Cloud Surf Inn: Report 2

Submitted To: Software Engineering

Submitted On: 03/23/2021

Submitted By: Group 8

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

Github: https://github.com/Cloud-Surf-Inn

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

Individual Contribution Breakdown:

	Chynna	Jakub	Sebastian	Ryder	Juan	JP	Chris	Sidonia	Bharath	Zach
Analysis & Domain Modeling	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Interaction Diagram	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Class Diagram & Interface Spcifications	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Alogortihms & Data Structures	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
User Interface Deisgn and Implementatio	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
Design of Tests	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Project Management	70%	30%	0%	0%	0%	0%	0%	0%	0%	0%
Formatting	50%	50%	0%	0%	0%	0%	0%	0%	0%	0%

Table of Contents

1. Analysis and Domain Modeling

1.1.1 Concept Definition

Room Matcher:

Concept	Responsibility	Type
DB Connection	Coordinate interactions/data exchange with the database.	Service
Customer Profile	Storing all of the data for all rooms in the hotel including number of beds, vacancy, cleanliness, and amenities.	Entity
Matching	System matches the user's preferences to a room that has them.	Service

Generate Key:

Concept	Responsibility	Type
DB Connection	Coordinate interactions/data exchange with the database.	Service
Key	A 10-digit alphanumeric code used for room access then emailed to the user.	Service
MasterKey	10-digit alphanumeric that has access to all rooms	Service
Login Page	Allows users to select profile and log in to account. Inside each account contains information in regards to each position	Service
Staff Profile	Services will be able to view the schedule of cleaning and the corresponding room’s pin.	Entity
Customer Profile	Profile will be used to view keys they have access too	Entity
Manager Profile	Same as staff profile + viewing master key	Entity

Housekeeping:

Concept	**Responsibility**	**Type**
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

Automatic Power:

Concept	Responsibility	Type
Temperature	Each room is able to view and set its particular temperature.	Service
PowerSaver	Room status changed to power-saving mode	Service
DB Connection	Coordinate interactions/data exchange with the database.	Service

1.1.2 Association Definitions

Room Matcher:

Concept Pair	**Association Definitions**
Matching ↔ Key	Matching uses information from the database to give each customer a room with the amenities and requirements they selected.

Generate Key:

Concept Pair	Association Definitions
LoginPage ↔ CustomerProfile	The credentials the user logs in with load the correct user profile and interface.
LoginPage ↔ StaffProfile	The credentials the employee logs in with 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 they are generating a key
StaffProfile ↔ Key	Employee has access to keys and new keys
CustomerProfile ↔ StaffProfile	The employee will be able to monitor any request the customer will make.
ManagerProfile ↔ MasterKey	Manager will be able to view master key which has access to all rooms

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.

Automatic Power:

Concept Pair	Association Definitions
CustomerProfile ↔ DatabaseConnection	Check out from the user interface updates the room in the database as vacant.
DatabaseConnection ↔ PowerSaver	A vacant room in the database signals the power saver to activate.
CustomerProfile ↔ Temperature	The temperature is shown on user profile which is fetched from the database.

1.1.3 Attribute Definitions

Room Matcher:

Concept	**Attribute**	**Definition**	**Constraints**
Matching	User Preferences	The user's selected requirements for the room they wish to find.	None.
	BookRoom	After room is selected customer selects book room to confirm	None
Customer Profile	Room Information	Displays all information relevant to the user's room and stay	None
	Booked Dates	The dates a room is already booked.	None.

Generate Key:

Concept	**Attribute**	**Definition**	**Constraints**
Key	GenerateKey ViewKey	The system will create a pin for people to gain access to rooms	Pin cannot repeat for more than one room at a time

MasterKey	GenerateMaterKey ViewMasterKey	The system will create a 10 digit pin that will allow access to any room	None
Manager Profile	ViewMasterKey	Displays and stores all information of which the manager has the authority to access.	None
Staff Profile	ViewKey	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	None

		profile it pertains to. The user will use this combination to log in to the system.	

Housekeeping:

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

Automatic Power:

Concept	Attribute	Definition	Constraints

PowerSaver	PowerControls	Power saver includes controls to the power which are turned off or on.	PowerControls are turned off or on depending on the state of the vacancy entity.
Temperature	ViewTemp	Shows the customer the current temperature	Database allows viewing, can not change temp to view.
	SetTemp	Allows the customer to input a new temperature.	Integer Value must be entered to change.
	Thermostat	Reads desired temperature from database and changes the temperature of the room accordingly.	Desired temperature must be in the ranges of 40 degrees and 90 degrees Fahrenheit.
DBConnector	DBChanger	Database changes status to “vacant”	Boolean operation turns to vacant after checkout
	CheckIn	Customer Changes status to “check-in”	Boolean operator from customer input
	CheckOut	Customer changes status to “check out”	Boolean operator from customer input

1.1.4 Traceability Matrix

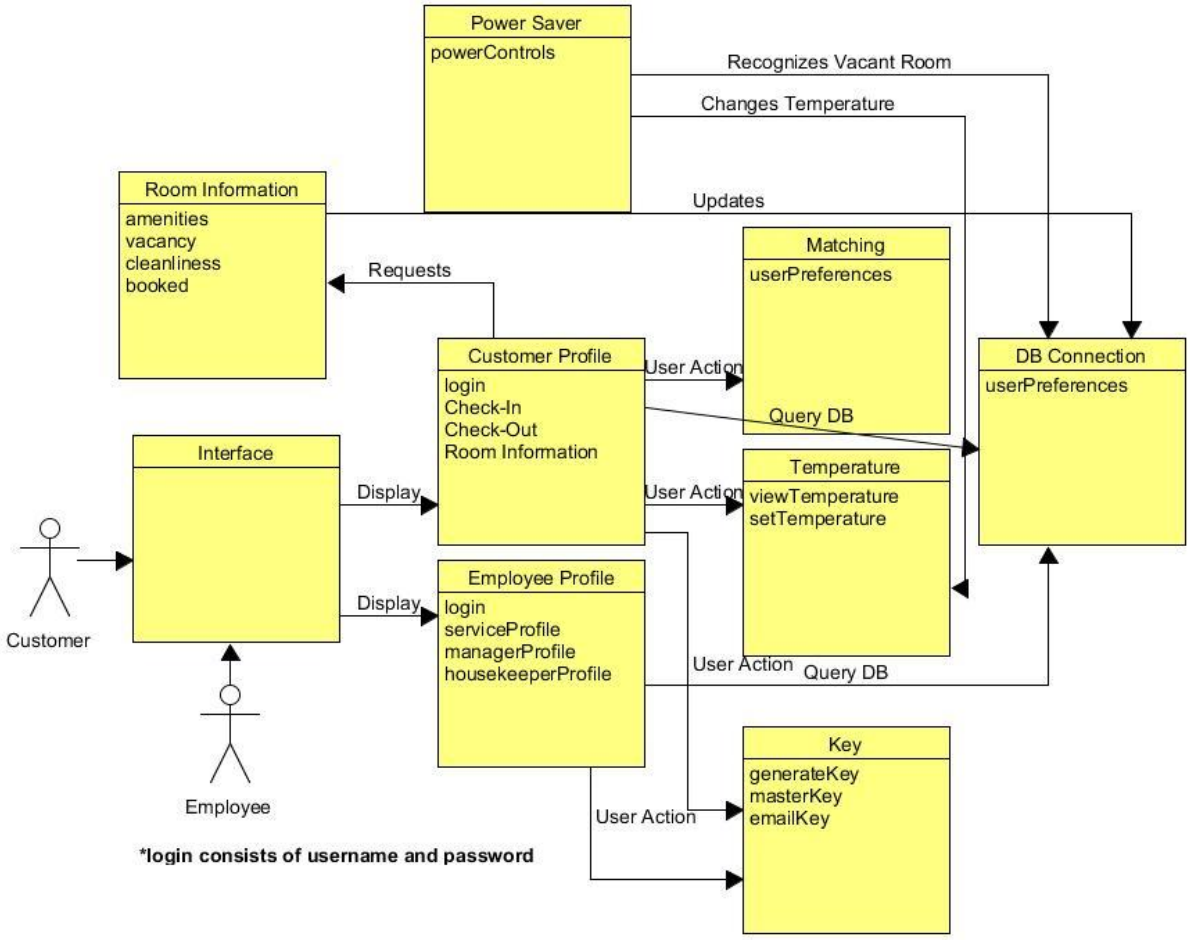
	Concept													
Use Case	**Customer Profile**	**Matching**	**DB Connection**	**Key**	**MasterKey**	**Login Page**	**Staff Profile**	**Manager Profile**	**Accept Task**	**Remove Task**	**Delete Task**	**Create Task**	**Temperature**	**Power Saver**
UC-1	X	X	X	X		X								
UC-2			X			X	X	X						
UC-3	X		X			X								
UC-4	X		X			X								
UC-5	X		X	X		X								
UC-6				X		X	X	X	X	X	X	x		
UC-7	X		X	X		X								
UC-8				X	X									

UC-9				X		X			X					
UC-10	X			X										
UC-11			X			X							X	
UC-12			X			X							X	
UC-13			X										X	X
UC-14			X										X	X
UC-15			X											X
UC-16	X		X						X	X		X		
UC-17			X	X		X	X		X	X	X			
UC-18			X			X	X							
UC-19	X					X								
UC-20	X					X								
UC-21	X		X			X				X	X			
UC-22			X			X		X						
UC-23	X					X						X		
UC-24	X		X			X	X			X				
UC-25			X			X	X		X	X				
UC-26			X			X	X		X	X				
UC-27	X		X			X		X						
UC-28			X			X		X						
UC-29				X	X	X		X						

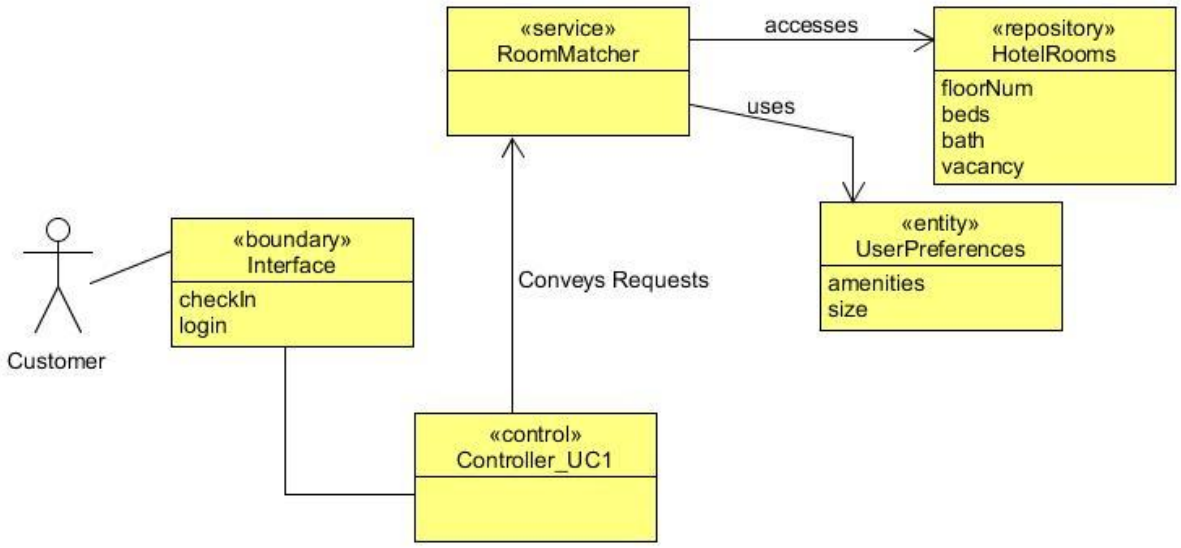
1.1.5 Domain Model

Prior to creating the concepts and their respective definitions, it was beneficial, to begin with formalizing the attributes associated with the Cloud Surf Inn system and then group them within the concepts they were directly associated with. Each subsystem and microservice has specific attributes that serve as elements, constructs, or indicators that are all vital to the system's successful operation. These attributes provide the foundation which subsystems will be based on and revolve around while they are in operation. Thus, the Use Cases served as the source from which attributes were ideated from. All attributes serve as the building blocks that allow Use Cases to be realizable, at least from a development perspective. Attributes, like “Generate Key, Master Key,” and “Email Key” for example, can be then grouped together under concepts (like “Key” in this illustration) to properly define the parameters each concept has. The concepts

themselves then serve as the high-level mechanisms and elements that Cloud Surf Inn will utilize and manipulate to function correctly and provide value to the customer.

