Kevin Shibu Chacko - 40241154
Richard Badir - 40249566

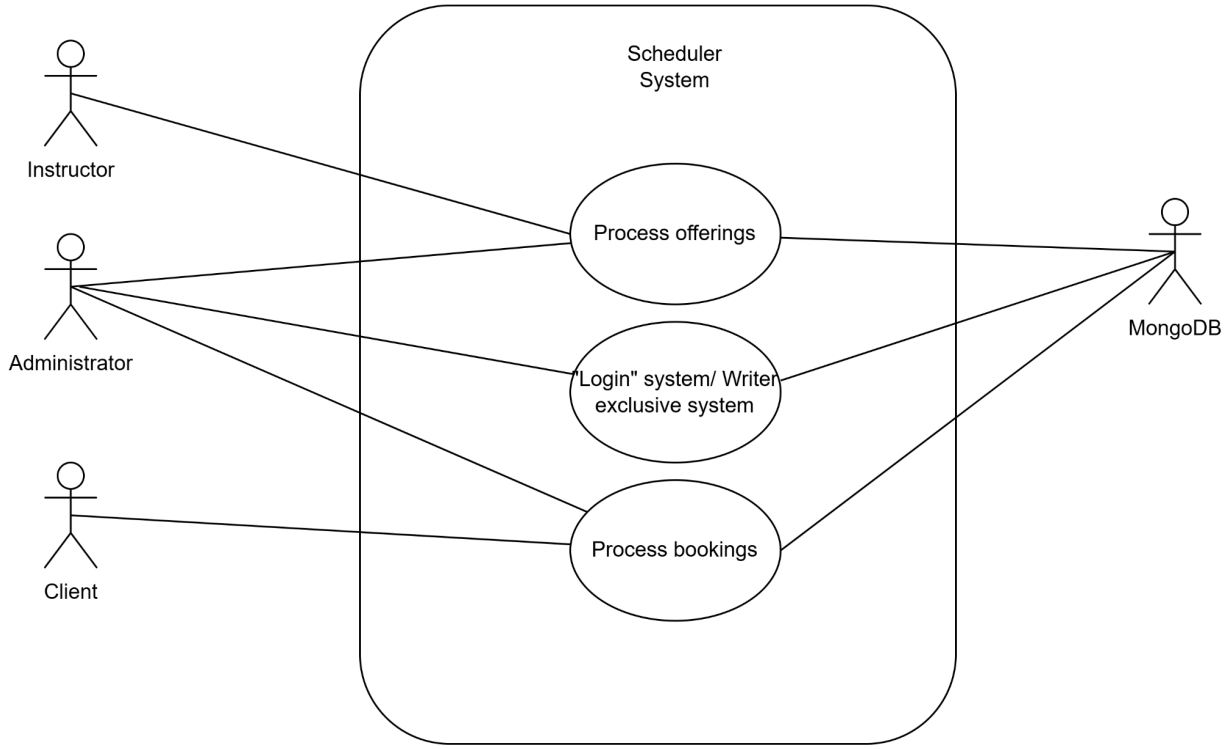
SOEN 342 Project Artifacts

SOEN 342
Section II

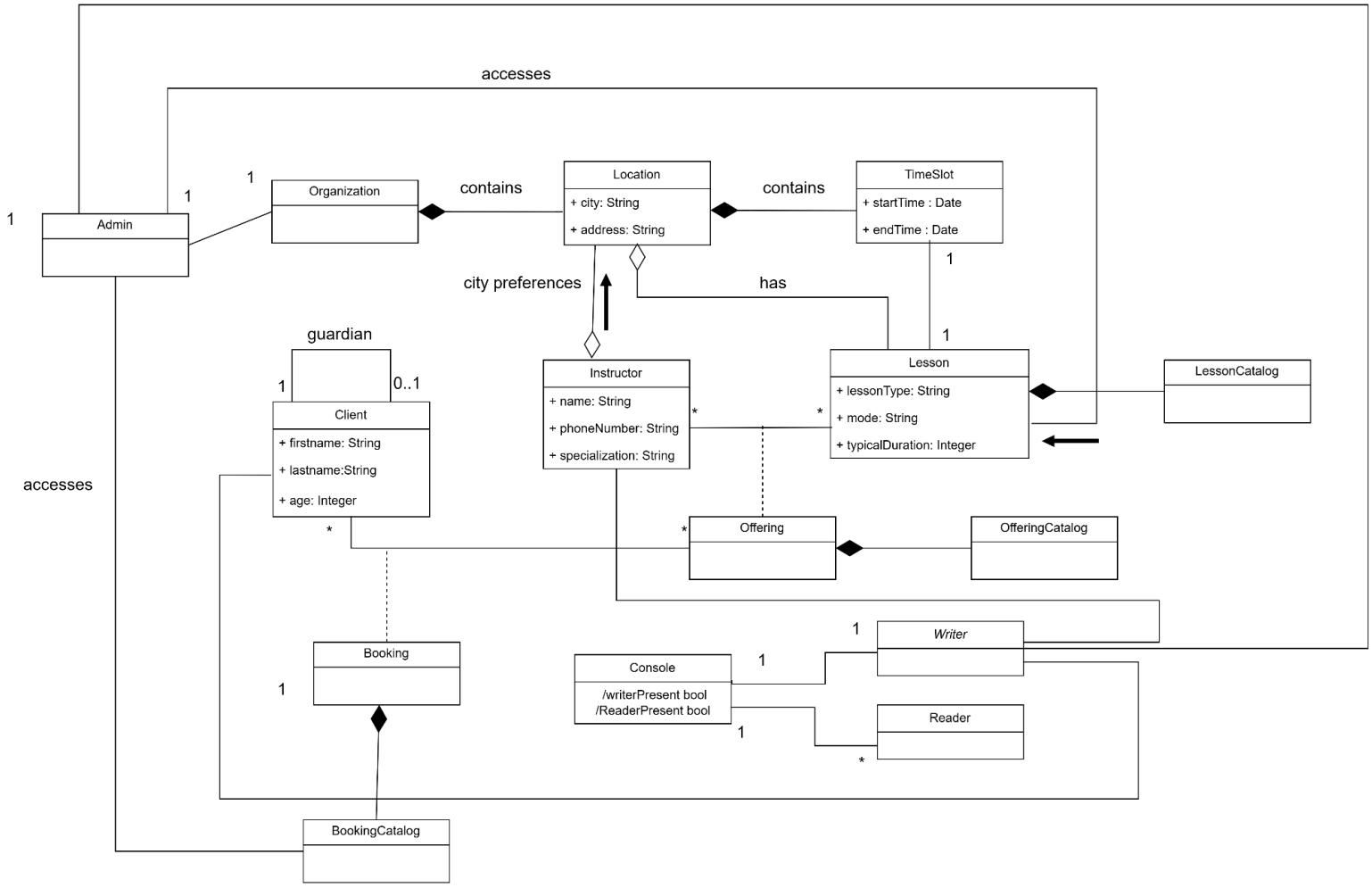
Constantino Constantinides
November 15th, 2024

Table of Contents

General Use Case:

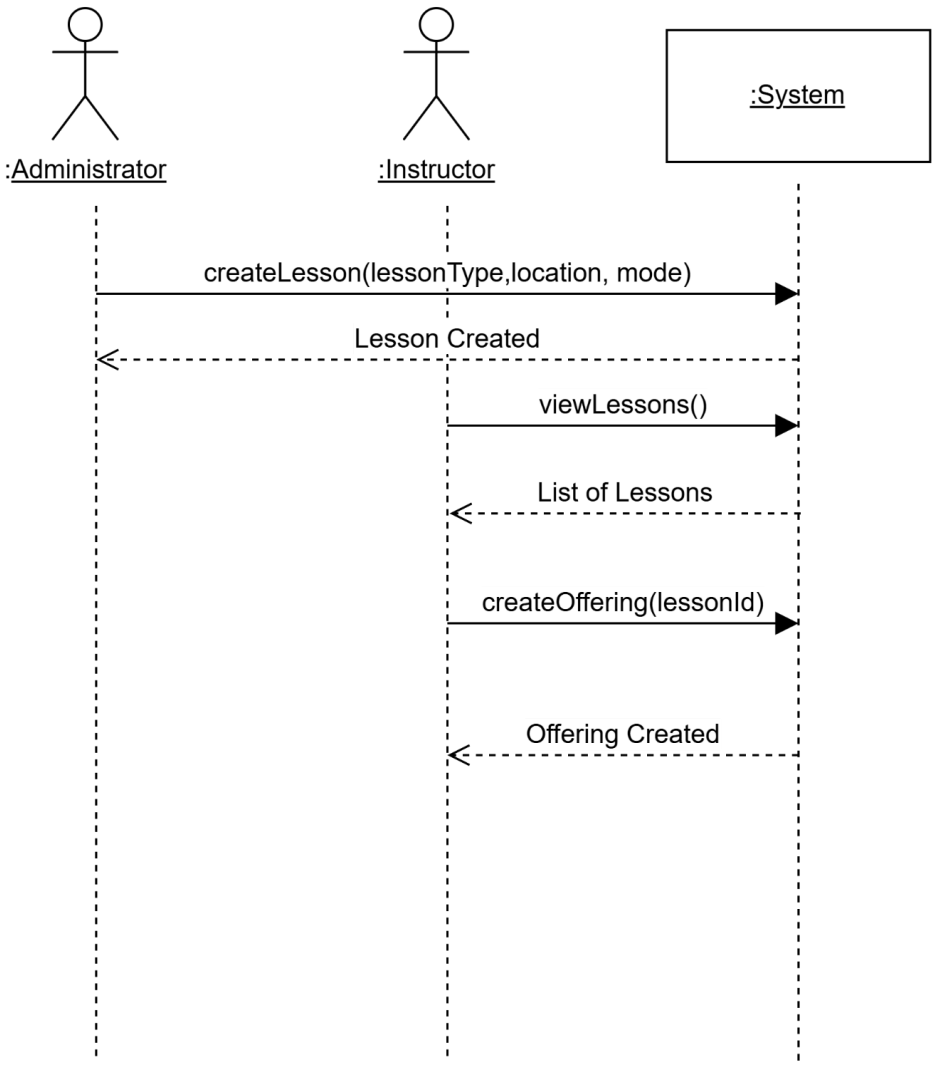


Domain Model:



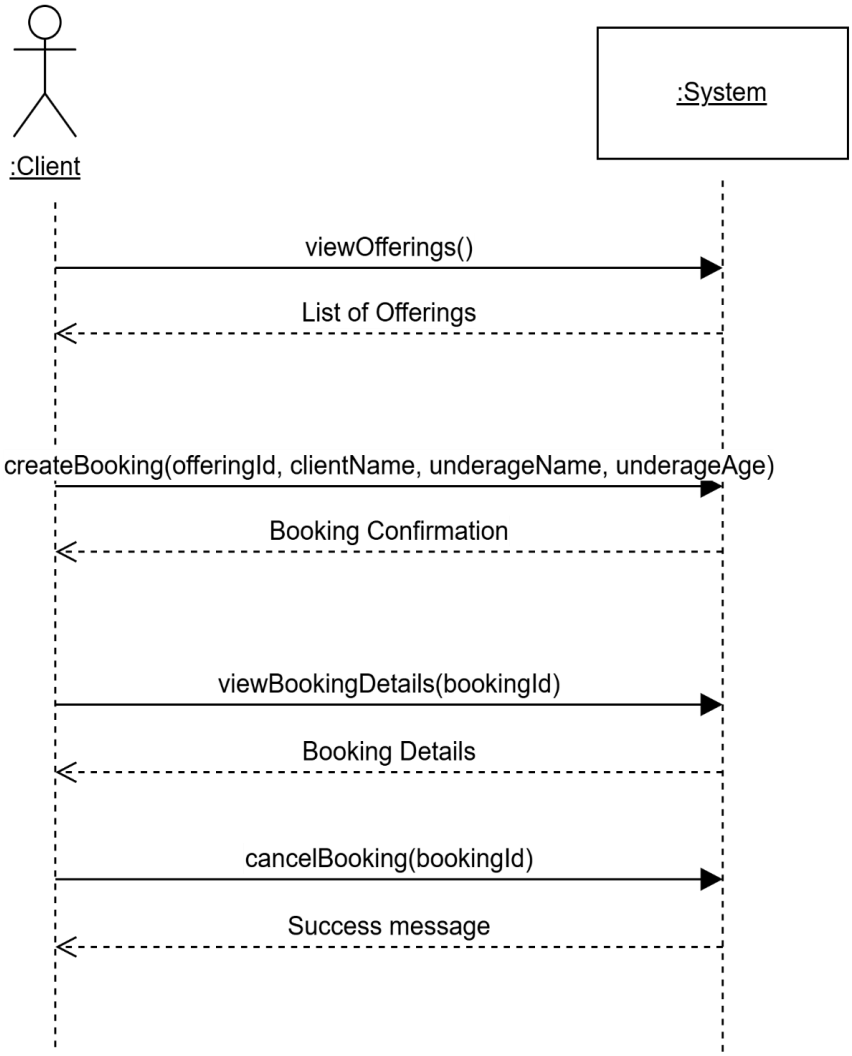
Success System Sequence Diagrams + System Operations:

Use Case 1: Process Offerings



System
createLesson(lessonType,location, mode)
viewLessons()
createOffering(lessonId)

