# Software Architecture Document

Version 1.0

for

Concordia Capstone Scheduler

Prepared by

| Name | StudentID | Email |
|---|---|---|
| Lance Lafontaine | 26349188 | lance.lafontaine92@gmail.com |
| Jason Tsalikis | 25892120 | jtsalikis@hotmail.ca |
| Lenz Petion | 26775837 | lenzpetion@gmail.com |
| Rameen Rastan-Vadiveloo | 27191863 | rameenrastanv@hotmail.com |
| Benny Zhao | 27205104 | bennyzhao@live.ca |
| Simeon Cvetkovic | 27430515 | bitugos@gmail.com |
| Lance Lafontaine | 26349188 | lance.lafontaine92@gmail.com |
| Sam Alexander Moosavi | 27185731 | sammoosavi94@gmail.com |

| | |
|---|---|
| Instructor | Professor Constantinos Constantinides |
| Course | SOEN 343<br>Software Architecture and Design I |

**Table of contents**

## List of figures

## List Of Tables

# 1. Introduction

## Purpose

The purpose of the Software Architecture Document (SAD) is to describe the system in terms of its high-level design (the system's architecture), as well as its low-level design (implementation level). Firstly, the document will provide an overview on the architecture of the system. It will proceed to present architectural representations of the system through different architectural views. It will then make reference to architecturally relevant functional and nonfunctional requirements, and will finally address size, performance, and quality attributes of the system. The document is intended for all team members as well as the clients of the product, as the different architectural views provide different abstractions of the system that is intended for different stakeholders.

## Scope

This document applies strictly to the software architecture of the system. As such, it details the high-level and low-level design of the system. This has a direct effect on the implementation of the system, as the implementation will be modeled by the system's architecture. Designers and implementers of the system will make reference to the architectural overview presented in this document when making design decisions during the development of this product.

**Definitions, acronyms, and abbreviations**

Table 1 . Glossary

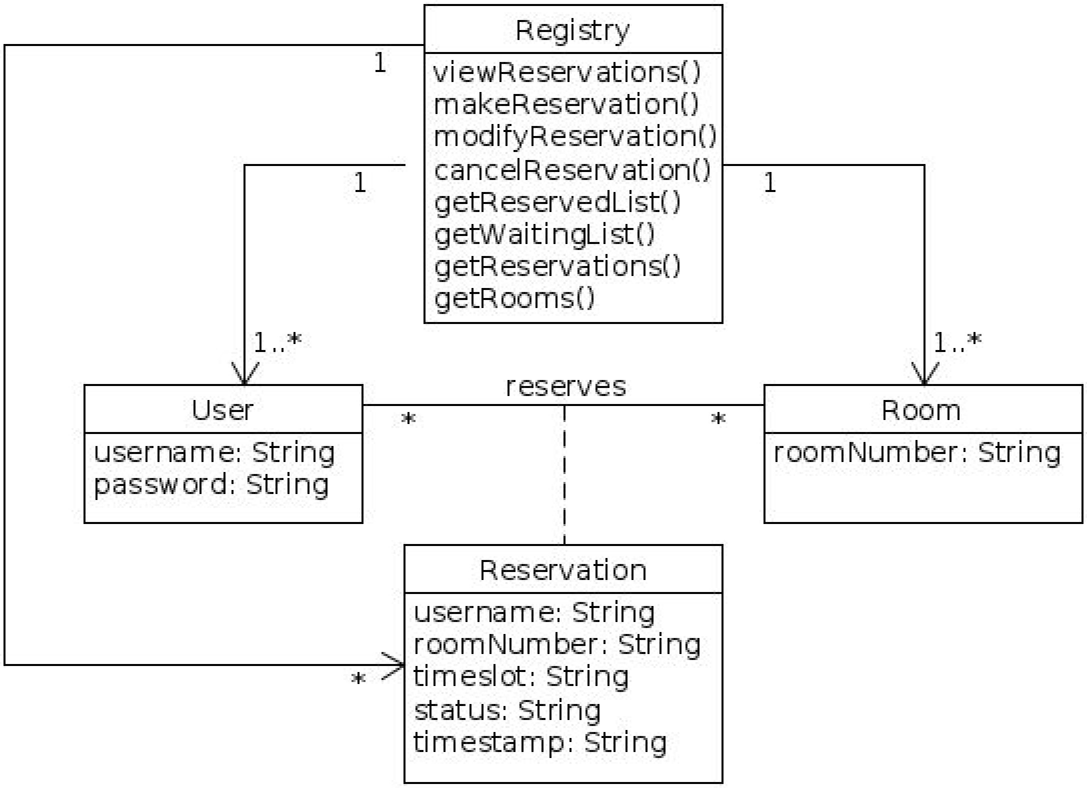
| Term | Definition |
|---|---|
| User | Someone who interacts with the reservation system. |
| Registry | A system for keeping an official record of room reservations. |
| Room | A space that students occupy for academic activities |
| Wait list | An ordered queue of students with a pending reservation at an unavailable time slot. |
| Reservation | Occupying a study room at a given time slot |
| Time Slot | A 1-hour time interval. |
| Weekly Calendar | A table displaying room information (time slots and availabilities) for a given week. |
| Stakeholder | An individual or group who is affected by the development of this project. |
| SRS | Software Requirements Specification |
| SAD | Software Architecture Documentation |
| Python | Python is a high-level, general-purpose programming language |
| SSD | System Sequence Diagram |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheet |

