
# Design Specification

## Team 14

2017310022 나경호

2017314698 김동락

2017314648 박세환

2016315178 이기요

2017314519 이은지

2016312961 조민재

# Table of Contents

# 1. Preface

Preface 에서는 본 문서의 예상 독자를 정의하고 문서의 전체적인 구조에 대해 간략히 설명한다. 문서에 변경 사항이 생겼을 때 변경 내용과 이유를 포함한 version history를 각 version 마다 기록한다.

## 1.1 Readership

본 문서는 Team 14의 조원들과, 소프트웨어 개발을 포함한 Software engineering에 도움을 주실 교수님 및 조교님을 예상 독자로 둔다. 본 문서에서 기술하는 시스템이 차후 타 프로젝트와 호환되어 활용될 시, 타 프로젝트의 소프트웨어 개발자도 예상 독자가 된다. 소프트웨어 개발자는 Software architect, Software engineer, Map manager, 서버 운영자를 포함한다.

## 1.2 Scope

본 Design Specification의 범위는 교수님과의 office hour meeting room인 Neo Metaverse Office를 구현하면서 Software engineering에서 사용될 design이다.

## 1.3 Objective

본 문서의 목적은 Neo Metaverse Office의 design에 대한 기술적인 설명을 제공하는 데에 있다. Neo Metaverse Office의 구현에 있어서 소프트웨어 측면에서의 architecture와 design을 제시한다. 다양한 diagram을 통해 본 시스템의 use case와 structure, flow 등을 나타내어 예상 독자가 본 문서를 읽고 시스템의 design을 이해하는 데에 도움을 주고자 한다.

## 1.4 Document Structure

본 문서는 Preface, Introduction, System architecture, Testing plan, Development plan, Supporting Information의 순서로 구성되어 있다. 다음은 각 장에 대한 간략한 설명이다.

**A. Preface**

이 장에서는 먼저 본 문서의 예상 독자를 제시한다. 다음으로는 Design Specification의 범위를 기술하고, 이어서 본 문서의 목적을 제시한다. 마지막으로 본 문서의 구조에 대해 설명한다.

**B. Introduction**

이 장에서는 Design Specification에 사용된 diagram과 편집 tool, 본 프로젝트의 개발 범위, 본 문서의 reference를 제시한다.

**C. System Architecture**

이 장에서는 본 문서에서 다루는 시스템에 무슨 구성요소가 있는지를 제시하고, 각 요소의 구조, 기능, interface를 다양한 diagram을 사용하여 나타낸다.

**D. Testing Plan**

시스템이 요구사항을 잘 지키면서 제작되었는지, 구조적 · 기능적 오류가 없는지 검사하는 작업에 대해 계획을 세우고, 이를 제시한다.

**E. Development Plan**

이 장에서는 시스템 제작 시 사용되는 개발 환경 및 개발 도구를 제시한다. 또한, 시스템 구현에의 assumption과 constraint, dependency를 기술한다.

**F. Supporting Information**

이 장에서는 문서 작성의 기준과 본 문서의 최신화 정보를 제시한다.

# 2. Introduction

프로젝트에서는 메타버스 오피스를 만들어 교수자와 학생이 공간의 제약 없이 만날 수 있게 해 준다. 이 시스템은 메타버스 오피스 안에 존재하는 사람들끼리는 자유롭게 소통하게 하고, 인가된 자 이외의 접근 및 방해를 막아 원활한 만남 기회를 제공해야 한다. 본 문서는 프로젝트를 구현하면서 쓰이는 design에 대한 것이다. 이하 기술될 design은 이미 제출된 Team 14의 Software Requirements Specification의 내용을 바탕으로 한다.

## 2.1 Objectives

Design process에 사용된 diagram과 tool을 소개하고, 본 프로젝트의 scope와 본 문서를 작성하면서 참고한 reference를 제시한다.

## 2.2 Applied Diagrams

**A. UML**

UML이란, Unified Modeling Language, 통합 모델링 언어이다. 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어로, 이 표준은 OMG(Object Management Group)에서 관리한다.

UML을 이용하여 시스템을 모델로 표현할 수 있으며, 객체 지향적으로 문제를 해결하는 데에 도움을 준다. UML의 구성요소는 구조, 행위, 그룹

등의 사물과 각 사물 간의 관계, 그리고 이를 나타내는 diagram이 있다. Diagram을 통해서 본 문서의 예상 독자가 본 시스템의 structure, flow, interface를 보다 잘 이해할 수 있게 한다. UML diagram에는 activity diagram, use case diagram, sequence diagram, class diagram, state diagram 등이 있다.

### B. Use Case Diagram

Use case diagram은 시스템과 시스템이 동작하는 주변 환경과의 상호작용을 그린 diagram이다. Use case diagram에는 Use case와 Actor가 있다. Use case란 시스템과의 external interaction을 수반하는 개별 task를 나타낸 것이고, Actor는 시스템을 사용하는 사람 혹은 타 시스템이다. 즉, Actor가 본 시스템에서 어떤 기능을 어떻게 사용하는지를 보여준다.

