Airline Reservation System Test Report

Han Xiao - hxia0019@student.monash.edu
Yuhang Huang - yhua0226@student.monash.edu
Jinhui Yuan - jyua0030@student.monash.edu

Testing Objectives

The system integration test of the Airline Reservation System should validate from the requirements that:

1. Getting the flight information function works correctly.
2. Display the correct value to the passengers when they choose flight information.
3. Buying direct flight tickets function works correctly.
4. Buying tickets with the transfer flights function works correctly.
5. Display the correct value to the passengers when they buy a ticket.
6. All calculations are correct.
7. Display prompts when the user enters the error message。
8. The system is easy to use by the end-users.

Scope of Testing

The system integration testing for the Airline Reservation System will cover the aeroplane, flight, flight collection, passenger, ticket, ticket collection, and ticket system components.

How unit tests were identified

For testing the Flight class, unit tests can be written to check that all fields are required, date is in DD/MM/YY format, time is in HH:MM:SS format. For example, the EVT method can be used to split the input time format into a date and a time, and test for conformance to the format separately.

When testing the Passenger class, unit tests can be written to check that all fields of a passenger are required, phone numbers follow a valid pattern, email follows a valid pattern, passport number is not more than 9 characters long, and the passenger's first name, last name, age, and gender are included

when a passenger is being added. For example, for the format requirement of passport numbers, the BVT method can be used to test boundary values by inputting 8-digit, 9-digit, and 10-digit numbers to test whether the passport number meets the requirement. BVT testing can ensure that the system works properly when handling boundary values and can handle errors correctly when invalid data is entered. In this way, the basic functionality of the system can be ensured, and it can work properly under different input conditions.

To test the FlightCollection class, unit tests can be written to check if adding a flight to the system meets the requirements of both the flight and FlightCollection, and if valid city names are used. Additionally, tests can be conducted to verify if the system returns the correct flight information when trying to retrieve flight information. If an invalid city name is entered, the test case can verify if the system correctly issues a warning, while if a city without flights is input, the test case can verify if the system prompts correctly. And if a user inputs a flight which has already existed in the flight collection, our application will also throw an error.

How integration tests were identified

In the provided code for the TicketSystem, integration tests could be identified by observing the interactions between different components and ensuring that they are working together correctly. Specifically, in this system, there are several components including Ticket, Passenger, Flight, TicketCollection, and FlightCollection. Here's how some of the identified integration tests are performed:

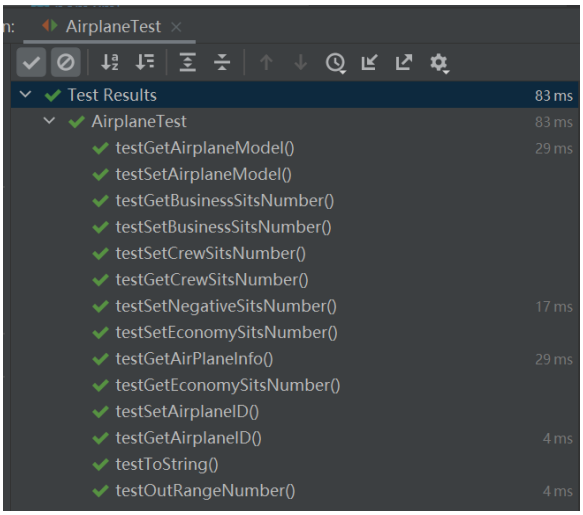
Ticket and Passenger: Tests are conducted to validate the passenger information when buying a ticket. For example, an exception is expected to be thrown if the passenger information is not set correctly. And a null value in passenger or flight parameters is also invalid, the checking method will return an exception message if it checks the null value. In addition, we also check whether the ticket price is valid when users input the price value, our system will throw an error if the price value is lower than zero.

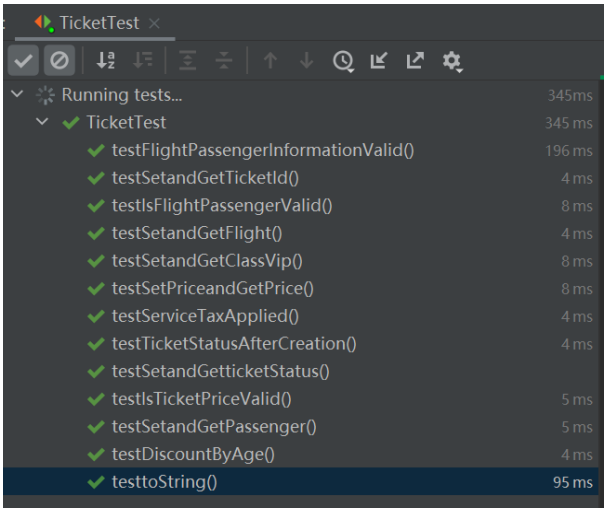
Ticket and Flight: Tests are conducted to validate the flight information when buying a ticket. For instance, an exception is expected to be thrown if a ticket for a non-existent flight is selected.

