Car Hire System

Car Hire System report containing system design planning using various UML diagrams including sequence, class, state and use case diagrams to illustrate the behavior of an online Car Hire System.

Robert Jacob 20089386

Contents

[Introduction 2](#_Toc71230909)

[Use Case 1: Search Car 3](#_Toc71230910)

[Main Success Scenario 4](#_Toc71230911)

[Sequence Diagram UC1 MSS 4](#_Toc71230912)

[Use Case 2: Book Car 4](#_Toc71230913)

[Main Success Scenario 5](#_Toc71230914)

[Sequence Diagram UC2 MSS 6](#_Toc71230915)

[6](#_Toc71230916)

[Alternative Success Scenario 7](#_Toc71230917)

[Sequence Diagram UC2 AS 7](#_Toc71230918)

[Failure Scenarios 8](#_Toc71230919)

[Sequence Diagram UC2 FS 8](#_Toc71230920)

[Use Case 3: Guest Booking 9](#_Toc71230921)

[Main Success Scenario 9](#_Toc71230922)

[Sequence Diagram UC3 MSS 10](#_Toc71230923)

[Failure Scenarios 10](#_Toc71230924)

[Sequence Diagram UC3 FS 11](#_Toc71230925)

[Use Case 4: Login 12](#_Toc71230926)

[Main Success Scenario 12](#_Toc71230927)

[Sequence Diagram UC4 MSS 13](#_Toc71230928)

[Alternative Success Scenario 13](#_Toc71230929)

[Sequence Diagram UC4 AS 14](#_Toc71230930)

[Failure Scenarios 15](#_Toc71230931)

[Sequence Diagram UC4 FS 15](#_Toc71230932)

[Use Case 5: Edit Booking 16](#_Toc71230933)

[Main Success Scenario 16](#_Toc71230934)

[Sequence Diagram UC5 MSS 17](#_Toc71230935)

[Failure Scenarios 18](#_Toc71230936)

[Sequence Diagram UC5 FS 18](#_Toc71230937)

[Use Case Diagram of Car Hire System 19](#_Toc71230938)

[Class Diagram of Car Hire System 20](#_Toc71230939)

[Hire System Test code 21](#_Toc71230940)

## Introduction

This online car hire system will service a car hire business for a medium sized local market. This car hiring system will facilitate the user’s ability to search the system for a suited car, book a chosen car and edit a previous car booking. A login facility is available for users that wish to create an account, accounts are created with the use of an email address accompanied by a password which will allow for previous bookings to be saved, as well as personal details including payment details.

The process of booking a car for hire will require a user to search for a car and then select a suitable car. The user will then proceed to book the car which will prompt an option to login or to continue as a guest, using his/her payment details saved to their account or in the case of a guest will require entering payment details at the time of booking. Bookings must be made using credit or debit cards, also including third party online payment means such as PayPal. A user may also avail of the ability to edit a booking which can be simply accessed through the users account after login or by entering the booking ID in the case of a guest user, which would have been sent automatically to the email of the user entered at the booking payment.

The brief description of the general scenarios above will be discussed in greater detail below with a main success scenario (MSS), an alternative success scenario and a failure scenario. Each use case will be given a section and a general description of each scenario will be illustrated by a UML sequence diagram that will provide a more clear and easily translatable template for coding the system. The diagram should illustrate a sequence of steps describing an interaction between a user and the system.

A class diagram is also present below which describes the types of objects in the system and the various kinds of static relationships that exist among them. This class diagram will also display attributes and operations of a class and the other constraints that apply to the way objects are connected. Overall the class diagram should provide a more clear description of the relationships in the system and allow for the system to be physically visible before code is written.

Lastly, a use case diagram will be included as an overview of the system in illustrated form. This diagram will allow the designing of the system from the perspective of the end user, which overall is key as catering to the user is of the utmost importance. This diagram is generally helpful in coding the system as it should give an overall view of the systems behaviour although concise and clearly without showing detail of any use cases which are covered quite heavily through diagram and discussion.

