Project Description

Group : { Andrew Spano, Jimmy Nguyen, Zak Gudlin }

This project is a travel assistant application which can be used by a frequent flyer to book and manage their flights. Users will be able to book flights based on arrival time & destination, set up their baggage plans (with additional costs), and manage their flights as well. Management features would include viewing the itinerary of their flight and cancellation of said flight. Users will also have the option to book a rental car from the destination airport. Users will be able to choose from several different vehicles. All payments would be handled through the in-house system.

System Requirements

1 is highest, 3 is lowest

Identifier	Priority	Requirement
REQ1	1a	The system shall allow an administrator to create flights for the user to select from.
REQ2	1b	The system shall allow an administrator to create planes that can be assigned to flights.
REQ3	1c	The system shall allow users to book flights (view arrival and destination).
REQ4	1d	The system shall allow users to setup their baggage plan for their flight.
REQ5	1e	The system shall allow users to cancel their flight.
REQ6	1f	The system shall allow users to view the itinerary of their flight.
REQ7	2a	The system should have a login feature where users are able to create/login to their user accounts.
REQ8	2b	The system shall allow users to pay for the flight using our application.
REQ9	2c	The system shall allow the user to logout of the application.
REQ10	3a	The system shall allow users to utilize a car rental service at their airport destination.
REQ11	3b	The system shall display various types of vehicles to rent (with rates).
REQ12	3c	The system shall allow users to book their car rental.
REQ13	3d	The system shall allow users to pay for car rentals using our application.
REQ14	3e	The system shall allow users to view their booked car rentals.

User Stories

Person = 1 pts, 1 Hour = 2 pts, Meeting = 3 pts

Identifier	User Story	Size
ST-1	As an administrator of the system, I will be able to create flights for a user to select from.	3 pt(s)
ST-2	As an administrator of the system, I will be able to create planes to be assigned to specific flights.	3 pt(s)
ST-3	As a user, I will be able to book flights that I am interested in based on arrival/destination.	3 pt(s)
ST-4	As a user, I will be able to choose my baggage plan for my booked flight.	3 pt(s)
ST-5	As a user, I will be able to cancel my booked flight.	3 pt(s)
ST-6	As a user, I will be able to find and view the itinerary of my booked flight.	4 pt(s)
ST-7	As a prospective client, I will be able to create a user account to access the services.	4 pt(s)
ST-8	As a user, I will be able to pay for my booked flight.	3 pt(s)
ST-9	As a user, I should be able to log out of the system.	2 pt(s)
ST-10	As a user, I will be able to utilize the rental car service at my destination airport	3 pt(s)
ST-11	As a user, I will be able to view the vehicles available for rental with rates included.	4 pt(s)
ST-12	As a user, I will be able to rent a vehicle based on type and make.	6 pt(s)
ST-13	As a user, I will be able to pay for the vehicle rental.	3 pt(s)
ST-14	As a user, I will be able to view my vehicle rentals.	2 pt(s)

Use Cases

Use Case UC-1:	Flight Booking
Related Requirements:	REQ3, REQ4
Initiating Actor:	A Customer
Actor's Goal:	To successfully book a flight
Participating Actors:	Flight, Airplane, User, Booking, UserController
Preconditions:	There must be a user, a flight, a rental system, and an airplane
Postconditions:	The user should have a booked trip saved in a trip list with automatic seating assigned

Flow of Events for Main Success Scenario	X	X
<-	1.	The user is presented with flight options to choose from
->	2.	The user is able to select a flight
<-	3.	The user is presented with baggage options to choose from
->	4.	The user is able to select a baggage plan
<-	5.	The user is notified that their flight has been successfully booked

Use Case UC-2:	Car Booking
Related Requirements:	REQ10, REQ11, REQ12
Initiating Actor:	A Customer
Actor's Goal:	To successfully book a rental car
Participating Actors:	User, RentalSystem, UserController
Preconditions:	There must be a user and a rental system
Postconditions:	The user should have a rental car assigned

Flow of Events for Main Success Scenario	X	X
<-	1.	The user is presented with rental car options
->	2.	The user is able to select a car by make and model
<-	3.	The user is notified that their car has been rented and set aside for them

Use Case UC-3:	**Payment**
Related Requirements:	REQ8, REQ13
Initiating Actor:	A User
Actor's Goal:	To make payment for trips/rental car
Participating Actors:	UserController, User, Payment
Preconditions:	To have a user who has booked a flight (car rental optional)
Postconditions:	User has added payment details to pay for services available

Flow of Events for Main Success Scenario	**X**	**X**
<-	1.	The user is presented with input fields for payment details
->	2.	The user inputs required information
<-	3.	The user is notified that their payment method is valid and has been confirmed

Use Case UC-4:	**Create User**
Related Requirements:	REQ7
Initiating Actor:	A Customer
Actor's Goal:	To successfully create a user account
Participating Actors:	UserController, User
Preconditions:	Application is running
Postconditions:	The user has created an account which they can use to login

Flow of Events for Main Success Scenario	**X**	**X**
<-	1.	The user is presented with proper fields to enter their details
->	2.	The user inputs their user name, password, and confirms their details
<-	3.	(If a user name exists, user must choose different username)
<-	4.	(If a password is invalid, user must choose different password)
->	5.	The user inputs a valid user name and password and confirms details
<-	6.	The user is notified of a successful account creation and an account is generated with the input information

Use Case UC-5:	**Login/Logout**
Related Requirements:	REQ7, REQ9
Initiating Actor:	A Customer
Actor's Goal:	To login to the APZ application
Participating Actors:	UserController, User
Preconditions:	Application is running and has loaded a list of saved created users
Postconditions:	User has successfully logged into the application and can access internal services