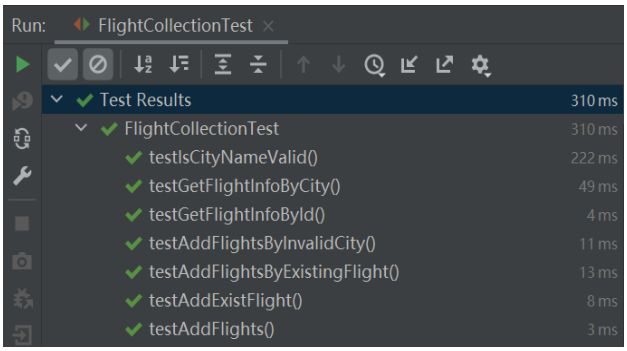
TicketSystem and TicketCollection: Tests are conducted to check if the ticket system can directly interact with the ticket collection, such as retrieving a ticket based on the ticket ID.

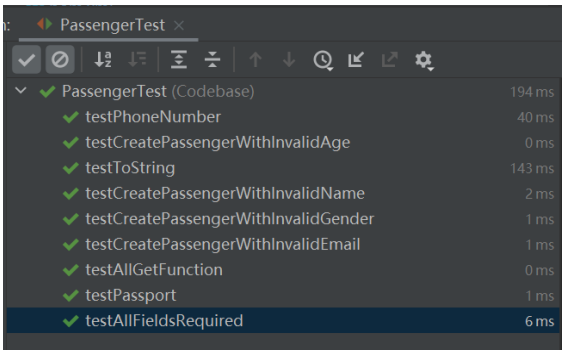
TicketSystem and FlightCollection: Tests are conducted to validate if the ticket system can directly interact with the flight collection, such as retrieving flight information based on the departure and destination cities. And whether the returned confirm information is correct to passengers when buying a ticket will also be checked.

