|  |  |
| --- | --- |
|  | Test DOCUMENT  **Team04 Jing Yang, Fu Zhan Jiang Miao, Bin Yu** |

**Content**

[Object model (updated) 1](#_Toc418119374)

[Test Plan 2](#_Toc418119375)

[Test Approaches: 2](#_Toc418119376)

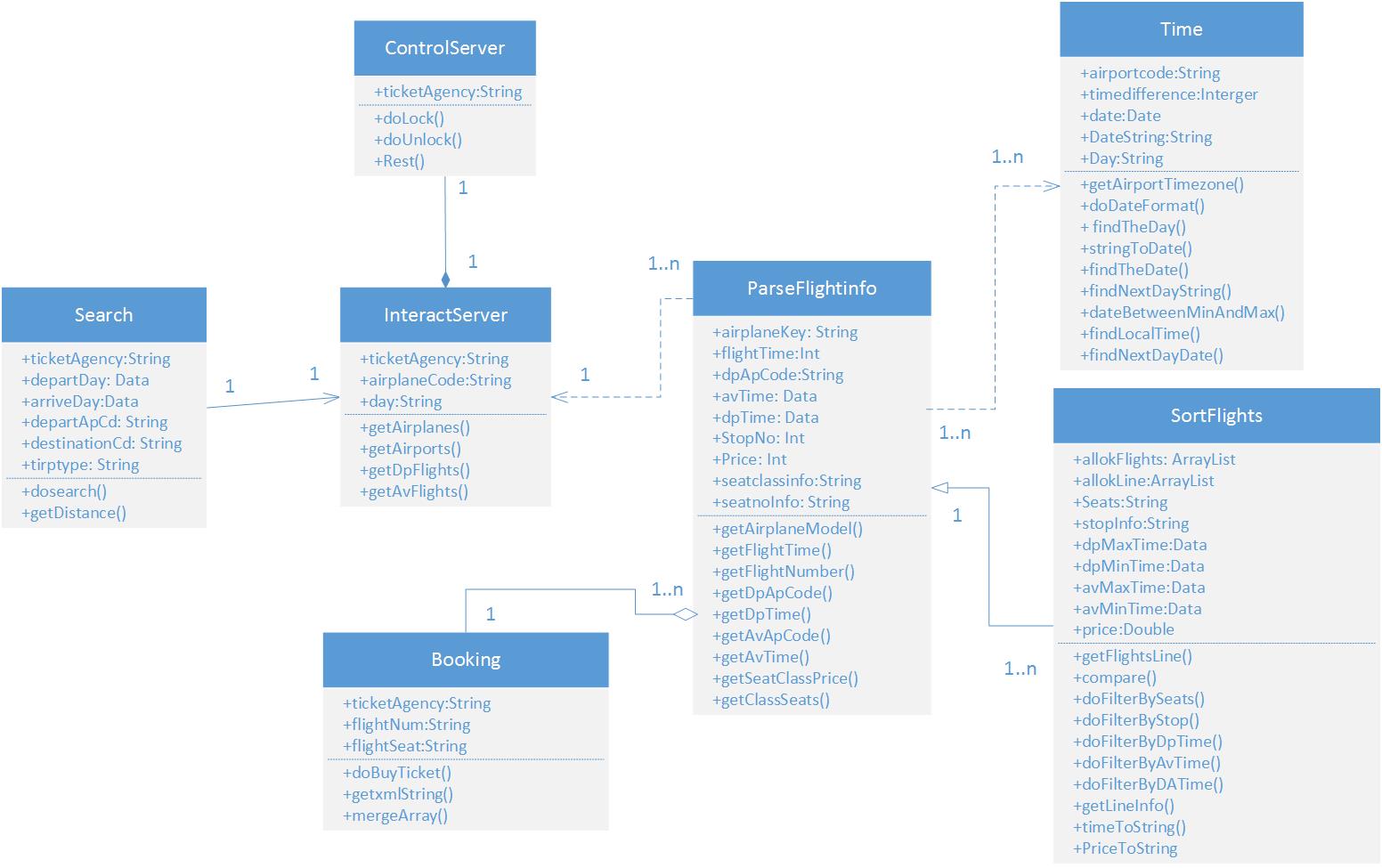
[Unit & Component Test Plan: 2](#_Toc418119377)

[Integration Test Plan: 2](#_Toc418119378)

[Test Result 3](#_Toc418119379)

[Integration Test 37](#_Toc418119380)

# Object model (updated)



**Search**

Do basic operation to submit the search information to server.

**InteractServer:**

Exchange data with the server to get all information from database.

**ControlServer:**

Basic operation to control database, ensure the user will not be disturbed by other users.

**ParseFlightinfo:**

Parse the information from the server, display information which confirm to the search condition

**Time:**

Control all the information which are related to the time.

**SortFlights:**

Sort all the parsing flight information in different ways.

**Booking:**

Submit the requirement to database to ensure orders have already been received.

# Test Plan

## Test Approaches:

Manual Testing, Boundary Testing

## Unit & Component Test Plan:

We are going to test almost every java class in our workspace to make sure every single unit works well. These unit testing including testing of interacting with the back-end database, parsing the xml string requested from database, searching flights, managing data of connecting flights, sorting and filtering flights, converting GMT time to airport’s local time.

We tested two cases to lock/unlock/buyTicket. One case is the database is idle, the other case is when database has already be locked by other teams. In both cases, our system works well.

In all this test cases, we consider testing of search function as the most crucial to our system. So we are going to test our search function in detail, including normal case when customer specify a valid day, border case of travel day, the special case when customer want to travel on 2015\_5\_7 (which seems an invalid travel day to our system at first glance, but actually it is a special border case) and invalid case when customer specifies an invalid day. We also test the case of red-eye flight. Another case is when departing airport and destination airport are very far from each other, and there are so many connecting flights we need to request from database server, in this case our system was extremely slow, we discovered this issue during testing and have fixed it up.

