4. System Architecture -Frontend

4.1. Objectives

4.2. Subcomponents

4.2.3. Checking Reservation

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

4.2.3.1. Attributes

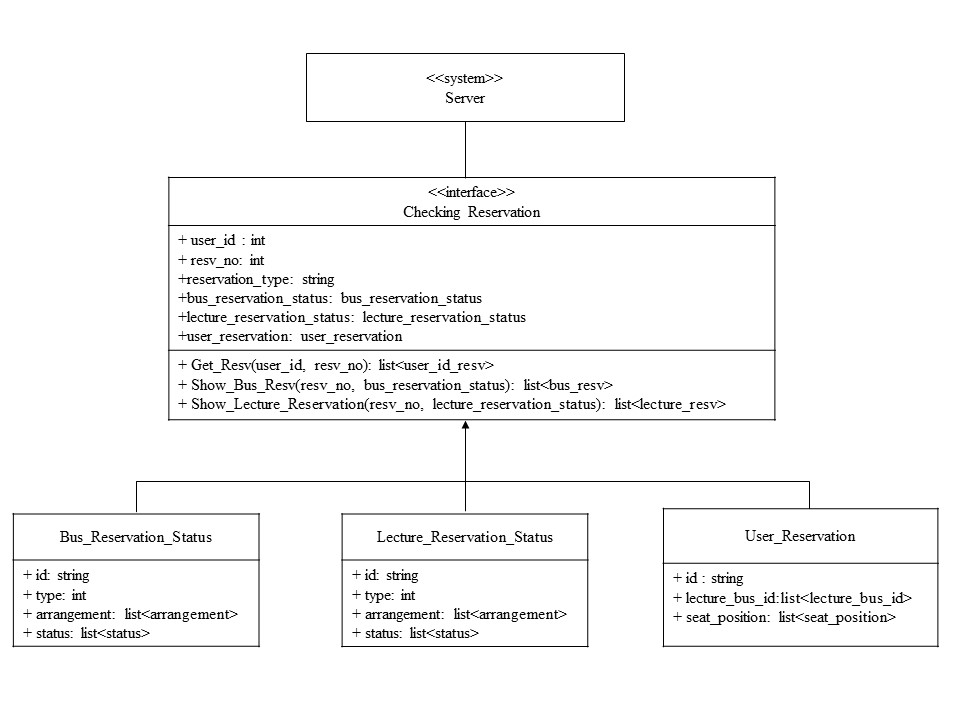
Checking Reservation 객체가 갖는 특성은 다음과 같다:

* 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.

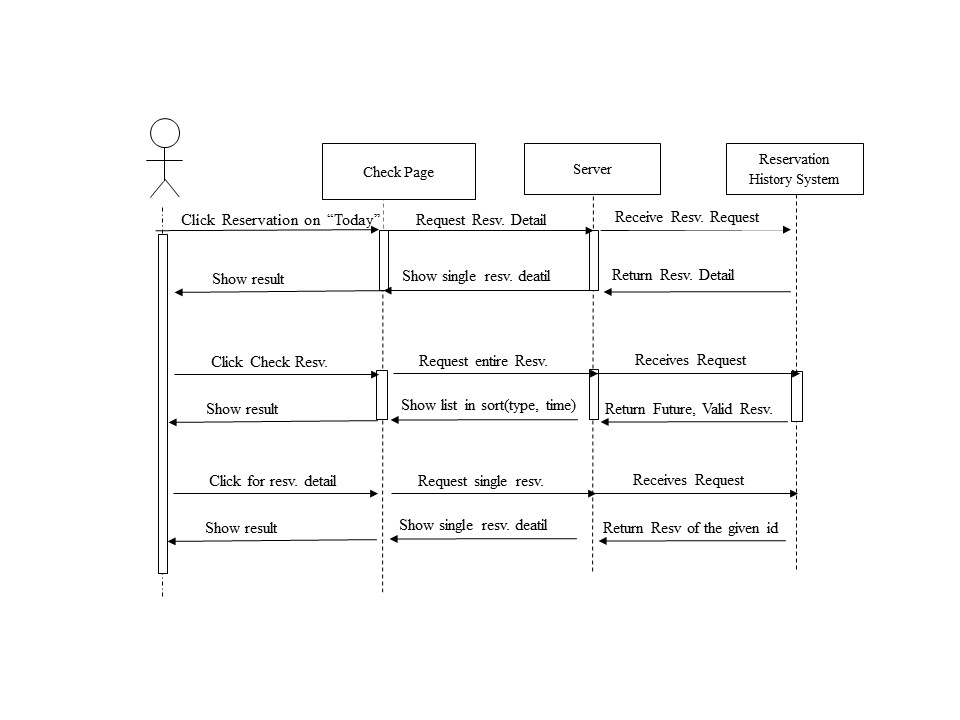
**4.2.3.2. Methods**

* Get\_Resv(user\_id, resv\_id): 전체 예약 리스트를 요청하여 종류별, 시간별로 나타낸다.
* Show\_Bus\_Resv(resv\_no, bus\_reservation\_status): 버스 개별예약의 세부사항을 표시한다.
* Show\_Lecture\_Resv(resv\_no, lecture\_reservation\_status):강의실 개별예약의 세부사항을 표시한다.

