UCLL

Case Study OSA

Simple Airline

Maria Verdonck, r0298778, 2TX/1

2016-2017

Contents

Short Introduction

This is an analysis of an Airline in its basic functions, which are further described in detail below. In this first definition study, it is assumed people travel without luggage and there is no security check necessary. There are also no flight personnel included in this first analysis besides the pilot and the person checking the boarding passes, the flight attendant. We also assume maintenance like repair and refuel of the planes is done automatically by the airports that our planes use. Our airline only operates out of one airport, so all travelers depart and return out of this one airport. The destination airport of all flights can also be their departure airport if it's a returning flight.

Business Analysis

Roles

- The **customer**: the one buying a plane ticket and making the payment
- The **traveler**: the person flying the plane
- The **payer**: the one making the payment for the booking and receiving the corresponding invoice
- The **clerk** at the desk: the one responsible for booking the tickets, checking the boarding passes, and answering incoming calls, cancelling a booking
- The **manager**: the one who makes flying and work schedules
- The **pilot**: the one who flies the plane
- The **flight attendant**: the one who checks the boarding pass before the traveler can board the plane

Business Analysis (AS IS)

Customer is an individual person booking one or multiple tickets. The customer must give the names, e-mail addresses, ID card numbers and birth dates of all the people who will be flying, as well as their destination and whether they want a return ticket, their chosen departure and return, and whether they would like a first-class ticket or not. There are only two types: first class and coach. Based on destination, whether it is one-way or two way and their destination, the price per ticket is set. The customer also gives the name, credit card information and address of the one paying for the tickets. The customer can book via site or via clerk.

The price of this 'order' is automatically paid with the credit card information connected to the order, and an invoice is sent to the address of the one paying for the order. If the credit card information is incorrect or there are insufficient funds, the booking does not get made. If the booking gets made, it gets a booking number.

This way the one paying for the tickets does not necessarily mean they are one of the people travelling, since it could be for example a travel agent.

Cancellation is possible until 1 week before departure in which you get a refund of half of the price. Cancellation must be done by calling the clerk or doing it via site.

Starting 1 week before departure the boarding passes can be printed out (the are sent to the e-mail addresses given at booking time). These are sent automatically daily based on the bookings.

Every three months the manager makes the flying schedule which contains departure and return times, as well as destinations and pricing. Every month the manager makes the work schedule containing who works the desk every day and who will fly with and be flight attendant on all flights on the flying schedule.

The flight attendant checks whether the traveller has a boarding pass before allowing them on board the plane.

Problem description (TO BE)

Functional requirements

The manager:

- Get an overview of all bookings made by clerk
- Change information on departure and return times, destinations, and prices
- Make a work schedule for each employee every month
- Make the flight schedule every three months

The clerk:

- Make a booking
- Cancel a booking
- Get own work schedule

The customer:

- Make a booking
- Make a cancellation

The traveller:

- Enter plane (by showing boarding pass to flight attendant)

The flight attendant:

- Get overview of all bookings made for a specific plane
- Get own work schedule

Pilot:

- Get own work schedule

Reports to be generated

- Overview of all bookings made per flight
- Work schedules for clerks, flight attendants and pilots

Business events

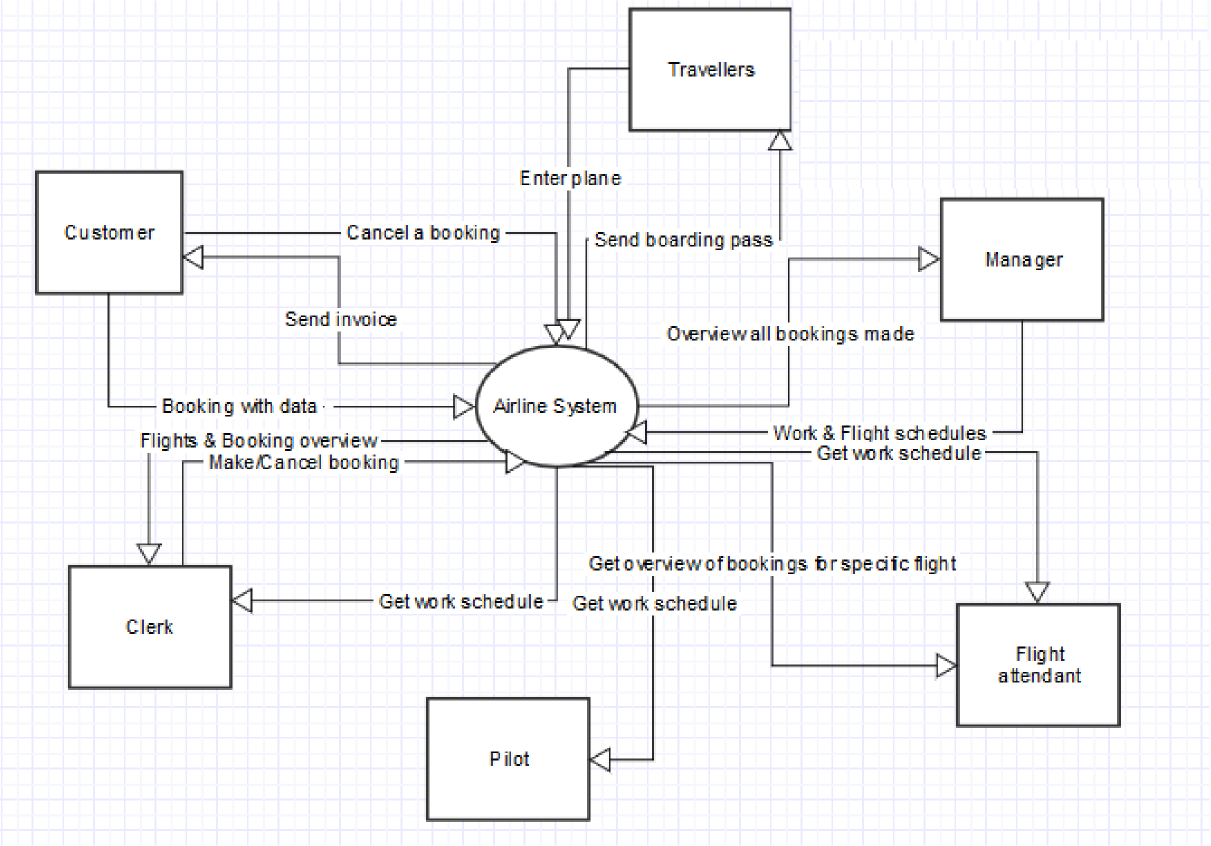
1. CUSTOMER makes a booking
2. CUSTOMER cancels a booking
3. TRAVELLER boards plane

Business use-cases

1. Make a booking
2. Cancel a booking
3. Send invoice
4. Create work schedule
5. Create flight schedule
6. Get overview of all bookings
7. Get overview of bookings for specific flight

8. Send boarding pass
9. Enter plane

Context Diagram



Requirement Analysis

Use Case Descriptions

1. Make a booking

Name	Make a booking
Summary	Customer or Clerk makes a booking
Actors	Customer or Clerk
Precondition	-
Scenario	Customer goes to site and makes a booking OR Customer calls Clerk who makes the booking Data needed: names, e-mail addresses, ID card numbers of all people travelling; departure and return times; destination; name, address and credit card information of the person paying If credit card information is incorrect or there are insufficient funds, booking does not get made (exception 1) Booking gets made, with connected booking number, customer gets booking number. Invoice gets sent (see use case 3)
Exceptions	1: Credit card information is incorrect or there are insufficient funds, booking does not get made, customer gets error message (via clerk or site)
Postconditions	A booking (with all connected data) is made with a connected booking number, event 3 send invoice gets started or customer gets error

2. Cancel a booking

Name	Cancel a booking
Summary	Customer or clerk cancels a booking
Actors	Customer or Clerk
Precondition	Departure time is farther than 1 week away
Scenario	Customer goes to site and does the cancellation OR Customer calls Clerk does the cancellation Data needed: booking number Booking gets cancelled, and half of the booking price gets sent to credit card of the one who paid for the booking If departure time is within 1 week, booking cannot get cancelled (exception 1) If booking number is non-existent, booking cannot get cancelled (exception 2)
Exceptions	1: Departure time is within 1 week; booking cannot get cancelled 2: Booking number is non-existent; booking cannot get cancelled
Postconditions	Booking with relevant booking number is cancelled or customer gets error

3. Send invoice

Name	Send invoice
Summary	There is an invoice sent to the address of the one paying for the booking
Actors	Payer
Precondition	There is a successful booking made (see event 1)
Scenario	After a successful booking is made (event 1) and invoice is automatically sent to the address of the one paying for the booking
Exceptions	-
Postconditions	Invoice is sent

4. Create work schedule

Name	Create a work schedule
Summary	Manager makes work schedule for Clerk, Pilot or Flight attendant
Actors	Manager, Clerk, Pilot, Flight attendant
Precondition	It’s the beginning of the month
Scenario	Every month the manager makes the work schedules for the clerks, pilots, and flight attendants. Each work schedule is a list of shifts which contain a time, location, and duration.
Exceptions	-
Postconditions	Work schedules are made

5. Create flight schedule

Name	Create flight schedule
Summary	The manager makes the flight schedule
Actors	Manager, Pilot, Flight attendant
Precondition	It’s been 3 months since the last flight schedule.
Scenario	Every 3 months the manager makes the flight schedule, containing destinations, departure, and arrival times, whether it’s a returning flight or not, and pricing
Exceptions	-
Postconditions	Flight schedule is created

6. Get overview of all bookings

Name	Get overview of all bookings
Summary	Get an overview of all the bookings made at this airline
Actors	Clerk or Manager
Precondition	-
Scenario	When necessary the Clerk and Manager can ask for an overview of all bookings made
Exceptions	-
Postconditions	Clerk or Manager get an overview of all bookings.

7. Get overview of bookings for specific flight

Name	Get overview of bookings for specific flight
Summary	Get an overview of all the bookings made for a specific flight of this airline
Actors	Flight Attendant
Precondition	-
Scenario	The Flight attendant can ask for an overview of all bookings for a specific flight, so she can check the boarding passes of the travellers boarding.
Exceptions	-
Postconditions	Flight attendant gets an overview of all bookings for a specific flight.

8. Send boarding pass

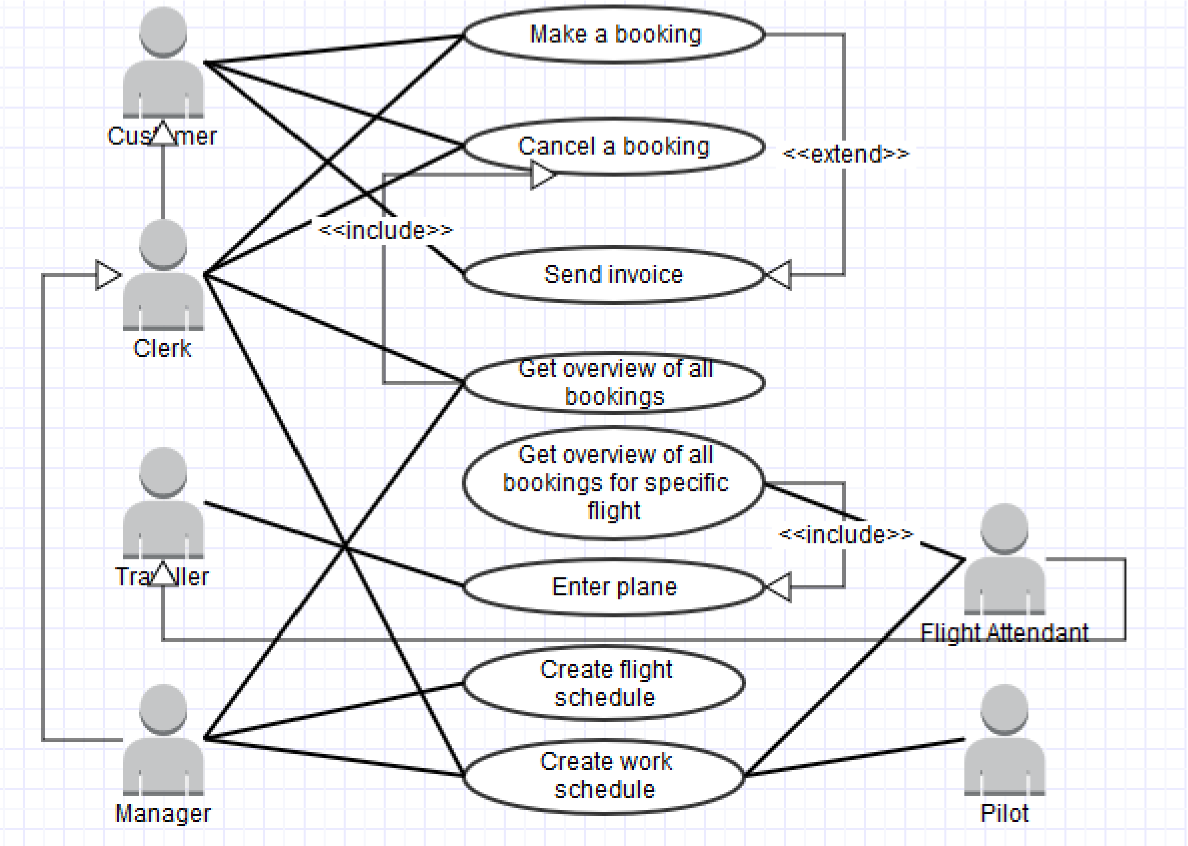
Name	Send boarding pass
Summary	The boarding pass is sent to the e-mail address of the traveller (data in booking)
Actors	Traveller
Precondition	It is 1 week before departure of the flight
Scenario	One week before the departure date of each flight, every traveller who has a booking for this flight gets the boarding pass sent to their e-mail address (data in their booking info). The Airline System checks and sends these daily.
Exceptions	-
Postconditions	Boarding pass gets sent to the e-mail address of the traveller.