## Integration Test Plan:

We are going to test if customers can search one-way and round-trip flights, then test if they can book flights.

When testing search function, we considered normal case and error case (when there is no flight found). Our system is implemented in the way to minimize customers’ chances to specify invalid inputs (such as invalid travel day, invalid airport).

When customer specifies “first class”, but there are no “first class” flights available or maybe not all the legs have “first-class” seats, in these cases, our system informs customers that we still find some flights without “first-class” seats, displays flights with “first class” seats and without “first-class” seats separately, customer can decide if they want to change their mind or not.

When testing book function, we tested the case when customers want to buy a one-way direct flight, the case when they want to buy multiple flights (different legs from Airport A to Airport B), and the case when the server is locked by someone else.

# Test Result

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 01 | Test Case Name | GetAirplanesTest |
| Test Case Description | Connect with the server to verify that the system can get all the Airplanes information from the server. | | |
| Test Driver |  | | |
| Test Stub | AirplaneParseTest | | |
| Pre-conditions | System can get the XML document which contains all the airplanes information on the server | | |
| Input | ticketangency=“Team04”; | | |
| Expected Output | The server XML document of airplanes | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 02 | Test Case Name | GetAirportsTest |
| Test Case Description | Same like GetAirplanesTest, get all the Airports information from server. | | |
| Test Driver |  | | |
| Test Stub | AirportParseTest | | |
| Pre-conditions | System can get the XML document which contains all the airports information on the server | | |
| Input | ticketangency=“Team04”; | | |
| Expected Output | The server XML document of airports | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 03 | Test Case Name | GetAvFlightsTest |
| Test Case Description | Connect with the server to verify that the system can get all the Arrival Flights information on the specific airport and the date from the server. | | |
| Test Driver |  | | |
| Test Stub | FlightParseTest | | |
| Pre-conditions | System can get the flight XML document from the server. | | |
| Input | ticketangency=“Team04”; airplaneCode = "PHL"; day= "2015\_05\_09"; | | |
| Expected Output | The related server XML document of arrival flights | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 04 | Test Case Name | GetDpFlightsTest |
| Test Case Description | Connect with the server to verify that the system can get all the Departure Flights information on the specific airport and the date from the database. | | |
| Test Driver |  | | |
| Test Stub | FlightParseTest  buyTicketTest  SearchTest  FilterTest  SortTest | | |
| Pre-conditions | System can get the flight XML document from the server. | | |
| Input | ticketangency=“Team04”; airplaneCode = "JFK"; day= "2015\_05\_17"; | | |
| Expected Output | The related server XML document of departure flights | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 05 | Test Case Name | AirplaneParseTest |
| Test Case Description | Inspect whether or not the system can parse the Airplane XML document which sending from the server. if it can be then try to display all the airplanes information which have already been saved in hash map. | | |
| Test Driver | GetAirplanesTest | | |
| Test Stub |  | | |
| Pre-conditions | The system has already built the Hashmap to save the Airplanes information. | | |
| Input | Airplane Model="A380" | | |
| Expected Output | Specific information of airplane”A380” | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 06 | Test Case Name | AirportParseTest |
| Test Case Description | Inspect whether or not the system can parse the Airports XML document which sending from the server, if it can be then try to display all the airports information which have already been saved in array list. | | |
| Test Driver | GetAirportsTest | | |
| Test Stub |  | | |
| Pre-conditions | The system has already built the Arraylist to save the Airports information. | | |
| Input | ticketAgency = "Team04"; | | |
| Expected Output | All the airports information in array list. | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 07 | Test Case Name | AirportParse2Test |
| Test Case Description | Inspect whether or not the system can parse the Airpors XML document which sending from the server, if it can be then try to display all the airports information which have already been saved in Hashmap. | | |
| Test Driver | GetAirportsTest | | |
| Test Stub |  | | |
| Pre-conditions | The system has already built the Hashmap to save the Airports information. | | |
| Input | ticketAgency = "Team04"; | | |
| Expected Output | All the airports information in hash map | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 08 | Test Case Name | FlightParseTest |
| Test Case Description | Inspect whether or not the system can parse the Flight XML document and search the specific flight of user’s requirements. | | |
| Test Driver | GetDpFlightsTest/GetAvFlightsTest  TimeRelatedTester\_findLocalTimeTest | | |
| Test Stub |  | | |
| Pre-conditions | The system has already built the Arraylist to save the Flight information. | | |
| Input | ticketAgency = "Team04"; airportCode = "JFK"; day = "2015\_05\_13"; | | |
| Expected Output | The specific airports information in array list. | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 09 | Test Case Name | lockTest |
| Test Case Description | Inspect the database situation, if two users access the database at the same time, once a user has locked the database, the other one cannot lock it until waiting two minutes. | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions | System can interact with the server to get the situation of database | | |
| Input | ticketangency=“Team04”; ticketangency=“Team03”; | | |
| Expected Output | “Team03” cannot lock the database, server response the error code(417) | | |
| Actual Output1 |  | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 10 | Test Case Name | lockTest2 |
| Test Case Description | The same user can repeat locking the database. | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions | System can interact with the server to get the situation of database | | |
| Input | ticketangency=“Team04”; ticketangency=“Team04”; | | |
| Expected Output | “Team04” can still lock the database, server response the right code(202) | | |
| Actual Output 2 |  | | |
| Test Result 2 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 11 | Test Case Name | buyTicketTest\_getxmlStringTest |
| Test Case Description | Give a requirement on a specific flight and seats kind to the server, then compare the result with input to verify the booking information is correct. | | |
| Test Driver |  | | |
| Test Stub | buyTicketTest | | |
| Pre-conditions | System can get xml information from the server. | | |
| Input | Flight number: “12100","12950";  Flight seat: "FirstClass","Coach" | | |
| Expected Output | Flight number="12100" seating="FirstClass"; Flight number="12950" seating="Coach" | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 12 | Test Case Name | buyTicketTest |
| Test Case Description | Give a requirement on a specific flight and seats kind to the server, then buy the specific flight with input information | | |
| Test Driver | interactWithServer.lock  interactWithServer.unlock  buyTicketTest\_getxmlStringTest | | |
| Test Stub |  | | |
| Pre-conditions | The database situation is unlock | | |
| Input | Flight number: “12100","12950";Flight seat: "FirstClass","Coach"  Flight “12100, Coach” | | |
| Expected Output | Booking specific flight success | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 13 | Test Case Name | unlockTest |
| Test Case Description | Inspect the database situation, change it from lock to unlock so that the database can receive the requirement from the users. | | |
| Test Driver | lockTest | | |
| Test Stub |  | | |
| Pre-conditions | Two testers, one make the database becoming lock and the next one try to unlock the database. | | |
| Input | doUnlockTest1()\_\_lock; doUnlockTest2()\_\_unlock; | | |
| Expected Output | The situation of database is unlock, which can be normal exchanged data. | | |
