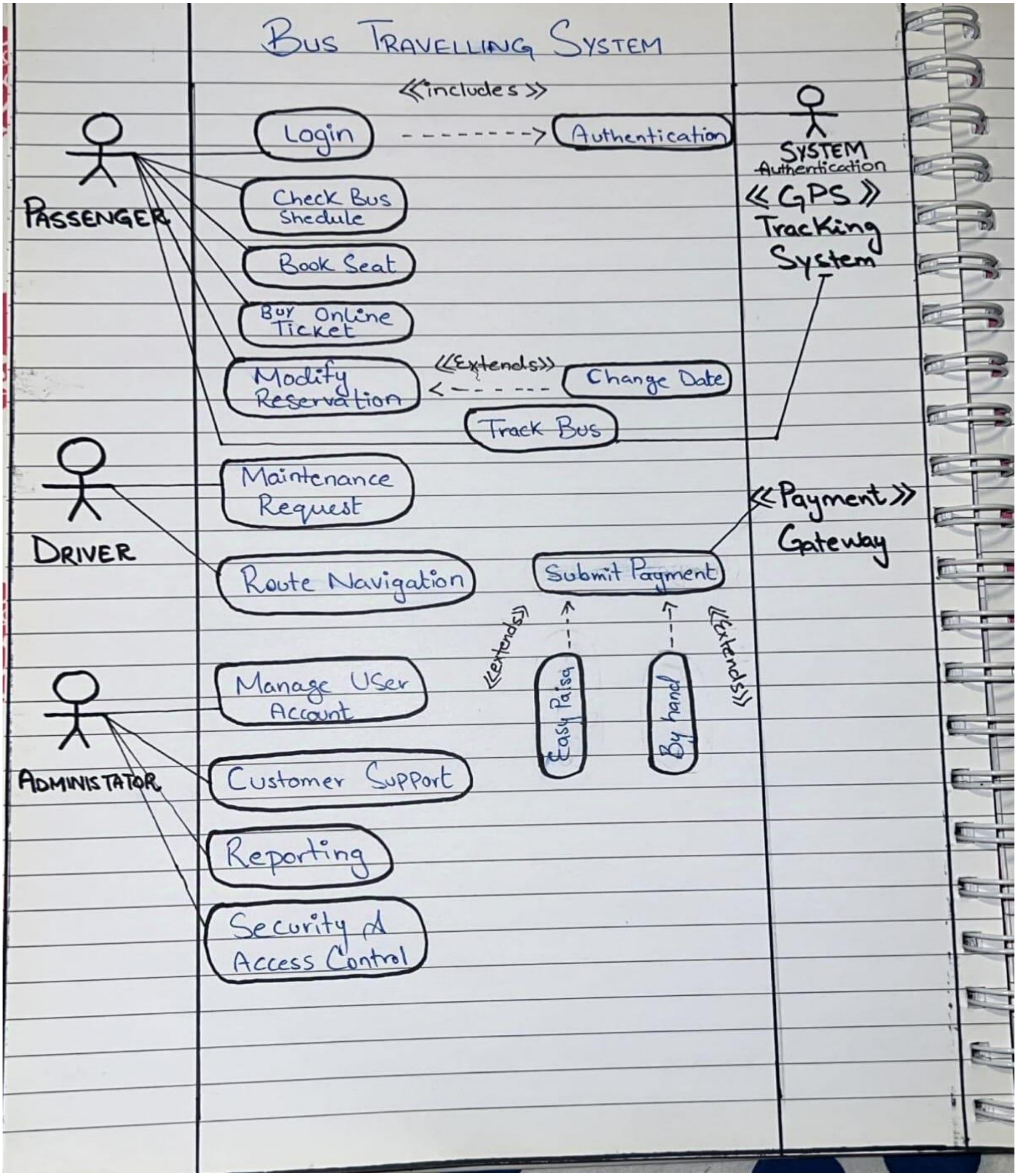
- Software Design and Architecture

Bus Travelling system

- **Submitted By**: **Noman Shakir**
- **Submitted To**: **Sir Mukhtiar Zamin**
- **Reg No**: **FA22-BSE-115**
- **Class** = **BSE-5A**