9. Enter plane

Name	Enter plane
Summary	The traveller enters the plane after showing boarding pass
Actors	Traveller, Flight attendant
Precondition	Traveller has his boarding pass (see event 8), the Flight attendant has an overview of all bookings made for this flight (see event 7)
Scenario	The traveller shows his boarding pass to the flight attendant who checks the booking overview and the traveller enters the plane. If the traveller doesn't have a or the correct boarding pass, they cannot enter the plan (exception 1) If the traveller is not on the overview of all the bookings made for this flight, they cannot enter the plane (exception 2) Flight adds this traveller to its list of travellers on flight
Exceptions	1: Traveller does not have a or the correct boarding pass 2: Traveller is not on the overview of all bookings made for this flight

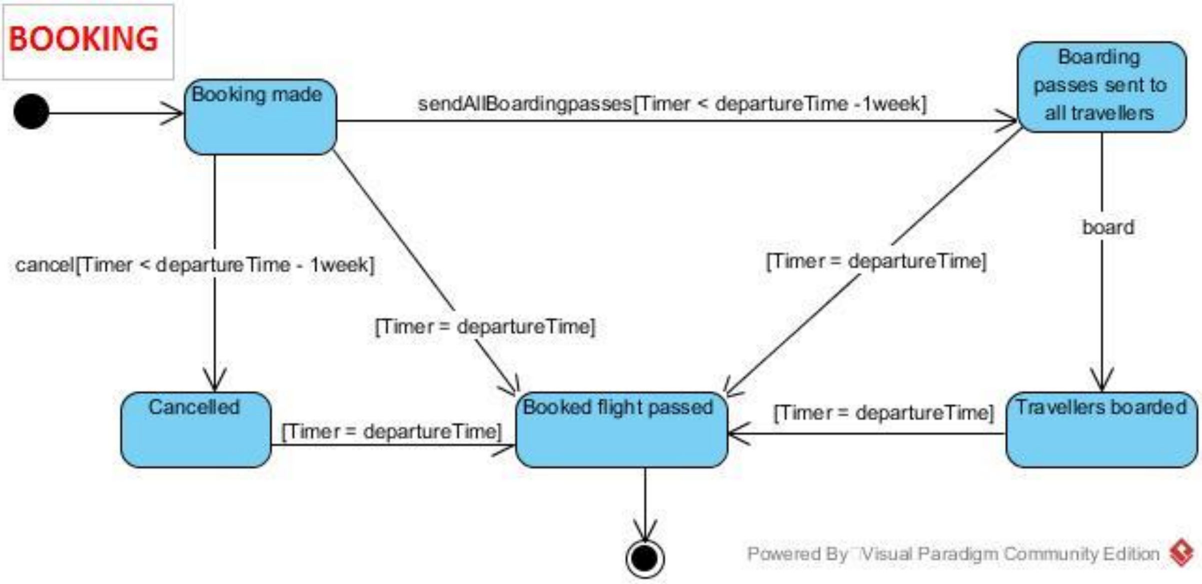
Postconditions	Traveller enters plane, unless exception happens, and traveller is added to list of travellers on the flight

Use Case Diagram



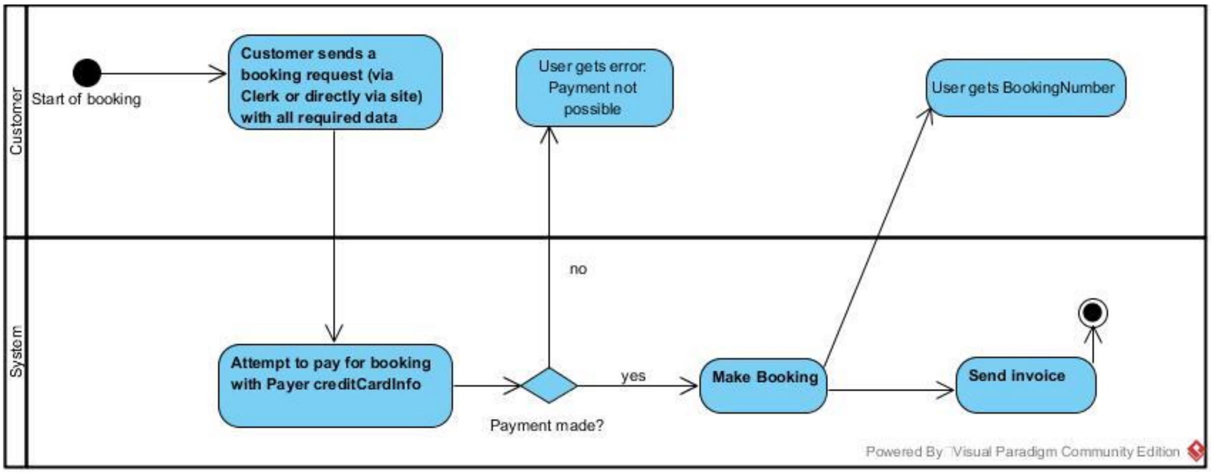
Dynamic Layer

State Diagram

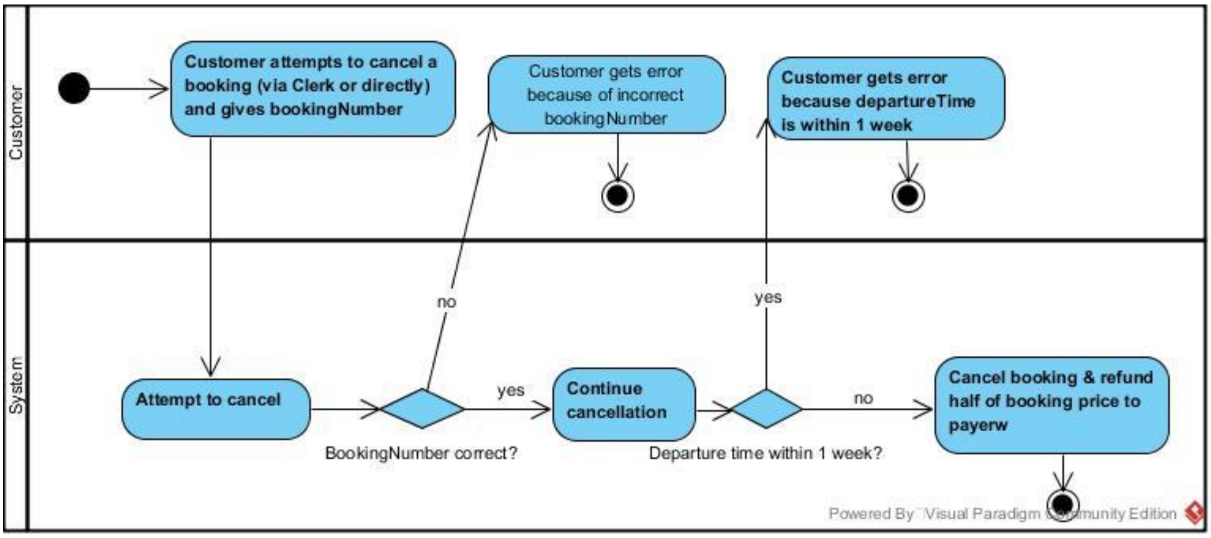

BOOKING
Booking made
sendAllBoardingpasses[Timer < departureTime -1week]
Boarding passes sent to all travellers
board
cancel[Timer < departureTime - 1week]
[Timer = departureTime]
[Timer = departureTime]
Cancelled
[Timer = departureTime]
Booked flight passed
[Timer = departureTime]
Travellers boarded
Powered By Visual Paradigm Community Edition

Activity Diagrams

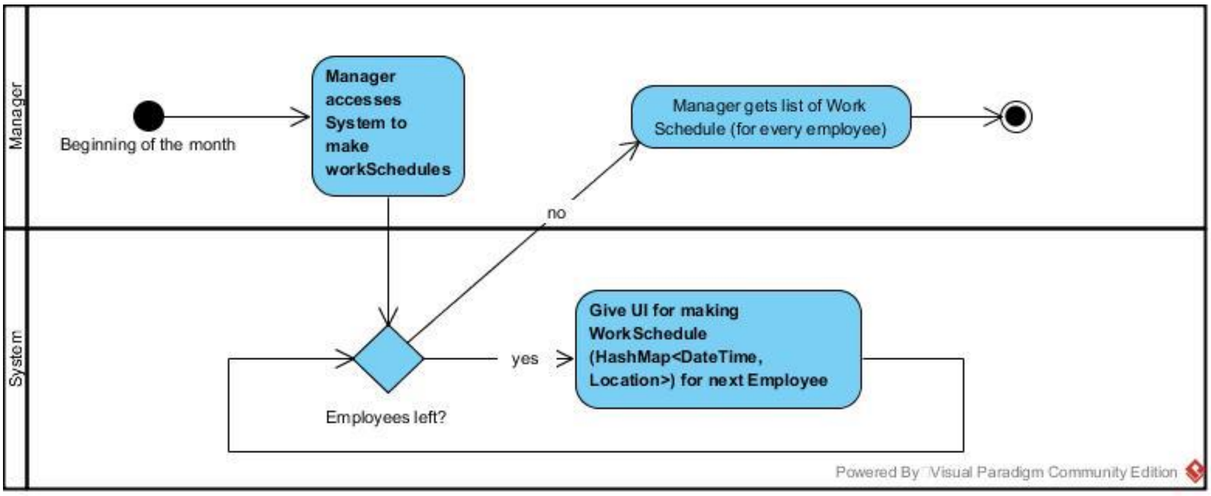
1&3. Make a Booking


Customer
System
Start of booking
Customer sends a booking request (via Clerk or directly via site) with all required data
User gets error: Payment not possible
User gets BookingNumber
no
Attempt to pay for booking with Payer creditCardInfo
yes
Make Booking
Send invoice
Payment made?
Powered By Visual Paradigm Community Edition

