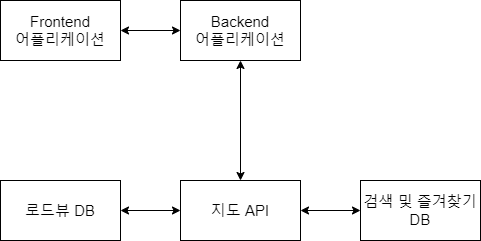
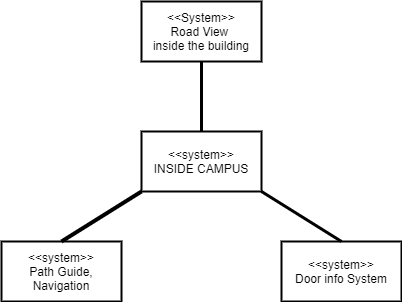
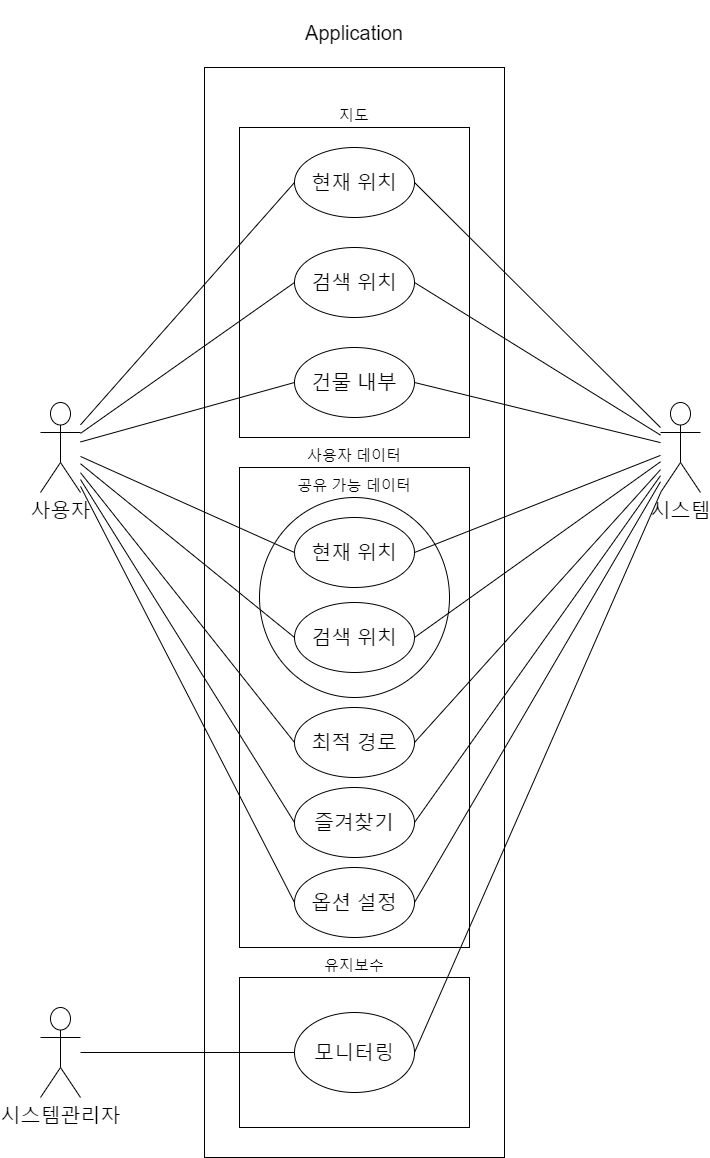
1. Preface  
   이 챕터는 독자층에 대한 정보와 디자인 명세서의 목적, 범위, 독자층, 그리고 INSIDE CAMPUS의 소프트웨어 디자인 명세서의 구조가 포함되어 있습니다.
   1. Readership  
      이 소프트웨어 디자인 명세서는 다양한 소제목을 포함한 10가지 부분으로 나뉘어져 있습니다. 그리고 이 명세서의 메인 독자층은 Team13이고, 추가적으로 교수님과 조교님들, 그리고 소프트웨어 공학 개론을 수강하고 있는 학생들 또한 메인 독자층이 될 수 있습니다.
   2. Scope  
      이 디자인 명세서는 캠퍼스 지도를 일반적인 도로와 건물위치 뿐만 아니라 건물 내부까지 표시해주고, 위치 검색 및 원하는 위치까지의 경로를 표시하는 시스템을 구현하는데 사용할 소프트웨어 엔지니어링 및 소프트웨어 품질 엔지니어링에서 사용됩니다.
   3. Objective  
      이번 소프트웨어 디자인 명세서의 본 목적은 INSIDE CAMPUS 앱의 기술적 디자인 측면에 대한 설명을 제공하는 것입니다. 이 문서는 INSIDE CAMPUS를 구현하는 데에 필요한 소프트웨어 아키텍쳐와 소프트웨어 디자인을 설명합니다. 또한 시스템의 다른 측면을 묘사하기 위한 시스템의 구조 개요를 제공합니다.   
      추가적으로 SRS문서 (요구사항 명세서)에서 논의된 일부 모듈의 구조와 설계를 명시하고, 프로그래밍 팀이 특정 모듈을 구현하는 방법을 보여주는 클래스 다이어그램을 포함하여 순차 및 활동 다이어그램으로 변환된 일부 사용 사례를 표시합니다.   
      이 명세서의 열람이 가능한 사람은 INSIDE CAMPUS 앱의 이해관계자, 개발자, 설계자 및 소프트웨어 테스터입니다.
   4. Document Structure
      * 1.Preface: 이 챕터는 독자층과 명세서의 범위, 시스템의 목적, 그리고 명세서의 구조를 설명합니다.
      * 2.Introduction: 이 챕터는 이 명세서에 사용된 몇가지 도구와, 명세서와 레퍼런스에 사용된 몇가지 다이어그램, 그리고 이 프로젝트의 목적을 설명합니다.
      * 3.Overall System Architecture: 이 챕터는 컨텍스트 다이어그램, 시퀀스 다이어그램, 유즈 케이스 다이어그램을 이용해 시스템의 전체적인 아키텍처를 설명합니다.
      * 4.System Architecture - Frontend: 이 챕터는 클래스 다이어그램과 시퀀스 다이어그램을 이용해 프론트엔드 시스템의 아키텍처를 설명합니다.
      * 5.System Architecture - Backend: 이 챕터는 클래스 다이어그램과 시퀀스 다이어그램을 이용해 백엔드 시스템의 아키텍처를 설명합니다.