## 2. Architectural representation

### Logical view

Class Diagram

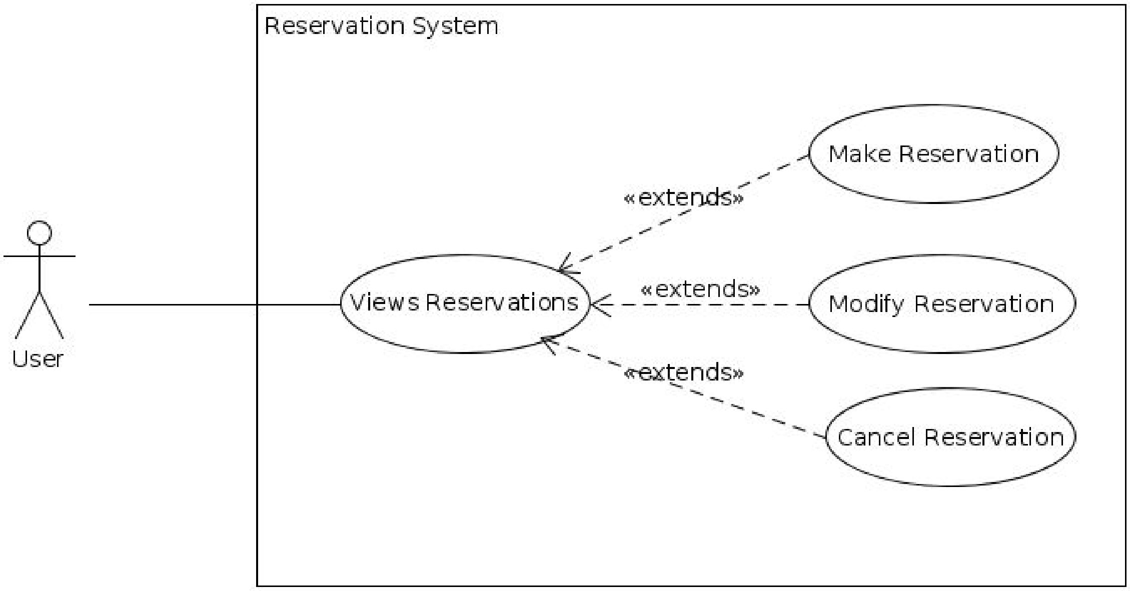
Figure 1. Class Diagram



1. **Use case view** :