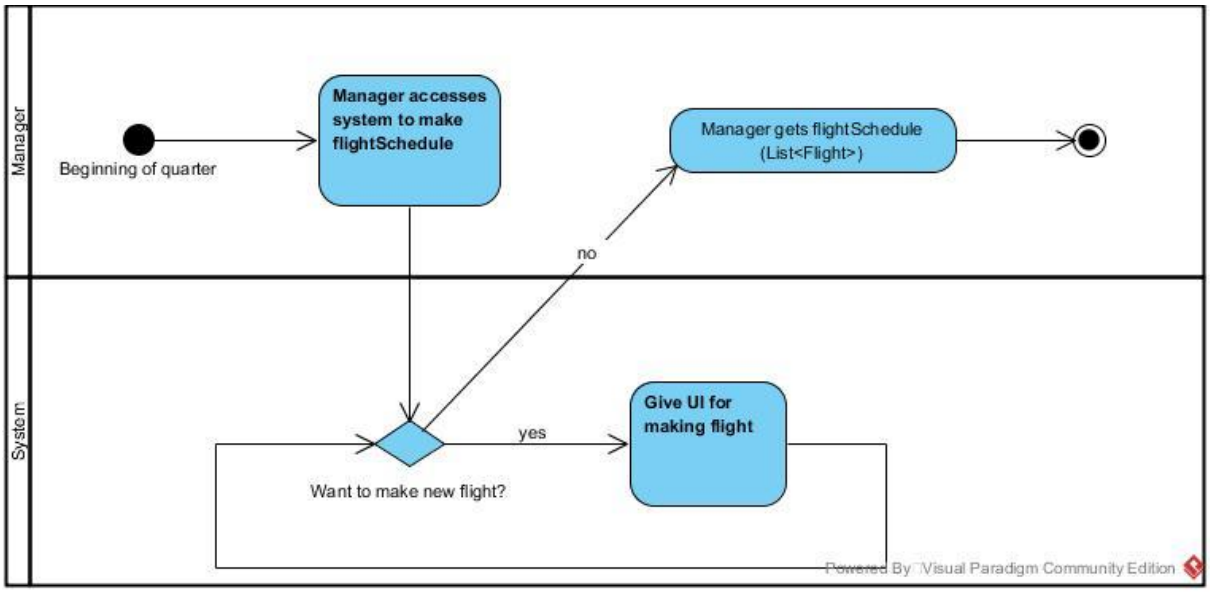
2. Cancel a Booking


Customer
System
Customer attempts to cancel a booking (via Clerk or directly) and gives bookingNumber
Customer gets error because of incorrect bookingNumber
Customer gets error because departureTime is within 1 week
no
yes
Attempt to cancel
yes
Continue cancellation
no
Cancel booking & refund half of booking price to payerw
BookingNumber correct?
Departure time within 1 week?
Powered By Visual Paradigm Community Edition

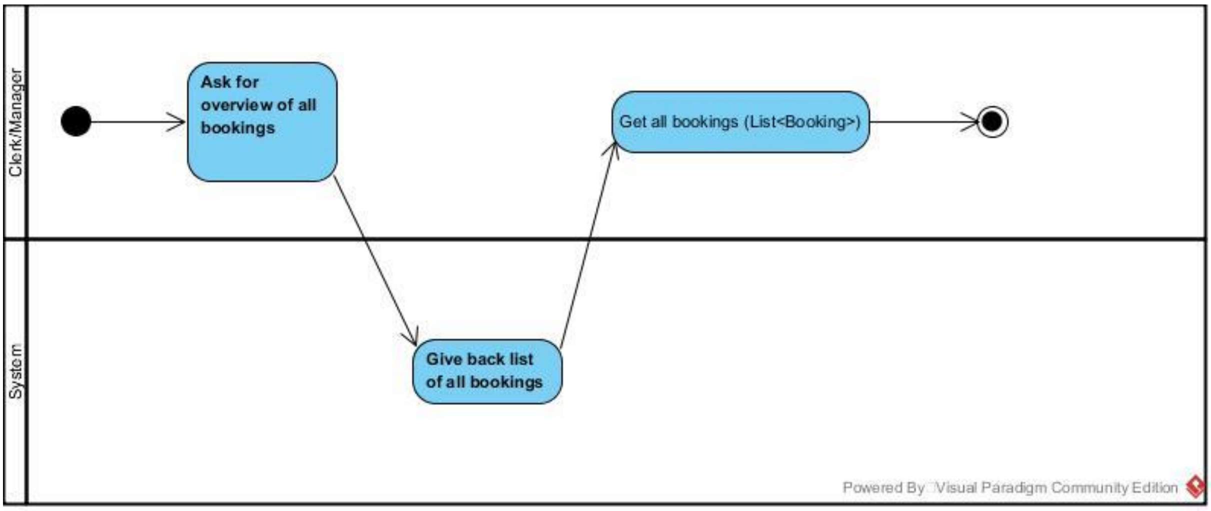
4. Create work schedule


Manager
System
Beginning of the month
Manager accesses System to make workSchedules
Manager gets list of Work Schedule (for every employee)
no
yes
Give UI for making WorkSchedule (HashMap<DateTime, Location>) for next Employee
Employees left?
Powered By Visual Paradigm Community Edition

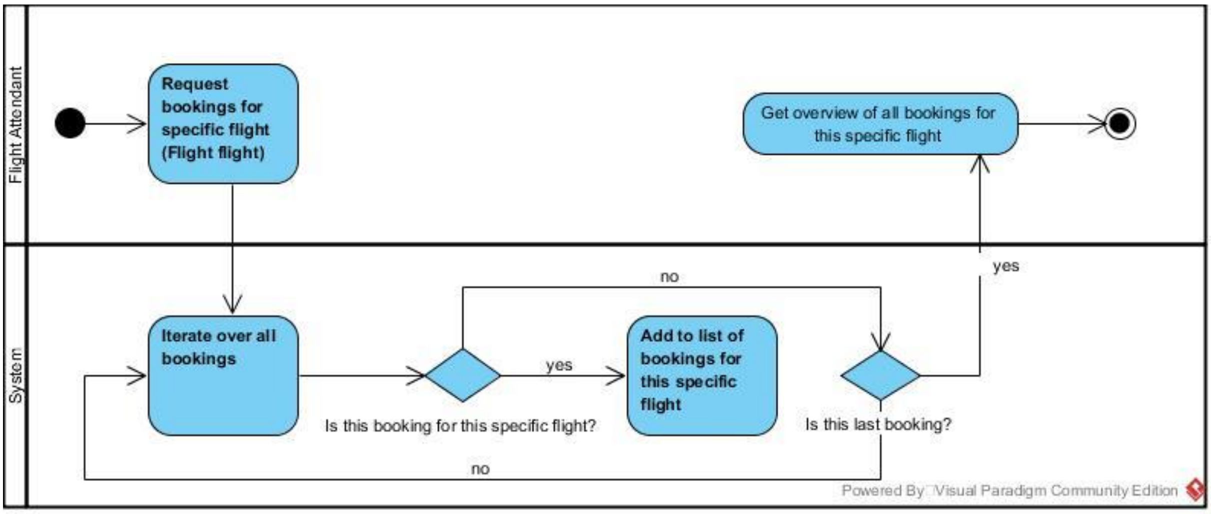
5. Create flight schedule


Manager
System
Beginning of quarter
Manager accesses system to make flightSchedule
Manager gets flightSchedule (List<Flight>)
no
yes
Want to make new flight?
Give UI for making flight
Powered By Visual Paradigm Community Edition

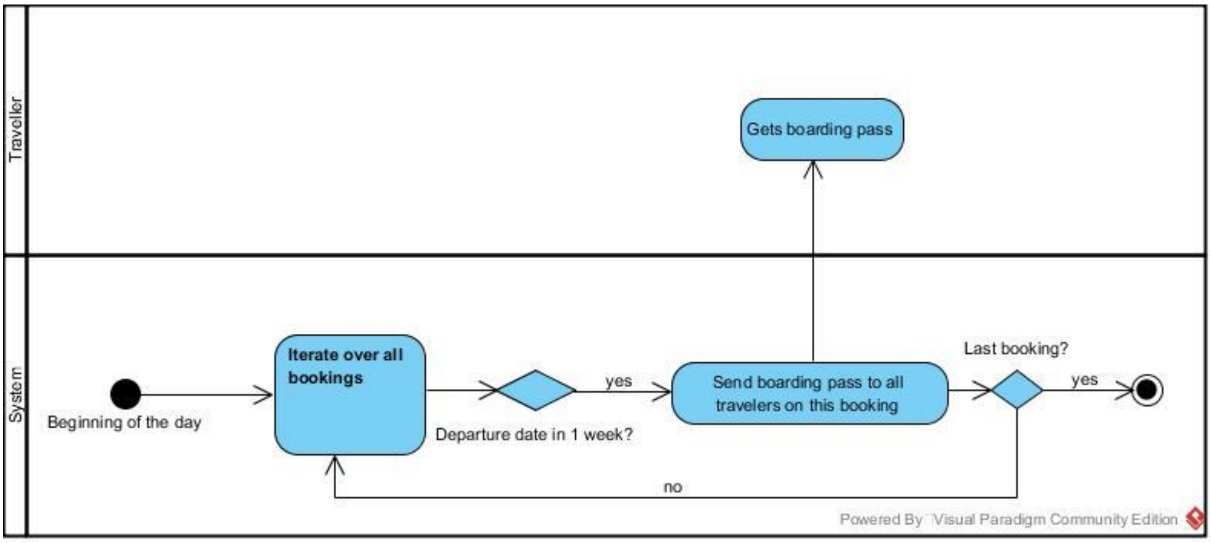
6. Get overview of all bookings


Clerk/Manager
System
Ask for overview of all bookings
Get all bookings (List<Booking>)
Give back list of all bookings
Powered By Visual Paradigm Community Edition

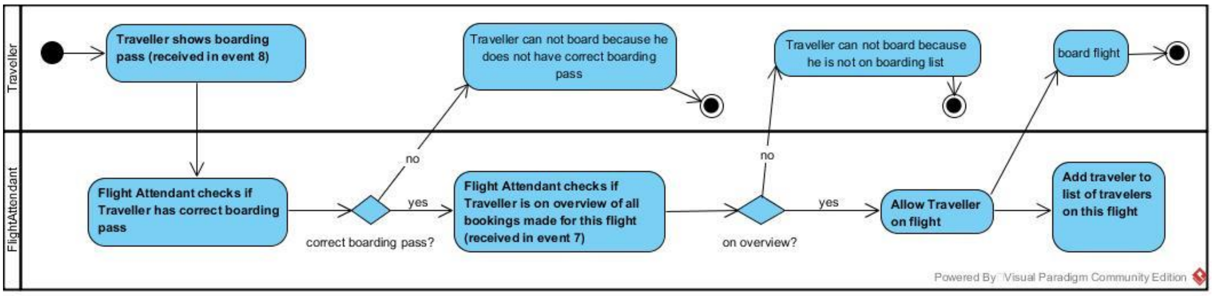
7. Get bookings specific flight


Flight Attendant
System
Request bookings for specific flight (Flight flight)
Get overview of all bookings for this specific flight
yes
no
Iterate over all bookings
yes
Add to list of bookings for this specific flight
Is this booking for this specific flight?
Is this last booking?
no
Powered By Visual Paradigm Community Edition

8. Send boarding pass


Traveller
System
Gets boarding pass
Beginning of the day
Iterate over all bookings
yes
Departure date in 1 week?
Send boarding pass to all travelers on this booking
Last booking?
yes
no
Powered By Visual Paradigm Community Edition

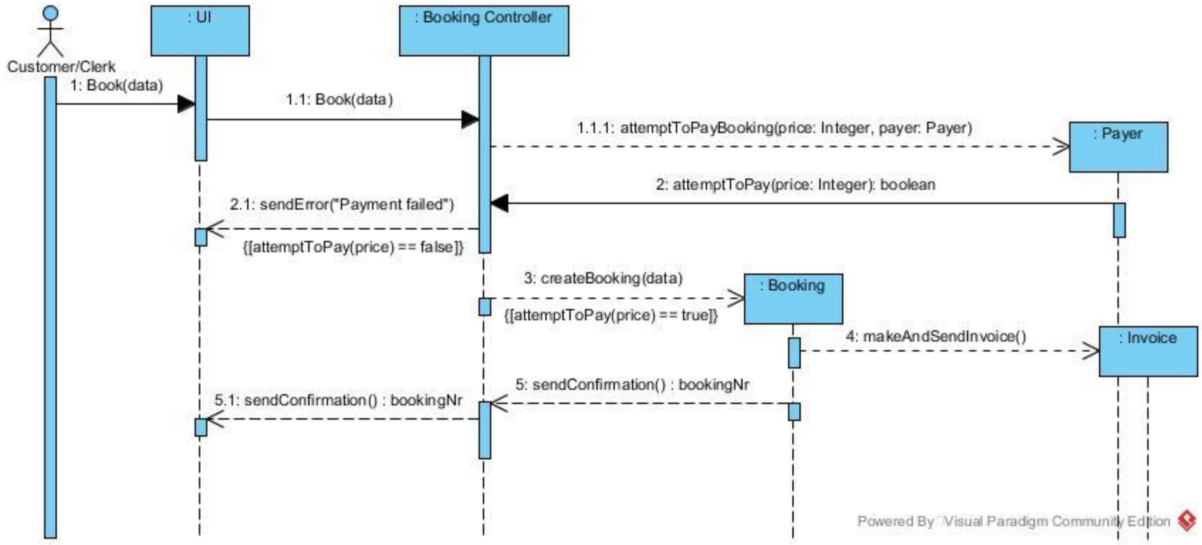
9. Enter plane


Traveller
FlightAttendant
Traveller shows boarding pass (received in event 8)
Traveller can not board because he does not have correct boarding pass
Traveller can not board because he is not on boarding list
board flight
Flight Attendant checks if Traveller has correct boarding pass
no
yes
correct boarding pass?
Flight Attendant checks if Traveller is on overview of all bookings made for this flight (received in event 7)
no
yes
on overview?
Allow Traveller on flight
Add traveler to list of travelers on this flight
Powered By Visual Paradigm Community Edition

