로그인을 요구하는 메시지를 띄워준다.  3b. 만약 저장한 하이라이트가 없을 경우  3b.1 저장한 것이 없다는 메세지를 띄워준다. |
| Activity Diagram |  |

Table 43: Use-case 7

## 2.5. Supplementary Requirements

### 2.5.1. Interface requirements

웹 페이지를 통해 서비스를 운영한다.

유저 인터페이스는 회원가입/로그인, 하이라이트 리스트 홈페이지, 영상 시청 및 댓글 페이지, 하이라이트 개인 저장 페이지, 축구 일정 페이지로 총 5가지 메뉴로 구성된다.

### 2.5.2. Physical requirements

시스템은 AI 모델 구동을 위한 GPU(CUDA)를 사용 가능한 서버가 요구된다.

### 2.5.3. Design requirements

React 기반의 웹 페이지가 프론트 엔드로 사용된다. 서비스에 필요한 AI 모델은 다른 GPU를 이용 가능한 서버에서 미리 학습시켜 모델 파라미터 만을 DB에 저장해서 사용한다. 기타 유저 정보나 게시글 정보 등 데이터도 DB에 저장하여 사용한다. 노드(Express)를 사용하여 백엔드를 구성한다.

기존 영상 링크로부터 원본 영상을 받아와 서버에 저장되어 있는 DL모델을 통해 하이라이트 필름을 제작하고 DB에 저장한다. 프론트에서 유저의 요청에 따라 백엔드 서버에서는 제작된 영상을 포함한 각종 데이터를 프론트를 통해 서비스한다.

### 2.5.4. Implementation requirements

AI – Pytorch Transformer VIVIT,

Back End – Node.js Express

Front Ed – React

Data Base – MySQL

## 2.6. Requirements Analysis

### 2.6.1. overall analysis domain model diagram

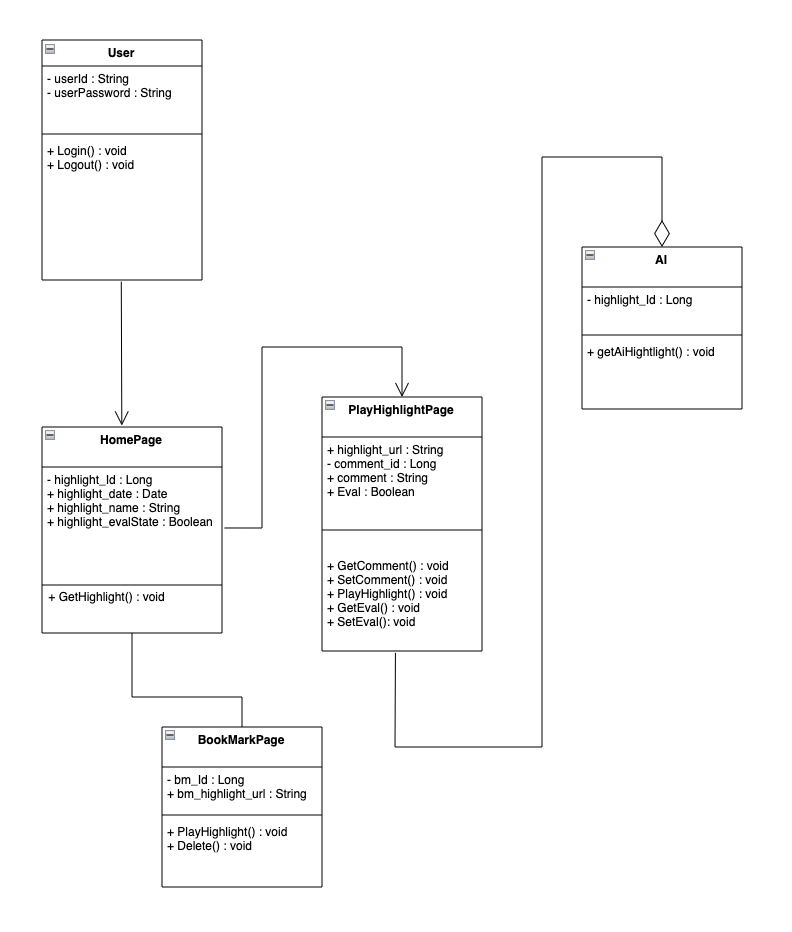


Figure 2: domain model diagram

### 2.6.2 Briefly describe each analysis class

|  |  |
| --- | --- |
| Class Name | User |
| Description | 유저는 ID와 Password로 로그인과 로그아웃을 한다. |
| Relation | 계정 로그인 후 영상이 있는 Homepage와 연관관계가 있다. |
| Attribute | useId : String, usePassword : String |

Table 44 : User class

|  |  |
| --- | --- |
| Class Name | HomePage |
| Description | HomePage에 게시된 highlight영상은 게시 번호, 게시 생성된 날짜, 게시물 제목, 추천,비 추천 여부를 표시한다. BookMarkPage 추가여부에 따라서 Homepage에 highlight영상에 속한 북마크 표시 여부가 달라진다.  PlayHighlightPage 에 속한 highlight url영상은 비/추천 표시 여부에 따라서 표시가 달라진다. highlight영상은 PlayHighlightPage와 주고 받는다. |
| Relation | PlayHighlightPage와 BookMarkPage에 연관 관계가 있다. |
| Attribute | highlight\_Id : Long, highlight\_date : Date, highlight\_name :String, highlight\_evalState : Boolean |

Table 45 : HomePage class

|  |  |
| --- | --- |
| Class Name | BookMarkPage |
| Description | BookMarkPage는 유저가 북마크 저장을 한 하이라이트 영상을 리스트를 볼 수 있다. 가지고 있어서 저장된 북마크를 표시 여부가 Homepage와 동시에 달라진다. |
| Relation | BookMarkPage는 Homepage와 연관 관계를 가진다. |
| Attribute | bm\_Id : Long , bm\_hight\_url : String |

Table 46 : BookMarkPage class

|  |  |
| --- | --- |
| Class Name | PlayHighlightPage |
| Description | PlayHighlightPage에서 영상을 재생할 수 있다. 영상에 달린 댓글 내용과 댓글 번호 포함하여 볼 수 있다. 사용자의 추천 여부를 알 수 있게 True라면 추천 False라면 비추천으로 설정 한다. 공백으로 Null을 둔다. PlayHighlightPage는 AI로부터 highlight 영상을 받아온다. |
| Relation | PlayHighlightPage는 AI와 연관관계를 가진다 |
| Attribute | highlight\_url : String, comment\_id : Long , comment : String, Eval : Boolean |

Table 47 : PlayHighlightPage class

|  |  |
| --- | --- |
| Class Name | AI |
| Description | AI모델에 적용 후 생성한 영상이Highlight\_Id에 추가되어 만들어진 영상을 PlayHighlightPage제공한다.  AI로 만들어진 Highlight 영상 가져와PlayHighlightPage에 넣어준다. |
| Relation | AI는 PlayHighlightPage와 연관관계를 가진다 |
| Attribute | Hightlight\_Id : Long |

Table 48 : AI class

## 2.7. Validation Criteria (Metrics & Measure)

### 2.7.1 [use-case 1] 회원가입

유저의 아이디는 이미 존재하는 사이트의 다른 유저의 아이디와 중복되지 않아야 한다. 또한 유저의 아이디와 비밀번호는 최대 15자를 넘지 않아야 한다.

### 2.7.2 [use-case 2] 영상 검색

유저는 팀 이름 또는 날짜를 통해 본인이 보고싶은 경기의 하이라이트를 검색할 수 있다. 두 조건 모두 적용한다면 정확히 한 경기를 얻을 수 있을 것이고, 유저는 두 조건 중 한 조건만을 적용할 수도 있다. 시스템은 팀 이름을 경기 영상에서 적절히 파싱하여(대/ 소문자, 공백 처리 등) 조건에 부합하는지 확인해야 한다. 경기 날짜 역시 단순 text가 아닌 Date Picker등을 적용하여 정확히 날짜로 검색할 수 있도록 한다.

### 2.7.3 [use-case 3] 영상 시청

유저는 시스템에서 제공하는 AI에 의해 생산된 하이라이트 영상을 시청한다. 유저는 웹 페이지의 동영상 표시 형식에 맞게 정지, 재생, 뒤로 감기, 앞으로 감기, 선택한 부분 시청하기 기능을 이용할 수 있어야 한다.

### 2.7.4 [use-case 4] 로그인

시스템은 유저가 입력한 아이디와 비밀번호를 검증하여 로그인 기능을 제공한다. 유저가 입력한 아이디와 비밀번호가 맞지 않는다면 잘못된 비밀번호라고 알린다. 유저가 입력한 아이디와 비밀번호가 맞는다면 유저에게 로그인 된 유저 정보 및 권한을 부여한다.

### 2.7.5 [use-case 5] 댓글 작성

시스템은 유저가 로그인 되어있는지 확인한다. 확인된 유저는 영상에 대해 댓글을 작성할 수 있다. 유저가 영상에 대해 댓글을 작성하면 유저의 이름과 댓글 내용을 통해 댓글을 표시한다. 댓글의 총 길이는 120자로 제한한다. 한 유저가 연속으로 3번 이상의 댓글을 작성하지 못하도록 제한한다. 댓글을 작성한 당사자에게만 댓글을 삭제할 수 있는 권한을 부여한다.

### 2.7.6 [use-case 6] 영상 평가

시스템은 유저가 로그인 되어있는지 확인한다. 시스템은 유저가 해당 영상을 한 번이라도 재생했는지 확인한다. 확인된 유저는 영상을 시청하고 해당 영상에 대해 추천 또는 비 추천을 통해 평가를 할 수 있다. 유저는 한 영상에 대해서 한 번만 평가할 수 있다. 이미 내린 평가에 대해서 취소 또는 수정을 할 수 있다.

### 2.7.7 [use-case 7] 하이라이트 개인 저장

시스템은 유저가 로그인 되어있는지 확인한다.

### 2.8.8 [use-case 8] 경기 일정 검색

유저는 팀 이름 또는 날짜를 통해 경기 일정을 검색할 수 있다. 팀 이름을 통해 검색한다면 해당 팀의 과거 및 미래의 경기 일정에 대해 표시한다. 날짜를 통해 검색한다면 해당 날짜의 경기 일정에 대해 표시한다. 시스템은 팀 이름을 경기 영상에서 적절히 파싱하여(대/ 소문자, 공백 처리 등) 조건에 부합하는지 확인해야 한다. 경기 날짜 역시 단순 text가 아닌 Date Picker등을 적용하여 정확히 날짜로 검색할 수 있도록 한다.

# 3. Design Efforts

## 3.1. Architecture Design

### 3.1.1. System Architecture

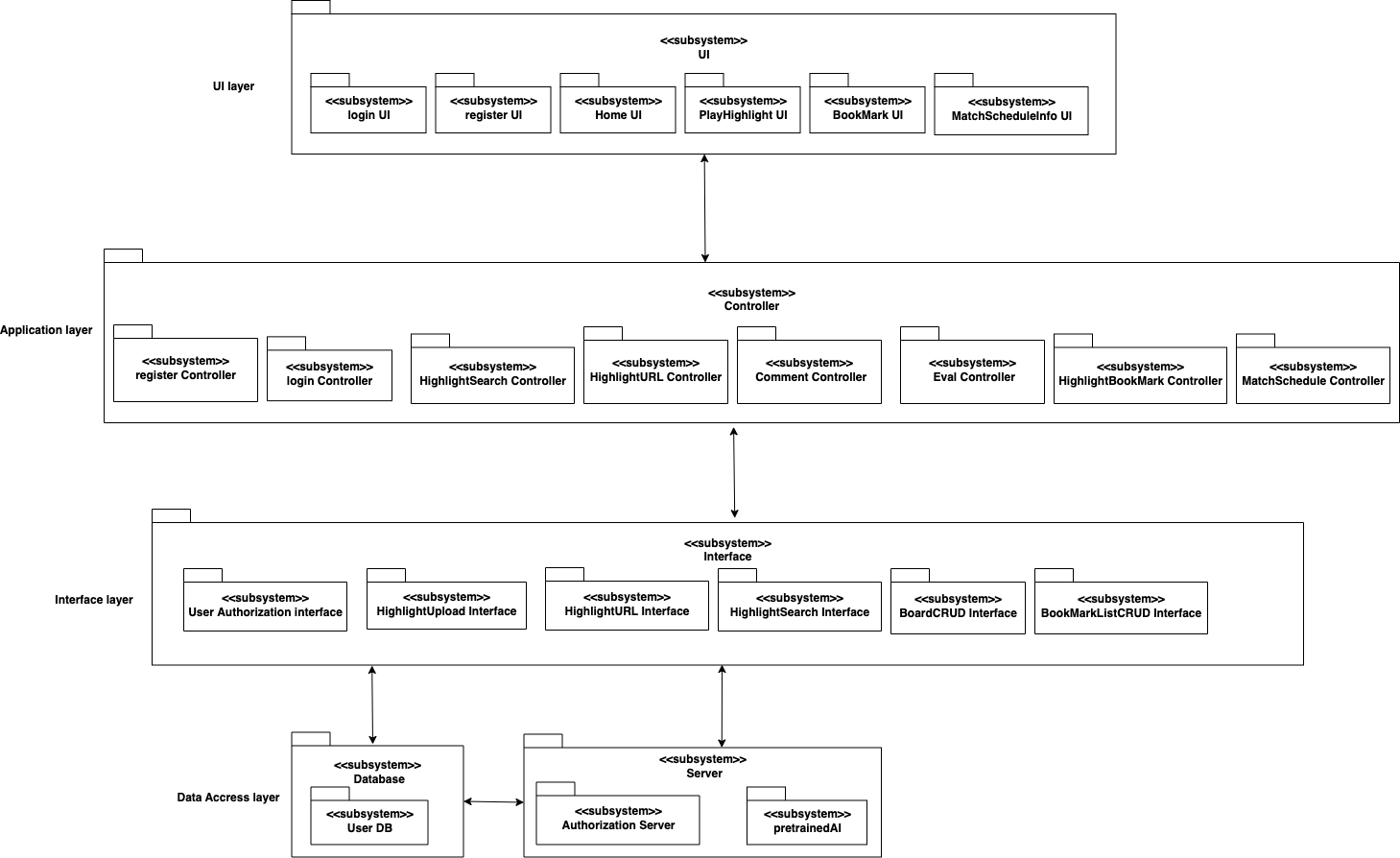


Figure 3 : System Architecture

### 3.1.1. System Description

본 시스템은 UI layer, Application layer, Interface layer, Data Access layer로 구성 된다.

UI layer는 브라우저 기반 사용자 인터페이스이다.

