Software Requirements Specification

for

< Virtual Room Reservation >

Version 1.0 approved

Prepared by <author>

< NSYSU CSE >

< 2023.11.09 >

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Online Classroom Reservation system, aimed at digitizing and streamlining the process of reserving classrooms in educational institutions. It encompasses the complete requirements and is intended as a guideline for the development and implementation of the system, adhering to best practices in software engineering. This SRS covers the entire system, focusing on user interaction, data management, and system functionality.

1.2 Glossary

- User: Individuals who interact with the system, including students, faculty, and administrative staff.
- Classroom: Physical or virtual spaces available for reservation.
- Reservation: The action or process of booking a classroom.
- Schedule: The system component managing the availability and booking times of classrooms.
- Authentication: The process of verifying the identity of a user.

1.3 Intended Audience and Reading Suggestions

This document is intended for the professors overseeing the project and the development team.

Professors will gain insight into how the system will meet their operational needs, while developers will find detailed specifications for building the system.

Readers should start with the overview in Section 1, then proceed to Section 2 for a general description, and finally focus on the detailed requirements in Section 3.

1.4 Product Scope

The Virtual Room Reservation software is a solution designed to automate and optimize the process of booking classrooms. By transitioning to an online platform, the system aims to enhance efficiency, reduce manual workload, and improve resource utilization. This aligns with the institution's goal of modernizing its infrastructure and providing a seamless experience for staff and students.