Flow of Events for Main Success Scenario	**X**	**X**
<-	1.	The user is presented with an input for user, password, and an action button
->	2.	The user inputs their details in proper fields and action with the button
<-	3.	The user is notified of their successful action and brought to the main application page
Flow of Events for Extensions (Alternate Scenarios)	X	X
<-	1.	The user is presented with the option to logout of the application
->	2.	The user selects the option to logout
<-	3.	The user is notified of their successful action and brought to the application's login screen

Use Case UC-6:	**Cancel Trip**
Related Requirements:	REQ3, REQ5
Initiating Actor:	A Customer
Actor's Goal:	To cancel a booked trip
Participating Actors:	UserController, User, Booking
Preconditions:	To have a user who has booked a trip (flight)
Postconditions:	The user is notified that their trip has been cancelled

Flow of Events for Main Success Scenario	**X**	**X**
<-	1.	The user is presented with options to cancel a trip based on a generated list
->	2.	The user selects one of the trips to cancel and confirms their cancellation
<-	3.	The user is notified that their trip is cancelled and it is removed from their list of trips

Use Case UC-7:	View Itinerary
Related Requirements:	REQ3, REQ6
Initiating Actor:	A Customer
Actor's Goal:	To view details of a booked trip
Participating Actors:	UserController, User, Booking, Flight, Airplane, Seating, Payment
Preconditions:	To have a user who has booked a trip and has made payment
Postconditions:	User is presented with a page with details of their scheduled trips

Flow of Events for Main Success Scenario	X	X
<-	1.	The user is presented with a list of their trips to view
->	2.	The user selects a trip from the list to generate a detailed itinerary
<-	3.	A detailed itinerary is presented to the user to view

Use Case UC-8:	Display Car Rental
Related Requirements:	REQ12, REQ14
Initiating Actor:	A Customer
Actor's Goal:	To view a booked car rental
Participating Actors:	UserController, User, RentalSystem, Booking
Preconditions:	To have a user who has booked a trip and a rental car
Postconditions:	User is presented with a list of car rentals that they have made

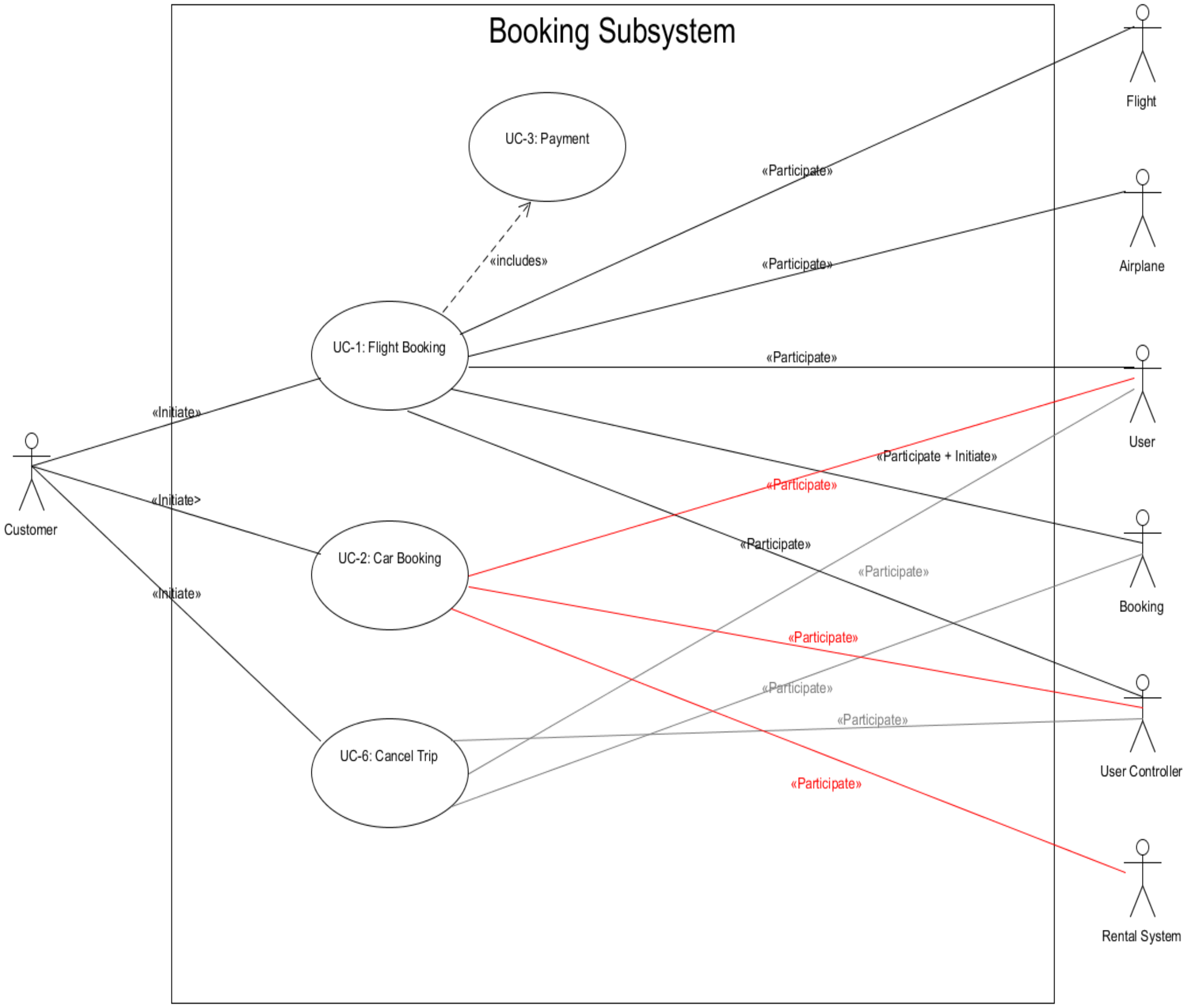
Flow of Events for Main Success Scenario	X	X
<-	1.	The user is presented with a list of car rentals that they have made, including the date booked and the cost of each rental