Use Case Diagram

Figure 2. Use Case Diagram



## 2. Architectural requirements: goals and constraints

### Functional requirements (Use case view)

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

Table 1. Use Case View

| **Source** | **Name** | **Architectural relevance** | **Addressed in:** |
|---|---|---|---|
| UC1 | Make Reservation | Requires us to interactive with database and verify other reservations; this was the bulk of business logic complexity, designed around this use case. | Sequence Diagram 2 |
| UC2 | View Reservation | Needs to be updated often in the front end; forced us to implement some form of an observer | Sequence Diagram 1 |
| UC3<br>UC4 | Modify Reservation<br>Cancel Reservation | Given that we may be modifying and cancelling often we implemented a unit of work to reduce load on the database. | Sequence Diagram 3, Sequence Diagram 4 |

**Non-functional requirements**

**Accessibility**

Only ENCS registered students in the database should have access to the web application.

**Usability**

90% of users will be able to perform all major activities provided by the system in less than 3 minutes.

**Portability**

The web application should function on all major web browsers and operating systems.

Table 2. NFR Table

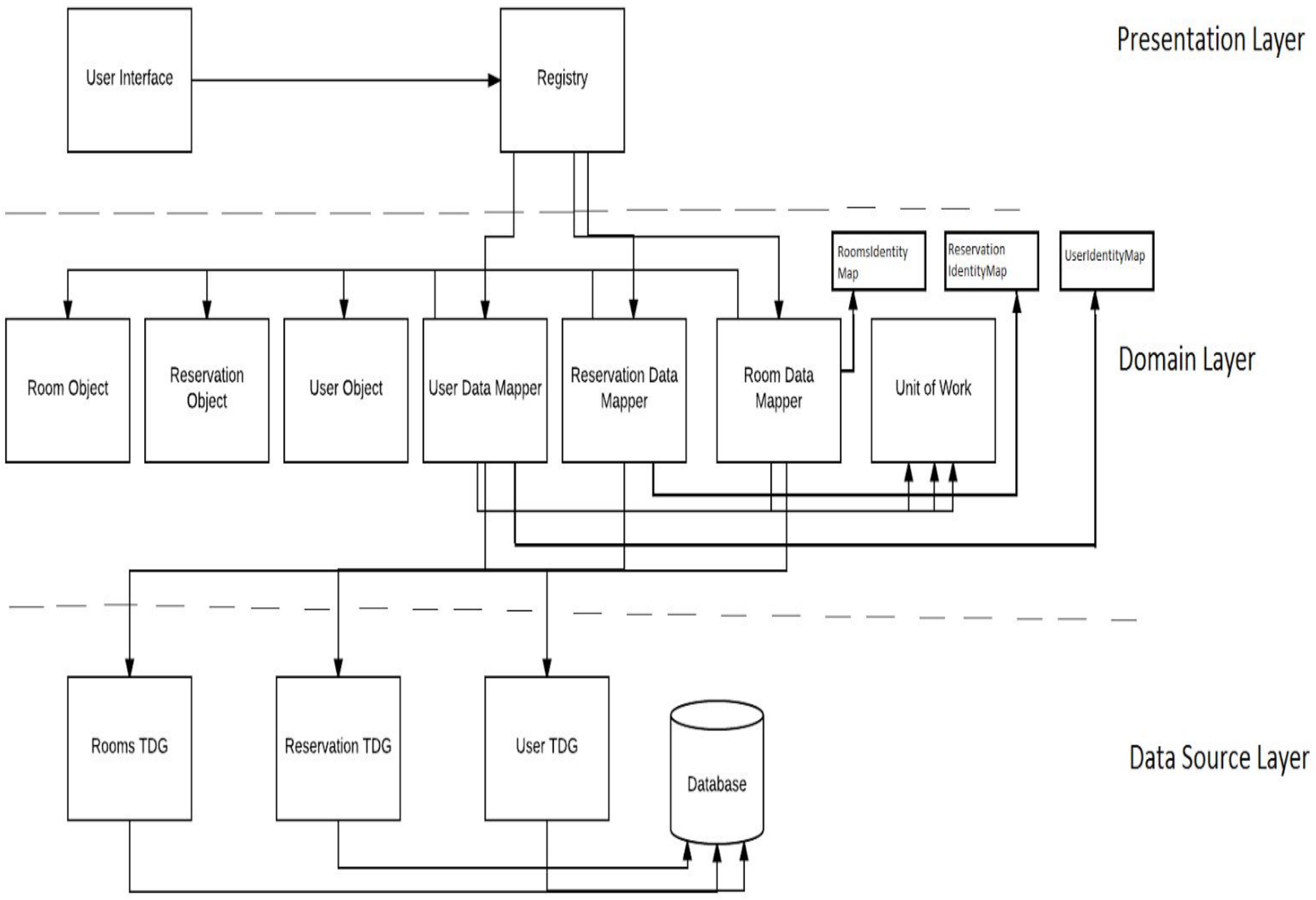
| Source | Name | Architectural relevance | Addressed in: |
|---|---|---|---|
| SRS | Accessibility | Login and logout functionality has to access database of registered students | Logical View |
| SRS | Usability | A Calendar UI that marks which timeslots were taken, this visual value makes the system easier to use | Logical View |
| SRS | Portability | Used standard libraries, web standards and popular frameworks to ensure compatibility on all major operating systems and web browsers. | Physical View |