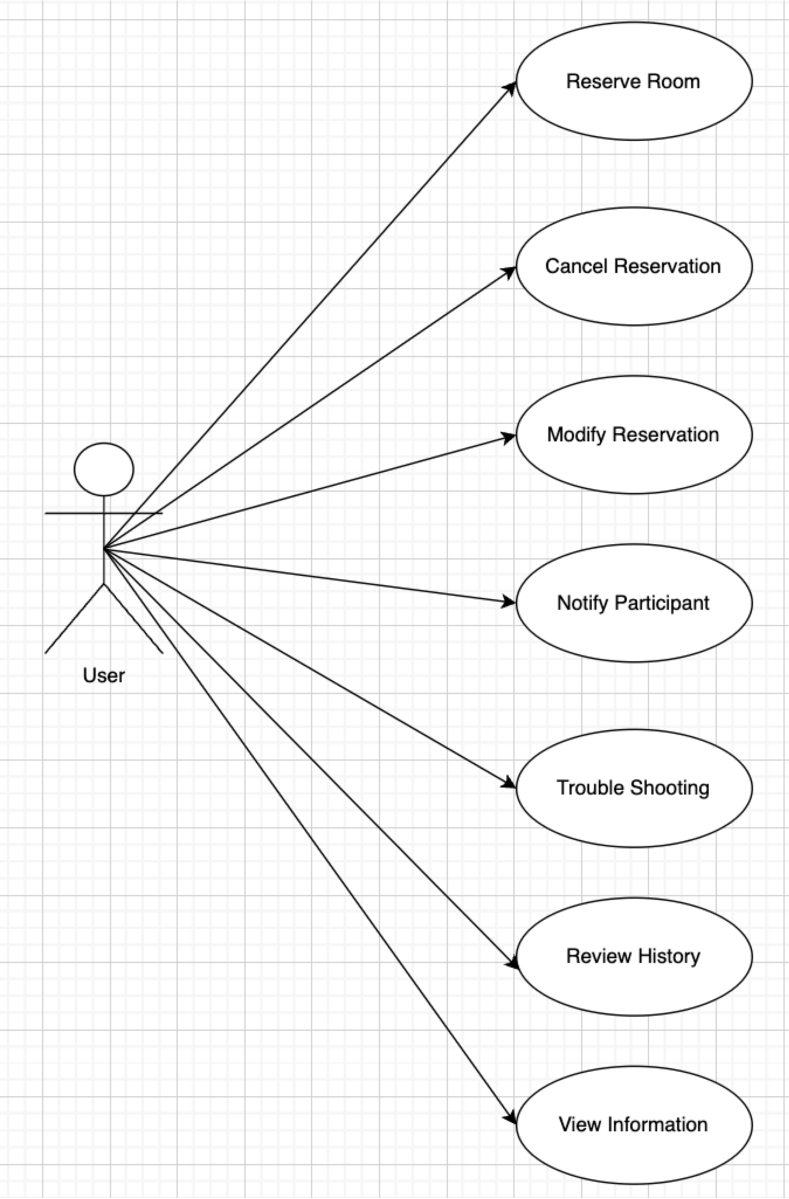
1.5 References

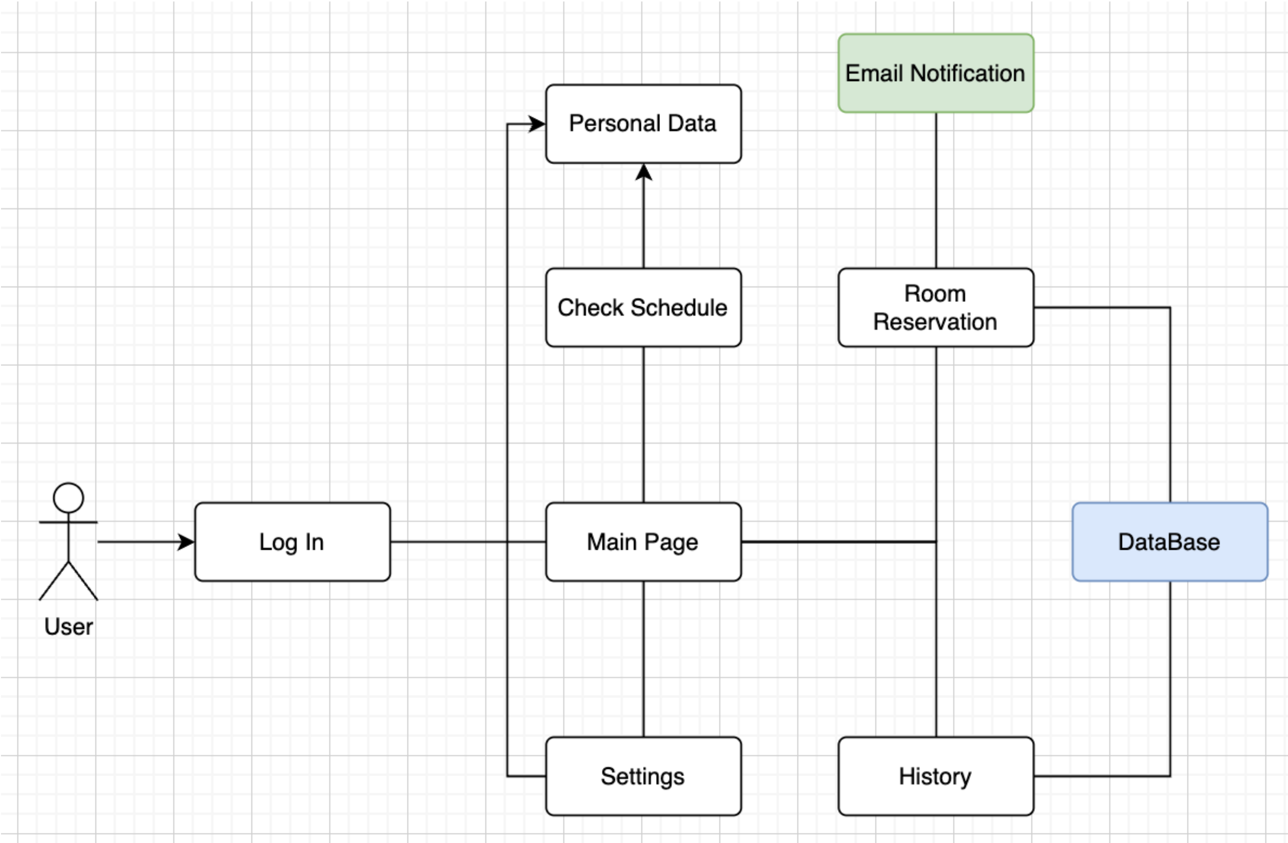
Software Engineering Textbook.

