



BOOKINGO

Software Requirement Specification

2021.5.16.

Introduction to Software Engineering

TEAM 1

팀장 임서현

조원 강종현

조원 이승우

조원 임재현

목차

| | |
|---|----|
| 1. Preface..... | 9 |
| 1.1. Readership | 9 |
| 1.2. Scope | 9 |
| 1.3. Objective | 9 |
| 1.4. Document Structure..... | 9 |
| 2. Introduction..... | 10 |
| 2.1. Objectives | 10 |
| 2.2. Applied Diagrams | 10 |
| 2.2.1. UML..... | 11 |
| 2.2.2. Use case Diagram | 11 |
| 2.2.3. Sequence Diagram | 11 |
| 2.2.4. Class Diagram..... | 11 |
| 2.2.5. Context Diagram..... | 11 |
| 2.2.6. Entity Relationship Diagram..... | 12 |
| 2.3. Applied Tools..... | 12 |
| 2.3.1. Microsoft PowerPoint | 12 |
| 2.3.2. SequenceDiagram.org | 12 |
| 2.3.3. Lucidchart | 12 |
| 2.3.4. Visual Paradigm | 12 |
| 2.4. Project Scope | 13 |
| 2.5. References | 13 |
| 3. System Architecture - Overall..... | 14 |
| 3.1. Objectives | 14 |
| 3.2. System Organization..... | 14 |
| 3.2.1. Context Diagram..... | 15 |

| | |
|--|----|
| 3.2.2. Sequence Diagram | 16 |
| 3.2.3. Use Case Diagram..... | 17 |
| 4. System Architecture - Frontend | 18 |
| 4.1. Objectives | 18 |
| 4.2. Subcomponents..... | 18 |
| 4.2.1. Login&Out..... | 18 |
| 4.2.2.1. Attributes | 18 |
| 4.2.2.2. Methods | 18 |
| 4.2.2.3. Class Diagram | 18 |
| 4.2.2.4. Sequence Diagram..... | 19 |
| 4.2.2. Booking..... | 20 |
| 4.2.2.1. Attributes | 21 |
| 4.2.2.2. Methods | 21 |
| 4.2.4.3. Class Diagram | 21 |
| 4.2.4.4. Sequence Diagram..... | 22 |
| 4.2.3. Checking Reservation | 23 |
| 4.2.3.1. Attributes | 23 |
| 4.2.3.2. Methods | 24 |
| 4.2.3.3. Class Diagram | 25 |
| 4.2.3.4. Sequence Diagram..... | 25 |
| 4.2.4. Arranging Lecture Room Seats | 26 |
| 4.2.4.1. Attributes | 26 |
| 4.2.4.2. Methods | 26 |
| 4.2.4.3. Class Diagram | 27 |
| 4.2.4.4. Sequence Diagram..... | 27 |
| 4.2.5. Changing/Canceling Reservation..... | 28 |
| 4.2.5.1. Attributes | 28 |

| | |
|--|----|
| 4.2.5.2. Methods | 29 |
| 4.2.5.3. Class Diagram | 29 |
| 4.2.5.4. Sequence Diagram..... | 30 |
| 5. System Architecture – Backend | 31 |
| 5.1. Objectives | 31 |
| 5.2. Overall Architecture | 32 |
| 5.3. Subcomponents..... | 32 |
| 5.3.1. Cloud Function..... | 32 |
| 5.3.1.1. DB Handler Class | 33 |
| 5.3.1.2. Firebase Authentication Class | 33 |
| 5.3.2. Reservation System..... | 33 |
| 5.3.2.1. Class Diagram | 33 |
| 5.3.2.2. Sequence Diagram..... | 34 |
| 5.3.3. Reservation History System..... | 34 |
| 5.3.3.1. Class Diagram | 34 |
| 5.3.3.2. Sequence Diagram..... | 36 |
| 5.3.4. Seat Management System | 36 |
| 5.3.4.1. Class Diagram | 36 |
| 5.3.4.2. Sequence Diagram..... | 37 |
| 6. Protocol Design..... | 38 |
| 6.1. Objectives | 38 |
| 6.2. JSON | 38 |
| 6.3. OAuth | 39 |
| 6.4. Log In | 39 |
| 6.5. Reservation..... | 40 |
| 6.5.1. Set Reservation | 40 |
| 6.5.2. Check Reservation | 41 |

| | |
|--|----|
| 6.7. Change Reservation..... | 41 |
| 6.8. Cancel Reservation..... | 42 |
| 6.9. Seat Arrangement | 43 |
| 6.9.1. Set Seat Arrangement..... | 43 |
| 6.9.2. Get Seat Arrangement | 44 |
| 7. Database Design..... | 44 |
| 7.1. Objectives | 44 |
| 7.2. ER Diagram..... | 45 |
| 7.2.1. Entities | 46 |
| 7.2.2.1. User | 46 |
| 7.2.2.2. Lecture..... | 46 |
| 7.2.2.3. Bus..... | 47 |
| 7.2.2.4. Seat_Arrangement | 48 |
| 7.3. Relational Schema | 48 |
| 7.4. SQL DDL | 49 |
| 7.4.1. User..... | 49 |
| 7.4.2. Lecture | 50 |
| 7.4.3. Bus | 50 |
| 7.4.4. Seat_Arrangement..... | 50 |
| 7.4.5. Bus_Reservation_Status | 51 |
| 7.4.6. Lecture_Reservation_Status..... | 51 |
| 7.4.7. Lecture_Taken..... | 52 |
| 7.4.8. User_Reservation | 52 |
| 8. Testing Plan..... | 53 |
| 8.1. Objectives | 53 |
| 8.2. Testing Policy | 53 |
| 8.2.1. Development Testing | 53 |

| | |
|---|----|
| 8.2.1.1. Usability | 53 |
| 8.2.1.2. Performance..... | 54 |
| 8.2.1.3. Dependability | 54 |
| 8.2.1.4. Security..... | 54 |
| 8.2.2. Release Testing | 54 |
| 8.2.3. User Testing | 54 |
| 8.2.4. Testing Case | 55 |
| 9. Development Plan..... | 55 |
| 9.1. Objectives | 55 |
| 9.2. Frontend Environment..... | 55 |
| 9.2.1. Adobe Illustrator | 55 |
| 9.2.2. Adobe Xd | 55 |
| 9.2.3. Android Studio | 56 |
| 9.3. Backend Environment | 56 |
| 9.3.1. Firebase..... | 56 |
| 9.4 Tools and APIs..... | 57 |
| 9.4.1. Github | 57 |
| 9.4.2. SKKU Member API..... | 57 |
| 9.5. Constraint | 58 |
| 9.6. Assumptions and Dependencies | 59 |
| 10. Supporting Information..... | 59 |
| 10.1. Software Design Specification | 59 |
| 10.2. Document History | 59 |

List of Figures

| | | |
|-------------|---|----|
| [Figure 1] | System Architecture | 15 |
| [Figure 2] | Context Diagram..... | 16 |
| [Figure 3] | Sequence Diagram | 17 |
| [Figure 4] | Use Case Diagram for BOOKINGO..... | 17 |
| [Figure 5] | Class Diagram - Login&out..... | 19 |
| [Figure 6] | Sequence Diagram – Login&out | 20 |
| [Figure 7] | Class diagram – Booking | 22 |
| [Figure 8] | Sequence diagram – Booking | 23 |
| [Figure 9] | Class diagram – Checking Reservation | 25 |
| [Figure 10] | Sequence diagram – Checking Reservation..... | 26 |
| [Figure 11] | Class diagram – Arranging Lecture Room Seats | 27 |
| [Figure 12] | Sequence diagram – Arranging Lecture Room Seats..... | 28 |
| [Figure 13] | Class Diagram – Changing/Canceling Reservation | 30 |
| [Figure 14] | Sequence Diagram – Changing/Canceling Reservation | 31 |
| [Figure 15] | System Architecture - Backend..... | 32 |
| [Figure 16] | Class diagram – Reservation System..... | 33 |
| [Figure 17] | Sequence diagram – Reservation System | 34 |
| [Figure 18] | Class diagram – Reservation History System | 35 |
| [Figure 19] | Sequence diagram – Reservation History System | 36 |
| [Figure 20] | Class diagram – Seat Management System Class..... | 37 |
| [Figure 21] | Sequence diagram – Seat Management System..... | 38 |
| [Figure 22] | Overall ER diagram | 45 |
| [Figure 23] | ER diagram – User Entity | 46 |
| [Figure 24] | ER diagram – Lecture Entity | 47 |
| [Figure 25] | ER diagram - Bus..... | 47 |

| | | |
|-------------|-------------------------------------|----|
| [Figure 26] | ER diagram – Seat_Arrangement | 48 |
| [Figure 27] | Relational Schema | 49 |
| [Figure 28] | Adobe Illustrator 로고..... | 55 |
| [Figure 29] | Adobe Xd 로고 | 56 |
| [Figure 30] | Android Studio 로고 | 56 |
| [Figure 31] | Firebase 로고 | 57 |
| [Figure 32] | Github 로고..... | 57 |
| [Figure 33] | 성균인 로그인 화면..... | 58 |

List of Tables

| | | |
|------------|--|----|
| [Table 1] | Log In request | 39 |
| [Table 2] | Log In response..... | 39 |
| [Table 3] | Set reservation request | 40 |
| [Table 4] | Set reservation response..... | 40 |
| [Table 5] | Check reservation request | 41 |
| [Table 6] | Check reservation response..... | 41 |
| [Table 7] | Change reservation request | 41 |
| [Table 8] | Change reservation response..... | 42 |
| [Table 9] | Cancel reservation request | 42 |
| [Table 10] | Reservation response | 42 |
| [Table 11] | Set seat arrangement request..... | 43 |
| [Table 12] | Set seat arrangement response | 43 |
| [Table 13] | Get seat arrangement request | 44 |
| [Table 14] | Get seat arrangement response..... | 44 |
| [Table 15] | SQL DDL – User Table..... | 49 |
| [Table 16] | SQL DDL – Lecture Table | 50 |
| [Table 17] | SQL DDL – Bus Table | 50 |
| [Table 18] | SQL DDL – Seat_Arrangement Table | 50 |
| [Table 19] | SQL DDL – Bus_Reservation_Status Table | 51 |
| [Table 20] | SQL DDL – Lecture_Reservation_Status Table | 51 |
| [Table 21] | SQL DDL – Lecture_Reservation_Status Table | 52 |
| [Table 22] | SQL DDL – Lecture_Reservation_Status Table | 52 |
| [Table 23] | Document History..... | 59 |

1. Preface

Preface에서는 본 문서의 예상 독자, 범위, 목표 및 BOOKINGO용 설계 문서의 문서 구조에 대해 서술한다.

1.1. Readership

본 문서는 총 10개의 카테고리로 구분된다. 본 문서의 대상 독자는 소프트웨어를 개발하는 Team1, 개발을 지도하는 지도교수, 그리고 본 소프트웨어의 개발과 유지를 담당하는 모든 사람들이다.

1.2. Scope

본 문서는 셔틀 버스와 강의실 좌석 예약 시스템을 운영하는 BOOKING을 구현하는데 사용할 기술적 설계에 대한 정의를 위해 사용된다. BOOKING은 크게 셔틀 좌석 예약, 강의실 좌석 예약 두 부분으로 구성된다.

1.3. Objective

본 문서는 학교 시설 내 좌석 예약 서비스를 제공하기 위한 시스템인 BOOKINGO의 기술적 설계에 대한 설명을 제공하기 위한 목적으로 작성되었다.

1.4. Document Structure

- **1. Preface:** 이 장에서는 본 문서의 독자, 범위, 구조를 설명한다.
- **2. Introduction:** 이 장에서는 본 문서에 사용된 도구, 다이어그램, 목적을 설명한다.
- **3. Overall System Architecture:** 이 장에서는 context diagram, sequence diagram, use case diagram을 이용한 시스템의 구조에 대해 설명한다.

- **4. System Architecture - Frontend:** 이 장에서는 class diagram, sequence diagram을 이용한 Frontend 시스템 구조에 대해 설명한다.
- **5. System Architecture - Backend:** 이 장에서는 class diagram, sequence diagram을 이용한 Backend 시스템 구조에 대해 설명한다.
- **6. Protocol Design:** 이 장에서는 클라이언트와 서버가 소통을 위해 사용하는 프로토콜 디자인에 대해 설명한다.
- **7. Database Design:** 이 장에서는 ER 다이어그램과 SQL DDL을 이용한 데이터베이스 설계를 설명한다.
- **8. Testing Plan:** 이 장에서는 시스템의 테스트 계획을 설명한다.
- **9. Development Plan:** 이 장에서는 시스템 개발을 위한 도구, 제약조건 등에 대해 설명한다.
- **10. Supporting Information:** 이 장에서는 본 문서의 지침과 문서의 이력을 서술한다.

2. Introduction

2.1. Objectives

Introduction에서는 본 문서의 System Design을 위한 Diagram과 Tool에 대해 소개하고 본 프로젝트의 범위에 대해 설명한다.

2.2. Applied Diagrams

2.2.1. UML

UML은 Unified Modeling Language의 약자로 소프트웨어 공학에서 사용되는 표준화된 모델링 언어이고, 여러 Diagram들을 통합하여 만들어졌다. UML은 시스템을 이해하기 쉬운 형태로 표현하여 기획자, 개발자, 아키텍처가 효율적으로 의사소통 할 수 있게 도와준다. 본 문서에서는 UML의 Class Diagram, Sequence Diagram, State Diagram을 이용하여 시스템의 구조를 시각화한다.

2.2.2. Use case Diagram

Use case Diagram은 시스템과 사용자의 상호작용을 다이어그램으로 표현한 것으로 사용자의 관점에서 시스템의 서비스 혹은 기능 및 그와 관련된 외부 요소에 대해 설명한다. 일반적으로 프로젝트를 시작하고, 프로젝트의 개발 범위를 정하거나, 사용자의 요구사항을 정의하고 프로그램에 수행해야 하는 기능의 명세를 알아야 할 때 이용한다.

2.2.3. Sequence Diagram

Sequence Diagram은 문제 해결을 위한 객체를 정의하고 객체들 사이의 상호작용을 시간의 흐름에 따라 나타낸다. Sequence Diagram은 현재 존재하는 시스템이 어떠한 시나리오로 움직이고 있는지에 대해 설명하는데 장점이 있다.

2.2.4. Class Diagram

Class Diagram은 UML에서 시스템의 클래스, 속성, 동작 방식, 객체사이의 관계를 표현함으로써 구조를 기술하는 정적 구조의 다이어그램이다. 클래스는 객체의 구성 블록이기 때문에 Class Diagram은 UML의 구성 블록이다. 위쪽 행에는 클래스의 이름, 가운데 행에는 클래스의 속성, 아래쪽 행에는 Method가 표시된다.

2.2.5. Context Diagram

Context Diagram은 가장 높은 레벨의 다이어그램으로 시스템의 Context와 경계를 정의하여 시스템과 상호 작용하는 객체를 보여준다. Context Diagram의 경우 프로젝트의 모든 이해당사자

가 이용하기 때문에 쉬운 언어로 작성해야 한다. Context Diagram은 시스템 요구 사항 및 제약 조건을 개발할 때 고려해야 할 외부 요인 및 이벤트에 집중한다.

