```
#include <iostream>
#include <vector>
#include <map>
#include <string>
#include <ctime>
#include <iomanip>

using namespace std;

class Flight {
protected:
  string flightNumber;
  string destination;
  string departure;
  int totalSeats;
  int availableSeats;
  double basePrice;
public:
  Flight(string fn, string dest, string dep, int seats, double price)
    : flightNumber(fn), destination(dest), departure(dep),
     totalSeats(seats), availableSeats(seats), basePrice(price) {}

  virtual void showFlightDetails() const {
    cout << "Flight Number: " << flightNumber << endl;
    cout << "Destination: " << destination << endl;
    cout << "Departure: " << departure << endl;
    cout << "Available Seats: " << availableSeats << "/" << totalSeats << endl;
  }

  virtual double calculateDynamicPrice() const = 0;
```

```
    bool bookSeats(int seats) {
        if (availableSeats >= seats) {
            availableSeats -= seats;
            return true;
        }
        return false;
    }

    int getAvailableSeats() const { return availableSeats; }
    string getFlightNumber() const { return flightNumber; }
};

class DynamicFlight : public Flight {
public:
    DynamicFlight(string fn, string dest, string dep, int seats, double price)
        : Flight(fn, dest, dep, seats, price) {}

    double calculateDynamicPrice() const override {
        double demandFactor = 1.0 + ((double)(totalSeats - availableSeats) / totalSeats);
        time_t now = time(0);
        tm* currentTime = localtime(&now);

        if (currentTime->tm_hour >= 19) demandFactor += 0.5;

        return basePrice * demandFactor;
    }
};

template <class T>
class ReservationSystem {
private:
    map<string, T*> flights;
public:
    void addFlight(T* flight) {
        flights[flight->getFlightNumber()] = flight;
    }

    void showAllFlights() const {
        cout << "\nAvailable Flights:\n";
        for (const auto& [flightNum, flight] : flights) {
            flight->showFlightDetails();
            cout << "Price: $" << fixed << setprecision(2) << flight->calculateDynamicPrice() << "\n\n";
        }
    }

    void bookFlight(const string& flightNum, int seats) {
        if (flights.find(flightNum) != flights.end()) {
            T* flight = flights[flightNum];
            double pricePerSeat = flight->calculateDynamicPrice();
            if (flight->bookSeats(seats)) {
                cout << "\nBooking successful!\n";
                cout << "Flight Details:\n";
                flight->showFlightDetails();
                cout << "Seats Booked: " << seats << endl;
                cout << "Total Price: $" << fixed << setprecision(2) << pricePerSeat * seats << endl;
            } else {
                cout << "Sorry, not enough seats available on this flight." << endl;
            }
        } else {
            cout << "Flight not found!" << endl;
        }
    }
```

```
  ~ReservationSystem() {
    for (const auto& [flightNum, flight] : flights) {
      delete flight;
    }
  }
};

int main() {
  ReservationSystem<DynamicFlight> system;

  // Adding flights to the system
  system.addFlight(new DynamicFlight("AI101", "New York", "06:00", 100, 500.00));
  system.addFlight(new DynamicFlight("BA202", "London", "10:00", 80, 400.00));
  system.addFlight(new DynamicFlight("CA303", "Paris", "19:30", 50, 350.00));

  int choice;
  string flightNum;
  int seats;

  do {
    cout << "\nFlight Reservation System\n";
    cout << "1. Show All Flights\n";
    cout << "2. Book a Flight\n";
    cout << "3. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
    case 1:
      system.showAllFlights();
```

```
            break;
        case 2:
            cout << "Enter Flight Number to book: ";
            cin >> flightNum;
            cout << "Enter number of seats to book: ";
            cin >> seats;
            system.bookFlight(flightNum, seats);
            break;
        case 3:
            cout << "Exiting the system.\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 3);

    return 0;
}
```