University Room Management System

Sean Nagel, Akshay Chikhalkar, Mohamed Bassam Kassabji

Department of Electrical Engineering and Computer Science
Technische Hochschule Ostwestfalen-Lippe, University of Applied Sciences and Arts
Lemgo, Germany
akshay.chikhalkar@stud.th-owl.de, sean.nagel@stud.th-owl.de, mohamed.kassabji@stud.th-owl.de

Abstract

Contents

I. Introduction

II. System description

As already mentioned in the introduction, this document focuses on the proposed University Room Management System (URMS). As can b guessed from the name, the system should provide multiple features regarding management of a university's rooms. Users of the system shall be students and employees of the university. Additionally, guests can also use the system, although they will not have access to the system's complete functionality.

The basic functionality of the system should allow all users to search and find rooms. The system shall also be able to recommend rooms based of some user defined criteria. The users are able to look at room schedules. The system should also provide users with navigational instructions to rooms.

Students and university staff should be able to book rooms for lectures or learning groups. Employees of the university should be granted more privileges regarding room booking than students.

The university's rooms are fitted with IOT technologies. Electronic locks are built into the rooms' doors. Members of the university are able to unlock these doors, if they booked the room. The rooms are also fitted with movement sensors, to analyze if the room is currently occupied. Additionally, rooms might have a display, which shows the room's schedule and availability.

To further describe the functionality of the URMS a specific business process should be examined. This business process describes a student booking and using a room. The business process according to Business Process Model and Notation (BPMN) is shown in Fig. 1-2

III. Use cases and user stories

To further identify the desired functionality of the University Room Management System, two use cases and three user stories were identified and formulated. One use case and two user stories will be described in more detail in this section.

The identified use cases are *Searching and booking a room* and *Getting navigation instructions to a room*. For the use case *Searching and booking a room* a use case specification was created. The specification is shown in TABLE I. Some aspects of this use case specification will be further elaborated on.

Extension 1a describes searching for a room via room properties. Possible properties for searching and filtering rooms could be: type of the room (lecture room or seminar room), room capacity or the availability of projectors and blackboards.

In extension 5b the maximum allowed number of rooms is noted. This limit can be different depending on the user. Our current specification limits students to be allowed of booking only a single room at a given time. University staff is allowed to book multiple rooms at a time (e.g. a Professor booking multiple rooms for an exam). The specific limits for each group of staff should be selected according to the universities specific needs.

Extension 5c notes the possibility to book a room for future use according to a time schedule. This option shall not be available to students but only to university staff. For example a room can be booked for a specific time frame on a weekly basis. This stands in contrast to the general booking presented in the main success scenario. In that case the room is booked from the time of booking without specifying the length of the booking. The booking has to be terminated manually in the end. Alternatively the booking is also terminated, if the system registers that the room has not been used for a while or a previously scheduled booking is about to start. Please see the BPMN diagram above for this process.

The open issues of the use case note the booking of multiple rooms at once. The process for this will likely be very similar to booking a single room. Showing information for multiple rooms at once and managing the unavailability of a single room ot of the selection need to be further deliberated. Similarly to scheduled booking students should not be allowed to book multiple rooms.

Following three user stories were formulated:

1) As a Student, I want to book a room, so that I have a place to work.
2) As a Guest, I want to have navigational instructions, so that I am able to locate a room.
3) As a Professor, I want to book a room according to a schedule, so that I can give regular lectures in that room.