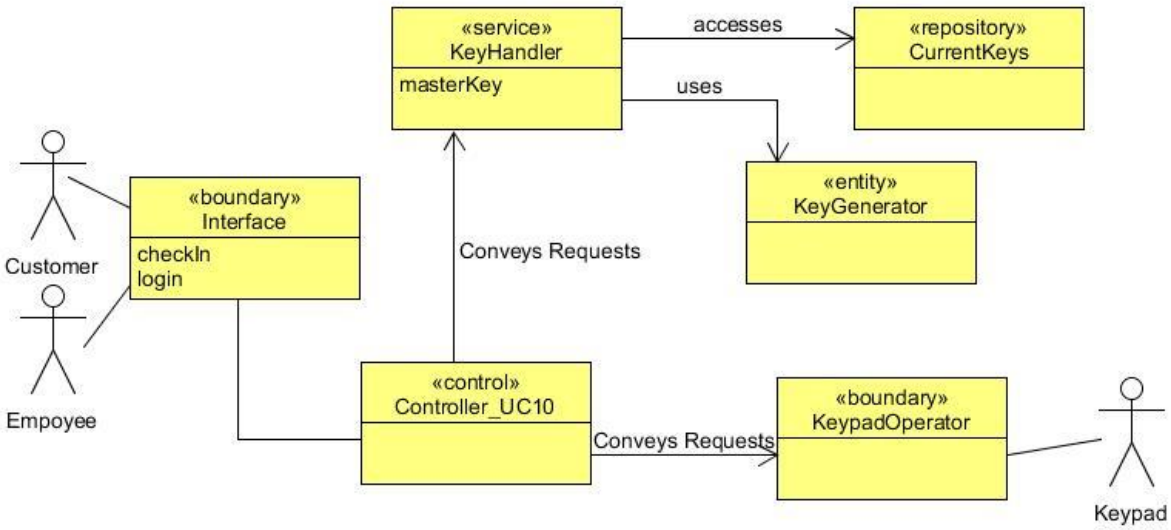


Room Matcher:



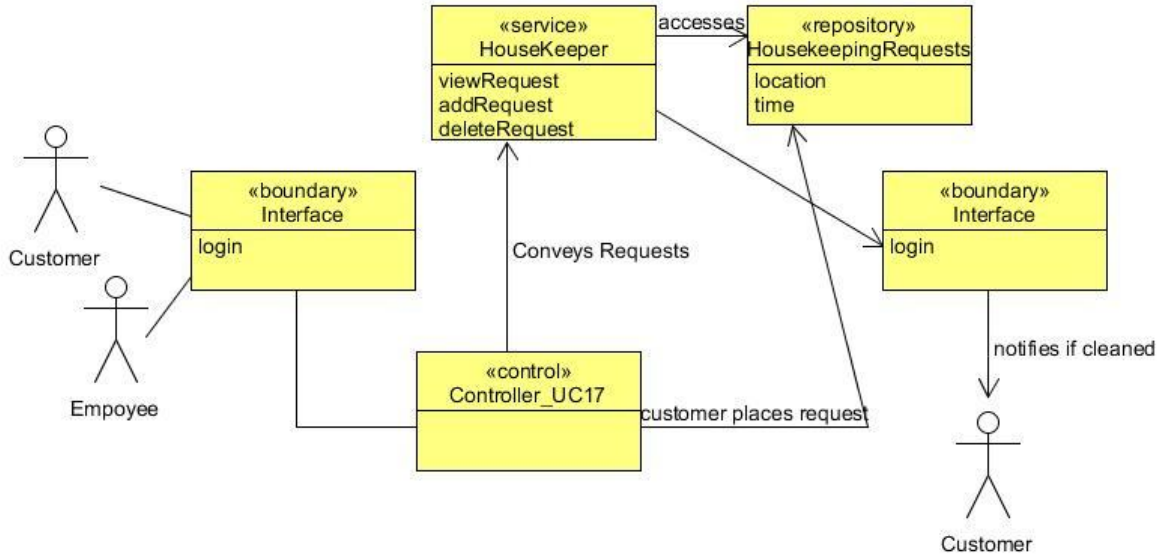
In order to keep the design modular, we will create a separate function for the matching algorithm. This function will take in the user's preferences as input when called, as well as the rooms from the database. Using this information, the matching algorithm will run, and the output will be taken by the driver function to be displayed to the customer.

Mobile Key:

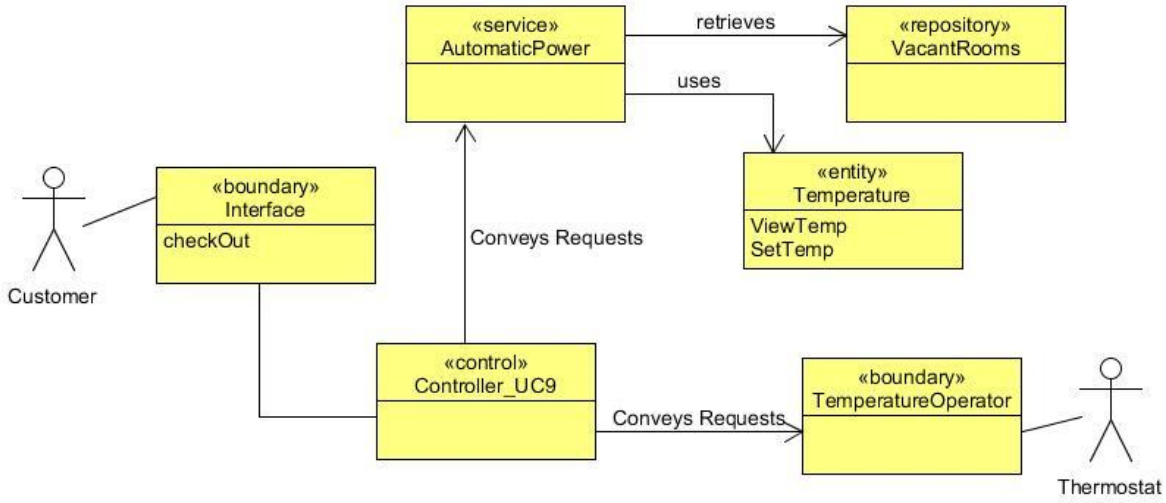


A key generator function is created, which provides the master key (for hotel manager) and key(s) for hotel guest use. Room access is determined by inputs provided by the actor (either manager or guest) to the keypad.

Housekeeping:

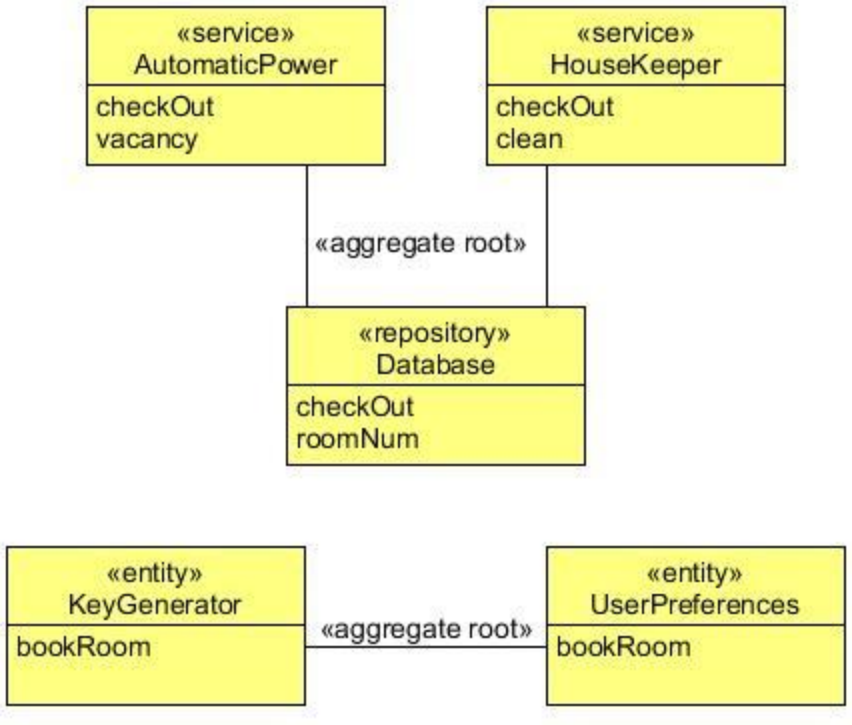


Power Control:



To ensure a modular design, the various tasks of the sub-system to perform this use case are split up. The customer interacts with the boundary of the interface and checks out. The controller handles the interaction between the AutomaticPower service which must retrieve which room became vacant and use the temperature entity to modify the temperature. The controller relays this request to the TemperatureOperator which interacts directly with the sensor and thermostat.

Aggregates:



These aggregates are identified to ensure the system does not have issues with concurrency. Our first aggregate comes between the key generator and user preferences, each will interact with the database at the same time when the user attempts to book a room. The second aggregate comes between the automatic power and house keeper. Each of these also becomes active when a customer checks out of a room and must access similar data in the database, we do not want to lock either system out so we must ensure the concurrency is accounted for.

1.2 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.

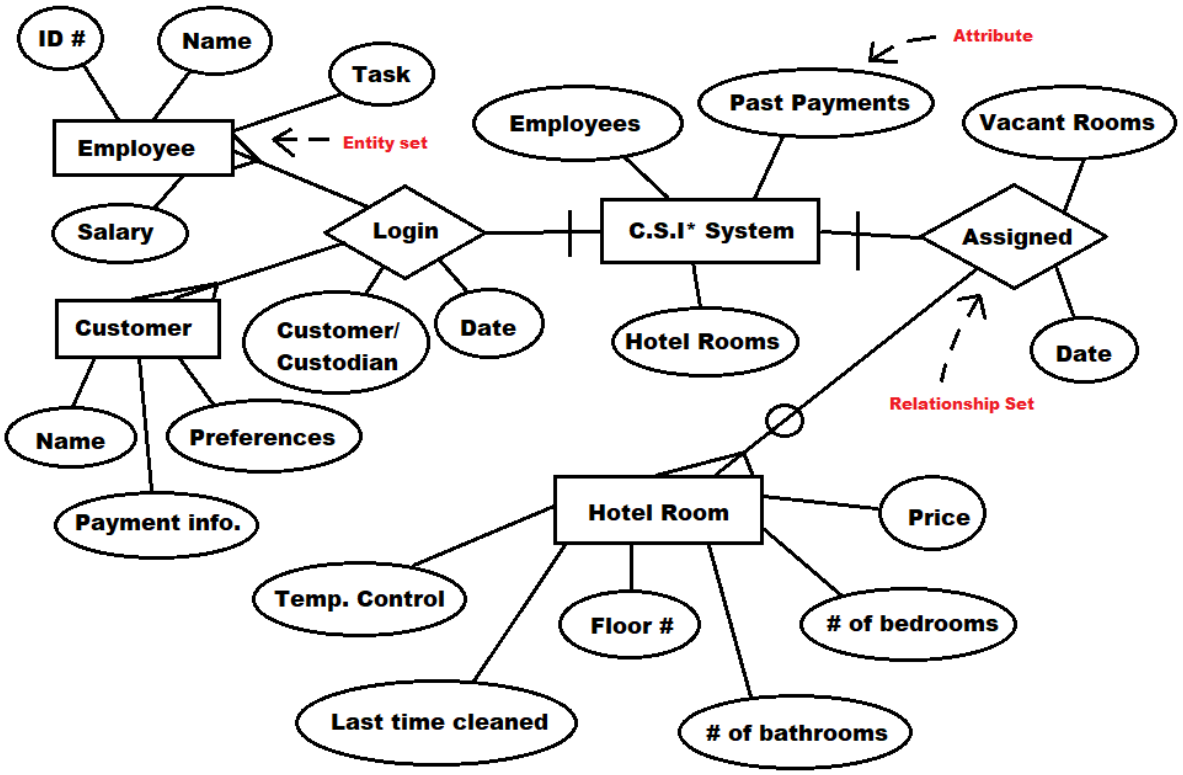
Operation	**Room Key Generated**

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

Operation	**Room Booking**
Use Case:	UC-1
Preconditions:	➔ Customer(s) have a Cloud Surf Inn Account.
Postconditions:	➔ Customer(s) have been assigned a room.

