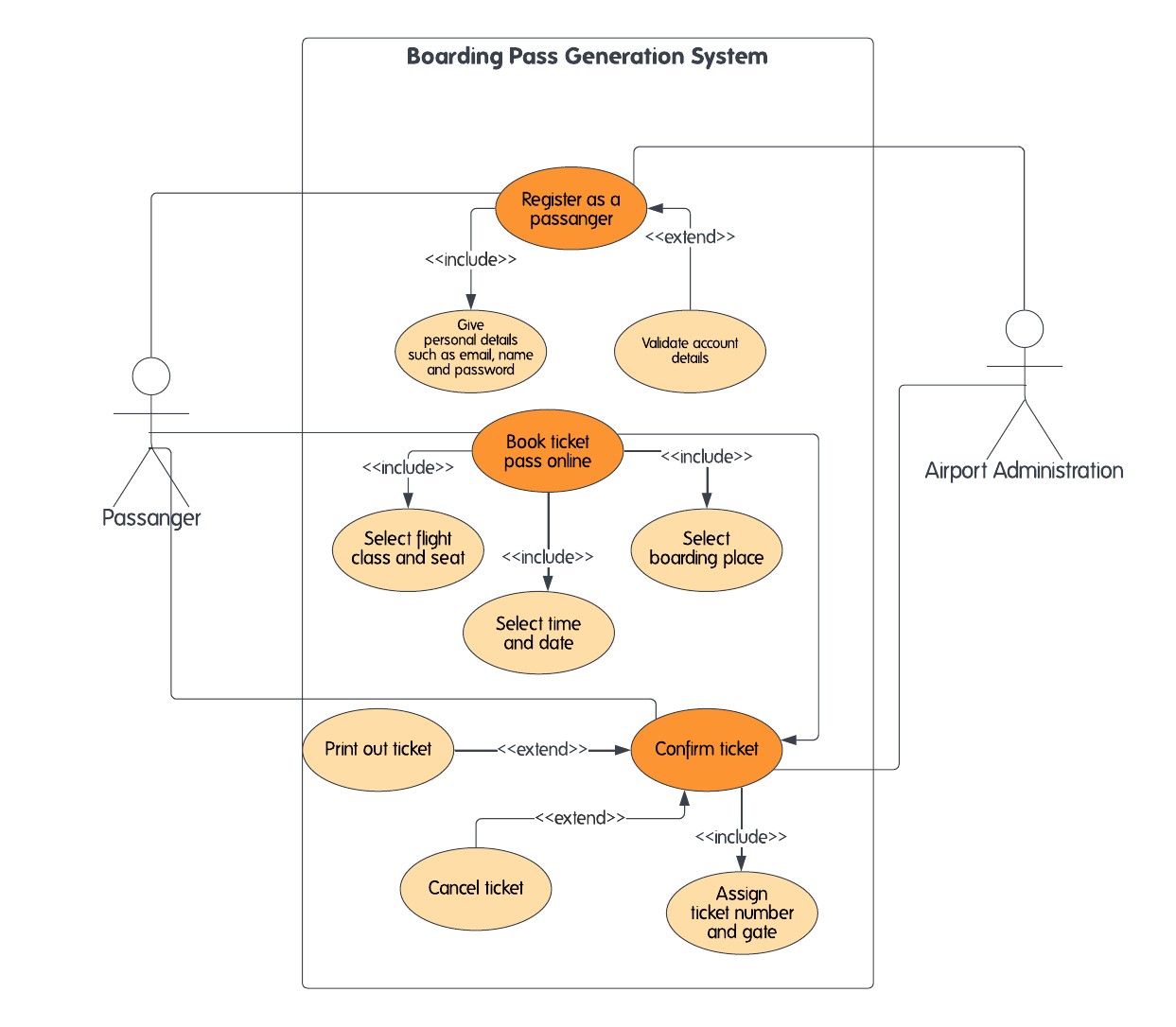
**Part A**

**Usecase Diagram:**



**Usecases:**

* Give personal details (includes email, name, and password)
* Select flight class and seat\* Register as a passenger
* Print out ticket
* Book ticket pass online
* Select time and date
* Cancel ticket
* account details
* Select boarding place
* Confirm ticket
* Assign ticket number and