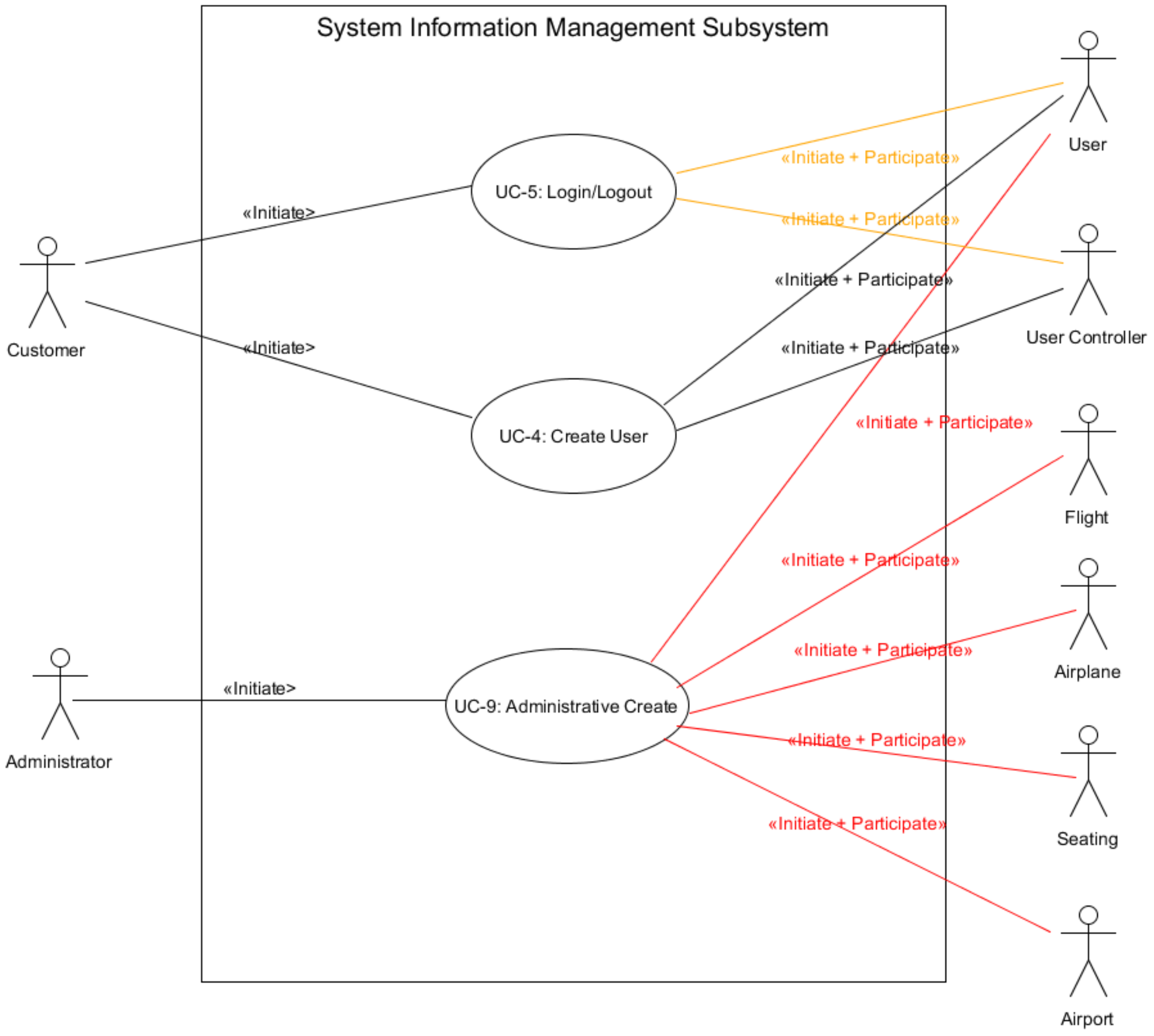
Use Case UC-9:	**Administrative Create**
Related Requirements:	REQ1, REQ2
Initiating Actor:	An Admin
Actor's Goal:	To create flights, planes, airports and users
Participating Actors:	Flight, Airplane, Seating, Airport, User
Preconditions:	To have the application running
Postconditions:	To generate a flight/airplane/airport/user