2.2.6. Entity Relationship Diagram

ER Diagram은 구조화된 데이터에 대한 표현으로 데이터베이스에 저장된 개체들의 관계를 보여준다. ER Diagram은 객체와 속성을 정의하고 이들 사이의 관계를 표시함으로써 데이터베이스의 논리적 구조를 보여준다. ER Diagram은 데이터베이스 설계를 디자인하는데 사용된다.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

다이어그램을 작성하기 위해 가장 기본적으로 쓸 수 있는 툴이다. 텍스트와 다양한 도형을 지원해주고 도형의 위치 및 크기에 대해서 자유도가 매우 높다. 가장 익숙한 툴이기 때문에 다수의 팀원이 사용한 툴이기도 하다. 하지만 자유도가 높은 만큼 형식의 통일성을 유지하며 수정하기 어렵다는 단점이 있다.

2.3.2. SequenceDiagram.org

Sequence Diagram 제작을 위한 web tool이다. 코드 형식의 글에 맞춰 작성하면 Sequence Diagram을 위한 박스와 선을 그려준다. Powerpoint보다 더 쉽고 빠르게 만들고 수정할 수 있다.

2.3.3. Lucidchart

ER diagram 제작을 위한 온라인 툴이다. ER diagram뿐만 아니라 데이터베이스, sequence diagram, class diagram 등 다양한 UML 형식의 다이어그램을 쉽게 만들 수 있는 툴을 제공하고 있다. ER diagram을 만들 때 도형을 만들고 선을 이을 때 간단한 클릭 몇 번으로 생성할 수 있다.

2.3.4. Visual Paradigm

Sequence Diagram 제작에 쓰인 또 다른 온라인 툴이다. UML 다이어그램은 물론이고 Agile 메소드를 위한 BPMN Modeling Platform 등 다양한 툴을 제공하는 사이트다. 마찬가지로 간단한 클릭 몇 번으로 도형과 선을 깔끔하게 만들 수 있게 도와준다.

2.4. Project Scope

본 문서에서 서술할 어플리케이션 서비스의 이름은 BOOKINGO이다. BOOKINGO는 기존 KINGO앱에 없는 셔틀 버스와 강의실 좌석 예약 시스템을 운영한다. 학교 내 시설을 사용하면서 학생과 교직원이 느꼈던 불편한 점을 해소해주는 것을 목표로 한다. BOOKING은 크게 셔틀 좌석 예약, 강의실 좌석 예약 두 부분으로 구성된다. 학교 구성원임을 인증한 유저는 로그인 후 크게 셔틀 좌석 예약 서비스와 강의실 좌석 예약 서비스를 이용할 수 있다. 본 프로젝트의 자세한 구현 범위는 다음과 같다.

‘Login & Out’을 통해 사용자의 접속관리가 이루어진다. 정상적으로 접속이 완료되면 ‘Booking Bus & Lecture Room Seat’을 통해 셔틀버스와 강의실 좌석에 대한 예약을 진행할 수 있고, ‘Check Reservation’으로 예약을 확인하거나 ‘Changing/Canceling Reservation’으로 예약을 수정 변경할 수 있다. 또한 ‘Arrange Lecture Room Seating’으로 좌석의 위치, 타입을 변경할 수 있고, ‘Seat Management System’을 통해 좌석에 대한 정보를 저장하고 관리한다. 추가적으로 ‘Reservation History System’에서는 시스템의 유지보수와 관련된 내용들을 저장한다.

https://github.com/skkuse/2020spring_41class_team1/blob/master/docs/SRS_TEAM1.pdf

2.5. References

▷ 2020spring_41class_team1, “SDS_TEAM1.pdf”, Last Edited: May 20, 2020.

https://github.com/skkuse/2020spring_41class_team1/blob/master/docs/SDD_TEAM1.pdf

▷ Alex Gallia, “UML Tutorial: : How to Model any Process or Structure in Your Business”, *Process*, October 26, 2018.

<https://www.process.st/uml-tutorial/>

▷ Kymberly Fergusson, “Entity Relationship Diagrams with draw.io”, *draw.io*, March 8th, 2018.

<https://drawio-app.com/entity-relationship-diagrams-with-draw-io/>

▷ InterSystems Documentation “Data Types”,

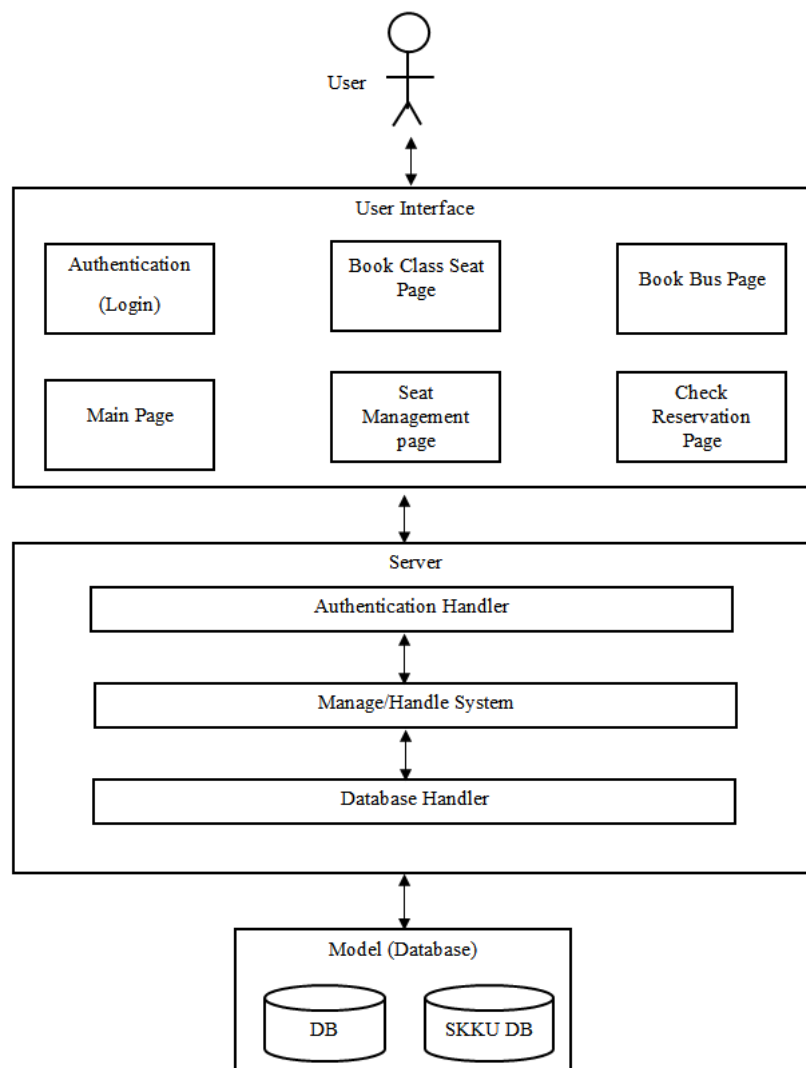
https://docs.intersystems.com/irislatest/csp/docbook/Doc.View.cls?KEY=RSQL_datatype#RSQL_datatype_ddltable

3. System Architecture - Overall

3.1. Objectives

‘System Architecture – Overall’에서는 BOOKINGO 시스템의 전체적인 구조에 대해 서술한다. 시스템의 각 subcomponents의 구조와 관계는 UML의 대표 다이어그램을 통해 상세하게 알 수 있다.

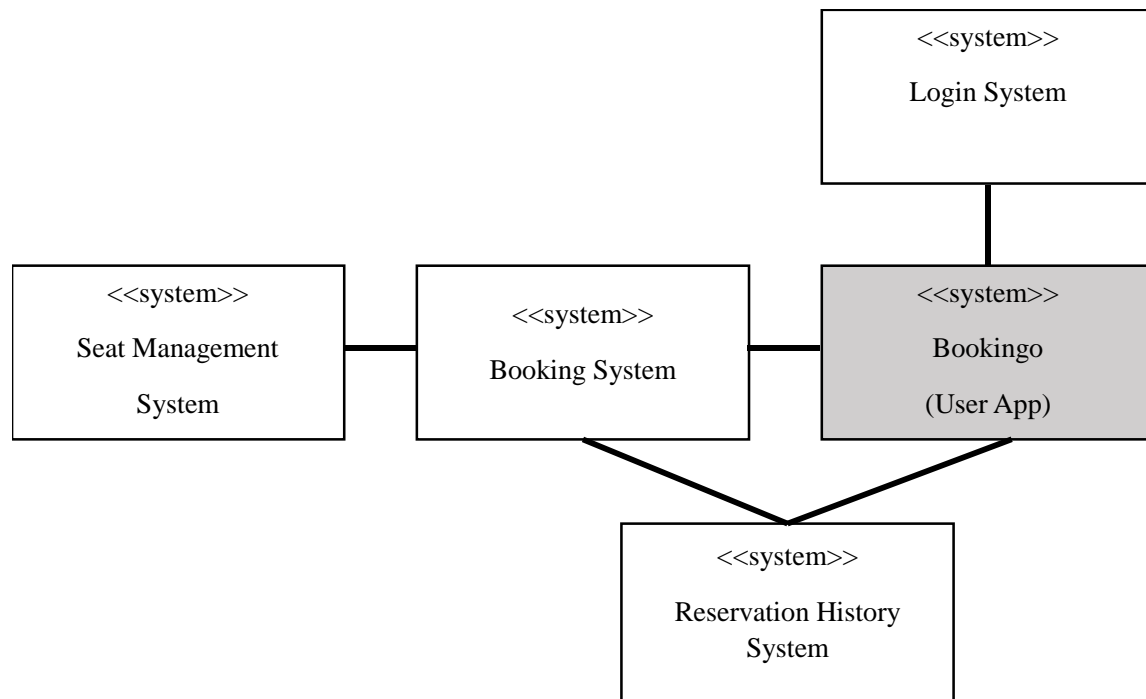
3.2. System Organization



[Figure 1] System Architecture

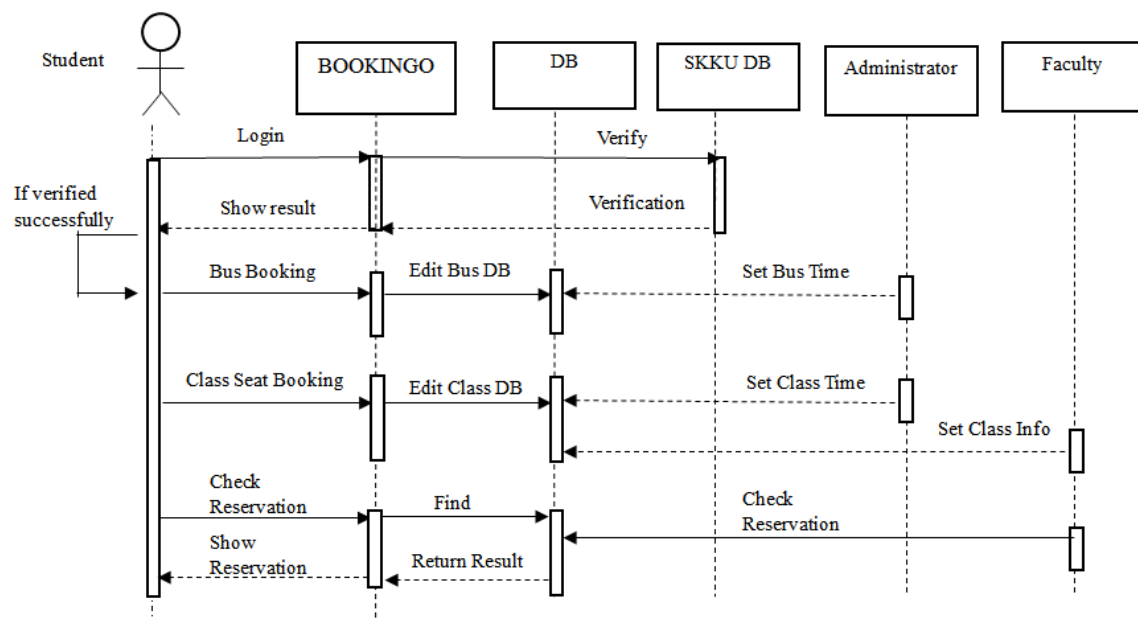
BOOKINGO 서비스는 클라이언트-서버 모델을 적용해 설계했다. Frontend Application은 사용자와의 모든 상호작용을 맡는다. 시스템 간 상호작용에는 JSON을 기반으로 데이터를 송수신한다. BOOKING 시스템에서 Backend Application은 Firebase를 통해 데이터베이스에서 필요한 정보를 저장한다. 데이터 요청이 들어오면 JSON을 기반으로 가공하여 전달한다. Frontend Application에서 사용자가 선택을 완료하면 완료한 결과에 따라 Backend Application에서 요청에 따라 데이터베이스를 업데이트하는 과정을 거친다.

