BOOKINGO

Software Test Plan Document

2021.5.30.

Introduction to Software Engineering

TEAM 1

팀장 임서현

조원 강종현

조원 이승우

조원 임재현

목차

List of Tables

1. Introduction

1.1. Purpose

본 문서는 학교 시설 내 셔틀 버스와 강의실 좌석 예약 서비스를 제공하기 위한 시스템을 테스트하는 계획을 기술하고 있다. 해당 문서는 성균관대학교 2021학년도 1학기 소프트웨어공학개론 Team1(강종현, 이승우, 임서현, 임재현)에 의하여 작성됐다. 본 문서의 Readership는 소프트웨어를 개발하는 Team1, 개발을 지도하는 지도교수, 그리고 본 소프트웨어의 요구사항을 파악하고 싶은 사용자들이다.

1.2. Scope

본 문서에서 서술할 시스템의 이름은 BOOKINGO이다. BOOKINGO는 기존 학교 앱에 없는 셔틀 버스와 강의실 좌석 예약 시스템을 운영해서 학생과 교직원이 느꼈던 불편한 점을 해소해주는 것을 목표로 한다. BOOKINGO는 크게 셔틀 좌석 예약, 강의실 좌석 예약으로 구성되어 있고 학교 구성원임을 인증한 유저는 두 서비스를 자유롭게 사용할 수 있다.

1.3. Definitions, Acronyms, and Abbreviation

다음 테이블은 이 문서에서 사용된 기술적 용어들의 정의를 정리했다.

[Table 1] Definition

Terms	Definitions
서버(Server)	클라이언트에게 네트워크를 통해 정보나 서비스를 제공하는 컴퓨터 또는 프로그램
클라이언트(Client)	네트워크를 통하여 서버라는 다른 컴퓨터 시스템 상의 원격 서비스에 접속할 수 있는 응용 프로그램이나 서비스
사용자(User)	서비스를 이용하는 사람. 본 서비스에서는 학생 및 교직원이 있다.
시스템 관리자 (System Administrator)	서비스를 관리하는 사람. 예약 시스템을 모니터링하고 예약 창을 관리한다.

시스템(System)	BOOKINGO를 system으로 칭하며 문서에 서술되지 않은 BOOKINGO의 모든 기능을 포함한다.
데이터베이스 (Database)	System에서 사용되는 data의 집합으로 일반적으로 컴퓨터 시스템에 저장된 구조화된 정보 또는 데이터의 체계적인 집합을 의미한다.
클라우드	인터넷 기반의 컴퓨팅으로 인터넷 통신망에 접속해 컴퓨팅 자원을 원하는 대로 가져다 쓸 수 있는 서비스를 의미한다.
클라우드 데이터베이스	Firebase와 같은 클라우드 컴퓨팅 플랫폼에 상주하는 데이터베이스
의사코드 (Pseudocode)	프로그램의 각 모듈이 작동하는 논리를 간략하게 표현한 언어로 프로그래밍 언어의 문법을 본뜬 모양으로 만든 코드다.
API	당사자들 간 계약을 나타내는 문서를 갖출 계약으로 비유되는 어플리케이션 간의 인터페이스.
핑(Ping)	네트워크를 통해 상대에게 접근할 수 있는가를 확인하기 위한 프로그램
쓰레드(Thread)	프로세스가 할당 받은 자원을 이용하는 실행 흐름의 단위
데이터 패킷(Packet)	통신망으로 전송하기 쉽게 작은 단위로 쪼갠 데이터.
브로드캐스트 (Broadcast)	로컬 랜에 붙어있는 모든 네트워크 장비들에게 보내는 통신
쿼리(Query)	데이터베이스에 정보를 요청하는 것
타임아웃(Timeout)	작업이 완료되기 전에 경과했거나 서버가 응답하지 않은 시간 초과 기간
데이터 스토리지 (Data Storage)	텍스트 파일, 이미지, 영상 등 다양한 종류의 데이터가 저장될 수 있는 저장 매체이자 자료 저장소.

다음 테이블은 머리글자(acronym) 및 약어(abbreviation)들의 정의를 정리했다.

[Table 2] Acronyms & Abbreviations

Acronyms & Abbreviations	**Explanation**

UI	User Interface
DB	Database
DBMS	Database Management System
OS	Operating System
GB	Gigabyte
MB	Megabyte
JSON	JavaScript Object Notation
N/A	Not Available or Not Applicable
API	Application Programming Interface
REST API	Representational State Transfer Application Programming Interface
HTTPS	Hypertext Transfer Protocol Secure

1.4. References

- ▷ Ariya.io, “Practical Testing of Firebase Projects”, Last updated: April 29, 2020. https://ariya.io/2020/04/practical-testing-of-firebase-projects
- ▷ CredibleSoft, “Ecommerce Websites Testing Guide Tutorial”, Last Updated: March 10th, 2021. https://crediblesoft.com/ecommerce-websites-testing-guide-tutorial/
- ▷ Damian T.Gordon, “Pseudocode: Software Testing”, Last Updated: September 11, 2015. https://www.youtube.com/watch?v=_jz0B7HnjAY
- ▷ Firebase, “Realtime Database Limits”, https://firebase.google.com/docs/database/usage/limits
- ▷ Firebase, “Firebase Database REST API”, https://firebase.google.com/docs/reference/rest/database
- ▷ Guru99, “What is Interface Testing? Types & Example”, https://www.guru99.com/interface-testing.html
- ▷ Jmpovedar, “Test Plan Outline(IEEE 829 Format)”, https://jmpovedar.files.wordpress.com/2014/03/ieee-829.pdf
- ▷ Kenneth Leroy, “Programming Fundamentals - A Modular Structured Approach using C++”, Openstax CNX, https://cnx.org/contents/MDgA8wfz@22.2:YzfkjC2r@17/Preface
- ▷ Kenneth Leroy Busbee, “Programming Fundamentals: Software Testing”, Rebus Community,

https://press.rebus.community/programmingfundamentals/chapter/software-testing/

▷ Kenneth Leroy Busbee, “programming Fundamentals: Pseudocode”, Rebus Community, https://press.rebus.community/programmingfundamentals/chapter/pseudocode/

▷ Milecia, “How to write pseudo code”, Last Updated: August 27, 2019. https://dev.to/flippedcoding/how-to-write-pseudo-code-2jfe

▷ StrongQA, “Test Case”. https://strongqa.com/qa-portal/testing-docs-templates/test-case

▷ Tutorialspoint, “Interface Testing”, https://www.tutorialspoint.com/software_testing_dictionary/interface_testing.htm