Operation	**Automatic Power Controls**
Use Case:	UC-13
Preconditions:	➔ Customer(s) have changed status to “check out.”
Postconditions:	➔ Power in the room is shut off.

1.3 Data Model and Persistent Data Storage



Relationship Examples:

One A is associated with one B:

One A is associated with one or more B's:

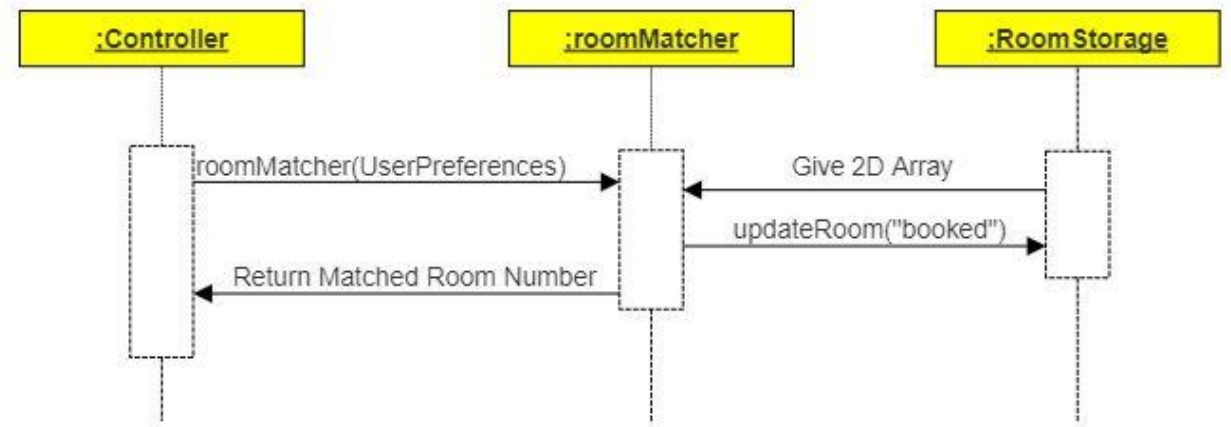
One A is associated with zero or more B's:

1.4 Mathematical Model

No mathematical models are utilized.

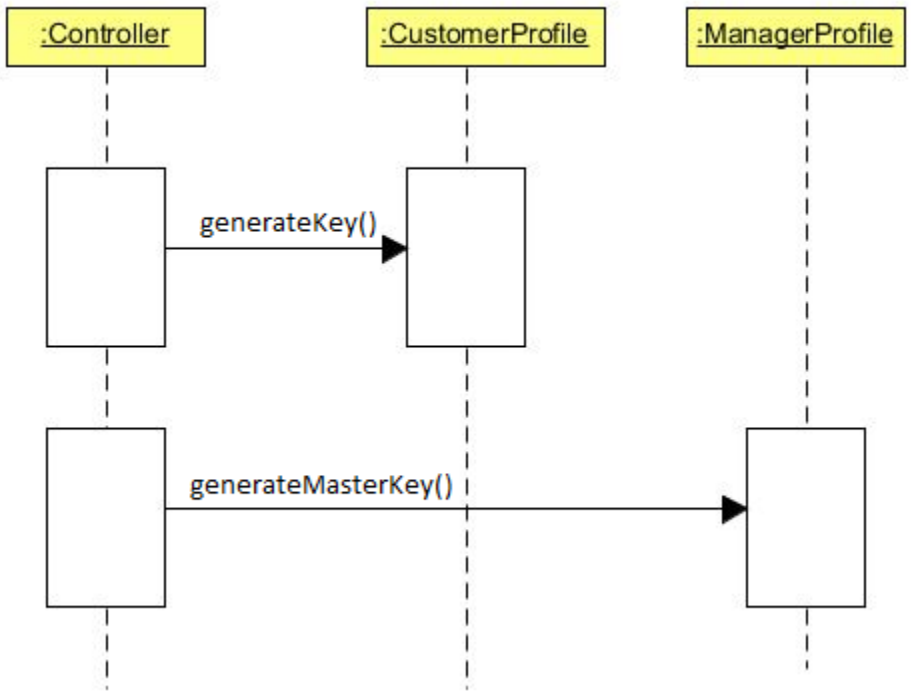
2. Interaction Diagrams

Room Matcher:



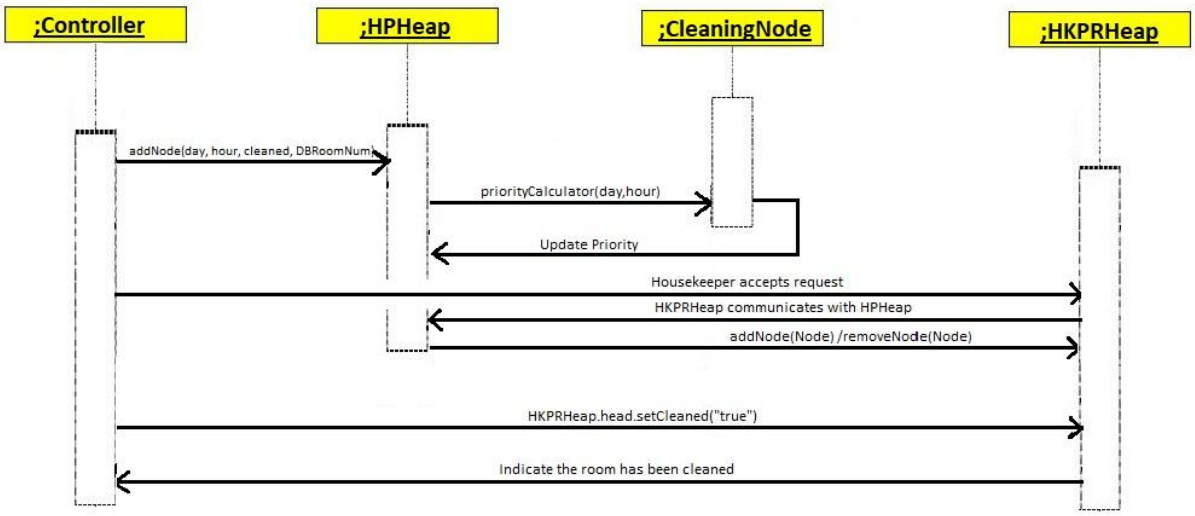
The controller calls roomMatcher() function with inputs from (UserPreferences) - these are the customer's room requirements. The roomMatcher function takes in the 2D database array of rooms, and matches the inputs with this array to best fit the customer's needs. When finished, it updates the matched room by calling the updateRoom() function, and returns back the matched room number to the controller. This interaction diagram was derived from the system sequence diagram of UC-1.

GenerateKey:



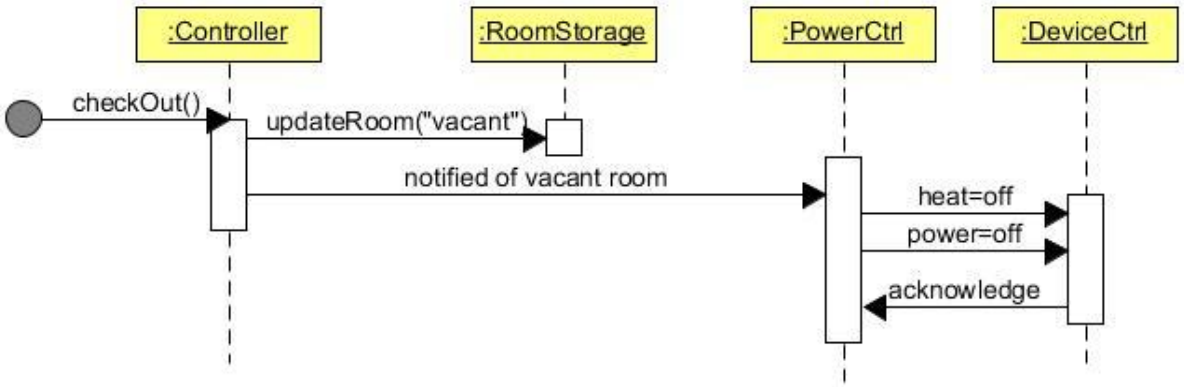
The controller will output both a generateKey() and update it into the customerProfile, and the controller will also output a generateMasterKey() and update that into the managerProfile. This interaction diagram is based off of the system sequence diagram for UC-10. It details the different classes that interact to insure there is a master key for the manager and a key for the customer profile.

Housekeeping:



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.

Power Controls:



This figure represents the interaction diagram for fully-dressed UC-13. It is derived from the system sequence diagram for fully-dressed UC-13. It is initiated by the checkOut() click from the customer which tells the database to update the room to "vacant" and then moves on to the power control. This then tells the device controller to turn the heat and power off, which the power controller will acknowledge when done. The single responsibility principle was employed here, each component has only one responsibility. For example the device control only has to access the devices to turn the heat and power off and the power control must signify the device controller to do those things. Another design principle used is the interface segregation principle, each controller only depends on the part of the interface it needs, making it resistant to modifications in other parts of the system. This interaction diagram was derived from the system sequence diagram of UC-13.

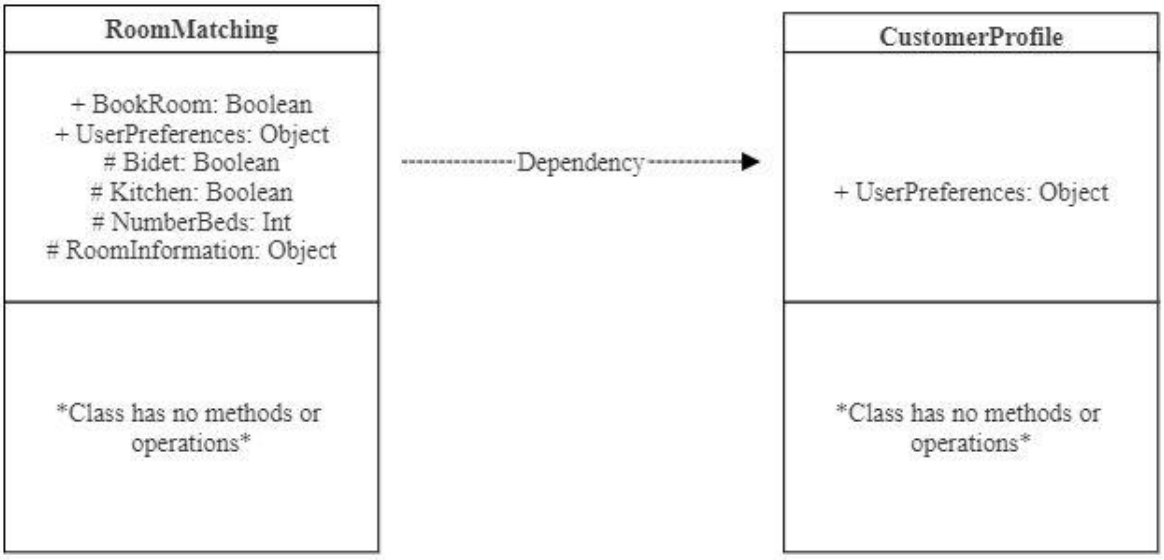
3. Class Diagram and Interface Specification

3.1 Class Diagram

Room Matcher:

Class	Meaning/Use of Class	Associations
RoomMatching	Matches customer's given room parameters with room from database.	RoomStorage CustomerProfile