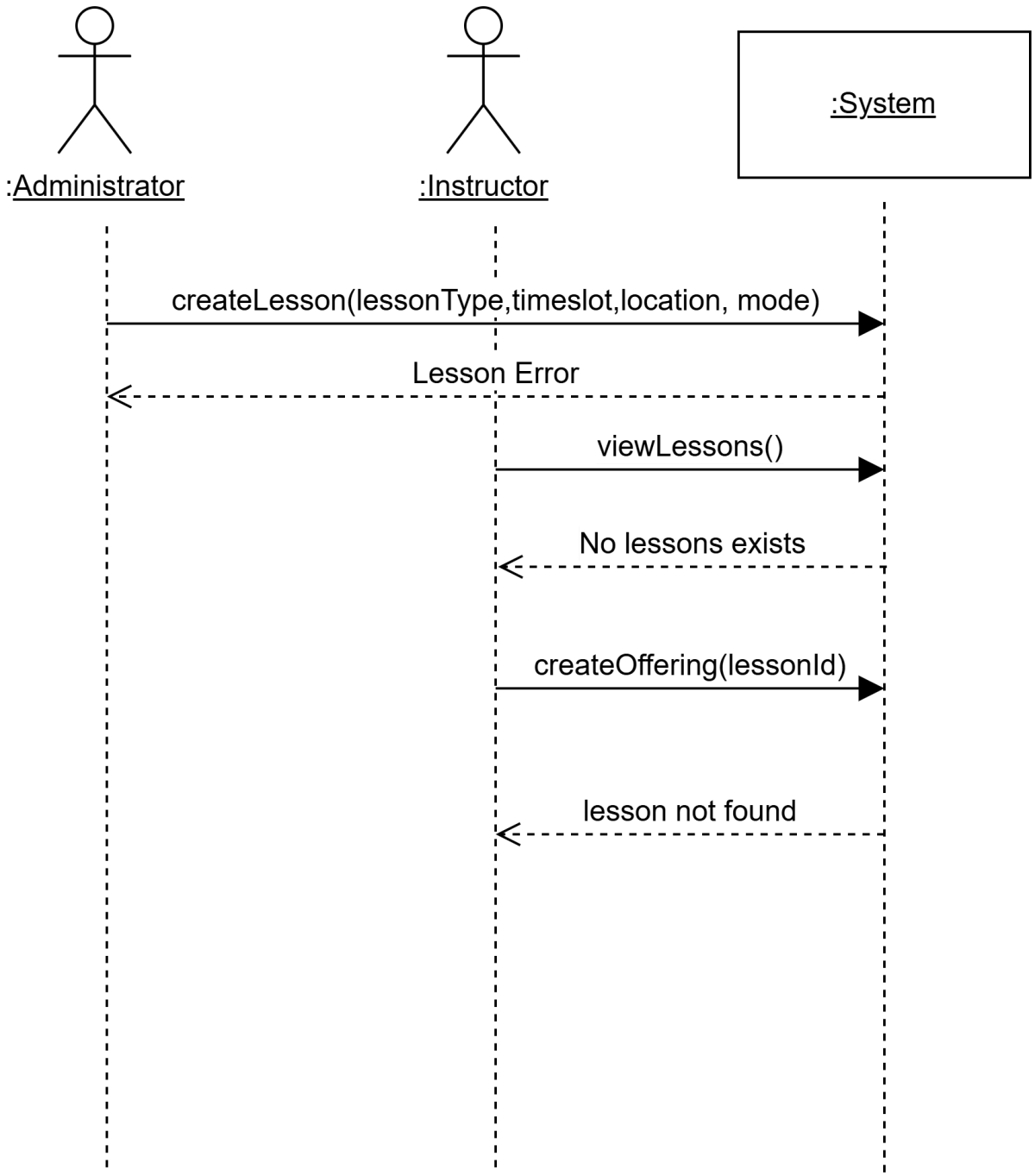
Use Case 2: Process Bookings



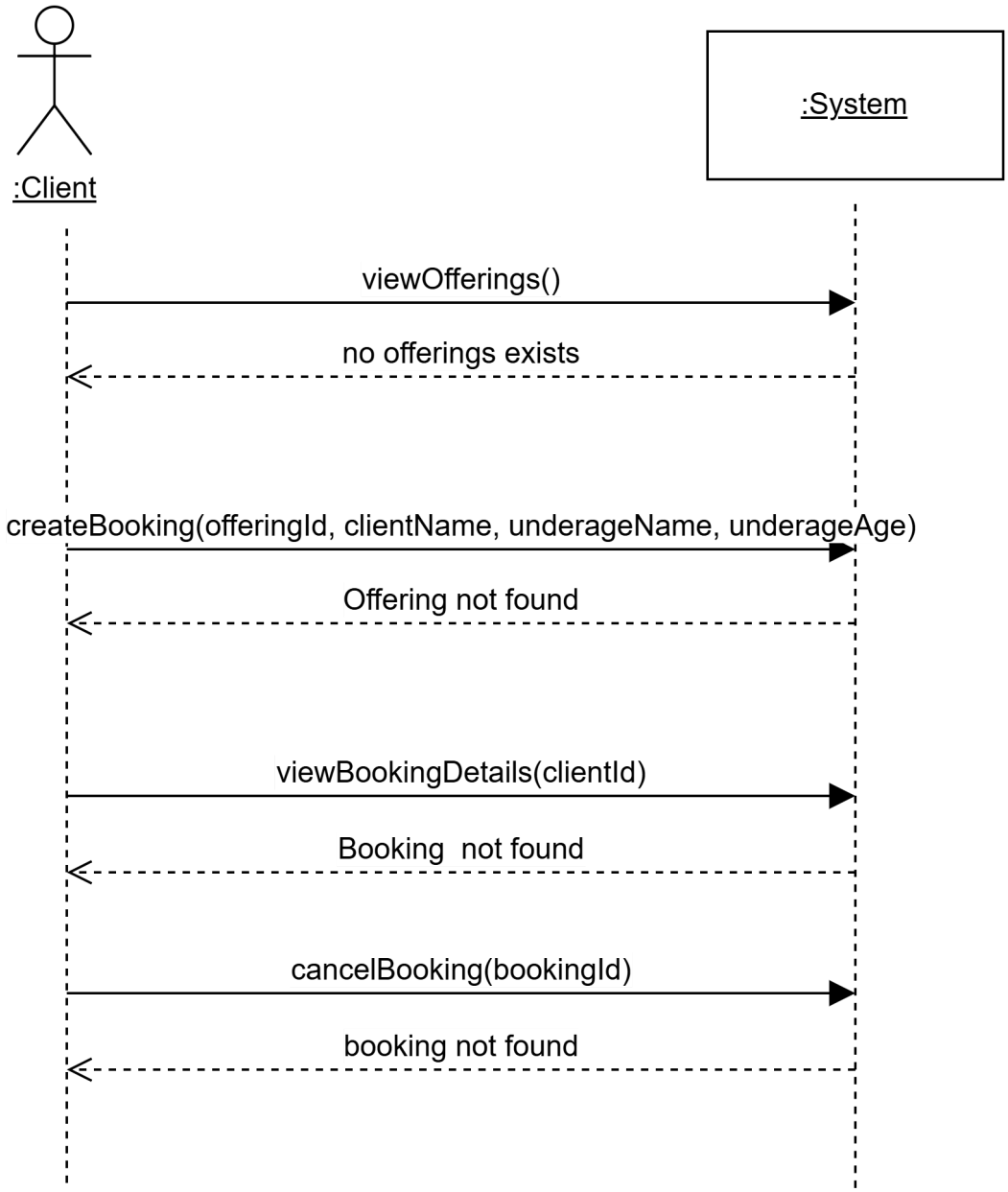
System
viewPublicOfferings()
createBooking(offeringId, clientName, underageName, underageAge)
viewBookingDetails(bookingId)
cancelBooking(bookingId)

Failure System Sequence Diagrams:

Use Case 1: Process Offerings



Use Case 2: Process Bookings



Contract Operations:

