**Air Route Planner**

Test Document

*Version 2.1*

**That “One” Team**

SE300, Section 1

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **A/M/D** | **Author** | **Description** |
| 1.0 | 04/15 | A | Yutong Zhu | Skeleton layout |
| 2.0 | 04/19 | A | B.Powell  B. Rompa  C. Wilkerson  Y.Zhu | Re-work document to meet specificiations |
| 2.1 | 04/24 | AMD | B.Powell  B. Rompa  C. Wilkerson  Y.Zhu  M.Khan | Testing of cases. Added more cases |

` \*A - Added, M - Modified, D - Deleted

**Table of Contents**

[**Purpose** 4](#_Toc354523600)

[**Team Project Information** 4](#_Toc354523601)

[**Test Cases** 5](#_Toc354523602)

# **Purpose**

The purpose of this document is to detail and lay out what cases were tested and in which fashion. This document is designed so that any test conducted may be replicated and verified for accuracy. This document also serves to demonstrate what functionality the program achieves, as well as where it lacks.

# **Team Project Information**

Course: Spring 2013 SE300 Section 1

Team: That “One” Team

Members/Roles:

* **Team Leader:** Brian Powell
* **Development Manager:** Brittany Rompa
* **Planning Manager:** Craig Wilkerson
* **Quality Manager:** Yutong Zhu
* **Req/Support Manager:** Muraad Khan

