2023

Final Report

동아리 스케줄 공유를 통한 공동공간 예약 및 조회 시스템 WePlan

7 조 박종범, 정유환, 이승주, 김동령

목차

Table of Figures

Table of Tables

1. Introduction

1.1. Motivation and Background

많은 대학교에서는 다양한 동아리가 있으며, 각 동아리는 여러 활동을 진행한다. 이러한 동아리 활동이 원활하게 진행되기 위해서는 무엇보다도 동아리 구성원 간의 동아리 방과 같은 공용 공간의 이용 시간이 즉각적이고 직관적으로 공유되어야 한다. 그러나, 평소 학교에서 동아리 활동을 진행하면서, 이러한 동아리방 및 교내 시설 등 동아리 구성원 간의 공용 공간의 이용 시간에 대한 동아리 구성원 간 공유가 효율적으로 이루어지지 않아 불편함을 겪었던 경험이 여러 번 있었다.

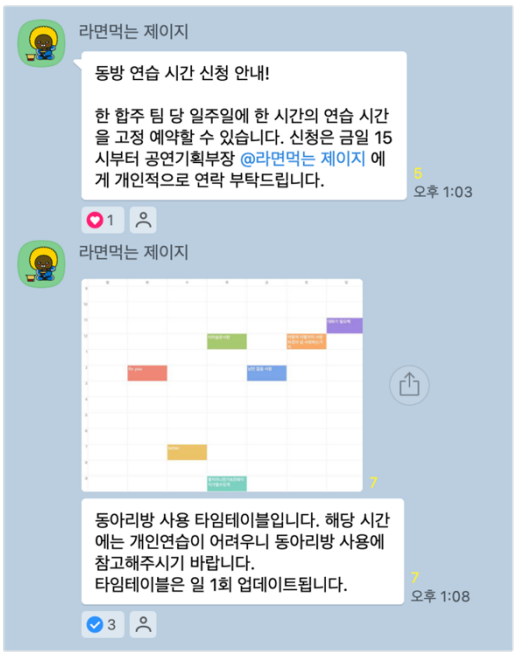


Figure 1-1 밴드 동아리 메신저

예를 들어, 밴드 동아리의 경우, 동아리방을 같이 사용하는 팀끼리 연습 시간을 조율해야 했는데, 이를 위해 공연부장에게 동아리 내 그룹 메신저를 통해 직접 예약을 받고 매주 시간표를 공지하는 방식으로 운영되었다(Figure 1-1). 이와 마찬가지로 테니스 동아리의 경우에도 동아리장에게 직접 예약 신청을 받고 확정된 해당 주차의 시간표를 메신저를 통해 공지하는 방식으로 테니스장 사용에 대한 예약 및 관리가 이루어져 왔다.

이렇게 외부 메신저를 통해 동아리 임원들이 동아리 일정을 공지하고, 동아리원들이 동아리 방 이용을 문의하는 방식에서는 일정에 대한 공지를 효율적으로 확인할 수 있는 UI가 제공되지 않아 동아리의 중요한 공지가 누락되거나 다른 메시지에 밀려 확인되지 못하는 문제가 있다. 또한, 동아리 구성원들 간의 착오로 인해 스케줄이 중복되거나 누락되는 문제가 존재했다.

또한, 위의 두 예시 모두 공통적으로 예약을 진행할 때 해당 시간에 예약이 가능한지는 동아리 임원만이 알고 있다. 이로 인해 동아리원은 매번 임원과의 연락을 통해 확인 및 승인을 받아야 하는 불편함이 있으며, 임원 역시 이를 확인하고 정리하여 동아리원들에게 공지해야 하는 번거로운 과정을 거쳐야 한다. 이러한 불편함을 겪었던 경험과 함께, 밴드 동아리 회장으로부터 동아리 공용 공간을 예약하는 시스템을 제작할 수 있는지에 대한 제안을 받았던 경험이 있어 이번 기회를 통해 이 프로젝트를 기획하게 되었다.

1.2. Project Problem Statement

- 효율적인 동아리 일정 공지 및 확인이 가능한 플랫폼의 부재
  - 일정에 대한 공지를 효율적으로 확인할 수 있는 UI가 제공되지 않아 중요한 공지가 누락되거나 다른 메시지에 밀려 확인되지 못하는 문제가 발생하고 있다.
- 수동으로 관리되는 동아리 일정 예약 및 문의
  - 동아리 구성원들 간의 착오로 인해 스케줄이 중복되거나 누락되는 문제가 발생하고 있다.
  - 예약을 진행할 때 해당 시간에 예약이 가능한지를 동아리 임원만 알고 있어 동아리원들은 매번 임원과의 연락을 통해 확인 및 승인을 받아야 하는 불편함이 있다.
  - 동아리 임원이 매회 예약 요청을 직접 확인하고 정리하여 동아리원들에게 공지하는 번거로운 과정이 필요하다.

1.3. Objectives

본 시스템의 목적은 동아리 예약 관리 기능을 하나의 플랫폼 내에서 통합하여 제공하여 원활한 동아리 예약 확인 및 관리를 가능하게 하는 것이다. 시스템은 사용자에게 각 동아리별로 현재 예약 현황을 한눈에 확인할 수 있는 타임 테이블을 제공한다. 동아리 구성원은 이를 기반으로 자신이 원하는 시간에 예약을 요청할 수 있다. 예약 요청은 동아리 관리자가 조회하고 승인 또는 거절할 수 있으며, 시스템은 중복 예약을 방지하기 위해 시간 중복 여부를 확인한다. 본 시스템의 가장 큰 특징 중 하나는 바로 실시간으로 예약과 관련된 모든 과정이 반영된다는 것이다. 궁극적으로, 사용자는 하나의 플랫폼(애플리케이션) 내에서 동아리 예약과 관련된 모든 기능을 편리하게 사용할 수 있게 된다.

1.4. Related Works/Trends

- IoT Space - 회의실 예약 시스템

Figure 1-2 IoT Space - 회의실 예약 시스템

Figure 1-3 IoT Space - 회의실 예약 시스템

해당 서비스는 특히 공유 오피스를 대상으로, 회의실 예약과 각 회의실의 예약 정보 및 일정(회의명, 예약시간, 예약자 명)등을 효율적으로 등록 및 조회하는 서비스이다. 서비스 내에서 각 회의실의 예약 현황 및 일정을 조회할 수 있고, 회의실 입구에 별도의 디스플레이 기기를 부착하여 해당 회의실의 예약 및 현황을 확인할 수 있도록 제공하고 있다. 또한, 별도의 앱이나 프로그램 설치 없이 데스크탑 및 모바일 기기로 해당 서비스에 접속할 수 있는 접근성을 확보하고 있다.

- Slack

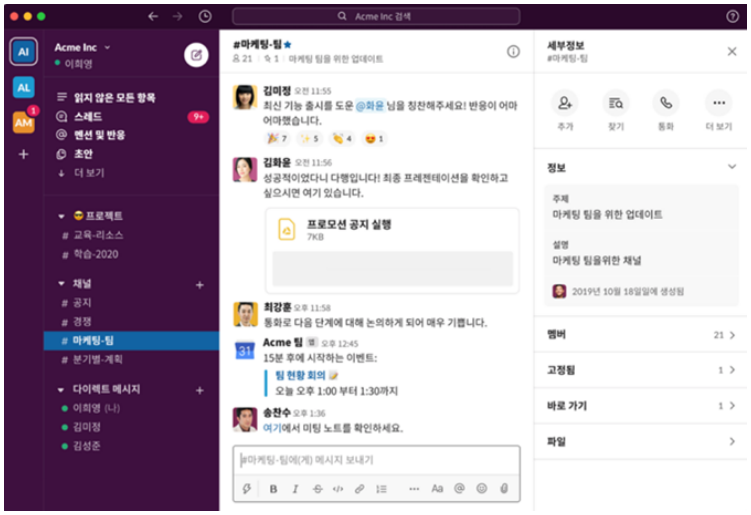


Figure 1-4 Slack

슬랙은 클라우드 컴퓨팅 기반 인스턴트 메신저 및 프로젝트 관리용 협업 툴이다. 현재 웹 앱과 안드로이드/iOS 모바일 앱 형태로 제공이 되고 있다. 슬랙은 메신저 기능 중심의 업무용 협업 툴로서 프로젝트 단위로 구성한다. 메시징, 채널, 음성 및 영상 통화, 파일 공유 등의 기능을 제공하고 있다. 또한, 개방성을 중점으로 슬랙 API를 제공하여 사용자가 필요한 어플리케이션을 만들어 연동할 수 있도록 하고 있다.

- Notion

Figure 1-5 Notion

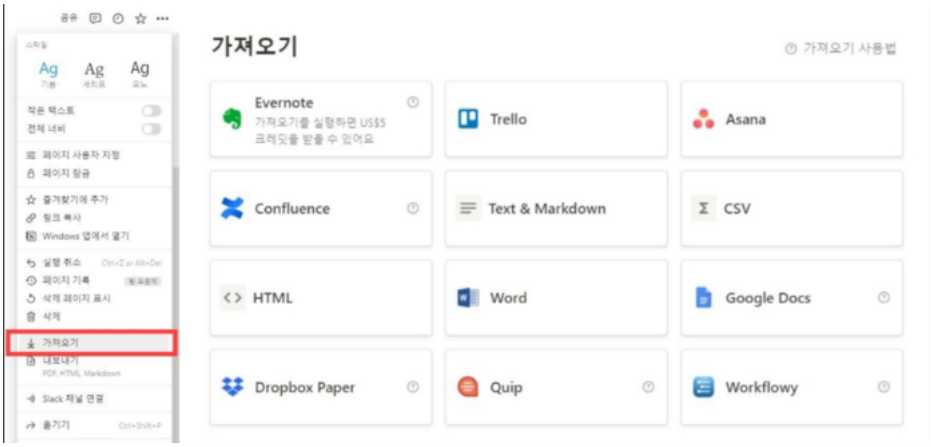


Figure 1-6 Notion

Notion은 개인적인 문서 및 노트 작업을 중심으로, 이를 다른 사용자와 공유 및 관리할 수 있는 협업 기능을 제공하고 있는 서비스이다. 워크스페이스 단위로 프로젝트를 나누고 문서를 페이지 단위로 작성하는 구조이며, 페이지안에 이미지, 영상, 링크, 파일, 코드, 외부 API 등 다양한 형태의 데이터를 삽입할 수 있으며 페이지안에 페이지도 삽입할 수 있다. 개인의 문서 작성을 중심으로 다양한 기능을 임베드하여 일정 관리도 수행할 수 있으며, 페이지 단위로 다른 사용자를 초대하여 협업도 진행할 수 있다.

1.4.1. Current Trends

현재 일정 관리와 예약 시스템은 사용자들이 어디서든 접근할 수 있고, 모바일 환경에서도 편리하게 서비스를 이용할 수 있도록 제공되고 있다. 이러한 시스템은 사용자들이 일정을 관리하고 예약을 신속하게 처리할 수 있도록 돕는 기능을 제공한다. 뿐만 아니라, 이러한 시스템은 실시간으로 업데이트 사항을 확인할 수 있는 유비쿼터스한 환경을 제공하고 있다. 사용자들은 언제 어디서든 시스템의 최신 정보를 확인하고 업데이트된 내용을 받아볼 수 있다. 이는 사용자들이 항상 최신 정보를 가지고 일정을 관리하고 예약을 진행할 수 있게 해준다.

또한, 이러한 일정 관리 및 예약 시스템은 단순한 일정 등록과 공유 기능을 넘어서 협업에 필요한 다양한 기능들을 함께 제공하고 있는 추세이다. 예를 들어, 여러 사용자들이 동시에 일정을 공

유하고 업무 협업을 할 수 있는 기능, 일정을 효율적으로 조율하고 관리할 수 있는 기능, 그리고 다양한 알림 기능 등이 포함되어 있다. 이를 통해 사용자들은 팀원과의 협업을 원활하게 진행하고 일정을 효율적으로 관리할 수 있다.
따라서, 현재 일정 관리와 예약 시스템은 사용자들에게 편리하고 유연한 환경을 제공하며, 다양한 기능들을 통해 협업과 일정 관리를 보다 효율적으로 할 수 있도록 지원하고 있다.

1.4.2. Drawbacks / Weakness in the Current Trends or Existing Solutions

해당 프로젝트에서 제공하고자 하는 서비스는 '대학교 커뮤니티 내에서 동아리 별 일정 관리 및 동아리 공간 예약 시스템' 이다. 그러나 앞서 제시한 세 서비스는 학교 커뮤니티 뿐만 아니라 다양한 프로젝트를 일반적으로 포괄할 수 있는 '올인원 협업툴'로 제공되고 있기 때문에, 프로그램의 규모자체가 매우 크다. 따라서, 현재 상용화된 서비스들은 다양한 기기에서 접속할 수 있는 접근성은 보장하고 있으나, 보다 빠르고 간편한 사용성을 제공하기에는 어려운 환경이다. 이에 따라, WePlan은 교내 동아리 행사 일정, 동아리 별 공용 공간 예약을 조회할 수 있는 타임 테이블을 제공하고 동아리 공간 사용 일정을 빠르게 추가하고 삭제할 수 있는 두 기능을 중심으로 하여, 대학생 커뮤니티에 적합한 가볍고 직관적인 서비스를 제공하고자 한다.

1.5. Proposed Solution

WePlan은 대학교 내 동아리 활동을 지원하기 위한 서비스로, 동아리 별 일정 관리와 동아리 공간 예약 시스템을 제공하는 데 방점을 두고 있다. 이에 반해 기존의 일반적인 협업 툴은 학교 커뮤니티뿐만 아니라 다양한 프로젝트를 포괄하는 대규모의 서비스로 제공되고 있기 때문에, 규모 자체가 매우 크다. 따라서, 현재 상용화된 서비스들은 다양한 기기에서 접속할 수 있는 접근성은 보장하고 있지만, 사용성 측면에서 보다 빠르고 간편한 환경을 제공하기에는 어려움이 있다.
이에 반해 WePlan은 대학생 커뮤니티의 특성을 고려하여 가벼우면서도 직관적인 사용자 경험을 제공하고자 한다. 사용자들은 간편하게 일정을 등록하고 조회할 수 있으며, 동아리 공간 예약도 신속하게 처리할 수 있다. 이를 통해 사용자들은 더욱 편리하고 효과적으로 동아리 활동을 계획하고 관리할 수 있다.

- 예약을 하나의 타임 테이블로 시각화
  기존에는 동아리 예약을 위해 동아리장이 수동으로 테이블을 만들고 메신저 등을 통해 예약 정보를 공지해야 했다. 이는 여러 툴과 플랫폼을 거치는 번거로운 과정이다. 그러나 우리는 이러한 불편함을 해소하기 위해, 모든 예약을 하나의 테이블로 관리하고, 동아리장과 동아리원 모두가 직관적인 UI를 통해 예약을 확인하고 관리할 수 있는 플랫폼(애플리케이션)을 제공하고자 한다.

- 하나의 시스템으로 관리되는 예약 요청 및 관리
  동아리 예약 및 취소 요청을 구두로 전달받고, 매번 업데이트할 때마다 동아리원에게 알리고 테이블을 수정하는 것은 매우 비효율적이다. 따라서, 우리는 동아리원이 주어진 타임 테이블을 참고하여 간편하게 스케줄 예약 및 취소를 할 수 있는 기능을 제공하고자 한다. 또한, 각 공간의 관리자가 예약 요청을 한눈에 관리하고 승인 및 거절을 간편하게 할 수 있는 페이지를 제공하고자 한다. 이를 통해 예약 관리 과정을 보다 효율적으로 수행할 수 있다.

- 중복 예약 문제의 근본적 차단
  기존의 방식에서는 착오 등으로 인해 중복 예약 문제가 발생할 수 있었다. 그러나 본 애플리케이션은 시스템적으로 이러한 문제를 근본적으로 차단할 수 있는 차별화된 기능을 제공한다. 이를 통해 중복 예약을 방지함으로써 예약 관리의 효율성을 높일 수 있다.

1.6. Expected Results

본 시스템은 교내 동아리 및 자치활동에서 폭넓게 사용될 수 있으며, 향후 추가적인 발전을 통해 본 시스템은 학생들이 쉽게 시스템을 사용하고 활동을 조직하고 참여할 수 있도록 도움이 되는 몇 가지 기능들을 추가하여 학교 내에서 활동하는 학생들에게 많은 편의 및 혜택을 제공할 수 있을 것으로 기대한다. 또한 기존의 메신저 애플리케이션이나 공유 및 협업 툴보다 보다 교내 활동에 초점을 맞추어 기능을 제공하고 있으므로, 향후 에브리타임과 같이 다양한 대학의 요구에 맞춘 기능을 추가하고, 대학별로 커스터마이즈된 시스템을 제공하거나, 교내 학사 시스템과 연동하여 학교의 학사 시스템이나 일정 관리 시스템과의 연동을 통해 활동 일정을 자동으로 동기화하는 등의 기능을 추가할 수 있을 것이다.

1.7. Risk and Concerns

이 프로젝트에서 가장 큰 도전은 다양한 사용자가 불편을 겪는 상황을 정확하게 이해하고, 이를 시스템에 반영하는 것이다. 각각의 동아리와 그룹이 서로 다른 운영 방식과 요구 사항을 가지고 있기 때문에, 이들 모두를 만족시키는 범용적인 시스템을 만드는 것은 쉽지 않다. 사용자 인터뷰나 설문조사 등을 통해 실제 사용자의 필요성을 파악하고, 이를 기반으로 기능 목록을 작성하는 과정이 필요하다.
또한, 여러 사용자가 동시에 같은 시간대를 예약하려고 할 때 발생할 수 있는 동시성 문제 역시 중요한 과제이며, 데이터의 일관성을 유지하면서 충돌 없이 예약 처리를 해야 하기 때문에 성능이 떨어지지 않으면서 효율적인 동시성 관리 방법을 찾아야 한다.
사용자 인터페이스 디자인 역시 중요한 과제이다. 사용자가 쉽게 예약할 수 있도록 직관적이고 친숙한 인터페이스를 제공해야 한다.
다음으로, 사용자 알림 기능은 이 시스템에서 중요한 부분을 차지하게 될 것이다. 사용자가 예약을 하거나, 예약이 승인되거나 거절되었을 때 즉시 알림을 받아야 한다. 이러한 알림 기능은 사용자 경험을 향상시키는 데 크게 기여할 수 있다. 사용자는 시스템에 대해 지속적으로 인지하고 상호작용할 수 있으며, 필요한 정보를 얻어서 효율적으로 자신의 일정을 관리할 수 있다. 하지만 이러한 알림 기능 구현은 단순하지 않기 때문에 추가적인 기술적 도전 과제이다.
마지막으로 보안 문제도 중요하다. 사용자 개인 정보와 예약 정보 등 민감 정보를 안전하게 보호하는 기능이 필요하며, 권한 있는 사용자만 특정 작업(예: 예약 승인)을 수행할 수 있어야 한다.
이런 복잡한 요소들에 대응하기 위해서는 충분한 계획과 설계 단계가 필수적이다. 기술적 지식과 경험 외에도 심층적인 사용자 연구 및 테스트가 요구된다.

- 관리자 권한 위임

각 동아리 채널의 관리자가 변경되는 경우 어떠한 방식으로 관리자 권한을 위임할 것인지에 대한

지정이 필요하다. 각 관리자 계정에 대하여 계정 하나를 지정해 놓고 자치적으로 양수자에게 양도하는 방식으로 진행할 것인지, 아니면 관리자 권한을 인계하는 기능을 사용할 것인지에 대한 선택이 필요할 것이다. 전자의 경우, 계정이 외부로 유출되는 등의 보안 이슈가 있을 것이고, 후자의 경우 해당 기능을 위한 별도의 구현이 필요할 것이다.

- 동일 사용자의 다중 예약 시도 가능성

동일 사용자가 무분별한 다중 예약을 시도하는 경우, 해당 요청이 서버에 부하를 일으킬 가능성이 존재한다. 따라서 한 계정 당 일정 기간 내에 최대 예약 신청 횟수를 제한하거나, 예약 간 제한시간을 두는 등의 추가적인 구현이 필요할 것이다.

- 사용자 정보 수집

사용자의 개인정보 수집의 범위를 어디까지 두어야 하는지에 대한 면밀한 검토가 필요하다. 학교 구성원 여부에 대한 판별 및 타임 테이블에 보여주기 위한 최소한의 범위 내에서 사용자 정보를 수집해야 할 것이며, 또 타임 테이블에는 이름, 소속 등의 정보를 어디까지 노출할 것인지에 대한 논의가 필요할 것이다. 마지막으로, 이들을 외부에 유출시키지 않기 위해 추가적인 보안 조치가 필요할 것이다.

- 예약 마감 시간

인가제로 예약이 진행되는 만큼, 신청하는 사용 기간이 신청하는 시점 바로 직후인 경우 관리자의 예약 승인이 곧바로 이루어지기 어렵다. 따라서, 직전에 신청되는 예약을 동아리 구성원 내에서 자율적으로 처리하도록 할 것인지, 아니면 일정 시간 이후의 예약은 자동으로 취소되도록 할 것인지에 대하여 고찰이 필요하다.

2. Requirement Efforts

2.1. Main Features of the System-to-be

- 타임 테이블 시각화
  - 각 동아리의 공용 공간 예약 일정을 타임 테이블 형식으로 시각화하여 예약/공실 여부를 쉽게 확인할 수 있으며 예약하는 팀마다 색을 다르게 하는 등 사용자가 쉽게 확인할 수 있는 직관적인 UI를 제공한다.
- 타임 테이블 시간 예약
  - 사용자는 각 동아리의 타임 테이블을 참고하여 예약 현황을 빠르게 조회하고 예약/취소를 할 수 있다.
- 관리자 승인
  - 각 동아리의 관리자가 사용자의 예약을 승인하거나 시간에 따라 자동승인 될 수 있도록 하여 중복 예약을 방지한다.
- 동아리 채널 개설
  - 동아리의 관리자가 해당 서비스를 시작하기 위한 과정을 구현하여 시작하기 수월하도록 한다.

2.2. High Level Requirements

- 동아리 채널의 타임 테이블에 대한 실시간 업데이트가 필요하다. 타임 테이블 시각화 시, 예약/취소는 실시간으로 시스템에 업데이트되어, 다른 사용자가 동시에 예약 현황을 실시간으로 확인할 수 있도록 해야 한다. 이로써 사용자들에게 최신 정보를 제공한다. 또, 중복된 예약으로 충돌이 생기지 않도록 해야 한다.
- 관리자 권한 및 인증 프로세스가 필요하다. 각 공간의 관리자에게 관리 권한을 부여할 때, 별도의 관리자 인증 및 권한 부여 프로세스를 구현해야 한다. 이를 통해 관리자와 일반 사용자의 역할 및 권한이 구분되며 예약 관리를 효율적으로 처리할 수 있다.
- 타임 테이블 시각화를 할 때 예약/취소가 다른 사용자에게도 실시간으로 보일 수 있도록 가능한 빠른 업데이트가 필요하다. 공간에 대한 예약 승인은 각 동아리 채널의 관리자가 담당하는데 이 관리자에게 관리 권한을 줄 때 다른 사용자와는 다른 인증 과정이 필요하다.
- 예약 신청 결과가 관리자에 의해 결정되었을 때 사용자가 알림을 받을 수 있는 형태로 제공되어야 한다.

2.3. Architecture View of Use Case Model

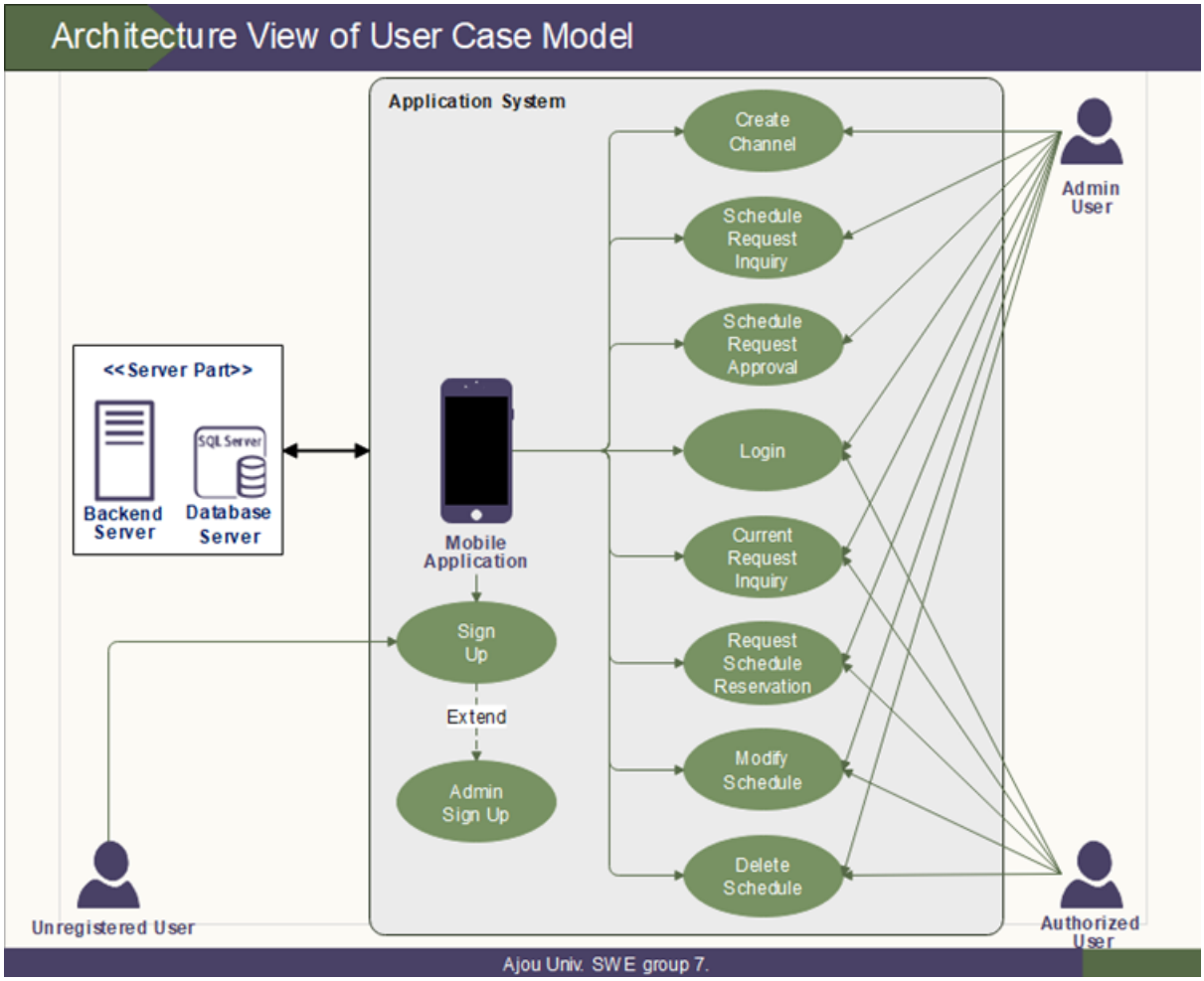


Figure 2-1 Architecture View of Use Case Model

2.4. User Requirements

Table 2-1 User Requirement 001

No.	USER_RQ_001
Title	시스템을 이용하는 유저는 관리자와 가입 유저로 구분되어야 한다.
Description	본 동아리 채널 시스템은 학생들이 주도적으로 동아리 공간을 예약 및 사용할 수 있도록 하는 것을 목적으로 한다. 동아리 구성원은 크게 동아리 활동을 총괄하는 동아리장과 그 외 동아리원으로 나뉘어지므로, 동아리 채널 시스템 역시 이에 대응하여 각 동아리의 채널을 관리하는 관리자와 일반 회원인 가입 유저로 나뉘어야 한다.

Table 2-2 User Requirement 001-01

No.	USER_RQ_001_01
Title	관리자 유저는 가입 유저의 모든 기능 및 동작을 제공하여야 한다.
Description	가입 유저와 같이 관리자 유저 역시 같은 동아리 공간을 공유하고 사용하는 학생이다. 만약 관리자 유저에게 가입 유저와 동일한 기능에 접근할 수 있는 기능을 부여하지 않는다면, 관리자 유저는 가입 유저용 계정을 따로 만들어 두 개의 계정을 관리해야 할 수도 있으며, 유저는 이런 불편함을 겪으며 시스템을 사용하려고 하지 않을 것이다.

Table 2-3 User Requirement 001-02

No.	USER_RQ_001_02
Title	관리자 유저는 가입 유저가 접근할 수 없는 채널 관리 페이지에 대한 접근 권한을 갖는다.
Description	다른 사용자의 스케줄 예약을 승인 및 거절할 수 있는 권한을 불특정 다수의 동아리 유저에게 부여한다면 예약의 무분별한 승인 및 무단 취소 등 전체 시스템에 혼란을 가져올 가능성이 크다. 따라서 이러한 관리 권한은 한 명의 동아리 관리자 유저에게만 부여되어야 한다.

Table 2-4 User Requirement 002

No.	USER_RQ_002
Title	시스템은 서비스를 이용하는 모든 유저에 대하여 유효한 인증 절차를 마련하여야 한다.
Description	본 동아리 채널 시스템은 보다 빠르고 편리한 동아리 공간 이용을 위하여 대학생들을 대상으로 하는 동아리 예약 및 관리 시스템이다. 동아리의 예약 관리는 오롯이 동아리 구성원들만의 몫이며, 본 시스템에 접근 가능한 유저는 대학

	교 학생 구성원 및 해당 동아리의 구성원이어야 한다. 그러므로 해당 요구사항이 충족되지 않는다면 중대한 문제가 발생할 것이다. 따라서 시스템은 인가된(시스템에 등록된) 유저만이 시스템을 이용할 수 있도록 해야 한다.

Table 2-5 User Requirement 003

No.	USER_RQ_003
Title	유저는 자동 로그인을 통해 간편하고 빠르게 시스템에 접속할 수 있어야 한다.
Description	동아리의 일정은 실시간으로 갱신되므로, 유저는 이를 시스템에 접근하여 주기적으로 확인하게 된다. 만약 시스템 접근 시마다 반복적으로 인증 절차를 거치게 된다면, 이는 사용자 편의성을 저해할 것이다. 해당 시스템의 유저의 사용 주기와 접근성을 고려하였을 때 시스템에 한 차례 인증을 거친 유저는 자동 로그인 시스템으로 로그인을 유지할 수 있도록 해야 한다.

Table 2-6 User Requirement 004

No.	USER_RQ_004
Title	유저의 인증 정보가 탈취되더라도 탈취자가 해당 정보로 인증할 수 없도록 해야 한다.
Description	악의적인 사용자가 다른 사용자의 정보를 탈취해 시스템의 인증 절차를 거쳐 시스템에 접근하게 된다면, 시스템은 중대한 위협에 직면할 수 있다. 따라서 시스템은 유저 정보가 탈취되더라도 실제 데이터를 확인할 수 없도록 정보를 암호화해 저장하고 있어야 한다.

Table 2-7 User Requirement 005

No.	USER_RQ_005
Title	관리자 유저는 동아리 채널을 생성, 수정, 삭제할 수 있어야 한다.
Description	스케줄은 유저의 예약 일정을 지칭하는 연속적인 시간의 단위이며, 채널은 동일한 공간 내에서 공유되는 스케줄의 집합을 공유하는 가상 공간이다. 시스템은 각 동아리마다 개별적으로 시간을 공유하고 관리할 수 있도록 관리자 유저에게 개별 채널을 관리하는 기능을 제공해 채널을 생성, 수정, 삭제하며, 가입 유저들이 이에 접근할 수 있도록 하여야 한다.

Table 2-8 User Requirement 006

No.	USER_RQ_006
Title	유저는 채널의 스케줄 예약 현황을 타임 테이블 형태로 조회할 수 있어야 한다.

Description	유저의 예약이 등록되면 이를 시각적으로 보여주는 수단이 필요하다. 따라서 시스템은 유저가 현재의 예약 현황들을 파악할 수 있도록 타임 테이블 형식을 사용하여 예약 정보를 제공하여야 한다.

Table 2-9 User Requirement 006-01

No.	USER_RQ_006_01
Title	유저는 타임 테이블 상에서 각 스케줄을 쉽게 구분하고 이해할 수 있어야 한다.
Description	만약 각각의 스케줄 블록 간의 경계가 모호하거나, 각 스케줄 블록이 시각적으로 동일하거나, 스케줄 블록이 서로 유사한 색상으로 제공되는 경우, 유저는 예약을 확인하는 데 큰 어려움을 겪을 것이다. 이는 사용자 경험을 저해하는 요소로 작용함과 동시에, 보다 편리한 동아리 스케줄 예약 플랫폼을 제공하고자 하는 본 프로젝트의 취지와 반하는 사항이다. 따라서, 시스템은 타임 테이블 상의 스케줄들이 서로 구분이 가능하도록 보색 관계 등의 디자인을 이용하여 뚜렷한 구분 경계를 가지도록 하여야 한다.