## 3. Logical view

### Layers, tiers etc.

Layered Architecture Class Diagram

Figure 3. Layered Class Diagram



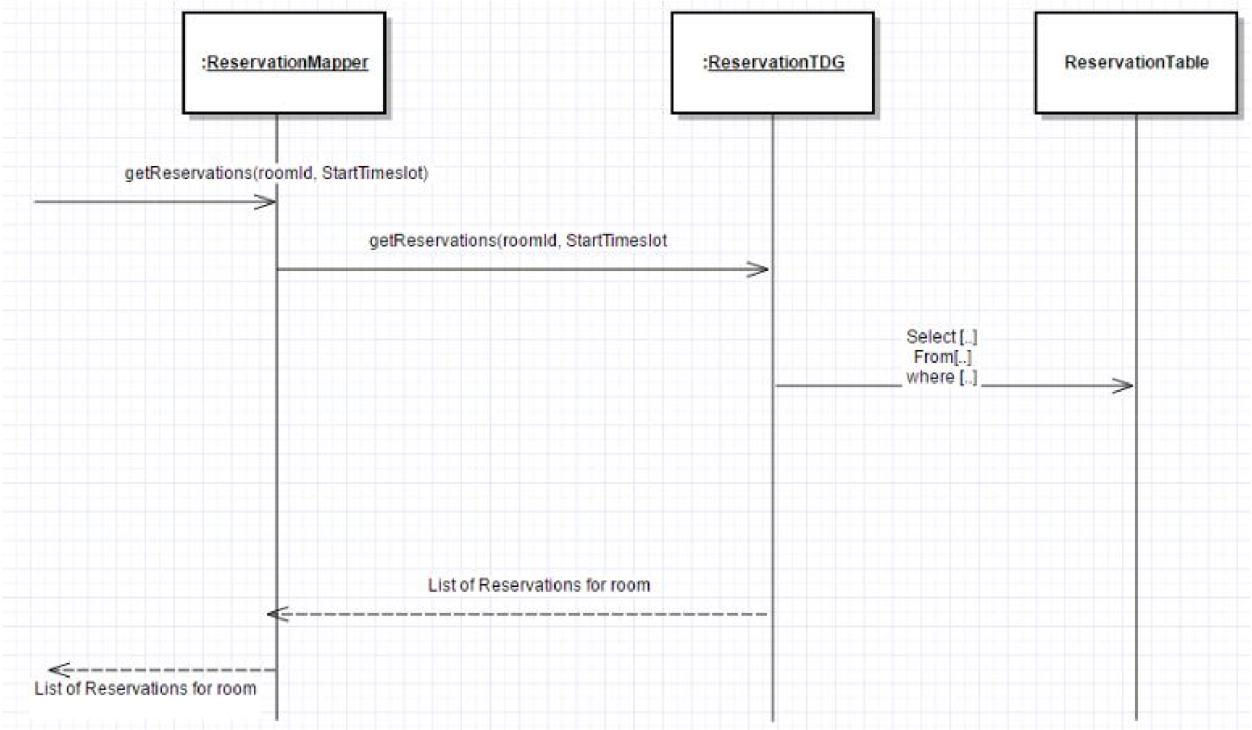
### Subsystems

The web application can be divided into two discrete subsystems: the client and the server. The client handles the presentation layer of the system, which is the only portion of the overall system that is visible to the user. This includes the Graphical User Interface, and also handles all user actions and requests. If the client receives a request from a user, it is then processed and sent to the next subsystem: the