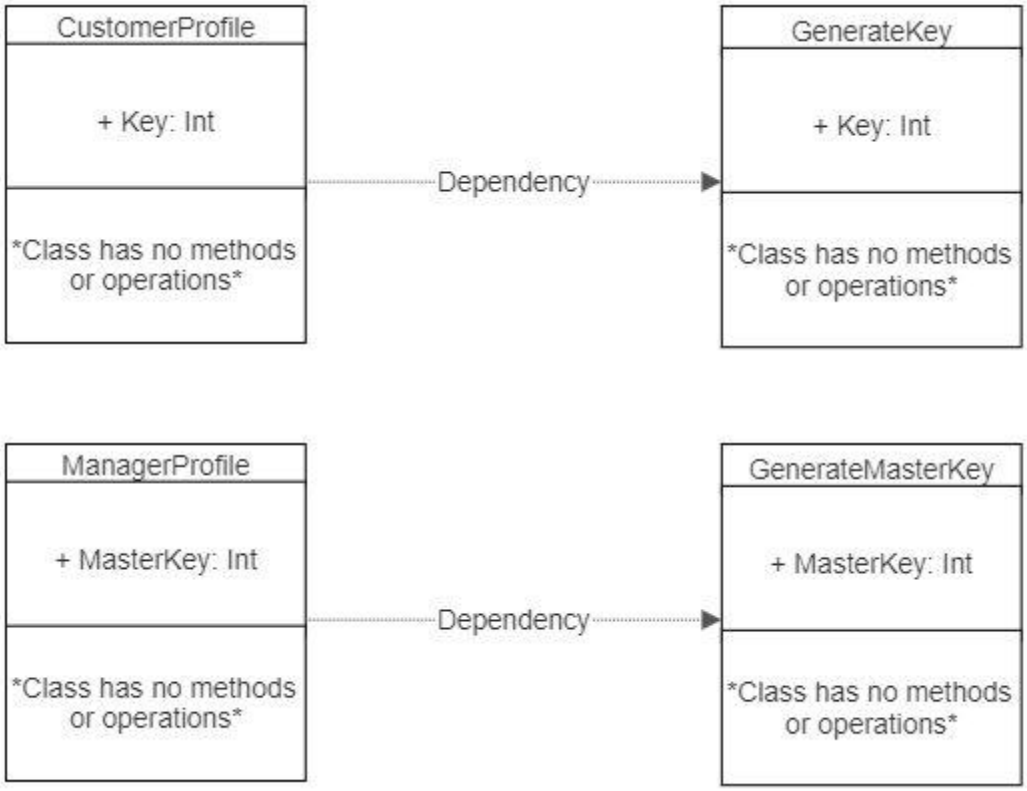
CustomerProfile	Stores customer's information and room preferences.	N/A
RoomStorage	A repository that is part of the database that holds information related to a room number, such as the device number of a thermostat.	N/A



The RoomMatching class is dependent upon the CustomerProfile class because of its need for inputs from UserPreferences. All other values required for execution of RoomMatching are inherited from the controller parent function when calling roomMatching, all in RoomInformation

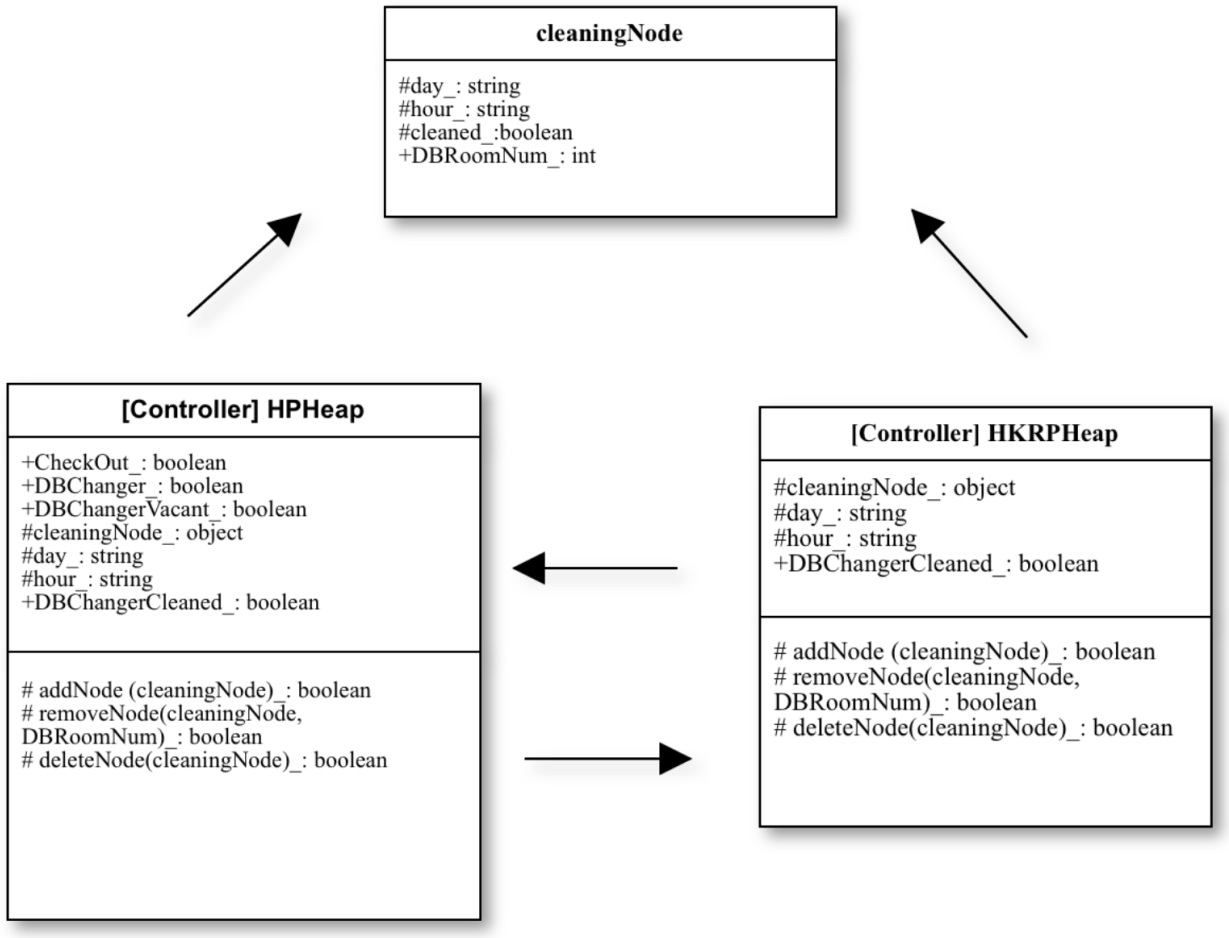
GenerateKey:

Class	**Meaning/Use of Class**	**Associations**
generateKey	Creating an alphanumeric number for guest/user to use in order to access rooms	StaffProfile ↔ Key UserProfile ↔ Key
generateMasterKey	Creating an alphanumeric number for manager to use in order access all rooms	ManagerProfile ↔MasterKey
CustomerProfile	Stores customer's information and room preferences.	N/A
MangerProfile	Stores manager information and masterkeys	N/A



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

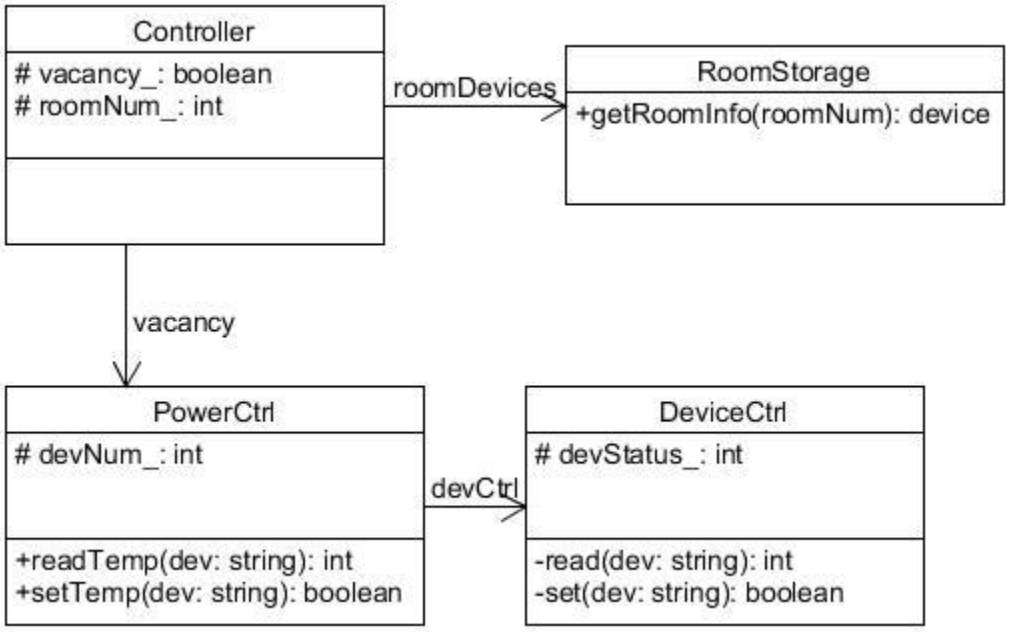


The HPHeap and HKPRHeap classes both utilize the CleaningNode class to perform their operations. The HPHeap and HKPRHeap classes also interact with each other to transfer CleaningNodes, delete CleaningNodes, and create new CleaningNodes. All three classes rely on the controller and each other to acquire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

Power Controls:

Class	Meaning/Use of Class	Associations
Controller	The main controller, it has the responsibility of gathering room information data from the database and notifying the power controller if a vacancy occurs.	RoomStorage (database), PowerCtrl
RoomStorage	A repository that is part of the database that holds information related to a room number, such as the device	N/A

	number of a thermostat.	
PowerCtrl	The interim object between the controller and device controller, it recognizes the change in vacancy variable and asks the device controller for a specific device's attributes.	DeviceCtrl
DeviceCtrl	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



3.2 Data Types and Operation Signatures

Room Matcher:

Class	Attributes and Operation	Data Type
RoomMatching	BookRoom UserPreferences *Bidet*	Boolean Object Boolean

	Kitchen *Jacuzzi* *NumberBeds* RoomInformation	Boolean Boolean Integer Object Object
CustomerProfile	UserPreferences	Object
RoomStorage	RoomInformation	Object

GenerateKey:

Class	**Attributes and Operation**	**Data Type**
GenerateKey	Key	Array
GenerateMasterKey	MasterKey	Array
CustomerProfile	Key	Array
ManagerProfile	MasterKey	Array

Housekeeping:

Class	**Attributes and Operation**	**Data Type**
cleaningNode	day hour cleaned DBRoomNum priority *priorityCalculator* *CleaningNode (constructor)* *compareTo* *equals*	String String Boolean Integer Integer Operation Operation Operation Operation
HPHeap	CheckOut DBChanger DBChangerVacant cleaningNode day hour DBChangerCleaned *addNode* *removeNode* *deleteNode*	Boolean Object Boolean Object String String Boolean Operation Operation Operation

	HPHeap(constructor) *size* *peek* *poll* *contains*	Operation Operation Operation Operation Operation
HKPRHeap	cleaningNode *addNode* *removeNode* *deleteNode* day hour DBChangerCleaned *size* *peek* *poll* *contains*	Object Operation Operation Operation String String Boolean Operation Operation Operation Operation

Power Controls:

Class	**Attributes and Operation**	**Data Type**
Controller	vacancy roomNUm	boolean int
RoomStorage	getRoomInfo(roomNum): device	operation
PowerCtrl	devNum readTemp(dev: string): int setTemp(dev: string): boolean	int operation operation
DeviceCtrl	devStatus read(dev: string): int set(dev: string): boolean	int operation operation

3.3 Traceability Matrix

Domain Concepts	**Classes**												
	RoomMatching	**CustomerProfile**	**GenerateKey**	**GenerateMasterKey**	**Controller**	**RoomStorage**	**PowerCtrl**	**DeviceCtrl**	**cleaningNode**	**HPHeap**	**HKPRHeap**	**SetTemperature**	**AutomaticPower**
DB	X	X			X	X	X	X				X	X

Connection													
CustomerProfile	X	X	X		X		X		X				
Matching	X				X	X							
Key		X	X		X								
MasterKey			X	X	X								
Login Page		X	X	X	X								
Staff Profile			X		X				X	X	X		
Manager Profile			X	X	X				X	X			
Accept Task			X		X				X	X	X		
Remove Task					X				X	X	X		
Delete Task					X				X	X	X		
Create Task					X				X	X	X		
Temperature		X			X	X	X	X				X	X
PowerSaver		X			X	X	X	X					X