UI layer는 register UI(회원가입), login UI(로그인), home UI(하이라이트목록 리스트 및 검색), PlayHighlight UI(영상시청 및 댓글,하이라이트 저장), BookMark UI(하이라이트 개인 저장) , 일정 UI(축구 일정 페이지)로 구성 된다.  
Application layer는 사용자 인터페이스를 관리한다.  
웹 브라우저를 통해 전달되는 사용자 인터페이스 기능을 제공한다. 이것은 사용자가 시스템에 회원가입, 로그인 할 수 있게 해주는 컴포넌트들과 사용자 작업에 해당하는 검색, 영상평가, 영상 댓글, 영상 저장 컴포넌트를 포함한다. 그리고 사용자는 시스템에 이용하는 영상 시청, 영상 일정 검색 컴포넌트를 가진다.

Interface layer는 비즈니스 로직을 수행한다. 애플리케이션 기능, 시스템 공통 기능 컴포넌트를 포함한다. 시스템 기능을 구현으로 사용자 인증, 하이라이트 영상, 댓글 작성,수정,삭제 와 추천 여부 표시, 영상 저장 및 삭제를 구현하는 컴포넌트를 포함한다 처리된 데이터는 Data Access layer와 상호작용한다. Data Access layer에 정보를 입력과 요청 및 데이터 출력을 한다.

Data Access layer는 DB, 시스템 지원 소프트웨어를 포함한다.데이터 영구저장 및 요청 데이터를 전송한다

시스템의 기능은 분리된 계층들로 구성되고 각 계층은 바로 아래 층에서 제공하는 기능과 서비스에만 의존한다. 계층적인 접근은 시스템의 점증적 개발을 지원한다. 계층이 계발될 때 그 계층에서 제공되는 서비스의 일부는 사용자에게 제공 될 수 있다. 이 아키텍처는 변경 가능하고 이식 가능하다. 만일 인터페이스가 변경되지 않는다면 시스템의 다른 부분을 변경하지 않고 확장된 기능을 가지는 새로운 계층으로 기존의 계층을 대체 할 수 있다. 또한 계층 인터페이스가 변경되거나 계층에 새로운 기능이 추가되었을 때 인접한 계층만 영향을 받는다. 즉, 인접한 계층은 상호작용 한다.

## 3.2. Class Diagram

### 3.2.1. Class Diagram

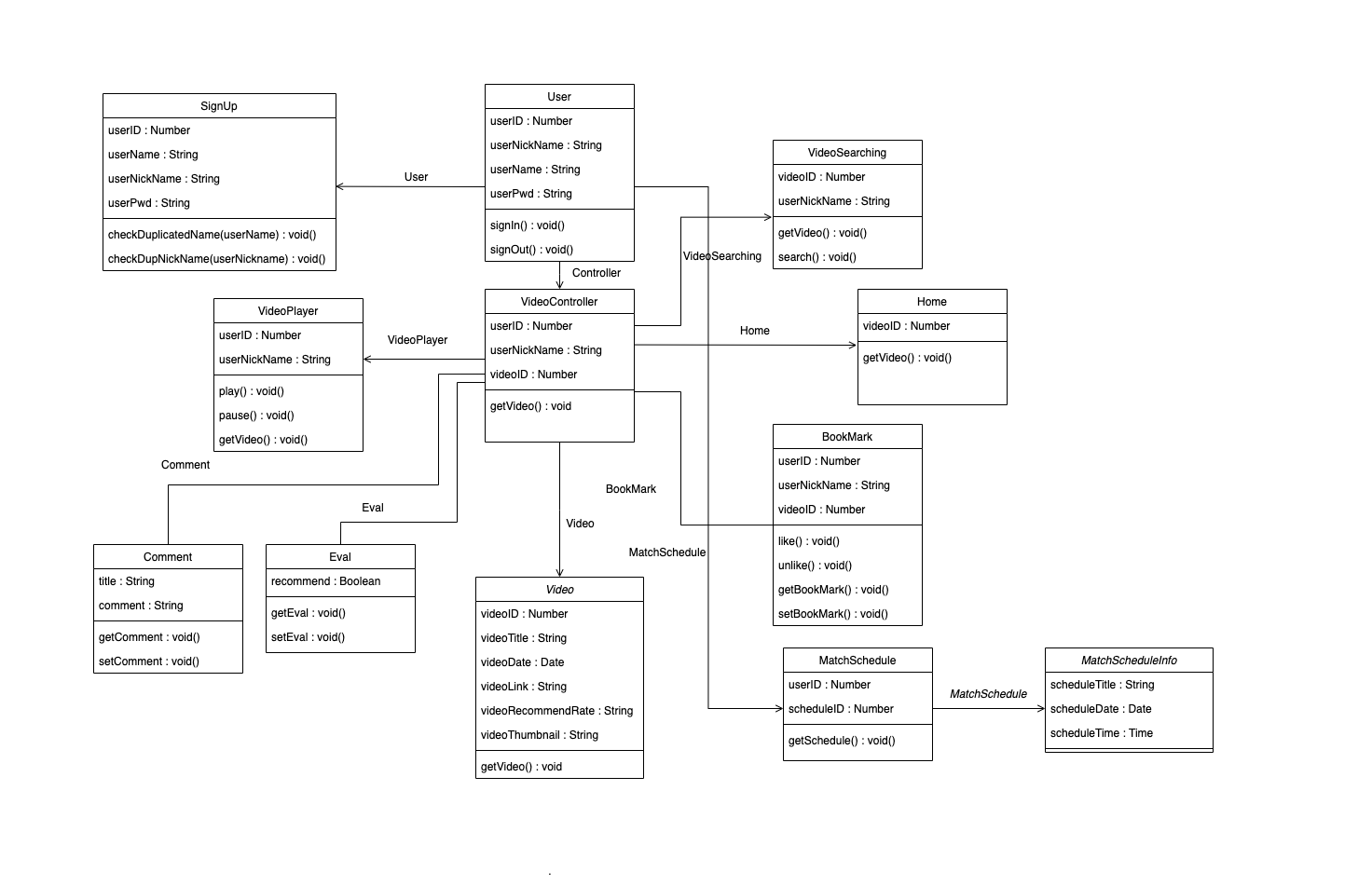


Figure 4 : Class Diagram

### 3.2.2. Class Specification

|  |  |  |
| --- | --- | --- |
| SignUp | | |
| Type | Data | Descriptions |
| Number | userID | User 식별하는 ID. |
| String | username | 로그인에 필요한 User ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| String | userPwd | 로그인에 필요한 User Pas. |
| Type | Data | Descriptions |
| void | checkDuplicateId(userID) | User의 아이디가 중복되는지 확인. |
| void | checkDupNickName(userNickname) | User의 닉네임이 중복되는지 확인. |

Table 49 : Class Specification 1

|  |  |  |
| --- | --- | --- |
| User | | |
| Type | Data | Descriptions |
| Number | userID | User를 식별하는 ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| Type | Data | Descriptions |
| void | login() | 로그인한다. |
| void | logout() | 로그아웃한다. |

Table 50 : Class Specification 2

|  |  |  |
| --- | --- | --- |
| VideoController | | |
| Type | Data | Descriptions |
| Number | userID | User를 식별하는 ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| Number | videoID | Video를 식별하는 ID. |
| Type | Data | Descriptions |
| void | getVideo() | Video 정보들을 가져옴. |

Table 51 : Class Specification 3

|  |  |  |
| --- | --- | --- |
| Home | | |
| Type | Data | Descriptions |
| Number | videoID | Video를 식별하는 ID. |
| Type | Data | Descriptions |
| void | getVideo() | Video 정보들을 가져옴. |

Table 52 : Class Specification 4

|  |  |  |
| --- | --- | --- |
| Video | | |
| Type | Data | Descriptions |
| Number | videoID | Video를 식별하는 ID. |
| String | videoTitle | Video의 제목. |
| Data | videoDate | Video영상날짜. |
| String | videoLink | Video 원본 링크. |
| String | videoRecommandRate | Video 추천 수. |
| String | videoThumbnail | Video 썸네일. |
| Type | Data | Descriptions |
| void | getVideo() | Video 정보들을 가져옴. |

Table 53 : Class Specification 5

|  |  |  |
| --- | --- | --- |
| VideoSearching | | |
| Type | Data | Descriptions |
| Number | videoID | Video를 식별하는 ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| Type | Data | Descriptions |
| void | getVideo() | Video 정보들을 가져옴. |

Table 54 : Class Specification 6

|  |  |  |
| --- | --- | --- |
| VideoPlayer | | |
| Type | Data | Descriptions |
| Number | videoID | Video를 식별하는 ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| Type | Data | Descriptions |
| void | play() | Video를 재생한다. |
| void | pause() | Video를 멈춘다. |
| void | getVideo() | Video 정보들을 가져옴. |

Table 55 : Class Specification 7

|  |  |  |
| --- | --- | --- |
| Comment | | |
| Type | Data | Descriptions |
| String | title | Video 댓글 제목. |
| String | comment | Video댓글 내용. |
| Type | Data | Descriptions |
| void | getComment() | 댓글을 가져온다. |
| void | setComment() | 댓글을 생성한다. |

Table 56 : Class Specification 8

|  |  |  |
| --- | --- | --- |
| Eval | | |
| Type | Data | Descriptions |
| Boolean | recommend | Video의 추천 |
| Type | Data | Descriptions |
| void | getEval() | 추천을 가져온다. |
| void | setEval() | 추천을 생성한다. |

Table 57 : Class Specification 9

|  |  |  |
| --- | --- | --- |
| BookMark | | |
| Type | Data | Descriptions |
| Number | userID | User를 식별하는 ID. |
| String | userNickName | User가 웹 페이지에서 사용하는 닉네임. |
| Number | videoID | Video를 식별하는 ID. |
| Type | Data | Descriptions |
| void | like() | 영상을 북마크한다. |
| void | unlike() | 영상 북마크를 취소한다. |
| void | getLikeVideo() | 북마크를 가져온다. |
| void | setLikeVideo() | 북마크를 생성한다. |

Table 58 : Class Specification 10

|  |  |  |
| --- | --- | --- |
| MatchSchedule | | |
| Type | Data | Descriptions |
| Number | userID | User를 식별하는 ID. |
| Number | scheduleID | 경기 일정을 식별하는 ID |
| Type | Data | Descriptions |
| void | getSchedule() | 경기 일정을 가져온다 |

Table 59 : Class Specification 11

|  |  |  |
| --- | --- | --- |
| MatchScheduleInfo | | |
| Type | Data | Descriptions |
| String | scheduleTitle | 경기 일정의 제목. |
| Date | scheduleDate | 경기 일정의 날짜 |
| Time | scheduleTime | 경기 일정 시간대 |

Table 60 : Class Specification 12

## 3.3. Use-Case Realization

### 3.3.1. Whole design sequence diagram

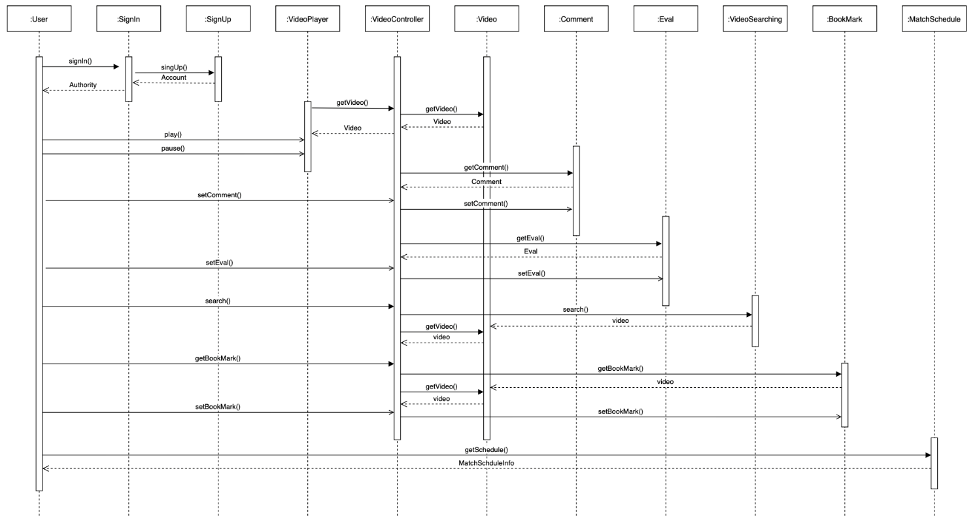


Figure 5 : Whole Design Sequence Diagram

### 3.3.2 Use Case 1 Realization. 회원가입 및 로그인



Figure 6 : Use case1 Interaction Diagram

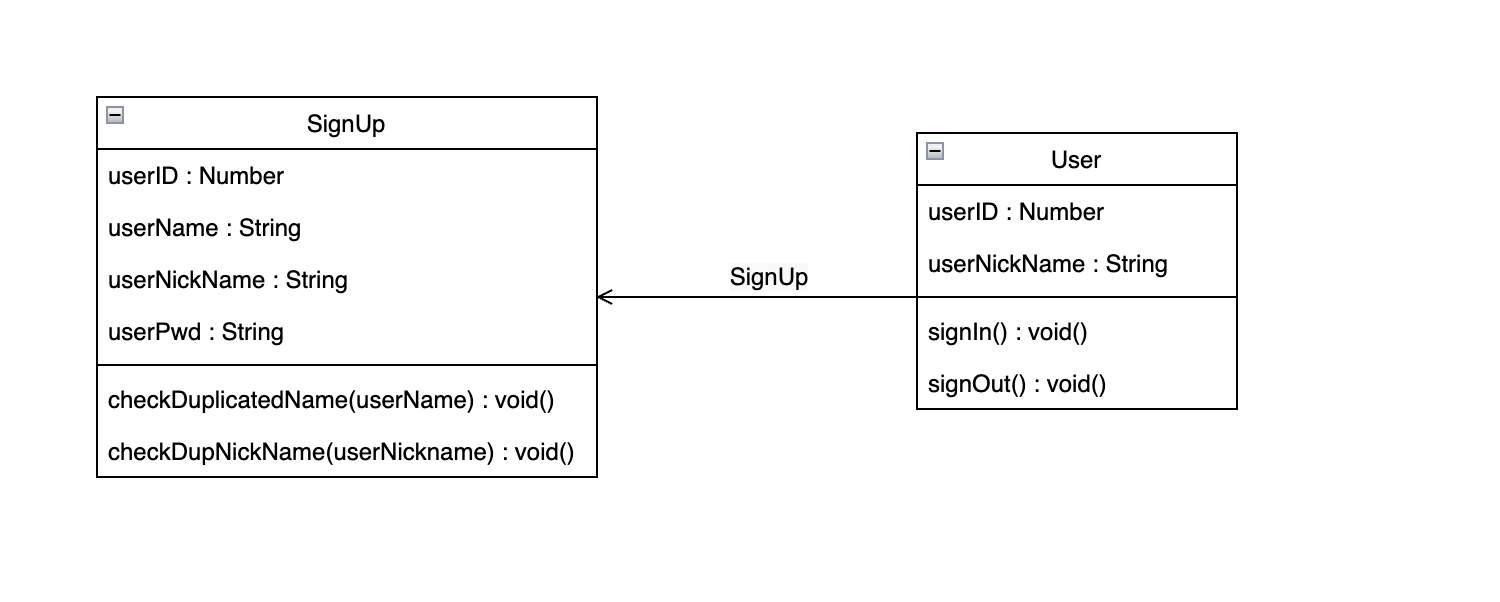


Figure 7 : Use case1 DCD

User는 자신의 아이디와 비밀번호를 parameter로 하는 singIn()을 통해 계정에 접속한다. SignIn이 완료되면 댓글 및 평가와 하이라이트를 저장할 수 있는 권한 Authority를 준다.   
만약 계정이 없을 경우 회원가입 signUp()을 하며 회원가입 시 아이디는 중복이 불가능하며 아이디와 비밀번호의 최대 길이는 15를 넘지 않는다. 회원가입이 완료되면 계정을 Account준다.