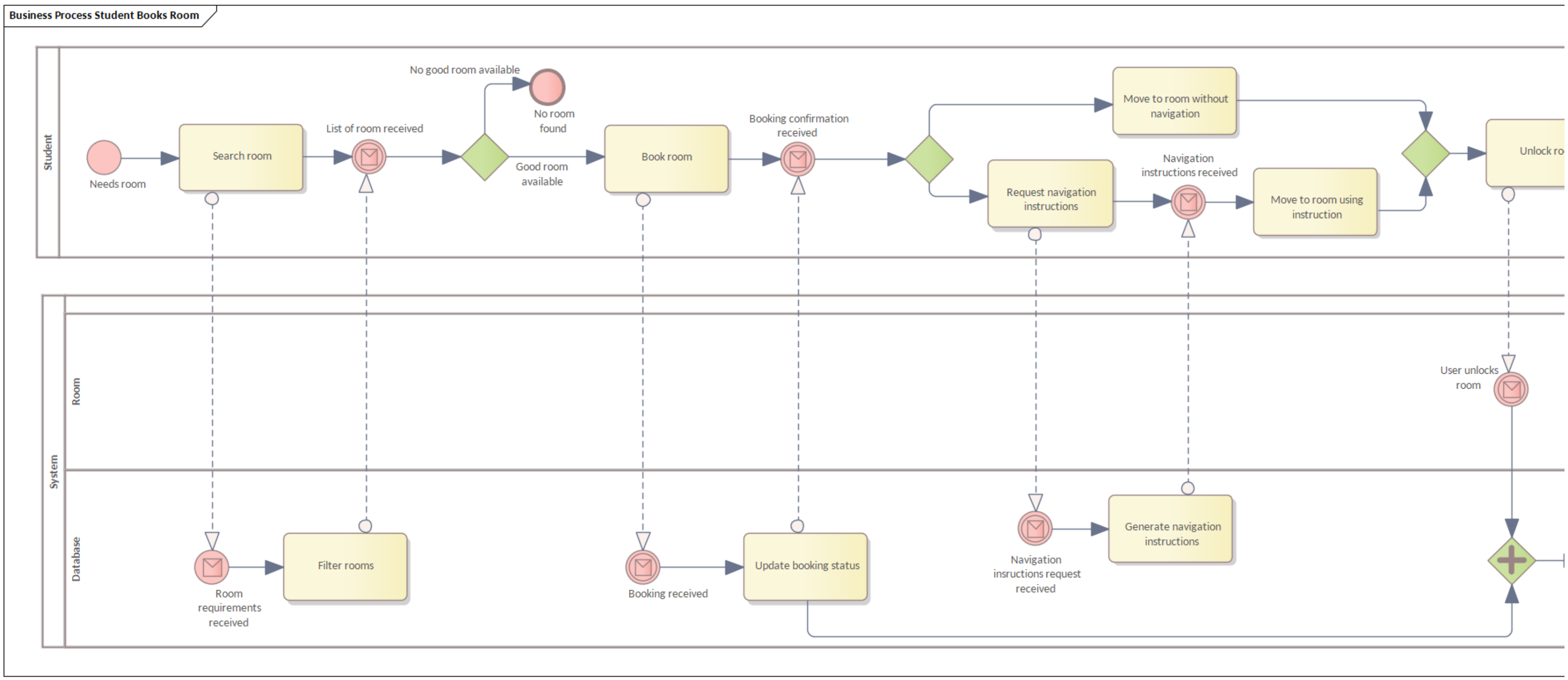


Fig. 1: Business Process “Student Books Room” (Part 1)

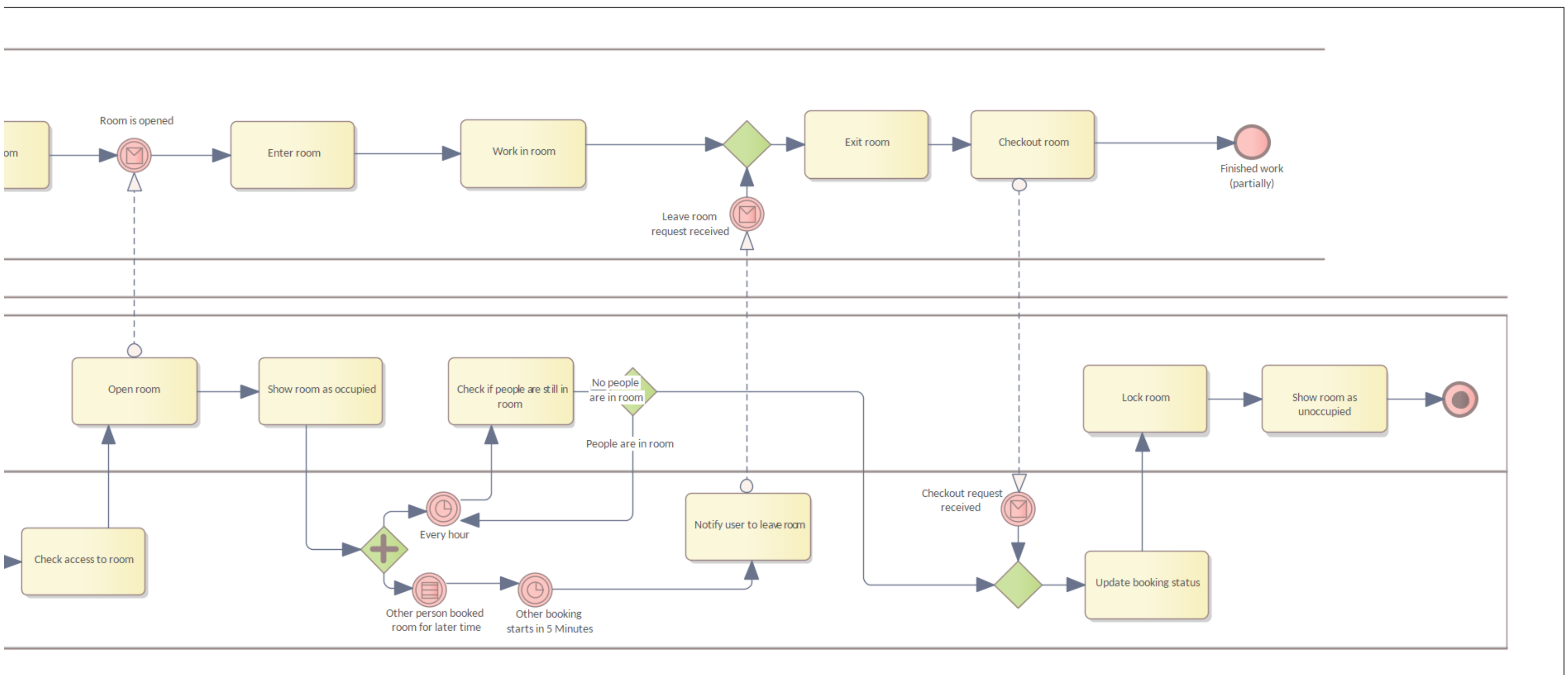


Fig. 2: Business Process “Student Books Room” (Part 2)