## Use Case 1: Search Car

**Actor:** Online Customer

**Trigger**: User chooses to search for car.

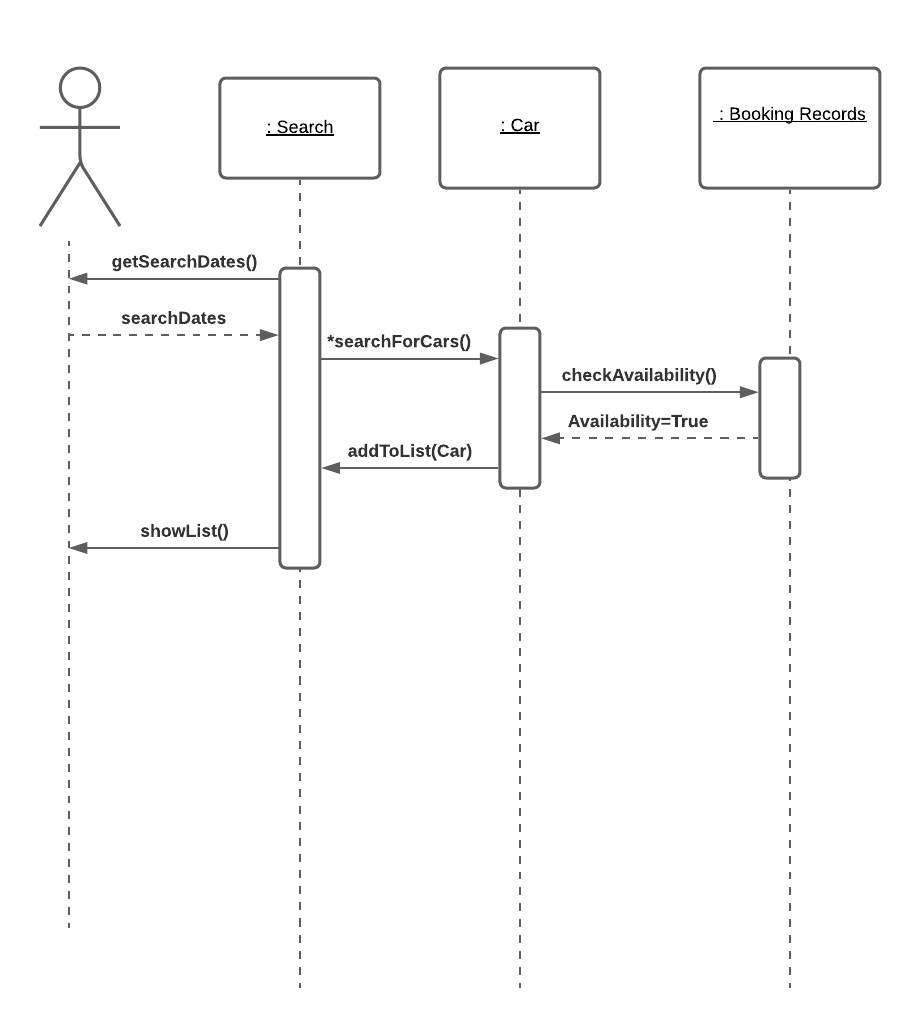
**Post-conditions:**

* Available cars have been displayed to user.

### Main Success Scenario

1. User browses website and selects to search for car within date range.
2. System updates car availability and selection.
3. System displays list of currently available cars that meet date range.
4. User selects car from list.

### Sequence Diagram UC1 MSS



## Use Case 2: Book Car

**Actor:** Online Customer

**Trigger**: User chooses to book car.

**Post-conditions:**

* User has successfully booked a car.
* System has displayed booking number.
* Tax is calculated and saved.
* Payment authorisation approvals recorded.
* Payment method is billed.
* Confirmation email of booking is sent to user with invoice.
* Remove booked car from available cars.

### Main Success Scenario

1. User selects option to book car.
2. System invites user to log in to their account or checkout as guest.
3. User chooses option to login.
4. User is moved to UC4.MSS.
5. System returns user from UC4.MSS.
6. System displays booking details and full payment required.
7. User confirms payment.
8. System authorises purchase and confirms booking.
9. User is prompted with booking number accompanied by hire details.
10. System sends confirmation email of booking to user’s email containing invoice.
11. System logs completed purchase.

### Sequence Diagram UC2 MSS

### 

### Alternative Success Scenario

**AS1** **User continues without login.**

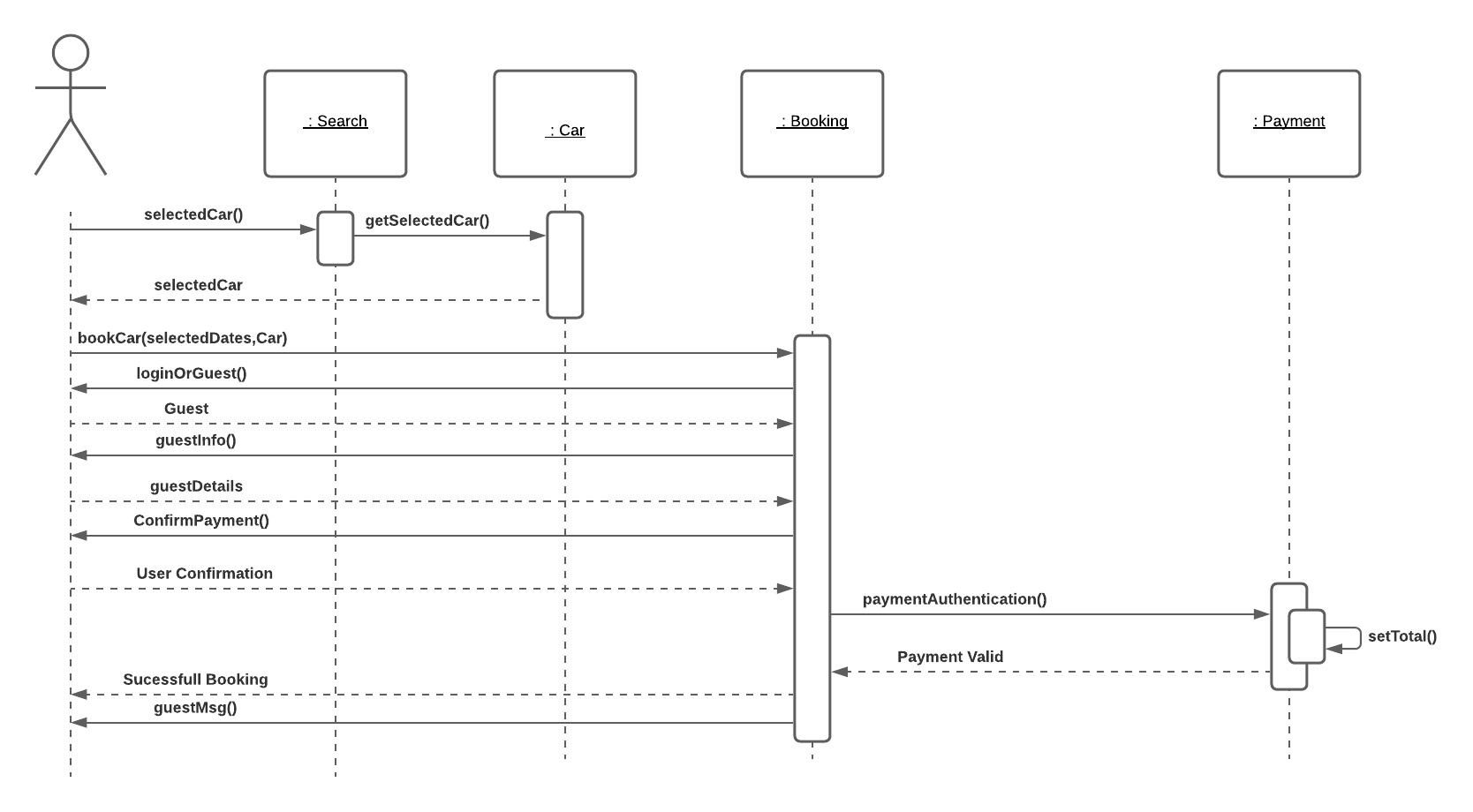
**AS1.1** At MSS.3, User selects to continue with guest option.