### 3.3.3 Use Case 2 Realization. 영상 검색

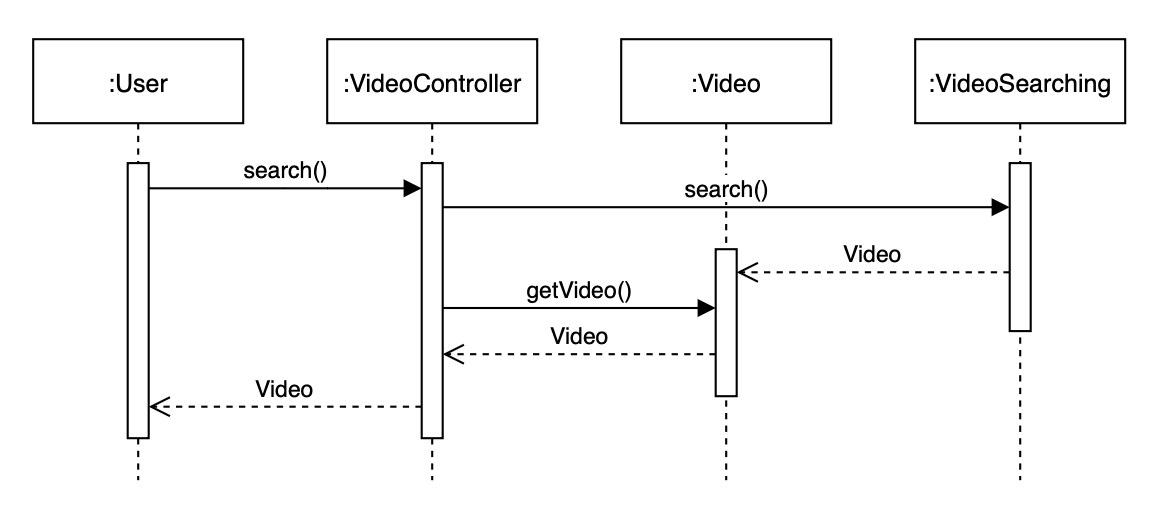


Figure 8 : Use case2 Interaction Diagram



Figure 9 : Use case2 DCD

User는 팀 명과 경기 날짜 등 검색어 및 필터를 통해 영상을 검색할 수 있다. User가 VideoController를 통해 search()로 검색을 시도하면, search()의 parameter인 팀 명과 경기 날짜를 통해 VideoController가 VideoSearching Service에게 검색을 요청한다. VideoSearching은 VideoController의 getVideo의 결과인 Video를 제공한다. VideoController는 그 Video를 User의 search 결과로 return한다.

만약 search()의 parameter와 부합하는 결과가 존재하지 않을 경우, videoSearching은 return Video로 결과가 존재하지 않는다는 메시지를 준다.

### 3.3.4. Use Case 3 Realization. 영상 시청

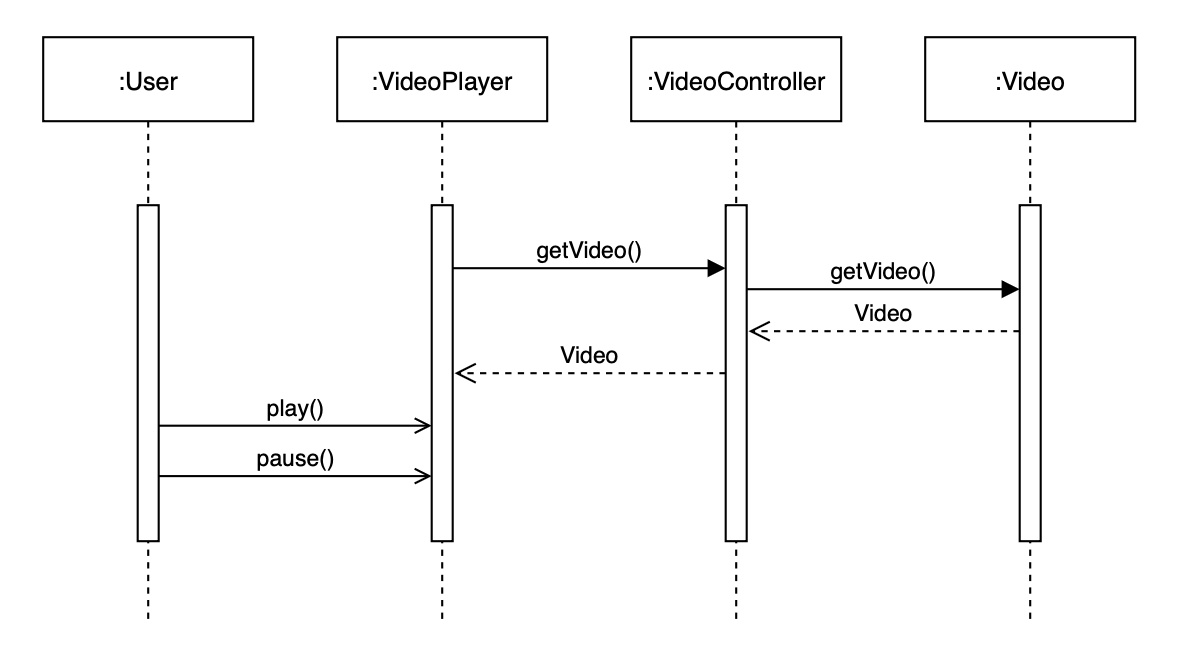


Figure 10 : Use case3 Interaction Diagram

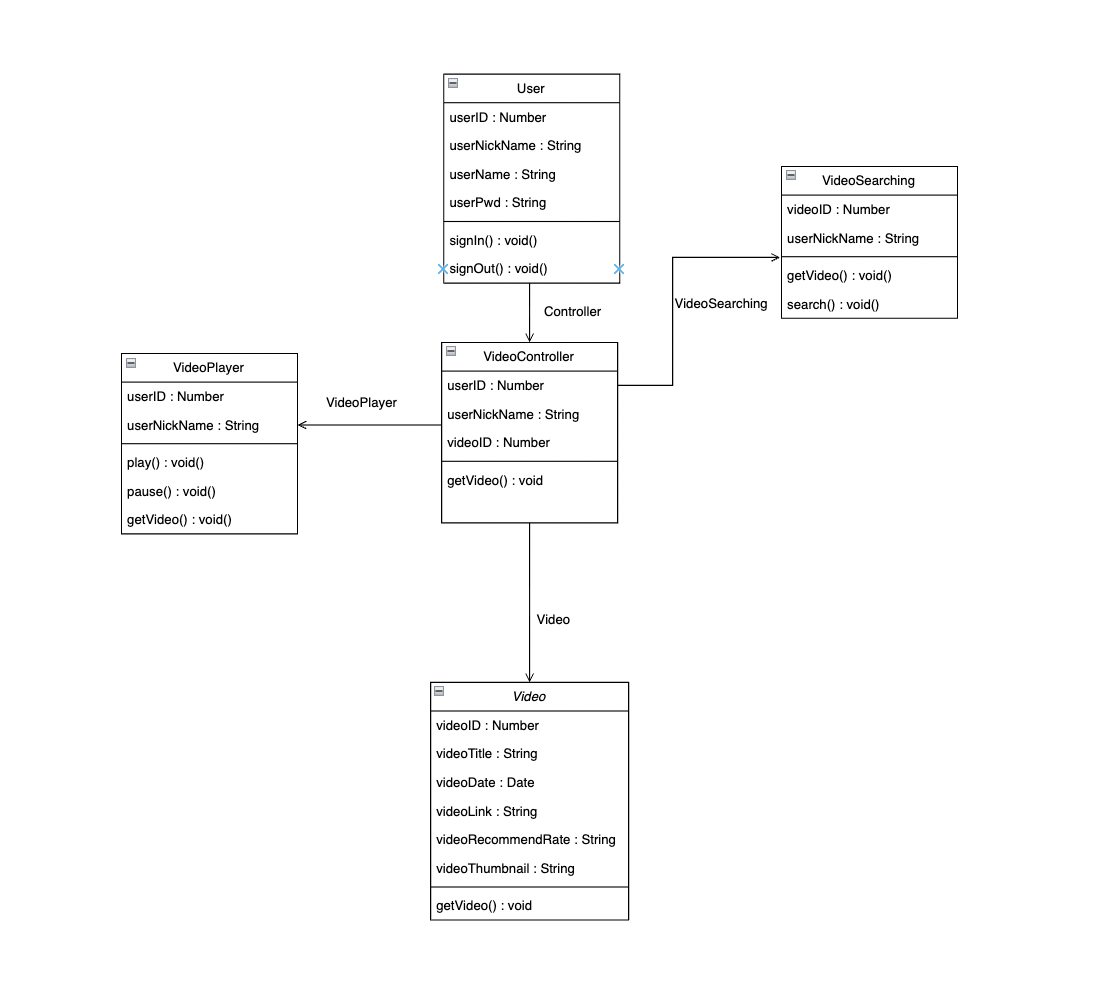


Figure 11 : Use case3 DCD

User는 VideoPlayer를 통해 경기를 시청할 수 있다. 홈페이지, 개인 하이라이트 저장 페이지, 또는 검색 결과 페이지에서 하이라이트 영상 목록 중 선택하여videoPlayer에게 getVideo()의 parameter로 제공한다. VideoPlayer는 getVideo()를 통해서 VideoController에게 적절한 Video를 제공받는다. User는 VideoController의 getVideo를 통해 Video로부터 제공받은 Video를 가지는 VideoPlayer에게 play()를 통해 재생, pause()를 통해 일시정지를 할 수 있다.

### 3.3.5. Use Case 4 Realization. 댓글 작성

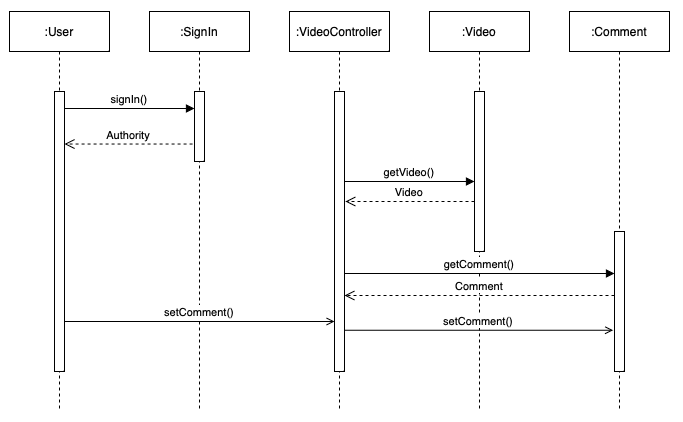


Figure 12 : Use case4 Interaction Diagram

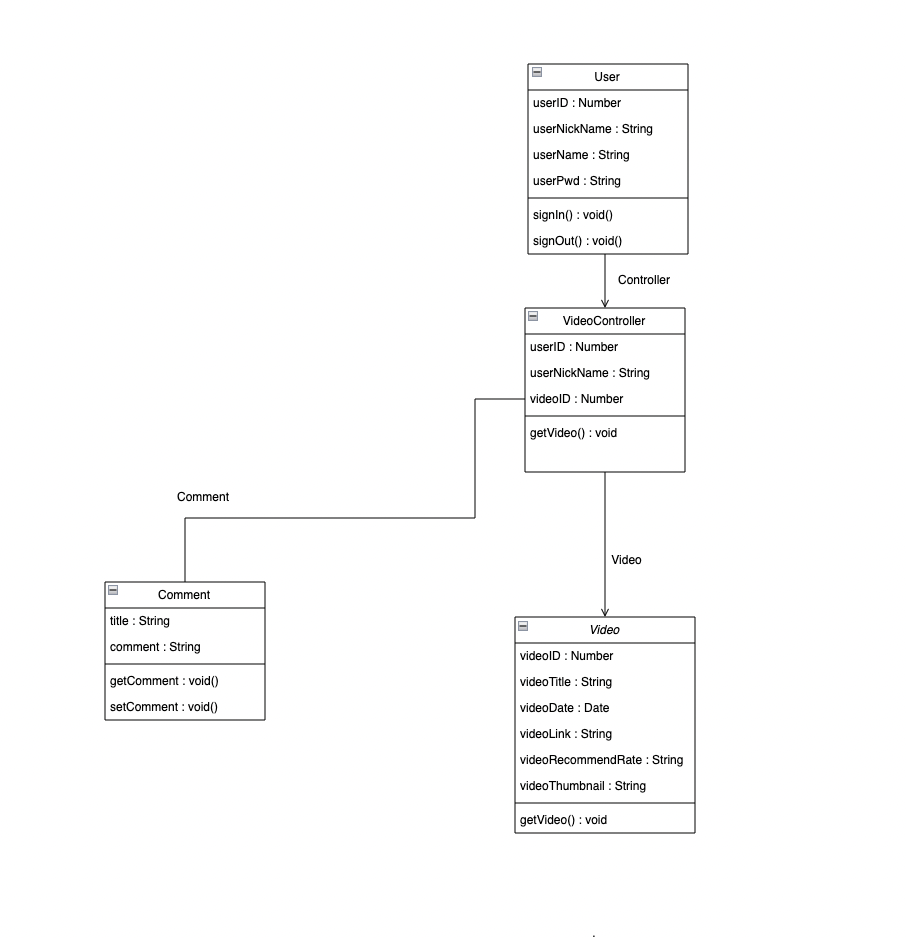


Figure 13 : Use case4 DCD

User는 각 경기 영상 하단에 위치한 댓글 창에 자신의 의견을 표현할 수 있는 댓글을 등록할 수 있다. 댓글 등록을 위해 User는 회원가입을 통해 생성한 자신의 계정으로 로그인이 요구된다. signIn()을 통해 로그인 후, 댓글 등록을 원하는 하이라이트 영상 제공 페이지와 VideoController를 통해 소통이 가능하다. User는 해당 영상의 하단에 위치한 댓글 창에 setComment()를 통해 댓글을 입력하고 등록할 수 있으며getComment()를 통해 댓글 목록에서 등록된 댓글들을 확인할 수 있다.