      * 6.Protocol Design: 이 챕터는 클라이언트와 서버의 커뮤니케이션을 위한 몇 가지 프로토콜 디자인을 설명합니다.
      * 7.Database Design: 이 챕터는 ER 다이어그램과 SQL DDL을 이용한 데이터베이스 디자인을 설명합니다.
      * 8.Testing Plan: 이 챕터는 시스템의 테스트 계획을 설명합니다.
      * 9.Development Plan: 이 챕터는 시스템을 개발하기 위해 사용된 도구와 제약 조건, 가정 및 종속성을 설명합니다.
      * 10.Supporting Information: 이 챕터는 이 명세서의 기록과 기준을 설명합니다.
2. Introduction  
   이 프로젝트는 캠퍼스 내의 상세한 지도를 표현함으로써 신입생들이나 학교를 처음 방문하는 사람들에게는 보다 정확한 위치를 표시해주고 현위치에서 원하는 위치까지의 경로 및 시간을 표시해줍니다. 또한 기존 재학생들과 학교 관계자분들, 혹은 캠퍼스에 자주 방문하신 분들에게도 생소한 위치나 건물 내부의 강의실 위치, 해당 강의실의 강의 시간 등을 자세히 알려줍니다. 이 디자인 명세서에는 프로젝트 구현에 사용되거나 사용될 설계가 나와 있습니다. 설명된 디자인은 프로젝트를 위해 미리 준비한 소프트웨어 요구사항 명세서에 지정된 요구사항을 따릅니다.
   1. Objectives  
      이 챕터에서는 디자인 단계에서 이 프로젝트에 적용한 다양한 도구와 도표를 설명합니다.
   2. Applied Diagrams
      * UML (통합 모델링 언어)  
        소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어이다. 이 표준은 UML을 고안한 객체 관리 그룹에서 관리하고 있다. UML은 데이터 모델링(개체-관계 모델)과 비즈니스 모델링(업무 흐름), 객체 모델링, 부품 모델링의 최선의 기술을 조합한다. UML은 소프트웨어 개발 프로세스뿐만 아니라 다른 구현 기술의 모든 공정에서 사용될 수 있다. UML은 Booch 방법론의 객체 모델링 기법(OMT)와 객체 지향 소프트웨어 공학(OOSE)을 광범위하게 사용할 수 있는 단일한 공통 모델링 언어로 통합한다. UML의 목표는 동시적 분산 시스템을 모델링 하는 표준 언어다. UML은 산업의 실질적 표준으로서, 객체 관리 그룹(OMG)에 의해 개선되고 있다. 초기에 OMG가 엄격한 소프트웨어 모델링 언어를 만들기 위해 객체 지향 방법론적인 통지를 요청했고, 많은 산업 선구자가 UML 표준 제작을 돕기위해 진지하게 응답하였다. UML 모델은 객체 관리 그룹이 지원하는 QVT와 같은 변환 언어 등을 이용해 다른 표현(예를 들면 자바)으로 자동적으로 변환된다. UML은 확장할 수 있으며 커스터마이제이션을 위한 메커니즘인 프로파일 (UML), 스테레오타입 (UML)을 제공한다. 프로파일을 이용한 확장의 의미는 UML 2.0에서 개선되었다.
      * Use case Diagram  