CO1

Contract: createLesson

Operation: createLesson(lessonType: String, mode: String, Location: Location, TimeSlot: TimeSlot)

Cross reference(s): Use Case Processing Offerings

Preconditions:

1. The operation is performed by administrator
2. The lessonType is valid and exists in the system
3. The mode is either 'private' or 'group'
4. The specified location exists in the system and has available time slots

Postconditions:

1. A new Lesson instance is created in the system (instance creation)
2. the new Lessons' attributes are set: lessonType, timeslot, location, mode (attribute modification).
3. A new lessonId is generated and assigned to new Lesson (attribute modification)
4. An association is formed between the new Lesson and specified Location (Association formed)

CO2

Contract: viewLessons

Operation: viewLessons()

Cross reference(s): Use Case Processing Offerings

Preconditions:

1. The instructor is logged in and authenticated

Postconditions:

none

CO3

Contract: createOffering

Operation: createOffering(lessonId: string)

Cross reference(s): Use Case Processing Offerings

Preconditions:

1. The Instructor is logged in and authenticated
2. The specified lessonId exists in the system
3. The status of lesson is available (not taken by another Instructor)
4. The Instructor's specialization match the lessons lesson type.

Postconditions:

1. A new Offering instance is created in the system (instance creation)
2. A new OfferingId is generated and assigned to new Offering (attribute modification)
3. The Lessons status is changed from 'available' to 'taken'. (attribute modification)
4. An Association is formed between Instructor and Lesson (Association formed)

CO4

Contract: viewOfferings

Operation: viewOfferings()

Cross reference(s): Use Case Processing Booking

Preconditions:
1. The Client is logged in and authenticated

Postconditions:
None

CO5

Contract: createBooking

Operation: createBooking(offeringId: String, clientName: String, underageName: String, underageAge: String)

Cross reference(s): Use Case Processing Booking

Preconditions:
1. The Client is logged in and authenticated
2. The specified offeringId exists in the system and is valid
3. The status of the Offering is available (not booked)
4. If underageName is provided, underageAge must also be provided
5. if underageAge is provided must be under 18.
6. If Offering mode is 'group', available spots must be greater than 0

Postconditions:
1. A new Booking instance is created in the system (instance creation)
2. The Booking's attributes are set: offeringId, clientName, underageAge (attribute modification)
3. A new BookingId is generated and assigned to new Booking (attribute modification)
4. The Booking's attribute status is set to 'active'
5. If Offering mode is 'private', the Offering's status is updated to 'booked' (attribute modification
6. If Offering mode is 'group'

- The Offering's available spots are decreased by 1 (attribute modification)
- If available spots reach 0, the Offering's status is updated to 'booked' (attribute modification)

7. An association is formed between the Booking and Client (Association formed)
8.An association is formed between the Booking and the specified Offering(association formed)

CO6

Contract: viewBookingDetails

Operation: viewBookingDetails(clientId:String)

Cross reference(s): Use Case Processing Booking

Preconditions:
1. The Client is logged in and authenticated

Postconditions:
None

CO7

Contract: CancelBooking

Operation: cancelBooking(bookingId: String)

Cross reference(s): Use Case Processing Booking

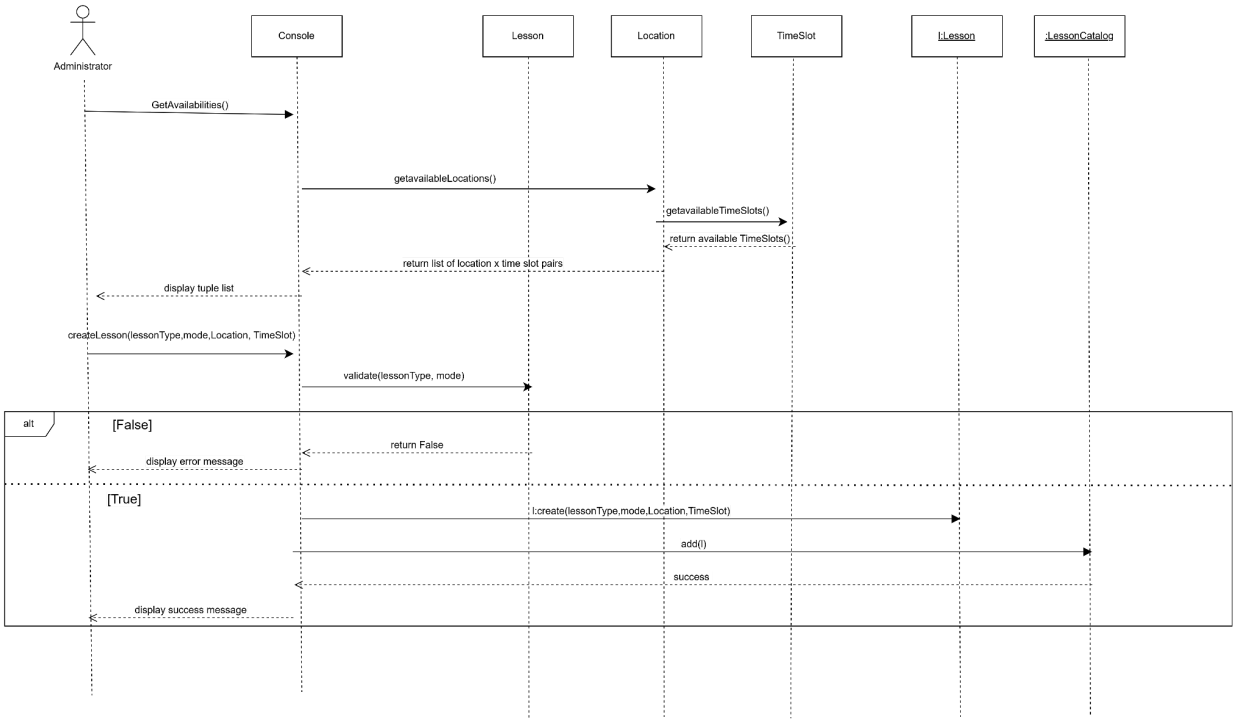
Preconditions:
1. The Client is logged in and authenticated
2. The specified bookingId exists in the system
3. The Booking belongs to the requesting Client
4. The Booking status is 'active'

