SSR WPI Reservation System Test Plan

Prepared by Team SSR

Gunnar Horve,
Yujun Tian,
Linan Xu,
Xiaoting Cui

26 March 2017
Version 1.0

Introduction

Objectives

This document is to define the testing methods to be used throughout the WPI/SSR Airplane Reservation System design. It is intended to be referenced periodically, as each test defined here will be required to be passed prior to system release.

Scope

This document is intended to limit the scope of testing to the previously defined and agreed upon requirements in the Requirements Analysis Document (RAD). For convenience, the requirements earlier specified are re-listed below

Functional Requirements

1. Airport Selection: Customers shall be able to specify the departure airport they wish to travel from and the airport they wish to arrive at.
2. Stopover Selection: Customers shall be able to specify the maximum number of stopovers when traveling from departure to destination airports.
3. Stopover Limit: The system shall allow a maximum of 2 stopovers when traveling from departure to destination airports. (The system will support 0, 1 or 2 stopovers when traveling from departure to destination airport.)
4. Class Selection: Customers shall be able to select first class or economy reservation for each flight between departure and destination airport.
5. Layover Range: The system will only display flights whose layover time is between 30 min and 4 hours.
6. Time Display: The system will show the departure time and arrival time of each leg which will be displayed airport local time.
7. Roundtrip Selection: Customers will have the ability to reserve flights to travel either one-way or reserve a round-trip flight.
8. Date Selection: Customers will be able to search for flights using departure date.
9. Roundtrip UI: Customers will be able to perform a round trip flight reservation as a sequence of two single-direction flight reservations
10. Pricing Display: The system will display the pricing of flights from departure to destination.
11. Flight Sorting: The system will allow customer to display flights sorted by price or travel time.
12. Confirmation Screen: The system will allow the customer to select flights and confirm the selection prior to the reservation being made.

13. Confirmation UI: The system will allow customer to select and reserve flights from a searching list.
14. No Deletions: The system will not allow a reservation to be deleted once it is made.
15. Request Method: The system should be able to request data using HTTP Get queries.
16. Server Reading: The system should be able to read data from XML returned from database.

Non-Functional Requirements

1. Reliability

1. Tests Exist: All classes and use cases will be unit tested through the use of a test plan document
2. No Space UI: The system will be able to return the information of 'there is not seat' when requested seat is not available for all legs of the flight.
3. Servery Busy UI: The system will be able to return the information of 'server is busy' to customer when the database is locked by others.
4. No race cond: The system should be able to lock the database to prevent race conditions

2. Performance

1. Fast UI: Response time for any requested actions will reasonable. Operations in excess of 3 seconds will provide indication to the customer the system is operating.
2. Search Speed: The system should provide results to the customer's searching action within 60 seconds

3. Supportability

1. Code is JAVA: The application will use the JAVA programming language for platform independence.

Assumptions/Risks

Assumptions

This testing document rests on several key assumptions. Namely:

1. WPI-provided database returns data consistent with its specifications
2. System will be used with access to a stable internet connection
3. System will be used on a reasonable modern (post 2009) laptop

4. System will be used by moderate (<10,000) users at once

Risks

1. Data loss: WPI database is rendered inoperable, halting development
2. Search complexity: depth 3 exhaustive search may be too demanding for real-time use
3. Locking issues: simplicity of WPI database locking likely won't scale to large user bases
4. Maintainability: system UI is currently implemented in a slapdash manner--potentially not maintainable by future devs

Test Approach

Req. #	Title	Verification Method				Verification Test Number
		Test	Analys.	Demo	Insp.	
F.1	Airport Selection	●				1
F.2	Stopover Selection			●		2
F.3	Stopover Limit			●		3
F.4	Class Selection	●				4
F.5	Layover Range			●		5
F.6	Time Display				●	6
F.7	Roundtrip Selection	●				7
F.8	Date Selection	●				8
F.9	Roundtrip UI				●	9
F.10	Pricing Display				●	10
F.11	Flight Sorting	●				11
F.12	Confirmation Screen				●	12
F.13	Confirmation UI				●	13
F.14	No Deletions	●				14

