**Problem Statement**

Customers seeking to book tickets for a flight mainly require two things:-

1) Ease of booking through the reservation system, and

2) A ticket at the cheapest possible cost.

If interfaces of airline reservation systems are tricky to handle, it can cause complications in the booking process and result in dissatisfaction of the customer.

Furthermore, if a customer is unable to find the best possible deal for a ticket through the system, they might not go through with the ticket purchase.

The aim is to provide for every customer to purchase not more than one ticket for themselves, on the flight with the lowest ticket price for their route. This is provided there are seats available on the specific flight. Or else, the flight with the next lowest price and available seats is chosen.

**Aging**

|  |  |  |
| --- | --- | --- |
| **Past** | **Present** | **Future** |
| Airline reservations systems were first implemented in the late 1950s, when American Airlines required a system that would allow real-time access to flight details in all of its offices, and the integration and automation of its booking and ticketing processes.    As a result, Sabre was developed and launched in 1964. Sabre's breakthrough was its ability to keep inventory correct in real time, accessible to agents around the world.    Prior to this, manual systems required centralized reservation centers, groups of people in a room with the physical cards that represented inventory, in this case, seats on airplanes. | Airline reservation system eventually evolved into the Computer Reservation System(CRS).     Computer reservation systems are used for the reservation of a particular airline and interface with a Global Distribution System (GDS) which travel agency and other distribution channels in making reservations for most major airlines in a single system. | Air travel has become a norm over recent years, and this trend doesn’t show any signs of changing soon. As airlines strive to provide cheaper fare prices, air travel will continue to flourish, and the need to provide more efficient airline reservation systems will only become more eminent. |

**Costing**

A basic flight reservation system with an API integration usually costs between 6K to 9K USD. Upper-end airline reservation systems come with an XML integration, which is ideally provided by some API or GDS (Global Distribution System) consolidators.

API/GDS is a network operated by a company that enables automated transactions between third parties and booking agents in order to provide travel-related services to the end consumers. For a booking engine with GDS integration, say Amadeus, you can add 25% to above cost.

**Requirements**

## HIGH LEVEL REQUIREMENTS

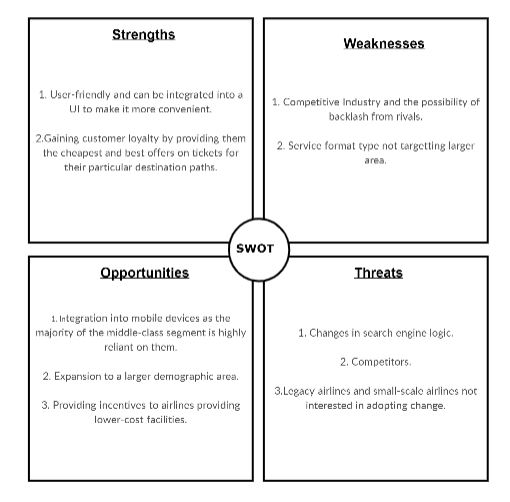
|  |  |
| --- | --- |
| **ID** | **Description** |
| HH\_01 | Book Passenger |
| HH\_02 | Cancel Passenger |
| HH\_03 | Change Price |
| HH\_04 | Recover Seat |
| HH\_05 | Get Available Seats |
| HH\_06 | Flight is full |
| HH\_07 | Initialize Flight |
| HH\_08 | Compare prices for two flight to which one is cheaper |