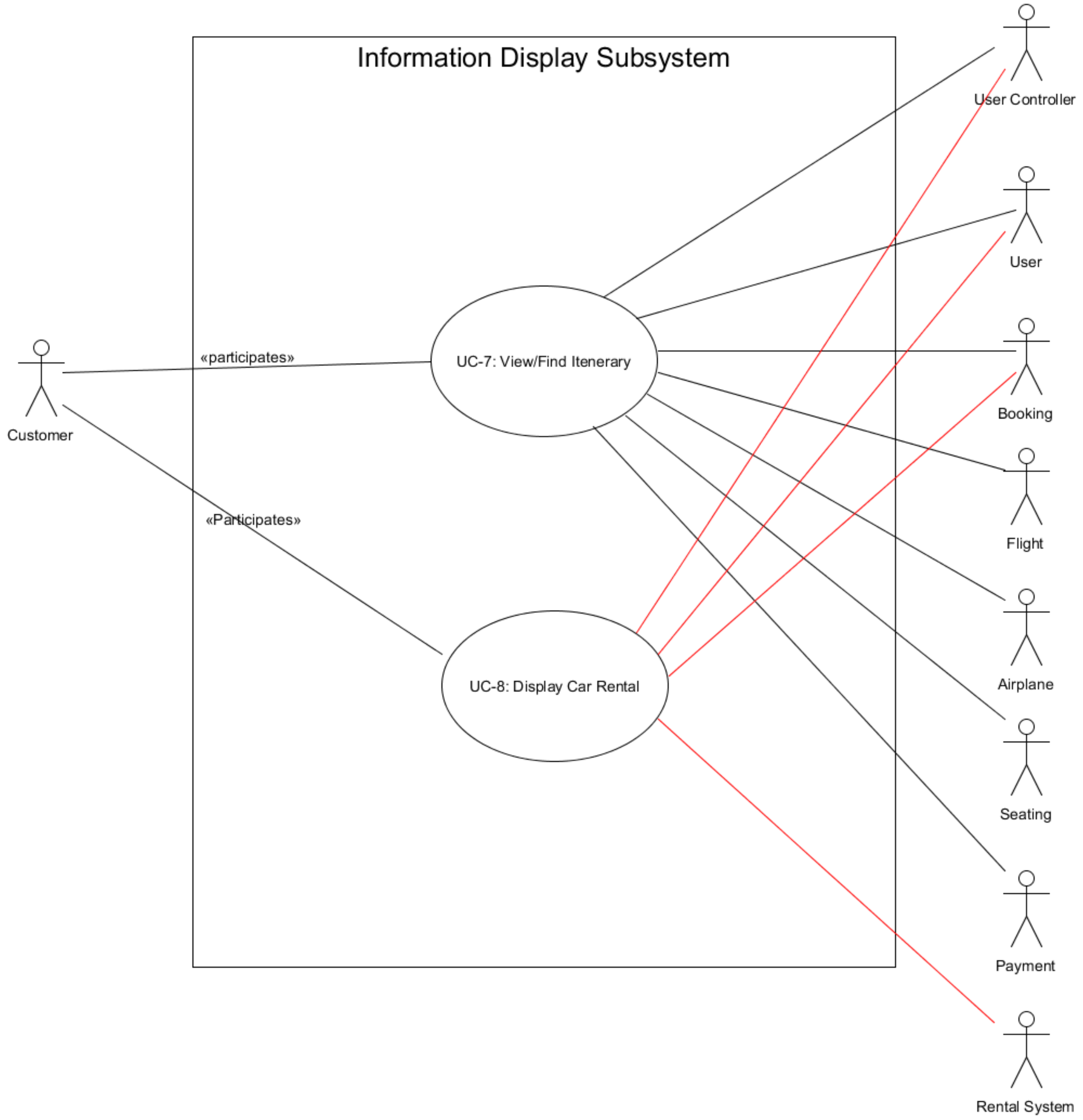
Flow of Events for Main Success Scenario	**X**	**X**
->	1.	The admin inputs details for a flight to be created (or plane/airport/user)
->	2.	The admin confirms the details entered
<-	3.	A flight/plane/airport/user is generated and is available for service

Traceability matrix

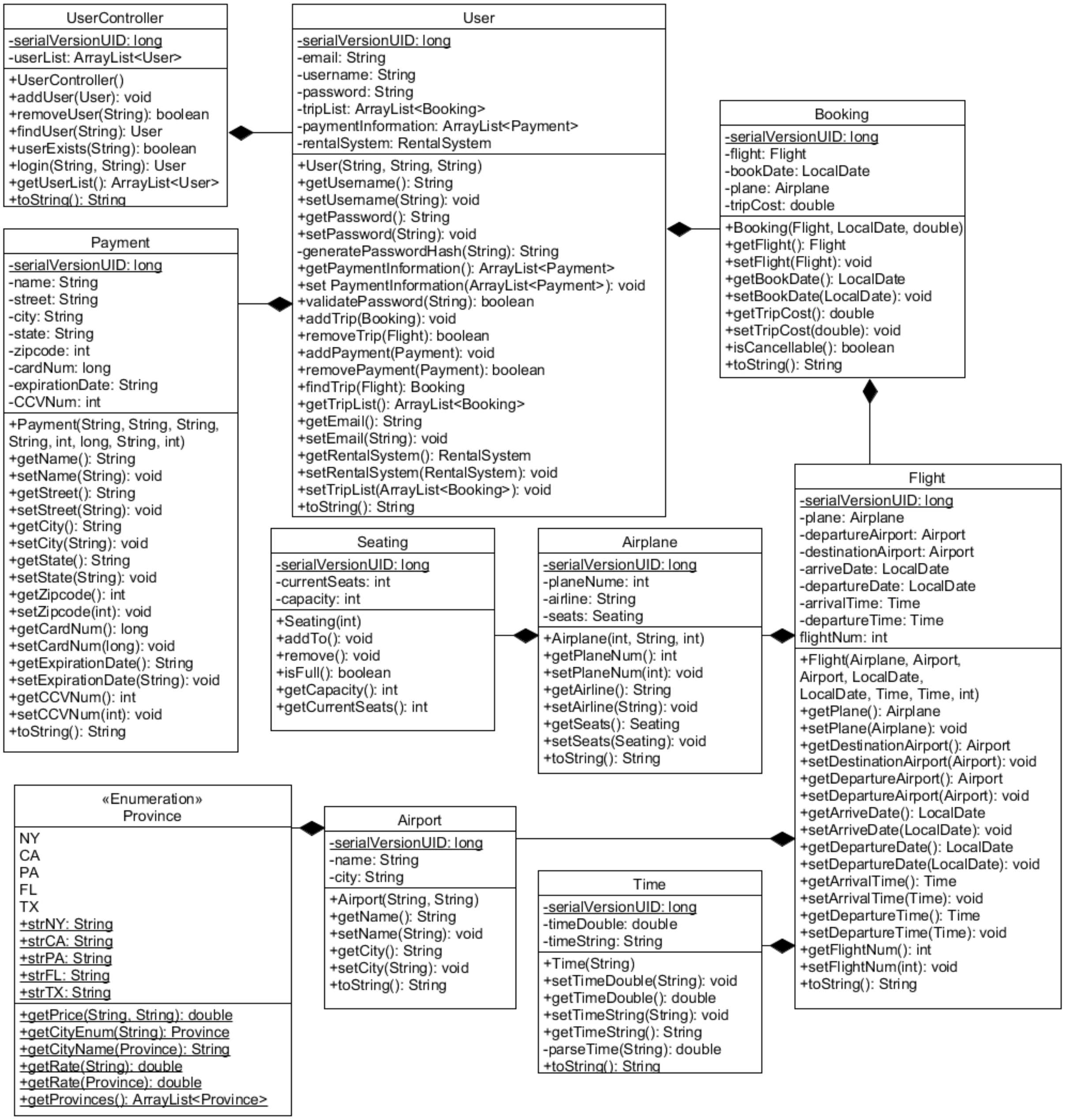
Requirements	**UC-1**	**UC-2**	**UC-3**	**UC-4**	**UC-5**	**UC-6**	**UC-7**	**UC-8**	**UC-9**
REQ1									X
REQ2									X
REQ3	X					X	X		
REQ4	X								
REQ5						X			
REQ6							X		
REQ7				X	X				
REQ8			X						
REQ9					X				
REQ10		X							
REQ11		X							
REQ12		X						X	
REQ13			X						
REQ14								X	


