

Group Project Progress Report

Date: 8-Apr-2021

Course Code and Course Title (CRN Number): CEN4072 Software Testing (CRN 10421)

Software Application Title/Name: SWT - Airline Reservation System

Project Manager: Jana Grunewald

Group Members Name:

Jana Grunewald

Jeffry Munoz

Breanna Rhodes

Tamara Vergara

Project Phase (Milestone) Integration Testing

From Report Date: 25-Mar-2021

Report Date: 8-Apr-2021

Complete by (Milestone): 8-Apr-2021

Description of Group Project Milestone

Project Milestone Name: Integration Testing								
Requirement Description	Task ID	Task Description	Pass/Fail/Not Executed	Test Date	Responsible Primary Team Member	Responsible Secondary Team Member	Comment	Additional Comment
Research and install integration testing tools	01	Find suitable tools to facilitate the integration unit phase	Pass	1-Apr-2021	Jeffry Munoz	Breanna Rhodes	Tools such as Mockito were found and installed for later use	
Create presentation	02	Create the presentation for delivery during class	Pass	7-Apr-2021	Breanna Rhodes	Jeffry Munoz	We ensured the presenter could run all test cases need for the deliverable	
Create progress report documents	03	Create the progress	Pass	7-Apr-2021	Jeffry Munoz	Breanna Rhodes	Created the	

		report and deliverable documentation need to for the second deliverable					needed document ation and screenshots for the third deliverable	
Finish the Deliverable 3 document	04	Update and finish the deliverable documentation needed for class.	Pass	8-Apr-2021	Tamara Vergara	Jana Grunewald		
Create stub to test database connection for the searchCustomer class	05	Create a stub to verify that the search customer page of the GUI displays information correctly	Pass	7-Apr-2021	Jeffry Munoz	Jana Grunewald	The stub was created to return a specified test customer	
Create a mock for the cancel button for the userCreation class	06	Create a mock to verify the userCreation cancel button works properly	Pass	7-April-2021	Jana Grunewald	Tamara Vergara		
Create a mock for the cancel button for the addCustomer class	07	Create a mock to verify the addCustomer cancel button works properly	Pass	7-April-2021	Jana Grunewald	Tamara Vergara		
Create a stub for the login page, username and password fields	08	Create a stub to check the username	Pass	7-April-2021	Jana Grunewald	Tamara Vergara	The stub was created to return	

		and password fields work properly.					“user” and “pass”	
Create a mock for the login class, login button	09	Create a mock to make sure the login button works properly	Pass	7-April-2021	Jana Grunewald	Tamara Vergara		
Documentation completion	10	Added the final touches to the deliverable documentation	Pass	7-April-2021	Tamara Vergara	Jana Grunewald		
Create a mock for the ticket class.	11	Create a mock in the ticket class.	Pass	7-April-2021	Tamara Vergara	Jana Grunewald		
Create a mock for the ticketReport class.	12	Create a mock in the ticketReport class.	Pass	7-April-2021	Tamara Vergara	Jana Grunewald		
Create a mock in the searchCustomer class.	13	Create a mock in the searchCustomer class.	Pass	7-April-2021	Tamara Vergara	Jana Grunewald		

