Cloud Surf Inn:
Report 3

Submitted To: Software Engineering

Submitted On: 05/03/2021

Submitted By: Group 8

Website: https://sites.google.com/view/cloud-surf-inn/home?authuser=0

Github: https://github.com/Software-Engineering-Team-8/Cloud-Surf-Inn

Group Members: Chynna Walsh, Sidonia Mohan, Ryder Morrello, JP Dangler, Juan Escudero, Zach LeMunyon, Chris Kline, Bharath Selvaraj, Jakub Vogel, Sebastian Matiz

Contribution Breakdown within Subsystem:

All team members contributed equally within their subsystem. The work breakdown structure is explained in the first section of the report.

Table of Contents

1. Work Breakdown

From the decomposition of sub-problems, four teams were made to address various functional features that if successful will become solutions to the problems. The functional features that address the main problems are labeled as, “Housekeeping”, “Mobile Key”, “Power Controls”, and “Room Matcher.” They subsequently have team codes of A, B, C, and D. Teams B and D work together on the same solution but are separated to give more individual focus to certain features. Group assignments were based on past experience working together, similar skill sets, and knowledge of the subsystems.

Team Assignments are as followed:

Team	Members	Team Code
Housekeeping	Juan, Ryder, Zach	Team A
Mobile Key	Sidonia, Bharath	Team B
Power Controls	Jakub, Sebastian, Chynna	Team C
Room Matcher	JP, Chris	Team D

The assignment of use cases was decided by individual student preference and subteam vision. In summary of changes for each subteam, a breakdown of use case evolovment should be discussed.

Report 1 Use Case Assignments by Team:

Team A:

Housekeeping UC-16, UC-17, UC-21, UC-25

Online Ordering UC-19, UC-23, UC-24, UC-20

Team B:

Mobile Key UC-6, UC-8, UC-9, UC-10, UC-29

Management UC-7, UC-18, UC-22, UC-27, UC-28

Team C:

Power Controls UC-11, UC-12, UC-14, UC-14, UC-15

Team D:

Room Matcher UC-1, UC-2, UC-3

Payment UC-3, UC-4

Functional Use Cases:
UC-6, UC-7, UC-10, UC-1, UC-2, UC-5, UC-11, UC-12, UC-13, UC-14, UC-15, UC-25, UC-16, UC-17, UC-19, UC-23, UC-25, UC-26

2. Customer Statement

Customer (Hotel Patron)

I usually love staying at hotels but the pandemic has brought a damper on going out and staying at my favorite hotels. I have a high-risk condition and because of that, I am very conscientious of the health and safety standards of the place I am staying at. A lot of the interactions I have with hotel staff force me to be physically closer to other people than I am comfortable with being during this pandemic. If there was a way for me to get my hotel key without having to go to the front desk and potentially wait in a line and have to interact with the front desk, I would be much happier to go there and would feel much safer. I also hate how my room key is not only often misplaced but becomes demagnetized next to my phone! However, when I finally get in my room and have some time to myself, I can never find the food I want or the shows I enjoy. Oftentimes the menu is scattered on one of the tables, or sometimes even missing entirely! And when I finally get some good food to eat, what better way to eat than with my favorite show! But half the time I try to find it, I end up fumbling through channels, trying to find the channel list, or giving up and just calling the front desk. It would be beneficial and eliminate frustration if there was just one single place where I could get my TV shows or movies, and order my food, that way I would not have to waste my time looking in several different places. I also enjoy my rooms being kept very tidy and sanitary but can never find a maid when I need her. I wish I could just schedule housekeeping to come and clean my room at a certain time so that I do not have to interact with the maid directly, and I also get my room cleaned right when I want it.

An overall downside with hotels is the room temperatures! I find myself too cold or too hot and the thermostats are so old you can not easily change the temperature or I have to get up to change it! It would be so much more convenient to change the room temperature through my phone! I also get frustrated when I am assigned a room without any note of my preferences! Or worse, I have to search through hundreds of rooms and none of them are desirable.

How will Cloud Surf Inn help me? A multifaceted app that could handle different aspects of my hotel experience would be a big boost to my experience. Since I ordered my room on the

app, why can it not just serve as my hotel pass as well? This way I would not have to worry about interacting with the hotel staff and could go straight to my room, keycard on my phone. And, if this app was integrated into the TV and had the TV shows, movies, and menus listed all in one place, I would have a much easier time ordering pizza while watching *The Office*. If I could also schedule housekeeping on the app, it would make my day being able to request housekeeping even when I am not in the hotel. Plus, being matched with a room according to my preferences removes the need for me to switch rooms because I am unsatisfied. I imagine the app to be easy to use, everything straightforward so I don't need to worry about clicking around finding the information! A website or any user interface would work just as well, as long as I have no human interaction.

Owner

As the owner of a large hotel, making sure guests feel comfortable and are highly satisfied with their stay is my number one priority. With the pandemic, however, I have seen fewer and fewer people go through my doors and rent rooms than ever. A large part of that I believe is because people do not feel comfortable with the hotel stay experience right now. If I could eliminate all but the vital interactions between customer and employee, I would have a much better time advertising to customers about the safety of my hotel. Another thing I would like to do is cut my costs wherever I can. I feel like some of my employees sometimes spend their time doing non-productive work. The maids, for example, spend a larger portion of their time walking from room to room, checking if there are people there, rather than performing the housekeeping service. I would like a better way for their services to be used and only schedule their shifts when people need the service to save money spent unnecessarily. I am also very passionate about being environmentally friendly; and unfortunately, I feel a lot of the energy we use is wasted. I would love to be able to turn down thermostats and turn off lights automatically when guests are not in use of a room; it is too difficult to get the staff to individually do this and I would end up paying them more than I would save!

How would Cloud Surf Inn help me? Contactless check-ins or housekeeping requests would greatly increase the comfortability of my guests and also reduce the time my employees spend doing non-productive things. I would love it too if the lights would automatically dim and the heat would be turned down when a room becomes vacant. I would then feel confident in

saying that my hotel is doing the best it can to be sustainable. The streamlined management on top of customer satisfaction reducing room vacancies is sure to increase revenue.

Staff (Housekeeper/Custodian)

I have lots of rooms to clean in this hotel and it gets challenging to keep track of what rooms I have taken care of and what rooms still need to be cleaned. I also often accidentally run into customers if they think the cleaning is done, or they do not announce themselves and I accidentally enter. During these pandemic times, I would especially like to avoid unnecessary contact with the customers. If I just knew when customers needed cleaning and could let them know somehow that it was done, it would be much easier to provide housekeeping that better serves them and also lets me get my work done quicker and easier. It is concerning to use one master key to open all bedrooms because if I lose it, whoever finds it has access to all the rooms in the hotel- compromising security. Also, with the pandemic especially, it is hard to keep track of high-traffic areas, like elevators, and to keep them clean. If I was able to tell when someone last cleaned that area, I would have a much easier time keeping the hotel clean for myself and others. It is also frustrating to waste so many extra towels and shampoo bottles as they must be thrown out after a guest checks out. People often steal them from the cart as well making me run out before I finish cleaning!

How would Cloud Surf Inn help me? Having a schedule of when the rooms need to be cleaned as requested by guests would help me greatly. I would be able to get my work done much faster and be able to keep track of what rooms I have done, and what rooms still need to be cleaned. I also reduce the amount of wasted resources as guests request how many towels, or shampoo bottles they need. If the system also had a part that could keep track of high-traffic areas, and let me know if they need to be cleaned again, I would be able to keep the hotel much safer for everyone.

Managers

As a hotel manager, things are stressful and hectic, to say the least. Properly communicating things to other staff is a tough thing to get right, and when it comes to serving our guests, every detail matters. It is incredibly difficult to keep track of things happening around the hotel on any given day, and it is even harder to keep track of the guests and their requests. As a manager, guests constantly seek me out for things and I have to make sure I remember and tend to everything they tell me. Whether it is to get clean towels for a room,

request a room change, or request a new key, I have to remember everything and make sure it gets done. What makes this worse is keeping track of the rest of the staff. Oftentimes, keeping track of everyone else can be a problem all on its own. I worry about micromanaging my fellow staff members by reminding them of small tasks that need to be done and who they need to be done for.

I also feel that the hotel is pretty wasteful. Not just environmentally, but also in terms of how we employees spend our time. We waste a great deal of electricity powering rooms, hallways, and services when guests are not using them, and sometimes staff forgets to power things off when not in use. It feels incredibly wasteful. I wish we could allow rooms to go into some sort of "power-saving" mode where they consume as little as possible. Regarding our time, on slow days or days where guests spend most of their time outside of the hotel, the staff stays relatively inactive, and sometimes the hotel is even overstaffed.

How will Cloud Surf Inn help me? Well, I think it addresses a lot of the issues I mentioned. Having a system where I can manage and view information about guests and the hotel itself is incredibly useful; it lets me focus on other aspects of management. Instead of micromanaging my fellow employees, I can concentrate on making sure everything is moving smoothly in addition to making sure high-level priorities are kept in check. It also goes without saying that making sure that the hotel is functioning in an efficient and environmentally friendly manner is solved. With Cloud Surf Inn, I can change my management style so that my sole purpose is to satisfy clients and let the system handle the rest.

Staff (Reception Area)

Working at the front desk is a much harder task than most people would think. Whenever guests need someone for something (whether good or bad), they come to me and my fellow receptionists. Checking guests in and out is also a lengthy process that stresses both me and the guests. On hectic days, it is very easy to make a mistake, and I can feel the frustration of the guests when these things happen. Naturally, I feel devastated when I cannot do my job perfectly, but there is very little I can do when the check-in and check-out processes are so tedious. I am also tasked with dealing with surly customers who are unsatisfied with the room they have been given, and getting them a new one requires much face-to-face interaction to assign a new key. To make matters worse, our current system for keeping track of rooms and guests is incredibly outdated and prone to errors. I have to be incredibly careful to not give a guest a room that is

already occupied, and sometimes, empty rooms are marked as occupied by the system. The Covid-19 pandemic has also made things even more difficult. Carelessly handling money, checks, and credit cards is something that cannot be taken lightly. We also limit the number of people in the reception area at any given time for safe social-distancing measures but this makes communication with guests difficult. It also makes giving out keys harder as we want to be as sanitary as possible. There has to be a simpler and more sanitary way to handle guest reception.

How will Cloud Surf Inn help me? Having guests able to book a room from the comfort of their own home makes everything much easier when they arrive. On top of that, guests can check themselves out, so even that is automated. Handling payment information is easier and more straightforward than ever. Cloud Surf Inn also accurately tracks information about the rooms and guests so I no longer have to rely on our old and obsolete system. I don't even have to worry about hotel keys as the automated hotel gives a unique room code in the confirmation email, and automatically re-generates a new code if there's a room change- eliminating guests needing to ask me for a new keycard. In short, Cloud Surf Inn makes everything about my job easier.

Staff (Service)

Serving guests and keeping them satisfied is my specialty, but I will admit that doing so is not always easy. It is incredibly difficult to keep track of which guests request what, especially when things are busy. For example, room service to a guest is simple when they request a couple of small items, but when multiple guests request service of multiple items, it becomes nearly impossible to properly queue and coordinate orders with the kitchen. Taking orders with phone calls and writing them done with a pencil and paper is not enough to keep up with orders on a busy day. I also find myself handling money or needing credit card signatures forcing face-to-face contact.

How will Cloud Surf Inn help me? I think it goes without saying that Cloud Surf Inn is going to make my life a whole lot easier. Being able to see orders and requests neatly organized within a user interface doesn't just help me, it helps the kitchen and other service staff as well. We can focus our energy on keeping guests happy and not on keeping ourselves organized. The automatic payment processing is also valuable to reduce contact and increase sanitation. I think it is also important to mention that being able to request services is also useful to our guests. They

can browse menus, request utilities, and see what we have to offer without needing to call anyone to find out; everything is at their fingertips.

3. User Interface Specifications

Power Controls:

Team Members: Jakub Vogel, Chynna Walsh, Sebastian Matiz

Requirements:

Identifier	Prioirty	Description
CREQ-1	4	As a user, I can view the current room temperature in the room from the interface.
CREQ-2	3	As a user, I can input a new room temperature in the room from the interface.
CREQ-3	2	As a user, I can turn off/on ventilation in the room from the interface.
CREQ-4	2	As a user, I can turn off/on the lights in the room from the interface.
CREQ-5	1	As a user, I can view the thermostat constraints on the interface.

Requirements Not Implemented:

Identifier	Description
CREQ-*1	As a user, I can view the thermostat constraints on the interface.
CREQ-*2	As a user, I can turn comfort mode/motion sensors off.
CREQ-*3	As a user, I can schedule an arrival time when I return to my room.
CREQ-*4	As a user, I can input my preferred comfort settings on the interface.

User Interface Use Case:

Identifier	Description	REQs Used
CUC-*1	“Room Control Page” The user interface has a room controls page that a user can access.	CREQ-1, CREQ-2, CREQ-3, CREQ-4,

		CREQ-5, CREQ-*1, CREQ-*2, CREQ-*3, CREQ-*4,

Report 1 Preliminary Design:

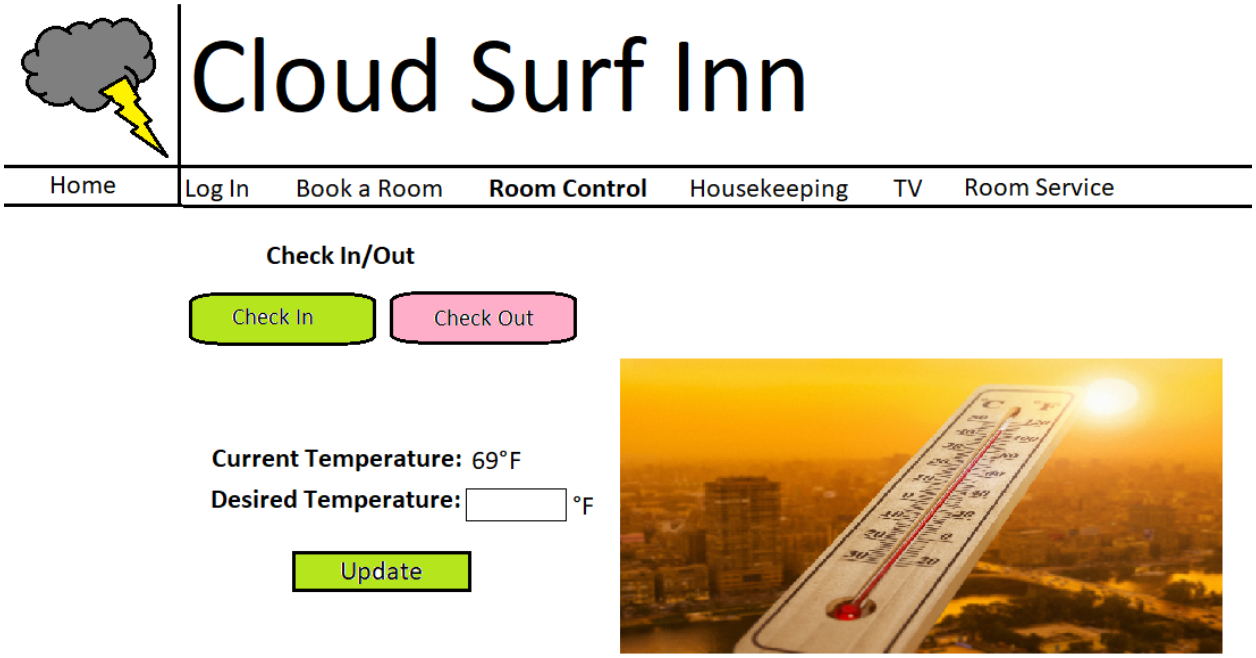


Figure: 3.1 This is the Room Control page where users will view the temperature in the room, and control the room temperature. To get to this page the user must click the "Room Control" tab near the top of the screen. The current temperature will be displayed above a textbox where the user can enter their desired temperature. Once they have entered the desired temperature into the textbox, the user selects the "update" button to update their preferences.

Updated Preliminay Design:

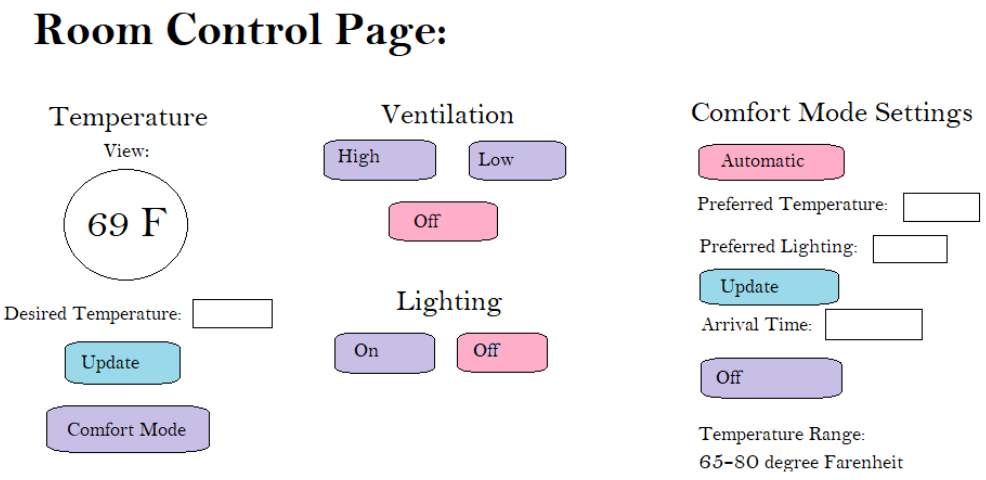


Figure 3.2: The room control page's updated preliminary design includes additional REQs and Use Cases. The user would login to the interface and select the room control tab. The features from the previous design are still intact and a user can input and update temperature. They now have the option to turn the lights on or off, decide what preferences they want for comfort mode, and level of fan ventilation.

Current Implemented Design:

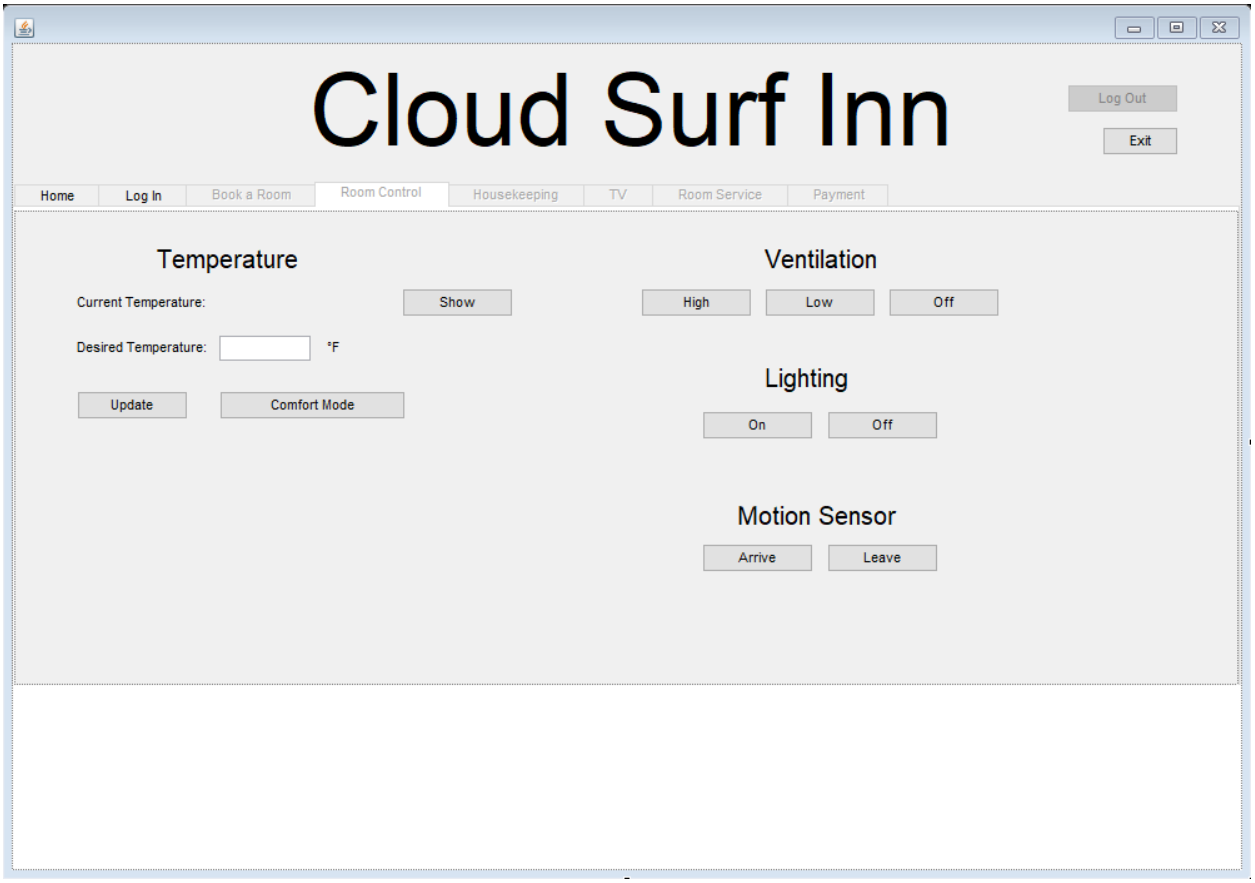


Figure 3.3: Implemented user interface for power controls. Use cases UC-11, UC-12, UC-13, UC-14, and UC-15 are shown. Motion sensor is used to signal the system as it is not yet implemented. (UC-*1)

User Effort Estimation:

Room Control Page:

Average 3 keystrokes and 5 mouse clicks.

Starting at Homepage

a. Select Room Control tab
b. Type in desired temperature in textbox
c. Select update to change
d. Select ventilation high/low/off

e. Select lighting on/off

f. Select off for automatic comfort mode.

Housekeeping:

Team Members: Ryder Morrello, Juan Escudero, Zach LeMunyon

Requirements:

Identifier	Priority	Description
HREQ-1	4	Housekeepers must be able to see when requests for cleaning/housekeeping services are made in a queue.
HREQ-2	3	Housekeepers must be able to update the status of cleaning/housekeeping requests.
HREQ-3	3	Housekeepers must be able to clock in and clock out.

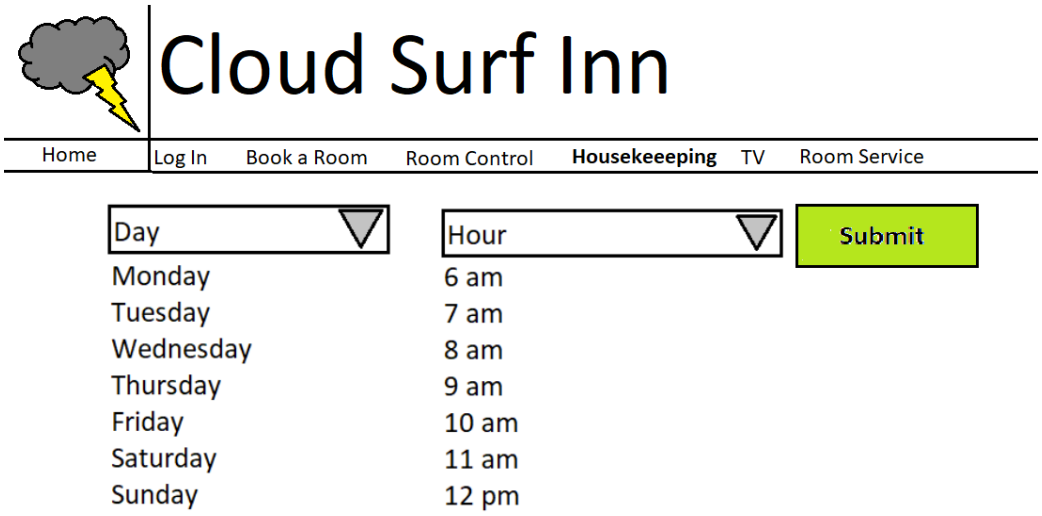
Requirements Not Implemented:

Identifier	Description
HREQ-*1	Housekeepers must be able to see requests for when the room is vacated and must be cleaned before the next guest moves in.

User Interface Use Case:

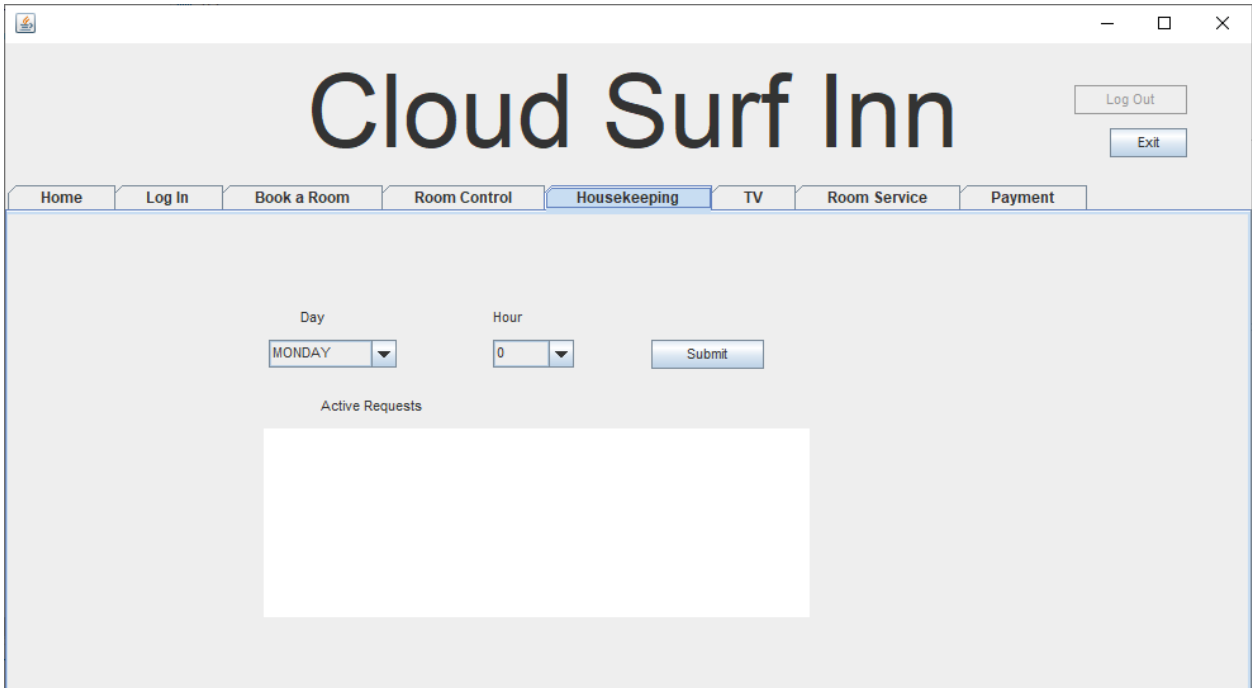
Identifier	Description	REQs Used
HUC-*1	“Housekeeping Page” The user interface has a housekeeping page where all housekeeping requirements are met.	HREQ-1, HREQ-2, HREQ-3, HREQ-*1

Report 1 Preliminary Design:



This is the Customer Housekeeper page where customers can select and time and day they would like their room cleaned (REQ-11,CREQ-9). To get to this page you must be a logged in customer who is staying at a room, and click the “Housekeeping” tab. There will be two combo boxes on this page, one for the day of the week and one for the specific hour you would like housekeeping to come and clean the room. After a day and time have been selected, the customer must click the “Submit” button to have their request fulfilled.

Updated/Current Preliminary Design:



Here customers can select a time and day for a room cleaning (REQ-11, CREQ-9). Customers use up to 5 clicks to request housekeeping, login, the tab selection, a day, hour, and to submit.

Mobile Key:

Team Members: Sidonia Mohan and Bharath Selvaraj

Requirements

Identifier	Priority	Description
CREQ-1	4	Customers will choose the login page.
CREQ-2	3	Customer inputs login information.
SREQ-1	4	Staff will choose the login page.
SREQ-2	3	Staff inputs login information.
MREQ-1	4	Manager will choose the login page.

MREQ-2	3	Manager will login to the manager page.

Requirements Not Implemented:

Identifier	Description
CREQ-*1	Customers will have the option to choose either “Forgot Password” or “Forgot Username” if they forget their login information.
SREQ-*1	Staff will have the option to choose either “Forgot Password” or “Forgot Username” if they forget their login information.
MREQ-*1	Manager will have the option to choose either “Forgot Password” or “Forgot Username” if they forget their login information.

Report 1 Preliminary Design

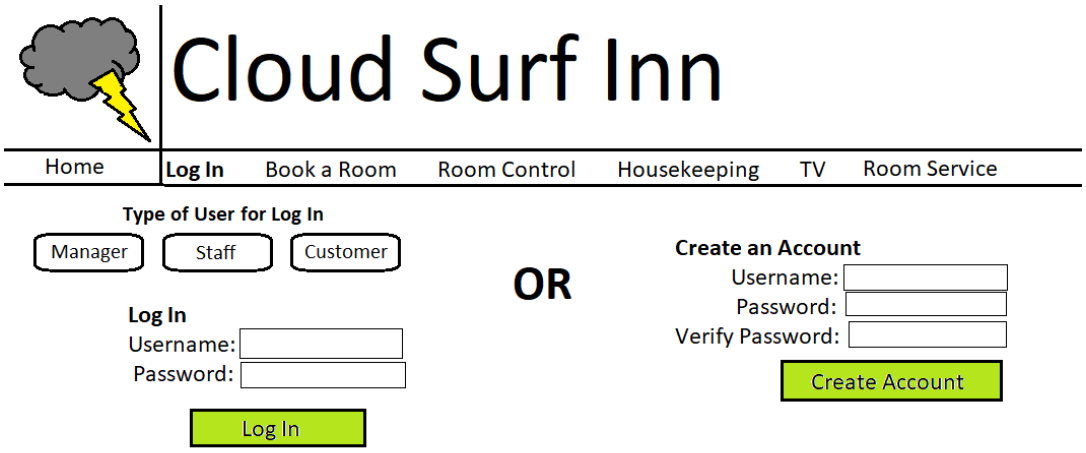


Figure 3.4: This was the original idea that we chose to implement for the login page. We wanted to have a UI that could both create and account for a user and to be able to log into the page, but we also wanted to make it look organized and neat. Above you can see all the things we were planning on implementing which we have done and created, such as housekeeping requests and booking a room etc.

Updated Preliminary Design

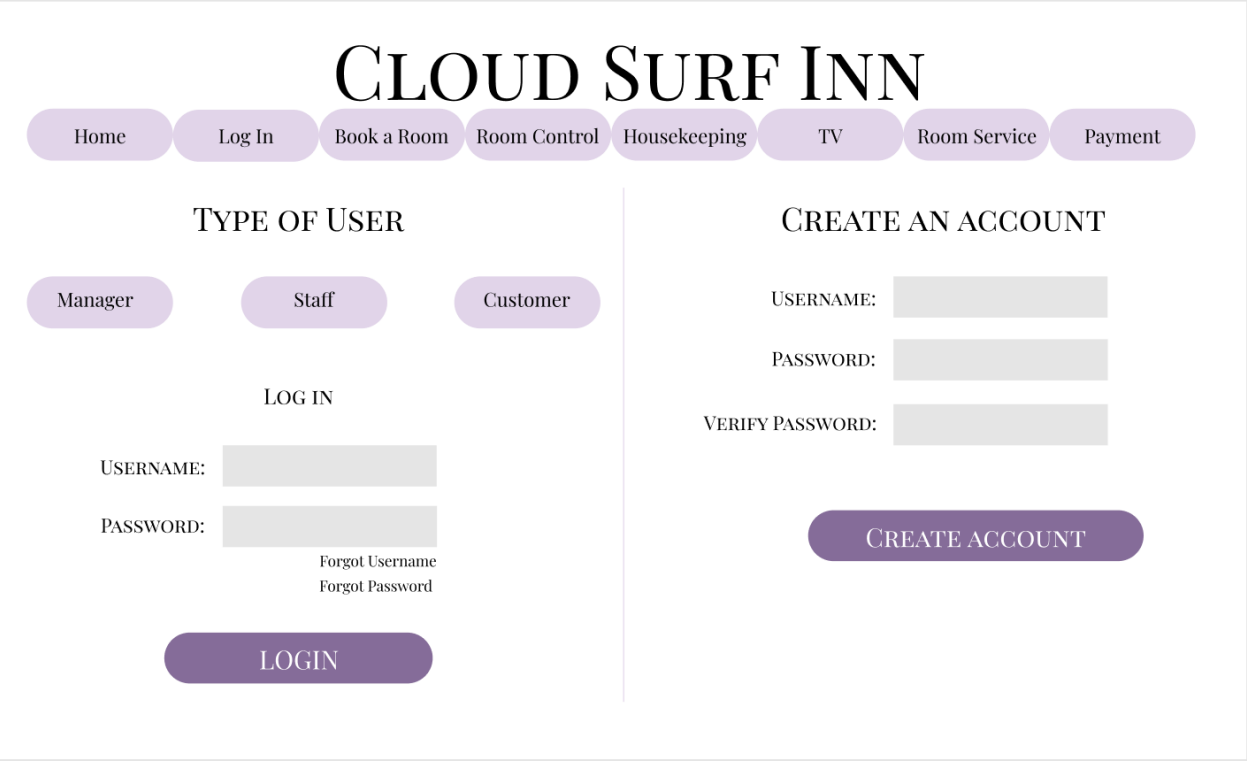


Figure 3.5: The login page updated preliminary design includes additional requirements. The user will be able to access this under the “Log In” tab. While the functionality of the login page remains intact, the updated version takes into account “Forgot Username” and “Forgot Password”. This gives the user the opportunity to still login to their respective account, even if they forget their login information.