**Usecase description:**

* Give personal details: This use-case describes the process of providing personal information, including email, name, and password, when registering as a new passenger.
* Select flight class and seat: This use-case describes the process of selecting a flight class and seat assignment when booking a ticket.
* Register as a passenger: This use- describes the process of creating a new passenger account, which includes providing personal, selecting a flight, and confirming ticket details.
* Print ticket: This use-case describes the process of printing a physical ticket for a passenger, either at a self-service kiosk or at a check-in counter.
* Book ticket pass online: This use-case describes the process of booking a ticket online, which includes selecting a flight, class, and seat, providing payment information, and confirming ticket details.
* Select time and date: This use-case describes the process of selecting a time and date for a flight when booking a ticket.
* Cancel ticket: This use-case describes the process of canceling a ticket, either because the passenger has changed their plans or for some other reason.
* Validate account details: This use-case describes the process of verifying the accuracy of a passenger's account information, such as their name, email, and password.
* Select boarding place: This use-case describes the process of selecting a boarding place when booking a ticket, such as a specific airport or terminal.
* Confirm ticket: This use-case describes the process of confirming ticket details, such as the flight number, date, time, and seat assignment, before finalizing a booking.
* Assign ticket number and gate: This use-case describes the process of assigning a unique ticket number and gate information to a passenger's booking.

**Part B**

**Class Diagram:**

A diagram of a computer program

Description automatically generated with medium confidence

**Class description:**

The classes in this diagram include DomesticFlight, Flight, Seat, Passenger, and BoardingPass. The DomesticFlight class contains information about the aircraft type and the operating airline. The Flight class includes details about the flight number, departure and arrival airports, and date and time of departure. The Seat class contains information about the seat number and class type. The Passenger class includes information about the passenger's name, email, and password. The BoardingPass class contains details about the passenger, flight, seat, ticket number, gate, and boarding time.

In terms of relationships, the **DomesticFlight** class is the child class of **Flight**, which means it inherits the attributes of the parent class. Then, we have a one-to-many composition relationship between the **Flight** class and **Seat** class. One flight must have one to many seats in it.

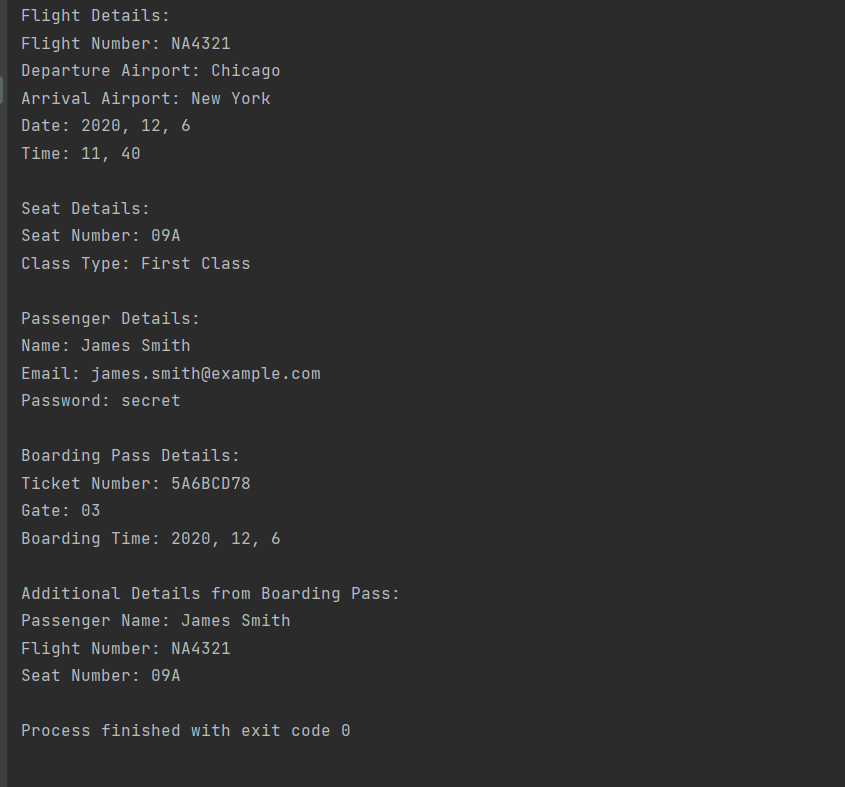
Next, we have a one-to-many aggregation relationship between **Flight** and **Passenger** classes. A flight can have one to many passengers. **Passenger** and **BoardingPass** have a one-to-one association relationship. One passenger has a boarding pass to board a flight, so that's why it's a one-to-one relationship.