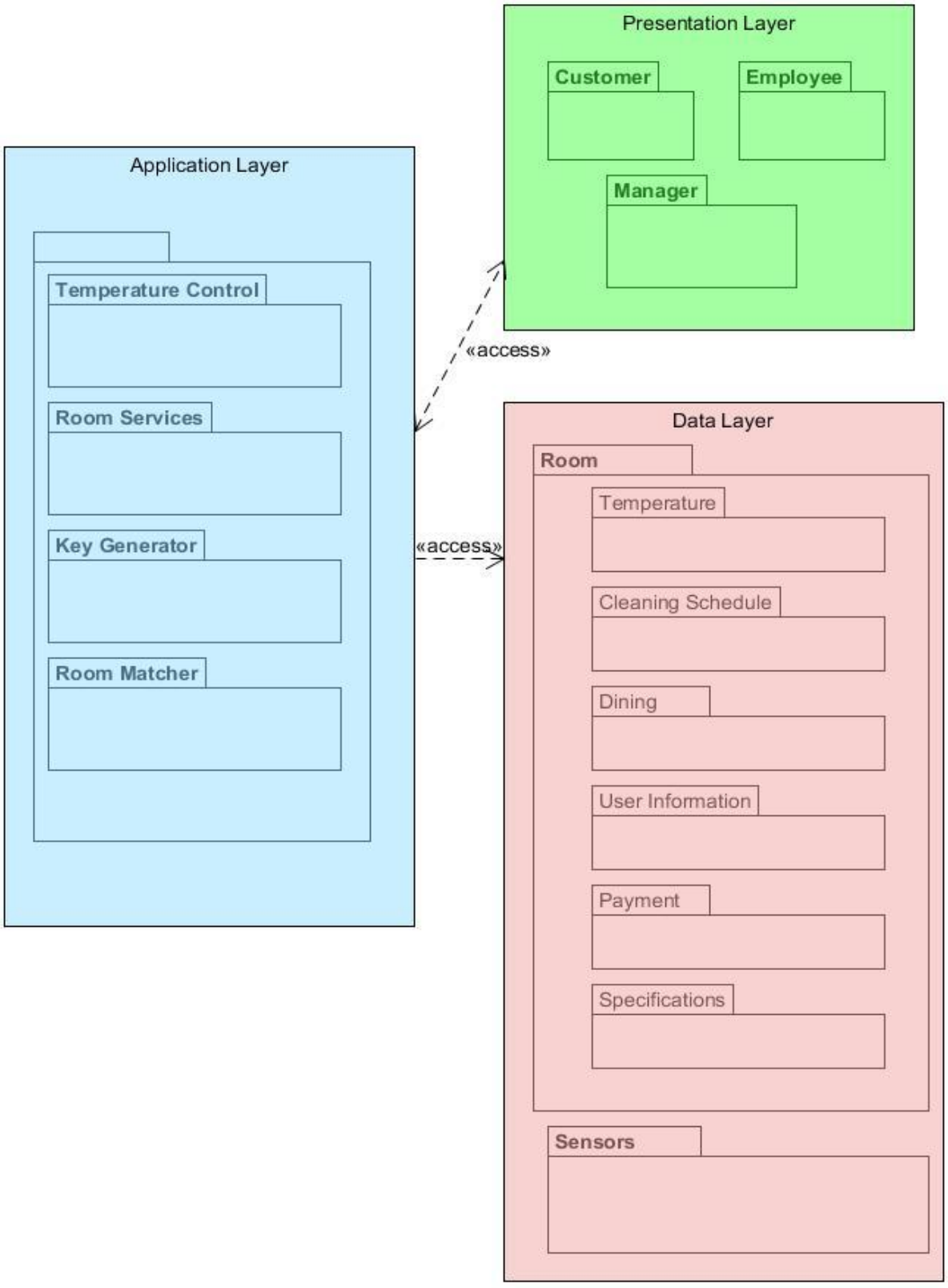
4. System Architecture

4.1 Identifying Subsystems

The subsystem package diagram depicts the three-tier architecture decided upon for the system. Starting with the presentation layer, different user interfaces are tailored to individual user's needs depending on if they are customers, employees, or managers. Following the presentation layer comes the application layer, which holds the business logic for fulfilling various functions the user might require such as matching to a room and especially for the automatic power-saving mode. The application layer handles most of the interaction between the presentation layer and the data layer and manages the overall system; because of this, there is an access dependency between these two layers, the application layer must utilize the input

information from the presentation layer, and the presentation layer must communicate that information to the application layer to respond to requests. The data layer interacts with the application layer to provide the large amounts of data stored about the hotel, it includes things such as individual room specifications, i.e., number of beds, amenities provided, as well as information about the sensors needed to control the temperature and lighting. This layer is accessed by the application layer in order for the application to access the information in the data layer and specify queries. The data layer contains a room package with other separate packages because of the large amounts of information that is associated with specific rooms and the users that interact with those rooms, and the need for the different aspects of the room package to be divided and conquered.

The separation into layers, as well as the distinct packages in those layers, allow separate teams in our group to tackle different parts of the system that are seemingly related. This maintains a degree of separation between layers and even packages so that a problem in one group does not negatively affect another.


Presentation Layer
Customer
Employee
Manager
Application Layer
Temperature Control
Room Services
Key Generator
Room Matcher
«access»
«access»
Data Layer
Room
Temperature
Cleaning Schedule
Dining
User Information
Payment
Specifications
Sensors

4.2 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 data 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 data layer contain access to critical information regarding guests as well as 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 from the data layer. The presentation layer has to do with the user interface that interacts with the customers and employees. 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 data layer. It also has the responsibility of performing the automatic features of the system, such as the power-saving mode. The data layer stores information about each room and the hotel operations.

Cloud Surf Inn also utilizes the Client-server model. The clients consist of the customers who must access the temperature and thermostat server, and the database to reach information about rooms. The managers are also clients who access the power saving mode feature to enable the automatic power control system and can view room statuses and housekeeping schedules. The employees are another set of clients that access room information and housekeeping schedules as well. The different clients here also have access to different user interfaces depending on which features apply to them. The clients must interact with the server to receive temperature information, room information, to enable the power control system, to receive ordering information, room key information, and various other services. The client-server model allows Cloud Surf Inn to connect its many potential clients with the information or service needed from the database or application layer.

4.3 Mapping Subsystems to Hardware

In the Cloud Surf Inn, we are using a client-server model, we will need to have a client and a server, which will also hold the database, running on different computing devices. The client will run on mobile devices and will be the way the users, both customers, and managers,

interact with Cloud Surf Inn. The server itself will have the responsibility of managing the user's requests regarding temperature, interacting with the database to retrieve the relevant information. The manager will also need to enable the power saving mode feature, which will automate the information in the database to execute the power saving tasks.

4.4 Connectors and Network Protocols

The temperature and power control is designed in a server/client fashion implementing Sockets for communication between Raspberry Pi and DataBase/GUI. The Raspberry Pi listens for a message from the GUI that signifies if a room is 'Vacant' or 'Booked'. Once this message is received, the ROOM_STATE parameter is set to True or False depending on whether 'Vacant' or 'Booked' is received. If ROOM_STATE equals false the temperature is turned off and Power Saving Mode is activated. If ROOM_STATE equals true Comfort Mode is Activated. If a client from within the GUI requests to 'view' or 'set' the temperature. The Raspberry Pi receives the request and the appropriate data is returned to the client.

4.5 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 book a room, they will input specific, required information (number of room occupants, amenities wanted) through the client, which will be sent to the controller upon submission. This controller performs the needed matching algorithm given data from both the client and the server containing the database then marks that room is booked; this happens for every customer every time they wish to book a room. 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 custodian would. At this point, the system is waiting for certain events to occur before

responding accordingly. Yet for things like scheduling housekeeping, the process itself is linear regardless of who is requesting it.

Time Dependency - Our system is 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, but the system does not time these requests and automatically handle their resolution. Continuing with the housekeeping example, a custodian 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.

4.6 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.

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 used will include 2pcs DHT22 Temperature Humidity Sensor.

5. Algorithms and Data Structures

5.1 Algorithms

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

The temperature and power control algorithm is designed in a server/client fashion implementing Sockets for communication between Raspberry Pi and DataBase/GUI. The Raspberry Pi listens for a message from the GUI that signifies if a room is 'Vacant' or 'Booked'. Once this message is received, the ROOM_STATE parameter is set to True or False depending on whether 'Vacant' or 'Booked' is received. If ROOM_STATE equals false the temperature is turned off and Power Saving Mode is activated. If ROOM_STATE equals true Comfort Mode is Activated. If a client from within the GUI requests to 'view' or 'set' the temperature. The Raspberry Pi receives the request and the appropriate data is returned to the client.

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.

5.2 Data Structures

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.

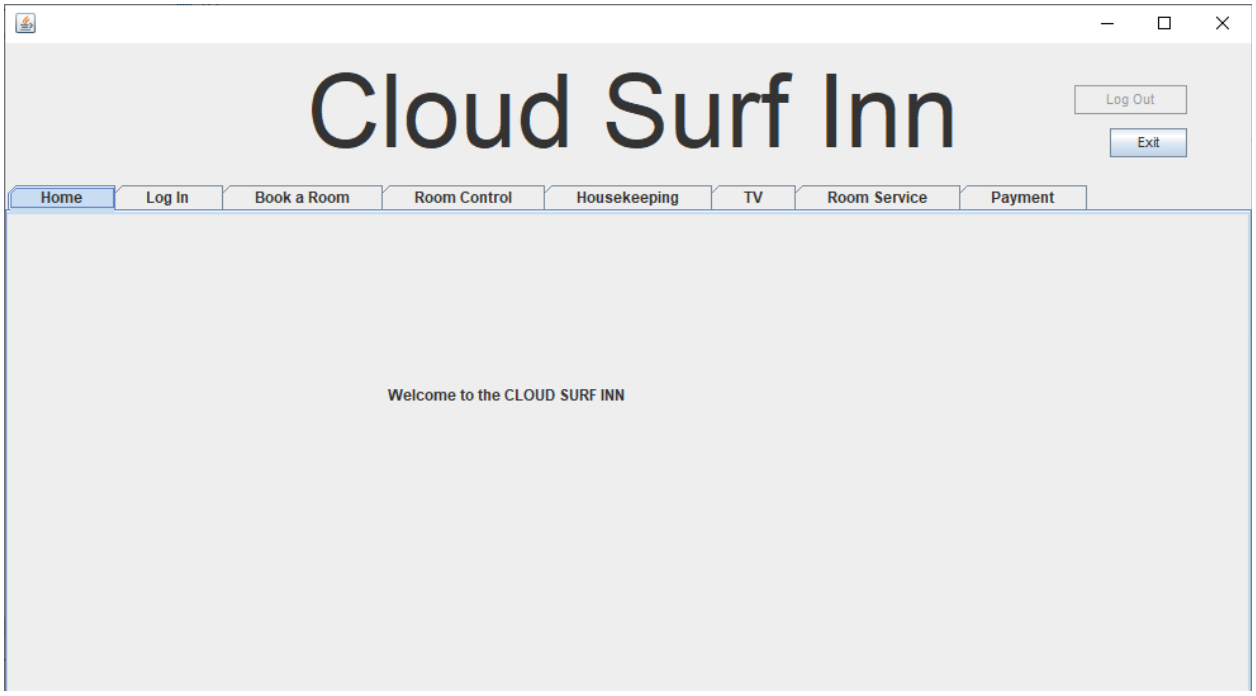
The Housekeeping System uses the Priority Queue data structure (Java's implementation) to provide the framework for the HPHeaps and HKPRHeaps. Many of the HPHeaps and HKPRHeaps methods are inherited directly from the Priority Queue Class that Java includes.