Current Implemented Design:

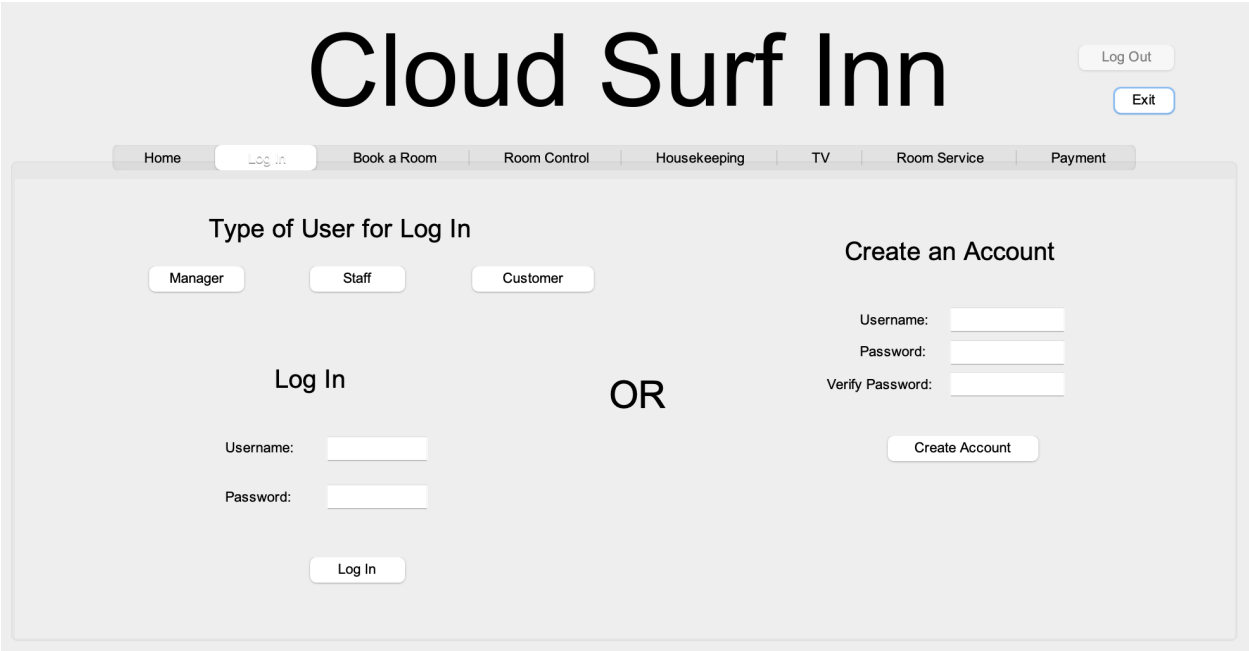


Figure 3.6: This is the most current version of our login page. It is not much different from what we originally designed, but still has all the same features and then some. We wanted to make the login page look clean, as the other designs we had implemented were cluttered so we opted for this apple inspired look. Above you can see we also added a payment tab which was not in our original idea.

User Effort Estimation

Login Page:

Average 3 keystrokes and 6 mouse clicks

Starting at Homepage

a. Select Log In tab
b. Type in desired username under “Create an Account”
c. Type in desired password under “Create an Account”
d. Select “Create Account”
e. Pick type of user

f. Type in username
g. Type in password
h. Select “Log in”

Room Matcher:

Team Members: John Paul Dangler, Christopher Kline

Requirements:

Identifier	Priority	Description
REQ-17	5	The system has a database of rooms containing occupancy, room number, cleanliness.
REQ-18	4	The system matches rooms to guests based on preferences.
REQ-19	4	The system allows customers to individually choose from available rooms and opt-out of the room matcher.
REQ-16	2	The system allows customer to input credit card number, holder, expiration date, and CVC

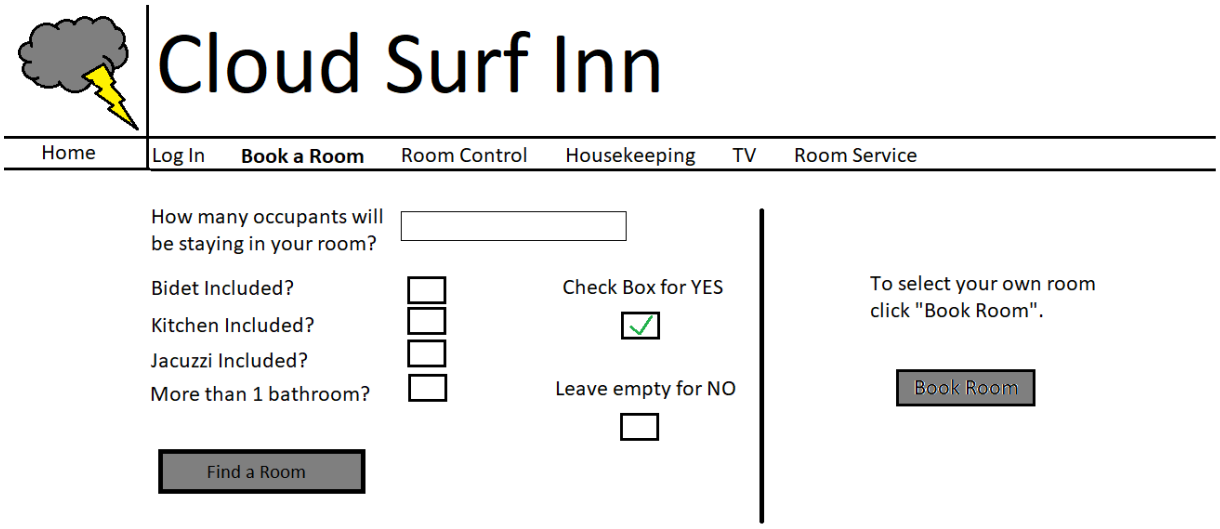
Requirements Not Implemented:

Identifier	Description
REQ-31	All items ordered by a customer are kept on file to pay at the end of visit

User Interface Use Case:

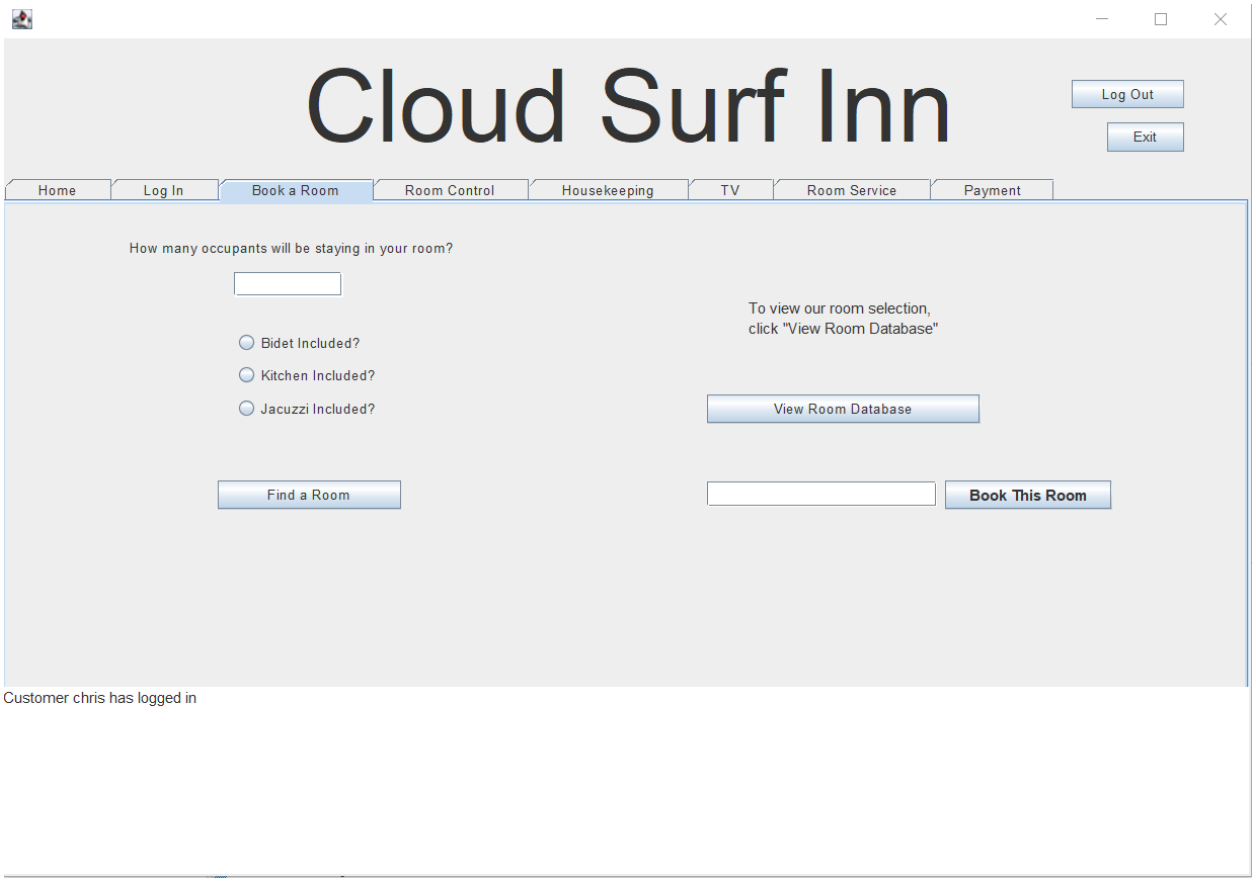
Identifier	Description	REQs Used
RUC-*1	“Book a Room Page” The user interface has a room booking page where all room booking/room matching use cases are met.	REQ-17, REQ-18, REQ-19
PUC-*1	“Payment Page” The user interface has a payment page where the payment use case is met.	REQ-16

Report 1 Preliminary Design:



This is the Book a Room page where customers and guests are able to book a room to stay at. To get to this page the user must click the “Book a Room” page near the top of the screen. Displayed is one text box for telling the application how many occupants will be in the room, and all other boxes are for radio style buttons for preferences included in rooms as well as opting out of room matching where the customer will be able to view available rooms (REQ-19). After they have filled out everything they would like in the room, they must click the “Book Room" (CREQ-3, CREQ-4) button at the very bottom of the page. They will then be matched to a room (CREQ-5).

Updated/Current Preliminary Design:



On the updated book a room page, we have a functional “View Room Database” button, allowing the user to see all rooms and their data. Furthermore, the room matching function now simple outputs a room number for the user to choose to book or not. From here, the user would use the “Book this room” button, and input a room number to book. All requirements for the room matcher page have been met.

User Effort Estimation:

Starting at Homepage

1. Select login tab
2. Enter username
3. Enter password
4. Click “Book a room” tab
5. Enter Room number
6. Click “Book this room”

Average of 3 keystroke and 3 mouse clicks.

4. Subsystem: Power Controls

Team C: Chynna Walsh, Jakub Vogel, Sebastian Matiz

4.1 Summary of Changes

The requirements and use cases were all updated to accommodate the upgrading of the subsystem. Functionality of use cases and requirements remains intact however, all were renumbered or revised. The addition of use cases and requirements were added to accommodate updated functionality, any added REQ or UC is marked with *. Additionally, all diagrams were updated to accommodate the changes made to the requirements and use cases. The diagram changes can be observed in the indicated section of the report. The concept models were renamed and the boundaries were updated but in essence they are still the same.

Report 1 Description	Update
UC-11 “Set Thermostat” - Allow customers to set the thermostat online to their desired temperature.	UC-11 “Thermostat Control” - Allow customers to set the thermostat online to their desired temperature and view the current temperature.
UC-12 “View Current Temperature” - Allow customers to view current temperature.	UC-12 “Ventilation Control” - Allow customers to set ventilation levels online.
UC-13 “Automatic Power Saving Mode” - If room is vacant “power save” mode is turned on	UC-13 “Automatic Power Saving Mode” - If a room is vacant, “power save” mode is turned on.
UC-14 “Automatic Comfort Mode” - If room is booked, “comfort” mode is turned on	UC-14 “Automatic Comfort Mode” - If a room is occupied, “comfort” mode is turned on.
UC-15 “Automatic Hallway Lights” - At night lights turn off and are motion sensored.	UC-15 “Lighting Control” - Allow customers to control the lighting in their room online.

Report 1 Description	Change/Update
REQ-3 The control system receives input upon check-in or check-out that a room is occupied or vacant from the database.	REQ-3 The system shall automatically shut off the thermostat in an empty room upon check-out and if the customer leaves the room.

REQ-4 The control system automatically shuts off heat and power in a room when vacant.	REQ-4 The system shall automatically shut off lights and power in an empty room upon check-out and if the customer leaves the room.
REQ-5 If temperature is set to an invalid setting a warning message is displayed.	REQ-5 The system shall automatically ventilate an empty room upon check-out.
REQ-6 A temperature sensor sends the temperature data to the system.	REQ-6 As a user, I have access to the room lighting controls.
REQ- 7 The system displays the temperature.	REQ-7 As a user, I have access to the room thermostat controls.
REQ-8 System allows the temperature to be manually changed through the user interface.	REQ-8 As a user, I have access to the room ventilation controls.
REQ- 29 The system turns off lights in hallways during the night.	REQ-29 As a user, I can expect my room to be at a comfortable temperature when I first arrive in my room, and at any point when I come back.
REQ- 30 Motions sensors turn on dimmed lighting at night.	REQ-30 As a user, I can expect my room to be well lit when I first arrive in my room, and at any point when I come back.
REQ- 41 Room turns on comfort mode when check-in activated	REQ-41 The system will not enter power-saving mode if the customer leaves and then enters the room within 10 minutes.

4.2 Glossary of Terms

Power Saving Mode: A mode that automatically turns down the heat, turns off lights, and signals ventilation to turn on. The main goal of this mode is sustainability.

Comfort Mode: A mode that automatically turns heat to a preset level and turns on lights. The main goal of this mode is user comfort.

Power Controls: The hardware system contains sensors, raspberry pi, fans, LEDs and a breadboard that controls the thermostat and sensors.

4.3 System Requirements

4.3.1 Enumerated Functional Requirements

Identifier	Priority	Description
REQ-3	5	The system shall automatically shut off the thermostat in an empty room upon check-out and if the customer leaves the room.
REQ-4	5	The system shall automatically shut off lights and power in an empty room upon check-out and if the customer leaves the room.
REQ-5	5	The system shall automatically ventilate an empty room upon check-out.
REQ-6	2	As a user, I have access to the room lighting controls.
REQ-7	4	As a user, I have access to the room thermostat controls.
REQ-8	2	As a user, I have access to the room ventilation controls.
REQ-29	3	As a user, I can expect my room to be at a comfortable temperature when I first arrive in my room, and at any point when I come back.
REQ-30	3	As a user, I can expect my room to be well lit when I first arrive in my room, and at any point when I come back.
REQ-41	2	The system will not enter power-saving mode if the customer leaves and then enters the room within 10 minutes.

Requirements Not Implemented:

Identifier	Description
REQ-*1	The system overrides arrival time of guests if the motion sensor is activated.
REQ-*2	The system turns on comfort mode settings 10 minutes prior to arrival time.
REQ-*3	As a user, I have access to the room control settings.

4.3.2 Enumerated Nonfunctional Requirements

None.

4.4 Functional Requirement Specification

4.4.1 Stakeholders

Hotel Owners: Have a personal interest in Cloud Surf Inn as it allows for greater power efficiency in the hotel and a more pleasurable experience for their guests.

Hotel Guests: Cloud Surf Inn will lead to a more enjoyable, relaxing, pandemic-friendly experience for hotel guests who will utilize the system repeatedly throughout their stay.

Software Engineers: Want to make a system that is easily used that has the potential to maximize the efficiency of the hotel business and continuously improve the system.

Prisons Owners: Want to see a power-saving system that can be transferred over and used in their prisons.

4.4.2 Actors and Goals

Initiating Actors

Actor	Goal
Guest/Customer	The guest is a customer staying at the hotel; they use the Cloud Surf Inn system to satisfy their desires and necessities.
Check-In/Out Subsystem (Team B)	Another subsystem responsible for check-in/out that sends a signal to the power control subsystem of a user check-in/out.
Motion Sensor	A device that detects when a guest enters or leaves the room and signals the power control subsystem of such an event.

Participating Actors

Actor	Role

ThermostatDevice	The thermostat has the responsibility of sensing the current temperature in the room as well as being able to modify it.
LightDevice	The lighting device will turn on and off the lights in the room.
FanDevice	The fan device turns on and off the fan in the room for ventilation.
Database	The database will log important events when they happen.

4.4.4 Casual Descriptions

Identifier	Description	REQ-# Used
UC-11	“Thermostat Control” - Allow customers to set the thermostat online to their desired temperature and view the current temperature.	CREQ-1, CREQ-2, REQ-7
UC-12	“Ventilation Control” - Allow customers to set ventilation levels online.	CREQ-3, REQ-8,
UC-13	“Automatic Power Saving Mode” - If a room is vacant, “power save” mode is turned on.	REQ-3, REQ-4, REQ-5
UC-14	“Automatic Comfort Mode” - If a room is occupied, “comfort” mode is turned on.	REQ-29, REQ-30
UC-15	“Lighting Control” - Allow customers to control the lighting in their room online.	CREQ-4, REQ-6

Use Cases Not Implemented:

Note: The motion sensor can be turned off to accommodate a guest not wanting it to be functional during their stay because they are in and out of the room frequently.

Identifier	Description	REQ-# Used
UC-*1	“Comfort Mode Settings” - Allow customers to pick arrival time, input custom comfort mode settings, or turn off comfort mode/motion sensor.	REQ-*1, REQ-*2, REQ-*3, REQ-*4, REQ-*5, REQ-29, REQ-30

4.4.4 Use Case Diagram

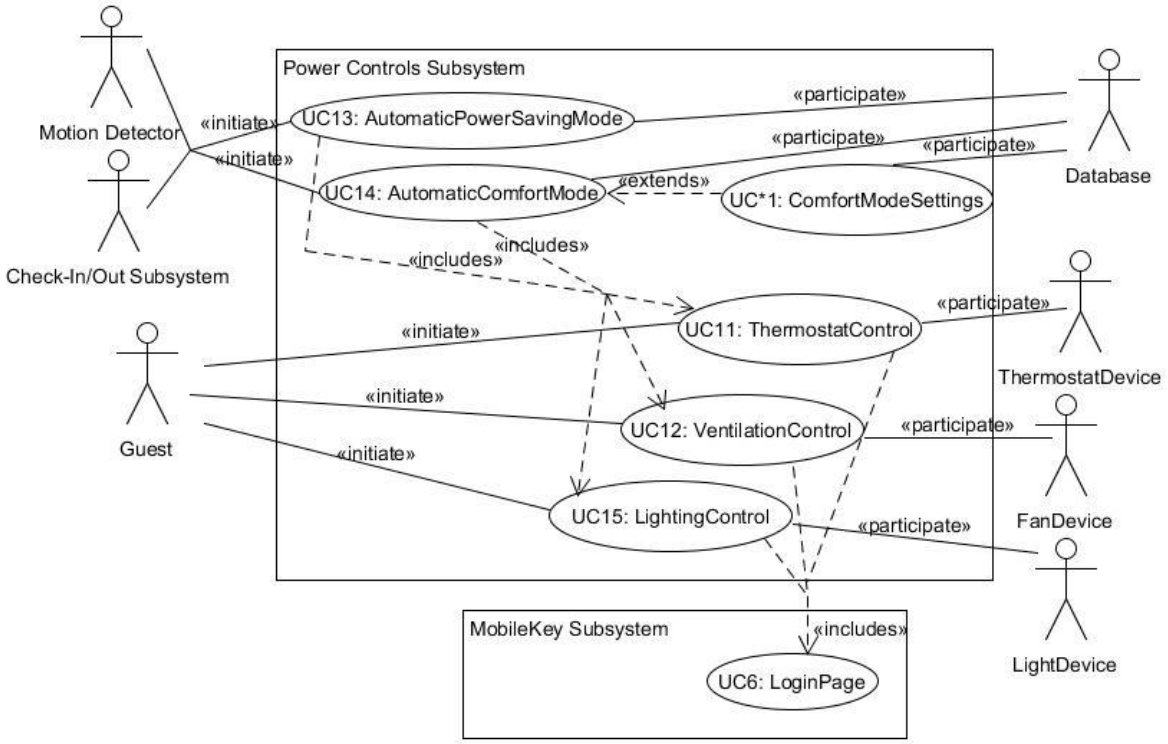


Figure 4.1: Use Case Diagram

The use case diagram above is for the power controls subsystem. On the top "row", a signal is initiated from the check in/out subsystem or the motion detector that triggers the beginning of UC13 and UC14, the automatic power saving and comfort modes. These include UC11, UC12, and UC15 as subroutines which help carry out the functions of power saving/comfort mode. For example, for the automatic power saving mode it has to use the thermostat, ventilation, and light controls to achieve that power saving mode. The database is a participating actor for all use cases, however, for sake of clarity it has been omitted from the subroutine use cases. The database will store information such as the state change of a device, or if power saving mode was entered, and at what time that occurred. The thermostat, ventilation, and lighting controls can also be separately initiated by the customer, to personally activate/disable them. UC*1 (ComfortModeSettings) extends the automatic comfort mode settings by allowing the user to adjust the settings of their comfort mode, if they choose to do so. Without this use case, UC14 is still complete.

4.4.5 Fully Dressed Descriptions

UC-13: Automatic Power Saving Mode
UC-14 functions similarly.
Related Requirements: REQ-3, REQ-4, REQ-5
Initiating Actors: Check-out Subsystem or Motion Sensor
Participating Actor: Thermostat Device, Light Device, Fan Device, Database
Actor Goal: The goal is to achieve room status of PowerSavingMode.
PreCondition: Customer has logged into system and selected "check-out"
PostCondition: Room is in power-saving mode.
Minimum Guarantee: The thermostat and lights in the room are shut off. The ventilation is turned on.

Success Guarantee: The thermostat and lights in the room are shut off and the ventilation turns off after a set time.
Flow Of Events for Success: 1. → The customer checks out of the room 2. ← The system turns on power-saving mode for the room. 3. ← The heat (thermostat device) is turned off for the room. 4. ← The light device is turned off for the room. 5. ← The fan device is turned on for the room. 6. ← The system logs the occurrence to the database. 7. ← The fan device turns off after 15 minutes to allow proper ventilation.
Alternative Flow Of Events: 1. → The motion sensor detects a customer leaving the room. 2. ← The system waits 10 minutes for another motion sensor detection. 3. ← If no detection occurs, the system turns on power-saving mode for the room. Otherwise, the flow of events ends with no action. 4. ← The heat (thermostat device) is turned off for the room. 5. ← The light device is turned off for the room. 6. ← The fan device is turned on for the room. 7. ← The system logs the occurrence to the database. 8. ← The fan device turns off after 15 minutes to allow proper ventilation.

UC-11: Thermostat Control
UC-12 and UC-15 function similarly.
Related Requirements: REQ-3, REQ- 4, REQ-6
Initiating Actors: Customer
Actor Goal: The goal is to have the temperature change in the room and update the UI.
Participating Actors: Database, Thermostat Device
PreCondition: Customer has logged into system and selected “Room Control” page.
PostCondition: Temperature has successfully changed.
Success Guarantee: The temperature of the room has changed and it is reflected on the UI.
Flow Of Events for Success: 1. → The customer views the current temperature.

2. → The customer inputs a new temperature.
3. ← The system reads the inputted temperature.
4. ← The system changes the temperature on the thermostat device.
5. ← The system updates the UI with the new temperature.
6. ← The system logs the occurrence to the database.

Alternative Flow Of Events:

1. → The customer views the current temperature.
2. → The customer inputs a new temperature.
3. ← The system rejects the update as the input is not within constraints.
4. → The customer inputs a new temperature within the constraints.
5. ← The system reads the inputted temperature.
6. ← The system changes the temperature on the thermostat device.
7. ← The system updates the UI with the new temperature.
8. ← The system logs the occurrence to the database.

UC-*1: Comfort Mode Settings
Related Requirements: REQ-*1, REQ-*2, REQ-*3, REQ-*4, REQ-*5, REQ-29, REQ-30
Initiating Actors: Customer
Actor Goal: The goal is for the customer to adjust their comfort mode settings, if they wish to do so.
Participating Actors: Motion Sensor, Database
PreCondition: Customer has logged into system and selected “Room Control” page.
PostCondition: The customer’s comfort mode settings are updated and activated.
Success Guarantee: The system turns on comfort mode.
Flow Of Events for Success: 1. → The customer inputs comfort mode temperature. 2. → The customer inputs comfort mode lighting. 3. → The customer inputs future arrival time. 4. ← The system logs the preferences in the database. 5. ← The system will use these preferences for future arrivals.
Alternative Flow Of Events: 1. → The customer selects off for comfort mode.

2. ← The system will turn the motion sensor off.
3. ← The system logs the preferences in the database.
4. ← The system will no longer automatically enter comfort nor power saving mode during the guest's stay.

4.4.6 System Sequence Diagram

Report 1 UC-13:

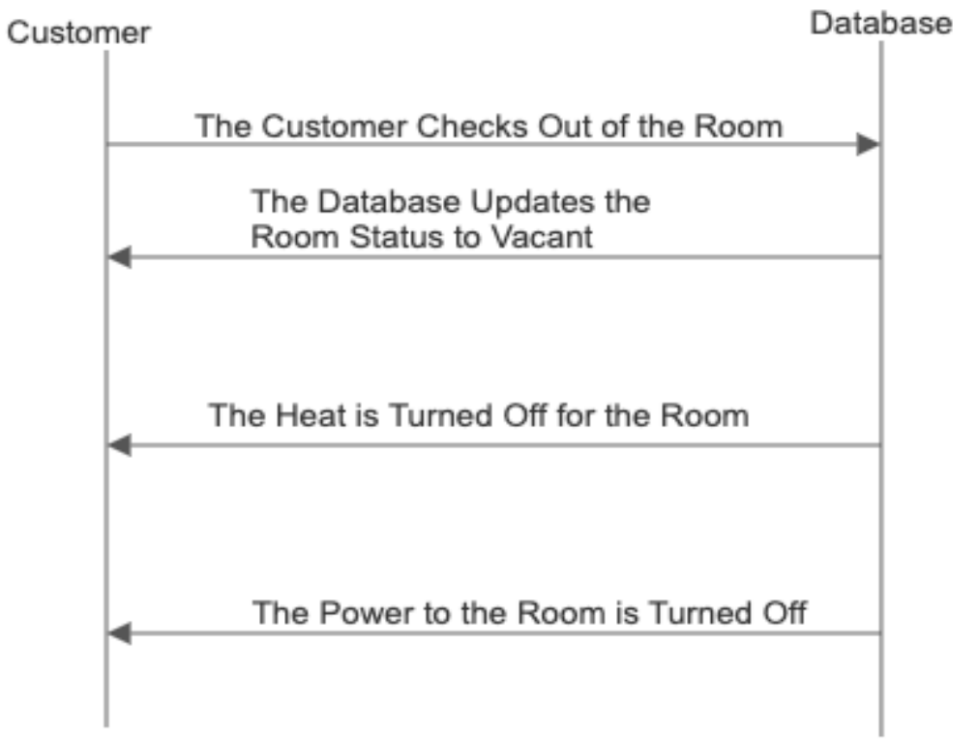


Figure 4.2: Old UC-13 System Sequence Diagram

Updated UC-13 (Automatic Power Saving Mode):

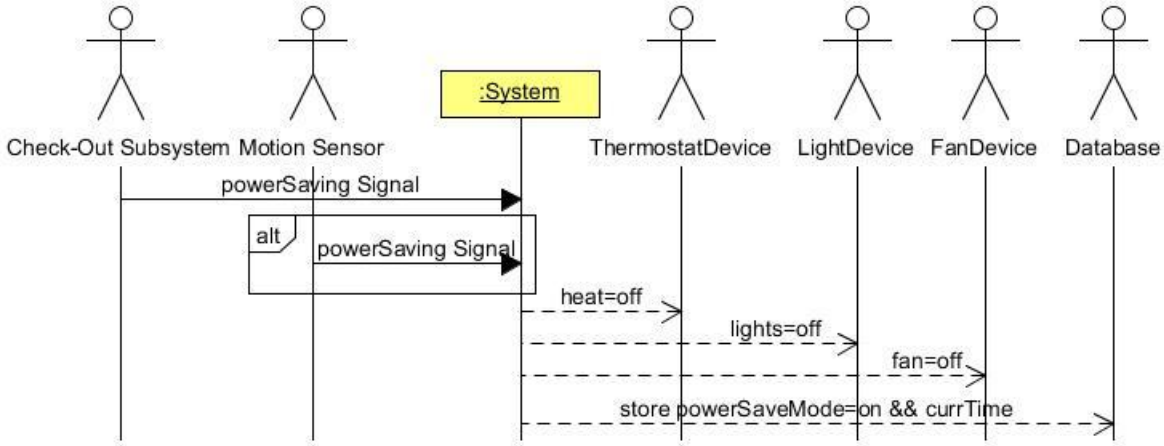


Figure 4.3: UC-13 System Sequence Diagram

Above is the system sequence diagram for the power controls subsystem. It details the sequence of UC-13 (automatic power saving mode) goes through. The checkout subsystem or the motion sensor sends a signal to the system which tells it to begin the power saving mode. The system then instructs the several devices to turn off. After this is complete, the system will store the state change in the database and the time at which it occurred.

UC-11 (Thermostat Control):

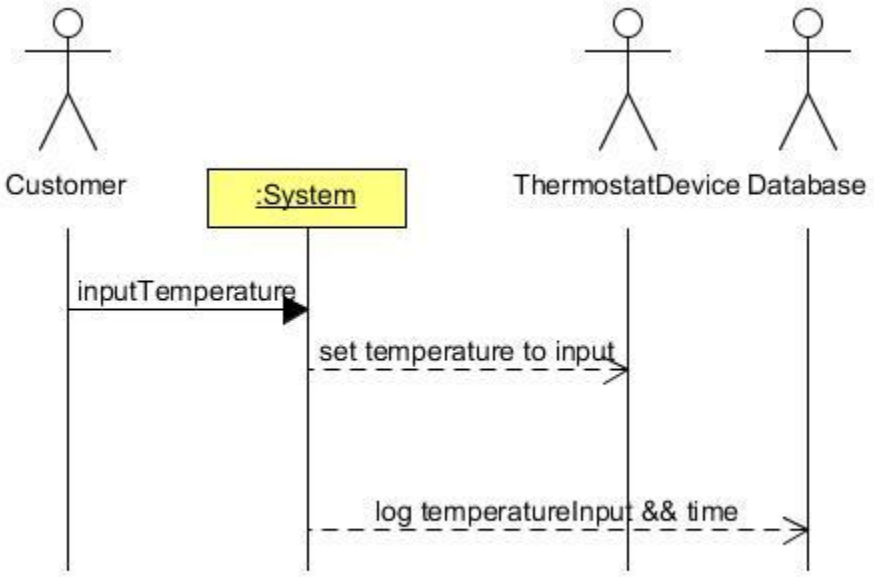


Figure 4.4: UC-11 System Sequence Diagram

UC-11 begins with the customer as an initiating actor inputting some kind of temperature into the interface part of the system. The system tells the thermostat device to change the temperature to the value inputted by the user. The system then completes by logging the temperature input and the time at which it occurred into the database.

UC-*1 (Comfort Mode Settings):

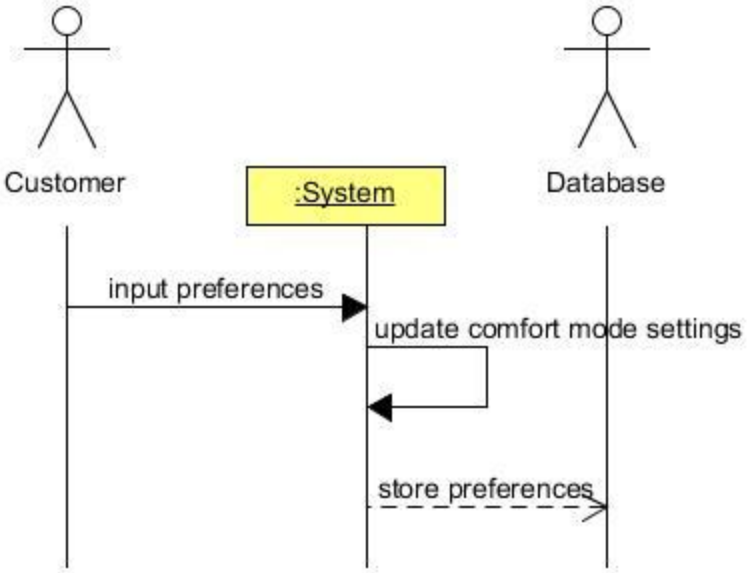


Figure 4.5: UC-*1 System Sequence Diagram

UC-*1 begins with the customer as an initiating actor inputting their comfort mode preferences into the interface part of the system. The system updates itself to use those new settings. The system then completes by logging the preferences into the database.