**Part C**

**Code :**

# Represents a flight with a flight number, departure airport, arrival airport, date, and time.  
class Flight:  
 def \_\_init\_\_(self, flight\_number, departure\_airport, arrival\_airport, date, time):  
 self.flight\_number = flight\_number  
 self.departure\_airport = departure\_airport  
 self.arrival\_airport = arrival\_airport  
 self.date = date  
 self.time = time  
  
 def get\_flight\_number(self):  
 return self.flight\_number  
  
 def set\_flight\_number(self, flight\_number):  
 self.flight\_number = flight\_number  
  
 def get\_departure\_airport(self):  
 return self.departure\_airport  
  
 def set\_departure\_airport(self, departure\_airport):  
 self.departure\_airport = departure\_airport  
  
 def get\_arrival\_airport(self):  
 return self.arrival\_airport  
  
 def set\_arrival\_airport(self, arrival\_airport):  
 self.arrival\_airport = arrival\_airport  
  
 def get\_date(self):  
 return self.date  
  
 def set\_date(self, date):  
 self.date = date  
  
 def get\_time(self):  
 return self.time  
  
 def set\_time(self, time):  
 self.time = time  
  
# Represents a domestic flight which is a type of flight with an aircraft type and an operation airline.  
class DomesticFlight(Flight):  
 def \_\_init\_\_(self, aircraft\_type, operation\_airline, flight\_number, departure\_airport, arrival\_airport, date, time):  
 super().\_\_init\_\_(flight\_number, departure\_airport, arrival\_airport, date, time)  
 self.aircraft\_type = aircraft\_type  
 self.operation\_airline = operation\_airline  
  
 def get\_aircraft\_type(self):  
 return self.aircraft\_type  
  
 def set\_aircraft\_type(self, aircraft\_type):  
 self.aircraft\_type = aircraft\_type  
  
 def get\_operation\_airline(self):  
 return self.operation\_airline  
  
 def set\_operation\_airline(self, operation\_airline):  
 self.operation\_airline = operation\_airline  
  
# Represents a passenger with a name, email, and password.  
class Passenger:  
 def \_\_init\_\_(self, name, email, password):  
 self.name = name  
 self.email = email  
 self.password = password  
  
 def get\_name(self):  
 return self.name  
  
 def set\_name(self, name):  
 self.name = name  
  
 def get\_email(self):  
 return self.email  
  
 def set\_email(self, email):  
 self.email = email  
  
 def get\_password(self):  
 return self.password  
  
 def set\_password(self, password):  
 self.password = password  
  
# Represents a seat with a seat number and class type.  
class Seat:  
 def \_\_init\_\_(self, seat\_number, class\_type):  
 self.seat\_number = seat\_number  
 self.class\_type = class\_type  
  
 def get\_seat\_number(self):  
 return self.seat\_number  
  
 def set\_seat\_number(self, seat\_number):  
 self.seat\_number = seat\_number  
  
 def get\_class\_type(self):  
 return self.class\_type  
  
 def set\_class\_type(self, class\_type):  
 self.class\_type = class\_type  
  
# Represents a boarding pass with a passenger, flight, seat, ticket number, gate, and boarding time.  
class BoardingPass:  
 def \_\_init\_\_(self, passenger, flight, seat, ticket\_number, gate, boarding\_time):  
 self.passenger = passenger # Passenger object  
 self.flight = flight # Flight object  
 self.seat = seat # Seat object  
 self.ticket\_number = ticket\_number  
 self.gate = gate  
 self.boarding\_time = boarding\_time  
  