Name	Searching and booking a room
Scope	University Room Management System
Level	User goal
Primary Actor	Student, Professor or other university staff
Stakeholders / interests	Other members of the university (students, professors, other staff) University / university management
Preconditions	User is in possession of a valid university card and has a user account
Postconditions	Status of the rooms in the room database is updated
Main success scenario	1. User searches a room via a room number 2. System returns a list of rooms fulfilling the search query 3. User selects a room from the list 4. System shows detailed information about the selected room 5. User books the room 6. System updates booking status of the room 7. System shows booking information and confirmation
Extensions	1a. User searches room via room properties 1. User opens page to filter specific properties 2. User selects room requirements and confirms 2a. No room fulfilling the search criteria exists 1. System shows message: “No rooms found” 2. System presents rooms similar to the search query 5a. Room is already booked 1. System shows message: “Room is already booked” 2. System suggests a list of similar rooms 5b. User already booked the allowed maximum amount of rooms 1. System shows message: “You cannot book another room” 2. System shows list of rooms already booked by the user 5c. User books room according to a schedule (for future use) 1. System presents a day and time picker 2. User selects day of week and time 3. System asks for the frequency of the booking 4. User selects how often booking should be repeated 5. System books room for all selected time slots 5a. Room is not available in all time slots 1. System shows message: “Room is not available at <DAY / TIME>” 2. System provides alternative time slots and rooms 3. User selects alternatives
Special requirements	Stable database Simple UI design Fast search
Technology / data variation	Available via a terminal at the university or via a web app
Frequency of occurrence	Very regularly, up to multiple requests per minute at most active times Long pauses without interaction also likely
Open issues	Booking of multiple rooms at once

TABLE I: Use case specification for the use case *Searching and booking a room*

A good practice is to formulate user stories according to the INVEST principle. This principle describes that a user story should be independent, negotiable, valuable, estimable, small and testable. User stories 1 and 2 were elaborated further and analyzed with regard to the INVEST principle.

The first user story “As a Student, I want to book a room, so that I have a place to work.” is extended by following points:

- A student is only allowed to book at most one room at any given time.
- Students can only book rooms for the current time and are not able to reserve them for later.
- A student is identified by their university id.
- Bookings by students do not have an ending time specified.
- Students are able to terminate their bookings manually.
- Bookings by students are terminated automatically, if a booking by a university employee starts.

The user story fulfills the INVEST principle, as it does not depend on other user stories, leaves room for negotiation and presents a value to the students. A development team should be able to estimate the work required for realization. The story itself is small, as it only focuses on bookings for a single user group. The story can be tested, by students trying to book rooms.

The second user story “As a Guest, I want to have navigational instructions, so that I am able to locate a room.” is extended by following points:

- The user story is also valid for students and university employees.
- Navigation is realized trough building schematics, as well as indoor and outdoor maps.
- Other points of interest (e.g. parking spots, water fountains) are out of scope and might be part of standalone user stories.
- Users do not need to be verified for navigation.

This user story is independent of other user stories and still leaves room for negotiation. The value of better navigation at the university campus is provided. The use case is small and the required amount of work can be estimated. Tests might include the navigation between rooms or buildings of the university by external people. Thus, this use case also fulfills the INVEST principle.

IV. Using machine learning models as specification instruments

One use case and one user story from the previous section were intentionally not further specified. Instead, they were used to analyze the ability of language models like ChatGPT[1] to generate parts of a system specification. In particular, we used the aforementioned ChatGPT by OpenAI to generate a use case specification and elaborate on a user story according to the INVEST principle.

The conversation with ChatGPT can be found in appendix A.

The initial prompt included a short description of the system and asked to write a use case specification for the specific use case. ChatGPT returned a specification including a name, ID, primary actor, preconditions, postconditions, trigger, main flow, alternate flow, exceptional flow and notes. To simplify comparisons ChatGPT was asked to provide the specification with the same aspects as those of the specification in TABLE I.

On first sight, the generated response seems to be good. Some points even include additional explanations. But there are other interesting things to note:

The use case's level was identified as being a user goal and a sub-function. The level should be either one or the other, but not both. by investigating the specification it can be seen, that it should probably be a user goal instead of a sub-function.

The three stakeholders students, professors and guests are identical to the primary actors previously noted in the specification. The only stakeholder, which is not a primary actor is the university management. In this case the primary actors should not be listed in the stakeholders section.

The generated preconditions are relatively simple and are not necessarily needed for the use case specification. Additionally, the third precondition, that the user needs to be logged in, directly contradicts the main success scenario, in which the user is logging in during step 2. Either the precondition or the following scenario is wrong.

The main success scenario starts with the user opening the system application. This is different to our approach, where the scenario started with the application being ready for user interaction. The approach chosen by ChatGPT is not wrong, but just different. Step 7 and 8 describe how the scenario continues depending on whether a room was found or not. The main success scenario should only contain the successful case of the room being found. The other case should be included as an alternative flow. Interestingly enough, the single alternative flow of the specification handles this case. Thus, step 8 of the main flow is unnecessary.

For the point frequency of occurrence the generated specification is defined as "frequently throughout the academic year". This is a very non-descriptive specification, as it is unclear if the use cases will occur multiple times per hour, per day or per week.

After generating the discussed use case specification ChatGPT was prompted to write a user story using the INVEST principle for user story 3: "As a Professor, I want to book a room according to a schedule, so that I can give regular lectures in that room."

The answer starts by listing the six facets of INVEST. It is declared that the given user story fulfills all aspects of the INVEST principle. Additionally, acceptance criteria for the use case are defined by the language model. The acceptance criteria seem sensible to the use case. Although the first five acceptance criteria are structured very similar to a use cases main success scenario. The answer also includes some additional considerations, which also seem sensible in the context of the system.

[1] https://openai.com/chatgpt

While reading ChatGPT's answer the question arose, how the model would handle a user story, which does not fulfill all aspects of INVEST. Thus, the model was prompted to write another user story according to the INVEST principle. The user story requested in this prompt is: "As a University member, I want to use the Room Management System, because it will make life easier." This user story is very broad and unspecific, such that it is neither estimable nor small. The user story also is not independent, as it focuses on the entire room management system at once.