4.4.7 Traceability Matrix

UC #	REQ-3	REQ-4	REQ-5	REQ-6	REQ-7	REQ-8	REQ-29	REQ-30	REQ-*1	REQ-*2	REQ-*3	REQ-*4
UC-11					X				X	X		
UC-12						X					X	
UC-13	X	X	X									
UC-14							X	X				
UC-15				X								X
UC-*1							X	X	X	X	X	X

4.5 System Architecture and System Design

4.5.1 Identifying Subsystems

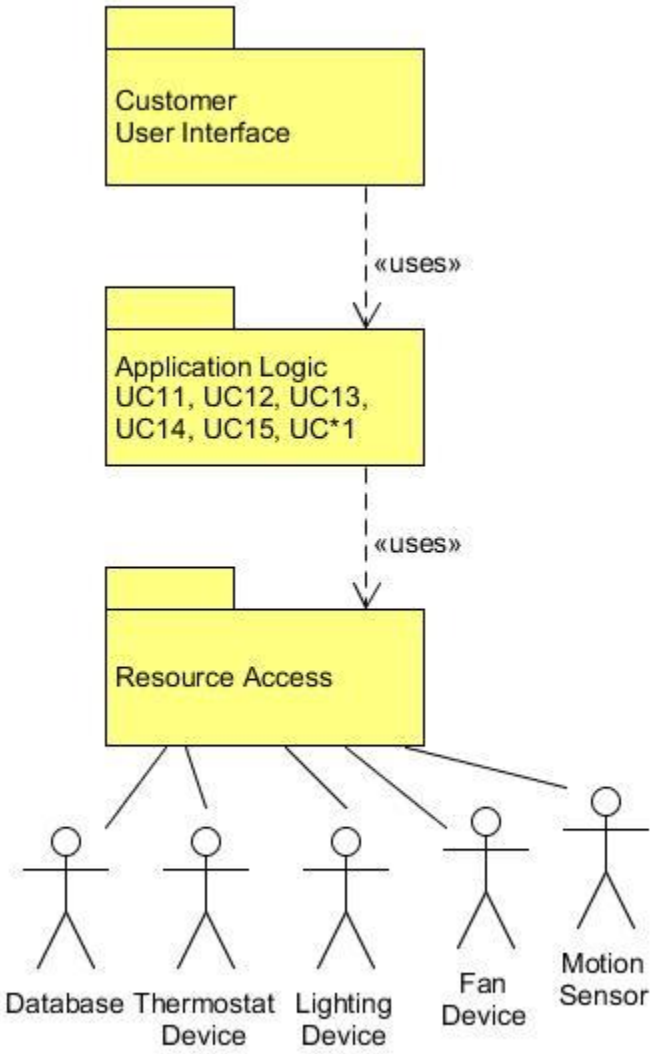


Figure 4.6: Architecture Package Diagram

The power controls subsystem is designed to be a layered architecture. The customer interacts with the user interface to be able to control the devices in the room. The user interface package uses the application logic package. This is where the requests from the user interface are received and where the business logic is held. The automatic power saving mode is processed in the application logic layer. The application logic uses the resource access layer which is a generalized layer to interact with various participating actors. Here, the system stores information to the database as well as interacting directly with the various devices in the room.

4.5.2 Architecture Styles

The first main architecture style used is the Layered architecture, more specifically, the three-tier architecture. It encompasses a presentation layer, an application layer, and a resource access layer. This allows Cloud Surf Inn to have more resilience and adaptability since each tier can be modified and upgraded separately. It also allows for more security, the application layer and the resource access layer contain access to various hotel controls which are separated by a boundary from the presentation layer. Additionally, it allows for greater stability; for example, a change in the presentation layer is much less likely to corrupt information in the resource access layer. The presentation layer has to do with the user interface that interacts with the customers. The application layer is a back-end part of the system, which takes requests from the presentation layer and matches them with the information required from the resource access layer. It also has the responsibility of performing the automatic features of the system, such as the power-saving mode.

The power controls subsystem also utilizes the Client-server model. The clients consist of the customers who must access the server to reach the controls for their rooms. There also needs to be a dedicated server in order to run the business logic behind the automatic power saving and comfort modes. The client-server model allows several guests to connect to the application at one time, in order for multiple users to access the devices in their room.

4.5.3 Mapping Subsystems to Hardware

The user interface layer runs on the client's computer. This allows for a simple, lightweight application that any guest can run and allows for multiple guests to access the application layer at one time. The application and resource layer are on the server computer to handle requests from customers as well as manage the interactions to the database and physical hardware components, as described next.

When the Raspberry Pi receives power control instructions from the user interface, our custom circuit board simulates what power control settings are active and inactive. The custom perforated board was custom soldered to fit our specific needs. On the board are 2 8 pin out rails that are connected to the Raspberry Pi Pin 2 and 5. Pin 2 is 5V power, and pin 5 is Ground. These rails provide Ground and 5V for all the devices attached to the rails. Our ventilation system is emulated by MakerFocus 2pcs Raspberry Pi Fans. The fans are powered by the 5V rail. The negative terminal of the fans are soldered to 2 NPN Transistors which are connected to GPIO Pin 27 and 22 of the Raspberry Pi. GPIO Pin 27 and GPIO Pin 22 are HIGH/HIGH for high ventilation, HIGH/LOW for low ventilation, and LOW/LOW for ventilation off. There is also a ventilation indicator RGB led that we have installed to show which ventilation mode we are in. The RGB indicator led is a common cathode therefore one pin is soldered to the ground rail. The other pins are soldered to wires that connect to GPIO Pin 23(Red), GPIO Pin 24(Green), and GPIO Pin(25)(Blue). The configurations for LOW/HIGH are in the order RGB. For high ventilation RGB is LOW/HIGH/LOW, low ventilation is LOW/LOW/HIGH, and ventilation off is HIGH/LOW/LOW. The lighting system is emulated with a single LED that is soldered and connected to GPIO pin 17, lights on correspond to pin 17 being HIGH and off correspond to pin 17 being LOW. The motion detection system is a single yellow LED that is soldered and connected to GPIO pin 16. Motion detected corresponds to pin 16 being HIGH, and no motion corresponds to pin 16 being LOW. Lastly the temperature system consists of a DHT Temperature sensor that has 3 pins. Positive pin is soldered to the 5V power rail, Negative pin is soldered to the Ground rail, and O pin is connected to GPIO pin 4. Pin 4 reads data from the sensor when it is requested to do so, i.e. when a customer selects display temp.

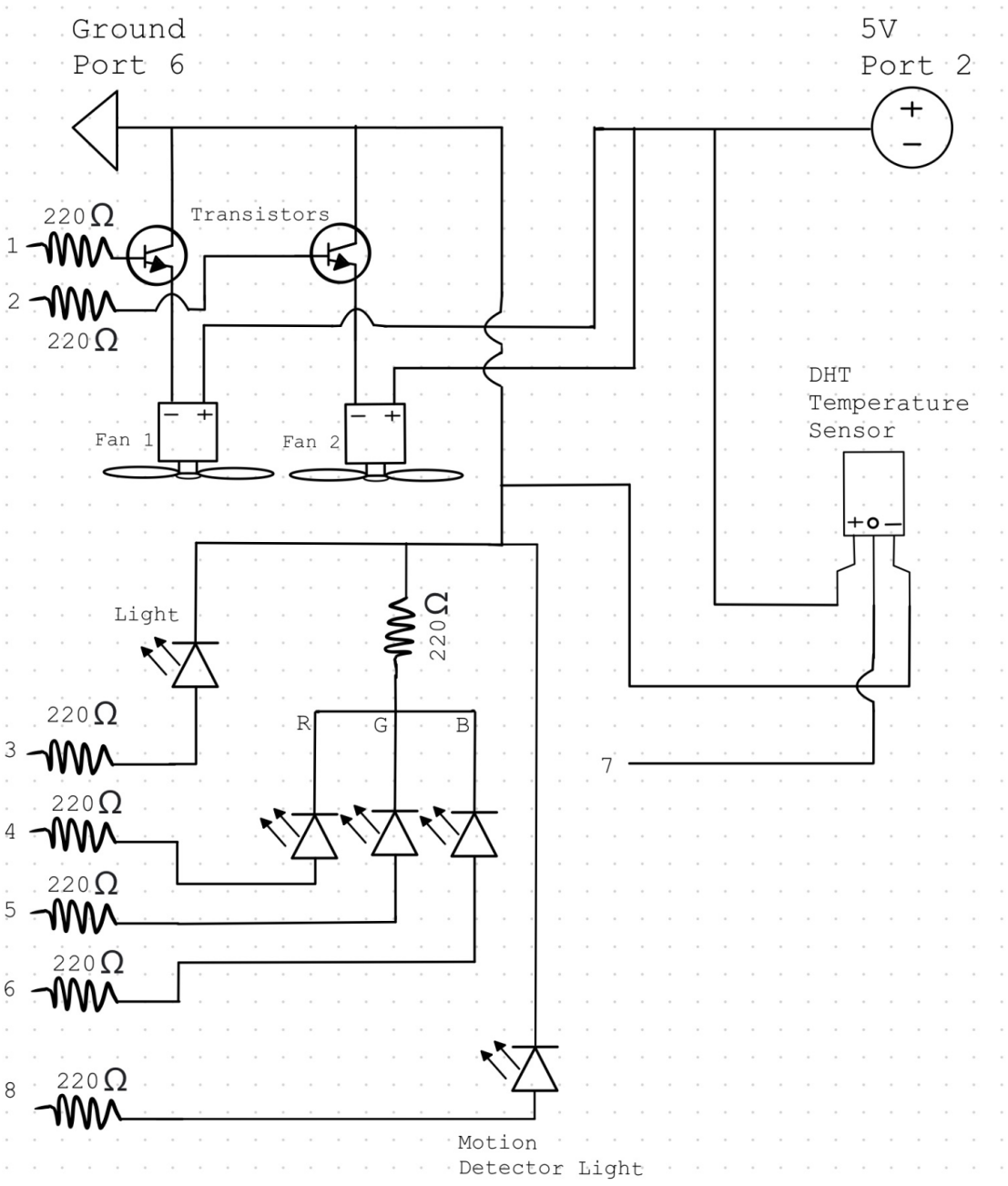


Figure 4.7: Hardware Schematic
This picture shows the hardware schematic for our custom power control board.

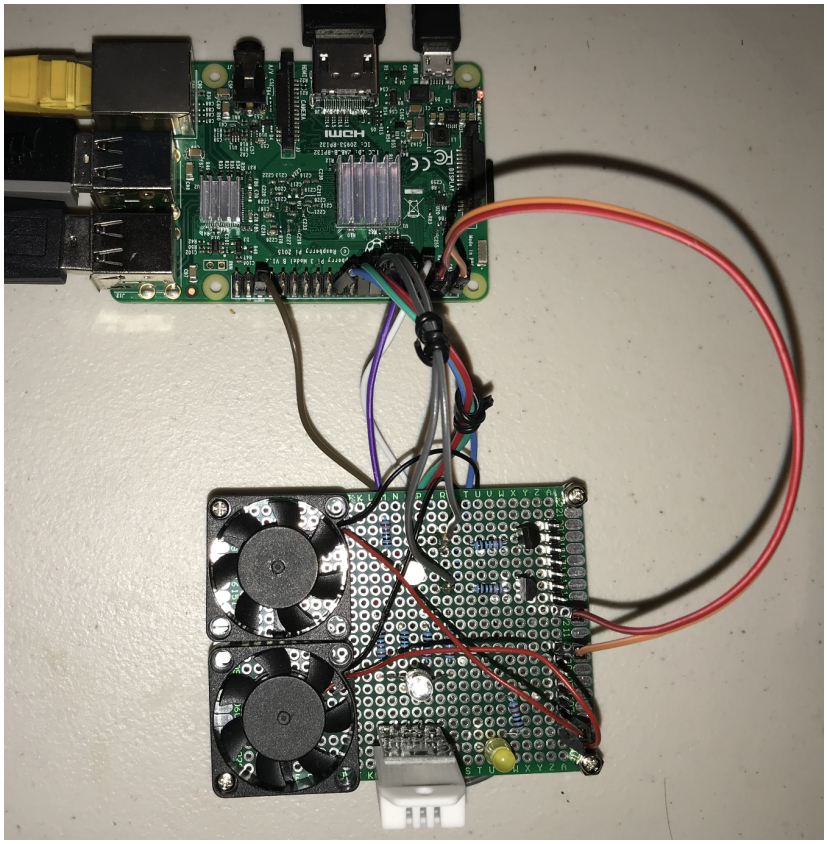


Figure 4.8: Raspberry Pi and Hardware

This figure shows the Raspberry Pi's connection to the physical hardware devices.

4.5.4 Connectors and Network Protocols

The Raspberry Pi that powers our custom power control board uses TCP Sockets using a multithreaded client/server connection, and Ethernet to communicate with our application. The Raspberry Pi must know the IPv4 of the computer that the application is running on. Both the computer that the application is on and the Raspbery Pi must use the same port number. Once a user checks into their room, a TCP message is sent from the main application to the Raspberry Pi indicating that a certain room is booked. The Raspberry Pi then configures the proper settings to the power control board for Comfort Mode. The Raspberry Pi is always listening for the next TCP message, moreover since some of our features run on timers, we spawn additional threads to handle simultaneous requests from the application whenever we need to implement timers. The only TCP messages that the Raspberry Pi handles are, "Booked", "Vacant", "FanOff",

"FanOnHigh", "FanOnLow", "LightOn", "LightOff", "Comfort Mode", "SetTemp", "DisplayTemp", "Leave", and "Arrive". All other messages are ignored and the Raspberry Pi continues listening for more incoming messages. Once the customer checks out of their room, we we intitate power saving mode. We then close the socket and wait to connect to a new customer on the original port number plus one.

4.5.5 Global Control Flow

The power controls subsystem can be considered partly procedure-driven and partly event-driven. Use cases 11, 12, and 15 can be considered largely procedure-driven. These use cases relate to the customer having remote control to the devices of their room, such as the thermostat, fans, and lights. As such, the user will go through the same steps each time they would like to interact with the devices. For example, for the thermostat, the guest will always have to input and select a temperature to change the thermostat. This will then trigger the thermostat to update upon receiving the new information. The automated parts of this subsystem, the power saving and comfort modes, are event-driven. This part of the system will wait for a customer to check-in or out, or, to enter or leave the room to initiate the power saving or comfort modes, respectively. This part of the system's responsibility is to detect these events and perform the corresponding automated tasks. Thus, it is largely event-driven.

The power controls subsystem is largely of the event-response type. There are two timers: one relating to a guest leaving the room, and one concerns the ventilation device. The first timer counts for about 10 to 15 minutes upon a customer leaving the room, before initiating the power saving mode. This is done to avoid a situation where a customer leaves the room briefly only to find that the power saving mode has already been entered. The second timer is for how long the ventilation system will remain activated upon entering power saving mode. The fan timer will run for about 15 minutes to ventilate the room before signaling the fan to turn off. The rest of the system relies on event-responses. The automated controls rely on events relating to the customer entering or leaving, and the device controls do not have a concern for real time.

4.5.6 Hardware Requirements

The temperature sensor components rely on a Raspberry Pi 3 Model B, which will need an 8 GB micro sd card, a micro USB power source, and a reliable internet connection. The sensors/additional hardware used will include 2pcs DHT22 Temperature Humidity Sensor, MakerFocus 2pcs Raspberry Pi Fans, RGB led, 2 leds, 8pcs 220 Ohm Resistors, 2pcs NPN Transistor, 2pcs 8-pin header, and a Perforated Circuit Board.

The user interface is run on a computer needing a minimum of 512 MB of memory, 300 MB of storage/disk space, and a minimum processor clock speed of 800 MHz.

4.6 Domain Analysis

4.6.1 Concept Definitions

Concept	Responsibility	Type
Interface	A graphical user interface where the user is shown what actions can be taken and select to have them done.	Boundary
Controller	Handles and coordinates the actions between the interface and the various use cases that must be completed and delegates work to other concepts.	Control
DatabaseConnection	Coordinate interactions and data exchanges with the database.	Boundary
PowerControl	Serves as an intermediary between the controller and the various devices, deciding which devices to be activated depending on input.	Service
ComfortPreferences	Consists of the unique comfort mode preferences of a room such as the desired temperature and at what time to enable comfort mode.	Entity
LightControl	Controls lights on/off.	Boundary
FanControl	Controls ventilation on/off.	Boundary
ThermostatControl	Controls heat/cool on/off and reads current temperature data.	Boundary
ControllerSignal	Sends a signal to the controller indicating a check-in/out or	Boundary

	that the motion detector detected a customer entering/leaving that activates comfort/power saving mode.	

4.6.2 Association Definitions

Concept Pair	Association Description	Type
Controller ↔ Power Control	Controller passes appropriate requests to power control.	Conveys Requests
Power Control ↔ Comfort Preferences	Power control obtains the unique comfort preferences associated with a room when initiating comfort mode.	Obtains
Database Connection ↔ Comfort Preferences	The database connection provides comfort preferences with the unique settings associated with the room.	Provides Data
Power Control ↔ Light Control	Power control instructs light control on turning on/off.	Conveys Requests
Power Control ↔ Fan Control	Power control instructs fan control on turning on/off.	Conveys Requests
Power Control ↔ Thermostat Control	Power control instructs thermostat control on turning on/off and to read temperature data.	Conveys Requests
Power Control ↔ Database Connection	Power control saves the update states of the devices and the time at which they were updated.	Requests Save

4.6.3 Attribute Definitions

Concept	Attributes	Description
Power Control	Device Status	Used to store information about the device's status upon completion of a request to change some device's parameters.
	Time	Used to store the time at which a change was completed.
Fan Control	Fan Speed	Consists of the high, low, and off settings at which the ventilation/fan can be run.
	Timer	A timer starts when power saving mode is entered. When

		the timer expires, the fan turns off.
Thermostat Control	Set Temperature	Sets the temperature of the thermostat device.
	View Temperature	Asks for a live reading of the current temperature.
Comfort Preferences	Room Number	The room number in which a set of comfort preferences is associated with.
	Operation Mode	The operation mode selected for the comfort mode, it can be automatic (with the motion sensor), on a schedule (the user inputs a time at which they arrive/leave the room), or completely off (always in comfort mode).
	Comfort Settings	Comfort settings the user has set for their room. Consists of things like the temperature or the light setting.

4.6.4 Traceability Matrix

Concept:	UC-11 (Thermostat)	UC-12 (Power-Saving)	UC-13 (Comfort)	UC-14 (Fan)	UC-15 (Lights)	UC-*1 (Comfort Settings)
Interface	x			x	x	x
Controller	x	x	x	x	x	x
Database Connection		x				x
Power Control		x	x			x
Light Control		x	x		x	
Fan Control		x	x	x		
Thermostat Control	x	x	x			
Controller Signal		x	x			

4.6.5 Domain Model

Domain Model Report 2:

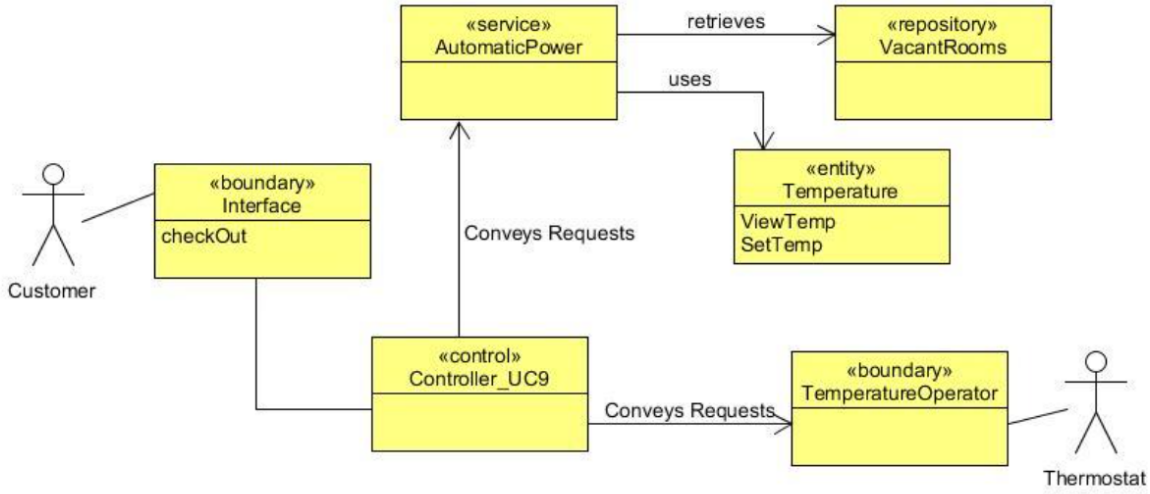


Figure 4.9: UC-13 System Sequence Diagram

Above is the system sequence diagram for the power controls subsystem. It details the sequence of UC-13 (automatic power saving mode) goes through. The checkout subsystem or the motion sensor sends a signal to the system which tells it to begin the power saving mode. The system then instructs the several devices to turn off. After this is complete, the system will store the state change in the database and the time at which it occurred.

Updated Domain Model:

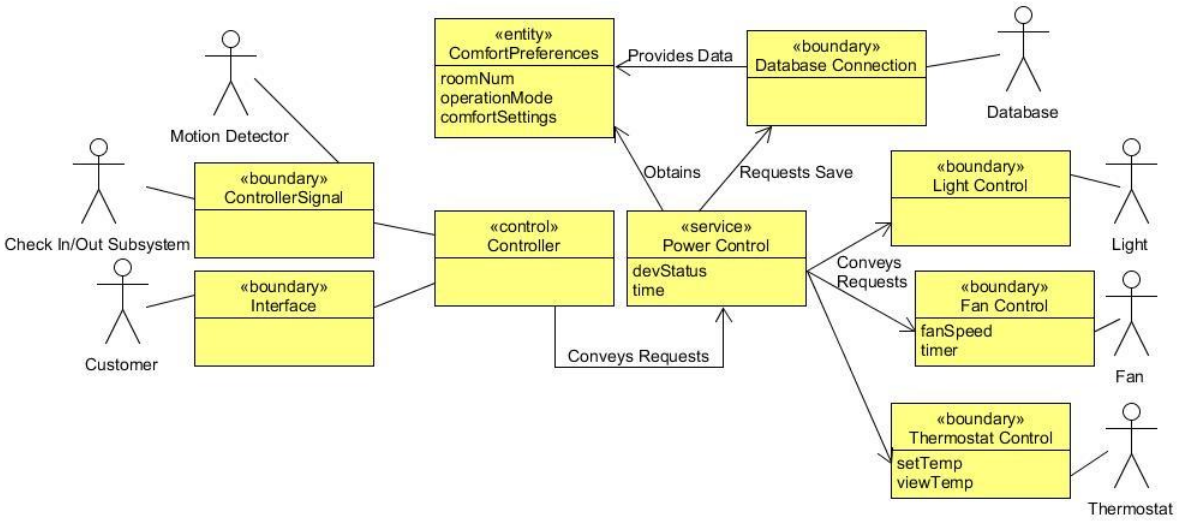


Figure 4.10: Domain Model

The domain model is the culmination of the concepts, attributes, and attribute associations. There are three main actors, the check in and out subsystem, the motion detector, and the customer. These serve as the boundaries, where the customer interacts with the interface. These both go to the controller which must designate the different responsibilities to the power control service. The power control service has devStatus and time in its properties which are stored for the various devices it communicates with for storing into the database. On the right hand side we have each device control interacting with its own acting device, with the thermostat control having some additional responsibilities such as setting the temperature and viewing the temperature. The fan control also has a fanSpeed attribute that describes the different states the fan can be in. The power control also obtains the unique comfort preferences associated with a room when instructed to activate comfort mode. This data is provided to the comfort preferences entity from the database connection. Comfort preferences include like the room number they are associated with, the operation mode of the room, and the specific comfort settings, such as the temperature of the room.

4.6.6 System Operation Contracts

Operation	ThermostatCtrl
Reference:	UC-11
Preconditions:	The user is logged in and has accessed the room control page on the UI.
Postconditions:	The thermostat temperature changed in the system and UI.

Operation	FanCtrl
Reference:	UC-12
Preconditions:	The user is logged in and has accessed the room control page on the UI. Fans in the room are off.
Postconditions:	Fans in the room turn on.

Operation	PowerCtrl
Reference:	UC-13
Preconditions:	Customer(s) have changed status to “check out.”
Postconditions:	Thermostat and power in the room is shut off. Ventilation is turned on.

Operation	PowerCtrl
Reference:	UC-14
Preconditions:	Customer(s) have changed status to “check in.”
Postconditions:	Power in the room is turned on, lights and thermostat.

Operation	LightCtrl
Reference:	UC-15
Preconditions:	The user is logged in and has accessed the room control page on the UI.
Postconditions:	Lights in the room change state (turn on/off).

Operation	comfortPreference
Reference:	UC-*1

Preconditions:	The user is logged in and has accessed the room control page on the UI.
Postconditions:	Automatic comfort mode is updated with user preferences.

4.6.7 Data Model & Persistent Data Storage

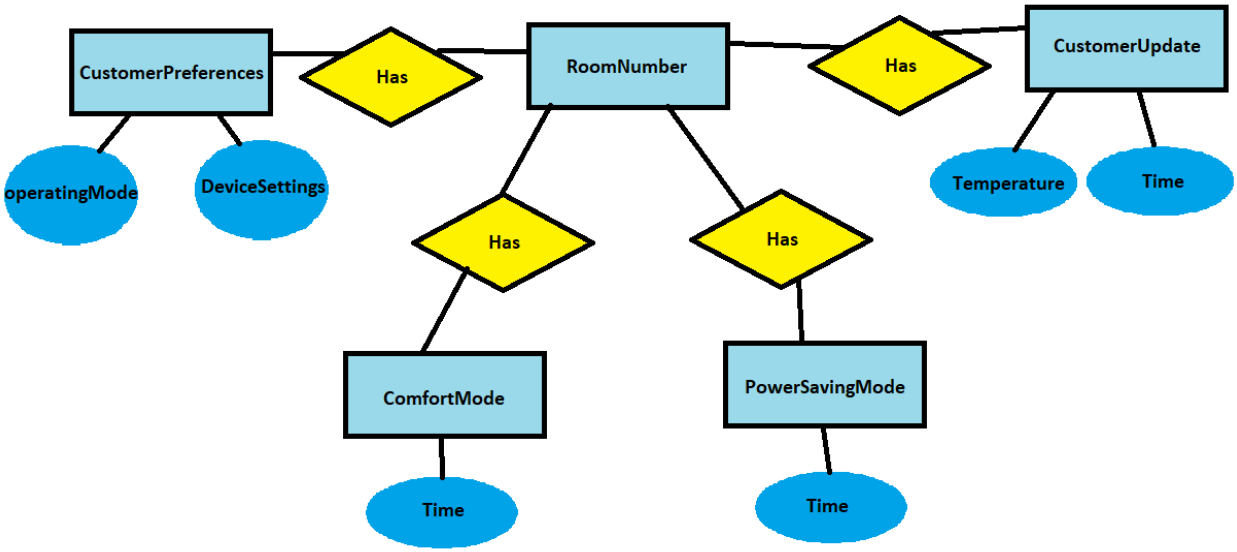


Figure 4.11: Data Model

Our subsystem is integrated with the applications database in that we log important information related to the power controls and the room. These records can be used to reference specific room states from the past. When the customer first books a room we log "Comfort Mode" along with the time at which setting was logged, "CM: year-month-dayUTCinfo", when the customer selects checkout we log the setting "Power Saving Mode" along with date and time information, "PSM: year-month-dayUTCinfo". When a customer inputs a desired temperature and selects update the temperature is logged along with the date and time information. Temperature is stored along with "Customer Update", "Customer Update: year-month-dayUTCinfo". A theoretical data model for customer preference information is included as well.

4.6.8 Mathematical Model

No mathematical models are utilized.

4.7 Interaction Diagrams

Old UC-13 Interaction Diagram Report 2:

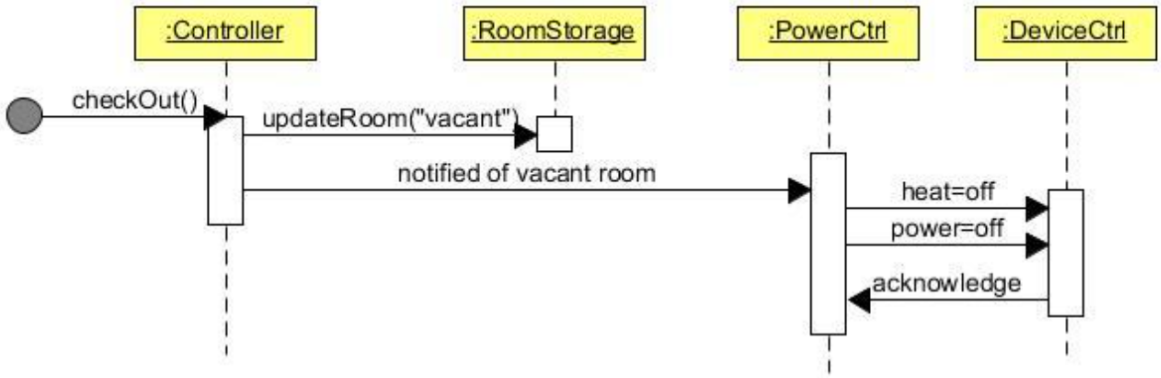


Figure 4.12: Old UC-13 Interaction Diagram

Updated UC-13 Interaction Diagram (Automatic Power Saving Mode):

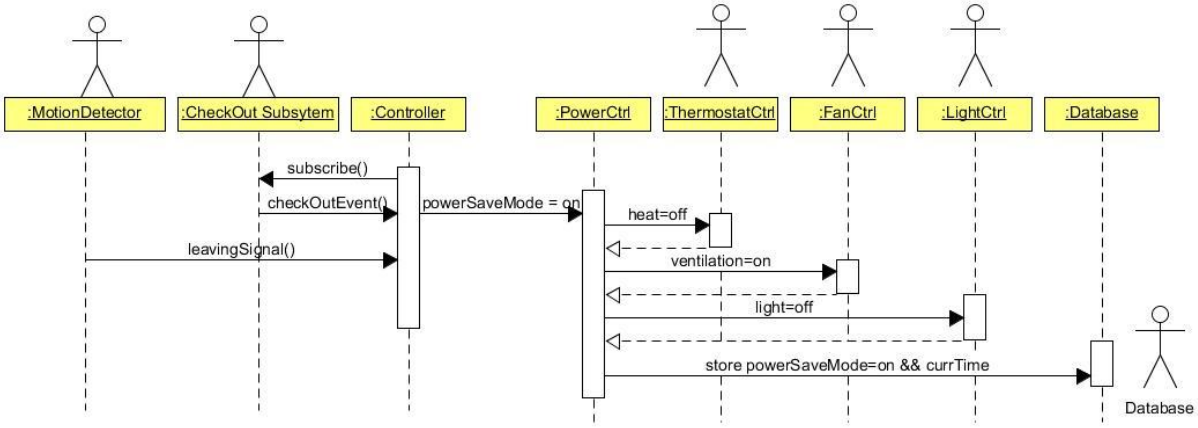


Figure 4.13: UC-13 Interaction Diagram

The interaction diagram is a direct development from the system sequence diagram for UC13. The publisher-subscriber pattern has been implemented in this stage, where the controller is a subscriber to the events of the checkOut subsystem and the motion detector. This is so that if more events are added where the power controls might need to be activated, it is much easier for future implementations to simply subscribe to that system. When a checkoutEvent() is sent from the check-out subsystem, or a leavingSignal() is sent from the motion detector, it tells power control to turn powerSavingMode on. Increased device specialization has been implemented, with a separate control for the thermostat, fan, and lights. The power control sends a message to each device controller to turn the heat=off, turn the ventilation on, and turn the light off. The thermostat control, fan control, and light control each interact with their corresponding participating actors, as shown by the actor symbol above those classes. After that is complete, the power control will log the power saving mode occurrence and the time at which it happened into the database.

UC-11 (Thermostat Control):

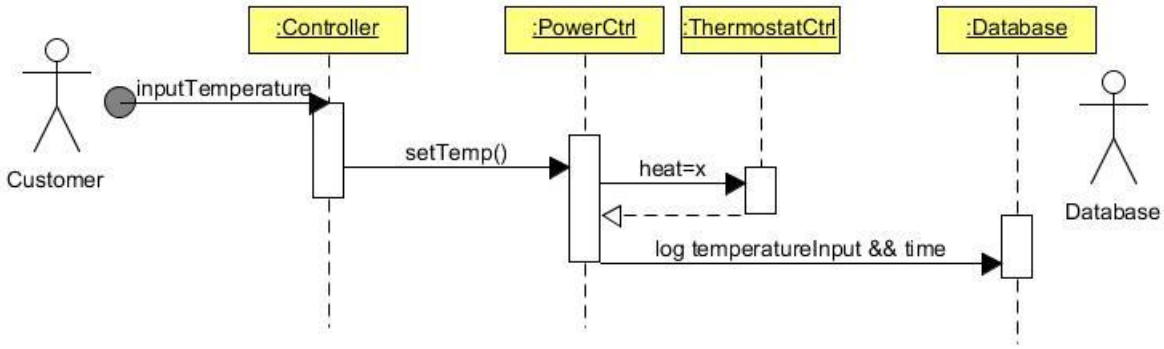


Figure 4.14: UC-11 Interaction Diagram

The interaction diagram for UC11 is a direct development from the system sequence diagram developed before. Here the customer is an initiating actor from the user interface, and will tell the controller that the guest would like to set the temperature to the thermostat. The controller conveys this request to the power control which will contact the thermostat control to set the temperature. The thermostat control interacts with the thermostat device participating actor. Which is omitted from the diagram for clarity. Once this occurs, the power controller will store this temperature input and the time at which it occurred into the database.

UC-*1 (Comfort Mode Settings):

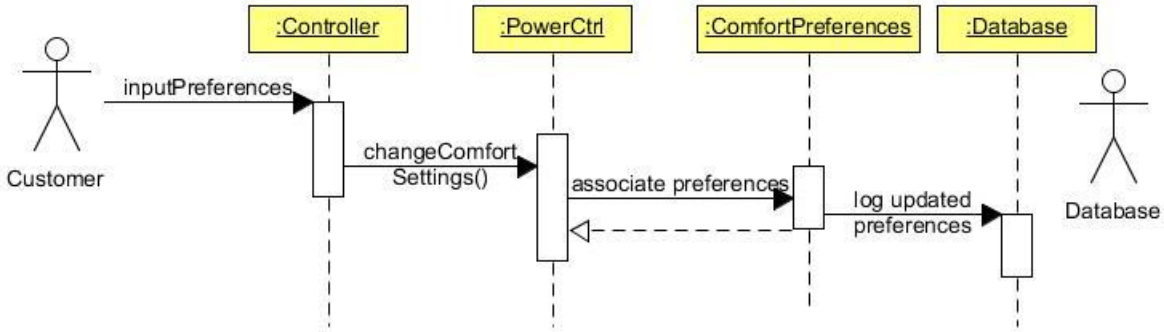


Figure 4.15: UC-*1 Interaction Diagram

The interaction diagram for UC*1 is a direct development from the system sequence diagram developed before. Here the customer is an initiating actor from the user interface, and will tell the controller of their comfort preferences. The controller conveys this request to the power control which will contact comfort preferences to store the information and associate it with a room number. The comfort preferences are now configured for that customer.