Group Project Milestone Screenshot

master

1 branch

0 tags

Go to file

Add file

Code

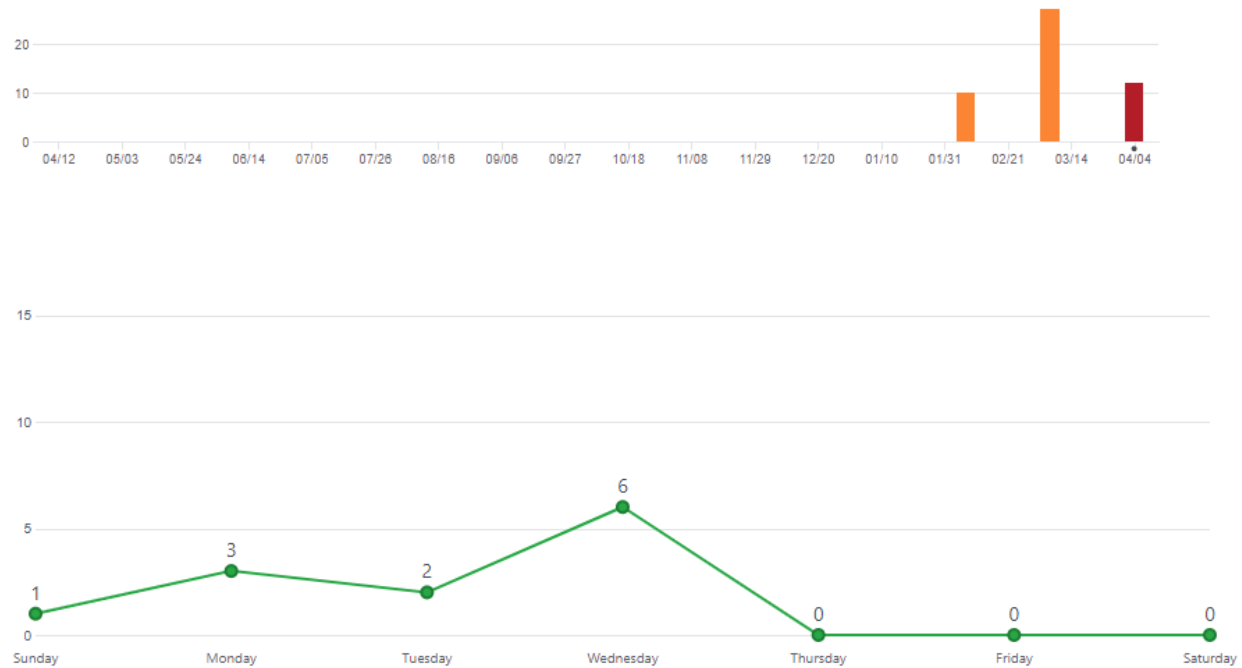
JeffMunoz

re-added Stub for DB connection in search customer

330a75a 1 hour ago

53 commits

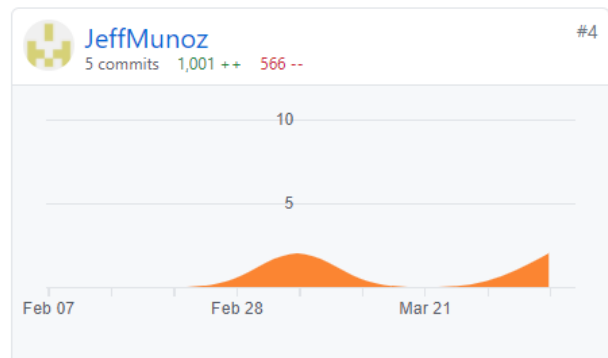
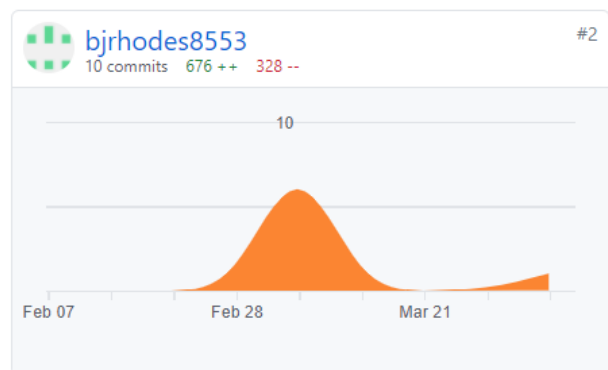
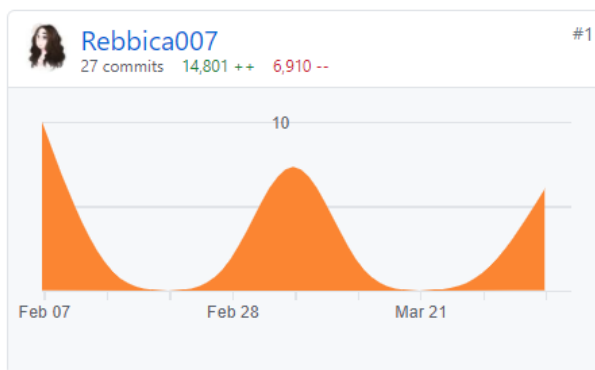
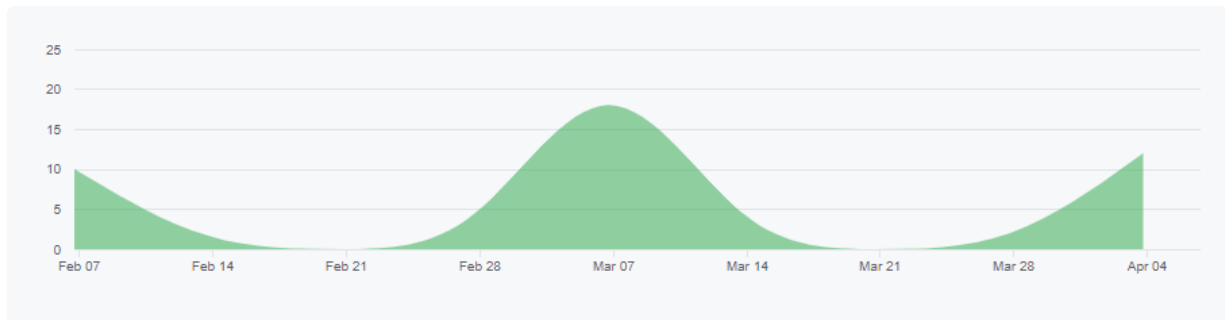
.idea	re-added Stub for DB connection in search customer	1 hour ago
Airline-SWT	re-added Stub for DB connection in search customer	1 hour ago
target	re-added Stub for DB connection in search customer	1 hour ago
.gitignore	Running code and database.	2 months ago
Airline-SWT.iml	re-added Stub for DB connection in search customer	1 hour ago
README.md	Update README.md	yesterday
jcalendar-1.4.jar	Added Stub for DB connection in search customer	4 hours ago
jgoodies-common-1.2.0.jar	Added Stub for DB connection in search customer	4 hours ago
jgoodies-looks-2.4.1.jar	Added Stub for DB connection in search customer	4 hours ago
junit-4.6.jar	Added Stub for DB connection in search customer	4 hours ago
mockito-core-3.8.0.jar	Added Stub for DB connection in search customer	4 hours ago
mysql-connector-java-8.0.23.jar	Added Stub for DB connection in search customer	4 hours ago
pom.xml	re-added Stub for DB connection in search customer	1 hour ago



