Airline Reservation System Final Project

Austin Albrechtsen
Professor Jafrina Jabin
Info-C451: System Implementation
May 3, 2024

Table of Contents:

Cover Page……1
Table of Contents……..2
Problem Statement……...3
System Requirements……..4
Functional Requirements Specifications……5-6
System Sequence Diagrams……7
Activity Diagram……8
User Interface Specification……...9-11
Traceability Matrix……12
System Architecture and System Design……13-14
Interface Design and Implementation……15
Project Plan……16
References……17

Problem Statement:

An airline ticket booking system is utilized in order to successfully manage customer information and user information. The system will be needed to also add flight data which will then need to be provided to customers as well as provide admin capabilities to modify and search for flights. The system will need to keep track of bookings and generate a ticket report. THe system will need to provide login capabilities as well as account creation capabilities.

System Requirements:

No.	Priority	Description
Req-1 (System Booking)	High	System should allow customers to book tickets with a ticket generated
Req-2 (Database)	High	System should have a database that communicates with interface input and stores this input.
Req-3 (Cancellation)	High	System should allow for users to cancel actions to provide user error forgiveness.
Req-4 (Flights)	High	System should allow for flights to be created and modified.
Req-5 (Class Bookings)	Medium	All classes should be made available to customers to book in as long as they are open.
Req-6 (User Creation)	Low	System should allow for guest login feature as well as ability to create user account
Req-7 (Customer Information)	High	System should allow for customers to input personal information and generate customer ID.
Req-8 (Search Flight)	Medium	System should allow customer to narrow flight search with specification
Req-9 (Price)	Medium	System should provide a price for booking.
Req-10 (Seats)	Low	System should allow for customers to choose the amount of seats rather than making multiple bookings.

Functional Requirements Specifications:

Stakeholders:
● Airports
● Users/Customers

Primary Actors:
● Customer- This actor can create a booking on a desired flight as well as search flights available. Actors can also upload customer information and create a user account.
● Airport- This actor can add flights as well as flight schedules, and they may view ticket reports to view all customer bookings.
Secondary Actors:
● Admin- This actor can update flight information as well as view user accounts and customer information.
● System- The system will reflect booking information as well as flight information. System will also provide user account information and customer information stored in a database.

Use Cases:

Customer- (8 total)
- Login/Logout- to authenticate customer into system(2)
- Book flight- to book flight (2)
- Booking information- to receive information about flight booking (2)
- Search Flight- to search for a flight (2)