| Actual Output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 14 | Test Case Name | AirportTimeZoneTest |
| Test Case Description | For all airports in the back-end database,  get the time differences between local time of the airport and GMT time | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input |  | | |
| Expected Output | time differences between local time of airports and GMT time | | |
| Actual Output1 |  | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 15 | Test Case Name | TimeRelatedTester\_findTheDayTest |
| Test Case Description | Given a Date like "2015\_05\_02 11:30:00", find the String of the day "2015\_5\_2" | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date input = "2015\_05\_02 11:30:00" | | |
| Test result |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 16 | Test Case Name | TimeRelatedTester\_stringToDateTest |
| Test Case Description | convert string like "13:58 5/13/2015" to date | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | String input ="13:58 5/5/2015" | | |
| Expected output | Tue May 05 13:58:00 EDT 2015 | | |
| Actual output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 17 | Test Case Name | TimeRelatedTester\_findTheDateTest |
| Test Case Description | convert string like "2015\_5\_13" to date "2015\_05\_13 00:00" | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | String input = "2015\_5\_13"; | | |
| Expect result | Wed May 13 00:00:00 2015 | | |
| Actual result |  | | |
| Test result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 18 | Test Case Name | TimeRelatedTester\_findTheNextDayTest |
| Test Case Description | Given a string of day like "2015\_5\_17", find the string of next day "2015\_5\_18" | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | String input = "2015\_5\_2"; | | |
| Test result |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 19 | Test Case Name | TimeRelatedTester\_dateFormatTest |
| Test Case Description | Convert a date to string in the format of "HH:mm MM/dd/yyyy" | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date input = "2015\_03\_27 23:39:00" | | |
| Test result |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 20 | Test Case Name | TimeRelatedTester\_dateBetweenMinAndMax |
| Test Case Description | Find the list of date between two dates | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date datemin = 2015\_3\_27 23:39:00  Date datemax = 2015\_3\_28 10:46:00 | | |
| Expect output | All date string between 3/27 23:00 and 3/28 11:00 | | |
| Actual output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 21 | Test Case Name | TimeRelatedTester\_findNextTimeTest |
| Test Case Description | Given a time, find the time which is certain minutes later than the time | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date inputdate = 2015\_3\_27 23:39:00  int inputminutes = 40 | | |
| Expect output | 2015\_3\_28 00:19:00 | | |
| Actual output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 22 | Test Case Name | TimeRelatedTester\_findLocalTimeTest |
| Test Case Description | Given a date in GMT format and the airport Code, get the date in local time format | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date inputdate = 2015\_3\_27 23:39  String inputAirport = "BOS"; | | |
| Expect output | 2015\_3\_27 19:39:00 | | |
| Actual output |  | | |
| Test result | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 23 | Test Case Name | TimeRelatedTester\_doCompareTest |
| Test Case Description | Compare two dates. One of the date is specified by "date1", the other date is on the same day with date1, but its hour and minutes are specified by integer "hours" and "minutes". | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date inputdate = 2015\_3\_27 23:39  **int** inputhours = 3;  **int** inputminutes = 25; | | |
| Test result |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 24 | Test Case Name | TimeRelatedTester\_subtractTwoDateTest |
| Test Case Description | Subtract two dates, get the time difference in minutes | | |
| Test Driver |  | | |
| Test Stub |  | | |
| Pre-conditions |  | | |
| Input | Date input1 = 2015\_3\_27 23:39  Date input2 = 2015\_3\_28 10:46 | | |
| Test result |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 25 | Test Case Name | SearchTest\_findFlightsOnDayTest\_\_normal case |
| Test Case Description | Get all Flights departing from “Depart Airport” on a specific day which is valid to our database. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest | | |
| Test Stub | SearchTest\_doSearchTest\_normal case  SearchTest\_doSearchTest\_red eye  SearchTest\_doSearchTest\_time out  FilterTest  SortTest | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Depart Airport Code: JFK  Depart Day: 2015\_05\_15 | | |
| Expected Output | All possible flights from JFK to MCO on 2015\_05\_15 | | |
| Actual Output1 |  | | |
| Test result1 | fail | | |
|  |  | | |
| Actual Output 2 |  | | |
| Test Result 2 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 26 | Test Case Name | SearchTest\_findFlightsOnDayTest\_\_border case1 |
| Test Case Description | Get all Flights departing from “Depart Airport” on a specific day which is a border day of our database. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest | | |
| Test Stub | FilterTest  SortTest | | |
| Pre-conditions | 1. “Depart airport”, “arrive airport” specified by customer are valid to our back-end database;  2. “departDay” specified by customer is one of the border day of our database; | | |
| Input | Depart Airport Code: JFK  Depart Day: 2015\_5\_8 | | |
| Expected Output | All possible flights from JFK to MCO on 2015\_05\_08 | | |
| Actual Output1 |  | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 27 | Test Case Name | SearchTest\_findFlightsOnDayTest\_\_border case2 |
| Test Case Description | Get all Flights departing from “Depart Airport” on a specific day which is a border day of our database. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest | | |
| Test Stub |  | | |
| Pre-conditions | 1. “Depart airport”, “arrive airport” specified by customer are valid to our back-end database;  2. “departDay” specified by customer is one of the border day of our database; | | |
| Input | Depart Airport Code: JFK  Depart Day: 2015\_5\_17 | | |
| Expected Output | No flights departing from JFK on 2015\_5\_17 (local time of JFK) | | |
| Actual Output1 |  | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 28 | Test Case Name | SearchTest\_findFlightsOnDayTest\_\_special case |
| Test Case Description | Get all Flights departing from “Depart Airport” on a specific day which is one day before our border case, that is “2015\_5\_7” in local time of the depart airport.  This means customer want to depart from certain airport between local time 2015\_5\_7 00:00 and local time 2015\_5\_7 24:00, that is GMT time 2015\_5\_7 offset:00 and GMT time 2015\_5\_8 offset:00.  So, at first glance, 2015\_5\_7 is invalid to our database, but if converting it to GMT time, we still can find flights that can satisfied customer’s requirement. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest | | |
| Test Stub | SearchTest\_doSearchTest\_special case | | |
| Pre-conditions | 1. “Depart airport”, “arrive airport” specified by customer are valid to our back-end database  2. “departDay” is “2015\_5\_7” in local time of the departairport | | |
| Input | Depart Airport Code: JFK  Depart Day: 2015\_5\_7 | | |
| Expected Output | All flights departing from “JFK” on the local day of “2015\_5\_7” | | |
| Actual Output1 | …… | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 29 | Test Case Name | SearchTest\_findFlightsOnDayTest\_\_invalid case |
| Test Case Description | Test if the program respond correctly on an invalid day | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest | | |
| Test Stub | SearchTest\_doSearchTest\_invalid case | | |
| Pre-conditions | 1. “Depart airport”, “arrive airport” specified by customer are valid to our back-end database  2. “departDay” is an invalid day to our database | | |
| Input | Depart Airport Code: JFK  Depart Day: 2015\_5\_19 | | |
| Expected Output | No flights can be found | | |
| Actual Output1 |  | | |
| Test result1 | success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 30 | Test Case Name | SearchTest\_doSearchTest\_normal case |