Table 2-10 User Requirement 007

No.	USER_RQ_007
Title	유저는 타임 테이블에서 각 스케줄의 상세한 정보를 추가적으로 확인할 수 있어야 한다.
Description	예약에 관한 상세한 정보를 제공하지 않는다면 유저는 누가, 어떤 목적으로 해당 시간을 예약한 것인지 파악할 수 없다. 예약에 관한 상세한 정보는 동아리 구성원 누구나가 파악할 수 있도록 상세한 정보를 제공하여야 한다.

Table 2-11 User Requirement 008

No.	USER_RQ_008
Title	유저는 타임 테이블에서 시스템 접속 시점부터 한 달 까지의 예약 현황을 조회할 수 있어야 한다.
Description	유저에게 단기간의 예약 현황만을 제공하는 것은 유저가 장기적인 계획을 수립하는 것을 어렵게 만든다. 유저가 충분한 정보를 기반으로 일정을 계획할 수 있도록 접속 시점부터 한 달 까지의 예약 현황들을 타임 테이블에서 확인할 수 있도록 하여야 한다.

Table 2-12 User Requirement 009

No.	USER_RQ_009

Title	유저는 채널에서 이용 가능한 시간 중 원하는 시간을 선택하여 스케줄을 예약할 수 있어야 한다.
Description	유저는 자신이 속한 채널에서 모바일 앱을 통해 시공간의 제약 없이, 이용 가능한 시간 중 본인이 원하는 시간을 자유롭게 선택하여 스케줄을 예약할 수 있어야 한다.

Table 2-13 User Requirement 009-01

No.	USER_RQ_009_01
Title	유저가 이미 존재하는 스케줄 블록과 시간이 겹치는 스케줄 예약을 신청할 수 없어야 한다.
Description	이전에 이미 다른 가입 유저가 스케줄을 예약하여 승인 및 반영된 시간에 대해서는 예약이 불가능하게 하여 충돌을 방지해야 한다.

Table 2-14 User Requirement 010

No.	USER_RQ_010
Title	관리자는 관리하고 있는 채널의 예약 현황을 별도로 확인할 수 있어야 한다.
Description	관리자는 별도로 가입 유저의 예약내용을 확인하고, 예약의 타당성 및 중요도 등을 예약 상세정보 조회를 통해 판단할 수 있어야 한다.

Table 2-15 User Requirement 011

No.	USER_RQ_011
Title	관리자는 가입 유저의 예약을 승인 혹은 거절할 수 있어야 한다.
Description	시스템은 관리자가 예약을 한 눈에 조회하고, 이를 승인 및 거절하는데 필요한 모든 기능들을 제공해주어야 한다.

Table 2-16 User Requirement 012

No.	USER_RQ_012
Title	가입 유저의 예약 신청이 승인 혹은 거절될 경우, 가입 유저는 이를 앱을 통해서 알림을 받아야 한다.
Description	예약의 승인 여부를 유저에게 직접적으로 알리지 않는다면 유저는 이를 확인하기 위해 주기적으로 시스템에 접속해야 하는 불편을 감수해야 한다. 사용자 경험 증진을 위해 시스템에서 이를 자동으로 가입 유저에게 알려주어야 한다.

Table 2-17 User Requirement 013

No.	USER_RQ_013
Title	유저는 타임 테이블에서 최신의 예약 정보를 확인할 수 있어야 한다.
Description	모든 유저의 입장에서, 다수의 유저가 동시에 접근하는 타임 테이블은 스케줄 예약에 대한 관리자 승인, 가입 유저의 스케줄 취소 등으로 인해 임의의 시간에 자주 변경될 것이다. 이러한 변경 사항들이 최대한 빠르게 타임 테이블에 업데이트되어야 모두가 똑같은 타임 테이블을 조회할 수 있게 되며, 예약 신청에 있어서의 충돌을 사전에 1차적으로 방지할 수 있다. 또한, 관리자 유저도 새로운 스케줄 예약을 이미 승인된 스케줄과의 상호 관계를 판단하여 승인 여부를 결정해야 하기 때문에, 해당 요구사항은 전반적인 서비스 품질을 향상시키는 데 있어 중요한 요소이다.

2.5. System Requirements

2.5.1. Functional Requirements

Table 2-18 System Requirements - Functional Requirement 001

No.	SYS_RQ_FR_001	**Related Requirement**	USER_RQ_001 USER_RQ_002
Title	시스템은 회원가입 시 관리자 권한을 인증하기 위해 관리자 인증번호를 요청하고 이를 확인해야 한다.		
Description	사용자가 회원 가입을 진행할 때, 관리자 권한을 인증하기 위해 시스템은 관리자 인증번호를 요청하고 이를 확인해야 한다.		

Table 2-19 System Requirements - Functional Requirement 002

No.	SYS_RQ_FR_002	**Related Requirement**	USER_RQ_002
Title	시스템은 유저가 아이디, 비밀번호를 통해 로그인을 진행하도록 하며, 성공적으로 인증이 되어야 로그인이 되도록 한다.		
Description	시스템이 관리하는 아이디가 중복되지 않도록 체크하며 비밀번호 유효성 검사를 진행하여 보안을 높일 수 있도록 한다		

Table 2-20 System Requirements - Functional Requirement 003

No.	SYS_RQ_FR_003	**Related Requirement**	USER_RQ_003
Title	시스템은 토큰을 기반으로 하여 이전에 수행된 로그인에 대해서는 최소 1달 간 유저의 로그인 상태를 유지해야 한다		
Description	한번 로그인한 유저에게 추가적으로 로그인을 요청하지 않도록 시스템은 사용자		

	의 로그인을 유지할 있도록 한다.

Table 2-21 System Requirements - Functional Requirement 004

No.	SYS_RQ_FR_004	**Related Requirement**	USER_RQ_004
Title	시스템은 보안을 위해 비밀번호를 해싱 알고리즘을 통해 변환한 다음 저장하여야 한다.		
Description	사용자의 비밀번호는 보안을 최우선으로 고려하여, 해싱 알고리즘을 통해 변환된 형태로 저장된다. 이 방식은 원본 비밀번호를 직접적으로 알아낼 수 없게 하여 사용자의 정보를 보호한다.		

Table 2-22 System Requirements - Functional Requirement 005

No.	SYS_RQ_FR_005	**Related Requirement**	USER_RQ_005
Title	시스템은 채널을 생성, 수정, 삭제할 수 있는 관리자 페이지의 접근 권한 및 UI를 관리자 유저에게만 제공하여야 한다.		
Description	시스템은 관리자에게만 채널 생성, 수정, 삭제 권한을 부여하여, 해당 권한을 가진 사용자만이 새로운 채널을 생성, 수정, 삭제할 수 있는 사용자 인터페이스를 제공한다. 이를 통해 채널의 관리 및 통제를 강화할 수 있다.		

Table 2-23 System Requirements - Functional Requirement 006

No.	SYS_RQ_FR_006	**Related Requirement**	USER_RQ_006
Title	시스템은 모든 종류의 유저에게 채널의 스케줄 예약 현황을 타임 테이블 형태로 제공하여야 한다.		
Description	시스템은 사용자가 현재 채널의 스케줄 현황에 관한 정보들을 확인할 수 있는 타임 테이블을 제공하여, 각 스케줄의 일시 및 간단한 대표 정보를 타임 테이블 안에 블록 형태로 나타나도록 한다.		

Table 2-24 System Requirements - Functional Requirement 007

No.	SYS_RQ_FR_007	**Related Requirement**	USER_RQ_007
Title	시스템은 채널의 타임 테이블에서, 각 스케줄 블록의 스케줄 상세 정보를 확인할 수 있도록 하는 기능을 별도로 제공한다.		
Description	타임 테이블에서, 각 스케줄 예약에 대해 더욱 상세한 정보를 쉽게 확인할 수 있도록 하는 별도의 기능을 시스템이 제공해야 함을 명시한다. 이 기능은 예약된 시간, 예약자, 예약 목적 등 가입 유저가 스케줄 예약 신청 시 기입한 정보들을		

	포함해야 한다.

Table 2-25 System Requirements - Functional Requirement 008

No.	SYS_RQ_FR_008	**Related Requirement**	USER_RQ_008
Title	시스템은 접속 시점부터 최소 한달까지의 스케줄 정보를 채널의 타임 테이블에 출력해야 한다.		
Description	시스템은 사용자에게 한 동아리 채널의 최대 한달 간의 스케줄 정보를 제공하는 기능을 갖추고 있다. 이를 통해 사용자들은 중/장기적인 스케줄을 예약하고 관리할 수 있다.		

Table 2-26 System Requirements - Functional Requirement 009

No.	SYS_RQ_FR_009	**Related Requirement**	USER_RQ_009
Title	시스템은 사용자가 스케줄에 대한 예약을 신청할 수 있는 별도의 UI를 제공하여야 한다.		
Description	시스템은 사용자에게 한 동아리 채널의 최대 한 달 간의 스케줄 정보를 제공하는 기능을 갖추고 있다. 이를 통해 사용자들은 중/장기적인 스케줄을 예약하고 관리할 수 있다.		

Table 2-27 System Requirements - Functional Requirement 009-01

No.	SYS_RQ_FR_009_01	**Related Requirement**	USER_RQ_009_01
Title	시스템은 이미 예약이 이루어진 스케줄과 중복되는 새로운 스케줄을 가입 유저가 예약하는 것과, 그러한 새로운 스케줄 예약을 관리자가 승인하는 것이 발생하지 않도록 하여 스케줄 구성 시 충돌이 일어나지 않도록 한다.		
Description	시스템은 중복 예약과 같은 충돌이 일어나지 않도록 경고 메시지 출력 등의 방법을 통해 기능적 장치를 마련하여 데이터 충돌을 사전에 방지한다. 또한, 이를 통해 관리자와 가입 유저의 서비스 이용에 있어 혼선이 빚어지지 않도록 한다.		

Table 2-28 System Requirements - Functional Requirement 010

No.	SYS_RQ_FR_010	**Related Requirement**	USER_RS_010
Title	시스템은 관리자가 해당 채널의 예약 신청 현황을 조회할 수 있는 UI를 별도로 제공하여야 한다.		
Description	시스템은 관리자가 해당 채널의 예약 현황을 쉽게 확인할 수 있는 별도의 UI를 제공한다. 이 UI는 관리자에게 각 스케줄 예약 신청의 각 시간, 이용 목적, 신청		

	자 정보 등의 상세한 예약 정보를 조회하고 관리하는 기능을 제공하여, 효과적인 채널 운영을 지원한다.

Table 2-29 System Requirements - Functional Requirement 011

No.	SYS_RQ_FR_011	**Related Requirement**	USER_RS_011
Title	시스템은 관리자가 스케줄 예약 요청에 대한 승인 혹은 거절할 수 있는 별도의 UI를 제공하여야 한다.		
Description	시스템은 스케줄 예약 요청에 대한 승인 또는 거절 권한을 관리자에게만 부여한다. 오직 관리자에게만 이러한 결정을 할 수 있는 사용자 인터페이스를 제공하며, 이를 통해 예약 요청의 관리와 통제가 가능하게 된다. 특정 스케줄 신청을 승인하여 시스템에 반영됐을 경우, 그 스케줄과 시간대가 겹치는 신청은 모두 자동으로 거절되도록 한다.		

Table 2-30 System Requirements - Functional Requirement 012

No.	SYS_RQ_FR_012	**Related Requirement**	USER_RQ_012
Title	시스템은 관리자의 승인/거절 결과를 해당 예약을 신청한 가입 유저에게 알림을 발송하여 알린다.		
Description	시스템은 관리자의 승인 또는 거절 결과에 따라 가입 유저에게 알림을 발송하는 기능을 제공한다. 이를 통해 가입 유저는 신속하게 예약 상태를 확인하고 필요한 조치를 취할 수 있다.		

2.5.2. Non-Functional Requirements

Table 2-31 System Requirements - Non-Functional Requirement 001

No.	SYS_RQ_NFR_001	**Related Requirement**	
Title	시스템은 사용자 친화적이고 자연스러운 흐름의 UX를 제공해야 한다.		
Description	시스템은 사용자가 별다른 지체 없이 목표로 하는 기능을 쉽게 수행할 수 있도록 자연스러운 화면 설계 순서, 독립적인 기능 분리 등 사용자에게 편리한 이용 경험을 보장해야 한다.		

Table 2-32 System Requirements - Non-functional Requirement 002

No.	SYS_RQ_NFR_002	**Related Requirement**	USER_RQ_006 USER_RQ_008
Title	시스템은 한 채널의 스케줄을 주 단위의 타임 테이블로 표시하여 제공하여야 한다.		

Description	시스템은 한 채널의 스케줄을 주 단위로 보여주는 기능을 제공한다. 이는 타임 테이블 형태로 표현되며, 각각의 예약 사항이 해당하는 시간대에 표시된다. 이를 통해 사용자들은 한 눈에 주간 스케줄을 파악하고, 예약 현황을 확인할 수 있다

Table 2-33 System Requirements - Non-functional Requirement 003

No.	SYS_RQ_NFR_003	**Related Requirement**	USER_RQ_006 USER_RQ_006_01 USER_RQ_008
Title	시스템은 한 채널의 테이블 내에서 각 스케줄을 각기 다른 색상으로 표기하여야 한다.		
Description	시스템은 한 채널 내의 다양한 스케줄을 쉽게 구분할 수 있도록, 각각의 스케줄을 다른 색상으로 표기하는 기능을 제공한다. 이를 통해 사용자들은 각 스케줄이 무엇인지 명확하게 인식하고, 복잡한 일정 정보를 직관적으로 이해할 수 있다.		

Table 2-34 System Requirements - Non-functional Requirement 004

No.	SYS_RQ_NFR_004	**Related Requirement**	USER_RQ_006 USER_RQ_007
Title	시스템은 사용자가 특정 스케줄 블록을 선택 시 제공하는 스케줄 상세 정보를 모달 형태로 출력하여야 한다.		
Description	시스템은 사용자가 특정 스케줄을 선택할 경우, 모달 형태로 해당 스케줄의 자세한 정보를 제공하는 기능을 갖추고 있다. 이 모달에는 예약자의 이름, 예약 시간, 예약 내용 등 필요한 상세 정보가 포함되어 있어 사용자들이 쉽게 스케줄에 대한 정보를 파악할 수 있다.		

Table 2-35 System Requirements - Non-functional Requirement 005

No.	SYS_RQ_NFR_005	**Related Requirement**	USER_RQ_013
Title	시스템은 정해진 시간(30초)이 지나면 자동으로 타임 테이블 업데이트를 제공한다.		
Description	시스템은 정해진 시간(30초)이 경과하면 자동으로 타임 테이블을 업데이트하는 기능을 제공한다. 이를 통해 사용자들은 항상 최신의 스케줄 정보를 확인할 수 있어, 일정 관리의 효율성을 높일 수 있다.		

Table 2-36 System Requirements - Non-functional Requirement 006

No.	SYS_RQ_NFR_006	**Related Requirement**	
Title	시스템은 사용자의 입력에 1초 이내로 반응하여야 한다.		

Description	사용자는 사용자 동작에 대한 애플리케이션의 빠른 반응을 통해 좋은 사용자 경험을 유지할 수 있다.

Table 2-37 System Requirements - Non-functional Requirement 007

No.	SYS_RQ_NFR_007	**Related Requirement**	
Title	시스템은 데이터 요청에 대하여 3초 이내에 응답하여야 한다.		
Description	사용자 경험을 위해 서버와의 통신 속도는 빨라야 한다. 데이터 요청에 대한 응답은 정상적으로 네트워크에 연결된 상황 (Chrome Network Throttling Fast 3G 기준)에서 3초 이내에 이루어져야 한다.		

2.6. Use Case Scenario

Table 2-38 Use Case 1

Use Case	1
Case Name	로그인
Actors	유저(게스트(가입 유저), 관리자), 미가입 유저, 모바일 애플리케이션, 백엔드 애플리케이션 서버, DB 서버
Description	시스템을 이용하고자 하는 유저는 이전에 등록한 유저 ID와 유저 패스워드를 사용하여 로그인해야 한다.
Pre-conditions	유저는 회원가입을 통해 유저의 정보와 유저 ID, 유저 패스워드를 미리 등록해야 한다.
Post-conditions	시스템을 이용할 수 있게 된다.
Primary Flow	1. 유저는 로그인 창에 유저 ID와 유저 패스워드를 입력한다. 2. 서버는 모바일 애플리케이션으로부터 데이터를 수신한다. 3. 서버는 수신한 데이터와 저장된 데이터의 비교를 수행한다. 3.1. 만약 데이터가 일치하지 않으면, 서버는 모바일 애플리케이션으로 데이터 불일치 여부를 송신한다. 3.2. 일치되는 데이터가 있으면, 유저는 시스템을 이용할 수 있게 된다.

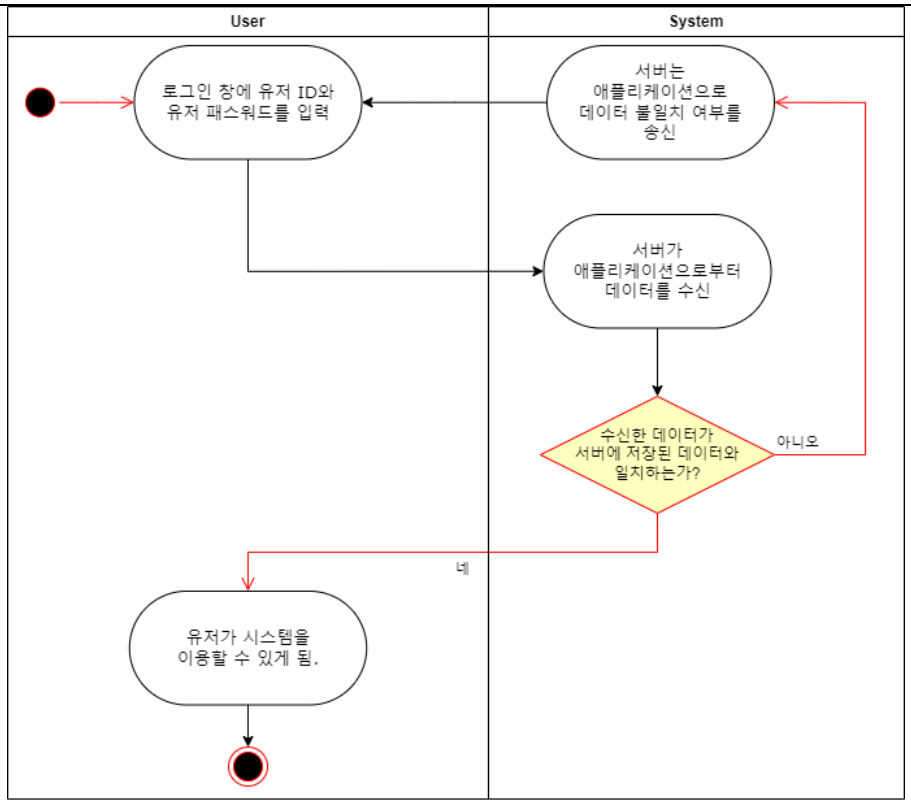
Activity Diagram	

Figure 2-2 Use Case 1 Activity Diagram

Table 2-39 Use Case 2

Use Case	2
Case Name	유저 등록
Actors	사용자(게스트(가입 유저) 및 관리자), 모바일 애플리케이션, 백엔드 애플리케이션 서버, DB 서버
Description	유저는 시스템을 이용하기 위해 등록이 필요하다.
Pre-conditions	유저는 애플리케이션을 사용하기 위한 디바이스를 소유해야 하며, 디바이스에 애플리케이션을 설치해야 한다.
Post-conditions	시스템을 이용할 수 있게 된다.
Primary Flow	1. 유저가 애플리케이션을 실행한다. 2. 유저가 로그인 페이지에서 회원가입 페이지로 이동한다. 3. 유저는 회원가입 폼에 유저 ID, 유저 비밀번호, 이름, 전화번호 등의 정보를 입력한다. 3.1. 만약 회원가입 폼에 입력되지 않은 정보가 있다면, 애플리케이션은 유저에게 경고 메시지를 전달한다. 3.2. 등록 폼에 모든 필수 정보가 입력되면, 해당 정보는 서버로 송신된다. 3.2.1. 만약 수신된 데이터가 아이디, 비밀번호, 전화번호의 생성 제약 조건을 만족하지 않으면, 서버는 유저에게 경고 메시지를 전달한다.

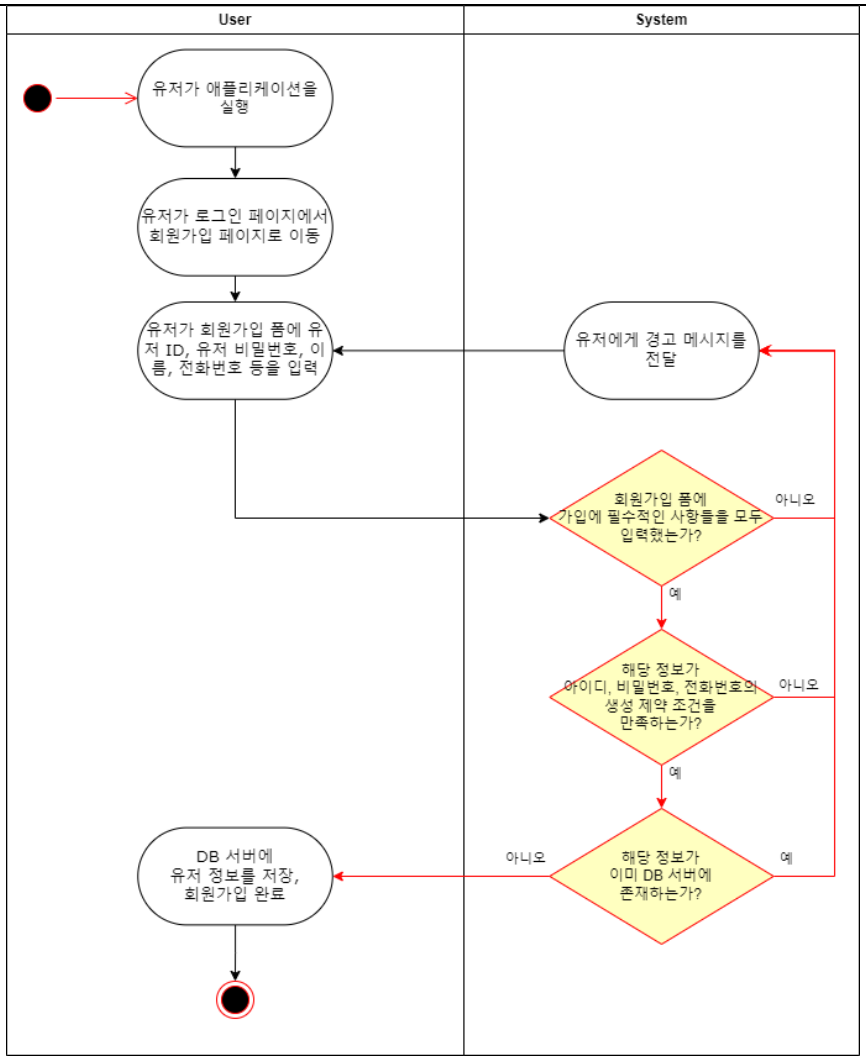
	3.2.2. 만약 수신된 데이터가 이미 서버에 존재하면, 서버는 유저에게 경고 메시지를 전달한다. 3.2.3. 만약 수신된 데이터가 서버에 존재하지 않는 데이터라면, 해당 데이터를 DB 서버에 저장한다. 3.3. 유저가 시스템에 로그인할 수 있게 된다.
Activity Diagram	

Figure 2-3 Use Case 2 Activity Diagram

Table 2-40 Use Case 3

Use Case	3
Case Name	동아리 채널 생성
Actors	유저(관리자), 모바일 애플리케이션, DB 서버
Description	관리자 권한을 가진 사용자가 동아리 채널을 생성한다.
Pre-conditions	유저는 관리자 권한이 부여된 계정으로 로그인을 성공해야 한다.

Post-conditions	채널 정보가 DB에 등록되고, 유저는 추가된 채널에 접근할 수 있게 된다.
Primary Flow	1. 관리자가 채널 리스트 보기 버튼을 눌러 리스트 메뉴를 연다. 1.1. 만약 현재 접속한 계정이 관리자 계정이 아니라면, 채널 추가 버튼을 활성화하지 않는다. 1.2. 관리자는 채널 추가 메뉴를 선택하고, 애플리케이션은 새 채널 추가 뷰를 보여준다. 1.3. 관리자는 채널 추가 폼에 추가하려는 채널의 이름과 장소 정보를 기입한다. 1.3.1. 만약 채널 추가 폼에 입력되지 않은 정보가 있다면, 애플리케이션은 유저에게 경고 메시지를 전달한다. 1.3.2. 등록 폼에 모든 필수 정보가 입력되면, 해당 정보는 DB 서버로 송신된다. 1.3.2.1. 만약 수신된 데이터와 일치하는 내용의 데이터가 이미 서버에 존재하면, DB 서버는 유저에게 경고 메시지를 전달한다. 1.3.2.2. 채널 정보를 DB 서버에 추가하여 새 채널을 추가하고, 서버는 애플리케이션을 통해 새 채널이 정상적으로 추가되었음을 보여주는 메세지를 전달한다.

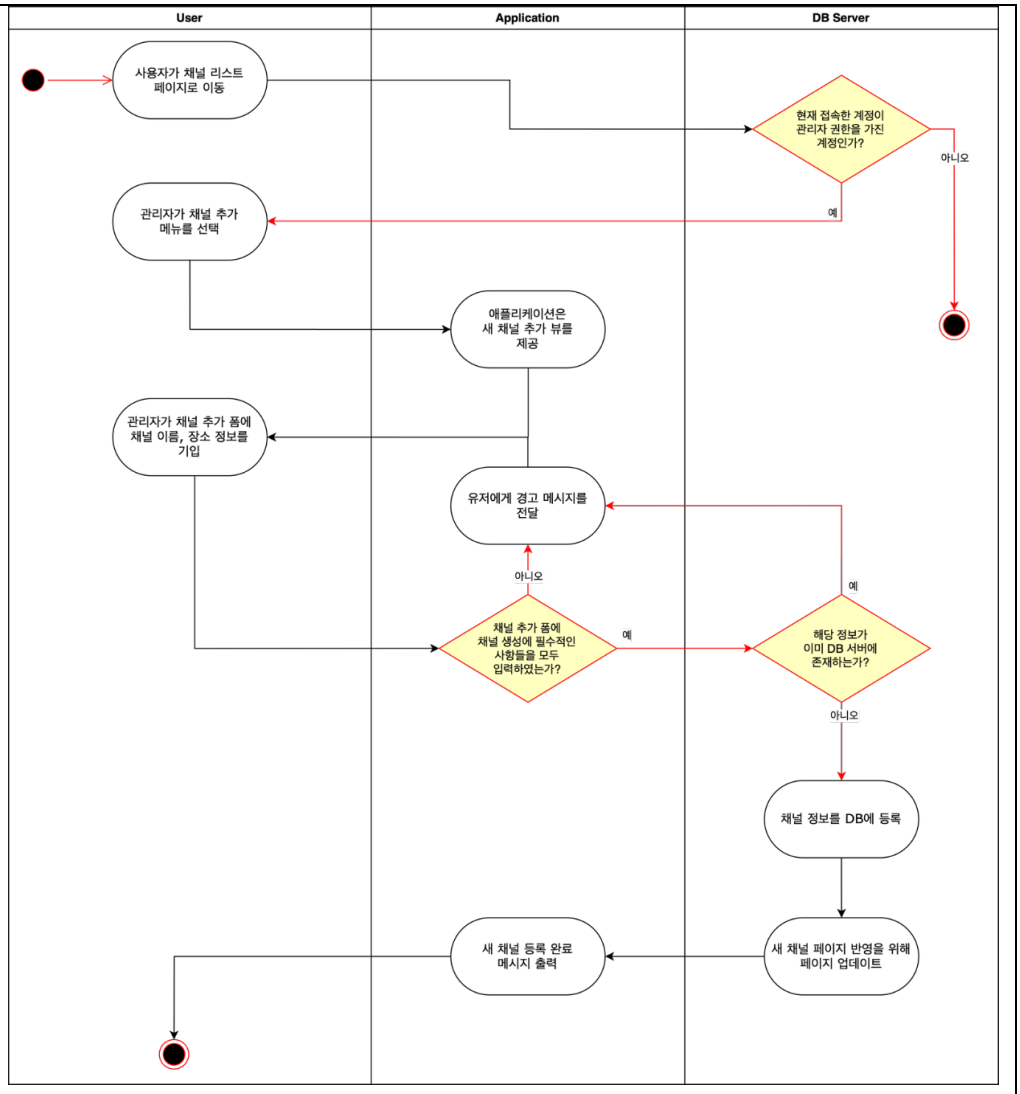
Activity Diagram	

Figure 2-4 Use Case 3 Activity Diagram

Table 2-41 Use Case 4

Use Case	4
Case Name	스케줄 예약 요청
Actors	게스트 유저(가입 유저), 모바일 애플리케이션, 백엔드 애플리케이션 서버, DB 서버
Description	게스트가 스케줄 예약을 요청한다.
Pre-conditions	게스트가 현재 로그인 상태여야 한다.
Post-conditions	게스트가 예약 요청한 스케줄이 관리자의 스케줄 승인 페이지로 이동한다
Primary Flow	1. 게스트가 동아리 채널의 타임 테이블 페이지로 이동한다. 2. 게스트가 타임 테이블 페이지에서 스케줄 예약 버튼을 누른다. 3. 애플리케이션은 스케줄 예약 페이지를 보여준다.

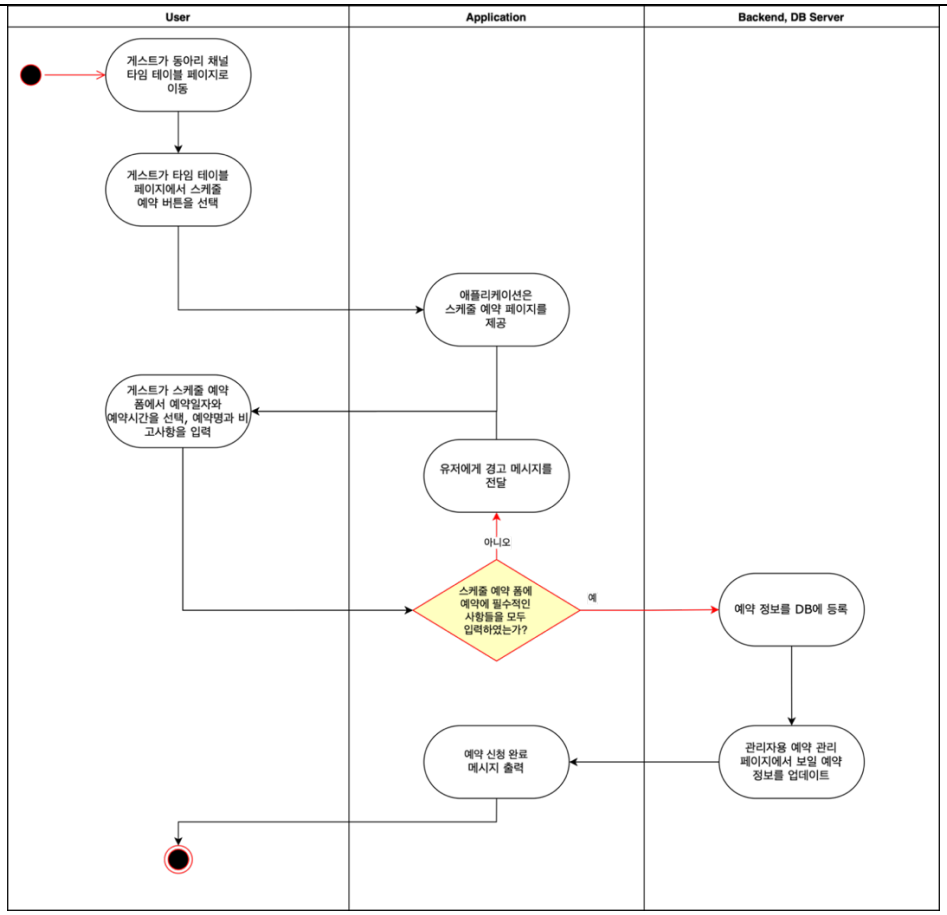
	4. 게스트는 스케줄 예약 폼에서 예약일자와 예약시간을 선택하고, 예약명과 비고사항을 입력한다. 5. 게스트가 확인 및 제출 버튼을 누른다. 5.1. 만약 비고사항을 제외한 예약일자, 예약시간이 선택되지 않았거나, 예약명이 입력되지 않았다면, 애플리케이션은 유저에게 경고 메시지를 전달한다. 5.2. 모든 필수 정보가 정상적으로 입력되었다면, 해당 예약 데이터가 DB 서버에 저장되고, 관리자용 예약 관리 페이지에 해당 정보가 업데이트된다. 5.3. 서버는 애플리케이션을 통해 새 채널이 정상적으로 추가되었음을 보여주는 메세지를 전달한다.
Activity Diagram	 **Figure 2-5 Use Case 4 Activity Diagram**

Table 2-42 Use Case 5

Use Case	5
Case Name	스케줄 상세정보 조회 및 삭제
Actors	유저(관리자, 게스트(가입 유저)), 모바일 애플리케이션, 백엔드 애플리케이션 서버, DB 서버
Description	유저가 예약 상세정보를 조회하고, 게스트가 현재 예약된 스케줄을 삭제한다.
Pre-	유저가 현재 로그인 상태이고, 현재 조회 가능한 예약이 존재해야 한다.