The answer of ChatGPT is disappointing, as it is argued that the given user story fulfills all aspects of INVEST. Especially statements like "The user story has a focused scope", or "It focuses on the specific need [...] to utilize the Room Management System" show that the problems of the user story are not detected by ChatGPT. The task to write the use case done according to the INVEST principle is done, but there is no plausibility check performed.

Although the answers by ChatGPT are not completely correct, the amount of sensible output by the language model are quite impressive, especially if you consider the small amount of information given in the prompts. The answers of ChatGPT need to be checked for plausibility and cannot be used as is. But the generated use case specifications and user story elaborations can be used as a good starting point, which requires some manual fine-tuning by human designers. Using language models for initial specification drafts can probably decrease the required time for a system specification.

It should also be noted that the used ChatGPT is a general purpose language model. More specified models might yield even better results. Currently, there are developments on models with the specific task to generate such specification and even write some initial code. One example that should be noted in this context is "MetaGPT"[2], which according to its development page is able to generate user stories, requirements, APIs and more.

V. Domain model

A domain model is a conceptual framework utilized in software engineering to encapsulate the fundamental elements comprising a particular problem domain [1]. It involves creating a visual or diagrammatic representation that outlines the essential entities, their attributes, interconnections, and governing principles within the domain. This model is instrumental in fostering a comprehensive understanding among diverse stakeholders, including developers, designers, and domain experts [2]. It involves defining entities like"Customer", "Product" or "Order", detailing their characteristics and the relationships that bind them. By delineating associations, constraints, and business rules, a domain model ensures that the software system mirrors the intricacies of the real-world context it aims to address. Typically illustrated through UML class diagrams, domain models facilitate effective communication and serve as a foundation for crafting software solutions aligned with the domain's intricacies and requirements [3].

A. Conceptual classes

The conceptual classes for the University Room Management System (URMS) are shown in TABLE II below.

The conceptual classes help to clarify how different elements interact within the system. These conceptual classes provide a way to categorize the key aspects of the system. The system's "business transactions" involve actions like making room reservations or bookings. Each transaction is further detailed through "transaction line items," which specify aspects such as the timeframe for a room reservation. The "product/service" category encompasses items like rooms and navigation instructions, while "actors" refers to the different individuals who interact with the system, such as students, professors, guests, and university staff.

"Place" pertains to relevant locations, including the university campus and the user interface of the system itself. Events like bookings are tracked under "events," while "physical objects" encompass tangible items like rooms, buildings, and sensors.

"Container" refers to broader categories like buildings, rooms, and the campus, while "things in container" delves into specific instances within those categories, such as individual rooms and door locks. The concept of "neighboring systems" highlights potential interactions with other external systems.

"Records" represent the detailed information about bookings made within the system. "Schedules/manuals" involve information related to room schedules or guidelines. These conceptual classes help paint a comprehensive picture of how the University Room Management System operates and how its various components fit together.

[2]https://github.com/geekan/MetaGPT

Business transactions	Booking (includes reservations implicitly)
Transaction line items	Booking time frame
Product/service	Room Navigation Instructions
Transaction record	-
Actors	Students Guests Professors Other university employees
Place	University campus Software Frontend (System user interface)
Events	Bookings
Physical objects	Rooms Buildings Room display Door lock Movement sensor
Descriptions	-
Catalouge	-
Container	Room Building Campus
Things in container	Building Room Room display Door lock Movement sensor
Neighboring systems	-
Records	Booking
Financial instruments	-
Schedules/manuals	Room schedule

TABLE II: Conceptual classes for the University Room Management System

B. Class diagram

The class diagram illustrates the structure and relationship among classes in the system. The university room management system includes a total of 13 classes which are shown in Fig. 3 below.