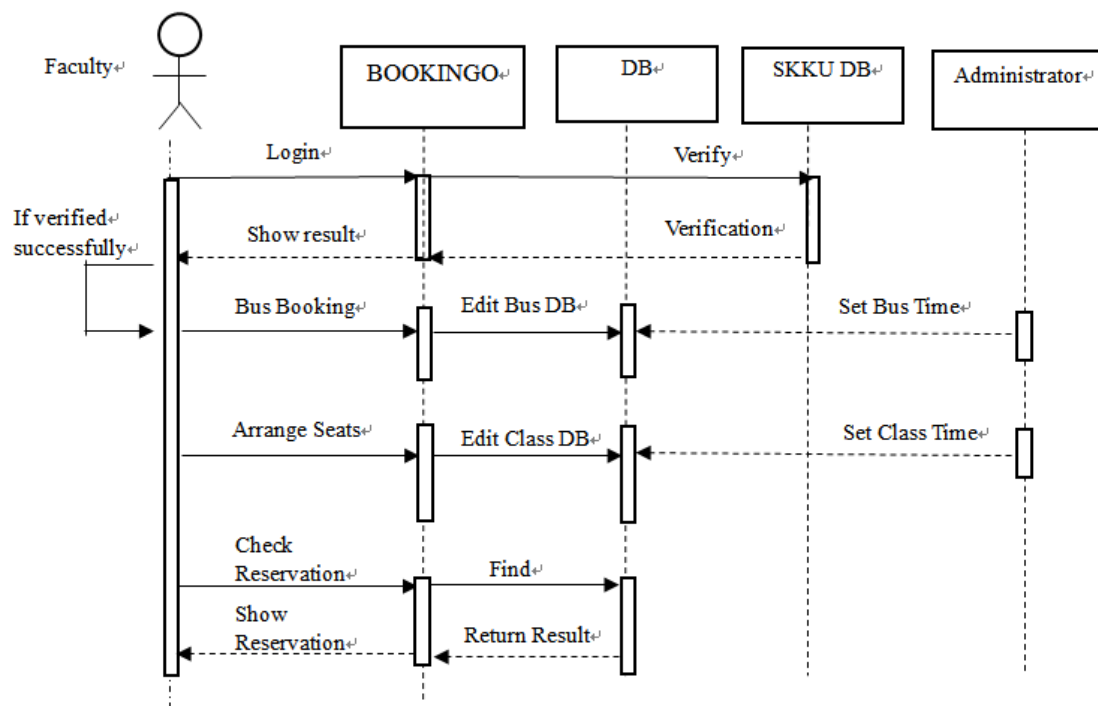
3.2.1. Context Diagram



[Figure 2] Context Diagram

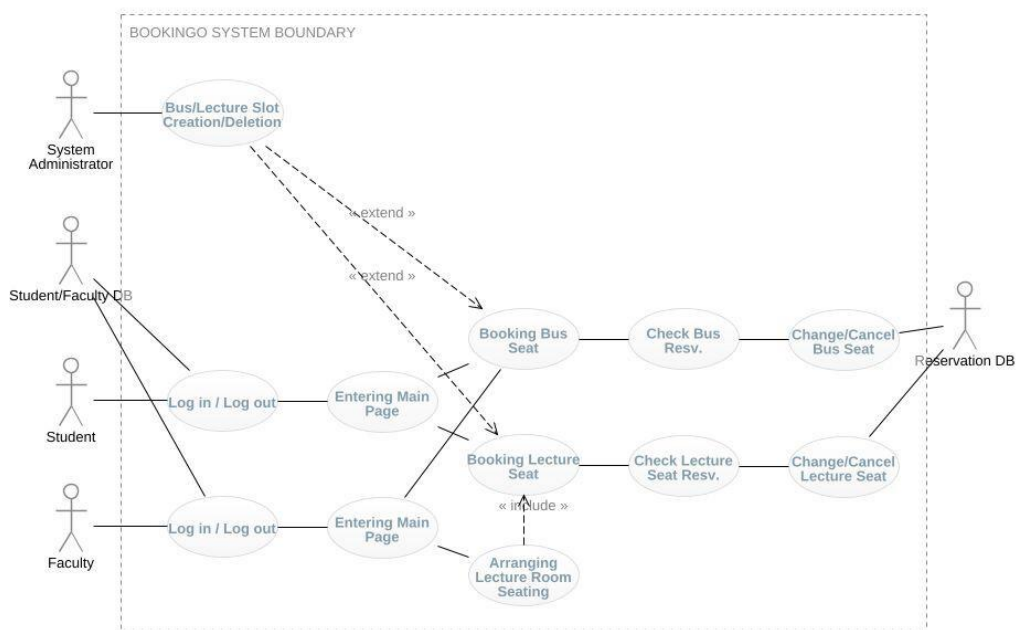
3.2.2. Sequence Diagram





[Figure 3] Sequence Diagram

3.2.3. Use Case Diagram



[Figure 4] Use Case Diagram for BOOKINGO

4. System Architecture - Frontend

4.1. Objectives

이 장에서는 Frontend 시스템의 구조, attributes, 그리고 function을 기술한다. 각 component마다의 관계도 기술한다.

4.2. Subcomponents

4.2.1. Login&Out

Login은 학내 구성원 DB에 포함된 인원이 서비스 이용을 위해 접속을 하는 절차를 의미한다.

Logout은 로그인 상태인 사용자가 시스템에서 나가는 절차를 의미한다.

4.2.2.1. Attributes

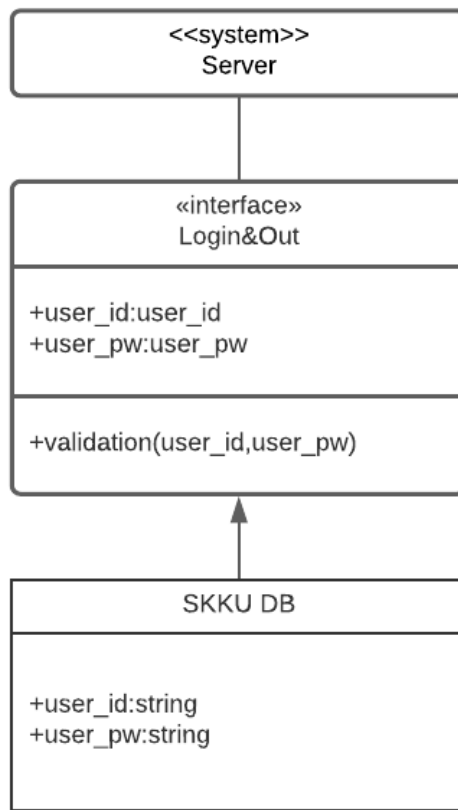
Login&Out 객체가 갖는 특성은 다음과 같다

- **User id:** 사용자의 성균인 아이디
- **User pw:** 사용자의 성균인 비밀번호

4.2.2.2. Methods

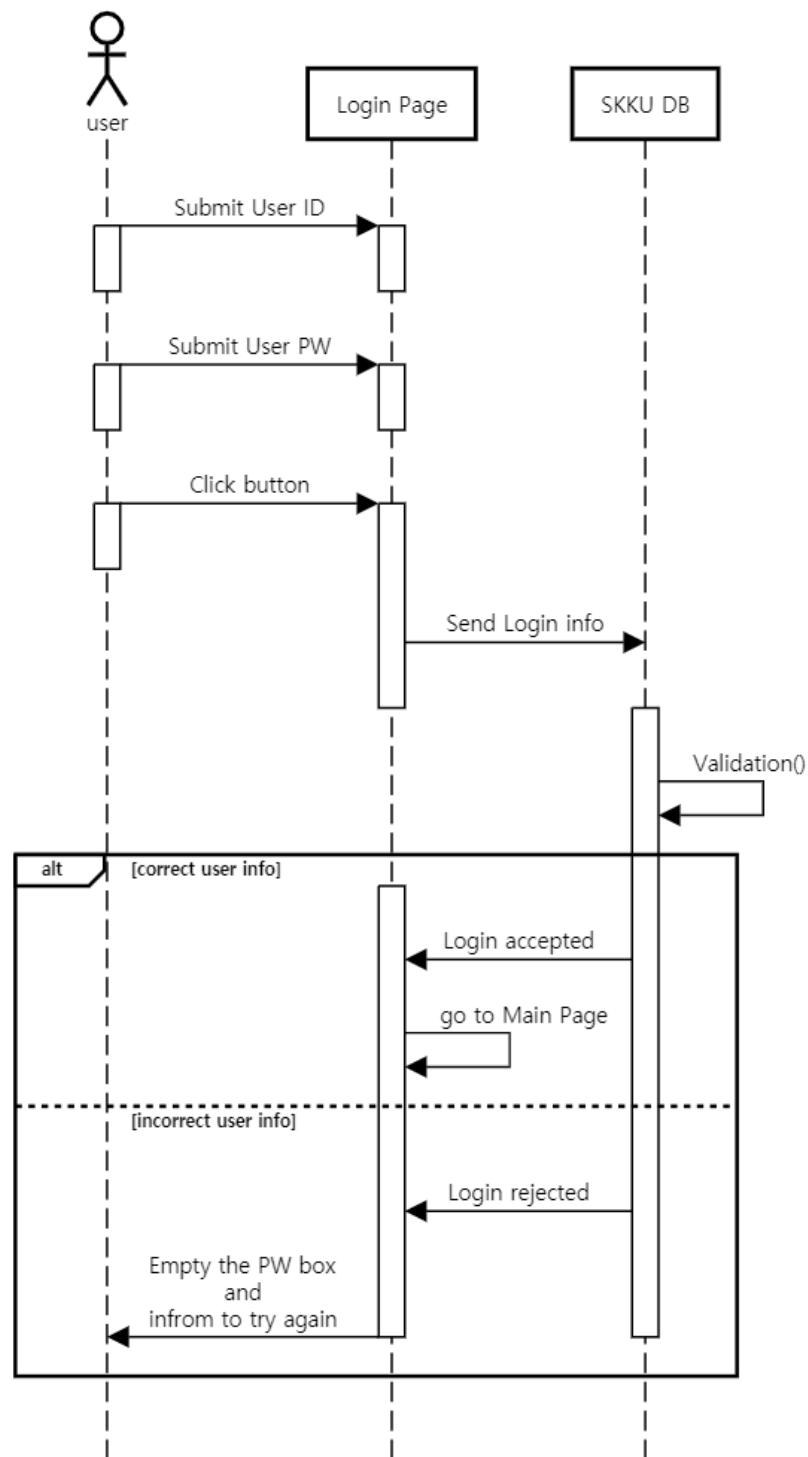
- **Validation():** 사용자가 입력한 정보를 확인한다.

4.2.2.3. Class Diagram



[Figure 5] Class Diagram - Login&out

4.2.2.4. Sequence Diagram



[Figure 6] Sequence Diagram – Login&out

4.2.2. Booking

Booking은 사용자의 예약 기능을 담당하고 있다. 사용자는 자신이 원하는 날짜 및 시간의 강

의실 및 셔틀 버스를 예약할 수 있다.

4.2.2.1. Attributes

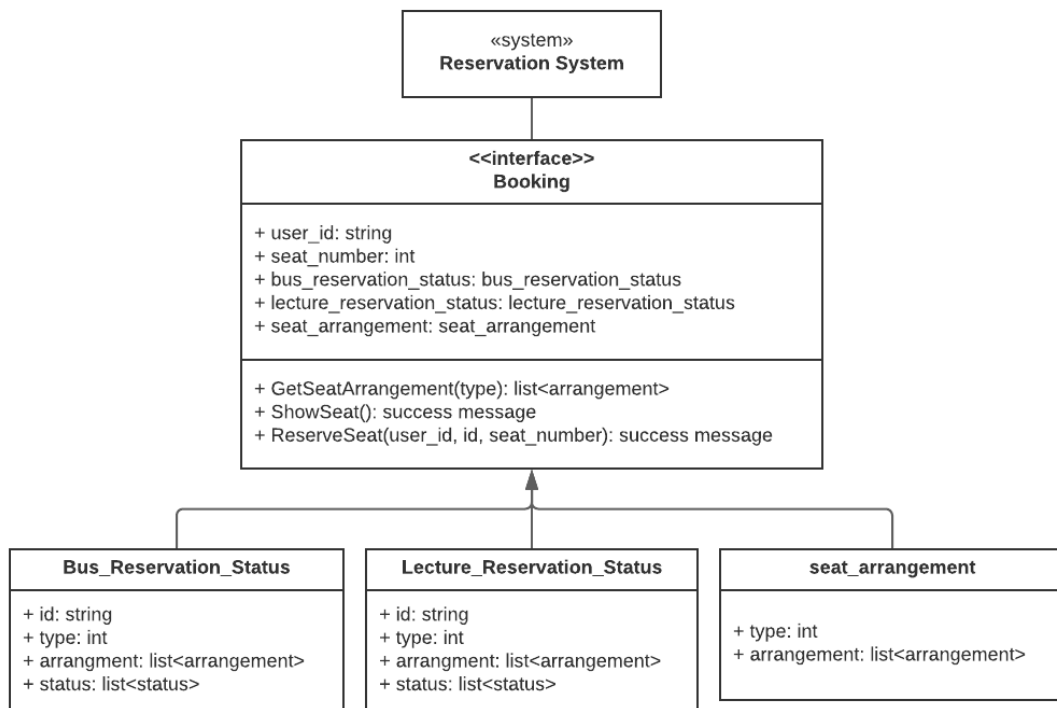
Booking 객체가 갖는 특성은 다음과 같다.

- **User id:** 유저의 kingo id
- **Reservation type:** 예약 유형 (강의실/버스)
- **Id:** 강의실/버스의 id
- **Time:** 강의 시간/버스 출발 시간
- **Location:** 강의 장소/버스 출발 장소
- **Seat type:** 좌석 배치 유형
- **Seat arrangement:** 좌석 배치

4.2.2.2. Methods

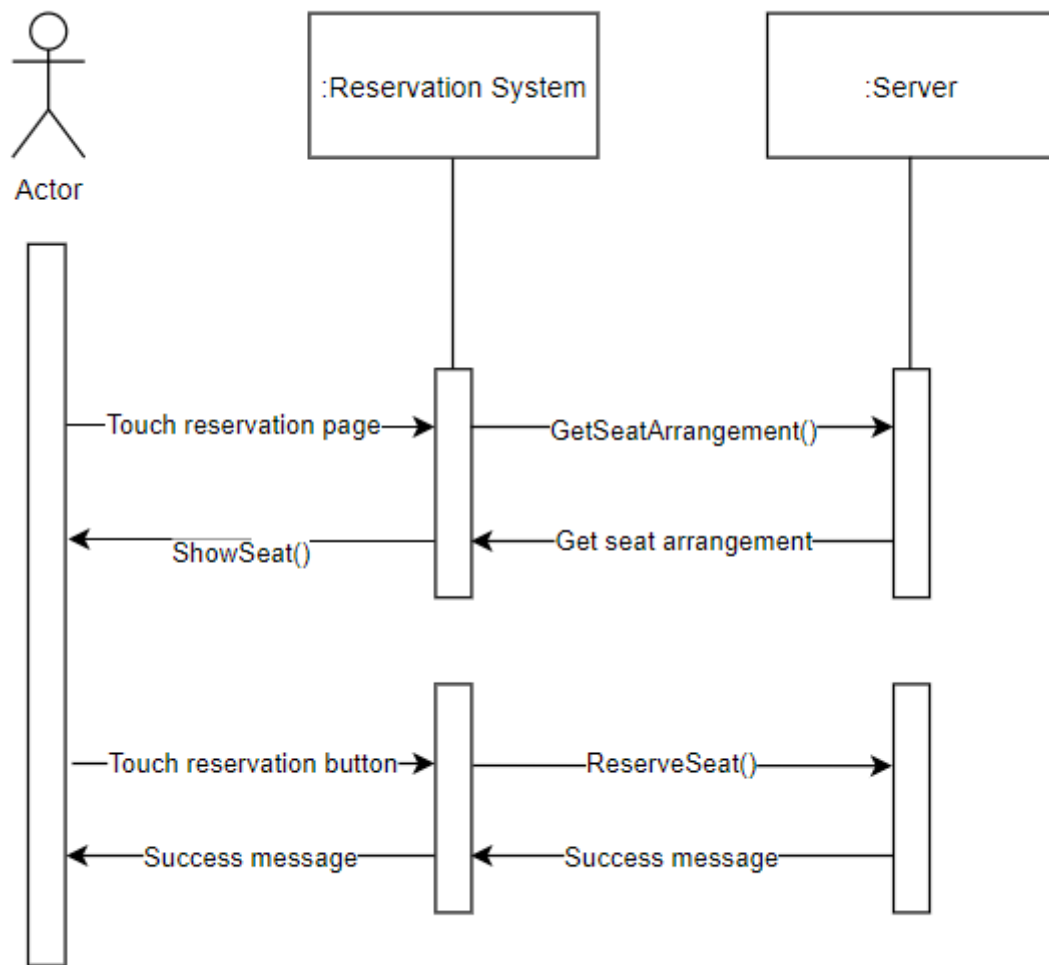
- **GetSeatArrangement():** 좌석의 배치 상태를 요청한다.
- **ShowSeat():** 좌석의 배치 상태를 보여준다.
- **ReserveSeat():** 사용자의 입력에 따라 예약을 완료한다.

