Airport System: Flight Booking System

You are tasked with building a simple Flight Booking System for an airport. The system should handle flight details, booking passengers, and assigning seats.

Requirements:

1. **Class Definitions**:
   - **Flight**: Represents a flight with attributes like flight number, destination, departure time, and available seats.
   - **Passenger**: Represents a passenger with attributes like name, passport number, and contact information.
   - **Booking**: Represents a booking made by a passenger for a specific flight. It includes the passenger, the flight, and the seat assigned.
2. **Functionality**:
   - Each **Flight** can have multiple passengers. However, there are limited seats available for each flight.
   - Passengers can book seats on a flight, but they must be assigned a seat based on availability.
   - The **Booking** class will represent a successful booking and should contain methods for checking if a seat is available and for assigning a seat to the passenger.
3. **Behavior**:
   - Implement the `book_ticket()` method in the **Flight** class to handle the booking process.
   - Implement a method to check if a seat is available (`is_seat_available()` in the **Flight** class).
   - Use OOP principles like **Encapsulation**, **Inheritance** (optional), **Polymorphism**, and **Abstraction** to structure the solution cleanly.

Steps:

1. **Class `Flight`**:
   - Attributes:
     - `flight_number` (str): The unique identifier for the flight.
     - `destination` (str): The destination of the flight.
     - `departure_time` (str): The time of departure in "HH:MM" format.
     - `available_seats` (int): The number of available seats on the flight.
     - `booked_passengers` (list): A list to store passengers who have booked tickets for this flight.
   - Methods:
     - `is_seat_available()`: Returns a boolean indicating whether there are available seats.

        - `book_ticket(passenger)`: Books a ticket for a passenger and assigns them a seat if available.
2. **Class `Passenger`**:
    - Attributes:
        - `name` (str): The name of the passenger.
        - `passport_number` (str): The unique identifier for the passenger.
        - `contact_info` (str): The contact information of the passenger.
    - Methods:
        - `get_details()`: Returns a string with the passenger's details.
3. **Class `Booking`**:
    - Attributes:
        - `flight`: The flight associated with the booking.
        - `passenger`: The passenger who made the booking.
        - `seat_number`: The assigned seat number.
    - Methods:
        - `get_booking_info()`: Returns the booking information including flight details and passenger details.

Example Flow:

```
# Create some flights
flight1 = Flight(flight_number="A123", destination="Paris",
departure_time="15:30", available_seats=2)
flight2 = Flight(flight_number="B456", destination="London",
departure_time="18:45", available_seats=1)

# Create some passengers
passenger1 = Passenger(name="John Doe", passport_number="J1234567",
contact_info="john@example.com")
passenger2 = Passenger(name="Alice Smith", passport_number="A9876543",
contact_info="alice@example.com")

# Book tickets
flight1.book_ticket(passenger1)
flight1.book_ticket(passenger2)  # Should successfully book the second
seat

# Attempt to book a third passenger on a full flight
flight1.book_ticket(Passenger(name="Bob White",
passport_number="B6543210", contact_info="bob@example.com"))
```

```
# Display booking information
print(flight1.booked_passengers[0].get_details())  # John Doe details
print(flight1.booked_passengers[1].get_details())  # Alice Smith
details
```