server, which contains the domain and data source layer of the system. The server handles the core business logic of the application and also contains the relational database, which contains tables to store domain object information in order to develop persistence in the overall system. These two subsystems interact with each other (the client sends a request to the server upon user input, the server then processes the action and sends a response), to form the overall system.

**Use case realizations**

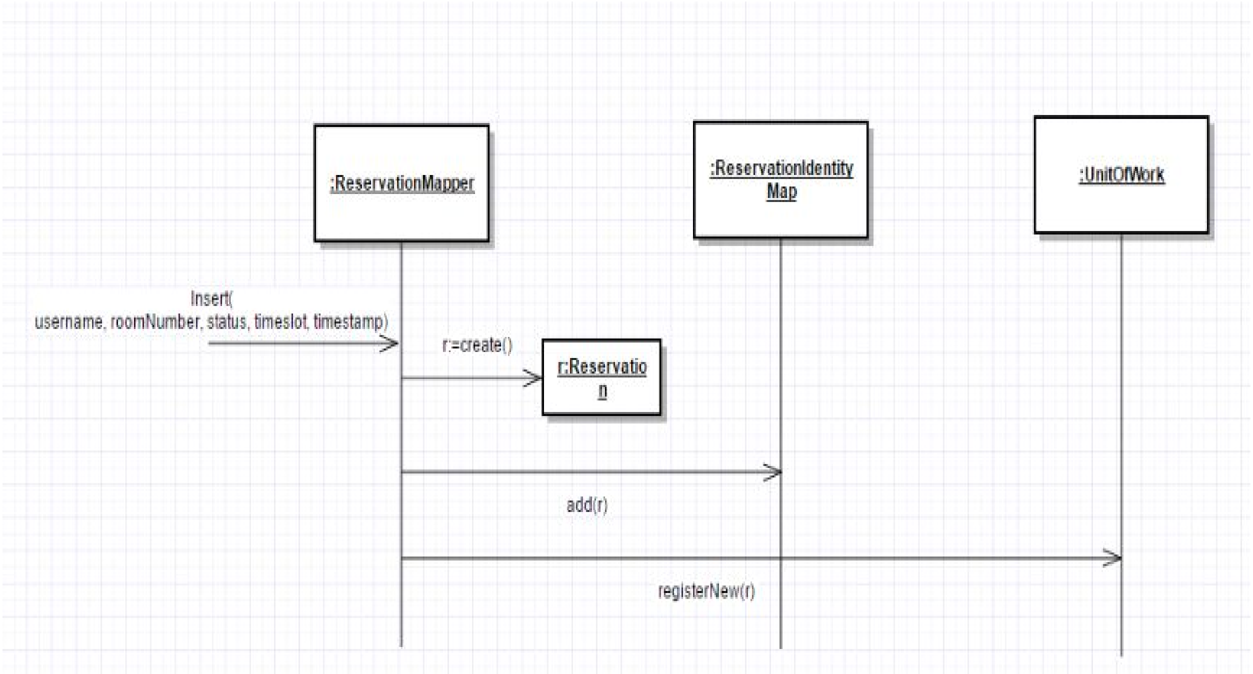
Figure 4. View Reservation Sequence Diagram