▷ Wikipedia, “Software testing “, Last updated: May 14, 2021. https://en.wikipedia.org/wiki/Software_testing

1.5. Overview

본 Test Plan Document 문서는 5개의 section으로 구성된다. Introduction 섹션에서는 문서의 개요로서 본 문서의 목적과 대상독자, 시스템의 범위, 문서를 이해하기 위해 필요한 용어의 정의 및 뜻, 참조문서 그리고 문서의 구성에 대해서 다룬다. 두 번째 section인 Approach에서는 종합적 기술로서 unit test와 interface test의 test methods에 대해 간략하게 기술한다. 세 번째 Software Unit Test에서는 BOOKINGO의 product function에 따른 각 unit test에 대해 기술한다. 네 번째 Software Interface Test는 BOOKINGO 시스템 외부의 API와의 인터페이스를 기술하고 그 인터페이스를 테스트하는 케이스들에 대해 알아본다. 마지막으로 Supporting Information section에서는 문서 업데이트 타임라인을 알 수 있는 document history를 기술한다.

2. Approach

2.1. Test Method

2.1.1. Software Unit Test Methods

2.1.1.1. Login & Out

유저가 키보드로 입력한 아이디와 비밀번호를 SKKU DB에서 query authorization으로 정보를

확인한 후 검색 및 정보가 확인되면 로그인이 성공한다.

해당 기능의 Pseudocode는 다음과 같다.

Input

유저의 아이디를 화면 키보드로 입력 받는다.
유저의 비밀번호를 화면 키보드로 입력 받는다.

Processing

입력 받은 아이디 정보를 통해 SKKU DB에 등록된 아이디인지 확인한다.
올바른 아이디면 해당 유저의 비밀번호가 맞는지 확인한다.

Output

유저 정보를 authorization 성공하면 로그인 성공 메시지를 띄운다.
유저에게 메인 홈 화면을 보여준다.

2.1.1.2. Booking Shuttle Bus Seat, Lecture Seat

유저가 예약 화면에서 다양한 기능을 터치하며 선택을 하면 그에 따라 데이터베이스 업데이트 요청을 하게 된다.

해당 기능의 Pseudocode는 다음과 같다.

Input

유저가 어떤 예약 서비스를 이용할 지 터치를 통해 입력 받는다.
유저가 예약할 세부 정보를 터치를 통해 입력 받는다.
유저가 예약할 좌석을 터치를 통해 입력 받는다.

Processing

입력 받은 예약 서비스를 이용해 해당 서비스의 세부 정보를 불러온다.
입력 받은 세부 정보를 이용해 좌석 배치 화면을 불러온다.

Output

예약이 성공하면 예약 완료 메시지를 띄운다.
예약이 실패할 경우 예약 실패 메시지를 띄운다.

2.1.1.3. Arranging Lecture Room Seating

사용자인 교직원은 좌석을 배치할 수 있는 화면에서 다양한 기능을 터치하며 선택을 한다. 선택이 완료되면 서버는 데이터베이스 업데이트 요청을 보내게 된다.

해당 기능의 Pseudocode는 다음과 같다.

Input

사용자가 배치 혹은 배정하고자 하는 강의와 해당 강의의 일정을 선택한다.
선택한 강의와 일정에 대해 좌석 배치/수동 배정/랜덤 배정 중 원하는 작업의 버튼을 터치한다.
좌석 배열에 수정이 필요한 경우 좌석 배치 버튼을 터치하고 드래그를 통해 수정을 진행한다.
좌석 형태에 수정이 필요한 경우 좌석 배치 버튼을 터치하고 좌석을 선택한 다음 팝업에서 원하는 형태를 선택한다.
수동으로 학생을 배정하고 싶은 경우 수동 배정 버튼을 터치하고 좌석과 학생을 선택한다.
자동으로 학생을 배정하고 싶은 경우 랜덤 배정 버튼을 터치한다.

Processing

좌석 배치버튼을 터치한 경우 변경된 좌석 배열 혹은 좌석 형태를 저장한다.
수동 배정버튼을 터치한 경우 사용자가 선택한 좌석과 학생의 정보를 저장한다.
랜덤 배정버튼을 터치한 경우 해당 강의와 학생의 정보를 통해 임의로 좌석을 부여하고 저장한다.

Output

좌석 배치/수동 배정/랜덤 배정을 각각 정상적으로 완료한 경우 ‘정상적으로 처리되었습니다’라는 메시지를 출력한다.

2.1.1.4. Checking Reservation

유저가 메인 홈 화면이나 개별 예약 내역을 확인한다는 요청을 보내면 서버에서 예약 내역을 불러온다.

해당 기능의 Pseudocode는 다음과 같다.

Input

사용자가 전체 예약 내역을 확인하고 싶다면 “예약 확인” 탭을 터치한다.
사용자가 개별 예약 세부사항을 확인하고 싶다면 개별 예약 내역을 터치한다.
사용자가 당일 개별 예약을 확인하고 싶다면 메인 화면에서 “Today”란의 개별 예약 내역을 터치한다.

Processing

사용자의 전체 예약 내역을 불러온다.
개별 예약의 세부사항을 불러온다.
당일 개별 예약 내역의 세부사항을 불러온다.

Output

전체 예약 내역의 페이지를 표시한다.
개별 예약의 세부사항이 기재된 페이지를 표시한다.
당일 개별 예약의 세부사항이 기재된 페이지를 표시한다.

2.1.1.5. Changing/Cancelling Reservation

유저가 예약을 변경하거나 취소하겠다는 요청을 보내면 서버에서 기존 예약을 새로운 예약 상태로 업데이트하거나 취소한다.

해당 기능의 Pseudocode는 다음과 같다.

Input

사용자는 변경하고자 하는 개별 예약 내역을 예약 확인 리스트에서 터치한다.
변경하고자 하는 정보(출발지점, 시간, 좌석위치 등)을 변경한다.
취소를 희망하는 경우, “취소” 버튼을 터치한다.
“확인” 버튼을 터치하여 변경 내역을 최종적으로 저장한다.

Processing

사용자가 터치한 개별 예약 세부사항을 불러온다.
사용자가 변경할 수 있는 정보를 불러온다.
사용자가 변경한 예약 세부사항(변경 및 취소)을 새로이 전달한다.

Output

“변경이 완료되었습니다” 라는 성공 메시지를 표시한다.
예약 취소의 경우, “예약을 취소하였습니다”라는 성공 메시지를 표시한다.