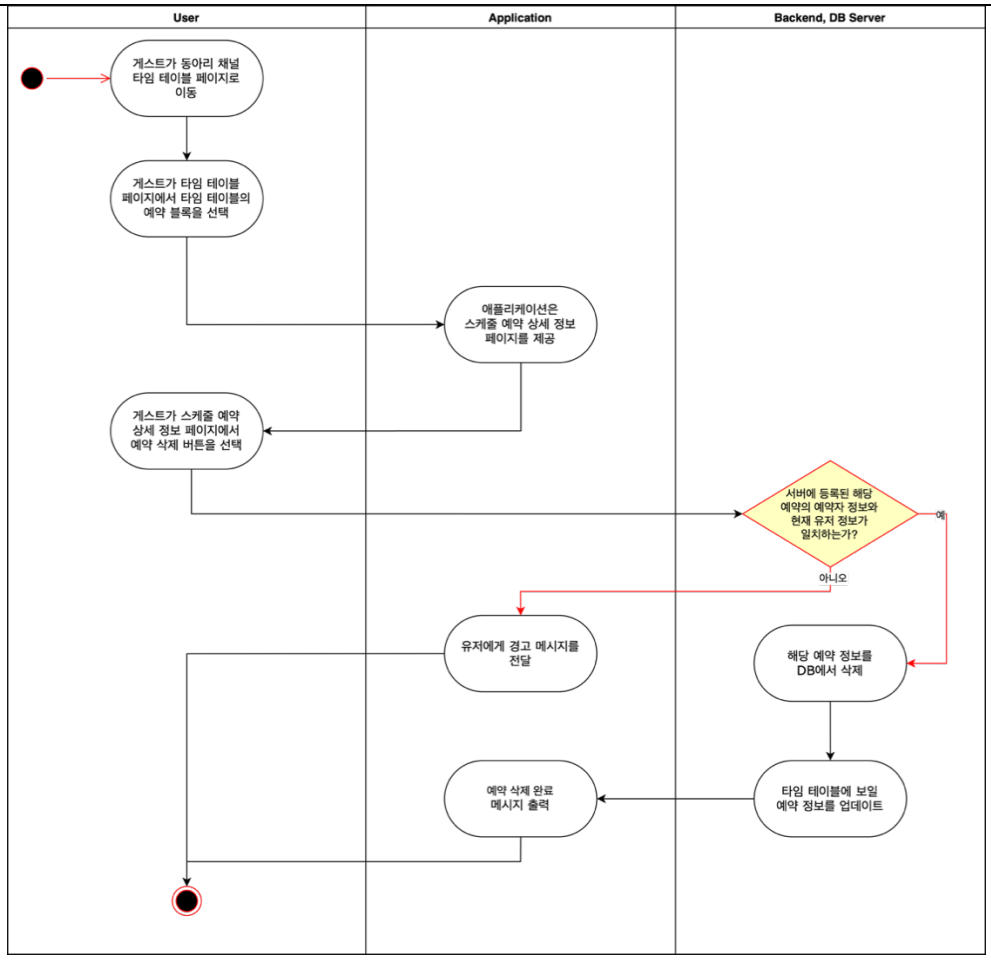
conditions	
Post-conditions	예약 삭제 신청에 성공하면 게스트의 예약이 삭제된다.
Primary Flow	1. 게스트가 동아리 채널의 타임 테이블 페이지로 이동한다. 2. 게스트가 현재 타임 테이블상의 예약 블록을 선택한다. 3. 애플리케이션은 스케줄 예약 상세 정보 페이지를 보여준다. 4. 게스트는 스케줄 예약 상세 정보 페이지에서 예약 삭제 버튼을 선택한다. 4.1. 만약 서버에 등록된 예약자 정보와 현재 유저의 정보가 일치하지 않으면, 애플리케이션은 유저에게 경고 메시지를 보낸다. 4.2. 만약 서버에 등록된 예약자 정보와 현재 유저의 정보가 일치하면, 해당 예약 데이터가 DB 서버에서 삭제된다. 4.2.1 서버는 애플리케이션을 통해 예약 정보가 삭제되었음을 보여주는 메세지를 전달하고, 타임 테이블을 업데이트한다.
Activity Diagram	

Figure 2-6 Use Case 5 Activity Diagram

Table 2-43 Use Case 6

Use Case	6

Case Name	관리자의 스케줄 승인/거절
Actors	유저(관리자), 모바일 애플리케이션, 백엔드 애플리케이션 서버, DB 서버
Description	관리자가 가입 유저가 요청한 스케줄을 승인하거나 거절할 수 있다
Pre-conditions	유저는 관리자 권한이 부여된 계정으로 로그인을 성공해야 하며, 현재 승인을 대기 중인 예약이 존재해야 한다.
Post-conditions	예약의 승인 여부에 따라 해당 예약 정보가 타임 테이블에 반영된다.
Primary Flow	1. 사용자가 로그인을 한다. 2. 접속한 계정의 권한을 확인한다. 2.1. 만약 현재 접속한 계정이 관리자 계정이 아닐 경우 2.1.1. 채널 예약 관리 페이지 버튼을 활성화하지 않는다. 2.2. 만약 현재 접속한 계정이 관리자 계정일 경우 2.2.1. 관리자는 채널 예약 관리 페이지에 접속하고, 애플리케이션은 채널 관리 페이지를 보여준다. 2.2.2. 관리자는 가입 유저의 예약 요청 정보들을 확인하고, 해당 예약을 승인하거나 거절한다. 2.2.3. 예약 요청의 승인 여부 2.2.3.1. 만약 관리자가 예약 요청을 승인하면, 백엔드 애플리케이션 서버에서 가입 유저에게 승인 알림을 전송한다. 2.2.3.1.1. DB 서버는 해당 스케줄 정보를 저장한다. 2.2.3.1.2. 백엔드 애플리케이션 서버에서 예약 정보를 업데이트한다. 2.2.3.2. 만약 관리자가 요청을 거절하면 모바일 애플리케이션 서버에서 가입 유저에게 거절 알림을 전송한다. 2.2.3.2.1 백엔드 애플리케이션 서버에서 예약 정보를 업데이트한다.

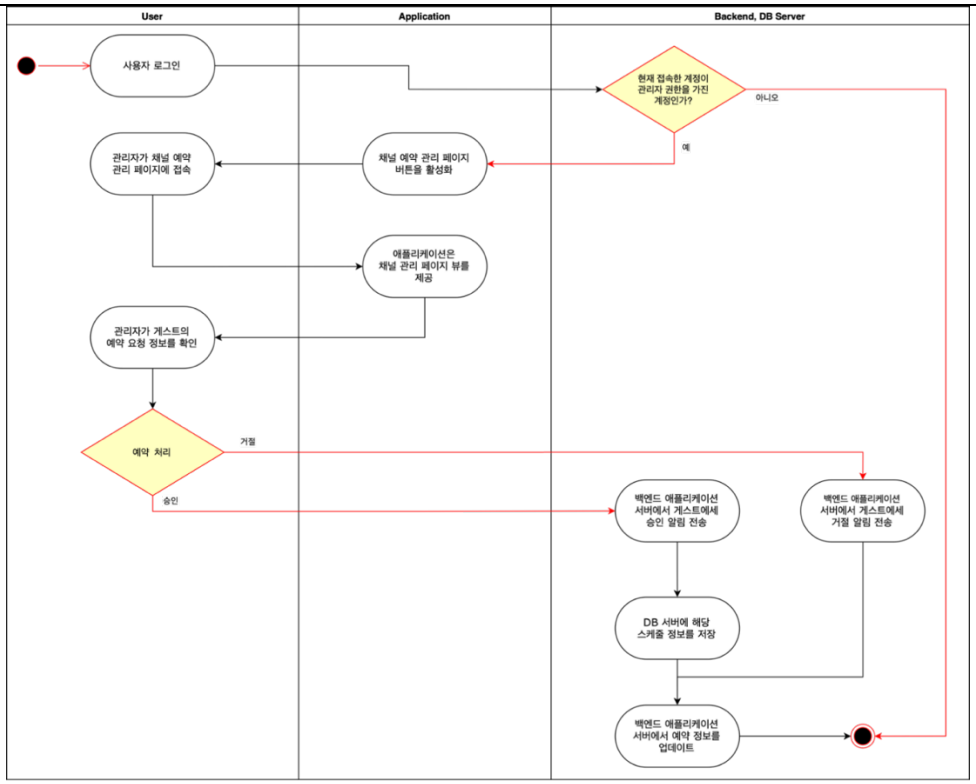
Activity Diagram	

Figure 2-7 Use Case 6 Activity Diagram

2.7. Domain Specific Requirements

Table 2-44 Domain Specific Requirement 1

No.	DM_RS_001	**Related Requirement**	
Title	예약 시스템은 각 대학별 학사규정 및 규약을 준수해야 한다.		
Description	예약 시스템은 각 대학 내 학칙을 준수하며, 예약 결과가 대학의 교내 행사 및 공간 사용 요청보다 우선시되거나, 예약 시 언제나 해당 시간대에 사용이 가능할 것을 보장하지 않는다.		

Table 2-45 Domain Specific Requirement 2

No.	DM_RS_002	**Related Requirement**	
Title	예약 시스템은 대학의 요청에 따라 서비스를 변경 또는 제공을 중단할 수 있어야 한다.		
Description	각 대학은 개별적으로 다양한 학사 시스템 등을 운영하고 있고, 본 예약 시스템이 기존의 예약 시스템과 충돌할 가능성을 배제할 수 없다. 따라서, 이러한 충돌이 발생하는 등의 이유로 대학의 요구가 있는 경우 해당 대학에 대한 서비스를 변경 또는 중단할 수 있다.		

Table 2-46 Domain Specific Requirement 3

No.	DM_RS_003	Related Requirement	USER_RQ_001 USER_RQ_002 SYS_RQ_FR_001 SYS_RQ_FR_002
Title	예약 시스템은 각 대학별로 재학생 인증 절차를 거쳐야 하며, 해당 유저가 재학 상태가 아닐 경우 서비스 제공을 제한할 수 있어야 한다.		
Description	재학 중이 아닌 유저(휴학, 제적, 졸업, 자퇴 등)가 해당 서비스로 로그인해 예약을 시도하는 경우, 해당 학교의 예약 시스템을 제공하는 데 혼선을 줄 수 있다. 따라서, 서비스에 대한 제공은 계정을 가지고 있는 현재 재학 중인 학생들을 대상으로 하여야 한다.		

2.8. Supplementary Requirements

2.8.1. Interface requirements

- 안드로이드 / iOS 플랫폼에 관계없이 일관된 UI를 제공하여야 한다.
- 디자인 가이드라인과 색상 스킴을 포함한 모든 기능이 모든 플랫폼에서 동일하게 작동해야 한다.
- UI는 직관적이고 사용하기 쉬워야 하며, 모든 기능에 대한 사용 방법이 명확하게 표시되어 있어야 한다.

2.8.2. Physical requirements

- 앱의 응답은 즉각적이어야 한다.
- 서버와의 통신은 느리지 않아야 한다.

2.8.3. Design requirements

- client-server design을 사용하여야 한다.
- 추후 확장성을 고려하여 유지보수가 용이하고 모듈화된 설계를 진행한다.
- 현재로서는 하나의 동아리에 대한 앱을 만들고 있지만, 추후에는 여러 개의 동아리를 한 앱에서 볼 수 있는 추가적인 구현을 할 수 있음을 고려하여 설계한다.

2.8.4. Implementation requirements

- Dart Flutter
- Java, Spring Boot, Spring Data JPA
- RESTful API
- DateTime ISO8601 준수
- RDBMS(MySQL, H2)

2.8.5. Additional NFRs

- 검증된 외부 패키지들을 사용하여야 하며, 보안 관련 최신 업데이트와 패치를 받을 수 있도록 한다.
- 해당 시스템에 외부 시스템이 접근하지 못하도록 하여 시스템 안정성을 보장한다.
- 사용자 데이터는 관련 법규를 준수하여 처리되어야 하며, 개인정보의 기밀성과 무결성을 보장

한다.

3. Design Efforts

3.1. Architectural Design

3.1.1. Frontend Layers

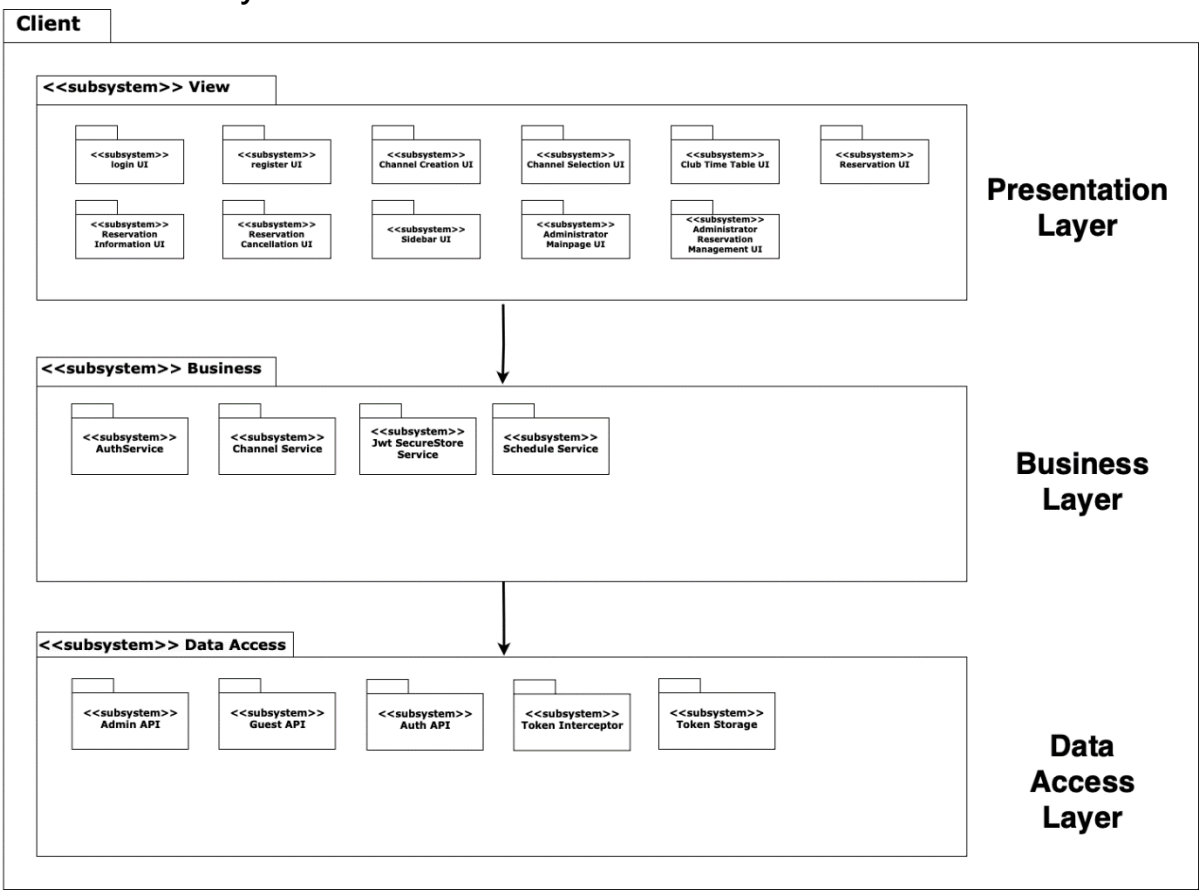


Figure 3-1 Client System Architecture Diagram

- **Presentation Layer**
  - 외부에 화면을 보여주는 View 영역
  - 외부에서 사용자에게 입력을 받는 영역
  - 실제 앱에서 보여지는 대부분의 영역으로 구성되는 계층이다.

- **Business Layer**
  - 화면에 나타낼 State를 저장해두는 영역
  - 사용자의 입력에 따라 데이터를 불러와 State를 업데이트 하는 영역
  - 앱을 표시할 때 필요한 데이터와 비즈니스 로직을 담아두는 계층이다.

- **Data Access Layer**
  - 서버에서 직접 요청을 받아오는 영역

- 클라이언트 디바이스에 저장된 데이터를 불러오는 영역
- 즉, 앱을 구동하기 위해 필요한 데이터를 받아오는 레이어이다.

3.1.2. Backend Layers

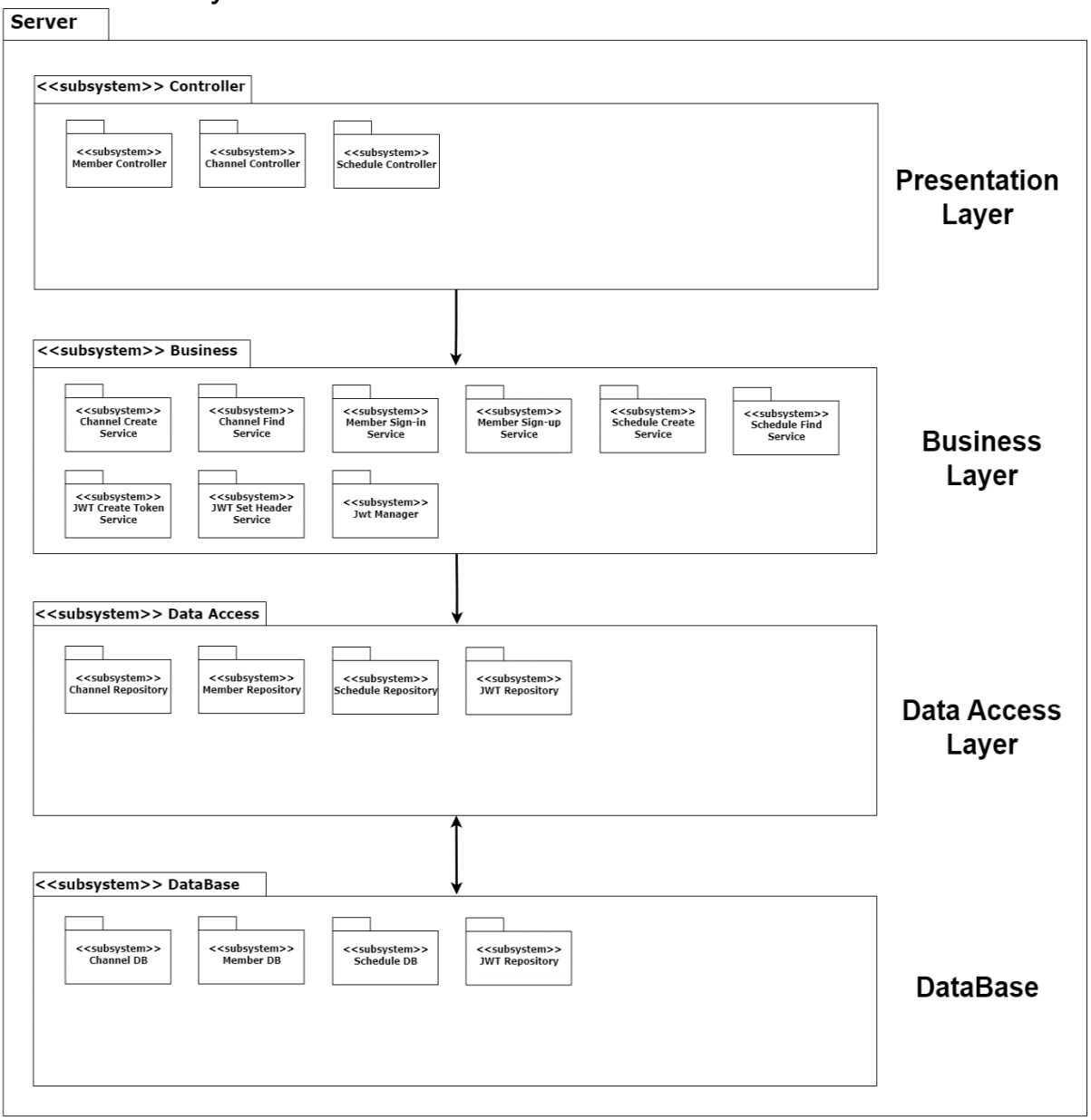


Figure 3-2 Server System Architecture Diagram

- **Presentation Layer**
  - 외부에 요청, 응답을 담당하는 계층
  - 외부 변화에 민감한 외부 의존성이 높은 영역
  - 외부 영역에 대한 처리를 담당하는 코드나 요청, 응답을 담당하는 클래스가 모두 여기에 속한다.

- **Business Layer**

- 비즈니스 로직을 담당하는 클래스
- 전체적인 비즈니스 로직 흐름을 읽을 수 있도록 작성한다.
- 상세한 구현은 각각의 도메인에 위임한다.

- **Data Access Layer**
  - Business Layer가 DB에 접근할 수 있도록 인터페이스를 제공한다.
  - 이때 레이어는 위에서 아래로 순방향으로만 참조 되어야 한다. 참조 방향이 역류되지 않게 한다.
  - 레이어의 참조가 하위 레이어를 건너뛰지 않는다. 동일 레이어 간에는 서로 참조하지 않아야 한다.

3.2. Detailed Design Class Diagram

3.2.1. Frontend Design Class Diagram

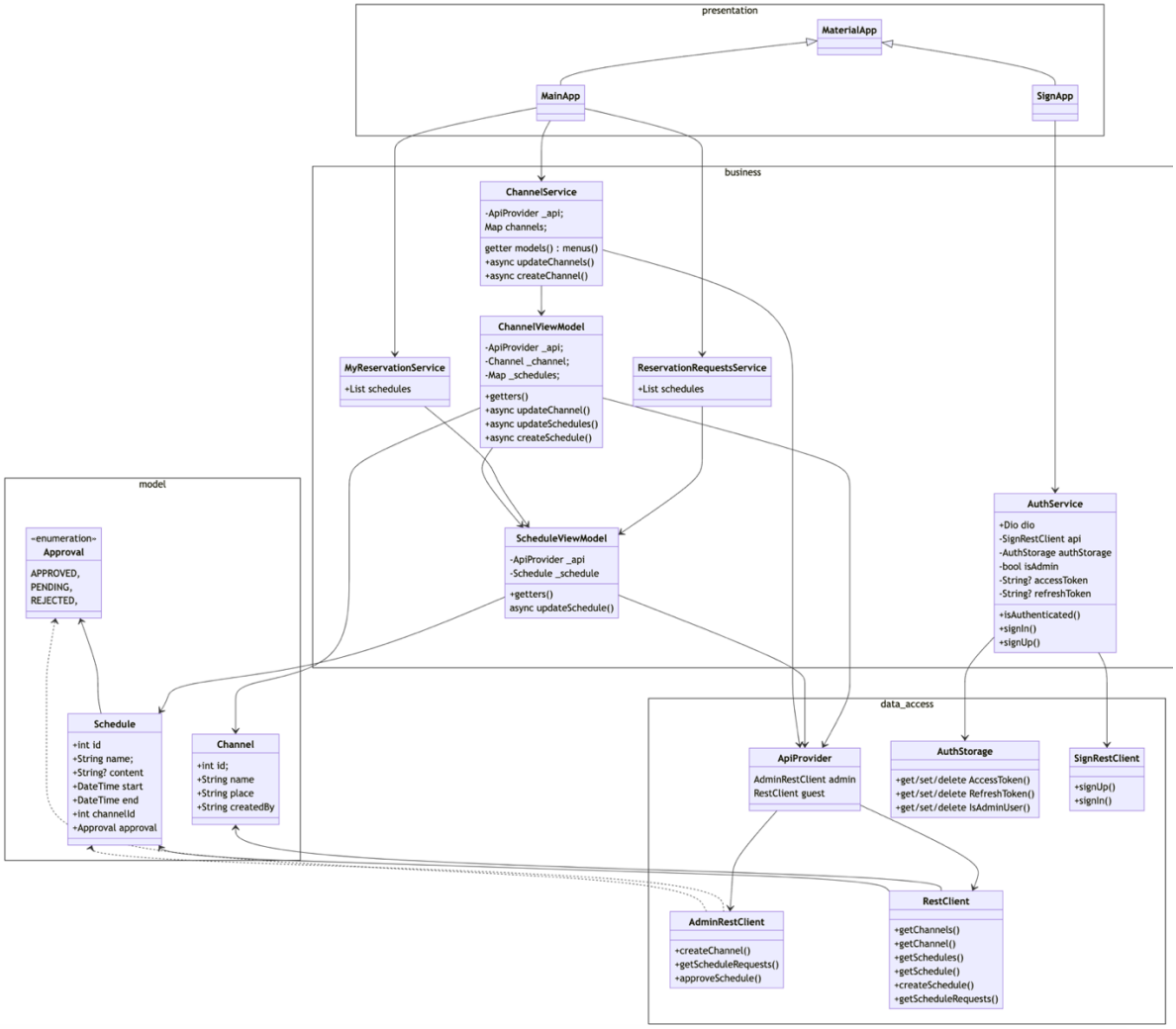


Figure 3-3 Frontend Design Class Diagram

해당 다이어그램은 프론트엔드 애플리케이션의 디자인 클래스 다이어그램으로, 크게 앞서 2.1의 Layered Architecture에서 언급한 Data Access Layer, Business Layer, Presentation Layer로 구성되어

있다.

- Data Access Layer에서는 Retrofit 라이브러리로 제작된 API RestClient에 AuthService를 통해 사용자의 Token이 인증된 Dio 인스턴스를 Injection하여 ApiProvider 인터페이스를 제공한다.
- Business Layer에서는 ApiProvider 인터페이스를 통해서 백엔드 데이터에 액세스하고, 결과를 저장하는 역할을 수행한다.
- Presentation Layer에서는 최종 사용자에게 보일 정보를 적합한 UI를 통해 제공하며, 보여질 정보를 Business Layer와 Binding하여 state가 변경되는대로 화면에 보여지게 한다.
- 추가적으로, Model은 서버 Domain과 일치하도록 하였다.

3.2.2. Backend Design Class Diagram

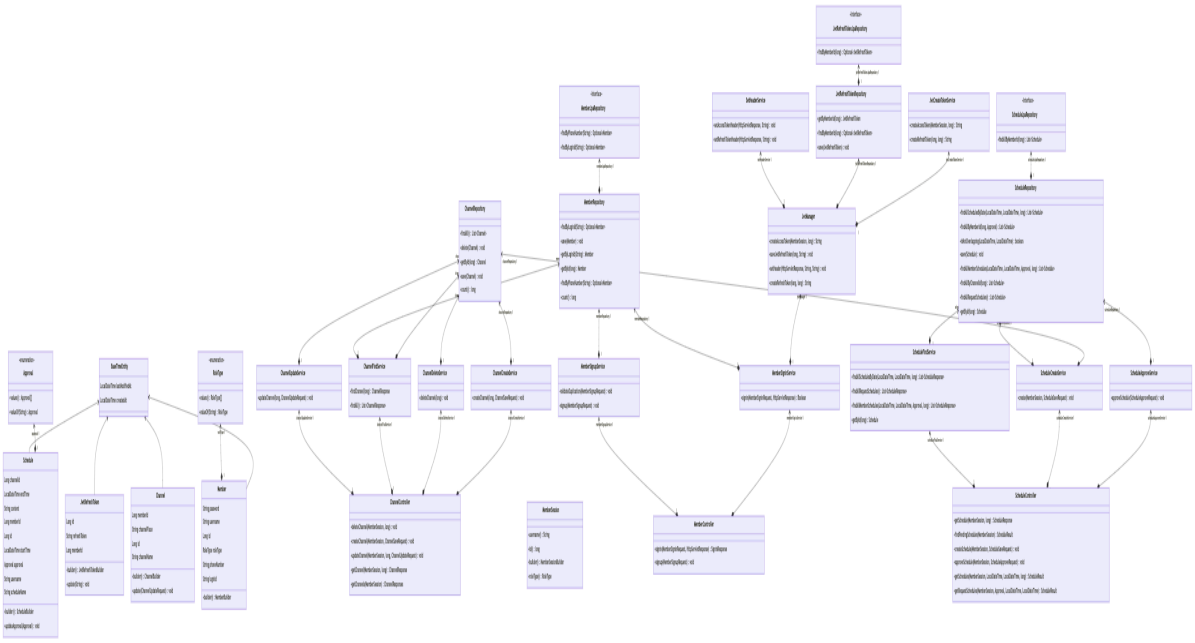


Figure 3-4 Backend Design Class Diagram

3.2.3. Backend Domain Diagram

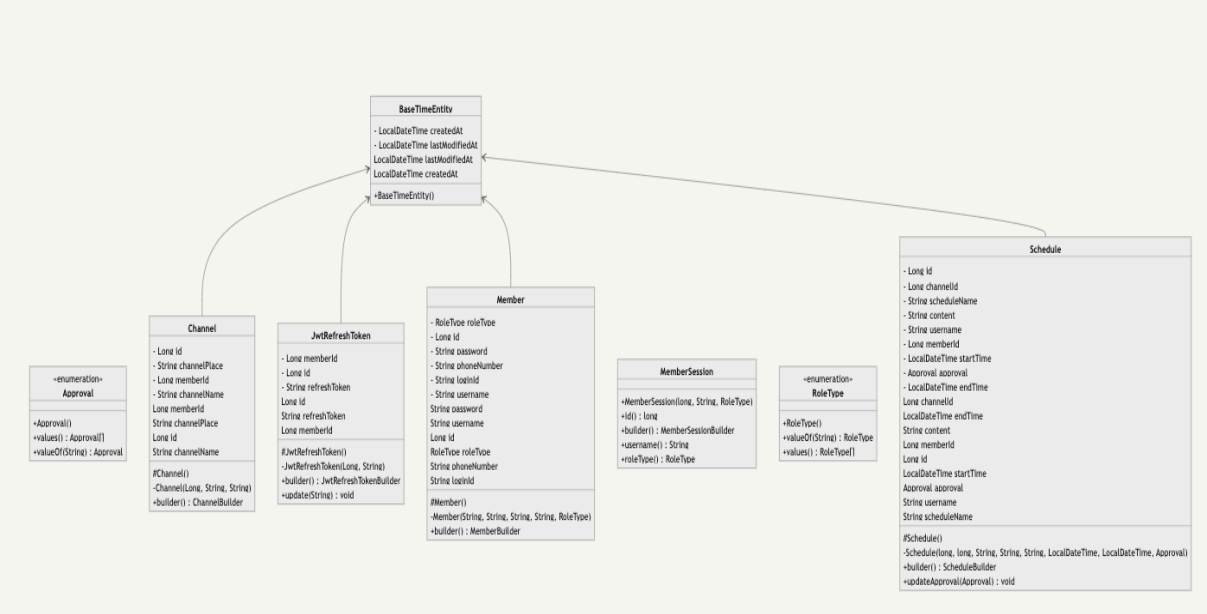


Figure 3-5 Backend Domain Diagram

이 클래스 다이어그램은 도메인 계층의 다이어그램을 나타낸다. 각 클래스에는 해당 클래스의 구조와 기능을 나타내는 메소드와 변수가 포함되어 있다.

- Approval, RoleType: 이들은 열거형 클래스로, 특정 변수에 허용되는 값의 집합을 정의한다.
- BaseTimeEntity: 이 클래스는 시간 관련 변수를 가지고 있으며, 다른 클래스들 (Channel, JwtRefreshToken, Member, Schedule)은 이 클래스를 상속받아 시간 정보를 활용한다.
- Channel: 이 클래스는 채널에 대한 정보를 나타낸다. 'id', 'channelPlace', 'memberId', 'channelName' 등의 변수를 가지고 있다.
- JwtRefreshToken: 이 클래스는 JWT(refresh token)에 대한 정보를 나타낸다. 'id', 'refreshToken', 'memberId'와 같은 변수를 가지고 있다.
- Member: 이 클래스는 회원에 대한 정보를 나타낸다. 'id', 'password', 'phoneNumber', 'loginId', 'username', 'roleType' 등의 변수를 가지고 있다.
- MemberSession: 이 클래스는 회원 세션에 대한 정보를 나타낸다. 'id', 'username', 'roleType' 등의 변수를 가지고 있다.
- Schedule: 이 클래스는 일정에 대한 정보를 나타낸다. 'id', 'channelId', 'scheduleName', 'content', 'username', 'memberId', 'startTime', 'endTime', 'approval' 등의 변수를 가지고 있다.