2. Overall Description

2.1 System Environment

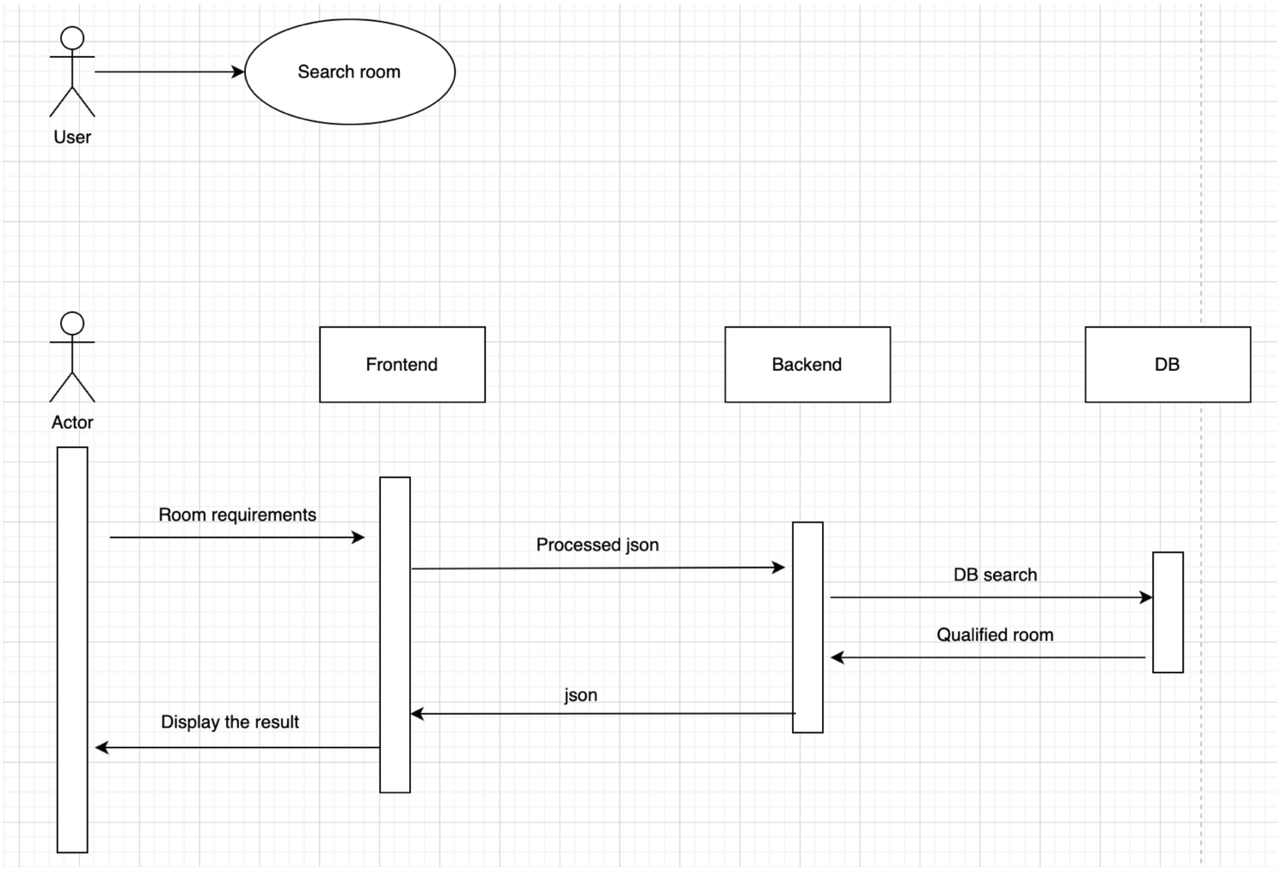
The system comprises a front-end interface for users, a back-end server handling data processing, and a database managing user and classroom information. It interfaces with external systems like institutional databases for user verification. UML diagrams illustrating the system environment and interactions will be provided.