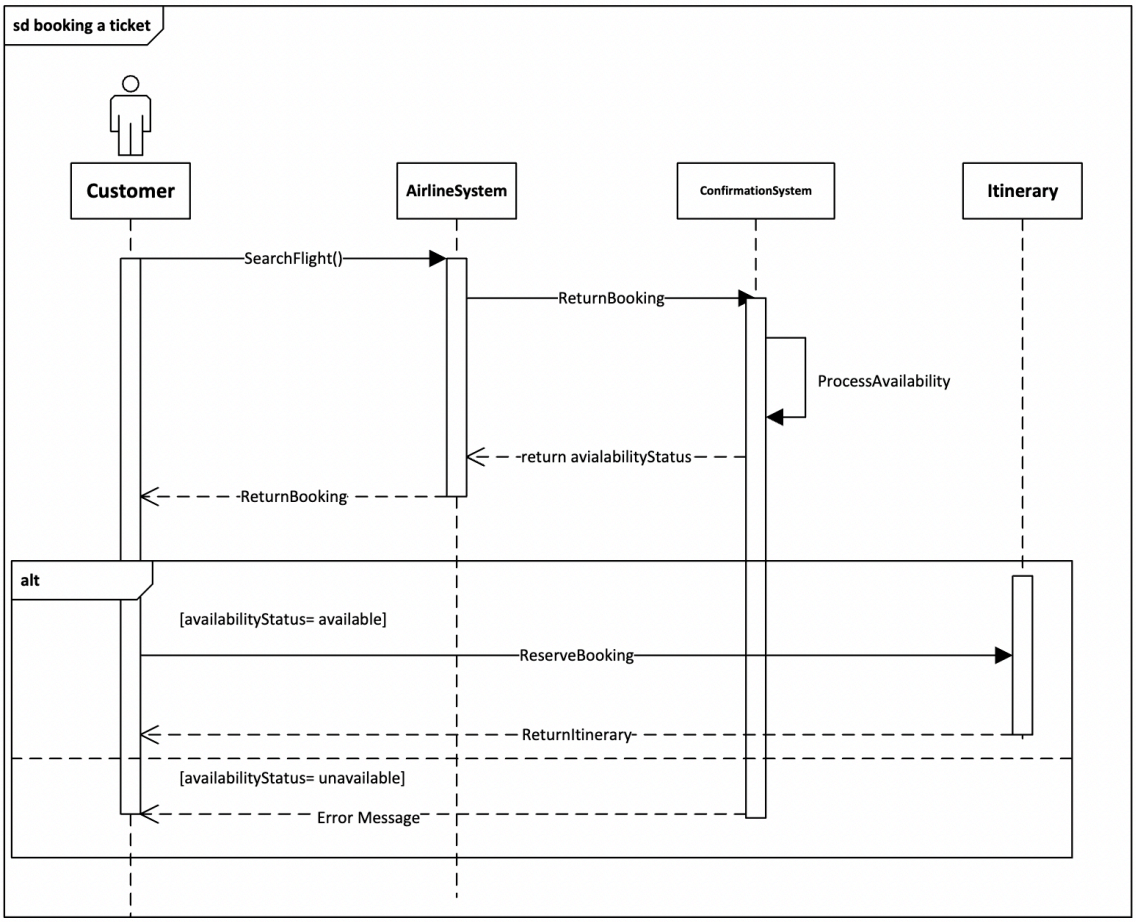
Airline- (12 total)
● Update flight- to update flight information (2)
● Login/Logout- to login/logout of the airline account or admin account (2)
● View account- to view customer booking information(2)
● Add/Remove flight- to add/remove flights on flight schedule (2)
● Take flight ticket- to provide booking number (2)
● Search Flight- to search the flight (2)

Admin- (14 total)
● Add/Remove flight- to add/remove a flight on flight schedule (2)
● Update flight- to update flight information (2)
● Login/Logout- to login/logout of admin or airline account (2)
● Update flight rate- to update the price of bookings (2)
● View Bookings- to view booking information (2)
● Update account- to update bookings(2)
● View account- to view customer booking account details (2)