| Test Case Description | Search Flights from “Depart Airport” to “Arrive Airport” on a specific day after a specific time | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case | | |
| Test Stub | manageLineDataTest  FilterTest  SortTest | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “MCO”  Depart Day: String departDay: “2015\_5\_15”  Desire Depart Time: Date desireDpTime: 2015\_5\_15 00:00 | | |
| Expected Output | All possible flights from JFK to MCO on 2015\_05\_15 | | |
| Actual Output |  | | |
| Test Result | Fail | | |
| Fail analysis |  | | |
| Fail fix |  | | |
| Test Output |  | | |
| Test Result 2 | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 31 | Test Case Name | SearchTest\_doSearchTest\_special case |
| Test Case Description | Search Flights from “Depart Airport” to “Arrive Airport” on a specific day after a specific time | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_special case | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “IAH”  Depart Day: String departDay: “2015\_5\_7”  Desire Depart Time: Date desireDpTime: 2015\_5\_15 00:00 | | |
| Expected Output | All possible flights from JFK to IAH on 2015\_5\_07 | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 32 | Test Case Name | SearchTest\_doSearchTest\_invalid case |
| Test Case Description | Search Flights from “Depart Airport” to “Arrive Airport” on a specific day after a specific time | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_invalid case | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “IAH”  Depart Day: String departDay: “2015\_5\_19”  Desire Depart Time: Date desireDpTime: 2015\_5\_19 00:00 | | |
| Expected Output | No flights can be found | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 33 | Test Case Name | SearchTest\_doSearchTest\_red eye |
| Test Case Description | Search Flights from “Depart Airport” to “Arrive Airport” on a specific day after a specific time | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd = "JFK";  Code of Arrive Airport: String destinationCd = "MCO";  Depart Day: String departDay = "2015\_5\_15";  Desire Depart Time: Date desireDpTime: 2015\_5\_15 00:00  Desire Seat Class: String seat = “Coach” | | |
| Expected Output |  | | |
| Actual Output1  (without considering red eye flight) |  | | |
| Actual Output2  (considering red eye flight) | **When considering red eye, we have one more result shown as following which takes off from MSY to MCO at 22:47:00.(local time of MSY)** | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 34 | Test Case Name | SearchTest\_doSearchTest\_time out |
| Test Case Description | Search Flights from “Depart Airport” to “Arrive Airport” on a specific day after a specific time | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd = "JFK";  Code of Arrive Airport: String destinationCd = "SFO";  Depart Day: String departDay = "2015\_5\_13";  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00  Desire Seat Class: String seat = “Coach” | | |
| Expected Output |  | | |
| Actual output1 | Cannot retrieve searched results in 5 minutes | | |
| Test Result1 | Fail | | |
| Actual Output2 | 29 flights from JFK to SFO was found. It took about 110 seconds for system to give back the search results. | | |
| Test Result2 | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 35 | Test Case Name | manageLineDataTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), for each line, get total flight duration, total price, take off time and landing time, flight of each leg, layover time of each leg. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case | | |
| Test Stub | SearchTest\_doSearchTest\_normal case  FilterTest  SortTest | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | For each flights from JFK to PHL on 2015\_5\_13,  duration, price, take off time, landing time, flights of each leg, layover time are displayed. | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 36 | Test Case Name | FilterTest\_doFilterBySeatsTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), filter the list according to specified seat class, the first list is flights in which each leg has specified seats available, the second list is flights in which some legs may don’t have specified seats available. | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | Two lists, the first is flights from JFK to PHL on 2015\_5\_13 with first-class seats available, the second is flights from JFK to PHL on 2015\_5\_13 without first-class seats available | | |
| Actual Output | Flights with first-class seats:      Flights without first-class seats | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 37 | Test Case Name | FilterTest\_doFilterByStopTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), filter the list according to specified stop information | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00  Specified Stop information: String stopInfo: “NonStopOneStop” | | |
| Expected Output | The same output of “manageLineDataTest” except the 2-stop flights | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 38 | Test Case Name | FilterTest\_doFilterByDATimeTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), filter the list according to specified time window | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00  The earliest taking off time: Date dpMinTime: 2015\_5\_13 00:00  The latest landing timeDate dpMaxTime: 2015\_5\_13 12:00  The earliest landing time: Date avMinTime: 2015\_5\_13 12:00  The latest landing time: Date avMaxTime: 2015\_5\_13 24:00 | | |
| Expected Output | The flights taking-off from “JFK” between 2015\_5\_13 00:00 and 2015\_5\_13 12:00, landing at “PHL” between 2015\_5\_13 12:00 and 2015\_5\_13 24:00 | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 39 | Test Case Name | SortTest\_sortByDurationTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), sort the list according to their fly duration (shorter to longer) | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | The flights taking-off from “JFK” to “PHL” on 2015\_5\_13 are managed according to their flight duration time shorter to longer | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 40 | Test Case Name | SortTest\_sortByDurationTest2 |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), sort the list according to their fly duration (longer to shorter) | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | The flights taking-off from “JFK” to “PHL” on 2015\_5\_13 are managed according to their flight duration time longer to shorter | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 41 | Test Case Name | SortTest\_sortByPriceTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), sort the list according to their price (lower to higher) | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | The flights taking-off from “JFK” to “PHL” on 2015\_5\_13 are managed according to their flight price lower to higher | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 42 | Test Case Name | SortTest\_sortByTakeoffTimeTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), sort the list according to their taking off time earlier to later | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | The flights taking-off from “JFK” to “PHL” on 2015\_5\_13 are managed according to their taking off time earlier to later | | |
| Actual Output |  | | |
| Test Result | Success | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | 43 | Test Case Name | SortTest\_sortByLandingTimeTest |
| Test Case Description | Given a list of flights from airport A to airport B (including direct and connecting flights), sort the list according to their landing time earlier to later | | |
| Test Driver | TimeRelatedTester\_findGMTTimeTest  TimeRelatedTester\_findTheDayTest  GetDpFlightsTest  FlightParseTest  GetAirportsTest  AirportParse2Test  SearchTest\_findFlightsOnDayTest\_normal case  SearchTest\_doSearchTest\_normal case  manageLineDataTest | | |
| Test Stub |  | | |
| Pre-conditions | “Depart airport”, “arrive airport” and “depart day” specified by customer are valid to our back-end database | | |
| Input | Code of Depart Airport: String sourceApCd: “JFK”  Code of Arrive Airport: String destinationApCd: “PHL”  Depart Day: String departDay: “2015\_5\_13”  Desire Depart Time: Date desireDpTime: 2015\_5\_13 00:00 | | |
| Expected Output | The flights taking-off from “JFK” to “PHL” on 2015\_5\_13 are managed according to their landing time earlier to later | | |
| Actual Output |  | | |
| Test Result | Success | | |