4.2.4.3. Class Diagram



[Figure 7] Class diagram – Booking

4.2.4.4. Sequence Diagram



[Figure 8] Sequence diagram – Booking

4.2.3. Checking Reservation

Check reservation 기능을 통해 당일, 또는 이후의 강의실과 셔틀버스 예약 내역을 확인하는 것이 가능하다.

4.2.3.1. Attributes

Checking Reservation 객체가 갖는 특성은 다음과 같다:

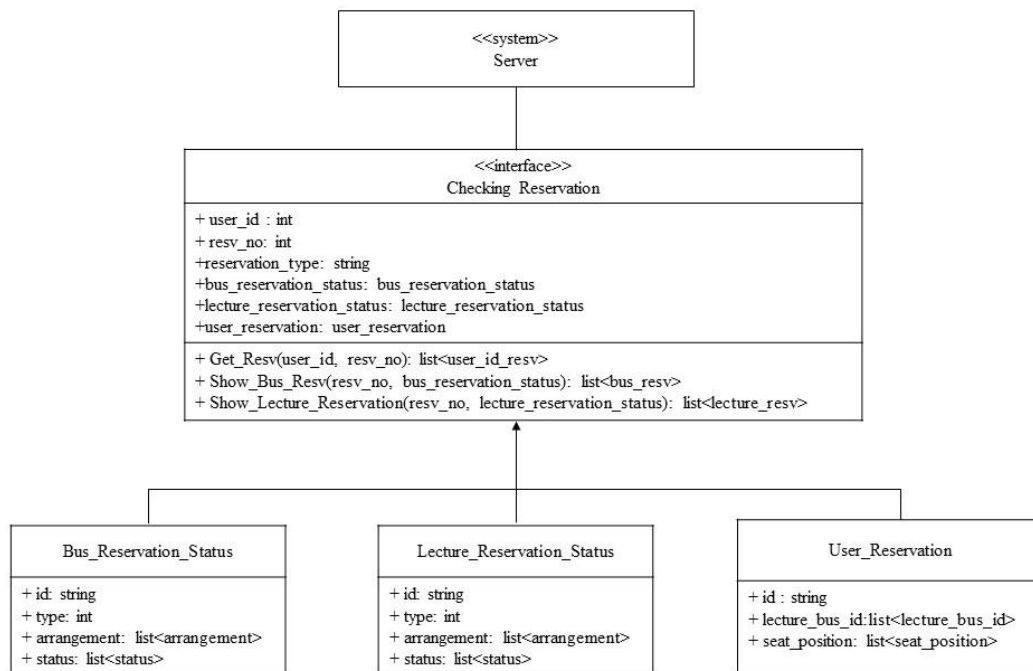
- **User id:** 접속시에 사용한 사용자의 성균인 ID
- **Resv no:** 각 예약별 고유 번호를 의미한다.

- **Reservation type:** 강의실 및 버스의 예약 유형을 의미한다.
- **Bus reservation status:** 버스 예약 상태를 담은 객체이다.
- **Lecture reservation status:** 강의실 예약 상태를 담은 객체이다.
- **User reservation:** 사용자 예약정보를 담은 객체이다.

4.2.3.2. Methods

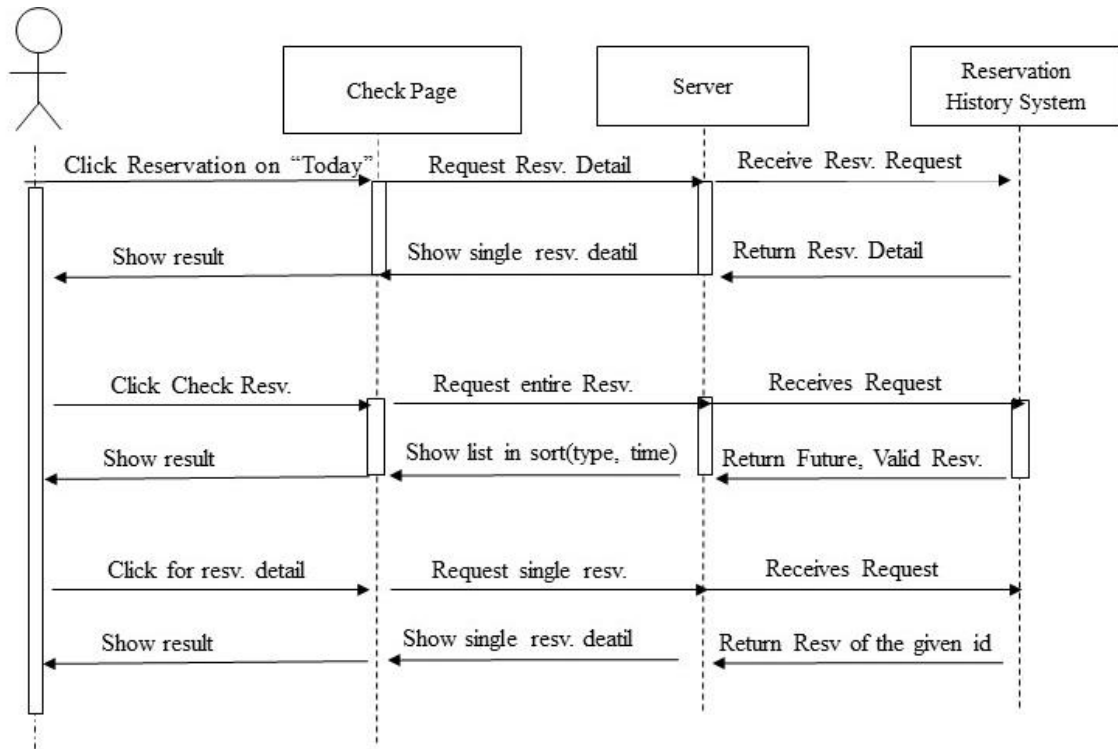
- **Get Resv():** 전체 예약 리스트를 요청하여 종류별, 시간 별로 나타낸다.
- **Show Bus Resv():** 버스 개별예약의 세부사항을 표시한다.
- **Show Lecture Resv():** 강의실 개별예약의 세부사항을 표시한다.

4.2.3.3. Class Diagram



[Figure 9] Class diagram – Checking Reservation

4.2.3.4. Sequence Diagram



[Figure 10] Sequence diagram – Checking Reservation

4.2.4. Arranging Lecture Room Seats

Arranging lecture room seats은 교직원이 직접 좌석을 배치하고, 학생 좌석을 배정할 수 있는 component이다. 이 클래스는 교직원이 선택을 완료하면 데이터베이스에 데이터를 반영하도록 한다.

4.2.4.1. Attributes

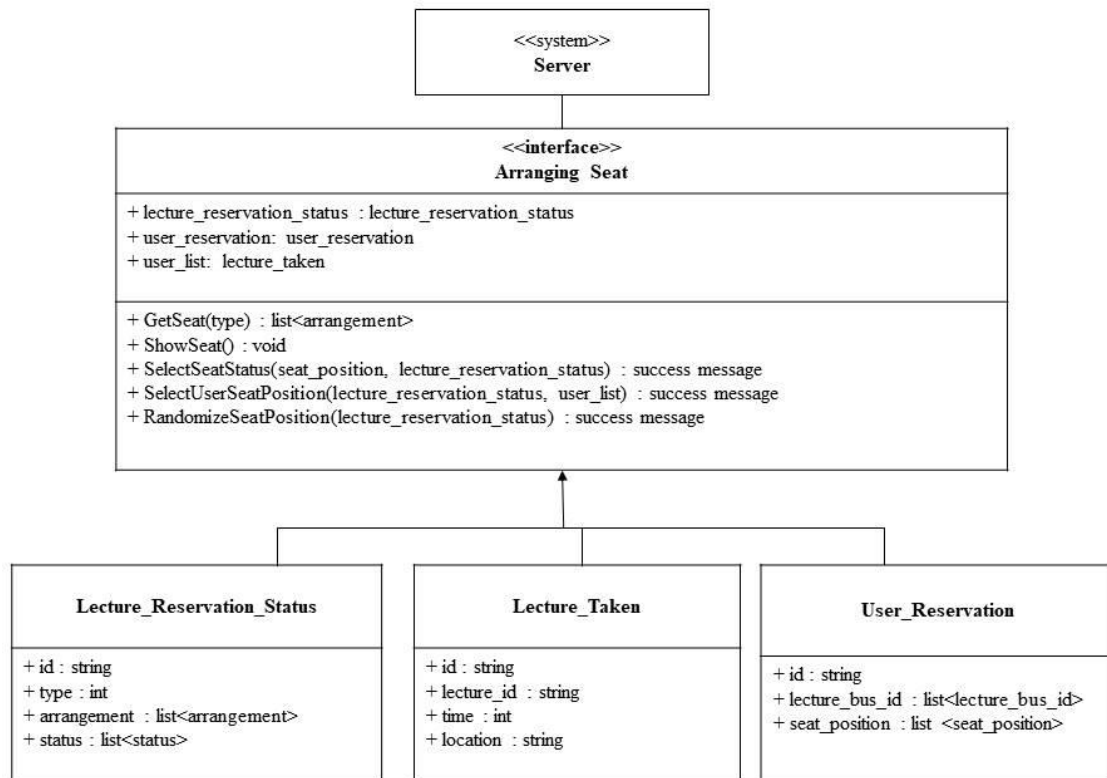
Arranging Lecture Room Seats 객체가 갖는 특성은 다음과 같다:

- **Lecture reservation status:** 강의실 예약 정보가 담긴 객체
- **User reservation:** 유저의 예약 내역 정보가 담긴 객체
- **User list:** 강의를 듣는 학생 목록

4.2.4.2. Methods

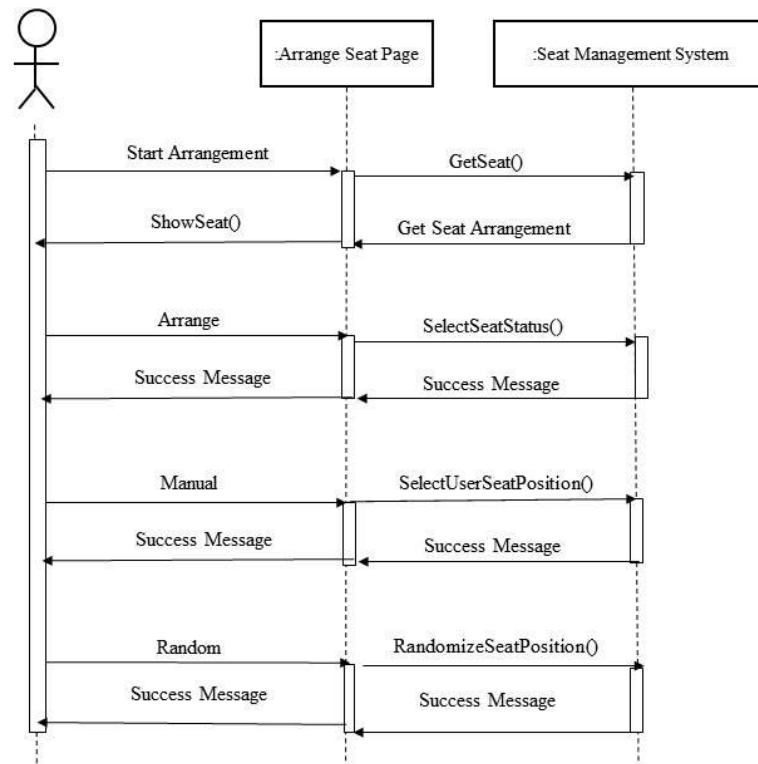
- GetSeat(): 좌석의 배치 상태를 요청한다.
- ShowSeat(): 좌석의 배치 상태를 보여준다.
- SelectSeatstatus(): 좌석의 예약 상태를 변경한다.
- SelectUserSeatPosition(): 학생을 수동으로 좌석에 배정한다.
- RandomizeSeatPosition(): 학생을 랜덤으로 좌석에 배정한다.

4.2.4.3. Class Diagram



[Figure 11] Class diagram – Arranging Lecture Room Seats

4.2.4.4. Sequence Diagram



[Figure 12] Sequence diagram – Arranging Lecture Room Seats

4.2.5. Changing/Canceling Reservation

사용자가 예약한 강의실의 좌석, 셔틀버스의 좌석을 변경하거나 취소할 수 있는 기능을 지원한다.

4.2.5.1. Attributes

Changing/Canceling Reservation 객체가 갖는 특성은 다음과 같다:

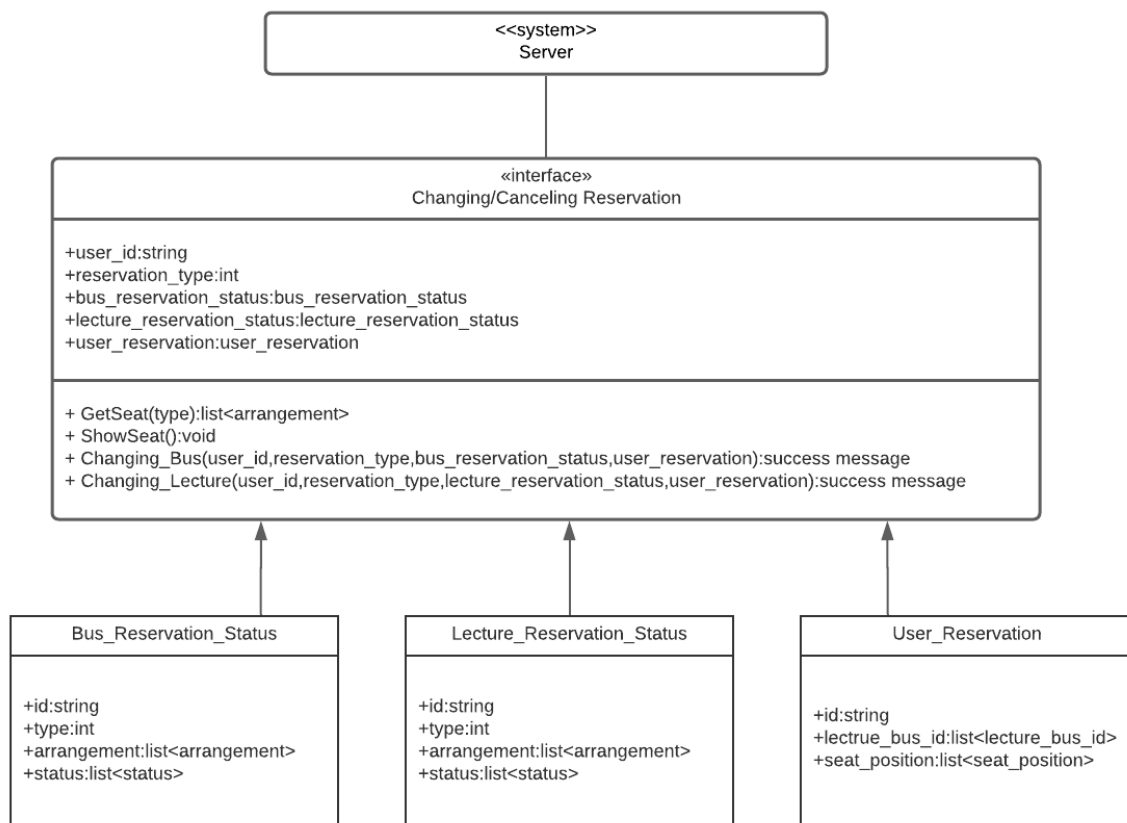
- **User id:** 사용자 고유 아이디
- **Reservation type:** 예약하고자 하는 기능
- **Id:** 강의실/버스의 id

