텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**변 경 이 력**

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **일자** | **변경 내역** | **작 성 자** |
| 1.0.0 | 2024.11.17 | 1차 작성 | 일조매 |
| 1.0.1 | 2024.12.14 | 2차 작성 | 일조매 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[***1.***](#_30j0zll) ***High level architecture 4***

[***2.***](#_3dy6vkm) ***Class Diagrams for static view*** 5

[***3.***](#_1t3h5sf) ***Sequence for dynamic view*** 6

# **High Level Architecture**

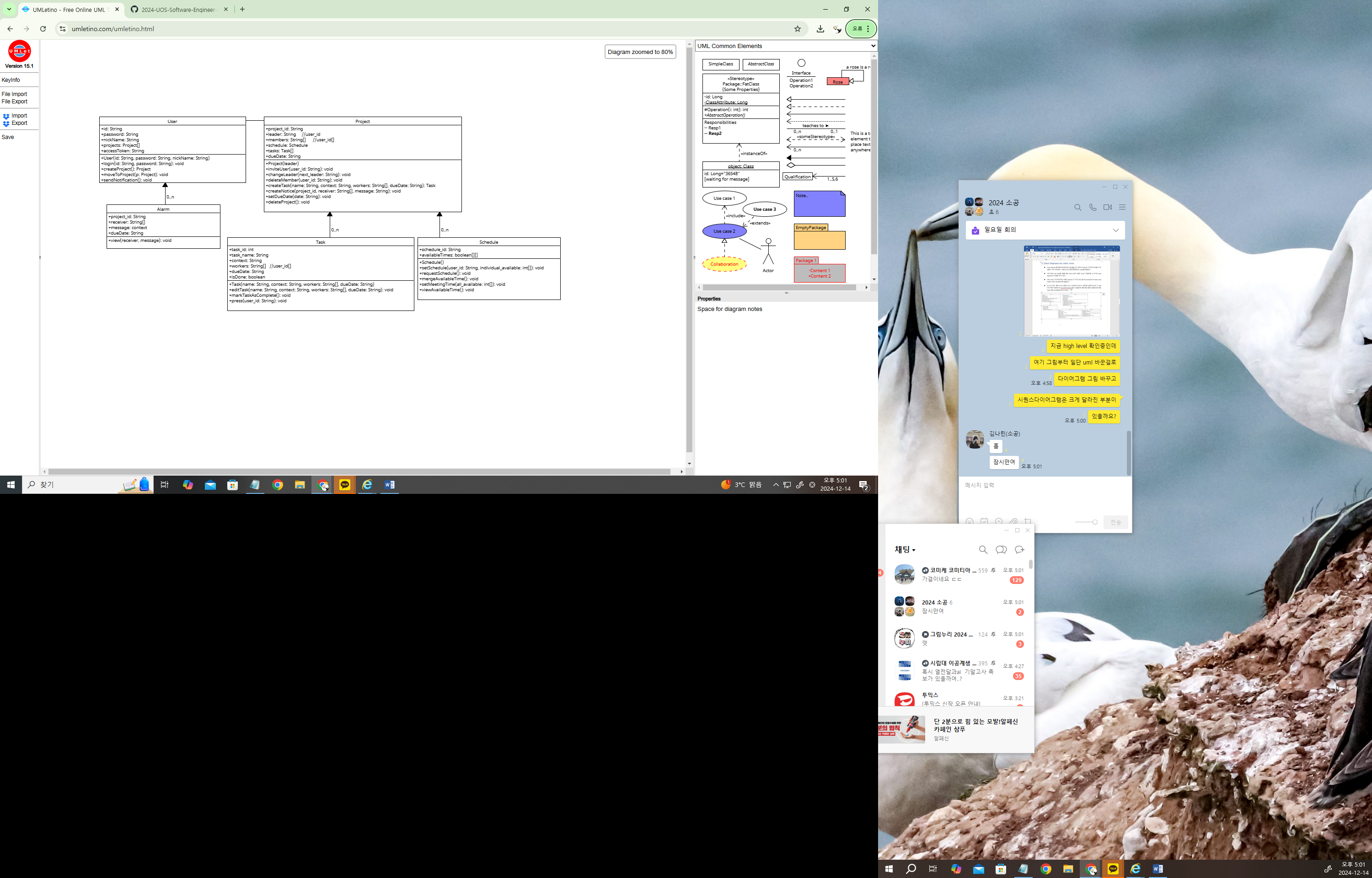
본 프로젝트는 High Level Architecture를 낮은 비용으로 쉽게 구현하기 위해 Single client-server 구조를 사용한다. Client(Front-end)는 React(Javascript)를 사용하고, Server(Back-end)는 node.js(javascript)를 사용한다. 추가로, Database(DB) 관리에는 SQL Developer를 사용한다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

# **Class Diagrams for Static View**

* Class Diagram을 만들 때 유의한 점은 ‘재사용성’이다. 특히 DB가 어떤 데이터를 가지고 있을지 가장 고려하였고, 사용하는 DB Table을 중점으로 Class를 만들었다.
* 다만 실제 Front-end를 개발할 때는 버튼 등의 수많은 Class가 만들어질 수 있기에 핵심이 되는 클래스만 작성하며, Class가 추가되더라도 아래의 Diagram이 가지고 있는 함수의 Parameter와 Return Type 자체는 수정이 최소화되도록 만들었다.
* Actor의 경우, 팀원 Actor와 팀장 Actor가 분리되어 있으나, 프로젝트 설계상 로그인 시 user 이미 객체가 만들어지고 프로젝트 접근 이후에 프로젝트 내에서 팀장 혹은 팀원이 결정되므로 팀장과 팀원을 따로 분리하지 않고, User Actor 하나로 통일했다.
* User가 가지고 있는 accessToken은 로그인 시 서버에서 받아오고, 해당 토큰이 없으면 서비스에 접근할 수 없도록 설계했다.



# **Sequence for Dynamic View**

많은 use case 중, 구현이 복잡한 SCHE01(스케쥴 시간대 입력), SCHE02(회의시간 설정)을 하나의 sequence diagram으로 표현하였고, 본 프로젝트의 핵심 기능인 CAL01(Task 목록 및 독촉 알림 전송)을 하나의 sequence diagram으로 표현하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 유스케이스 명 | 설명 | 우선 순위 |
| SCHE01 | 스케쥴  시간대 입력 | 팀원은 주간 스케줄에서 여유로운 시간대를 입력할 수 있다.  다른 팀원이 입력한 여유로운 시간대를 확인할 수 있다. | 1 |
| SCHE02 | 회의시간 설정 | 팀장은 여유로운 시간대를 바탕으로 회의시간을 설정할 수 있다.  설정한 회의시간은 캘린더에 기록된다. | 2 |
| CAL01 | Task 목록 및 독촉 알림 전송 | 팀원은 날짜를 선택하여 세부 Task를 확인, 특정 Task 독촉 알림을 전송할 수 있다. | 1 |

High-level architecture로 single client-server 구조를 고려하고 있기에, actor가 시스템, 데이터베이스와의 상호작용하는 과정과 use case의 실패대 안을 중심으로 diagram을 표현하였다.