5.3 Concurrency

6. User Interface Design and Implementation

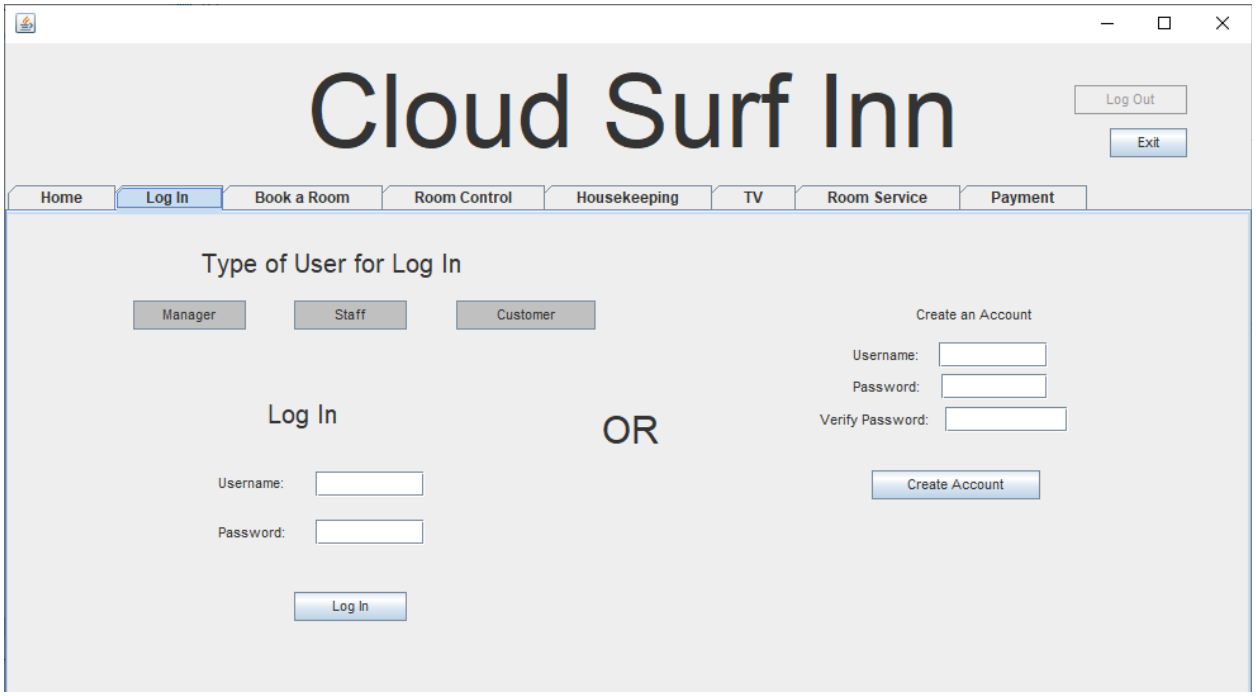
All buttons and tabs were made to look exactly the same as the mock up version of the GUI, so there are no significant changes. Additionally, there is no effect on the user effort estimation because everything works exactly as planned. Currently, the user interface runs at the same time with the database. The database is fully implemented, so it can talk with the user interface. The user interface can take in values, store them locally, then store them to the database. Additionally, the database can update values for the GUI when needed. Furthermore, most teams have worked with the database group and GUI group to implement their code into the project.

Home Page



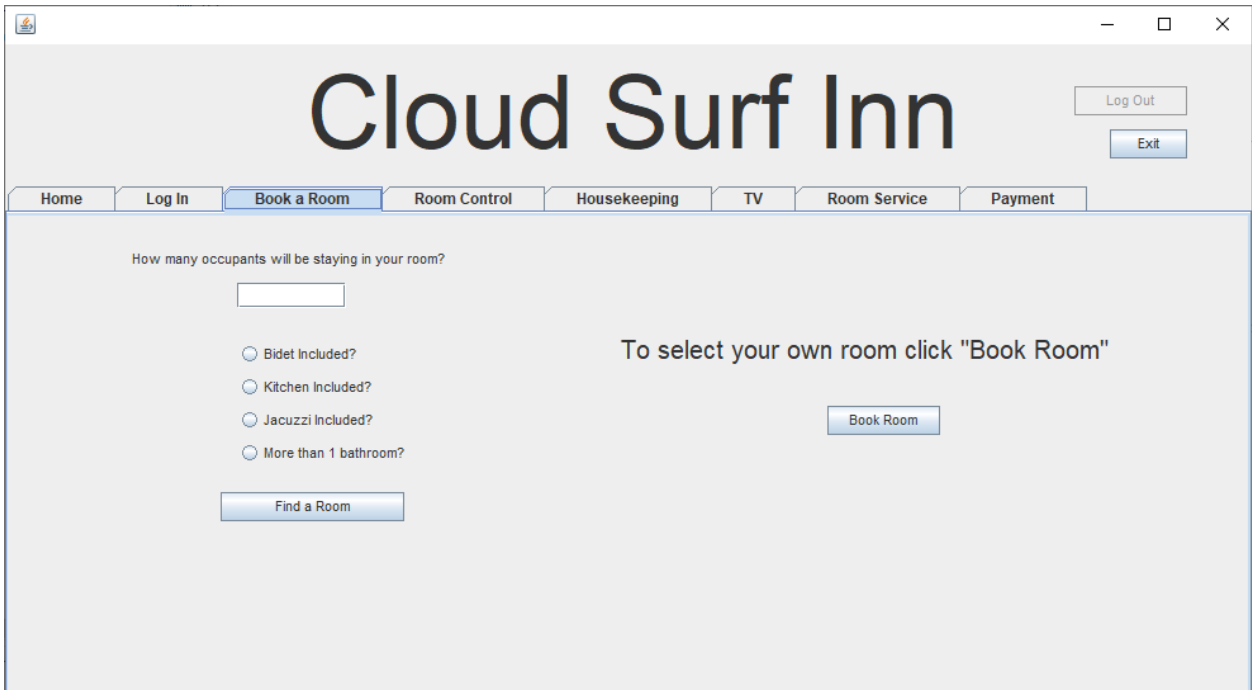
The home page is the first page the user sees. Right now it is a rudimentary implementation but will be updated with more aesthetically pleasing graphics and information for the user.

Login Page



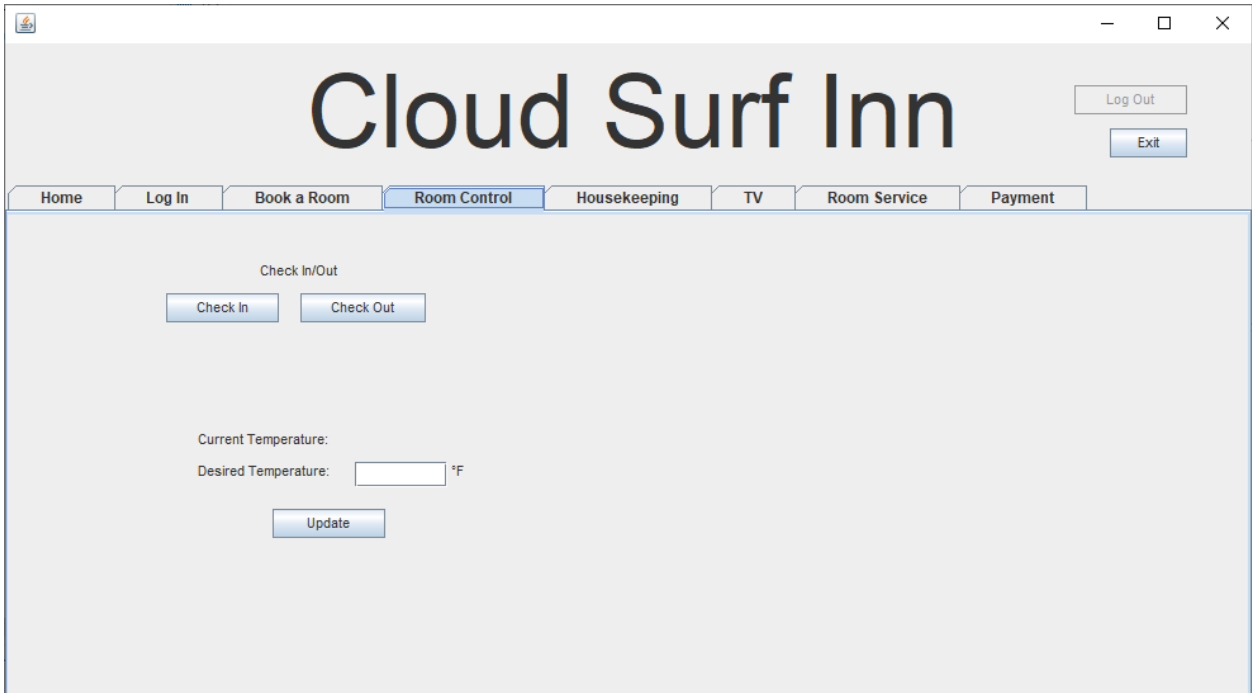
The login page has been implemented. It allows users to login or create accounts (CREQ-1, SREQ-1, SREQ-2, MREQ-1, MREQ-2). The login tab must simply be selected from the tabs, only 1 user click is needed. Different users need to click which member they are, i.e. manager, staff, or customer to have additional screens needed for specific users (REQ-39).

Book a Room Page



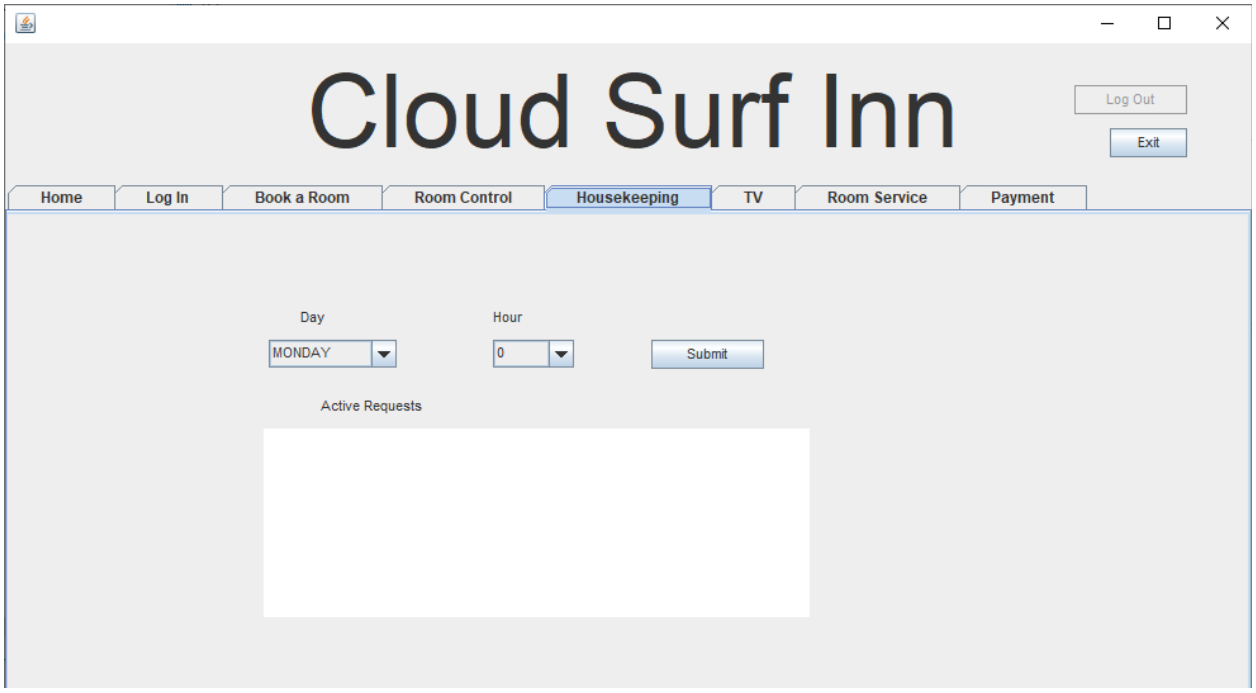
The "book room" page is implemented. It is modeled closely off of the sketch from report 1 and fulfills the requirement of match rooms with use preferences (REQ-19). They then click "book room" to complete and are matched to a room (CREQ-3, CREQ-4, CREQ-5). To book a room, a user needs to click one to log in, and click up to 4 times to select their preferences and type rooms to find a room.