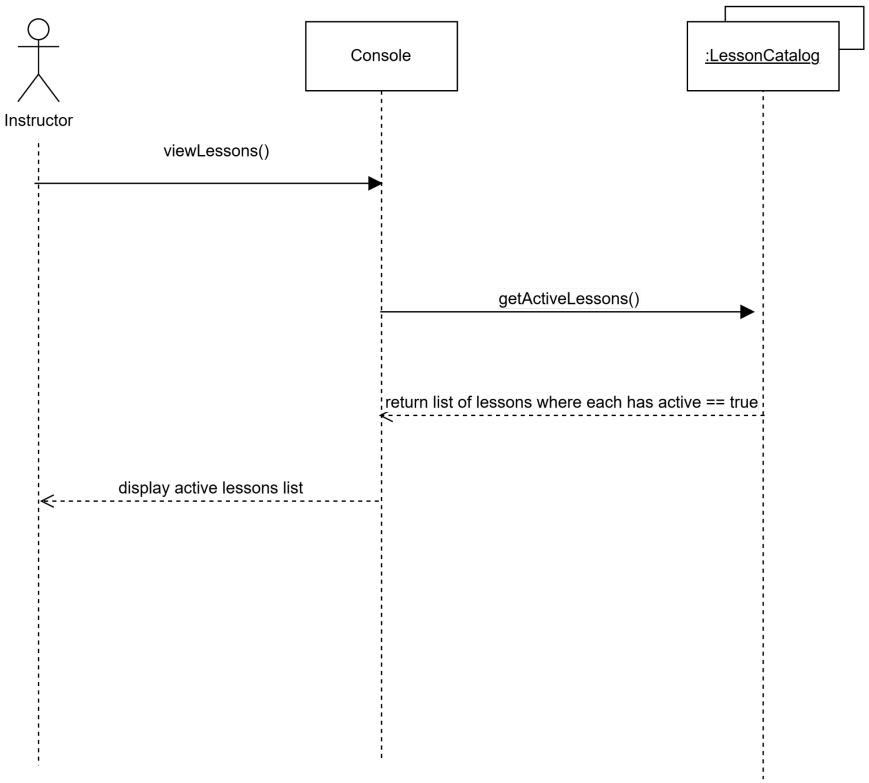
Postconditions:
1. The Booking's status is updated to 'cancelled' (attribute modification)
2. if associated Offering mode is private, the Offering's status is updated to 'available' (attribute modification)
3. If associated Offering mode is 'group'

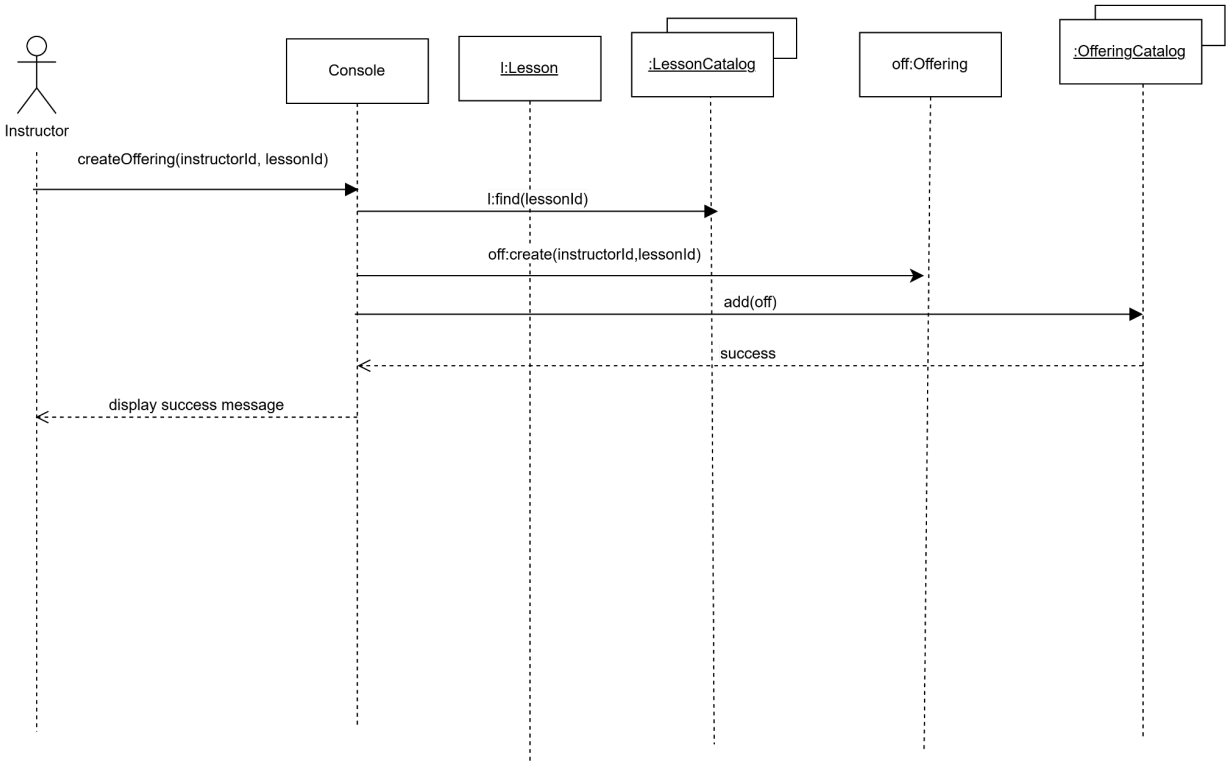
- The Offering's available spots in increased by 1 (attribute modification)
- The Offering's status is updated to 'available; if it was 'booked' (attribute modification)

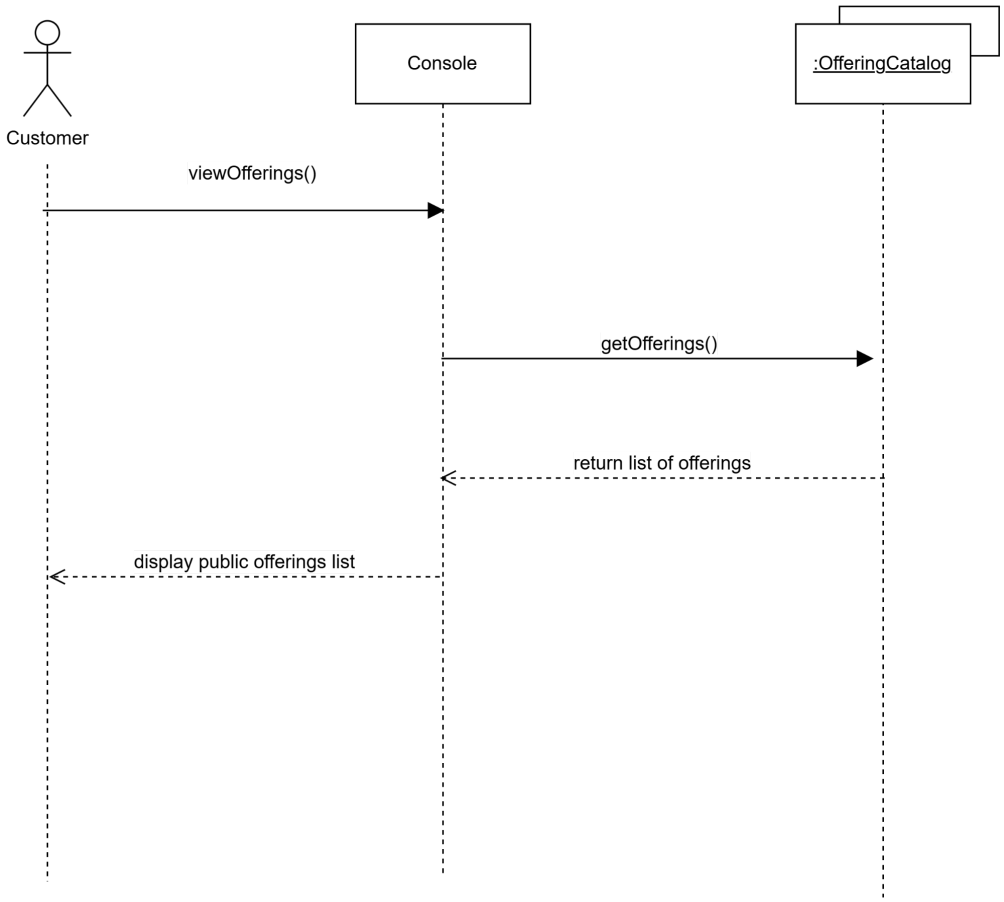
4. Association between Booking and Client is removed (Association removed)
5. Association between Booking and Offering is removed (Association removed)
6. Booking instance is deleted (instance deletion)