2.1.2. Software Interface Test Methods

2.1.2.1. Firebase

BOOKINGO 데이터베이스는 Firebase에 있다. 유저의 업데이트 요청을 확인하면 데이터베이스에 읽기 또는 수정 요청을 보낸다.

Firebase의 pseudocode는 다음과 같다.

Input

시스템 서버가 유저의 업데이트 요청을 확인한다.
Firebase에 저장된 데이터베이스에 수정 요청을 보낸다.

Processing

변경된 내역에 따라 DB를 업데이트한다.

Output

DB 업데이트가 성공적으로 완료되면 성공 메시지를 띄운다.

2.1.2.2. SKKU API

SKKU API는 유저가 로그인할 때 유저의 정보를 확인할 때 필요하다. 로그인 요청을 확인하면 데이터베이스 읽기 요청을 보내 유저 정보를 확인한다.

SKKU API의 pseudocode는 다음과 같다.

Input

시스템 서버가 유저의 로그인 요청을 확인한다.
SKKU API에 저장된 데이터베이스에 읽기 요청을 보낸다.

Processing

요청에 따라 DB를 확인한다.

Output

읽기가 성공적으로 완료되면 성공 메시지를 띄운다.

3. Software Unit Test

이 장에서는 BOOKINGO의 기능들의 unit test cases에 대해 기술한다. BOOKINGO의 기능들은 다음과 같다.

- Login & Out
- Booking Shuttle Bus Seat, Lecture Seat
- Arranging Lecture Room Seating
- Checking Reservation
- Changing/Cancelling Reservation

Function의 actor, description을 살펴보고 기능을 수행하기 위한 전제조건들을 알아본다. 그리고 다양한 기능들을 테스트할 수 있는 테스트 케이스들을 알아본다.

3.1. Login & Out

Login & Out 기능을 사용하는 사용자는 학생과 교직원이 있다. 사용자는 로그인 시 성균인 아이디 및 비밀번호를 사용하여 접속한다. 만약 입력한 정보가 올바르다면, 메인 홈 화면으로 들어갈 수 있다. 사용자가 서비스 이용을 마치고 싶다면, 어플리케이션을 종료한다. 이 때 세션이 종료되면서 자동으로 로그아웃이 이루어진다.

Login & Out의 Precondition으로 사용자의 정보가 SKKU DB에 포함되어 있어야 한다. 그리고 로그아웃을 하기 위해서는 사용자가 현재 로그인 된 상태임을 가정하고 있다.

3.1.1. User Account Login Test Case

[Table 3] Login & out Unit Test – 로그인 성공

Test Case Name	**Login & out – 로그인 성공**
Test case object	올바른 로그인 정보가 들어왔을 때 작동을 확인
Test Inputs	SKKU DB에 등록된 로그인 정보(ex. jihol123)
Action	1. SKKU DB에 소속된 유저가 올바른 유저 아이디와 비밀번호를 넣는다. 2. 확인 버튼을 터치한다. 3. SKKU DB에 등록된 학생정보를 authorization 한다.

	4. 로그인이 성공한다.
Expected Results	로그인 성공 메시지를 띄운다. 로그인이 완료되면 메인 홈화면을 띄운다. 유저의 type에 맞게 화면이 띄워져야 한다.

[Table 4] Login & out Unit Test – 로그아웃 성공

Test Case Name	**Login & out – 로그아웃 성공**
Test case object	로그인 된 상태에서 로그아웃
Test Inputs	SKKU DB에 등록된 로그인 정보(ex. jihol123)가 올바르게 로그인 되어 있고 유저는 로그아웃 버튼을 터치한다.
Action	1. SKKU DB에 소속된 유저가 로그인 된 상태에서 로그아웃 버튼을 터치한다. 2. 확인 버튼을 터치한다. 3. 서버에 등록된 유저 세션을 끝낸다. 4. 로그아웃이 성공한다.
Expected Results	로그아웃 성공 메시지를 띄운다. 유저는 처음 로그인 화면을 띄운다.

3.1.2. User Account Login Test Case - Negative

[Table 5] Login & out Unit Test – 로그인 실패1

Test Case Name	**Login & out – 잘못된 아이디로 인한 로그인 실패**
Test case object	잘못된 아이디 정보가 들어왔을 때 작동을 확인
Test Inputs	SKKU DB에 등록되지 않은 아이디(ex. abcd)
Action	1. SKKU DB에 소속된 유저가 잘못된 유저 아이디와 비밀번호를 넣는다. 또는 SKKU DB에 소속되지 않은 유저가 로그인을 시도한다. 2. 확인 버튼을 터치한다. 3. SKKU DB에 등록된 학생정보를 authorization 한다. 4. query가 실패해서 로그인이 되지 않는다.
Expected Results	‘등록되지 않은 아이디입니다’ 메시지를 띄운다. 로그인이 되지 않는다.

[Table 6] Login & out Unit Test – 로그인 실패2

Test Case Name	**Login & out – 잘못된 비밀번호로 인한 로그인 실패**
Test case object	잘못된 비밀번호 정보가 들어왔을 때 작동을 확인
Test Inputs	SKKU DB에 등록된 아이디지만 비밀번호가 다를 때 (ex. 143jk!)
Action	1. SKKU DB에 소속된 유저가 아이디를 기입한다. 하지만 비밀번호를 회원정보와 다르게 입력한다. 2. 확인 버튼을 터치한다. 3. SKKU DB에 등록된 학생정보를 authorization 한다. 4. query가 실패해서 로그인이 되지 않는다.
Expected Results	'잘못된 비밀번호입니다' 메시지를 띄운다. 로그인이 되지 않는다.

[Table 7] Login & out Unit Test – 아이디 자료형 테스트

Test Case Name	**Login & out – 지나치게 긴 아이디를 다룰 수 있는지 확인**
Test case object	자료형(VARCHAR30)을 넘어선 긴 문자를 집어넣을 때 handle할 수 있는지 여부 확인
Test Inputs	지정한 자료형 크기를 넘어서는 긴 아이디 (ex. asdfasdfasdfasdfasdfasdfasdfasdfasdfasdfasdf)
Action	1. 유저가 긴 아이디를 입력한다. 2. 확인 버튼을 터치한다. 3. 로그인이 실패한다.
Expected Results	'등록되지 않은 아이디입니다.' 메시지를 띄운다. 오버플로로 시스템이 다운되지 않아야 한다. 로그인이 되지 않는다.