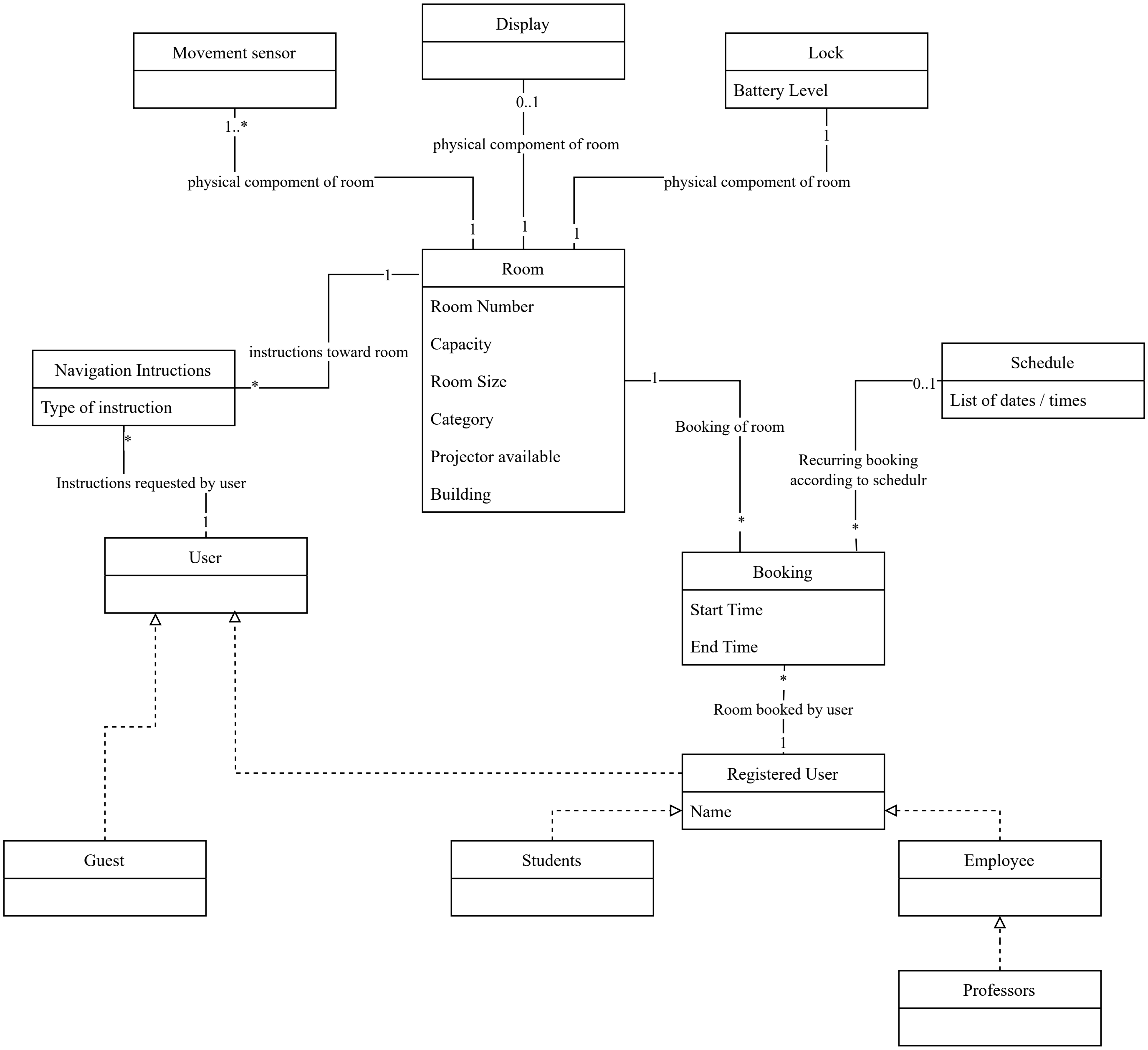


Fig. 3: Domain model of the University Room Management System

Classes: The classes are an amalgamation of physical objects and software components. The classes are described below: Movement sensor: The movement sensor is a physical object that is used to detect the presence of a person in the room. It is connected to the room display and the door lock through the system (Room). At least one movement sensor is present in each room and connected to the class "Room" in a many-to-one association.

Display: The display is a physical object that displays the room schedule and other booking information. In this system we consider one or more than one display per room and connected to the class "Room" in one-to-many association.

Lock: The lock is again a physical object that is used to lock and unlock the room. In this system, we consider one lock per room and connected to the class "Room" in one-to-one association. The class "Lock" has one attribute "Battery Level" which is used to check the status of the Battery and helps to keep track of maintenance cycles.

Navigation Instructions: The navigation instructions are a service which is used to provide directions or visual insight to the user. Navigation instructions can be invoked by the system itself or by the user. This can be achieved by requesting navigation instructions at the time of booking or by accessing the navigation service and searching for the room number. The class “Navigation Instructions” has one attribute “Type of instruction” which defines whether instructions are requested in the building schematics or maps form. It is associated with the class “Room” and “User” in a many-to-one association.

Schedule: The class “Schedule” holds the schedule metadata about the lectures according to the scheduler. This schedule can be referred to by the user at the time of booking to create recurring room reservations. The attribute “List of dates/times” returns all the days and times of the week when the lecture is scheduled. The class “Schedule” is associated with the class “Booking” in one-to-many (0...1 to *) association.

Booking: The class “Booking” holds the booking metadata about the room reservations. The attributes “Start Time” and “End Time” returns the start and end time of the booking. It invokes class “Schedule” to get a lecture schedule if required. The class “Booking” is associated with the class “Room”, “Schedule” and “Registered User” in many-to-one association.

Registered User: The class “Registered User” is a master class for university students, professors and other university staff. It means these users are authenticated via valid student or employee ID. The classes “Student”, “Professors” and “Employee” are inherited from the class “Registered User” and associated with the class “Booking” in a one-to-many (1 to *) association.

Guest: The class “Guest” is a user who is not registered in the system. It is inherited from the class “User” and has limited privileges in the system such as requesting navigational instructions.

User: The class “User” is a master class for all the users in the system. It is associated with the class “Navigation Instructions” in a one-to-many (1 to *) association.

Room: The class “Room” is a software component that is used to represent the physical room. It is associated with the classes “Booking”, “Navigation Instructions”, “Display”, “Lock” (1 to 1), “Movement Sensor” and “Booking” in one-to-many (1 to *) association. “Room” contains a total of 6 attributes which are described below: “Room Number”: The attribute “Room Number” is used to identify the room uniquely. It is a primary key for the class “Room”. “Capacity”: The attribute “Room Capacity” is used to identify the capacity of the room or the number of people it can fit. “Room Size”: The attribute “Room Size” is used to identify the size of the room in square meters. “Category”: The attribute “Category” is used to identify the type of the room such as lecture room, seminar room, etc. “Projector available”: The attribute “Projector available” is used to identify whether the room has a projector or not. “Building”: The attribute “Building” is used to identify the building in which the room is located.

VI. System architecture

System architecture serves as the foundational blueprint for the design and operation of intricate systems. It encompasses the configuration of components, their interactions, and the pathways through which data flows to accomplish specified objectives [4]. A well-organised architecture promotes modularity, facilitating efficient development and maintenance by compartmentalising the system into manageable modules. Furthermore, it enhances scalability by accommodating growth and demands without disrupting the entire system [5]. With a focus on reliability, architecture minimises vulnerabilities by distributing tasks and implementing redundancy [4]. By addressing performance, security, and interoperability, the architecture ensures a robust and adaptable system that can evolve over time whilst optimising resource utilisation. Essentially, system architecture constitutes the essential framework that underpins a system’s efficiency, durability, and effectiveness in a dynamic technological landscape [5]. Fig. 4 shows the system architecture of the University Room Management System.