**4.2.3.3. Class Diagram**



**4.2.3.4. Sequence Diagram**

****

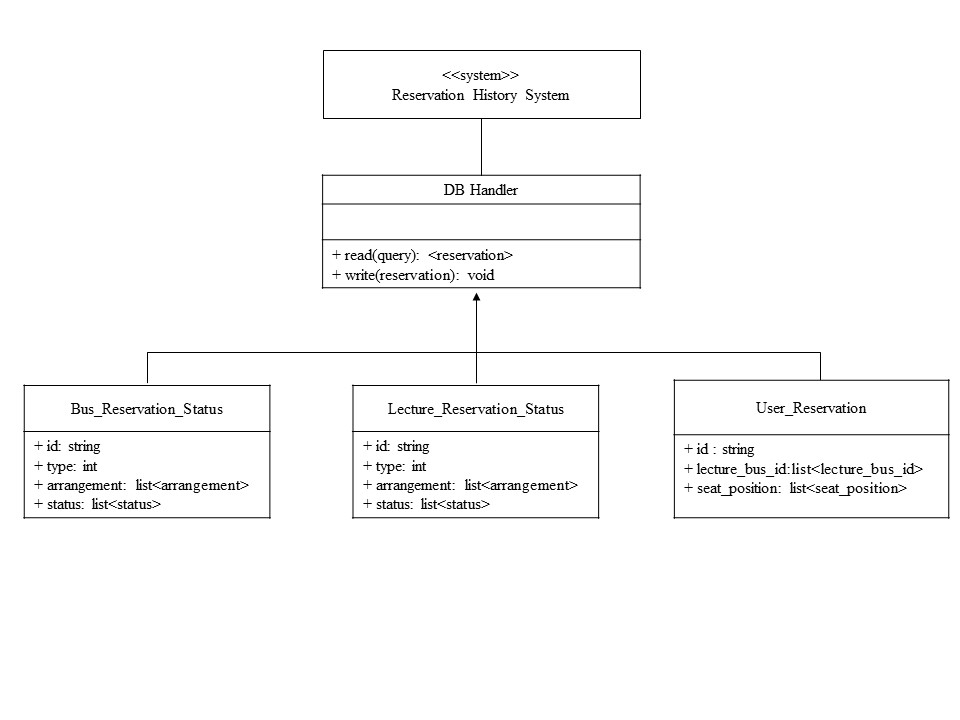
5. System Architecture -Backend

5.1. Objectives

5.2. Subcomponents

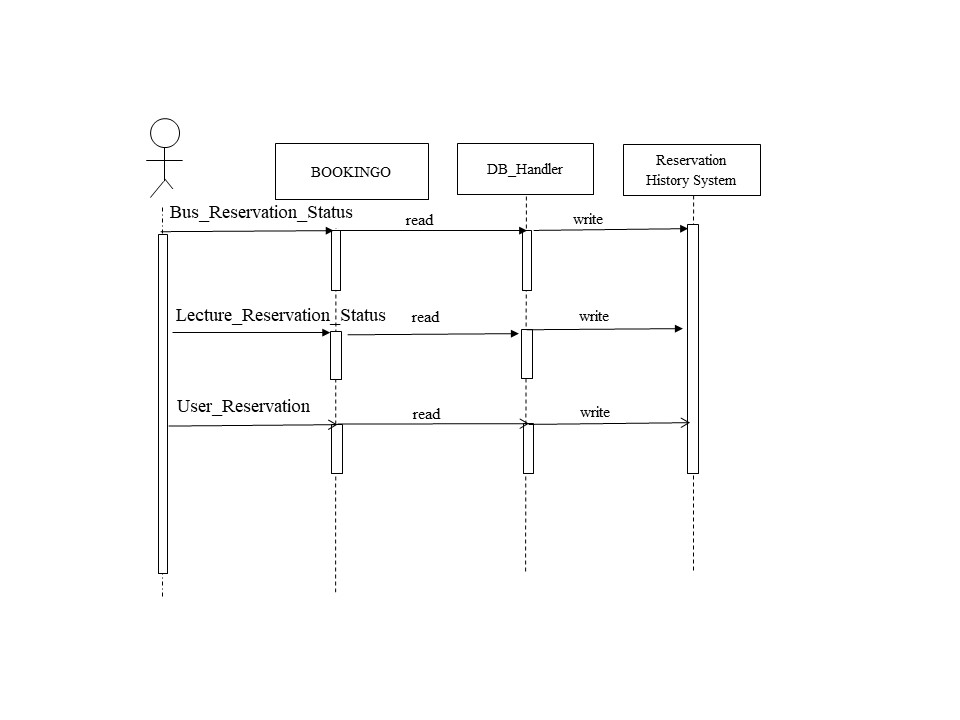
5.2.3. Reservation History System

5.2.3.1. Class Diagram



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.