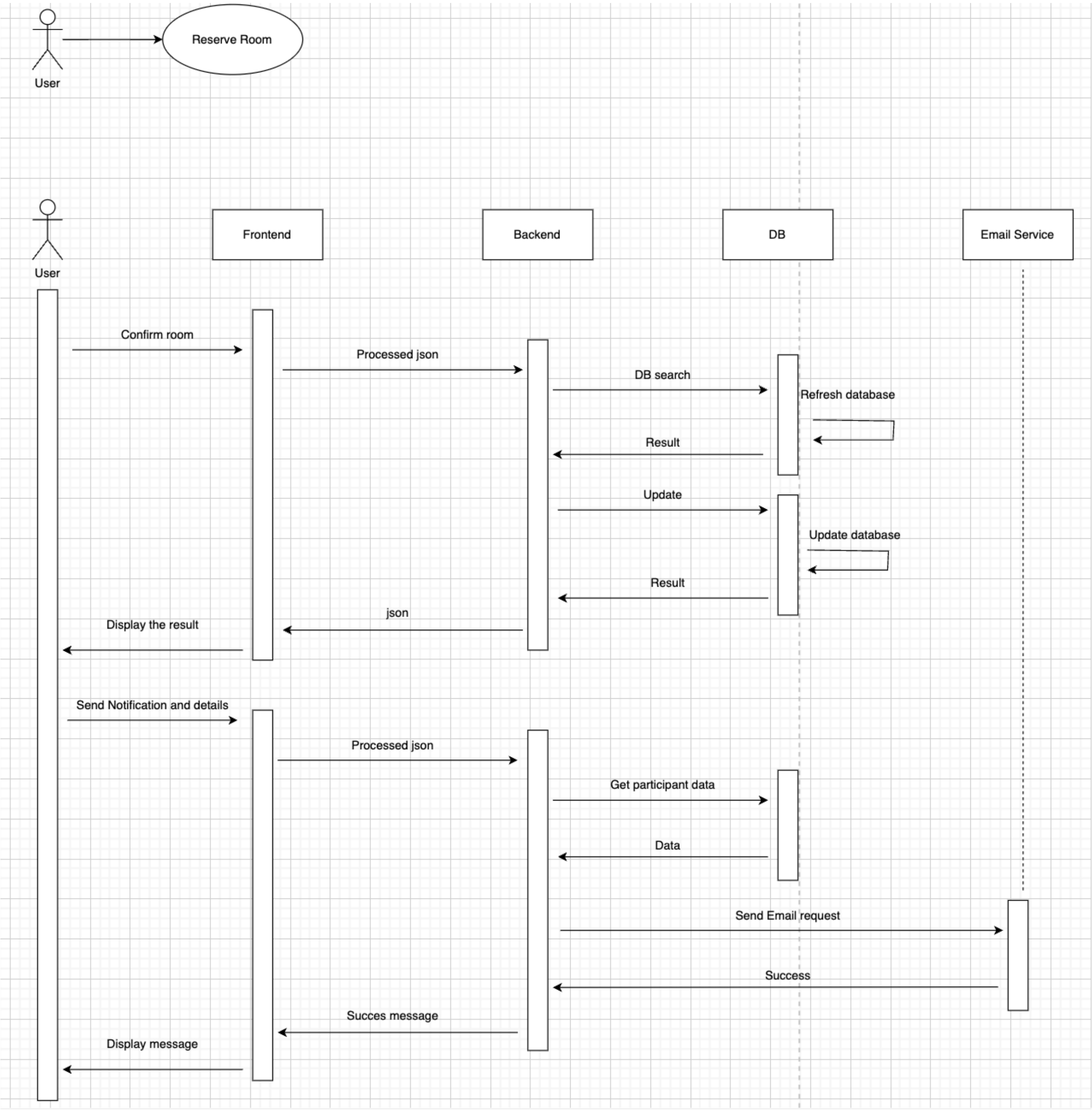


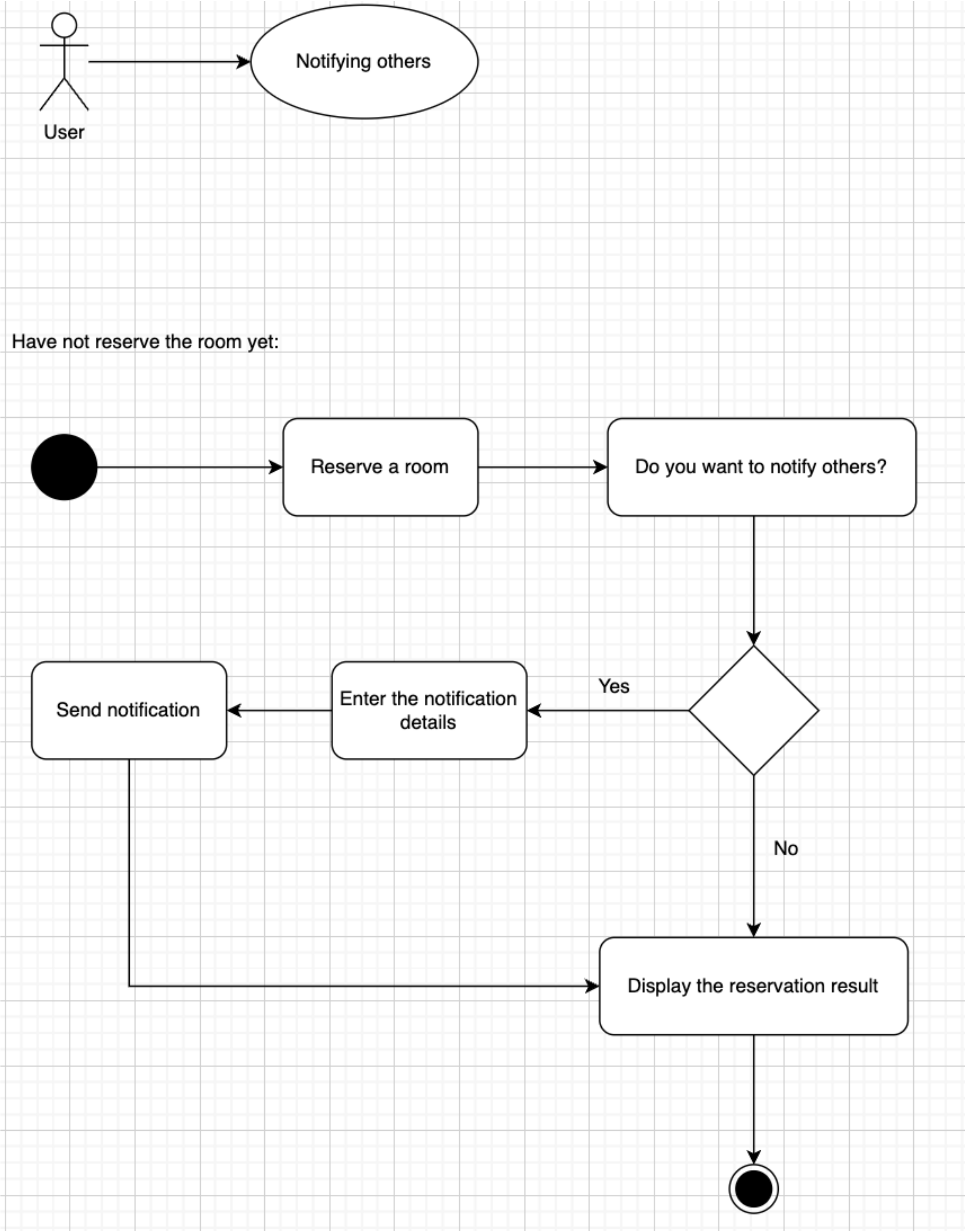


2.2 Functional Requirements Definition

The system provides functionalities such as user registration and login, classroom searching, reservation making, and schedule management. A detailed overview of these functionalities will be cross-referenced with use case descriptions in Section 3.


User
Search room
Actor
Frontend
Backend
DB
Room requirements
Processed json
DB search
Qualified room
json
Display the result


User
Reserve Room
User
Frontend
Backend
DB
Email Service
Confirm room
Processed json
DB search
Refresh database
Result
Update
Update database
Result
json
Display the result
Send Notification and details
Processed json
Get participant data
Data
Send Email request
Success
Succes message
Display message


User
Notifying others
Have not reserve the room yet:
Reserve a room
Do you want to notify others?
Yes
Enter the notification details
Send notification
No
Display the reservation result

Use case: <<Use Case Name>>

Diagram:

<< insert diagram or reference here>>

Brief Description

The use case <<name>> is initiated by the <<actor>> to <<explicit functionality>>. Possible relation to other use cases.

Initial Step-By-Step Description
Before this use case can be initiated, the <<actor>> has already <<preconditions>>.