Table 1 : High Level Requirement

## 2. LOW LEVEL REQUIREMENTS

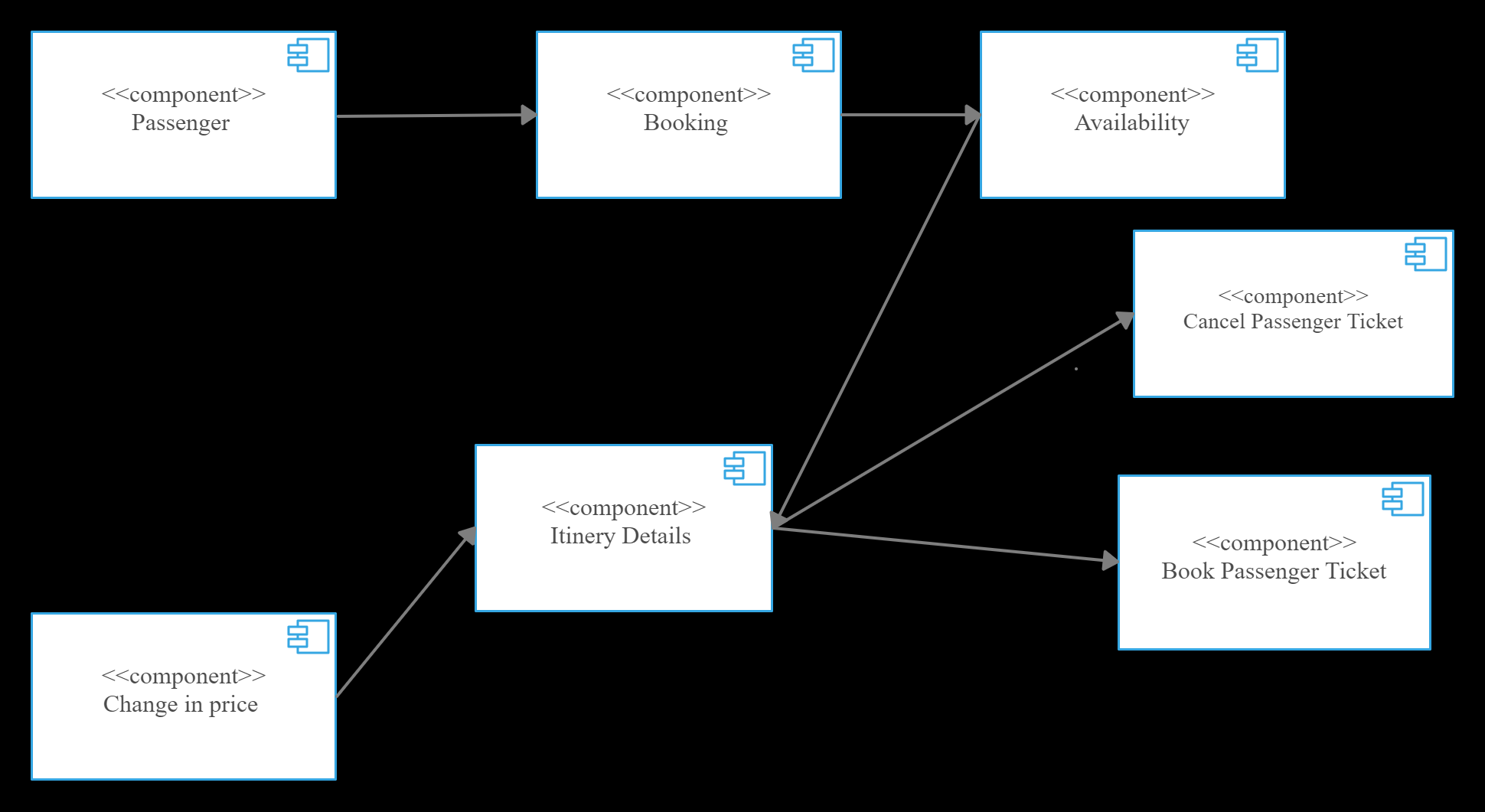
|  |  |
| --- | --- |
| **ID** | **Description** |
| LL\_01 | Create a .txt file for creating dataset for flight details |
| LL\_02 | Create a .txt file for creating dataset for passenger details to book or cancel reservations |
| LL\_03 | Ensure that FlightReservationSystem.class is able to access input1.txt file |
| LL\_04 | Ensure that FlightReservationSystem.class is able to access input2.txt file |
| LL\_05 | Ensure that output file is generated |