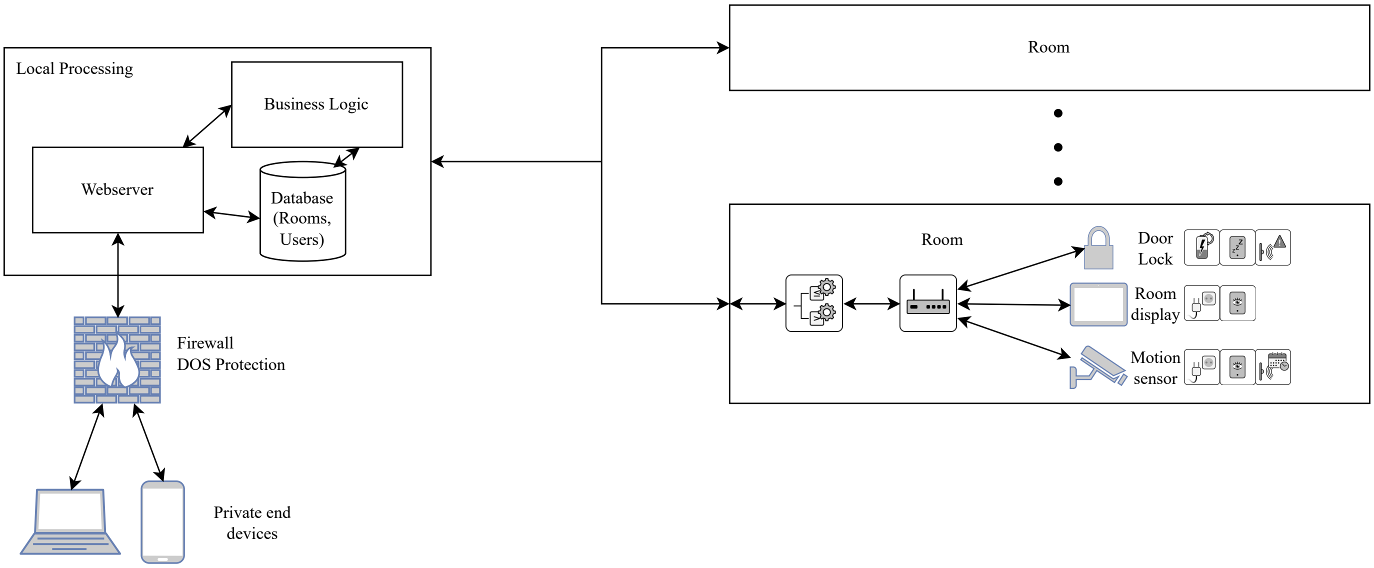


Fig. 4: System architecture of the University Room Management System

The system is bifurcated into three parts, namely, Frontend, Backend, Database and Devices. The Front end is the user interface that is used to interact with the system. The Backend is the core of the system which is responsible for processing the user requests and generating the response. The Database is used to store the system data such as infrastructure metadata or user data. The Frontend and Backend are connected to each other via APIs. The Backend and Database are connected through a network firewall via a physical Ethernet connection. The Frontend and Database are not connected to each other directly but via Backend.

The web server, business logic and database are hosted on different on-prim servers and connected to each other using a physical connection. As this server is locally deployed, all the processing takes place locally and no data is stored on the cloud.

Fig. 4 shows multiple rooms, each equipped with devices such as a room display, door lock and movement sensor are connected to the local servers via wireless connection through a network Gateway and Rule Engine. Each device inside the room may use a different communication protocol (i.e. RF, WIFI, Zigbee, Bluetooth, etc.) To deal with all these different protocols, a Gateway is used to convert the data into a common format and connect to local servers. The Rule Engine is used to process the data and generate the response. The Gateway and Rule Engine are connected to each other via a physical Ethernet connection.

Devices: IoT devices indicated in the system architecture are door locks, room displays and motion sensors. Each device and its nature is represented in the schematic using IoT Patterns. The IoT patterns are used to represent the IoT devices and their nature. The IoT patterns are shown in TABLE III below.

Door Lock: Each room has one door lock that is battery operated has sleep mode and wake-up mode. The only drawback of this is batteries need to be replaced once in a while. The lock has event-based sensing which means it will send the data only when the event occurs. The event in this case is when the door is locked or unlocked or on system request.

Room Display: The displays are connected to an active power grid as it is always on and requires a continues power supply. The display shows the booking schedule of the respective room.

Motion Sensor: The motion is again always on the device, hence connected to the power grid. The motion sensor is scheduled-based sensing which means it will send the data periodically as per schedule. The event in this case is when the motion is detected in the room.

All rooms can be interconnected through a network switch to local servers and subsequently linked to the front end via APIs. To safeguard against external threats, a firewall filters incoming and outgoing traffic. This firewall is connected to the network switch via a physical Ethernet link.

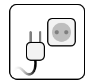
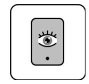
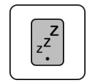
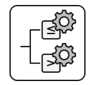
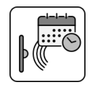
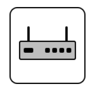
Mains Powered Device	
Period Energy Limited Device	
Always-On Device	
Normally-Sleeping Device	
Rules Engine	
Event-Based Sensing	
Schedule-Based Sensing	
Device Gateway	

TABLE III: IoT Patterns[3]

VII. Experiment and Results

VIII. Discussion

IX. Conclusion

Acknowledgment

Appendix

A. Conversation with ChatGPT

The conversation with ChatGPT described in section IV:

Prompt:

A University Manager system, which is a system where students can book lecture rooms at the university for study and professors can book seminar rooms to give lectures. In addition the University Manager system, give the possibility for students, professor and guests to have navigation instruction for a seminar room they search for.