4.8 Class Diagram

Class	Meaning/Use of Class	Associations
Controller	This main controller receives signals from outside subsystems. For example, for UC13 (automatic power saving mode) it would receive a signal from the checkOut subsystem which initiates the power saving mode. It relays this information to the power control, instructing it to enable various modes. For example, a power saving mode=on sent would instruct the power control to turn the thermostat heat off, fan ventilation on, and light off.	PowerCtrl
PowerCtrl	The power control object is the facilitator between the main controller and the actual device controls themselves. It has device numbers as variables that keep track of the various devices, and also has functions to activate and deactivate certain devices. For example, an activate() call placed to the light control would turn it on.	ThermostatCtrl, FanCtrl, LightCtrl
ComfortPreferences	The comfort preferences object holds the user's customized preferences for comfort mode.The operating mode first specifies which mode in which the comfort mode of the system is operating. Automatically, based on a schedule, or off completely. The arrival time and exit time are integers used for the scheduling mode. And comfortSettings is an object that stores the user's preferences while in comfort mode, such as the temperature they set the room to, or whether they want the fan on or off when they enter.	PowerCtrl
ThermostatCtrl	This object interacts directly with the devices, such as the temperature reader, and therefore is in its own class so it can focus on connecting to devices. It has direct control over the devices and instructs them to set or read temperatures.	N/A
FanCtrl	This object interacts directly with the fan device, setting it to various speeds: high, low, and off. The timer is used to automatically switch the fan off after a certain time has occurred.	N/A
LightCtrl	This object interacts directly with the light device, connecting to it and turning it off.	N/A

Class Diagram Report 2:

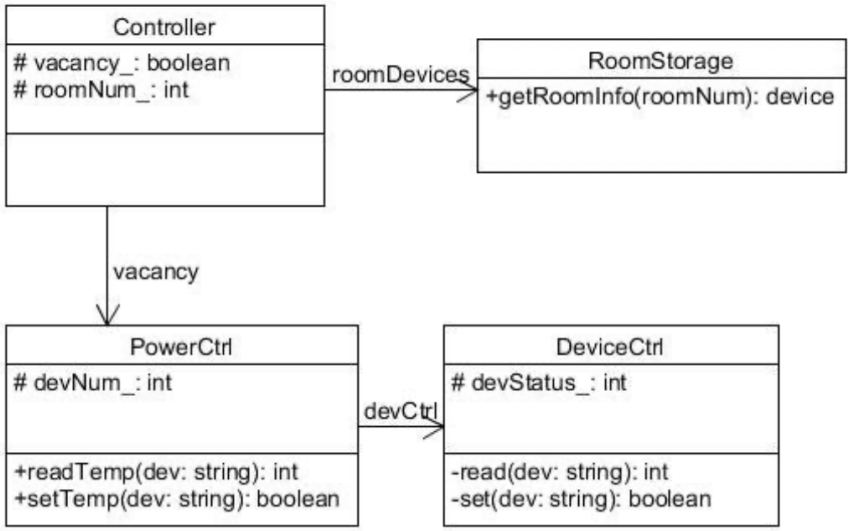


Figure 4.16: Previous Class Diagram

Updated Class Diagram:

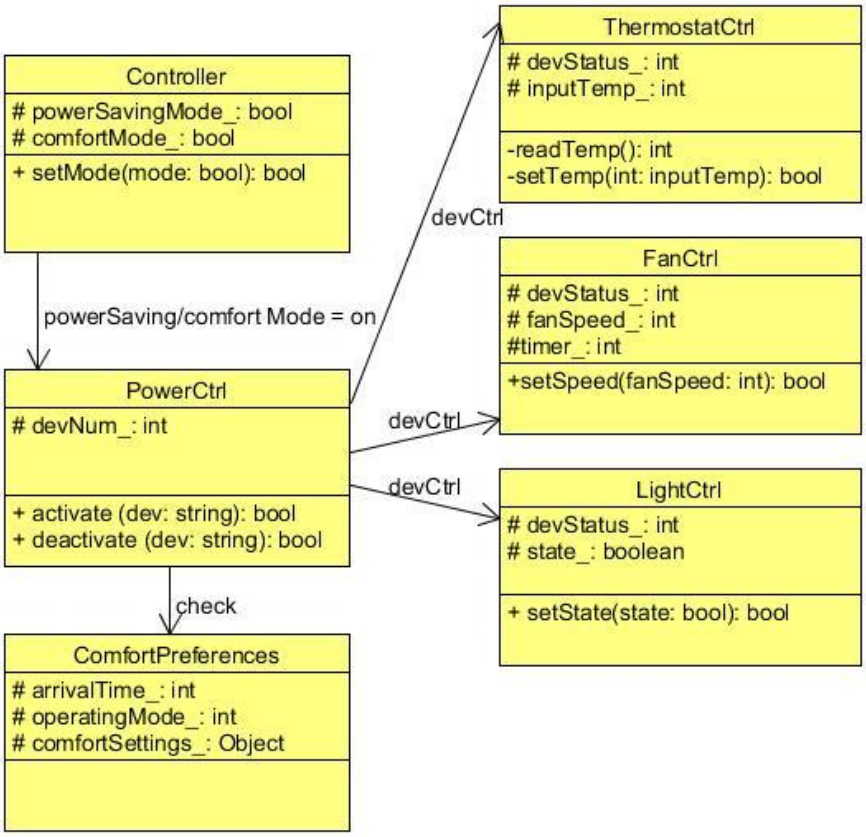

Controller
powerSavingMode_: bool
comfortMode_: bool
+ setMode(mode: bool): bool
powerSaving/comfort Mode = on
PowerCtrl
devNum_: int
+ activate (dev: string): bool
+ deactivate (dev: string): bool
check
ComfortPreferences
arrivalTime_: int
operatingMode_: int
comfortSettings_: Object
devCtrl
ThermostatCtrl
devStatus_: int
inputTemp_: int
-readTemp(): int
-setTemp(int: inputTemp): bool
devCtrl
FanCtrl
devStatus_: int
fanSpeed_: int
#timer_: int
+setSpeed(fanSpeed: int): bool
devCtrl
LightCtrl
devStatus_: int
state_: boolean
+ setState(state: bool): bool

Figure 4.17: Class Diagram

The updated class diagram is shown above. The controller is the main class, with attributes powerSavingMode and comfortMode which represent which state the system should be in. In includes a function to set the mode based on the incoming signals. This mode change is relayed to the power control class. The power control class contains device numbers for the different devices, as well as the controls to activate and deactivate different devices. The devices are connected to the thermostat control, fan control, and light control. These each have devStatuses associated with them to ensure that the devices are working properly. The thermostat control has an additional inputTemperature attribute which is there for the customer to change the device's temperature. It also has read temperature and set temperature functions to control the device. The fan control has a fan speed integer for the different modes, high, low, and off. And a set speed function to interact with the device. The timer is used to ensure that the fan only turns on for a set period for ventilating. The light control has a simple state boolean for on and off, and a set state function to interact with the device. The power control also checks the comfort preferences of the user to ensure that comfort mode is initiated in accord with the user's preferences. These attributes can include the arrival time (what time to start comfort mode) if they choose to schedule a time, or operating mode, which indicates whether its automatic, on scheduling mode, or completely off. The comfort settings have to do with the temperature the customer chooses for comfort, or if they want the ventilation on while they are in comfort mode.

4.9 Algorithms and Data Structures

4.9.1 Algorithms

There are no complex algorithms in this subsystem.

4.9.2 Data Structures

No complex data structures are used.

4.9.3 Concurrency

Our system spawns multiple threads when we need to time specific actions while still being able to handle multiple requests. When the customer first checks into the room we spawn a thread to set ventilation on a timer as per ventilation protocol while the main thread is waiting for incoming messages from the server. Furthermore when the motion detector has sensed that the

customer has left the room we spawn a thread to observe the amount of time the customer has been gone. This is done while the main thread is able to handle simultaneous requests. The same check-in threading protocol is done with check-out.

As for thread synchronization. No methods of thread synchronization were used because multiple threads were not manipulating critical variables or code.

4.10 Design of Tests

UC#	**Test Case Description**	**Test Coverage**
UC-11	A unit test is conducted on the ThermostatCtrl class by using an external heat source. readTemp will be evaluated by showing the current temperature, then using a heat source to increase the control room temperature. To pass, the temperature must increase at least 10 degrees after being exposed to a heat source. setTemp is also evaluated for error by inputting a temperature not within the constraints. The constraints shall be no greater than 80 degrees and no less than 65 degrees. To pass, the temperature of the rooms shall not change when outside of these bounds.	This test covers the requirements of the thermostat requested by the customer and also covers the constraints of attributes in the ThermostatCtrl class.
UC-12	A unit test is conducted on the FanCtrl class using state based testing. FanCtrl has 3 states, off, low, high. Assigning the default setting to fan = off then iterating through each state change. An LED is installed to confirm the state of the fans. When high both fans will be on and the LED will be green. When low one fan is on and the LED will be blue. When in the default setting (off) the LED will be red.	This test covers the constraints involving the FanCtrl class attributes. setSpeed can be one of three states: off, low, high.

UC-13	An integration test is conducted by signaling the controller class that the state of the system is powerSaving = on. This evaluates the composition of the system components, such as lighting control, thermostat control, and ventilation control. When signaling the initial class, the system should signal the component classes and bring them to their default setting on powerSave mode. To pass Lighting Control and Thermostat Control will be off. Ventilation Control will be high. The ventilation state will be evaluated with a timer as well. Setting a mock time of 15 seconds, the state of VentilationCtrl will go to its default state of off from high. To pass, the controls will all turn to an off state.	This test covers all the requirements of power saving mode and evaluates the integration of the system. It also covers the requirements of ventilation control using a timer.
UC-11	The thermostat controller will be asked to input a temperature of 74 degrees. In the thermostat controller and the interface, it must show that the set temperature is now 74 degrees and be overriding the read temperature data that is shown by default.	This test covers the set temperature feature that this class must perform. We ensure that the class is able to read the temperature from the sensor.
UC-13 UC-14	The power control class will take as input the parameter powerSavingMode as “on” in one test, and “off” as the other test. The output should enact the power saving mode when on, and comfort mode when off, i.e. turn off the heat, lights, and turn on ventilation in a vacant room, and turn it back on, i.e. heat, lights, and turn off ventilation in a non-vacant room.	This test covers the features in UC-13 of the power saving mode and comfort mode, ensuring that when the power control receives the signal to turn power saving mode on, it is sent from the controller and the power control responds appropriately as described before.

4.11 Effort Estimation using Use Case Points

Unadjusted Actor Weight (UAW):

Actor	Description	Complexity	Weight
Guest	Guests interact with the system via a graphical user interface.	Complex	3
Motion Sensor	The motion sensor interacts with our system through an API.	Simple	1
Thermostat Device	Thermostat communicates with the system via a network communication protocol (TCP).	Average	2
LightDevice	Light communicates with the system via a network communication protocol (TCP).	Average	2
FanDevice	Fan communicates with the system via a network communication protocol (TCP).	Average	2
Database	Database interacts with our system through a protocol.	Average	2

Unadjusted actor weight comes to 3+1+2*4=12

Unadjusted Use Case Points (UUCW):

Use Case	Description	Category	Weight
Thermostat Control (UC-11)	Moderate interface design. 2 participating actors (thermostat, database). Up to 4 steps.	Average	10
Ventilation Control (UC-12)	Simple user interface. 2 participating actors (fan, database). Up to 2 steps.	Average	10
Automatic Power Saving Mode (UC-13)	No user interface. 4 participating actors (thermostat, light, fan, database). No steps required.	Average	10
Automatic Comfort Mode (UC-14)	No user interface. 4 participating actors (thermostat, light, fan, database). No steps required.	Average	10
Lighting Control (UC-15)	Simple user interface. 2 participating actors (light, database). Up to 2 steps.	Simple	5
Comfort Mode	Moderate interface design. 1	Average	10

Settings (UC-*1)	participating actor. Up to 4 steps.		
		Use Case Weight Total	55

Unadjusted Use Case Weight = 5*10 + 5=55

Technical Complexity Factor (TCF):

Technical Factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight*Perceived Complexity)
T1	System is not distributed.	2	0	2*0=0
T2	Users expect good performance but no critical response times or throughput.	1	3	1*3=3
T3	Users expect high efficiency but it is not exceptionally critical.	1	3	1*3=3
T4	Internal processing is relatively simple.	1	1	1*1=1
T5	System should be able to be reused in other hotels.	1	5	1*5=5
T6	Software needs to be easily installed for different hotels.	0.5	5	0.5*5=2.5
T7	Ease of use is very important.	0.5	5	0.5*5=2.5
T8	No portability concerns.	2	0	2*0=0
T9	System should be able to handle new features but is unlikely to need to do so.	1	2	1*2=2
T10	Concurrent use is required.	1	4	1*4=4
T11	Security is a significant concern.	1	5	1*5=5
T12	No direct access for third parties.	1	0	1*0=0

T13	No unique training needs.	1	0	1*0=0
			Technical Factor Total:	28

Technical complexity factor comes to 0.6+0.01*28=0.88

Environmental Complexity Factor (ECF):

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor (Weight*Perceived Impact)
E1	Beginner familiarity with UML-based development.	1.5	1	1.5*1=1.5
E2	Little familiarity with application problems.	0.5	1	0.5*1=0.5
E3	Some knowledge of object-oriented approach.	1	2	1*2=2
E4	Beginner lead analyst.	0.5	1	0.5*1=0.5
E5	Motivated with some team members lacking.	1	4	1*4=4
E6	Stable requirements expected.	2	5	2*5=10
E7	No part-time staff.	-1	0	-1*0=0
E8	Programming language of average difficulty will be used.	-1	3	-1*3=-3
			Environmental Factor Total:	15.5

Environmental complexity factor comes to 1.4-0.03*15.5= 0.935

Use Case Points and Duration:

UCP = UUCP * TCF * ECF = (55 + 12) * 0.88 * 0.935 = 55.1276 = 55 use case points.

Duration = UCP * PF = 55 * 28 = 1,540

4.12 History, Current, & Future Work

Future Work:

There are many future applications of the work done here. For the current hotel alone, fully incorporating the motion detectors into the subsystem would be the first line of work. Even upgrading this to also include custom scheduling of when they plan to enter or leave the room, or a even a more advanced bluetooth proximity tracker to receive more accurate information on when the guest truly leaves or enters their room. Additionally, the idea of creating an even smarter hotel room is very appealing. Adding devices such as electronic shades and integrating virtual assistant support would further provide convenience and luxury for hotel guests. An Alexa or Google activated smart speaker could be placed in each room that the customer can use to manage any one of the connected devices in the room. They can tell their Alexa to wake them up at sometime and in the morning, the electronic shades slowly open, the lights are set to a low setting, and the virtual assistant greets them to start the morning.

Our system can extend into many other business sectors. The power control subsystem can be used in homes, prison cells, offices, and all other large, confined spaces to accommodate automatic controls and increase power efficiency. Various input signals could trigger the power saving mode to on, it does not have to be as specialized as checking out of a hotel room. We truly believe the applications are endless and ever more appropriate for an increasingly connected, modern, and efficient future.

Current Work:

Team C has completed all use cases developed for the second demo, integrated the subsystem into the full system, and developed another use case that uses motion sensors. The motion sensors are not implemented nor tested but have been fully developed. Current work would be to develop a test case for UC-*1 and implementing then integrating it into the system.

History of Work:

Use Case	Development	Inital Implementation	Updated	Completion
UC-11	02/28-03/15	03/15	03/21-03/24	03/25
UC-12	04/04-04/10	04/12	04/19-04/22	04/22

UC-13	02/28-03/15	03/15	03/20-03/22	03/25
UC-14	02/28-03/15	03/15	03/21-03/24	03/25
UC-15	4/04-04/10	04/12	04/19-04/22	04/22
UC-*1	04/14-04/28	N/A	N/A	Not Completed

5. Subsystem: Mobile Keys

Team B: Sidonia Mohan and Bharath Selvaraj

5.1 Summary of Changes

The requirements and use cases were all updated to accommodate the upgrading of the subsystem. The functionality of use cases and requirements remains intact however, all were revised.

REQ-0, REQ-1,REQ-22, REQ-28, REQ-35, REQ-36 - We changed the wording of everything to be in the actor point of view

UC10 - used to be separate for each role, but now we created a subroutine that lets one UC make keys for everyone and uses the code to create a master key that is set as a global variable.

UC-6 - used to be separate for each role, created a subroutine, which generalizes the login and based on what login credentials were used, features would be displayed according to the login type.

5.2 Glossary of Terms

Key: a 4 digit pin that will be distributed to customers after they book a room that will be used to enter their specific room. The staff will also be provided with a temporary 4 digit pin that will be available to them when they accept a housekeeping request and expire after the request is completed.

Master Key: the master key is a 4 digit pin that is only provided to the manager and this key will give the manager access to enter any room at any time. This is only implemented for any cases of emergencies that immediate access to a room is needed.

Staff Key: the staff key is a temporary key given to the staff/housekeeping when a cleaning request is made. The temporary pin will only be given after the request is accepted and the pin is set to expire after the request is marked as completed.

5.3 System Requirements

5.3.1 Enumerated Functional Requirements

Identifier	Priority	Description
REQ-0	5	The system shall generate a 4 digit key that will be accessible through the user interface
REQ-15	3	The system shall allow staff to view the room database.
REQ-22	5	As a staff member, I must be able to log in and see a corresponding pin for a room that requested cleaning.
REQ-28	1	As a Customer, I shall have access to the pins available to me
REQ-32	4	As a Customer, I will have to input the driver's license pin for identification purposes in payment to receive room.
REQ-35	3	As a Manger, I have access to a master 4 digit pin to access all rooms.

5.3.2 Enumerated Nonfunctional Requirements

Identifier	Priority	Requirement
REQ-24	5	The system shall generate a unique key for each room to increase room security.
REQ-42	3	The system shall make sure there is a keypad to enter the room key into every door.
REQ-36	3	The system shall make it so that the pin will no longer work after 11 am on the day of checkout for guest

5.4 Functional Requirement Specification

5.4.1 Stakeholders

Hotel Owners: Have a personal interest in Cloud Surf Inn as it allows for greater power efficiency in the hotel and a more pleasurable experience for their guests.

Hotel Guests: Cloud Surf Inn will lead to a more enjoyable, relaxing, pandemic-friendly experience for hotel guests who will utilize the system repeatedly throughout their stay.

Software Engineers: Want to make a system that is easily used that has the potential to maximize the efficiency of the hotel business and continuously improve the system.

5.4.2 Actors and Goals

Initiating :

Actor	Goal
Guest/Customer	The goal is to use a pin to get into their respective room to lodge there.
Staff/housekeeping	The goal is to maintain cleanliness around the hotel and rooms.
Manager	The goal is to keep track of guests and housekeeping and provide additional support in crisis

Participating:

Actor	Role
Database(Customer)	The database will save the customers key and allows them to gain access to their rooms until their checkout date and time
Database(Manager)	The database will give the manager the master key to access any rooms and provide the manager with access to everything related to the staff.
Database(Staff)	The database will keep track of what rooms need cleaning and provide the housekeeping with the appropriate tools

5.4.3 Use Case Casual Description

Identifier	Description	REQ-# Used
UC-6 (Subroutine)	"Login Page" - Allow User to see available keys 1. Allows customer to see keys to access certain rooms associated with them 2. Allows managers to see the master key which allows access to all rooms 3. Allows staff to see keys that are associated with rooms that need cleaning/assistance	REQ-22, REQ-33, REQ-34, REQ-35, REQ-37, REQ-38, REQ-39

UC-8	“Keypad” - Customer, manager, and staff input key pin into the keypad	REQ-36, REQ-35, REQ-22, REQ-0, REQ-1
UC-10 (Subroutine)	“Key Creation” System will create a 4 digit pin when its prompted to 1. Creates Key for manager (MasterKey) 2. Creates Key for guests when they book a room 3. Creates Key for staff when they need access to a room	REQ-0, REQ-1, REQ-20, REQ-2, REQ-35

5.4.4 Use Case Diagram

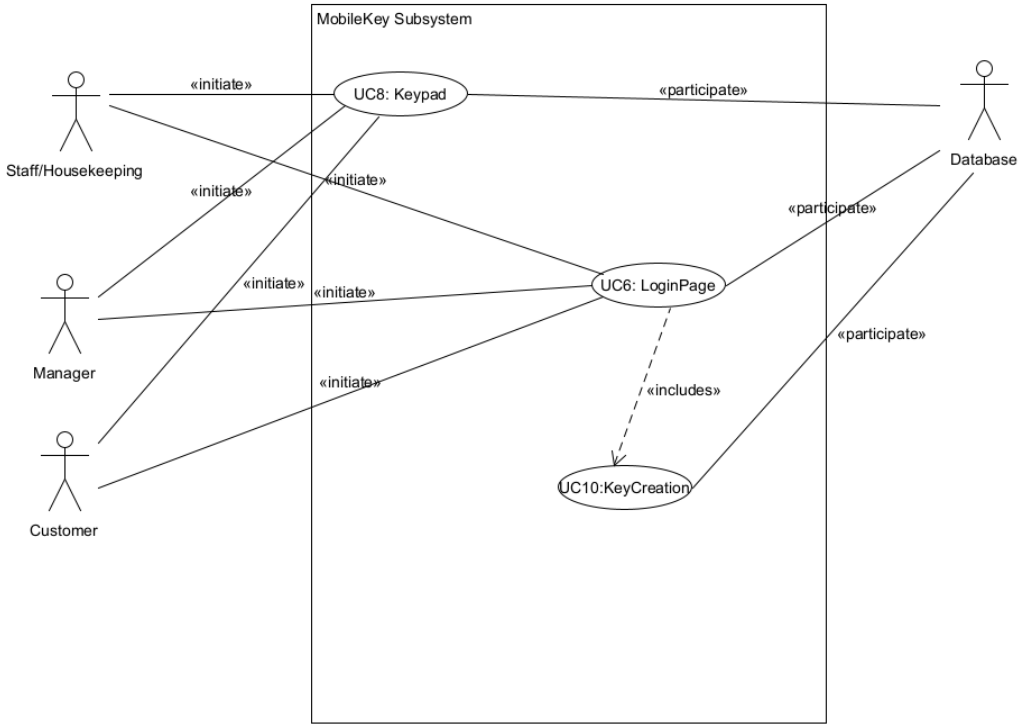


Figure 5.1: Use Case Diagram

The use case diagram above represents our subsystem, mobile keys. All main actors essentially have the same functions when it comes to using the use cases, however, what's in their respective roles will be shown in others parts. This UC diagram includes UC-6, UC-8, and UC-10. All users will first use the login page to view keys or create keys, if it is not already available to them, and

it will generate them a key via UC-10. Once that key is acquired, the user can input the key into the keypad for the designated room and will be able to enter. The database will store all information of what keys are related to what room, and UC-8 will reference the database to make sure that the proper key is entered for that specific room.

5.4.5 Fully Dressed Descriptions

UC - 6: Login Page
Related Requirements: REQ-22, REQ-35
Initiating Actors: Customer, Manager, Staff
Actor Goal: To log in to their respective accounts
Participating Actors: Database
PreCondition: The user has the ability to create an account or login through Cloud Surf Inn
PostCondition: To see the user's information
Minimum Guarantee: The system shows the available pins available to the user
Success Guarantee: The system shows the available pins available to the user
Flow Of Events for Success 1. → User creates an account 2. → User logins to their respective accounts 3. ← User can see their available information

UC-10: Key Creation
Related Requirements: REQ-0, REQ-20
Initiating Actors: Customer, Manager, Staff
Actor Goal: To log in and receive a 4 digit pin (key) for use
Participating Actors: Database
PreCondition: The user has a Cloud Surf Inn account that is validated (Manger/Staff have

accounts when hired, Customer needs to input information to create an account)
PostCondition: Be provided a pin
Minimum Guarantee: The system outputs a 4 digit pin for a specific user
Success Guarantee: The system outputs a 4 digit pin specific user
Flow Of Events for Success 4. → User logins 5. ←A 4 digit numerical key is generated 6. ←User can view available keys

5.4.6 System Sequence Diagram

UC-6 (Login Page)

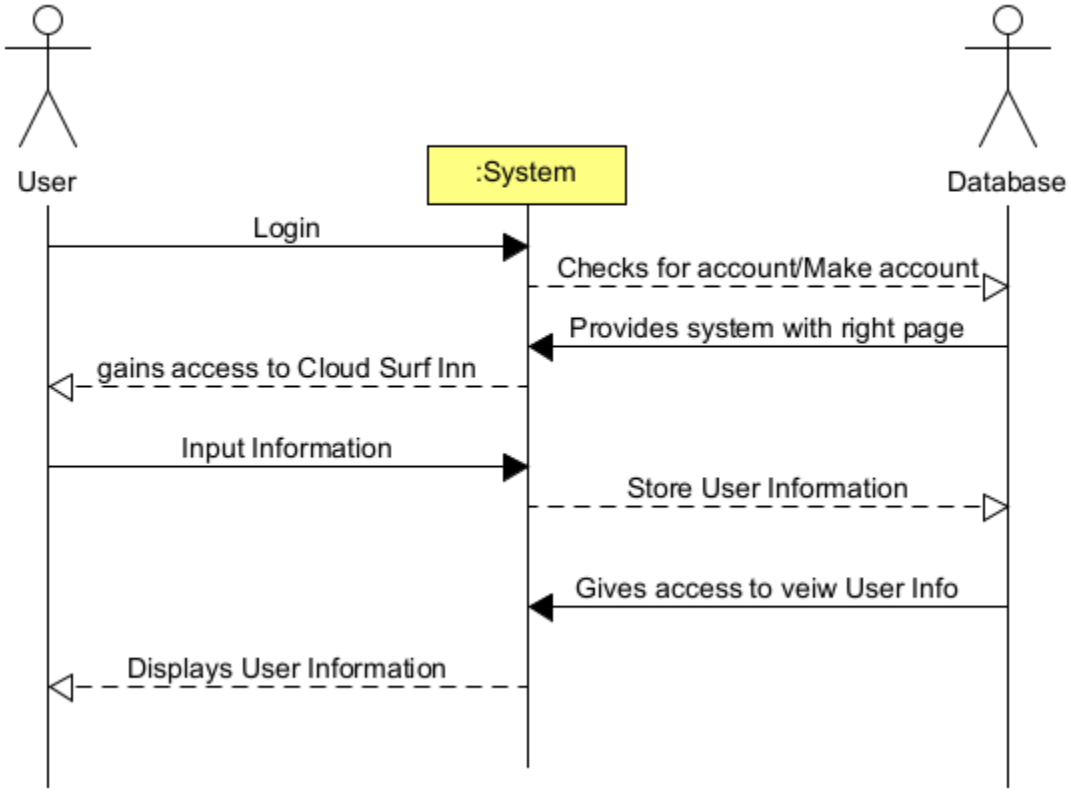


Figure 5.2: UC-6 System Sequence Diagram

The above figure is showing the steps of UC-6, the login page. First, a user must log in to the system, if there is no user account, one will be made and stored in the database. If a user already has one, nothing changes. Once logged in, the user will be required to input any extra information for the case of room booking. Once that happens the system will store all information. Then the system will assign rooms, and generate keys for the user, which will be displayed on the GUI for that specific user to see.

UC-10 (Key Creation)

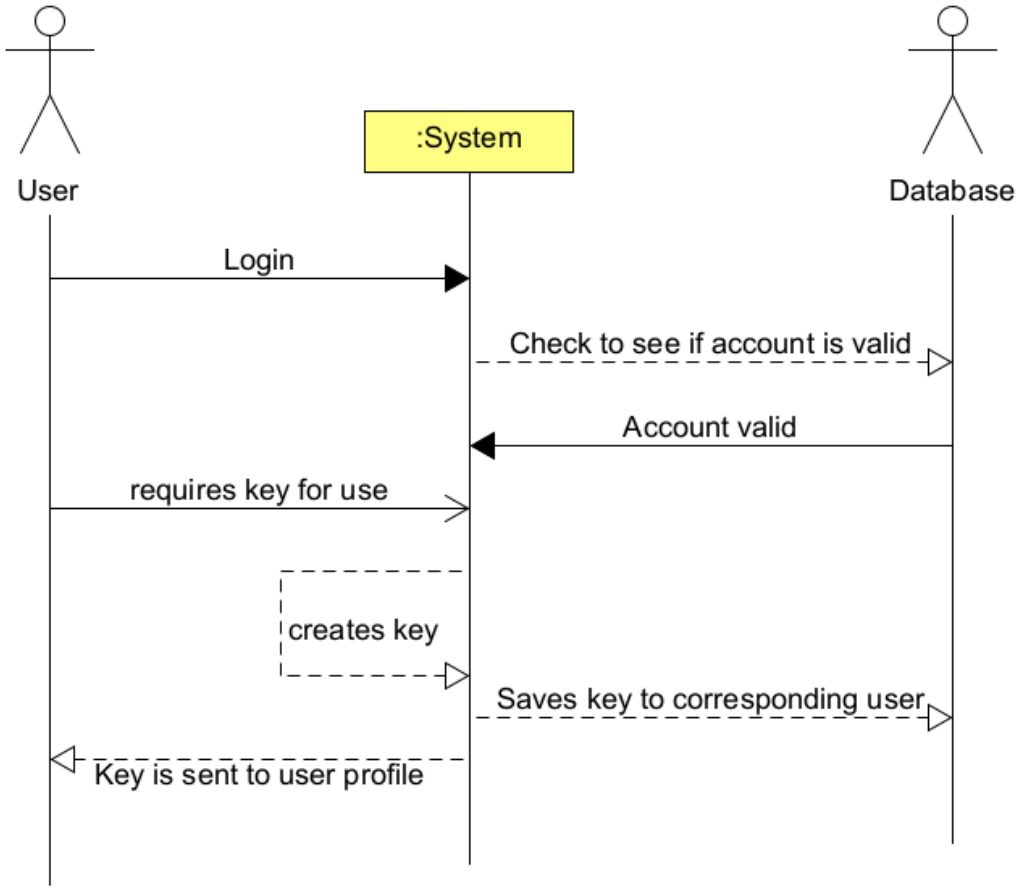


Figure 5.3: UC-10 System Sequence Diagram

The above figure shows UC-10, which is key creation. The user first logs in via the login page and proceeds as normal. When the user requires a key, whether it may be a guest checking in to the hotel room and needs their key or if it's for housekeeping that needs access to clean a room,

the system will create a key and save it to the database matched to the corresponding room and then the key is sent to the user interface, allowing them to view the key.