Booking Subsystem
UC-3: Payment
«includes»
UC-1: Flight Booking
UC-2: Car Booking
UC-6: Cancel Trip
Customer
«Initiate»
«Initiate>
«Initiate»
«Participate»
«Participate»
«Participate»
«Participate + Initiate»
«Participate»
«Participate»
«Participate»
«Participate»
«Participate»
«Participate»
«Participate»
Flight
Airplane
User
Booking
User Controller
Rental System

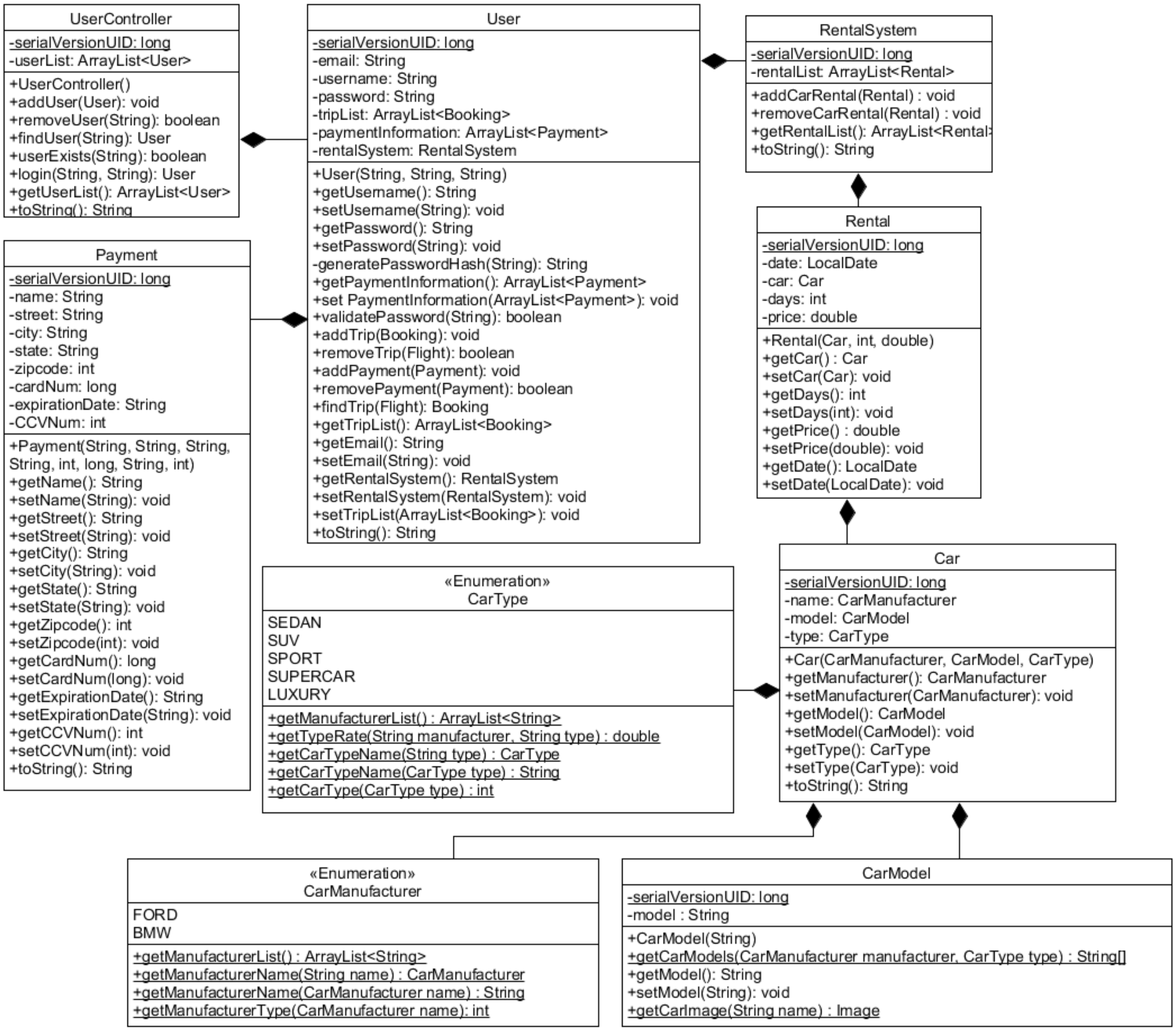

System Information Management Subsystem
UC-5: Login/Logout
UC-4: Create User
UC-9: Administrative Create
Customer
Administrator
User
User Controller
Flight
Airplane
Seating
Airport
«Initiate>
«Initiate>
«Initiate>
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»
«Initiate + Participate»