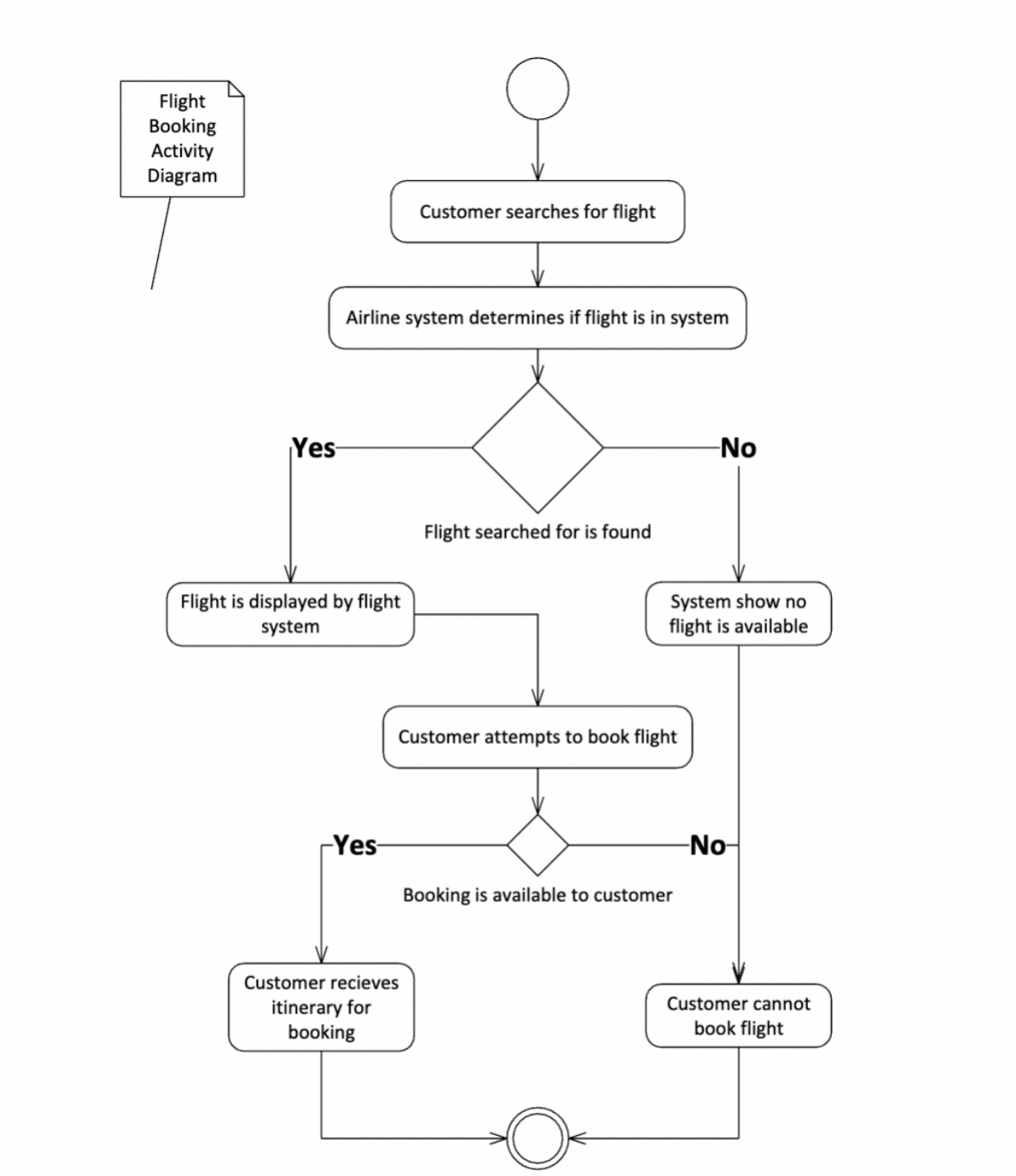
System- (8 total)
● Assign customer to booking- to assign a customer to a specific booking and flight (2)

- Display Flight- to display flight information(2)
- Display booking- to display booking that a customer has purchased (2)

System Sequence Diagrams:

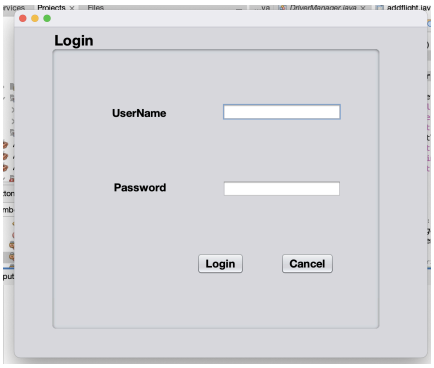


Activity Diagram:

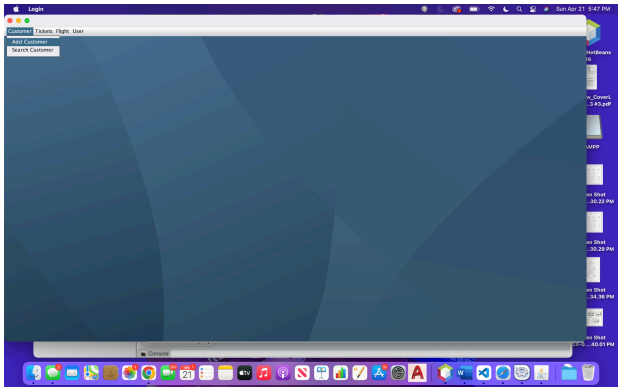


User Interface Specification:

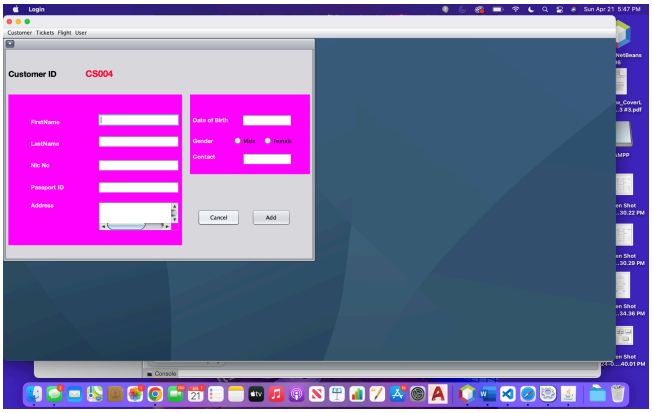
Use Case- Add Customer Information



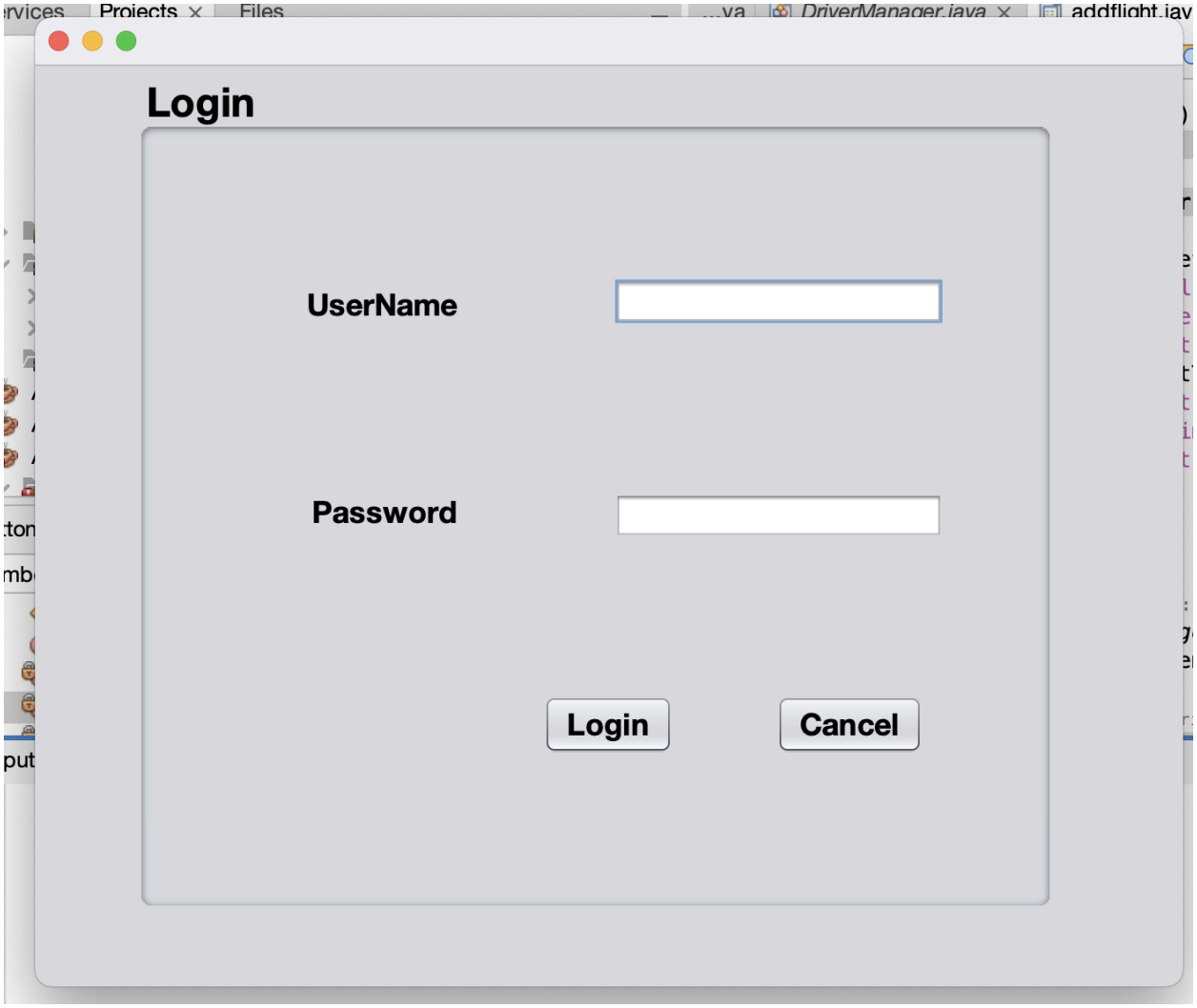
——————————>



——————————>

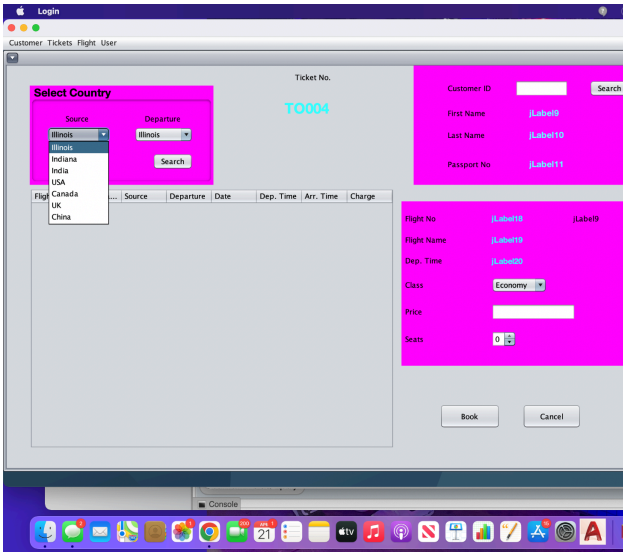


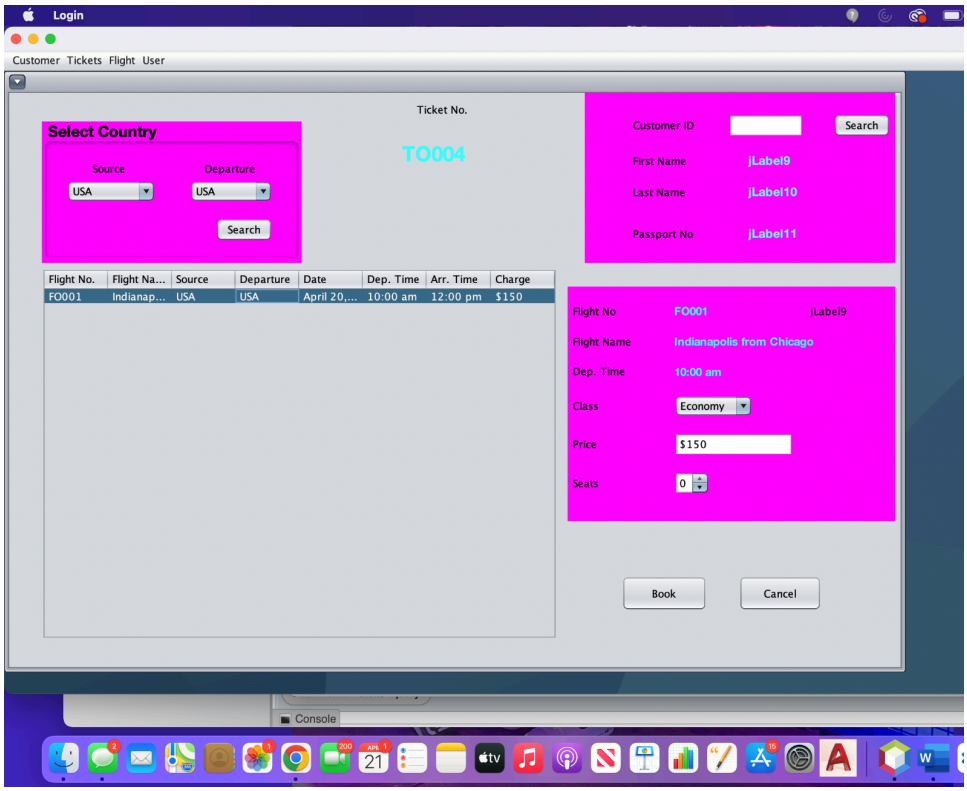