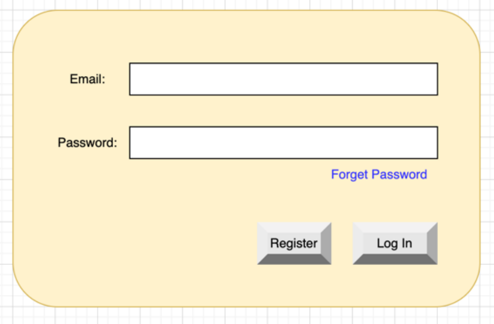
1. The <actor>> <<initiates>>.
2. <<next action>>.
3. <<next action>>.etc.

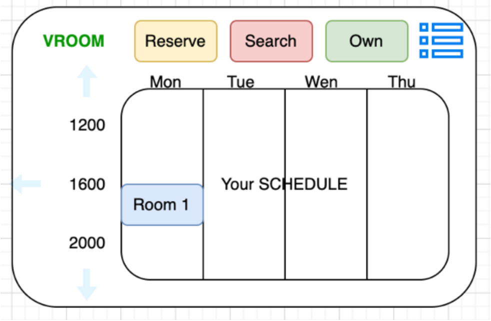
Use *active* not *passive* voice. Remember that this is a *black box* approach. If two consecutive steps above are both system operations, you may have started doing *design* instead of specification. Normally, the user and the system alternate steps. Describe the most common path only in the step-by-step.

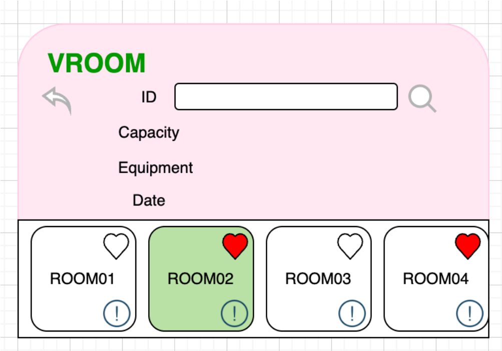
This is referred to by Sommerville as the *requirement definition* and must be readable by the user with more detail available in the *requirement specification* in the next section. As a practical matter, many of the brief descriptions can be filled in before starting to fill in the initial step-by-step descriptions.>

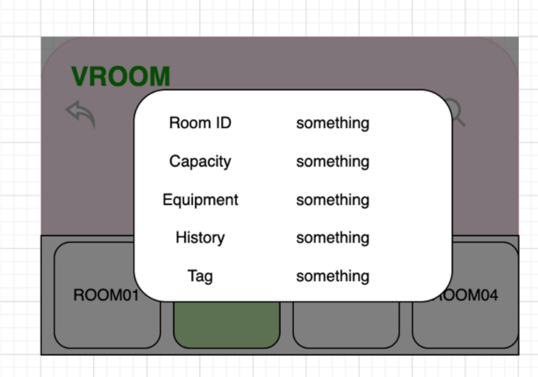
2.3 User Interface Specifications

The interface is designed for users with varying levels of technical expertise. It will feature intuitive navigation, simple reservation processes, and clear error messages. Prototyping will be used to refine the interface, ensuring consistency and user-friendliness.









2.4 Non-Functional Requirements

- Data Security and Privacy: The system must be secure. The password entered must be asterisk, stored in Database in an encrypted manner. The password must be at least 8 characters, including English and numbers, and uppercase and lowercase characters.

- User-Friendly Registration: To register an account, you only need to fill in the minimum (name, email, password), and the other parts are optional.

- Real-Time Availability: Database needs to be updated within 0.5 seconds.

3. Requirements Specifications

3.1 External Interface Requirements

The system will interface with user devices, institutional databases, and potentially other scheduling systems. All interfaces will be detailed in this section.

3.2 Functional Requirements

A detailed specification of each system functionality, including use case name, priority, trigger, precondition, basic path, alternative paths, postconditions, exception paths, and additional notes. UML diagrams will supplement these descriptions.

A possible grid for a requirement specification is as follows:

Use Case Name	
Priority	(this field is optional) *essential*, *desired* or *optional*
Trigger	
Precondition	
Basic Path	
Alternative Paths	(If none, so state.)
Postcondition	
Exception Paths	(If none, so state.)
Other	(this field is optional)

4. Other Nonfunctional Requirements

4.1 Performance Requirements

Performance requirements will be specified to guide design choices, including response times and system capacity.

4.2 Safety Requirements

Safety requirements will be outlined, including data integrity and system operation safeguards.

4.3 Security Requirements

Security measures for user authentication and data privacy.