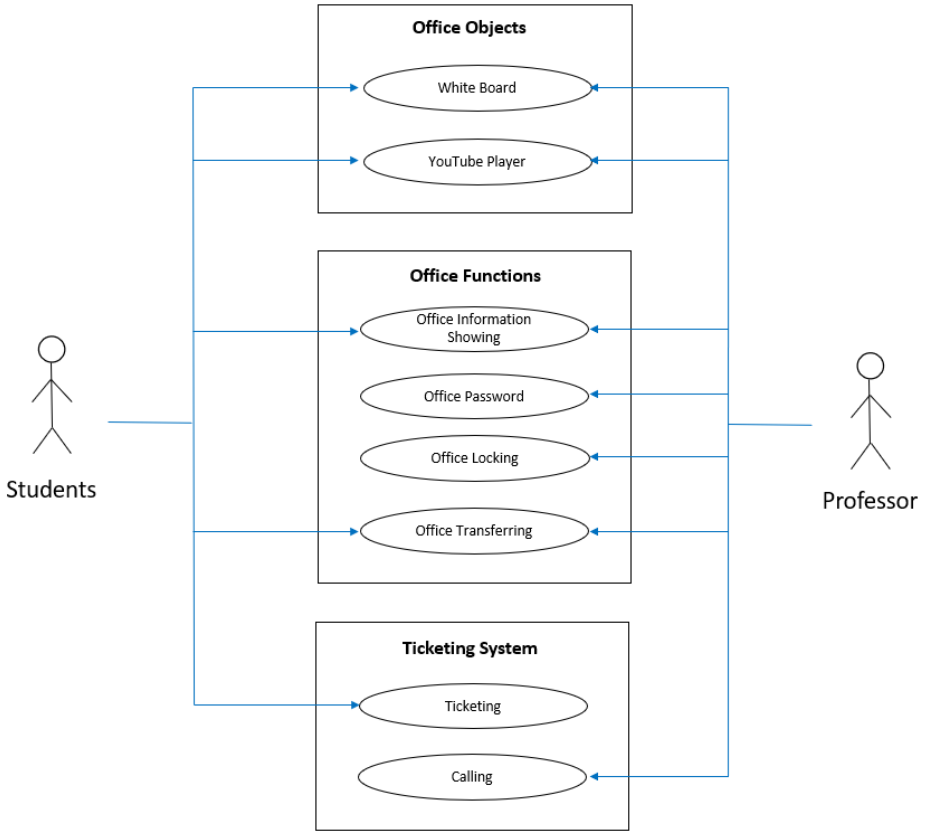


Figure 1: Use case diagram

### C. Sequence Diagram

Sequence diagram은 시스템 내부에서 actor와 시스템 구성 요소 간의 상호작용을 모델링하는 데에 사용된다. Sequence diagram은 특정한 use case에서 발생하는 상호작용의 흐름을 시간 순서대로 보여준다.

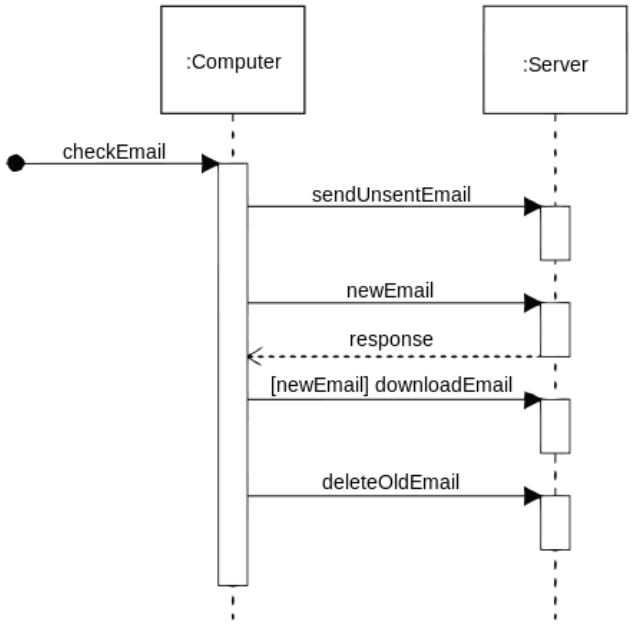


Figure 2: Sequence diagram의 예

### D. Class Diagram

Class diagram은 시스템의 클래스, 클래스의 attribute와 method, 클래스 간의 관계를 표현한 static structure diagram이다. Class diagram은 object-oriented modeling의 기본이 된다. 시스템 구조의 general conceptual modeling 뿐만 아니라 model을 programming code로 변환하는 detailed modeling에도 활용될 수 있다.

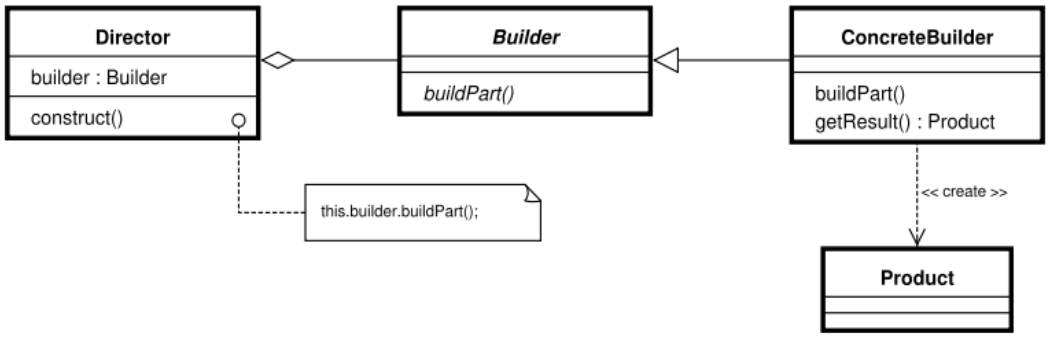


Figure 3: Class diagram의 예

## 2.3 Applied Tools

### A. Lucidchart

Lucidchart는 각종 chart나 diagram을 그리고 수정할 수 있는 웹 기반 편집 tool이다. HTML5를 지원하는 browser에서 사용 가능하다. 회원 가입 이후 무료로 사용 가능하며, 무료 사용 시 수정 가능한 document의 수는 3개로 제한된다.

Figure 4: LucidChart

### B. Microsoft PowerPoint

Power Point는 MS사가 제공하는 PPT 제작 Tool으로, 다양한 도형, 아이콘 등을 삽입할 수 있고, 필요한 diagram을 그리는데 유용하게 사용되며, 텍스트도 삽입할 수 있다.

Figure 5: Microsoft PowerPoint

## 2.4 Project Scope

Team 14의 주제는 Neo Metaverse Office로, 'Smart campus in metaverse'의 범주 하에 design 되었다. Neo Metaverse Office는 교수자의 office hour 시간에 교수자와 학생 간의 meeting을 공간적 제약 없이 수행하도록 도와주는 Unity 기반 VRChat 메타버스 만남 공간이다. 본 시스템은 기존의 meeting 방식(직접 방문, 화상 회의 플랫폼 사용)의 불편함을 해소하기 위해서 White Board로 판서를 하는 기능, Youtube나 Google Drive를 사용하는 Video Player 기능, Office의 정보를 나타내는 기능, Office로 이동하는 기능, Office의 접근을 허용하고 막는 기능, Office 출입 대기 시스템인 Ticketing System 등으로 이루어져 있다.