### 3.3.6. Use Case 5 Realization. 영상 평가

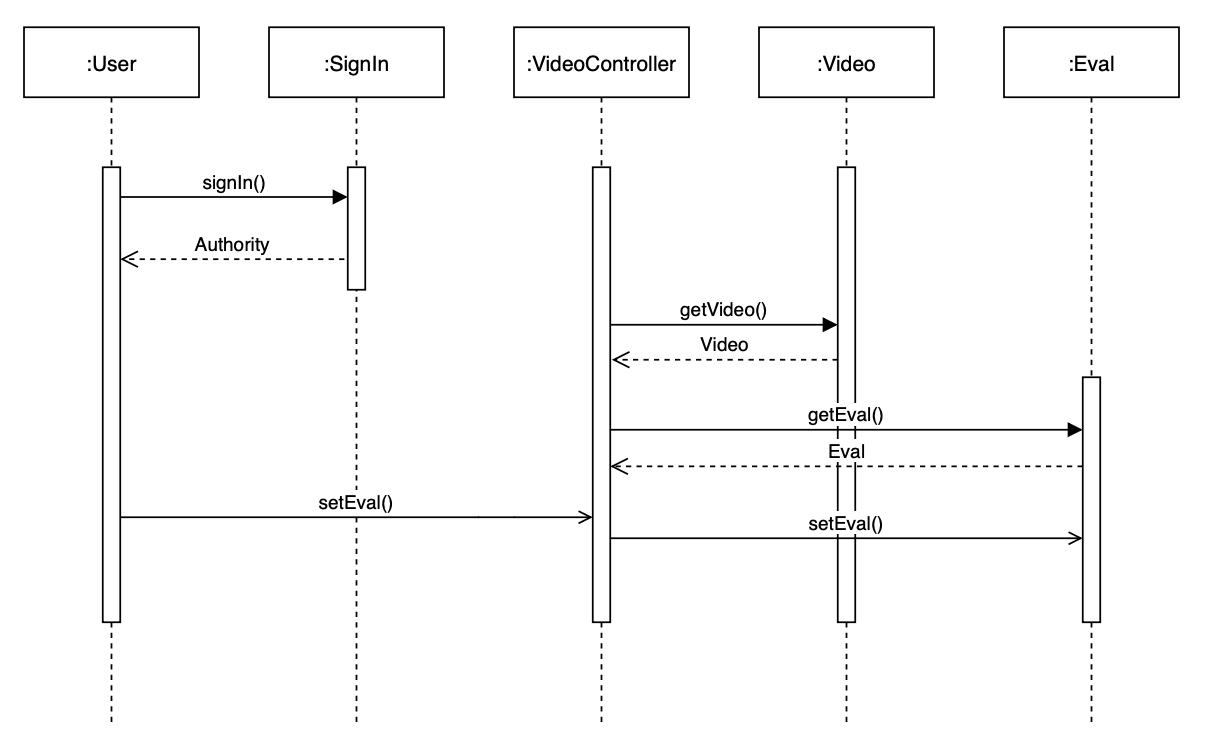


Figure 14 : Use case5 Interaction Diagram

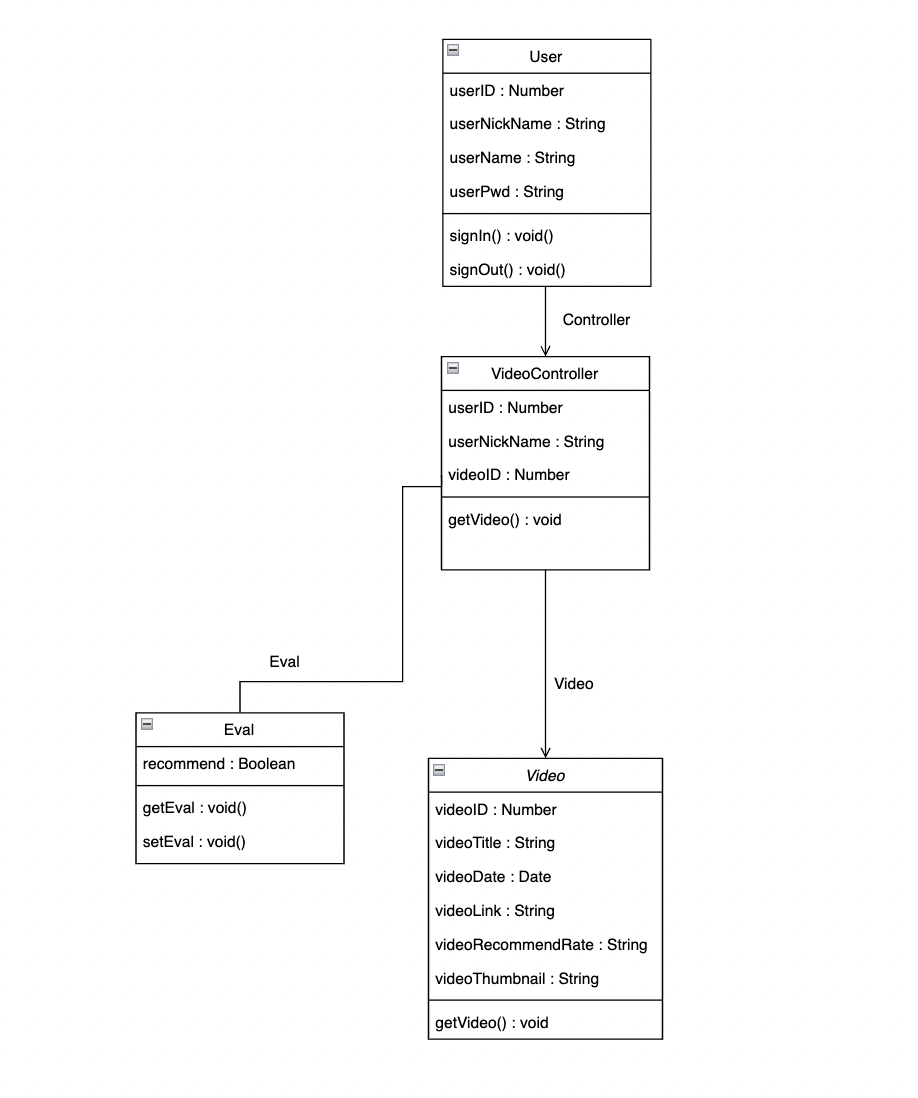


Figure 15 : Use case5 DCD

User는 각 하이라이트 영상 마다 추천/비 추천 형식으로 평가를 할 수 있다. 영상 평가를 위해 user는 회원가입을 통해 생성한 자신의 계정으로 로그인이 요구된다. signIn()을 통해 로그인 후, VideoController를 통해 평가를 하고자 하는 하이라이트 영상 제공 페이지로 소통이 가능하다. setEval()를 통해 해당 영상 하단에 위치한 추천/비 추천 버튼을 통해 영상을 평가하고 영상 평가를 마치면 User는 getEval()을 통해 해당 버튼 클릭 유무로 자신의 평가 기록 확인이 가능하다.

### 3.3.7. Use Case 6 Realization. 개인 저장

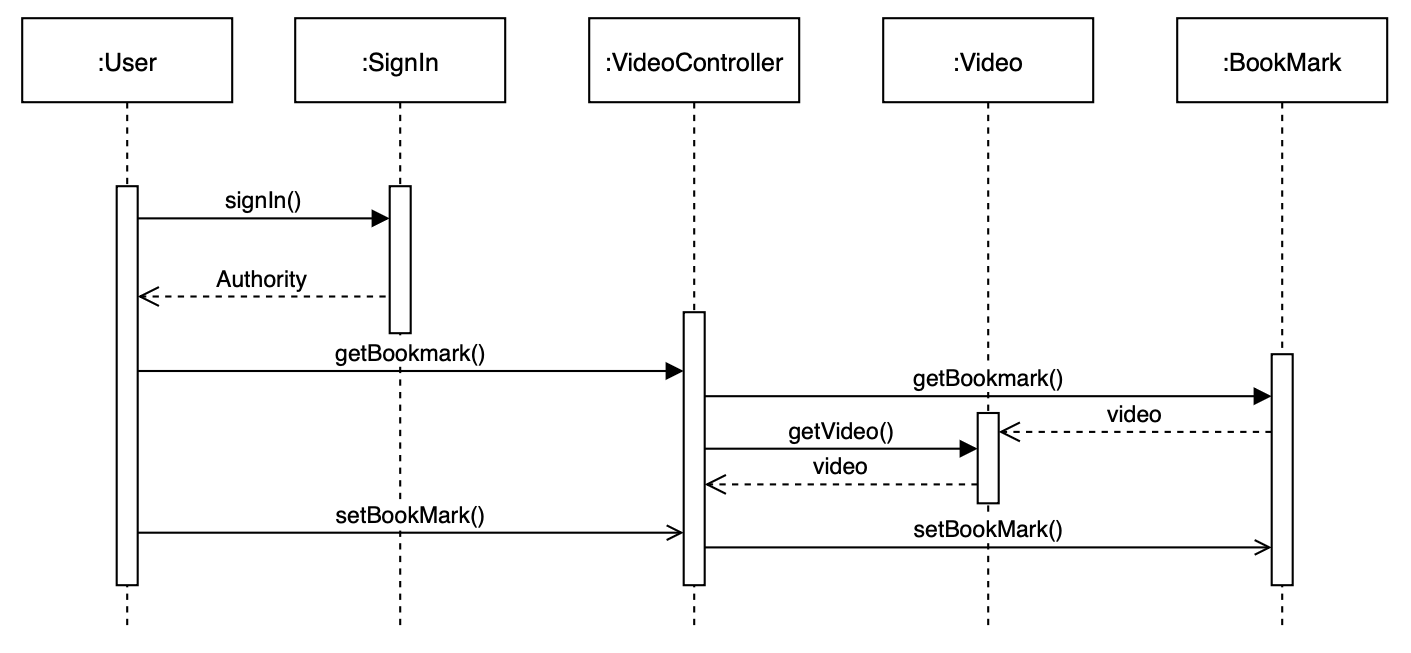


Figure 16 : Use case6 Interaction Diagram

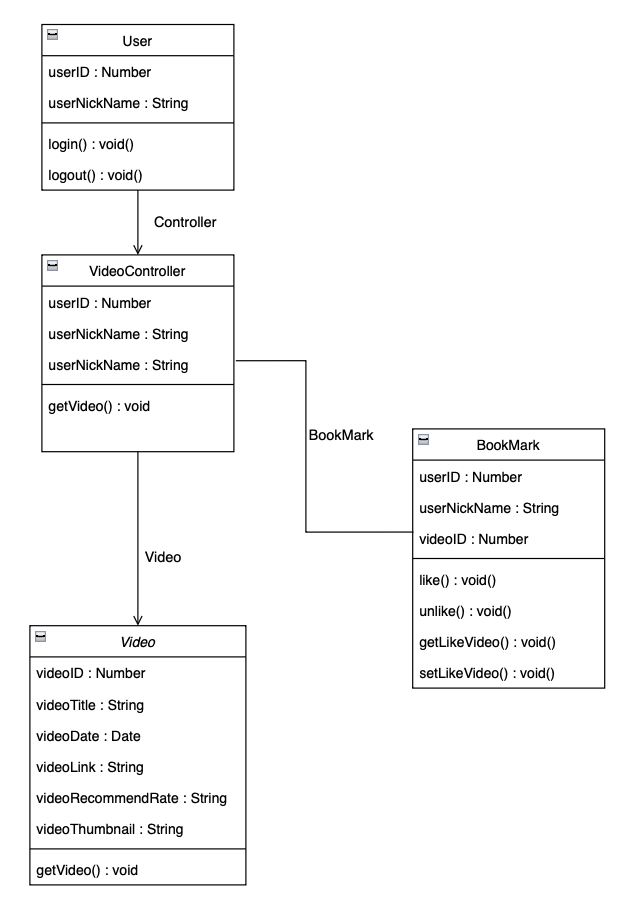


Figure 17 : Use case6 DCD

User는 로그인이 되어 있다면 북마크(하이라이트 개인 저장) 기능을 이용할 수 있다.

User는 VideoController를 통해서 BookMark 서비스와 소통한다. User는 getBookMark()를 통해 VideoController에게 BookMark 목록을 요청한다. videoController는 getBookMark()를 통해 BookMark Video를 return한다. User는 setBookMark()를 통해 VideoController에게 특정 Video를 parameter로 저장 요청할 수 있다. VideoController는 setBookMark()를 통해 BookMark를 저장한다.