[Table 8] Login & out Unit Test – 비밀번호 자료형 테스트

Test Case Name	**Login & out – 지나치게 긴 비밀번호를 다룰 수 있는지 확인**
Test case object	자료형(VARCHAR30)을 넘어선 긴 문자를 집어넣을 때 handle할 수 있는지 여부 확인
Test Inputs	지정한 자료형 크기를 넘어서는 긴 비밀번호

	(ex. asdfasdfasdfasdfasdfasdfasdfasdfasdfasdfasdfasdf!!)
Action	1. 유저가 아이디와 긴 비밀번호를 입력한다. 2. 확인 버튼을 터치한다. 3. 로그인이 실패한다.
Expected Results	잘못된 아이디라면 ‘등록되지 않은 아이디입니다’ 메시지를 띄운다. 올바른 아이디라면 ‘잘못된 비밀번호입니다’ 메시지를 띄운다. 오버플로로 시스템이 다운되지 않아야 한다. 로그인이 되지 않는다.

[Table 9] Login & out Unit Test – 해킹 시도 방지 테스트

Test Case Name	**Login & out – 해킹 시도를 적절하게 대응하는지 확인**
Test case object	SQL injection을 통해 로그인을 가능하게 하는 방법에 적절하게 대응하는지 확인
Test Inputs	SQL injection등으로 비밀번호 입력에 대한 결과를 True로 만들 수 있는 값 Ex) <Or ‘1’=’1’>
Action	1. 유저가 비밀번호 입력란에 <Or ‘1’=’1’>을 입력한다. 2. 확인 버튼을 터치한다. 3. 로그인이 실패한다.
Expected Results	SQL injection으로 인한 로그인 거부의 경우 ‘정상적이지 않은 접근입니다.’ 메시지를 띄운다. 로그인이 되지 않는다.

3.1.3. User Account Login Test Case – Session Validation

[Table 10] Login & out Unit Test – 세션 만료 테스트

Test Case Name	**Login & out – 세션 만료되었을 때 로그아웃 되는지 확인**
Test case object	세션이 만료되었을 때 로그아웃 되는지 확인
Test Inputs	세션이 만료되고 예약 기능을 사용하려고 한다.
Action	1. 세션이 만료된 유저가 강의실 좌석 예약 기능을 터치한다.
Expected Results	‘세션이 만료되어 로그아웃 합니다’ 메시지를 띄운다.

	로그아웃이 성공한다. 로그인 화면을 띄운다.

3.1.4. Constraints

Login & Out 기능은 아이디와 패스워드를 입력하고 확인 버튼을 터치한 뒤 5초 내로 성공 혹은 실패에 대한 결과가 나와야 한다.

보안을 위해 패스워드를 입력할 경우 별표(*, asterisk) 로 대신해서 표현된다.

3.2. Booking

Booking 기능을 사용하는 사용자는 학생과 교직원이 있다. 사용자는 로그인을 한 후 메인 화면에서 강의실과 버스 좌석 중 어떤 예약 서비스를 이용할 지 결정한다. 그 후 해당 예약 서비스의 예약 세부 정보를 선택한 후 좌석을 선택하여 예약을 진행한다.

Booking의 Precondition으로 예약 세부 정보는 SKKU API를 통해 받아온 정보이다. 예약할 좌석은 자유석이나 이미 예약이 되어있는 좌석이 아닌, 예약 가능한 좌석이어야 한다.

3.1.1. Booking Test Case

[Table 11] Booking Unit Test – 예약 성공

Test Case Name	**Booking – 예약 성공**
Test case object	올바른 입력 값을 받았을 때 예약 기능의 정상적인 작동 확인
Test Inputs	이용할 예약 서비스 예약 세부 정보 예약 좌석
Action	1. 로그인 한 유저가 메인 화면에서 이용할 예약 서비스 버튼을 누른다. 2. 예약 세부 정보를 선택한 후 좌석 조회 버튼을 누른다. 3. 예약 좌석을 선택한 후 예약 버튼을 누른다. 4. 예약이 성공한다.
Expected Results	예약 성공 메시지가 띄워진다. 예약한 좌석의 정보가 DB에 저장된다.

[Table 12] Booking Unit Test – 자유석 예약

Test Case Name	**Booking – 자유석 예약**
Test case object	잘못된 입력 값을 받았을 때 예약 기능의 정상적인 작동 확인
Test Inputs	이용할 예약 서비스 예약 세부 정보 자유석
Action	1. 로그인 한 유저가 메인 화면에서 이용할 예약 서비스 버튼을 누른다. 2. 예약 세부 정보를 선택한 후 좌석 조회 버튼을 누른다. 3. 자유석을 선택한다
Expected Results	“자유석은 선택할 수 없습니다.” 메시지가 띄워진다.

[Table 13] Booking Unit Test – 예약 완료 좌석 예약

Test Case Name	**Booking – 예약 완료 좌석 예약**
Test case object	잘못된 입력 값을 받았을 때 예약 기능의 정상적인 작동 확인
Test Inputs	이용할 예약 서비스 예약 세부 정보 예약 좌석
Action	1. 로그인 한 유저가 메인 화면에서 이용할 예약 서비스 버튼을 누른다. 2. 예약 세부 정보를 선택한 후 좌석 조회 버튼을 누른다. 3. 이미 예약된 좌석을 선택한다.
Expected Results	“예약된 좌석은 선택할 수 없습니다.” 메시지가 띄워진다.

[Table 14] Booking Unit Test – 동시 예약

Test Case Name	**Booking – 동시 예약**
Test case object	한 좌석을 동시에 누른 상황에서 예약 기능의 정상 작동 확인 동시에 좌석을 누르게 되면 먼저 예약을 마친 사람에게 해당 좌석

	이 예약되며, 뒤늦게 누른 사람에게는 예약 실패 메시지가 뜬다.
Test Inputs	이용할 예약 서비스 예약 세부 정보 예약 좌석
Action	1. 로그인 한 유저가 메인 화면에서 이용할 예약 서비스 버튼을 누른다. 2. 예약 세부 정보를 선택한 후 좌석 조회 버튼을 누른다. 3. 좌석을 선택한 후 예약 버튼을 누른다. 4. 예약이 실패한다.
Expected Results	이미 예약된 좌석이라는 메시지가 띄워진다.

3.3. Arranging Lecture Room Seating

Arranging Lecture Room Seating 기능을 사용하는 사용자는 교직원이다. 사용자는 수업 방식에 적절한 형태로 강의실의 좌석 배치를 설정할 수 있다. 일반적인 이론 수업의 경우 사각형 배열의 좌석 배치도를 설정할 수 있고 수강인원이 적은 경우 강의실 뒷좌석들은 예약 불가로 변경하는 등 적절한 좌석 배치를 설정할 수 있다. 실습, 팀, 실험 수업 등에서 조별로 좌석을 배치할 수도 있다.