## 2.5 References

이 문서를 작성하기 위해 사용한 references는 다음과 같다.

A. Team 11, 2021 Spring. Software Design Document, SKKU.

B. Appleton, Brad. A Software Design Specification Template. N.d.

C. https://commons.wikimedia.org/wiki/File:CheckEmail.svg

D. https://ko.m.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:Builder_UML_class_diagram.svg

# 3. System Architecture

## 3.1 Objective

System Architecture에서는 전체 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트 사이의 관계를 기술한다.

## 3.2 System Organization

본 시스템은 교수-학생 듀얼 인터페이스의 형식으로 설계되어 있다. 교수가 방을 만들고 학생의 신청을 동의한다. 학생은 교수님의 방을 찾을 수 있고 최종 신청이 완료된 후에는 방에서 White Board 등의 기능을 사용할 수 있다.

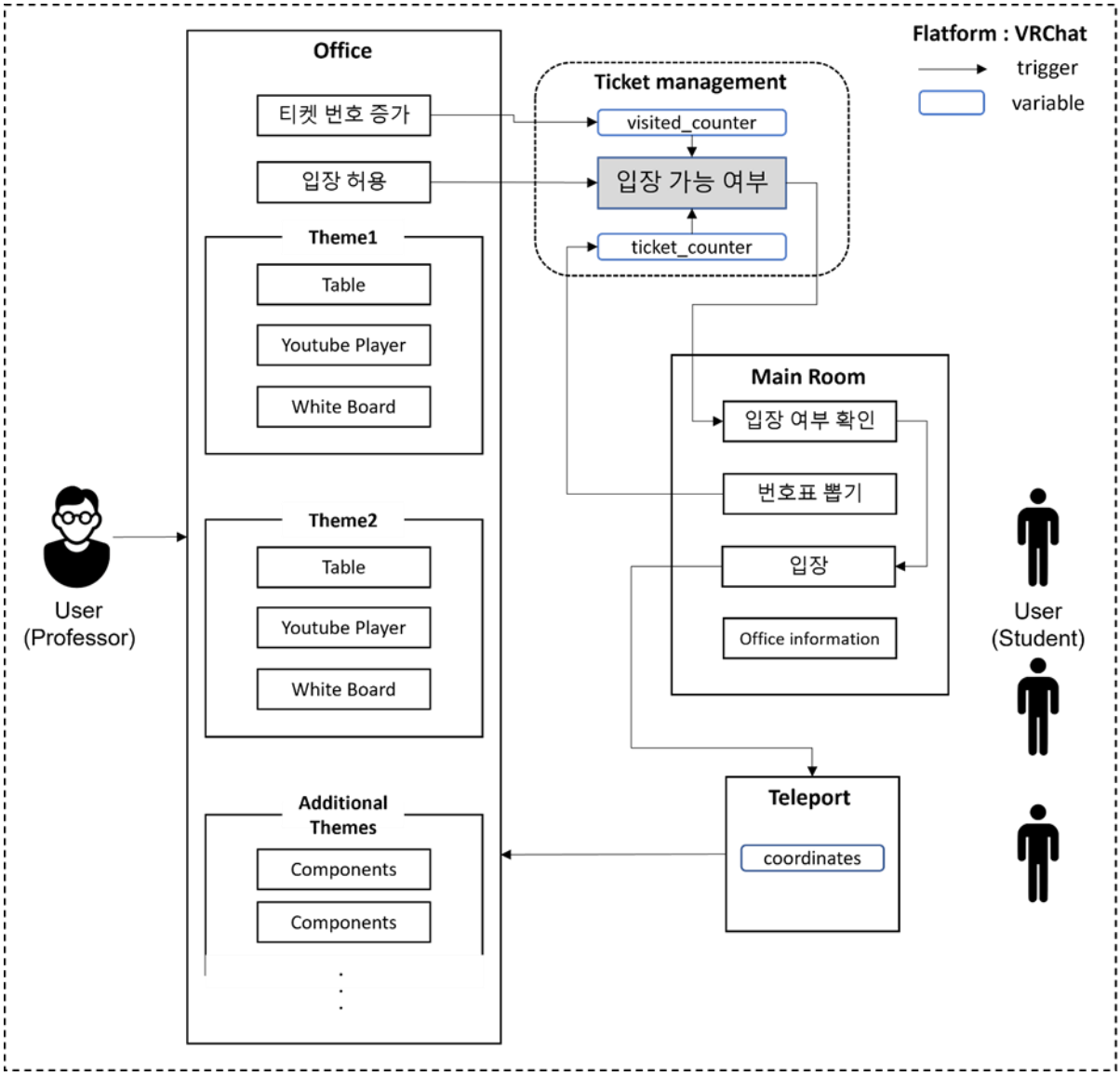


Figure 6: Overall System Architecture

### A. Student

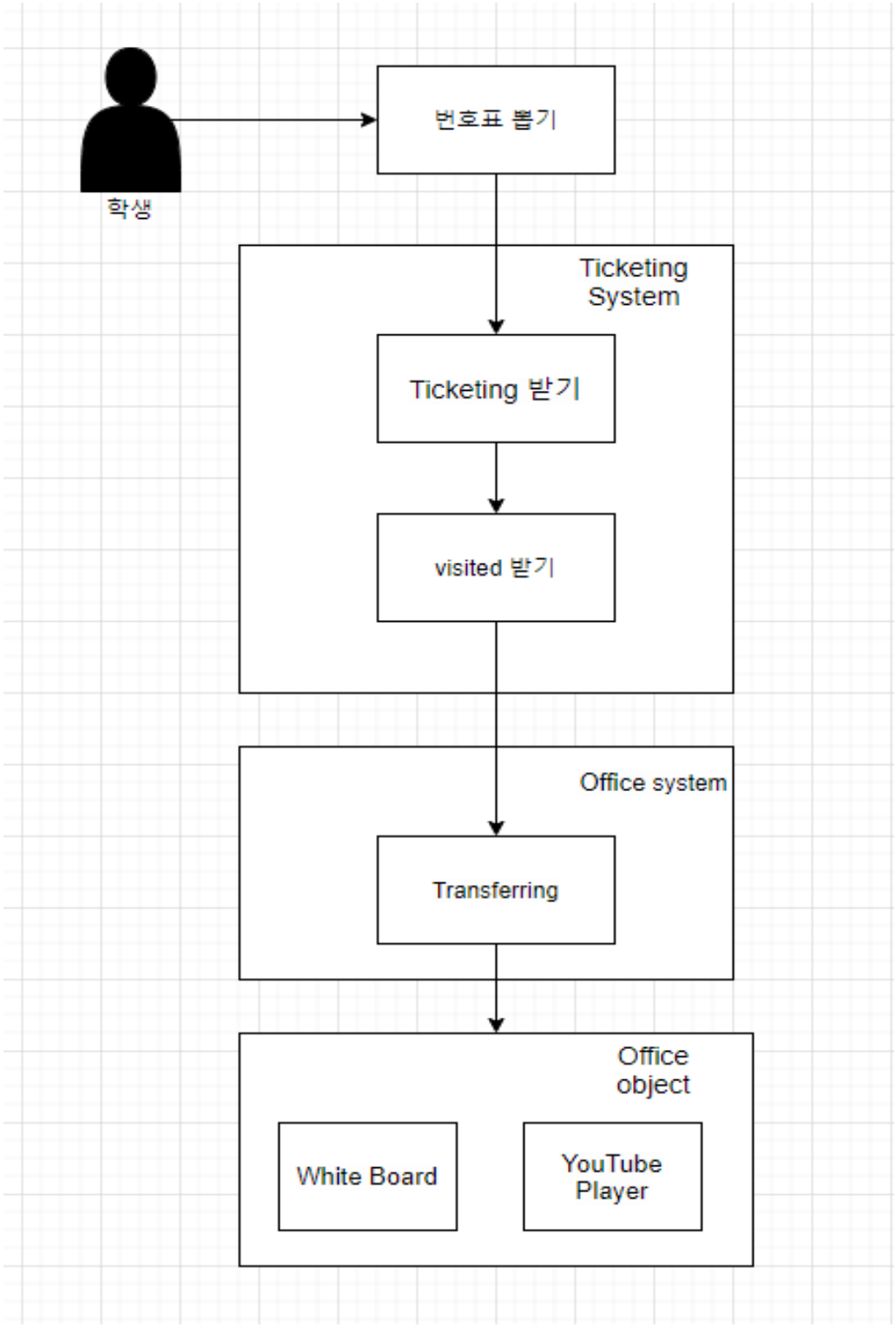


Figure 7: System Architecture for student

학생들은 Ticketing System을 통해 대기번호를 받고 교수의 요청을 받아

Office Transferring을 통해 방으로 전송되며, 방안에서는 학생들이 White Board, YouTube Player 등의 기능을 이용해 교수에게 질문할 수 있다.