**AS1.2** User is prompted to enter guest payment details.

**AS1.3** User enters card details to prompt.

**AS1.4** System confirms details and resumes at MSS.7.

### Sequence Diagram UC2 AS



### Failure Scenarios

**FS1** **Authorisation Failure**

**FS2.1** At MSS.8, system sends request to users’ bank for authorisation of purchase.

**FS2.2** System receives authorisation failure from users’ bank for purchase.

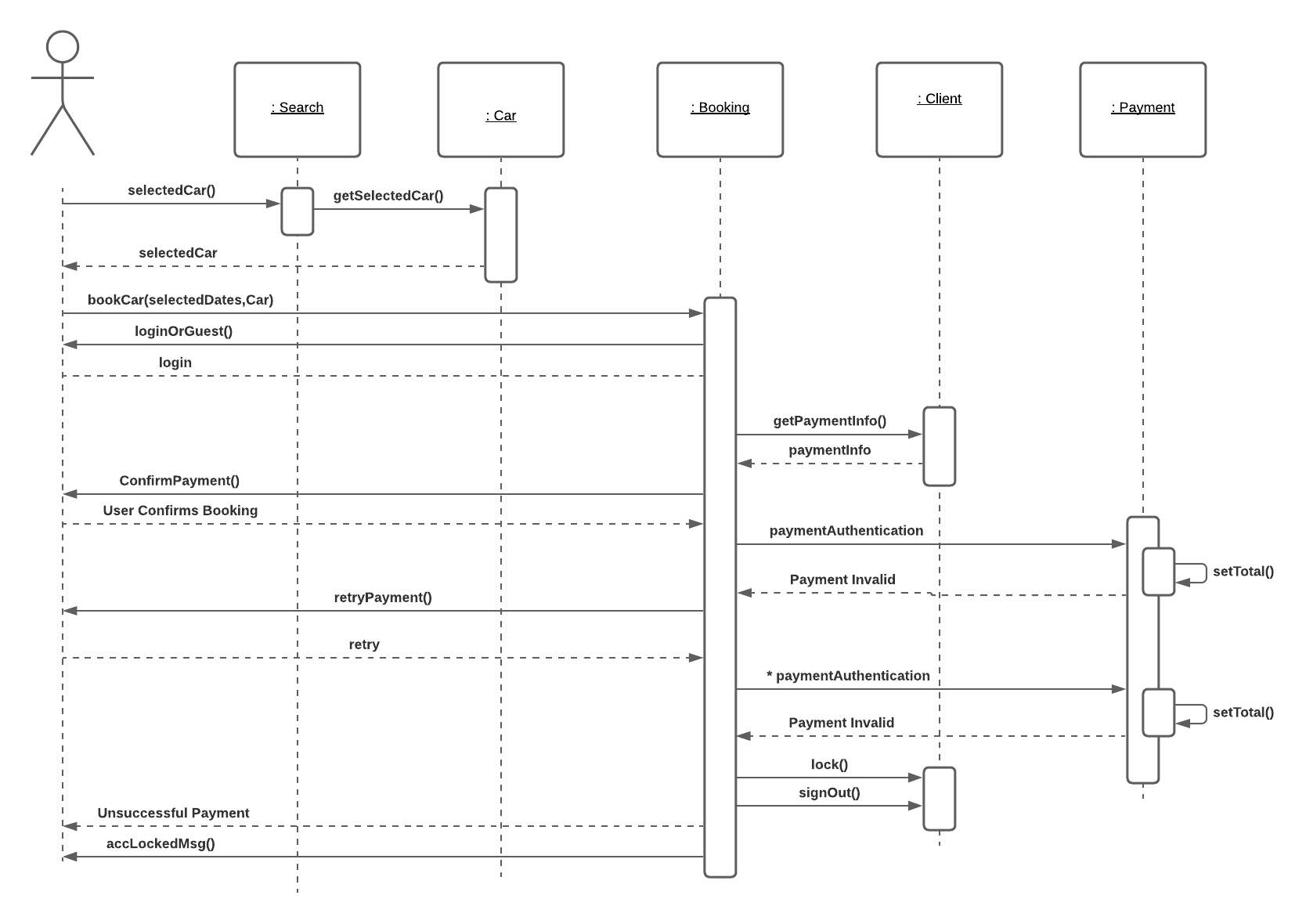
**FS2.3** System requests user to retry new details for purchase.