Arranging Lecture Room Seating의 Precondition으로 사용자가 강의실 Tab을 터치하기 전까지 변경된 정보를 불러온다. 그리고 사용자가 강의실 배치/수동 배정/랜덤 배정 기능을 이용하기 위해서는 사용자가 현재 Login된 상태임을 가정하고 있다.

3.3.1. Arranging Lecture Room Seating Unit Test

[Table 15] Arranging Lecture Room Seating Unit Test – 좌석 배열 변경 성공

Test Case Name	**Arranging Lecture Room Seating – 좌석 배열 변경 성공**
Test case object	좌석 배열 변경에 대한 작동 확인
Test Inputs	기존 좌석박스를 정상적인 범위 내로 드래그
Action	1. 원하는 강의와 일정을 선택한다. 2. 좌석 배치 버튼을 터치한다.

	3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 위치를 변경하고 싶은 좌석을 정상적인 범위 내에서 원하는 위치로 드래그 한다. 5. 확인 버튼을 터치한다. 6. 좌석 배열에 성공하고 변경된 정보를 저장한다.
Expected Results	좌석 배열 성공 메시지를 띄운다. 변경된 내용이 반영된 강의실 배치 화면을 띄운다.

[Table 16] Arranging Lecture Room Seating Unit Test – 좌석 형태 변경 성공

Test Case Name	**Arranging Lecture Room Seating – 좌석 형태 변경 성공**
Test case object	좌석 형태 변경에 대한 작동 확인
Test Inputs	좌석 선택 후 팝업에서 원하는 좌석 형태 선택
Action	1. 원하는 강의와 일정을 선택한다. 2. 좌석 배치 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 좌석 형태를 변경하고자 하는 좌석을 터치한다. 5. 팝업에서 원하는 좌석 형태를 선택한다. 6. OK버튼을 터치한다. 7. 좌석 형태 변경에 성공하고 변경된 정보를 저장한다.
Expected Results	좌석 형태 변경 성공 메시지를 띄운다. 변경된 내용이 반영된 강의실 배치 화면을 띄운다.

[Table 17] Arranging Lecture Room Seating Unit Test – 좌석 수동 배정 성공

Test Case Name	**Arranging Lecture Room Seating – 좌석 수동 배정 성공**
Test case object	좌석 수동 배정 기능 작동 확인
Test Inputs	강의와 일정 선택 후 수동 배정 버튼 터치
Action	1. 원하는 강의와 일정을 선택한다. 2. 수동 배정 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 수동 배정하고자 하는 좌석을 선택한다. 5. 해당 좌석에 배정하고자 하는 학생을 선택한다. 6. OK버튼을 터치한다.

	7. 수동 배정에 성공하고 변경된 정보를 저장한다.
Expected Results	수동 배정 성공 메시지를 띄운다. 변경된 내용이 반영된 강의실 배치 화면을 띄운다.

[Table 18] Arranging Lecture Room Seating Unit Test – 좌석 랜덤 배정 성공

Test Case Name	**Arranging Lecture Room Seating – 좌석 랜덤 배정 성공**
Test case object	좌석 랜덤 배정 기능 작동 확인
Test Inputs	강의와 일정 선택 후 랜덤 배정 버튼 터치
Action	1. 원하는 강의와 일정을 선택한다. 2. 랜덤 배정 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 해당 강의실의 좌석에 학생들을 임의로 배정한다. 5. 좌석 랜덤 배정에 성공하고 변경된 정보를 저장한다.
Expected Results	좌석 랜덤 배정 성공 메시지를 띄운다. 변경된 내용이 반영된 강의실 배치 화면을 띄운다.

3.3.2. Arranging Lecture Room Seating Unit Test - Negative

[Table 19] Arranging Lecture Room Seating Unit Test – 좌석 배열 변경 실패 1

Test Case Name	**Arranging Lecture Room Seating – 좌석 배열 변경 실패**
Test case object	지정된 강의실 범위를 벗어난 경우에 대한 작동 확인
Test Inputs	기존 좌석박스를 지정된 범위 밖으로 드래그
Action	1. 원하는 강의와 일정을 선택한다. 2. 좌석 배치 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 위치를 변경하고 싶은 좌석을 지정된 범위 밖으로 드래그 한다. 5. 좌석 배열에 실패한다.
Expected Results	좌석 배열 실패 및 재시도 요청 메시지를 띄운다. 드래그 이전 강의실 배치 화면을 띄운다.

[Table 20] Arranging Lecture Room Seating Unit Test – 좌석 배열 변경 실패 2

Test Case Name	Arranging Lecture Room Seating – 좌석 배열 변경 실패
Test case object	다른 좌석과 겹치게 배열한 경우에 대한 작동 확인
Test Inputs	기존 좌석박스를 다른 좌석의 위치와 겹쳐지도록 드래그
Action	1. 원하는 강의와 일정을 선택한다. 2. 좌석 배치 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 위치를 변경하고 싶은 좌석을 다른 좌석과 겹치도록 드래그 한다. 5. 좌석 배열에 실패한다.
Expected Results	좌석 배열 실패 및 재시도 요청 메시지를 띄운다. 드래그 이전 강의실 배치 화면을 띄운다.

[Table 21] Arranging Lecture Room Seating Unit Test – 좌석 수동 배정 실패

Test Case Name	Arranging Lecture Room Seating – 좌석 수동 배정 실패
Test case object	선택한 학생이 다른 좌석에 배정되어 있는 경우에 대한 작동 확인
Test Inputs	강의와 일정 및 배정하고자 하는 좌석 선택 후 다른 좌석에서 배정되어 있는 학생을 선택
Action	1. 원하는 강의와 일정을 선택한다. 2. 수동 배정 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 수동 배정하고자 하는 좌석을 선택한다. 5. 이미 다른 좌석에 배정되어 있는 학생을 선택한다. 6. OK버튼을 터치한다. 7. 수동 배정에 실패한다.
Expected Results	‘중복되는 학생입니다’ 메시지를 띄운다. 수동 배정 버튼을 터치하기 전 화면을 띄운다.