Room Control Page



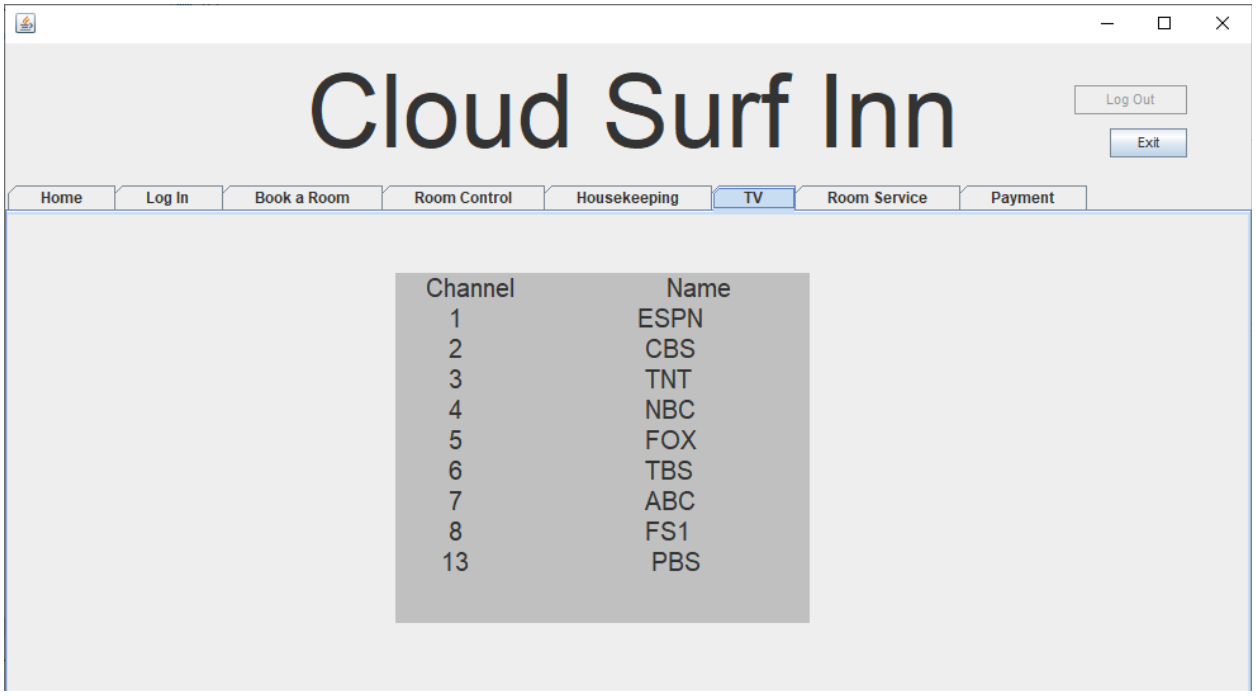
The room control page is implemented to allow users to check in and out (REQ-34, CREQ-8, CREQ-11). The current temperature is shown and the desired temperature can be updated (REQ-7, REQ-8, CREQ-12). Users can go directly to room control to check in/out for a total of two clicks and also update the temperature in a total of 3 clicks.

Housekeeping Page



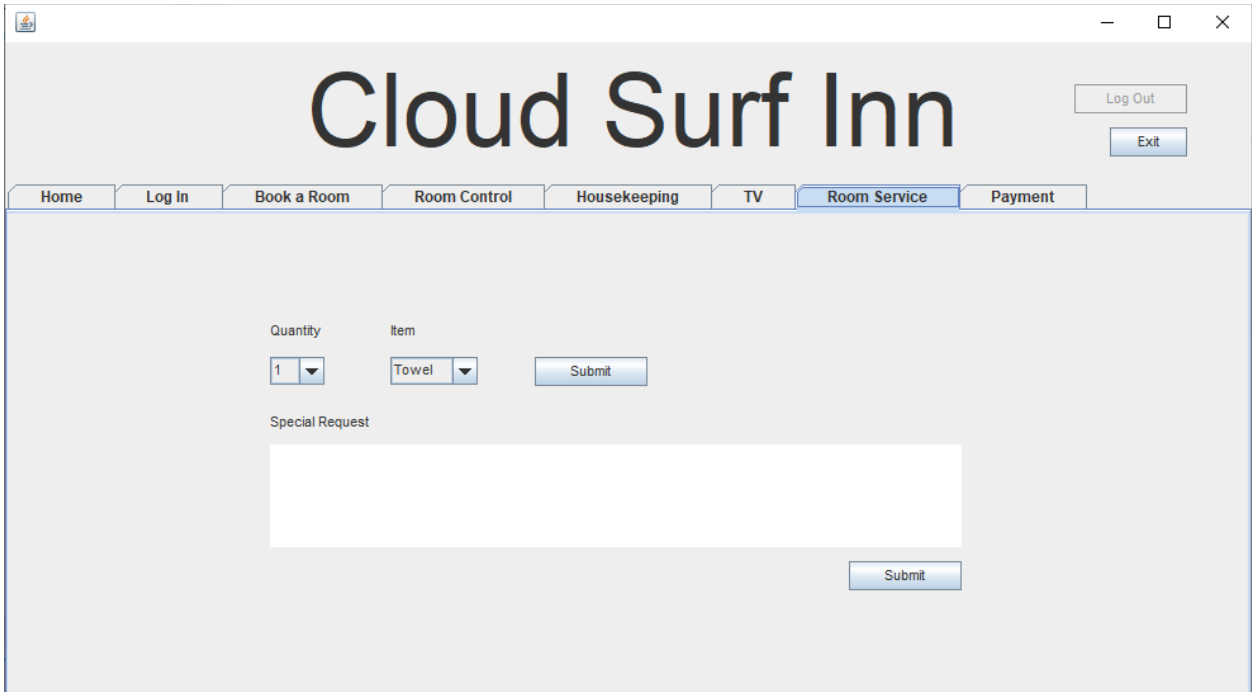
Here customers can select a time and day for a room cleaning (REQ-11, CREQ-9). Customers use up to 5 clicks to request housekeeping, login, the tab selection, a day, hour, and to submit.

TV Page



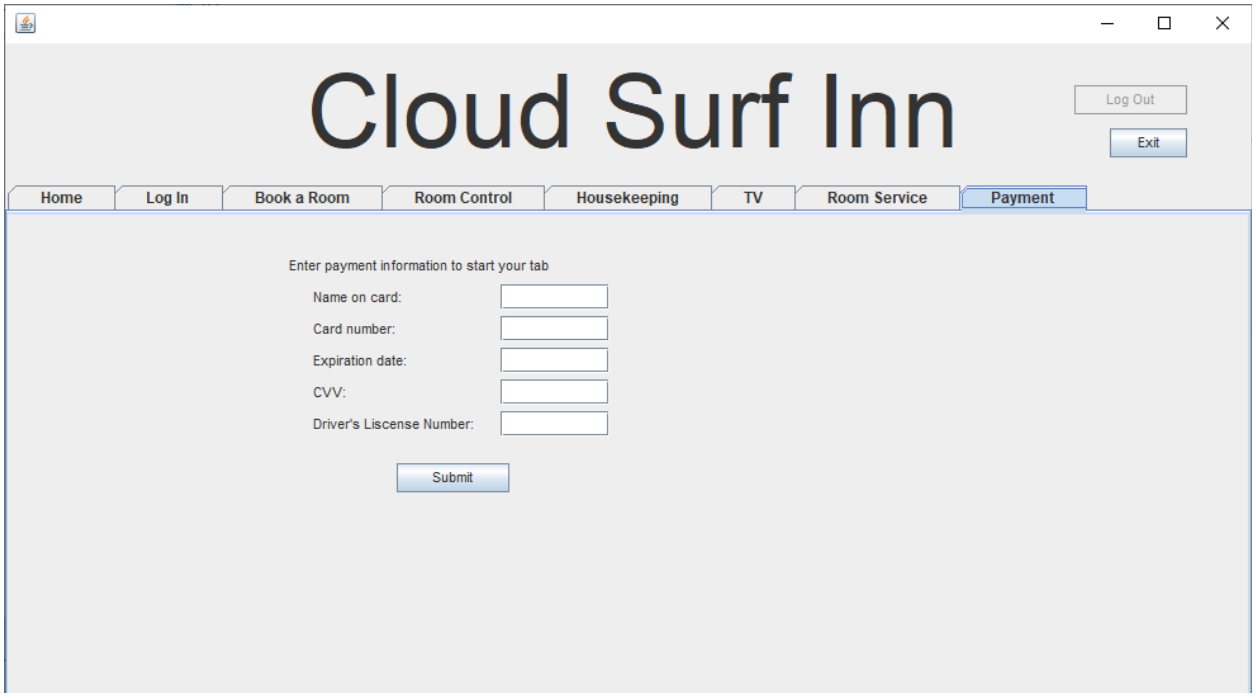
Customers can easily access a TV guide (REQ-9, CREQ-10), all in one click.

Room Service Page



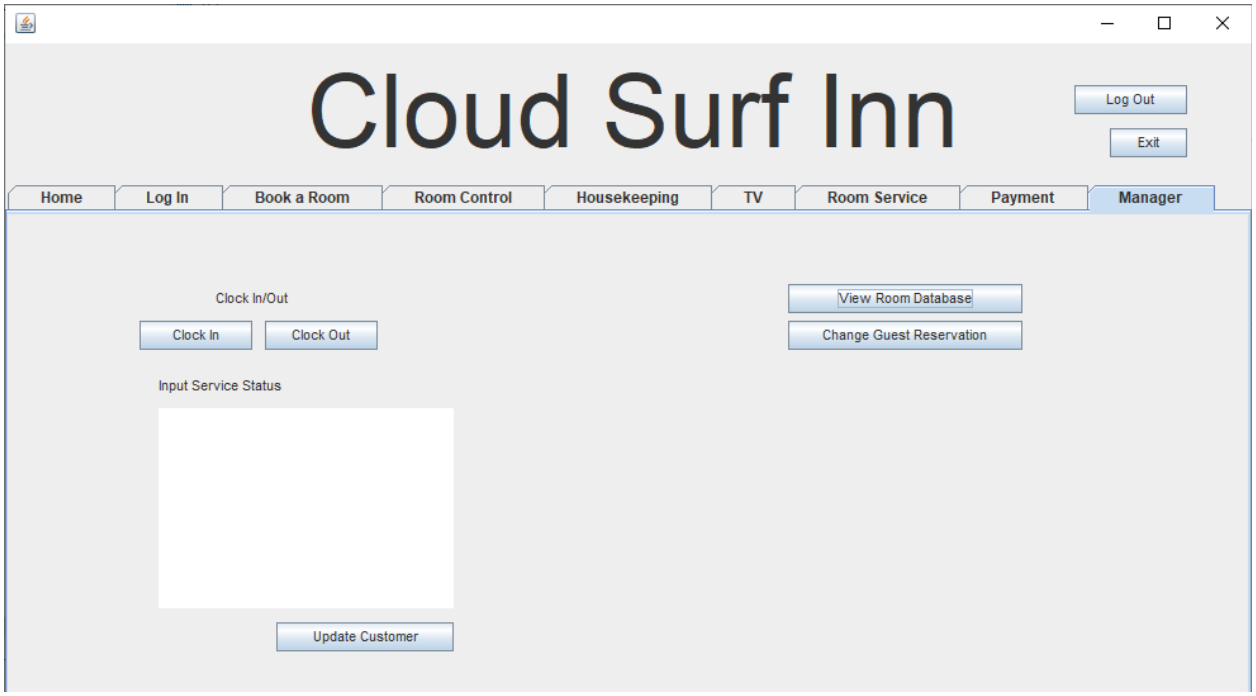
This page allows customers to order food, items, and services (REQ-10, CREQ-13).