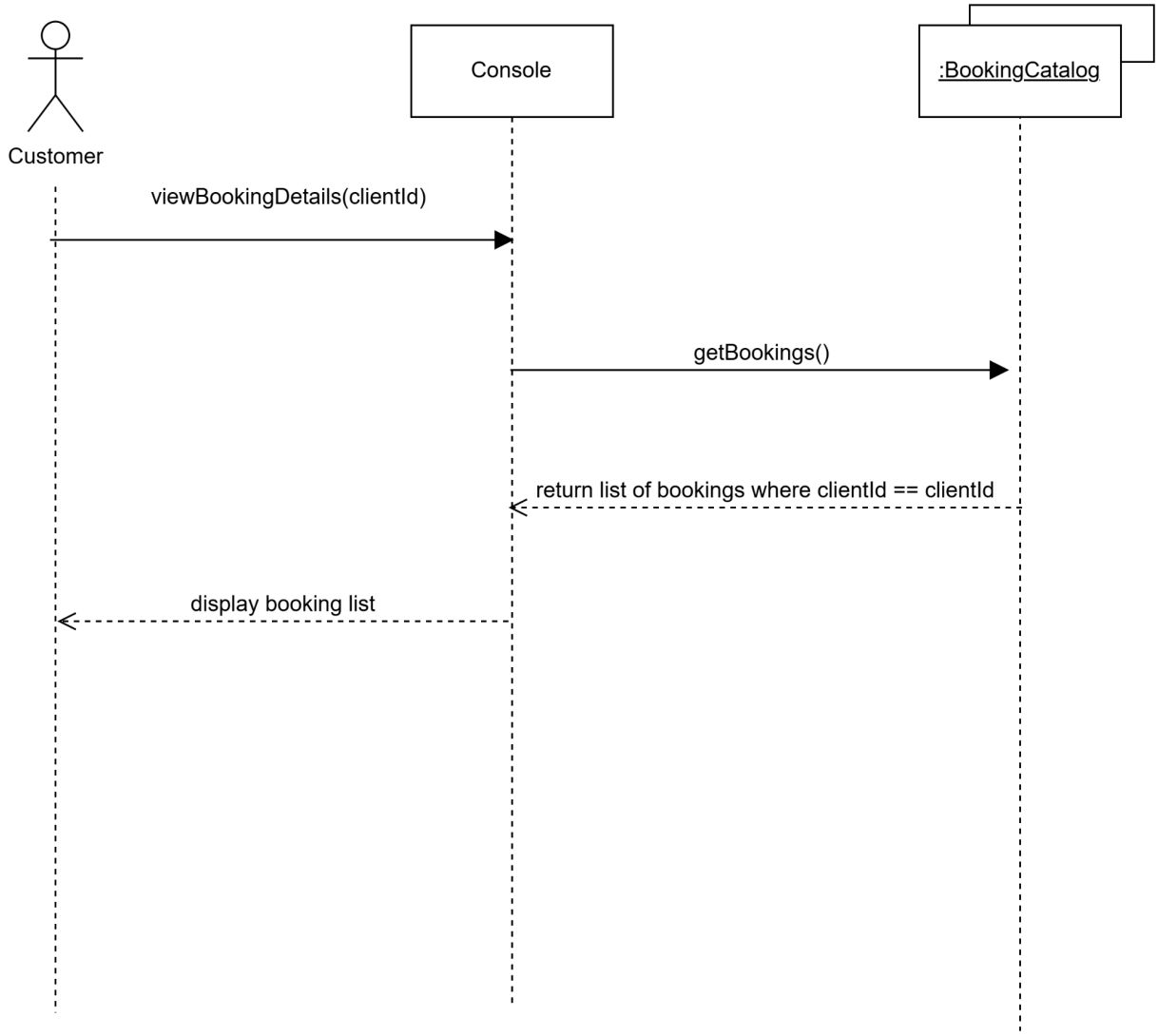
Interaction Diagrams:

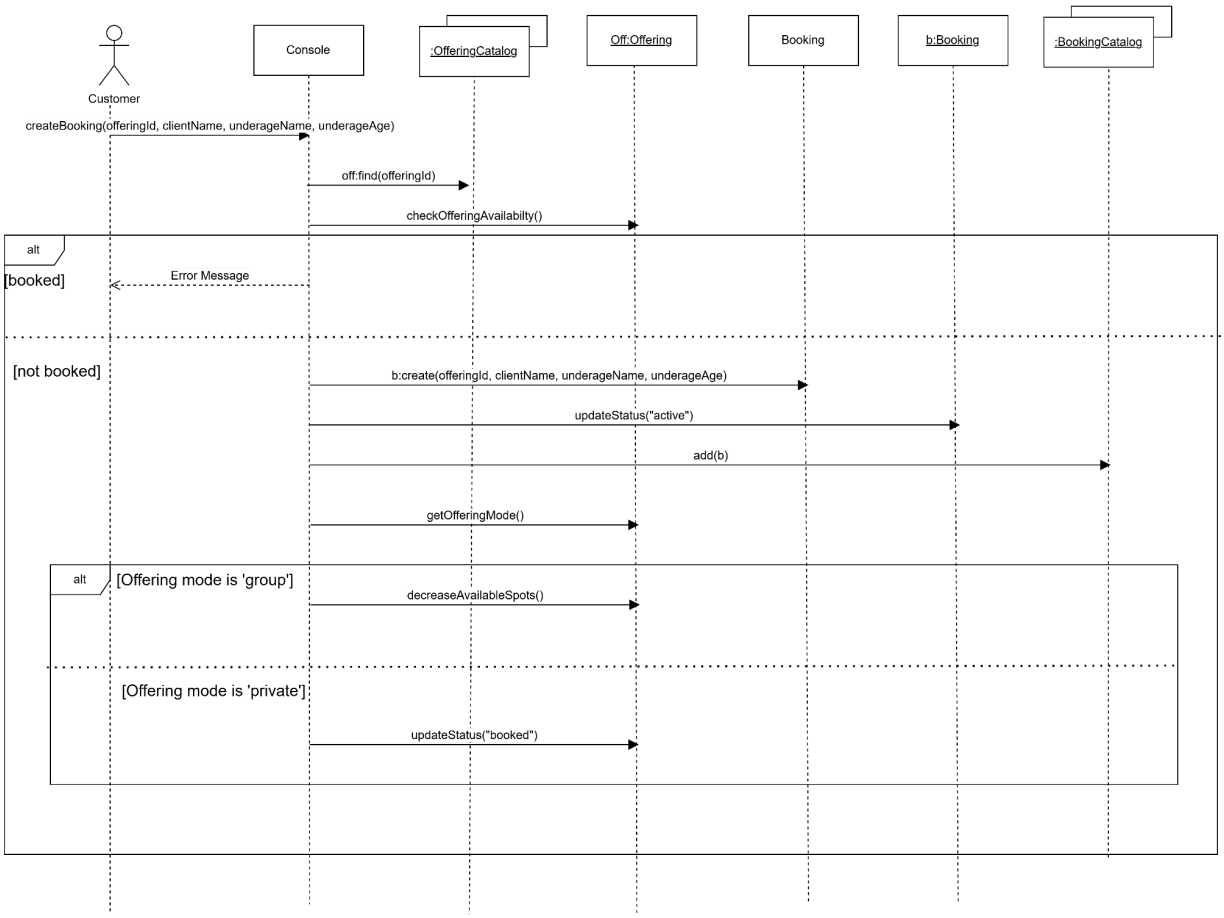


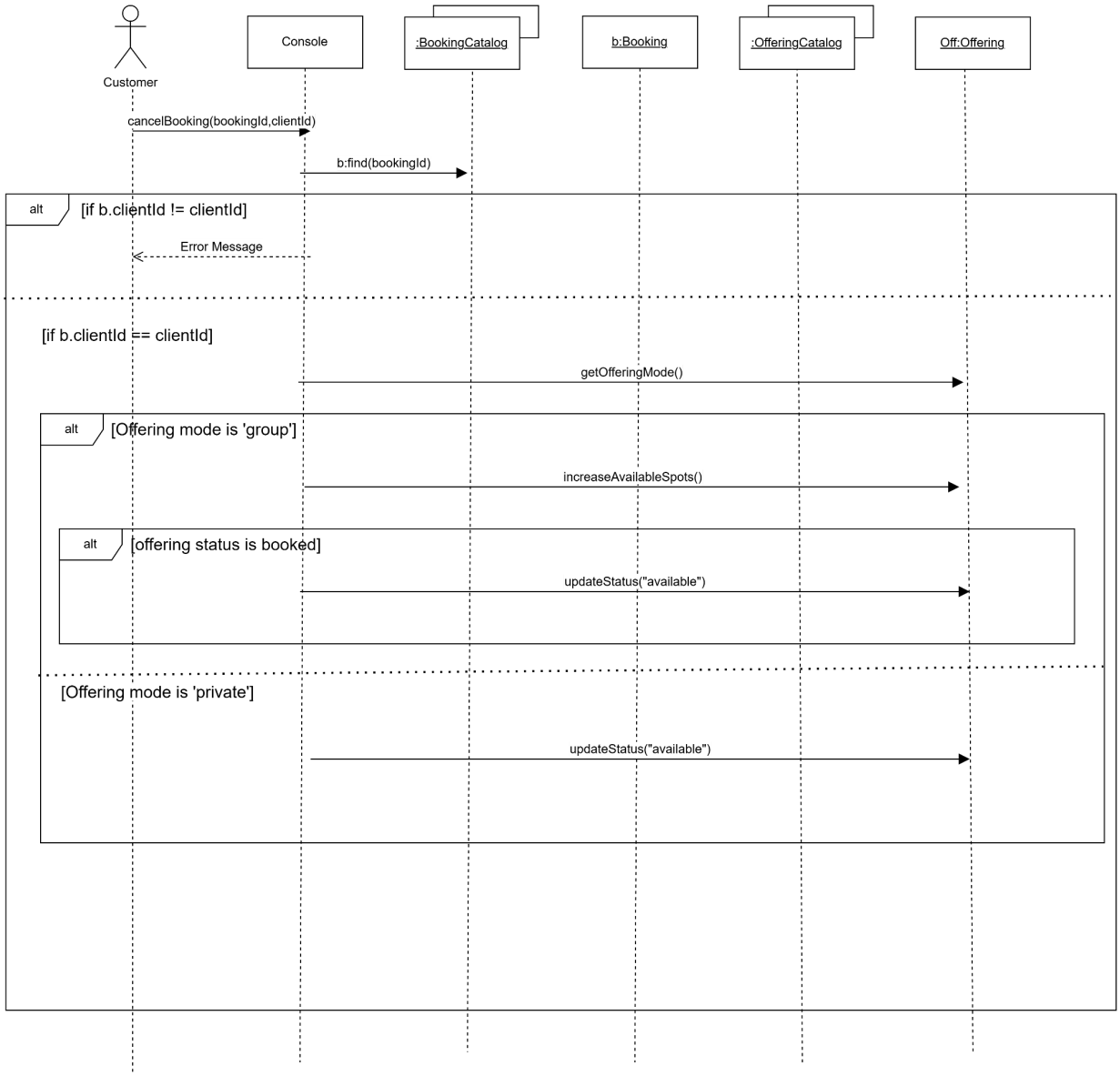

Instructor
Console
:LessonCatalog
viewLessons()
getActiveLessons()
return list of lessons where each has active == true
display active lessons list


Instructor
Console
l:Lesson
:LessonCatalog
off:Offering
:OfferingCatalog
createOffering(instructorId, lessonId)
l:find(lessonId)
off:create(instructorId,lessonId)
add(off)
success
display success message