[Table 22] Arranging Lecture Room Seating Unit Test – 좌석 랜덤 배정 실패

Test Case Name	Arranging Lecture Room Seating – 좌석 랜덤 배정 실패
Test case object	강의실에 부여된 좌석 수 보다 해당 강의를 수강하는 학생 수가 많을 경우
Test Inputs	해당 강의실의 좌석 수 보다 수강 학생 수가 많은 강의와 일정 선택

	후 랜덤 배정 버튼 터치
Action	1. 해당 강의실의 좌석 수 보다 수강 학생 수가 많은 강의와 일정을 선택한다. 2. 랜덤 배정 버튼을 터치한다. 3. 이전까지 저장된 해당 강의실에 대한 정보를 불러온다. 4. 좌석 수와 학생 수를 비교한다. 5. 랜덤 배정에 실패한다.
Expected Results	'해당 강의실의 좌석 수가 부족합니다. 좌석 배치를 수정하세요.' 메시지를 띄운다. 좌석 배치 기능을 지원하는 페이지를 띄운다.

3.3.3. Constraints

좌석 배치/수동 배정/랜덤 배정은 모두 확인 버튼 혹은 OK버튼을 터치한 후 5초 이내에 결과가 표현되어야 한다.

3.4. Checking Reservation

Reservation check 기능을 활용하는 사용자에는 학생과 교직원이 있다. 사용자는 메인 화면의 "Today"란, 또는 예약 확인 탭을 통해 기존 예약 내역을 확인할 수 있다.

Reservation check 기능의 precondition으로는 사용자 정보가 SKKU API에 등록되어 있어야 한다는 것, 그리고 확인하고자 하는 사전 예약 내역이 존재해야 한다는 것이 있다.

3.4.1. Reservation Check Test Case

[Table 23] Check – 전체 예약 내역 확인

Test Case Name	**Check – 전체 예약 내역 확인**
Test case object	올바른 입력을 받았을 때, 확인 기능의 정상적인 활동 확인
Test Inputs	유효하거나, 종료, 만료된 예약 내역의 존재
Action	사용자가 "예약 확인" 탭을 터치한다.
Expected Results	사용자의 과거 및 현재 예약 내역을 최근 순서로 표시한다.

[Table 24] Check – 개별 예약 세부사항 확인

Test Case Name	Check – 개별 예약 세부사항 확인
Test case object	올바른 입력을 받았을 때, 확인 기능의 정상적인 활동 확인
Test Inputs	유효한 개별 예약 내역
Action	사용자가 전체 예약 내역에서 확인하고자 하는 개별 예약을 터치한다. 메인 화면의 “Today” 란에서 확인하고자 하는 개별 예약을 터치한다.
Expected Results	해당 예약의 세부사항(출발위치, 시간, 장소, 좌석위치 등)이 화면에 표시된다.

[Table 25] Check – 이미 취소된 예약의 세부사항 확인

Test Case Name	Check – 이미 취소된 예약의 세부사항 확인
Test case object	올바른 입력을 받았을 때, 확인 기능의 정상적인 활동 확인
Test Inputs	기존에 취소된 예약 및 과거에 완료 및 만료된 예약내역
Action	이미 종료된 수업 또는 셔틀 예약 내역, 또는 취소한 예약 내역을 터치한다.
Expected Results	해당 예약의 세부사항(출발위치, 시간, 장소, 좌석위치 등)이 표시된다. 이와 함께, 종료된 예약의 경우 “만료된 예약입니다” 메시지가, 취소된 예약의 경우, “취소된 예약입니다” 메시지가 하단에 함께 표시된다.

[Table 26] Check – 예약내역이 존재하지 않는 경우의 예약 확인

Test Case Name	Check – 예약내역이 존재하지 않는 경우의 예약 확인
Test case object	잘못된 입력을 받았을 때, 확인 기능의 정상적인 활동 확인
Test Inputs	예약내역의 부재
Action	사용자는 “예약 확인” 탭을 터치한다.
Expected Results	화면에 “예약 내역이 존재하지 않습니다”라는 메시지를 표시한다.

3.5. Changing / Cancelling Reservation

Reservation change/cancellation 기능의 사용자는 학생 및 교직원이다. 사용자는 개별 예약 내역에서 변경/취소 기능을 사용할 수 있으며, 이를 통해 예약 시간, 구간, 좌석 위치 등의 정보를 수정 또는 예약 자체를 취소하는 것이 가능하다.

해당 기능의 precondition은 사용자 정보가 SKKU API에 등록되어 있어야 한다는 것, 그리고 수정 및 취소의 대상이 되는 예약 내역이 사전에 존재해야 한다는 것이 있다.

3.5.1. Reservation Change Test Case

[Table 27] Change/Cancellation - 예약 변경

Test Case Name	Change/Cancellation - 예약 변경
Test case object	올바른 입력을 받았을 때, 변경/취소 기능의 정상적인 활동 확인
Test Inputs	유효한 예약 건 예약 세부사항 (좌석 위치, 출발지점, 시간, 장소)
Action	사용자가 예약 확인 탭, 당일 예약의 경우, “Today”란에서 개별 예약을 터치한다. 표시되는 개별 예약 페이지 하단의 변경/취소 버튼을 터치한다. 변경할 수 있는 정보가 표시되며, 사용자는 새로운 예약 정보를 입력한다. “확인” 버튼을 터치한다.
Expected Results	“변경이 완료되었습니다” 메시지를 표시한다.

[Table 28] Change/Cancellation - 예약 취소

Test Case Name	Change/Cancellation - 예약 취소
Test case object	올바른 입력을 받았을 때, 변경/취소 기능의 정상적인 활동 확인
Test Inputs	유효한 예약 건
Action	사용자가 예약 확인 탭, 당일 예약의 경우, “Today”란에서 개별 예약을 터치한다. 표시되는 개별 예약 페이지 하단의 변경/취소 버튼을 터치한다. 팝업창의 “취소” 버튼을 터치한다. 취소 확인을 묻는 확인창에서 “확인” 버튼을 터치한다.
Expected Results	“취소가 완료되었습니다” 메세시를 표시한다. 당일 예약이 취소된 경우, “Today”란에 해당 예약이 표시되지 않는다.

[Table 29] Change/Cancellation - 이미 만료된(취소, 과거) 예약의 변경 및 취소

Test Case Name	Change/Cancellation -

	이미 만료된(취소, 과거) 예약의 변경 및 취소
Test case object	잘못된 입력을 받았을 때, 변경/취소 기능의 정상적인 활동 확인
Test Inputs	유효하지 않은 개별 예약 건 취소가 완료된 예약 건
Action	사용자가 유효하지 않은 예약 또는 취소된 개별 예약을 터치한다. 개별 예약 세부사항에서 변경 및 취소 버튼을 터치한다.
Expected Results	“해당 예약은 만료(취소)되어 변경 및 취소가 불가합니다”라는 메시지가 표시된다.

