Project Deliverables

Version 04
05/02/2017

Minoa's Reservation System

Object Oriented Software Engineering
SE 6329 – R. Z. Wenkstern

Preethi Sekar
Umang Shah
Nishigandha Narendra Rane
Radhika Kalaiselvan
Gajanan Golegaonkar

Table of Contents

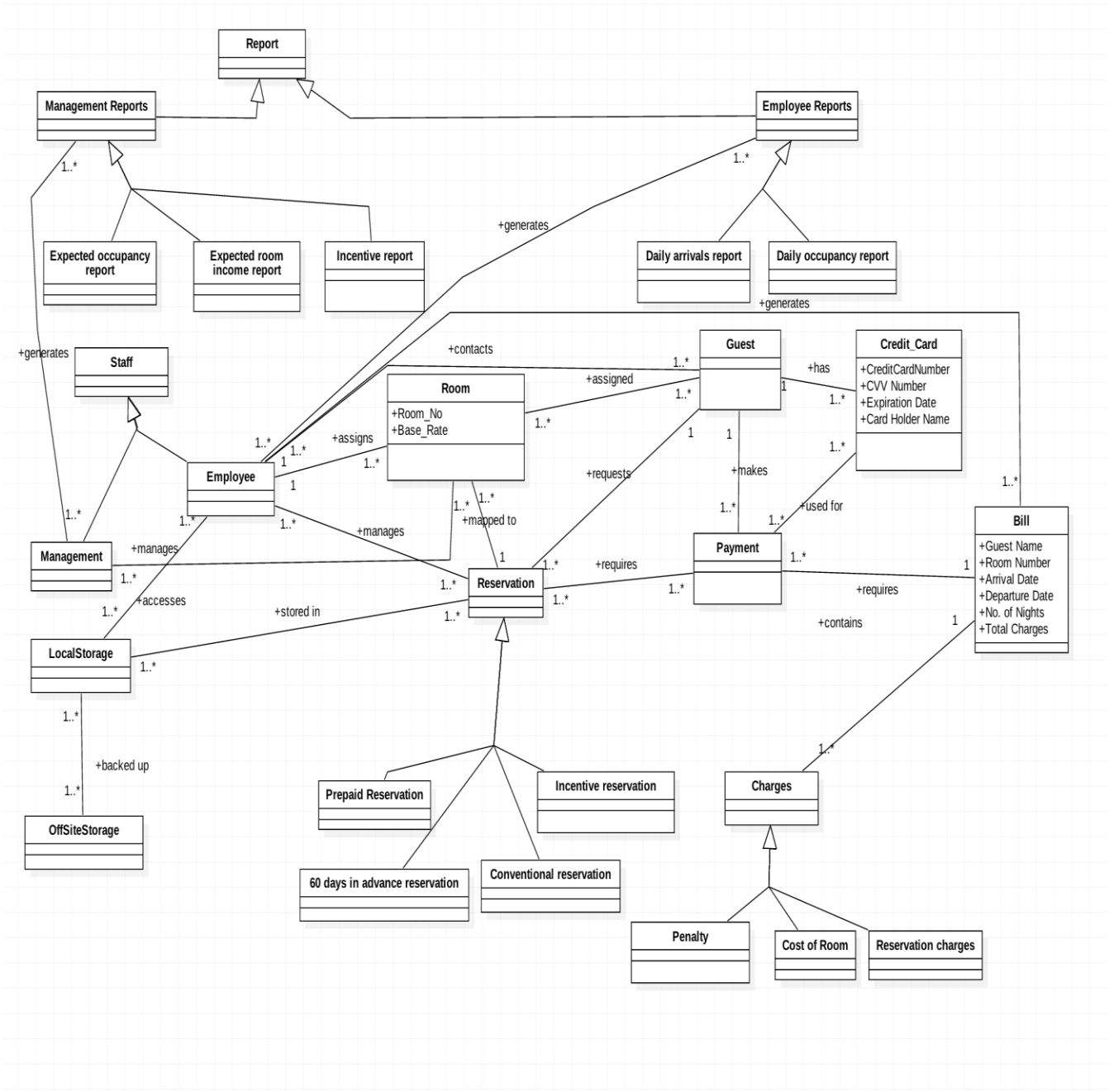
Part I: Use Case Analysis

Version 4.0
05/02/2017

Revision History

Version - description	Date	Description	Author
V 1.0 - Inception draft	04/10/2017	First draft. To be refined primarily during elaboration.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar
V 2.0 - Elaboration 1 draft 1	04/18/2017	Refined Use Case Model and Domain Model	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar
V 3.0 - Elaboration 2 draft 1	04/22/2017	1.Small modifications to the Domain model 2.Small modifications to the UCD. 3.The Black Box Sequence Diagrams were fine tuned to reflect the Use Case Descriptions. 4.The Operation Contracts were updated accordingly and made more specific and technical.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar
V 4.0 - Elaboration 3 draft 1	05/02/2017	1.Minor modifications to the use case description. 2.Black Box and White Box Sequence Diagrams are changed accordingly.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar

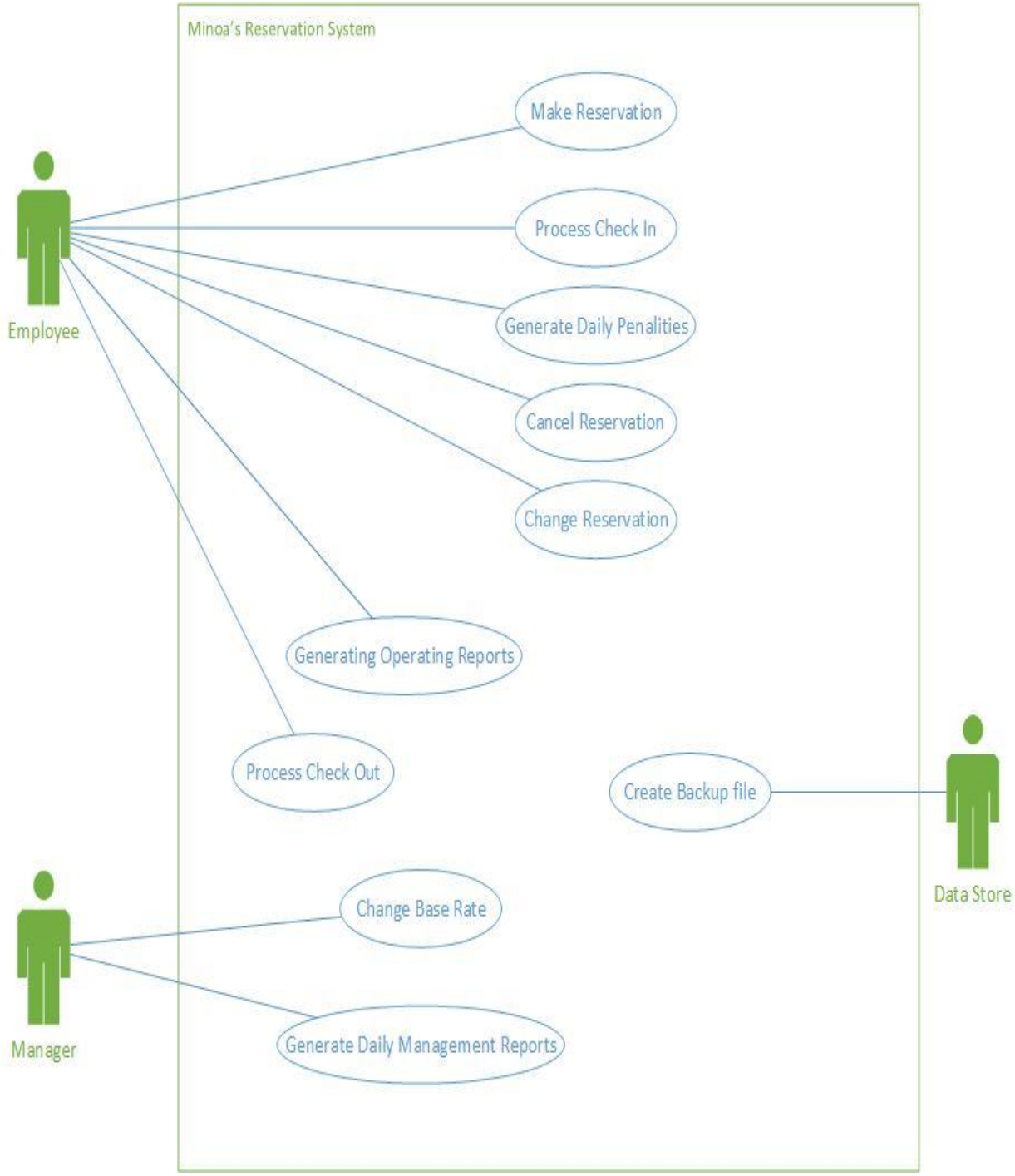
1. Domain Model



Use Case Diagrams (UCD)

1.1. UCD Level 0

2.1.1 Diagram



2.1.2 UC Descriptions in brief format

MakeReservation:
The employee must be able to make a reservation for the guest through the Minoa's Reservation System. The type of reservation is decided by the employee when the customer contacts the hotel to make a reservation.

ProcessCheckIn:
The Employee is responsible for checking in the guest. The Employee requests for the guest's identification and the other details and check in the quest into the room.

GenerateDailyPenalties:
The employee is responsible for generating the daily penalties. For instance, if the guest does not show up for check in after the reservation I made, the employee generates a "no show" penalty.

CancelReservation:
The Employee cancels a reservation for the guest. The employee gets the personal information and other room details to cancel a reservation.

ChangeReservation:
The Employee changes a reservation for the guest. The employee gets the personal information and other room details to changes a reservation.

GeeratingOperatingReports:
The Employee is responsible for creating three main operating reports. The expected occupancy report that shows the number of rooms currently reserved each night for the next 30 days. The expected room income report which describes the expect income for the expected occupancy report. The incentive report which describes the amount of money lost for next 30 days because of the incentive reservations.

