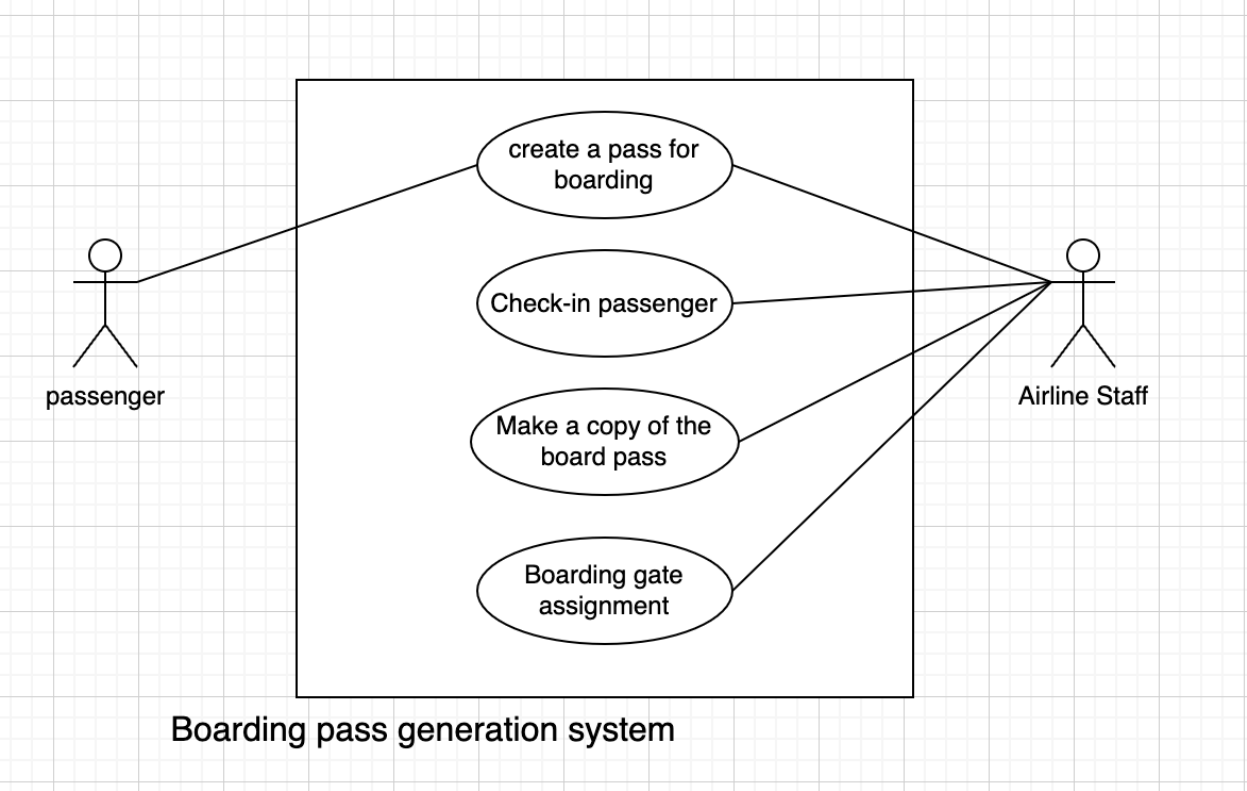
Use Case:	create a pass for boarding
Trigger:	- The passenger arrives at the check-in desk for their flight.
Precondition:	- The passenger has confirmed booking for a flight and he/she has not checked in yet. - Passenger has provided necessary personal and flight information.
Main scenarios:	
1	Passenger requests a boarding pass.
2	System prompts the passenger to provide their personal information (e.g., first name, last name, passport number….)
3	System prompts the passenger to provide their flight detail (e.g., flight number, departure time)
4	System validates the provided information.
5	System generates a boarding pass with relevant details (e.g., passenger name, seat number, hate information)
6	System presents the boarding pass to the passenger for viewing and/or printing.

UML use case:



Use Case:	Check-in passenger
Trigger:	Passenger arrives at the airport for check-in.
Precondition:	Since the passenger's reservation has been confirmed, the system has the original information already. - Passenger possesses necessary travel documents (e.g., ID, passport)
Main scenarios:	
1	Passenger approaches the check-in counter