F.15	Request Method	●				15
F.16	Server Reading			●		16
NF.1.1	Tests Exist	●				17
NF.1.2	No Space UI				●	18
NF.1.3	Server Busy UI				●	19
NF.1.4	No Race Cond.	●				20
NF.2.1	Fast UI				●	21
NF.2.2	Search Speed		●			22
NF.3.1	Code is JAVA				●	23

Test Specifications

Test Case:F.1 Airport Selection			Requirement of origin: Customers shall be able to specify the departure airport they wish to travel from and the airport they wish to arrive at.		
Preconditions: There is a list of departure and arrival airports on UI					
Dependencies: the lists of all airports which can be selected on UI					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Select one of the depature airports	Choose BOS	BOS airport is selected		
2	Select one of the arrival airports	Choose SFO	SFO is selected		
Postconditions: two separate airports serving as departure airport and arrival airport are selected					

Test Case: F.2 Stopover Selection			Requirement of origin: Customers shall be able to specify the maximum number of stopovers when traveling from departure to destination airports.		
Preconditions: a list of stopover(0,1,2) on UI					
Dependencies: the number of stopover can be selected on UI					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Selected a number of stopover	Selected 2	2 is selected		
Postconditions: a number of stopover is selected					

Test Case: F.3 Stopover Limit			Requirement of origin: The system shall allow a maximum of 2 stopovers when traveling from departure to destination airports. (The system will support 0, 1 or 2 stopovers when traveling from departure to destination airport.)		
Preconditions: software is collected to database					
Dependencies: able to input data, able to search through database, able to display flights					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Verify all flights with less than 3 stopovers from departure to arrival airport on a specific day are retrieved from database	Input BOS as depart, SFO as destination, 5/10/2016, one way	All displayed flights have less than 3 stopover		
2					
Postconditions: all valid flights are displayed					

Test Case: F.4 Class Selection	Requirement of origin: Customers shall be able to

	select first class or economy reservation for each flight between departure and destination airport.

Preconditions: all valid flights are displayed between depart and destination (all valid flights between BOS and SFO)

Dependencies: a list of first class and coach can be selected located after each flight, a list of flights is displayed

Step	Actions	Data	Expected Results	Actual Results	Notes
1	select class for the first flight	Select coach	Coach is selected for 1st flight		
2	Select class for the second flight	Select first class	First class is selected for 2nd flight		

Postconditions: seat class is selected for each flight between depart and destination

Test Case: F.5 Layover Range	Requirement of origin: The system will only display flights whose layover time is between 30 min and 4 hours.

Preconditions: software is collected to database

Dependencies: able to input data, able to search through database, able to display flights

Step	Actions	Data	Expected Results	Actual Results	Notes
1	Verify all flights with reasonable layover time from departure to arrival airport on a specific day are retrieved from database	Input BOS as depart, SFO as destination, 5/10/2016, one way	All flights have layover time from 30min to 4h		

Postconditions: all valid flights with reasonable layover time are displayed

Test Case: F.6 Time Display			Requirement of origin: The system will show the departure time and arrival time of each leg which will be displayed airport local time.		
Preconditions: all valid flights are displayed between depart and destination (a list of all valid flights between BOS and SFO)					
Dependencies: a list of flights is displayed					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Verify all legs are displayed in local time	BOS as depart, SFO as destination, 5/10/2016, one way	all legs are displayed in local time		
2					
Postconditions: each leg is displayed in local time between depart and destination					

Test Case:F.7 Roundtrip Selection			Requirement of origin: Customers will have the ability to reserve flights to travel either one-way or reserve a round-trip flight.		
Preconditions: The UI is provided.					
Dependencies: 1) Ability to show the UI with “round-trip” button. 2)Ability to select flight with chosen departure, arrival airport and departure date.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Click on “Round-trp” button.	Dep Bos, Arrival SFO, Departure date 4/5/201	After ordering the Bos to SFO ticket, customers get back to the main UI.		