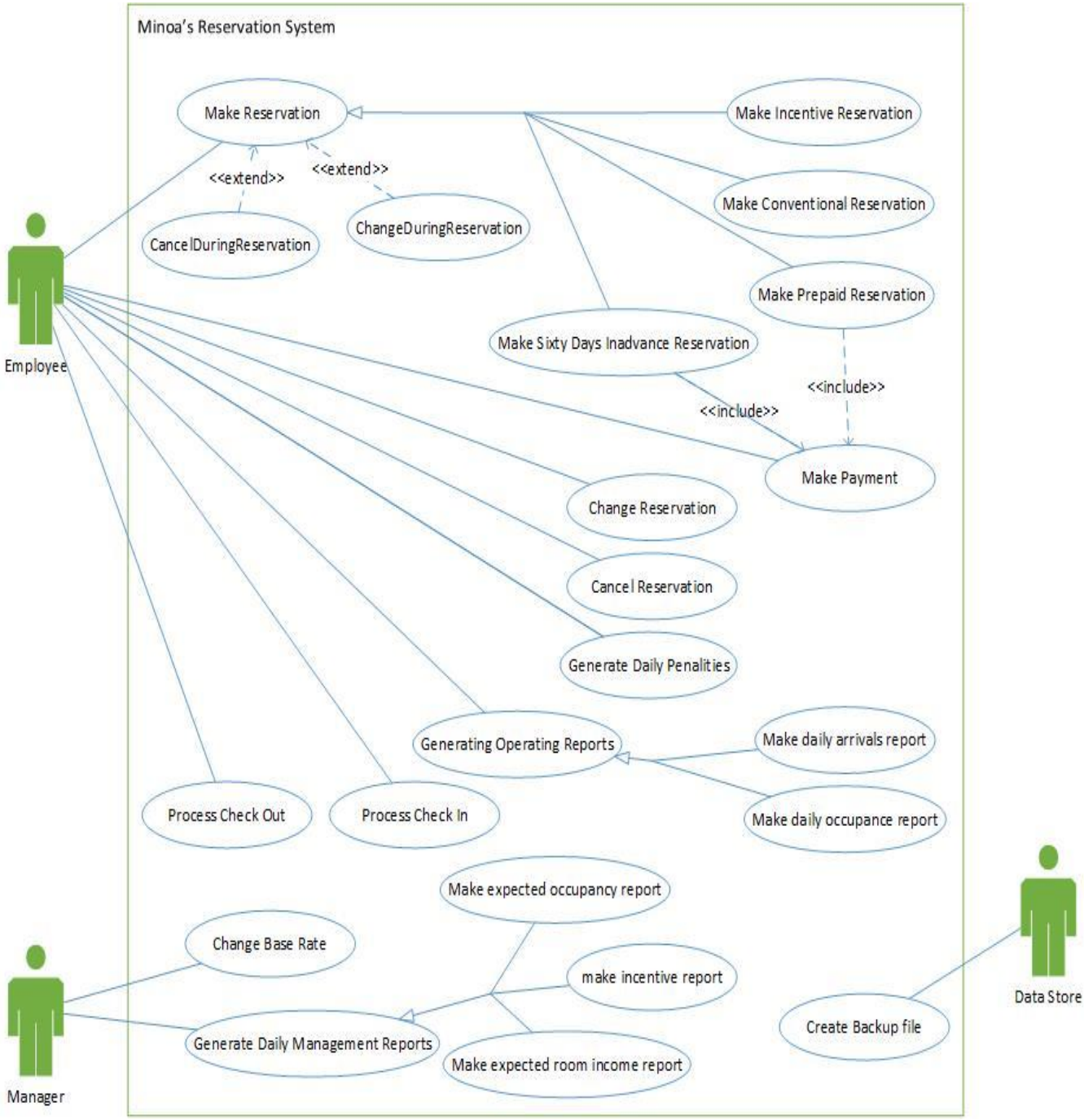
PocessCheckOut:
The employee is responsible for checking out a guest and processing the payment.

CreateBackupFile:
The System creates a backup of all the files each day and stores it away offsite.

ChangeBaseRate:
The Management fixes the base rate each day. The base rate is fixed based on the staffing, availability of rooms and various other factors.

GenerateDailyManagementReports:
The Management is responsible for generating two major management reports, the daily arrivals report which describes when the guests are expected to arrive and the daily occupancy report which lists the guests currently staying at the hotel.

1.2. UCD Level 1



1.3. UCD Prioritization

UC Prioritization (Level 0)

Fully Dressed	Casual	Brief
1. Make Reservation	3. Process Check-in	6. Generate Daily Penalty Charges
2. Cancel Reservation	4. Process Check-out	7. Generate Operating Reports
	5. Change Base Rate	8. Generate Daily Management Reports
		9. Create backup files

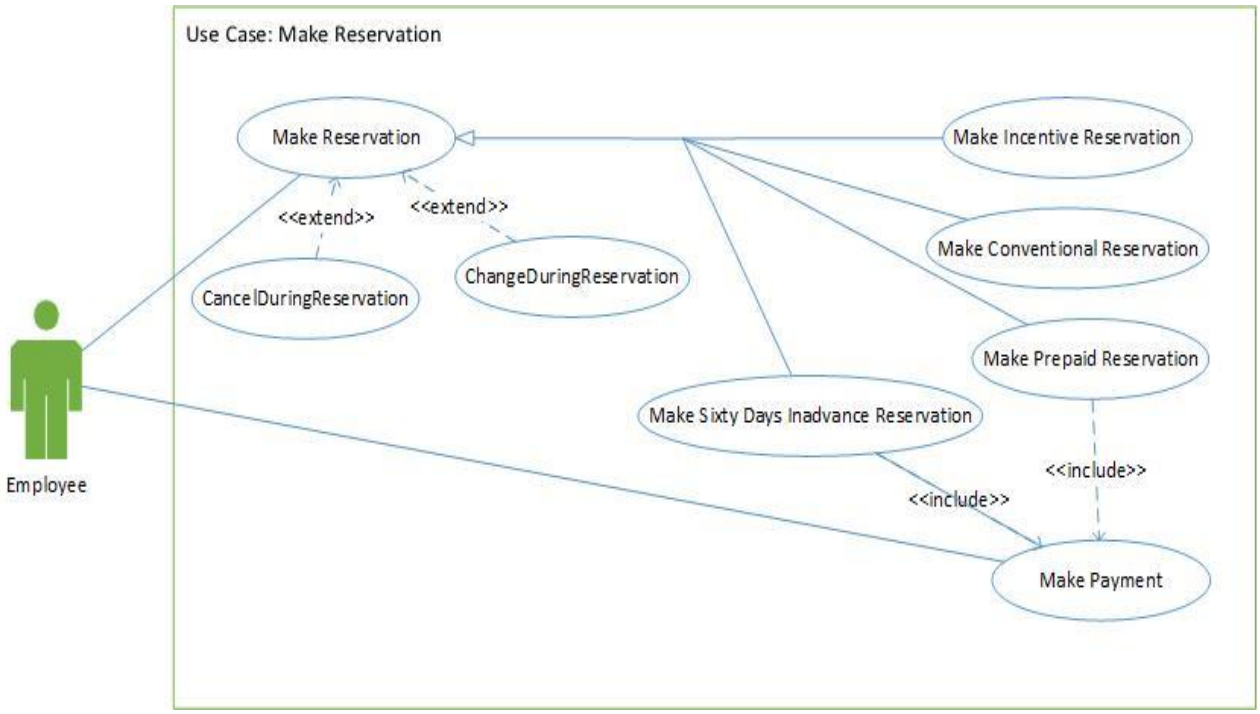
"Make Reservation" UC Prioritization (Level 1)

Fully Dressed	Casual	Brief
1. Make Conventional Reservation	5. Make Payment	6. Change During Reservation
2. Make Incentive Reservation		7. Cancel During Reservation
3. Make 60 day advance Reservation		
4. Make Prepaid Reservation		

2. Use Case Descriptions

2.1. UC Make Reservation

3.1.1 Level 1 UCD for UC Make Reservation



3.1.2 UC Make Reservation Main Flow

Use Case: MakePrepaidReservation
ID: 1
Brief Description: Guest makes a reservation 90 days prior to start of their stay
Primary Actors: Employee
Secondary Actors:
Preconditions: Employee should be logged in to the system.
Main flow: 1. Guest contacts the Minoa's Employee to make a reservation. 2. Employee requests Guest for personal details. 3. Employee requests Guest for the date. 4. Employee checks for the room availability. 5. System checks if rooms are available.

6. System calculates the difference between current date and date of the reservation.
7. System enables the reservation options. (PrepaidReservation Option is enabled only when difference is 90 days away and if the rooms are available)
8. Employee initiates to make a prepaid reservation.
9. Employee asks for details.
10. Guest provides the details.
11. Employee enters the details in the system.
12. System calculates the total cost of the stay.
13. Include(MakePayment)
14. Employee confirms reservation.
15. Guest confirms booking details.

extensionPoint: guestDoesNotWantToBook, Extension: CancelDuringReservation
extensionPoint: guestWantsToChangeDuringReservation, Extension: ChangeDuringReservation

Postconditions:
1. A reservation is made in the name of the guest.
2. The number of available rooms are updated.