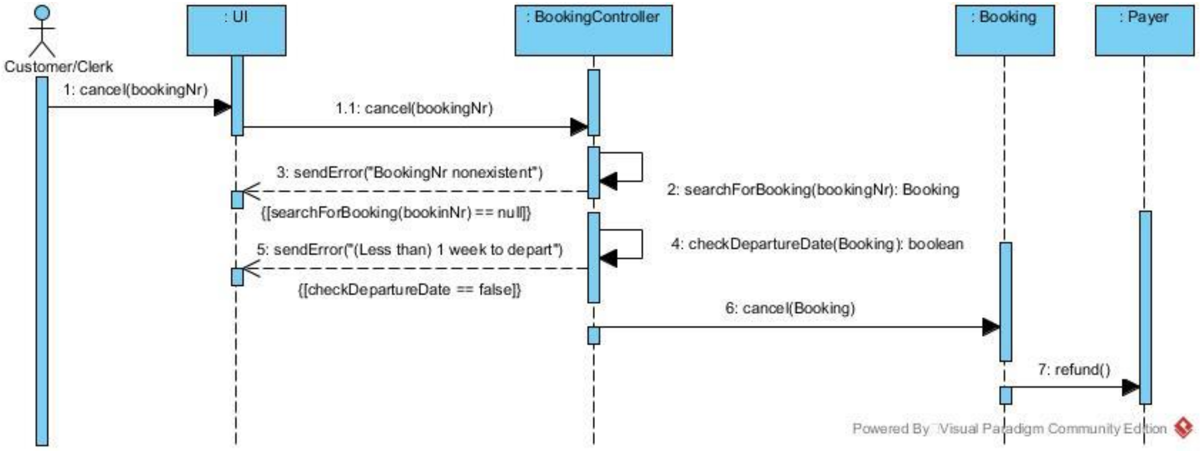
Application Layer

Sequence Diagrams

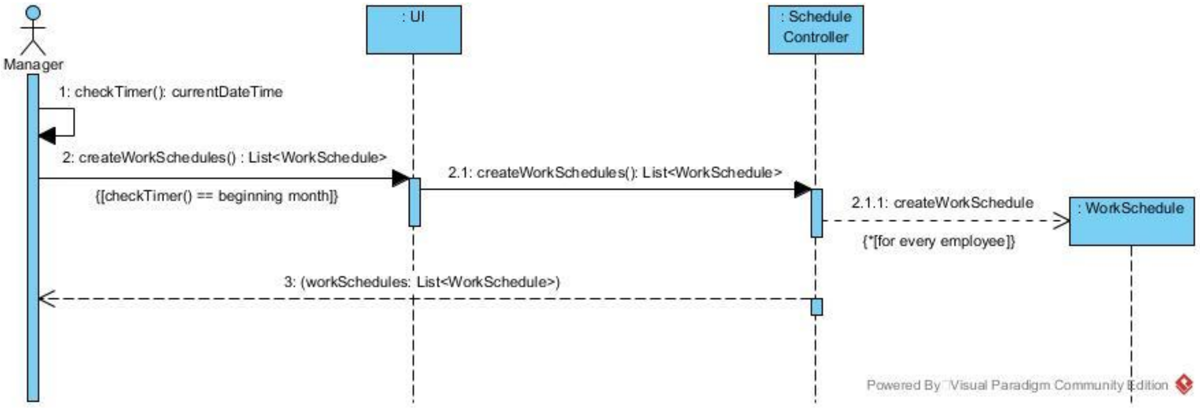
1&3. Make a Booking


Customer/Clerk
: UI
: Booking Controller
: Payer
: Booking
: Invoice
1: Book(data)
1.1: Book(data)
1.1.1: attemptToPayBooking(price: Integer, payer: Payer)
2: attemptToPay(price: Integer): boolean
2.1: sendError("Payment failed")
{[attemptToPay(price) == false]}
3: createBooking(data)
{[attemptToPay(price) == true]}
4: makeAndSendInvoice()
5: sendConfirmation() : bookingNr
5.1: sendConfirmation() : bookingNr
Powered By Visual Paradigm Community Edition

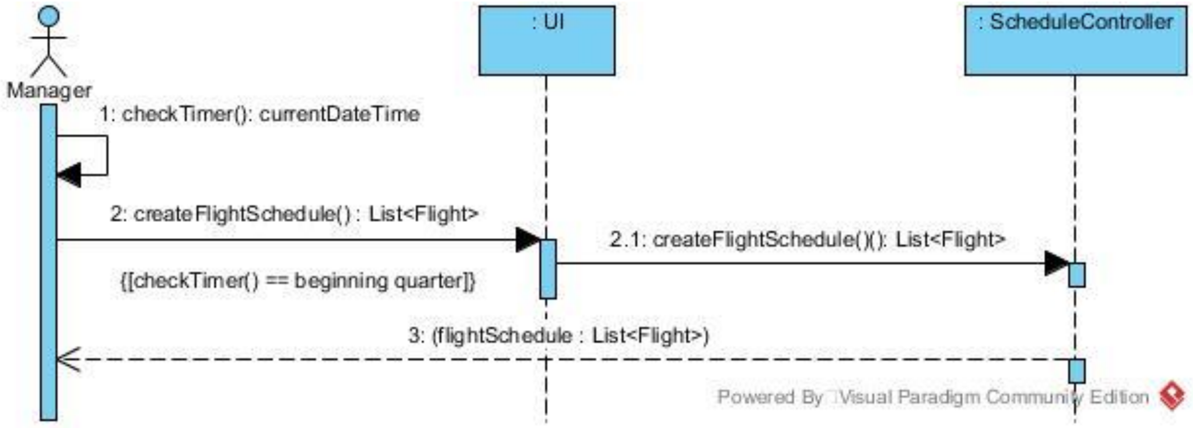
2. Cancel a Booking


Customer/Clerk
: UI
: BookingController
: Booking
: Payer
1: cancel(bookingNr)
1.1: cancel(bookingNr)
3: sendError("BookingNr nonexistent")
{[searchForBooking(bookinNr) == null]}
2: searchForBooking(bookingNr): Booking
5: sendError("(Less than) 1 week to depart")
{[checkDepartureDate == false]}
4: checkDepartureDate(Booking): boolean
6: cancel(Booking)
7: refund()
Powered By Visual Paradigm Community Edition

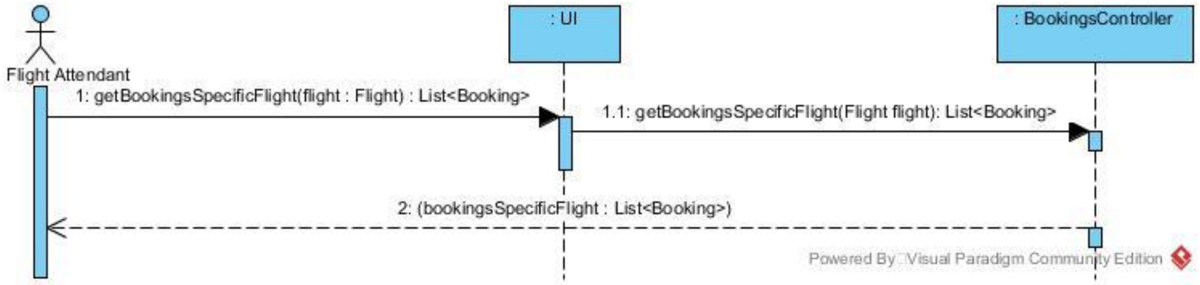
4. Create work schedule


Manager
: UI
: Schedule Controller
1: checkTimer(): currentDateTime
2: createWorkSchedules() : List<WorkSchedule>
{[checkTimer() == beginning month]}
2.1: createWorkSchedules(): List<WorkSchedule>
2.1.1: createWorkSchedule
{*[for every employee]}
: WorkSchedule
3: (workSchedules: List<WorkSchedule>)
Powered By Visual Paradigm Community Edition

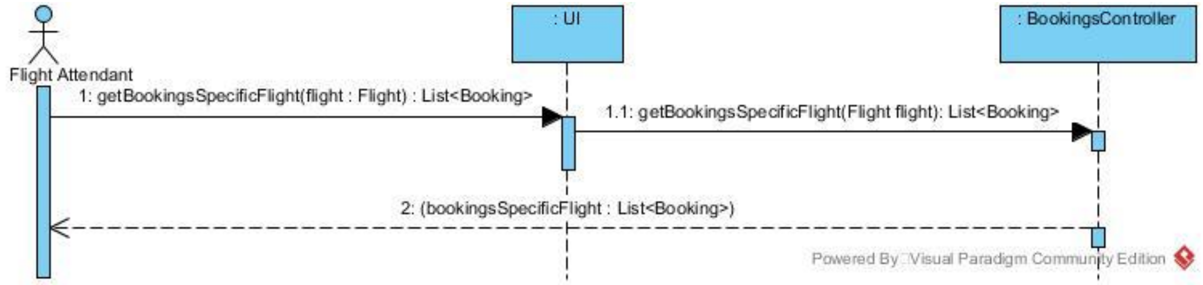
5. Create flight schedule


Manager
: UI
: ScheduleController
1: checkTimer(): currentDateTime
2: createFlightSchedule() : List<Flight>
{[checkTimer() == beginning quarter]}
2.1: createFlightSchedule()(): List<Flight>
3: (flightSchedule : List<Flight>)
Powered By Visual Paradigm Community Edition

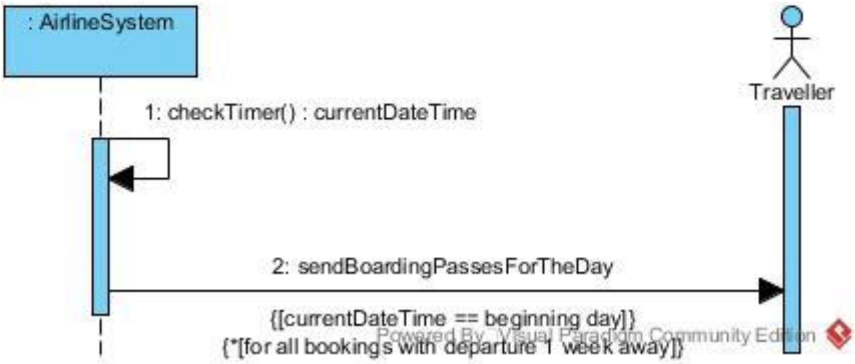
6. Get overview of all bookings


Flight Attendant
: UI
: BookingsController
1: getBookingsSpecificFlight(flight : Flight) : List<Booking>
1.1: getBookingsSpecificFlight(Flight flight): List<Booking>
2: (bookingsSpecificFlight : List<Booking>)
Powered By Visual Paradigm Community Edition

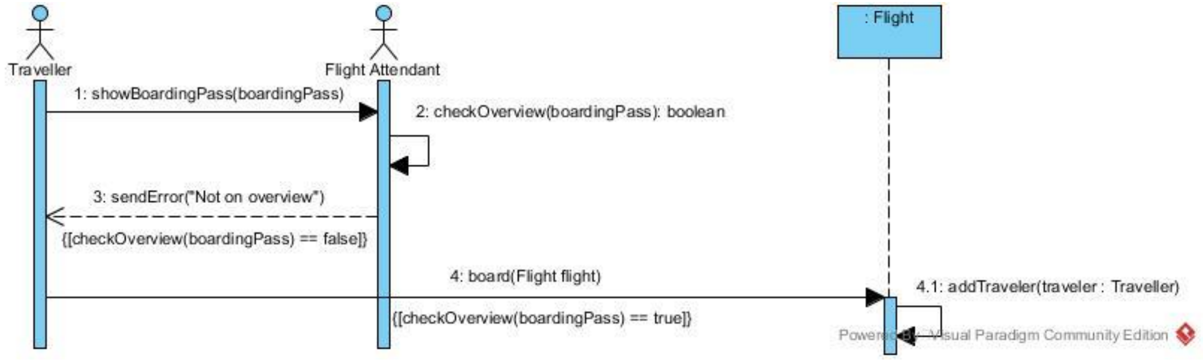
7. Get bookings specific flight


Flight Attendant
: UI
: BookingsController
1: getBookingsSpecificFlight(flight : Flight) : List<Booking>
1.1: getBookingsSpecificFlight(Flight flight): List<Booking>
2: (bookingsSpecificFlight : List<Booking>)
Powered By Visual Paradigm Community Edition