2	Select departure from SFO, arrival BOS, and choose return date.		Customers can order the ticket again.		
Postconditions:Two tickets can be ordered successfully.					

Test Case:F.8 Date Selection			Requirement of origin:Customers will be able to search for flights using departure date.		
Preconditions: The UI can provide departure date to choose from.					
Dependencies: 1) Ability to choose the departure date from UI. 2) Ability to search the flight according to the date customers has chosen.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Click on the “date of travel” column in User Interface.	The date which has been chosen.	A list of flight, which would departure on that date, will be displayed.		
2	Test the search function by passing exact “date” parameter and run the code.		A list of flights sharing the same date of what we have passed into,		
3	Verify same step of step 1.		Same flight combinations are sorted by departure time.		
4	Verify same step of step 2.		Same flight combinations are returned from the system.		
Postconditions:All valid flights, which would departure on that date, are displayed.					

Test Case:F.9 Roundtrip UI			Requirement of origin:Customers will be able to perform a round trip flight reservation as a sequence of two single-direction flight reservations.		
Preconditions: Costumers has ordered two tickets, one is from A to B, and one is from B to A.					
Dependencies: 1) Ability to show the UI .2) Ability to provide the “round-trip” selection.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	After making confirmation to the round way tickets	Dep BOS, arrival SFO, departure date 4/5/2017; Dep SFO, arrival Bos, departure date 6/72017	A confirmation screen shows that: “Dep BOS, arrival SFO, departure date 4/5/2017; Dep SFO, arrival Bos, departure date 6/72017”.		
Postconditions: The UI will show the information of round trip customers has ordered.					

Test Case:F.10 Pricing Display:			Requirement of origin:The system will display the pricing of flights from departure to destination.		
Preconditions:The price of the each flight is included in the original dataset.					
Dependencies: 1) Ability to get XML by HTTP query.2)Ability to parse data from XML files.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Choose departure and arrival airport and departure date	Flights, which satisfy the requirement has been	A list of flights, with the price, would		

	on the UI.	sorted.	displayed.		
2	Parse flights data from XML file.	Flight XML file	Get the String data of flights and have price in every record.		
Postconditions: All the flights shown from the UI have price displayed.					

Test Case:F.11 Flight Sorting			Requirement of origin:The system will allow customer to display flights sorted by price or travel time.		
Preconditions: At least 3 flights from departure to destination airport on specified day have been retrieved from database					
Dependencies: 1) Ability to command sorting by travel time. 2) Ability to display list of flights.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Verify flights are sorted by price or travel time.	Depart BOS, ARRIVAL SFO, one way, 4/5/2016	Over 3 flights are sorted from lowest price to highest price, or from shortest time to longest time.		
2	Type “sort_by_time” and press<return>.		System responds with “working...”		
3	Verify flights are sorted by flight time(shortest time first).		Shortest time first, longest time last.		

4	Type “sort_by_price” and press<return>		System responds with “working...”		
5	Verify flights are sorted by price(cheapest price first).		Cheapest flight first, most expensive last.		
Postconditions: Valid flights are displayed by order from cheapest to most expensive price, or by order from shortest time to longest time.					

Test Case:F.12 Confirmation Screen			Requirement of origin:The system will allow the customer to select flights and confirm the selection prior to the reservation being made.		
Preconditions:The desired flight has been clicked.					
Dependencies: Ability to select departure, arrival and departure date from UI.					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Click on the “confirmation” button.		The message box would show up: “Do you confirm this flight”.		
2	Click on the “confirm”.		The ticket would be reserved successfully and get back to the main UI.		
3	Click on the “no”.		The ticket is not ordered, and get back to the main UI.		