Open Issues:

Use Case: MakeSixtyDaysInAdvanceReservation
ID: 2
Brief Description: Guest makes a reservation 60 days prior to check-in.
Primary Actors: Employee
Secondary Actors:
Preconditions: 1. Employee should be logged in to the system.
Main flow: 1. Guest contacts the Minoa's Employee to make a reservation. 2. Employee requests Guest for personal details. 3. Employee requests Guest for the date. 4. Employee checks for the room availability. 5. System checks if rooms are available. 6. System calculates the difference between current date and date of the reservation. 7. System enables the reservation options. (SixtyDayInAdvanceReservation Option is enabled only when difference is 60 days away and if the rooms are available) 8. Employee initiates to make a SixtyDaysInAdvance reservation. 9. Employee asks for email id, additional details. 10. Guest provides the details. 11. Employee enters the details in the system. 12. Employee confirms reservation. 13. Guest confirms booking details. 14. System calculates the total cost of the stay. 15. System adds guest to email reminder list for the date. 16. Guest contacts the employee within 15 days. 17. include (MakePayment) 18. Employee confirms reservation. 19. Guest confirms booking details. extensionPoint: guestDoesNotWantToBook, Extension: CancelDuringReservation

extensionPoint: guestWantsToChangeDuringReservation, Extension: ChangeDuringReservation
Postconditions: 1. A reservation is made in the name of the guest. 2. The number of available rooms are updated.
Open Issues: 1. 45 days before the booking date, employee generates reminder emails. 2. Reminder email is sent to the guest to make a payment.

Use Case: MakeConventionalReservation
ID: 3
Brief Description: Guest makes a conventional reservation.
Primary Actors: Employee
Secondary Actors:
Preconditions: 1. Employee should be logged in to the system.
Main flow: 1. Guest contacts the Minoa's Employee to make a reservation. 2. Employee requests Guest for personal details. 3. Employee requests Guest for the date. 4. Employee checks for the room availability. 5. System checks if rooms are available. 6. System calculates the difference between current date and date of the reservation. 7. System enables the reservation options. (ConventionalReservation Option is enabled only if the rooms are available) 8. Employee initiates to make a Conventional reservation. 9. Employee asks for details. 10. Guest provides the details. 11. Employee enters the details in the system. 12. System calculates the total cost of the stay. 13. Employee asks for credit card details. 14. Guest provides the credit card details. 15. Credit card details are stored. 16. Employee confirms reservation. 17. Guest confirms booking details. extensionPoint: guestDoesNotWantToBook, Extension: CancelDuringReservation extensionPoint: guestWantsToChangeDuringReservation, Extension: ChangeDuringReservation
Postconditions: A reservation is made in the name of the guest. The number of available rooms are updated.
Open Issues:

Use Case: MakeIncentiveReservation
ID: 4
Brief Description: An Incentive reservation is made for the guest by the employee.
Primary Actors:

Employee
Secondary Actors:
Preconditions: 1. Employee should be logged in to the system. 2. There should be less than 60% occupancy.
Main flow: 1. Guest contacts the Minoa's Employee to make a reservation. 2. Employee requests Guest for personal details. 3. Employee requests Guest for the date. 4. Employee checks for the room availability. 5. System checks if rooms are available. 6. System calculates the difference between current date and date of the reservation. 7. System enables the reservation options. (IncentiveReservation Option is enabled only if the rooms are available) 8. Employee initiates to make an Incentive reservation. 9. Employee asks for details. 10. Guest provides the details. 11. Employee enters the details in the system. 12. System calculates the total cost of the stay. 13. Employee asks for credit card details. 14. Guest provides the credit card details. 15. Employee enters the credit card details. 16. Credit Card Details are stored. 17. Employee confirms reservation. 18. Guest confirms booking details. extensionPoint: guestDoesNotWantToBook, Extension: CancelDuringReservation extensionPoint: guestWantsToChangeDuringReservation, Extension: ChangeDuringReservation
Postconditions: 1. A reservation is made in the name of the guest. 2. The number of available rooms are updated.
Open Issues:

3.1.3 UC Make Reservation «Includes»

Use Case: MakePayment
ID: 5
Casual Description: The system charges the guest and processes the payment. 1. Employee initiates payment process 2. Employee searches for the guest reservation record. 3. The reservation details are displayed. 4. System asks the employee for confirmation. 5. Employee confirms the charges. 6. Employee gets the credit card details. 7. Employee enters the credit card details into the system. 8. System stores the credit card details. 9. The system charges from the credit card user provided. 10. Update the system record. 11. System displays the success message.

3.1.4 UC Make Reservation «Extends»

Use Case: ChangeDuringReservation
ID: 6
Description: The employee changes the reservation for the guest. The employee checks for room availability for the new booking date. Then proceeds to make the modification under suitable conditions. The employee then records the payment and the changes are confirmed in the system.

Use Case: CancelDuringReservation
ID: 7
Description: The employee cancels the reservation when the guest requests. Employee clears the previously made progress and aborts without making any reservation.

3.1.5 UC Make Reservation Supplementary Specification

Product Non-Functional Requirements:

Reliability:

Details of reservation once made should be always reflected in daily processing.

Recoverability:

Local storage logs all details about transaction and records of transaction is backed up on an off-site storage. Logs can be used to recover any lost reservation details.

Usability:

• System should ask for confirmation before important tasks like booking confirmation, payment processing, and cancellation.

• Employee should be able to navigate the sys Product Non-Functional Requirements:

Reliability:

Details of reservation once made should be always reflected in daily processing.

Recoverability:

Local storage logs all details about transaction and records of transaction is backed up on an off-site storage. Logs can be used to recover any lost reservation details.

Usability:

• System should ask for confirmation before important tasks like booking confirmation, payment processing, and cancellation.

• Employee should be able to navigate the system.

• Employee must be able to retrieve guest information from the system for change or cancellation tasks.

• Good GUI should be available as Employees are not technically sound.

Security:

Employee must have proper authorization for each task.

Process Non-Functional Requirements:

Implementation Constraints:

Minoa does not have very high end computer systems so the operating logic of program must be light weight and be handled on generic PC setup. Example: 2GB RAM, 500 GB HDD, i3 Processor.

Free or Open Source and Other Components:

• We are using MYSQL Database for storing essential Hotel Information about Employee, Managers their credentials, Room details and also maintaining details about Reservations.

• We will use XYZ open source tool as a payment gateway.

External Non Functional Requirements:

Legal Issues:

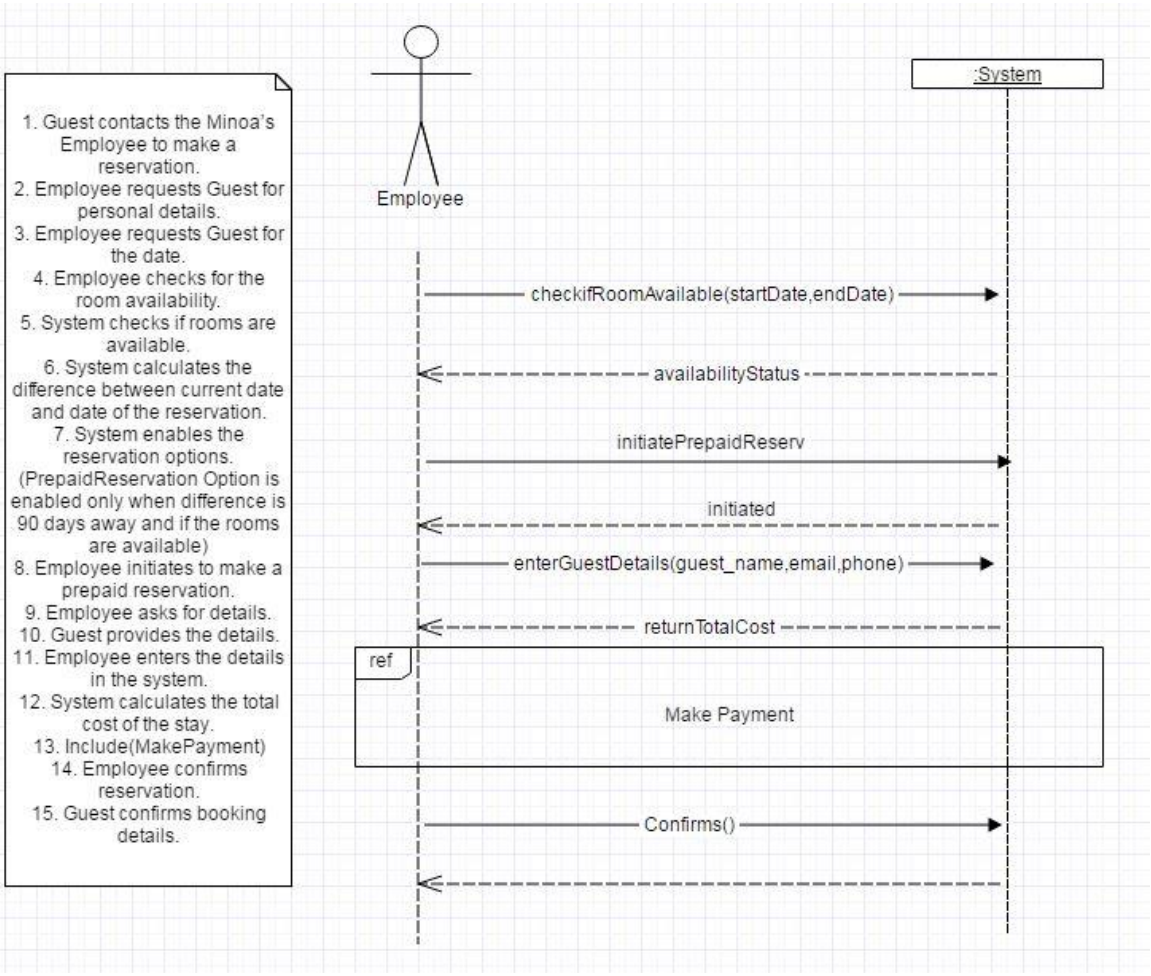
• Guest should be notified that credit card details are being stored by the system till the end of their reservation.