3.2.4. Backend Presentation/Business/Data Access Layer Diagram (Member, Channel,

Schedule)

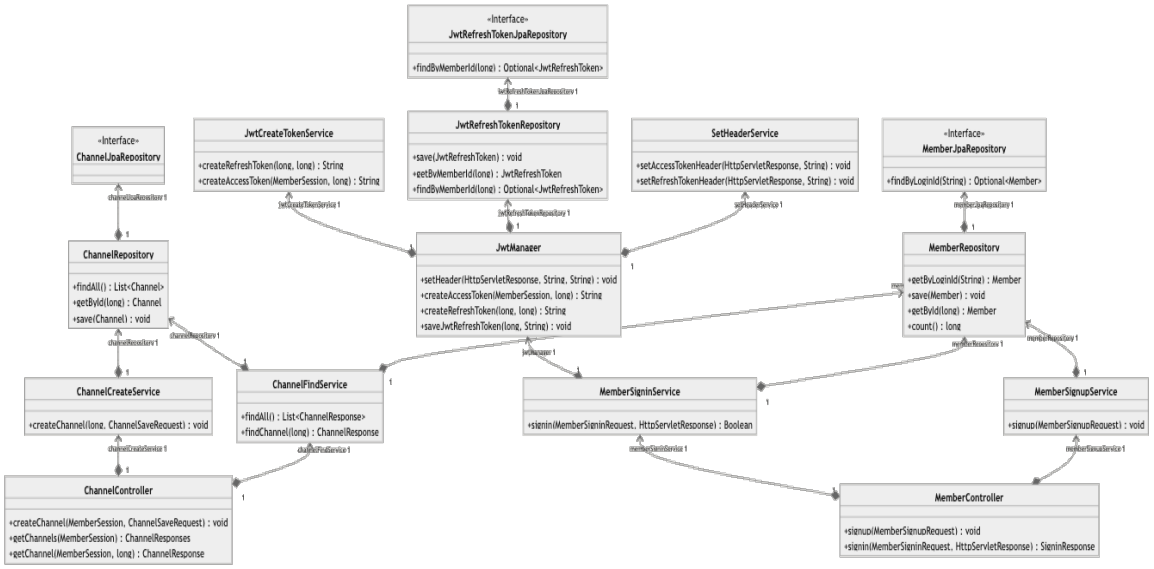


Figure 3-6 Backend Presentation/Business/Data Access Layer Diagram (Member, Channel)

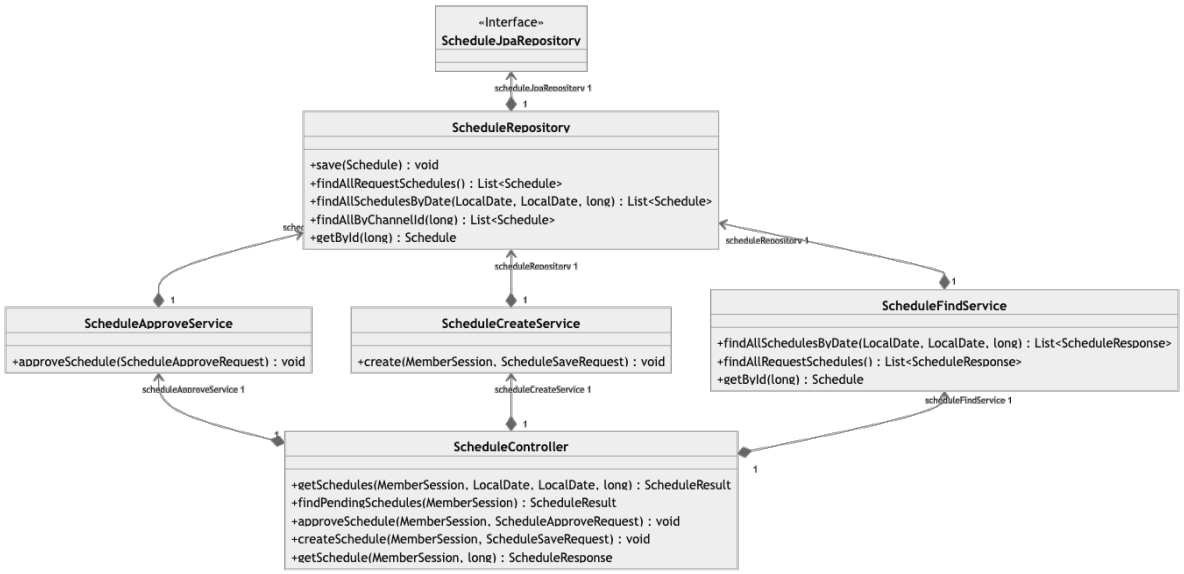


Figure 3-7 Backend Presentation/Business/Data Access Layer Diagram

이 클래스 다이어그램은 웹 애플리케이션의 RESTful API를 처리하는 여러 서비스와 컨트롤러, 그리고 이들이 상호작용하는 방식을 보여준다.

- Controller 클래스 (MemberController, ChannelController, ScheduleController) Presentation Layer에 해당하는 클래스들로 사용자의 요청을 받아 처리하고, 적절한 서비스를 호출하여 비즈니스 로직을 실행한다. 이후 결과를 사용자에게 응답한다.
- Service 클래스 (ChannelCreateService, ChannelFindService, JwtManager,

MemberSigninService, MemberSignupService 등) Business Layer에 해당하는 클래스들로 각각의 서비스 클래스는 특정 비즈니스 로직을 수행한다. 이 과정에서 이 클래스들은 필요한 데이터를 Repository로부터 가져올 수 있다.

- Repository 클래스 (MemberRepository, ChannelRepository, ScheduleRepository, JwtRefreshTokenRepository 등) 이들은 데이터베이스와의 직접적인 상호작용을 담당하며, 데이터를 저장하거나 검색하는 역할을 한다. 또한 JpaRepository는 Spring Data JPA의 인터페이스로 각 도메인의 CRUD 작업을 추상화한다. Repository 클래스들은 JpaRepository를 사용하여 데이터베이스에 접근한다.

3.3. Use Case Realization (Sequence Diagram)

3.3.1. Sign-up

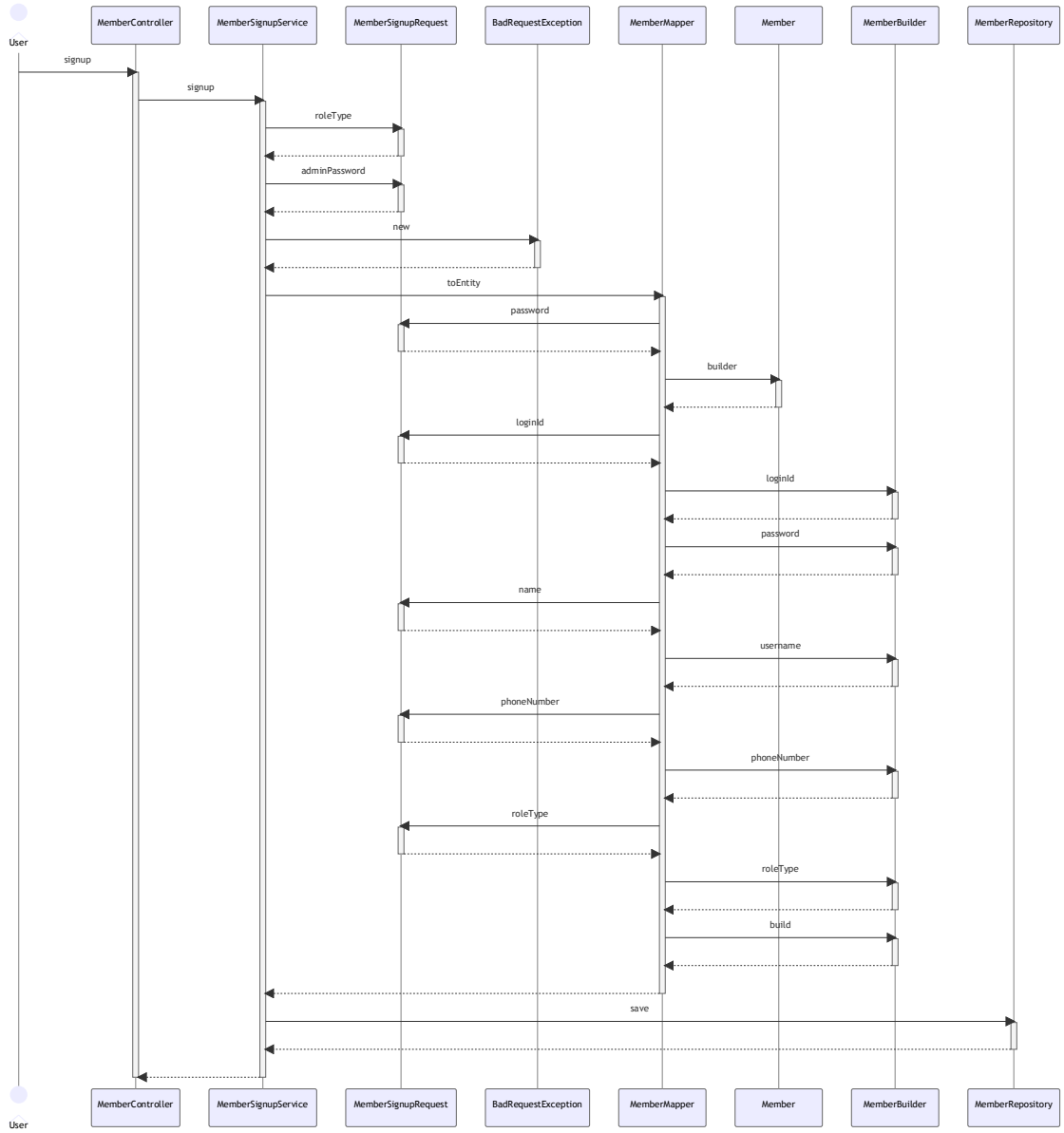


Figure 3-8 Sign-up sequence diagram

이 시퀀스 다이어그램은 유저가 서비스에 회원가입 시의 일련의 흐름을 나타낸다.

- MemberController: 이 클래스에서는 회원 가입 요청을 받고, MemberSignupService에 처리를 위임한다.
- MemberSignupService: 이 클래스에서는 signup() 메소드를 통해 회원 가입을 처리하는 역할을 한다.
  - signup(): 해당 메소드는 회원 가입 요청을 받아 처리한다.
- MemberSignupRequest: 이 클래스에서는 회원 가입 요청 정보를 담고 있다.
- BadRequestException: 이 클래스는 예외 클래스로, 잘못된 요청이 있을 경우 에러를 발생시키는 역할을 한다.
- MemberMapper: 이 클래스에서는 toEntity() 메소드를 통해 회원 가입 요청 정보를 회원 엔티티로 변환하는 역할을 수행한다.
  - toEntity(): 이 메소드에서는 회원 가입 요청 정보를 회원 엔티티로 변환한다.
- Member: 이 클래스는 회원 정보를 담고 있는 엔티티 클래스로, 회원 객체로 인스턴스화된다.
- MemberBuilder: 이 클래스는 회원 객체를 생성하고 필드 값을 설정하는 빌더 클래스로 기능한다.
- MemberRepository: 이 클래스는 save() 메소드를 통해 회원 정보를 저장하고 조회하는 역할을 수행하는 클래스이다.
  - save(): MemberRepository 클래스 내에서 회원 정보를 저장하는 역할을 수행하는 메소드이다.

각 클래스와 메소드는 회원 가입 요청을 처리하고, 필요한 정보를 변환하며, 회원 객체를 생성하여 저장한다. 이를 통해 사용자의 회원 가입 요청에 대한 처리를 수행한다.

3.3.2. Sign-in

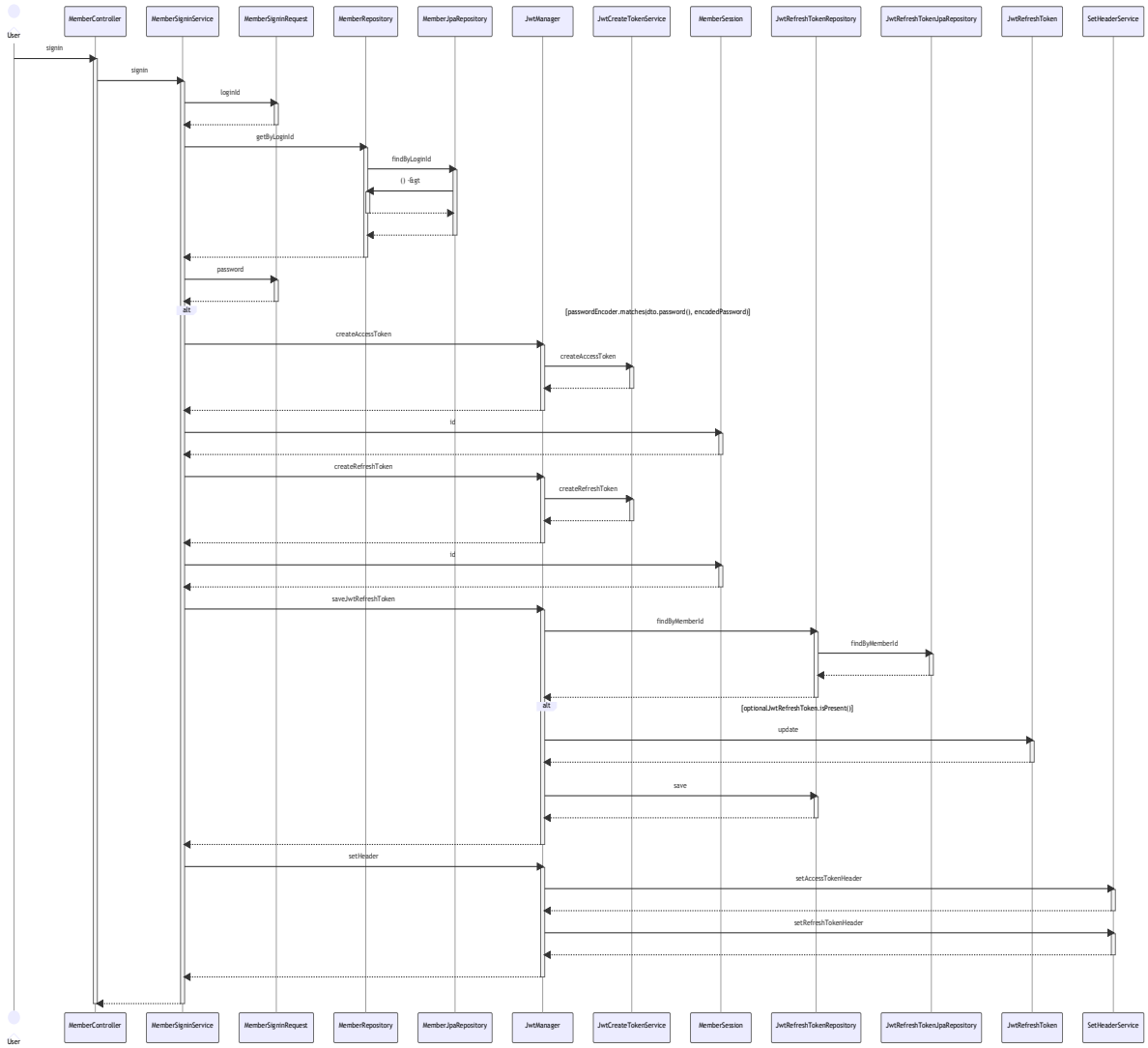


Figure 3-9 Sign-in sequence diagram

이 시퀀스 다이어그램은 사용자가 서비스에 로그인하는 일련의 과정을 나타낸다.

- MemberController: 이 클래스에서는 signin() 메소드로 로그인 요청을 받고, MemberSigninService에 그 처리를 위임한다.
- MemberSigninService: 이 클래스는 signin() 메소드를 통해 회원 로그인 요청을 처리하고, 회원 정보를 조회하고 인증을 수행하는 역할을 한다.
  - signin(): 해당 메소드는 회원 로그인 요청을 처리하는 역할을 한다.
- MemberRepository: 이 클래스는 getByLoginID() 메소드를 통해 회원 정보를 저장하고 조회하는 역할을 한다.
  - getByLoginId(): 이 메소드에서는 로그인 ID에 해당하는 회원 정보를 조회한다.
- NotFoundException: 이 클래스는 예외 클래스로, 조회된 회원 정보가 없을 경우 에러를 발생시키는 역할을 수행한다.
- Member: 이 클래스는 회원 정보를 담고 있는 엔티티 클래스로, 회원 객체로 인스턴스화된다.
  - getPassword(): 이 메소드는 회원의 비밀번호를 가져오는 역할을 수행한다.

- getRoleType(): 이 메소드는 회원의 역할 유형을 가져온다.

- MemberSigninRequest: 이 클래스는 회원 로그인 요청에서 필요한 정보를 포함한다.
- MemberMapper: 이 클래스에서는 toMemberSession 메소드를 통해 회원 정보를 MemberSession으로 변환하는 역할을 수행한다.
  - toMemberSession(): 이 메소드는 MemberMapper 클래스 내에서 회원 정보를 MemberSession으로 변환하는 역할을 수행한다.
- MemberSession: d이 클래스는 로그인 된 회원의 세션 정보를 담고 있다.
- JwtManager: 이 클래스는 JWT 토큰을 생성하고 처리하는 역할을 수행한다.
  - createAccessToken(): 액세스 토큰을 생성하는 메소드이다.
  - createRefreshToken(): 리프레시 토큰을 생성하는 메소드이다.
  - saveJwtRefreshToken(): 리프레시 토큰을 저장하는 메소드이다.
- JwtRefreshTokenRepository: 이 클래스는 리프레시 토큰을 저장하고 조회하는 역할을 수행한다.
- JwtCreateTokenService: 이 클래스에서는 JWT 토큰 생성에 관련된 기능을 수행한다.
- BadRequestException: 이 클래스는 잘못된 요청 예외를 처리하는 역할을 수행하는 클래스이다.
- SigninResponse: 이 클래스는 로그인 응답을 담고 있는 클래스다.

이와 같이 각 클래스와 메소드는 로그인 과정에서 필요한 역할을 수행하며 사용자의 로그인 요청을 처리하고, 액세스 토큰과 리프레시 토큰을 생성하여 반환한다.

3.3.3. Channel creation

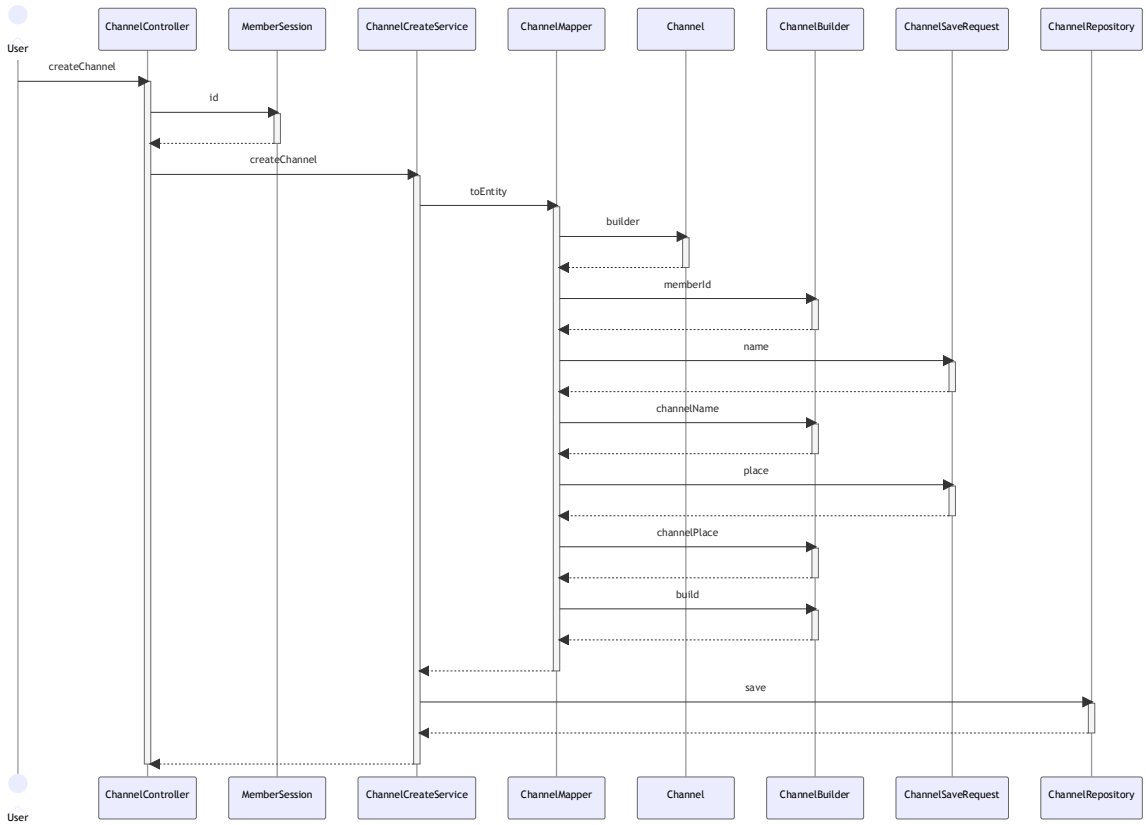


Figure 3-10 Channel creation sequence diagram

이 시퀀스 다이어그램은 관리자가 채널을 생성하는 일련의 과정을 나타낸다.

- ChannelController: 이 클래스에서는 채널 생성 요청을 받고, MemberSession에서 회원 ID를 가져와 처리한다.
- MemberSession: 이 클래스는 회원 세션 정보를 담고 있다.
- ChannelCreateService: 이 클래스에서는 createChannel() 메소드를 통해 채널 생성 요청을 처리하고 ChannelMapper를 사용하여 채널 엔티티를 생성한다.
  - createChannel(): 이 메소드는 채널 생성 요청을 처리한다.
- ChannelMapper: 이 클래스를 통해 채널 정보를 채널 엔티티로 변환하는 역할을 수행한다.
  - toEntity(): 이 메소드는 채널 정보를 채널 엔티티로 변환하는 역할을 수행한다.
- Channel: 이 클래스는 채널 정보를 담고 있는 엔티티이다.
- ChannelBuilder: 이 클래스는 채널 엔티티를 생성하는 빌더 클래스이다.
- ChannelSaveRequest: 이 클래스는 채널 생성 요청에서 필요한 정보를 담고 있다.
- ChannelRepository: 이 클래스를 통해 채널 정보를 저장하고 조회하는 역할을 수행하는 클래스이다.
  - save(): 이 메소드에서는 채널 정보를 저장한다.

이와 같이 각 클래스와 메소드는 채널 생성 과정에서 필요한 역할을 담당하며, 사용자의 요청을 처리하여 채널 엔티티를 생성하고 저장한다.

3.3.4. Single channel find service

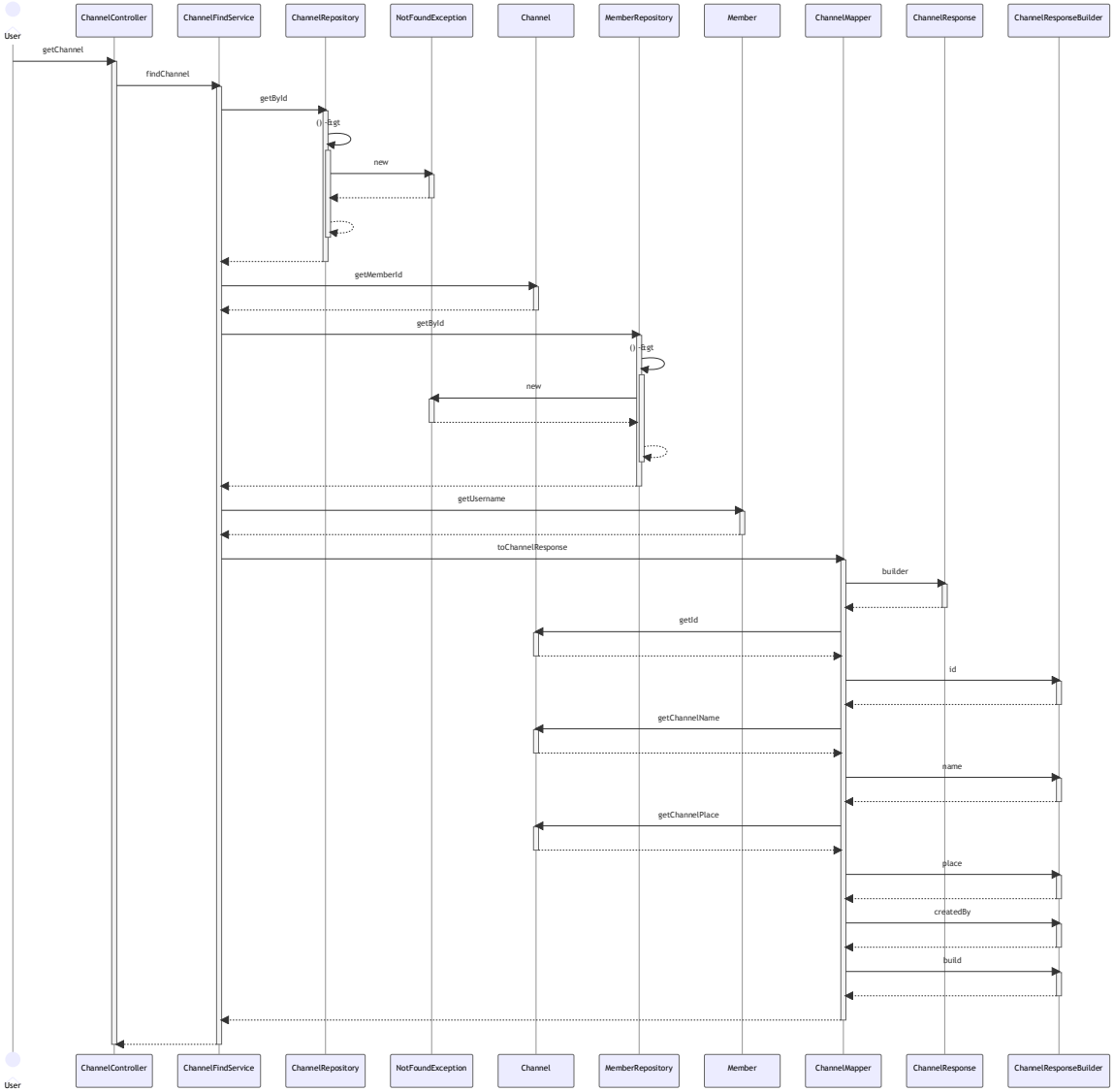


Figure 3-11 Single channel find service sequence diagram

이 시퀀스 다이어그램은 사용자가 채널 정보를 조회하는 일련의 과정을 나타낸다.

- ChannelController: 이 클래스는 채널 정보 조회 요청을 받고 ChannelFindService에 처리를 위임하는 역할을 한다.
- ChannelFindService: 이 클래스에서는 채널 정보 조회를 처리하고 ChannelRepository와 MemberRepository를 사용하여 채널과 회원 정보를 조회한다.
  - findChannel(): 이 메소드는 채널 정보 조회 요청을 처리하는 역할을 수행한다.
- ChannelRepository: 이 클래스는 getById() 메소드를 통해 채널 정보를 저장하고 조회하는 역할을 수행한다.
  - getById(): 이 메소드는 ChannelRepository 클래스 내에서 채널 ID에 해당하는 채널 정보를 조회한다.
- NotFoundException: 이 클래스는 예외 클래스로, 조회된 채널이나 회원 정보가 없을 경우 에러를 발생시킨다.

- MemberRepository: 이 클래스에서는 getById() 메소드를 통해 회원 정보를 저장하고 조회하는 역할을 수행한다.
  - getById(): 이 메소드는 MemberRepository 클래스 내에서 회원 ID에 해당하는 회원 정보를 조회한다.
- Member: 이 클래스는 회원 정보를 담고 있는 엔티티 클래스로, 회원 객체로 인스턴스화된다.
  - getUsername(): 이 메소드는 회원의 사용자명을 가져오는 역할을 수행한다.
- ChannelMapper: 이 클래스는 채널 정보를 채널 응답(ChannelResponse)로 변환하는 역할을 수행한다.
  - toChannelResponse(): 이 메소드는 ChannelMapper 클래스 내에서 채널 정보를 채널 응답으로 변환하는 역할을 수행한다.
- ChannelResponse: 이 클래스는 채널 정보를 담고 있는 응답 클래스이다.
- ChannelResponseBuilder: 이 클래스는 채널 응답을 구성하는 빌더 클래스이다.

이와 같이 각 클래스와 메소드는 채널 정보 조회 과정에서 필요한 역할을 담당하며, 사용자의 요청에 따라 채널과 회원 정보를 조회하여 채널 응답을 생성하여 반환한다.

3.3.5. Total channel find service

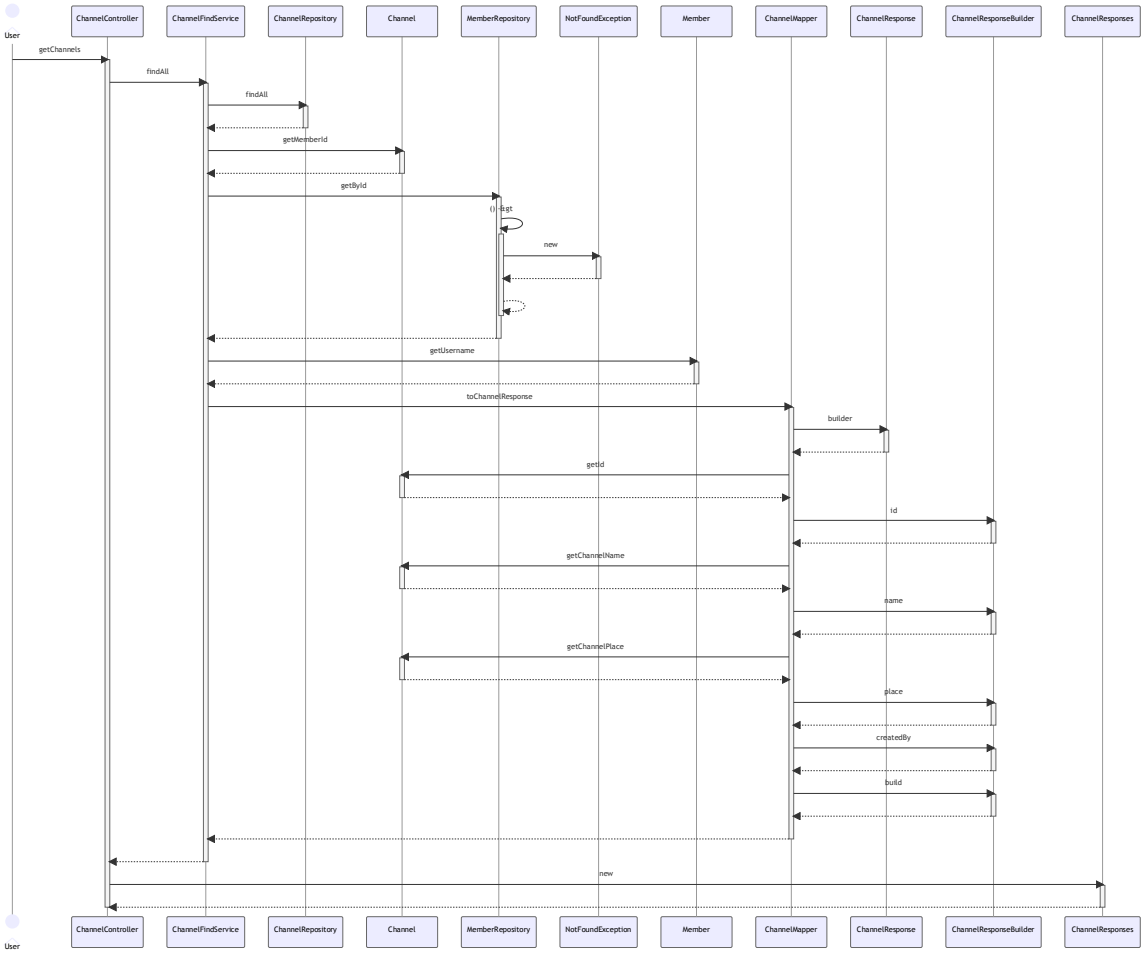


Figure 3-12 Total channel find service sequence diagram

이 시퀀스 다이어그램은 사용자가 모든 채널 정보를 조회한다.