Feb 7, 2021 – Apr 7, 2021

Contributions: Commits ▾

Contributions to master, excluding merge commits



Group Project Meeting Minutes

Meeting Date: 3/26/21

Meeting Length: 37 minutes

Team Members present:

Jana Grunewald

Jeffry Munoz

Breanna Rhodes

Tamara Vergara

Summary: During this meeting, tasks were assigned for the completion of the third deliverable. It was decided that Jeff would research different integration testing tools that could be used for this project.

Breanna and Tamara pointed out that only one mock and one stub were needed for this submission but

more would be needed for the final project. Jana covered the requirements for the next deliverable and the improvements needed based on the feedback from the previous submission.

Meeting Date: 4/02/21

Meeting Length: 43 Minutes

Team Members present:

Jana Grunewald

Jeffrey Munoz

Breanna Rhodes

Tamara Vergara

Summary: During this meeting, we discussed the project progress report. We discussed the feedback further to improve this submission. We also discussed trying to maintain only one test per case as we had included multiple tests in some of the test cases in the previous phase. We also discussed the selected tools for integrations testing. We decided that Jeff and Jana would create the stub needed for this deliverable while Breanna and Tamara created the mock. The deadline for the completion of this was also agreed on by the team to be Wednesday April 7th to allow time for corrections.

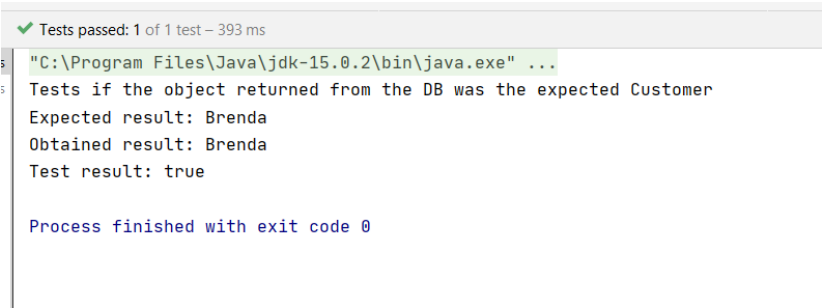
Additional Important Requirements:

1. Submit one (1) Microsoft Word Document or PDF on Canvas
 - a. **Do not submit a zipped file (-10 points)**
2. Your progress report must include all information specified and the subsections/subcategories included in this template
 - a. **Missing or incorrect information will result in a 10-point deduction from the overall project grade (-10 points)**
3. Progress Report File Naming Convention
 - a. **Course Code-CRN-ProgressReport-ProjectPhaseName-Group#**
 - i. e.g. CEN4072-10421-ProgressReport-UnitTestingPhase-Group7