Use Case Diagram:



Fully Dressed diagram:

1. Use Case Name:

Book Seat

2. Primary Actor:

- **Passenger** (An individual who wishes to book a seat on the bus)

3. ***Stakeholders and Interests:***

- **Passenger**: Wants to book a bus seat easily and quickly based on their travel plans.
- **System Administrator**: Ensures that the system performs seat booking efficiently without overloading the system and preventing double bookings.
- **Bus Operator**: Wants to manage bookings effectively to ensure proper seat allocation and avoid overbooking.

4. Preconditions:

- The user is logged in to the system (if login is required).
- The bus schedule and seating availability data are up to date.
- The user has access to the booking system.

5. Postconditions:

- **Success Postcondition**: The seat is successfully booked, and a confirmation is displayed to the user.
- **Failure Postcondition**: The user is informed of any issues preventing seat booking (e.g., seat unavailable, payment failure).

6. Main Success Scenario (Basic Flow):

- The **Passenger** selects the "Book Seat" option on the system interface.
- The system displays available buses and seating options.
- The **Passenger** selects a bus, date, and seat from the available options.
- The system checks the seat availability for the selected bus and date.
- The system prompts the **Passenger** to confirm booking details.
- The **Passenger** confirms the seat booking.
- The system processes the payment for the booking (if applicable).
- The system successfully books the seat and generates a booking confirmation.

- The **Passenger** is provided with a booking reference and travel details.

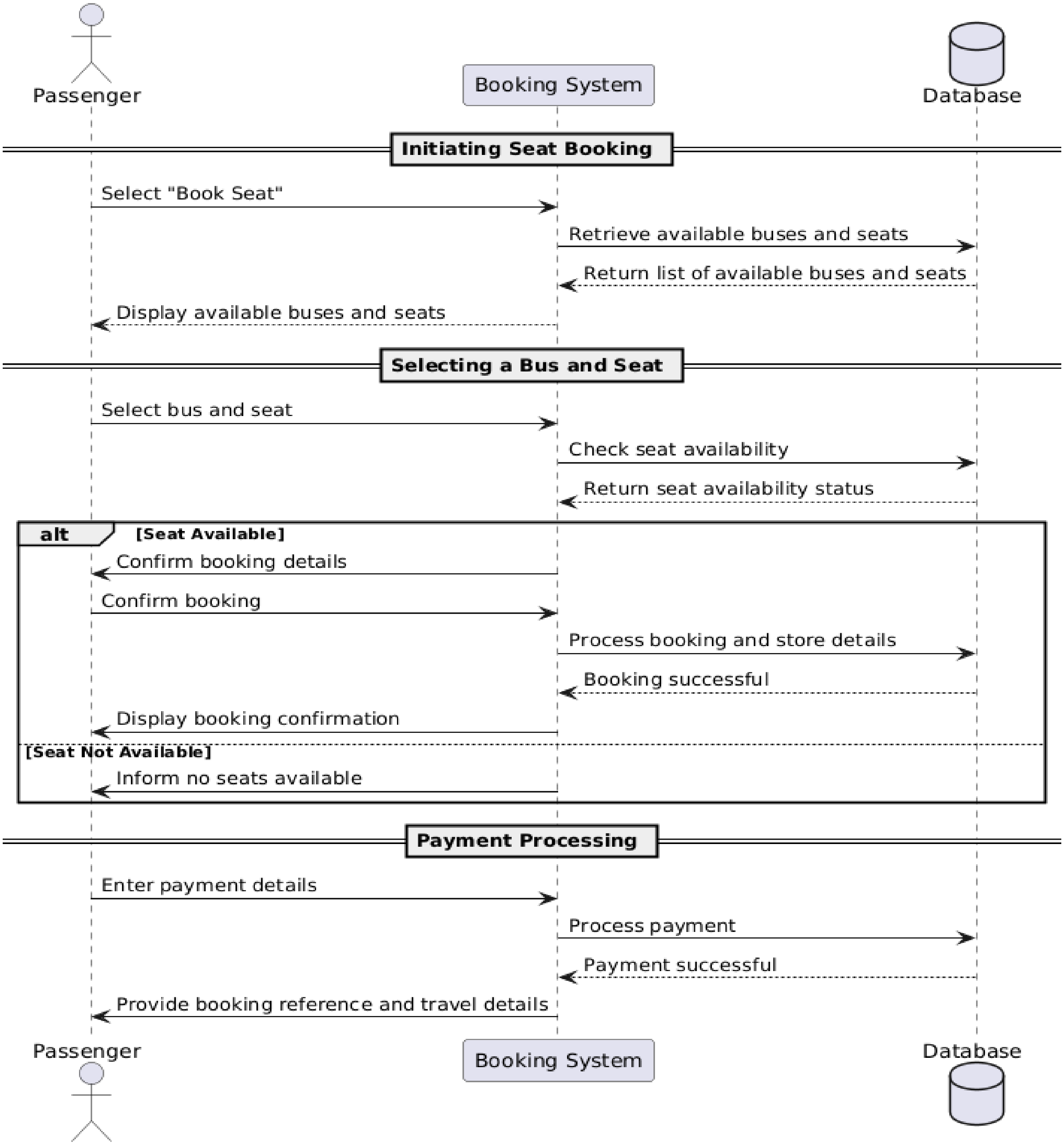
7. Extensions (Alternative Flows):