4. Software Interface Test

이 장에서는 BOOKINGO의 external interface test cases에 대해 기술한다. BOOKINGO의 외부 API는 Firebase, SKKU API가 있다. API의 역할을 알아보고 그리고 API의 기능들을 테스트할 수 있는 테스트 케이스들을 알아본다.

4.1. Firebase

BOOKINGO 시스템은 Firebase를 통해 데이터베이스 관리를 한다. Firebase의 Firestore를 이용하여 저장된다. Firebase를 통해 FCM 및 Real-Time database 서비스 등을 사용할 수 있다. 데이터는 JSON 객체의 형태로 저장된다.

4.1.1. Connection Test

[Table 30] Firebase Connection Test

Test Case Name	**Firebase Connection Test**
Test case object	Firebase와의 통신이 정상적으로 작동하는지 확인
Test Inputs	Ping command
Processing	Ping command code를 통해 네트워크가 잘 연결되는지 확인한다.
Expected Results	Ping status code 0: Success

4.1.2. Simultaneous Connections

동시 연결은 데이터베이스에 연결된 휴대기기, 브라우저 탭 또는 서버 앱 하나를 의미한다. 최

대 동시 연결 수는 전체 사용자 수 및 사용자가 앱에서 소비하는 평균 시간에 따라 다르다. Firebase 데이터 스토리지의 최대 한도는 200,000개 미만이다. 시스템의 병목 지점을 확인하기 위해 단계적으로 동시 연결을 테스트한다.

[Table 31] Firebase Simultaneous Connections – 100000 connections

Test Case Name	**Firebase Simultaneous Connections – 100000 connections**
Test case object	Firebase 에서 동시 연결 한도를 확인
Test Inputs	Ping Command
Processing	100,000 번의 Ping 요청을 동시에 서버에 보낸다.
Expected Results	Ping status code 0: Success

[Table 32] Firebase Simultaneous Connections – 200000 connections

Test Case Name	**Firebase Simultaneous Connections – 100000 connections**
Test case object	Firebase 에서 동시 연결 한도를 확인
Test Inputs	Ping Command
Processing	200,000 번의 Ping 요청을 동시에 서버에 보낸다.
Expected Results	Ping status code 1: Failure

4.1.3. Simultaneous Responses

단일 데이터베이스는 브로드캐스트 작업 및 개별적인 읽기와 관련된 데이터 패킷을 보내게 된다.

[Table 33] Firebase Simultaneous Responses – 50000 connections

Test Case Name	**Firebase Simultaneous Responses – 50000 connections**
Test case object	Firebase 단일 데이터베이스에서 전송한 동시 요청 처리 동작 및 한도를 확인
Test Inputs	GET Command
Processing	50,000 번의 GET요청을 동시에 서버에 보낸다.
Expected Results	HTTPS Status code

	200: OK

[Table 34] Firebase Simultaneous Responses – 100000 connections

Test Case Name	**Firebase Simultaneous Responses – 100000 connections**
Test case object	Firebase 단일 데이터베이스에서 전송한 동시 요청 처리 동작 및 한도를 확인
Test Inputs	GET Command
Processing	100,000 번의 GET요청을 동시에 서버에 보낸다.
Expected Results	HTTPS Status code 500: Internal Server Error

4.1.4. Size of a Single Write Event

Firebase에서 쓰기를 실행하기 위한 이벤트는 쓰는 위치의 기존 데이터 값과 쓰기 위치에 새 데이터를 업데이트하는데 필요한 데이터 값으로 구성된다. 이 쓰는 이벤트는 1MB 크기를 초과하면 정상적으로 작동하지 않는다.

[Table 35] Size of a single write event

Test Case Name	**Single write event size**
Test case object	단일 데이터베이스에서 전송한 데이터 이벤트 크기 확인
Test Inputs	좌석 내역 수정하고 쓰기 요청 시도
Processing	데이터 업데이트 요청
Expected Results	이벤트 크기 1MB를 넘지 않아 정상적으로 요청이 이루어진다.

4.1.5. Size of a Single Response Served by the Database

단일 데이터베이스에서 제공하는 단일 응답의 크기는 256MB 미만이어야 한다. 예약 내역 및 좌석 배치를 불러올 때 데이터를 읽어야 하는데 이때 데이터 크기를 확인해야 한다.

[Table 36] Size of a single read response

Test Case Name	**Single read response size**

Test case object	단일 데이터베이스에서 제공하는 단일 응답의 크기를 확인
Test Inputs	좌석 배치 UI를 위해 데이터 요청.
Processing	데이터 읽기 요청
Expected Results	256MB를 넘지 않아 정상적으로 요청이 이루어진다.

4.1.6. Length of Time a Single Query

Firebase에서 단일 쿼리가 실패하기 전에 최대 15분 동안 실행될 수 있다. 만약 timeouts을 specify하면 설정한 시간에 따라 timeout이 발동한다. 최대 실행시간이지만 유저가 그렇게 오래 기다리지 않도록 해야 한다.

[Table 37] Length of time a single query can run

Test Case Name	**Single query time**
Test case object	단일 쿼리가 최대로 실행될 수 있는 시간을 확인
Test Inputs	학생 query를 실행해본다.
Processing	쿼리 실행
Expected Results	15분을 넘지 않고 실패했다는 메시지가 뜬다.

[Table 38] Length of time a single query can run - Negative

Test Case Name	**Single query time**
Test case object	단일 쿼리가 최대로 실행될 수 있는 시간을 확인 만약 timeouts 값이 음수거나 15분을 초과하면 HTTP 400 error가 난다.
Test Inputs	학생 query를 실행해본다.
Processing	1. timeouts을 음수 또는 15분을 초과한 숫자로 설정한다. 2. 쿼리 실행
Expected Results	HTTP 400 error status를 확인한다.

4.1.7. Size of a Single Write Request to the Database

Firebase에서 각 쓰기 작업의 총 데이터는 256MB 미만이어야 한다.

[Table 39] Size of single write request

Test Case Name	**Single write request size**
Test case object	데이터베이스에 대한 단일 쓰기 요청의 크기 확인. 256MB 미만이어야 한다.
Test Inputs	쓰기 작업을 요청한다.
Processing	좌석 배치 내역을 변경하고 쓰기 요청을 한다.
Expected Results	쓰기 작업의 총 데이터가 256MB이고 업데이트가 이상 없이 이루어진다.