**FS2.4** User re-enters information.

**FS2.5** System does not authorise the purchase and locks the account for safety.

**FS2.6** System sends confirmation of locked account to user.

### Sequence Diagram UC2 FS



## Use Case 3: Guest Booking

**Actor:** Online Customer

**Trigger**: User selects to book as guest.

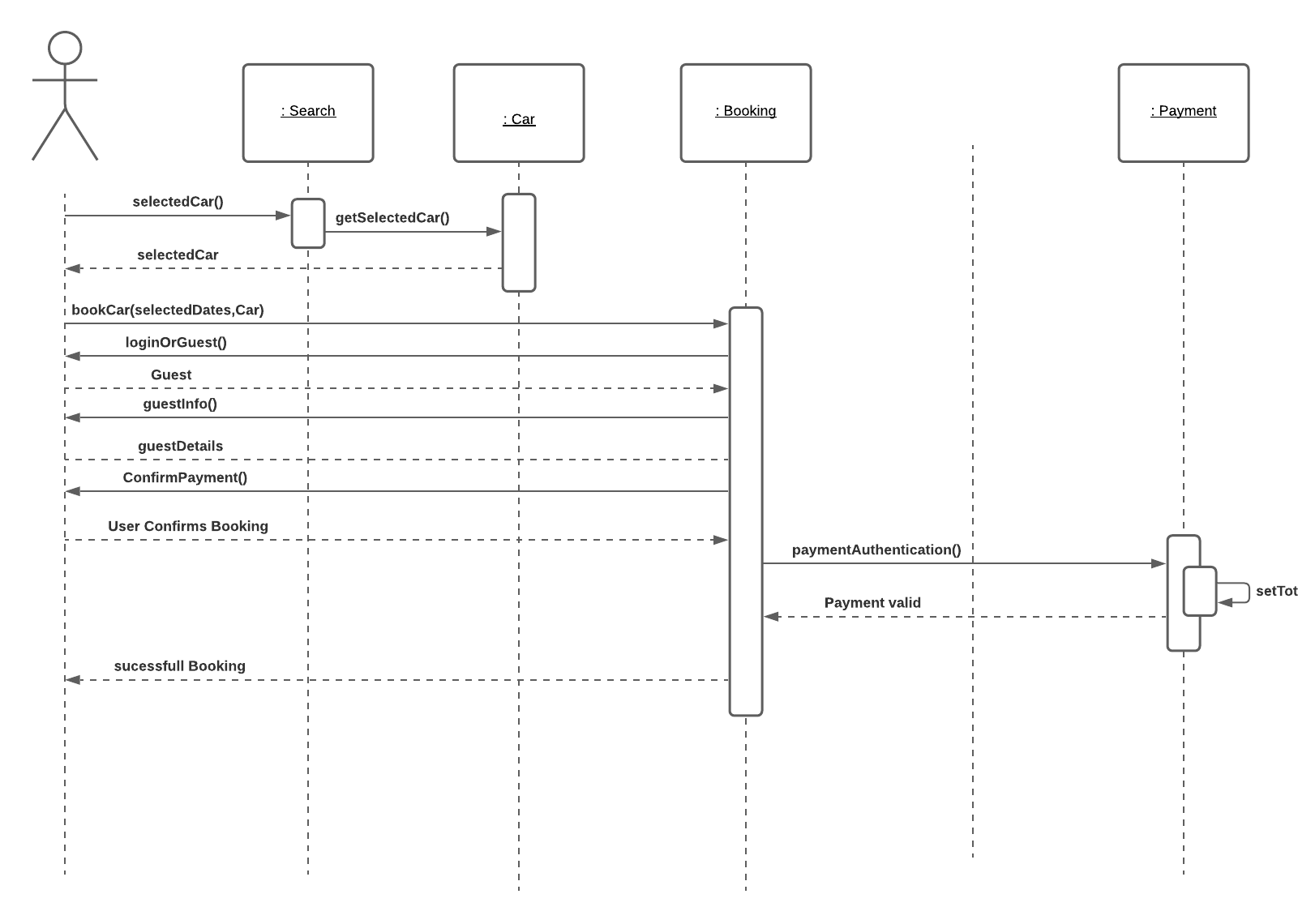
**Post-conditions:**

* User has successfully booked a car.
* System has displayed booking number.
* Tax is calculated and saved.
* Payment authorisation approvals recorded.
* Payment method is billed.
* Confirmation email of booking is sent to user with invoice.
* Remove booked car from available cars.

### Main Success Scenario

1. User selects to proceed as guest.
2. System asks user for guest information.
3. System prompts user to enter payment details.
4. User enters details to system.
5. System prompts confirmation of payment.
6. User confirms payment.
7. System authenticates payment.
8. System displays successful booking and logs completed booking.

## Sequence Diagram UC3 MSS



### Failure Scenarios

**FS1** **Login Failure.**

**FS2.1** At MSS.7, system sends request to users’ bank for authorisation of purchase.

**FS2.2** System receives authorisation failure from users’ bank for purchase.

**FS2.3** System requests user to retry new details for purchase.

**FS2.4** User re-enters information.

**FS2.5** System does not authorise the purchase and locks the account for safety.

**FS2.6** System sends confirmation of locked account to user.

### Sequence Diagram UC3 FS

## Use Case 4: Login

**Actor:** Online Customer

**Trigger**: User selects to login.

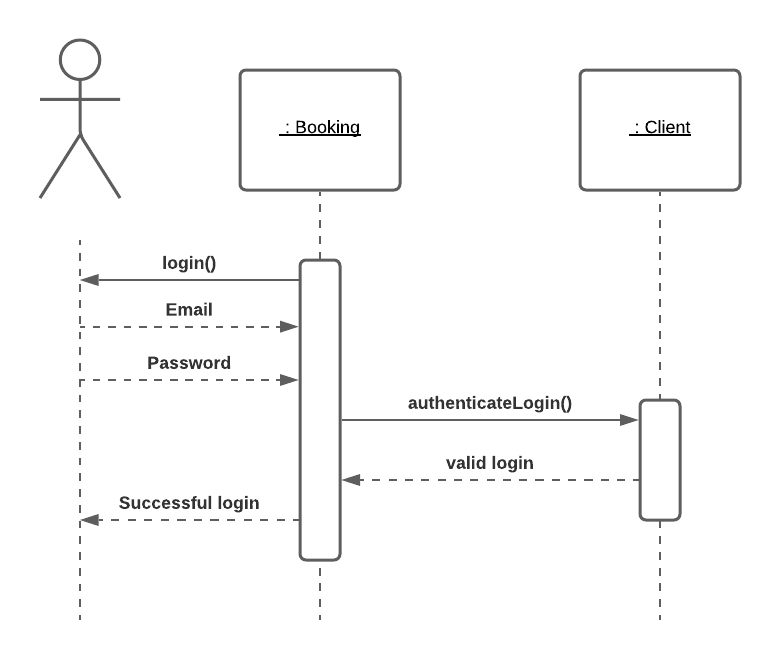
**Post-conditions:**

* User has been logged into account.
* System loads account details.
* System updates details.
* System updates login status.

### Main Success Scenario

1. User selects to login to account.
2. System prompts user to enter their email address and password.
3. System authenticates login information.
4. System logs user into account.
5. System loads user information and account details.
6. System returns user to previous action.