# **Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Product | Action | Expected | Actual | Pass/Fail | Comments |
| 1 | File Input | 1. Source code launched without file in local directory 2. File chooser window opens 3. User picks a valid input file | Program launches normally | Program launches normally | Pass |  |
|  |  |  |  |  |  |  |
| 2 | File Input | 1. Source code launched without file in local director 2. File chooser window opens 3. User picks an invalid input file 4. Program prompts user to pick an input file again | Re-prompt user for valid input file | Program launches bad input file | Fail | Took a bad input file  ~60% successful  // logic needs changing (ITL) |
|  |  |  |  |  |  |  |
| 3 | GUI | 1. Follow steps in test case 1 2. User selects “ATL” as origin 3. User selects “ATL” as destination 4. Click “Find” button | Returns Invalid airport entry in history | Returns Invalid airport entry in history | Pass |  |
|  |  |  |  |  |  |  |
| 4 | GUI | 1. Follow steps 3.a and 3.b 2. User selects “DCA” as destination 3. Click Find | Nothing happens | Nothing happens | Pass | // waiting for category selection |
|  |  |  |  |  |  |  |
| 5 | GUI | 1. Repeat steps 4.a 4.b 2. User selects cost category 3. User selects lowest sub category 4. Click “Find” button | Returns cheapest route cost | Returns cheapest route cost | Pass |  |
|  |  |  |  |  |  |  |
| 6 | GUI | 1. Repeat steps 4.a 4.b 2. User selects highest sub category 3. Click “find” button | Returns highest route cost | Returns highest route cost | Pass |  |
|  |  |  |  |  |  |  |
| 7 | GUI | 1. Repeat step 4.a 4.b 2. User selects airline filter 3. User selects “Delta” 4. User clicks find | Returns all routes with Delta Airlines | Returns all routes with Delta Airlines | Pass | // Set resizable GUI (ITL) – airline selection pushes scroll bar off frame |
|  |  |  |  |  |  |  |
| 8 | GUI | 1. Repeat steps 4.a 4.b 2. User selects “Air Tran” 3. User clicks find | Returns invalid for path selected in history | Returns invalid for path selected in history | Pass |  |
|  |  |  |  |  |  |  |
| 9 | GUI | 1. Repeat steps 4.a 4.b 2. User selects time category 3. User selects shortest sub category 4. User clicks find | Returns shortest route |  | Pass/Fail | output is incorrect  // ITL – need to tweak the time output |
|  |  |  |  |  |  |  |
| 10 | GUI | 1. Repeat steps 4.a 4.b 2. User selects time category 3. User selects longest sub category 4. User clicks find | Returns longest route |  | Pass/Fail | output is incorrect  // ITL – need to tweak the time output |
|  |  |  |  |  |  |  |
| 11 | GUI | 1. Follow steps in test case 1 2. User selects “ORL” as origin 3. User clicks “More Info” | Returns ORL information in info box | Returns ORL information in info box | Pass |  |
|  |  |  |  |  |  |  |
| 12 | GUI | 1. Follow steps in test case 1 2. User selects “JRK” as destination 3. User clicks “More Info” | Returns JFK information in info box | Returns JFK information in info box | Pass |  |
|  |  |  |  |  |  |  |
| 13 | Add Routes | 1. Follow steps in test case 1 2. User clicks route tab 3. User clicks create route 4. User clicks accept |  |  | Pass |  |
|  |  |  |  |  |  |  |
| 14 | Add Routes | 1. Follow test case 13 a-c 2. User changes destination to “ORL” 3. User enters depart time of 0830 4. User enters an arrival time of 0730 5. User enters airline of “UNITED” 6. User enters cost of 130.00 | Returns Invalid time input | Returns Invalid time input | Pass |  |
|  |  |  |  |  |  |  |
| 15 | Add Routes | 1. Follow test case 13 a-c 2. User changes destination to “DCA” 3. User enters depart time of “089-“ 4. User enters arrival time of 0930 5. User enters airline of “United” 6. User enters cost of 130.0 | Returns Invalid for depart and arrival times | Returns Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 16 | Add Routes | 1. Follow test case 13a-c 2. User changes destination to “JFK” 3. User enters depart time of 0830 4. User enters arrival time of 0830 5. User enters airline of UNITED 6. User enters cost of 130.0 | Invalid for depart and arrival times | Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 17 | Add Routes | 1. Follow test case 13a-c 2. User changes destination to “JFK” 3. User enters depart time of 0830 4. User enters arrival time of 0820 5. User enters airline of UNITED 6. User enters cost of 130.0 | Invalid for depart and arrival times | Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 18 | Add Routes | 1. Follow test case 13 a-c 2. User changes destination to “JFK” 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of UNITED 6. User enters cost of 130.0 | Route adds to system | Route is added to the system | Pass |  |
|  |  |  |  |  |  |  |