 def get\_passenger(self):  
 return self.passenger  
  
 def set\_passenger(self, passenger):  
 self.passenger = passenger  
  
 def get\_flight(self):  
 return self.flight  
  
 def set\_flight(self, flight):  
 self.flight = flight  
  
 def get\_seat(self):  
 return self.seat  
  
 def set\_seat(self, seat):  
 self.seat = seat  
  
 def get\_ticket\_number(self):  
 return self.ticket\_number  
  
 def set\_ticket\_number(self, ticket\_number):  
 self.ticket\_number = ticket\_number  
  
 def get\_gate(self):  
 return self.gate  
  
 def set\_gate(self, gate):  
 self.gate = gate  
  
 def get\_boarding\_time(self):  
 return self.boarding\_time  
  
 def set\_boarding\_time(self, boarding\_time):  
 self.boarding\_time = boarding\_time  
  
  
  
# Create flight object  
flight = Flight(flight\_number='NA4321', departure\_airport='Chicago', arrival\_airport='New York', date='2020, 12, 6' ,time='11, 40')  
  
# Create seat object  
seat = Seat(seat\_number='09A', class\_type='First Class')  
  
# Create passenger object  
passenger = Passenger(name='James Smith', email='james.smith@example.com', password='secret')  
  
# Create boarding pass object  
boarding\_pass = BoardingPass(passenger=passenger, flight=flight, seat=seat, ticket\_number='5A6BCD78', gate='03', boarding\_time='2020, 12, 6')  
  
# Print flight details  
print("Flight Details:")  
print("Flight Number:", flight.get\_flight\_number())  
print("Departure Airport:", flight.get\_departure\_airport())  
print("Arrival Airport:", flight.get\_arrival\_airport())  
print("Date:", flight.get\_date())  
print("Time:", flight.get\_time())  
  
# Print seat details  
print("\nSeat Details:")  
print("Seat Number:", seat.get\_seat\_number())  
print("Class Type:", seat.get\_class\_type())  
  
# Print passenger details  
print("\nPassenger Details:")  
print("Name:", passenger.get\_name())  
print("Email:", passenger.get\_email())  
print("Password:", passenger.get\_password())  
  
# Print boarding pass details  
print("\nBoarding Pass Details:")  
print("Ticket Number:", boarding\_pass.get\_ticket\_number())  
print("Gate:", boarding\_pass.get\_gate())  
print("Boarding Time:", boarding\_pass.get\_boarding\_time())  
  
# Additional details about passenger, flight, and seat in boarding pass  
print("\nAdditional Details from Boarding Pass:")  
print("Passenger Name:", boarding\_pass.get\_passenger().get\_name())  
print("Flight Number:", boarding\_pass.get\_flight().get\_flight\_number())  
print("Seat Number:", boarding\_pass.get\_seat().get\_seat\_number())

**Output:**

****

**Part D**

[GitHub - najlakhalid/flight-management](https://github.com/najlakhalid/flight-management)

**Part E**

**Summary of earnings:**

Through this project, I've gained a better understanding of software development. First, I learned about creating use case diagrams which show the different things the software needs to do, like booking a flight or checking in. These diagrams helped me understand what users expect from the system. Then, I explored creating use cases, which are like step-by-step instructions for each action in the diagram, detailing how users interact with the system. After that, I worked on class diagrams, which shows the structure of the software by showing the classes (like Flight or Passenger), their attributes (such as flight number or name), and how they relate to each other (like how a Flight has many Passengers). Finally, I put all this knowledge into practice by implementing these concepts in code, turning the diagrams into real software components. This experience has been instrumental in understanding how to design and build software systems effectively.