- **Type:** 강의실/버스의 좌석 타입
- **Arrangement:** 강의실/버스의 좌석 배치
- **Status:** 강의실/버스의 예약 상태
- **Lecture bus id:** 강의실/버스의 고유번호
- **Seat position:** 예약된 좌석의 위치

4.2.5.2. Methods

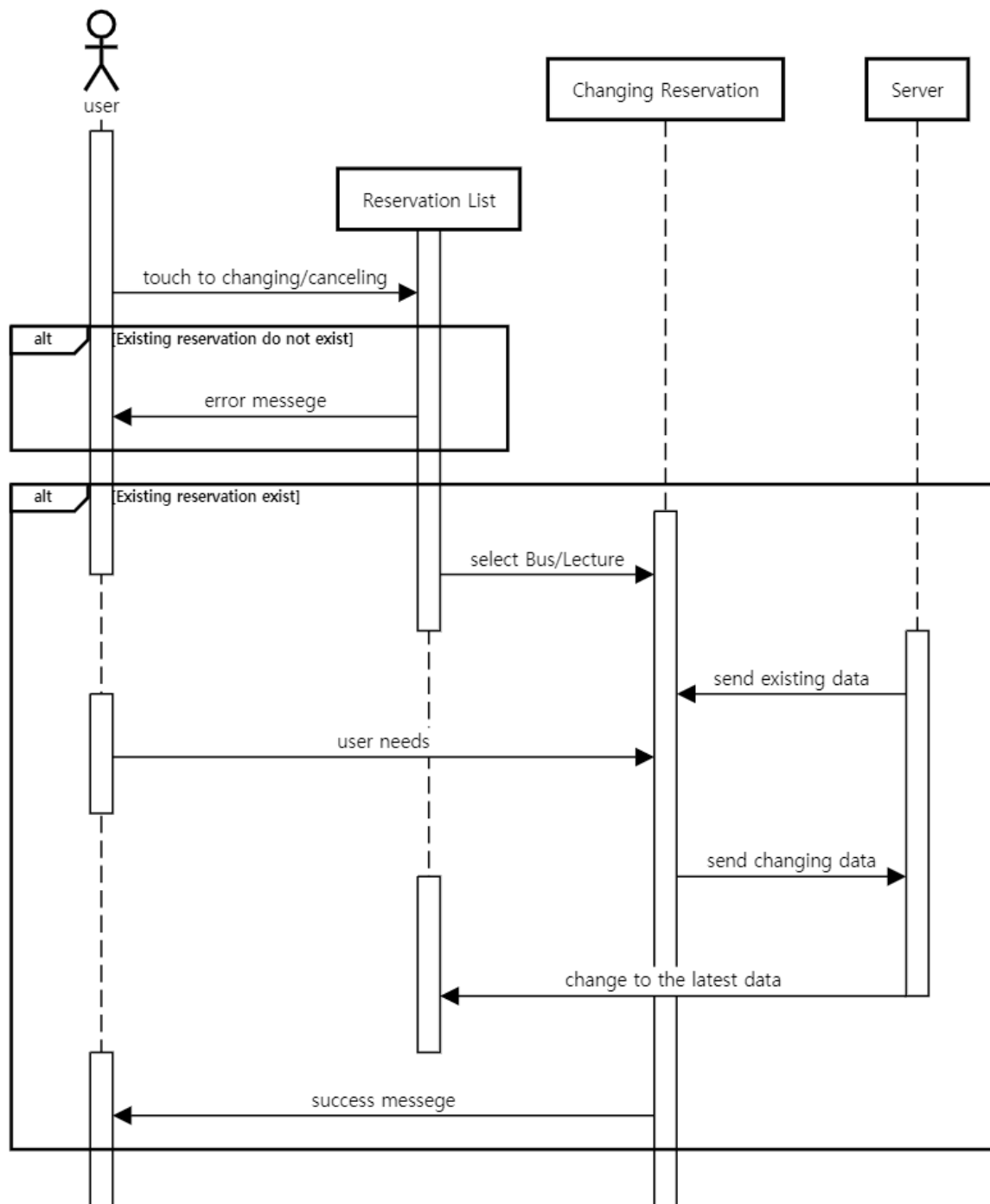
- **Getseat():** 좌석의 배치 상태를 요청한다.
- **Showseat():** 좌석의 배치 상태를 보여준다.
- **Changing bus():** 기존 버스 좌석 예약을 변경한다.
- **Changing lecture():** 기존 강의실 좌석 예약을 변경한다.

4.2.5.3. Class Diagram



[Figure 13] Class Diagram – Changing/Canceling Reservation

4.2.5.4. Sequence Diagram



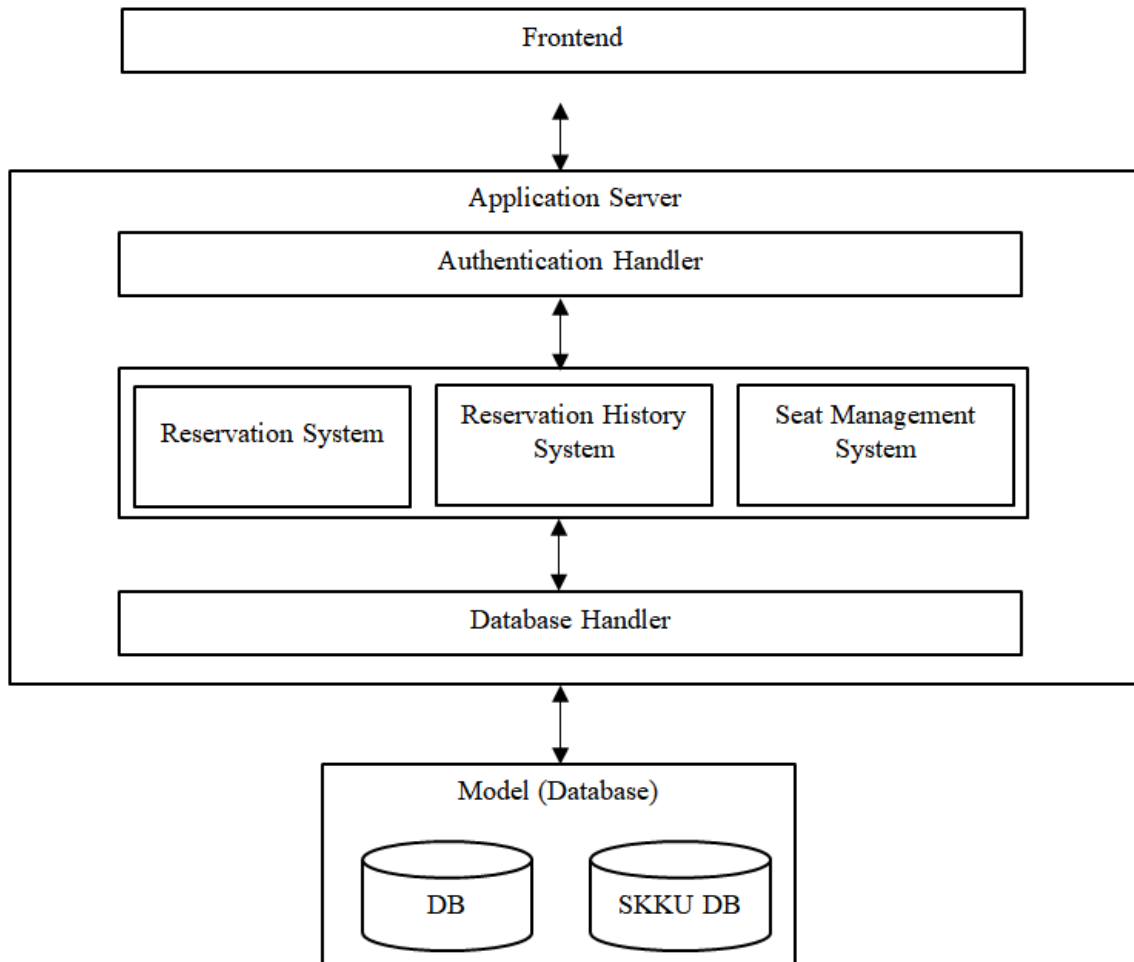
[Figure 14] Sequence Diagram – Changing/Canceling Reservation

5. System Architecture – Backend

5.1. Objectives

이 장에서는 Backend 시스템의 구조를 기술한다. DB와 클라우드 API들에 대해 기술한다.

5.2. Overall Architecture



[Figure 15] System Architecture - Backend

Backend 시스템의 전반적인 아키텍처는 다음과 같다. 컨트롤러에서 API 콜에 해당하는 클라우드 API를 사용해서 요청을 수행하게 된다. Authentication은 외부 API를 사용한다. Authentication이 완료되면 알맞은 Backend system을 거치고 DB handler를 통해 데이터베이스를 업데이트한다.

5.3. Subcomponents

5.3.1. Cloud Function

5.3.1.1. DB Handler Class

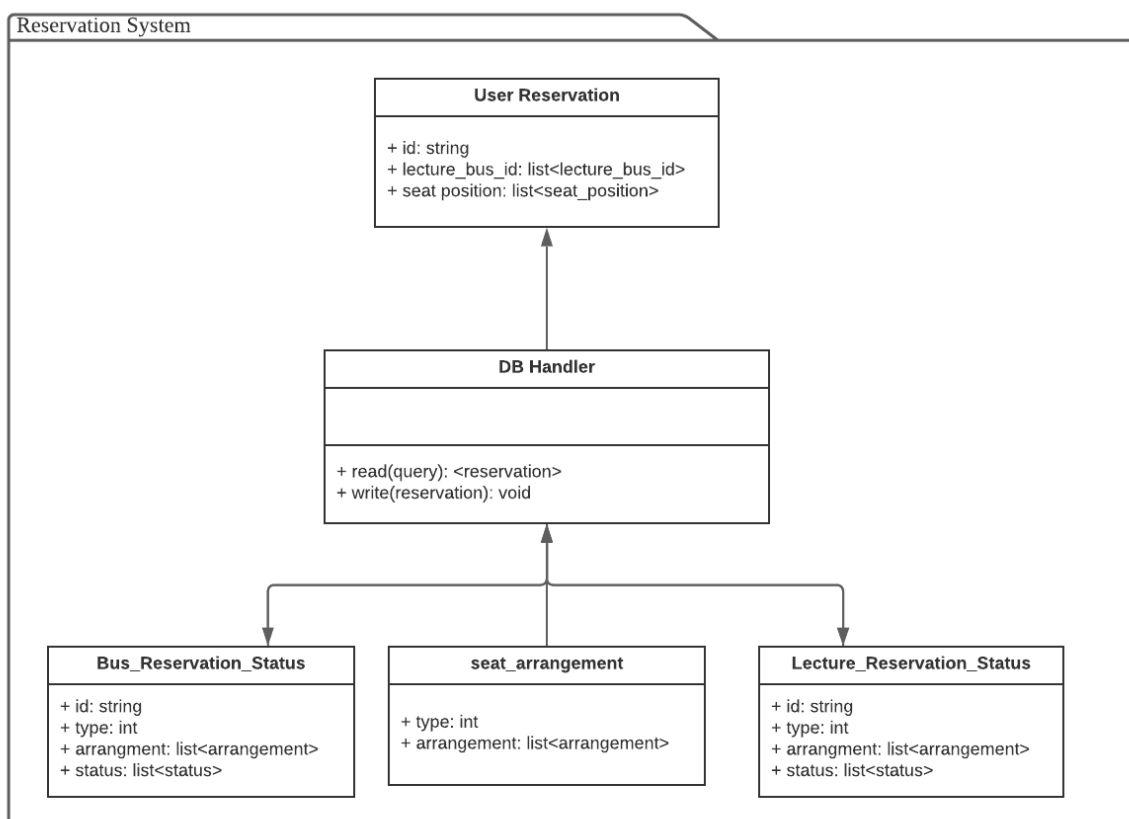
데이터베이스의 입출력을 관리한다. 사용자가 입력한 요청에 맞게 데이터베이스 정보를 불러 오거나 업데이트한다.

5.3.1.2. Firebase Authentication Class

Firebase에서 제공해주는 Authentication 기능을 사용한다.

5.3.2. Reservation System

5.3.2.1. Class Diagram

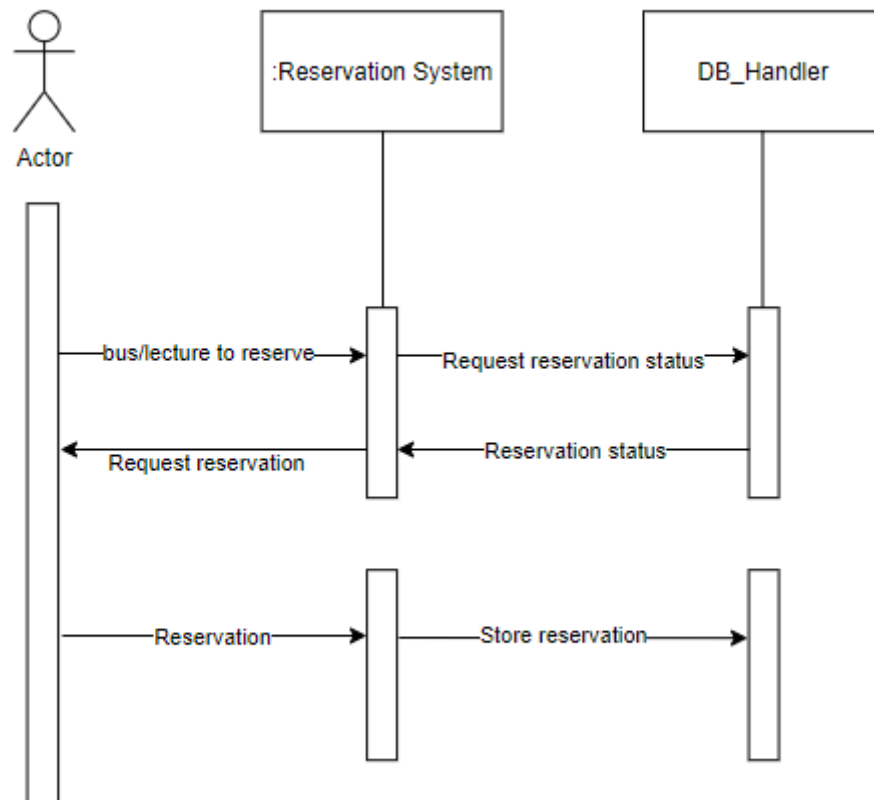


[Figure 16] Class diagram – Reservation System

- Class Description

Reservation System Class: 예약을 담당하는 역할을 한다. 유저가 시스템에 예약을 요청하면 시스템은 유저 예약 목록에 해당 예약을 더해준다. 동시에 해당 예약 사항에 대해 버스나 강의의 좌석 예약 상태를 업데이트 한다.