Information Display Subsystem
UC-7: View/Find Itenerary
UC-8: Display Car Rental
«participates»
«Participates»
Customer
User Controller
User
Booking
Flight
Airplane
Seating
Payment
Rental System

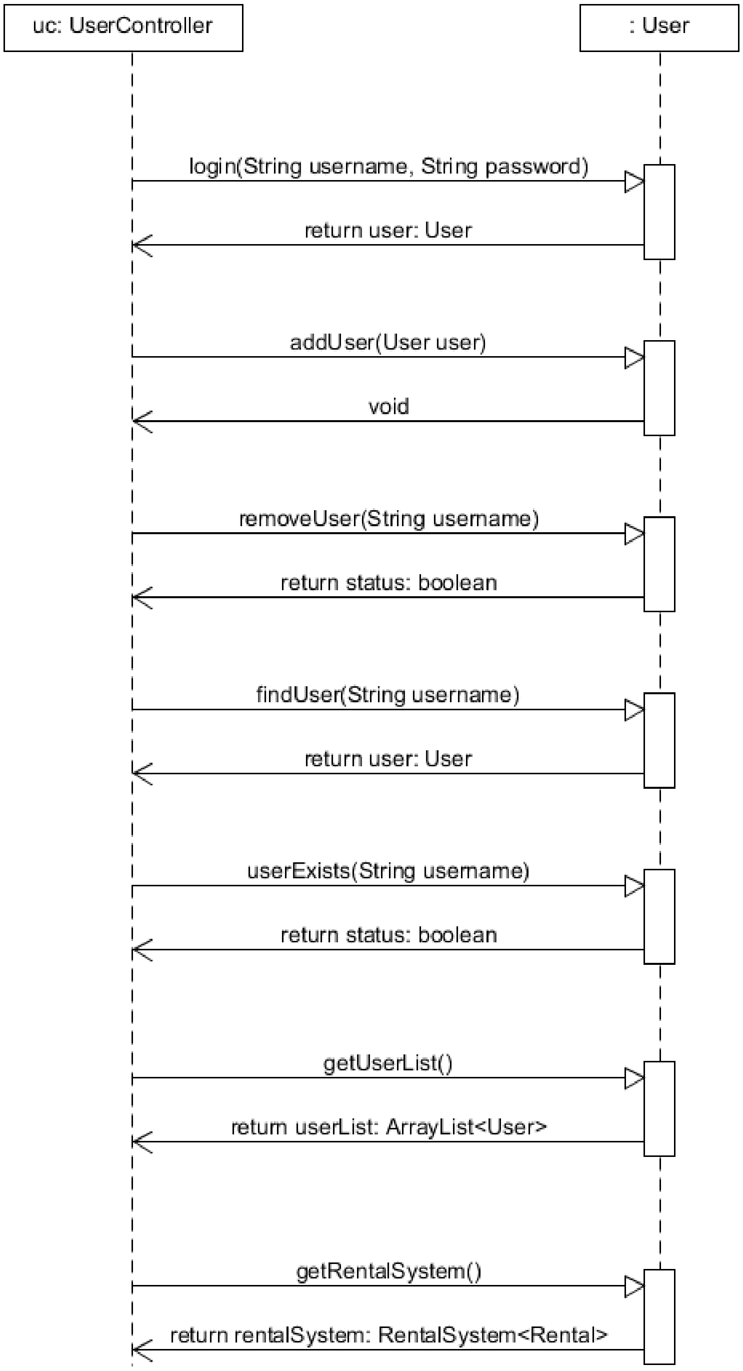
Flight Booking UML Class Diagrams



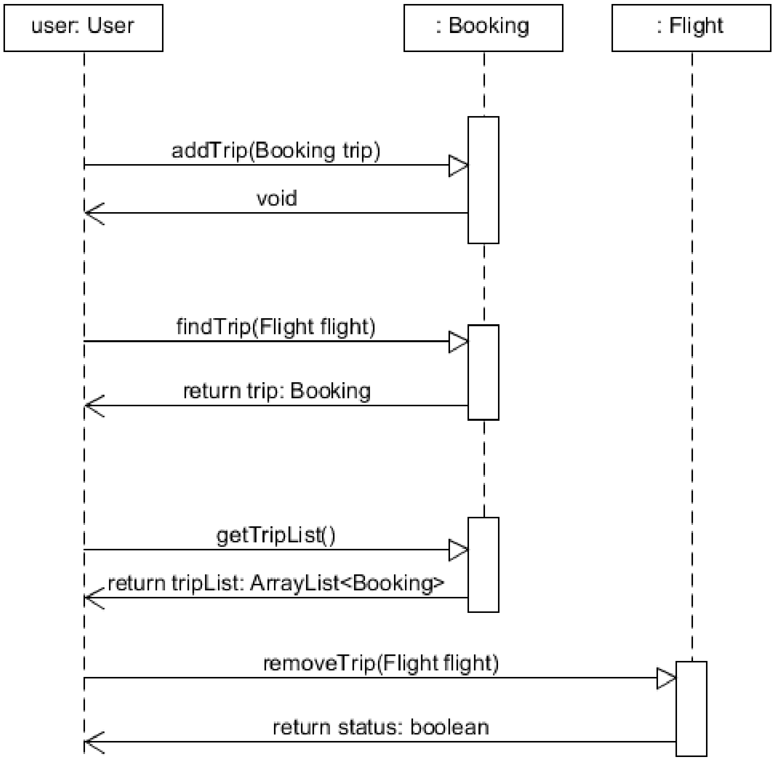
Car Booking UML Class Diagrams


UserController
-serialVersionUID: long
-userList: ArrayList<User>
+UserController()
+addUser(User): void
+removeUser(String): boolean
+findUser(String): User
+userExists(String): boolean
+login(String, String): User
+getUserList(): ArrayList<User>
+toString(): String
User
-serialVersionUID: long
-email: String
-username: String
-password: String
-tripList: ArrayList<Booking>
-paymentInformation: ArrayList<Payment>
-rentalSystem: RentalSystem
+User(String, String, String)
+getUsername(): String
+setUsername(String): void
+getPassword(): String
+setPassword(String): void
-generatePasswordHash(String): String
+getPaymentInformation(): ArrayList<Payment>
+set PaymentInformation(ArrayList<Payment>): void
+validatePassword(String): boolean
+addTrip(Booking): void
+removeTrip(Flight): boolean
+addPayment(Payment): void
+removePayment(Payment): boolean
+findTrip(Flight): Booking
+getTripList(): ArrayList<Booking>
+getEmail(): String
+setEmail(String): void
+getRentalSystem(): RentalSystem
+setRentalSystem(RentalSystem): void
+setTripList(ArrayList<Booking>): void
+toString(): String
RentalSystem
-serialVersionUID: long
-rentalList: ArrayList<Rental>
+addCarRental(Rental) : void
+removeCarRental(Rental) : void
+getRentalList(): ArrayList<Rental
+toString(): String
Rental
-serialVersionUID: long
-date: LocalDate
-car: Car
-days: int
-price: double
+Rental(Car, int, double)
+getCar() : Car
+setCar(Car): void
+getDays(): int
+setDays(int): void
+getPrice() : double
+setPrice(double): void
+getDate(): LocalDate
+setDate(LocalDate): void
Payment
-serialVersionUID: long
-name: String
-street: String
-city: String
-state: String