### 3.3.8. Use Case 7 Realization. 경기 일정 검색

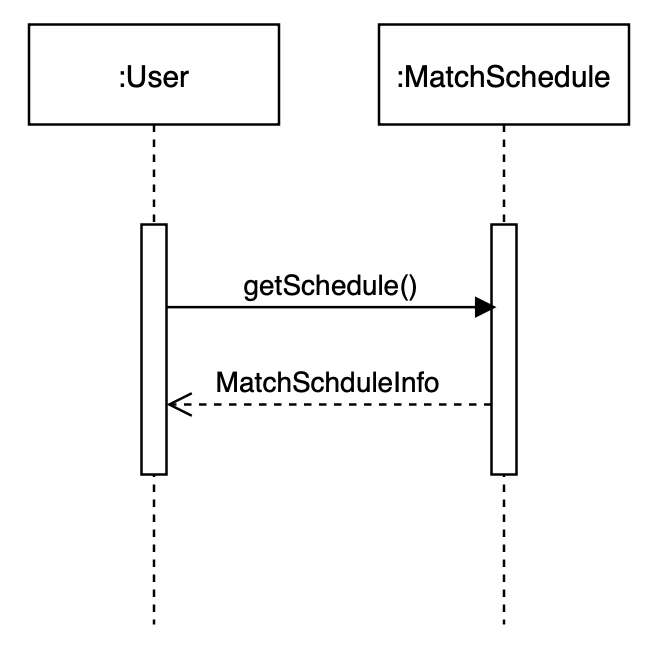


Figure 18 : Use case7 Interaction Diagram

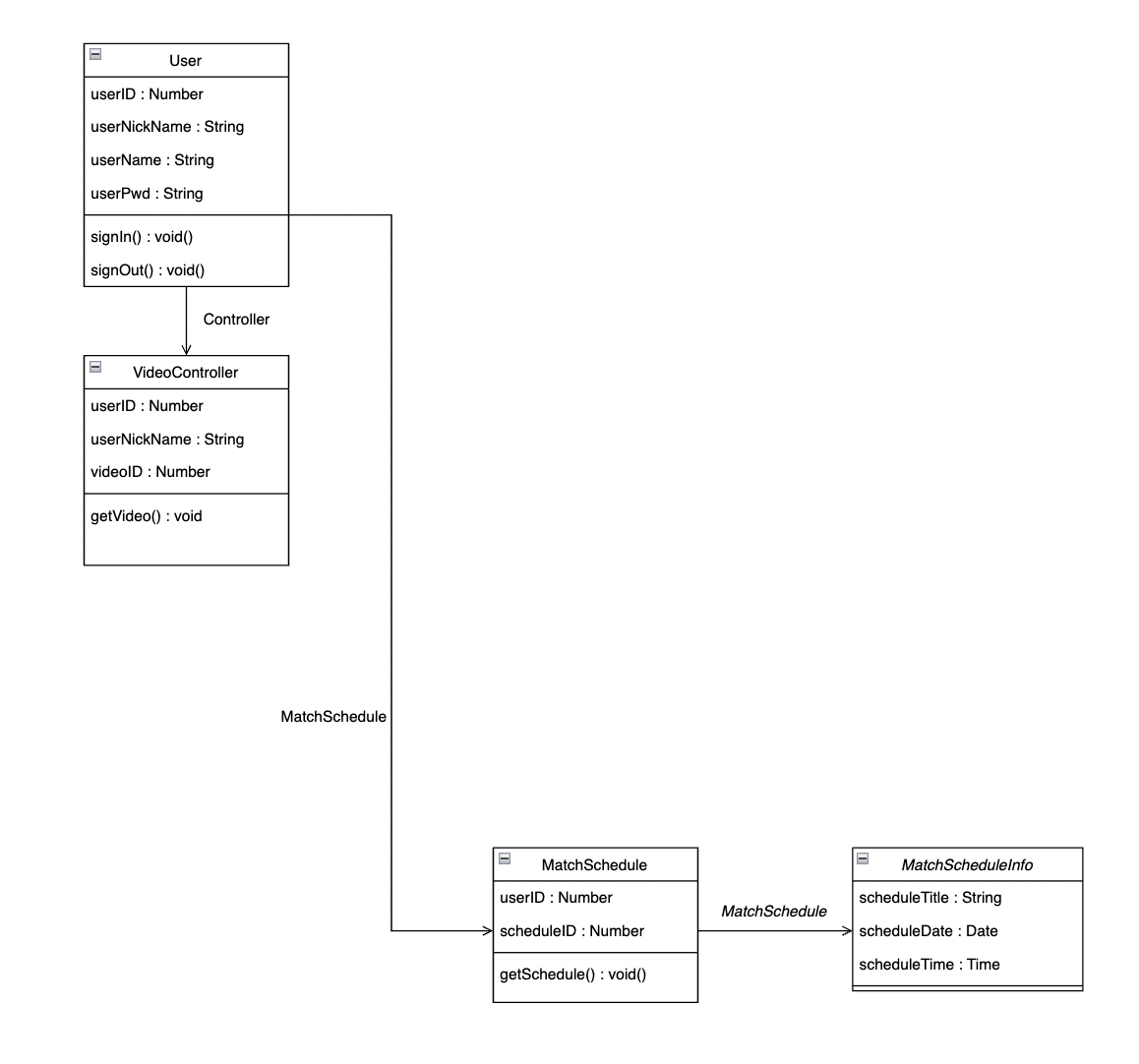


Figure 19 : Use case7 DCD

getSchedule()을 통해 현재 날짜를 기준으로 한 일정인 MathScheduleInfo를 가져온다. 경기 일정은 현재 날짜를 기준으로 지난 이틀동안 있었던 경기와 앞으로 이틀동안 있을 경기를 보여준다.

# 4. Implementation

## 4.1. Highlight Generator

### 4.1.1. Overview

텍스트, 표지판이(가) 표시된 사진

자동 생성된 설명

Figure 20 : Highlight Generator AI Architecture

하이라이트 영상 제작을 위한 Highlight Generator AI를 구현하였다. SoccerNet V2 dataset [1]을 통해 GPU(cuda)를 구동할 수 있는 server에서 경기 장면마다 장면을 분류하는 Action Spotter를 구현 및 학습시킨다.

Admin이 전체 경기 영상을 제공하면, 영상에서 Audio Extractor를 통해 음성 파일을 추출하고 VGGish [2] Feature Extractor를 통해 VGGish npy파일로 추출한다. 영상에 대해서는 ResNet152 [3]를 통해 Feature Extraction을 진행한다.

Feature Extracted된 두 영상/소리 modality에 대해서 NetVLAD [4] pooling layer를 통해서 pooling을 진행한다. 맞춰진 channel data들에 대해 Dropout을 적용하고, Affine FC Layer를 통과시켜 logit을 얻은 뒤 activation function을 진행해서 각 modality에 대한 장면 별 예측 값을 얻는다. 각 layer 사이 적절한 fusion layer를 두어 두 modality를 융합한다. 이렇게 구성한 multi-modal based AI model output으로 각 시간대 별 특정한 장면이 classifying되어있는 action time-table json파일을 얻는다.

이 Time-table을 통해 class 중 중요한 장면이라고 생각되는 장면들 앞 뒤로 적절하게 추출하여 붙임으로서 완성된 하이라이트 필름을 제공한다. 모델의 각 파트별로 자세한 내용을 아래에 기술한다.

### 4.1.2. Dataset

Dataset으로는 SoccerNet V2를 사용한다. SocceNet pre task인 action spotting base module [1]을 사용한다. 해당 Dataset은 EPL, Champions League, France ligue1, Bundesliga, Serie-A, Laliga 총 5개의 리그에 대해 2014-2015, 2015-2016, 2016-2017의 경기들로 전반, 후반으로 구분되어 총 580개의 45분(half-quarter)길이의 경기 영상과 label로 이루어져 있다.

경기 label은 json 형식의 파일로 어떤 특정한 장면들에 대해 그 장면이 나온 시간(gameTime), 그리고 그 장면에 관한 설명(label)등으로 이루어져 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 21 : Labels of game

나올 수 있는 class는 다음과 같이 규정되어 있다.

|  |  |
| --- | --- |
| Class | |
| Background | 기본 |
| Penalty | 패널티 킥 |
| Kick-off | 킥 오프 |
| Goal | 골 |
| Substitution | 선수 교체 |
| Offside | 오프사이드 |
| Shots on target | 유효 슈팅 |
| Shots off target | 슈팅 |
| Clearance | 클리어링 |
| Ball out of play | 라인 아웃 |
| Throw-in | 쓰로 인 |
| Foul | 파울 |
| Indirect free-kick | 간접 프리킥 |
| Direct free-kick | 직접 프리킥 |
| Corner | 코너킥 |
| Yellow card | 옐로 카드 |
| Red card | 레드 카드 |
| Yellow->red card | 경고 누적 |

Table 61 : Classes of highlight video

### 4.1.3. Audio

모델은 영상(Video)과 소리(Audio) modality로 나누어 융합하는 multi-modal model이기 때문에 일단 영상에서 소리를 추출한다. ffmpeg [6]를 통해 mkv 영상 파일에서 wav오디오 파일을 추출한다. 획득한 소리(Audio) modality에 대해 VGG를 통해 feature extraction을 진행한다. VGG는 AudioSet [7]을 통해 pre-trained된 모델이다. 해당 모델에서는 Pytorch [8]의 VGGish [9]라는 버전의 모델을 구현하였다. 해당 모델을 512 차원의 VGGish.npy Audio Feature를 얻게 된다.

### 4.1.3. Video

영상(Video) modality에 대해서는 ResNet-152를 통해 feature extraction을 진행한다.

ResNet152는 ImageNet을 통해 pre-trained 된 152layer의 deep-CNN Architecture이다. 해당 모델에서는 영상의 각 frame마다 2048 차원의 feature vector 표현을 사용하였다.

각 영상은 25 fps로 잘려 224\*224 크기로 전처리되어 해당 모델에 들어가 장면에 대한 feature vector output을 얻는다. PCA를 통해 512차원의 Video Feature를 얻게 된다.

### 4.1.5. Action Spotting

두 modality에 대해 NetVLAD [2] Pooling layer를 통해 pooling 을 진행한다. 1초마다 각 modality에 대해서 multi-binary cross entropy loss와 adam optimizer 등으로 구성된 전체 deep learning architecture에 대해서 학습되고 평가된다. 이렇게 얻은 적절한 각각 두 modality에 대한 예측 값을 late fusion mechanism을 적용하여 last layer에서 융합한다.

그렇게 얻는 최종 예측 값은 json file로 구성되어 있으며, 경기의 각 frame time table마다 그 시간의 예측 장면과, 그에 대한 confidence를 기록한다.

### 4.1.6. Highlight Generating

Model의 output으로 획득한 action spotting result json file에 대해, 0.65의 확신도 threshold를 넘는 confidence를 가진 예측 장면들만 필터링하고 정렬한다.

moviepy [10]를 사용하여 만들어낸 해당 장면들에 대한 시간 목록들에 대해 앞으로 5초, 뒤로 30초씩 패딩하여 해당 장면에 대해 35초 짜리 clip을 만든다.

생성된 모든 clip들을 moviepy를 통해 이어 붙이고, 새로운 비디오파일로 만들어 하이라이트 영상으로 제공한다.

## 4.2. BackEnd

### 4.2.1. Rest API

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Path | Message Format | Description |
| POST | /auth/signUp |  | 회원 가입 |
| POST | /auth/signIn |  | 로그인 |
| POST | /auth/signOut | - | 로그아웃 |
| DELETE | /auth/deleteAccount |  | 회원 탈퇴 |
| GET | / | - | 영상 리스트 |
| GET | /search?videoTitle={videoTitle} | - | 영상 검색 |
| GET | /random?num={num} | - | 영상 랜덤호출 |
| POST | /likeVideo/:videoID | - | 영상 추천 |
| POST | /reply/:videoID/comment |  | 댓글 등록 |
| GET | /reply/deleteComment |  | 댓글 삭제 |
| GET | /reply/getUserComment | - | 사용자 별  댓글조회 |
| GET | /reply/getVideoComment/:videoID |  | 영상 별  댓글조회 |
| POST | /bookmark/:videoID | - | 북마크 등록 |
| GET | /bookmark/getList | - | 북마크 조회 |
| GET | /schedule | - | 경기일정 전체조회 |
| GET | /schedule/date/:date | - | 특정 날짜 경기 일정 조회 |

Table 62 : Rest API

### 4.2.2. Database

#### Database Diagram

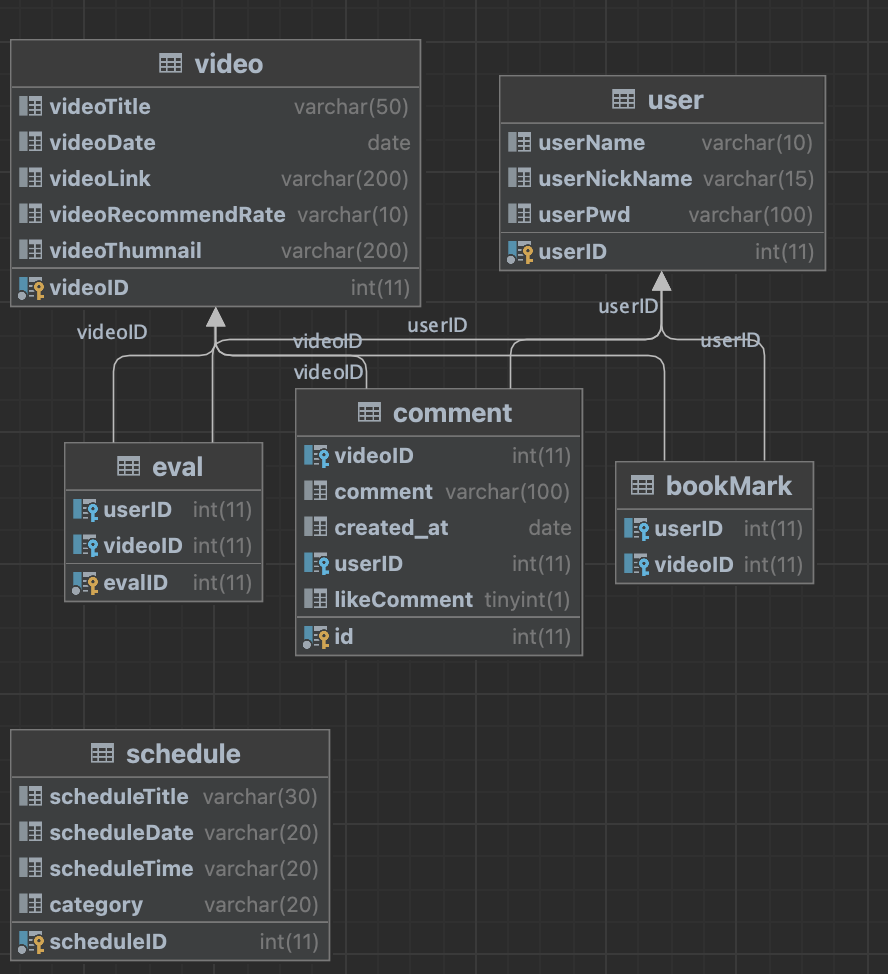


Figure 22 : Database diagram

#### Sequelize

|  |
| --- |
| Table: User |
| module.exports = class User extends Sequelize.Model {    static init(sequelize) {      return super.init(        {          userID: {            type: Sequelize.INTEGER,            autoIncrement: true,            allowNull: false,            primaryKey: true,          },          userName: {            type: Sequelize.STRING,            allowNull: false,          },          userNickName: {            type: Sequelize.STRING,            allowNull: false,          },          userPwd: {            type: Sequelize.STRING,            allowNull: false,          },        }, {          ...        });    }    static associate(db) {      db.User.hasMany(db.Comment, { foreignKey: 'userID', sourceKey: 'userID' });      db.User.hasMany(db.Eval, { foreignKey: 'userID', sourceKey: 'userID' });      db.User.hasMany(db.Bookmark, { foreignKey: 'userID', sourceKey: 'userID' });  }  }; |
| Sequelize를 이용하여 Database의 user table과 연동하였다. user table은 user 구분을 위한 userID(PK)와 로그인 시, ID로 사용될 userName, user의 닉네임인 userNickName, 그리고 user의 비밀번호인 userPwd로 구성 되어있다. user table과 comment, eval, bookmark table은 각각 1:N 관계를 갖는다. |

Table 63 : User table with Sequelize

|  |
| --- |
| Table: Video |
| module.exports = class Video extends Sequelize.Model {    static init(sequelize) {      return super.init(        {          videoID: {            type: Sequelize.INTEGER,            autoIncrement: true,            allowNull: false,            primaryKey: true,          },          videoTitle: {            type: Sequelize.STRING,          },          videoDate: {            type: Sequelize.DATE,          },          videoLink: {            type: Sequelize.STRING,          },          videoRecommendRate: {            type: Sequelize.INTEGER,          },          videoThumnail: {            type: Sequelize.STRING,          },        },        {          ...        }      );    }    static associate(db) {      db.Video.hasMany(db.Comment, { foreignKey: 'videoID', sourceKey: 'videoID' });      db.Video.hasMany(db.Eval, { foreignKey: 'videoID', sourceKey: 'videoID' });      db.Video.hasMany(db.Bookmark, { foreignKey: 'videoID', sourceKey: 'videoID' });  }  }; |
| Sequelize를 이용하여 Database의 video table과 연동하였다. video table은 video 구분을 위한 videoID(PK)와 영상의 제목인 videoTitle, 영상의 게시 날짜인 videoDate, 영상의 url 주소 정보를 담고 있는 videoLink, 영상의 추천 수를 확인할 수 있는 videoRecommenRate, 그리고 영상의 썸네일 이미지를 담고 있는 videoThumnail로 구성 되어있다. video table과 comment, eval, bookmark table은 각각 1:N 관계를 갖는다. |

