Code-

```
 #include <iostream>
using namespace std;
struct Passenger {
    string name;
    int age;
};
struct Node {
    Passenger passenger;
    Node* next;
};
struct Queue {
    Node* front;
    Node* rear;
    int size;
    Queue() {
        front = nullptr;
        rear = nullptr;
        size = 0;
    }

    void enqueue(const string& name, int age) {
        Node* newNode = new Node;
        newNode->passenger.name = name;
```

```
    newNode->passenger.age = age;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = newNode;
        rear = newNode;
    } else {

        rear->next = newNode;
        rear = newNode;
    }
    size++;
}
void dequeue() {
    if (!isEmpty()) {
        Node* temp = front;
        front = front->next;
        delete temp;
        size--;
        if (isEmpty()) {
            rear = nullptr;
        }
    }
}

bool isEmpty() {
    return size == 0;
```

```
    }
    void clear() {
        while (!isEmpty()) {
            dequeue();
        } }
};
struct Stack {
    Node* top;
    int size;
    Stack() {
        top = nullptr;
        size = 0;
    }
    void push(const string& name, int age) {
        Node* newNode = new Node;
        newNode->passenger.name = name;
        newNode->passenger.age = age;
        newNode->next = top;
        top = newNode;
        size++;
    }
    void pop() {
        if (!isEmpty()) {
            Node* temp = top;
            top = top->next;
            delete temp;
```

```
            size--;
        }}
    bool isEmpty() {
        return size == 0;
    }
    void clear() {
        while (!isEmpty()) {
            pop();
        }
    }
};
struct Flight {
    int flightNumber;
    string origin;
    string destination;
    string departureTime;
    string arrivalTime;
    int capacity;
    int bookedSeats;
    Queue waitingList; // Using queue to manage waiting list
    Stack cancellationStack; // Using stack to manage cancellations
};
void initializeFlight(Flight& flight) {

    cout << "Enter Origin: "; cin >> flight.origin;
    cout << "Enter Destination: "; cin >> flight.destination;
```

```
    cout << "Enter Departure Time: "; cin >> flight.departureTime;
    cout << "Enter Arrival Time: "; cin >> flight.arrivalTime;
    cout << "Enter Capacity: "; cin >> flight.capacity;
    flight.bookedSeats = 0;
}
void addPassenger(Flight& flight, const string& name, int age) {
    if (flight.bookedSeats < flight.capacity) {
        flight.bookedSeats++;
        cout << "**************************************************" << endl;
        cout << "Seat booked successfully!" << endl;
        cout << "Flight number - 69" << endl;
        cout << name << " " << age << " " << "years" << endl;
        cout << flight.origin << " " << "TO" << " " << flight.destination << endl;
        cout << "Departure time" <<  " " << flight.departureTime <<endl;
        cout << "Arrival time" <<  " " << flight.arrivalTime <<endl;
        cout << "**************************************************" << endl;
    } else {
        cout << "Flight is full. Adding to waiting list..." << endl;
        flight.waitingList.enqueue(name, age); // Add to waiting list
    }
}
void cancelBooking(Flight& flight) {
    if (!flight.waitingList.isEmpty()) {
        flight.waitingList.dequeue();
        cout << "Booking cancelled successfully!" << endl;
    } else {
```

```
        cout << "No passengers in waiting list." << endl;
    }
}
void processCancellations(Flight& flight) {
    if (!flight.cancellationStack.isEmpty()) {
        flight.cancellationStack.pop();
        cout << "Cancellation processed successfully!" << endl;
    } else {
        cout << "No cancellations to process." << endl;
    }
}
int main() {
    Flight flight1;
    cout << "Enter Flight Details:" << endl;
    initializeFlight(flight1);
    cout << "Enter number of passengers: ";
    int numPassengers;
    cin >> numPassengers;
    for (int i = 0; i < numPassengers; ++i) {
        string name;
        int age;
        cout << "Enter Passenger " << i + 1 << " Name: "; cin >> name;
        cout << "Enter Passenger " << i + 1 << " Age: "; cin >> age;
        addPassenger(flight1, name, age);
    }
    cancelBooking(flight1);
```

```
    processCancellations(flight1);
cout << "**THANK YOU FOR VISITING GURJAR AIRLINES**";
    return 0;
}
```