-zipcode: int
-cardNum: long
-expirationDate: String
-CCVNum: int
+Payment(String, String, String, String, int, long, String, int)
+getName(): String
+setName(String): void
+getStreet(): String
+setStreet(String): void
+getCity(): String
+setCity(String): void
+getState(): String
+setState(String): void
+getZipcode(): int
+setZipcode(int): void
+getCardNum(): long
+setCardNum(long): void
+getExpirationDate(): String
+setExpirationDate(String): void
+getCCVNum(): int
+setCCVNum(int): void
+toString(): String
«Enumeration»
CarType
SEDAN
SUV
SPORT
SUPERCAR
LUXURY
+getManufacturerList() : ArrayList<String>
+getTypeRate(String manufacturer, String type) : double
+getCarTypeName(String type) : CarType
+getCarTypeName(CarType type) : String
+getCarType(CarType type) : int
Car
-serialVersionUID: long
-name: CarManufacturer
-model: CarModel
-type: CarType
+Car(CarManufacturer, CarModel, CarType)
+getManufacturer(): CarManufacturer
+setManufacturer(CarManufacturer): void
+getModel(): CarModel
+setModel(CarModel): void
+getType(): CarType
+setType(CarType): void
+toString(): String
«Enumeration»
CarManufacturer
FORD
BMW
+getManufacturerList() : ArrayList<String>
+getManufacturerName(String name) : CarManufacturer
+getManufacturerName(CarManufacturer name) : String
+getManufacturerType(CarManufacturer name): int
CarModel
-serialVersionUID: long
-model : String
+CarModel(String)
+getCarModels(CarManufacturer manufacturer, CarType type) : String[]
+getModel(): String
+setModel(String): void
+getCarImage(String name) : Image

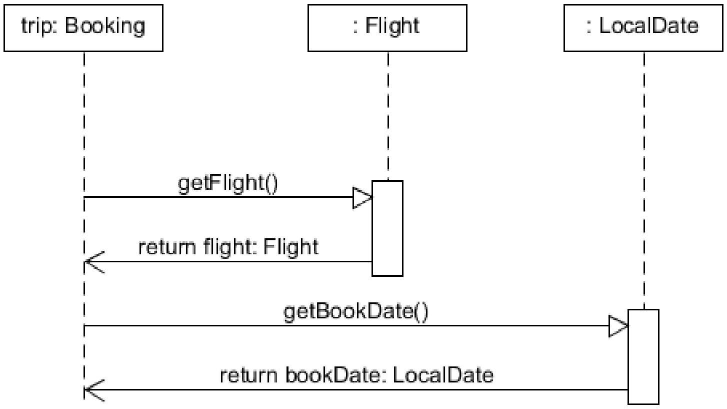
UserController UML Sequence Diagram


uc: UserController
: User
login(String username, String password)
return user: User
addUser(User user)
void
removeUser(String username)
return status: boolean
findUser(String username)
return user: User
userExists(String username)
return status: boolean
getUserList()
return userList: ArrayList<User>
getRentalSystem()
return rentalSystem: RentalSystem<Rental>