Postconditions: Go back to the main UI.

Test Case:F.13 Confirmation UI			Requirement of origin: The system will allow customer to select and reserve flights from a searching list		
Preconditions: The system has listed a search result					
Dependencies: 1)Ability to command reserving flights 2)Ability to display list of flights					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Select the flight the user prefer	Get all flights satisfied with all requirements	A list of flights		
2	Click “reserve the flight” button		A checkbox to confirm the selection		
3	Click “Yes” to reserve the flight		Return to the main user interface		
4	Click “No” to cancel the reservation		Return to the searching list		
Postconditions: Show a box to say that order successfully					

Test Case:F.14 No Deletions			Requirement of origin: The system will not allow a reservation to be deleted once it is made		
Preconditions: The system has reserved the order of a flight					
Dependencies: 1)Ability to command reserving flights 2)Ability to reject deletions of orders					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Display all the orders the user ordered	Get all the order of the user	A list of flights		

2	Orders cannot be deleted by user		No deletion function provided		
Postconditions: All the information of the order but cannot be deleted					

Test Case: F.15 Request Method			Requirement of origin: The system should be able to request data using HTTP Get queries		
Preconditions: The system wants to get the data					
Dependencies: 1)Ability to get data by using HTTP queries					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Verify airports data get from HTTP Get query	HTTP server	Get the airports data stored by XML file		
2	Verify flights data get from HTTP Get query	HTTP server	Get flights data stored by XML file		
3	Verify airplanes data get from HTTP Get query	HTTP server	Get airplanes data stored by XML file		
Postconditions: Get the XML file stored with airports, flights and airplanes data					

Test Case: F.16 Server Reading			Requirement of origin: 1. The system should be able to read data from XML returned from database		
Preconditions: The system has got the XML file by HTTP Get query					
Dependencies: 1) Ability to get XML by HTTP query 2)Ability to parse data from XML files					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Parse airports data from XML file	Airport XML file	Get the String data of airports		
2	Parse flights data from XML file	Flight XML file	Get the String data of flights		
3	Parse airplanes data from XML file	Airplane XML file	Get the String data		

			of airplanes		
Postconditions: Get the String data of airports, flights and airplanes					

Test Case: NF.1.1 Tests Exist			Requirement of origin: All classes and use cases will be unit tested through the use of a test plan document		
Preconditions: Classes and use cases are existed					
Dependencies: 1) Ability to test all the classes 2) Ability to test all the use cases					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Select one of the classes to test	All the classes	Result of the unit test		
2	Select one of the use cases	All the use cases	Result of the unit test		
Postconditions: Get the result of all unit test					

Test Case: NF.1.2 No Space UI			Requirement of origin: The system will be able to return the information of 'there is not seat' when requested seat is not available for all legs of the flight		
Preconditions: The system has listed a search result					
Dependencies: 1) Ability to show the seat type 2) Ability to show the seat number					
Step	Actions	Data	Expected Results	Actual Results	Notes
1	Select one of the seat type	Seat number by seat type of this flight	Show if there are available seats		
2	Choose the available seat type		Decrease the number of the seat		
Postconditions: Return "there is no seat" if requested seat is not available for all legs of the flight					

Test Environment

For tests in which it is relevant to do so, J-Unit tests will be conducted in an Eclipse project which is a directory sibling of the main project.

Milestones/Deliverables

Test Schedule

As of this moment, there are four weeks until project delivery. In response, the following schedule is proposed:

- Week 1: Search testing. All tests pertinent to legitimizing search done here.
- Week 2: UI testing. All tests pertinent to legitimizing the SRC UI are done here.
- Week 3: Performance Testing. All remaining tests are intended to be done here.
- Week 4: Slack fill. All tests which inevitably failed the first time are intended to be done here.

Deliverables

Pending the submission of this document, there are no intended deliverables from SSR to WPI prior to project completion. That said, it is understood that all tests will have been passed by that time.