Table 64 : Video table with Sequelize

|  |
| --- |
| Table: Comment |
| module.exports = class Comment extends Sequelize.Model{      static init(sequelize){          return super.init({              comment: {                  type: Sequelize.STRING,                  allowNull: true              }              created\_at: {                  type: Sequelize.DATE,                  allowNull: true,                  defaultValue: Sequelize.NOW              },          },{              ...          });      }      static associate(db){          db.Comment.belongsTo(db.User, {foreignKey: 'userID', targetKey: 'userID'});      }      static associate(db){          db.Comment.belongsTo(db.Video, {foreignKey: 'videoID', targetKey: 'videoID'});      }  }; |
| Sequelize를 이용하여 Database의 comment table과 연동하였다. comment table은 comment 구분을 위한 id(PK)가 sequelize에서 자동으로 생성되고 댓글 내용인 comment, 댓글 작성 시간인 created\_at으로 구성 되어있다. comment table은 user의 table의 userID와 video table의 videoID를 외래키(FK)로 사용한다. |

Table 65 : Comment table with Sequelize

|  |
| --- |
| Table: Eval |
| module.exports = class Eval extends Sequelize.Model{      static init(sequelize){          return super.init({              evalID: {                  type: Sequelize.INTEGER,                  primaryKey: true,                  autoIncrement: true,                  allowNull: false              }          },{              ...          });      }      static associate(db){          db.Eval.belongsTo(db.User, {foreignKey: 'userID', targetKey: 'userID'});      }      static associate(db){          db.Eval.belongsTo(db.Video, {foreignKey: 'videoID', targetKey: 'videoID'});      }  }; |
| Sequelize를 이용하여 Database의 eval table과 연동하였다. eval table은 evaluation 구분을 위한 evalID(PK)로 구성 되어있다. eval table은 user의 table의 userID와 video table의 videoID를 외래키(FK)로 사용한다. |

Table 66 : Eval table with Sequelize

|  |
| --- |
| Table: Bookmark |
| module.exports = class Bookmark extends Sequelize.Model{      static init(sequelize){          return super.init({              bookmarkID: {                  type: Sequelize.INTEGER,                  primaryKey: true,                  autoIncrement: true,                  allowNull: false              }          },{              ...          });      }      static associate(db){          db.Bookmark.belongsTo(db.User, {foreignKey: 'userID', targetKey: 'userID'});      }      static associate(db){          db.Bookmark.belongsTo(db.Video, {foreignKey: 'videoID', targetKey: 'videoID'});      }  }; |
| Sequelize를 이용하여 Database의 bookmark table과 연동하였다. bookmark table은 bookmark 구분을 위한 bookmarkID(PK)로 구성 되어있다. Bookmark table은 user의 table의 userID와 video table의 videoID를 외래키(FK)로 사용한다. |

Table 67 : Bookmark table with Sequelize

|  |
| --- |
| Table: Schedule |
| module.exports = class Schedule extends Sequelize.Model {  static init(sequelize) {  return super.init(  {  scheduleID: {  type: Sequelize.INTEGER,  autoIncrement: true,  allowNull: false,  primaryKey: true,  },  scheduleTitle: {  type: Sequelize.STRING,  },  scheduleDate: {  type: Sequelize.STRING,  },  scheduleTime: {  type: Sequelize.STRING,  },  category: {  type: Sequelize.STRING,  },  },  {  …  }  );  } |
| Sequelize를 이용하여 Database의 schedule table과 연동하였다. schedule table은 schedule들을 구분을 위한 schedule ID(PK), 타이틀인 scheduleTitle, 경기날짜를 나타내는 scheduleDate, 경기 시간을 나타내는 scheduleTime, 경기정보를 나타내는 category로 구성되어있다. |

Table 68 : Schedule table with Sequelize

### 4.2.3. Controller

#### signController: 회원정보와 관련된 서비스를 제공 및 관리한다.

* 1. signup

: 회원가입 기능을 제공하는 모듈로 user로부터 user가 가입하고자 하는 아이디(username), 닉네임(userNickName), 비밀번호(userPwd), 그리고 비밀번호 확인 값(userPwdCheck)을 입력 받는다. 이 때, 아이디와 비밀번호는 입력 받은 값의 길이가15 이하인지 확인하고 초과된 경우, “15자 이하여야 함”을 알리는 메시지를 띄운다. 이 후, 아이디와 닉네임의 중복체크를 진행하고 이미 데이터베이스에 동일한 값이 있는 경우, “중복된 값이 있음”을 알려준다. 마지막으로 user가 입력한 비밀번호가 비밀번호 확인 값과 같은 지 비교한 후, 같지 않으면 “일치하지 않다”는 메시지를 띄우고 같다면 user가 입력한 값으로 데이터베이스에 user 정보를 추가한다. 이 때, 보안을 위해 비밀번호는 블로피시 암호에 기반을 둔 암호화 해시 함수인 bcrypt를 사용하여 암호화된 값을 저장한다. 회원가입이 완료되면 “정상적으로 회원가입이 완료되었음”을 알리는 메시지를 띄운다.

* 1. signin

: 로그인 기능을 제공하는 모듈로 user가 가입한 아이디(userName)와 비밀번호(userPwd)를 입력 받는다. User가 입력한 아이디가user테이블에 존재하지 않는다면, “등록되지 않은 사용자”임을 알리는 메시지를 띄운다. 반면, 아이디는 존재하지만 입력된 비밀번호가 데이터베이스에 저장된 값과 일치하지 않는다면 “비밀번호가 일치하지 않음”을 알려준다. 해당 단계를 모두 통과한 후, 로그인을 시도하게 되면 인터넷 표준 인증방식인 jwt 토큰을 부여한다. 토큰은 userID(Primary Key)를 기반으로 서명하고 미리 설정해둔 환경변수로 암호화를 진행한다. 해당 토큰은 쿠키에 저장하고 “정상적으로 로그인 되었음”을 알리는 메시지를 띄운다.

* 1. signout

: 로그아웃 기능을 제공하는 모듈로 요청이 오면 토큰을 저장해 둔 쿠키를 삭제함으로써 로그아웃이 실행되고 “정상적으로 로그아웃 되었음”을 알려준다.

* 1. deleteAccount

: 회원탈퇴 기능을 제공하는 모듈로 해당 요청이 들어왔을 때, user 본인 확인을 위해 비밀번호를 요구한다. 이 때, 비밀번호가 user의 정보와 일치하지 않으면 “비밀번호가 일치하지 않음”을 알리는 메시지를 띄우고 일치하는 경우에는 “정상적으로 탈퇴 되었음”을 알리며 user의 정보를 데이터베이스에서 삭제함으로써 회원 탈퇴 서비스를 제공한다.

#### commentController : 댓글과 관련된 서비스를 제공 및 관리한다.

* 1. addComment

: 댓글 작성 기능을 제공하는 모듈로 user가 작성하고자 하는 댓글의 내용과 userID(user 테이블의Primary Key), 댓글을 작성하고자 하는 비디오의 videoID(video 테이블의 Primary Key)를 입력 받는다. 입력 받은 값을 기반으로 comment 테이블에 정보를 추가한다. 이 때, userID와 videoID는 comment 테이블에서 Foreigner Key로 저장된다.

* 1. deleteComment

: 댓글 삭제 기능을 제공하는 모듈로 user가 삭제하고자 하는 댓글의 commentID(Primary Key)를 기반으로 댓글의 존재 유무와 현재user와 댓글의 작성자가 동일한 지, 검증 과정을 거친다. 만약 삭제하고자 하는 댓글이 존재하지 않는다면, “존재하지 않는 댓글”임을 알려주고 user와 댓글의 작성자가 동일하지 않는다면, “자신의 댓글만 삭제 가능함”을 알리는 메시지를 띄운다. 이 과정을 통과했다면 해당 commentID에 해당되는 댓글의 tuple을 comment 테이블에서 삭제함으로써 댓글 삭제를 진행한다. 삭제 후, “댓글이 삭제되었음”을 알리는 메시지를 띄운다.

* 1. getUserComment

: user별로 댓글 리스트를 보여주는 서비스이다. 쿠키에 저장된 userID를 기반으로 user가 작성한 모든 댓글들을 comment 테이블에서 찾아서 보여준다.

* 1. getVideoComment

: video 별로 댓글 리스트를 보여주는 서비스이다. Parameter로 받은 videoID를 기반으로 해당 비디오에 달린 모든 댓글들을 comment 테이블에서 찾아서 보여준다. 이 때, 각 댓글의 작성자의 닉네임(userNickName)도 같이 제공한다.

#### evalController : 영상 평가와 관련된 서비스를 제공 및 관리한다.

* 1. likeVideo

: 영상 평가 기능을 제공하는 모듈로 쿠키에 저장된 userID와 parameter로 받은 videoID를 기반으로 영상 평가의 유무를 판단한다. eval 테이블에 동일한userID와 videoID가 존재한다면, 이미 해당 영상에 대한 평가를 진행한 것으로 판단하여 “평가를 취소”했음을 알리며 eval 테이블에서 해당 tuple을 삭제함과 동시에video 테이블의 videoRecommendate column의 값을 ‘-1’(감소) 시킨다. 반면, 동일한 userID와 videoID가 존재하지 않는다면, 해당 영상에 대한 평가가 아직 진행되지 않았으므로 “평가가 등록되었음”을 알리며 eval 테이블에서 해당 데이터 값을 기반으로 tuple을 생성하고 video 테이블의 videoRecommendate column의 값을 ‘+1’(증가) 시킴으로써 영상 평가를 진행한다.

#### videoController : 영상 재생과 관련된 서비스를 제공 및 관리한다.

* 1. findAll

: 데이터베이스의 Video Table에 저장되어 있는 모든 비디오 파일을 보여준다. 지정된 주소로 GET요청만 하면 findAll을 통해 모두 가져온다.

* 1. findSome

: 동영상 시청페이지에서 랜덤으로 프론트엔드에서 지정한 개수만큼 랜덤으로 영상들을 반환한다. 랜덤으로 가져오기 위해 js의 random함수를 통해서 videoTable의 동영상을 랜덤으로 가져온다. 이때, 가져올 수 있는 개수를 가변적으로 설정했다. 만일 보유한 동영상보다 요청하는 동영상 개수가 더 많다면, 이에 대한 메세지와 모든 데이터를 반환한다.

* 1. findAllTitle

: 검색 키워드를 통해 비디오 리스트를 보여준다. query로 비디오의 타이틀을 프론트엔드에서 인자로 받는다. like 구문을 통해서 그 키워드가 들어간 비디오 영상들을 반환한다.

#### scheduleController : 경기 일정과 관련된 서비스를 제공 및 관리한다.

* 1. findDate

: 특정 날짜에 대한 경기 일정을 나타낸다. 프론트 엔드로부터 params로 특정 날짜를 (ex : 12-04) 받아서 like 구문을 통해서 조건을 추가한 다음 findAll로 조건에 해당되는 경기일정을 반환한다.

#### bookmarkController : 영상 북마크 서비스를 제공 및 관리한다.

* 1. bookmarkVideo

: 비디오 영상을 북마크하거나 해제할 수 있다. jwt에 포함되어있는 userID와 params로 받는 videoID를 통해서 where절을 통해 데이터베이스의 bookmark table에서 찾는다. 만약 값이 없을 시, 북마크를 추가하고 ‘북마크에 추가되었습니다.’라는 메세지를 반환한다. 만약 값이 있을 시, 데이터베이스에서 삭제하고 ‘북마크가 해제되었습니다.’라는 메세지를 반환한다.

* 1. getBookmark

: 각 유저가 저장한 비디오 리스트를 나타낸다. 먼저 jwt에 포함되어있는 userID를 통해서 userNickName을 가져온다. 마이페이지에서 userNickName이 필요하기 때문이다. 그 다음, Bookmark table에 userID를 조건으로 video table에서 video 리스트를 가져온다. 이후 json으로 userName과 유저가 북마크한 video리스트를 반환한다.

## 4.3. FrontEnd

### 4.3.1. overview

피그마를 사용해 전반적인 UI를 구체화하였으며 리액트를 사용하여 구현하였다. 백엔드와의 통신은 axios 모듈을 사용하였다.

### 4.3.2. UI

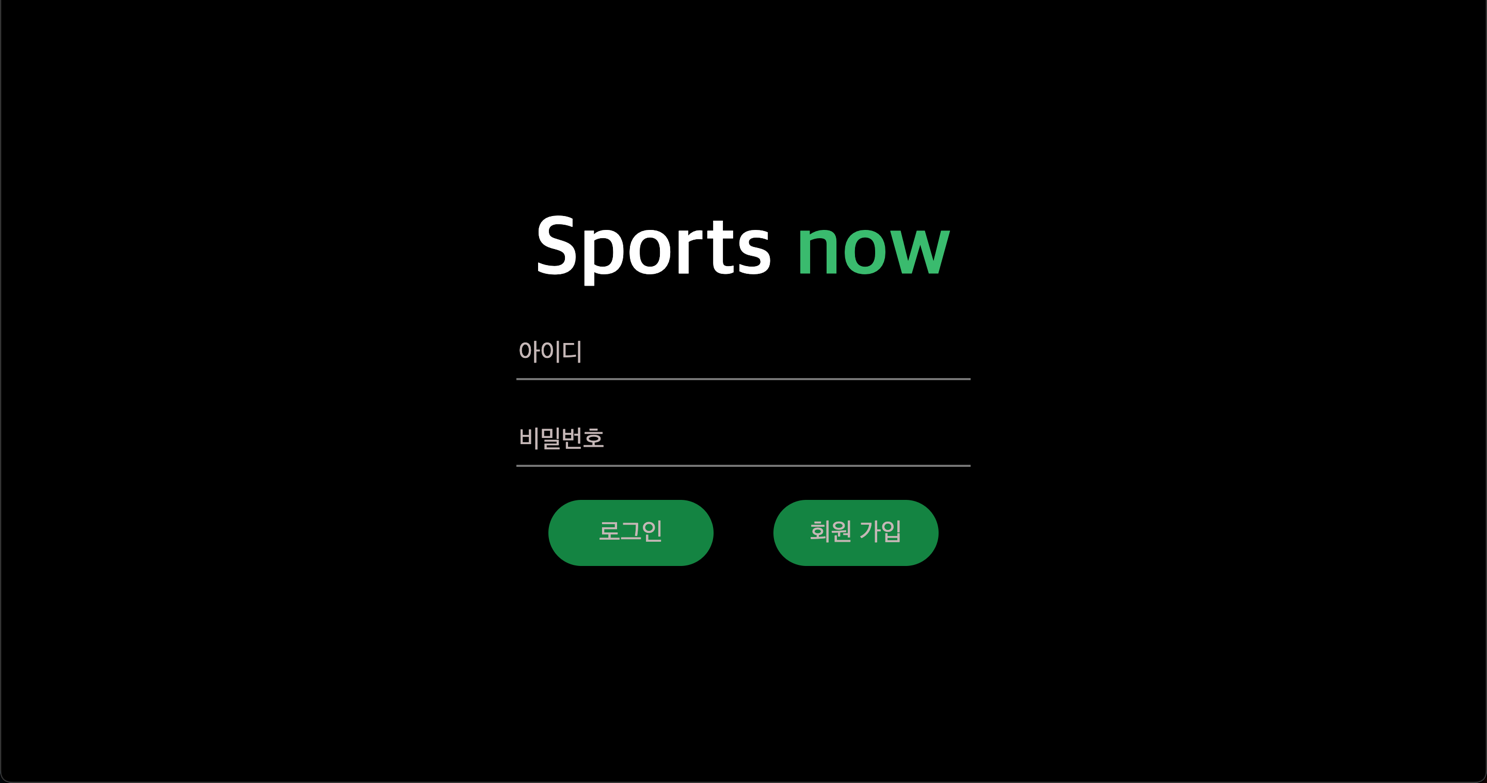


Figure 23. UI – 로그인

사용자의 아이디와 비밀번호를 input value로 받은 뒤에, 백 엔드에 body로 POST 요청을 한다. 로그인을 실패할 시 로그인 실패 alert를 띄우고, 성공할 시 로그인 성공했다는 alert를 띄운다. 이때 쿠키에 jwt토큰이 저장되는데, jwt토큰이 있다면 topNavBar를 보여주고, 없다면 위 화면 처럼 보여주지 않도록 구현했다. (회원가입과 동일) 마지막으로 home으로 라우팅된다.