1 ChannelController: 이 클래스는 채널 정보 조회 요청을 받고 ChannelFindService에 처리를 위임하는 역할을 한다.
2 ChannelFindService: 이 클래스에서는 findAll() 메소드를 통해 모든 채널 정보를 조회하고, MemberRepository를 사용하여 회원 정보를 조회한다.
    2.1 findAll(): 이 메소드는 모든 채널 정보를 조회하는 기능을 수행한다.
3 ChannelRepository: 이 클래스에서는 findAll() 메소드를 통해 채널 정보를 저장하고 조회하는 역할을 수행한다.
    3.1 findAll(): 이 메소드는 모든 채널 정보를 조회하는 기능을 수행한다.
4 NotFoundException: 이 클래스는 예외 클래스로, 조회된 회원 정보가 없을 경우 에러를 발생시킨다.
5 MemberRepository: 이 클래스는 getById() 메소드를 통해 회원 정보를 저장하고 조회하는 역할을 수행한다.
    5.1 getById(): 이 메소드는 회원 ID에 해당하는 회원 정보를 조회하는 기능을 수행한다.
6 Member: 이 클래스는 회원 정보를 담고 있는 엔티티 클래스이다.
    6.1 getUsername(): 이 메소드는 회원의 사용자명을 가져오는 역할을 수행한다.
7 ChannelMapper: 이 클래스에서는 toChannelResponse() 메소드를 통해 채널 정보를 채널 응답(ChannelResponse)로 변환하는 역할을 수행한다.
    7.1 toChannelResponse(): 이 메소드는 채널 정보를 채널 응답으로 변환하는 역할을 수행한다.
8 ChannelResponse: 이 클래스는 채널 정보를 담고 있는 응답 클래스이다.
9 ChannelResponseBuilder: 이 클래스는 채널 응답을 구성하는 빌더 클래스이다.
10 ChannelResponses: 이 클래스는 채널 응답들을 담고 있다.

이와 같이 각 클래스와 메소드는 모든 채널 정보 조회 과정에서 필요한 역할을 담당하며, 채널과 회원 정보를 조회하여 채널 응답들을 생성하여 반환한다.

3.3.6. Schedule creation

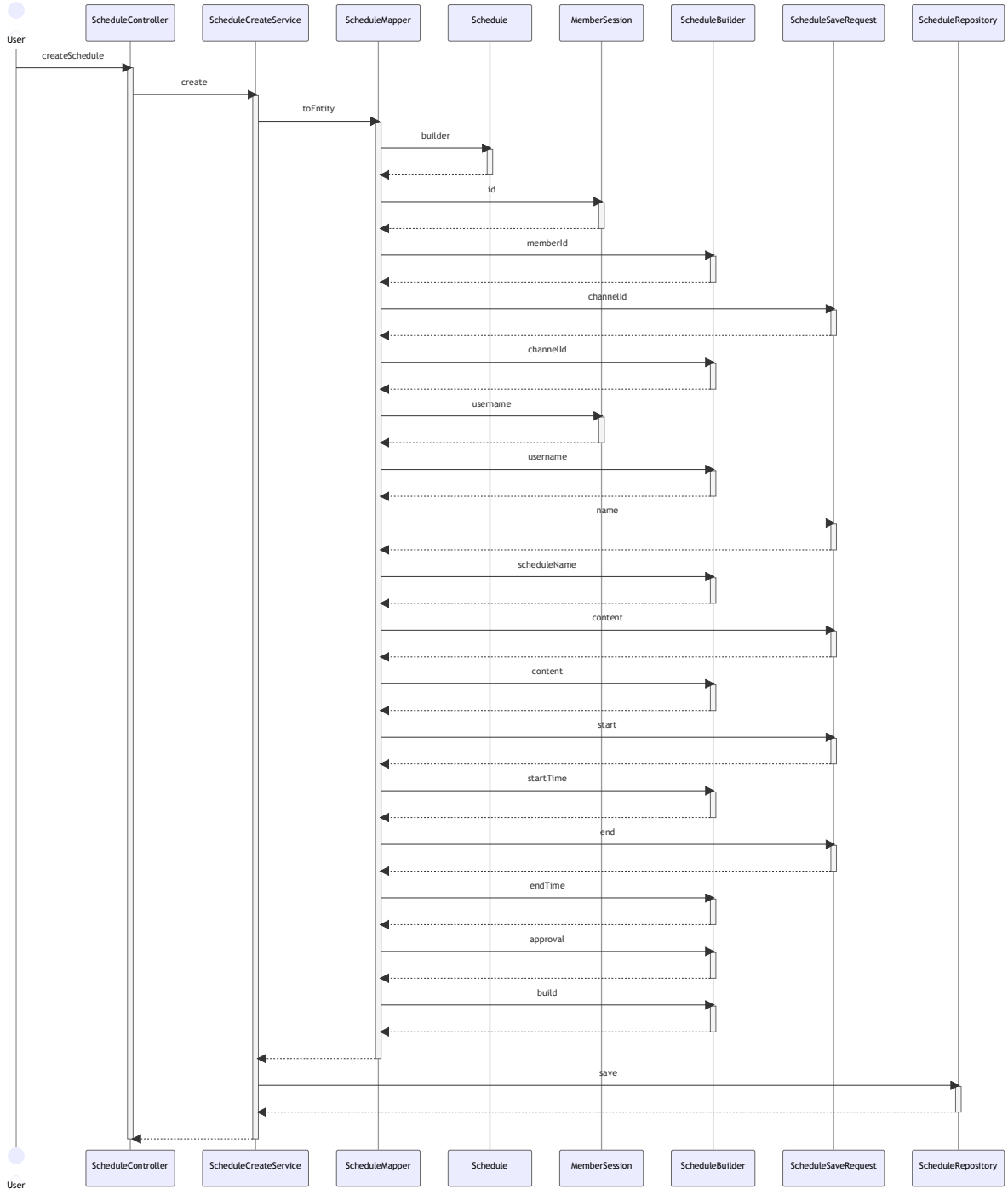


Figure 3-13 Schedule creation sequence diagram

이 시퀀스 다이어그램은 사용자가 스케줄을 생성하는 일련의 과정을 나타낸다..

- ScheduleController: 이 클래스는 일정 생성 요청을 받고, ScheduleCreateService에 처리를 위임하는 역할을 한다.
- ScheduleCreateService: 이 클래스에서는 create() 메소드를 통해 일정 생성 요청을 처리하고, ScheduleMapper를 사용하여 일정 엔티티를 생성한다.
  - create(): 이 메소드는 ScheduleCreateService 클래스에서 일정 생성 요청을 처리하는 역

할을 한다.

- ScheduleMapper: 이 클래스에서는 toEntity() 메소드를 통해 일정 정보를 일정 엔티티로 변환하는 역할을 수행한다.
  - toEntity(): 이 메소드는 ScheduleMapper 클래스 내에서 일정 정보를 일정 엔티티로 변환하는 역할을 수행한다.
- Schedule: 이 클래스는 일정 정보를 담고 있는 엔티티 클래스이다.
- MemberSession: 이 클래스는 회원 세션 정보를 담고 있다.
- ScheduleBuilder: 이 클래스는 일정 엔티티를 생성하는 빌더 클래스이다.
- ScheduleSaveRequest: 이 클래스는 일정 생성 요청 시 필요한 정보를 포함하고 있다.
- ScheduleRepository: 이 클래스는 save() 메소드를 통해 일정 정보를 저장하고 조회하는 역할을 수행한다.
  - save(): 이 메소드는 일정 정보를 저장하는 기능을 수행한다.

이와 같이 각 클래스와 메소드는 일정 생성 과정에서 필요한 역할을 담당하여 사용자의 요청을 처리하고 일정 엔티티를 생성하여 저장한다.

3.3.7. Single schedule find service

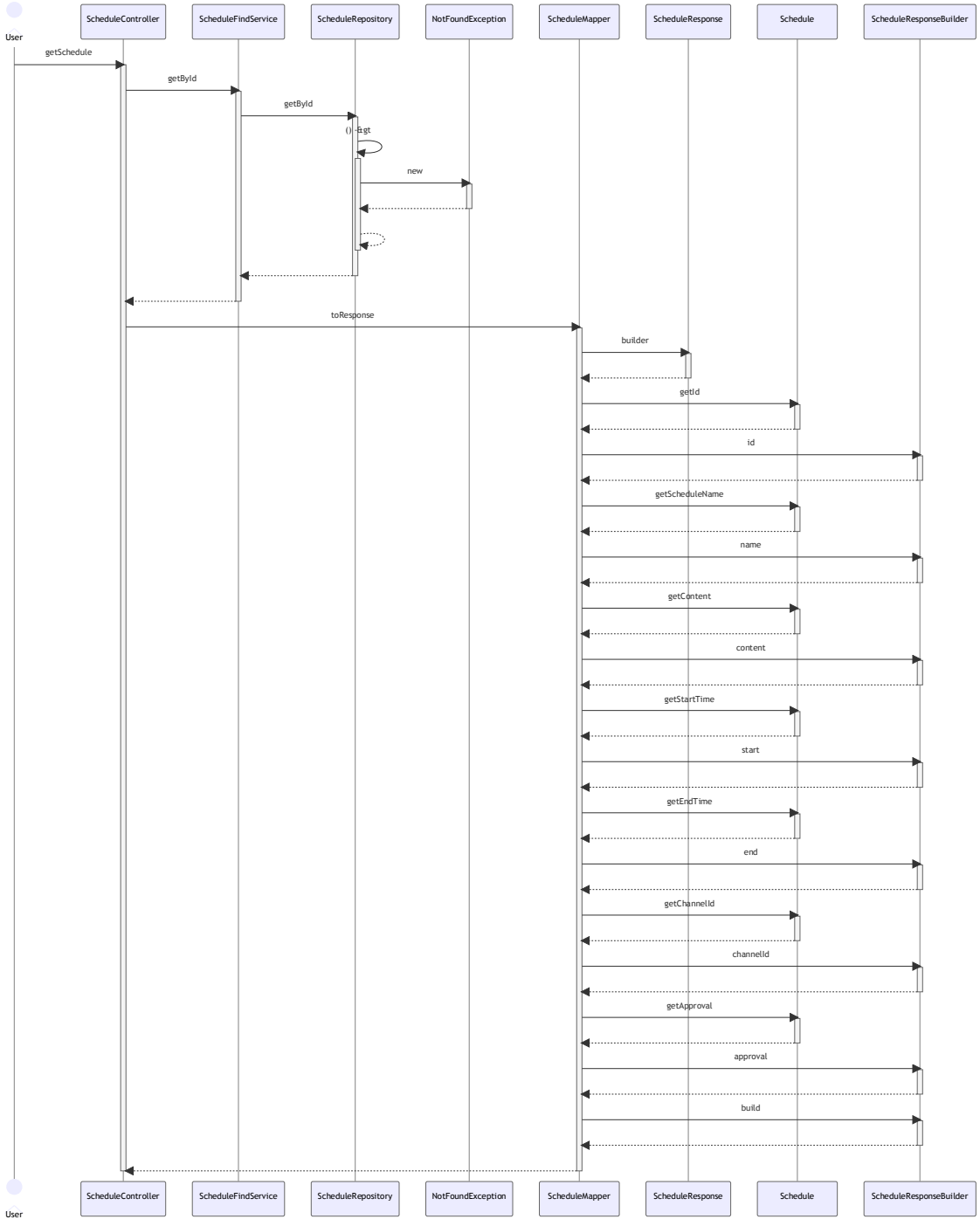


Figure 3-14 Single schedule find service sequence diagram

이 시퀀스 다이어그램은 사용자가 특정 스케줄을 조회 시의 일련의 과정을 나타낸다.

- ScheduleController: 이 클래스에서는 일정 정보 조회 요청을 받고, ScheduleFindService에 처리를 위임하는 역할을 한다.
- ScheduleFindService: 이 클래스에서는 getById() 메소드를 통해 특정 일정의 정보를 조회하고 ScheduleRepository를 사용하여 일정 정보를 조회한다.
  - getById(): 이 메소드는 특정 일정 ID에 해당하는 일정 정보를 조회하는 기능을 수행한

다.

- ScheduleRepository: 이 클래스에서는 getById() 메소드를 통해 일정 정보를 저장하고 조회하는 역할을 수행한다.
  - getById(): 이 메소드는 일정 ID에 해당하는 일정 정보를 조회하는 기능을 수행한다.
- NotFoundException: 이 클래스는 예외 클래스로, 조회된 일정 정보가 없을 경우 에러를 발생시킨다.
- ScheduleMapper: 이 클래스에서는 toResponse() 메소드를 통해 일정 정보를 일정 응답(ScheduleResponse)로 변환하는 역할을 수행한다.
  - toResponse(): 이 메소드는 일정 정보를 일정 응답으로 변환하는 역할을 수행한다.
- Schedule: 이 클래스는 일정 정보를 담고 있는 엔티티 클래스이다.
- ScheduleResponse: 이 클래스는 일정 정보를 담고 있는 응답 클래스이다.
- ScheduleResponseBuilder: 이 클래스는 일정 응답을 구성하는 빌더 클래스이다.

이와 같이 각 클래스와 메소드는 특정 일정 정보 조회 과정에서 필요한 역할을 담당하여 사용자의 요청을 처리하고 일정 응답을 생성하여 반환한다.

3.3.8. Total schedule find service

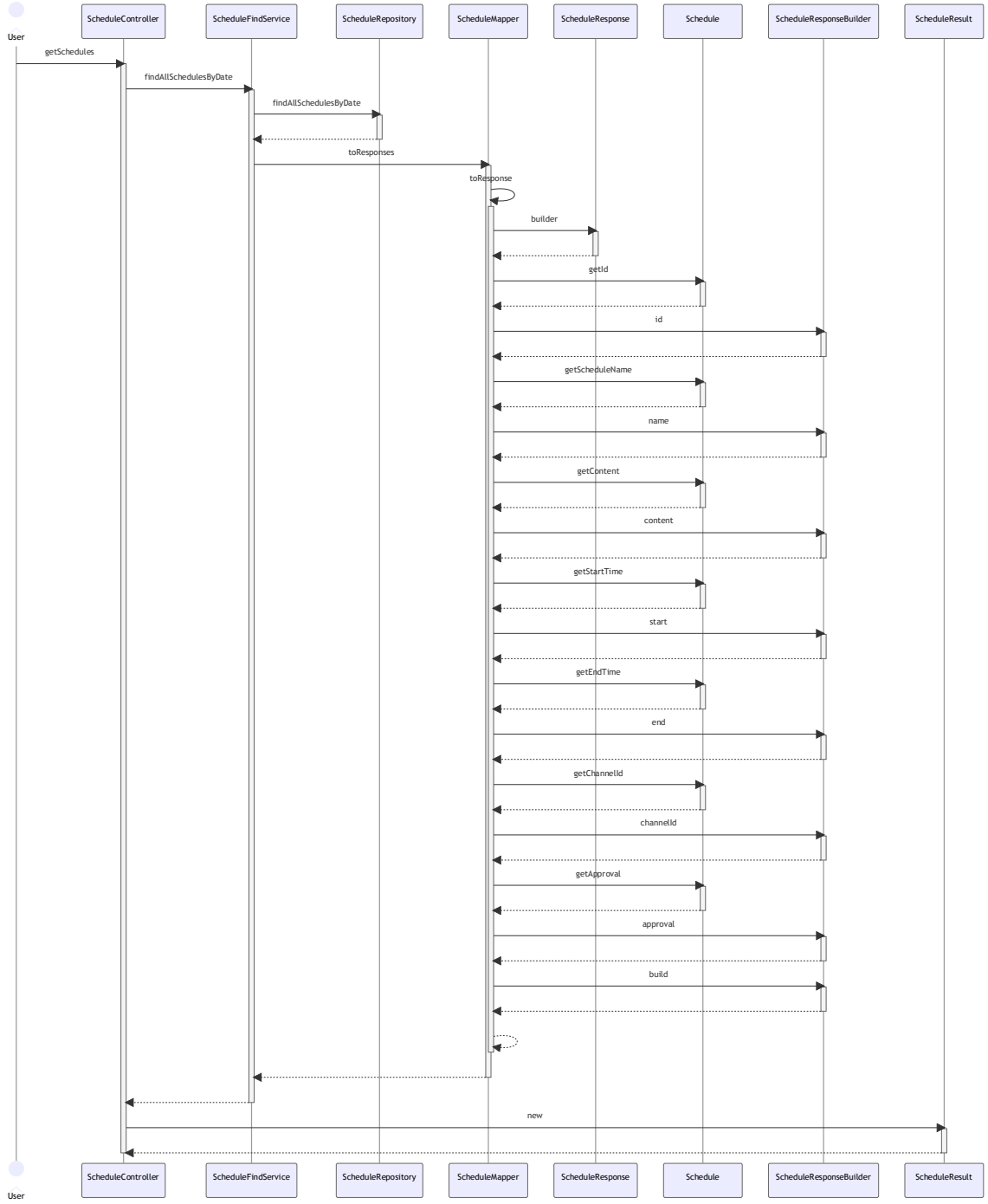


Figure 3-15 Total schedule find sevice sequence diagram

이 시퀀스 다이어그램은 사용자가 특정 날짜의 모든 일정 정보를 조회 시의 일련의 과정을 나타낸다.

- ScheduleController: 이 클래스에서는 일정 정보 조회 요청을 받고, ScheduleFindService에 처리를 위임한다.
- ScheduleFindService: 이 클래스에서는 findAllSchedulesByDate() 메소드를 통해 특정 날짜에 해당하는 모든 일정 정보를 조회하고, ScheduleRepository를 사용하여 일정 정보를 조회한다.

- findAllSchedulesByDate(): 이 메소드는 특정 날짜에 해당하는 모든 일정 정보를 조회하는 기능을 수행한다.

- ScheduleRepository: 이 클래스에서는 findAllSchedulesByDate() 메소드를 통해 일정 정보를 저장하고 조회하는 역할을 수행한다.
  - findAllSchedulesByDate(): 이 메소드는 특정 날짜에 해당하는 모든 일정 정보를 조회하는 기능을 수행한다.
- ScheduleMapper: 이 클래스는 일정 정보를 일정 응답(ScheduleResponse)으로 변환하는 역할을 수행한다.
  - toResponse(): 이 메소드는 일정 정보를 일정 응답으로 변환하는 기능을 수행한다.
- Schedule: 이 클래스는 일정 정보를 담고 있는 엔티티 클래스이다.
- ScheduleResponse: 이 클래스는 일정 정보를 담고 있는 응답 클래스이다.
- ScheduleResponseBuilder: 이 클래스는 일정 응답을 구성하는 빌더 클래스이다.
- ScheduleResult: 이 클래스에서는 조회된 일정 정보들을 담고 있다.

이와 같이 각 클래스와 메소드는 특정 날짜의 모든 일정 정보 조회 과정에서 필요한 역할을 담당하며, 사용자의 요청을 처리하고 일정 응답들을 생성하여 반환한다.

3.3.9. Schedule creation request find service

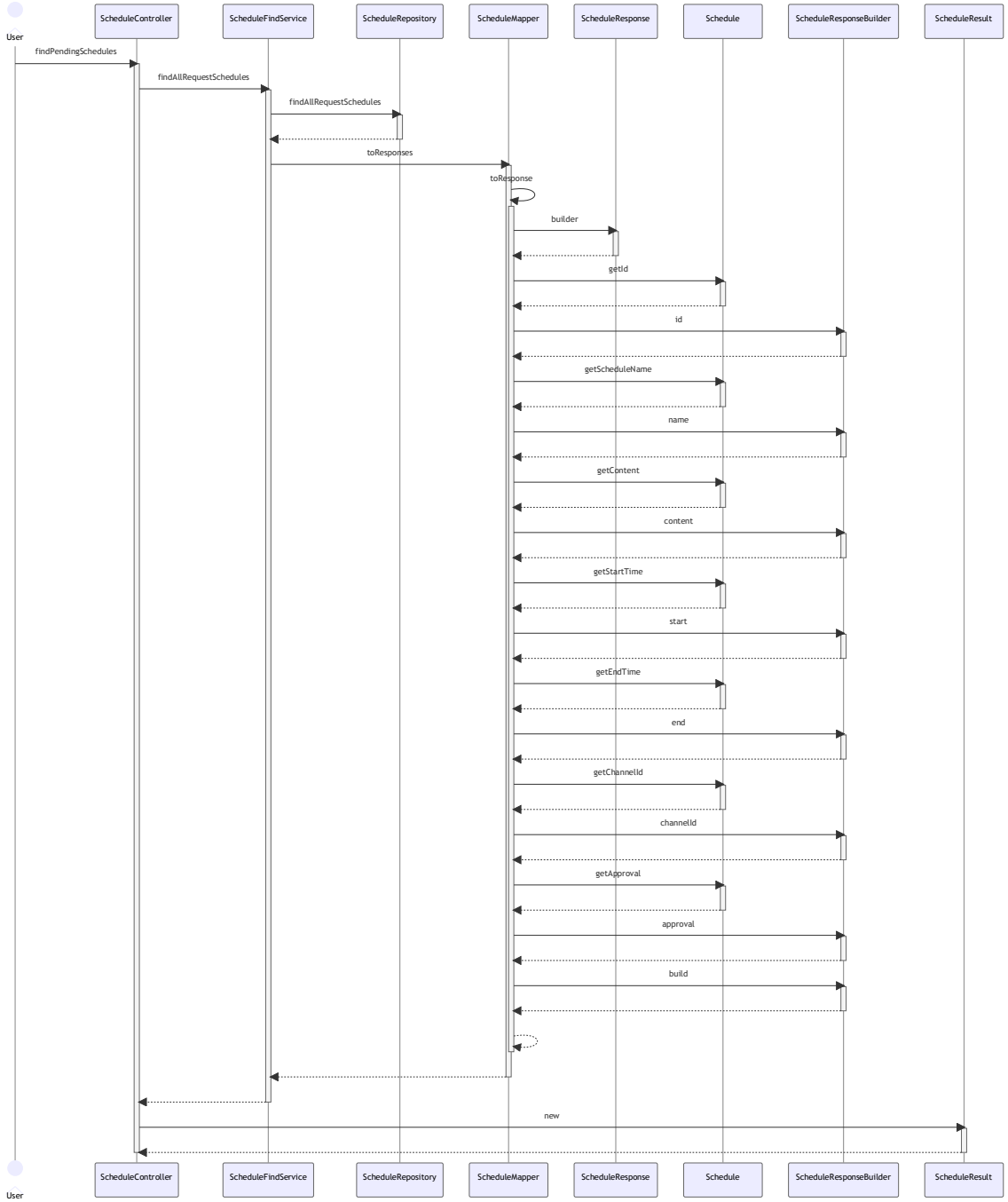


Figure 3-16 Schedule creation request find service sequence diagram.

이 시퀀스 다이어그램은 관리자가 스케줄 생성 요청 목록을 조회하는 일련의 과정을 나타낸다.

- ScheduleController: 이 클래스에서는 일정 정보 조회 요청을 받고, ScheduleFindService에 처리를 위임한다.
- ScheduleFindService: 이 클래스에서는 findAllRequestSchedules() 메소드를 통해 보류 중인 모든 일정 정보를 조회하고, ScheduleRepository를 사용하여 일정 정보를 조회한다.
  - findAllRequestSchedules(): 이 메소드는 보류 중인 모든 일정 정보를 조회하는 기능을 수행한다.

- ScheduleRepository: 이 클래스는 findAllRequestSchedules() 메소드를 통해 일정 정보를 저장하고 조회하는 역할을 수행한다.
  - findAllRequestSchedules(): 이 메소드는 보류 중인 모든 일정 정보를 조회하는 기능을 수행한다.
- ScheduleMapper: 이 클래스는 toResponse() 메소드를 통해 일정 정보를 일정 응답(ScheduleResponse)로 변환하는 역할을 수행한다.
  - toResponse(): 이 메소드는 일정 정보를 일정 응답으로 변환하는 기능을 수행한다.
- Schedule: 이 클래스는 일정 정보를 담고 있는 엔티티 클래스이다.
- ScheduleResponse: 이 클래스는 일정 정보를 담고 있는 응답 클래스이다.
- ScheduleResponseBuilder: 이 클래스는 일정 응답을 구성하는 빌더 클래스이다.
- ScheduleResult: 이 클래스는 조회된 일정 정보들을 담고 있다.

이와 같이 각 클래스와 메소드는 보류 중인 모든 일정 정보 조회 과정에서 필요한 역할을 담당하며, 사용자의 요청을 처리하고 일정 응답들을 생성하여 반환한다.

3.3.10. Schedule approval

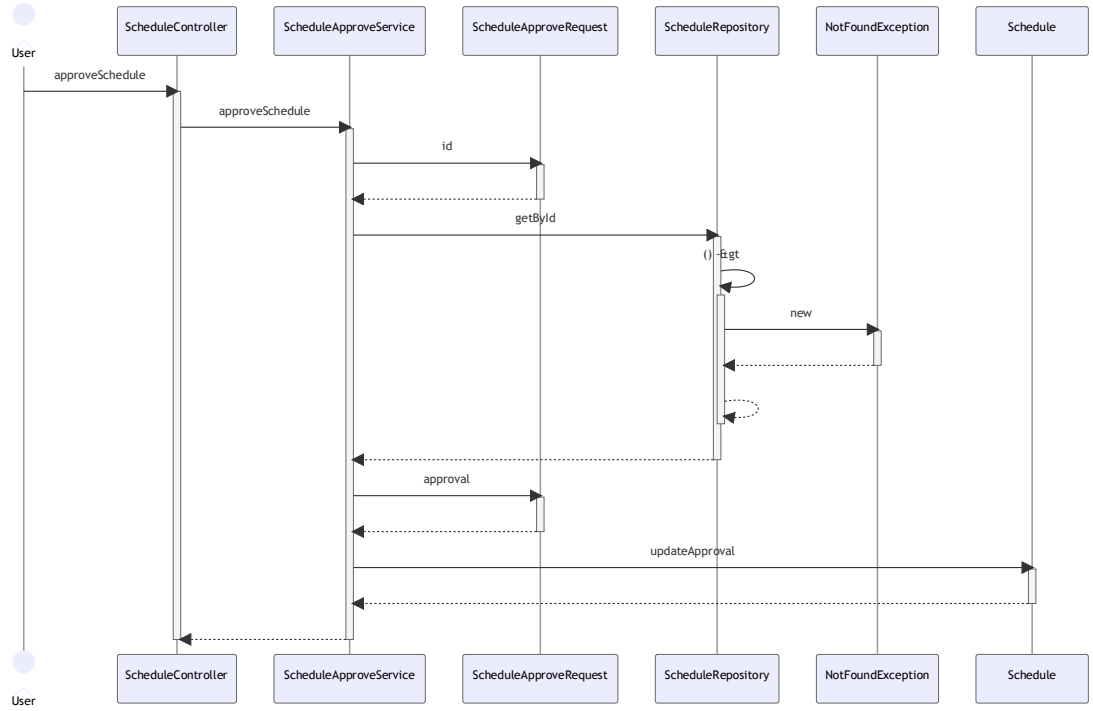


Figure 3-17 Schedule approval sequence diagram

이 시퀀스 다이어그램은 관리자가 특정 일정을 승인하는 일련의 과정을 나타낸다.

- ScheduleController: 이 클래스에서는 일정 승인 요청을 받고, ScheduleApproveService에 처리를 위임한다.
- ScheduleApproveService: 이 클래스는 aproveSchedule() 메소드를 통해 일정 승인을 처리한다.
  - approveSchedule(): 이 메소드는 특정 일정의 승인을 처리하는 기능을 수행한다.

- ScheduleApproveRequest: 이 클래스는 일정 승인 요청 정보를 담고 있다.
- ScheduleRepository: 이 클래스는 getById() 메소드를 통해 일정 정보를 저장하고 조회하는 역할을 수행한다.
  - getById(): 이 메소드는 특정 일정 ID에 해당하는 일정 정보를 조회하는 기능을 수행한다.
- NotFoundException: 이 클래스는 예외 클래스로, 조회된 일정 정보가 없을 경우 에러를 발생시킨다.
- Schedule: 이 클래스는 일정 정보를 담고 있는 엔티티 클래스이다.
  - updateApproval(): 이 메소드는 일정의 승인 상태를 업데이트하는 기능을 수행한다.

이와 같이 각 클래스와 메소드는 일정 승인 요청을 처리하고, 필요한 정보를 조회하며, 일정의 승인 상태를 업데이트한다. 또한 이를 통해 사용자의 승인 요청에 대한 처리를 수행한다.

4. Implementation Details

4.1. Common

4.1.1. Rest API

클라이언트와 서버 간의 효율적인 통신을 위해 REST API를 설계하였다. 이를 통해 클라이언트는 서버에 요청을 보내고, 서버는 그에 대한 응답을 제공하여 데이터를 주고받을 수 있다. 이에 사용된 API 명세는 다음과 같다.

로그인/회원가입 (Token 없이 가능한 API 요청)

Table 4-1 로그인 API

요청	로그인	
HTTP 요청 타입	POST	
경로	/api/signin	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
loginId	string	6자 이상 16자 미만의 숫자 포함가능한 영문 소문자
password	string	8자 이상 16자 미만의 문자열
응답		
Header		
AccessToken	string	JWT access token string
RefreshToken	string	JWT refresh token string
200 성공		
Body		
isAdmin	boolean	Admin 계정일 경우 true, 그렇지 않을 경우 false를 반환한다. 이는 UI 구성에 반영된다.
실패		
Body		

statusCode	int	상태코드
message	string	실패 원인에 대한 설명

Table 4-2 회원가입 API

요청	회원가입	
HTTP 요청 타입	POST	
경로	/api/signup	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
loginId	string	6자 이상 16자 미만의 숫자 포함가능한 영문 소문자
password	string	8자 이상 16자 미만의 문자열
name	string	1자 이상 11자 미만의 문자열
phoneNumber	string	'-' 를 포함하지 않은 13자리 숫자로만 구성된 문자열
roleType	Enum RoleType	Enum 타입으로, "ADMIN" 혹은 "GUEST" 두가지 종류 중 하나
adminPassword	string?	8자 이상 15자 미만의 문자열로 roleType이 ADMIN일 경우에만 필요한 필드
응답		
Header		
AccessToken	string	JWT access token string
RefreshToken	string	JWT refresh token string
200 성공		
Body		
isAdmin	boolean	Admin 계정일 경우 true, 그렇지 않을 경우 false를 반환한다. 이는 UI 구성에 반영된다.
실패		
Body		
statusCode	int	상태코드
message	string	실패 원인에 대한 설명

관리자 및 게스트 인증 및 에러 관련 공통 요청 및 응답 내용

Table 4-3 요청시 Header

요청 Header		
AccessToken	string	JWT access token string
RefreshToken	string	JWT refresh token string

Table 4-4 응답시 Header

응답 Header		
AccessToken	string	서버에서 갱신을 요청할 때에만 적재함. JWT access token string
RefreshToken	string	서버에서 갱신을 요청할 때에만 적재함. JWT refresh token string

Table 4-5 실패시 응답

실패시 응답		
Body		
statusCode	int	상태코드
message	string	실패 원인에 대한 설명

관리자 관련 API (Token 요구)

Table 4-6 채널 생성 API

요청	채널 생성	
HTTP 요청 타입	POST	
경로	/api/admin/channels	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
name	string	채널 이름, 1자 이상 11자 미만의 문자열
place	string	채널 장소, 1자 이상 11자 미만의 문자열
응답		
200 성공		
isAdmin	boolean	Admin 계정일 경우 true, 그렇지 않을 경우 false를 반환한다. 이는 UI 구성에 반영된다.