Use Case: Book TIcket

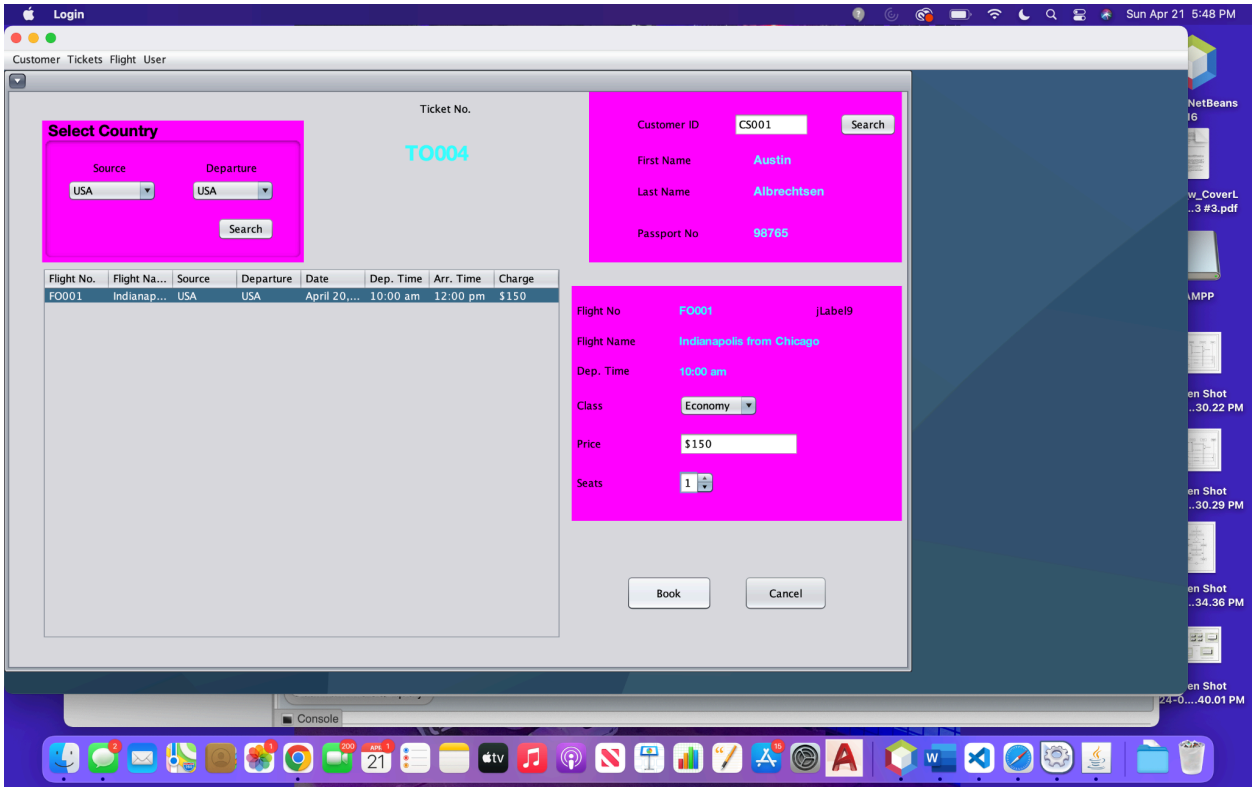

Login
UserName
Password
Login
Cancel


Login
Customer
Tickets
Flight
User
Book Ticket
Ticket Report
Console

——————————————————————>


Login
Customer
Tickets
Flight
User
Ticket No.
TO004
Select Country
Source
Departure
Illinois
Search
Indiana
India
USA
Canada
UK
China
Date
Dep. Time
Arr. Time
Charge
Customer ID
First Name
Last Name
Passport No
Flight No
Flight Name
Dep. Time
Class
Economy
Price
Seats
Book
Cancel
Console