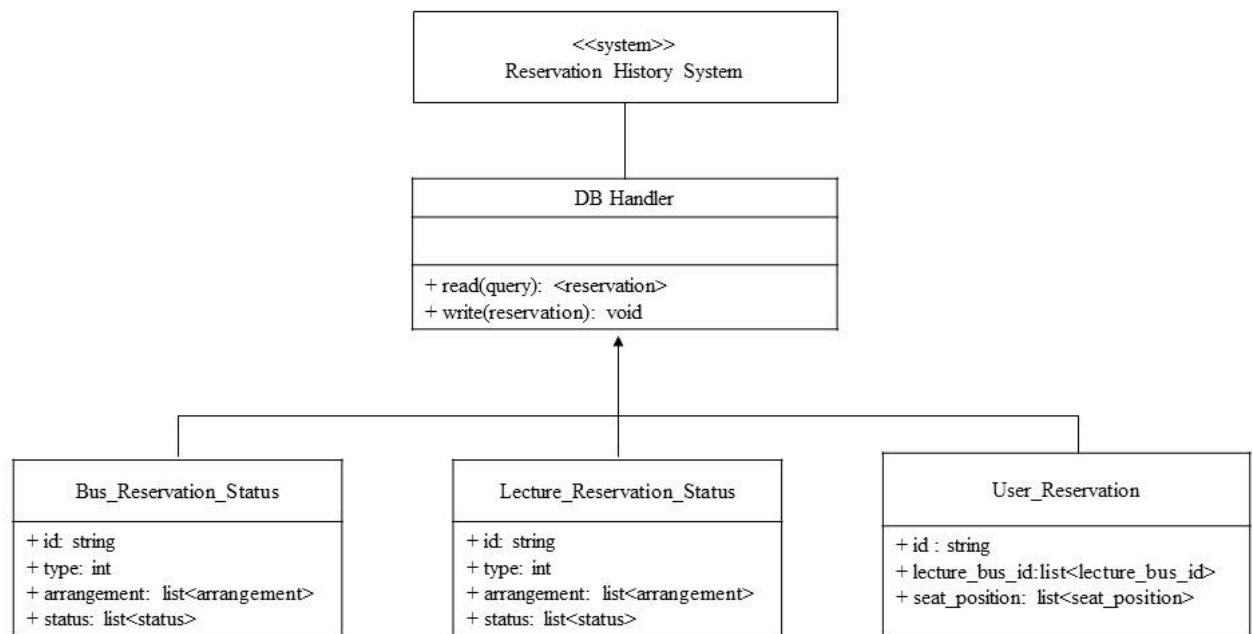
5.3.2.2. Sequence Diagram



[Figure 17] Sequence diagram – Reservation System

5.3.3. Reservation History System

5.3.3.1. Class Diagram

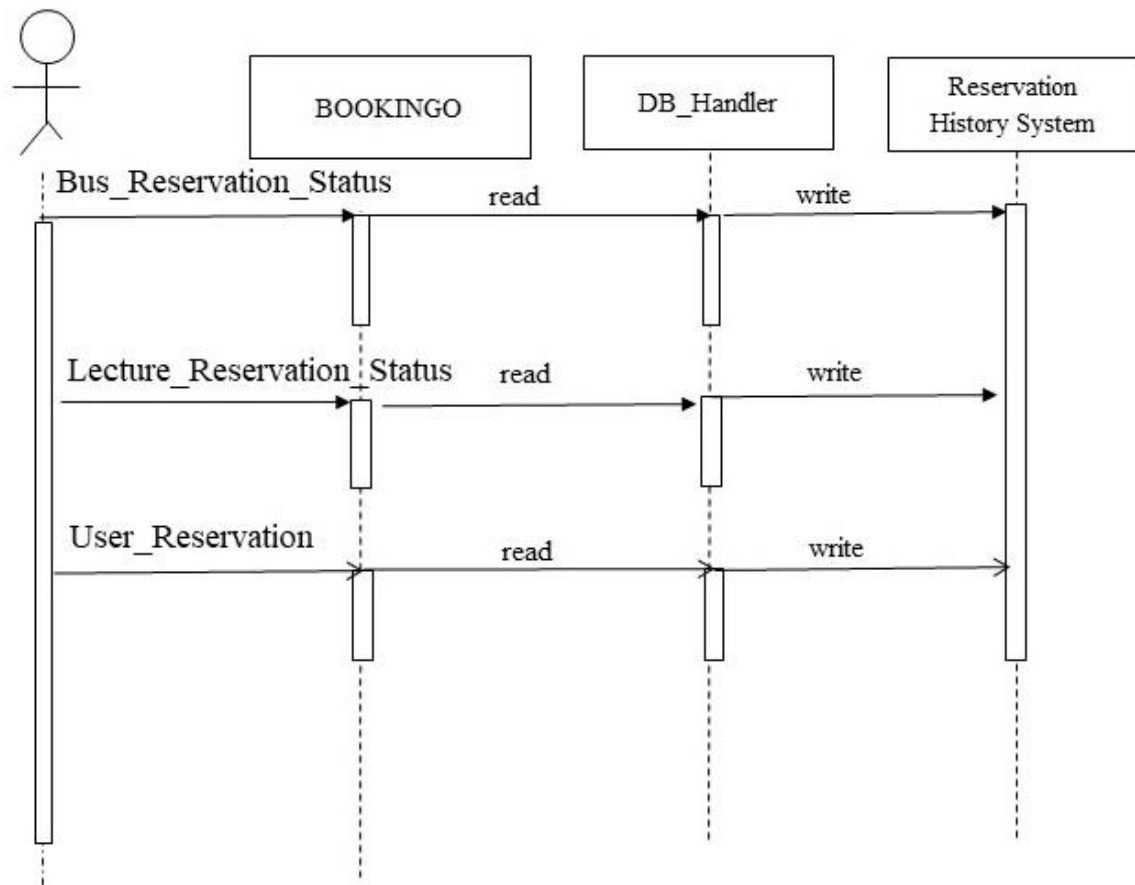


[Figure 18] Class diagram – Reservation History System

- Class Description

Reservation History System: 각 예약의 최소 생성 및 변경사항이 있을 때 마다 해당 내역을 반영하는 역할을 한다. 예약 변경/취소 내역 등을 reservation check기능을 통해 확인할 수 있다.

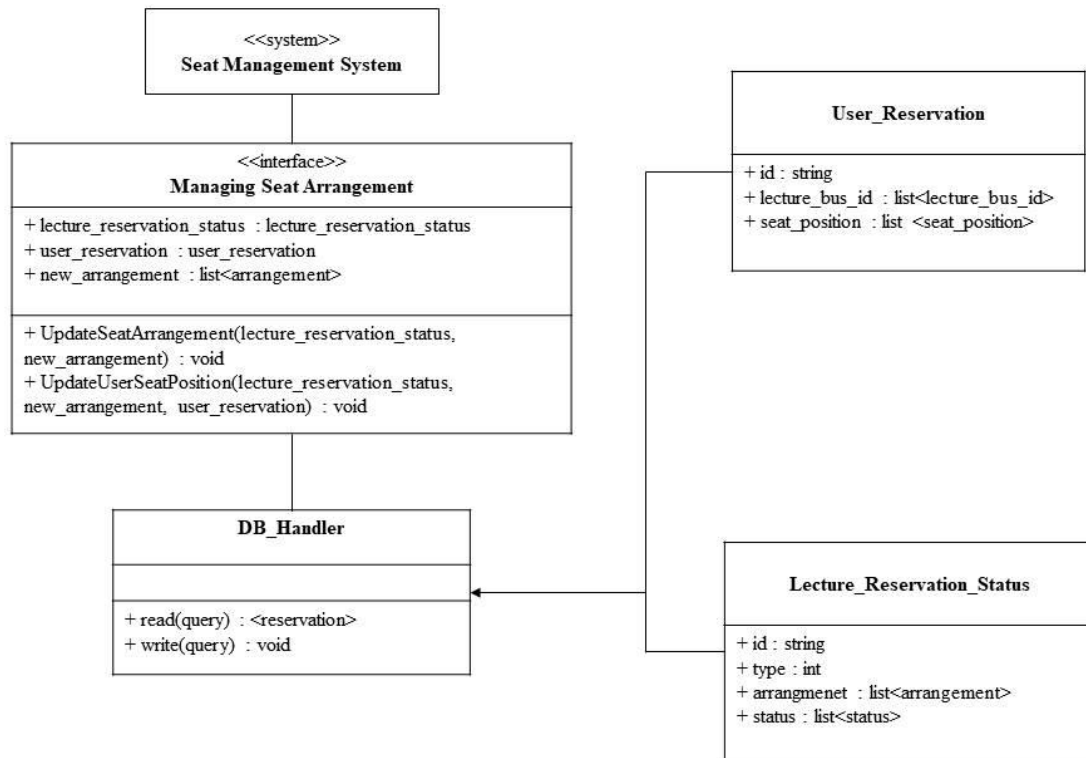
5.3.3.2. Sequence Diagram



[Figure 19] Sequence diagram – Reservation History System

5.3.4. Seat Management System

5.3.4.1. Class Diagram

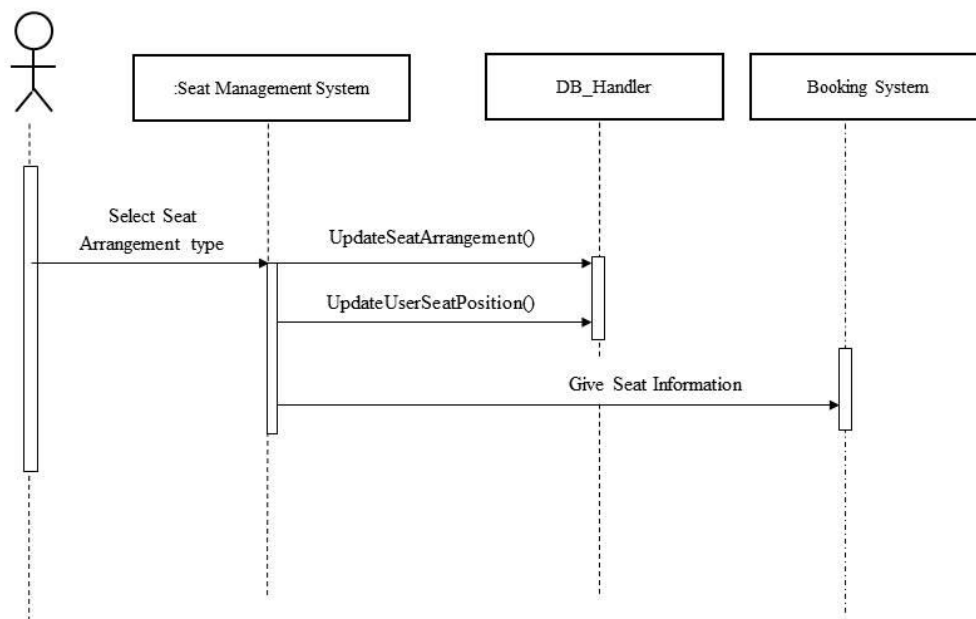


[Figure 20] Class diagram – Seat Management System Class

- Class Description

Seat Management System Class: 교직원의 예약을 돕는 시스템이다. 교직원이 시스템에 예약과 관련된 활동을 하면 시스템은 유저의 요청이 끝나는 순간 데이터베이스를 업데이트해준다. 동시에 해당 예약 사항에 관한 강의실의 좌석 예약 상태를 업데이트한다.

5.3.4.2. Sequence Diagram



[Figure 21] Sequence diagram – Seat Management System

6. Protocol Design

6.1. Objectives

이 장은 각 시스템 간 상호작용에 쓰일 프로토콜의 구조와 각 인터페이스가 어떻게 정의되었는지에 대해서 다룬다.

6.2. JSON

JSON(JavaScript Object Notation)은 키-값의 한 쌍으로 이루어진 데이터를 전달하기 위해 텍스트를 사용해서 표현하는 개방형 표준 포맷이다.

Firebase의 실시간 데이터베이스에서 모든 데이터는 JSON 객체로 저장되기 때문에 데이터베이스를 JSON 트리로 볼 수 있다. 이 JSON 트리는 SQL 데이터베이스와는 다르게 테이블이나 레코드가 존재하지 않는다. JSON 트리에 데이터를 추가된 데이터는 이미 존재하는 JSON 구조에 연관된 키를 갖는 노드가 된다.

6.3. OAuth

OAuth란 인터넷 사용자들이 비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보에 대해 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 공통적인 수단으로 사용되는 개방형 표준이다. 이 방법은 여러 기업들이 사용하고 있으며 사용자들이 타사의 애플리케이션이나 웹사이트의 계정에 관한 정보를 공유할 수 있게 해준다.

Firebase 역시 OAuth 2.0을 이용하여 백엔드와 쉽게 연동하여 사용할 수 있다.