Table 4-7 채널 수정 API

요청	채널 수정	
HTTP 요청 타입	PATCH	
경로	/api/admin/channels/:id	
Field	**Type**	**설명 / 제약조건**
요청		
Path variable		
id	int	수정할 채널의 id
Body		
name	string?	1자 이상 11자 미만의 문자열

place	string?	1자 이상 11자 미만의 문자열
응답		
200 성공		

Table 4-8 채널 삭제 API

요청	채널 삭제	
HTTP 요청 타입	DELETE	
경로	/api/admin/channels/:id	
Field	**Type**	**설명 / 제약조건**
요청		
Path variable		
id	int	삭제할 채널의 id
응답		
200 성공		

Table 4-9 스케줄 승인요청 승인/거절 API

요청	스케줄 요청에 대한 승인/거절	
HTTP 요청 타입	POST	
경로	/api/admin/schedules/requests	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
id	int	수정할 채널의 id
Approval	Enum Approval	승인/거절 여부 (APPROVED / PENDING / REJECTED)
응답		
200 성공		

Table 4-10 스케줄 승인요청목록 확인 API

요청	스케줄 요청목록 확인	
HTTP 요청 타입	GET	
경로	/api/admin/schedules/requests	
Field	**Type**	**설명 / 제약조건**
응답		
200 성공		
schedules	List<Schedule>	Schedule object들의 배열

게스트 관련 API

Table 4-11 특정채널 상세정보 조회 API

요청	특정채널 상세정보 조회	
HTTP 요청 타입	GET	
경로	/api/guest/channels/:id	
Field	**Type**	**설명 / 제약조건**
요청		
Path variable		
id	int	수정할 채널의 id
Body		
id	int	채널 ID
name	string	채널 제목
place	string	채널 장소
createdBy	string	채널 생성자명
응답		
200 성공		

Table 4-12 전체채널 정보 조회 API

요청	전체채널 정보 조회	
HTTP 요청 타입	GET	
경로	/api/guest/channels	
Field	**Type**	**설명 / 제약조건**
응답		
200 성공		
channels	List<Channel>	Channel Object에 대한 리스트

Table 4-13 특정스케줄 상세정보 조회 API

요청	특정스케줄 상세정보 조회	
HTTP 요청 타입	GET	
경로	/api/guest/schedules/:id	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
id	int	스케줄 ID
응답		
200 성공		
id	int	스케줄 ID
name	string	스케줄 이름

content	string?	스케줄 내용
start	ISO8601 String	스케줄 시작시간
end	ISO8601 String	스케줄 종료시간
channelId	int	스케줄이 생성된 채널 id
approval	Enum Approval	스케줄의 승인 여부 (APPROVED/REJECTED/PENDING)

Table 4-14 전체스케줄 정보 조회 API

요청	전체스케줄 정보 조회	
HTTP 요청 타입	GET	
경로	/api/guest/schedules	
Field	**Type**	**설명 / 제약조건**
요청		
Query Parameter		
start	ISO8601 String	조회할 스케줄들이 시작되는 시점
end	ISO8601 String	조회할 스케줄들이 종료되는 시점
channelId	int	스케줄을 조회할 채널의 id
응답		
200 성공		
schedules	List<Schedule>	Schedule Object들의 배열

Table 4-15 스케줄 삭제 API

요청	스케줄 삭제	
HTTP 요청 타입	DELETE	
경로	/api/guest/schedules/:id	
Field	**Type**	**설명 / 제약조건**
요청		
Path variable		
id	int	삭제할 스케줄의 id
응답		
200 성공		

Table 4-16 스케줄 수정 API

요청	채널 수정	
HTTP 요청 타입	PATCH	
경로	/api/admin/channels/:id	
Field	**Type**	**설명 / 제약조건**
요청		

Path variable		
id	int	수정할 스케줄의 id
Body		
name	string	스케줄 이름
content	string?	스케줄 내용
응답		
200 성공		

Table 4-17 새로운 스케줄 승인요청 API

요청	새로운 스케줄 승인요청	
HTTP 요청 타입	POST	
경로	/api/guest/schedules/requests	
Field	**Type**	**설명 / 제약조건**
요청		
Body		
name	string	스케줄 이름
content	string?	스케줄 내용
start	ISO8601	스케줄 시작시간
end	ISO8601	스케줄 종료시간
channelId	int	스케줄을 생성할 채널
응답		
200 성공		
schedules	List<Schedule>	Schedule Object들의 배열

Table 4-18 승인요청한 스케줄 조회 API

요청	승인요청한 스케줄 불러오기	
HTTP 요청 타입	GET	
경로	/api/guest/schedules/requests	
Field	**Type**	**설명 / 제약조건**
요청		
Query Parameter		
start	ISO8601	조회할 스케줄들의 시작시간
end	ISO8601	조회할 스케줄들의 종료시간
approval	Enum Approval	조회할 스케줄의 현재 승인상태
응답		
200 성공		
schedules	List<Schedule>	Schedule Object들의 배열

API는 Swagger Document로 생성하여 팀원들과 공유하였으며, 해당 문서는 https://weplan.parkjb.com/docs/index.html 에서 확인할 수 있다.

4.1.2. Github

코드 변경 사항을 추적하고 팀원 간의 협업을 위해 Git과 Github를 활용하였다. 이를 통해 코드의 버전 관리와 변경 이력을 관리하고, 여러 팀원들이 동시에 작업을 수행하였다.

4.1.3. Notion

프로젝트의 진행 상황과 문서를 공유하고 일정을 관리하기 위해 Notion을 이용하였다. Notion 내의 Gantt Chart를 통해 프로젝트의 전체 일정을 시각화하여 관리하였다. 또한, 회의록 등의 문서를 공유하고 팀원 간의 연락처와 일정을 관리하는 데 활용하였다.

4.2. Frontend

4.2.1. Flutter

Flutter의 경우 Android와 iOS, Web, macOS, Windows 등 다양한 플랫폼에서 동일한 코드로 앱을 개발할 수 있는 편리함이 있고, 이를 통해 플랫폼에 관계없이 일관된 사용자 경험을 제공할 수 있다. 또한, Flutter는 초기 제공되는 리소스가 풍부하여 개발 속도를 높일 수 있다는 장점이 있다. 다른 Cross-Platform 프론트엔드 개발 라이브러리인 React Native도 고려해보았지만, 단기간에 빠른 완성물을 만들기 위해서는 Flutter가 더 적합하다고 판단하였다. 이러한 이유로 프론트엔드 모바일 앱 개발을 위해 이번 프로젝트에 Flutter를 선택하였다.

4.3. Backend

4.3.1. Spring

백엔드 개발에서 단순한 I/O 처리가 많을 경우 성능이 더 좋은 Node.js를 고려할 수 있지만, 안정적인 멀티 쓰레드 기반 서버 운영과 팀원들의 숙련도를 고려하여 Java 기반의 Spring을 선택하였다. 또한, Spring Boot를 사용하여 내장 톰캣, 자동 구성 기능, 의존성 관리에 활용하였다. 이를 통해 개발과 배포 과정을 간편하게 처리하였다.

데이터베이스 접근 기술로는 Spring Data JPA와 QueryDsl을 활용하여 SQL 쿼리를 작성하였다. 이를 통해 효율적이고 유지보수가 용이한 데이터베이스 접근을 구현하였다.

4.3.2. DB Details

MEMBER

회원 정보를 관리하기 위한 테이블

Table 4-19 Member DB Table

MEMBER					
키	컬럼 설명	컬럼명	값	데이터 타입	옵션
PK	회원 id	member_id	1	bigint	auto_increment, not null
	로그인아이디	login_id	yhwjd	varchar(15)	not null

	비밀번호	password	dslafn@f22fasdlc	varchar(255)	not null
	이름	username	회원 실명	varchar(10)	
	전화번호	phone_number	010-5036-3655	varchar(11)	
	권한	role_type	ADMIN	varchar(10)	not null
	생성일자	created_at	2023-01-01	timestamp	not null
	수정일자	last_modified_at	2023-01-02	timestamp	not null

CHANNEL

채널 정보를 관리하기 위한 테이블

Table 4-20 Channel DB Table

CHANNEL					
키	컬럼 설명	컬럼명	값	데이터 타입	옵션
PK	채널 id	channel_id	1	bigint	auto_increment, not null
	채널생성자 id	member_id	1	bigint	not null
	채널 이름	channel_name	코딩	varchar(10)	not null
	채널 공간	channel_place	구학 101호	varchar(10)	not null
	생성일자	created_at	2023-01-01	timestamp	not null
	수정일자	last_modified_at	2023-01-02	timestamp	not null

SCHEDULE

회원-채널의 연결 테이블로 특정 회원, 특정 채널의 스케줄 정보를 관리하기 위한 테이블

Table 4-21 Schedule DB Table

SCHEDULE					
키	컬럼 설명	컬럼명	값	데이터 타입	옵션
PK	스케줄 id	schedule_id	1	bigint	auto_increment, not null
	회원 id	member_id	1	bigint	not null
	채널 id	channel_id	1	bigint	not null
	회원 이름	username	회원 실명	varchar(10)	not null
	예약명	schedule_name	소공 과제	varchar(10)	not null
	예약 내용	content	6명이서 과제하러 갑니다	varchar(100)	
	시작 시간	start_time	10/1 12:00	timestamp	not null
	종료 시간	end_time	10/1 14:00	timestamp	not null
	예약승인여부	approval	PENDING	varchar(10)	not null

	생성일자	created_at	2023-01-01	timestamp	not null
	수정일자	last_modified_at	2023-01-02	timestamp	not null

JWT_REFRESH_TOKEN

회원 로그인 유지를 위한 JWT Refresh Token을 관리하기 위한 테이블

Table 4-22 JWT_REFRESH_TOKEN DB Table

JWT_REFRESH_TOKEN					
키	컬럼 설명	컬럼명	도메인	데이터 타입	옵션
PK	토큰 id	jwt_refresh_token_id	1	bigint	auto_increment, not null
	회원 id	member_id	1	bigint	not null
	리프레시 토큰	refresh_token	dalkfdasf@fdasf	varchar(255)	not null
	생성일자	created_at	2023-12-01 12:00:01.019147	timestamp	not null
	수정일자	last_modified_at	2023-12-02 00:16:52.536158	timestamp	not null

4.3.3. Linux Server

리눅스 기반의 홈서버를 구성하여 배포를 진행하였다. nginx를 이용하여 리버스 프록시 서버를 구성하였고, 보안을 위해 인증서를 발급받아 https 프로토콜을 적용하였다. 해당하는 서버의 주소명은 weplan.parkjb.com 이다.

5. Testing Details

5.1. Validation Criteria

Table 5-1 Validation Criteria 1

No.	SYS_RQ_VA_001	Related Requirement	USER_RQ_009 SYS_RQ_NFR_001
Title	시스템은 사용자 친화적이고 자연스러운 흐름의 UI 및 UX를 제공해야 한다.		
Metrics	설문 조사 및 레퍼런스 조사		
Measure	예비 사용자 설문 조사 및 비슷한 서비스 레퍼런스 조사		

Table 5-2 Validation Criteria 2

No.	SYS_RQ_VA_002	Related Requirement	SYS_RQ_NFR_006
Title	클라이언트 앱의 반응 시간은 1초 이내여야 한다		
Metrics	시간		
Measure	Dart Testing tool		

Table 5-3 Validation Criteria 3

No.	SYS_RQ_VA_003	Related Requirement	SYS_RQ_NFR_007
Title	정상적으로 네트워크에 연결된 상황 (Chrome Network Throttling Fast 3G 기준)에서 3초 이내에 백엔드 서버의 요청에 대한 응답을 받을 수 있어야 한다.		
Metrics	시간		
Measure	Java Testing tool, Dart Testing Tool		

5.2. Scope and Objective of Testing

본 프로젝트에서 수행할 테스트의 종류로는 단위 테스트, 통합 테스트, QA 테스트가 있다.

- 단위 테스트(Unit Test)

단위 테스트는 소프트웨어의 가장 작은 단위인 모듈 또는 함수를 개별적으로 테스트하는 과정이다. 이 테스트는 개별 모듈이 예상대로 작동하는지 확인하고, 모듈 내의 버그를 발견하고 수정하는 데 사용된다. 단위 테스트는 모듈의 기능, 입력값, 출력값, 예외 처리 등을 확인하여 코드의 신뢰성과 품질을 향상시키는 역할을 한다. 주로 테스트 도구를 사용하여 개발자가 작성한 테스트 케이스를 실행하고 결과를 분석한다.

- 통합 테스트(Integration Test)

통합 테스트는 단위 테스트된 모듈들을 조합하여 전체 시스템이 예상대로 작동하는지 확인하는 과정이다. 이 테스트는 모듈 간의 상호작용, 데이터 흐름, 인터페이스 호환성 등을 검증하여 전체 시스템의 정상적인 작동을 확인한다. 통합 테스트는 개별 모듈의 동작만 검증하는 것이 아니라, 모듈 간의 통합에서 발생할 수 있는 문제점을 찾아내는 역할을 한다. 마찬가지로 테스트 도구를 이용하며 시스템의 일부 또는 전체를 대상으로 테스트 케이스를 수행한다.

- QA 테스트(Quality Assurance Test)
QA 테스트는 소프트웨어가 사용자의 요구사항을 충족하고, 품질 기준을 만족하는지 평가하는 과정이다. 이 테스트는 사용자의 관점에서 소프트웨어를 테스트하며, 기능적 요구사항, 비기능적 요구사항, 사용자 경험 등을 평가한다. QA 테스트는 시스템의 완전성, 안정성, 보안성, 성능 등을 확인하여 사용자에게 안정적이고 품질 좋은 소프트웨어를 제공하는 역할을 한다. 배포 직전에 팀원들이 직접 앱을 이용해보고 다양한 테스트 기법과 도구를 사용하여 시스템을 평가한다.
또한 단위 테스트, 통합 테스트에서는 네트워크 지연 시간, 토큰의 만료 시간, 스케줄 등록 후 변하는 시간에 따른 상황 변화에 대한 테스트는 진행하기 어렵다. 이 부분은 QA 테스트에서 해당 요구사항을 만족하는지 확인한다.

이렇게 단위 테스트, 통합 테스트, QA 테스트는 소프트웨어 개발 과정에서 각각의 목적과 범위에 따라 다른 종류의 테스트를 수행한다. 이러한 테스트들을 통해 소프트웨어의 품질을 향상시키고, 안정적인 시스템을 제공할 수 있다.

5.3. Unit Test Cases

5.3.1. Channel

- CH_UT_01: 형식에 맞는 데이터로 채널을 생성
- CH_UT_02: 형식에 맞지 않는 데이터로 채널을 생성
- CH_UT_03: 서버에 등록된 채널 ID로 단일 채널 조회
- CH_UT_04: 서버에 등록되지 않은 채널 ID로 단일 채널 조회
- CH_UT_05: 채널 ID 리스트로 전체 채널 조회
- CH_UT_06: 서버에 등록된 채널 ID로 채널 삭제
- CH_UT_07: 서버에 등록되지 않은 채널 ID로 채널 삭제
- CH_UT_08: 채널 삭제 권한이 없는 관리자가 채널 삭제
- CH_UT_09: 서버에 등록된 채널 ID로 형식에 맞는 데이터로 채널 정보 업데이트
- CH_UT_10: 서버에 등록된 채널 ID로 형식에 맞지 않는 데이터로 채널 정보 업데이트
- CH_UT_11: 서버에 등록되지 않은 채널 ID로 채널 정보 업데이트
- CH_UT_12: 채널 수정 권한이 없는 관리자가 채널 정보 업데이트

5.3.2. Member

- MEM_UT_01: 서버에 등록되지 않은 정보로 형식에 맞는 데이터로 관리자 회원가입
- MEM_UT_02: 서버에 등록되지 않은 정보로 형식에 맞는 데이터로 게스트 회원가입
- MEM_UT_03: 서버에 등록되지 않은 정보로 형식에 맞지 않는 데이터로 회원가입
- MEM_UT_04: 서버에 이미 등록된 정보로 회원가입
- MEM_UT_05: 등록된 ID 및 패스워드로 로그인
- MEM_UT_06: 등록되지 않은 ID 및 패스워드로 로그인

5.3.3. JWT Token

- JWT_UT_01: 회원 정보를 가지고 있는 AccessToken을 생성
- JWT_UT_02: 회원 ID를 가지고 있는 RefreshToken을 생성
- JWT_UT_03: HTTP 헤더 필드 AccessToken에 토큰 삽입
- JWT_UT_04: HTTP 헤더 필드 RefreshToken에 토큰 삽입
- JWT_UT_05: HTTP 헤더 필드에 로그인 토큰 정보 삽입
- JWT_UT_06: 회원 정보를 가지고 있는 AccessToken을 발급
- JWT_UT_07: 회원 ID를 가지고 있는 RefreshToken을 발급
- JWT_UT_08: 로그인한 회원의 토큰 관련 정보를 DB에 저장
- JWT_UT_09: 이미 존재하는 회원의 토큰 관련 정보 DB에서 업데이트

5.3.4. Schedule

- S_UT_01: 형식에 맞는 데이터로 DateTime이 겹치는 스케줄 생성
- S_UT_02: 형식에 맞는 데이터로 DateTime이 겹치지 않는 스케줄 생성
- S_UT_03: 형식에 맞지 않는 데이터로 스케줄 생성
- S_UT_04: 단일 스케줄 조회
- S_UT_05: 해당 기간에 승인된 스케줄 목록 조회
- S_UT_06: 요청받은 스케줄 조회
- S_UT_07: 승인 대기 스케줄 조회
- S_UT_08: 승인 대기 스케줄을 승인 상태로 변경
- S_UT_09: 승인 대기 스케줄을 거절 상태로 변경
- S_UT_10: 스케줄 삭제
- S_UT_11: 등록된 스케줄 ID를 사용해 형식에 맞는 데이터로 스케줄 업데이트
- S_UT_12: 등록된 스케줄 ID를 사용해 형식에 맞지 않는 데이터로 스케줄 업데이트
- S_UT_13: 등록되지 않은 스케줄 ID를 사용해 스케줄 업데이트
- S_UT_14: 스케줄 변경 요청 시도
- S_UT_15: 스케줄 생성/조회/삭제/변경 시 서버와의 통신 시간 3초 준수 여부 확인

5.3.5. Mobile

- MO_UT_01: 형식에 맞는 데이터로 회원가입
- MO_UT_02: 형식에 맞지 않는 데이터로 회원가입
- MO_UT_03: 관리자 계정으로 회원가입
- MO_UT_04: 등록된 회원정보로 로그인
- MO_UT_05: 등록되지 않은 회원정보로 로그인
- MO_UT_06: 채널 생성
- MO_UT_07: 스케줄 생성
- MO_UT_08: 스케줄 상세정보 확인
- MO_UT_09: 채널 테이블 색상 해시값 적용 여부 확인
- MO_UT_10: 스케줄 삭제

- MO_UT_11: 스케줄 변경
- MO_UT_12: 관리자 계정으로 관리 페이지 활성화 여부 확인 및 접속
- MO_UT_13: 요청받은 스케줄 목록 확인
- MO_UT_14: 요청 스케줄 승인
- MO_UT_15: 요청 스케줄 거절
- MO_UT_16: 30초마다 타임 테이블 업데이트 여부 확인
- MO_UT_17: 상호작용 시 애플리케이션 반응 속도 1초 준수 여부 확인

5.4. Test Cases for each Use Case

5.4.1. T_01 Test: 회원 가입

- T_01_1 Test: 사용자가 서비스에 가입한다.
- T_01_2 Test: 사용자가 형식에 맞지 않는 데이터를 입력하면 회원 가입에 실패한다.
- T_01_3 Test: 사용자가 이미 존재하는 회원정보를 입력 시 회원 가입에 실패한다.

Table 5-4 Test Case 01-1

TEST STEPS	
Test Case ID :	T_01_1
Test Case Name :	회원 가입
For Use Case :	사용자가 서비스에 가입한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, SYS_RQ_FR_001, SYS_RQ_FR_004
Test Type :	Negative
Precondition :	사용자는 시스템에 현재 등록되어 있지 않아야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 앱에 접속한다.	사용자에게 서비스 첫 페이지를 보여준다.
2	사용자가 회원가입 버튼을 누른다.	사용자에게 회원가입 페이지를 보여준다.
3	사용자가 유효한 형식의 회원가입 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
4	백엔드 서버에서 형식에 맞는 데이터를 전송받는다.	유효성 검사 후 해당 정보를 DB에 저장한다.
5	회원가입에 성공한다.	사용자가 회원가입에 성공한다.

Table 5-5: Test Case 01-2

TEST STEPS

Test Case ID :	T_01_2
Test Case Name :	회원 가입
For Use Case :	사용자가 형식에 맞지 않는 데이터를 입력하면 회원 가입에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, SYS_RQ_FR_001, SYS_RQ_FR_004
Test Type :	Negative
Precondition :	사용자는 시스템에 현재 등록되어 있지 않아야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 앱에 접속한다.	사용자에게 서비스 첫 페이지를 보여준다.
2	사용자가 회원가입 버튼을 누른다.	사용자에게 회원가입 페이지를 보여준다.
3	사용자가 유효하지 않은 회원가입 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
4	백엔드 서버에서 형식에 맞지 않는 데이터를 전송받는다.	유효성 검사 후 해당 정보를 저장하지 않고 예외를 발생시킨다.
5	회원가입에 실패한다.	애플리케이션에서 사용자에게 데이터 형식이 일치하지 않음을 알린다.

Table 5-6: Test Case 01-3

TEST STEPS	
Test Case ID :	T_01_3
Test Case Name :	회원 가입
For Use Case :	사용자가 이미 존재하는 회원정보를 입력 시 회원 가입에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, SYS_RQ_FR_001, SYS_RQ_FR_004
Test Type :	Negative
Precondition :	사용자는 시스템에 현재 등록되어 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 앱에 접속한다.	사용자에게 서비스 첫 페이지를 보여준다.
2	사용자가 회원가입 버튼을 누른다.	사용자에게 회원가입 페이지를 보여준다.
3	사용자가 이미 등록되어 있는 회원가입 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.

4	백엔드 서버에서 중복 데이터를 전송받는다.	유효성 검사 후 해당 정보를 저장하지 않고 예외를 발생시킨다.
5	회원가입에 실패한다.	애플리케이션에서 사용자에게 해당 가입 정보가 이미 존재함을 알린다.

5.4.2. T_02 Test: 로그인

- T_02_1 Test: 사용자가 서비스에 로그인한다.
- T_02_2 Test: 사용자가 등록되어 있지 않은 회원 정보를 입력하면 로그인에 실패한다.
- T_02_3 Test: 사용자가 올바르지 않은 토큰으로 로그인을 시도할 경우 로그인에 실패한다.
- T_02_4 Test: 관리자가 서비스에 로그인할 경우 관리자만이 접근 가능한 UI를 제공한다.

Table 5-7: Test Case 02-1

TEST STEPS		
Test Case ID :	T_02_1	
Test Case Name :	로그인	
For Use Case :	사용자가 서비스에 로그인한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002, SYS_RQ_FR_003	
Test Type :	Positive	
Precondition :	사용자는 시스템에 현재 등록되어 있어야 하며, 현재 AccessToken을 갖고 있지 않거나 기간이 만료되었어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 로그인 페이지에 접속한다.	사용자에게 서비스의 로그인 페이지를 보여준다.
2	사용자가 유효한 로그인 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 유효한 데이터를 전송받는다.	사용자가 입력한 정보와 일치하는 사용자 정보가 존재하면 로그인에 성공한다.
4	회원가입에 성공한다.	사용자에게 메인 페이지를 보여준다.
5	사용자에게 토큰을 발급한다.	사용자에게 AccessToken, RefreshToken을 발급하여 다음 요청시에도 로그인이 유지될 수 있도록 한다.

Table 5-8: Test Case 02-2

TEST STEPS		
Test Case ID :	T_02_2	
Test Case Name :	로그인	
For Use Case :	사용자가 등록되어 있지 않은 회원 정보를 입력하면 로그인에 실패한다	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002, SYS_RQ_FR_003	
Test Type :	Negative	
Precondition :	시스템에 등록된 사용자 정보가 있어야 하며, 현재 AccessToken을 갖고 있지 않거나 기간이 만료되었어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 로그인 페이지에 접속한다.	사용자에게 서비스의 로그인 페이지를 보여준다.
2	사용자가 DB에 등록되지 않은 로그인 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 중복되지 않은 데이터를 전송받는다.	서버는 해당 정보와 일치하는 데이터가 DB에 없음을 확인하고 예외를 발생시킨다.
4	로그인에 실패한다.	애플리케이션은 사용자에게 해당 데이터 정보가 없음을 알린다.

Table 5-9: Test Case 02-3

TEST STEPS		
Test Case ID :	T_02_3	
Test Case Name :	로그인 실패	
For Use Case :	사용자가 올바르지 않은 토큰정보로 로그인을 시도할 경우 로그인에 실패한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002	
Test Type :	Negative	
Precondition :	사용자는 시스템에 현재 등록되어 있어야 하며, 기간이 만료되거나 올바르지 않은 AccessToken을 가지고 었어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**

1	로그인 페이지에 접속한다.	사용자에게 서비스의 로그인 페이지를 보여준다.
2	디버깅 페이지에서 임의로 AccessToken을 조작한다.	애플리케이션은 서버에 해당 AccessToken으로 요청을 보낸다.
3	백엔드 서버에서 유효하지 않은 토큰임을 확인하여 응답한다.	애플리케이션은 해당 AccessToken이 유효하지 않음을 확인하고 즉시 로그인 화면으로 복귀한다.

Table 5-10: Test Case 02-4

TEST STEPS	
Test Case ID :	T_02_4
Test Case Name :	관리자 로그인
For Use Case :	사용자가 올바르지 않은 토큰정보로 로그인을 시도할 경우 로그인에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002, SYS_RQ_FR_005
Test Type :	Positive
Precondition :	사용자는 시스템에 현재 관리자로 등록되어 있어야 하며, 현재 AccessToken을 갖고 있지 않거나 기간이 만료되어 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	로그인 페이지에 접속한다.	사용자에게 서비스의 로그인 페이지를 보여준다.
2	관리자 권한을 가진 계정으로 로그인을 수행한다.	애플리케이션은 서버에 인증토큰을 요청한다.
3	백엔드 서버에서 관리자임을 확인시켜준다.	애플리케이션은 해당 사용자가 관리자임을 서버에서 확인하고 이에 관리자만 이용가능한 UI를 제공한다.

5.4.3. T_03 Test: 채널 생성

- T_03_1 Test: 관리자 권한을 가진 사용자가 채널을 생성한다.
- T_03_2 Test: 관리자 권한을 가진 사용자가 채널 생성 시 유효하지 않은 값을 입력한 경우 채널 생성에 실패한다.

Table 5-11: Test Case 03-1

TEST STEPS		
Test Case ID :	T_03_1	
Test Case Name :	채널 생성	
For Use Case :	관리자 권한을 가진 사용자가 채널을 생성한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002, SYS_RQ_FR_003, SYS_RQ_FR_005	
Test Type :	Positive	
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 생성 페이지로 이동한다.	관리자에게 채널 생성 페이지를 보여준다.
2	관리자가 채널 생성 페이지에서 채널 생성에 필요한 값을 입력한다	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 유효한 데이터를 전송받는다.	관리자가 유효한 값을 입력한 경우, 애플리케이션은 관리자 사용자에게 채널 생성에 성공했음을 알린다.
4	DB에 채널 정보를 저장한다.	해당 데이터를 DB에 저장하고 채널을 생성한다.

Table 5-12: Test Case 03-2

TEST STEPS		
Test Case ID :	T_03_2	
Test Case Name :	채널 생성	
For Use Case :	관리자 권한을 가진 사용자가 채널 생성 시 유효하지 않은 값을 입력한 경우 채널 생성에 실패한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_002, USER_RQ_003, USER_RQ_004, SYS_RQ_FR_002, SYS_RQ_FR_003, SYS_RQ_FR_005	
Test Type :	Negative	
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**

1	관리자가 채널 생성 페이지로 이동한다.	관리자에게 채널 생성 페이지를 보여준다.
2	관리자가 채널 생성 페이지에서 유효하지 않은 값을 입력한다	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 유효하지 않은 데이터를 전송받는다.	관리자가 유효하지 않은 값을 입력한 경우, 서버는 예외를 발생시킨다.
4	채널 생성에 실패한다.	애플리케이션은 관리자 사용자에게 유효하지 않은 값을 입력했음을 알린다.

5.4.4. T_04 Test: 채널 관리자 페이지 접속

- T_04_1 Test: 관리자 권한을 가진 사용자가 채널 관리자 페이지에 접속한다.

Table 5-13: Test Case 04-1

TEST STEPS		
Test Case ID :	T_04_1	
Test Case Name :	채널 관리자 페이지 접속	
For Use Case :	관리자 권한을 가진 사용자가 채널 관리자 페이지에 접속한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_02, SYS_RQ_FR_005	
Test Type :	Positive	
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 정보 창을 연다.	애플리케이션은 관리자에게 채널 정보 창을 보여주고, 관리자 페이지 버튼을 활성화시킨다.
2	관리자가 관리자 페이지 접속 버튼을 누른다.	서버는 현재 접속한 유저의 멤버의 관리자 type이 true인지를 확인하고, 관리자 페이지로 연결시킨다.
3	관리자 페이지로 이동한다.	관리자가 관리자 페이지에서 작업할 수 있게 된다.

5.4.5. T_05 Test: 채널 목록 조회

- T_05_1 Test: 사용자가 서비스에서 채널 목록을 조회한다.

Table 5-14: Test Case 05-1

TEST STEPS

Test Case ID :	T_05_1
Test Case Name :	채널 목록 조회
For Use Case :	사용자가 서비스에서 채널 목록을 조회한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_008, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008
Test Type :	Positive
Precondition :	사용자는 시스템에 로그인해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 모든 채널 조회 페이지로 이동한다.	사용자에게 채널 목록 조회 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	DB로부터 가져온 정보를 보여준다.	사용자에게 조회 페이지에서 채널 목록 정보를 보여준다.

5.4.6. T_06 Test: 특정 채널 조회

- T_06_1 Test: 관리자 권한을 가진 사용자가 채널을 생성한다.

Table 5-15: Test Case 06-1

TEST STEPS	
Test Case ID :	T_06_1
Test Case Name :	특정 채널 조회
For Use Case :	사용자가 서비스에서 특정 채널을 조회한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_008, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008
Test Type :	Positive
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 채널 목록 페이지로 이동한다.	사용자에게 채널 목록 조회 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	사용자가 특정 채널을 클릭한	DB로부터 해당 채널의 ID값으로 채널 정보를 조회해

	다.	상세 정보를 가져온다.
4	사용자에게 DB로부터 가져온 정보를 보여준다.	사용자에게 해당 채널 페이지를 보여주고, 테이블 형식으로 해당 채널의 현재 스케줄을 보여준다..

5.4.7. T_07 Test: 특정 채널 수정

- T_07_1 Test: 관리자 권한을 가진 사용자가 서비스에서 특정 채널을 수정한다.
- T_07_2 Test: 관리자 권한을 가진 사용자가 서비스에서 유효하지 않은 정보로 특정 채널의 수정을 시도할 경우, 수정에 실패한다.
- T_07_3 Test: 관리자 권한을 가진 사용자가 해당 계정에서 생성하지 않은 채널의 수정 페이지에 접근하는 경우 수정에 실패한다.

Table 5-16: Test Case 07-1

TEST STEPS	
Test Case ID :	T_07_1
Test Case Name :	특정 채널 수정
For Use Case :	관리자 권한을 가진 사용자가 서비스에서 특정 채널을 수정한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_005, SYS_RQ_FR_001, SYS_RQ_FR_002, SYS_RQ_FR_005
Test Type :	Positive
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 하며, 수정되는 채널은 해당 관리자 계정에 의해 생성된 계정이어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 채널 관리 페이지로 이동한다.	관리자에게 채널 관리 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	사용자가 수정할 채널 정보를 입력하여 수정을 시도한다.	해당 정보를 백엔드 서버로 전송한다.
4	백엔드 서버에서 유효한 데이터를 전송받는다.	관리자가 유효한 값을 입력한 경우, 서버는 해당 정보로 채널 정보를 업데이트하고 애플리케이션은 관리자 사용자에게 채널 수정에 성공했음을 알린다
5	DB에서 채널 수정 정보를 반영한다.	DB에서 입력받은 정보로 채널 정보를 업데이트한다.

Table 5-17: Test Case 07-2

TEST STEPS	
Test Case ID :	T_07_2
Test Case Name :	특정 채널 수정
For Use Case :	관리자 권한을 가진 사용자가 서비스에서 유효하지 않은 정보로 특정 채널의 수정을 시도할 경우, 수정에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_005, SYS_RQ_FR_001, SYS_RQ_FR_002, SYS_RQ_FR_005
Test Type :	Negative
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 하며, 수정되는 채널은 해당 관리자 계정에 의해 생성된 계정이어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 관리 페이지로 이동한다.	관리자에게 채널 관리 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	관리자가 유효하지 않은 수정할 채널 정보를 입력하여 수정을 시도한다.	해당 정보를 백엔드 서버로 전송한다.
4	백엔드 서버에서 유효한 데이터를 전송받는다.	관리자가 유효하지 않은 값을 입력한 경우, 서버는 예외를 발생시킨다.
5	채널 정보 수정에 실패한다.	애플리케이션은 관리자 사용자에게 입력한 정보가 유효하지 않음을 알린다.