Write me a use-case specification using the proper and formal structure for that.

A use case specification for the use case called (getting navigation instructions to a seminar room in the university) that is implemented in the University Manager system.

Answer:

Use Case Specification: Getting Navigation Instructions to a Seminar Room in the University

Use Case Name: Get Navigation Instructions to a Seminar Room

ID: UC-NAV-01

Description: This use case allows students, professors, and guests using the University Manager system to obtain navigation instructions to reach a specific seminar room within the university campus. Users can search for a seminar room by its room number or name, and the system will provide step-by-step directions to help them reach the desired location.

Primary Actor: User (Student, Professor, or Guest)

Preconditions:

- The University Manager system is up and running.
- The user is logged into the system (for registered students and professors).

Postconditions:

- The user receives navigation instructions to the requested seminar room.

Trigger: User initiates the request to get navigation instructions to a seminar room.

Main Flow:

1) The user opens the University Manager system's application or website.
2) If required, the user logs into the system using their credentials. (Skipped for guests)
3) The user selects the option to "Get Navigation Instructions" from the main menu.
4) The system presents the user with a search bar to enter the seminar room number or name they wish to navigate to.
5) The user enters the seminar room number or name in the search bar.
6) The system validates the entered information and searches for the requested seminar room in the database.
7) If the seminar room is found:
   a. The system calculates the best route to reach the seminar room from the user's current location or a specified starting point within the campus.
   b. The system generates step-by-step navigation instructions, including directional cues (e.g., left, right, straight), landmarks, and approximate distances.
   c. The navigation instructions are displayed on the user's device screen.
8) If the seminar room is not found:
   a. The system displays an error message indicating that the seminar room could not be located.
   b. The use case terminates.

Alternate Flow:

6a. If the entered seminar room number or name is not valid or does not exist:
i. The system displays an error message indicating that the seminar room could not be found.

ii. The user is given the option to re-enter the room number or name.
iii. The user may choose to cancel the navigation request.
iv. The use case continues from step 4.

Exceptional Flow:

1) If the University Manager system is not operational due to maintenance or technical issues:
   a. The user is informed about the unavailability of the system.
   b. The user may be asked to try again later or contact support for assistance.
   c. The use case terminates.

Notes:

- The University Manager system may integrate with campus maps or GPS services to provide accurate navigation instructions.
- The system could offer additional features, such as displaying accessibility information or estimated time of arrival to the seminar room.
- For security reasons, certain seminar rooms (e.g., faculty offices) may have restricted access for navigation instructions. In such cases, the system should inform the user accordingly.

Prompt:

Use Case Specification: Getting Navigation Instructions to a Seminar Room in the University in terms of :
Name
Scope
Level
Primary actor
Stakeholders
preconditions
postconditions
Main success scenario
Extensions
Special requirements:
Technology / data variation:
Frequency of occurence:
Open issues:

Answer:

Name: Getting Navigation Instructions to a Seminar Room in the University

Scope: University Manager System

Level: User Goal (Primary), Subfunction (Secondary)

Primary Actor: User (Student, Professor, or Guest)

Stakeholders:

- Students: Require navigation instructions to find seminar rooms for academic purposes.
- Professors: Need navigation guidance to reach seminar rooms for conducting lectures or meetings.
- Guests: Visitors to the university who may require navigation assistance to attend seminars or events.
- University Management: Ensures the system enhances campus navigation and user experience.

Preconditions:

- The University Manager system is operational and accessible to the primary actor.
- The user has a compatible device with access to the University Manager system.
- For registered students and professors, the user is logged into the system.

Postconditions:

- The user receives accurate navigation instructions to the requested seminar room within the university campus.

Main Success Scenario:

1) The user opens the University Manager system's application or website.
2) If required, the user logs into the system using their credentials. (Skipped for guests)
3) The user selects the option to "Get Navigation Instructions" from the main menu.
4) The system presents the user with a search bar to enter the seminar room number or name they wish to navigate to.
5) The user enters the seminar room number or name in the search bar.
6) The system validates the entered information and searches for the requested seminar room in the database.
7) If the seminar room is found:
   a. The system calculates the best route to reach the seminar room from the user's current location or a specified starting point within the campus.
   b. The system generates step-by-step navigation instructions, including directional cues (e.g., left, right, straight), landmarks, and approximate distances. c. The navigation instructions are displayed on the user's device screen.
8) If the seminar room is not found:
   a. The system displays an error message indicating that the seminar room could not be located.
   b. The user is given the option to re-enter the room number or name.
   c. The user may choose to cancel the navigation request.

Extensions:

6a. If the entered seminar room number or name is not valid or does not exist:
i. The system displays an error message indicating that the seminar room could not be found.
ii. The user is prompted to re-enter the room number or name.
iii. The user may choose to cancel the navigation request.
iv. The use case continues from step 4.

Special Requirements:

- The University Manager system should integrate with campus maps or GPS services for accurate navigation.
- The system may provide additional features such as displaying accessibility information or estimated time of arrival to the seminar room.
- Certain seminar rooms may have restricted access, and the system should handle these cases appropriately.

Technology / Data Variation:

- The system should be accessible via various platforms (web, mobile apps) to accommodate different user devices.
- The availability of campus map data and GPS accuracy may vary depending on the university's infrastructure.