        유스 케이스 다이어그램은 사용자, 그리고 사용자가 수반한 다른 유스 케이스 간의 관계를 보여주는 사용자-시스템 간 상호작용의 표현이다. 유스 케이스 다이어그램은 각기 다른 종류의 시스템 사용자와 각기 다른 유스 케이스를 식별할 수 있으며 다른 유형의 다이어그램이 수반되기도 한다. 유스 케이스는 원이나 타원으로 표현된다.
      * Sequence Diagram  
        시퀀스 다이어그램은 컴퓨터 과학 커뮤니티뿐만 아니라 비즈니스 애플리케이션 개발을 위한 설계 레벨 모델로서도 아마도 가장 중요한 UML 다이어그램일 것입니다. 최근에는 시각적 자기 설명적 특성 때문에 비즈니스 프로세스를 묘사하는 데 인기를 얻고 있습니다. 이름에서 알 수 있듯이, 시퀀스 다이어그램은 행위자와 객체 간에 발생하는 메시지 및 상호 작용의 순서를 설명합니다. 행위자 또는 사물은 필요하거나 다른 사물과 통신하려는 경우에만 활성화될 수 있습니다. 모든 통신은 시간순으로 표시됩니다.
      * Class Diagram  
        UML의 클래스 다이어그램은 시스템의 클래스, 속성, 작업(또는 메서드) 및 개체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조 다이어그램의 한 유형입니다. 클래스는 객체의 구성 블록이므로 클래스 다이어그램은 UML의 구성 블록입니다. 클래스 다이어그램의 다양한 구성 요소는 실제로 프로그래밍될 클래스, 주 객체 또는 클래스와 객체 사이의 상호 작용을 나타낼 수 있습니다. 클래스 모양 자체는 세 개의 행이 있는 직사각형으로 구성됩니다. 맨 위 행에는 클래스의 이름이 포함되고 가운데 행에는 클래스의 속성이 포함되며 맨 아래 섹션에는 클래스에서 사용할 수 있는 메서드 또는 작업이 표시됩니다. 클래스와 하위 클래스가 함께 그룹화되어 각 개체 간의 정적 관계를 표시합니다.
      * Context Diagram  
        시스템 컨텍스트 다이어그램(레벨 0 DFD)은 데이터 흐름 다이어그램에서 가장 높은 레벨이며 모델링할 시스템의 컨텍스트와 경계를 설정하는 전체 시스템을 나타내는 하나의 프로세스만 포함합니다. 이는 시스템과 외부 실체(즉, 행위자) 사이의 정보 흐름을 식별합니다. 상황별 다이어그램은 일반적으로 요구 사항 문서에 포함되어 있습니다. 이 내용은 모든 프로젝트 관계자가 읽어야 하므로 이해 관계자가 항목을 이해할 수 있도록 쉬운 언어로 작성해야 합니다. 시스템 컨텍스트 다이어그램의 목적은 전체 시스템 요구 사항 및 제약 조건 집합을 개발할 때 고려해야 하는 외부 요인 및 이벤트에 주의를 집중하는 것입니다. 시스템 컨텍스트 다이어그램은 프로젝트 초기에 조사 대상 범위를 결정하는 데 종종 사용됩니다. 따라서, 문서 내에 있습니다. 시스템 컨텍스트 다이어그램은 시스템과 상호 작용할 수 있는 모든 외부 엔터티를 나타냅니다. 전체 소프트웨어 시스템이 단일 프로세스로 표시됩니다. 이러한 다이어그램은 내부 구조에 대한 세부 정보가 없는 중앙의 시스템을 모든 외부 도면요소, 상호 작용하는 시스템 및 환경으로 표시합니다.