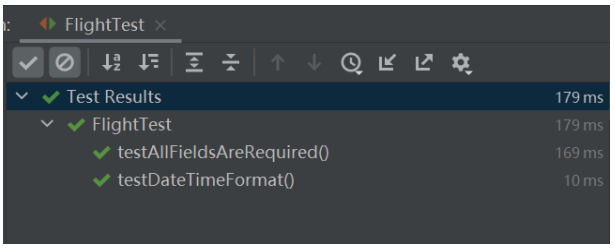
Unit test result

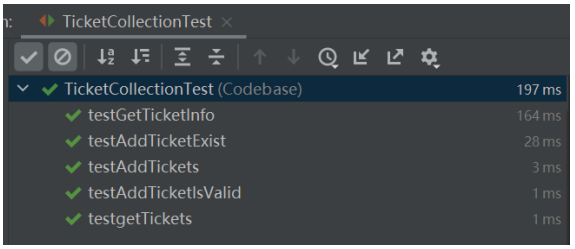




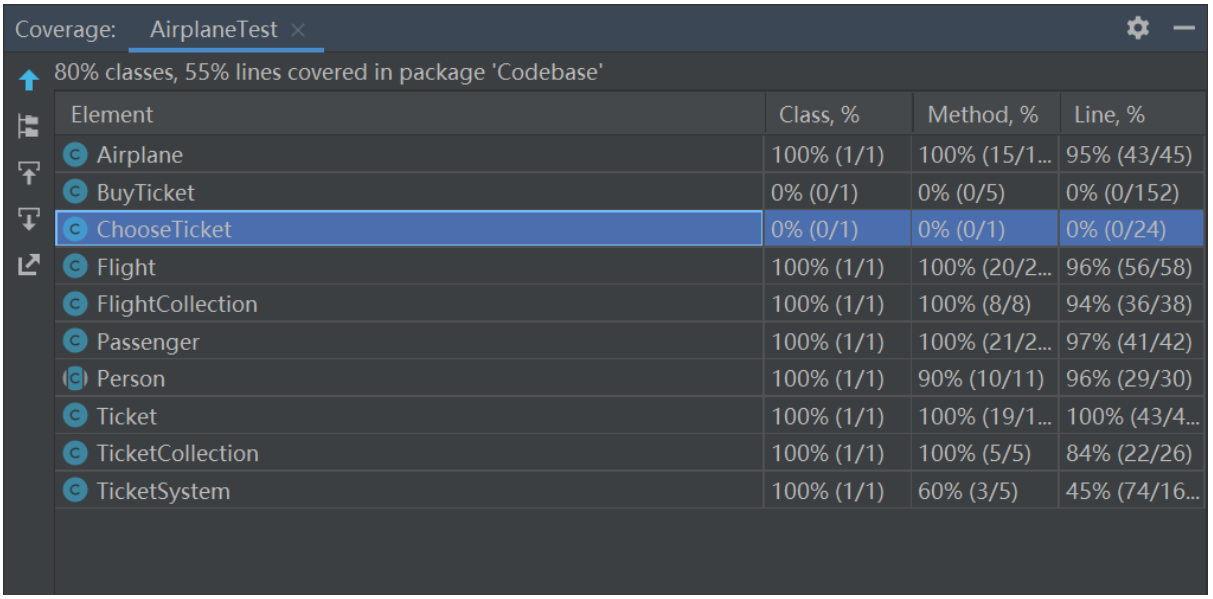






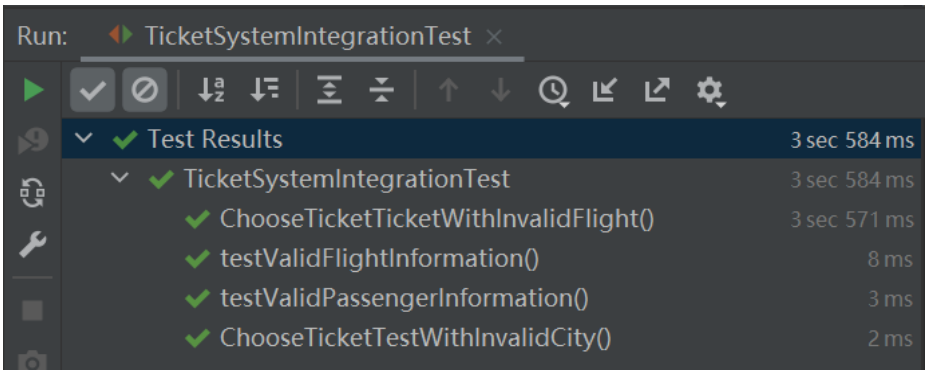


Unit test coverage

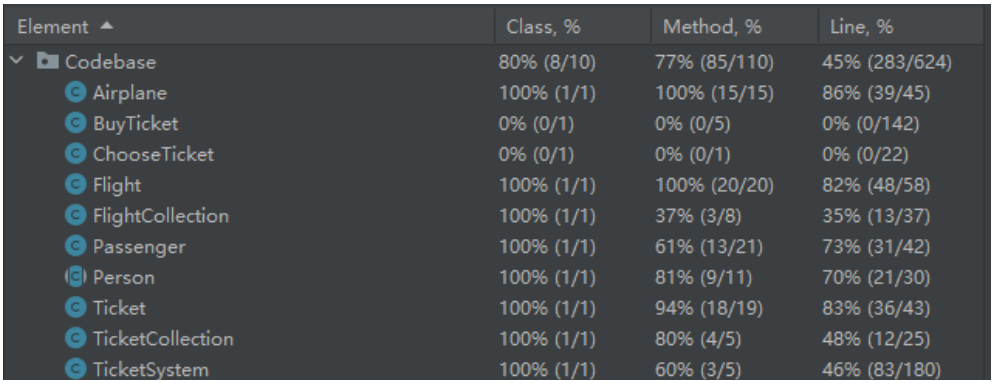


The unit test coverage of the system has reached 100% (excluding BuyTicket and ChooseTicket, because we have integrated the functionality of BuyTicket and ChooseTicket into the TicketSystem class).

Integration test result



Integration test coverage



The integration test coverage of the system has reached 100% (excluding BuyTicket and ChooseTicket, because we have integrated the functionality of BuyTicket and ChooseTicket into the TicketSystem class).

