Context Diagram & Architecture Design

|  |  |
| --- | --- |
| 프로젝트 이름 | 회의실 예약 시스템 |
| 조원 | 이지원(2013105080)  장효원(2013105088)  장호택(2013105087)  임영규(2013105084) |

|  |
| --- |
| 회의실 예약 시스템의 기본적인 구조 |
| 회의실 예약 시스템은 시스템을 총괄하는 메인프레임 컴퓨터를 기준으로 여러 명의 사용자들이 동시다발적으로 시스템에 접속하여 회의실을 등록하거나 등록을 삭제하고, 예약하거나 예약을 취소할 수 있다. 또한 이러한 회의실을 처리하는 데이터들은 실시간으로 처리되어야 한다. |

|  |
| --- |
| **Architecture Design** |
| 1. **MVC 패턴** |
| * 1. MVC가 현재 시스템에 적합한가? NO.   2. 적합하지 않은 이유는 무엇인가?      1. MVC패턴은 비즈니스 영역의 상태 정보를 처리하는 Model과 비즈니스 영역의 대한 결과 화면을 담당하는 View와 사용자의 요청 값에 처리 및 흐름 제어를 담당하는 Controller로 구성된다. 이러한 패턴은 개별 사용자가 시스템에 접근하는 방식으로는 적합하다. 하지만, 전체적으로 보았을 때 회의실 예약 시스템은 여러 사람들이 동시다발적으로 사용할 수 있어야 하고, 또한 사용 할 데이터가 asynchronize되어 입력되는데, MVC 패턴은 이러한 입력을 처리하는데 적절하지 않다. |
| 1. **Layered 아키텍처** |
| * 1. Layered 아키텍처가 현재 시스템에 적합한가? NO.   2. 적합하지 않은 이유는 무엇인가?      1. Layered 아키텍처는 크기가 큰 프로젝트에서 레이어를 여러 가지로 나누어 플랫폼에 독립적이고, 보안을 중요시 할 때 유용한 방식이다. 하지만 회의실 예약 시스템은 기능이 수직적이 아니라 같은 레벨에 있는 사용자들이 시스템을 이용하는 등 수평적인 기능들을 많이 가지고 있어 레이어를 여러 가지로 나누는 것이 불필요하다. |
| 1. **Repository 아키텍처** |
| * 1. Repository 아키텍처가 현재 시스템에 적합한가? NO.   2. 적합하지 않은 이유는 무엇인가?      1. Repository 아키텍처는 모든 데이터가 중앙 repository에서 관리하고 데이터는 오직 repository를 통해서만 접근할 수 있는 데이터를 중요시하는 모델이다. 이러한 모델은 매우 많은 양의 데이터가 생성되어 오랜 기간 동안 시스템에 저장되어야 할 때 적합한 모델이다. 이 모델을 사용하면 각 요소가 독립적 이여서 각 요소가 다른 요소의 존재에 대해 알 필요가 없다. 하지만 우리가 개발하고 있는 회의실 예약 시스템은 추가로 응용 프로그램을 붙일 일도 잘 없고, 데이터의 크기도 크지 않다. 또한 데이터 Evolution이 어렵기 때문에 적합하지 않다. |
| 1. **Client-Server 아키텍처 (적합한 모델)** |
| * 1. Client-Server 아키텍처가 현재 시스템에 적합한가? YES.   2. 적합한 이유는 무엇인가?      1. Client-Server 아키텍처는 시스템의 각 기능들이 서비스로 구성되고, 각 서비스가 분리된 서버에서 전달되는 방식이다. Client들은 이러한 서비스들의 사용자이고, 이러한 서비스를 사용하기 위해 서버에 접속한다. 이 모델은 여러 장소에서 접속할 필요가 있는 공유되는 데이터베이스 데이터를 사용할 때 사용한다. 우리가 개발하는 시스템은 공유되는 데이터를 사용하므로 이러한 모델이 적합하다. 또한, 다중 사용자가 동시다발적으로 접속하기에 편리하고, 또한 데이터를 Synchronize하기에 적합하므로 좋다. |

|  |
| --- |
| **Context Diagram** |
|  |