6.4. Log In

- Request

[Table 1] Log In request

| Attribute | Detail | |
|--------------|---------------|-------------------------------|
| Protocol | OAuth | |
| Request Body | Id | User Kingo Id |
| | Request Token | Token for OAuth |
| | User | Basic information of the user |

- Response

[Table 2] Log In response

| Attribute | Detail |
|--------------|-----------------------|
| Success Code | 200 OK |
| Failure Code | HTTP error code = 400 |

| | | |
|--------------------------|--------------|------------------|
| Success Response Body | Access Token | Token for access |
| | Message | Success message |
| Failure Response Body | Message | Fail message |

6.5. Reservation

6.5.1. Set Reservation

- Request

[Table 3] Set reservation request

| Attribute | Detail | |
|-----------|-----------------------|---------------------|
| Method | POST | |
| URI | /user/:id/reservation | |
| Parameter | User | User id |
| | Lecture Bus Id | 예약 대상 (강의, 버스) id |
| | Seat position | 좌석 위치 |
| Header | Authorization | User authentication |

- Response

[Table 4] Set reservation response

| Attribute | Detail | |
|-----------------------|----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 409 (Conflict) | |
| Success Response Body | Message | Success message |
| Failure Response Body | Message | Fail message |

6.5.2. Check Reservation

- Request

[Table 5] Check reservation request

| Attribute | Detail | |
|--------------|-----------------------|---------------------|
| Method | GET | |
| URI | /user/:id/reservation | |
| Request Body | id | id |
| Header | Authorization | User authentication |

- Response

[Table 6] Check reservation response

| Attribute | Detail | |
|-----------------------|-----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 404 (Not Found) | |
| Success Response Body | Message | Success message |
| Failure Response Body | Message | Not Reserved |

6.7. Change Reservation

- Request

[Table 7] Change reservation request

| Attribute | Detail | |
|-----------|-----------------------|---------|
| Method | POST | |
| URI | /user/:id/reservation | |
| Parameter | User | User id |

| | | |
|--------|----------------|---------------------|
| | Lecture Bus Id | 예약 대상 (강의, 버스) id |
| | Seat position | 좌석 위치 |
| Header | Authorization | User authentication |

- Response

[Table 8] Change reservation response

| Attribute | Detail | |
|-----------------------|----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 409 (Conflict) | |
| Success Response Body | Message | Success message |
| Failure Response Body | Message | Fail message |

6.8. Cancel Reservation

- Request

[Table 9] Cancel reservation request

| Attribute | Detail | |
|--------------|-----------------------|---------------------|
| Method | DELETE | |
| URI | /user/:id/reservation | |
| Request Body | User | User id |
| | Lecture Bus Id | 예약 대상 (강의, 버스) id |
| Header | Authorization | User authentication |

- Response

[Table 10] Reservation response

| Attribute | Detail | |
|-----------------------|-----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 404 (Not Found) | |
| Success Response Body | Message | Success message |
| Failure Response Body | Message | Fail message |

6.9. Seat Arrangement

6.9.1. Set Seat Arrangement

- Request

[Table 11] Set seat arrangement request

| Attribute | Detail | |
|--------------|-----------------------|---------------------|
| Method | POST | |
| URI | /seat/:id/arrangement | |
| Request Body | arrangement | list<arrangement> |
| Header | Authorization | User authentication |

- Response

[Table 12] Set seat arrangement response

| Attribute | Detail | |
|-----------------------|-----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 404 (Not Found) | |
| Success Response Body | Message | Success message |

| Attribute | Detail | |
|-----------------------|---------|--------------|
| Failure Response Body | Message | Fail message |

6.9.2. Get Seat Arrangement

- Request

[Table 13] Get seat arrangement request

| Attribute | Detail | |
|--------------|-----------------------|---------------------|
| Method | GET | |
| URI | /seat/:id/arrangement | |
| Request Body | id | lecture / bus id |
| Header | Authorization | User authentication |

- Response

[Table 14] Get seat arrangement response

| Attribute | Detail | |
|-----------------------|-----------------------------------|-----------------|
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 404 (Not Found) | |
| Success Response Body | Message | Success message |
| Failure Response Body | Message | Fail message |

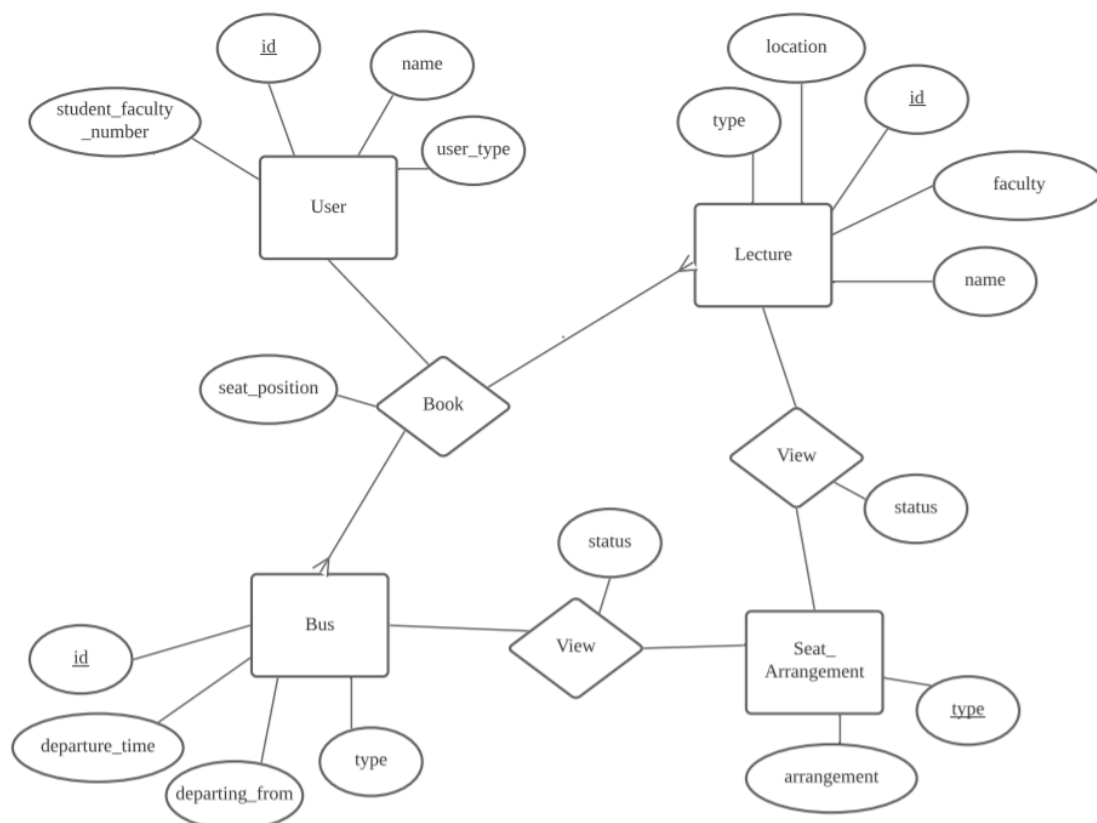
7. Database Design

7.1. Objectives

Database Design은 요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 전체적인 Entity 간의 관계를 표현한다. 그리고 Relational Schema, SQL DDL 명세를 만든다.

7.2. ER Diagram

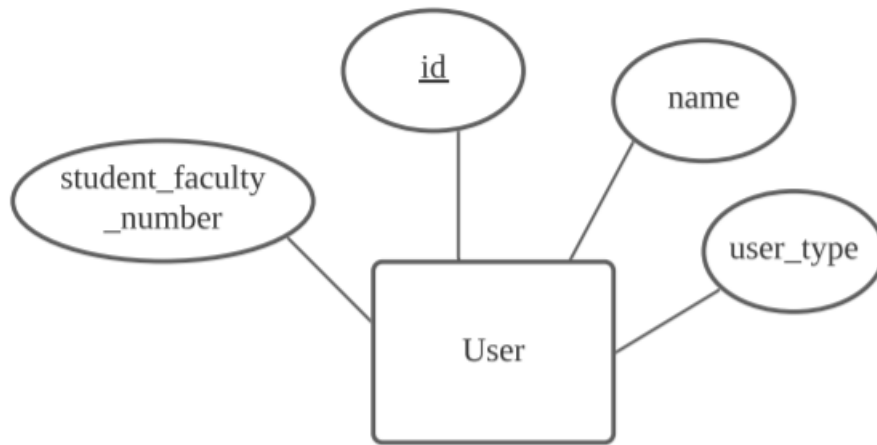
BOOKINGO는 User, Lecture, Bus, Seat_Arrangement로 총 4개의 Entity가 존재한다. ER Diagram에서 각각의 Entity는 네모 박스로 표현되고 Entity간의 관계는 마름모 형태로 표현된다. 특정 Entity가 다른 entity와 복수(1:M)관계를 가질 수 있을 때에는 해당 entity 쪽에는 삼지창 모양으로 선을 세 개 그어 나타낸다. 각 entity가 가질 수 있는 속성들은 타원으로 표현된다. 모든 개체는 객체를 고유하게 식별할 수 있는 특성 집합을 가지고 있어야 하며 최소한의 고유 식별 집합을 primary key라고 부른다. 이 Primary key attribute는 타원 안에 밑줄을 그어 표현한다.



[Figure 22] Overall ER diagram

7.2.1. Entities

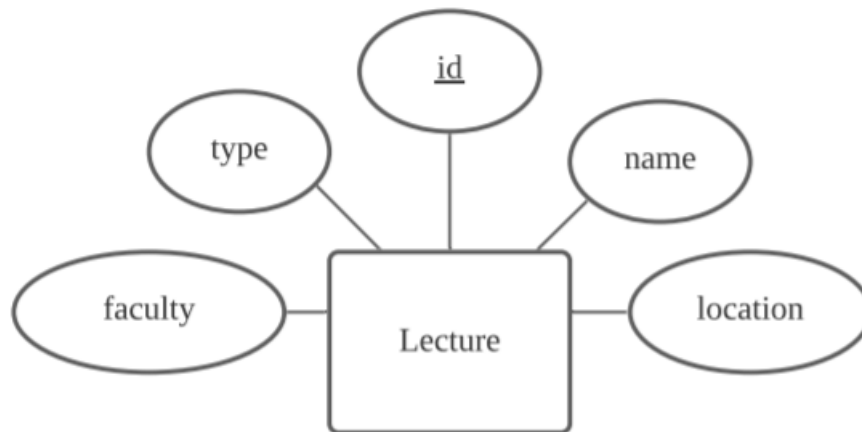
7.2.2.1. User



[Figure 23] ER diagram – User Entity

User Entity는 사용자의 정보를 나타낸다. BOOKINGO의 User는 학생과 교직원으로 구성되어 있다. User는 기본적으로 id, name, user_type, student_faculty_number 속성들을 가지고 있다. skku 포털로 로그인하는 아이디인 id가 primary key로 객체 식별에 이용된다. Name은 회원의 이름을 의미한다. User_type은 사용자가 학생인지, 교직원인지 구분하기 위한 변수다. Student_faculty_number은 아이디와 별개로 학번, 교직원 번호를 의미한다. Primary key가 될 수 있는 집합으로는 {student_faculty_number}도 있지만 BOOKINGO는 아이디를 이용해서 예약을 하기 때문에 id를 Primary key로 선정했다.

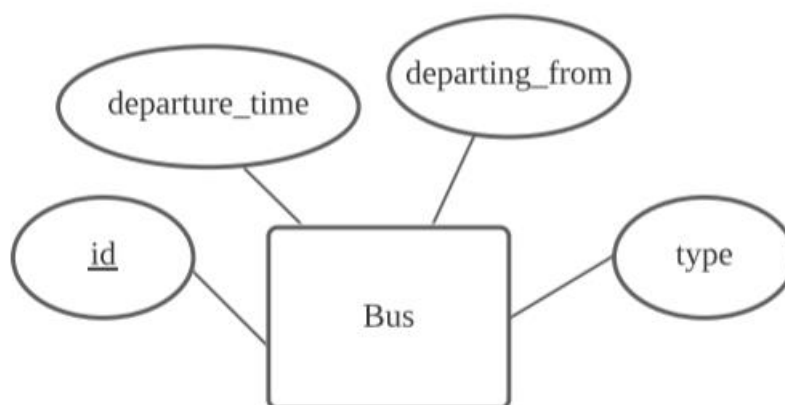
7.2.2.2. Lecture



[Figure 24] ER diagram – Lecture Entity

Lecture Entity는 강의실 정보를 표현한다. Lecture은 id, name, type, location, faculty를 기본적으로 속성들로 가지고 있다. Id는 강의마다 가지고 있는 고유 번호다. Name은 강의 이름, faculty는 담당 교직원 이름을 뜻한다. Time은 강의 시간, location은 강의 장소 정보를 담고 있다. Type는 강의실 좌석 배치 유형 정보를 담고 있다. Lecture마다 가지고 있는 고유 id가 Lecture Entity의 primary key다.

7.2.2.3. Bus

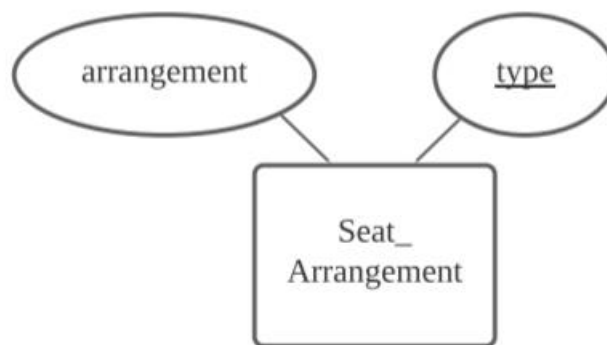


[Figure 25] ER diagram - Bus

Bus Entity는 셔틀 버스의 기본 정보를 담고 있는 객체다. Bus는 id, departure_time,

departing_from, type을 기본적으로 속성으로 갖고 있다. Id는 버스마다 고유로 가지고 있는 번호를 의미한다. 셔틀 버스의 출발날짜와 시간 정보는 departure_time에 모두 들어간다. Departing_from은 셔틀 버스가 자과캠 출발인지, 인사캠 출발인지 구별할 수 있는 정보가 담겨 있다. 마지막으로 type은 버스의 좌석 형태 유형 정보가 담겨져 있다. Bus Entity 역시 버스마다 고유로 구별할 수 있는 id가 primary key다.

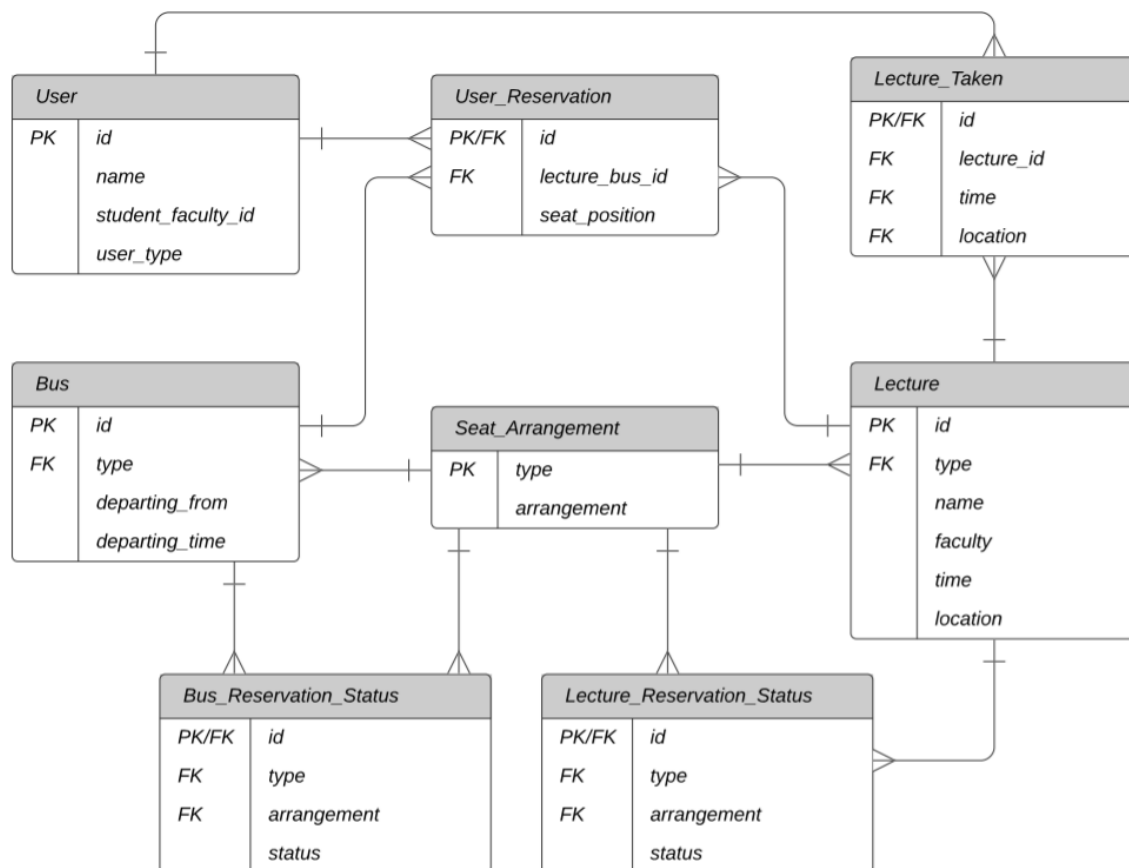
7.2.2.4. Seat_Arrangement



[Figure 26] ER diagram – Seat_Arrangement

Seat_Arrangement Entity는 좌석의 배치 유형과 배치된 좌석의 위치 정보를 담고 있다. Type은 강의실 및 버스 좌석의 배치를 구분할 수 있는 유형 번호다. Arrangement는 각 type마다 가지고 있는 좌석 배치 형태 정보를 담고 있다. Seat_Arrangement Entity에서 type이 primary key다.