- **3a. Passenger Enters Invalid Details**:
    - 3a1. The system detects invalid inputs such as invalid bus date or seat number.
    - 3a2. The system prompts the **Passenger** to correct the invalid details.
    - 3a3. The **Passenger** corrects the input and resubmits the booking request.
- **6a. System Experiences Payment Error**:
    - 6a1. The system fails to process the payment.
    - 6a2. The system displays an error message explaining the payment issue.
    - 6a3. The **Passenger** may choose to retry the payment or cancel the booking.
- **7a. No Seats Available**:
    - 7a1. The system finds no available seats on the selected bus.
    - 7a2. The system informs the **Passenger** of the lack of available seats.
    - 7a3. The system suggests alternative buses or dates or allows the **Passenger** to modify their search.

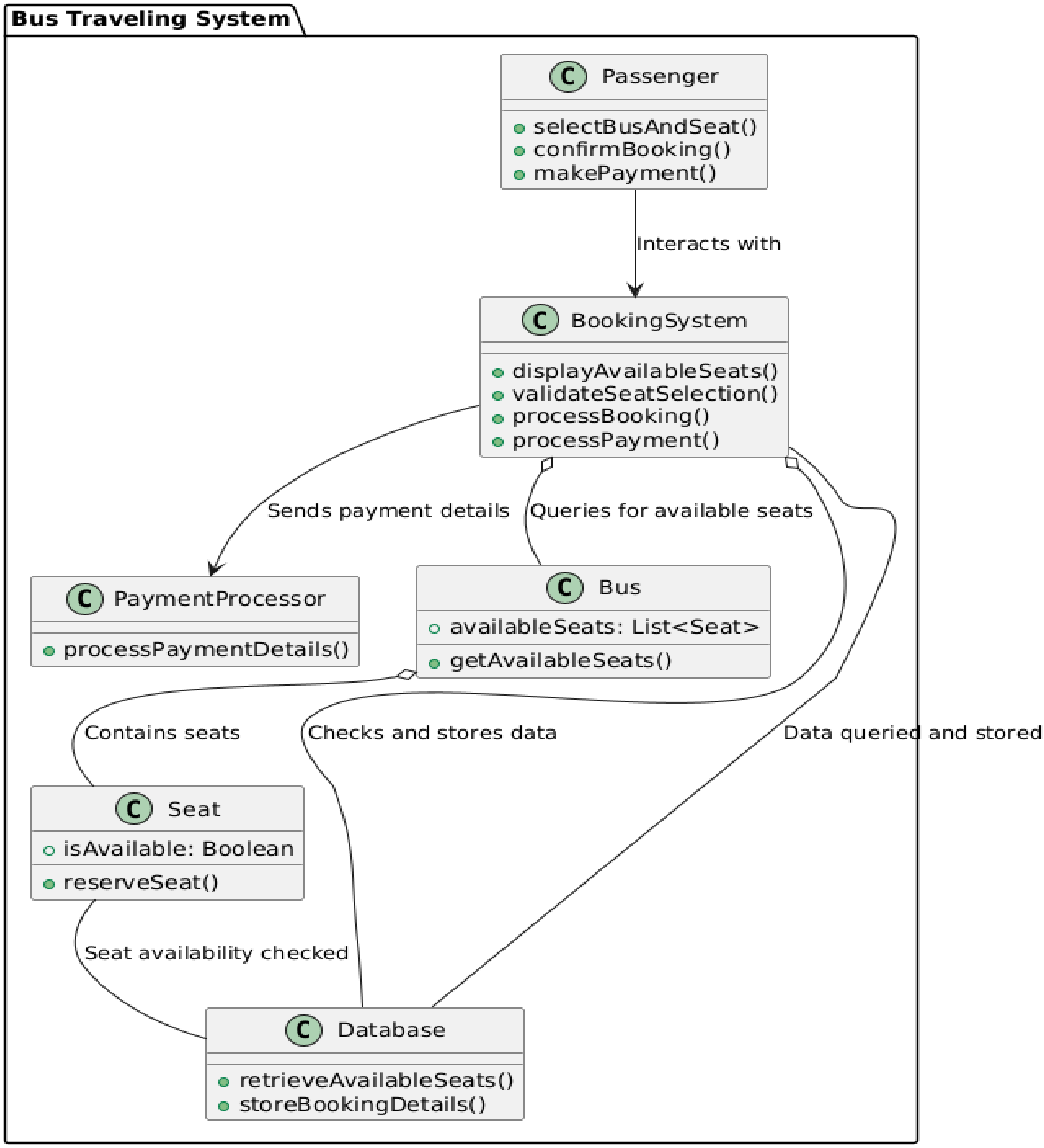
8. Special Requirements:

- **Performance**: The system should process seat bookings within 2 seconds under normal load conditions.
- **Usability**:
    - The booking interface should be intuitive and user-friendly.
    - Support for seat selection using a visual representation of the bus seating arrangement.
- **Scalability**: The system must handle a large number of concurrent seat booking requests without performance degradation.
- **Accessibility**: The system should comply with accessibility standards to support users with disabilities.