Table 5-18: Test Case 07-3

TEST STEPS	
Test Case ID :	T_07_3
Test Case Name :	특정 채널 수정
For Use Case :	관리자 권한을 가진 사용자가 해당 계정에서 생성하지 않은 채널의 수정 페이지에 접근하는 경우 수정에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_005, SYS_RQ_FR_001, SYS_RQ_FR_002, SYS_RQ_FR_005
Test Type :	Negative
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 하며,

	수정되는 채널은 해당 관리자 계정에 의해 생성된 계정이 아니어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 관리 페이지로 이동한다.	관리자에게 채널 관리 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	관리자가 해당 계정에서 생성하지 않은 채널의 수정 정보를 입력한다.	해당 정보를 백엔드 서버로 전송한다.
4	백엔드 서버에서 수정을 시도하는 유저 정보가 해당 채널 관리자 ID와 불일치함을 확인한다.	관리자 정보가 일치하지 않는 경우, 서버는 예외를 발생시킨다.
5	채널 정보 수정에 실패한다.	애플리케이션은 관리자 사용자에게 해당 채널을 수정할 권한이 없음을 알린다.

5.4.8. T_08 Test: 특정 채널 삭제

- T_08_1 Test: 관리자 권한을 가진 사용자가 특정 채널을 삭제한다.
- T_08_2 Test: 관리자 권한을 가진 사용자가 해당 계정에서 생성하지 않은 채널의 삭제를 시도하는 경우 삭제에 실패한다.

Table 5-19: Test Case 08-1

TEST STEPS		
Test Case ID :	T_08_1	
Test Case Name :	특정 채널 삭제	
For Use Case :	관리자 권한을 가진 사용자가 특정 채널을 삭제한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_005, SYS_RQ_FR_001, SYS_RQ_FR_002, SYS_RQ_FR_005	
Test Type :	Positive	
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 하며, 수정되는 채널은 해당 관리자 계정에 의해 생성된 계정이어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 관리 페이지로	관리자에게 채널 관리 페이지를 보여준다.

	이동한다.	
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	관리자가 삭제할 채널을 선택한다.	해당 정보를 백엔드 서버로 전송한다.
4	백엔드 서버에서 해당 채널 정보를 삭제한다.	서버는 해당 채널 ID와 일치하는 ID를 DB에서 찾아, 해당 채널 정보를 삭제한다.
5	채널 삭제에 성공한다.	애플리케이션은 관리자 사용자에게 채널 삭제가 완료되었음을 알린다.

Table 5-20: Test Case 08-2

TEST STEPS	
Test Case ID :	T_08_2
Test Case Name :	특정 채널 삭제
For Use Case :	관리자 권한을 가진 사용자가 해당 계정에서 생성하지 않은 채널의 삭제를 시도하는 경우 삭제에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_005, SYS_RQ_FR_001, SYS_RQ_FR_002, SYS_RQ_FR_005
Test Type :	Negative
Precondition :	사용자는 관리자 권한을 가진 계정을 사용해 시스템에 로그인해야 하며, 수정되는 채널은 해당 관리자 계정에 의해 생성된 계정이 아니어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 채널 관리 페이지로 이동한다.	관리자에게 채널 관리 페이지를 보여준다.
2	DB로부터 모든 채널의 정보를 가져온다.	채널 정보에는 채널명, 채널 장소, 채널 관리자명이 있다. 이 정보를 DB 서버로부터 가져온다.
3	관리자가 삭제할 채널로 해당 계정에서 생성하지 않은 채널을 선택한다.	해당 정보를 백엔드 서버로 전송한다.
4	백엔드 서버에서 삭제를 시도하는 유저 정보가 해당 채널 관리자 ID와 불일치함을 확인한다.	관리자 정보가 일치하지 않는 경우, 서버는 예외를 발생시킨다.
5	채널 삭제에 실패한다.	애플리케이션은 관리자 사용자에게 해당 채널을 삭제할 권한이 없음을 알린다.

5.4.9. T_09 Test: 채널 타임 테이블 조회

- T_09_1 Test: 사용자가 서비스에서 스케줄을 조회한다.
- T_09_2 Test: 스케줄이 30초 이내에 자동으로 타임테이블을 업데이트하여 제공한다.

Table 5-21: Test Case 09-1

TEST STEPS		
Test Case ID :	T_09_1	
Test Case Name :	채널 타임 테이블 조회	
For Use Case :	사용자가 서비스에서 스케줄을 조회한다.	
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_NFR_002	
Test Type :	Positive	
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 특정 채널을 클릭한다.	DB로부터 해당 채널의 ID값으로 채널 정보를 조회해 상세 정보를 가져온다.
2	DB로부터 특정 채널의 모든 스케줄의 정보를 모두 가져온다.	스케줄 정보에는 멤버와 채널 ID, 유저와 스케줄 이름, 시작 및 종료 시간 등의 정보 등이 있다. 이 정보들을 DB 서버로부터 가져온다.
3	사용자에게 DB로부터 가져온 정보를 보여준다.	사용자에게 해당 채널 페이지를 보여주고, 테이블 형식으로 해당 채널의 현재 스케줄을 보여준다.

Table 5-22: Test Case 09-2

TEST STEPS	
Test Case ID :	T_09_2
Test Case Name :	채널 타임 테이블 자동 업데이트
For Use Case :	사용자가 서비스에서 스케줄을 조회한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_NFR_002, SYS_RQ_NFR_005
Test Type :	Positive
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.
TEST STEPS	

#	Steps	Expected Results
1	사용자가 특정 채널을 클릭한다.	DB로부터 해당 채널의 ID값으로 채널 정보를 조회해 상세 정보를 가져온다.
2	DB로부터 특정 채널의 모든 스케줄의 정보를 모두 가져온다.	스케줄 정보에는 멤버와 채널 ID, 유저와 스케줄 이름, 시작 및 종료 시간 등의 정보 등이 있다. 이 정보들을 DB 서버로부터 가져온다.
3	사용자에게 DB로부터 가져온 정보를 보여준다.	사용자에게 해당 채널 페이지를 보여주고, 테이블 형식으로 해당 채널의 현재 스케줄을 보여준다.
4	일정시간(최소 30초)동안 아무 동작 없이 대기한다.	애플리케이션이 서버의 타임테이블 정보를 조회하여 자동으로 최신의 상태로 갱신하여 보여준다.

5.4.10. T_10 Test: 단일 스케줄 상세 조회

- T_10_1 Test: 사용자가 단일 스케줄에 대해 상세 정보를 조회한다.

Table 5-23: Test Case 10-1

TEST STEPS	
Test Case ID :	T_10_1
Test Case Name :	단일 스케줄 상세 조회
For Use Case :	사용자가 단일 스케줄에 대해 상세 정보를 조회한다.
Related Requirement IDs :	USER_RQ_007, SYS_RQ_FR_007
Test Type :	Positive
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 채널 페이지의 스케줄 테이블에서 특정 스케줄을 클릭한다.	애플리케이션은 사용자에게 해당 스케줄의 상세 정보를 모달 형태로 보여준다.
2	DB로부터 특정 단일 스케줄의 정보를 모두 가져온다.	스케줄 정보에는 멤버와 채널 ID, 유저와 스케줄 이름, 시작 및 종료 시간 등의 정보 등이 있다. 이 정보들을 DB 서버로부터 가져온다.
3	사용자에게 DB로부터 가져온 정보를 보여준다.	사용자에게 상세 정보 모달에서 해당 스케줄에 대한 상세 예약 정보를 보여준다.

5.4.11. T_11 Test: 스케줄 예약 요청

- T_11_1 Test: 사용자가 특정 시간에 대한 스케줄에 대하여 예약을 요청한다.
- T_11_2 Test: 사용자가 이미 예약이 존재하는 시간대에 대해 예약을 시도하는 경우, 예약

에 실패한다.

- T_11_3 Test: 사용자가 유효하지 않은 정보로 예약을 시도하는 경우, 예약에 실패한다.

Table 5-24: Test Case 11-1

TEST STEPS		
Test Case ID :	T_11_1	
Test Case Name :	스케줄 예약 요청	
For Use Case :	사용자가 특정 시간에 대한 스케줄에 대하여 예약을 요청한다.	
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, USER_RQ_009, USER_RQ_009_01, USER_RQ_013, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_FR_009, SYS_RQ_FR_009_01	
Test Type :	Positive	
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 특정 채널 페이지에서 스케줄 신청 버튼을 누른다.	애플리케이션은 사용자에게 스케줄 신청 페이지를 보여준다.
2	사용자가 유효한 스케줄 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 유효한 데이터를 전송받는다.	사용자가 유효한 정보를 입력했고, 해당 시간과 중복되는 다른 예약이 존재하는지를 DB에서 조회해 만약 없다면 해당 정보로 예약을 생성해 DB에 저장한다.
4	스케줄 예약에 성공한다.	애플리케이션은 사용자에게 예약이 성공했음을 알린다.
5	타임 테이블을 업데이트한다.	해당 채널의 타임 테이블을 업데이트해 새 예약을 표시한다.

Table 5-25: Test Case 11-2

TEST STEPS	
Test Case ID :	T_11_2
Test Case Name :	스케줄 예약 요청
For Use Case :	사용자가 이미 예약이 존재하는 시간대에 대해 예약을 시도하는 경우, 예약에 실패한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, USER_RQ_009, USER_RQ_009_01, USER_RQ_013, SYS_RQ_FR_006,

	SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_FR_009, SYS_RQ_FR_009_01
Test Type :	Negative
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 다른 예약이 존재해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 특정 채널 페이지에서 스케줄 신청 버튼을 누른다.	애플리케이션은 사용자에게 스케줄 신청 페이지를 보여준다.
2	사용자가 유효한 스케줄 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 이미 예약이 존재하는 시간대가 포함된 데이터를 전송받는다.	DB는 해당 시간과 중복되는 예약 정보가 이미 존재함을 확인 후 예외를 발생시킨다.
4	스케줄 예약에 실패한다.	애플리케이션은 사용자에게 이미 해당 시간대에 예약이 존재함을 알린다.

Table 5-26: Test Case 11-3

TEST STEPS	
Test Case ID :	T_11_3
Test Case Name :	스케줄 예약 요청
For Use Case :	사용자가 유효하지 않은 정보로 예약을 시도하는 경우, 예약에 실패한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, USER_RQ_009, USER_RQ_009_01, USER_RQ_013, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_FR_009, SYS_RQ_FR_009_01
Test Type :	Negative
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 특정 채널 페이지에서 스케줄 신청 버튼을 누른다.	애플리케이션은 사용자에게 스케줄 신청 페이지를 보여준다.
2	사용자가 유효하지 않은 스케줄 정보를 입력한다.	해당 정보를 DTO로 백엔드 서버로 전송한다.
3	백엔드 서버에서 유효하지 않	사용자가 유효하지 않은 값을 입력한 경우, 서버는 예

	은 스케줄 데이터를 전송받는다.	외를 발생시킨다.
4	스케줄 예약에 실패한다.	애플리케이션은 사용자에게 유효하지 않은 값을 입력했음을 알린다.

5.4.12. T_12 Test: 스케줄 삭제

- T_12_1 Test: 타임 테이블에 등록된 스케줄이 삭제된다.
- T_12_2 Test: 사용자가 해당 계정에서 생성하지 않은 스케줄의 삭제를 시도하는 경우 삭제에 실패한다.

Table 5-27: Test Case 12-1

TEST STEPS	
Test Case ID :	T_12_1
Test Case Name :	스케줄 삭제
For Use Case :	타임 테이블에 등록된 스케줄이 삭제된다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, USER_RQ_009, USER_RQ_009_01, USER_RQ_013, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_FR_009
Test Type :	Positive
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 삭제를 시도하는 유저가 신청해 승인된 스케줄이 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 채널 페이지의 스케줄 테이블에서 특정 스케줄을 클릭한다.	애플리케이션은 사용자에게 해당 스케줄의 상세 정보를 모달 형태로 보여준다.
2	사용자가 스케줄 삭제 버튼을 누른다.	백엔드 서버로 스케줄 삭제 요청을 전송한다.
3	백엔드 서버에서 데이터를 전송받는다.	서버는 해당 예약을 요청한 사용자의 ID와 스케줄을 등록한 사람의 ID를 비교해 일치하는지를 확인한 후, 스케줄을 삭제한다.
4	스케줄 삭제에 성공한다.	애플리케이션은 사용자에게 스케줄이 삭제되었음을 알린다.
5	타임 테이블을 업데이트한다.	해당 채널의 타임 테이블을 업데이트한다.

Table 5-28: Test Case 12-2

TEST STEPS	
Test Case ID :	T_12_2
Test Case Name :	스케줄 삭제
For Use Case :	사용자가 해당 계정에서 생성하지 않은 스케줄의 삭제를 시도하는 경우 삭제에 실패한다.
Related Requirement IDs :	USER_RQ_006, USER_RQ_006_01, USER_RQ_007, USER_RQ_008, USER_RQ_009, USER_RQ_009_01, USER_RQ_013, SYS_RQ_FR_006, SYS_RQ_FR_007, SYS_RQ_FR_008, SYS_RQ_FR_009
Test Type :	Negative
Precondition :	사용자는 시스템에 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 승인된 스케줄이 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	사용자가 채널 페이지의 스케줄 테이블에서 특정 스케줄을 클릭한다.	애플리케이션은 사용자에게 해당 스케줄의 상세 정보를 모달 형태로 보여준다.
2	사용자가 스케줄 삭제 버튼을 누른다.	백엔드 서버로 스케줄 삭제 요청을 전송한다.
3	백엔드 서버에서 데이터를 전송받는다.	서버는 해당 예약을 요청한 사용자의 ID와 스케줄을 등록한 사람의 ID를 비교해 일치하는지를 확인한 후, 일치하지 않으면 예외를 발생시킨다.
4	스케줄 삭제에 실패한다.	애플리케이션은 사용자에게 스케줄을 삭제할 권한이 없음을 알린다.
5	타임 테이블을 업데이트한다.	해당 채널의 타임 테이블을 업데이트한다.

5.4.13. T_13 Test: 스케줄 신청 목록 조회

- T_13_1 Test: 관리자 권한을 가진 사용자가 채널들에 대한 스케줄 요청 목록을 조회한다.

Table 5-29: Test Case 13-1

TEST STEPS	
Test Case ID :	T_13_1
Test Case Name :	스케줄 신청 목록 조회
For Use Case :	관리자 권한을 가진 사용자가 채널들에 대한 스케줄 요청 목록을 조회한다.

Related Requirement IDs :	USER_RQ_001_02, USER_RQ_010, SYS_RQ_FR_005, SYS_RQ_FR_010
Test Type :	Positive
Precondition :	사용자는 시스템에 관리자 권한으로 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 대한 스케줄 등록 요청이 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 스케줄 등록 요청 페이지로 이동한다.	관리자에게 스케줄 요청 현황 페이지를 보여준다.
2	DB로부터 모든 채널의 스케줄 등록 요청 정보를 가져온다.	DB는 현재 승인을 대기 중인 전체 스케줄 리스트를 반환한다.
3	DB로부터 가져온 정보를 보여준다.	관리자에게 스케줄 요청 현황 페이지에서 승인 대기 중인 스케줄 목록을 보여준다.

5.4.14. T_14 Test: 스케줄 예약 신청 승인

- T_14_1 Test: 관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 승인한다.
- T_14_2 Test: 관리자 권한을 가진 사용자가 이미 예약이 존재하는 시간대에 대한 예약을 승인하려고 하는 경우, 승인에 실패한다.

Table 5-30: Test Case 14-1

TEST STEPS	
Test Case ID :	T_14_1
Test Case Name :	스케줄 예약 신청 승인
For Use Case :	관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 승인한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_011, USER_RQ_012, SYS_RQ_FR_005, SYS_RQ_FR_009_01, SYS_RQ_FR_010, SYS_RQ_FR_011, SYS_RQ_FR_012
Test Type :	Positive
Precondition :	사용자는 시스템에 관리자 권한으로 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 대한 스케줄 등록 요청이 있어야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 스케줄 요청 현황 페이지에서 특정 스케줄의 스케줄 승인 버튼을 누른다.	백엔드 서버로 스케줄 승인 요청을 전송한다

2	백엔드 서버에서 데이터를 전송받는다.	서버는 해당 예약과 시간이 중복되는 다른 예약이 DB에 존재하는지 조회한 후, 존재하지 않는다면 해당 예약 상태를 승인으로 변경한다.
3	스케줄 승인에 성공한다.	애플리케이션은 관리자에게 스케줄이 승인되었음을 알린다.
4	타임 테이블을 업데이트한다.	해당 채널의 타임 테이블을 업데이트한다.
5	스케줄을 예약한 사용자에게 알림을 발송한다.	시스템은 해당 예약을 신청한 사용자에게 스케줄의 승인/거절이 완료되었음을 알리는 알림을 발송한다.

Table 5-31: Test Case 14-2

TEST STEPS	
Test Case ID :	T_14_2
Test Case Name :	스케줄 예약 신청 승인
For Use Case :	관리자 권한을 가진 사용자가 이미 예약이 존재하는 시간대에 대한 예약을 승인하려고 하는 경우, 승인에 실패한다.
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_011, USER_RQ_012, SYS_RQ_FR_005, SYS_RQ_FR_009_01, SYS_RQ_FR_010, SYS_RQ_FR_011, SYS_RQ_FR_012
Test Type :	Negative
Precondition :	사용자는 시스템에 관리자 권한으로 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 대한 스케줄 등록 요청이 있어야 한다. 또한, 해당 등록 요청 스케줄과 시간대가 중복되는 이미 승인된 스케줄이 존재해야 한다.

TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 스케줄 요청 현황 페이지에서 특정 스케줄의 스케줄 승인 버튼을 누른다.	백엔드 서버로 스케줄 승인 요청을 전송한다
2	백엔드 서버에서 기존 스케줄과 시간대가 중복되는 승인 요청 데이터를 전송받는다.	서버는 해당 예약과 시간이 중복되는 다른 예약이 DB에 존재하는지 조회한 후, 존재한다면 예외를 발생시킨다.
3	스케줄 승인에 실패한다.	애플리케이션은 관리자에게 이미 승인된 스케줄 중 해당 예약 요청과 시간대가 중복되는 스케줄이 이미 있음을 알린다.

5.4.15. T_15 Test: 스케줄 예약 신청 거절

- T_15_1 Test: 관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 거절한다.

Table 5-32: Test Case 15-1

TEST STEPS		
Test Case ID :	T_15_1	
Test Case Name :	스케줄 예약 신청 거절	
For Use Case :	관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 거절한다.	
Related Requirement IDs :	USER_RQ_001, USER_RQ_001_01, USER_RQ_001_02, USER_RQ_011, USER_RQ_012, SYS_RQ_FR_005, SYS_RQ_FR_010, SYS_RQ_FR_011, SYS_RQ_FR_012	
Test Type :	Positive	
Precondition :	사용자는 시스템에 관리자 권한으로 로그인해야 하며, 생성된 채널이 존재하고 해당 채널에 대한 스케줄 등록 요청이 있어야 한다.	
TEST STEPS		
#	**Steps**	**Expected Results**
1	관리자가 스케줄 요청 현황 페이지에서 특정 스케줄의 스케줄 거절 버튼을 누른다.	백엔드 서버로 스케줄 거절 요청을 전송한다.
2	백엔드 서버에서 데이터를 전송받는다.	서버는 해당 예약 상태를 거절 변경한다.
3	스케줄 거절에 성공한다.	애플리케이션은 관리자에게 스케줄이 거절되었음을 알린다.
4	스케줄을 예약한 사용자에게 알림을 발송한다.	시스템은 해당 예약을 신청한 사용자에게 스케줄의 승인/거절이 완료되었음을 알리는 알림을 발송한다.

5.5. Testing Results

Table 5-33 Testing Results

Test Case Number	Test Case Description	Test Case Test Result	Comments
T_01_1	회원 가입(Positive)	Success	
T_01_2	회원 가입(Negative)	Success	
T_01_3	회원 가입(Negative)	Success	
T_02_1	로그인(Positive)	Success	
T_02_2	로그인(Negative)	Success	

T_02_3	로그인(Negative)	Success	
T_02_4	관리자 로그인(Positive)	Success	
T_03_1	채널 생성(Positive)	Success	
T_03_2	채널 생성(Negative)	Success	
T_04_1	채널 관리자 페이지 접속 (Positive)	Success	
T_05_1	채널 목록 조회(Positive)	Success	
T_06_1	특정 채널 조회(Positive)	Success	
T_07_1	특정 채널 수정(Positive)	Success	
T_07_2	특정 채널 수정(Negative)	Success	
T_07_3	특정 채널 수정(Negative)	Success	
T_08_1	특정 채널 삭제(Positive)	Success	
T_08_2	특정 채널 삭제(Negative)	Success	
T_09_1	채널 타임 테이블 조회 (Positive)	Success	
T_10_1	단일 스케줄 상세 조회 (Positive)	Success	
T_11_1	스케줄 예약 요청(Positive)	Success	
T_11_2	스케줄 예약 요청(Negative)	Success	
T_11_3	스케줄 예약 요청(Negative)	Success	
T_12_1	스케줄 삭제(Positive)	Success	
T_12_2	스케줄 삭제(Negative)	Success	
T_13_1	스케줄 신청 목록 조회 (Positive)	Success	
T_14_1	스케줄 예약 신청 승인 (Positive)	Failed	예약 승인/거절 시 사용자 알림 기능이 구현되지 않음.
T_14_2	스케줄 예약 신청 승인 (Negative)	Success	
T_15_1	스케줄 예약 신청 거절 (Positive)	Failed	예약 승인/거절 시 사용자 알림 기능이 구현되지 않음.

Table 5-34 Testing Results Summary

Total Tests	28	
Tests Passed	26	93%
Tests Failed	2	%
Test Coverage	28/28	100%

Tests Quality	28/28	100%

5.6. Failed Test Analysis

Table 5-35: Failed Test Case - 14-1

Test Case No.	T_14_1
For Use Case	관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 승인한다.
Analysis	USER_RQ_012, SYS_RQ_FR_012에 대한 구현이 성공적으로 이루어지지 않았다. 이에 따라 사용자의 스케줄 예약 신청에 대한 승인 및 거절이 이루어지더라도 알림이 발송되지 않는 상황이다. 따라서, 해당 테스트는 최종적으로 실패하였다. 해당 Case를 구현하기 위해서는 Google의 Firebase Cloud Messaging (FCM) 서비스를 이용해야 하며, iOS의 경우 추가적으로 Apple Push Notification service (APNs) 인증서를 발급받아 FCM에 등록하는 절차가 필요하다. APNs 인증서를 위해서 Apple Developer Program을 이용하였지만 이를 실제 적용하고 테스팅하기에는 시간적인 제약이 존재하였다. 또, 각 클라이언트의 디바이스 토큰을 받아 서버에 저장해두는 등의 추가적인 DB구조 개선이 요구된다. 향후 해당 Requirements에 대한 구현이 완료된다면 해당 테스트가 성공할 것으로 기대된다.

Table 5-36: Failed Test Case - 15-1

Test Case No.	T_15_1
For Use Case	관리자 권한을 가진 사용자가 특정 스케줄에 대한 요청을 거절한다.
Analysis	본 테스트 역시 상기의 T_14_1의 이유와 동일한 이유로 테스트에 실패하였다.

5.7. QA Test

현재 모든 팀원들이 iOS 디바이스를 이용하고 있어, Apple에서 제공하는 테스팅 플랫폼인 TestFlight을 이용하여 테스팅을 진행하였다. Github에 푸쉬되는대로 파이프라인을 통해 자동으로 TestFlight에 배포되어 각 팀원들의 앱이 최신버전으로 유지되도록 하였다. 프론트엔드에서 새로운 기능을 추가할 때마다 즉각적으로 팀원들에게 배포하여 여러가지 디바이스에서 바로 테스트를 진행할 수 있어 유용하였다. 12/7 현재시각으로 1.0.0(Build 24)를 이용하여 테스팅을 진행하고 있으며, 해당 플랫폼을 통해 다음과 같이 여러가지 버그들을 사전에 찾아낼 수 있었다.

1. 로그인이 유지되지 않고 해제되는 버그
2. 예약 시작시간과 종료시간이 정확히 일치할 경우 예약이 되지 않는 버그
3. 서로 다른 채널에 대해 스케줄이 겹쳐 예약이 진행되지 않는 버그
4. 승인되지 않은 스케줄에 대해 시간검증이 진행된 버그
5. 시작시간이 종료시간보다 이후인데도 정상적으로 예약요청이 이루어지는 버그

6. 회원가입시 대문자 id를 입력받을 경우 에러를 출력하여 회원가입이 진행되지 않는 버그

버전 1.0.0

빌드	진행 상태	그룹	설치	충돌
23 내부	테스트 중 90일 후 무효화	WE	2	–
22 내부	테스트 중 90일 후 무효화	WE	–	–
21 내부	테스트 중 89일 후 무효화	WE	3	–
20 내부	테스트 중 89일 후 무효화	WE	–	–
19 내부	테스트 준비 완료 89일 후 무효화		–	–
18 내부	테스트 중 88일 후 무효화	WE	5	–
17 내부	테스트 중 81일 후 무효화	WE	5	–
16 내부	테스트 중 81일 후 무효화	WE	1	–
13 내부	테스트 중 81일 후 무효화	WE	5	–
11 내부	테스트 중 78일 후 무효화	WE	5	–
9 내부	테스트 중 77일 후 무효화	WE	5	–

Figure 5-1 TestFlight 배포 빌드

테스터(4)

필터 추가

테스터	진행 상태	그룹	세션	충돌	피드백	기기
insoyafear@gmail.com KimDongryeong · 내부	설치됨 1.0.0 (23) 2023년 12월 6일	WE	17			iPhone 13 Pro iOS 17.1.1
leesj2222@naver.com 이승주 · 내부	설치됨 1.0.0 (23) 2023년 12월 6일	WE	11			iPhone 14 iOS 17.1.2
yhwjd@naver.com 정유환 · 내부	설치됨 1.0.0 (23) 2023년 12월 6일	WE	8			iPhone 8 iOS 15.1
parkjb825@icloud.com 박종범 · 내부	설치됨 1.0.0 (23) 2023년 12월 6일	WE	134			iPhone 12 Pro Max 및 다른 기기 iOS 17.2

Figure 5-2 TestFlight 참여 테스터

6. Requirements Satisfaction

본 항목은 각각의 사용자/기능/품질 요구사항에 대하여 이를 검증하는 데 연관되어 있는 단위

테스트, 통합 테스트에 대해 정리하였다.

6.1. User Requirements

Table 6-1: User Requirement 001

No.	USER_RQ_001
Title	시스템을 이용하는 유저는 관리자와 가입 유저로 구분되어야 한다.
Verificable Unit Test	MEM_UT_01, MEM_UT_02, MEM_UT_03, M_UT_01, M_UT_02, M_UT_03
Verificable Integrated Test	T_01_1, T_01_2, T_01_3

Table 6-2: User Requirement 001-1

No.	USER_RQ_001_01
Title	관리자 유저는 가입 유저의 모든 기능 및 동작을 제공하여야 한다.
Verificable Unit Test	CH_UT_03, CH_UT_04, CH_UT_05, S_UT_01, S_UT_02, S_UT_03, S_UT_04, S_UT_05, S_UT_10, S_UT_11, S_UT_12, S_UT_14, MO_UT_07, MO_UT_08, MO_UT_10, MO_UT_11
Verificable Integrated Test	T_05_1, T_06_1, T_09_1, T_10_1, T_11_1, T_11_2, T_11_3, T_12_1, T_12_2

Table 6-3: User Requirement 001-2

No.	USER_RQ_001_02
Title	관리자 유저는 가입 유저가 접근할 수 없는 채널 관리 페이지에 대한 접근 권한을 갖는다.
Verificable Unit Test	CH_UT_01, CH_UT_02, CH_UT_06, CH_UT_07, CH_UT_09, CH_UT_10, CH_UT_11, S_UT_06, S_UT_07, S_UT_08, S_UT_09, S_UT_07, MO_UT_06, MO_UT_12, MO_UT_13, MO_UT_14, MO_UT_15
Verificable Integrated Test	T_04_1

Table 6-4: User Requirement 002

No.	USER_RQ_002
Title	시스템은 서비스를 이용하는 모든 유저에 대하여 유효한 인증 절차를 마련하여야 한다.
Verificable Unit Test	MEM_UT_01, MEM_UT_02, MEM_UT_03, MEM_UT_04, MEM_UT_05, MEM_UT_06, JWT_UT_01, JWT_UT_02, JWT_UT_03, JWT_UT_04, JWT_UT_05,

	JWT_UT_06, JWT_UT_07, JWT_UT_08, JWT_UT_09, MO_UT_01, MO_UT_02, MO_UT_03, MO_UT_04, MO_UT_05
Verificable Integrated Test	T_01_1, T_01_2, T_01_3, T_02_1, T_02_2

Table 6-5: User Requirement 003

No.	USER_RQ_003
Title	유저는 자동 로그인을 통해 간편하고 빠르게 시스템에 접속할 수 있어야 한다.
Verificable Unit Test	MEM_UT_05, MEM_UT_06, JWT_UT_01, JWT_UT_02, JWT_UT_03, JWT_UT_04, JWT_UT_05, JWT_UT_06, JWT_UT_07, JWT_UT_08, JWT_UT_09, MO_UT_04, MO_UT_05
Verificable Integrated Test	T_02_1, T_02_2

Table 6-6: User Requirement 004

No.	USER_RQ_004
Title	유저의 인증 정보가 탈취되더라도 탈취자가 해당 정보로 인증할 수 없도록 해야 한다.
Verificable Unit Test	MEM_UT_05, MEM_UT_06, JWT_UT_01, JWT_UT_02, JWT_UT_03, JWT_UT_04, JWT_UT_05, JWT_UT_06, JWT_UT_07, JWT_UT_08, JWT_UT_09, MO_UT_04, MO_UT_05
Verificable Integrated Test	T_02_1, T_02_2

Table 6-7: User Requirement 005

No.	USER_RQ_005
Title	관리자 유저는 동아리 채널을 생성, 수정, 삭제할 수 있어야 한다.
Verificable Unit Test	MEM_UT_01, CH_UT_01, CH_UT_02, CH_UT_06, CH_UT_07, CH_UT_08, CH_UT_09, CH_UT_10, CH_UT_11, CH_UT_12, MO_UT_03, MO_UT_06
Verificable Integrated Test	T_04_1, T_07_1, T_07_2, T_07_3, T_08_1, T_08_2

Table 6-8: User Requirement 006

No.	USER_RQ_006
Title	유저는 채널의 스케줄 예약 현황을 타임 테이블 형태로 조회할 수 있어야 한다.

Verificable Unit Test	S_UT_05, MO_UT_16
Verificable Integrated Test	T_09_1

Table 6-9: User Requirement 006-1

No.	USER_RQ_006_01
Title	유저는 타임 테이블 상에서 각 스케줄을 쉽게 구분하고 이해할 수 있어야 한다.
Verificable Unit Test	S_UT_05, MO_UT_09, MO_UT_16, MO_UT_17
Verificable Integrated Test	T_09_1

Table 6-10: User Requirement 007

No.	USER_RQ_007
Title	유저는 타임 테이블에서 각 스케줄의 상세한 정보를 추가적으로 확인할 수 있어야 한다.
Verificable Unit Test	S_UT_04, MO_UT_0
Verificable Integrated Test	T_09_1, T_10_1

Table 6-11: User Requirement 008

No.	USER_RQ_008
Title	유저는 타임 테이블에서 시스템 접속 시점부터 한 달 까지의 예약 현황을 조회할 수 있어야 한다.
Verificable Unit Test	S_UT_05, S_UT_06, S_UT_07
Verificable Integrated Test	T_09_1

Table 6-12: User Requirement 009

No.	USER_RQ_009
Title	유저는 채널에서 이용 가능한 시간 중 원하는 시간을 선택하여 스케줄을 예약할 수 있어야 한다.