### B. Professor

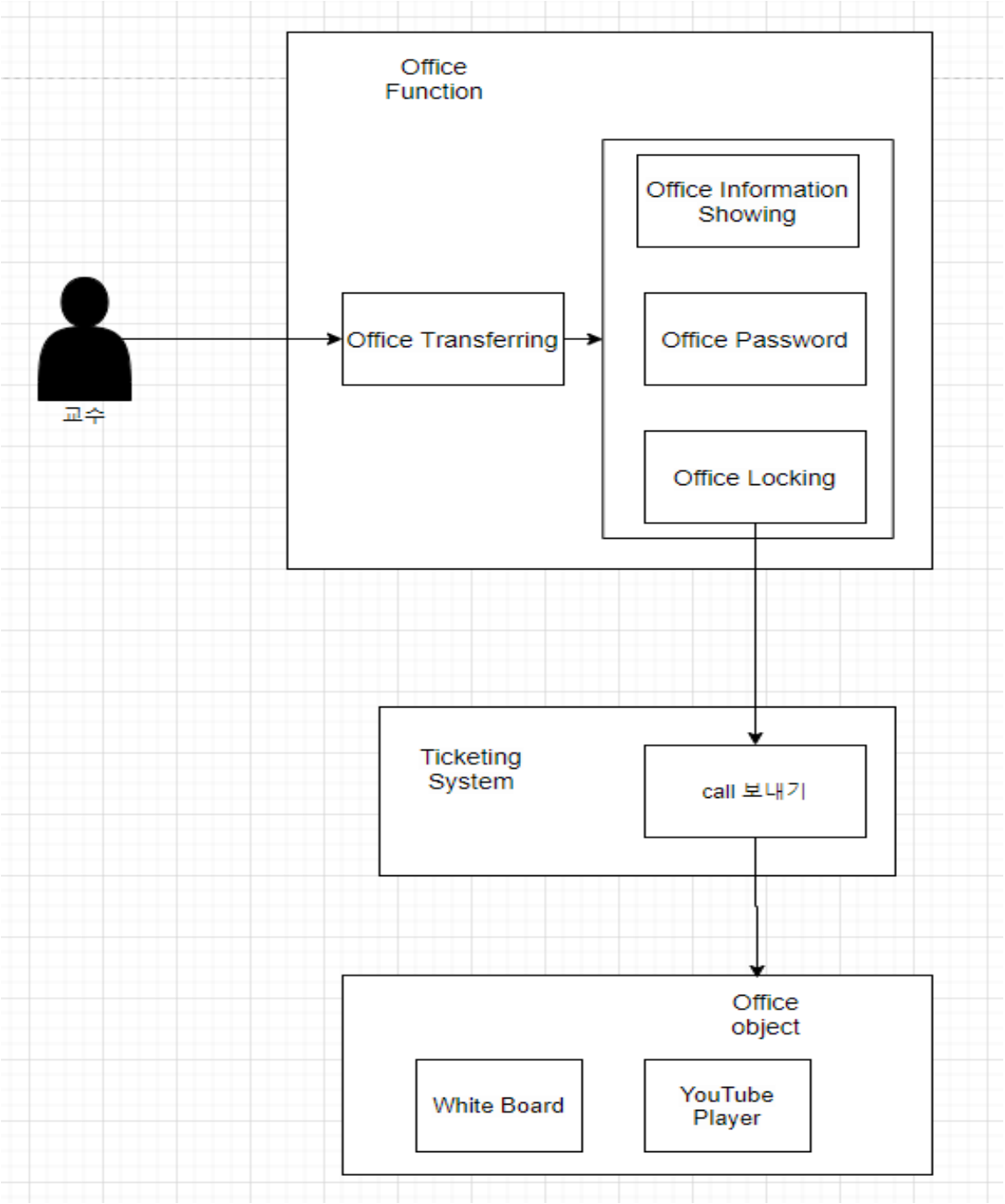


Figure 8: System Architecture for professor

교수는 Office Transferring을 통해 방으로 전송되며 Office Function의 기능을 통해 방을 수정할 수 있다. 학생들은 교수가 신청을 수락해야 방에 들어갈 수 있으며 이후 교수는 White Board , YouTube Player 등의 기능을 통해 학생들의 질문에 답변한다.