### Sequence Diagram UC4 MSS



### Alternative Success Scenario

**AS1** **User is prompted for authentication for login.**

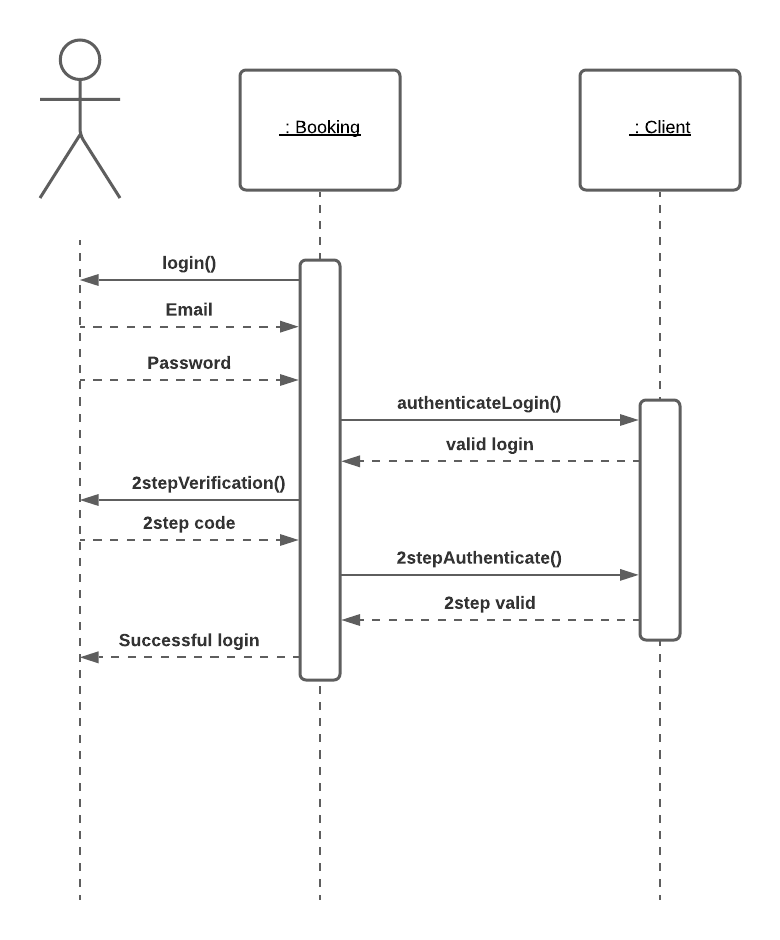
**AS1.1** At MSS.3, System prompts user to authenticate the login with paired device.

**AS1.2** User enters authentication code.

**AS1.3** System confirms correct auth code.

**AS1.4** System returns to MSS.4.

### Sequence Diagram UC4 AS



### Failure Scenarios

**FS1** **User is enters incorrect details twice.**

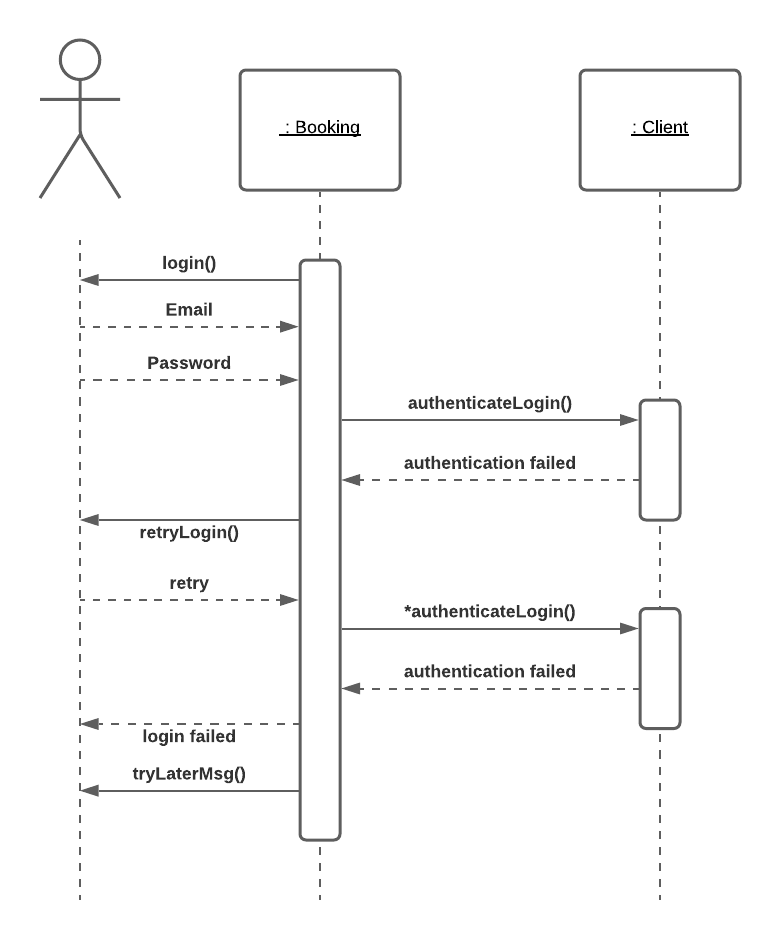
**FS1.1** At MSS.3, System prompts user to re-enter the login details.