5.4.7 Traceability Matrix

	REQ-0	REQ-1	REQ-20	REQ-22	REQ-24	REQ-33	REQ-34	REQ-35	REQ-36	REQ-37	REQ-38	REQ-39	REQ-42
UC-8	X	X		X				X	X				X
UC-6				X		X	X	X		X	X	X	
UC-10	X	X	X	X	X			X					X

5.5 System Architecture and System Design

N/A

5.6 Domain Analysis

5.6.1 Concept Definitions

Concept	Responsibility	Type
DB Connection	Coordinate interactions/data exchange with the database.	Service
Key	A 4 digit pin used for room access then emailed to the user.	Service
MasterKey	4 digit pin that has access to all rooms	Service
Login Page	Allows users to select a profile and log in to the account. Inside each account contains information in regards to each position	Service
Staff Profile	Staff will be able to view the schedule of cleaning and the corresponding room’s pin.	Entity
Customer Profile	Profile will be used to store the customer's room key.	Entity

Manager Profile	Manager will be able to view the schedule of cleaning and the master key.	Entity

5.6.2 Association Definitions

Concept Pair	Association Definitions
LoginPage ↔ CustomerProfile	The credentials the user logs in will load the correct user profile and interface.
LoginPage ↔ StaffProfile	The credentials the employee logs in will load the correct employee profile and interface.
LoginPage ↔ ManagerProfile	The credentials the manager uses to log in with load the correct manager profile and interface
CustomerProfile↔Key	When user registers and books a room, they are generating a key
StaffProfile ↔ Key	Employees have access to keys that are needed for room cleans
CustomerProfile ↔ StaffProfile	The employee will be able to monitor any request the customer will make and view keys that are available to them.
ManagerProfile ↔ MasterKey	Manager will be able to view the master key which has access to all rooms

5.6.3 Attribute Definitions

Concept	Attribute	Definition	Constraints
Key	generateKey viewKey saveKey	The system will create a pin for people to gain access to rooms	Pin cannot repeat for more than one room at a time
StaffKey	generateStaffKey viewStaffKey saveStaffKey	A Pin that will give housekeepers access to rooms that need cleaning	Cannot access room until time requested by customer
MasterKey	generateMasterKey viewMasterKey saveMasterKey	The system will create a 4 digit pin that will allow	Cannot be same as any other pins given to customer or staff

		access to any room	
Manager Profile	viewMasterKey	Displays and stores all information of which the manager has the authority to access.	None
Staff Profile	viewStaffKey	Displays schedule of cleaning for the day with all pending requests for cleaning services.	None
Customer Profile	viewKey	Displays and stores the key and the room number	None
LoginPage	Username	Is the combination of letters, numbers, and symbols a user's respective profile has associated with it as its "name." The user will use this combination to log in to the system and it is not necessarily private.	Usernames cannot be the same as any other customers

	Password	Is the combination of letters, numbers, and symbols a user's respective profile has associated with it as its key to authorize login for the corresponding profile it pertains to. The user will use this combination to log in to the system.	None

5.6.4 Traceability Matrix

Concept:	UC-6 (Login Page)	UC-8 (Keypad)	UC-10 (Key Creation)
Database Connection	X	X	
Key	X	X	X
MasterKey	X	X	X
Login Page	X		
Staff Profile	X		X
Customer Profile	X		X
Manager Profile	X		X

5.6.5 Domain Model

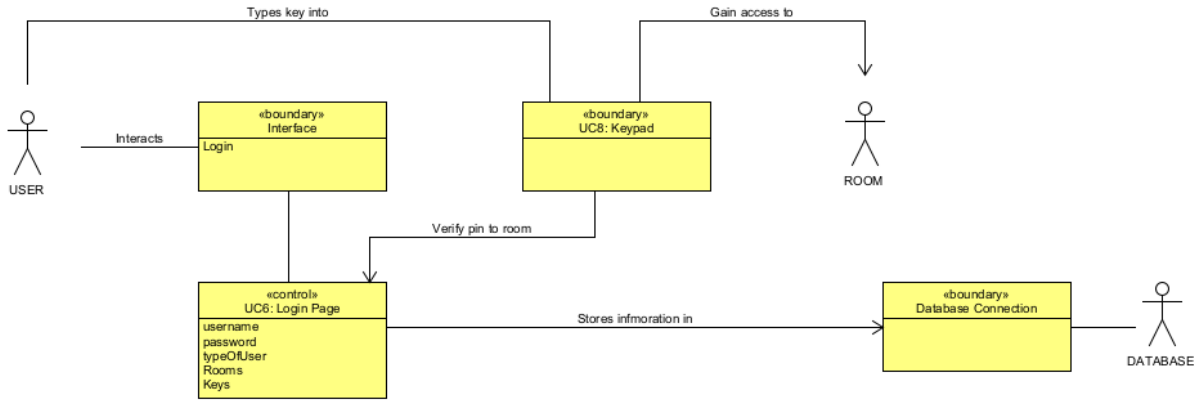


Figure 5.4: Domain Model

The domain model shows 1 main actor which is the user, the user can be any of the following: staff/housekeeping, Managerial Staff(aka Managers), and guest/clients of the Cloud Surf Inn. The boundary for the user is to first login via our GUI/interface, and once they log in they gain access to our control. For the login page, the user must input a username, password and specify what type of user they are (ex. managers selects manager). After that, they gain access to their specific user profile which will have keys available to them, and the rooms associated with those keys. The rooms and keys will be saved under the corresponding user in the database, and then using those credentials users can input their key into the keypad and they will gain access to the room.

5.6.6 System Operation Contracts

Operation	**Login Page**
Use Case:	UC-6 (Customer)
Preconditions:	➔ Customer(s) creates a unique username and password.
Postconditions:	➔ Customer(s) can view available pins.

Operation	**Login Page**
Use Case:	UC-6 (Manager)

Preconditions:	➔ Manager(s) creates a unique username and password.
Postconditions:	➔ Manager(s) can view master pin.

Operation	**Login Page**
Use Case:	UC-6 (Staff)
Preconditions:	➔ Staff(s) creates a unique username and password.
Postconditions:	➔ Staff(s) can view available pins.

Operation	**Key Creation**
Use Case:	UC-10 (Customer)
Preconditions:	➔ Customer(s) has logged into the system. ➔ Customer(s) has booked a room.
Postconditions:	➔ Customer(s) has been provided a pin.

Operation	**Key Creation**
Use Case:	UC-10 (Manager)
Preconditions:	➔ Manager(s) has logged into the system.
Postconditions:	➔ Manager(s) has been provided a master key.

Operation	**Key Creation**
Use Case:	UC-10 (Staff)
Preconditions:	➔ Staff(s) has logged into the system. ➔ Customer(s) has requested housekeeping.
Postconditions:	➔ Staff(s) has been provided a temporary pin.

5.6.7 Data Model & Persistent Data Storage

N/A

5.6.8 Mathematical Model

No mathematical models are utilized.

5.7 Interaction Diagram

UC-6 (Login Page) and UC-10 (Key Creation)

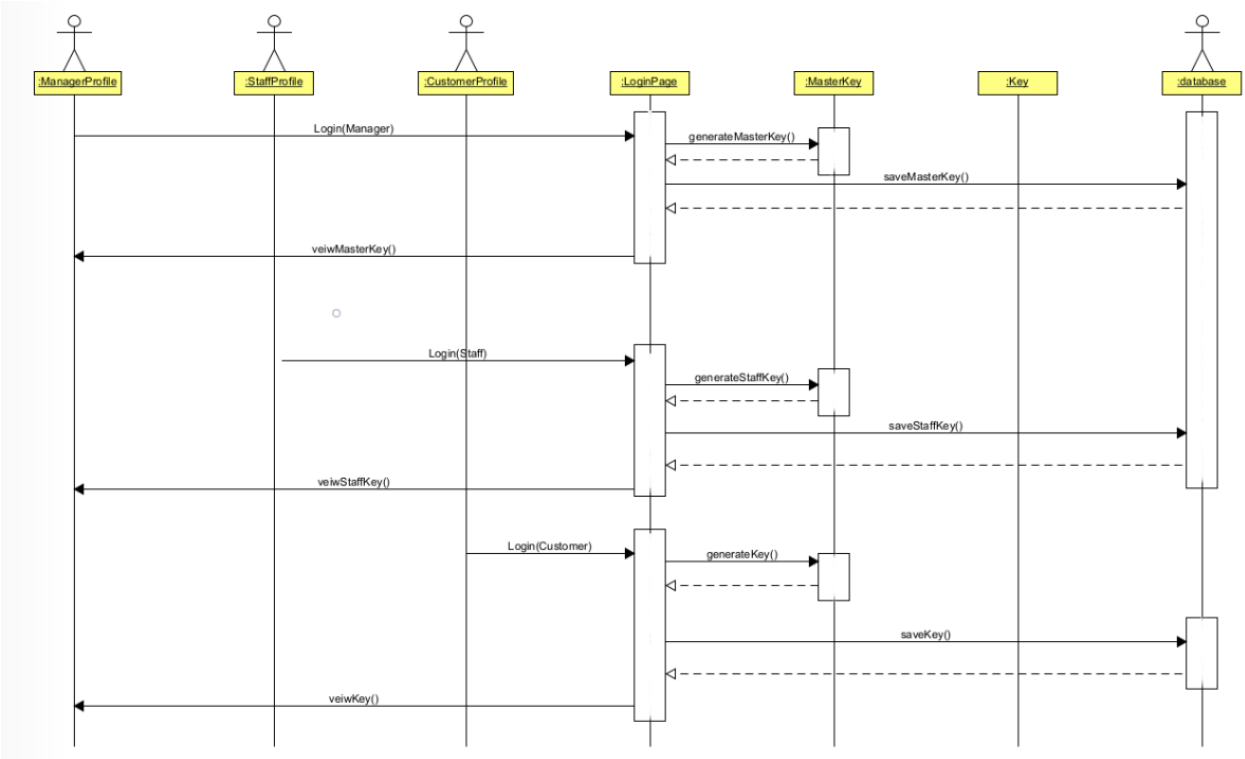


Figure 5.5: UC-6 and UC-10 Interaction Diagram

Each type of profile runs through the same steps, the only difference is that each key is different, masterkey unlocks all rooms, staff key is made when staff is prompted to by either manager or customer, and customer key is made when they book a room. The user will log in to their respective profile via the login() function on the GUI. The login page will create the specific type of key using either generateKey(), generateStaffKey(), or generateMasterKey() which will be sent back to the login page and the login page will save that key in the database, for any of the users to view their keys via the login page.

5.8 Class Diagram

Class	Meaning/Use of Class	Associations
generateKey	Creating a 4 digit pin for guest,	StaffProfile ↔ Key

	user, and manager to use to access rooms	UserProfile ↔ Key ManagerProfile ↔ Key
generateMKey	Creating a 4 digit pin for manager to use in order to access all rooms	ManagerProfile ↔MasterKey
CustomerProfile	Stores customer's information, room preferences, and room pin.	N/A
MangerProfile	Stores manager information and masterkeys	N/A
StaffProfile	Stores housekeeping requests and pin for designated cleaning rooms	N/A

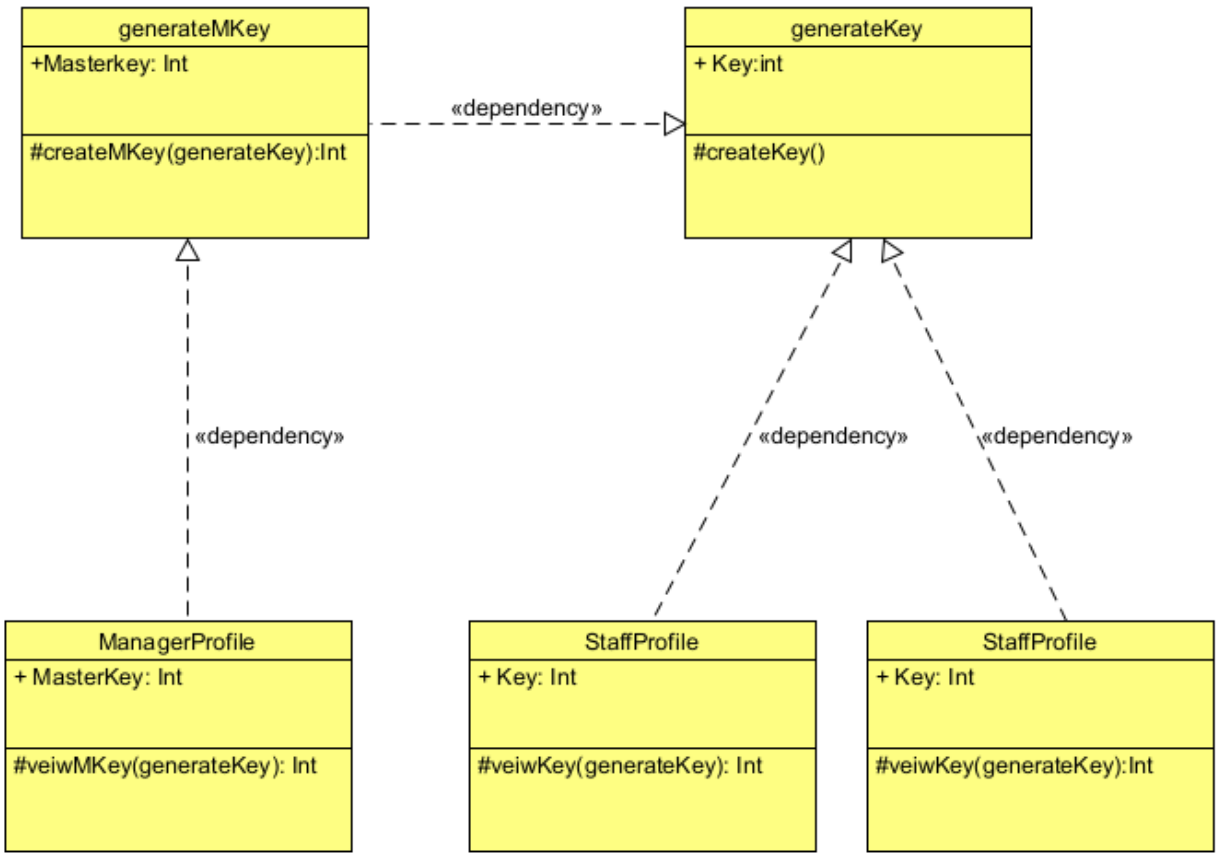


Figure 5.6: Class Diagram

5.9 Algorithms and Data Structures

N/A

5.10 Design of Tests

Use Case	Test Case Description	Test Coverage
UC-6	The manager will log in and be able to see the master key. The housekeeping will log in and see the key for the designated room they are trying to clean. The user will be able to log in and be able to view the key of the room that the Room Matcher chooses for the user.	This test will ensure that the correct key is displayed for the manager, housekeeping, and user when they log in to their designated accounts.
UC-10	When the user logs into the login page, they should be able to log into their accounts and view keys/ and tasks available to them.	This test will ensure that the key is shown, and so are the requests for housekeeping and staff

5.11 Effort Estimation using Use Case Points

Unadjusted Actor Weight (UAW):

Actor	Description	Complexity	Weight
Guest	Guests interact with the system via a graphical user interface.	Complex	3
Staff/housekeeping	Staff interacts with the system via a graphical user interface.	Complex	3
Manger	Manager interacts with the system via a graphical user interface.	Complex	3
Database	Database interacts with our system for storing information	Average	2

Unadjusted actor weight comes to 3*3 + 2 = 11

Unadjusted Use Case Points (UUCW):

Use Case	Description	Category	Weight
Login Page (UC-6)	Moderate interface design. 4 participating actors (guest, staff, manager, database). Up to 7 steps.	Average	10
Keypad (UC-8)	No user interface. 4 participating	Average	10

	actors (guest, staff, manager, database). Up to 5 steps.		
Key Creation (UC-10)	Moderate user interface. 1 participating actor (database). No steps required.	Average	10
		Use Case Weight Total	30

Unadjusted Use Case Weight = 3*10 = 30

Technical Complexity Factor (TCF):

Technical Factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight*Perceived Complexity)
T1	System is not distributed.	2	0	2*0=0
T2	Users expect good performance but no critical response times or throughput.	1	3	1*3=3
T3	Users expect high efficiency but it is not exceptionally critical.	1	3	1*3=3
T4	Internal processing is relatively simple.	1	1	1*1=1
T5	System should be able to be reused in other hotels.	1	5	1*5=5
T6	Software needs to be easily installed for different hotels.	0.5	5	0.5*5=2.5
T7	Ease of use is very important.	0.5	5	0.5*5=2.5
T8	No portability concerns.	2	0	2*0=0
T9	System should be able to handle new features but is unlikely to need to do so.	1	2	1*2=2
T10	Concurrent use is required.	1	4	1*4=4

T11	Security is a significant concern.	1	5	1*5=5
T12	No direct access for third parties.	1	0	1*0=0
T13	No unique training needs.	1	0	1*0=0
			Technical Factor Total:	28

Technical complexity factor comes to 0.6+0.01*28=0.88

Environmental Complexity Factor (ECF):

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor (Weight*Perceived Impact)
E1	Beginner familiarity with UML-based development.	1.5	1	1.5*1=1.5
E2	Little familiarity with application problem.	0.5	1	0.5*1=0.5
E3	Some knowledge of object-oriented approach.	1	2	1*2=2
E4	Beginner lead analyst.	0.5	1	0.5*1=0.5
E5	Motivated with some team members lacking.	1	4	1*4=4
E6	Stable requirements expected.	2	5	2*5=10
E7	No part-time staff.	-1	0	-1*0=0
E8	Programming language of average difficulty will be used.	-1	3	-1*3=-3
			Environmental Factor Total:	15.5

Environmental complexity factor comes to 1.4-0.03*15.5= 0.935

Use Case Points and Duration:

UCP = UUCP * TCF * ECF = (30 + 11) * 0.88 * 0.935 = 33.7348 = 34 use case points.

Duration = UCP * PF = 34 * 28 = 952

5.12 History, Current, and Future Work

Future Work:

One of the main aspects we would like to incorporate in our future work is the usage of QR codes to enter a room as opposed to a 4 digit pin. This would be more user-friendly and efficient because the user would be able to scan the QR code using a QR code scanner attached to the door handle. This, in turn, would give them access after the scanner has recognized the code as correct. The QR code would also significantly reduce the time the customer spends trying to open their designated room.

The QR system can also be implemented in other commercial and residential buildings. Instead of carrying a keycard around to enter buildings and rooms in commercial spaces, the user can just use the QR code to enter a facility. This would reduce the risk of forgetting keycards and not being able to enter. QR codes can also be used for student housing as students would be able to use their designated QR codes rather than use their student IDs.

Current Work:

For Team B's current work, we have completed all use cases developed for the second demo, integrated the subsystem into the full system, and are currently exploring avenues to implement additional test cases.

History of Work:

Use Case	Development	Initial Implementation	Updated	Completion
UC-6	3/01-3/24	3/20	3/24-3/26	3/26
UC-10	3/15-3/20	3/19	3/21-3/23	3/23

6. Subsystem: Housekeeping

Team A: Ryder Morrello, Juan Escudero, Zach LeMunyon

6.1 Summary of Changes/Current Work

The requirements and use cases were all updated to accommodate for the changes and updates made to the system. Despite this, the functionality of the use cases and requirements all remain the same; they were merely renumbered and/or revised.

Updated: REQ-11, REQ-12, REQ-13, REQ-14

New:REQ-44

REQ-45: When someone checks out and the room is marked vacant, all requests from that room are deleted and one housekeeping request is scheduled to clean the room for the next guest.

Furthermore, the Room Service microservice was implemented as a part of Cloud Surf Inn. The Room Service system can be expanded to feature things like food or other services a hotel may wish to offer guests. The use cases which were implemented and demonstrated in Demo 2 are the following:

REQ-2, REQ-9, REQ-10, REQ-21, REQ-37, REQ-40

UC-19, UC-23, UC-24

Housekeeping:

Since Demo 1, the Housekeeping System has undergone very few changes to the features that were already implemented. However, all features (including new ones) have been implemented in some way within the user interface and the database. The database now stores the primary housekeeping heap along with the housekeeper heaps of all housekeepers within the Cloud Surf Inn system. Thus, housekeeping requests persist in the system even if a session is terminated. The user interface now generates a temporary PIN for housekeepers to use to access the room which is next on their schedule. This PIN is regenerated every time a housekeeper will clean a different room and this helps maintain guests security and privacy. The user interface also has a tentative notification system that prints any notifications a guest has in regards to a

housekeeping request they have solicited. Another new feature is that whenever a customer checks out of Cloud Surf Inn and their room is marked as vacant, the Housekeeping System automatically deletes any and all cleaning requests the new departed guest has issued. This may seem like a small addition, but it is extremely critical in maintaining the integrity of the Housekeeping System so that all requests in the system are relevant and can be serviced without wasting any time and effort housekeepers will spend on cleaning rooms.

Room Service:

The Room Service system was nonexistent during Demo 1; so this is an entirely new subsystem within Cloud Surf Inn! We did not expect to have the time to implement a Room Service System, but since the notion of a service system is extremely similar to the Housekeeping System, we decided to do our best to implement it. As of now, this system allows customers to request a service in the form of items they wish to have delivered to their room. Items range from toiletries, bath products, and items for the bed. The request will then be added to a queue of service requests which will be managed by the hotel's service staff (service staff is a broad term that can include chef's, waiters, or anyone the hotel deems as a service employee). The service staff can then accept requests and service them while the system automatically notifies guests with the progress of their service request using a tentative notification system. Even though the Room Service system is functional, our vision for the system is to make it modular and thus allow a hotel to offer whatever services it wishes. These services can include food items, exercise equipment, or anything else a hotel may wish to offer its guests as services; a hotel could even offer no services should they desire to do so.

6.2 Glossary of Terms

HPHeap (Housekeeping Heap): The housekeeping heap is the heap/priority queue (a min-heap specifically) that stores and contains all housekeeping requests made by guests. These housekeeping requests are visible to all housekeepers logged in to the Cloud Surf Inn System and can be accepted by an available housekeeper to be serviced. There will only be one housekeeping heap at any given time in the Cloud Surf Inn System.

HKPRHeap (Housekeeper Heap): The housekeeper heap is a heap/priority queue (a min-heap specifically) that stores and contains all housekeeping requests accepted by a particular housekeeper. Each housekeeper logged in to Cloud Surf Inn will have one housekeeper heap associated with them.

High Traffic Area: Location in hotel where many guests traverse or congregate throughout any typical day. Examples could be elevators, hallways, doors, lounges, gyms, bathrooms, and stairwells. These areas demand special attention from housekeeping staff to ensure cleanliness.

Service Staff: Service staff/employees are employees that a hotel uses to offer services to a guest. They can include cooks, chefs, waiters, exercise coaches, or anyone whom a hotel utilizes to offer a service to a customer. The term is kept vague so that a hotel can decide whichever staff members are categorized as service staff.

Housekeeper: Housekeepers are custodial staff which clean and maintain a hotel utilizing Cloud Surf Inn. They are actual employees like maids, maintenance workers, and custodians that can clean and maintain a hotel's cleanliness. The term is kept vague so that a hotel can decide whichever staff members are categorized as housekeepers.

6.3 System Requirements

6.3.1 Enumerated Functional Requirements

Housekeeping Requests:

Identifier	Priority	Description
REQ-11	5	The system allows customers to book housekeeping through the interface.
REQ-12	2	The system allows the manager to view schedules that show when the maids will come in and clean.
REQ-13	3	The system displays which rooms or areas need cleaning for the maids.
REQ-14	2	The system allows maids to input when they last cleaned high-traffic areas or rooms.

Room Service:

Identifier	Priority	Description
REQ-2	2	The system allows guests to order food and be able to add/request additional information regarding that item.
REQ-9	2	The system allows customers to view TV channels and service options.
REQ-10	2	The system allows customers to order utilities or services through the user interface.
REQ-21	3	Managers can manipulate service requests.
REQ-37	4	Staff can login and view and update service requests.
REQ-40	2	Customers can track their ordered food/services.

6.3.2 Enumerated Nonfunctional Requirements

Housekeeping:

Identifier	Priority	Requirement
REQ-26	3	The hotel system should be able to schedule room cleanings so maids do not have spare time.

Room Service: *N/A*

6.4 Functional Requirement Specification

6.4.1 Stakeholders

Hotel Owners: Have a personal interest in Cloud Surf Inn as it allows for greater power efficiency in the hotel and a more pleasurable experience for their guests.

Hotel Guests: Cloud Surf Inn will lead to a more enjoyable, relaxing, pandemic-friendly experience for hotel guests who will utilize the system repeatedly throughout their stay.

Software Engineers: Want to make a system that is easily used that has the potential to maximize the efficiency of the hotel business and continuously improve the system.

6.4.2 Actors and Goals

Initiating Actors

Actor	Goal
Housekeeper	The housekeeper/custodian is responsible for keeping rooms, hallways, and areas within the hotel clean. They will accept requests to provide cleaning and indicate whenever requests have been completed. Housekeepers can also notify guests whenever their cleaning requests have been satisfied.
Guest/Customer	The guest is a customer staying at the hotel; they use the Cloud Surf Inn system to satisfy their desires and necessities. They can solicit housekeeping or room service requests whenever they wish.

Participating Actors

Actor	Role
Housekeeper	Housekeepers will be notified whenever housekeeping requests have been made by customers and accept them accordingly.
Service Employee	Service employees responsibilities include but are not limited to cooking food, delivering food/services, and indicating services are ready to be delivered/offered.
Manager	The manager will have the high-level responsibility of managing all the needs of the hotel and guests. The manager can view and edit housekeeping requests as well as room service requests.

6.4.4 Casual Description

Housekeeping:

Identifier	Description	REQ-# Used
UC-16	“Housekeeping Request” - Allows customers to send housekeeping requests when they are out of the room.	REQ-11

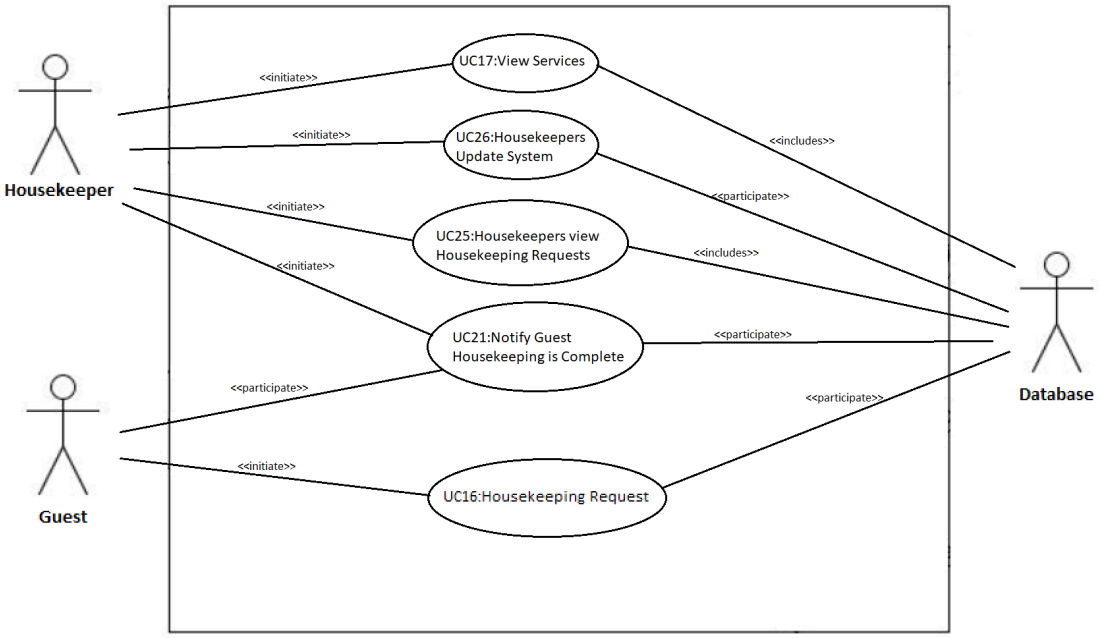
UC-17	“View services” - Housekeepers view what rooms need service and are provided a pin for that room designated for a specific time slot.	REQ-22, REQ-37, REQ-21
UC-21	“Notify Guest Housekeeping is Complete” - Allows the guests to be notified when their room is clean and the housekeeping has left.	REQ-14, REQ-15
UC-25	“Housekeepers view housekeeping requests” - Allows housekeepers to view housekeeping requests.	REQ-13
UC-26	“Housekeepers update system” - The housekeepers can input if they cleaned a room and when they cleaned a high traffic area.	REQ-14

Room Service

Identifier	Description	REQ-# Used
UC-19	“Ordering Services” - Allows customers to send orders to service staff or maintenance.	REQ-10, REQ-2
UC-23	“View the Services Menu” - Allows a guest to view the available menu of services that the hotel has to offer.	REQ-9, REQ-10
UC-24	“Service Status Update” - The employee notifies guests about the status of their requests.	REQ-40, REQ-21, REQ-37

6.4.4 Use Case Diagram

Housekeeping:

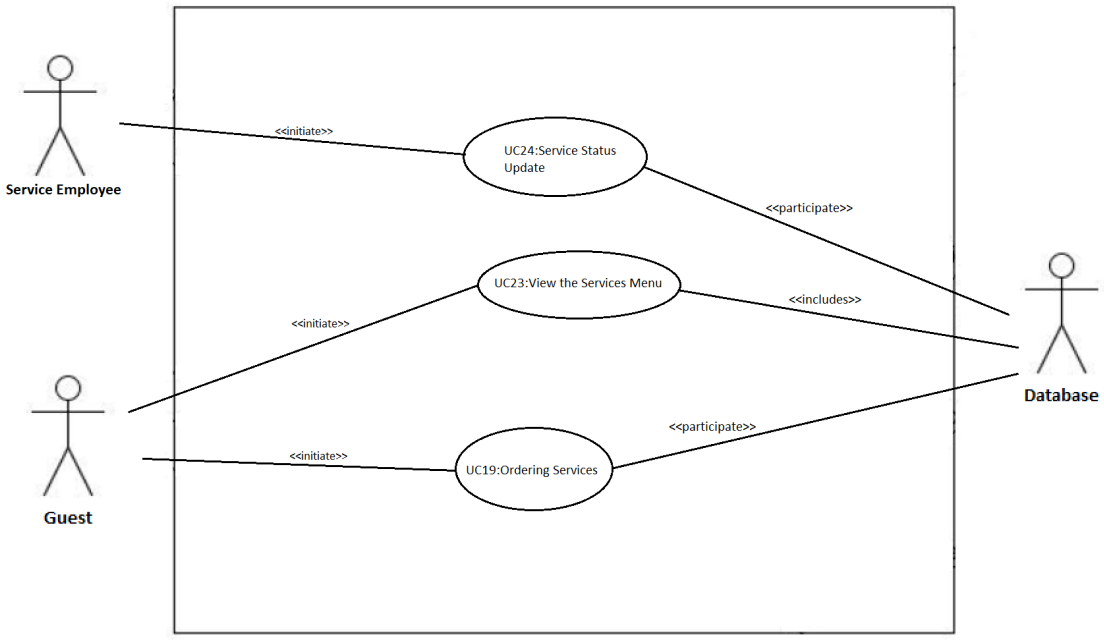


Revised Use Case Diagram for the Housekeeping Subsystem.

The Housekeeper and Guest can serve as both the initiating and participating actors depending on what usecase is being fulfilled.

The implemented usecases are shown in the diagram.

Room Service:



Revised Use Case Diagram for the Room Service Subsystem.

The Service Employee and Guest can serve as both the initiating and participating actors depending on what usecase is being fulfilled.

The implemented usecases are shown in the diagram.

6.4.5 Fully Dressed Descriptions

UC-25: Maids View Housekeeping Request
Related Requirements: REQ-12, REQ-13, REQ-15
Initiating Actors: Housekeeper
Actor Goal: The goal of the customer is to request housekeeping to their room at the Cloud Surf Inn.

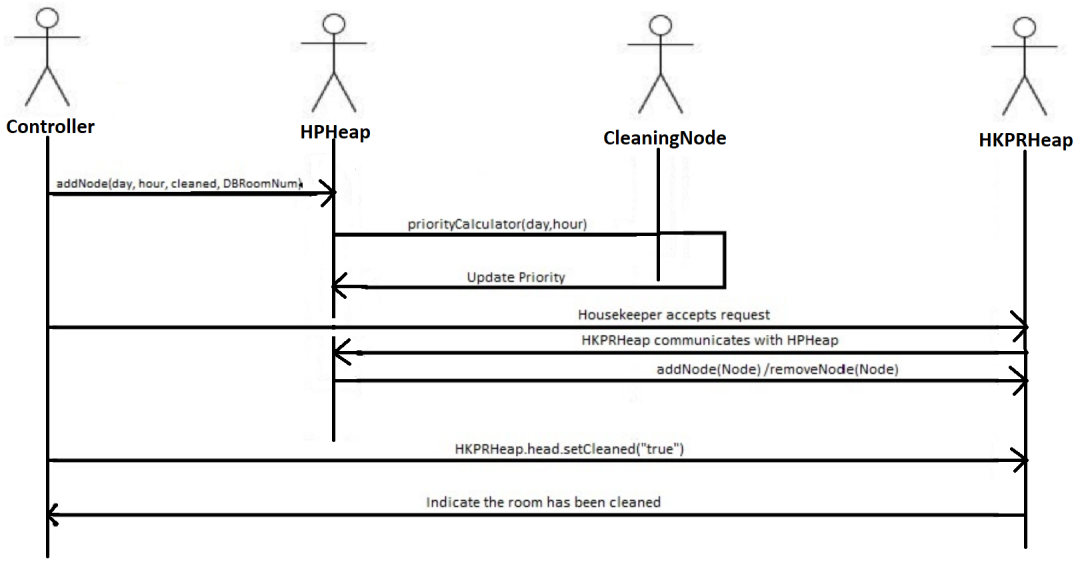
Participating Actors: N/A
PreCondition: Customer must have made a housekeeping request.
PostCondition: Housekeepers will know the time when they must clean the customer's room.
Minimum Guarantee: Housekeepers will accept a cleaning request and it will be added to their schedule if possible. They will be able to view scheduled times of cleaning requests (even if not so elegantly). They will be able to indicate when their task has been completed.
Success Guarantee: Housekeeper will accept cleaning requests and view them in a calendar format. They will be able to adjust their schedules, indicate when their task has been completed, and indicate if additional time is needed.
Flow Of Events for Success: 1. → The housekeeper logs in while using the "Staff" option. 2. → The housekeeper clicks the "Housekeeping" tab. 3. ←The system shows the housekeeper(s) the rooms that must be cleaned along with the times associated with them.
Alternate Flow of Events: 1. → The customer sets up an account or logs in. 2. → The customer selects the "Room Service" tab. 3. → The customer submits a "Special Request" specifically asking for housekeeping or a service only a custodian could accomplish. 4. ←The system updates the service staff's interface with the guest's "Special Request".

UC-19: Ordering Services
Related Requirements: REQ-2, REQ-9, REQ-10, REQ-40
Initiating Actors: Customer
Actor Goal: The goal of the customer is to order any services they want or need.
Participating Actors: Service Employee
PreCondition: Customers must have an account on the Cloud Surf Inn website. Customers must be staying in a room at the moment of ordering. Hotel must have services to offer guests.
PostCondition: After this the customer will have whatever item or service they wanted delivered to their room. Customer will be notified of progress up unto and including delivery of service item.
Flow Of Events for Success:

1. → The customer sets up an account or logs in. 2. → The customer clicks “Room Service” 3. ← The system shows the customer everything there is to order/everything offered. 4. → The customer selects everything they would like to order and clicks “Submit”. 5. ← The system then tells the Service Employee(s) the customers order. 6. → Service System updates guest with order progress. 7. → Service Employee(s) automatically notifies guest that order is complete. 8. → Service Employee(s)/ delivers order to customer.
Alternate Flow of Events: 1. → The customer sets up an account and logs in. 2. → The customer clicks “Room Service” 3. ← The system shows the customer everything there is to order. 4. → The customer selects “Other,” types in their custom note, and clicks “Submit”. 5. → The system then tells the Service Employee(s) the customer's request. 6. ← The Service Employee(s) then fulfill the service normally. 7. → Service Employee(s)/ delivers order to customer.