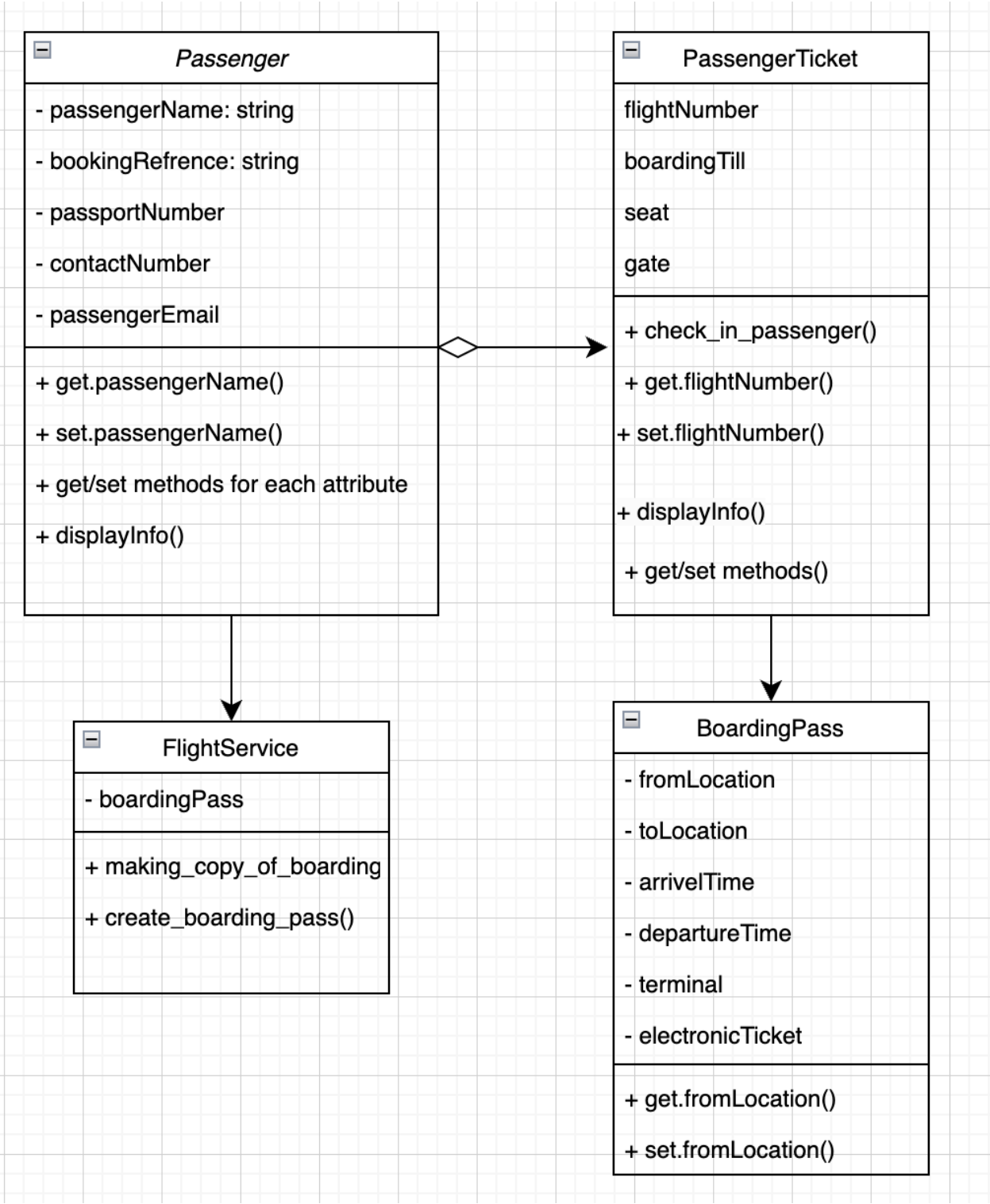
2	System prompts the passenger to provide their boarding pass or reservation code
3	System verifies the provided information and validates the passenger's identity
4	system collects additional required details from the passenger (e.g., baggage information, contact information)
5	system assigns a seat to the passenger
6	system updates the passenger's check-in status.

Use Case:	Make a copy of the board pass
Trigger:	Passenger requests a duplicate copy of their boarding pass
Precondition:	Passenger has a valid boarding pass or reservation code
Main scenarios:	
1	Passenger requests a duplicate copy of their boarding pass
2	System prompts the passenger to provide their boarding pass or reservation code.

3	System verifies the provided information
4	System generates a duplicate copy of the boarding pass.
5	system present the duplicate copy to the passenger for viewing and printing.

Use Case:	Boarding gate assignment
Trigger:	The flight is ready for gate assignment
Precondition:	-the flight details have been confirmed -the aircraft is ready for passengers
Main scenarios:	
1	Airline staff selects the flight from the system dashboard to assign a gate
2	The system present available gates based on the flight's size and schedule
3	The staff chooses an appropriate gate and confirms the assignment
4	The system updates the flight information with the assigned gate
5	the system disseminates the gate information to be displayed on airport monitors and updated on digital boarding passes.

UML class diagram



1. Passenger: A type of the `Passenger` class, this object represents a passenger. It has properties like {bookingReference}, {passportNumber}, {contactNumber}, {passengerEmail}, and {passengerName}. It includes ways to display the passenger's data and to access and set these properties.

2. PassengerTicket: An instance of the `PassengerTicket` class, this object represents a passenger's ticket. It has properties like {gate}, {seat}, {boardingTill}, and {flightNumber}. It includes methods to execute the passenger's check-in as well as methods to access and set these properties. It provides a way to show the ticket details.

3. BoardingPass: An instance of the `BoardingPass` class, this object represents a boarding pass. It has attributes like {electronicTicket}, {terminal}, {arrivelTime}, {departureTime}, {fromLocation}, and {toLocation}. It includes methods to display the boarding pass information and to access and set these characteristics.