## 3.3 Components

### 3.3.1 Context Diagram

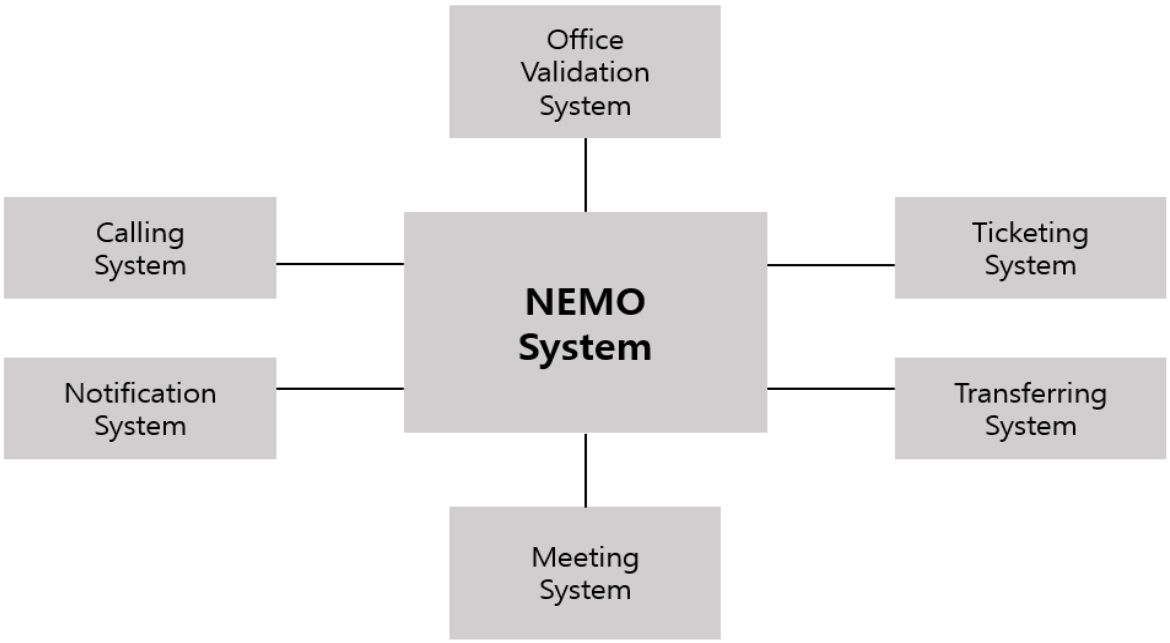


Figure 9: System Context Diagram

### 3.3.2 Class Diagram

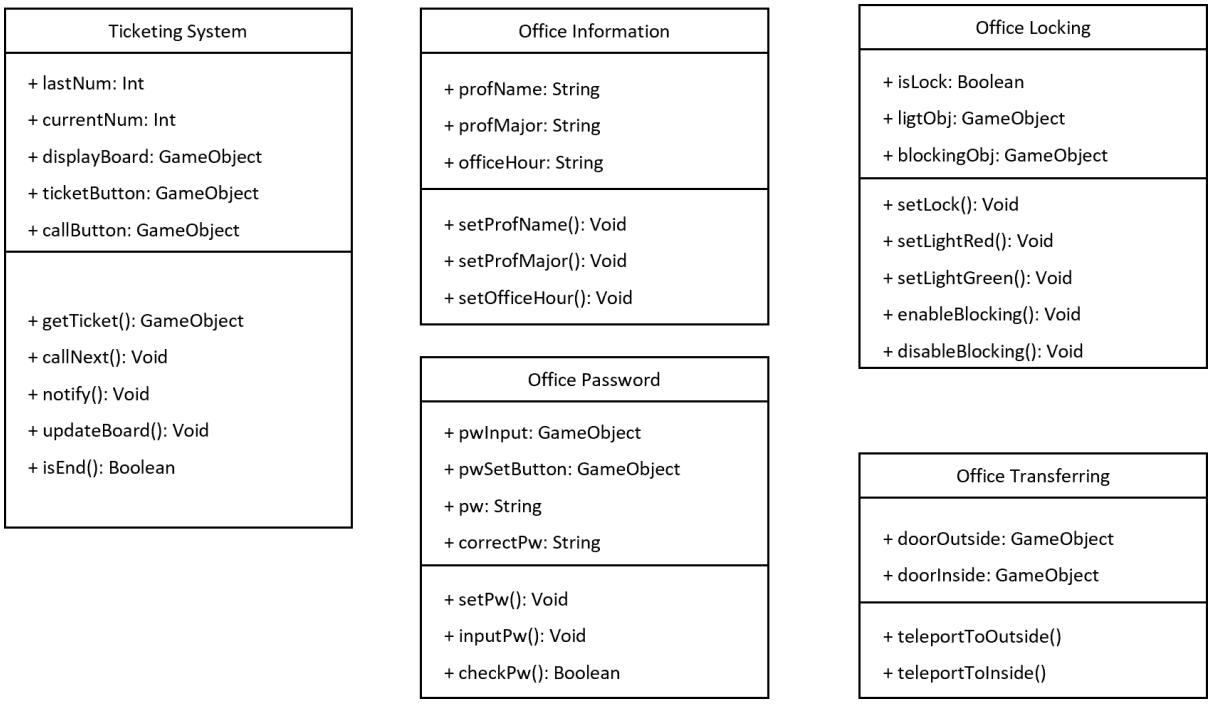


Figure 10: System Class Diagram

### A. Ticketing System

| Element | | Description |
|---|---|---|
| Attributes | +lastNum | 가장 마지막으로 뽑힌 티켓 순번 |
| | +currentNum | 현재 회의를 진행중인 학생의 티켓 순번 |
| | +displayBoard | 진행중 또는 대기중인 순번을 보여주는 오브젝트 |
| | +ticketButton | 티켓을 뽑는 버튼 |
| | +callButton | 다음 학생을 부르는 버튼 |
| Methods | +getTicket() | 티켓을 뽑는다 |
| | +callNext() | 다음 학생을 부른다. |

| | | |
|---|---|---|
| | +notify() | 학생에게 알림 메시지가 보인다. |
| | +updateBoard() | 순번이 바뀔 때 보드의 정보를 업데이트한다. |
| | +isEnd() | 마지막 티켓번호인지 확인한다. |

**B. Office Information**

| Element | | Description |
|---|---|---|
| Attributes | +profName | 해당 Office 에 배정된 교수님의 성명 |
| | +profMajor | 해당 Office 에 배정된 교수님의 전공 |
| | +officeHour | 해당 OfficeHour 의 시간 |
| Methods | +setProfName() | 교수님의 성명을 입력 받아 설정한다. |
| | +setProfMajor() | 교수님의 전공을 입력 받아 설정한다. |
| | +setOfficeHour() | Office Hour 시간을 입력 받아 설정한다. |

**C. Office Password**

| Element | | Description |
|---|---|---|
| Attributes | +pwSetButton | 비밀번호 설정 버튼 |
| | +pwInput | 비밀번호를 입력할 오브젝트 |
| | +password | 입력한 비밀번호 |