# Integration Test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Name/Number | | OneWay\_normal case | | | |
| Test Case Description | | Display flights from Departure to Arrival Airport | | | |
| Function / Module Under Test | | Flight Display | | | |
| Test Requirement | | They can specify the airport they depart from and the airport they arrive at. | | User Case | SearchFlight |
| Goals | | Verify the software can retrieve and display flights(one way) between departing and arrival airport | | | |
| Setup for Test | |  | | | |
| Pre-conditions | | flights exist between departure and arrival airports on specified date | | | |
| Step | Action | | Expected Results | Pass / Fail | Actual Results |
| 1 | Specify JFK as departure airport | | System accepts JFK as departure and displays “JFK(John F. Kennedy International)” in the “Going From” text window | Pass |  |
| 2 | Specify MCO as Arrival airport | | System accepts MCO as arrival and displays “MCO(Orlando International)” in the “Going To” text window | Pass |  |
| 3 | Specify 2015\_5\_13 as day of travel | | System accepts specified date and “2015\_5\_13 “displayed in text of “Departing on” window | Pass |  |
| 4 | Specify “First-class” seat | | System accepts specified seat class and displays “First-class” in text of “Departing on” window | Pass |  |
| 5 | Click “Search” button | | System transitions to flight display window and shows all flights from BOS to MCO on 13 May 2015 | pass | See Fig 1 |