Verificable Unit Test	S_UT_01, S_UT_02, S_UT_03, MO_UT_07
Verificable Integrated Test	T_11_1, T_11_2, T_11_3

Table 6-13: User Requirement 009-1

No.	USER_RQ_009_01
Title	유저가 이미 존재하는 스케줄 블록과 시간이 겹치는 스케줄 예약을 신청할 수 없어야 한다.
Verificable Unit Test	S_UT_01, S_UT_02, S_UT_03, MO_UT_07
Verificable Integrated Test	T_11_1, T_11_2, T_11_3

Table 6-14: User Requirement 010

No.	USER_RQ_010
Title	관리자는 관리하고 있는 채널의 예약 현황을 별도로 확인할 수 있어야 한다.
Verificable Unit Test	MEM_UT_01, S_UT_07, MO_UT_03, MO_UT_12, MO_UT_13
Verificable Integrated Test	T_13_1

Table 6-15: User Requirement 011

No.	USER_RQ_011
Title	관리자는 가입 유저의 예약을 승인 혹은 거절할 수 있어야 한다.
Verificable Unit Test	MEM_UT_01, S_UT_07, S_UT_08, S_UT_09, MO_UT_03, MO_UT_12, MO_UT_13, MO_UT_14, MO_UT_15
Verificable Integrated Test	T_14_1, T_14_2, T_15_1

Table 6-16: User Requirement 012

No.	USER_RQ_012
Title	가입 유저의 예약 신청이 승인 혹은 거절될 경우, 가입 유저는 이를 앱을 통해서 알림을 받아야 한다.
Verificable Unit	

Test	
Verificable Integrated Test	

Table 6-17: User Requirement 013

No.	USER_RQ_013
Title	유저는 타임 테이블에서 최신의 예약 정보를 확인할 수 있어야 한다.
Verificable Unit Test	MO_UT_16, MO_UT_17
Verificable Integrated Test	T_09_1

6.2. System Requirements

6.2.1. Functional Requirement

Table 6-18: System Requirements - Functional Requirement 001

No.	SYS_RQ_FR_001	**Related Requirement**	USER_RQ_001 USER_RQ_002
Title	시스템은 회원가입 시 관리자 권한을 인증하기 위해 관리자 인증번호를 요청하고 이를 확인해야 한다.		
Verificable Unit Test	CH_UT_03, CH_UT_04, CH_UT_05, S_UT_01, S_UT_02, S_UT_03, S_UT_04, S_UT_05, S_UT_10, S_UT_11, S_UT_12, S_UT_14, MO_UT_07, MO_UT_08, MO_UT_10, MO_UT_11		
Verificable Integrated Test	T_05_1, T_06_1, T_09_1, T_10_1, T_11_1, T_11_2, T_11_3, T_12_1, T_12_2		

Table 6-19: System Requirements - Functional Requirement 002

No.	SYS_RQ_FR_002	**Related Requirement**	USER_RQ_002
Title	시스템은 유저가 아이디, 비밀번호를 통해 로그인을 진행하도록 하며, 성공적으로 인증이 되어야 로그인이 되도록 한다.		
Verificable Unit Test	MEM_UT_05, MEM_UT_06, JWT_UT_01, JWT_UT_02, JWT_UT_03, JWT_UT_04, JWT_UT_05, JWT_UT_06, JWT_UT_07, JWT_UT_08, JWT_UT_09, MO_UT_04, MO_UT_05		
Verificable Integrated Test	T_02_1, T_02_2, T_02_3		

Table 6-20: System Requirements - Functional Requirement 003

No.	SYS_RQ_FR_003	**Related Requirement**	USER_RQ_003
Title	시스템은 토큰을 기반으로 하여 이전에 수행된 로그인에 대해서는 최소 1달 간 유저의 로그인 상태를 유지해야 한다		
Verificable Unit Test	MEM_UT_05, MEM_UT_06, JWT_UT_01, JWT_UT_02, JWT_UT_03, JWT_UT_04, JWT_UT_05, JWT_UT_06, JWT_UT_07, JWT_UT_08, JWT_UT_09, MO_UT_04, MO_UT_05		
Verificable Integrated Test			
Verificable Test	QA Test를 진행하며 이전에 수행된 로그인에 대해서 최소 1달간 로그인 상태를 유지하는지 지속적으로 테스트하였다. 초반에 로그인이 해제된 경험이 있어, 해당 시간에 확인된 서버 로그를 분석 후 디버깅하여 현재는 안정적으로 로그인 상태를 유지하고 있다.		

Table 6-21: System Requirements - Functional Requirement 004

No.	SYS_RQ_FR_004	**Related Requirement**	USER_RQ_004
Title	시스템은 보안을 위해 비밀번호를 해싱 알고리즘을 통해 변환한 다음 저장하여야 한다.		
Verificable Unit Test	MEM_UT_05, MEM_UT_06		
Verificable Integrated Test			

Table 6-22: System Requirements - Functional Requirement 005

No.	SYS_RQ_FR_005	**Related Requirement**	USER_RQ_005
Title	시스템은 채널을 생성, 수정, 삭제할 수 있는 관리자 페이지의 접근 권한 및 UI를 관리자 유저에게만 제공하여야 한다.		
Verificable Unit Test	MEM_UT_01, MO_UT_03, MO_UT_12		
Verificable Integrated Test	T_02_4		

Table 6-23: System Requirements - Functional Requirement 006

No.	SYS_RQ_FR_006	**Related Requirement**	USER_RQ_006

Title	시스템은 모든 종류의 유저에게 채널의 스케줄 예약 현황을 타임 테이블 형태로 제공하여야 한다.
Verificable Unit Test	S_UT_05, MO_UT_06, MO_UT_16
Verificable Integrated Test	T_09_1

Table 6-24: System Requirements - Functional Requirement 007

No.	SYS_RQ_FR_007	**Related Requirement**	USER_RQ_007
Title	시스템은 채널의 타임 테이블에서, 각 스케줄 블록의 스케줄 상세 정보를 확인할 수 있도록 하는 기능을 별도로 제공한다.		
Verificable Unit Test	S_UT_04, MO_UT_10		
Verificable Integrated Test	T_10_1		

Table 6-25: System Requirements - Functional Requirement 008

No.	SYS_RQ_FR_008	**Related Requirement**	USER_RQ_008
Title	시스템은 접속 시점부터 최소 한달까지의 스케줄 정보를 채널의 타임 테이블에 출력해야 한다.		
Verificable Unit Test	S_UT_05, MO_UT_06, MO_UT_07		
Verificable Integrated Test	T_09_1		

Table 6-26: System Requirements - Functional Requirement 009

No.	SYS_RQ_FR_009	**Related Requirement**	USER_RQ_009
Title	시스템은 사용자가 스케줄에 대한 예약을 신청할 수 있는 별도의 UI를 제공하여야 한다.		
Verificable Unit Test			
Verificable Integrated Test	T_11_1, T_11_2, T_11_3		

Table 6-27: System Requirements - Functional Requirement 009-1

No.	SYS_RQ_FR_009_01	**Related Requirement**	USER_RQ_009_01
Title	시스템은 이미 예약이 이루어진 스케줄과 중복되는 새로운 스케줄을 가입 유저가 예약하는 것과, 그러한 새로운 스케줄 예약을 관리자가 승인하는 것이 발생하지 않도록 하여 스케줄 구성 시 충돌이 일어나지 않도록 한다.		
Verificable Unit Test	S_UT_01, S_UT_08, S_UT_14, MO_UT_07, MO_UT_14		
Verificable Integrated Test	T_11_2, T_14_2		

Table 6-28: System Requirements - Functional Requirement 010

No.	SYS_RQ_FR_010	**Related Requirement**	USER_RS_010
Title	시스템은 관리자가 해당 채널의 예약 신청 현황을 조회할 수 있는 UI를 별도로 제공하여야 한다.		
Verificable Unit Test	MO_UT_12, MO_UT_13		
Verificable Integrated Test	T_13_1		

Table 6-29: System Requirements - Functional Requirement 011

No.	SYS_RQ_FR_011	**Related Requirement**	USER_RS_011
Title	시스템은 관리자가 스케줄 예약 요청에 대한 승인 혹은 거절할 수 있는 별도의 UI를 제공하여야 한다.		
Verificable Unit Test	MO_UT_14, MO_UT_15		
Verificable Integrated Test	T_14_1, T_14_2, T_15_1		

Table 6-30: System Requirements - Functional Requirement 012

No.	SYS_RQ_FR_012	**Related Requirement**	USER_RQ_012
Title	시스템은 관리자의 승인/거절 결과를 해당 예약을 신청한 가입 유저에게 알림을 발송하여 알린다		
Verificable Unit			

Test	
Verificable Integrated Test	

6.2.2. Non-Functional Requirement

Table 6-31: System Requirements – Non-Functional Requirement 001

No.	SYS_RQ_NFR_001	**Related Requirement**	
Title	시스템은 사용자 친화적이고 자연스러운 흐름의 UX를 제공해야 한다.		
Verificable Unit Test			
Verificable Integrated Test			
Verificable Test	QA Test 및 Target User 시연을 통해서 이용하기 편한지, 흐름은 자연스러운지 테스팅하였으며, 동아리 임원에게 이를 시연해본 결과 매우 유용하다는 평을 받았다.		

Table 6-32: System Requirements – Non-Functional Requirement 002

No.	SYS_RQ_NFR_002	**Related Requirement**	USER_RQ_006 USER_RQ_008
Title	시스템은 한 채널의 스케줄을 주 단위의 타임 테이블로 표시하여 제공하여야 한다.		
Verificable Unit Test	MO_UT_07		
Verificable Integrated Test	T_09_1		

Table 6-33: System Requirements – Non-Functional Requirement 003

No.	SYS_RQ_NFR_003	**Related Requirement**	USER_RQ_006 USER_RQ_006_01 USER_RQ_008
Title	시스템은 한 채널의 테이블 내에서 각 스케줄을 각기 다른 색상으로 표기하여야 한다.		
Verificable Unit Test	MO_UT_07, MO_UT_09		
Verificable Integrated Test	T_09_1		

Table 6-34: System Requirements – Non-Functional Requirement 004

No.	SYS_RQ_NFR_004	**Related Requirement**	USER_RQ_006 USER_RQ_007
Title	시스템은 사용자가 특정 스케줄 블록을 선택 시 제공하는 스케줄 상세 정보를 모달 형태로 출력하여야 한다.		
Verificable Unit Test	MO_UT_08		
Verificable Integrated Test	T_09_1		

Table 6-35: System Requirements – Non-Functional Requirement 005

No.	SYS_RQ_NFR_005	**Related Requirement**	USER_RQ_013
Title	시스템은 정해진 시간(30초)이 지나면 자동으로 타임 테이블 업데이트를 제공한다.		
Verificable Unit Test	MO_UT_16		
Verificable Integrated Test	T_09_2		

Table 6-36: System Requirements – Non-Functional Requirement 006

No.	SYS_RQ_NFR_006	**Related Requirement**	
Title	시스템은 사용자의 입력에 1초 이내로 반응하여야 한다.		
Verificable Unit Test	MO_UT_17		
Verificable Integrated Test			
Verificable Test	팀원들과 QA Test를 진행하며 총 170번의 세션을 통해 앱을 이용하였으며, 1초 이상의 딜레이가 발생한 입력은 보고되지 않았다.		

Table 6-37: System Requirements – Non-Functional Requirement 007

No.	SYS_RQ_NFR_007	**Related Requirement**	
Title	시스템은 데이터 요청에 대하여 3초 이내에 응답하여야 한다.		
Verificable Unit	S_UT_15		

Test	
Verificable Integrated Test	
Verificable Test	팀원들과 QA Test를 진행하며 총 약 50개의 스케줄을 계속하여 요청하고 수락하는 테스트를 진행하였으며, 3초 이상의 딜레이가 발생한 요청은 존재하지 않았다.

7. Implementation Result

7.1. Prototype

Figure 7-1

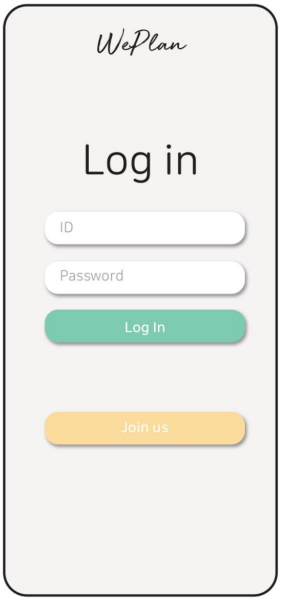


Figure 7-2

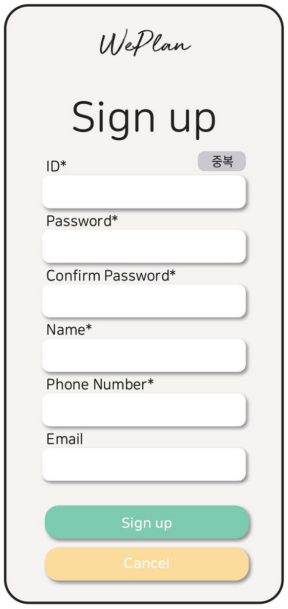


Figure 7-3

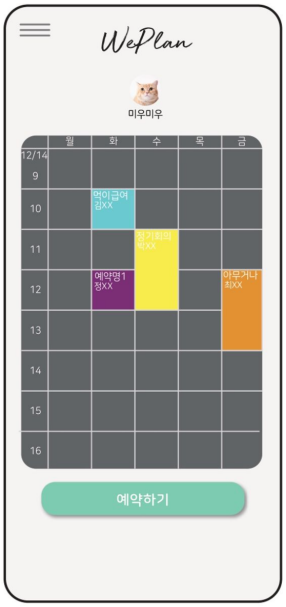


Figure 7-4

Figure 7-5

7.2. UI(Screenshots)

Figure 7-6 스케줄 타임테이블

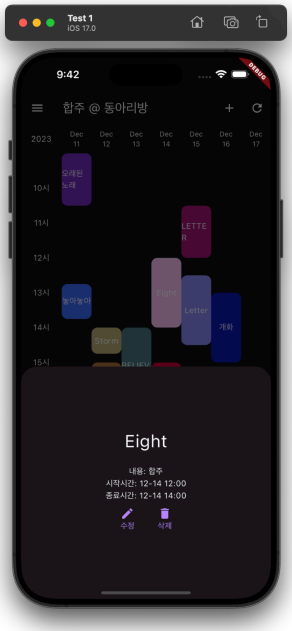


Figure 7-7 스케줄 상세정보

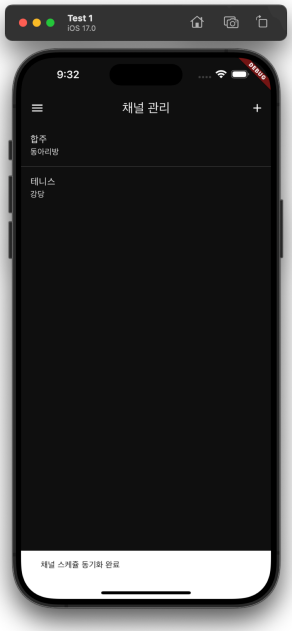


Figure 7-8 채널목록 (관리자)

Figure 7-9 채널 상세정보 (관리자)

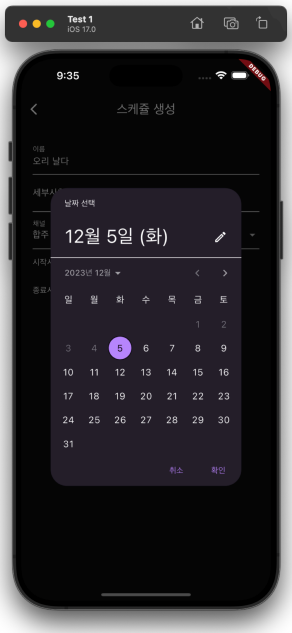


Figure 7-10 스케줄 생성일자 선택

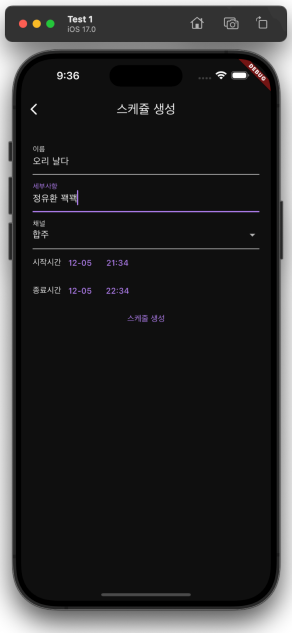


Figure 7-11 스케줄 생성

7.3. Swagger API Documents

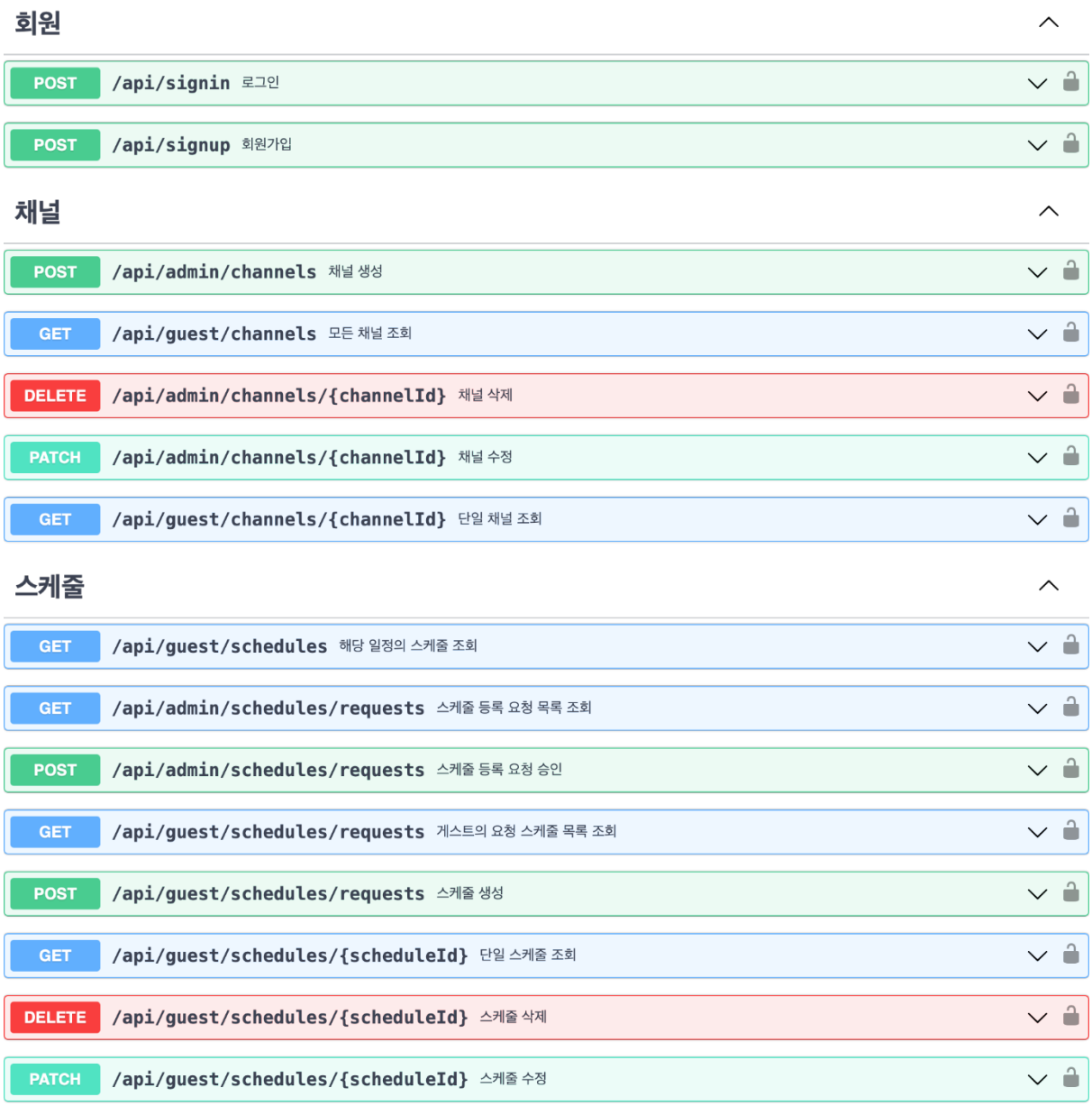


Figure 7-12 API

8. Project Status and Summary

8.1. Project Status

본 프로젝트는 일부 요구사항을 완전히 충족시키지 못하였으며, 일부 통합 테스트 역시 만족할 만한 결과를 얻지 못하였다. 따라서, 본 프로젝트는 성공적으로 완료되지 못했다.

앞서 5.6.장의 Failed Test Analysis에서 해당 테스트가 실패한 원인에 대해서 분석하였으며, 본 장에서는 만족시키지 못한 요구사항들에 대해서 분석하고, 해당 요구사항들을 만족시키기 위한 해결 방안에 대하여 모색하였다. 또한, 각 파트별로 전반적인 진행 상황에 대하여 평가하고, 각 팀원별로 프로젝트 진행에 있어 겪었던 challenge에 대해 기술하였다.

8.1.1. Unsatisfied Requirement

Table 8-1: Unsatisfied Requirement - User Requirement 012

No.	USER_RQ_012
Title	가입 유저의 예약 신청이 승인 혹은 거절될 경우, 가입 유저는 이를 앱을 통해서 알림을 받아야 한다.
Analysis	본래 해당 기능을 앱 푸쉬 알람 또는 카카오톡 메시지 발송 방식의 형태로 구현하고자 하였으나, 채널 및 스케줄 생성, 변경, 업데이트, 삭제 등의 핵심 로직을 구현하고 테스팅을 하는 과정에서 예상한 것보다 더 많은 시간을 소요해 해당 요구사항을 충족하는 데 필요한 기능을 구현하고 테스팅하지 못하였다.
Solution	해당 요구사항을 향후 개발 단계에서의 최우선 목표로 설정한다. 유저 편의성을 위하서라면 앱 내 푸쉬가 가장 적절할 것이고, 빠른 구현을 목표로 한다면 카카오톡 알림 발송 방식이 가장 적절할 것으로 보인다.

Table 8-2: Unsatisfied Requirement - System Requirement - Functional Requirement 012

No.	SYS_RQ_FR_012
Title	시스템은 관리자의 승인/거절 결과를 해당 예약을 신청한 가입 유저에게 알림을 발송하여 알린다.
Analysis	상기의 USER_RQ_012와 동일하다.
Solution	상기의 USER_RQ_012와 동일하다.

8.1.2. Additional Problems and Concerns

- 현재 아무 채널에나 유저가 스케줄 신청을 할 수 있도록 신청이 열려 있어, 해당 동아리에 가입되어 있지 않은 사용자도 해당 채널에 스케줄 등록 요청을 보낼 수 있다.

  Solution: 1차적으로는 해당 동아리의 관리자가 승인을 해야지만 스케줄 등록이 가능하므로, 동아리원이 아닌 사용자가 해당 채널에 스케줄 요청을 보내더라도 관리자가 거절할 수 있도록 되어 있다. 향후 개설된 채널에 대하여 채널별 멤버 등록 기능을 통해 등록되지 않은 멤버는 해당 채널에서 스케줄 등록 요청을 보낼 수 없도록 하는 것을 추가적인 요구사항으로 분류한 다음 구현하여 해결할 수 있다.

- 누군가 악의를 가지고 동시다발적으로 반복적으로 예약을 신청해 서버에 부하를 줄 가능성이 있다.

  Solution: 한 번 예약을 신청한 유저는 일정 시간이 지나야 추가적인 예약이 가능하도록 하는 것을 요구사항으로 추가해 해당 기능을 구현함으로써 해결할 수 있는 문제이다.

- 채널 관리자가 사전 예고 없이 채널을 삭제해 버리는 경우, 해당 채널을 이용 중인 사용

자들에게 피해가 갈 수 있다는 문제점이 있다.

Solution: 앞서 언급했던 채널별 멤버 등록 기능을 구현하게 된다면, 관리자가 채널 삭제를 시도하는 경우 기존에 이미 가입한 채널의 멤버들에게 삭제 여부에 동의하는지를 일정 기간 동안 질의하고, 질의에 답한 유저 중 과반이 이에 동의하는 경우 삭제가 되도록 하거나, 시스템 개발자 또는 운영자에게 직접 삭제 사유와 함께 요청을 보내면 최종 검토 후 삭제 여부를 결정하는 방식을 구현하여 해결할 수 있을 것으로 생각된다.

8.1.3. Overall Status

본 프로젝트 전반에 대하여 평가해 보았다. 전반적으로, 본 프로젝트는 각 대학생들의 보다 효율적인 동아리 일정 공지 및 확인에 있어서, 각 사용자가 채널을 생성 및 관리하며, 스케줄을 등록 및 수정, 삭제하고, 신청된 스케줄에 대해서 각 동아리장(관리자)이 이를 승인 또는 거절하는 핵심 기능에 대해서는 성공적으로 구현하였다. 그러나, 앞서 언급했듯 일부 만족시키지 못한 요구사항 및 실패한 테스트가 존재한다. 또한, 본 프로젝트의 구현 결과물을 실제 비즈니스에서 활용하기에는 미비한 점이 상당수 존재하는 것 역시 사실이다.

그렇지만, 구현한 핵심 요구사항들에 대해서는 프로그램이 의도했던 대로 작동하는 것을 확인하였으며, 따라서 본 프로젝트의 결과물은 향후 이어질 추가적인 요구사항을 만족시키는 과정에서 하나의 프로토타입으로서 기능할 수 있을 것이다. 향후 만족시키지 못한 요구사항들과 더불어 계속 추가되고 변경될 요구사항들을 반영하여 나간다면, 프로젝트 결과물의 활용도와 완성도를 한층 끌어올릴 수 있을 것이라 기대한다.

8.1.4. Each Part Status

- Channel: 관리자 계정을 사용해 채널을 등록하고, 채널 정보를 수정 및 삭제하는 기능은 완벽하게 구현되었다. 그러나, 모든 채널에 대해서 유저를 따로 구분하지 않아 모든 가입 유저가 모든 채널의 타임 테이블을 자유롭게 확인하고, 예약을 신청할 수 있는 상황이다. 또한, 관리자가 채널 삭제 시 채널 이용자의 의사와 관계없이 바로 삭제가 진행된다는 문제점이 존재한다. 8.1.2. Additional Problems and Concerns에서 언급했듯, 해당 기능을 향후 새로운 요구사항으로 설정해 구현할 필요가 있을 것이다.

- Member: 유저를 일반 가입 유저와 관리자 유저로 나누어, 관리자의 경우 관리자 코드를 가지고 있는 사람들만이 관리자 권한으로 회원가입을 진행할 수 있는 것을 확인하였다. 또한, 관리자 유저는 일반 유저가 사용가능한 모든 기능들을 정상적으로 이용 가능한 것을 확인하였으며, 관리자 전용 기능들 역시 기대한 대로 동작함을 확인하였다. 멤버 기능들에 대해서는 모두 의도한 대로 구현이 완료되었다.

- Schedule: 타임 테이블 형식으로 각 채널마다 일정이 표시되고, 일정이 비어있는 시간대에 원하는 예약을 시도하고, 관리자 계정이 해당 예약을 승인 및 반려할 수 있는 것을 확인하였다. 또한, 예약을 변경 및 삭제하는 기능 역시 정상적으로 작동한다. 그러나, 5.6. Failed Test Analysis 및 8.1.1. Unsatisfied Requirement에서 언급하였듯 예약 승인/거절 시

해당 예약의 신청자에게 알림을 전송하는 기능이 구현되지 않은 상태이다. 또한, 8.1.2. Additional Problems and Concerns에서 언급했듯 채널에 모든 멤버가 접근 가능한 문제 및 동시다발적으로 예약을 신청할 수 있다는 문제가 존재한다. 이 역시 해당 문제점을 해결하기 위한 기능들을 향후 새로운 요구사항으로 설정해 구현하는 것이 과제가 될 것이다.

- Mobile(UI): 모든 스케줄을 타임 테이블 형식으로 제공하며, 각 스케줄에 대해서도 해싱을 사용해 스케줄이 보색으로 출력되도록 하여 가시성을 성공적으로 높였다. 하지만, 보다 더 좋은 사용자 경험을 위해 지속적인 피드백을 통해 UI에 대한 개선은 꾸준히 이루어져야 할 것이며, 새로운 기능이 추가될 때마다 이에 맞춰 UI 역시 발전해야 할 것이다.

8.2. Difficulties Encountered

8.2.1. 박종범

Application을 제작할 때 어떠한 아키텍처를 구상하고 적용해야 의존성을 줄이고 재사용성이 용이할지 구상하는 과정에서 가장 큰 어려움이 뒤따랐다. 그래서 구현보다도 초반 앱 설계할 때에 시간을 더 많이 들이게 되었다. 하지만 다행히도 아키텍처 구상이 완료된 이후로는 오히려 수월하게 개발할 수 있었다.

8.2.2. 정유환

요구 사항에 대한 테스트를 진행할 때 테스트하기 어려운 부분을 최소화 하는 부분에서 어려움을 겪었다. 총 3가지 테스트 방식 중에서 코드 레벨에서 테스트 할 수 있는 부분은 단위 테스트와 통합 테스트인데 모든 요구 사항을 테스트하기는 어려웠다. 외부 의존성을 최소화하여 코드 레벨에서 최대한 테스트를 많이 할 수 있도록 했지만 여전히 코드로 작성하기 어려운 부분이 있어 QA 테스트로 진행한 부분이 조금 아쉬움으로 남는다.

8.2.3. 이승주

처음으로 Spring을 사용해 참여했던 프로젝트였던 만큼, 전반적인 스프링 기술의 이해 부족과 더불어 프로젝트 경험 부족으로 인해 단순 로직 구현 부분에서도 많은 시간을 소비하게 되었다. 또한, 구현한 코드를 바탕으로 테스트 코드를 작성하는 부분을 완전히 이해하기에는 주어진 시간이 부족했고, 이 부분은 다른 전문가 팀원에게 도움을 받아 해결하였다.

8.2.4. 김동령

프로젝트 초기 단계에 서비스 기획 및 요구 사항 명세 과정에서 팀원들이 서로가 서비스 컨셉을 다르게 이해하고 있어서 이를 토의를 통해 조정하는 과정이 조금 어려웠다. 그러나, 회의를 충분히 가졌기에 서로 일관된 그림을 그리고 완성도 높은 요구 명세를 작성할 수 있었고, 후에 요구사항이 크게 변경되지 않아서 최종 개발 비용을 줄일 수 있었다. 또한 Flutter 프레임워크를 처음 사용하다 보니 개발에 생각보다 많은 시간이 들었고, 개발 계획 일정에 맞추지 못해서 어려웠다. 이 과정에서 혼자 모든 것을 하려는 것 보다는 팀원의 도움도 받는 것이 매우 중요하고, 협업 과정에서 현재 맡은 task에 대해 기한을 맞추기 어렵다는 판단이 서면 팀원에게 도움을 잘 청하는

것 또한 엔지니어의 중요한 역량이라는 것을 깨달았다.

9. Appendix

9.1. Member Contribution

Table 9-1: Member Contribution

Name	Role
박종범	프론트엔드 구조설계 및 구현 / API 명세제작 / 물리서버 세팅 및 배포지원 / TestFlight 테스팅환경 배포
정유환	DB 설계, 백엔드 애플리케이션 개발(회원, 채널, 스케줄), 테스트 코드 작성 / API 명세제작
이승주	Prototyping 및 시각화, 백엔드 채널&스케줄 삭제 및 업데이트 로직
김동령	Prototyping, 프론트엔드 UI/UX디자인 및 구현

9.2. User Feedback

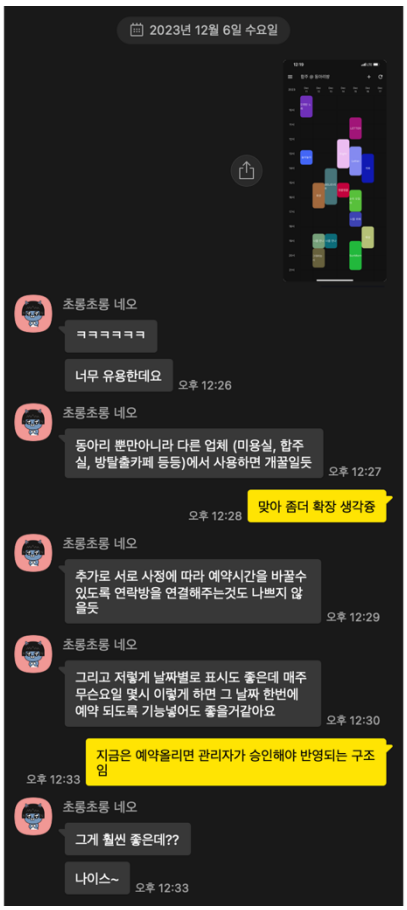


Figure 9-1 User Feedback

유저 피드백을 통해서 해당 앱의 유용성을 확인할 수 있었고, 보완해야 할 사항들에 대해서 피드백 받을 수 있었다.

9.3. Github Repository

- Client Repository: https://github.com/softwar7/weplan.git
- Server Repository: https://github.com/softwar7/weplan-server.git

10. Reference

- 조르테 간단등록군(https://torokun.jorte.com/ko/index.html)
- 스페이스 클라우드 (https://www.spacecloud.kr)
- 에브리 타임 (https://everytime.kr/)
- IOT SPACE - 회의실 예약 시스템(https://iotspace.co.kr/meetingroombooking)
- Slack (https://mokeya.tistory.com/142)
- 업무의 연속성 유지를 돕는 협업툴 '노션' – 동아일보 (https://www.donga.com/news/It/article/all/20220513/113388460/1)
- Software Architecture Patterns CH01 https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html
- "지속 성장 가능한 소프트웨어를 만들어가는 방법" https://geminikims.medium.com/%EC%A7%80%EC%86%8D-%EC%84%B1%EC%9E%A5-%EA%B0%80%EB%8A%A5%ED%95%9C-%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4%EB%A5%BC-%EB%A7%8C%EB%93%A4%EC%96%B4%EA%B0%80%EB%8A%94-%EB%B0%A9%EB%B2%95-97844c5dab63