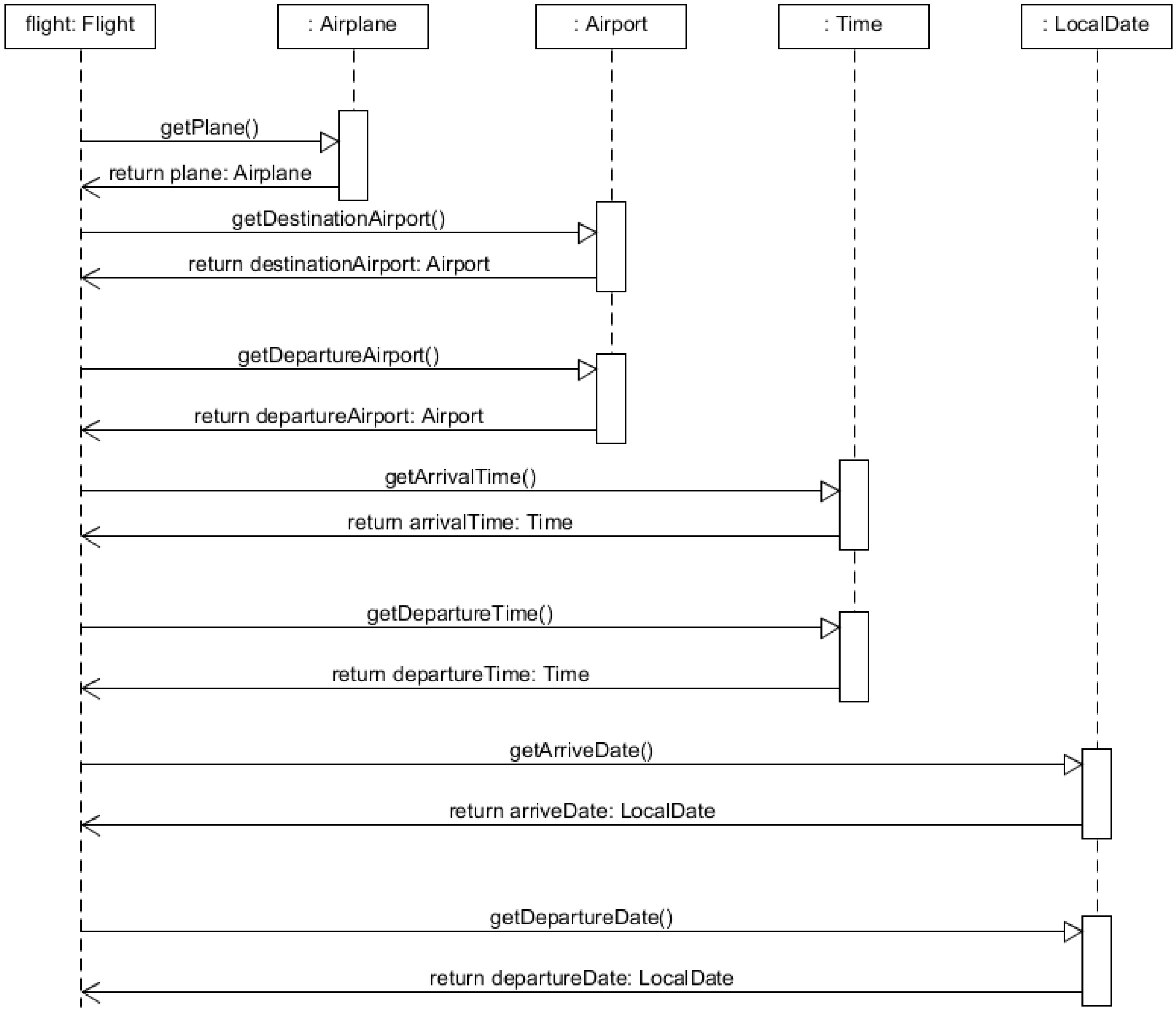
User UML Sequence Diagram



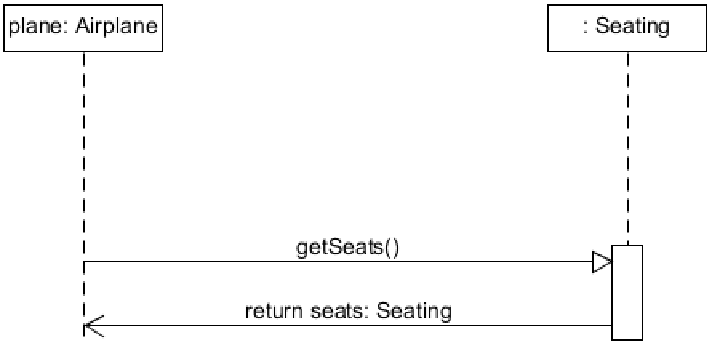
Booking UML Sequence Diagram



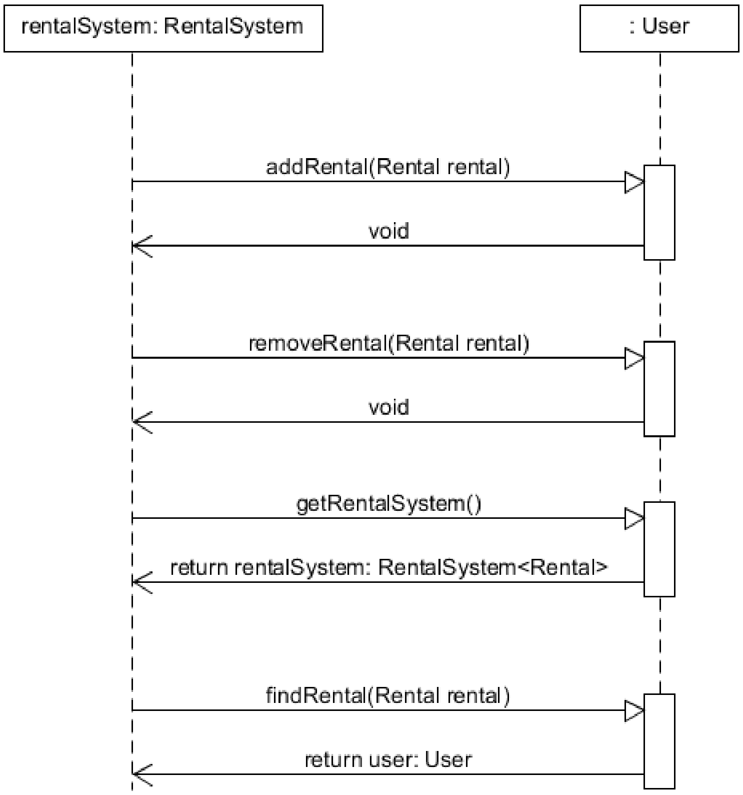
Flight UML Sequence Diagram



Airplane UML Sequence Diagram



Rental System UML Sequence Diagram


rentalSystem: RentalSystem
: User
addRental(Rental rental)
void
removeRental(Rental rental)
void
getRentalSystem()
return rentalSystem: RentalSystem<Rental>
findRental(Rental rental)
return user: User