Table 2 : Low Level Requirement

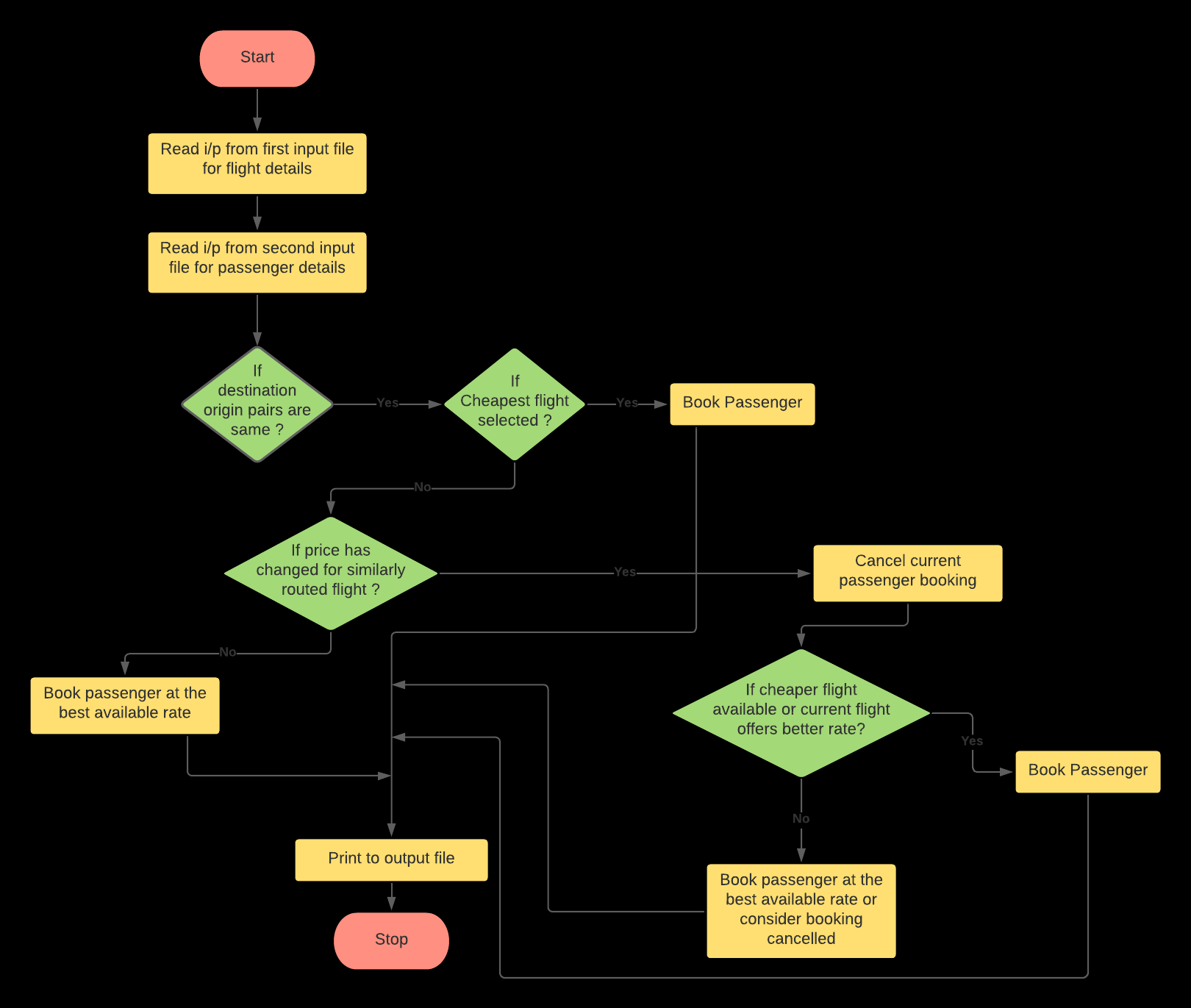


**Figure: SWOT Analysis**

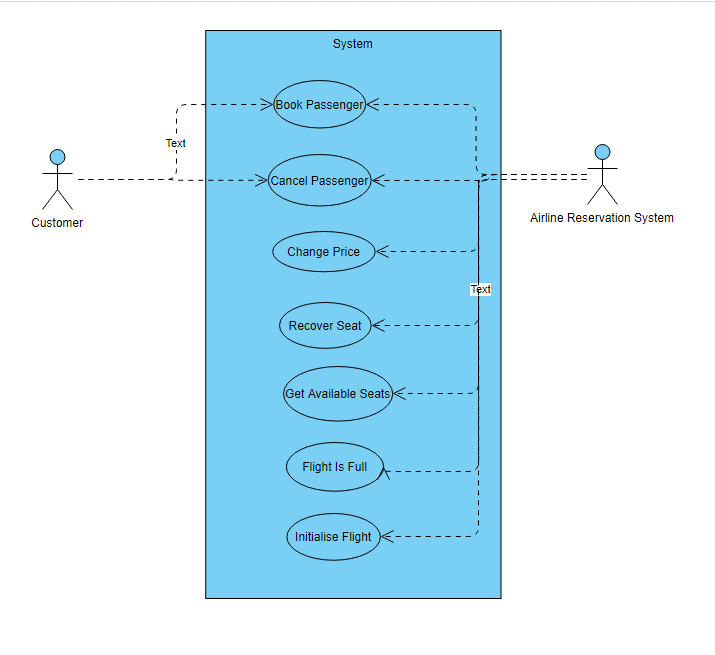
**Designs**



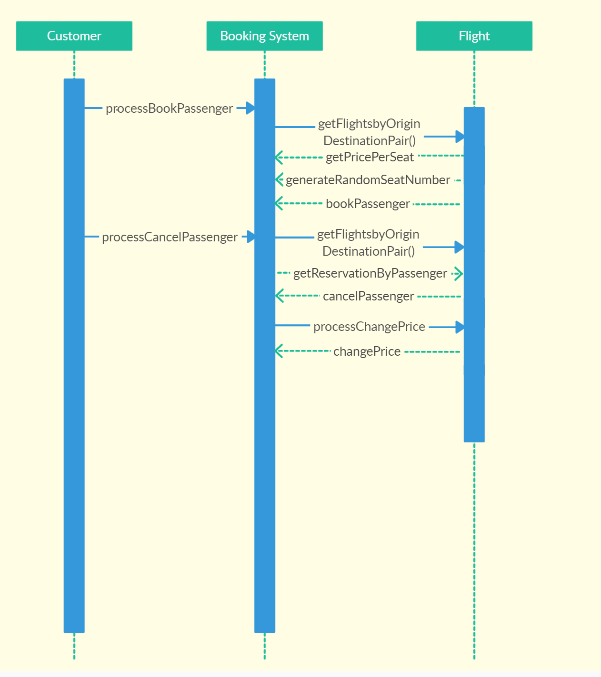
**Figure: Component Diagram**



**Figure: Flowchart**



**Figure: Use Case diagram**



**Figure: Sequence Diagram**

**Test Planning**

1. Unit testing:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Requirement mapping** | **Description** | **Expected input** | **Expected output** | **Actual output** |
| IT\_01 | HH\_01 | Book Passenger | Passenger Details | Client id, phone number, name, address |  |
| IT\_02 | HH\_02 | Cancel Passenger | Passenger Details | Client id, phone number, name, address |  |
| IT\_03 | HH\_03 | Change Price | New Price | Updated price |  |
| IT\_04 | HH\_04 | Recover Seat | seatNumber | seatNumber |  |
| IT\_05 | HH\_05 | Get Available Seats |  | Availability status |  |
| IT\_06 | HH\_06 | Flight is full | Trying to reserver for a new passengers | No seats available |  |
| IT\_07 | HH\_07 | Initialize Flight | Flight Details | New flight initialized |  |
| IT\_08 | HH\_08 | Compare prices for two flight to which one is cheaper | Flight details for both flights | Show flight details for the cheaper flight |  |

Table 1: Unit Testing

2. Integration testing:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Requirement mapping** | **Description** | **Expected input** | **Expected output** | **Actual output** |
| UT\_01 | LL\_1 | Create a .txt file for creating dataset for flight details | FLIGHT\_NUMBER,  NUMBER\_OF\_SEATS,  PRICE\_PER\_SEAT,  ORIGIN,  DESTINATION | Input1.txt file |  |
| UT\_02 | LL\_2 | Create a .txt file for creating dataset for passenger details to book or cancel reservations | Booking details | Input2.txt file |  |
| UT\_03 | LL\_03 | Ensure that FlightReservationSystem.class is able to access input1.txt file | FLIGHT\_NUMBER,  NUMBER\_OF\_SEATS,  PRICE\_PER\_SEAT,  ORIGIN,  DESTINATION | Input1.txt file |  |
| UT\_04 | LL\_04 | Ensure that FlightReservationSystem.class is able to access input2.txt file | Booking details | Input2.txt file |  |
| UT\_05 | UT\_05 | Ensure that output file is generated | Input2.txt | Output.txt Status |  |

Table 2: Integration testing