5.2.3.2. Sequence Diagram

****

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 세가지 단계로 나누어 볼 수 있으며, 각각 중점적으로 평가하고자하는 바가 상이하다.

8.2. Testing Policy

8.2.1. Development Testing

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

8.2.1.1. Usability

BOOKINGO 어플리케이션 내 예약 과정에서 사용자의 불편함을 최소화해야하며, 예약 관련 세부사항이 화면에 명확하게 나타나야한다. 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 지체없이 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 처리하게되는 정보는 학내 구성원의 이동 및 수업 정보를 포함하게되며, 이러한 민감한 개인 정보를 철저히 보호할 수 있어야만 한다. 이를 위해 개발 중 development testing을 통해 login/out 부분의 security를 component testing및 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 릴리즈하기 전 release testing을 진행한다.

8.2.3. User Testing

최종 사용자 중 일부, 또는 전체를 통해 어플리케이션 사용 경험에 관한 조언, 평가 및 피드백을 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 설정하고자하며, 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] Adobe Illustrator 로고

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

9.2.2. Adobe Xd



[Figure] 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 프로토 타입 UI 구성에 있어 효율적이기도 하다.

9.2.3. Android Studio

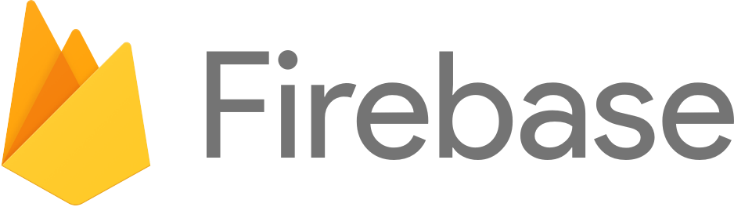


[Figure] Android Studio 로고

BOOKINGO 시스템의 개발은 안드로이드 어플리케이션 개발에 일반적으로 사용하는 Android Studio IDE를 활용하게 된다. 언어는 Kotlin을 사용하며, style guide 또한 Kotlin를 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] 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

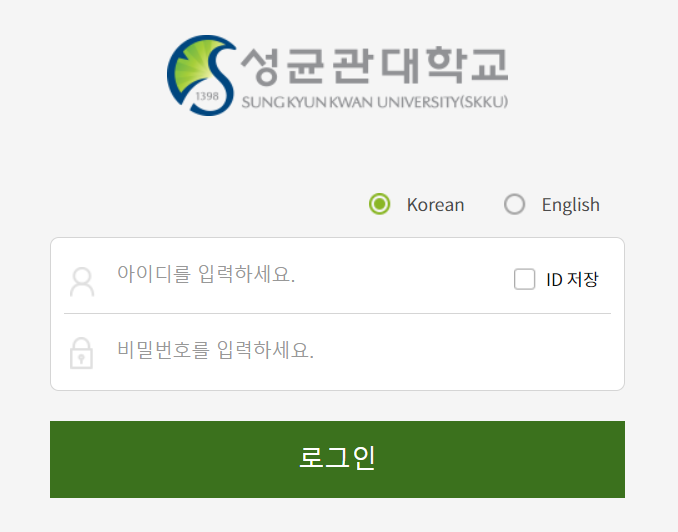


[Figure] Github 로고

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(가정)



[Figure] 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

|  |  |  |  |
| --- | --- | --- | --- |
| # | Date | Description | Writer |
| 0 | 2021/5/6 | 4.2.1, 4.2.5. | 강종현 |
| 1.1 | 2021/5/6 | 4.2.2, 5.2.2. | 강종현 |
| 1.2 | 2021/5/6 | 4.2.4., 5.2.1. | 임서현 |
| 1.3 | 2021/5/6 | 4.2.3., 5.2.3. | 임재현 |
| 1.4 | 2021/5/15 | 1. Preface, 2. Intro, 3. Architecture | 강종현 |
| 1.5 | 2021/5/15 | 6. Protocol Design | 이승우 |
| 1.6 | 2021/5/15 | 7. Database Design | 임서현 |
| 1.7 | 2021/5/15 | 8. Testing Plan, 9. Development Plan | 임재현 |