APPENDIX

Unit test cases

AirplaneTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test set negative seat number (BVT)	Input negative seat number : -1	Error message	Error message	P
002	Test out of range seat number (BVT)	Input seat number : 0	Error message	Error message	P
		Input seat number : 400	Error message	Error message	P
003	Test reasonable seat number	Input seat number : 300	No Error	No Error	P

FlightTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test wrong date time format	Input date and time : 2023-07-05 00:00:00	Error message	Error message	P
		05/07/2023 00-00-00	Error message	Error message	P
002	Test right date time format	Input date and time: 05/07/2023 13:55:00	No Error	No Error	P
003	Test all fields are required	Input new Flight lacking attributes	Error message	Error message	P

FlightCollectionTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test wrong city name	Input city:123	Error message	Error message	P
		Fuzhou//?	Error	Error	P

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
			message	message	
002	Test add Flights	Two Flight objects created and added to an ArrayList. Called addFlights method.	No Error	No Error	P
003	Test get Flight Info with valid parameter	Input city name and id of flight: ("Suzhou", "Beijing")	Return a valid flight	Return a valid flight	P
		3	Return a valid flight	Return a valid flight	P
004	Test add flight which is already in the system	Create a flight list containing two same flights, then call addFlights method.	Error message	Error message	P

PassengerTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test create passenger with invalid phone number	Input phone number:12345678	Error message	Error message	P
		+61 123456789	Error message	Error message	P
		0412345	Error message	Error message	P
002	Test create passenger with invalid Email	Input email: 123gmail.com	Error message	Error message	P
003	Test create passenger with invalid age	Input age: -22	Error message	Error message	P
004	Test create passenger with invalid name	Input name:123Maggie, Smith123	Error message	Error message	P

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
005	Test create passenger with invalid gender	Input gender: NotWoman	Error message	Error message	P

TicketTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test invalid ticket price value	Input price value: -1	Error message	Error message	P
002	Test invalid passenger	Create a ticket with: passenger=null	Error message	Error message	P
003	Test invalid flight	Create a ticket with: flight=null	Error message	Error message	P
004	Test set and get ticket_id	Set ticket_id = 123 and then get ticket_id	Return ticket_id = 123	Return ticket_id = 123	P
005	Test set and get price	Set ticket price = 100 and get price	Return ticket price = 100	Return ticket price = 100	P
006	Test set and get flight	Set flight and then get flight	Return flight	Return flight	P
007	Test set and get ClassVip	Set ClassVip = true and then get	Return true	Return true	P
008	Test set and get passenger	Set passenger and then get passenger	Return passenger	Return passenger	P
009	Test toString	Set ticket and Compare with String	Return true	Return true	P
010	Test ticket_status after creation	Get ticket_status after create a ticket	Return false	Return false	P

TicketCollectionTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass/ Fail
001	Test add invalid ticket	Create a ticket without passenger and add it to collection	Error message	Error message	P
002	Test add ticket which is already in the system	Create a ticket all list which contains a existing ticket	Error message	Error message	P
003	Test get tickets	Create a bunch of tickets and see if they were in collection	Return true	Return true	P
004	Test get ticket info	Set specific ticket id and get its info	Return true	Return true	P

Integration test cases

TicketSystemTest

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass /Fail
001	When choosing a ticket, a valid city is used	Input ticket with invalid city name : Shanghai123	Error message	Error message	P
002	Appropriate checks have been implemented to validate flight information	Input a flight with two cities which do not really exist in the ticket system. departFrom="SUZHOU" departTo="SHANGHAI"	Error message	Error message	P
003	Test for buying an already booked ticket	Set a ticket which status is true then buy this ticket	Error message	Error message	P
004	Test validate	Buy a ticket and	Age should be	Age should be	P

ID	Purpose for Test	Test Steps	Expected Result	Actual Result	Pass /Fail
	passenger information	then input different format information	positive	positive	
			phoneNumber cannot be empty	phoneNumber cannot be empty	P
			Passport number should not be more than 9 characters long	Passport number should not be more than 9 characters long	P
			cardNumber cannot be empty	cardNumber cannot be empty	P
			Invalid email format	Invalid email format	P
			Error choice for buying a ticket	Error choice for buying a ticket	P
			Security code format error	Security code format error	P
005	Test valid Ticket Information	Buy a ticket with valid input and then check the value returned by ticket.toString()	Return true	Return true	P
006	Test correct value displayed to passenger	But a ticket with valid input and then check the returned price value by ticket.gerPrice()	Return true	Return true	P