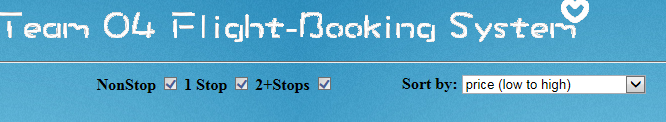






Fig 1---Part of Test results of Integration Test “OneWay\_normal case”

(Our system also recommends some extra flights without desired seat class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Name | | OneWay\_error case | | | |
| Test Case Description | | When no flights can be found, informs customer cannot find desired flights | | | |
| Function / Module Under Test | | Flight Display | | | |
| Test Requirement | | They can specify the airport they depart from and the airport they arrive at. | | User Case | SearchFlight |
| Goals | | Verify the software will handle situation when there are no flights (one way)for the specified day | | | |
| Setup for Test | |  | | | |
| Pre-conditions | | There are no flights for specified day in the database | | | |
| Step | Action | | Expected Results | Pass / Fail | Actual Results |
| 1 | Specify JFK as departure airport | | System accepts JFK as departure and displays “JFK(John F. Kennedy International)” in the “Going From” text window | Pass |  |
| 2 | Specify PHL as Arrival airport | | System accepts PHL as arrival and displays “PHL(Philadelphia International)” in the “Going To” text window | Pass |  |
| 3 | Specify 2015\_5\_9 as day of travel | | System accepts specified date and “2015\_5\_9 “displayed in text of “Departing on” window | Pass |  |
| 4 | Specify “First-class” seat | | System accepts specified seat class and displays “First-class” in text of “Departing on” window | Pass |  |
| 5 | Click “Search” button | | No flights are displayed and system displays dialog ‘’Oops, no flight found! Try another day!’’ | pass | See Fig 2 |



Fig 2---One way: fail to find flights

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Name | | RoundTrip\_normal case | | | |
| Test Case Description | | Display flights from Depart Airport to Arrival Airport, after customer choose an outbound flight, display returning flights from arrival airport to depart airport | | | |
| Function / Module Under Test | | Flight Display | | | |
| Test Requirement | | They can specify the airport they depart from and the airport they arrive at. | | User Case | SearchFlight |
| Goals | | Verify the software can retrieve and display all inbound and outbound flights(round trip) between departing and arrival airport | | | |
| Setup for Test | |  | | | |
| Pre-conditions | | Inbound and Outbound flights exist between departure and arrival airports on specified date | | | |
| Step | Action | | Expected Results | Pass / Fail | Actual Results |
| 1 | Specify JFK as departure airport | | System accepts JFK as departure and displays “JFK(John F. Kennedy International)” in the “Going From” text window | Pass |  |
| 2 | Specify MCO as Arrival airport | | System accepts PHL as arrival and displays “MCO(Orlando International)” in the “Going To” text window | Pass |  |
| 3 | Specify 2015\_5\_13 as departure day | | System accepts specified date and “2015\_5\_13 “displayed in text of “Departing on” window | Pass |  |
| 4 | Specify 2015\_5\_15 as return day | | System accepts specified date and “2015\_5\_15“displayed in text of “Returning on” window | pass |  |
| 5 | Specify “First-class” seat | | System accepts specified seat class and displays “First-class” in text of “Departing on” window | Pass |  |
| 6 | Click “Search” button | | System transitions to flight display window and shows all outbound flights from JFK to MCO on 13May 2015. | pass | See Fig 3.1 |
| 7 | After selecting the outbound flight, click “select” to choose inbound flight | | System transitions to flight display window and shows all outbound flights from MCO to JFK on 15 May 2015. | pass | See Fig 3.2 |