| 19 | Add Routes | 1. Follow test case 13a-c 2. User changes destination to “JFK” 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of 898air 6. User enters cost of 130.0 | Returns Invalid for airline | Returns Invalid for airline | Pass |  |
|  |  |  |  |  |  |  |
| 20 | Add Routes | 1. Follow test case 13a-c 2. User selects “JFK” as origin 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of “united” 6. User enters cost of 130.00 | Returns Invalid for airline | Returns Invalid for airline | Pass |  |
|  |  |  |  |  |  |  |
| 21 | Add Routes | 1. Repeat test case 13a-c 2. User selects “JFK” as origin 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of “UNITED” 6. User enters cost of 140 | Returns invalid for cost | Returns invalid for cost | Pass |  |
|  |  |  |  |  |  |  |
| 22 | Add Routes | 1. Repeat test case 13a-c 2. User selects “JFK” as origin 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of “UNITED” 6. User enters cost of a32 | Returns invalid for cost | Returns invalid for cost | Pass |  |
|  |  |  |  |  |  |  |
| 23 | Add Routes | 1. Repeat test case 13a-c 2. User selects “JFK” as origin 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of “UNITED” 6. User enters cost -340 | Returns invalid for cost | Returns invalid for cost | Pass |  |
|  |  |  |  |  |  |  |
| 24 | Add Routes | 1. Repeat test case 13a-c 2. User selects “JFK” as origin 3. User enters depart time of 0830 4. User enters arrival time of 0900 5. User enters airline of “UNITED” 6. User enters cost 130.0 | Route adds to the system | Route is added to the system | Pass |  |
|  |  |  |  |  |  |  |
| 25 | Delete Routes | 1. Follow test case 1 2. Select Route menu 3. Select delete route 4. User selects route 1 5. User clicks delete | Route is deleted from system | Route gets deleted | Pass |  |
|  |  |  |  |  |  |  |
| 26 | Add Airport | 1. Follow test case 1 2. User clicks airports 3. User clicks create new airport 4. User enters aur | Returns invalid for format of airport | Returns invalid for format of airport | Pass |  |
|  |  |  |  |  |  |  |
| 27 | Add Airport | 1. Follow test case 26 a-c 2. User enters DCA | Returns invalid for already existent airport | Returns invalid for already existent airport | Pass |  |
|  |  |  |  |  |  |  |
| 28 | Add Airport | 1. Follow test case 26 a-c 2. User enters AAA | Airport is added | Airport gets added | Pass |  |
|  |  |  |  |  |  |  |
| 29 | Add Airport | 1. Follow test case 26a-c 2. User enters 89AA | Returns invalid for incorrect format | Returns invalid for incorrect format | Pass |  |
|  |  |  |  |  |  |  |
| 30 | Add Airport | 1. Follow test case 26a-c 2. User enters AAAA | Returns invalid for incorrect format | Returns invalid for incorrect format | Pass |  |
|  |  |  |  |  |  |  |
| 31 | Add Airport | 1. Follow test case 26a-c 2. User enters AA | Returns invalid for incorrect format | Returns invalid for incorrect format | Pass |  |
|  |  |  |  |  |  |  |
| 32 | Delete Airport | 1. Follow test case 1 2. User clicks Airports 3. User clicks delete airport 4. User selects “ATL” | Deletes airport from system | Airport is deleted from the system | Pass |  |
|  |  |  |  |  |  |  |
| 33 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 14b-f | Returns Invalid for depart and arrival times | Returns Invalid for depart and arrival times | Pass | // ITL – airline field is empty after invalid edit (resolved) |
|  |  |  |  |  |  |  |
| 34 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 15b-f | Returns Invalid for depart and arrival times | Returns Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 35 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 16b-f | Invalid for depart and arrival times | Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 36 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 17b-f | Invalid for depart and arrival times | Invalid for depart and arrival times | Pass |  |
|  |  |  |  |  |  |  |
| 37 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 18b-f | Route is edited in the system | Route gets edited in the system | Pass |  |
|  |  |  |  |  |  |  |
| 38 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 19b-f | Returns Invalid for airline | Returns Invalid for airline | Pass |  |
|  |  |  |  |  |  |  |
| 39 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 20b-f | Returns Invalid for airline | Returns Invalid for airline | Pass |  |
|  |  |  |  |  |  |  |
| 40 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 21b-f | Returns invalid for cost | Route cost is edited | Fail | // ITL – set cost equal to false (resolved) |
|  |  |  |  |  |  |  |
| 41 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 22b-f | Returns invalid for cost | Returns invalid for cost | Pass |  |
|  |  |  |  |  |  |  |
| 42 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 23b-f | Returns invalid for cost | Returns invalid for cost | Pass |  |