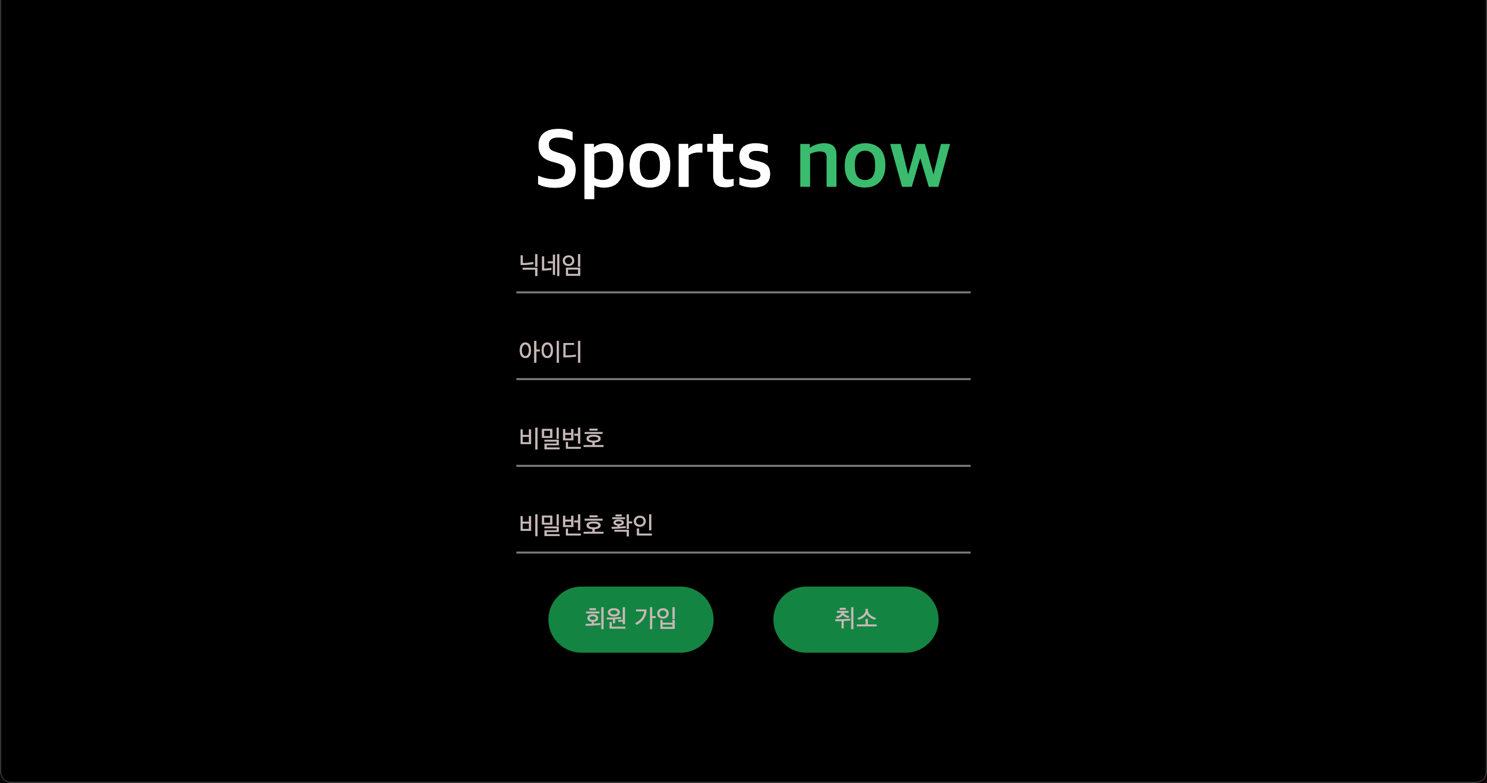


Figure 24. UI – 회원가입

사용자의 닉네임, 아이디, 비밀번호, 비밀번호 재확인 input value를 저장한 뒤에 백 엔드에 body로 POST요청을 한다. 회원가입이 제대로 완료되지 않았다면, 회원가입 실패 alert를 띄워주고, 제출 때 비밀번호 재확인이 불일치를 검증하여 불일치할 경우 비밀번호 불일치 alert를 띄워준다.



Figure 25. UI – 홈페이지

로그인 후 처음으로 보여지는 페이지이다. 해당 페이지에서는 영상에 대한 간단한 정보(제목, 날짜, 추천 수)를 담은 Component가 화면 너비에 맞게 주어지며 스크롤을 내려 밑에 있는 Component들도 확인 할 수 있다. 시청을 원하는 영상을 클릭하면 videoID와 함께 영상 재생 페이지로 넘어가게 된다.



Figure 26. UI - 영상 재생 페이지

영상을 시청할 수 있는 영상 재생 페이지이다. 영상 재생 페이지에는 하이라이트 영상이 아닌 원본을 볼 수 있는 원본 링크와 북마크, 추천 그리고 댓글 기능이 포함되어 있다.

북마크와 추천 버튼을 누르면 해당 videoID가 포함된 요청이 서버에 전달된다. 요청이 성공적으로 서버에 도착해 북마크 또는 추천이 추가가 완료가 되면 해당 완료 메시지를 유저에게 보여준다. 북마크 또는 추천을 한 후 다시 버튼을 누르게 되면 취소되며 해당 취소 메시지를 유저에게 보여준다.



Figure 27. UI - 댓글 작성

원하는 특정 비디오에 대한 댓글을 추가할 수 있는 기능이다. 댓글 입력란에 원하는 내용을 기입한 후 Enter key 또는 댓글 작성 버튼을 누르면 해당 비디오 댓글란 가장 밑에 추가된다.

휴지통 아이콘을 클릭하면 자신의 댓글을 삭제할 수 있으며 삭제를 완료 후 완료 메시지가 유저에게 보여진다. 자신의 댓글이 아닌 다른 유저의 댓글에 대한 휴지통 아이콘을 클릭하면 해당 댓글을 삭제 할 수 없다는 메시지가 유저에게 보여진다.



Figure 28. UI - 일정 페이지

특정 날짜의 경기 일정을 나타내는 페이지이다. 현재 달(month)을 기준으로 날짜와 요일을 설정했다. 평균 30일 정도의 버튼을 나타내기 위해 가로 스크롤 형식으로 구현했다.

특정 날짜를 선택하면 그 날짜의 값을 RestAPI를 통해 url에 담아서 백 엔드에 요청한다.

날짜마다 다른 데이터 값을 출력해야 하므로 경기 일정을 나타내는 컴포넌트는 데이터 값만 들어갈 수 있게끔 컴포넌트화 했다. 백 엔드에서 받아오는 데이터에 따라 데이터가 없다면 등록된 정보가 없다는 컴포넌트를 나타냈다.

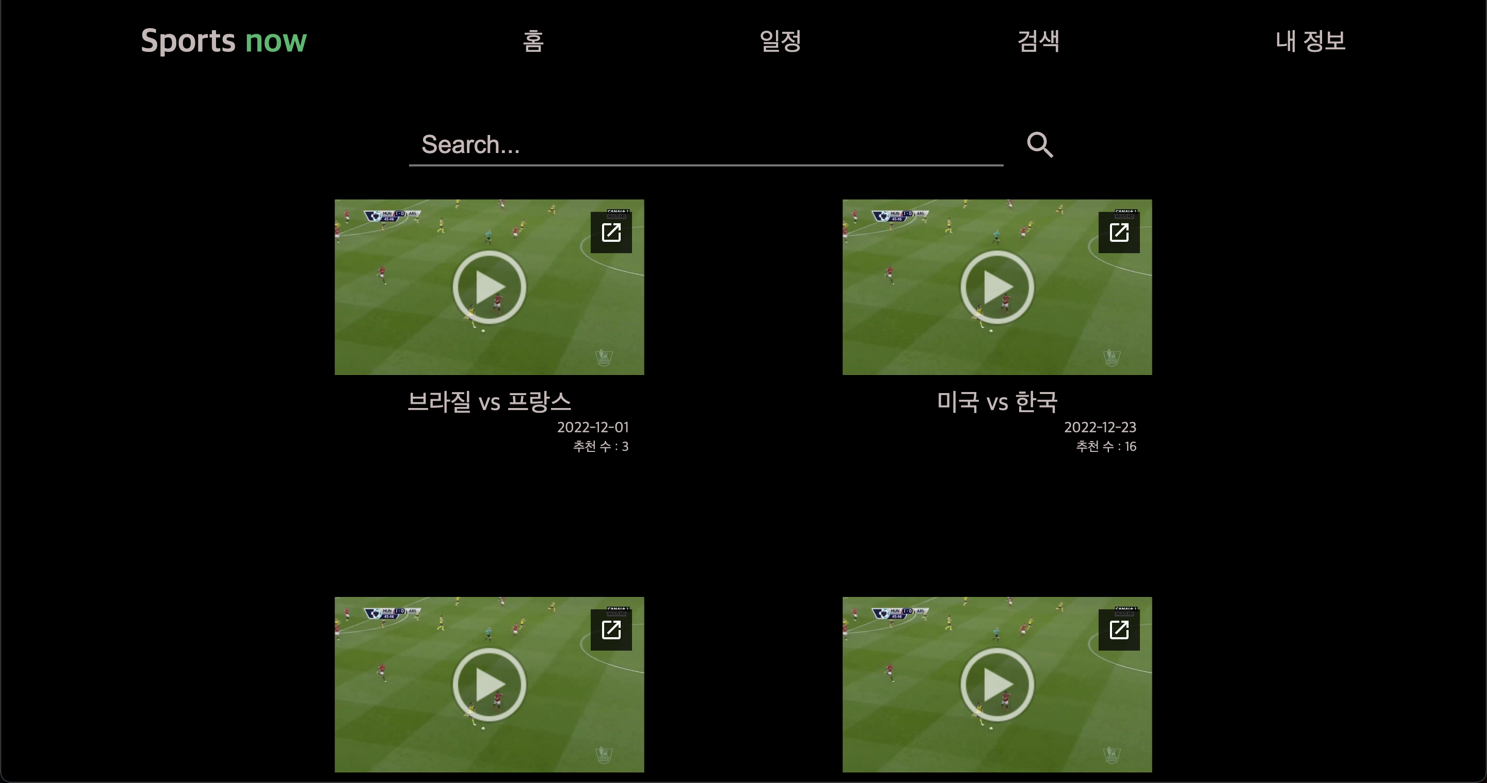


Figure 29. UI - 검색 페이지

제목을 통해 영상을 검색할 수 있는 검색 페이지이다. 제목에 대한 정보를 요청이 REST API를 통해 url에 담아 서버에 요청된다. 해당 요청에 대한 비디오 정보들이 반환이 되면 각 비디오에 대한 간단한 정보를 담은 Component들이 보여진다.

유저가 시청을 원하는 Component를 클릭하게 되면 해당 비디오에 대한 영상 재생 페이지로 넘어가게 된다.