Login
Customer Tickets Flight User
Ticket No.
Select Country
Source
Departure
USA
USA
Search
TO004
Customer ID
Search
First Name
jLabel9
Last Name
jLabel10
Passport No
jLabel11
Flight No. Flight Na... Source Departure Date Dep. Time Arr. Time Charge
FO001 Indianap... USA USA April 20,... 10:00 am 12:00 pm $150
Flight No
FO001
jLabel9
Flight Name
Indianapolis from Chicago
Dep. Time
10:00 am
Class
Economy
Price
$150
Seats
0
Book
Cancel
Console


Login
Sun Apr 21 5:48 PM
Customer Tickets Flight User
Ticket No.
Select Country
Source
Departure
USA
USA
Search
TO004
Customer ID
CS001
Search
First Name
Austin
Last Name
Albrechtsen
Passport No
98765
Flight No. Flight Na... Source Departure Date Dep. Time Arr. Time Charge
FO001 Indianap... USA USA April 20,... 10:00 am 12:00 pm $150
Flight No
FO001
jLabel9
Flight Name
Indianapolis from Chicago
Dep. Time
10:00 am
Class
Economy
Price
$150
Seats
1
Book
Cancel
Console

Traceability Matrix:

REQ's	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
REQ1	4	X							
REQ2	3	X	X						
REQ3	5	X						X	
REQ4	5	X		X	X	X	X		
REQ5	2	X				X		X	X
REQ6	1	X					X		
REQ7	4	X	X						
REQ8	4			X			X		
REQ9	2	X						X	
REQ10	3	X			X				X
MAX PW		5	4	4	1	3	3	2	3
TOTAL PW		15	7	5	3	2	4	2	1

Use Cases:

No.	Description
UC1	Book a flight as a customer
UC2	Update flight info
UC3	Add a flight as an airline
UC4	Create Login
UC5	Create Customer Profile
UC6	Remove Flights as an airline
UC7	Receive a seat assignment as a customer booking a flight
UC8	Customer should be able to choose which class they would like to fly in for booking

System Architecture and System Design:

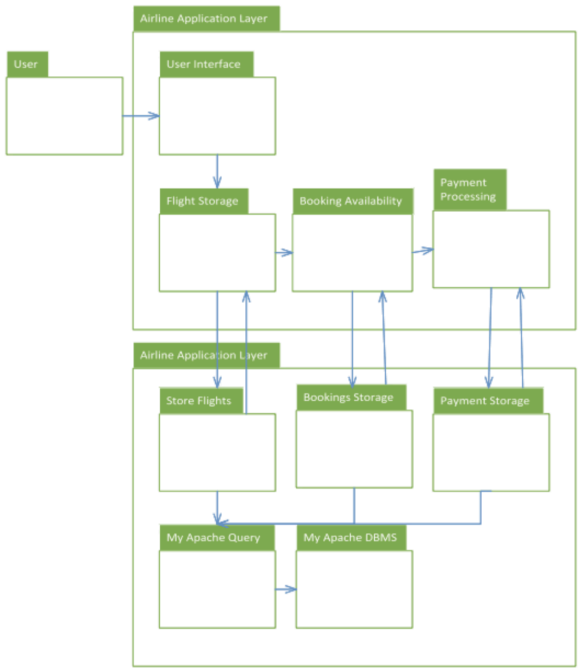
Architecture Style:

The airline system is a client-server architecture style. The system can function autonomously because the producer in this case is the airline as they provide the flights that the customer is able to book. The customer has requests in the form of flights that they wish to book, and they send requests to the server to book flights. The airline acts as the server, and, through providing available flights, they meet customer needs as the customer is the client, by addings flights to book. My system uses visual code studio in order to create code, and this allows for the server(airline) to add the flights for customers to book. The customer then would book flights through the database that runs through Apache Netbeans in order to choose what type of booking that they wish to proceed through. This system can run on only one machine as the data stored through the server can also be utilized through the database in Apache.