|  |  |  |  |  |  |  |
| 43 | Edit Route | 1. Follow test case 1 2. User selects routes 3. User selects edit route 4. Follow test case 24b-f | Route edited in the system | Route is edited in the system | Pass |  |
|  |  |  |  |  |  |  |
| 44 | Save | 1. Follow test case 1 2. User clicks file 3. User clicks save | Information saved, return information saved in history | Information saved, return information saved in history | Pass |  |
|  |  |  |  |  |  |  |
| 45 | Close Airport | 1. Follow test case 1 2. User selects Airports tab 3. User selects Open/Close Airport 4. User selects Close tab 5. User selects ATL 6. User selects OK | Returns invalid time for closure | Returns invalid time for closure | Pass |  |
|  |  |  |  |  |  |  |
| 46 | Close Airport | 1. Follow test case 45 a – d 2. User enters 0800 for begin 3. User enters 0730 for end | Returns invalid for end time | Returns invalid for end time | Pass |  |
|  |  |  |  |  |  |  |
| 47 | Close Airport | 1. Follow test case 45 a – e 2. User enters 0800 3. User enters 0800 4. User clicks OK | Returns Invalid time entry | Airport closes from 0800 - 0800 | Fail | // ITL – closes from 0800 – 0800 (resolved) |
|  |  |  |  |  |  |  |
| 48 | Close Airport | 1. Follow test case 45 a – e 2. User enters 0800 3. User enters 08a4 4. User clicks OK | Returns Invalid end time entry | Returns Invalid end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 49 | Close Airport | 1. Follow test case 45 a – e 2. User enters 0800 for begin 3. User enters 2500 for end 4. User clicks OK | Returns Invalid end time entry | Returns Invalid end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 50 | Close Airport | 1. Follow test case 45 a – e 2. User enters 2500 for begin 3. User enters 0830 for end 4. User clicks OK | Returns Invalid begin time entry | Returns Invalid begin time entry | Fail | // |
|  |  |  |  |  |  |  |
| 51 | Close Airport | 1. Follow test case 45 a – e 2. User enters 2500 for begin 3. User enters 600 for end 4. User clicks OK | Returns Invalid begin and end time entry | Returns Invalid begin and end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 52 | Close Airport | 1. Follow test 45 a – e 2. User enters –0830 for begin 3. User enters 1030 for end 4. User clicks OK | Returns Invalid begin time entry | Returns Invalid begin time entry | Pass |  |
|  |  |  |  |  |  |  |
| 53 | Close Airport | 1. Follow test case 45 a – e 2. User enters 0830 for begin 3. User enters 0930 for end 4. User clicks OK 5. User closes Close Airport box 6. User selects ATL as origin 7. User selects DCA as destination 8. User selects cost as category 9. User selects lowest as subcategory 10. User clicks Find | Airport successfully closes and search does not display routes falling under close time. | Airport successfully closes and search does not display routes falling under close time. | Pass |  |
|  |  |  |  |  |  |  |
| 54 | Open Airport | a. Follow test case 45 using Open tab | Returns invalid time for opening | Returns invalid time for opening | Pass |  |
|  |  |  |  |  |  |  |
| 55 | Open Airport | a. Follow test case 46 using Open tab | Returns invalid for end time | Returns invalid for end time | Pass |  |
|  |  |  |  |  |  |  |
| 56 | Open Airport | a. Follow test case 47 using Open tab | Returns Invalid time entry | Returns Invalid time entry | Pass |  |
|  |  |  |  |  |  |  |
| 57 | Open Airport | a. Follow test case 48 using Open tab | Returns Invalid end time entry | Returns Invalid end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 58 | Open Airport | a. Follow test case 49 using Open tab | Returns Invalid end time entry | Returns Invalid end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 59 | Open Airport | a. Follow test case 50 using Open tab | Returns Invalid begin time entry | Returns Invalid begin time entry | Pass |  |
|  |  |  |  |  |  |  |
| 60 | Open Airport | a. Follow test case 51 using Open tab | Returns Invalid begin and end time entry | Returns Invalid begin and end time entry | Pass |  |
|  |  |  |  |  |  |  |
| 61 | Open Airport | a. Follow test case 52 using Open tab | Returns Invalid begin time entry | Returns Invalid begin time entry | Pass |  |
|  |  |  |  |  |  |  |
| 62 | Open Airport | 1. Follow test case 53 a – e 2. User clicks Airports tab 3. User clicks Open/Close Airport 4. User selects Open tab 5. User selects ATL 6. User enters 0700 for begin 7. User enters 1000 for end 8. User clicks OK 9. User closes Open Airport box 10. User selects ATL as origin 11. User selects JFK as destination 12. User selects cost as category filter 13. User selects lowest as subcategory filter 14. User selects Find | Airport closes and reopens successfully, search will  display routes which fall under reopen time. | Airport does not reopen | Fail |  |