Deliverable 3

Integration Testing Plan

Testing Category: Integration Testing (Functional)	
Test Criteria#1	
Requirement#1: FUN-3	The application shall allow users to search for a customer stored in the database using their customer ID
Input:	String representing the customer ID - "CS015"
Mock/Stub description:	To verify that the customer information obtained from the database when searching for a valid customer ID matches the expected customer.
Mock/Stub(Code)	<pre>@Test public void test() throws SQLException, ParseException { Customer testCustomer = new Customer("Brenda", "Rhodes", "CS015", "12346", "654789", "10501 FGCU BLVD South", "05/01/2021","female","1234567890"); DBConnection customerStub = mock(DBConnection.class); when(customerStub.findCust()).thenReturn(testCustomer); System.out.println("Tests if the object returned from the DB was the expected Customer"); System.out.print("Expected result: Brenda" + "\n" + "Obtained result: " + testCustomer.getFirstName() + "\n");</pre>

	<pre> System.out.println("Test result: " + testCustomer.getFirstName().equalsIgnoreCase("Brenda")); Assert.assertEquals("Brenda", testCustomer.getFirstName()); } </pre>
Expected Output	The first name, last name, nic number, passport ID, address, date of birth, gender, and photo of the customer displayed on the GUI
Test Result (Actual Output)	<p>The test correctly returned the first name of the customer.</p> 
Discrepancies	N/A
Dependencies	Java Swing Library, MySQL JDBC, and Mockito

Branch Coverage (%)

0 %

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

We had thought Method % was the branch % when in fact it is not. Branch coverage also only covers if/when/else statements from what we researched and we are unsure how to bring those cases up.

Statement Coverage (%)

84%

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

Testing Category: Integration Testing (Input Validation Testing)	
Test Criteria#2	
Requirement# FUN-5:	The user shall be able to add a unique flight.
Input:	<ul style="list-style-type: none">The user must enter a string for the following fields<ul style="list-style-type: none">Flight Name<ul style="list-style-type: none">Specifically, the user enters a string with alphabetic characters (a-z, A-Z), as long as it does not exceed 255 characters in length such as 'JetBlue'

	<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> ■ Chosen test input: “yellowSky” ○ Dep Time <ul style="list-style-type: none"> ■ Specifically, the user enters a string leading with integers followed by the period (.) character or the colon character (:) with the option of ‘AM’ or ‘PM’ on the end of the string as long as the string does not exceed 255 characters, in the format hour:minutes, such as ‘8.00AM’ or ‘11:00’. ■ Chosen test input: “11:00AM” ○ Arr Time <ul style="list-style-type: none"> ■ Specifically, the user enters a string leading with integers followed by the period (.) character or the colon character (:) with the option of ‘AM’ or ‘PM’ on the end of the string OR can be a single digit as long as the string does not exceed 255 characters, in the format hour:minutes, such as ‘8.00AM’ or ‘8’. ■ Chosen test input: “08:33PM” ○ Flight Charge <ul style="list-style-type: none"> ■ Specifically, the user enters a string of integers as long as the string does not exceed 255 characters. ■ Chosen test input: “14000” ● The user must select from a drop down menu for the following: <ul style="list-style-type: none"> ○ Source <ul style="list-style-type: none"> ■ Specifically, from one of the following choices: ‘India’, ‘Srilanka’, ‘UK’, ‘Usa’, ‘Canada’, ‘Chinna’. ■ Chose test input: “Chinna” ○ Departure <ul style="list-style-type: none"> ■ Specifically, from one of the following choices: ‘India’, ‘Srilanka’, ‘UK’, ‘Usa’, ‘Canada’, ‘Chinna’. ■ Chosen test input: “Usa” ● The user must choose from a date picker for the following: <ul style="list-style-type: none"> ○ Date <ul style="list-style-type: none"> ■ Specifically the year, month and date of flight. ■ Chosen test input: “2021-04-03”
Mock/Stub description:	<p>A mock was created to test the input of the user as they are entering information into the text fields, and selecting their choice from the choicebox. Because these components cannot be accessed unless in the controllers, the addFlightTest class simulates the data that can be entered into the components by the user and checks it against regex patterns and string arrays containing the items in the choicebox. The results of the verification of each component are stored in a boolean array, which is then checked against a boolean</p>