Identifying Subsystems:

The UML package diagram portrays how the airline booking system interacts in order to complete customer bookings. THis system as mentioned above is a client-server system, and the user interface consists of components derived from what the airline system inputs into the system, being the flights available. The apache database then stores the available flights as well as the booking availability, and this allows the customer to choose what flight and class they would like to choose. The system then determines what bookings are available for the customer to choose from. Then, the payment interface determines if the details that the customer inputs are valid for the customer to reserve a booking for the customer to reserve. The database then communicates with the interface by updating the availability of flights by taking away the availability of a booking that the customer desires if the payment is valid.

UML Package Diagram:



Persistent Data Storage:

This system requires storage from the Apache database system as the airline adds flight availability. The flights are added into the system by the airline, and the bookings of different class availability are added for the customer to choose from. The customer will then attempt to reserve as booking after choosing by entering card information which the system will determine the accuracy of. If the payment information is valid, the system will also store the payment information in Apache.

Global Control Flow

Execution Orders:

The airline system will conduct operations in the same order for every customer. The customer will first choose a flight that they would like to book with. The customer will then choose the class that they wish to book with, and the system will determine if this booking is available. If available, the customer will enter payment information in the form of card details in order to reserve a booking. The system will then determine if the payment information is valid, and, if valid, the system will remove the availability of this specific booking from the flight that was inputted.

Time Dependency:

This system is time dependent because the flights that are added by the airline will no longer have availability of the flight time and date have passed. In this case, the flight will no longer be available for the customer to book. Booking from a customer also could cause the flight to be no longer for the customer to book, and, if a flight is full, the customer will no longer be able to choose to book this flight. The Apache database will determine that the flight is fully booked, and it will update to the system that the flight is no longer able to be booked by the customer.

Hardware Requirements:

The airline system will require Apache to be run in order to be run correctly as well as have github in order to retrieve the code needed to run the system.

- Color Display: Minimum Resolution: 640 x 480 pixels
- Desktop: Computer or Mobile Device
- Memory: 1 gB of RAM
- Hard Drive: 1 gB of free space on drive

Interface Design and Implementation:

Design of Tests

- List and describe the test cases that will be programmed and used for unit testing of your system.
    - Book Flight: customer may book a flight
    - Search Flight: customer and admin may search flight by flight number
    - Modify Flight: admin may modify flight after searching
    - Login: user logs in with username and password
    - Create User: user enter personal info and creates user in system
    - Generate Ticket Report: admin may generate report of all bookings
    - Add Flight: Admin may add flights to system
- Discuss the test coverage of your tests.
    - User testing will be done by creating multiple accounts, flights and bookings. After this, I will test login of the user accounts created. Then I will test bookings by generating a report as well as confirming that the report of bookings is also reflected in the database. Similarly, I will test flight use cases by creating and modifying flights, and I will confirm that these match the database that is connected. Lastly, I will test user creation by confirming that accounts created are in the database and allowed to be authenticated through the login screen.
- Describe the Integration Testing strategy and plans on how you will conduct it.
    - My code already is designed to be integrated as one so testing integration of use cases will be a matter of running the system. I am struggling to add a payment method in my system, so I am determining if this will be something my system will cover or not as integrating it seems unlikely.
- Describe your plans for testing algorithms, non-functional requirements, or user interface requirements that you stated in previous reports.

I have already begun testing interface requirements as I have written the code and designed the layouts as shown in my screenshots. I plan to just run my system and make sure that the database is also updating accurate information.

Project Plan:

- W1-2: determine the framework and establish the structure of the system, connecting front-end to back-end, connecting back-end to database (**Completed)**
- W3-4: build system login, customer booking abilities, and flight schedules (**Complete)**
- W5-7: Implement basic features for airline; add and cancel flight and implement customer ability to purchase flight and see flight details (**Completed)**
- W8: test accomplished features and record demo for midterm (**Completed)**
- W9-11: improve the current features based on customers' feedback if there are any, or continue working on implementing the basic features for customers booking flights; the system for airlines to add or cancel flights (**Completed)**
- W12-14: writing test cases for the implemented features or continue building possible features you'd like to implement **(Completed)**
- W15: record demo for final presentation **(Completed)**

References:

Tilley, Scott R., and Harry J. Rosenblatt. *Systems Analysis and Design*. 11th ed., Course Technology Cengage Learning, 2017.