| Methods | +setPw() | 비밀번호를 설정한다. |
| | +inputPw() | 비밀번호를 입력한다. |
| | +checkPw() | 비밀번호를 맞게 입력했는지 확인한다. |

### D. Office Locking

| Element | | Description |
|---|---|---|
| Attributes | +isLock | Office 의 잠금 여부 |
| | +lightObj | 현재 회의를 진행중인 학생의 티켓 순번 |
| | +blockingObj | 진행중 또는 대기중인 순번을 보여주는 오브젝트 |
| Methods | +setLock() | Office 를 잠근다. |
| | +setLightRed() | 오브젝트의 색깔을 빨간색으로 바꾼다. |
| | +setLightGreen() | 오브젝트의 색깔을 초록색으로 바꾼다. |
| | +enableBlocking() | 문 앞을 막는 오브젝트를 활성화한다. |
| | +disableBlocking() | 문 앞을 막는 오브젝트를 비활성화한다. |

### E. Office Transferring

| Element | | Description |
|---|---|---|
| Attributes | +doorOutside | 바깥쪽 문 (Hall) |
| | +doorInside | 안쪽 문 (Office) |
| Methods | +teleportToOutside() | 사용자를 Office 밖으로 이동시킨다. |
| | +teleportToInside() | 사용자를 Office 안으로 이동시킨다. |

### 3.3.3 Sequence Diagram

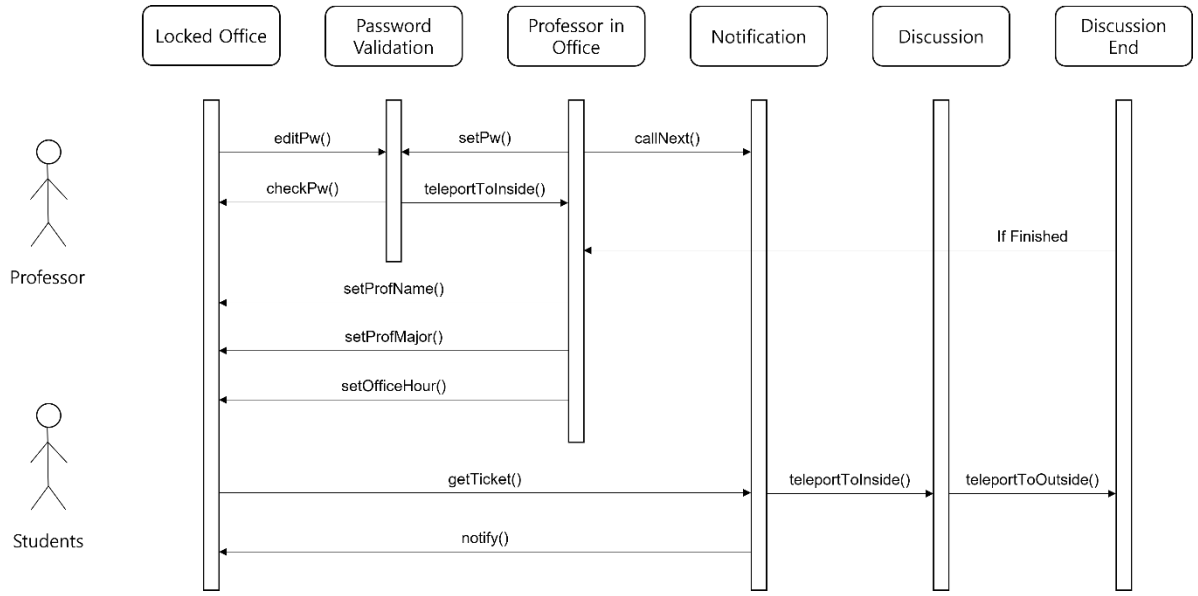


Figure 11: System Sequence Diagram

# 4. Testing Plan

## 4.1 Objective

이 chapter에서는 NEMO 시스템을 개발하는 과정에서부터 배포 이후 사용자가 직접 시스템을 사용해보기까지의 시스템 테스팅에 관한 계획을 설명한다. 테스팅 과정은 시스템의 잠재적인 결함과 버그를 발견하여 수정하고, 사용자의 요구명세서를 충족하는지 확인하며, 개발 환경과 사용 환경 간의 차이를 확인 및 반영할 수 있다는 점에서 그 중요성이 매우 큰 필수적인 단계이다. NEMO 시스템은 크게 unit testing, integration testing, system testing, 그리고 acceptance testing의 4단계 testing을 거칠 계획이다.