• All payments must account for additional taxes

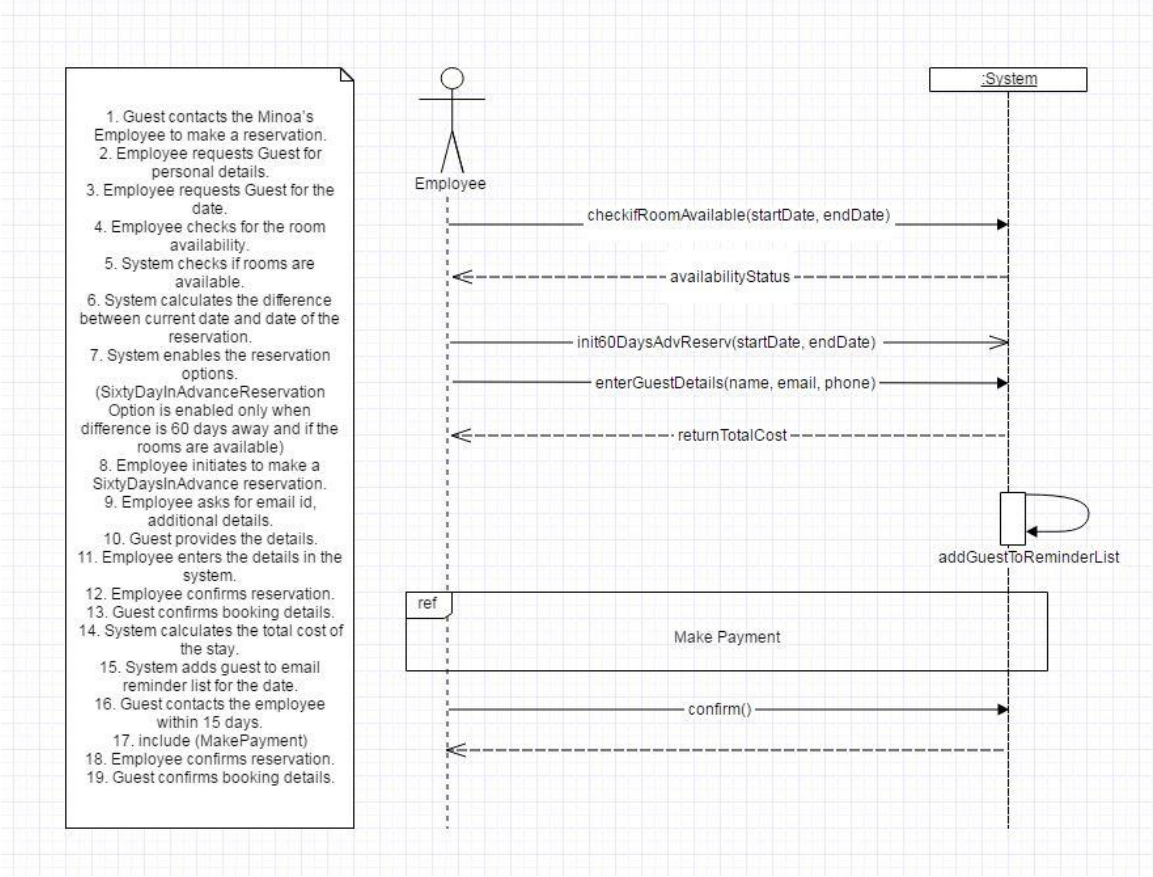
3. System Sequence Diagrams

3.1.1. UC Make Reservation

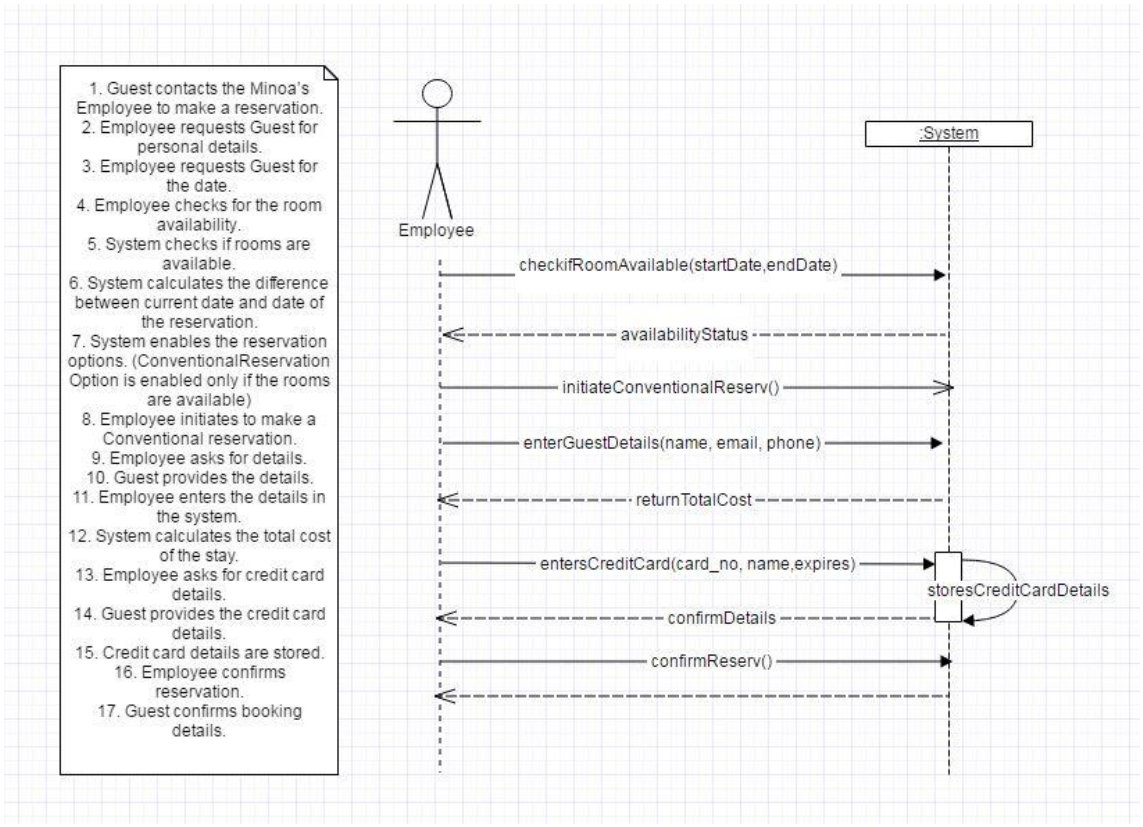
Make Prepaid Reservation:



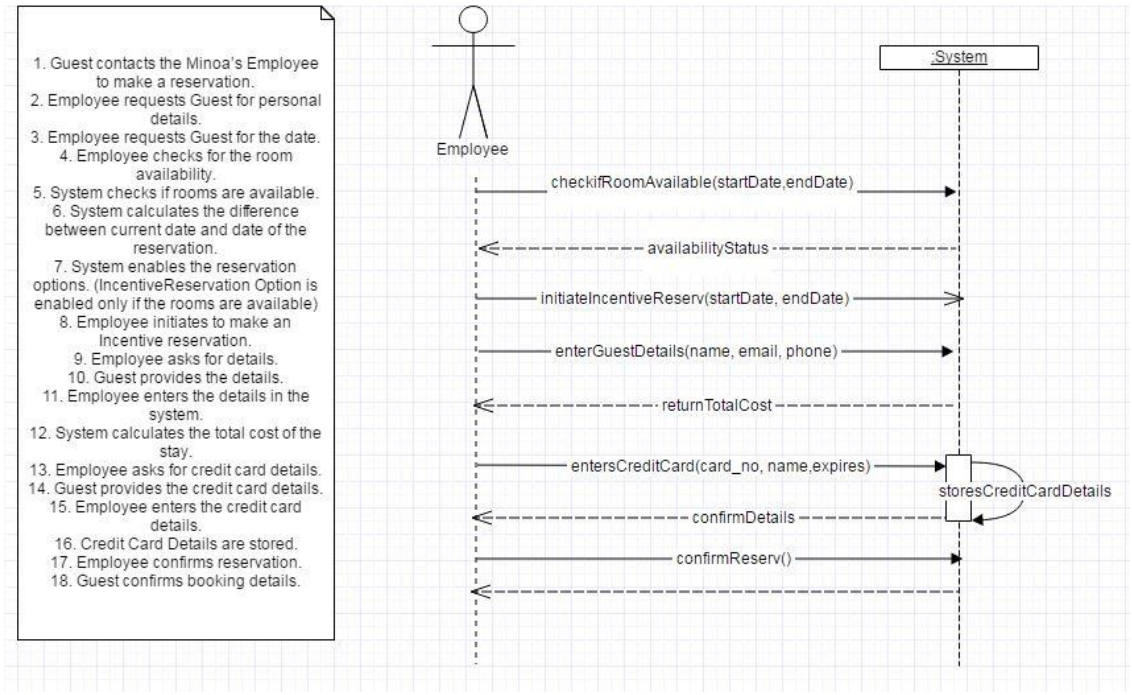
Make Sixty Day Reservation:



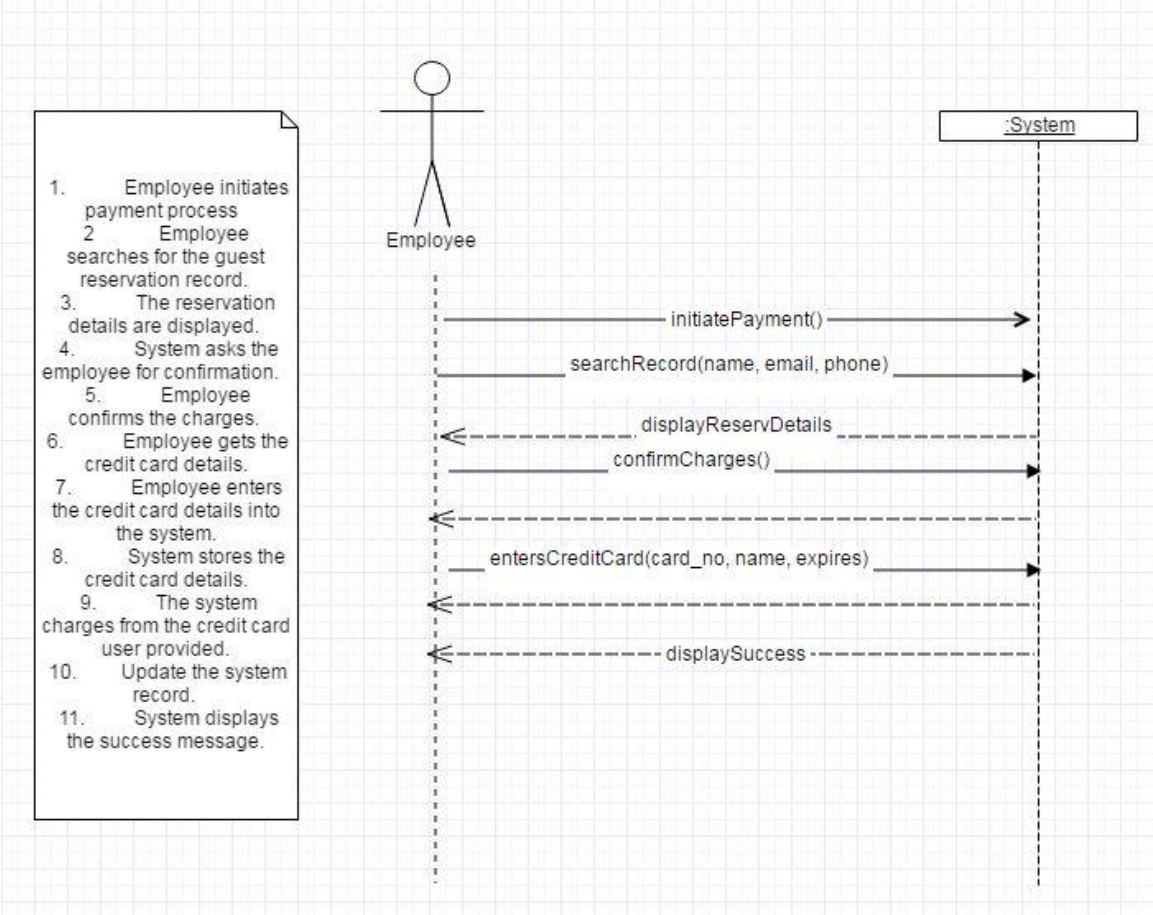
Make Conventional Reservation:



Make Incentive Reservation:



Make Payment:



4. Operation Contracts

Use Case: MakePrepaidReservation

Contract CO1: CheckIfRoomAvailable(startDate,endDate)
Cross Reference: MakePrepaidResevation
Preconditions: 1. Employer is logged into the system
2. Guest requested to book a room
Postconditions:

Contract CO2: initiatePrepaidReservation(startDate,endDate)
Cross Reference: MakePrepaidResevation
Preconditions: 1. Room is available and guest is interested.
Postconditions: 1. an instance of PrepaidReservation r is created by the controller.
2. the attributes of PrepaidReservation startDate,endDate are initialized.

Contract CO3: enterGuestDetails(name, email, phone)
Cross Reference: MakePrepaidResevation
Preconditions: 1. There is a reservation underway.
Postconditions: 1. an instance of guest g is created.
2. Attributes of g, name,email,phone are initialized.
3.g is associated with r.

Contract CO4: confirmResv()
Cross Reference: MakePrepaidResevation
Preconditions: 1. There is a reservation underway.
2. Reservation details are available.
Postconditions: 1. Update RoomMgr attribute No_of_ available_rooms.

Use Case: MakeSixtyDaysReservation
Contract CO1: CheckIfRoomAvailable(startDate,endDate)
Cross Reference: MakeSixtyDaysResevation
Preconditions: 1. Employer is logged into the system
2. Guest requested to book a room
Postconditions:

Contract CO2: init60DaysAdvReserv(startDate, endDate)
Cross Reference: MakeSixtyDaysResevation
Preconditions: 1. Room is available and guest is interested
Postconditions: 1. an instance of 60DaysAdvanceReservation r is created by the controller.
2. the attributes of 60DaysAdvanceReservation startDate,endDate are initialized.

Contract CO3: enterGuestDetails(name, email, phone)
Cross Reference: MakeSixtyDaysResevation
Preconditions: 1. There is a reservation underway
Postconditions: 1. an instance of guest g is created.
2. Attributes of g, name, email, phone are initialized.
3.g is associated with r.

Contract CO4: confirmResv()
Cross Reference: MakeSixtyDaysResevation
Preconditions: 1. There is a reservation underway.
2. Reservation details are available.

Postconditions: 1. Update RoomMgr attribute No_of_ available_rooms.

Use Case: MakeConventionalReservation
Contract CO1: CheckIfRoomAvailable(startDate, endDate)
Cross Reference: MakeConventionalReservation
Preconditions: 1. Employer is logged into the system
2. Guest requested to book a room
Postconditions:

Contract CO2: initiateConventionalReservation(startDate, endDate)
Cross Reference: MakeConventionalReservation
Preconditions: 1. Room is available and guest is interested
Postconditions: 1. an instance of ConventionalReservation r is created by the controller.
2. the attributes of ConventionalReservation startDate,endDate are initialized.

Contract CO3: enterGuestDetails(name, email, phone)
Cross Reference: MakeConventionalReservation
Preconditions: 1. There is a reservation underway
Postconditions: 1. an instance of guest g is created.
2. Attributes of g, name, email, phone are initialized.
3.g is associated with r.

Contract CO4: enterCreditCard(card_no,name,expires)
Cross Reference: MakeConventionalReservation
Preconditions: 1. There is a reservation underway
2. Customer details are entered and available
Postconditions: 1. an instance of Credit Card c is created.
2. Attributes of c, the card_no, name, expires are stored.
3. c is associated with g.

Contract CO5: confirmResv()
Cross Reference: MakeConventionalReservation
Preconditions: 1. There is a reservation underway.
2. Reservation details are available.
Postconditions: 1. Update RoomMgr attribute No_of_ available_rooms.

Use Case: MakeIncentiveReservation
Contract CO1: CheckIfRoomAvailable(startDate,endDate)
Cross Reference: MakeIncentiveReservation
Preconditions: 1. Employer is logged into the system

2. Guest requested to book a room

Postconditions:

Contract CO2: initiateIncentiveReservation(startDate,endDate)
Cross Reference: MakeIncentiveReservation
Preconditions: 1. Room is available and guest is interested
Postconditions: 1. an instance of ConventionalReservation r is created by the controller.
2. the attributes of ConventionalReservation startDate,endDate are initialized.

Contract CO3: enterGuestDetails(name, email, phone)
Cross Reference: MakeIncentiveReservation
Preconditions: 1. There is a reservation underway
Postconditions: 1. an instance of guest g is created.
2. Attributes of g, name, email, phone are initialized.
3.g is associated with r.

Contract CO4: enterCreditCard(card_no,name,expires)
Cross Reference: MakeIncentiveReservation
Preconditions: 1. There is a reservation underway
2. Customer details are entered and available
Postconditions: 1. an instance of Credit Card c is created.
2. Attributes of c, the card_no, name, expires are stored.
3. c is associated with g.

Contract CO5: confirmResv()
Cross Reference: MakeIncentiveReservation
Preconditions: 1. There is a reservation underway.
2. Reservation details are available.
Postconditions: 1. Update RoomMgr attribute No_of_ available_rooms.

Use Case: MakePayment
Contract CO1: initiatePayment
Cross Reference: MakePayment
Preconditions: 1. Employer is logged into the system
Postconditions: 1.an instance of payment p is created.

Contract CO2: searchRecord
Cross Reference: MakePayment
Preconditions: 1. Employer is logged into the system
2.Reservation Record is available.

Postconditions:

Contract CO3: confirmCharges
Cross Reference: MakePayment
Preconditions: 1. Employer is logged into the system
2.Reservation Record is available.
Postconditions:

Contract CO4: enterCreditCard(card_no,name,expires)
Cross Reference: MakeIncentiveReservation
Preconditions: 1. There is a reservation underway
2. Customer details are entered and available
Postconditions: 1. an instance of Credit Card c is created.
2. Attributes of c, the card_no, name, expires are stored.
3.c is associated with g.
4. Payment p is associated with c.
5. c 's balance is updated.

Part II: Use Case Design

Version 3.0
05/02/2017

Revision History

Version	Date	Description	Author
1.0	04/18/2017	First draft. To be refined primarily during elaboration.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar
2.0	04/22/2017	1.Logical View that is the component subsystem diagram was modified to represent the subsystems correctly. 2.use Case Realizations and Design Class Diagrams were created.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar
3.0	05/02/2017	Minor Changes to the deployment view based on the implementation.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar

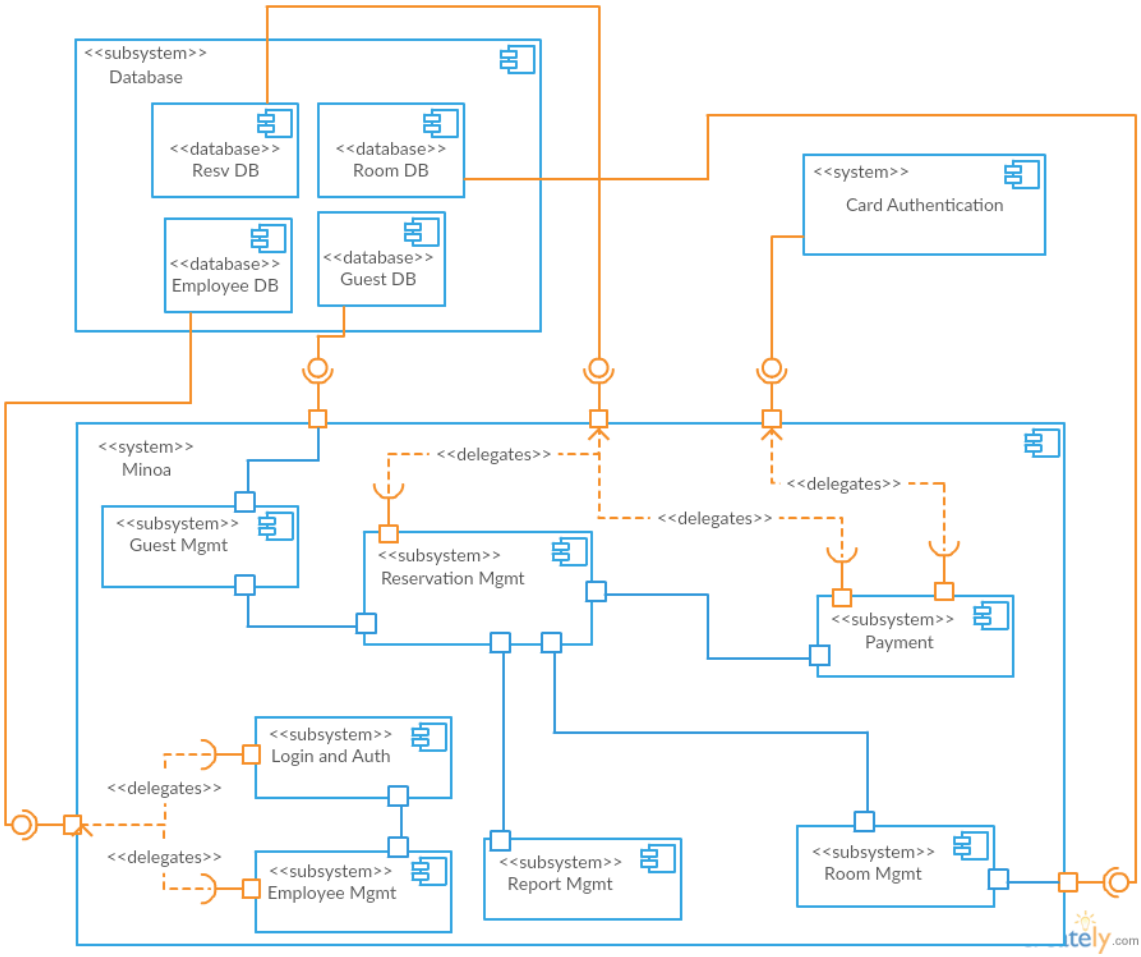
1. Software Architecture

The Software Architecture Diagram summarizes the architecture from the following views:

1. Logical View: A model of the System, Subsystems and Components that are most crucial parts of the Product.

2. Deployment View: View of the physical architecture of the system and the artifacts and where they run.

1. **Logical View:**

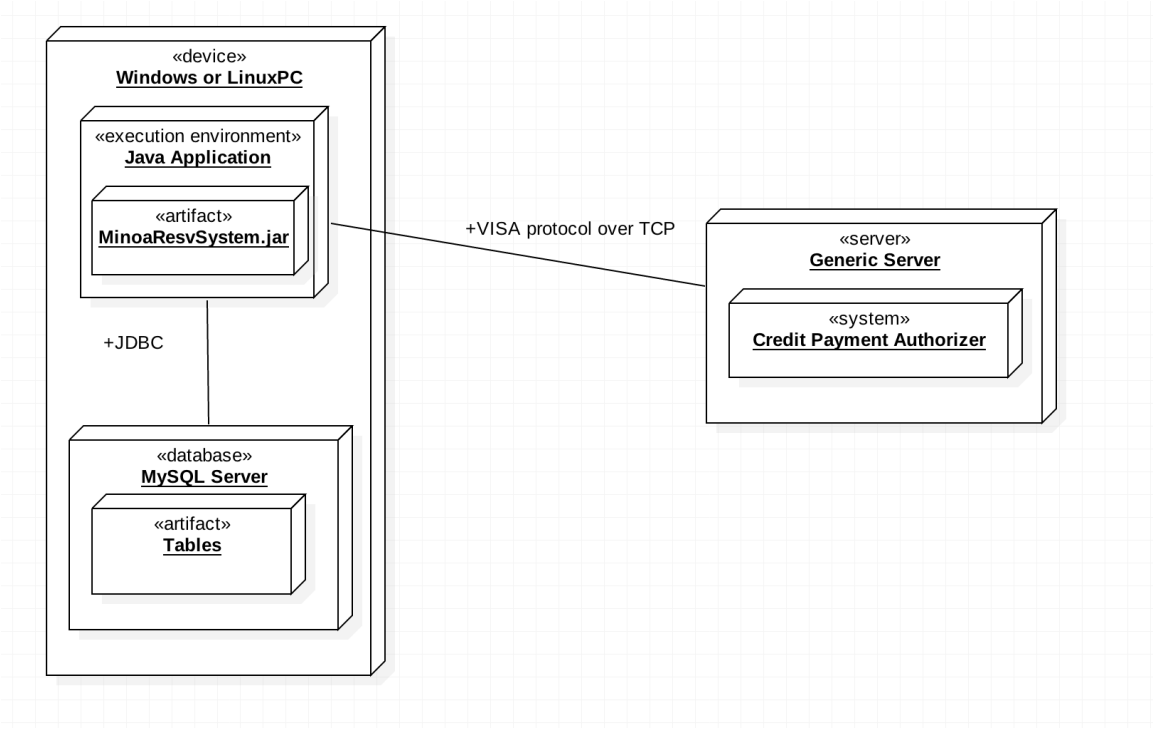


Discussion and Motivation:

The logical view takes into account the core Reservation System and it is consisted of various subsystems like the Reservation, Bill and Payment and GUI Subsystem. The Database Interface

component allows interaction with the external Database. The Card Authentication system is also required when processing Payments.

2. Deployment View:



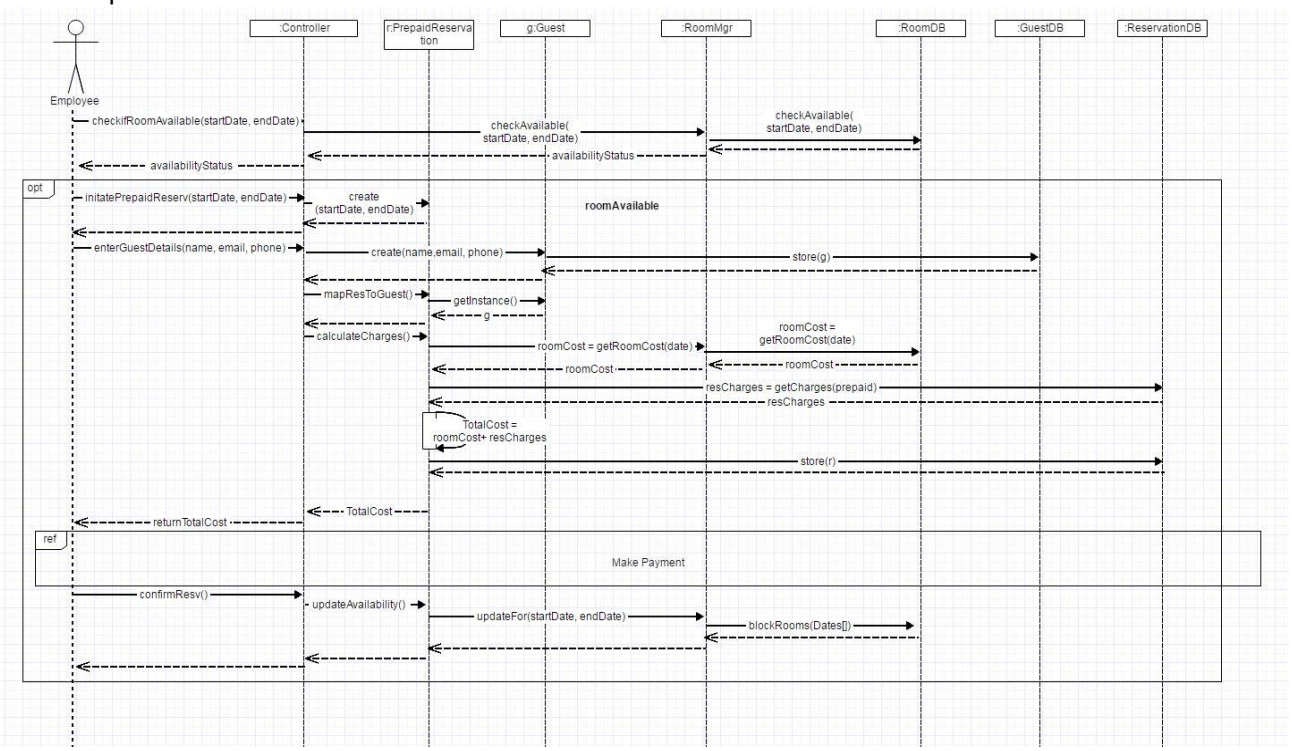
Discussion and Motivation:

The deployment diagram shows the distributed physical entities which are the Client Side PCs which will be accessed by Minoa Staff, the Database Server, Application Server (Glassfish) and the Credit Authentication System Server.