8. Send boarding pass

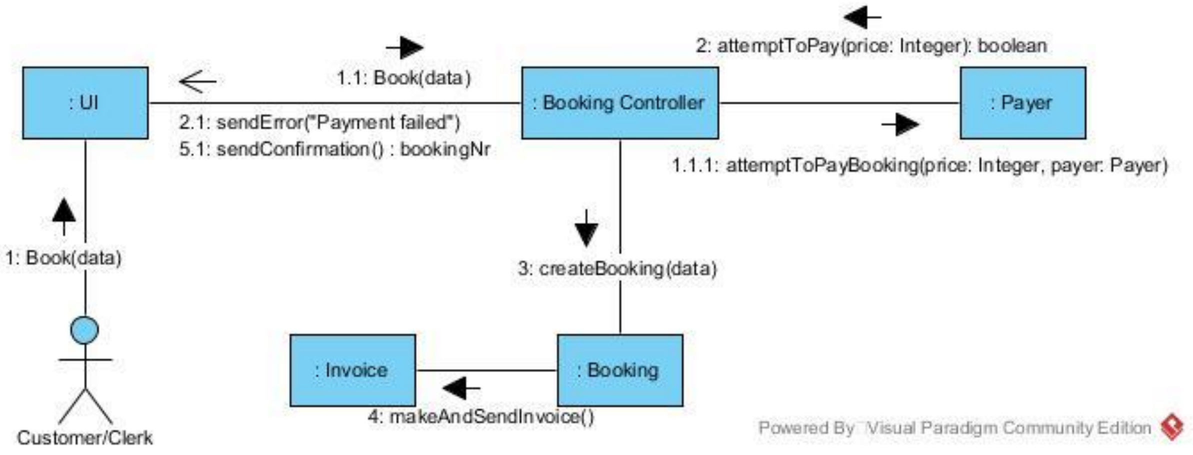

: AirlineSystem
Traveller
1: checkTimer() : currentDateTime
2: sendBoardingPassesForTheDay
{[currentDateTime == beginning day]}
{*[for all bookings with departure 1 week away]}
Powered By Visual Paradigm Community Edition

9. Enter plane

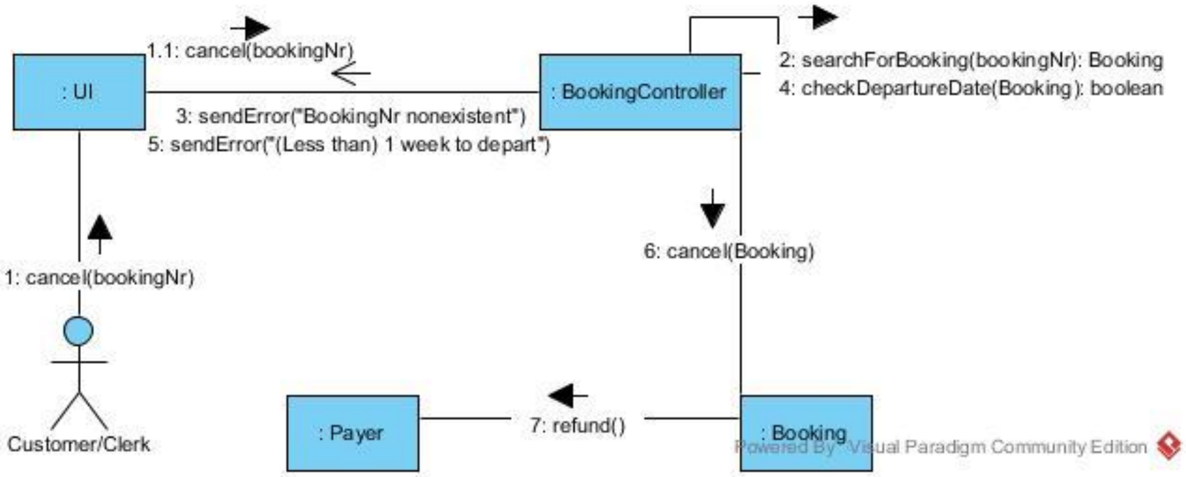

Traveller
Flight Attendant
: Flight
1: showBoardingPass(boardingPass)
2: checkOverview(boardingPass): boolean
3: sendError("Not on overview")
{[checkOverview(boardingPass) == false]}
4: board(Flight flight)
{[checkOverview(boardingPass) == true]}
4.1: addTraveler(traveler : Traveller)
Powered By Visual Paradigm Community Edition

Collaboration Diagrams

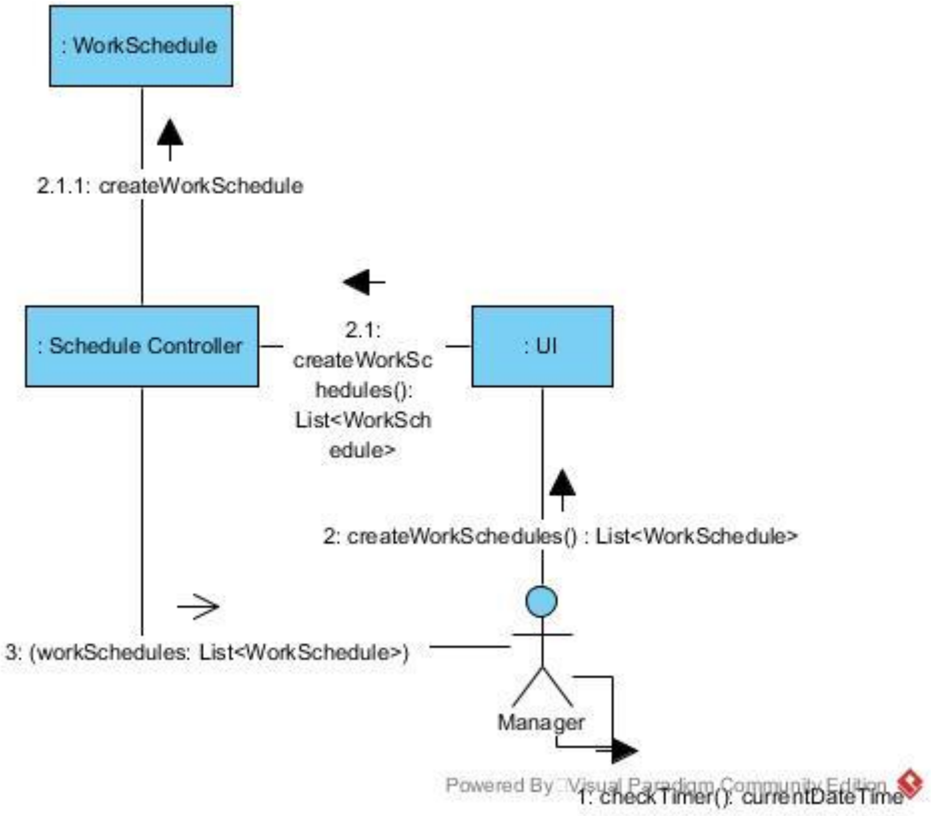
1&3. Make a Booking


: UI
: Booking Controller
: Payer
: Invoice
: Booking
Customer/Clerk
1: Book(data)
1.1: Book(data)
2.1: sendError("Payment failed")
5.1: sendConfirmation() : bookingNr
2: attemptToPay(price: Integer): boolean
1.1.1: attemptToPayBooking(price: Integer, payer: Payer)
3: createBooking(data)
4: makeAndSendInvoice()
Powered By Visual Paradigm Community Edition

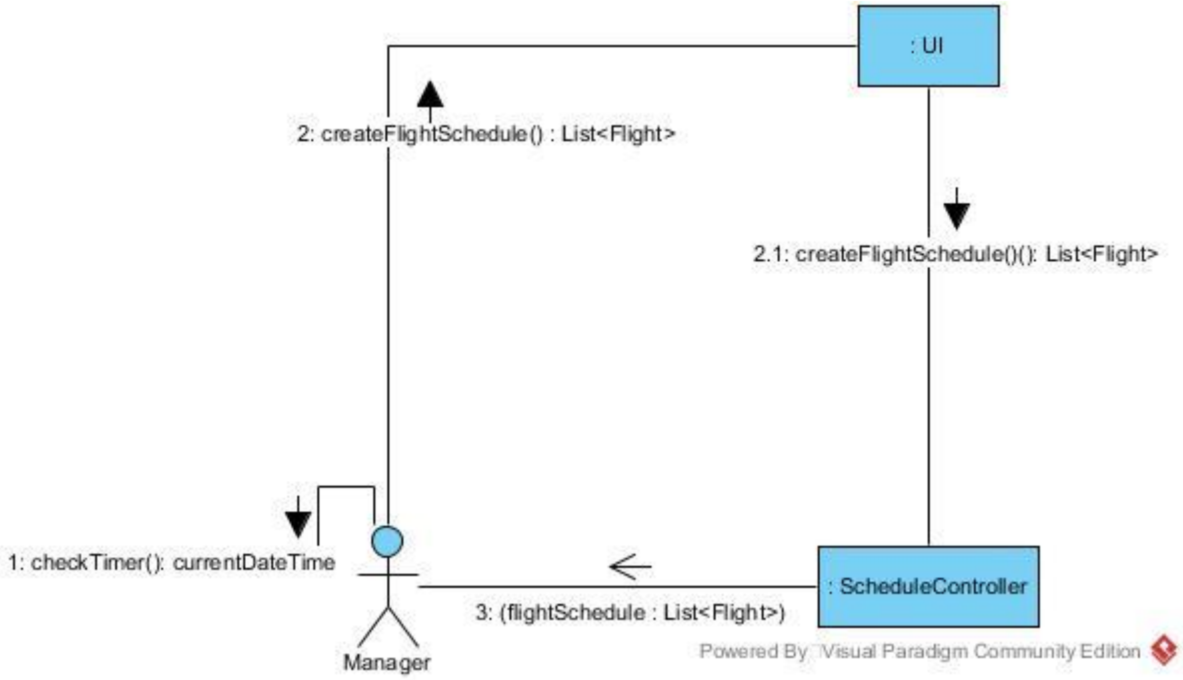
2. Cancel a Booking


: UI
1.1: cancel(bookingNr)
3: sendError("BookingNr nonexistent")
5: sendError("(Less than) 1 week to depart")
: BookingController
2: searchForBooking(bookingNr): Booking
4: checkDepartureDate(Booking): boolean
1: cancel(bookingNr)
6: cancel(Booking)
Customer/Clerk
: Payer
7: refund()
: Booking
Powered By: Visual Paradigm Community Edition

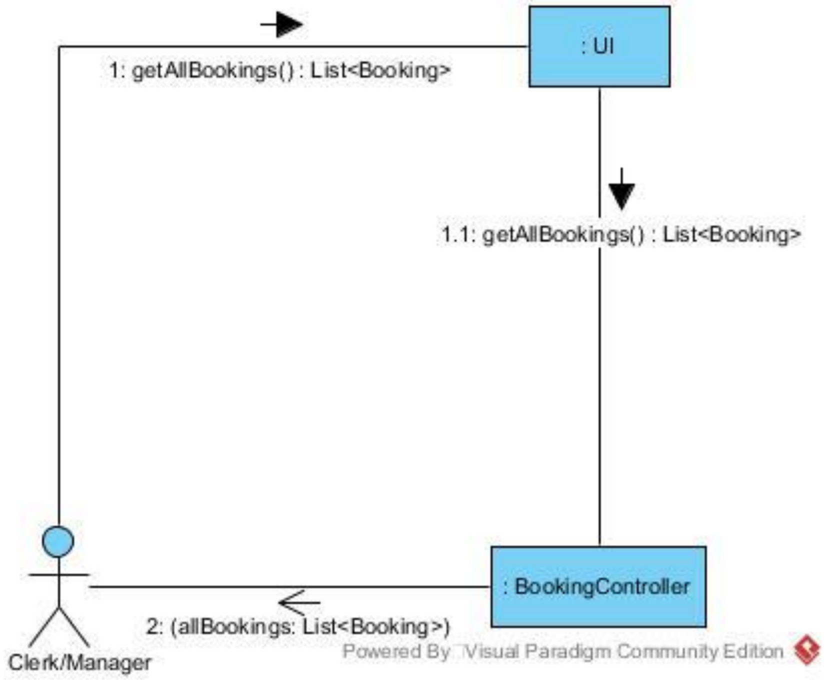
4. Create work schedule


: WorkSchedule
2.1.1: createWorkSchedule
: Schedule Controller
2.1: createWorkSchedules(): List<WorkSchedule>
: UI
2: createWorkSchedules() : List<WorkSchedule>
3: (workSchedules: List<WorkSchedule>)
Manager
1: checkTimer(): currentDateTime
Powered By: Visual Paradigm Community Edition