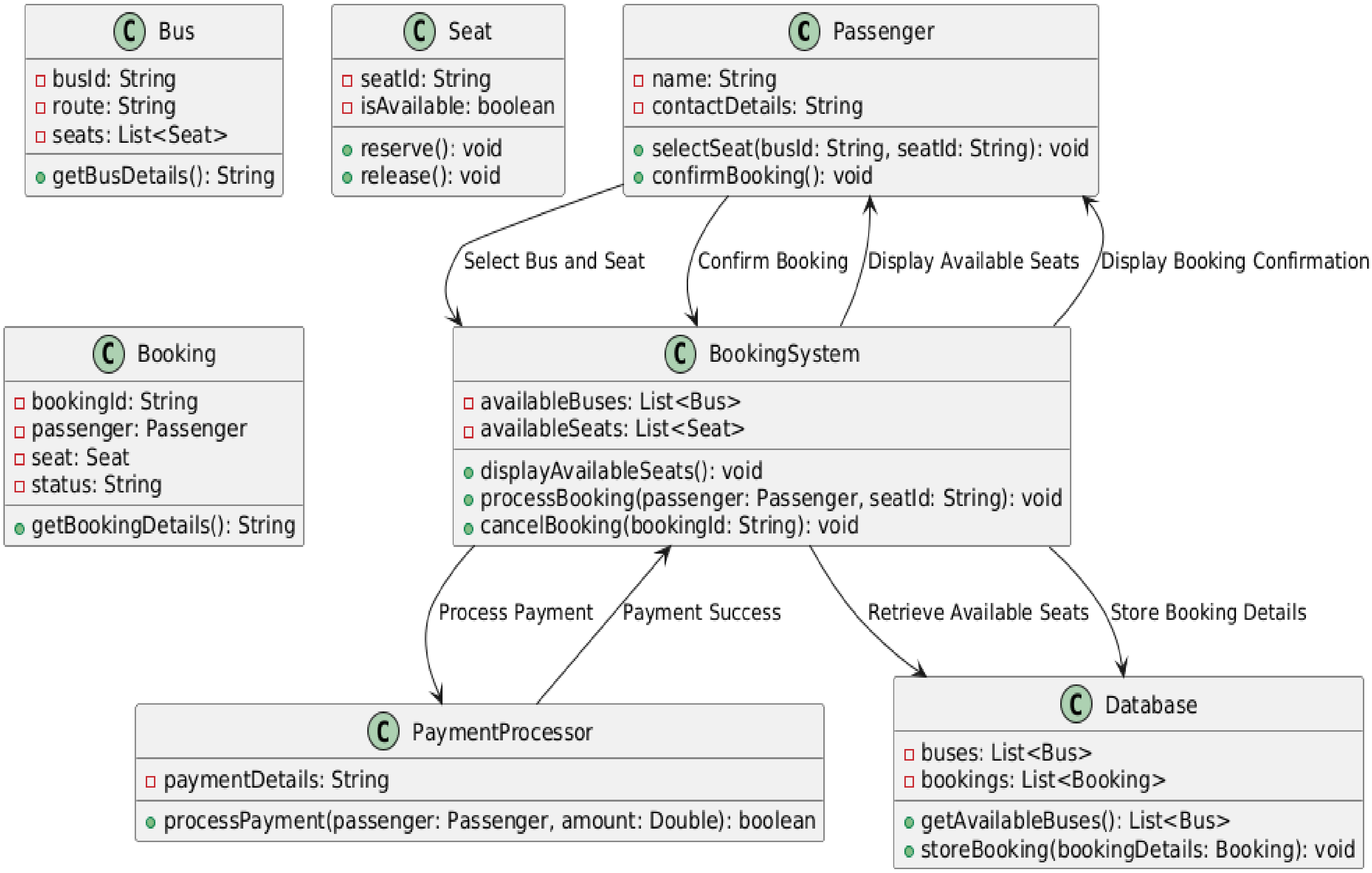
***Sequence diagram*:**



Composite Structure Diagram:



Class Diagram:


Bus
busId: String
route: String
seats: List<Seat>
getBusDetails(): String
Seat
seatId: String
isAvailable: boolean
reserve(): void
release(): void
Passenger
name: String
contactDetails: String
selectSeat(busId: String, seatId: String): void
confirmBooking(): void
Select Bus and Seat
Confirm Booking
Display Available Seats
Display Booking Confirmation
Booking
bookingId: String
passenger: Passenger
seat: Seat
status: String
getBookingDetails(): String
BookingSystem
availableBuses: List<Bus>
availableSeats: List<Seat>
displayAvailableSeats(): void
processBooking(passenger: Passenger, seatId: String): void
cancelBooking(bookingId: String): void
Process Payment
Payment Success
Retrieve Available Seats
Store Booking Details
PaymentProcessor
paymentDetails: String
processPayment(passenger: Passenger, amount: Double): boolean
Database
buses: List<Bus>
bookings: List<Booking>
getAvailableBuses(): List<Bus>
storeBooking(bookingDetails: Booking): void