4.1.8. REST API Responses

Firebase Database REST API의 반응들을 확인한다. HTTPS 클라이언트에서 요청을 보내고 반응을 확인한다.

[Table 40] REST API GET – Reading Data

Test Case Name	**GET – Reading Data**
Test case object	REST API GET response를 확인
Test Inputs	실시간 데이터베이스에 저장된 예약 내역을 read한다.
Processing	curl 명령어로 특정 예약 내역을 요청한다.
Expected Results	200 OK status code를 확인한다.

[Table 41] REST API PUT – Writing Data

Test Case Name	**PUT – Writing Data**
Test case object	REST API PUT response를 확인
Test Inputs	실시간 데이터베이스에 예약 내역을 생성한다.
Processing	curl 명령어와 PUT 명령어로 특정 예약 내역을 write한다.
Expected Results	200 OK status code를 확인한다.

[Table 42] REST API POST – Pushing Data

Test Case Name	**POST - Pushing Data**
Test case object	REST API POST response를 확인
Test Inputs	POST를 통해 리소스를 생성한다.
Processing	curl 명령어와 POST 명령어로 특정 예약 내역을 push한다.
Expected Results	200 OK status code를 확인한다.

[Table 43] REST API PATCH – Updating Data

Test Case Name	**PATCH – Updating Data**
Test case object	REST API PATCH response를 확인
Test Inputs	PATCH 명령어로 예약 내역을 업데이트해본다.
Processing	curl 명령어와 PATCH 명령어로 특정 예약 내역을 update한다.
Expected Results	200 OK status code를 확인한다.

[Table 44] REST API DELETE – Removing Data

Test Case Name	**DELETE – Removing Data**
Test case object	REST API Delete response를 확인
Test Inputs	예약 내역을 삭제해본다.
Processing	curl 명령어와 DELETE 명령어로 특정 예약 내역을 delete한다.
Expected Results	200 OK status code를 확인한다.

4.2. SKKU API

SKKU API는 Bookingo에 로그인 할 때 필요한 성균인 계정 정보, 예약 시스템에 필요한 버스 시간표, 강의 시간표, 수강인원, 강의실, 강사 등에 대한 정보를 제공해주는 서비스이다.

4.2.1. Connection Test

[Table 45] SKKU API Interface Test – Connection test

Test Case Name	SKKU API Connection Test
Test case object	SKKU API와의 통신이 정상적으로 작동하는지 확인
Test Inputs	Ping Command
Processing	Ping command를 통해 네트워크가 잘 연결되어 있는지 확인
Expected Results	Ping status code 0: Success

4.2.2. Functional Test

[Table 46] SKKU API Interface Test – Functional test 1

Test Case Name	SKKU API Functional Test
Test case object	SKKU API에서 Login 정보가 정상적으로 불러와지는지 확인
Test Inputs	GET Command 로 id와 pw가 올바른 정보인지 확인
Processing	GET 요청으로 임의의 ID와 PW의 로그인 가능 여부를 받는다.
Expected Results	HTTP Status code 200: OK 401: Unauthorized

[Table 47] SKKU API Interface Test – Functional test 2

Test Case Name	SKKU API Functional Test
Test case object	SKKU API에서 수강중인 강의가 정상적으로 불러와지는지 확인
Test Inputs	GET Command 로 id에 해당하는 수강 강좌를 불러온다
Processing	GET 요청으로 로그인 한 정보의 수강 강의 정보를 받는다.
Expected Results	HTTP Status code 200: OK 404: Not Found

[Table 48] SKKU API Interface Test – Functional test 3

Test Case Name	SKKU API Functional Test
Test case object	SKKU API에서 예약 관련 정보가 정상적으로 불러와지는지 확인
Test Inputs	GET Command 로 lecture/bus id에 해당하는 정보를 불러온다.

Processing	GET 요청으로 버스 / 강의 관련 정보를 받는다.
Expected Results	HTTP Status code 200: OK 404: Not Found

[Table 49] SKKU API Interface Test – Functional test 4

Test Case Name	SKKU API Functional Test
Test case object	SKKU API에서 좌석 배치가 정상적으로 불러와지는지 확인
Test Inputs	GET Command로 lecture/bus id에 해당하는 좌석 배치를 불러온다.
Processing	GET 요청으로 버스 / 강의 id를 받는다.
Expected Results	HTTP Status code 200: OK 404: Not Found

4.2.3. Performance Test

[Table 50] SKKU API Interface Test – Performance test 1

Test Case Name	SKKU API Performance Test
Test case object	SKKU API에서 동시 연결 한도를 확인
Test Inputs	Ping Command
Processing	100,000 번의 Ping 요청을 동시에 API 서버에 보낸다.
Expected Results	Ping status code 0: Success

[Table 51] SKKU API Interface Test – Performance test 2

Test Case Name	SKKU API Performance Test
Test case object	SKKU API에서 동시 연결 한도를 확인
Test Inputs	Ping Command
Processing	200,000 번의 Ping 요청을 동시에 API 서버에 보낸다.
Expected Results	Ping status code 1: Failure

[Table 52] SKKU API Interface Test – Performance test 3

Test Case Name	SKKU API Performance Test
Test case object	SKKU API에서 단일 데이터 베이스에 동시 전송 한도를 확인
Test Inputs	GET Command
Processing	50,000 번의 GET요청을 동시에 API 서버에 보낸다.
Expected Results	HTTP Status code 200: OK

[Table 53] SKKU API Interface Test – Performance test 4

Test Case Name	SKKU API Performance Test
Test case object	SKKU API에서 단일 데이터 베이스에 동시 전송 한도를 확인
Test Inputs	GET Command
Processing	100,000 번의 GET요청을 동시에 API 서버에 보낸다.
Expected Results	HTTP Status code 500: Internal Server Error

5. Supporting Information

5.1. Document History

[Table 54] Document History

#	Date	Description	Writer
0.	2021/5/20	문서 개요 작성 및 회의	강종현, 이승우, 임재현, 임서현
1.1	2021/5/23	Introduction, Document History, Firebase	임서현
1.2	2021/5/23	Arranging	강종현
1.3	2021/5/23	Checking, Cancelling & Changing Reservation	임재현

1.4	2021/5/23	Booking, SkkU API	이승우
1.5	2021/5/24	회의 및 수정, TPD_1	강종현, 이승우, 임서현, 임재현
1.6	2021/5/25	회의 및 수정, TPD_2	강종현, 이승우, 임서현, 임재현
1.7	2021/5/27	Test Plan Document	강종현, 이승우, 임서현, 임재현