**FS1.2** User enters login details.

**FS1.3** System cannot confirm authentication of details due to incorrect password.

**FS1.4** System prompts a failed login, imposing cooldown of 5 minutes and try again later message to user.

### Sequence Diagram UC4 FS



## Use Case 5: Edit Booking

**Actor:** Online Customer

**Trigger**: User selects to Guest booking.

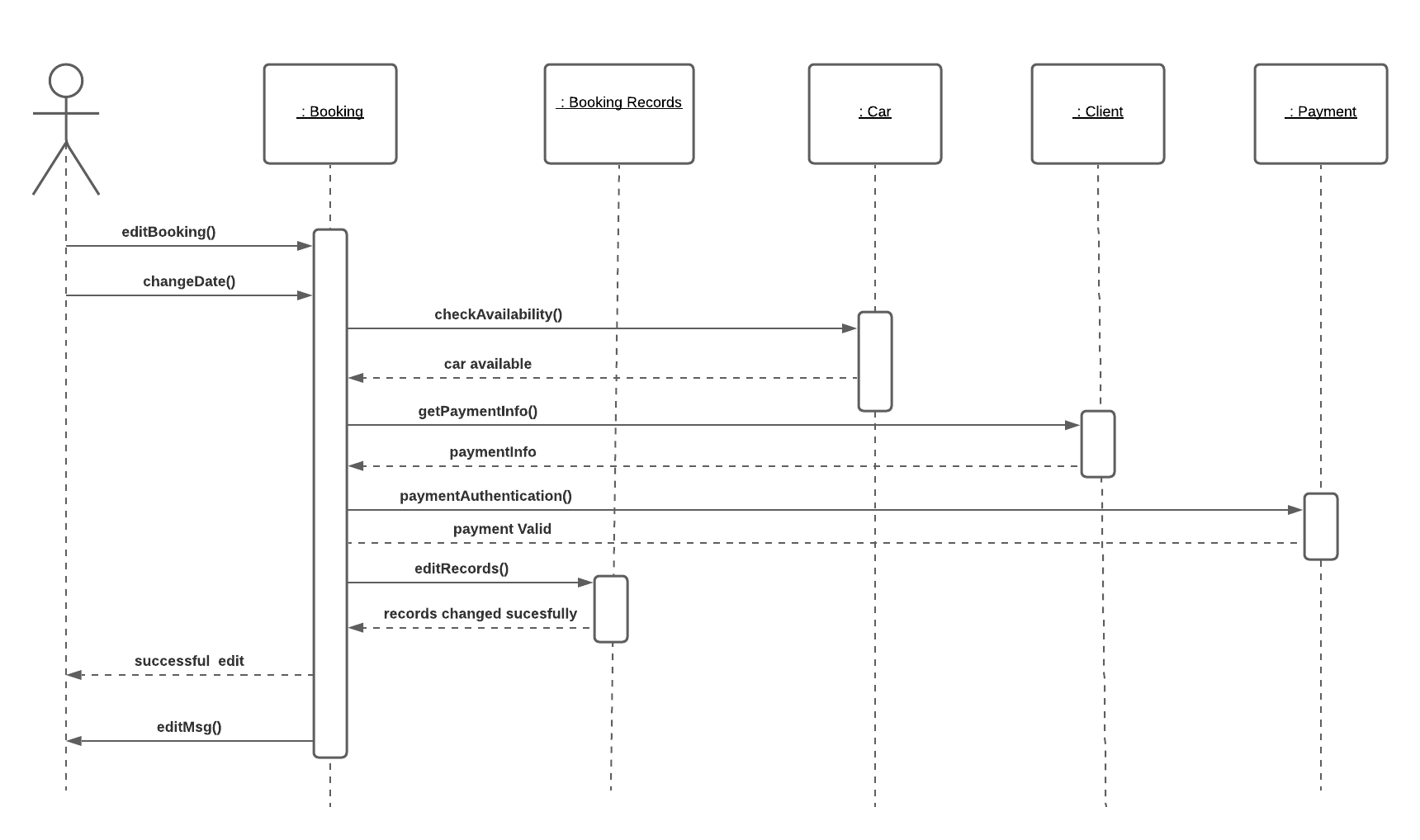
**Post-conditions:**

* User has successfully edited a booking.
* System has displayed booking number.
* Tax is calculated and saved.
* Payment authorisation approvals recorded.
* Payment method is billed.
* Confirmation email of edited booking is sent to user with invoice.
* Remove booked car from available cars.

### Main Success Scenario

1. User selects to edit booking.
2. System displays options to change information.
3. User selects to change date of booking.
4. System prompts user to enter new date for booking.
5. User enters new date of booking.
6. System searches available cars for entered date range.
7. System returns available booking entered.
8. System retrieves payment information from previous booking.
9. Payment is authenticated and the billed new total.
10. System amends edited booking records with new information.
11. User is prompted with successful booking and notification containing booking changes.

### Sequence Diagram UC5 MSS



### Failure Scenarios

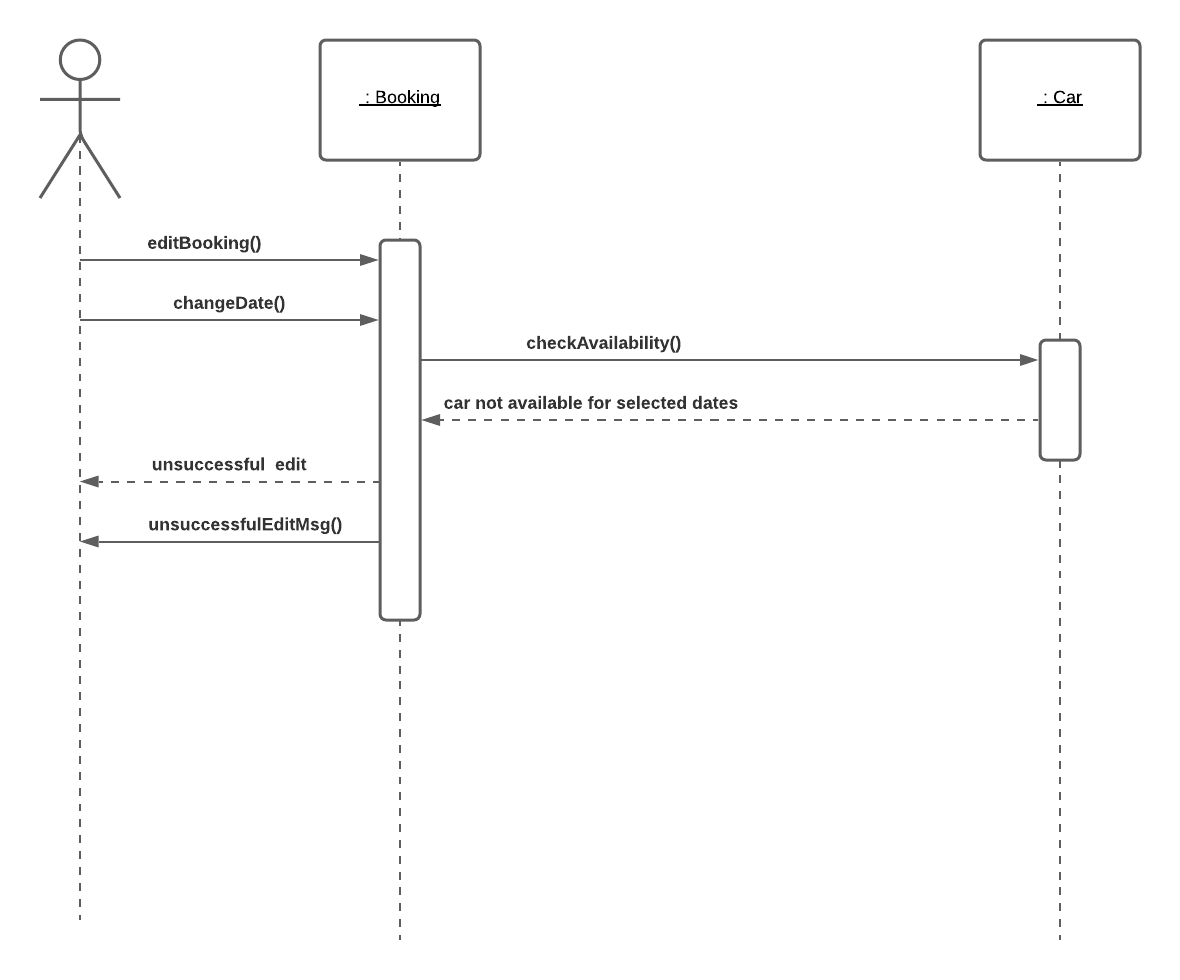
**FS1** **User enters unavailable date range.**