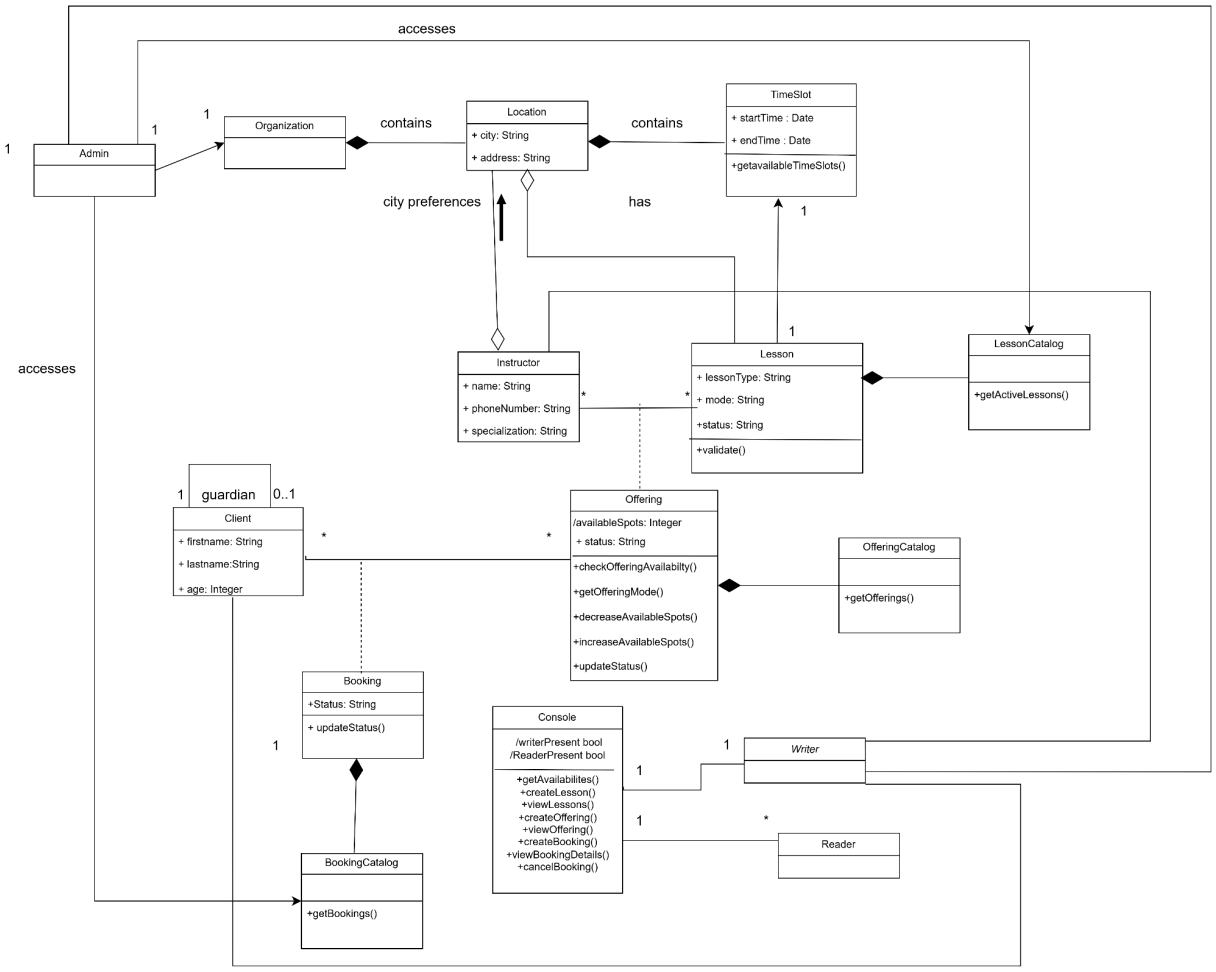

Customer
Console
:OfferingCatalog
viewOfferings()
getOfferings()
return list of offerings
display public offerings list


Customer
Console
:BookingCatalog
viewBookingDetails(clientId)
getBookings()
return list of bookings where clientId == clientId
display booking list

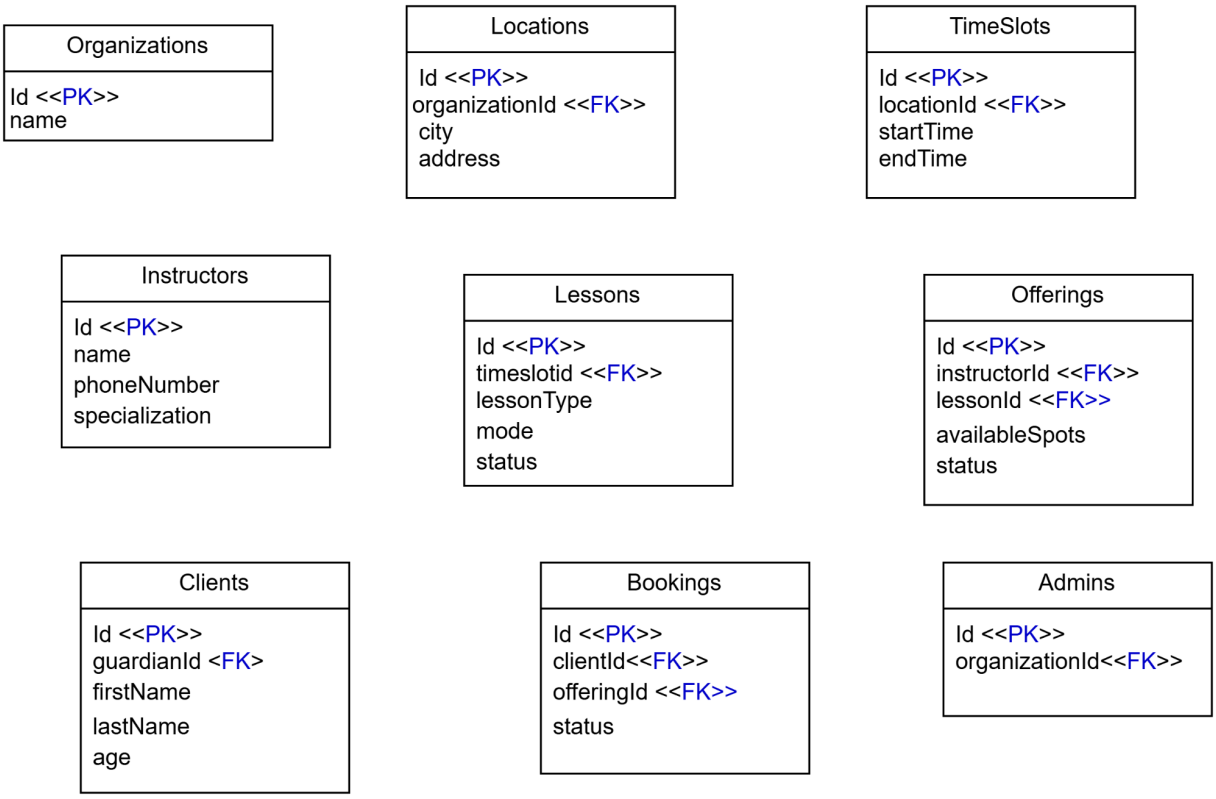

Customer
Console
:OfferingCatalog
Off:Offering
Booking
b:Booking
:BookingCatalog
createBooking(offeringId, clientName, underageName, underageAge)
off:find(offeringId)
checkOfferingAvailabilty()
alt
[booked]
Error Message
[not booked]
b:create(offeringId, clientName, underageName, underageAge)
updateStatus("active")
add(b)
getOfferingMode()
alt
[Offering mode is 'group']
decreaseAvailableSpots()
[Offering mode is 'private']
updateStatus("booked")


Customer
Console
:BookingCatalog
b:Booking
:OfferingCatalog
Off:Offering
cancelBooking(bookingId,clientId)
b:find(bookingId)
alt
[if b.clientId != clientId]
Error Message
[if b.clientId == clientId]
getOfferingMode()
alt
[Offering mode is 'group']
increaseAvailableSpots()
alt
[offering status is booked]
updateStatus("available")
[Offering mode is 'private']
updateStatus("available")

Class Diagram:



Relational Models:



OCL expressions:

1.

context OfferingCatalog
inv: self.getOfferings()→forAll(o1, o2 | o1 <> o2 implies
not (o1.lesson.timeSlot.startTime = o2.lesson.timeSlot.startTime and
o1.lesson.timeSlot.endTime = o2.lesson.timeSlot.endTime and
o1.lesson.location = o2.lesson.location))

2.

context Client
inv: self.age < 18 implies **self**.guardian <> null

3.

context Offering
inv: self.instructor.cityPreferences->includes(**self.**lesson.location.city)

4.

context Client
inv: self.bookings→forAll(b1, b2 | b1 <> b2 implies not (b1.offering.lesson.timeSlot.startTime = b2.offering.lesson.timeSlot.startTime and b1.offering.lesson.timeSlot.endTime = b2.offering.lesson.timeSlot.endTime))