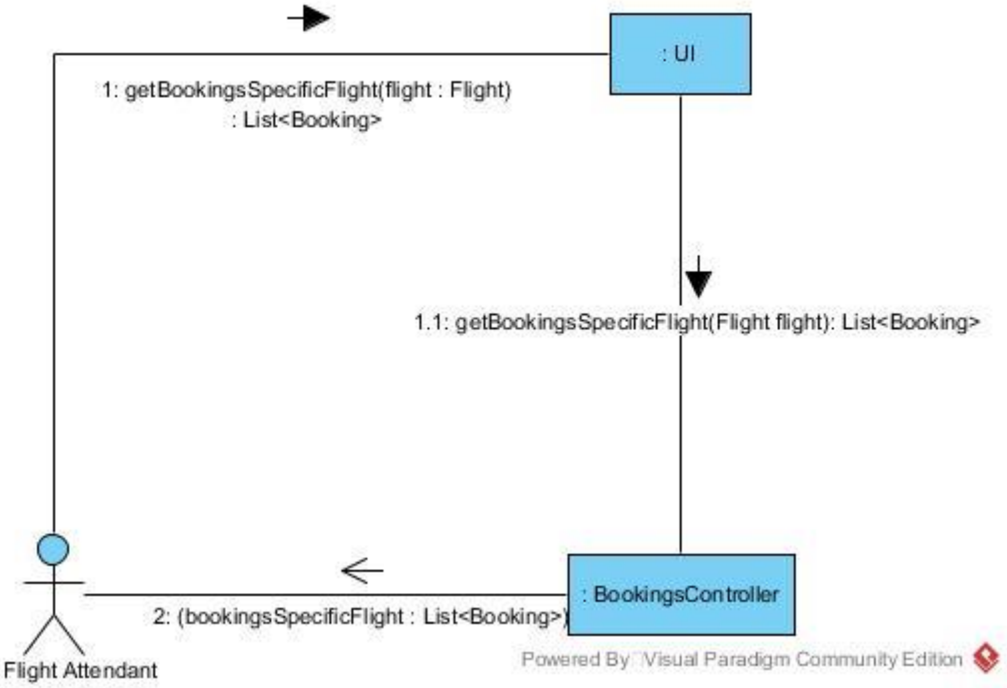
5. Create flight schedule


: UI
2: createFlightSchedule() : List<Flight>
2.1: createFlightSchedule()(): List<Flight>
1: checkTimer(): currentDateTime
: ScheduleController
3: (flightSchedule : List<Flight>)
Manager
Powered By Visual Paradigm Community Edition

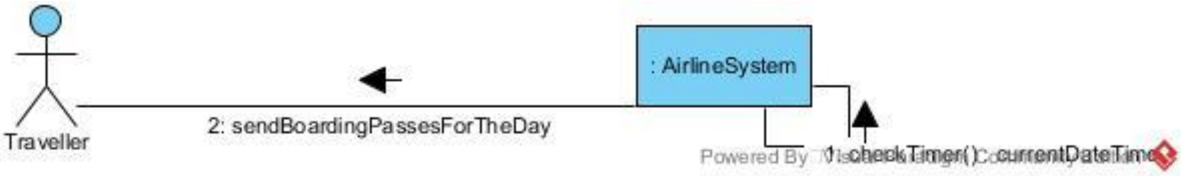
6. Get overview of all bookings


: UI
1: getAllBookings() : List<Booking>
1.1: getAllBookings() : List<Booking>
: BookingController
2: (allBookings: List<Booking>)
Clerk/Manager
Powered By Visual Paradigm Community Edition

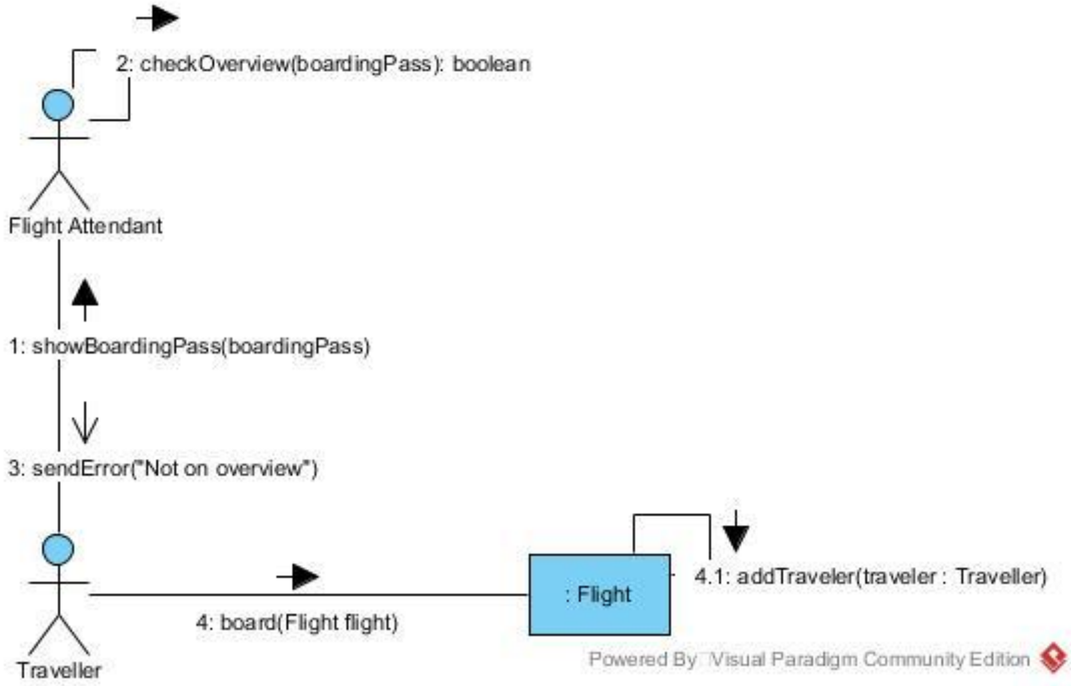
7. Get bookings specific flight


: UI
1: getBookingsSpecificFlight(flight : Flight)
: List<Booking>
1.1: getBookingsSpecificFlight(Flight flight): List<Booking>
: BookingsController
2: (bookingsSpecificFlight : List<Booking>)
Flight Attendant
Powered By Visual Paradigm Community Edition

8. Send boarding pass


: AirlineSystem
2: sendBoardingPassesForTheDay
Traveller
1: checkTimer(): currentDateTime
Powered By Visual Paradigm Community Edition

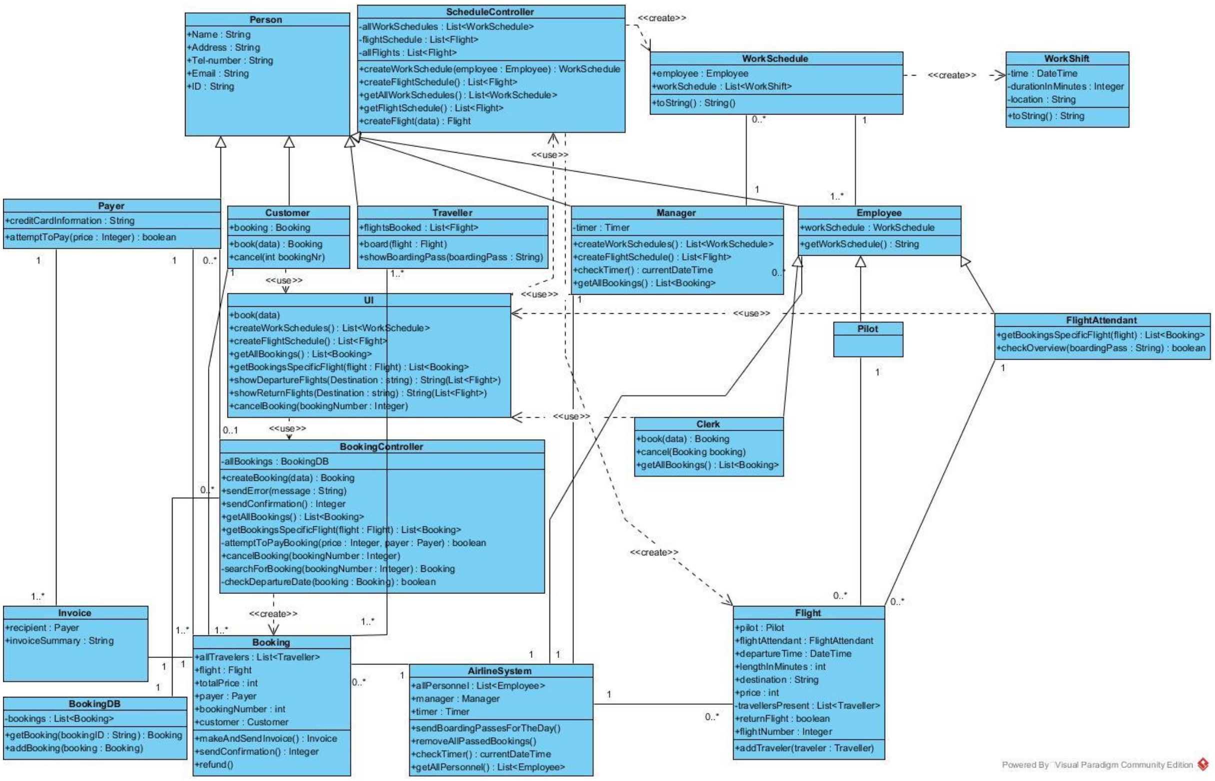
9. Enter plane


2: checkOverview(boardingPass): boolean
Flight Attendant
1: showBoardingPass(boardingPass)
3: sendError("Not on overview")
Traveller
4: board(Flight flight)
: Flight
4.1: addTraveler(traveler : Traveller)
Powered By Visual Paradigm Community Edition

Class Diagram

See ‘Class Diagram.jpg’

NOTE: I kept updating the class diagram as I made changes/additions, so I don’t have the older versions of the class diagram. But it they can be found in the submissions of the sub-assignments if they’re needed for reference.