7.3. Relational Schema



[Figure 27] Relational Schema

7.4. SQL DDL

7.4.1. User

[Table 15] SQL DDL – User Table

```

CREATE TABLE User
(
    id VARCHAR(30) NOT NULL,
    name VARCHAR(10) NOT NULL,
    student_faculty_id VARCHAR(30) NOT NULL,
    user_type VARCHAR(30) NOT NULL,
    PRIMARY KEY (id)
);
  
```

7.4.2. Lecture

[Table 16] SQL DDL – Lecture Table

```
CREATE TABLE Lecture
(
    id VARCHAR(30) NOT NULL,
    name VARCHAR(30) NOT NULL,
    faculty VARCHAR(10) NOT NULL,
    time INT NOT NULL,
    location VARCHAR(30) NOT NULL,
    type STRING NOT NULL
    PRIMARY KEY (id),
    FOREIGN KEY(type) REFERENCES seat_arrangement(type)
);
```

7.4.3. Bus

[Table 17] SQL DDL – Bus Table

```
CREATE TABLE Bus
(
    id VARCHAR(30) NOT NULL,
    departing_from VARCHAR(30) NOT NULL,
    departing_time INT NOT NULL,
    type STRING NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(type) REFERENCES Seat_Arrangement(type)
);
```

7.4.4. Seat_Arrangement

[Table 18] SQL DDL – Seat_Arrangement Table

```
CREATE TABLE Seat_Arrangement
```

```
(  
    type STRING NOT NULL,  
    arrangement STRING NOT NULL,  
    PRIMARY KEY (type)  
);
```

7.4.5. Bus_Reservation_Status

[Table 19] SQL DDL – Bus_Reservation_Status Table

```
CREATE TABLE Bus_Reservation_Status  
(  
    id VARCHAR(30) NOT NULL,  
    type STRING NOT NULL,  
    arrangement STRING NOT NULL,  
    status STRING NOT NULL,  
    PRIMARY KEY(id),  
    FOREIGN KEY(id) REFERENCES Bus(id),  
    FOREIGN KEY(type) REFERENCES Seat_Arrangement(type),  
    FOREIGN KEY(arrangement) REFERENCES  
        Seat_Arrangement(arrangement)  
);
```

7.4.6. Lecture_Reservation_Status

[Table 20] SQL DDL – Lecture_Reservation_Status Table

```
CREATE TABLE Lecture_Reservation_Status  
(  
    id VARCHAR(30) NOT NULL,  
    type STRING NOT NULL,  
    arrangement STRING NOT NULL,  
    status STRING NOT NULL,
```

```
PRIMARY KEY(id),  
FOREIGN KEY(id) REFERENCES Lecture(id),  
FOREIGN KEY(type) REFERENCES Seat_Arrangement(type),  
FOREIGN KEY(arrangement) REFERENCES  
Seat_Arrangement(arrangement)  
);
```

7.4.7. Lecture_Taken

[Table 21] SQL DDL – Lecture_Reservation_Status Table

```
CREATE TABLE Lecture_Reservation_Status  
(  
id VARCHAR(30) NOT NULL,  
lecture_id VARCHAR(30) NOT NULL,  
time INT NOT NULL,  
location VARCHAR(30) NOT NULL,  
PRIMARY KEY(id),  
FOREIGN KEY(id) REFERENCES User(id),  
FOREIGN KEY(lecture_id) REFERENCES Lecture(id),  
FOREIGN KEY(time) REFERENCES Lecture(time),  
FOREIGN KEY(location) REFERENCES Lecture(location)  
);
```

7.4.8. User_Reservation

[Table 22] SQL DDL – Lecture_Reservation_Status Table

```
CREATE TABLE User_Reservation  
(  
id VARCHAR(30) NOT NULL,  
lecture_bus_id VARCHAR(30) NOT NULL,  
seat_position STRING NOT NULL,
```

```
PRIMARY KEY(id),  
FOREIGN KEY(id) REFERENCES User(id),  
FOREIGN KEY(lecture_bus_id) REFERENCES Lecture(id) and  
Bus(id)  
);
```

8. Testing Plan

8.1. Objectives

Testing 기획 및 계획의 목적은 시스템 개발 중간 또는 이후 단계에서 나타날 수 있는 결함 및 오류를 찾아내어 이를 수정하고, 또 시스템이 당초 기획한 바와 같이 개발이 이루어졌는지, 또 그 성능이 적당한지를 평가하기 위함이다. 더불어, 사전에 기획된 testing plan은 개발에 있어, 일정부분 가이드라인을 제공하게 될 것이다. Testing plan은 크게 development testing, release testing, 그리고 user testing 세 가지 단계로 나누어 볼 수 있으며, 각각 중점적으로 평가하고자 하는 바가 상이하다.

8.2. Testing Policy

8.2.1. Development Testing

Development Testing은 개발 단계 도중에 이루어지는 만큼, 1차적으로 각 sub system의 평가에 중점을 두며, 개발 진척에 따라 integration testing도 진행하게 된다. Development testing 단계에서는, 시스템에서 꼭 달성해야 할 usability, performance, dependability 그리고 security를 평가 준거로 삼아 testing을 진행할 것이다.

8.2.1.1. Usability

BOOKINGO 어플리케이션 내 예약 과정에서 사용자의 불편함을 최소화해야 하며, 예약 관련 세부사항이 화면에 명확하게 나타나야 한다. 이를 위해 user testing에서 20명 내외의 테스터를

선정하여, 어플리케이션의 사용자 경험을 설문하고자 한다.

8.2.1.2. Performance

BOOKINGO 어플리케이션 내에서 다수의 사용자가 동시에 접속하여 실시간으로 예약을 진행하는 만큼, 예약 현황을 지체 없이 업데이트 하는 것이 매우 중요하다. 사용자의 예약에 관한 활동(예약, 변경 취소)을 즉각적(10초 이내)으로 반영하기 위해, 다양한 test case를 준비하여 각 예약 기능 및 좌석 배치 기능의 반영 속도를 측정하고, 서버와의 소통이 효율적으로 이루어졌는지 평가하고자 한다.

8.2.1.3. Dependability

시스템이 안정적으로 작동하게끔 하기 위해, BOOKINGO 내 sub system과 각 구성요소가 올바르게 존재하는지, 또 알맞게 연결되었는지 확인하는 것이 필요하다. 이를 위해 개발 중, development testing 단계에서 unit, component testing과 integration testing을 진행하고자 한다.

8.2.1.4. Security

BOOKINGO 어플리케이션 내에서 처리하게 되는 정보는 학내 구성원의 이동 및 수업 정보를 포함하게 되며, 이러한 민감한 개인 정보를 철저히 보호할 수 있어야만 한다. 이를 위해 개발 중 development testing을 통해 login/out 부분의 security를 component testing 및 앱과 성균인 DB의 integration testing을 진행해 보안성을 확인하고자 한다.

8.2.2. Release Testing

최종적으로 시스템의 delivery를 진행하기 전 또는, 다른 팀에게 전달하기 전 특정 릴리즈 버전의 시스템에 대해 testing을 진행하는 단계이다. BOOKINGO 어플리케이션의 경우, user testing을 위한 알파 및 베타 테스터 전달 전, 그리고 최종 사용자에게 어플리케이션을 배포하기 전 release testing을 진행한다.

8.2.3. User Testing

최종 사용자 중 일부, 또는 전체를 통해 어플리케이션 사용 경험에 관한 조언, 평가 및 피드백

을 받는 testing이다. BOOKINGO 어플리케이션 개발 과정에서는 20-30인 이내의 알파/베타 테스트를 진행하고자 하며, 최종 릴리즈 이후 사용자에게 acceptance test를 진행하게 된다.

8.2.4. Testing Case

위에서 언급한 4가지 emergent property(dependability, security, performance, usability)를 기반으로 test case를 설정하고자 하며, 각 property당 3-4개의 test case를 배당, 최대 20개의 testing case를 진행 및 평가하게 될 것이다.

9. Development Plan

9.1. Objectives

개발 환경 및 개발 단계에서 활용할 도구, API등을 기술한다.

9.2. Frontend Environment

9.2.1. Adobe Illustrator



[Figure 28] Adobe Illustrator 로고

Adobe Illustrator 툴을 통해 BOOKINGO 어플리케이션 내에서 필요한 각종 이미지, UI 요소를 디자인하게 된다.

9.2.2. Adobe Xd



[Figure 29] Adobe Xd 로고

웹 및 모바일 어플리케이션용 UX/UI 디자인 툴이다. BOOKINGO 개발에 있어, 전체적인 UI 구성을 위 툴을 활용할 것이다. 또한 작동하는 UI를 직관적으로 빠르게 생산해 낼 수 있다는 점에서 프로토타입 UI 구성에 있어 효율적이기도 하다.

9.2.3. Android Studio



[Figure 30] Android Studio 로고

BOOKINGO 시스템의 개발은 안드로이드 어플리케이션 개발에 일반적으로 사용하는 Android Studio IDE를 활용하게 된다. 언어는 Kotlin을 사용하며, style guide 또한 Kotlin을 기준으로 한다. Android Studio의 사용을 통해, 해당 IDE에서 제공되는 다양한 도구를 활용할 수 있으며, 또한 firebase와의 연동 또한 용이하다.

9.3. Backend Environment

9.3.1. Firebase



[Figure 31] Firebase 로고

Firebase는 클라우드 저장, 실시간 데이터베이스, 머신러닝 등의 기능을 제공하는 모바일 및 웹 어플리케이션이다. 이중, 클라우드 저장 및 실시간 데이터베이스 관리를 중점적으로 활용하여 예약 서비스를 이용자에게 제공하고자 한다. BOOKINGO에서는 예약 내역의 실시간 관리가 중요한 만큼, Firebase의 기능을 효율적으로 활용할 수 있을 것이다.

9.4 Tools and APIs

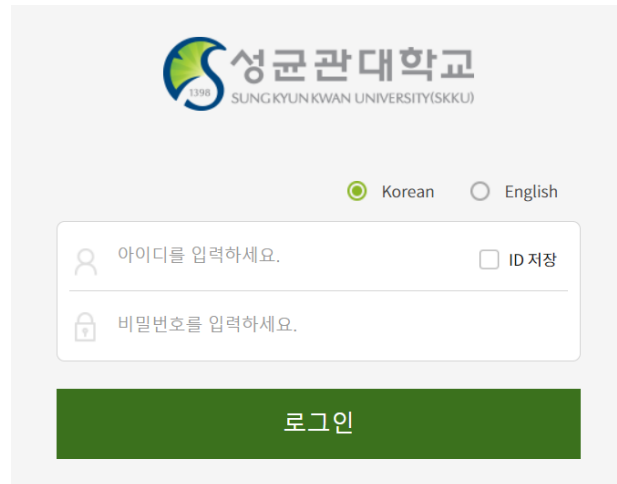


[Figure 32] Github 로고

9.4.1. Github

Github를 활용하여 BOOKINGO의 각 개발 및 릴리즈 버전 관리를 편리하게 하는 것이 가능하며, 이는 개발부터 각 sub-system 통합, 그리고 release testing까지 여러 부분에 걸쳐 편리성을 제공하기 된다. 또한 협업 기능을 통해 팀원간 소통을 도울 수 있다.

9.4.2. SKKU Member API



[Figure 33] 성균인 로그인 화면

교내 학술정보관, GLS, ASIS, 일자리 센터 등 교내 서비스를 이용할 때 학내 구성원임을 인증하는 API이다.

9.5. Constraint

시스템은 여러 가지 제약사항을 지키면서 서술된 내용을 바탕으로 구현되어야 한다.

- 시스템은 단일 터미널만 지원한다.
- 시스템이 원활하게 동작하기 위한 모바일 최소 사양은 1GB RAM과 512MB의 저장공간이 있어야 한다.
- 예약에 변동이 있다면 10초 이내에 데이터베이스가 업데이트되어야 한다.
- 좌석 배치 UI는 3초 이내에 로드되어야 한다.
- 모든 데이터는 JSON 객체의 형태로 저장되며 Data type은 Integer, String, List가 있다.
- Firebase가 따르는 Apache License를 따라야 한다.
- Google Developers에서 배포하는 Kotlin Style Guide를 따라 코드를 작성한다.
- 규제정책으로 저작권법을 준수해야 한다.
- 성균관대학교 구성원만 사용할 수 있으므로 로그인 시 교내 구성원임을 엄격하게 확인한다.
- 시스템은 상시 네트워크에 연결되어 예약 현황을 업데이트해야 한다.

- 사용자의 개인정보가 외부로 유출되어서는 안되며 개인정보를 수집하기 전 동의를 반드시 구한다.

9.6. Assumptions and Dependencies

이 문서의 서비스들은 모두 안드로이드 플랫폼임을 가정한다. 따라서 안드로이드 운영체제를 지원하지 않고 있는 모바일 기기에서는 작동이 어려울 수 있다. DBMS로는 firebase를 사용하고 client측 인터페이스를 작성할 때 Android Studio IDE를 사용한다. 교내 서비스이므로 교내 데이터베이스에 크게 의존하고 있다. 강의실 정보, 교직원 정보, 학생 정보 등을 참고할 수 있어야 한다.

10. Supporting Information

10.1. Software Design Specification

위 소프트웨어 설계 명세서는 IEEE 권고사항 (IEEE Recommended Practice for Software Design Description, IEE-Std1016)을 참고하여 작성하였다.

10.2. Document History

[Table 23] Document History

| # | Date | Description | Writer |
|-----|-----------|---------------------------------------|-----------------------|
| 0 | 2021/5/3 | 개요 작성 / 회의 | 강종현, 이승우, 임서현, 임재현 |
| 1.1 | 2021/5/6 | 4.2.1, 4.2.5. | 강종현 |
| 1.2 | 2021/5/6 | 4.2.2, 5.2.2. | 이승우 |
| 1.3 | 2021/5/6 | 4.2.4., 5.2.1. | 임서현 |
| 1.4 | 2021/5/6 | 4.2.3., 5.2.3. | 임재현 |
| 1.5 | 2021/5/15 | 1. Preface, 2. Intro, 3. Architecture | 강종현 |
| 1.6 | 2021/5/15 | 6. Protocol Design | 이승우 |

| | | | |
|-----|-----------|--------------------------------------|-----------------------|
| 1.7 | 2021/5/15 | 7. Database Design | 임서현 |
| 1.8 | 2021/5/15 | 8. Testing Plan, 9. Development Plan | 임재현 |
| 1.9 | 2021/5/16 | Design Specification Final | 강종현, 이승우, 임서현, 임재현 |