	<p>array containing all true objects using the mock object created using the Mockito classes. The mock object contains the results of verifying the input of each component, and then uses the Mockito method <code>verify().add()</code> to test the validity of the actual results. The first argument that goes into the <code>verify</code> parenthesis is the actual input stored in the mock object, and the argument that goes into the <code>add</code> parenthesis is the expected result. If the two match then every component was called correctly and the input was validated correctly. If the two do not match, then Mockito will display a message stating where exactly in the test case did the test fail. This is because the booleans stored in the array are created sequentially, so if the 3rd boolean is returned false by the mockito object, then the 3rd component that is sequentially listed is the component that is not currently working.</p>
Mock/Stub(Code)	<pre><i>//Creates the mock object which will store the test results.</i> List flightAddMock = mock(List.class); <i>//Add the test results to the mock object.</i> flightAddMock.add(testResults); <i>//Verifies if the actual output equates the expected output.</i> verify(flightAddMock).add(allTestPassed);</pre>
Expected Output	<p>The expected output should be True, this would mean the test passed. The items in the choice box are 6 strings. The mock object is expected to contain all 6 strings, which would fulfill the functionality of the method that returns the strings of the choice box.</p>
Test Result (Actual Output)	<p>The mocks containing the test inputs pass the regex pattern mocks for the text fields, and the choice box functionality mocks. This results in the test inputs being stored into the database each time the test is run and passed.</p>

	id	flightname	source	depart	date	deptime	arrtime	flightcharge
1	F0001	JetBlue	India	Uk	2019-06-14	8.00AM	10.00PM	50000
2	F0002	Delta	India	China	2019-06-15	8.00PM	2.00AM	15000
3	F0003	American Airlines	India	Srilanka	2019-06-15	9.00AM	10.00AM	9000
4	F0004	My Flight	Uk	India	2021-02-23	11:00	8	760
5	F0005	yellowSky	Chinna	Usa	2021-03-11	11:00AM	08:33PM	14000
6	F0006	yellowSky	Chinna	Usa	2021-03-11	11:00AM	08:33PM	14000
7	F0007	yellowSky	Chinna	Usa	2021-03-11	11:00AM	08:33PM	14000
8	F0008	yellowSky	Chinna	Usa	2021-03-11	11:00AM	08:33PM	14000
9	F0009	yellowSky	Chinna	Usa	2021-03-12	11:00AM	08:33PM	14000
10	F0010	yellowSky	Chinna	Usa	2021-04-03	11:00AM	08:33PM	14000
11	F0011	yellowSky	Chinna	Usa	2021-04-03	11:00AM	08:33PM	14000
12	F0012	yellowSky	Chinna	Usa	2021-04-03	11:00AM	08:33PM	14000
13	F0013	yellowSky	Chinna	Usa	2021-04-03	11:00AM	08:33PM	14000

Run: addflightTest

addflightTest7 s 651 ms

testUserInput7 s 651 ms

Tests passed: 1 of 1 test - 7 s 651 ms

exclude patterns:
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'.
true
Add flight has been hidden.
Class transformation time: 0.6147386s for 4422 classes or 1.3901822704658526E-4s per class
Process finished with exit code 0

Discrepancies

There were no discrepancies.

Dependencies

For this project we used JUnit, Mockito, and MySQL.

Branch Coverage (%)

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

Statement Coverage (%)

95%

Element	Class, %	Method, %	Line, %	Branch, %
 addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
 addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
 DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
 Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
 Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
 searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
 ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
 ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
 userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

Summary of Integration Testing Results

- Total number of requirements integration tests are associated with:
 - A total of 19 requirements were tested
- Total number of mocks and stubs written: 10 Mocks and 3 stubs
- Total number of passing mocks and stubs: 10 Mocks and 3 stubs pass
- Total number of failing mocks and stubs: 0
- Screenshot of overall branch coverage (Branch %): We had thought Method % was the branch % when in fact it is not. Branch coverage also only covers if/when/else statements from what we researched and we are unsure how to bring those cases up.

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

- Screenshot of overall statement coverage (Line %):

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)

- Screenshot of overall statement (line %) and branch (branch %) coverage percentage from IDE:

Element	Class, %	Method, %	Line, %	Branch, %
addCustomer	100% (6/6)	100% (23/23)	96% (295/3...	33% (1/3)
addflight	100% (3/3)	100% (17/17)	95% (211/2...	100% (0/0)
DBConnection	100% (1/1)	33% (1/3)	11% (2/18)	100% (0/0)
Login	100% (3/3)	100% (10/10)	85% (102/1...	100% (0/0)
Main	100% (8/8)	100% (23/23)	92% (98/106)	50% (1/2)
searchCustomer	100% (7/7)	86% (20/23)	84% (292/3...	0% (0/1)
ticket	100% (7/7)	66% (16/24)	82% (348/4...	0% (0/1)
ticketreport	100% (2/2)	100% (7/7)	93% (59/63)	50% (1/2)
userCreation	100% (3/3)	100% (11/11)	94% (145/1...	0% (0/1)