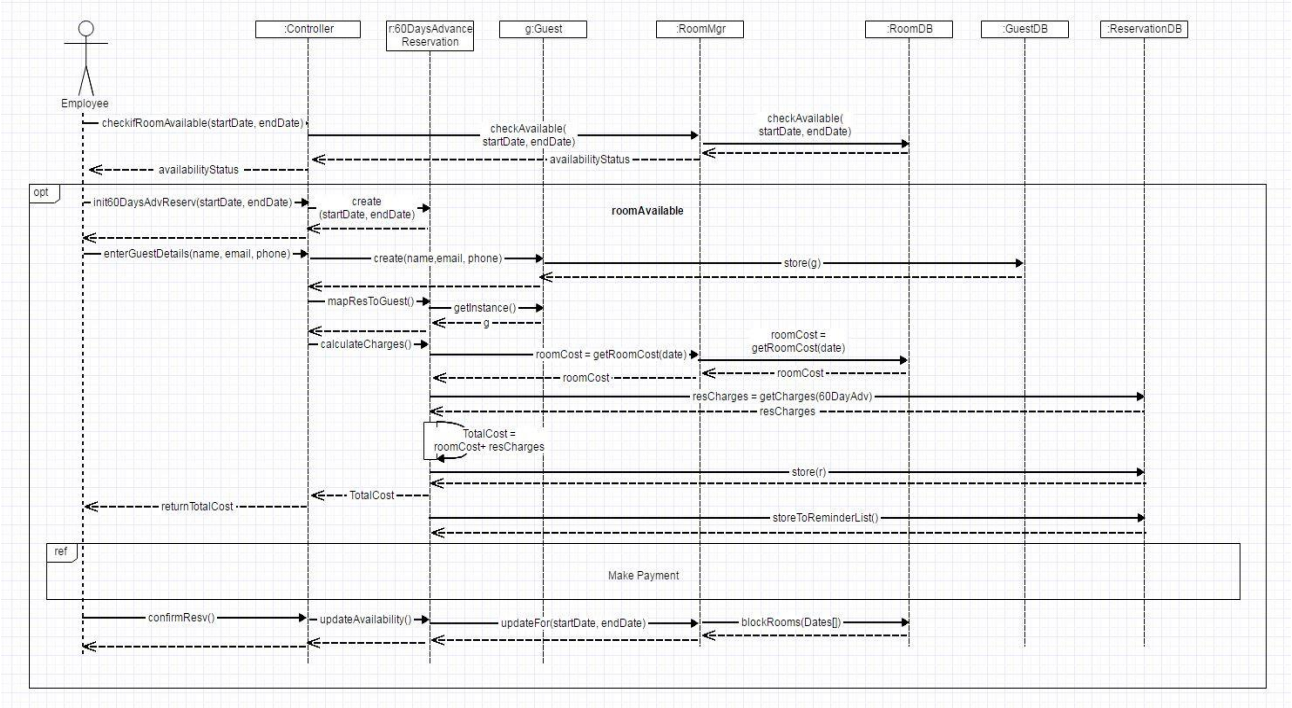
The connectivity between internal artefacts and their communication with external systems is shown.

USE CASE REALIZATIONS

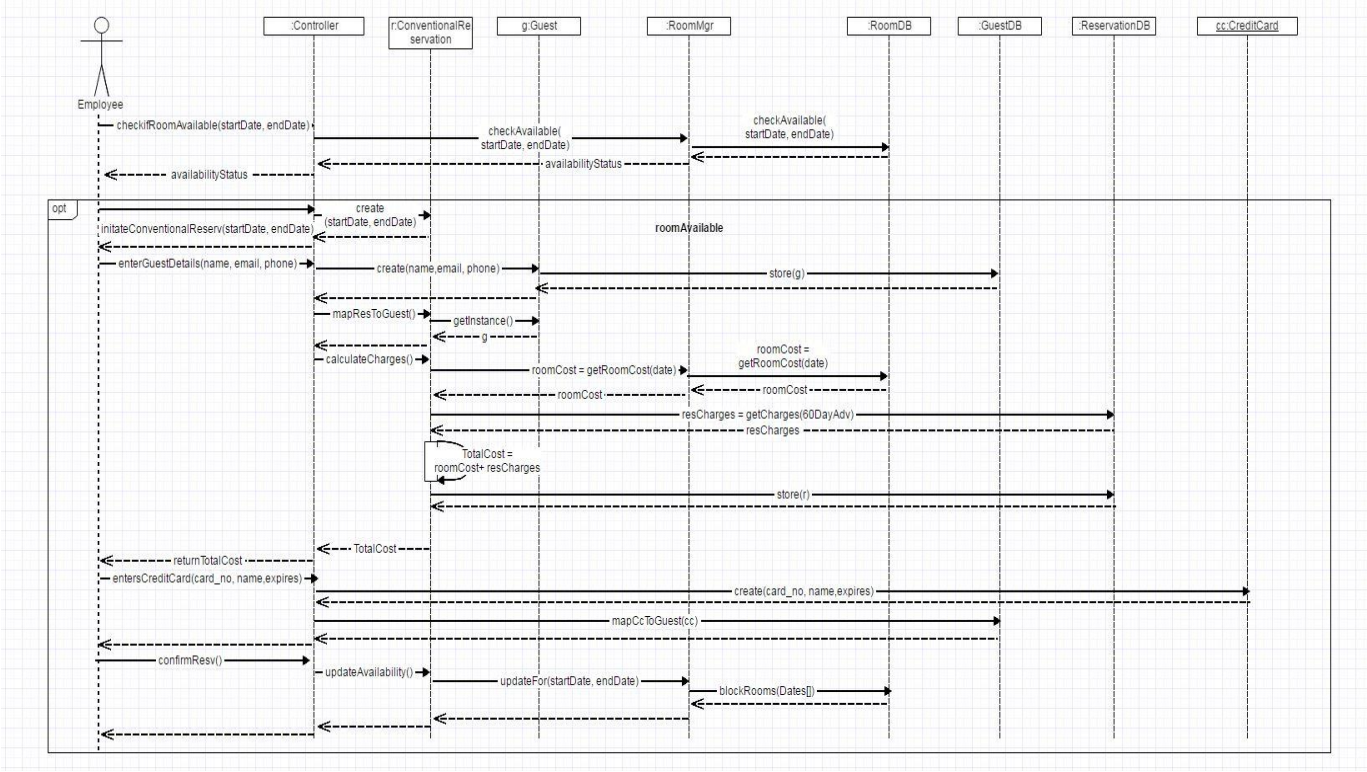
MakePrepaidReservation:



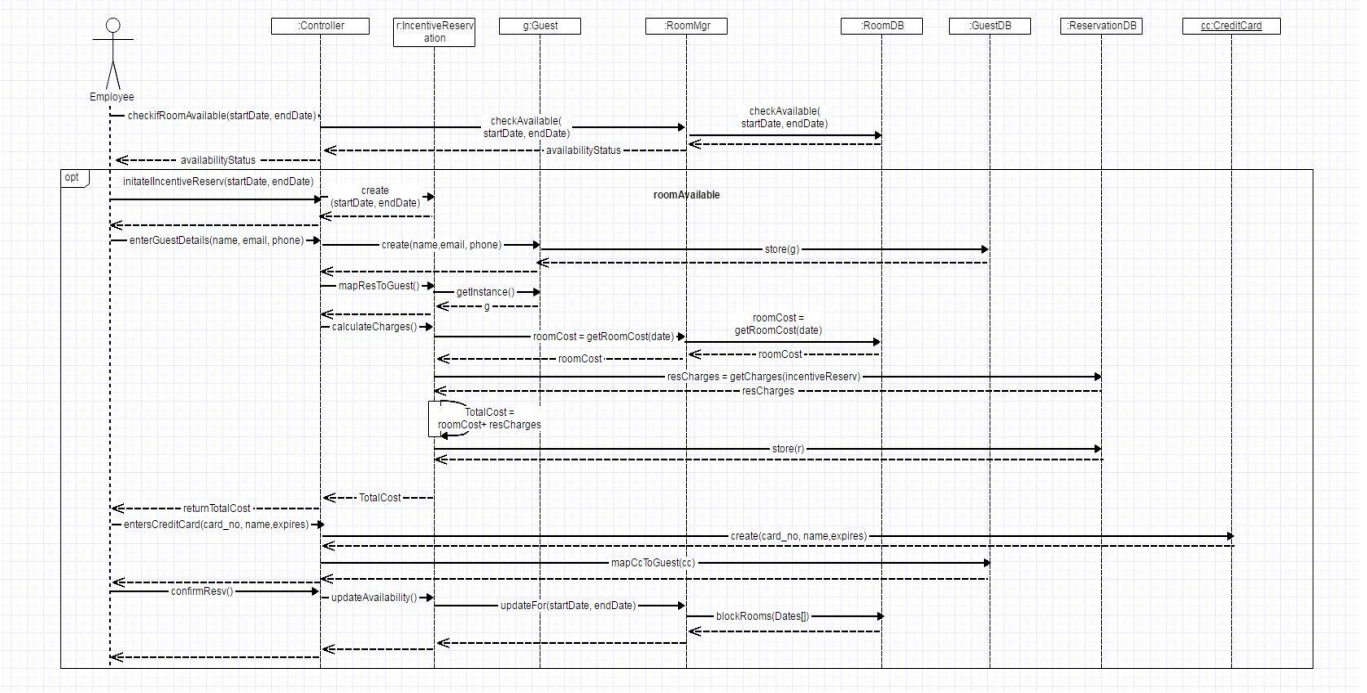
MakeSixtyDaysAdvanceReservation:



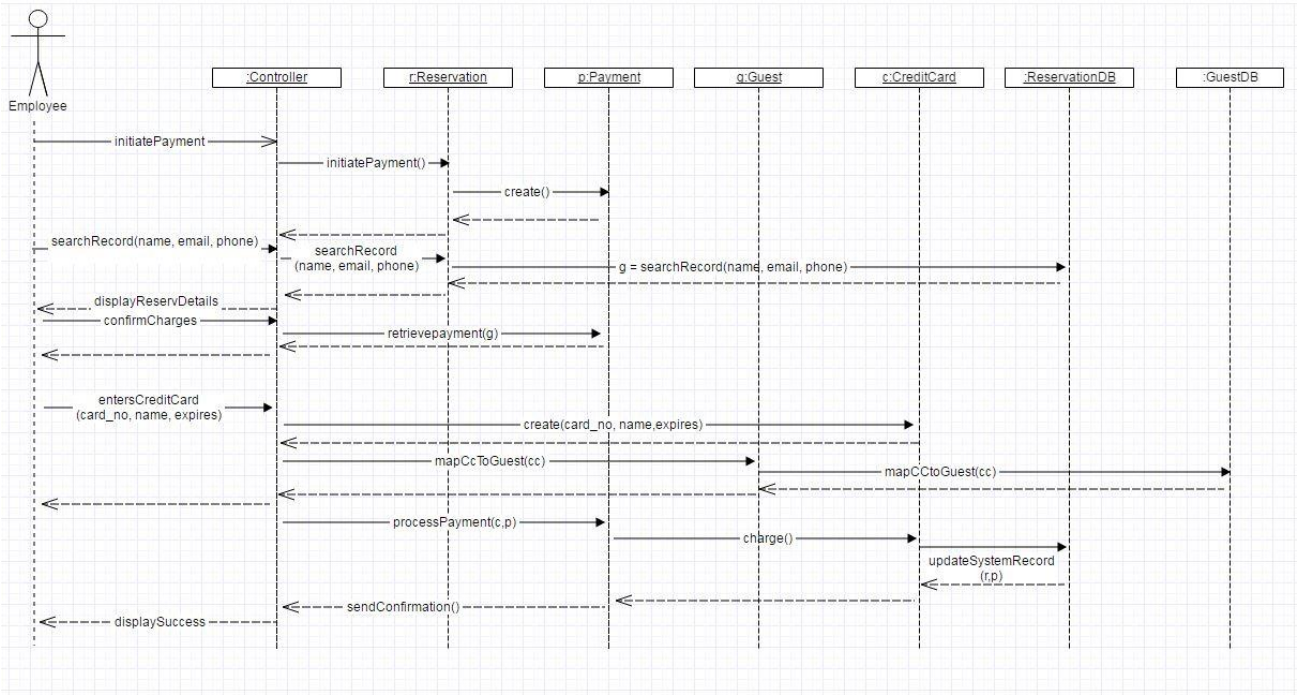
MakeConventionalReservation:



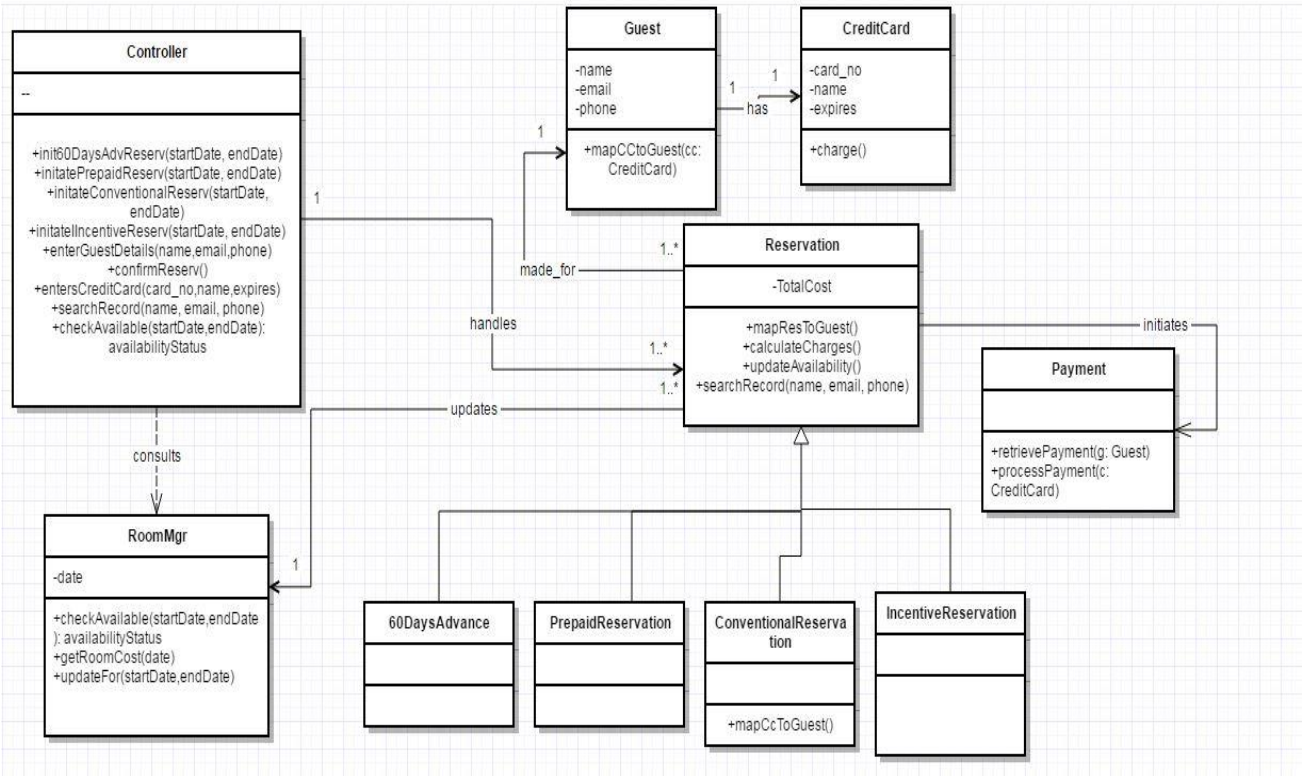
MakeIncentiveReservation:



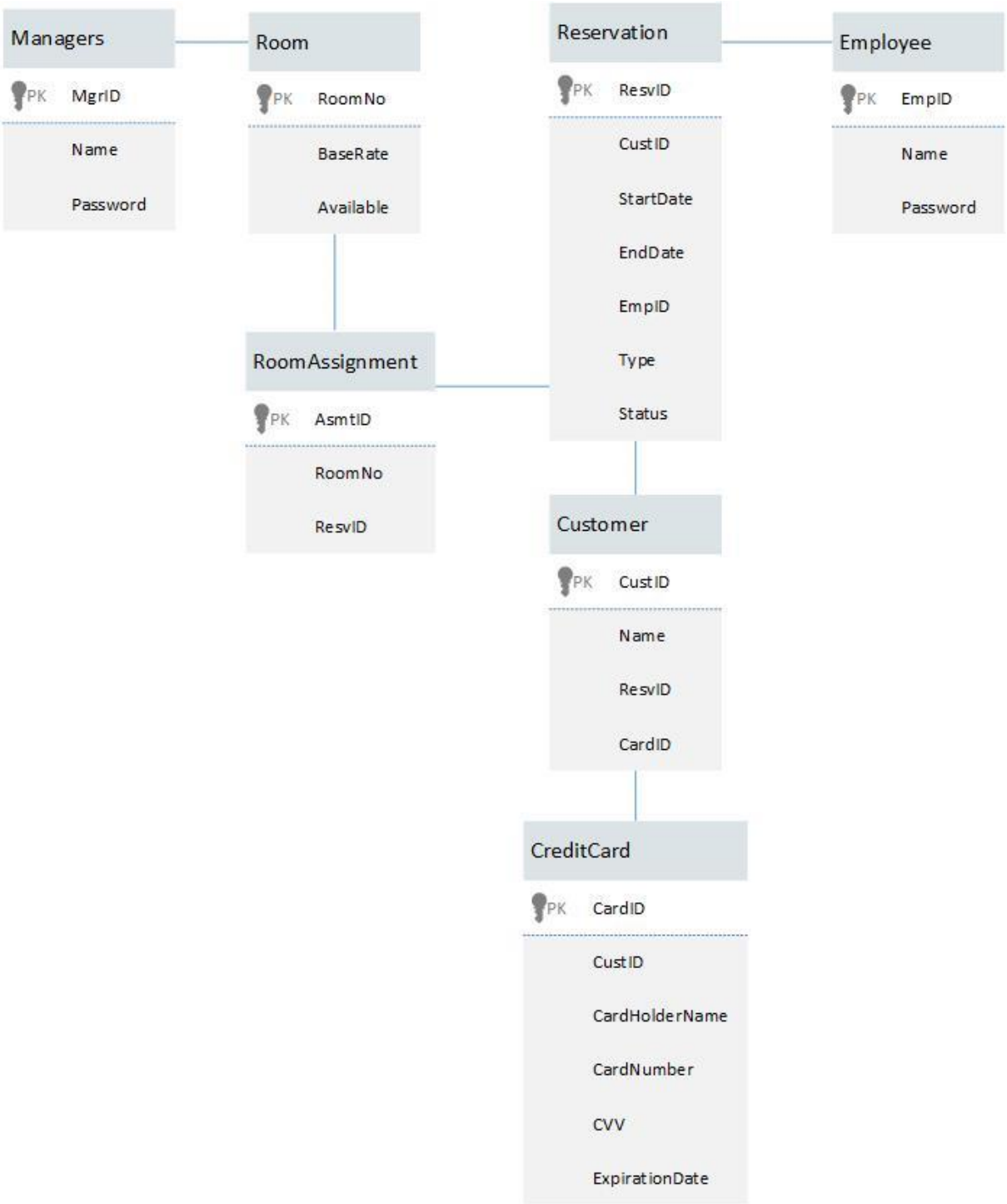
MakePayment



2. DESIGN CLASS DIAGRAMS



Database Design:



Part III: Implementation

Version 1.0

05/01/2017

Revision History

Version	Date	Description	Author
1.0 Implementation	05/01/2017	The implementation was done in JAVA based on the Design Class Diagram.	Preethi Sekar Umang Shah Nishigandha Narendra Rane Radhika Kalaiselvan Gajanan Golegaonkar

STATUS REPORT:

Task	Contributors
Updating the deliverables	Preethi,Umang
Domain Model	
Use Case Diagrams	
Use Case Description	
System Sequence Diagrams	
Operation Contracts	
Component Subsystem Diagram (Logical view)	
Deployment Diagram (Deployment View)	
Database Design	
Use Case Realization	
Design Class Diagram	
Implementation	Preethi, Nishigandha, Gajanan, Umang, Radhika
Formal Documentation	Preethi