Person
+Name : String
+Address : String
+Tel-number : String
+Email : String
+ID : String
ScheduleController
-allWorkSchedules : List<WorkSchedule>
-flightSchedule : List<Flight>
-allFlights : List<Flight>
+createWorkSchedule(employee : Employee) : WorkSchedule
+createFlightSchedule() : List<Flight>
+getAllWorkSchedules() : List<WorkSchedule>
+getFlightSchedule() : List<Flight>
+createFlight(data) : Flight
<<create>>
WorkSchedule
+employee : Employee
+workSchedule : List<WorkShift>
+toString() : String()
<<create>>
WorkShift
-time : DateTime
-durationInMinutes : Integer
-location : String
+toString() : String
0..*
1
<<use>>
1
1..*
Payer
+creditCardInformation : String
+attemptToPay(price : Integer) : boolean
Customer
+booking : Booking
+book(data) : Booking
+cancel(int bookingNr)
Traveller
+flightsBooked : List<Flight>
+board(flight : Flight)
+showBoardingPass(boardingPass : String)
Manager
-timer : Timer
+createWorkSchedules() : List<WorkSchedule>
+createFlightSchedule() : List<Flight>
+checkTimer() : currentDateTime
+getAllBookings() : List<Booking>
Employee
+workSchedule : WorkSchedule
+getWorkSchedule() : String
1
1
0..*
1
<<use>>
1..*
<<use>>
0..*
1
UI
+book(data)
+createWorkSchedules() : List<WorkSchedule>
+createFlightSchedule() : List<Flight>
+getAllBookings() : List<Booking>
+getBookingsSpecificFlight(flight : Flight) : List<Booking>
+showDepartureFlights(Destination : string) : String(List<Flight>)
+showReturnFlights(Destination : string) : String(List<Flight>)
+cancelBooking(bookingNumber : Integer)
<<use>>
Pilot
FlightAttendant
+getBookingsSpecificFlight(flight) : List<Booking>
+checkOverview(boardingPass : String) : boolean
1
1
<<use>>
Clerk
+book(data) : Booking
+cancel(Booking booking)
+getAllBookings() : List<Booking>
0..1
<<use>>
BookingController
-allBookings : BookingDB
+createBooking(data) : Booking
+sendError(message : String)
+sendConfirmation() : Integer
+getAllBookings() : List<Booking>
+getBookingsSpecificFlight(flight : Flight) : List<Booking>
-attemptToPayBooking(price : Integer, payer : Payer) : boolean
+cancelBooking(bookingNumber : Integer)
-searchForBooking(bookingNumber : Integer) : Booking
-checkDepartureDate(booking : Booking) : boolean
0..*
<<create>>
1..*
1..*
1..*
1..*
0..*
0..*
Invoice
+recipient : Payer
+invoiceSummary : String
1
1
1
BookingDB
-bookings : List<Booking>
+getBooking(bookingID : String) : Booking
+addBooking(booking : Booking)
Booking
+allTravelers : List<Traveller>
+flight : Flight
+totalPrice : int
+payer : Payer
+bookingNumber : int
+customer : Customer
+makeAndSendInvoice() : Invoice
+sendConfirmation() : Integer
+refund()
0..*
1
1
1
AirlineSystem
+allPersonnel : List<Employee>
+manager : Manager
+timer : Timer
+sendBoardingPassesForTheDay()
+removeAllPassedBookings()
+checkTimer() : currentDateTime
+getAllPersonnel() : List<Employee>
1
0..*
Flight
+pilot : Pilot
+flightAttendant : FlightAttendant
+departureTime : DateTime
+lengthInMinutes : int
+destination : String
+price : int
-travellersPresent : List<Traveller>
+returnFlight : boolean
+flightNumber : Integer
+addTraveler(traveler : Traveller)
Powered By Visual Paradigm Community Edition

Technical Design

Useful design patterns

State Pattern

For: state of a Booking

See ‘State Diagram – Booking’

MVC

- Controller: BookingController and ScheduleController
- View: UI
- Model (rest)

See: Class Diagram

Singleton

For: BookingDB (so you can only access one at a time)
This way you can only access one object, the one DB object, so there are no concurrency problems.

Also useful possibly for the two Controllers, but less necessary.

Factory

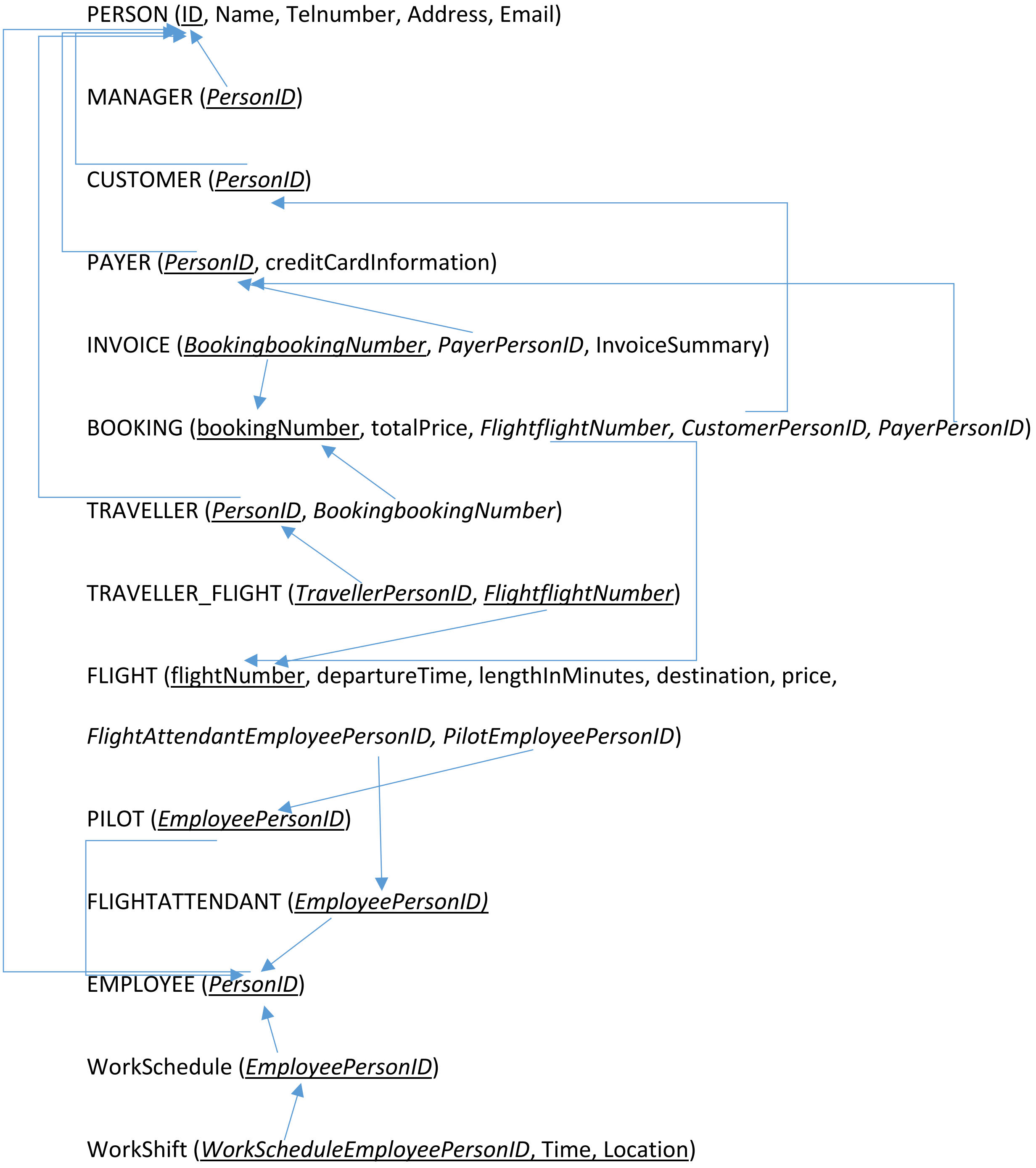
For: Person & Employee

The controllers can also be seen as simple factories, since schedules and bookings are created there.

Observer

- Over the timer object (as the Observeable) of the AirlineSystem, so that when the timer changes (let's say it changes daily), the corresponding tasks are done by the AirlineSystem (and other possible future observers).
- Over everything that wants to send an Error message to the one using the system so this message is displayed.

Relational data model



PERSON (ID, Name, Telnumber, Address, Email)

MANAGER (*PersonID*)

CUSTOMER (*PersonID*)

PAYER (*PersonID*, creditCardInformation)

INVOICE (*BookingbookingNumber*, *PayerPersonID*, InvoiceSummary)

BOOKING (bookingNumber, totalPrice, *FlightflightNumber, CustomerPersonID, PayerPersonID*)

TRAVELLER (*PersonID*, *BookingbookingNumber*)

TRAVELLER_FLIGHT (*TravellerPersonID*, *FlightflightNumber*)

FLIGHT (flightNumber, departureTime, lengthInMinutes, destination, price,

FlightAttendantEmployeePersonID, PilotEmployeePersonID)

PILOT (*EmployeePersonID*)

FLIGHTATTENDANT (*EmployeePersonID)*

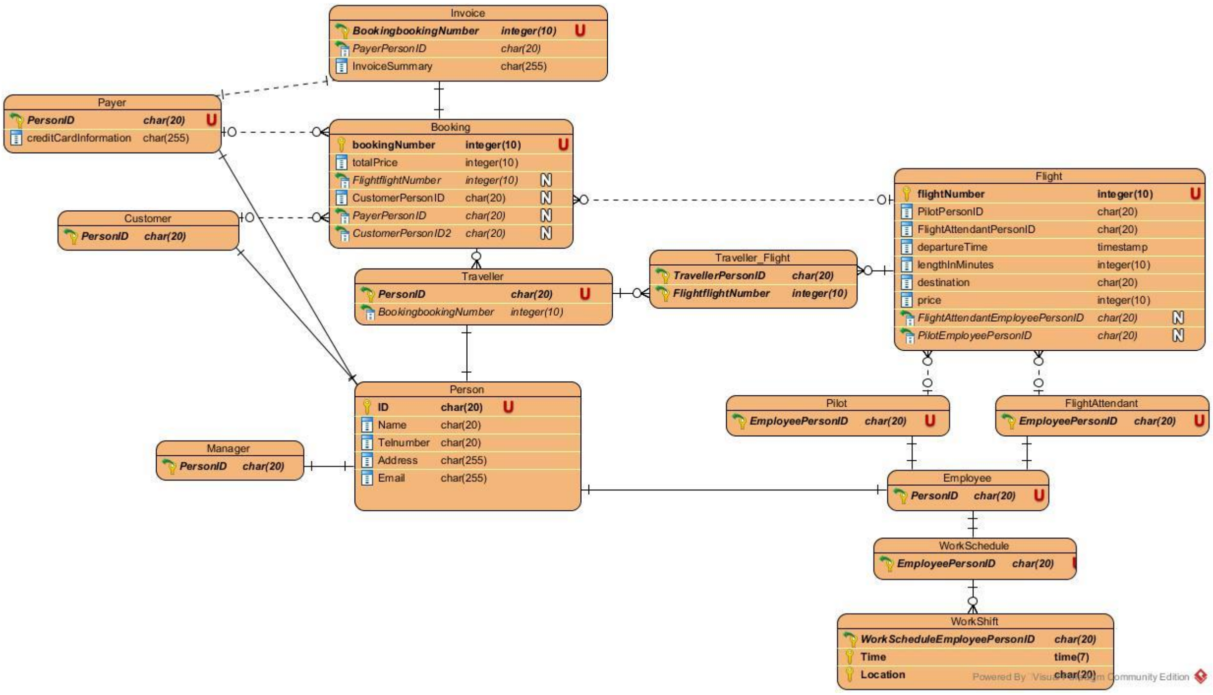
EMPLOYEE (*PersonID*)

WorkSchedule (*EmployeePersonID*)

WorkShift (*WorkScheduleEmployeePersonID*, Time, Location)