View Reservation Sequence Diagram


:ReservationMapper
:ReservationTDG
ReservationTable
getReservations(roomId, StartTimeslot)
getReservations(roomId, StartTimeslot
Select [..]
From[..]
where [..]
List of Reservations for room
List of Reservations for room

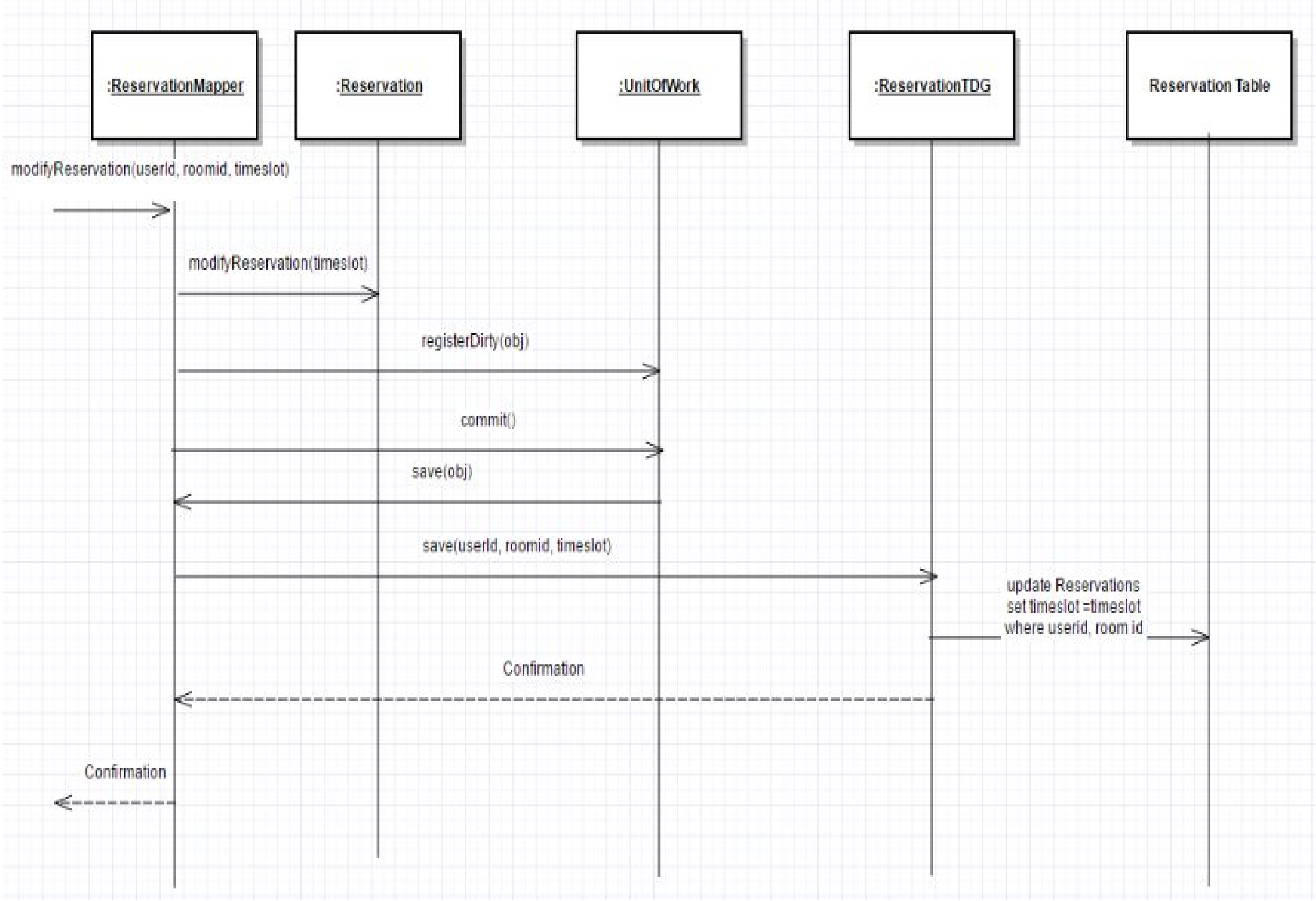
Make Reservation Sequence Diagram

Figure 5. Make Reservation Diagram



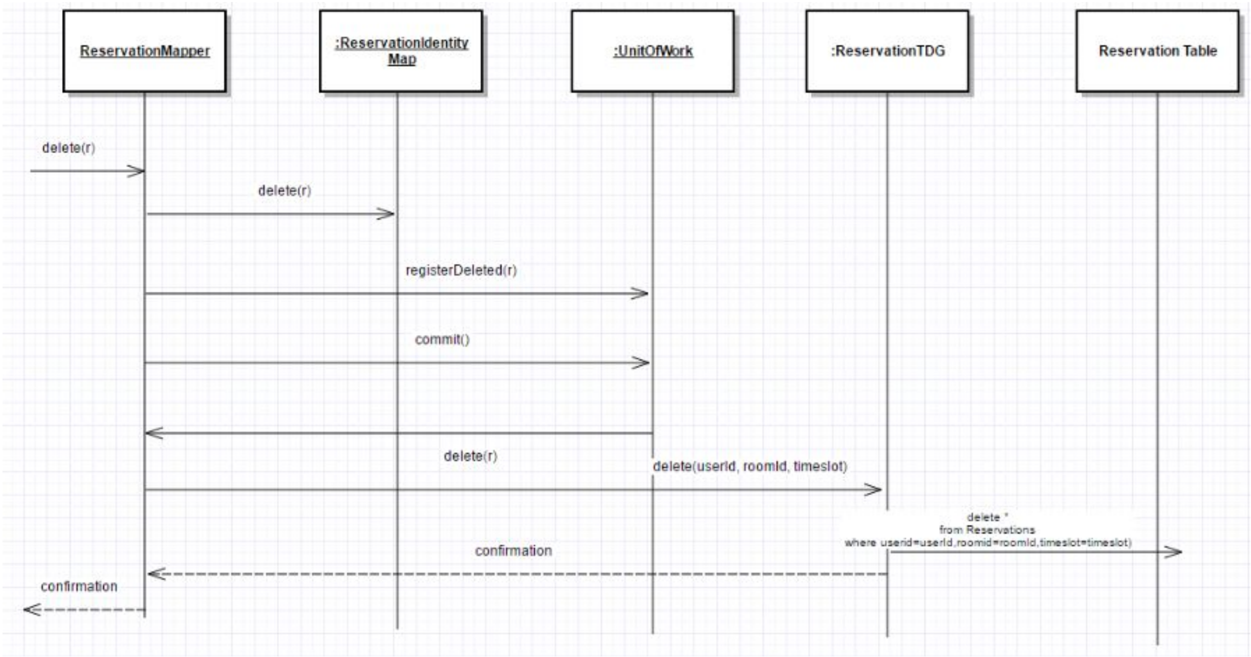
Modify Reservation Sequence Diagram

Figure 6. Modify Reservation Sequence Diagram



Cancel Reservation Sequence Diagram

Figure 7. Cancel Reservation Sequence Diagram



**Reuse of components and frameworks**

The web framework which was used in the implementation of this application was Django. Django is a python based web framework. Due to requirements in the project outline, we were restricted from incorporating certain functionalities of the framework, as certain limitations were imposed. In addition to this, Node.js, a run-time environment for JavaScript, was used to assist with front-end development.

## 4. Size and performance

Volumes:

- Estimated ENCS Capstone students : 400 students
- Number of Capstone rooms: 15
- Total number of Capstone room timeslots: 60

Performance:

- Time for a user to login: less than 5 seconds
- Time for the system to perform operations on the schedule (add, cancel, modify): less than 8 seconds