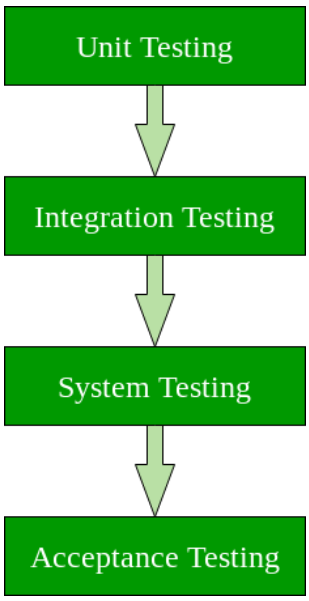

Figure 12: Testing Plan

## 4.2 Testing Policy

### 4.2.1. Unit Testing

Unit(Component) testing 단계에서는 시스템을 구성하는 각각의 unit/component가 문제없이 작동되는지를 확인한다. NEMO 시스템은 여러 맵으로 구성되어 있으며 각각의 공간 안에는 특정한 기능을 수행하는 버튼, Youtube 동영상을 재생하는 플레이어 등 많은 구성요소들이 포함되어 있다. Unit testing의 목표는 이러한 구성 요소들과 각각의 공간들이 정상적으로 작동함을 확인하고 문제가 있을 경우 수정하는 데에 있다. 이러한 unit/component testing의 과정 없이 각각의 구성요소를 하나의 시스템으로 통합하고자 할 경우, 이후에 문제가 생겼을 때 원인이 어느 부분에 있는지를 확인하기가 어렵기 때문에 이 과정은 필수적이라 할 수 있다.

### 4.2.2. Integration Testing

각각의 구성요소에 대한 testing을 마친 이후에는 이 구성요소를 하나씩 결합하며 각 component 간의 interaction이 정상적으로 작동하는지를 확인할 계획이다. 단, 모든 구성요소를 결합한 뒤 테스팅을 진행할 경우 버그의 원인을 찾아내기 어렵기 때문에, component를 하나씩 결합해가며 테스팅을 여러 번 진행하고자 한다. 이때 각 component의 interface 정보를 공유하고 문제가 생겼을 경우 이를 해결하기 위한 툴로 Github을 활용한다.

### 4.2.3. System Testing

모든 구성요소를 결합하여 완성된 하나의 시스템/소프트웨어에 대해서 testing을 진행한다. 특히 이 단계에서는 emergent properties(창발적 속성)에 대한 테스팅이 중요하다. 그 중에서도 NEMO system testing에서는 performance와 reliability에 집중할 계획이다.

#### 4.2.3.1 Performance

NEMO 시스템은 교수님 오피스, 대기 공간 등 여러 공간으로 구성되어 있다. 이 시스템을 사용하는 사용자는 이 공간들을 옮겨 다녀야 하기 때문에 공간을 이동할 때 소요되는 시간이 짧아야 한다. 따라서 최종 시스템에서 공간 이동에 소요되는 시간은 3초 내로 구현될 수 있도록 하며 여러 사용자가 동시에 시스템에 접속할 경우, 2인 이상의 사용자가 같은 장소로 동시에 이동할 경우 등 여러 상황을 가정하여 테스팅을 진행할 예정이다. 또한 NEMO의 중요한 구성요소로서 티켓팅 시스템이 있는데, 이 ticketing은 시간/순서와 직접적인 연관이 있으므로 delay가 발생해서는 안 된다. 따라서 학생이 번호표를 뽑는 경우, 교수가 특정 번호를 호출하는 경우 등 티켓팅과 관련한 기능들은 모두 1초 이내로 작업이 완료될 수 있도록 한다. 마지막으로 Youtube 플레이어, 화이트보드 등 기타 구

성요소의 사용에 있어서도 3초 내로 반응이 올 수 있도록 구현 및 테스팅을 진행할 예정이다.

#### 4.2.3.2 Reliability

시스템에 대한 사용자의 신뢰도를 높이기 위해선 시스템을 사용할 때 발생할 수 있는 오류를 제거하는 것이 중요하다. 이를 위해서는 시스템을 구성하는 각각의 component가 오류 없이 구현되어야 할 뿐만 아니라, component 사이의 잘못된 상호작용에 의해 예상치 못한 문제가 발생하지 않도록 하나의 시스템으로 결합해 나가는 과정이 필수적이다. 따라서 앞서 설명하였던 unit testing에 더불어 integration testing을 반복하며 최종적으로 결합된 하나의 시스템이 작동하는 데 있어 문제가 없는지를 이 단계에서 마지막으로 확인할 계획이다.

### 4.2.4. Acceptance Testing

Acceptance testing은 시스템이 배포되기 전 진행되는 마지막 testing으로서, 개발된 시스템을 배포할 수 있는지를 결정하는 단계이기 때문에 매우 중요하다. 이 테스팅은 크게 시스템 개발자들이 수행하는 테스팅과 시스템 사용자들이 수행하는 테스팅 2가지로 나누어진다. 먼저 시스템의 개발자들은 앞서 작성된 Software Requirements Specification(SRS)의 4가지 scenario(Connection scenario, Ticketing scenario, Meeting scenario, 그리고 Transfer scenario)가 정상적으로 수행되는지를 직접 실행해보며 확인한다.