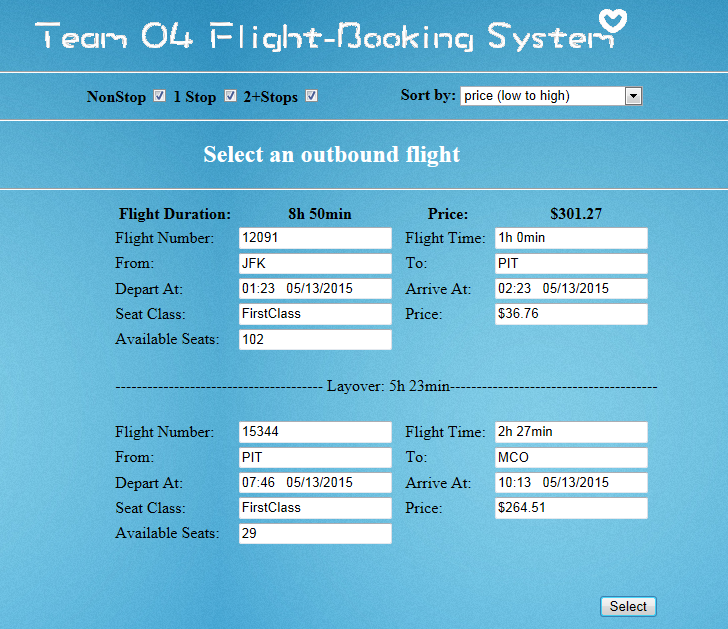


Fig 3.1--- RoundTrip\_normal case: success at finding an outbound flight

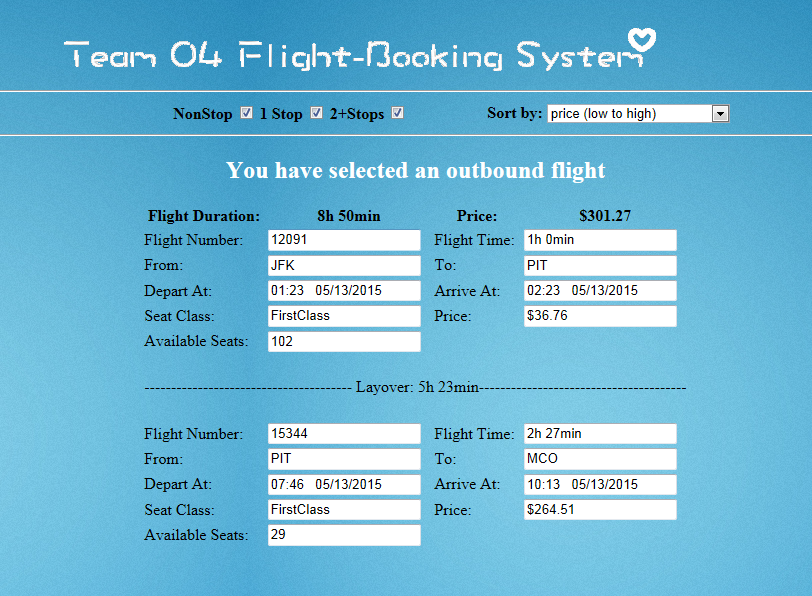




Fig 3.2--- RoundTrip\_normal case: success at finding an inbound flight

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case Name/Number | | RoundTrip\_error case1 | | |
| Test Case Description | | When no outbound flights can be found, informs customer cannot find outbound flights | | |
| Function / Module Under Test | | Flight Display | | |
| Test Requirement | | They can specify the airport they depart from and the airport they arrive at | User Case | SearchFlight |
| Goals | | Verify the software will handle situation when there are no outbound flights(round trip) for the specified day | | |
| Setup for Test | |  | | |
| Pre-conditions | | There is no outbound flights between departure and arrival airports on specified date | | |
| Step | Action | Expected Results | Pass / Fail | Actual Results |
| 1 | Specify JFK as departure airport | System accepts JFK as departure and displays “JFK(John F. Kennedy International)” in the “Going From” text window | Pass |  |
| 2 | Specify PHL as Arrival airport | System accepts PHL as arrival and displays “PHL(Philadelphia International)” in the “Going To” text window | Pass |  |
| 3 | Specify 2015\_5\_9 as departure day | System accepts specified date and “2015\_5\_9 “displayed in text of “Departing on” window | Pass |  |
| 4 | Specify 2015\_5\_13 as return day | System accepts specified date and “2015\_5\_13 “displayed in text of “Returning on” window | pass |  |
| 5 | Specify “First-class” seat | System accepts specified seat class and displays “First-class” in text of “Departing on” window | Pass |  |
| 6 | Click “Search” button | No flights are displayed and system displays dialog ‘’Oops, no flight found! Try another day!’’ | pass | See Fig 4 |

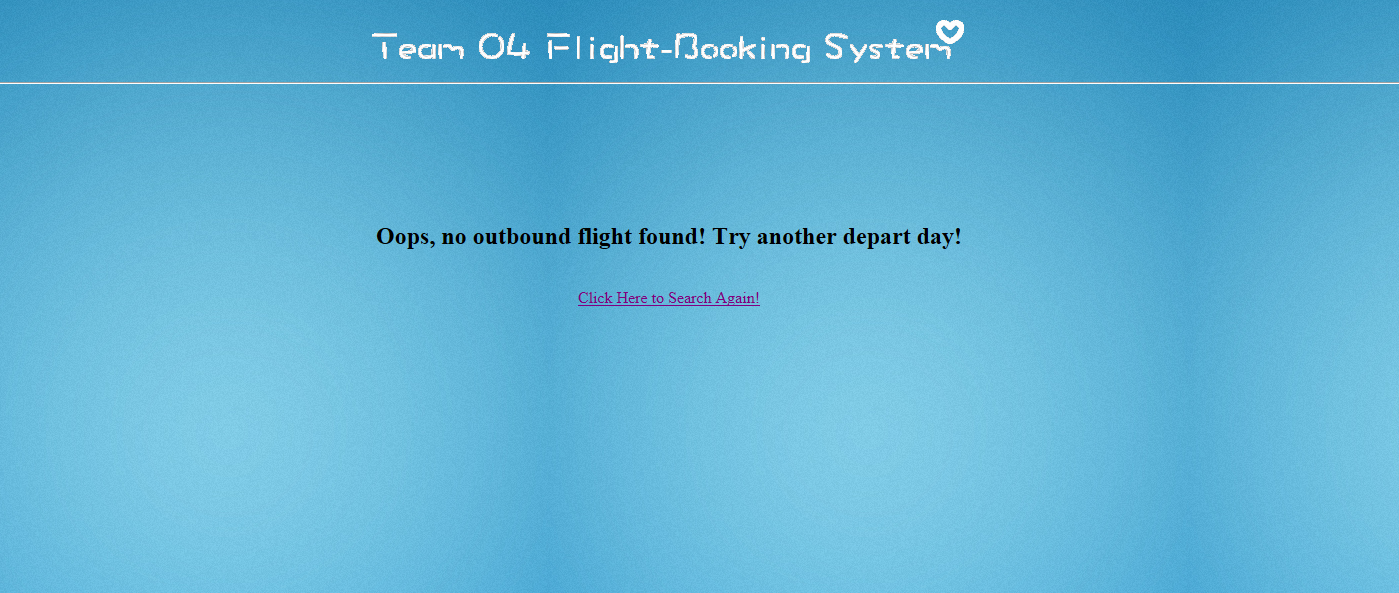


Fig 4---Round trip: fail to find an outbound flight

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Name/Number | | RoundTrip\_error case2 | | | |
| Test Case Description | | Displays outbound flights. After customer choose an outbound flight, if no inbound flights can be found, informs customer cannot find inbound flights. | | | |
| Function / Module Under Test | | Flight Display | | | |
| Test Requirement | | They can specify the airport they depart from and the airport they arrive at. | | User Case | SearchFlight |
| Goals | | Verify the software will handle situation when there are no inbound flights(round trip) for the specified day | | | |
| Setup for Test | |  | | | |
| Pre-conditions | | There is no inbound flights between departure and arrival airports on specified date | | | |
| Step | Action | | Expected Results | Pass / Fail | Actual Results |
| 1 | Specify JFK as departure airport | | System accepts JFK as departure and displays “JFK(John F. Kennedy International)” in the “Going From” text window | Pass |  |
| 2 | Specify PHL as Arrival airport | | System accepts PHL as arrival and displays “PHL(Philadelphia International)” in the “Going To” text window | Pass |  |
| 3 | Specify 2015\_5\_13 as departure day | | System accepts specified date and “2015\_5\_13 “displayed in text of “Departing on” window | Pass |  |
| 4 | Specify 2015\_5\_15 as return day | | System accepts specified date and “2015\_5\_15 “displayed in text of “Returning on” window | pass |  |
| 5 | Specify “First-class” seat | | System accepts specified seat class and displays “First-class” in text of “Departing on” window | Pass |  |
| 6 | Click “Search” button | | System transitions to flight display window and shows all outbound flights from JFK to PHL on 13 May 2015. | pass | See Fig 5.1 |
| 7 | After selecting outbound flight, click “select” to choose a return flight | | No flights are displayed. | pass | See Fig 5.2 |