      * ERD (Entity Relationship Diagram)  
        ERD는 데이터베이스에 저장된 엔티티 세트의 관계를 보여줍니다. 이 컨텍스트에서 엔티티는 개체이며 데이터의 구성 요소입니다. 엔티티 집합은 유사한 엔티티의 컬렉션입니다. 이러한 엔티티는 속성을 정의하는 속성을 가질 수 있습니다. 엔티티, 속성을 정의하고 개체 간의 관계를 표시함으로써 ER 다이어그램은 데이터베이스의 논리적 구조를 보여줍니다. 엔티티 관계 다이어그램은 데이터베이스 설계를 스케치하는 데 사용됩니다.
   3. Applied Tools
      * Draw.io  
        다이어그램을 그리는 데에 사용한 사이트 주소입니다. 전반적인 다이어그램을 그리는 데에 필요한 모양이나 선 등이 준비되어 있으며, 사이트에서 작성된 다이어그램은 사이트 내에서 열 경우, 언제 어디서든 누구나 수정이 가능하게끔 되어 있습니다.
      * Aquerytool.com  
        ERD를 보다 쉽게 표현할 수 있는 사이트 주소입니다. 웹 기반이라 인터넷만 있으면 시간과 장소에 제한이 없으며, 작성된 파일을 가진 누구나 수정이 가능합니다.
      * Gitmind.com  
        웹 기반 마인드맵핑 사이트입니다. 다양한 아이콘을 사용해 보다 직관적인 다이어그램을 그릴 수 있습니다.
   4. Project Scope  
      INSIDE CAMPUS는 캠퍼스 어디에 무엇이 있는지 잘 모르거나, 캠퍼스 내에서 강의실 혹은 특정 위치를 가고 싶을 때, 실제 거리와 문 위치, 복도 사진, 강의실 위치 등을 보여주면서 빠르고 정확하게 찾아가게끔 도와줍니다. 로드뷰 방식의 지도 데이터는 데이터베이스 서버에 저장하여 불러오는 방식을 채택할 것입니다.
   5. References
      * Team 1, 2020 Spring, Software Design Document, SKKU
3. System Architecture - Overall
   1. Objectives  
      프론트엔드 디자인부터 백엔드 디자인까지의 시스템 구성을 설명합니다.
   2. System Organization  
        
        
        
        
        
        
        
        
        
        
        
        
      저희는 client-server 모델을 적용시킨 시스템으로 디자인할 것입니다. Frontend 어플리케이션에서 이 앱의 모든 기능을 유저들에게 제공할 것이고, Backend 어플리케이션에선 Frontend 어플리케이션의 기능들에 사용되는 모든 데이터를 관리할 것입니다. Frontend와 Backend 어플리케이션은 JSON을 기반으로 데이터를 주고받습니다.

Figure system architecture - overall

* + - Context Diagram
    - Use case Diagram