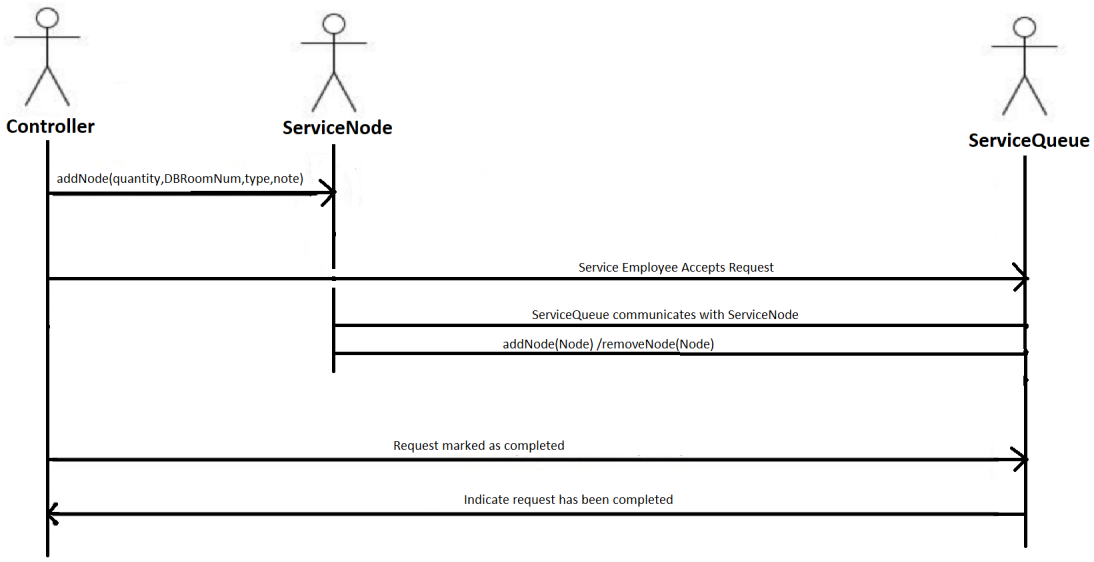
6.4.6 System Sequence Diagram

Housekeeping:



This is the System Sequence Diagram for the Housekeeping Subsystem. The controller is the user interface which the Guest interacts with to solicit a housekeeping request.

Room Service:



This is the System Sequence Diagram for the Room ServiceSubsystem. The controller is the user interface which the Guest interacts with to solicit a room service request.

6.4.7 Traceability Matrix

Housekeeping:

UC #	REQ-11	REQ-13	REQ-14	REQ-15	REQ-21	REQ-22	REQ-37	HREQ-1	HREQ-2
UC-16	X								
UC-17					X	X	X		
UC-21			X	X					X
UC-25		X						X	
UC-26			X						X

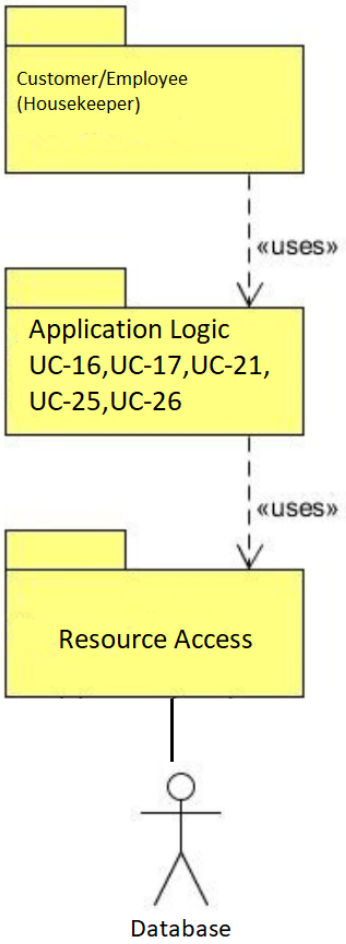
Room Service:

UC #	REQ-2	REQ-9	REQ-10	REQ-21	REQ-37	REQ-40	SREQ-2	SREQ-3
UC-19	X		X					
UC-23		X	X					
UC-24				X	X	X	X	X

6.5 System Architecture and System Design

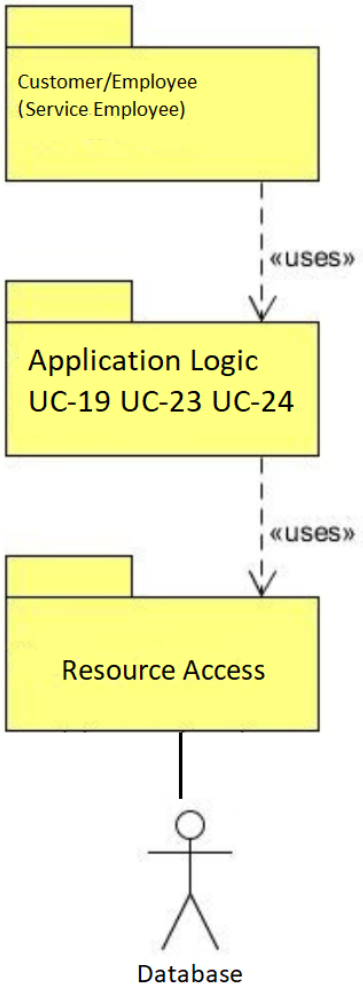
6.5.1 Identifying Subsystems

Housekeeping



This is the Subsystem diagram for the Housekeeping Subsystem. The customer/housekeeper interacts with the system to exchange information with the database.

Room Service



This is the Subsystem diagram for the Room Service Subsystem. The customer/service employee interacts with the system to exchange information with the database.

The Housekeeping and Room Service subsystems are both identical to each other in theory. Thus, when discussing their architecture and design, they will be treated the same and not differentiated. The subsystems are designed to be layered. The guest interacts with the user interface to solicit requests, and in turn, the user interface uses the application logic package.

Naturally, the application logic package is where requests from the user interface are received and where the business logic is stored. The data structures which compose the housekeeping and room service subsystems are processed in the application logic layer. The application logic layer utilizes the resource access layer which itself is a generalized layer that facilitates the interaction between the participating actors. This is where the system stores information within the database as well as updating the user interfaces with new, relevant information.

6.5.2 Architecture Styles

Just as with the Power Controls subsystem, the housekeeping and room service subsystems feature the Three-tiered Layered Architecture architectural style. The three tiers are the presentation layer, an application layer, and a resource access layer. This allows Cloud Surf Inn to be innovated and updated modularly; each tier can be modified independently from the others. The three-tiered architectural style also bolsters the system security because each layer cannot access all elements of the other layers. This also helps with overall system stability because errors and bugs in one layer cannot carry over and corrupt information in other layers. Naturally, the presentation layer is what customers and staff members directly interact with. The application layer is where the back-end of the system resides and thus where all of the important data processing, management, and storage occurs. The application layer works in tandem with the resources access layer to acquire information. The application layer also has the important task of performing the tasks that are automated, which of course offer important business value.

The housekeeping and room service also feature the Client-server model. The client-side consists of the customers, housekeepers, and service staff that must access the server to access features, create requests, relay requests, and notify other actors with new information. There must be a dedicated server to run and store the business logic that makes the housekeeping and room service systems possible. This client-server model facilities the ability for several guests and staff members to simultaneously connect to the application and manipulate it concurrently to perform their necessary and desired tasks.

6.5.3 Mapping Subsystems to Hardware

Cloud Surf Inn's user interface runs on the client's computer. This allows for the application to be a simple and lightweight application that any guest can run. This also allows for

multiple guests to access the application layer at one time without interfering with their individual experiences. The application and resource layer are on the server computer to handle requests from customers or staff as well as save and manage the exchanges with the database where critical information is stored.

6.5.4 Connectors and Network Protocols

Our subsystem runs on a single application. Therefore, we are not applicable for this section.

6.5.5 Global Control Flow

Execution Orderliness - Overall, Cloud Surf Inn is an event-driven system because we have different systems that wait for a customer/user to utilize the system and the system has to then respond accordingly. Anyone using the system can generate the actions in a different order, and for certain features like housekeeping, the system will wait until an actor who can service the requests participates in the system and updates the request. However, there is some procedure-driven "linearity" in the execution flow as well. Every user, regardless of their relationship to the hotel, will be required to sign in and perform a predefined sequence of steps to interact with specific parts of the system. For example, whenever any customer wishes to request room service, they will input specific, required information (service type/item, quantity, and a custom note) through the client, which will be sent to the controller upon submission. This controller performs the service node creation given data from both the client and the server containing the database then adds that service node to the queue of services ; this happens for every customer every time they request room service. After signing in, the linearity of the system diverges because users with different roles will have various levels of access to the different facets of the system. For instance, a manager will have significantly more options and sub-systems to interact with than a housekeeper would. At this point, the system is waiting for certain events to occur before responding accordingly. Yet for things like scheduling housekeeping or requesting room service, the process itself is linear regardless of who is requesting it.

Time Dependency - The housekeeping and room service subsystems are event-response type with no true concern for time. The system only interacts with time for user convenience. Things like scheduled housekeeping requests will have a time associated with them (to calculate the

priority), but the system does not time these requests and automatically handle their resolution. Continuing with the housekeeping example, a housekeeper will indicate that a request has been completed only when they have finished regardless if that was ahead or behind schedule; the system will not automatically notify the guest that their room has been cleaned at the time the cleaning was scheduled to end. Furthermore, for employees clocking-in and clocking-out, the system will not time them. Rather, it will simply keep track of when employees clock-in and when they clock-out out and store them for later use.

6.5.6 Hardware Requirements

There are no hardware requirements specific to our subsystem. As long as you follow the hardware requirements for the overall system, then everything will work inside our subsystem as well.

6.6 Domain Analysis

6.6.1 Concept Definitions

Housekeeping:

Concept	Responsibility	Type
Interface	A graphical user interface where the user is shown what actions can be taken and select to have them done.	Boundary
Controller	Handles and coordinates the actions between the interface and the various use cases that must be completed and delegates work to other concepts.	Control
DB Connection	Coordinate interactions/data exchange with the database.	Service
Accept Task	Allows the housekeeper to accept a cleaning request given by the customer.	Service
Remove Task	Allows housekeepers to remove a cleaning request they have chosen to no longer perform.	Service
Delete Task	Allows someone with higher levels of authority (Customer/Manager) to delete the cleaning request before it has been accepted by a housekeeper.	Service
Create Task	Allows the customer to create a cleaning request.	Service

Room Service:

Concept	Responsibility	Type
Interface	A graphical user interface where the user is shown what actions can be taken and select to have them done.	Boundary
Controller	Handles and coordinates the actions between the interface and the various use cases that must be completed and delegates work to other concepts.	Control
DB Connection	Coordinate interactions/data exchange with the database.	Service
Accept Task	Allows the service employee to accept a room service request given by the customer.	Service
Remove Task	Allows service employees to remove a service request that has been fulfilled.	Service
Create Task	Allows the customer to create a room service request.	Service

6.6.2 Association Definitions

Housekeeping:

Concept Pair	Association Definitions
Create Task ↔ Accept Task	Whenever a customer creates a housekeeping request, a housekeeper must be able to see that request and accept it as a task. Otherwise, the request can never be seen or manipulated by a housekeeper and thus will never be solicited. The HPHeap and HKPRHeap priority queues will communicate with each other to make this exchange possible and ensure that any and all requests can be accepted by a housekeeper.
All Tasks ↔ DB Connection	All tasks (accept, remove, delete, and create) must connect to the database and keep its information up to date so that the priority queue data structures can be properly maintained with the most recent information available. Any changes will be processed by the database. This ensures that all requests are all accounted for and able to be tracked and manipulated.

Room Service:

Concept Pair	Association Definitions
Create Task ↔ Accept Task	Whenever a customer creates a room service request, a service employee must be able to see that request and accept it as a task. Otherwise, the request can never be seen or manipulated by a service employee and thus will never be solicited. The Service Queue will ensure that any and all requests can be accepted by a service employee and handled accordingly.
All Tasks ↔ DB Connection	All tasks (accept, remove, delete, and create) must connect to the database and keep its information up to date so that the priority queue data structures can be properly maintained with the most recent information available. Any changes will be processed by the database. This ensures that all requests are all accounted for and able to be tracked and manipulated.

6.6.3 Attribute Definitions

Housekeeping:

Concept	Attribute	Definition	Constraints
DB Connection	DBRoomNum	Coordinate interactions/data exchange with the database.	Must be able to be changed from the database.
	DBChanger		
	DBChangerVacant		
	DBChangerCleaned		
Accept Task	cleaningNode	Allows housekeeper to accept a cleaning request given by the customer	Cannot have time conflict with another request.
Remove Task	cleaningNode	Allows housekeepers to remove a cleaning request they have chosen to no longer perform.	Must be more than 3 hours before cleaning is scheduled.
	DBRoomNum		
Delete Task	CleaningNode	Allows someone with higher levels of authority (Customer/Manager) to delete the cleaning request before it has been accepted by a	Must be more than 1 hour before cleaning is scheduled.
	DBRoomNum		

		housekeeper.	
Create Task	priority day hour	Allows a customer to schedule a cleaning request at a specified day and time. The priority sorts the priority queue accordingly.	Time for cleaning must be greater than 3 hours away - Must not have a conflict with another scheduled cleaning service at the same time.

Room Service:

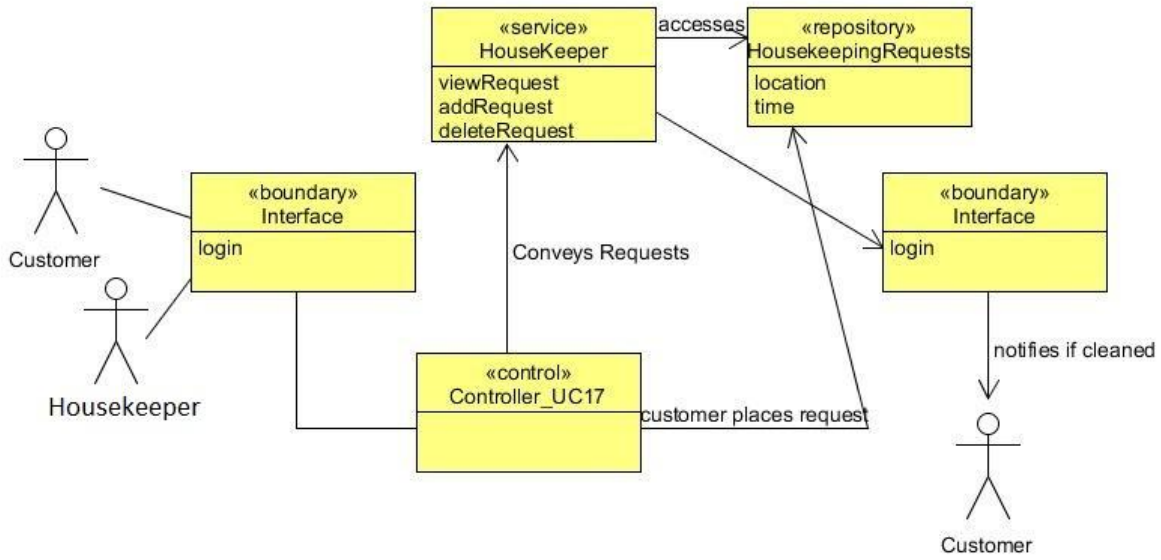
Concept	Attribute	Definition	Constraints
DB Connection	DBRoomNum DBChanger DBChangerVacant DBChangerCleaned	Coordinate interactions/data exchange with the database.	Must be able to be changed from the database.
Accept Task	ServiceNode	Allows service employee to accept a cleaning request given by the customer	Cannot have time conflict with another request.
Remove Task	ServiceNode DBRoomNum	Allows service employees to remove a cleaning request they have chosen to no longer perform.	Request must have been accepted being being removed.
Create Task	quantity time DBRoomNum type note accepted	Allows a customer to schedule a room service request detailing the type of service, an optional note, a quantity, and the current time for use in the Service Queue. The DBRoomNum and accepted status do not require user input.	- The customer must be booked at the hotel and have a room at the time of making the room service request. - Service employee(s) must be available.

6.6.4 Traceability Matrix

Concept:	UC-16	UC-17	UC-19	UC-21	UC-23	UC-24	UC-25	UC-26
Interface	X	X	X	X	X	X	X	X
Controller	X	X	X	X		X	X	X
DB Connection	X	X	X				X	X
Accept Task				X		X		X
Remove Task							X	
Delete Task					X			
Create Task	X		X					X

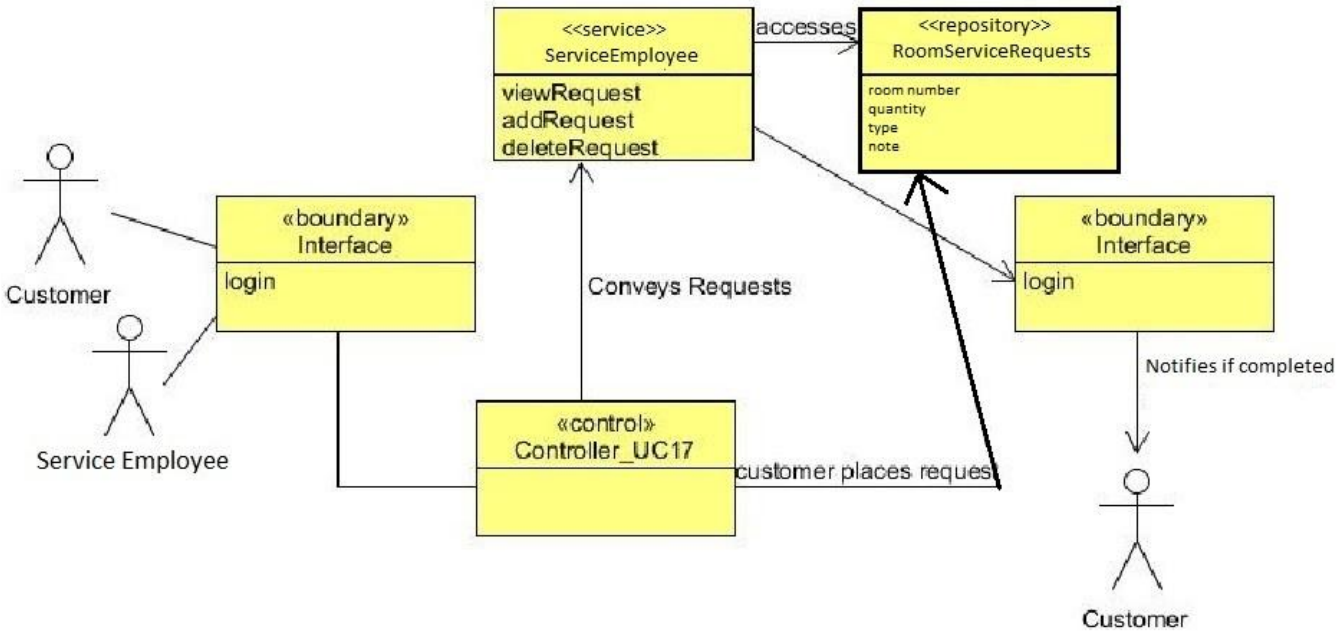
6.6.5 Domain Model

Housekeeping



This is the Domain Model for the Housekeeping Subsystem. It shows how the Customer and Housekeeper interact with Cloud Surf Inn to successfully clean the customer's room.

Room Service



This is the Domain Model for the Room Service Subsystem. It shows how the Customer and Service Employee interact with Cloud Surf Inn to successfully provide the customer with room service..

The domain model is where all of the concepts, attributes, and attribute associations are illustrated pictoriorally. There are two main actors in both the housekeeping and the room service systems: hotel staff (housekeepers and service employees) and customers. The controller is what designates what occurs within the housekeeping and room service subsystems. The housekeeping and room service subsystems both have a node they create to store for use in the system's corresponding data structures (either a heap or a queue). There is no hardware that the housekeeping and room service systems interact with.

6.6.6 System Operation Contracts

Operation	Maids View Housekeeping Request
Use Case:	UC-25
Preconditions:	➔ Customer(s) must have made a housekeeping request. ➔ Housekeeper must be logged in and have clocked-in
Postconditions:	➔ Housekeepers will know the time when they must clean the customer's room. ➔ Housekeepers will be able to accept a request that is within an available time slot. ➔ Housekeepers will notify customers when cleaning is done.

Operation	Ordering Services
Use Case:	UC-19
Preconditions:	➔ Customer(s) must have made a room service request and currently staying in the room. ➔ Service Employee must be logged in and have clocked-in
Postconditions:	➔ Service Employees will be able to view all requests. ➔ Service Employees will notify customers when the service is in progress and when the service is done.

6.6.7 Data Model & Persistent Data Storage

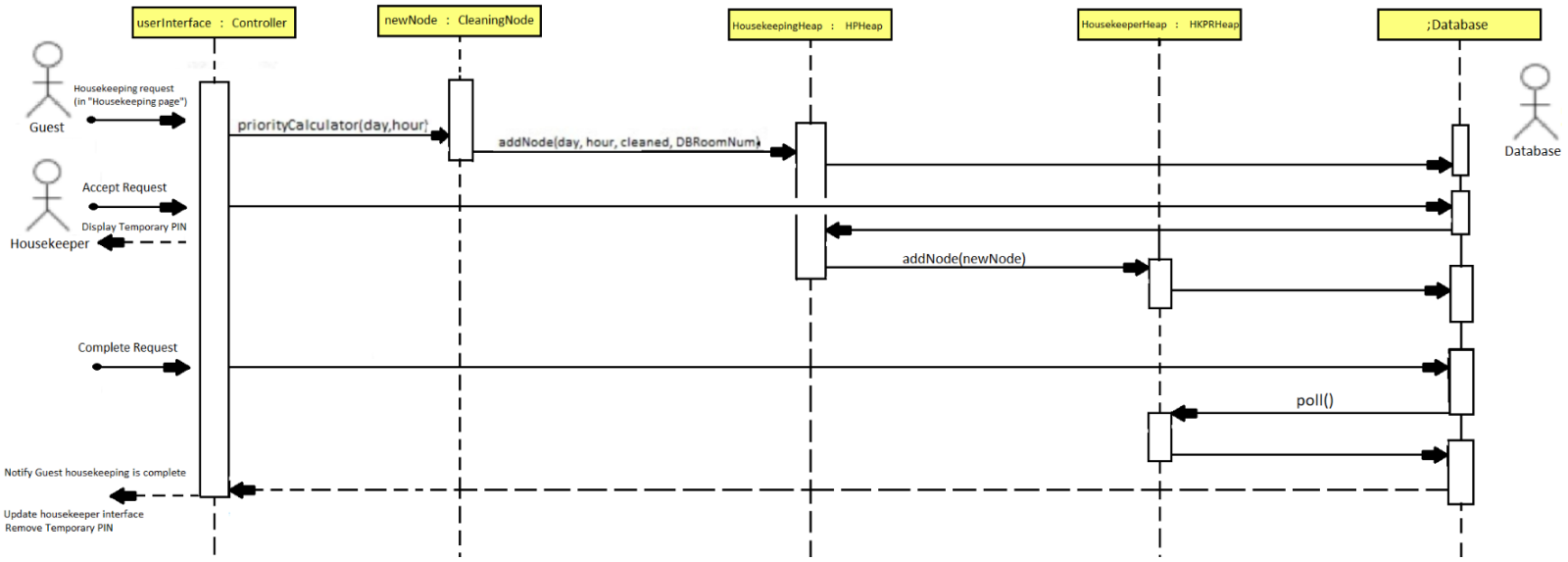
The housekeeping and room service systems do not have a data model because of the nature of the CleaningNode and ServiceNode classes that store all information relevant to their respective subsystems. Persistent data storage is managed in the Cloud Surf Inn database where the HKPRHeap, HPHeap, ServiceQueue, and their respective nodes are maintained and managed.

6.6.8 Mathematical Model

No mathematical models are utilized in either the Housekeeping or Room Service subsystems.

6.7 Interaction Diagrams

Housekeeping:

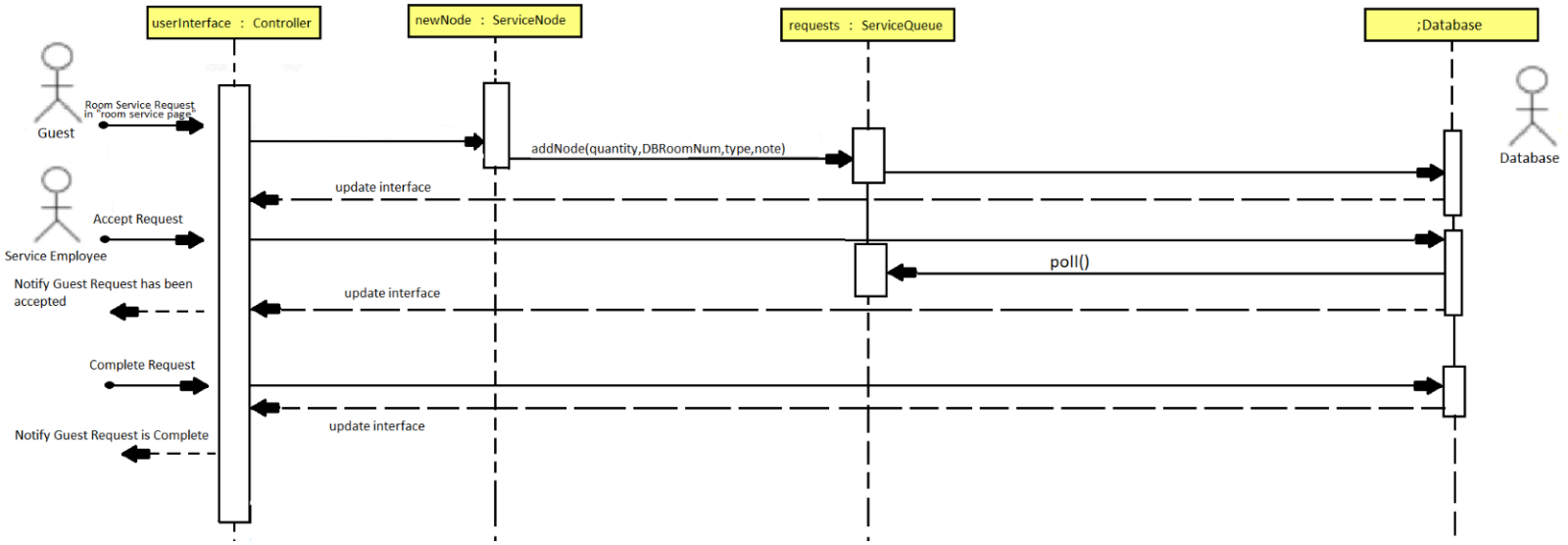


This is the revised Interaction Diagram for the Housekeeping Subsystem. It is more detailed than the previous version in addition to specifying the specific results that the actors witness while interacting with the system.

This figure represents the interaction diagram for the Housekeeping system. After logging in, the customer will request housekeeping which then initiates the process by having the controller add a CleaningNode to the HPHeap with the information the customer provided. After this occurs, the HPHeap will use the priorityCalculator from the CleaningNode class to calculate the priority the request has before updating the priority of the CleaningNode within the HPHeap.

Once the housekeeper accepts the next housekeeping request, the HKPRHeap will initiate the process of transferring the accepted CleaningNode from the HPHeap. The HPHeap will have the CleaningNode removed from it, and that removed CleaningNode will be added to the HKPRHeap. After the housekeeper indicates that the housekeeping service has been fulfilled, the HKPRHeap will then indicate the room has been cleaned. This interaction diagram is derived from the system sequence diagram for UC-25.

Room Service:



This is the revised Interaction Diagram for the Room Service Subsystem. It is more detailed than the previous version in addition to specifying the specific results that the actors witness while interacting with the system.

This figure represents the interaction diagram for the Room Service system. After logging in, the customer will request room service for an available item which then initiates the process by having the controller add a ServiceNode to the ServiceQueue with the information the customer provided. After this occurs, the newly created request will appear at the top of the ServiceQueue (because of its FIFO ordering). Once the service employee accepts the room service request, the ServiceQueue will remove the ServiceNode from itself and the information will be displayed in the interface for the service employee to see .When this occurs, the system will automatically notify the customer that their request has been accepted. After the housekeeper indicates that the room service request has been fulfilled, the system will

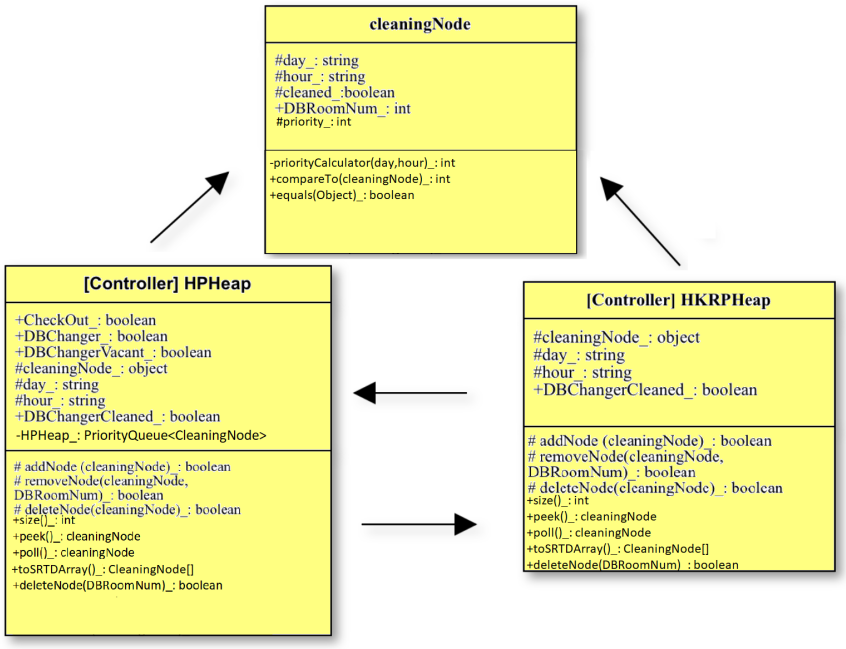
automatically notify the guest that their request is on its way. This interaction diagram is derived from the system sequence diagram for UC-2.

6.8 Class Diagram

Housekeeping:

Class	Meaning/Use of Class	Associations
CleaningNode	Each request will be a cleaning node with a day and an hour.	Housekeeping Priority Queue Housekeeper's Priority Queue
HPHeap	Tracks the cleaning service requests that are available to be accepted.	Interface Cleaning Node Housekeeper's Priority Queue
HKPRHeap	Tracks the cleaning services a housekeeper has accepted and will fulfill.	Interface Cleaning Node Housekeeping Priority Queue

The HPHeap and HKPRHeap classes both utilize the CleaningNode class to perform their operations. The HPHeap and HKPRHeap classes also interact with each other to transfer CleaningNodes, delete CleaningNodes, and create new CleaningNodes. All three classes rely on the controller and each other to acquire their values (day, hour, etc.).

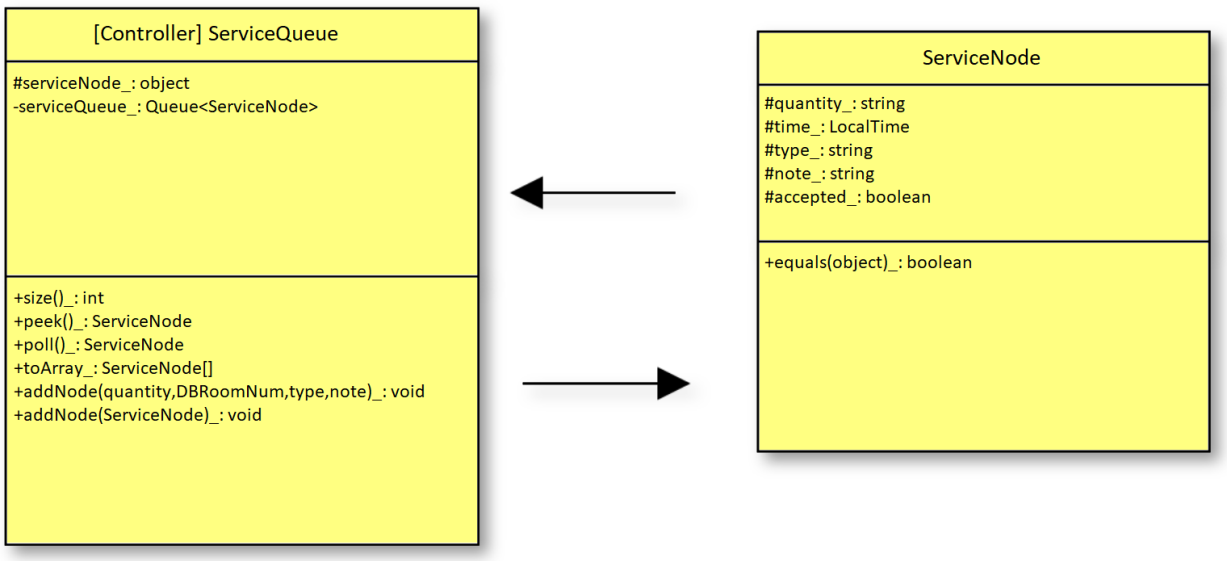


This is the revised Class Diagram for the Housekeeping Subsystem. It includes the new methods and variables that each class utilizes and manipulates to provide the functionality that the Housekeeping System has as of Demo #2.

Room Service:

Class	Meaning/Use of Class	Associations
ServiceNode	Each request will be a service node with a service type, quantity, time of request, an optional note, and whether or not the request has been accepted.	Service Queue
Service Queue	Tracks the room service requests that are available to be accepted.	Interface Service Node

The ServiceQueue class utilizes the ServiceNode class to perform its operations. The ServiceQueue class has the ability to create new ServiceNodes. The two classes rely on the controller and each other to acquire their values (time, item, quantity, etc.).