Payment Page



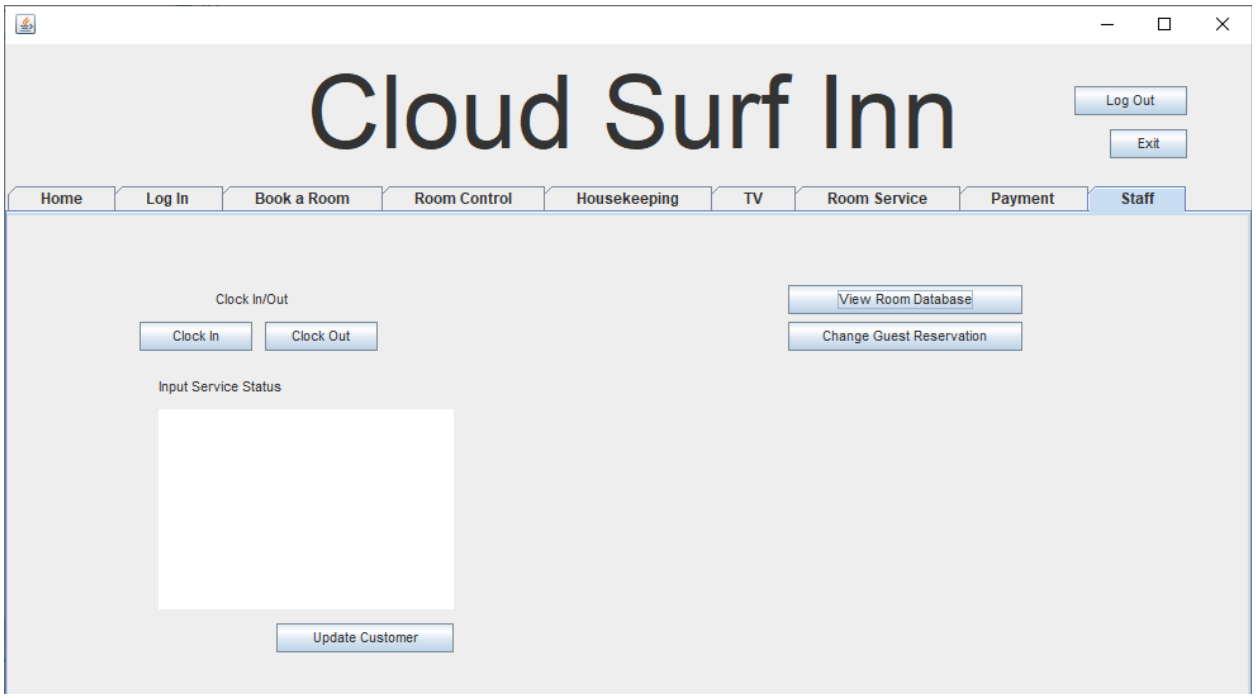
This page allows customers to make purchases during their stay (REQ-16, CREQ-6, CREQ-7). It clearly shows what information must be entered and identifies the customer (REQ-32).

Manager Page



Here managers can access their elevated permissions (MREQ-2). They can view the room database (REQ-15, REQ-17, MREQ-3, MREQ-4) and change room assignments (MREQ-5, MREQ-6). It also allows for control of housekeeping (REQ-12, MREQ-7, MREQ-8) and the master key (MREQ-9).

Staff Page



Here service employees have an easy method of clocking in and out (REQ-38, SREQ-4). They also update guests service requests and view housekeeping requests (REQ-37, SREQ-3, SREQ-5, SREQ-6, SREQ-7, SREQ-8).

7. Design of Tests

Room Matcher:

Use Case	Test Case Description	Test Coverage
UC-1	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, 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.	This test ensures that our matching system is able to meet requirements even if no room matches exactly. 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	A user logs into the system, clicks “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.

Integration Strategy:

To integrate the roomMatching class, we will first design it as a script to run on its own, separately from the database. This will give us a simple way to ensure that the algorithm works, where we can easily initialize and change data without needing interaction with the database. Then, we will be able to edit our code to take proper input from the controller when called.

For the viewing of rooms, we must display database data to the user interface. It is best to wait until those two services have been integrated, then we will be able to pull from the database and output to the user interface with ease. This way, we can ensure that the displaying and data pulling are working, and then implement a simple array printer using the correct inputs and outputs.

Mobile Keys:

Use Case	Test Case Description	Test Coverage
UC-6	The manager will login and be able to see the master key. The housekeeping will login and see the key for the designated room they are trying to clean. The user will be able to login 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 login to their designated accounts.
UC-10	When the customer logs into the page after booking a room, on check-in day the user will be given access to their room. The key they are provided is a random 10 digit pin.	This test will ensure that the key is randomly generated and matches the room given
UC-29	The manager will login and see the master key and be able to change the master key. The master key will grant access to all rooms at all times.	This test will ensure that the manager can access rooms just in case. This is implemented as a failsafe.

Integration Strategy:

To integrate the login page we have created a gui that allows each user to select their profile and gain access to what is on their personal account. For the manager he will see the

masterkey and everything the staff will see, the guest will see what rooms they have access to and the pins associated to that room, and the staff members will be able to view requests and keys for certain rooms to gain access.

To integrate generateKey, we first created a java script that will display a 10 digit alphanumeric pin that will grant access to the room the system assigned to it. We will need to create a database to save all the pins that match each room and allow Guest and Managers to view the key at any time, but only allow staff to view the pin if they are being requested.

To integrate the Master key, in the login page will have a separate tab in the Manager login page labeled masterkey. In that interface the manager has the ability to view the master and change the master key whenever. We created a java script that is similar to that of the generate key, however we also gave the manager the ability to create his own key.

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 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 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.	This test will ensure the housekeeper can accept jobs given to them by the customers.

Integration Strategy:

To integrate the Housekeeping system we will need to implement three classes into the main project. The classes will be implemented so they work separately just as well as if they worked

together. The housekeeping heap used for storing all submitted requests by the customer, will eventually be stored in a separate database from all other information. The individual housekeeper heaps will be included in this database as well. Eventually the manager will be able to utilize all of these functions as well.

Power Controls:

Use Case	Test Case Description	Test Coverage
UC-10	The device control class will be asked to read the ambient temperature and make sure that the reading makes sense. It will then be asked to set an ambient temperature and ensure that it stays that way.	This test covers all the requirements that this class must perform. We ensure that the class is able to read the temperature from the sensor and also set the temperature.
UC-13	The power control class will take as input the parameter “vacancy” as true in one test, and false as the other test. The output should enact the power saving mode when vacancy is true, and comfort mode when vacancy is false, i.e. turn off the heat in a vacant room, and turn it back on 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 vacancy parameter is sent from the controller, the power control responds appropriately.
UC-13	The main controller will be asked to retrieve the information about a room number, notifying the power control if the room is vacant and ensuring that power control receives the right device number.	This test covers the responsibilities of the main controller for this UC. It will have to notify the power control if a room it retrieves is vacant and tell power control what device to contact.

Integration Strategy:

The integration strategy of these separate classes and objects will be done from a bottom-up approach. In other words, we will make sure each class operates separately. Then, we will start with the interaction of the power control to the device control, ensuring that the power control asks the device controller to turn on/off the temperature appropriately depending on if the vacancy variable is set to true or false with a predefined device number. Then, we will continue

up to the main controller which will have to notify the power controller of a vacancy, as well as which device to connect to. Doing it this way, we can ensure that lower level communications are functioning properly and so while going upwards we can test an individual connection each time and see whether the object all the way at the end is still functioning as intended.

8. Project Management and Plan of Work

8.1 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 ensured that it was followed. 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.

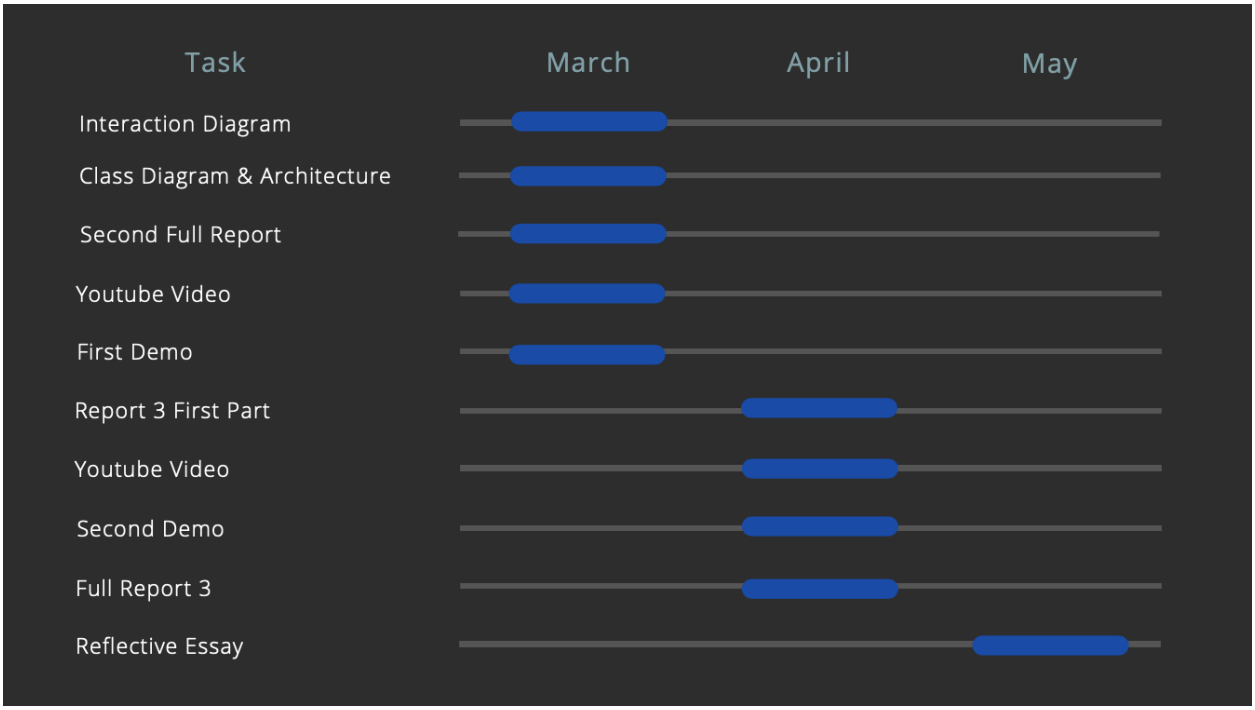
8.2 Project Coordination and Progress Report

Team Code	Use Cases	Implemented
Team A	UC-25 UC-16	Demo 1 Demo 1

	UC-26	Demo 2
	UC-21	Demo 2
Team B	UC-10	Demo 1
	UC-6 UC-7	Demo 1 Demo 1
Team C	UC-13 UC-14	Demo 1 Demo 2
	UC-11 UC-12	Demo 1 Demo 1
Team D	UC-1 UC-2	Demo 1 Demo 1
	UC-5	Demo 2

8.3 Plan of Work

The rest of the week our group plans on meeting to perform integration testing on the system and record the demo. We will also be discussing edge cases and alternative plans to optimize our system functionality. Primary developers will be ensuring the production of products that are systematic.



8.4 Breakdown of Responsibilities

The integration of our subsystems is being coordinated by Chris and Ryder. Chris will be performing the integration testing in regards to the database, and Ryder is performing the integration testing on the user interface along with being the primary developer for the user interface. Chynna and Jakub are aiding in developing the integration of all subsystems and ensuring consistency throughout the project.

Breakdown of Responsibilities of Classes:

Class	Developer	Coder	Tester
RoomMatching	JP	Chris	Chris
CustomerProfile	JP	Chris	Chris
GenerateKey	Bharath	Bharath	Sidonia
GenerateMKey	Bharath	Bharath	Sidonia
StaffProfile	Bharath	Bharath	Sidonia

CleaningNode	Ryder	Juan	Zach
HPHeap	Ryder	Juan	Zach
HKPRHeap	Ryder	Juan	Zach
Controller	JP	Chris	Chris
RoomStorage	JP	Chris	Chris
PowerCtrl	Jakub	Sebastian	Chynna
DeviceCrtl	Jakub	Sebastian	Chynna

8.5 Work Assignment

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 three main solutions are:

1. Checking out of the room makes the database switch to vacant which turns off power in the hotel room.
2. A guest logs into the system and is matched with a room based on preferences, then a mobile key is outputted.
3. Guests make a housekeeping request, housekeeping logs into the system, and reviews requests.

The functional features that address the problems are labeled as, “Housekeeping”, “Mobile Keys”, “Power Control”, 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. Each team has more responsibilities and additional functional features that optimize their subsystem, but will focus on one of the core four functional features; online ordering falls under housekeeping, management falls under mobile keys, and payment falls under room matcher. 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 Control	Jakub, Sebastian, Chynna	Team C
Room Matcher	JP, Chris	Team D

Team Members are as followed:

Chynna Walsh - Management & Organization, C++

Sebastian Matiz - Programming, Java, C

Juan Escudero - Writing and Documentation, Java, C, Python

Bharath Selvaraj - Programming, Java, C++

Sidonia Mohan - Java, JavaScript, HTML/CSS

Chris Kline - Eclipse, SQL, Java, C++

Ryder Morrello - Java, JavaScript

JP Dangler - Documentation, C++, C

Jakub Vogel - Report Writing, C++, Java

Zach LeMunyon - Programming

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/

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