**FS2.1** At MSS.6, System cannot find available car for date range.

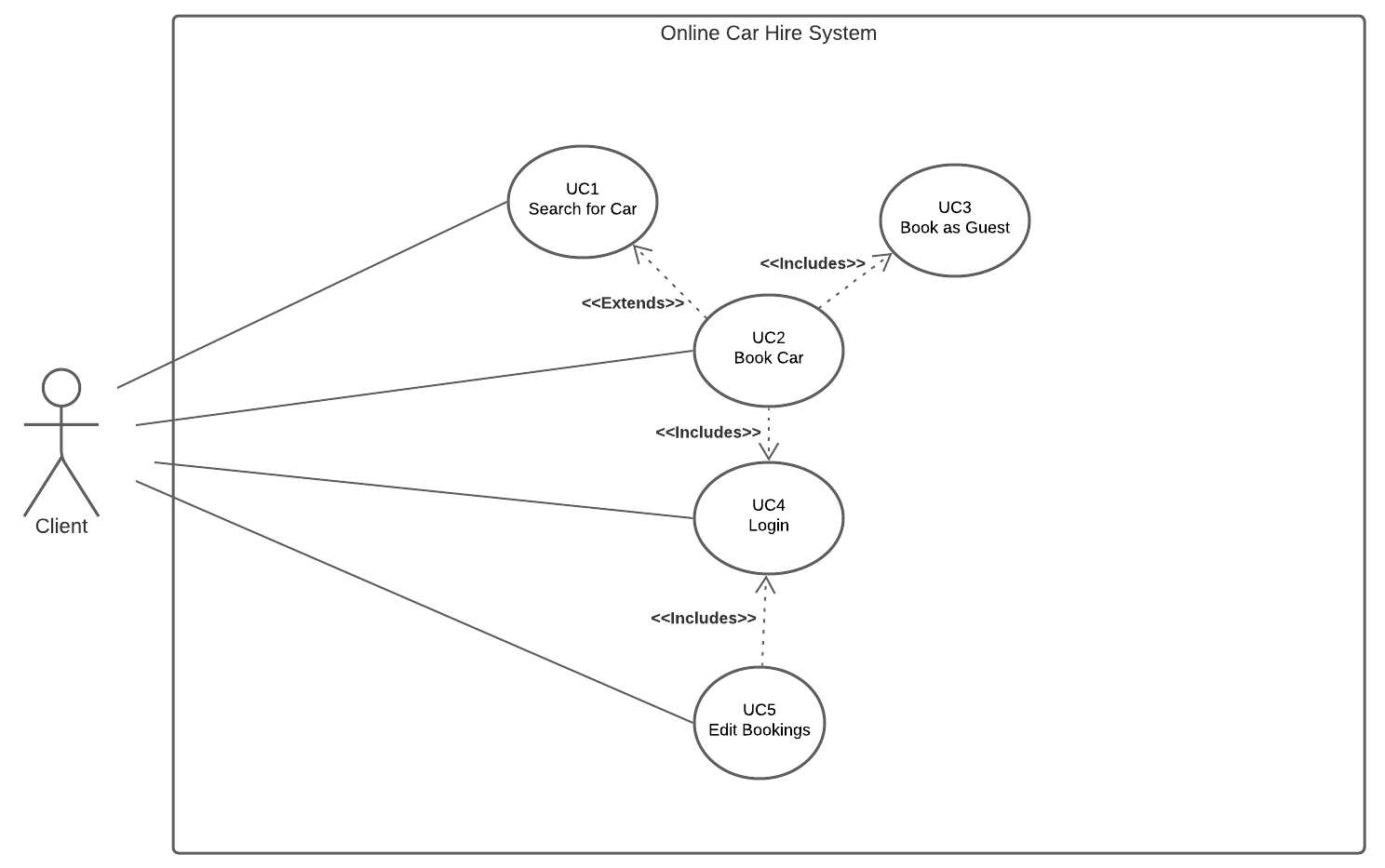
**FS2.2** System prompts user with unsuccessful booking edition.