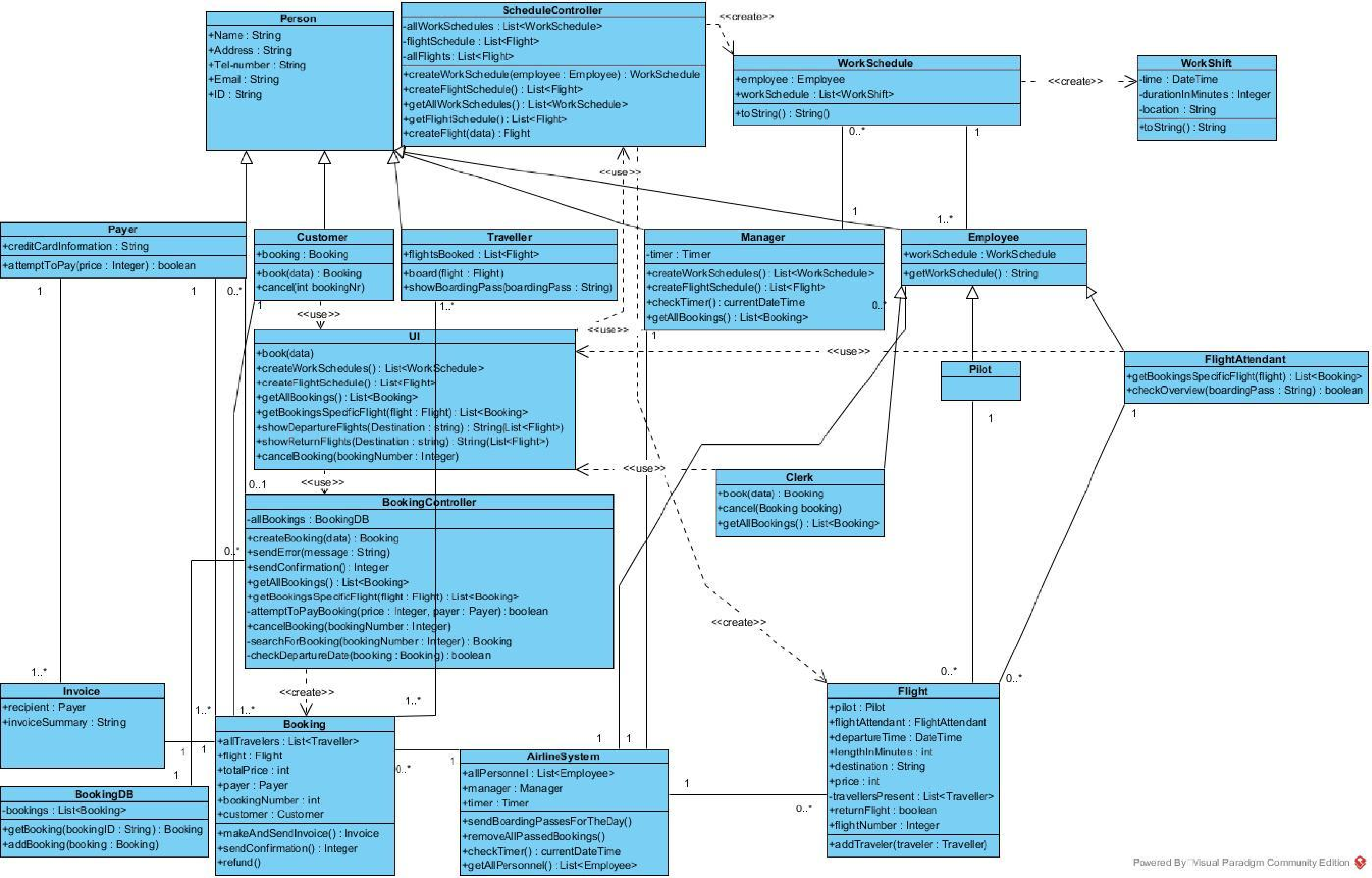
Conceptual data model

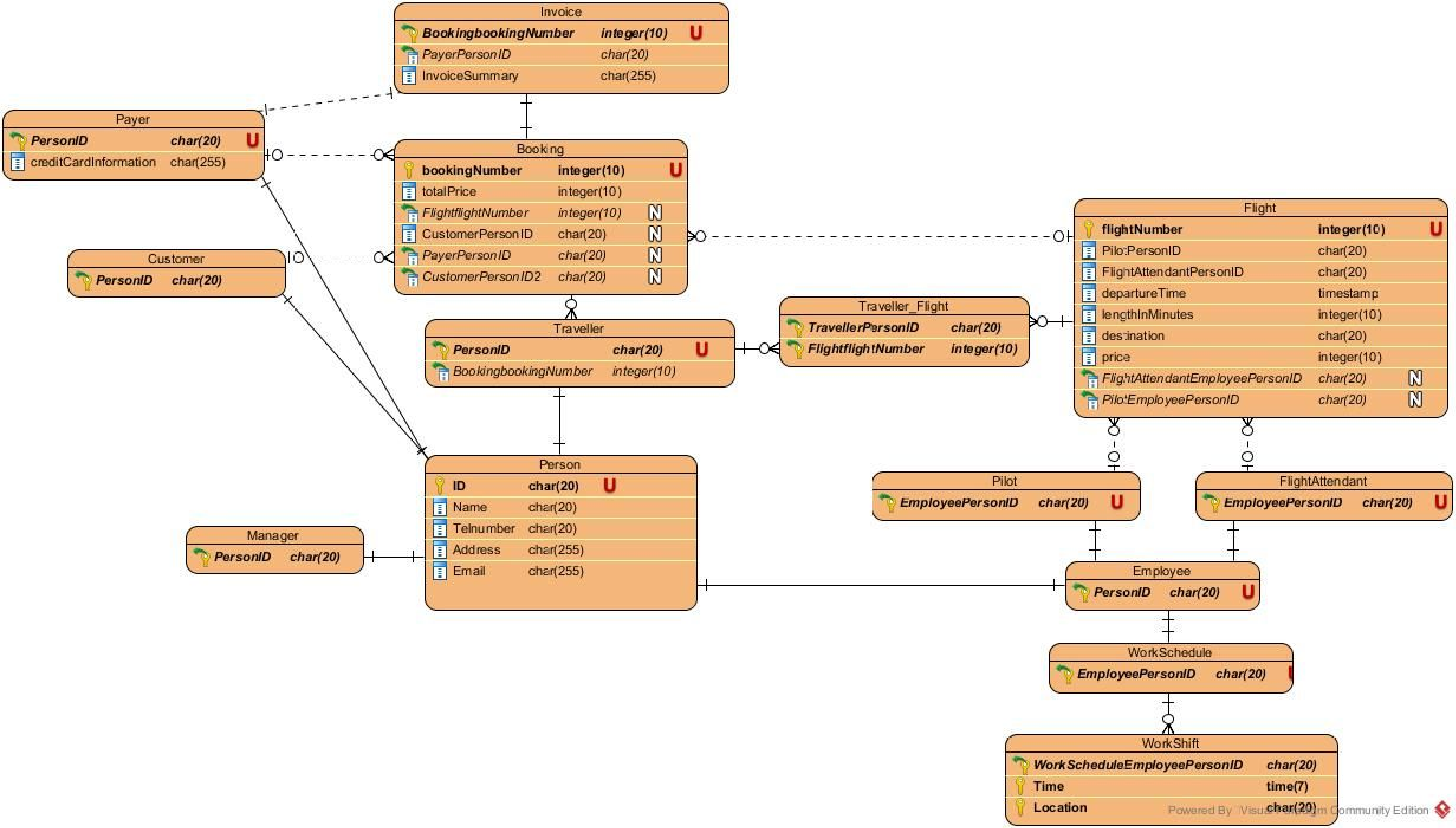
See ‘Entity Relationship Diagram.jpg’


Invoice
BookingbookingNumber integer(10) U
PayerPersonID char(20)
InvoiceSummary char(255)
Payer
PersonID char(20) U
creditCardInformation char(255)
Booking
bookingNumber integer(10) U
totalPrice integer(10)
FlightflightNumber integer(10) N
CustomerPersonID char(20) N
PayerPersonID char(20) N
CustomerPersonID2 char(20) N
Customer
PersonID char(20)
Traveller
PersonID char(20) U
BookingbookingNumber integer(10)
Traveller_Flight
TravellerPersonID char(20)
FlightflightNumber integer(10)
Flight
flightNumber integer(10) U
PilotPersonID char(20)
FlightAttendantPersonID char(20)
departureTime timestamp
lengthInMinutes integer(10)
destination char(20)
price integer(10)
FlightAttendantEmployeePersonID char(20) N
PilotEmployeePersonID char(20) N
Person
ID char(20) U
Name char(20)
Telnumber char(20)
Address char(255)
Email char(255)
Manager
PersonID char(20)
Pilot
EmployeePersonID char(20) U
FlightAttendant
EmployeePersonID char(20) U
Employee
PersonID char(20) U
WorkSchedule
EmployeePersonID char(20)
WorkShift
WorkScheduleEmployeePersonID char(20)
Time time(7)
Location char(20)
Powered By Visual Paradigm Community Edition

Things yet to be implemented

- Checking whether there is still room on the flights (now all bookings are made for flights unlimited capacity)
- Luggage system
- Hiring (adding) / Firing (removing) personnel/manager
- Operating out of different airports


Person
+Name : String
+Address : String
+Tel-number : String
+Email : String
+ID : String
ScheduleController
-allWorkSchedules : List<WorkSchedule>
-flightSchedule : List<Flight>
-allFlights : List<Flight>
+createWorkSchedule(employee : Employee) : WorkSchedule
+createFlightSchedule() : List<Flight>
+getAllWorkSchedules() : List<WorkSchedule>
+getFlightSchedule() : List<Flight>
+createFlight(data) : Flight
<<create>>
WorkSchedule
+employee : Employee
+workSchedule : List<WorkShift>
+toString() : String()
<<create>>
WorkShift
-time : DateTime
-durationInMinutes : Integer
-location : String
+toString() : String
0..*
1
<<use>>
1
1..*
Payer
+creditCardInformation : String
+attemptToPay(price : Integer) : boolean
Customer
+booking : Booking
+book(data) : Booking
+cancel(int bookingNr)
Traveller
+flightsBooked : List<Flight>
+board(flight : Flight)
+showBoardingPass(boardingPass : String)
Manager
-timer : Timer
+createWorkSchedules() : List<WorkSchedule>
+createFlightSchedule() : List<Flight>
+checkTimer() : currentDateTime
+getAllBookings() : List<Booking>
Employee
+workSchedule : WorkSchedule
+getWorkSchedule() : String
1
1
0..*
1
<<use>>
1..*
0..*
<<use>>
1
UI
+book(data)
+createWorkSchedules() : List<WorkSchedule>
+createFlightSchedule() : List<Flight>
+getAllBookings() : List<Booking>
+getBookingsSpecificFlight(flight : Flight) : List<Booking>
+showDepartureFlights(Destination : string) : String(List<Flight>)
+showReturnFlights(Destination : string) : String(List<Flight>)
+cancelBooking(bookingNumber : Integer)
<<use>>
Pilot
FlightAttendant
+getBookingsSpecificFlight(flight) : List<Booking>
+checkOverview(boardingPass : String) : boolean
1
1
<<use>>
Clerk
+book(data) : Booking
+cancel(Booking booking)
+getAllBookings() : List<Booking>
0..1
<<use>>
BookingController
-allBookings : BookingDB
+createBooking(data) : Booking
+sendError(message : String)
+sendConfirmation() : Integer
+getAllBookings() : List<Booking>
+getBookingsSpecificFlight(flight : Flight) : List<Booking>
-attemptToPayBooking(price : Integer, payer : Payer) : boolean
+cancelBooking(bookingNumber : Integer)
-searchForBooking(bookingNumber : Integer) : Booking
-checkDepartureDate(booking : Booking) : boolean
0..*
<<create>>
1..*
<<create>>
0..*
0..*
Invoice
+recipient : Payer
+invoiceSummary : String
1..*
1..*
1..*
Booking
+allTravelers : List<Traveller>
+flight : Flight
+totalPrice : int
+payer : Payer
+bookingNumber : int
+customer : Customer
+makeAndSendInvoice() : Invoice
+sendConfirmation() : Integer
+refund()
1
1
1
0..*
1
1
1
AirlineSystem
+allPersonnel : List<Employee>
+manager : Manager
+timer : Timer
+sendBoardingPassesForTheDay()
+removeAllPassedBookings()
+checkTimer() : currentDateTime
+getAllPersonnel() : List<Employee>
1
0..*
Flight
+pilot : Pilot
+flightAttendant : FlightAttendant
+departureTime : DateTime
+lengthInMinutes : int
+destination : String
+price : int
-travellersPresent : List<Traveller>
+returnFlight : boolean
+flightNumber : Integer
+addTraveler(traveler : Traveller)
BookingDB
-bookings : List<Booking>
+getBooking(bookingID : String) : Booking
+addBooking(booking : Booking)


Invoice
BookingbookingNumber integer(10) U
PayerPersonID char(20)
InvoiceSummary char(255)
Payer
PersonID char(20) U
creditCardInformation char(255)
Booking
bookingNumber integer(10) U
totalPrice integer(10)
FlightflightNumber integer(10) N
CustomerPersonID char(20) N
PayerPersonID char(20) N
CustomerPersonID2 char(20) N
Customer
PersonID char(20)
Flight
flightNumber integer(10) U
PilotPersonID char(20)
FlightAttendantPersonID char(20)
departureTime timestamp
lengthInMinutes integer(10)
destination char(20)
price integer(10)
FlightAttendantEmployeePersonID char(20) N
PilotEmployeePersonID char(20) N
Traveller_Flight
TravellerPersonID char(20)
FlightflightNumber integer(10)
Traveller
PersonID char(20) U
BookingbookingNumber integer(10)
Person
ID char(20) U
Name char(20)
Telnumber char(20)
Address char(255)
Email char(255)
Manager
PersonID char(20)
Pilot
EmployeePersonID char(20) U
FlightAttendant
EmployeePersonID char(20) U
Employee
PersonID char(20) U
WorkSchedule
EmployeePersonID char(20)
WorkShift
WorkScheduleEmployeePersonID char(20)
Time time(7)
Location char(20)