This is the Class Diagram for the newly-implemented Room Service Subsystem. This system was introduced during Demo #2. For the future of Cloud Surf Inn, this subsystem will undergo many changes to provide more business value for customers and staff members alike.

6.9 Algorithms and Data Structures

Housekeeping:

Algorithms

The Housekeeping system implements several algorithms to allow it to function correctly and efficiently. The HPHeap and HKPRHeap priority queues implement Java's built-in priority queue class. Additionally, the system also uses Java's time class to derive the priority used by the priority queues. Using Java's native priority queue gives the housekeeping system a means of organizing itself such that urgent requests are always at the head of the priority queue. This is because Java's priority queue classes utilizes the "natural ordering" of its constituent members to organize itself. For this particular system, a min-heap (elements with lower priority are at the top of the heap) is desirable because the most urgent requests will have a low priority value because the priority value is calculated based on the current time with the priorityCalculator function. Moreover, to compare priorities, the CleaningNodes are comparable to each other in order to establish the "natural ordering" the HPHeap and HKPRHeaps utilize to organize themselves in order.

Data Structures

The Housekeeping System uses the Priority Queue data structure (Java's implementation) to provide the framework for the HPHeaps and HKPRHeaps. Many of the HPHeaps and HKPRHeaps methods are inherited directly from the Priority Queue Class that Java includes.

Room Service:

Algorithms

The Room Service subsystem is extremely similar to the Housekeeping subsystem. The Room Service subsystem is simpler than the Housekeeping subsystem and this is directly reflected in its algorithmic design. Unlike the Housekeeping subsystem, Room Service does not

need a priority because the Room Service subsystem incorporates a Queue which has no priority to establish a more ornate ordering. Thus, the ServiceQueue implements Java's built-in Queue class because a FIFO (first-in, first-out) ordering is all that is necessary to have a robust room service system. Of course, as the Room Service subsystem evolves and becomes more complex, a more complex data structure (like a Priority Queue) may prove to be more useful. Thanks to the modular implementation of the Room Service system, large updates like these are easy to incorporate and test. Once again, Java's Time class is also utilized to acquire the time of the request for use in showing the request time in the service employee's user interface. Apart from this, the Room Service Subsystem does not have any algorithms to process any data in its current iteration. However, this will likely change in the future as the Room Service becomes more intricate and has more features added. Ordering things like food will require more information from the customer and also information that is provided by the hotel itself and fed into the Cloud Surf Inn application.

Data Structures

The Room Service System uses the Queue data structure (Java's implementation) to provide the framework for the ServiceQueue class. Many of the ServiceQueue's methods are inherited directly from the Queue Class that Java includes. Once again, a Queue was appealing to use because of its FIFO ordering. The added complexity of a PriorityQueue is unnecessary for the Room Service system's current offerings, but this can change in the future.

6.10 Design of Tests

Housekeeping:

Use Case	Test Case Description	Test Coverage
UC-16	The customer will login and navigate to the housekeeping page. The customer will then select the day of the week and hour they wish to schedule housekeeping and click the "Submit" button.	This test will ensure that the housekeeping request will be visible by the housekeepers.
UC-25	The housekeeper will login with the special username and password given to them. The housekeeper will then navigate to the housekeeping tab where both the HPHeap and HPKRHeap will be present.	This test will ensure the housekeeper is able to view all the requests given to them by the customers.
UC-26	The housekeeper must take all steps provided in	This test will ensure the

	UC-25, then the housekeeper will be able to click a button to indicate that the highest priority cleaning job to their specific housekeeper heap is complete. This will then remove the task from the HPHeap and inject it into the specific housekeepers HPKRHeap.	housekeeper can accept jobs given to them by the customers.

Room Service (Showing only the use case demonstrated in Demo 2):

Use Case	Test Case Description	Test Coverage
UC-19	- The customer will login and navigate to the "Room Service" page. The customer will then select the item they wish to receive, the quantity, and add an additional note for the service staff to see. The customer will then click the "Submit" button. This step can be repeated as many times as desired. - The service employee will login and navigate to the "Staff" page. The service employee will then be able to see the newly issued service request(s) in the "Incoming Service Requests" text box. They will then accept a request (as many as desired) and see the accepted request(s) in the "Ongoing Service Requests" text box along with all of the information related to the request(s). The service employee will then click the "Complete Request" button to mark the request(s) as completed. - The customer can login once again and see the updates for their request in the "Notifications:" line in the textbox at the bottom of the "Log In" page.	This test will ensure that the service request will be visible by the service staff after being issued by a customer. This test will also show that the details of the request can be seen. This test has the additional benefit of demonstrating the notification system that the Room Service subsystem relies on for communication between staff and customers.

6.11 Effort Estimation using Use Case Points

Housekeeping and Room Service:

Unadjusted Actor Weight (UAW):

Actor	Description	Complexity	Weight
Guest	Guests interact with the system via a graphical user interface.	Complex	3

Staff (Housekeeper, Service)	Staff members interact with the system via a graphical user interface.	Complex	3
Database	Database interacts with our system through a protocol.	Average	2

Unadjusted actor weight comes to: 3+3+2 = 8

Unadjusted Use Case Points (UUCW):

Use Case	Description	Category	Weight
Housekeeping Request (UC-16)	Simple user interface. 3 participating actors (housekeeper, database, and customer). Up to 3 steps.	Average	10
View Services (UC-17)	Moderate interface design. 3 participating actors (service employee, database, and customer). Up to 3 steps.	Average	10
Ordering Services (UC-19)	Moderate interface design. 3 participating actors (service employee, database, and customer). Up to 3 steps.	Average	10
Notify Guest Housekeeping is Complete (UC-21)	Simple user interface. 2 participating actors (housekeeper, database). Up to 2 steps.	Simple	5
View Services Menu (UC-23)	Simple user interface. 2 participating actors (customer, database). Up to 2 steps.	Simple	5
Service Status Update (UC-24)	Simple user interface. 2 participating actors (service employee, database). Up to 2 steps.	Simple	5
Housekeepers View Housekeeping Request (UC-25)	Moderate interface design. 2 participating actors (housekeeper and database). Up to 4 steps.	Average	10
Housekeepers Update System (UC-26)	Very Simple interface design. 2 participating actors (housekeeper and database). 1 step.	Very Simple	2

Use Case Weight Total	57

Unadjusted Use Case Weight = 4*10 + 3*5+2=57

Technical Complexity Factor (TCF):

Technical Factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight*Perceived Complexity)
T1	System is not distributed.	2	0	2*0=0
T2	Users expect good performance but no critical response times or throughput.	1	3	1*3=3
T3	Users expect high efficiency but it is not exceptionally critical.	1	3	1*3=3
T4	Internal processing is relatively simple.	1	1	1*1=1
T5	System should be able to be reused in other hotels.	1	5	1*5=5
T6	Software needs to be easily installed for different hotels.	0.5	5	0.5*5=2.5
T7	Ease of use is very important.	0.5	5	0.5*5=2.5
T8	No portability concerns.	2	0	2*0=0
T9	System should be able to handle new features but is unlikely to need to do so.	1	2	1*2=2
T10	Concurrent use is required.	1	4	1*4=4
T11	Security is a significant concern.	1	5	1*5=5
T12	No direct access for third parties.	1	0	1*0=0
T13	No unique training needs.	1	0	1*0=0

Technical Factor Total:	28

Technical complexity factor comes to 0.6+0.01*28=0.88

Environmental Complexity Factor (ECF):

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor (Weight*Perceived Impact)
E1	Beginner familiarity with UML-based development.	1.5	1	1.5*1=1.5
E2	Little familiarity with application problem.	0.5	1	0.5*1=0.5
E3	Some knowledge of object-oriented approach.	1	2	1*2=2
E4	Beginner lead analyst.	0.5	1	0.5*1=0.5
E5	Motivated with some team members lacking.	1	4	1*4=4
E6	Stable requirements expected.	2	5	2*5=10
E7	No part-time staff.	-1	0	-1*0=0
E8	Programming language of average difficulty will be used.	-1	3	-1*3=-3
			Environmental Factor Total:	15.5

Environmental complexity factor comes to 1.4-0.03*15.5= 0.935

Use Case Points and Duration:

UCP = UUCP * TCF * ECF = (57 + 8) * 0.88 * 0.935 = 53.482 = 53 use case points.

Duration = UCP * PF = 53 * 28 = 1,484

6.12 Future and Current Work

Housekeeping:

There is a great deal of future work for the Housekeeping system. The Housekeeping System's future innovation revolves around improving usability as well as introducing more automated functionality that emphasizes the utility of Cloud Surf Inn. First and foremost, the Cloud Surf Inn System has user concurrency as an important future that will eventually be implemented so that multiple users (housekeepers and guests for example) can interact with the system at the same time. This means that the HPHeap and HKPRHeap classes will need to have some sort of synchronization protocol to prevent complications when multiple users request housekeeping and multiple housekeepers edit the HPHeap. Measures to prevent issues with concurrency will be very important for the Housekeeping system as Cloud Surf Inn continues to evolve. Another crucial feature that will greatly improve the usability of the Housekeeping system is the advent of a calendar system to solicit and view requests. Having a visual means of viewing and issuing requests (with something like Google Calendar as inspiration) will allow housekeepers and guests alike to pictorially see their schedule rather than view it in a text box. The calendar solution will be a more elegant means of displaying requests. Similarly, the Housekeeping system will also benefit from having a map of the hotel be visible to the housekeepers so they can view where rooms are located as well as where the hotel's high traffic areas are located. Another minor feature that the housekeeping system could use is the ability to automatically request housekeeping for rooms that have been recently marked vacant. This way, whenever a customer vacates the hotel and their room, Cloud Surf Inn automatically creates a housekeeping request for the newly vacated room to be properly disinfected and prepared for the next guest. An interesting feature to enhance this system could be a "Demand" parameter that is shared across all of Cloud Surf Inn. Demand could be a value that varies based on the activity occurring within the hotel. Whenever there are many guests checking in and out of cloud surf inn, or the system detects many people moving around hallways and other hall traffic areas, the Demand value can rise to indicate that there is more activity occurring within the system so tolerances must be tightened to ensure that everything continues to run smoothly. In the case of the housekeeping system, this "tightening" could be reflected in the automatic housekeeping requests being given a low priority and thus making the requests more important so that

housekeepers can quickly disinfect rooms and open them up for new guests. In the event that the Demand value is low (indicating that there is less activity than usual) Cloud Surf Inn can relax its automated processes. For housekeeping, the automated requests can be given a higher priority which allow housekeepers to have more time to disinfect rooms so that new guests can occupy them.

Room Service:

The Room Service subsystem will share many of the new features that the Housekeeping subsystem will receive as a part of Cloud Surf Inn's future work. Just as with Housekeeping, the Room Service subsystem must have protocols and protections in place to protect the ServiceQueue once Cloud Surf Inn allows for multiple users to utilize the system simultaneously. For example, cases where two service employees accept a request at the same time must be handled to avoid issues with the ServiceQueue (which is a data structure that can be viewed and manipulated by all service staff). Another important feature that the Room Service subsystem will have is the ability to have the manager add new services, especially food and beverage services. This feature will make the Room Service system modular and allow each hotel to offer services that cater to its offerings. However, with food and beverages, the Room Service system will also need the ability to accommodate for canceled requests and allow service staff to reject requests. The Room Service system will also benefit from having properly developed notification and messaging systems. The notification system will eventually neatly detail all notifications a guest may have, including ones that have been acknowledged in the past. A formal messaging system for the guest with hotel staff can allow customers to directly message service staff for updates or additional details regarding their service request. These new features will make the Room Service system more powerful for a hotel utilizing Cloud Surf Inn while simultaneously improving the user experience guests and staff have while interacting with the system and its features.

Current Work:

Team A has completed all use cases developed for the second demo, fully integrated the subsystem into the full system, and is running additional test cases to ensure the correct functionality of the Room Service and Housekeeping subsystems even with edge cases.

7. Subsystem: Database, Room Matcher, Payment

7.1 Summary of Changes

1. The Room matcher function now finds the maximum number of beds offered in the given hotel database. If the customer attempts to enter a number above this maximum, a pop-up will appear notifying them they have exceeded the maximum number of beds for this hotel.
2. Upon matching the customer to a room, the customer must now input the outputted room number and click “Book this room”.
3. The payment tab has been fully implemented. Any information a user inputs will be saved under their entry in the customer database upon the user clicking “done”.
4. Upon opening the application, users can now choose to create an account within our system. All of the customer’s information such as username, password, assigned room, and room pin, will be stored in a user profile database.
5. The GUI will now have a button labeled “View Room Database”, appearing on the “Book a room”, “Manager”, and “Staff” pages. This button will allow the user to view all rooms in the database and their associated data. The user would use this to choose a room to book, whilst managers and staff would use it to gain needed information about each room.

7.2 Glossary of Terms

SQLiteDriver: This is the main class that is used to create and manipulate the hotel database that is used within our system. The class not only initializes the database with all its entries but also connects the database with all of the other parts of our system. JDBC (Java Database Connectivity) is used within the class in order to make our overall system and databases communicate with each other.

SQLiteCustomer: This is the main class that is used to create and manipulate the customer database that is used within our system. This database is structured in almost the exact same way as our hotel database. It uses JDBC in the same way as SQLiteDriver and also can be

manipulated with functions similar to those contained in SQLiteDriver. The only difference between these two databases is the entries that they contain.

viewDatabase: The viewDatabase class within our subsystem is used by the customer in order to view a list of all rooms within a hotel. Upon pressing a button, this class will be created and a pop-up list will be displayed via a call to a function within the class. The list will show all current rooms in the hotel along with all their respective information.

7. 3 System Requirements

7.3.1 Enumerated Functional Requirements

Room Matcher:

Identifier	Priority	Description
REQ-17	5	The system has a database of rooms containing occupancy, room number, cleanliness.
REQ-18	4	The system matches rooms to guests based on preferences.
REQ-19	4	The system allows customers to individually choose from available rooms and opt-out of the room matcher.
REQ-20	1	The system will generate a key for the guest.

Payment:

Identifier	Priority	Description
REQ-16	2	The system allows customer to input credit card number, holder, expiration date, and CVC

Enumerated Nonfunctional Requirements

None.

7.4 Functional Requirement Specification

7.4.1 Stakeholders

Hotel Owners: Ensuring our businesses stay up to date with current technology by enabling the customer to book entirely online. Face-to-face meetings are discouraged with COVID-19, and this system will allow for less of them.

Hotel Guests: Cloud Surf Inn will allow for a seamless booking experience in which the customer is able to easily find and book a room.

Software Engineers: Ensuring this system is easily updatable, conforms to industry standards, and is viable for any hotel.

7.4.2 Actors and Goals

Actor	Goal
Manager	The manager's goal is to ensure customers are happy with their stay and that staff are doing their jobs.
Staff	The staff are employees such as housekeepers or chefs, who perform roles to keep the hotel functioning.

Guest/Customer	The guest is a customer staying at the hotel; they use the Cloud Surf Inn system to satisfy their desires and necessities. Through the application, they wish to be able to view every room, find a room, and to pay for rooms and services.

7.4.3 Use Case Casual Description

Room Matcher:

Identifier	Description	REQ-# Used
UC-1	"Find Room" - Matches customer with room based on checked preferences and guest number input.	REQ-3, REQ-17, REQ-18, REQ-32
UC-2	"View Room Database" - Allows staff and managers to view all rooms and their information.	REQ-13, REQ-15, REQ-17
UC-5	"Book a Room" - Allows a potential guest to select a room of their choice	REQ-19, REQ-17

Payment:

Identifier	Description	REQ-# Used
UC-3	"Payment" - Allows customers to input payment information and charge things to the room.	REQ-31, REQ-16, REQ-10, REQ-2

7.4.4 Use Case Diagram

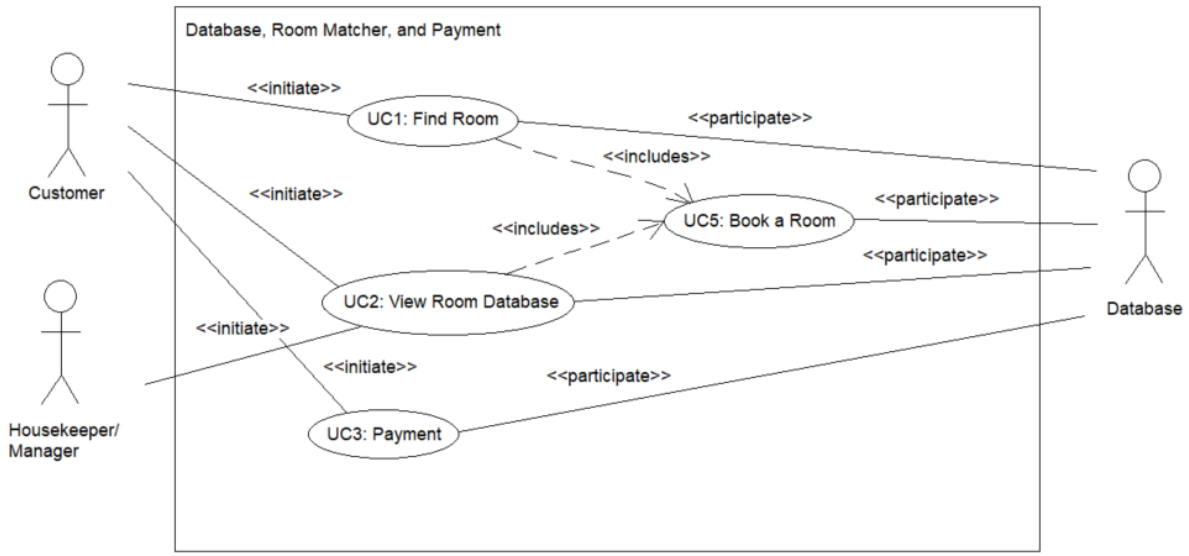


Figure 7.4.4: Use Case Diagram

The use case diagram above is for the database, room matcher, and payment subsystems. If a customer wishes to book a room, he/she can do so by either initiating UC1 or UC2. UC1 will utilize our room matcher subsystem which automatically assigns a customer a room based on his/her preferences. UC2 will provide a pop-up list of all the current hotel rooms along with their respective information. Either of these use cases will then allow a customer to access UC5 which books a customer to their desired room. UC5 will only book the room for the customer if they don't have a room beforehand and if their desired room is vacant. Once a customer has booked their room, the vacancy of the room will be updated in our system's database. Finally, a customer will have to initiate UC3 in order to input their payment information into the system. This can be done via our user interface which includes a payment tab that allows customers to input payment information. After payment information is successfully inputted, the user can then pay for whatever room they booked and make their booking final.

7.4.5 Fully Dressed Descriptions

UC-1: Find Room

Related Requirements: REQ-1, REQ-17, REQ-23
Initiating Actors: Customer
Actor Goal: To find a room that best accommodates his/her needs.
Participating Actors: N/A
PreCondition: The customer has a Cloud Surf Inn account.
PostCondition: The customer has an assigned room and selects “book room”
Minimum Guarantee: A guest will be matched to a room based on a preference selection.
Success Guarantee: A guest inputs their preferences and is matched to a room(s) and is able to pick a room from those selected.
Flow Of Events for Success 1. → The Customer signs in using the login interface. 2. → The customer chooses the “Room Matcher” tab. 3. → The customer inputs the dates of booking. 4. → The customer inputs the number of guests. 5. → The customer selects each amenity offered that they require. 6. → The customer selects “Find Room” 7. ← An available room that matches the preferences of the customer is selected through an algorithm. 8. → The customer is shown the room they are matched with, and allowed to select “Book This Room” or “No Thanks” 9. → The customer selects “Book This Room” When shown the matched room’s information. 10. ← The system prompts the customer to input payment information. 11. → The customer inputs their payment information. 12. → The selected room is assigned to the customer for the selected booking period.

Alternate Flow of Events

1. → The customer signs into the application.
2. → The customer chooses the “Room Matcher” tab.
3. → The customer clicks “Book a room”
4. → The customer chooses a room to view.
5. → The customer inputs booking dates.
6. → The customer clicks “Book This Room”
7. ← The system prompts the customer to input payment information.
8. → The customer inputs their payment information.
9. → The selected room is assigned to the customer for the selected booking period.

UC-2: View Room Database
Related Requirements: REQ-13, REQ-15, REQ-17
Initiating Actors: Customer, Staff, Manager
Actor Goal: To view the data about all rooms in the database.
Participating Actors: N/A
PreCondition: The customer has a Cloud Surf Inn account.
PostCondition: The customer is displayed the room database.
Minimum Guarantee: Each room in the database and its data will be visible.
Success Guarantee: The user is presented the room database in a neat and formatted way, able to be expanded to fit many rooms.
Flow Of Events for Success 1. → User signs in using the login interface. 2. → The user chooses the “view room database” tab.

3. → The user is displayed each room in the database. Clicking on the room displays its data.

UC-3: Payment
Related Requirements: REQ-31, REQ-16, REQ-10, REQ-2
Initiating Actors: Controller
Actor Goal: To receive payment data from user.
Participating Actors: Customer
PreCondition: The customer has clicked the “book room” button.
PostCondition: The customer’s payment information is stored in the database.
Minimum Guarantee: the customer’s payment information will be stored for the session.
Success Guarantee: The customer’s payment information is stored for the existence of their account.
Flow Of Events for Success 1. → The Customer signs in using the login interface. 2. → The customer chooses the “Room Matcher” tab. 3. → The customer inputs the dates of booking. 4. → The customer inputs the number of guests. 5. → The customer selects each amenity offered that they require. 6. → The customer selects “Find Room” 7. ← An available room that matches the preferences of the customer is selected through an algorithm. 8. → The customer is shown the room they are matched with, and allowed to select “Book This Room” or “No Thanks”

9. → The customer selects “Book This Room” When shown the matched room’s information.
10. ← The system prompts the customer to input payment information.
11. → The customer inputs their payment information.
12. → The payment information is stored in the database under their email.

Alternate Flow of events for success

1. → The customer signs into the application.
2. → The customer chooses the “Room Matcher” tab.
3. → The customer clicks “Book a room”
4. → The customer chooses a room to view.
5. → The customer inputs booking dates.
6. → The customer clicks “Book This Room”
7. ← The system prompts the customer to input payment information.
8. → The customer inputs their payment information.
9. → The payment information is stored in the database under their email.

UC-5: Book a Room
Related Requirements: REQ-31, REQ-16, REQ-10, REQ-2
Initiating Actors: Customer
Actor Goal: To choose a room to book from the database.
Participating Actors: N/A
PreCondition: The customer log ged into the system.

PostCondition: The customer has booked a room.
Minimum Guarantee: The customer will be able to book a room.
Success Guarantee: The customer will be able to book a room for a selected period of time.
Flow Of Events for Success 1. → The customer signs into the application. 2. → The customer chooses the “Room Matcher” tab. 3. → The customer clicks “Book a room” 4. → The customer chooses a room to view. 5. → The customer inputs booking dates. 6. → The customer clicks “Book This Room” 7. ← The system prompts the customer to input payment information. 8. → The customer inputs their payment information. 9. → The selected room is assigned to the customer for the selected booking period.

7.4.6 System Sequence Diagram

UC-1 (Find Room):

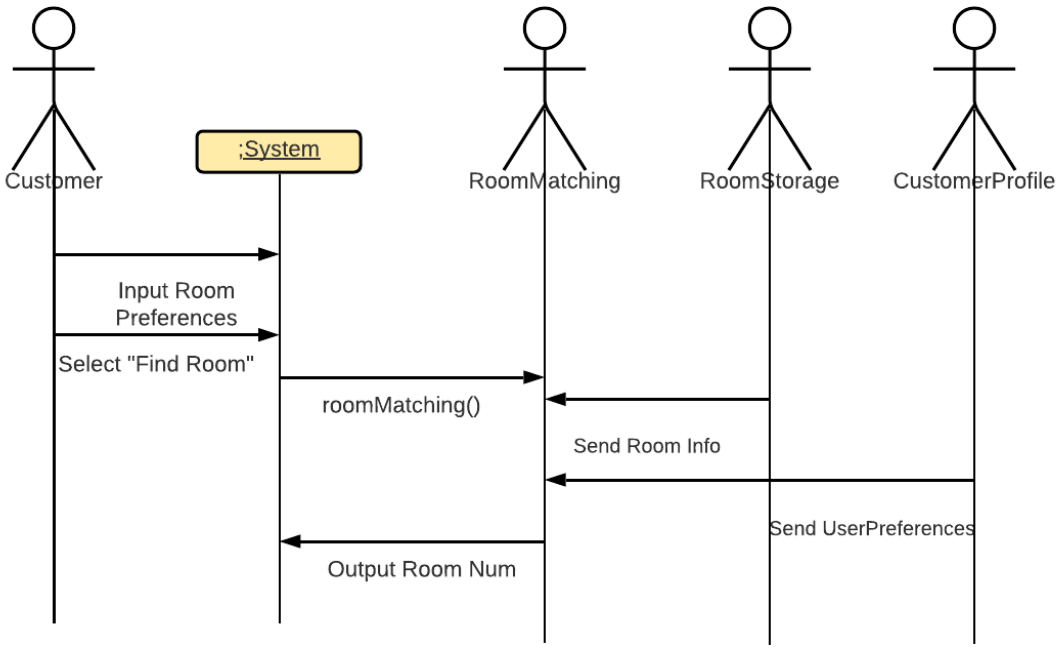


Figure 7.4.6.1: UC-1 System Sequence Diagram

Above is the system sequence diagram for the room matcher subsystem. It details the sequence that UC-1 goes through in order to operate correctly. The customer inputs their preferences and then proceeds to select the “Find Room” button. Our system will then utilize the room matching function which communicates with the hotel database. This communication will allow for a pairing to be made between a customer and their desired room. Once this pairing is made the system will output the room number of the desired room and assign the room to the customer.

UC-2 (View Room Database):

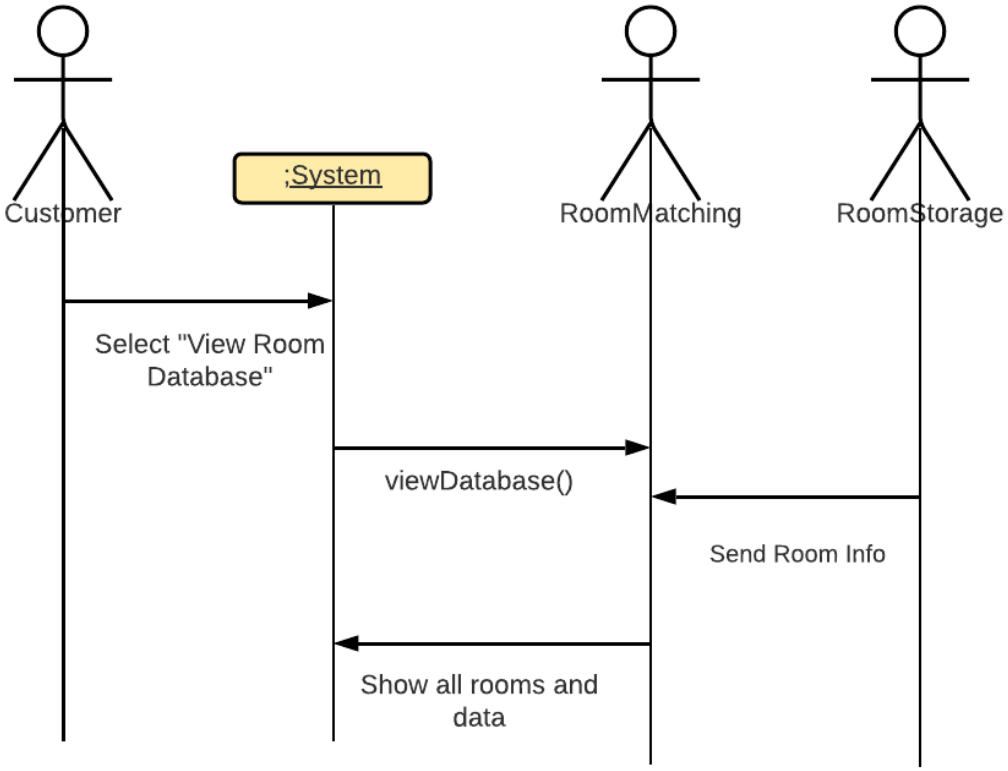


Figure 7.4.6.2: UC-2 System Sequence Diagram

Above is the system sequence diagram for the view room database subsystem. It details the sequence that UC-2 goes through in order to operate correctly. The customer starts by selecting the “View Room Database” button. Our system will then proceed to make communications with our database in order to retrieve hotel room information. After these communications are made, the system will then display a pop-up list with all the hotel rooms and their respective information.

UC-3 (Payment):

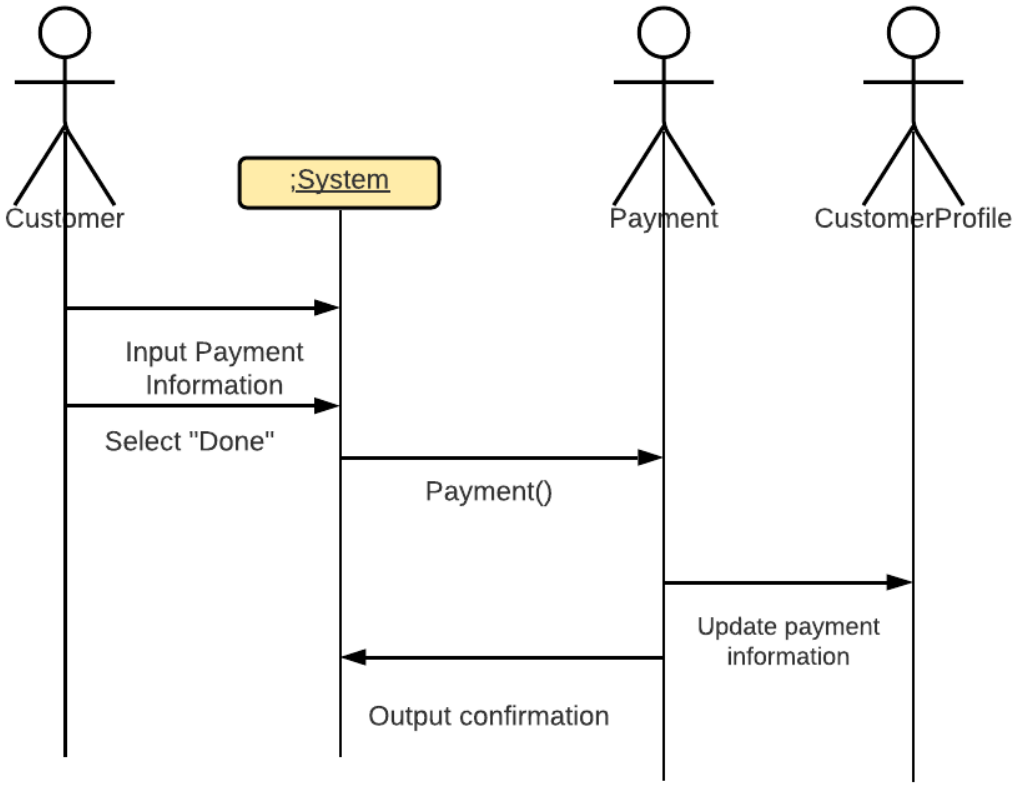


Figure 7.4.6.3: UC-3 System Sequence Diagram

Above is the system sequence diagram for the payment subsystem. It details the sequence that UC-3 goes through in order to operate correctly. A customer will begin by going to the payment tab located within our system's GUI. Once there, a customer will be able to input their payment information and select the "done" button. After this button is clicked, our system will make a connection to the integrated database which contains the payment information of all customers. This connection will allow for the inputted payment information to be stored in the database where it will be kept in a solid state. A customer can then use this stored payment information to pay for their booked room.

UC-5 (Book a Room):

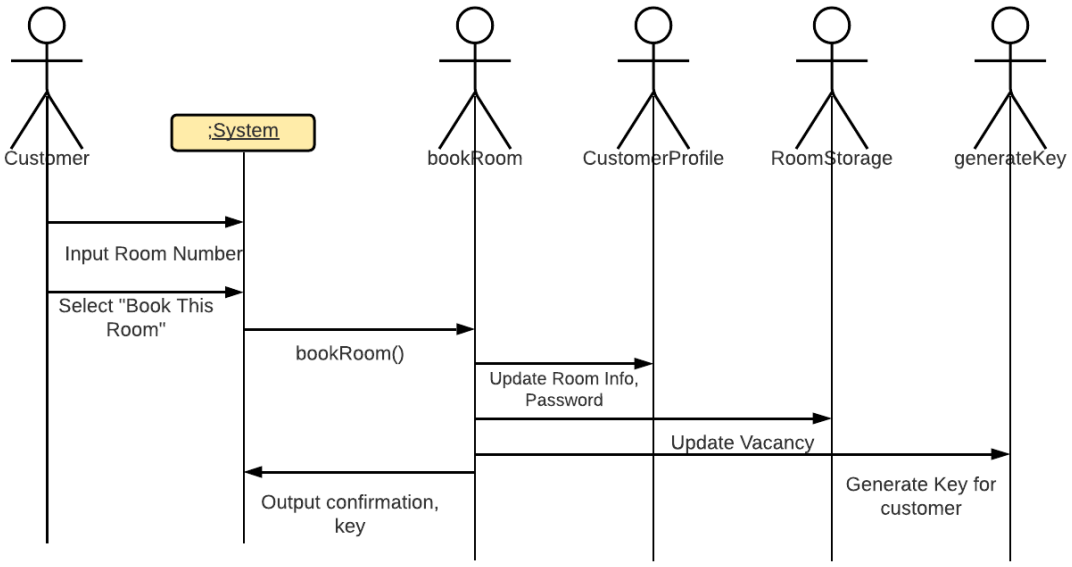


Figure 7.4.6.4: UC-5 System Sequence Diagram

Above is the system sequence diagram for the payment subsystem. It details the sequence that UC-5 goes through in order to operate correctly. A customer will begin by inputting the room number of the room they wish to book. The customer will know this room number before accessing UC-5 as they will obtain it from either UC-1 or UC-2. Once the desired room number is inputted and the “Book This Room” button is clicked, our system will make a connection with the hotel database and a randomly generated pin will be given to the customer. The connection with the hotel database exists so the vacancy of the room will be updated to reflect whether or not the room is booked. Finally, the customer’s pin will also be saved in the database using the same connection.