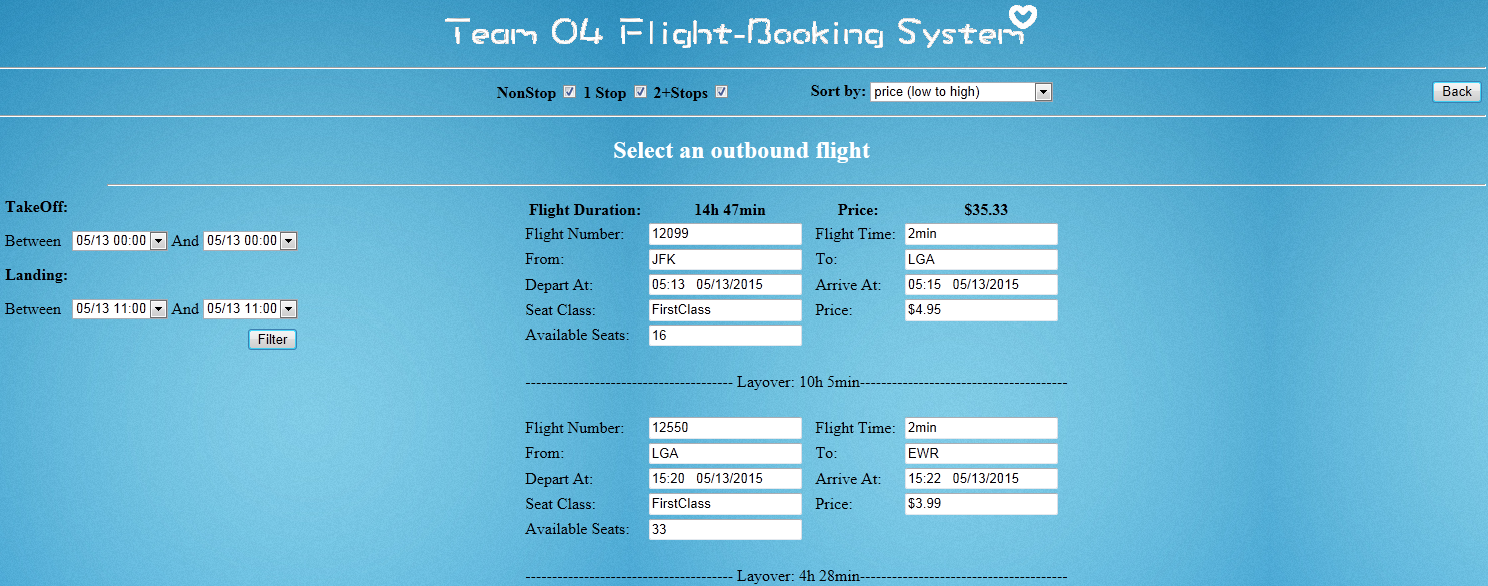




Fig 5.1---Round trip: success at finding an outbound flight

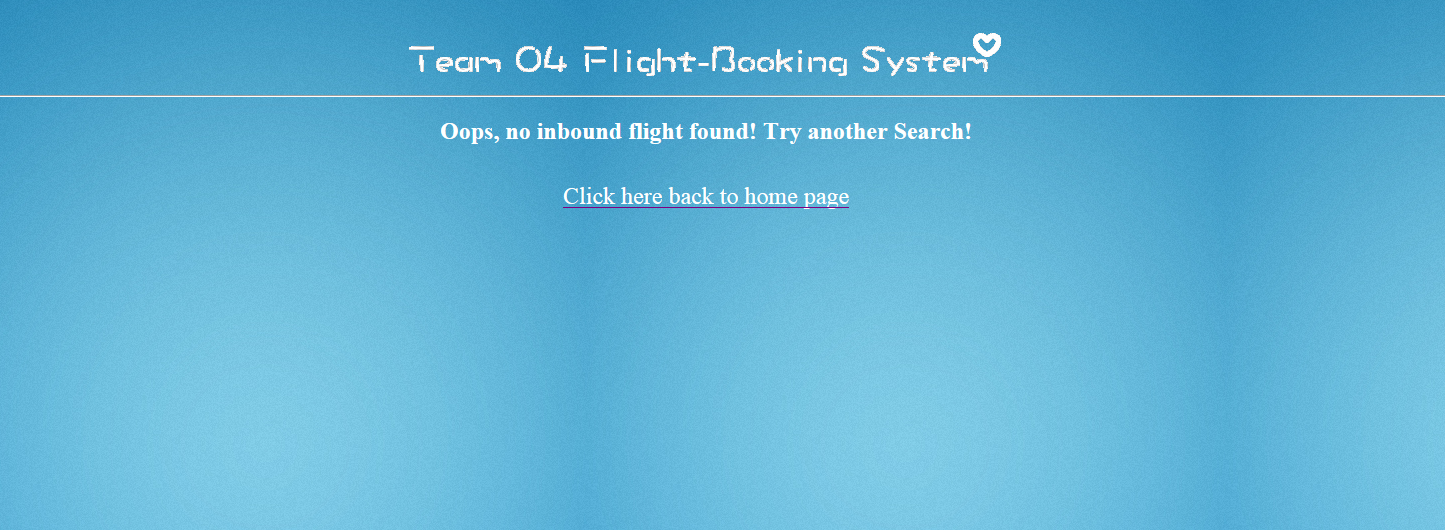


Fig 5.2---Round trip: fail to find an inbound flight