4. FlightService: An instance of the `FlightService` class, this object represents a flight service. It offers methods for creating a copy of the boarding pass and copying the pass depending on flight and passenger information. It also has a {boardingPass} attribute.

Python classes

```
class Passenger:
    def __init__(self, passengerName, bookingReference, passportNumber,
contactNumber, passengerEmail):
        self._passengerName = passengerName
        self._bookingReference = bookingReference
        self._passportNumber = passportNumber
        self._contactNumber = contactNumber
        self._passengerEmail = passengerEmail

    def get_name(self):
        return self._passengerName

    def set_passengerName(self, passengerName):
        self._passengerName = passengerName

    def get_bookingReference(self):
        return self._bookingReference

    def set_bookingReference(self, bookingReference):
        self._bookingReference = bookingReference

    def get_passengerEmail(self):
        return self._passengerEmail

    def set_passengerEmail(self, passengerEmail):
        self._passengerEmail = passengerEmail

    def get_contactNumber(self):
        return self._contactNumber

    def set_contactNumber(self, contactNumber):
        self._contactNumber = contactNumber

    def get_passpotNumber(self):
        return self._passportNumber

    def set_passport_number(self, passportNumber):
        self._passportNumber = passportNumber


    def displayInfo(self):
        print(f"Passenger Name: {self._passengerName}")
```

```
        print(f"Passenger: {self._bookingReference}")
        print(f"Phone Number: {self._contactNumber}")
        print(f"Passport Number: {self._passportNumber}")
        print(f"Passenger email: {self._passengerEmail}")



class PassengerTicket:
    def __init__(self, flightNumber, boardingTill, seat, gate):
        self._flightNumber = flightNumber
        self._boardingTill = boardingTill
        self._seat = seat
        self._gate = gate


    def check_in_passenger(self):
        # Perform the check-in(tickets) process for the passenger
        pass


    def get_flightNumber(self):
        return self._flightNumber

    def set_flightNumber(self, flightNumber):
        self._flightNumber = flightNumber

    def get_boardingTill(self):
        return self._boardingTill

    def set_boardingTill(self, boardingTill):
        self._boardingTill = boardingTill

    def get_seat(self):
        return self._seat

    def set_seat(self, seat):
        self._seat = seat

    def get_gate(self):
        return self._gate

    def set_gate(self, gate):
        self._gate = gate
```

```
    def displayInfo(self):
        print(f"Flight number: {self._flightNumber}")
        print(f"Check in time: {self._boardingTill}")
        print(f"Seat prefrences: {self._seat}")
        print(f"Gate Number: {self._gate}")

class BoardingPass:
    def __init__(self, fromLocation, toLocation, arrivelTime,
departureTime, terminal, electronicTicket):
        self._fromLocation = fromLocation
        self._toLocation = toLocation
        self._arrivelTime = arrivelTime
        self._departureTime = departureTime
        self._terminal = terminal
        self._electronicTicket = electronicTicket



    def get_fromLocation(self):
        return self._fromLocation

    def set_fromLocation(self, fromLocation):
        self._fromLocation = fromLocation

    def get_toLocation(self):
        return self._toLocation

    def set_toLocation(self, toLocation):
        self._toLocation = toLocation

    def get_arrivelTime(self):
        return self._arrivelTime

    def set_arrivelTime(self, arrivelTime):
        self._arrivelTime = arrivelTime

    def get_departureTime(self):
        return self._departureTime

    def set_departureTime(self, departureTime):
        self._departureTime = departureTime

    def get_terminal(self):
        return self._terminal
```

```
    def set_terminal(self, terminal):
        self._terminal = terminal

    def get_electronicTicket(self):
        return self._electronicTicket

    def set_electronicTicket(self, electronicTicket):
        self._electronicTicket = electronicTicket

    def displayInfo(self):
        print(f"Flight from: {self._fromLocation}")
        print(f"Flight to: {self._toLocation}")
        print(f"Arrivel Time: {self._arrivelTime}")
        print(f"Departure time: {self._departureTime}")
        print(f"Gate Number : {self._terminal}")
        print(f"Electronic Ticket: {self._electronicTicket}")

class flightService:
    def __init__(self):
        self.boardingPass = {}

    def make_copy_of_boarding_pass(self, PassengerTicket):
      # Create a duplicate copy of the boarding pass
        pass

    def create_boarding_pass(self, passenger):
        # Generate a boarding pass based on the passenger and flight
information
        pass




# example of the entire process
passenger = Passenger(passengerName="JAMES SMITH",
bookingReference="5A6BCD78",passengerEmail="James.smith@gmail.com",contact
Number="0567855433",passportNumber="NA132456788")
passengerTicket =
PassengerTicket(flightNumber="NA4321",boardingTill="11:20", seat=
"09A",gate="03")
boarding_pass = BoardingPass(fromLocation="Chicago ORD",toLocation="New
York JFK",arrivelTime="11:20",departureTime="11:40",terminal="03",
electronicTicket="629")
flight_service = flightService()
```

```
Passenger.displayInfo(passenger)
PassengerTicket.displayInfo(passengerTicket)
boarding_pass.displayInfo()
```