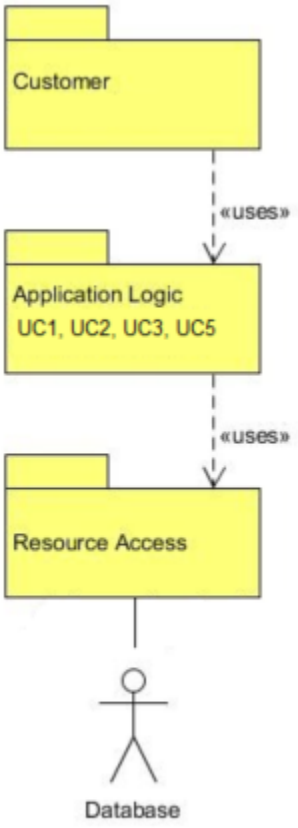
7.4.7 Traceability Matrix

	REQ-16	REQ-17	REQ-18	REQ-19	REQ-20
UC-1	X	X	X		
UC-2	X	X		X	

UC-3	X				
UC-5		X		X	X

7.5 System Architecture and System Design

7.5.1 Identifying Subsystems



This is the Subsystem diagram for the Database, Room Matcher, and Payment subsystems. The customer interacts with the system to exchange information with the database.

7.5.2 Architecture Styles

As mentioned above, our first architectural style was a layered architecture. The presentation layer accesses the resources layer, in order to allow each system to be edited separately. In the case of the payment system and matching system, the application must connect to the resource layer to access the database and perform its function, then give the necessary information to the presentation layer.

The client server model would also be used in order to allow for clients to all access the application at the same time. This way, the database would be held in a central server and edited and accessed remotely by all systems using the client at the time.

7.5.3 Mapping Subsystems to Hardware

All of this subsystem's applications and functions will run on the main client in whatever computer system the customer is using.

7.5.4 Connectors and Network Protocols

This application needs the SQLite extension, and will use java sockets to communicate database information between the client and database.

7.5.5 Global Control Flow

Both the room matcher and payment subsystems are procedure-driven. The payment subsystem can be accessed once the customer logs in, and this is the only necessary procedure before access is allowed. From here, the customer will choose to navigate to the payment tab and input their information.

The room matching and room booking features work the same way - they are procedure driven and only accessible once a customer has logged in. From this point, the customer will choose to navigate to the "Book a Room" tab, and follow procedure to book a room. All functions in our subsystem are procedure activated and require customer input to be called.

7.5.6 Hardware Requirements

Our Cloud Surf Inn application will run on computer systems featuring either macOS or Microsoft Windows operating systems with versions supporting the Eclipse Integrated Programming Environment. For a computer to run Cloud Surf Inn, it must have Java version 1.4.0 or later (the latest version is always preferable). In regards to hardware, most modern desktop and laptop computers have more than enough power, storage, and memory to run Cloud Surf Inn and the database it relies on. Nonetheless, because of Cloud Surf Inn's development in Eclipse, a computer running the system will need a minimum of 512 MB of memory, 300 MB of storage/disk space, and a minimum processor clock speed of 800 MHz.

7.6 Domain Analysis

7.6.1 Concept Definitions

Concept	Responsibility	Type
DB Connection	Coordinate interactions and data exchanges with the database.	Boundary
Customer Profile	Stores all customer information under their username to be accessed and used by other services.	Entity
Matching	Intakes user preferences, and outputs a room number that best matches the user's given preferences.	Service
Controller	Handles and coordinates the actions between the interface and the various use cases that must be completed and delegates work to other concepts.	Control

7.6.2 Association Definitions

Concept Pair	Association Description	Type
Customer Profile ↔Matching	Matching function requires UserPreference data from Customer Profile.	Obtains
Matching ↔ DB Connection	Matching function requires room information in order to match with User Preference data.	Obtains

7.6.3 Attribute Definitions

Concept	**Attributes**	**Definition**
Customer Profile	User Preferences	All of the user's room preferences are stored here.

7.6.4 Traceability Matrix

Concept:	UC-1 (Find Room)	UC-2 (View Room Database)	UC-3 (Payment)	UC-5 (Book a Room)

DB Connection	X	X		X
Customer Profile	X		X	X
Matching	X			X

7.6.5 Domain Model

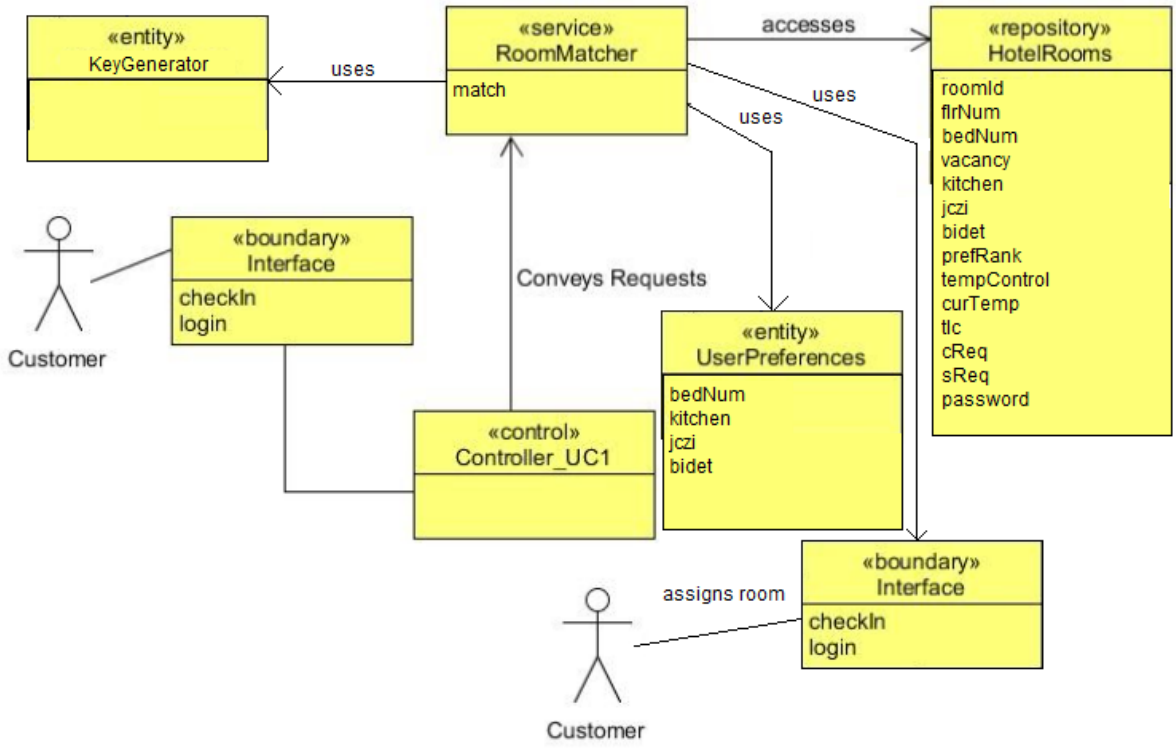


Figure 7.6.5: Domain Model

The domain model above is the culmination of the concepts, attributes, and attribute associations. There is only one main actor with our subsystem, the customer. The customer serves as a boundary for our system, which within the customer interacts with the interface. The interface acts also as a controller which allows the customer to use our RoomMatcher service. This service

accesses our hotel room repository (contained in our database) so it can pull the entity UserPreferences which contains all the necessary information for matching a customer to a room. The RoomMatcher service will then use this UserPreferences entity to pair a customer with a room that most closely matches their desires. Once this pairing is made, the RoomMatcher service will then use the interface as a boundary again in order to assign a room to a customer.

7.6.6 System Operation Contracts

Operation	**Find a Room**
Use Case:	UC-1 (Find Room)
Preconditions:	➔ Customer(s) have a Cloud Surf Inn Account. ➔ Customer(s) have logged into their account.
Postconditions:	➔ Customer(s) are given a room number to fit their needs.

Operation	**View Room Database**
Use Case:	UC-2 (View Room Database)
Preconditions:	➔ User(s) have a Cloud Surf Inn Account. ➔ User(s) have logged into their account.
Postconditions:	➔ User(s) are displayed all rooms in the database.

Operation	**Payment**

Use Case:	UC-2 (Payment)
Preconditions:	➔ Customer(s) have a Cloud Surf Inn Account. ➔ Customer(s) have logged into their account.
Postconditions:	➔ Customer(s) have been assigned a room.

Operation	**Book a Room**
Use Case:	UC-5 (Book Room)
Preconditions:	➔ Customer(s) have a Cloud Surf Inn Account. ➔ Customer(s) have logged into their account.
Postconditions:	➔ Customer(s) have been assigned a room.

7.6.7 Data Model & Persistent Data Storage

All persistent data in our subsystem is stored in two SQLite databases: the Room Storage database and the Customer Profile database. In Room Storage, all of the rooms in the hotel each have an entry. Contained in these entries are necessary information about each room: number of beds, the amenities it has, its floor and room number, and its temperature information. All of this persistent information is stored in this database and kept even when the application is not running. Along with this database, the customer profile is used. In this database, each created customer account has an entry. Each entry contains information about each customer, such as their username and password, user preferences, and room number and key if necessary. All of this data is also persistent and kept even when the application is closed. We keep this information in databases so that it is saved and easily accessible by each participant in our system. Keeping this information between sessions is not vital, and these SQLite databases allow for this.

7.6.8 Math Model

No mathematical models were used in this subsystem.

7.7 Interaction Diagrams

UC-1 (Find Room):

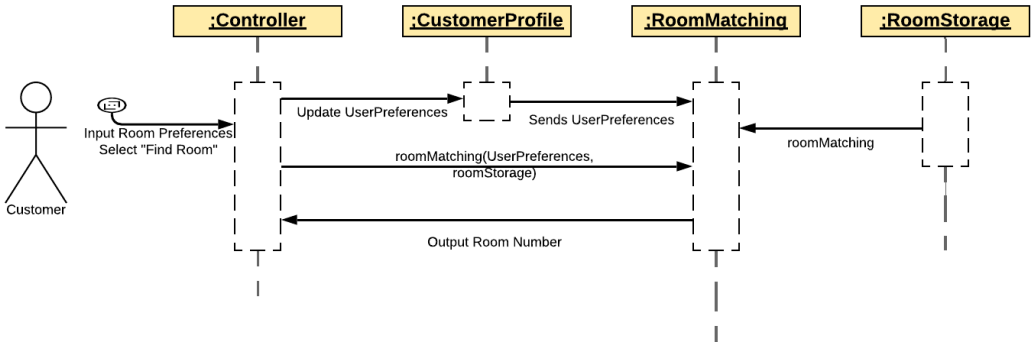


Figure 7.7.1 - This diagram shows what happens in the backend system when a customer wishes to use the room matching system to find a room suitable for their desires. Upon selecting “Find Room”, first their user preferences are updated under their profile. Then, the roomMatching() function is called with inputs to allow it to read the user’s preferences and room database. It then outputs to the controller the best suited room for the given customer.

UC-2 (View Room Database):

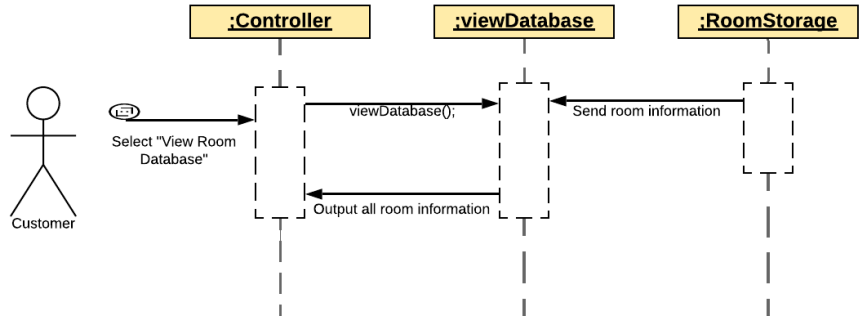


Figure 7.7.2 - viewDatabase is called by the user, then given the required information from roomStorage. It then outputs the room information neatly and visibly for the user.

UC-3 (Payment):

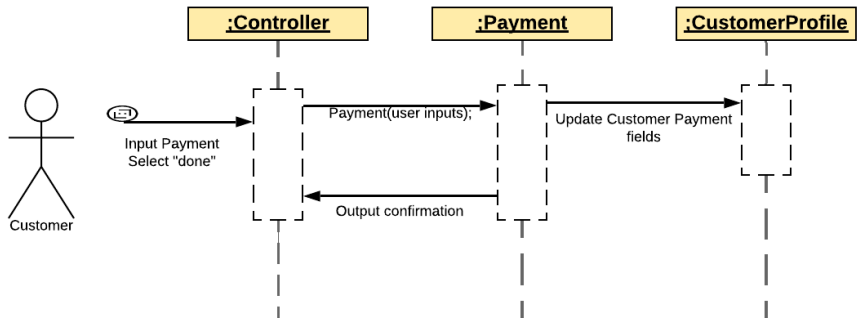


Figure 7.7.3 - The user will click done, prompting the payment function to input all data into the Customer Profile.

UC-5 (Book Room):

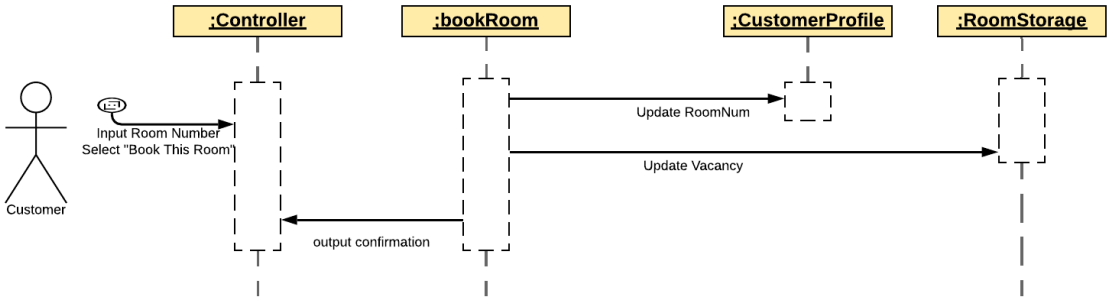


Figure 7.7.4 - Upon selecting the “Book This Room” button, the vacancy status of the room must be updated, and the customer’s profile must be updated to acknowledge that they are staying in the reserved room number.

7.8 Class Diagram

Room Matcher:

Class	Meaning/Use of Class	Associations
RoomMatching	This function matches the given customer's room parameters with a room from the database and outputs that room's number for the customer.	RoomStorage CustomerProfile
CustomerProfile	Stores customer's information and room preferences.	N/A
RoomStorage	A repository that is part of the database that holds information related to a room number, such as the device number of a thermostat.	N/A
Payment	This function intakes the user's inputted preferences and stores them in the CustomerProfile database under the user's entry.	CustomerProfile
bookRoom	This function is called to handle all updates necessary upon a customer booking a room. Currently this includes updating their room number in their profile and changing the vacancy status on the room.	RoomStorage CustomerProfile
viewDatabase	This function handles all necessary actions when the "View Room Database" button is pushed. Currently this includes formatting and outputting	RoomStorage

	all room data from the database.	

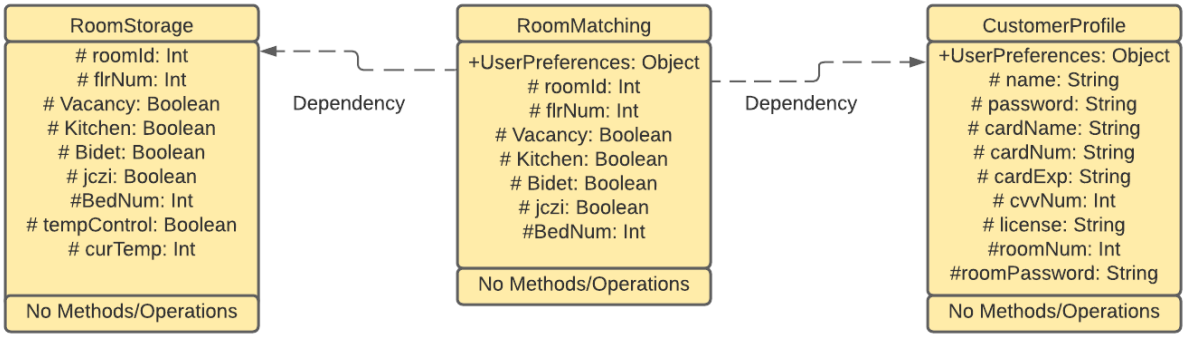


Figure 7.8.1 - RoomMatching function;s output is dependent upon all necessary data obtained from both RoomStorage and Customer Profile. Dependent on these values, the output of the function changes.

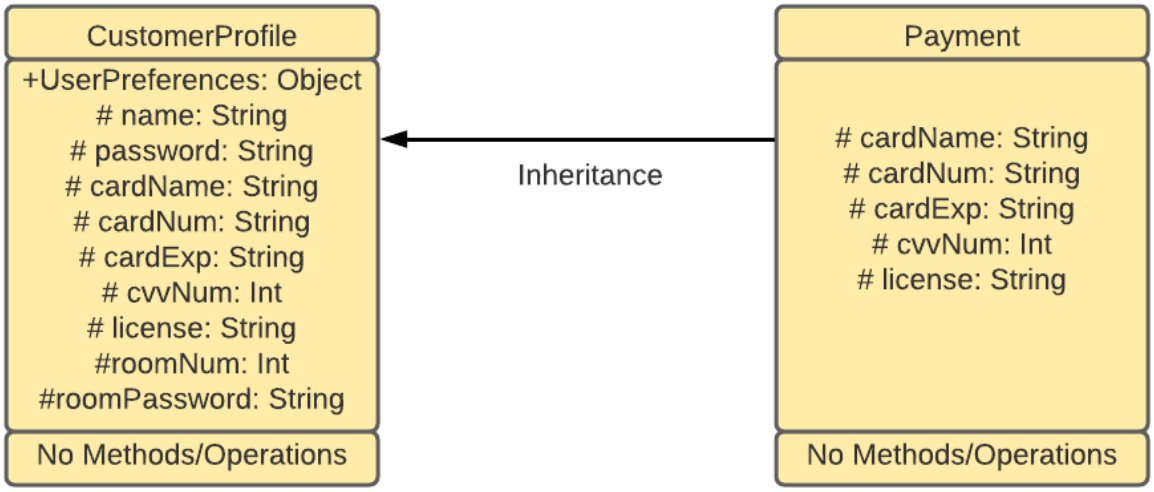


Figure 7.8.2 - All payment information inputted by the user is inherited by their customer profile and saved to be used and accessed any time the system needs a payment.

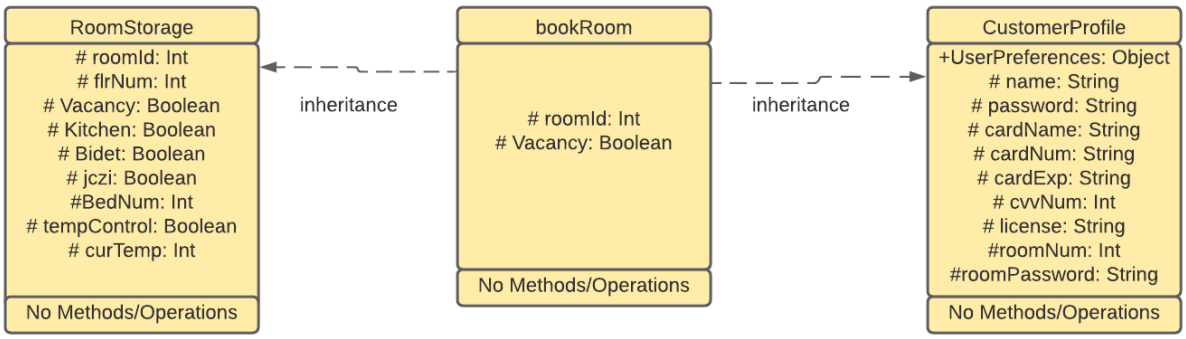


Figure 7.8.3 - Book Room inherits from both room storage and customer profile to edit and update data needed to set a room to non-vacant and assign a room number to a customer profile.

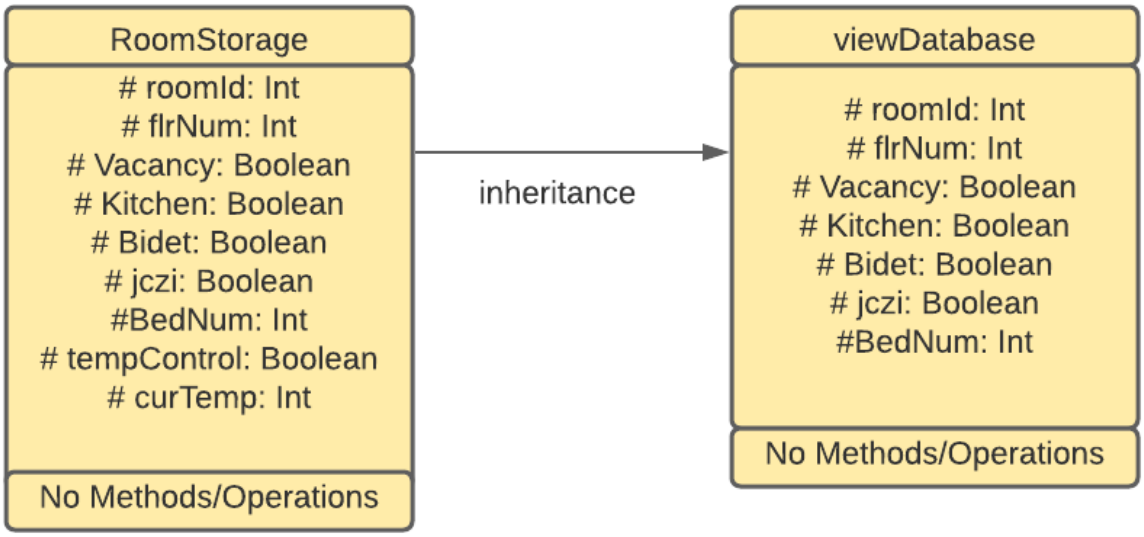


Figure 7.8.4 - View Database must inherit all given values from the Room database in order to correctly present all room information to the user requesting it.

7.9 Algorithms and Data Structures

The room matcher algorithm was designed in such a way to allow for the matching of a customer to a room even if an exact room is not available. The algorithm will allow for more

beds than normal to compensate for the customer's amenity wishes. To accomplish this, we implemented a preference ranking system, where the preference rank of a room would be increased based on each amenity that it had that the customer required. If these amenities were not present, or the number of rooms was below the required amount, the room's preference would not be increased for that amenity. This allows for the picking of a room.

There are three main data structures that our subsystem will be using: an SQLite database, a 2d Java Array, and an object of type ResultSet.

SQLite database: This is the main database that will be used by all parts of our system. In order to communicate with this database, our group will use JDBC (java database connectivity) to send / pull data to the database.This database will contain all the information for every room in a hotel. (It is possible that other systems will require additional SQLite databases).

2d Java Array: This data structure will be used to provide communication between the SQLite database and other systems. The first column in the array will contain room numbers that correspond to each individual room. The second column will contain all the data for the corresponding room number.

ResultSet: This object is similar to a linked list in terms of overall functionality. The ResultSet object will traverse through all entries within the database once (limitations of SQLite within eclipse). While the ResultSet is being traversed, it will also simultaneously fill the 2d Java Array.

7.10 Design of Tests

Use Case	Test Case Description	Test Coverage
UC-1 (Find Room)	User wants a room with at least 4 beds, a kitchen, a jacuzzi, and a bidet. You have no fitting room, but you have a room with 5 beds, and every needed amenity. You also have a room with 4 beds, a jacuzzi,	This test ensures that our matching system is able to meet requirements even if no room matches exactly.

	a bidet, but no kitchen. The matching system should prioritize needed amenities and allow for extra beds, but not less beds. Therefore, the matching system should pick the room with 5 beds.	We test in this case what happens when we have no room to match the exact requirements, but a room that matches with more beds and all amenities, or a room with the same number of required beds and not all amenities. This covers bases on how our algorithm handles prioritizing having the correct amenities over proper amount of beds, as well as its ability to compensate for lacking a proper room.
UC-2 (View Room Database)	A user logs into the system, clicks “Book a room” tab, and chooses“View Room Database”. The database is displayed in the system GUI for the user to view.	Tests the user logging in as a user with the ability to view the room database, as well as the ability to display the database.
UC-3 (Payment)	The user will log into the system, and click the “Payment” tab. They will input their payment information, and upon selecting done will be given a confirmation output.	Tests that the intake function successfully executes and deposits information into the customer profile database.
UC-5 (Book a Room)	A user logs into the system, clicks “Book a room” tab, and inputs their desired room number in the given box. The will now click “Book this room” and	Tests to see that the function correctly works, given either a vacant or

	be given a confirmation output or told that the room is not vacant depending on the selected room.	non-vacant room.

7.11 Effort Estimation Using Use Case Points

Unadjusted Actor Weight (UAW):

Actor	Description	Complexity	Weight
Guest	Guests interact with the system via a graphical user interface.	Complex	3
Manager	Managers interact with the system via a graphical user interface.	Complex	3
Staff	Staff interact with the system via a graphical user interface.	Complex	3
Database	Provides room information for functions to use.	Average	2

Unadjusted Actor Weight = 3+3+3+2 = 11

Unadjusted Use Case Points (UUCW):

Use Case	Description	Category	Weight
UC-1 (Find Room)	2 participating actors (User, database). Up to 5 steps.	Average	10
UC-2 (View Room Database)	2 participating actors (User, database). Up to 3 steps.	Simple	5

UC-3 (Payment)	2 participating actors (user, database). 4 steps required.	Average	10
UC-5 (Book a Room)	2 participating actors (user, database). 5 Steps required.	Average	10
		Use Case Weight Total	35

Technical Complexity Factor (TCF):

Technical Factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight*Perceived Complexity)
T1	System is not distributed.	2	0	2*0=0
T2	Users expect good performance but no critical response times or throughput.	1	3	1*3=3
T3	Users expect high efficiency but it is not exceptionally critical.	1	3	1*3=3
T4	Internal processing is relatively simple.	1	1	1*1=1
T5	System should be able to be reused in other hotels.	1	5	1*5=5
T6	Software needs to be easily installed for different hotels.	0.5	5	0.5*5=2.5

T7	Ease of use is very important.	0.5	5	0.5*5=2.5
T8	No portability concerns.	2	0	2*0=0
T9	System should be able to handle new features but is unlikely to need to do so.	1	2	1*2=2
T10	Concurrent use is required.	1	4	1*4=4
T11	Security is a significant concern.	1	5	1*5=5
T12	No direct access for third parties.	1	0	1*0=0
T13	No unique training needs.	1	0	1*0=0
			Technical Factor Total:	28

Technical complexity factor comes to 0.6+0.01*28=0.88

Environmental Complexity Factor (ECF):

Environmental Factor	**Description**	**Weight**	**Perceived Impact**	**Calculated Factor (Weight*Perceived Impact)**
E1	Beginner familiarity with UML-based development.	1.5	1	1.5*1=1.5
E2	Little familiarity with application problems.	0.5	1	0.5*1=0.5

E3	Some knowledge of object-oriented approach.	1	2	1*2=2
E4	Beginner lead analyst.	0.5	1	0.5*1=0.5
E5	Motivated with some team members lacking.	1	4	1*4=4
E6	Stable requirements expected.	2	5	2*5=10
E7	No part-time staff.	-1	0	-1*0=0
E8	Programming language of average difficulty will be used.	-1	3	-1*3=-3
			Environmental Factor Total:	15.5

Environmental complexity factor comes to 1.4-0.03*15.5= 0.935

Total UCP: (UUCW + UAW) * TCF * ECF

TOTAL UCP = 37.8488

Duration = UCP * PF = 35*28 = 980

7.12 History, Current, & Future Work

Future work for our subsystem mainly includes cleaning up convenience and simpleness. For each hotel, the system's database must be updated to match the hotel's layout of rooms. Making this process easy and understandable for hotel owners would be first on the list. Next, the room database could be more easily understandable as well, perhaps allowing for owners to upload floor maps would be a solution. The "View Room Database" button would display a floor

map of the hotel as an interactive image. This way, both customers and hotel staff can use it and easily navigate through the hotel, online or in person.

The next system that has room for improvement is our payment system. Currently, encryption and authentication systems are needed for verifying payment information. We have no way to tell if we are given a proper credit card or name, and perhaps third party connections could be used. Furthermore, the payment system could be updated to create a pop-up “prompt”. Users would be required to input valid payment information even before they were allowed to book the room.

Another potential application for our subsystem would be using it within a hospital environment. Our database can easily be structured in a way that could suit hospital rooms instead of hotel rooms. If this were done, patients or nurses would be able to interact with our system the same way customers do. The automation our system offers also provides health benefits to hospitals as it minimizes human interaction. Using our system in this manner could save lives and make healthcare more efficient.

History of Work:

Use Case	Development	Initial Implementation	Updated	Completion
UC-1	02.28-03/15	03/15	03/21-04/22	04/22
UC-2	04/04-04/15	04/15	04/16-04/22	04/22
UC-3	02/28-03/15	03/15	03/21-04/22	04/22
UC-4	04/04-04/15	N/A	N/A	N/A
UC-5	02/28-03/15	03/15	04/16-04/22	04/22

8. Project Management

History of Work Milestones:

Milestone	Deadline	Implementation Status
Creation of User Interface	Initial: 03/15 Final: 03/13	User Interface is operational.
Integration of Database to UI	Initial: 03/21 Final: 03/21	Database is fully integrated to User Interface
Raspberry Pi Integration	Initial: 03/21 Final: 03/25	Raspberry Pi is fully integrated into the system.
Subsystem Integration	Initial: 03/21 Final: 04/02	All subsystems are integrated into User Interface.
Housekeeping Request System Test	Initial: 03/21 Final: 03/21	Housekeeping test was successful, an updated test is now available.
Automatic Power Control System Test	Initial: 03/15 Final: 03/23	The Automatic Power test was successful.
Mobile Key System Test	Initial: 03/11 Final: 04/11	Mobile key administered successful new test.
Room Matching System Test	Initial: 03/11 Final: 03/15	Room matching test was successful. Development of a concurrent system in progress.
Creation and Implementation of the Room Service Subsystem	Initial: 04/19 Final: 04/23	Room Service system was created and implemented; allowing for intended operation.

Key Accomplishments:

- ★ Mobile Key Integration
- ★ Power Automatically Turns Off in Room
- ★ Thermostat Hardware Integrated into System
- ★ Ventilation System Integrated into System
- ★ Implementing and Integrating the Room Service Subsystem

Merging the Contributions from Individual Team Members:

Merging contributions from individual members is a challenge for every team. In our first iterations of designing diagrams, each sub-group ended up using a slightly different tool to create their diagram which created many inconsistent styles in the report. To ensure consistency, we aimed for one of the group members to lead with a format, and asked the other groups to emulate the format used so that design discrepancies were minimal. For example, one group would create a system sequence diagram and the other sub-groups would follow that style in creating theirs. Formatting throughout the report was also an issue that arised. With many different people inputting their work into the document, there quickly became many different appearances and styles that had to be normalized. To combat this, a project manager would create a table of contents and different headers to ensure that people placed their work in the correct spot, and matched the outline already put there. The project managers took on the responsibility of setting an agreed-upon format for the report, and sending a skeleton for all sub teams to follow. Another challenge faced was starting to connect the different systems. For example, while each sub-group had their own code for their system done, we needed to integrate that code to a singular database. Sub-groups met together to discuss and create a database format to work with to ensure integration would be possible with each sub-group. Meeting between sub-groups allowed for integration to begin on the system, database, and interface.

9. References

D-

Year = 2020

Date = March 11

Titles = A Guide to Mobile Keys for Independent Hotels

Publisher = webrezpro

Url = https://www.webrezpro.com/a-guide-to-mobile-keys-for-independent-hotels/

51

E-

Year = 2019

Date = september 30

Titles = 9 Smart Home Devices That Automate Your Home Publisher = nationwide

Url = https://blog.nationwide.com/9-wifi-home-automation-apps/

I-

Year = 2017

Date = August 12

Titles = DIY Smart Thermostat for cheap

Publisher = ecobots

Url =https://www.youtube.com/watch?v=HQk7H_XBCRo&ab_channel=Ecobots

J-

Year = 2019

Date = March 20

Titles = how to make motion sensor alarm

Publisher = easy tech

Url = https://www.youtube.com/watch?v=iEk5sajT5Lk&ab_channel=EASYTECH

K-

Year = 2008

Date = march 13

Titles = Creating Database Web Applications with Eclipse

Publisher = eclipse foundation

Url = https://www.eclipse.org/articles/article.php?file=Article-EclipseDbWebapps/index.html

L-

Year = 2019

Date = October 8

Titles = How to build a Raspberry Pi temp monitor

Publisher = initial state
Url=https://medium.com/initial-state/how-to-build-a-raspberry-pi-temperature-monitor-8c2f70acaea9

M-

Titles = RASPBERRY PI DS18B20 TEMPERATURE SENSOR TUTORIAL

Publisher = Circuit Basics

Author = Scott Campbell

Url = https://www.eclipse.org/articles/article.php?file=Article-EclipseDbWebapps/index.html