Figure 30. UI - 내 정보 페이지(북마크)

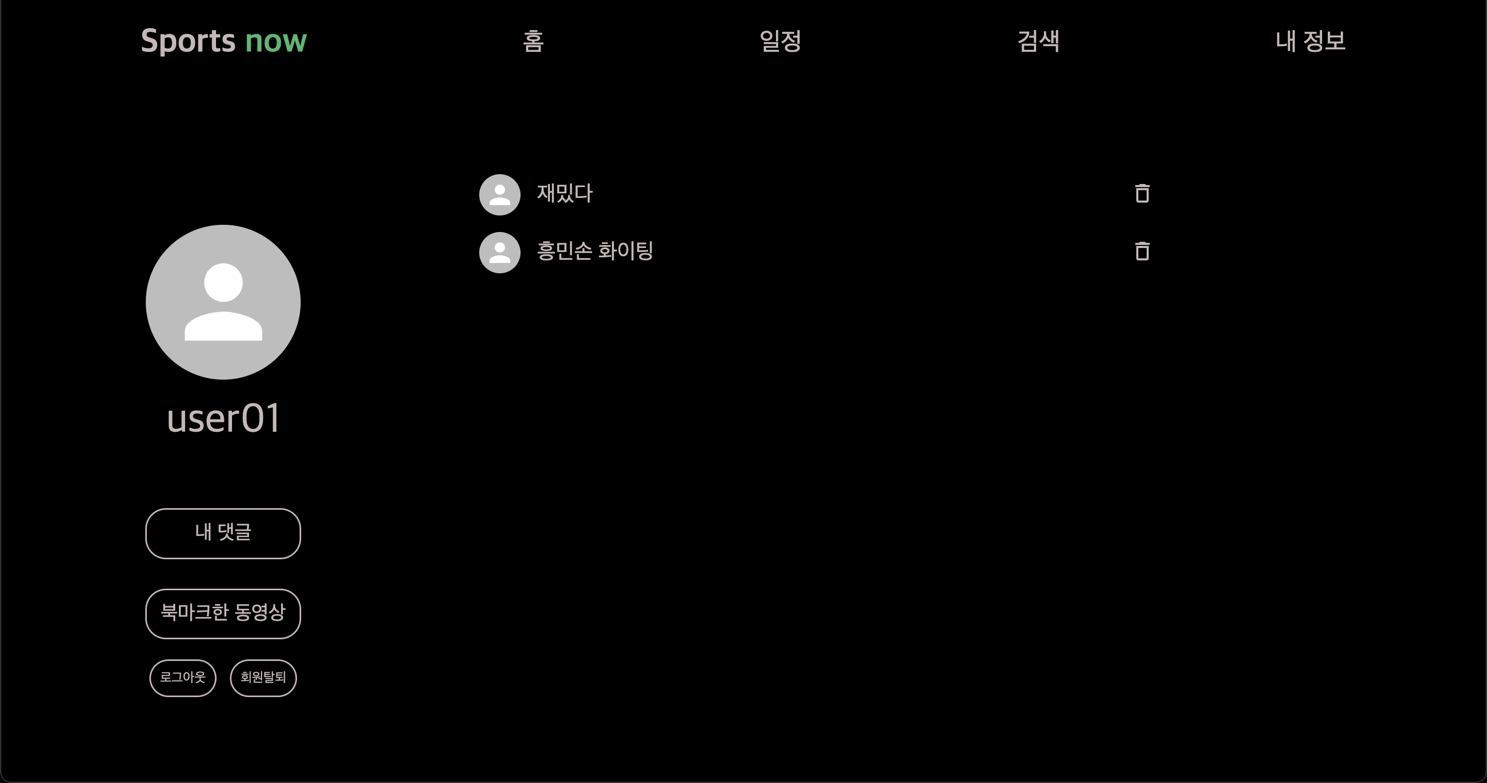


Figure 31. UI - 내 정보 페이지(댓글)

내 정보 페이지에서의 기능은 내가 쓴 댓글확인, 북마크한 동영상 보기, 로그아웃, 회원탈퇴 총 4개이다. 내 정보 페이지를 접속했을 때의 첫 화면은 북마크한 동영상을 나타내는 것으로 구현했다. 이후 내 댓글 버튼을 누르면 내가 쓴 댓글을 확인할 수 있다. 한 화면에서 두 개의 화면을 보여줘야하므로 toggle state를 통해서 내 댓글일 경우와 북마크한 동영상일 경우를 구분했다. 버튼 클릭시 해당 버튼state를 toggle state에 저장한 뒤에, toggle state에 따라서 두 개의 컴포넌트를 화면에 나타냈다.

로그아웃 시 쿠키에 저장된 jwt 토큰을 삭제하고, 로그인 페이지로 라우팅했다. 회원 탈퇴 기능은 RESTAPI를 통해 백 엔드에 요청한 뒤, 계정이 삭제되면, 쿠키의jwt 토큰을 삭제한다. 이후 마찬가지로 로그인 페이지로 라우팅한다.

# 5. Testing

## 5.1. 회원가입

5.1.1. 회원가입 성공

5.1.2. 회원가입 실패

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.1.1 | | |
| Test Case Name : 회원가입 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 회원가입 창에 접속 | 아이디와 닉네임, 비밀번호를 입력할 수 있는 회원가입 창이 보여진다 |
| 2 | 아이디와 닉네임, 비밀번호를 입력 후 회원가입 버튼 클릭 | 회원가입에 성공했다는 메시지가 보여지고 로그인 창으로 이동한다 |
| TEST INFORMATION | | |
| Test Case Number : 5.1.2 | | |
| Test Case Name : 로그인 | | |
| Test Type : 실패 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 회원가입 창에 접속 | 아이디와 닉네임, 비밀번호를 입력할 수 있는 회원가입 창이 보여진다 |
| 2 | 아이디와 닉네임, 비밀번호를 입력 후 회원가입 버튼 클릭 | 회원가입에 실패한 원인에 대한 메시지가 보여지며 회원가입에 실패한다. |

Table 69 : Testing for sign up

## 5.2. 로그인

5.2.1. 로그인 성공

5.2.2. 로그인 실패

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.2.1 | | |
| Test Case Name : 로그인 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 로그인 창에 접속 | 아이디와 비밀번호를 입력할 수 있는 로그인창이 보여진다 |
| 2 | 아이디와 비밀번호를 기입 후 로그인 버튼 클릭 | 로그인에 성공하며 홈페이지로 들어가진다. |
| TEST INFORMATION | | |
| Test Case Number : 5.2.2 | | |
| Test Case Name : 로그인 | | |
| Test Type : 실패 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 로그인 창에 접속 | 아이디와 비밀번호를 입력할 수 있는 로그인창이 보여진다 |
| 2 | 아이디와 비밀번호를 기입 후 로그인 버튼 클릭 | 로그인에 실패하며 로그인에 실패했다는 알림창이 띄워진다. |

Table 70 : Testing for sign in

## 5.3. 댓글 작성

5.3.1. 댓글 작성 성공

5.3.2. 댓글 작성 실패

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.3.1 | | |
| Test Case Name : 댓글 작성 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 해당 영상의 댓글 리스트와 댓글을 작성할 수 있는 창이 보여진다 |
| 2 | 댓글을 작성 후 버튼 클릭 | 댓글 작성이 성공했다는 메시지가 유저에게 보이고 댓글 리스트 밑에 해당 댓글이 추가된다 |
| TEST INFORMATION | | |
| Test Case Number : 5.3.2 | | |
| Test Case Name : 댓글 작성 | | |
| Test Type : 실패 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 해당 영상의 댓글 리스트와 댓글을 작성할 수 있는 창이 보여진다 |
| 2 | 댓글을 작성 후 버튼 클릭 | 로그인이 필요하다는 메시지가 유저에게 보이고 로그인창으로 이동한다. |

Table 71 : Testing for add comment

## 5.4. 댓글 삭제

5.4.1. 댓글 삭제 성공

5.4.2. 댓글 삭제 실패

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.4.1 | | |
| Test Case Name : 댓글 삭제 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 해당 영상의 댓글 리스트와 댓글을 작성할 수 있는 창이 보여진다 |
| 2 | 삭제를 원하는 댓글의 휴지통 아이콘을 클릭 | 댓글 삭제가 성공했다는 메시지가 유저에게 보여지고 해당 댓글이 삭제된 댓글 리스트가 최신화 되어 보여진다 |
| TEST INFORMATION | | |
| Test Case Number : 5.4.2 | | |
| Test Case Name : 댓글 삭제 | | |
| Test Type : 실패 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 해당 영상의 댓글 리스트와 댓글을 작성할 수 있는 창이 보여진다 |
| 2 | 삭제를 원하는 댓글의 휴지통 아이콘을 클릭 | 자신의 댓글만 삭제할 수 있다는 메시지가 유저에게 보여지고 댓글 삭제에 실패한다. |

Table 72 : Testing for delete comment

## 5.5. 유저 별 댓글 리스트 호출

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.5 | | |
| Test Case Name : 유저 별 댓글 리스트 호출 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 마이 페이지에 접속 | 유저가 북마크로 저장한 영상의 리스트와 왼쪽 사이드에 내 댓글 버튼이 보여진다. |
| 2 | 내 댓글 버튼을 클릭 | 내가 쓴 댓글의 목록이 보여진다. |

Table 73 : Testing for call the list of comments by user

## 5.6. 북마크 저장

5.6.1. 북마크 저장

5.6.2. 북마크 저장 취소

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.7.1 | | |
| Test Case Name : 북마크 저장 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 영상과 영상 하단에 북마크 아이콘이 보여진다. |
| 2 | 북마크 아이콘을 클릭 | 북마크에 추가되었다는 메시지가 유저에게 보여지고 성공적으로 북마크 저장이 완료된다. |
| TEST INFORMATION | | |
| Test Case Number : 5.7.2 | | |
| Test Case Name : 북마크 저장 취소 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 영상과 영상 하단에 이미 저장이 완료되었다는 표시(체킹)가 된 북마크 아이콘이 보여진다. |
| 2 | 북마크 아이콘을 클릭 | 북마크가 해제되었다는 메시지가 유저에게 보여지고 성공적으로 북마크 저장 취소가 완료된다. |

Table 74 : Testing for bookmark

## 5.7. 북마크 리스트 호출

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.7 | | |
| Test Case Name : 북마크 리스트 호출 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 마이 페이지에 접속 | 유저가 북마크로 저장한 영상의 리스트가 보여진다. |
| 2 | 마이 페이지의 다른 페이지 접속 후, 다시 북마크 배너를 클릭 | 유저가 북마크로 저장한 영상의 리스트가 보여진다. |

Table 75 : Testing for call the list of bookmark

## 5.9. 영상 평가

5.9.1. 영상 평가

5.9.2. 영상 평가 취소

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.9.1 | | |
| Test Case Name : 영상 평가 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 영상과 영상 하단에 영상 평가 아이콘이 보여진다. |
| 2 | 영상 평가 아이콘을 클릭 | 영상 평가가 완료되었다는 메시지가 유저에게 보여지고 성공적으로 영상 평가가 완료된다. |
| TEST INFORMATION | | |
| Test Case Number : 5.9.2 | | |
| Test Case Name : 영상 평가 취소 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 영상 시청 페이지에 접속 | 영상과 영상 하단에 이미 평가가 완료되었다는 표시(체킹)가 된 영상 평가 아이콘이 보여진다. |
| 2 | 영상 평가 아이콘을 클릭 | 영상 평가가 취소되었다는 메시지가 유저에게 보여지고 성공적으로 영상 평가 취소가 완료된다. |

Table 76 : Testing for evaluation of video

## 5.10. 경기 일정

|  |  |  |
| --- | --- | --- |
| TEST INFORMATION | | |
| Test Case Number : 5.10 | | |
| Test Case Name : 특정 날짜 경기 일정 호출 | | |
| Test Type : 성공 | | |
| TEST STEPS | | |
| # | Step | Expected Result |
| 1 | 경기 일정 페이지 클릭 | 현재 해당되는 달(month)의 날짜의 리스트가 가로 스크롤로 나타난다. |
| 2 | 특정 날짜 클릭 | 만일 해당 날짜의 경기 일정이 있다면, 경기 날짜, 팀, 경기 정보가 나타난다.  해당 날짜의 경기 일정이 없다면 정보가 없다는 메세지가 화면에 나타난다. |

Table 77 : Testing for call the specific schedule

# 6. Appendix : Contribution

## 6.1. Document

- 조완식

1. Introduction 작성

2.1, 2.2 하이라이트 영상 재생 페이지 관련 User / System Requirements 작성

2.4.3 2.4.4 하이라이트 영상 재생 페이지 관련 Use Case 작성

2.3 Domain Requirements 작성

2.5.2 Physical Requirements 작성

2.5.3 Design Requirements 작성

2.7 Validation & Criteria 작성

3.3.1 Whole Design Sequence Diagram 작성

3.3.3 영상 검색 Use Case Realization 작성

3.3.4 영상 시청 Use Case Realization 작성

3.3.7 개인 저장 Use Case Realization 작성

4.1 Highlight Generator Implementation 작성

- 김지수

1. Introduction 작성

2.1. 댓글 작성과 하이라이트 영상 평가 관련 User Requirements 작성

2.2. 댓글 작성과 하이라이트 영상 평가 관련 System Requirements 작성

2.4.5. 댓글 작성 관련 Use-Case model 작성

2.4.6. 하이라이트 영상 평가 Use-Case model 작성

3.3.5. 댓글 작성 관련 Use-Case Realization 작성

3.3.6. 영상 평가 관련 Use-Case Realization 작성

4.2. BackEnd Implementation 작성

5.5. 유저 별 댓글 리스트 호출 작성

5.7. 북마크 저장 작성

5.8. 북마크 리스트 호출 작성

5.9. 영상 평가 작성

- 홍성표

1 Introduction 작성

2.1, 2.2 회원가입/로그인, 경기 일정 관련 User/System Requirements 작성

2.4.2, 2.4.5 회원가입/로그인 관련 Use Case 작성

2.5.2 Physical Requirements 작성

2.7 Validation & Criteria 작성

3.2.2 Class Specification 작성

3.3.2 회원가입/로그인 Use Case Realization 작성

3.3.8 경기 일정 검색 Use Case Realization 작성

4.3 FrontEnd Implemetation 작성

5.1 5.2 5.3 5.4 Testing 작성

- 문민수

1. Introduction 작성

2.1, 2.2 하이라이트 개인 저장 페이지 관련 User / System Requirements 작성

2.4 Use-Case Model 작성

2.6.1 domain model diagram 작성

2.4.7 하이라이트 개인 저장 페이지 관련 Use Case 작성

3.2.1 class diagram 작성

3.2.2 class specification 작성

4.2. BackEnd Implementation 작성

4.3 FrontEnd Implemetation 작성

5.10 경기 일정 Testing 작성

## 6.2. Project

- 조완식

AI:

Hightlight Generator 관련

SoccerNet V2 Dataset PreProcessing

Video to Sound Extracting

Implementing VGGish

Implementing ResNet 152

Implementing NetVLAD Pooling

Make Fusioning multi-modal based action spotting deep learning model

Make Highlight Detecting module

Implementing Hightlight Generate module

- 김지수

Back-End :

Implementing signController/Router

Implementing commentController/Router

Implementing evalController/Router

Implementing bookmarkController/Router

- 홍성표

Front-End :

Design & Implementing 회원 가입,

Design & Implementing 로그인,

Design & Implementing 영상 재생 페이지,

Design & Implementing 홈페이지,

Design & Implementing 내 정보 페이지(유저 북마크 리스트, 작성한 댓글 리스트)

Design & Implementing 비디오 검색 페이지

- 문민수

Back-End :

Implementing scheduleController/Router

Implementing videoController/Router

Refactoring userController(jwt decoded)

Front-End :

Implementing 경기 일정 페이지

Implementing & Refactoring 내 정보 페이지(유저 북마크 리스트, 작성한 댓글 리스트)

Refactoring 회원가입, 로그인

# Reference.

|  |  |
| --- | --- |
| [1] | A. Deliège, "SoccerNet-v2 : A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos," in *CVPR*, 2021. |
| [2] | S. Hershey, "CNN Architectures for Large-Scale Audio Classification," in *ICASSP*, 2017. |
| [3] | K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *ILSVRC*, 2015. |
| [4] | R. Arandjelović, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016. |
| [5] | FFmpeg, *FFmpeg,* https://github.com/FFmpeg/FFmpeg. |
| [6] | J. F. Gemmek, "Audio Set: An ontology and human-labeled dataset for audio events," in *IEEE ICASSP* , New Orleans, 2017. |
| [7] | Pytorch, *Pytorch,* https://pytorch.org. |
| [8] | harritaylor, *torchvggish,* https://github.com/harritaylor/torchvggish, 2021. |
| [9] | Zulko, *moviepy,* https://zulko.github.io/moviepy/index.html. |