Frequency of Occurrence:

- The use case is expected to occur frequently throughout the academic year as students attend lectures, professors conduct classes, and events are held.

Open Issues:

- How will the system handle navigation to seminar rooms located in different buildings or floors within the campus?
- Should the system provide additional options such as displaying alternative routes or real-time updates for room availability?
- How will the system ensure privacy and security of users' location data while providing navigation instructions?

Prompt:

write me a user story using the INVEST principal for the following user story:

user story: As a Professor, I want to book a room according to a schedule so that I can give regular lectures in that room.

Answer:

User Story: As a Professor, I want to book a room according to a schedule so that I can give regular lectures in that room.

INVEST Principle:

- **Independent:** The user story is self-contained and can be implemented without dependencies on other user stories. It focuses on the specific need of the Professor to book a room for regular lectures.
- **Negotiable:** The details of the implementation are open to negotiation between the Professor (user) and the development team. The team can collaborate with the Professor to determine the specific room booking requirements and scheduling preferences.
- **Valuable:** The user story provides clear value to the Professor by enabling them to secure a designated room for their regular lectures, ensuring consistent teaching arrangements and an organized learning environment.
- **Estimable:** The user story is well-defined and has a clear scope, making it estimable for the development team. The effort required to implement the room booking functionality can be reasonably assessed.
- **Small:** The user story has a focused and concise scope, which allows the development team to implement it within a single development iteration.
- **Testable:** The user story's acceptance criteria are specific and measurable, allowing for effective testing. It is possible to verify the successful implementation of the room booking feature based on the defined criteria.

Acceptance Criteria:

- The Professor can access the University Manager system and navigate to the "Room Booking" section.
- The system presents a form with options to select the desired room, date, start time, and end time for the lecture.
- The Professor can enter the required details and submit the booking request.
- The system validates the availability of the selected room for the specified date and time.
- If the room is available, the system confirms the booking and sends a confirmation email to the Professor.
- The booked room should be reserved in the system's schedule for the specified date and time.
- The Professor can view, modify, or cancel their booked lectures within the system as needed.
- In case of any conflicts or unavailability, the system provides alternative room options or available time slots for the Professor's lectures.
- The system ensures that only authorized Professors can access the room booking functionality.

Additional Considerations:

- The system should have user authentication to ensure that only authorized Professors can access the room booking feature.
- The system may send reminders or notifications to the Professor before the scheduled lecture time.
- The room booking feature should be user-friendly and accessible via multiple platforms, such as web and mobile applications.
- The system should handle and display room availability in real-time to avoid double bookings.
- The user interface should be intuitive and provide clear feedback on the status of the room booking process.

Note: The user story may be further elaborated during backlog grooming or refinement sessions to capture more detailed requirements and user expectations.

Prompt:

write me a user story using the INVEST principal for the following user story:

user story: As a University member, I want to use the Room Management System, because it will make life easier.

Answer:

User Story: As a University member, I want to use the Room Management System because it will make life easier.

INVEST Principle:

- **Independent:** The user story is self-contained and can be implemented independently without relying on other user stories. It focuses on the specific need of the University member to utilize the Room Management System for simplifying room-related tasks.
- **Negotiable:** The user story allows for negotiation between the University member (user) and the development team to identify the specific features and functionalities that will enhance their experience with the Room Management System.
- **Valuable:** The user story holds clear value for the University member by offering a convenient and efficient solution for managing room-related activities, streamlining processes, and saving time.

- **Estimable:** The user story is well-defined and has a clear scope, making it estimable for the development team. The effort required to implement the Room Management System features can be reasonably estimated.
- **Small:** The user story has a focused scope, enabling the development team to implement it within a single development iteration.
- **Testable:** The user story's acceptance criteria are specific and measurable, enabling effective testing. The success of the Room Management System can be assessed based on the defined criteria.

Acceptance Criteria:

- The Room Management System should be accessible to all University members through a user-friendly web application or mobile interface.
- The system should have user authentication to ensure that only authorized University members can access and use the Room Management features.
- University members should be able to log in using their credentials or university ID.
- The Room Management System should provide a clear and intuitive user interface with navigational menus and visual cues.
- University members should be able to view available rooms, their schedules, and occupancy status in real-time.
- The system should offer a simple and efficient process for booking or reserving rooms for various purposes (e.g., lectures, study sessions, meetings).
- Room availability should be displayed in a visual calendar format, with filtering options based on date, time, location, or room amenities.
- The system should send notifications or reminders to users about upcoming room reservations or changes in room status.
- University members should be able to easily modify or cancel their room reservations within the system.
- The Room Management System should maintain accurate and up-to-date records of room bookings and usage.
- The system should have a responsive design to ensure usability across different devices and screen sizes.

Additional Considerations:

- The Room Management System could incorporate features like room occupancy limits, equipment availability, and room-specific details.
- To enhance user adoption, the system could provide user tutorials or help documentation for first-time users.
- The system should prioritize data security and privacy, ensuring that only authorized personnel can access sensitive room-related information.
- Regular user feedback and iterative improvements can be incorporated to enhance the Room Management System based on user preferences and needs.

Note: The user story may be further elaborated during backlog grooming or refinement sessions to capture more detailed requirements and user expectations

References

[1] C. Larman, *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. Pearson Education India, 2012.

[2] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[3] G. Booch, "Oo analysis and design with applications," 1994.

[4] M. v. Steen and A. S. Tanenbaum, "Distributed systems: principles and paradigms," *Pearson*, 2007.

[5] I. Kotseruba, O. J. A. Gonzalez, and J. K. Tsotsos, "A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications," *arXiv preprint arXiv:1610.08602*, pp. 1–74, 2016.