그러나 시스템의 개발환경과 사용자들의 사용환경 간에는 분명한 차이가 있다. 따라서 시스템을 배포하기 전 이 테스팅 단계에서는 사용자들이 직접 그들의 환경에서 각자의 데이터를 가지고 시스템을 사용할 때 문제가 없는지를 필수적으로 확인하여야 한다. NEMO 시스템은 총 20명의 사용자

에게 베타버전에 대한 평가를 받을 예정이며, 평가 항목은 시스템의 성능에 관한 performance, 시스템의 기능적 완성도와 신뢰도에 관한 reliability, 그리고 시스템을 얼마나 쉽고 간단하게 조작할 수 있는지에 관한 usability를 중점으로 할 계획이다.

# 5. Development Plan

## 5.1 Objective

시스템을 구현하는 데 사용하는 언어, 라이브러리, 플랫폼 등의 개발환경을 설명한다.

## 5.2 Develop Environment

### 5.2.1. Unity


Figure 13: Unity

유니티는 2D와 3D 모두 통틀어서 게임, 애니메이션과 같은 상호작용하는 콘텐츠를 만드는데 사용되는 게임엔진이다. 유니티를 이용해 스크립트를 구성하는데 사용하는 프로그래밍 언어로는 C#, 자바스크립트를 사용한다. VRChat SDK3를 사용해서 월드를 구성하고, 원하는 기능을 구성하는데 사용한다.

### 5.2.2. VRChat SDK3

Figure 14: VR CHAT

위에서 설명한 Unity 엔진을 기반으로 하는 가상현실 지향 음성 채팅 소프트웨어이다. Unity엔진을 이용하기 때문에 본인이 원하는 아바타 및 월드를 구현할 수 있는 것이 특징이다.

패키지로는 총 2가지가 있다. 월드를 구현하는 데 쓰이는 VRCSDK-WORLD, 아바타를 구현하는 데 쓰이는 VRCSDK-AVATAR이다. 하지만 이번 프로젝트에서는 아바타보다는 월드의 기능 구현이 우선이기 때문에 VRCSDK-WORLD에 집중한다. WORLD 패키지 내에는 다양한 사물을 구현하거나 기능이 추가된 거울, 앉고 일어날 수 있는 의자와 같은 기능들이 포함되어 있다.

### 5.2.3. Udon


Figure 15: UDON

VRChat Udon은 VRChat에서 자체 개발한 프로그래밍 언어이다. Node, wire를 이용해 입력, 출력을 연결하는 내장 프로그래밍 인터페이스인 VRChat Udon을 통해 안전하고 성능이 우수하며 사용하기 쉽도록 설계됐다. Trigger Action의 전체 동작을 VRChat Udon으로 복제할 수 있을 뿐만 아니라 자신만의 동작을 만들 수 있다. 또한 변수를 다른 사람과 동기화하며 장면과 상호 작용하고 User 간 상호 작용할 수 있다. VRChat Udon은 VRChat Client와 Unity Editor 모두 실행돼 손쉽게 창작물을 테스트하고 디버깅할 수 있다.

### 5.2.4. Github


Figure 16: GitHub

Github는 버전 관리와 협업을 위한 코드 Hosting Platform이다. 협업 프로젝트를 보다 쉽게 하기 위해서 사용한다. 깃허브를 통해서 팀원들이 프로젝트를 진행함에 용이하고, 쉽게 Integration 작업을 진행할 수 있다.

## 5.3 Constraints

시스템은 이 문서에 언급된 내용을 기반으로 하여 설계 구현된다. 밑의 사항을 준수한다면 그 외 부분은 개발자의 선호방식에 따라 구현될 수 있다.

- Unity 2019.4.30f1, VRChat SDK3을 사용하는 VRChat World를 구현한다.

- 실제 사무실과 최대한 비슷하도록 World를 구현하고, 기능을 구현할 때는 최적화 작업이 필요하다.
- 최대한 Open Source를 reuse하는 방향으로 구현한다.
- 시스템 개발뿐만 아니라 유지보수도 고려한다.

### 5.3.1. System Constraint

해당 시스템은 VRChat 기반이기 때문에 VRChat의 최소 사양을 맞춰야 시스템이 작동할 수 있다. VRChat의 최소 사양은 다음과 같다.

| 시스템 요구 사항 | |
|---|---|
| 구분 | 최소 사양 |
| 운영체제 | Windows 7<br>Windows 8.1<br>Windows 10 |
| 프로세서 | Intel® i5-4590 이상<br>AMD FX 8350 이상 |
| 램 | 4GB |
| 그래픽 카드 | NVIDIA GeForce® GTX 970 이상<br>AMD Radeon™ R9 290 이상<br>Intel UHD Graphics 610 이상 |
| DirectX | Version 11 |
| 저장 공간 | 1GB |

Figure 17: System Constraint for VRChat

## 5.4 Assumption Dependencies

이 문서의 시스템은 최소 Window 7 이상의 OS에서 Unity 2019.4.30f1을 기반으로 하는 VRChat SDK3에서 개발된다. Window 이외의 운영체제 및 Window 7 미만의 운영체제에서는 실행되지 않을 수 있다.

# 6. Supporting Information

## 6.1 Software Design Specification

이 문서는 IEEE Recommendation (IEEE Recommend Practice for Software Design Description, IEEE-Std-1016) 형식에 맞추어 작성되었다.

## 6.2 Figures

## 6.3 Tables



## 6.4 Document History

| Date | Version | Description | Writer |
|---|---|---|---|
| 21/11/16 | 1.0 | Part 1,2 | 박세환 |
| 21/11/17 | 1.5 | Part 3 | 이기요,김동락,조민재 |
| 21/11/18 | 2.0 | Part 4 | 이은지 |
| 21/11/19 | 3.0 | Part 5 | 나경호 |
| 21/11/20 | 4.0 | Integration | 나경호 |
| 21/11/21 | 5.0(final) | 최종 수정 | 모든 팀원 참여 |