**FS2.3** System notifies user with edited booking failure message.

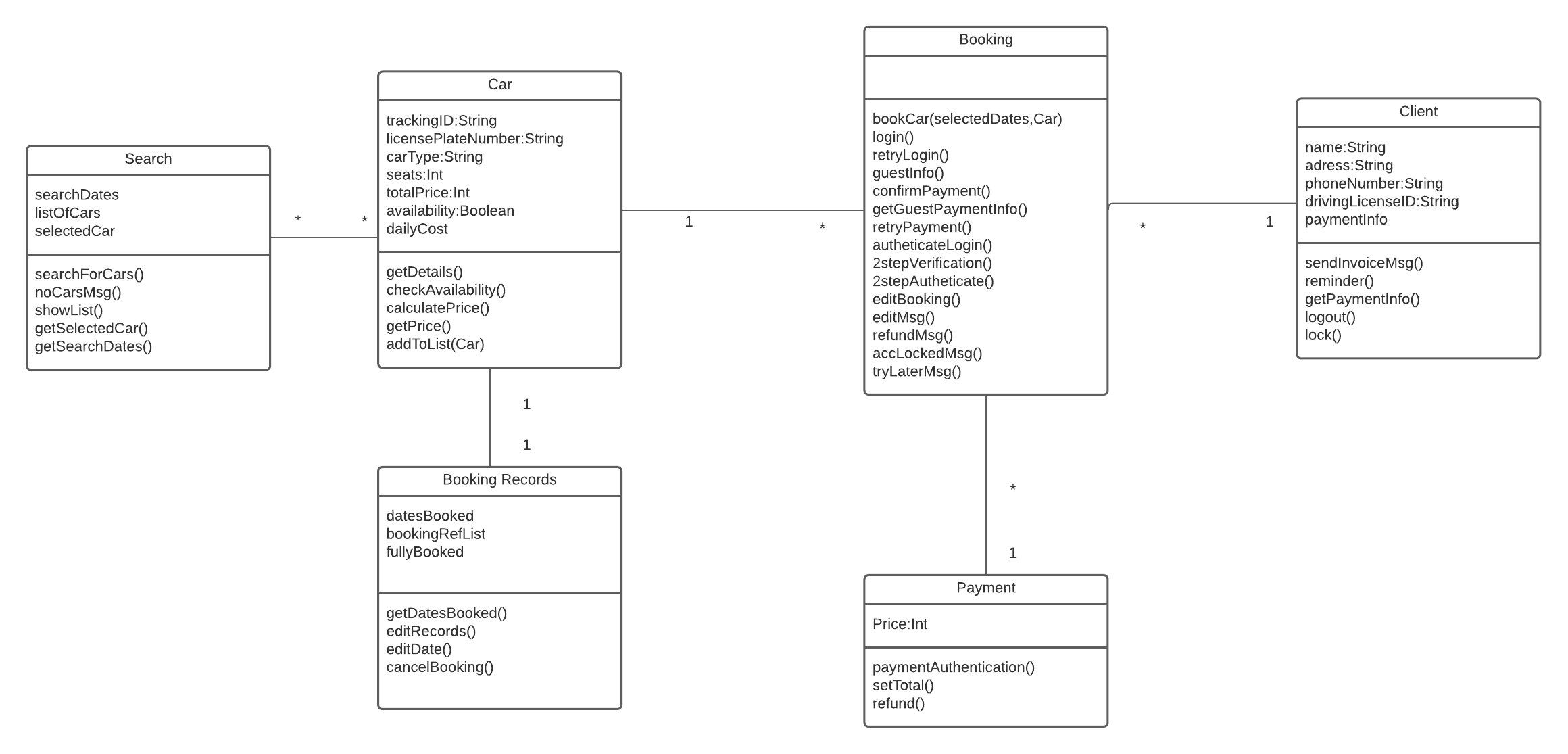
### Sequence Diagram UC5 FS



## Use Case Diagram of Car Hire System



## Class Diagram of Car Hire System



## Hire System Test code

import org.junit.Assert;  
import org.junit.Before;  
import org.junit.Test;  
  
  
public class BookingTest {  
 private carBooking car = new carBooking();  
 private Customer customer;  
 private CarType fourDoor;  
 private CarType twoDoor;  
 private Car car1;  
 private Car car2;  
 private Car car3;  
 private Car car4;  
 private Date startDate;  
 private Date endDate;  
  
 public BookingTest() {  
 }  
  
 @Before  
 public void setup() {  
 this.customer = new Customer("John", "Doe", "1234", "jd@test.com", "1234", "9999888877776666");  
 this.fourDoor = new carBooking("four door", 1);  
 this.twoDoor = new carBooking("two door", 1);  
 this.car1 = new Car(1, this.availableCar);  
 this.car2 = new Car(2, this.availableCar);  
 this.car3 = new Car(3, this.availableCar);  
 this.car4 = new Car(4, this.availableCar);  
  
 account1 = new Account("John Kelly", "3 yellow road", "11100000");  
 account2 = new Account("John murphy", "3 blue road", "12220000");  
  
 this.carsAvailable.addAllCars(new Room[]{this.car1, this.car2, this.car3, this.car4});  
 bookingRegister bookReg = bookingRegister.getAvailableCars();  
 bookReg.getCar(fourDoor, 2019);  
 bookReg.getCar(fourDoor, 2020);  
 bookReg.getCar(twoDoor, 111111);  
 this.startDate = bookingRegister.getDates();  
 bookingRegister.set("2/11/2020");  
 this.endDate = bookingRegister.getDates();  
 }  
  
 @Test  
 public void firstAvailableTwoDoor() {  
 Assert.assertEquals(this.car1, this.cars.getAvailableCars(this.startDate, this.endDate, this.twoDoor));  
 Assert.assertEquals(this.car2, this.cars.getAvailableCars(this.startDate, this.endDate, this.fourDoor));  
 }  
  
 @Test  
 public void firstAvailableFourDoor() {  
 Assert.assertEquals(this.car2, this.cars.getAvailableRoom(this.startDate, this.endDate, this.fourDoor));  
 }  
  
 @Test  
 public void newBookingIsAddedToBookingRegister() {  
 Car car = this.cars.getAvailableCar(this.startDate, this.endDate, this.twoDoor);  
 Car car2 = this.cars.getAvailableCar(this.startDate, this.endDate, this.fourDoor);  
 Booking booking = new Booking(this.customer, this.startDate, this.endDate, twoDoor);  
 Assert.assertTrue(car.getBookingRegister().contains(booking));  
 Assert.assertTrue(car2.getBookingRegister().contains(booking));  
 }  
  
  
 @Test  
 public void getAvailableCars() {  
 Assert.assertTrue(car.getAvailableCar().contains(car));  
 Assert.assertTrue(car2.getAvailable().contains(car2));  
 }  
  
  
 @Test  
 void getAccPhoneNumber() {  
 assertEquals("1110000", account1.getAccNumber());  
 assertEquals("11222000", account2.getAccNumber());  
 }  
  
  
 @Test  
 void getAccountName() {  
 assertEquals("John Kelly", account1.getAccountName());  
 assertEquals("John Murphy", account2.getAccNumber());  
 }  
  
  
 @Test  
 void getAccountAddress() {  
 assertEquals("3 Yellow Road", account1.getAccountAddress());  
 assertEquals("3 blue Road", account2.getAccNumber());  
 }  
  
  
}