SSN COLLEGE OF ENGINEERING
KALAVAKKAM-603110

Department of Computer Science and Engineering

UCS2313 – Object Oriented Programming Lab

II Year CSE - (III Semester)

Airline Booking System

Academic Year 2022-23

Batch: 2021- 2025

Faculty Incharge : S.Rajalakshmi

Project Students:

Parthiban M	3122 21 5001 065
Rohith M	3122 21 5001 085
Rajkumar S	3122 21 5001 077

Problem Statement:

The main objective of the Airline Booking System is to design and implement an efficient and flawless system which ultimately helps the employees of the airline system to issue reservation tickets for various air flights and maintain the records of various passengers and provide quick services to the passengers. Existing Airline reservation system includes various old processes such as maintaining the inventory, fares, enquiries and the reservation details of each passenger.

The main objective of the project includes:

- Higher productivity and effective management.
- Security and protection of confidential data pertaining to the passenger
- Maintaining the records of the passengers
- Reservation and cancellation of the tickets at their own convenience place
- Details about the availability of seats in the flight
- Quick information regarding the ongoing offers and discounts (if any)
- To minimize the repetitive work done by the system administrator and reservation clerks.

Motivation for the Problem:

The conventional method of ticket reservation involves physically traveling to the airport and waiting in long queues. There are scenarios where people are not able to get reservations due to their inability to stand in queues for long periods of time or the tickets got sold out by the time they reached the airport. There are also cases of overbooking, where the airline sells more tickets than the number of seats available, to avoid empty seats by no show passengers. Additionally, there is a need to manually fill the paper document and submit it to the respective authorities for booking the tickets. Working men often spend most of their time working and barely have any time for themselves. Traveling to the airport and waiting in queues for a ticket is difficult for them if there is an urgent meeting in some other place. This system provides an easy interface to book or cancel a ticket at any time at their convenience.

Scope and Limitations:

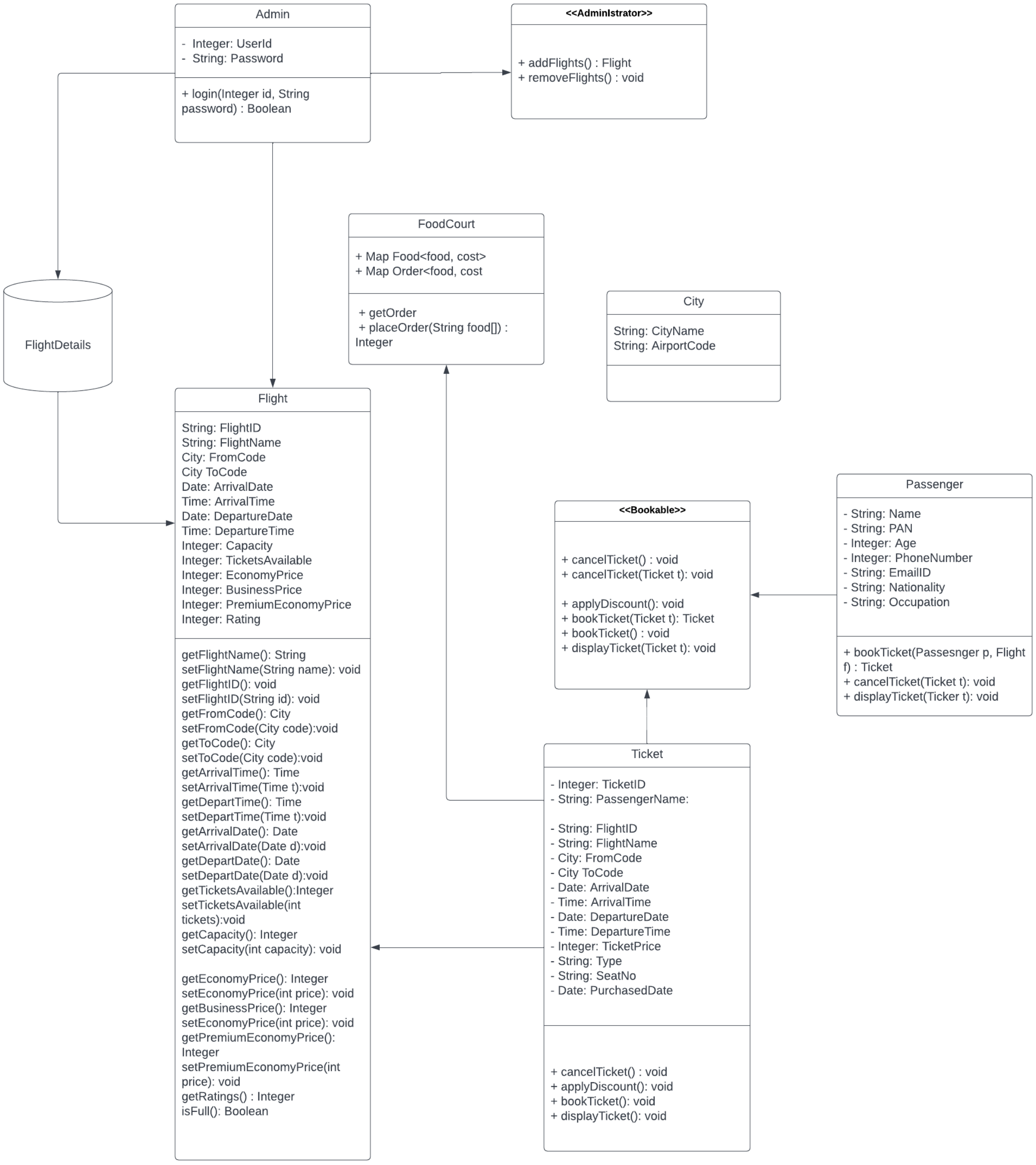
Scope of the project includes:

- Developing classes to represent different aspects of the airline booking system, such as the flights, food options, passengers, and tickets.
- Implementing various functions and methods to perform various tasks related to the booking process, purchasing food options, and generating tickets.
- Integrating the system with CSV files to store and retrieve data about flights, passengers and tickets.

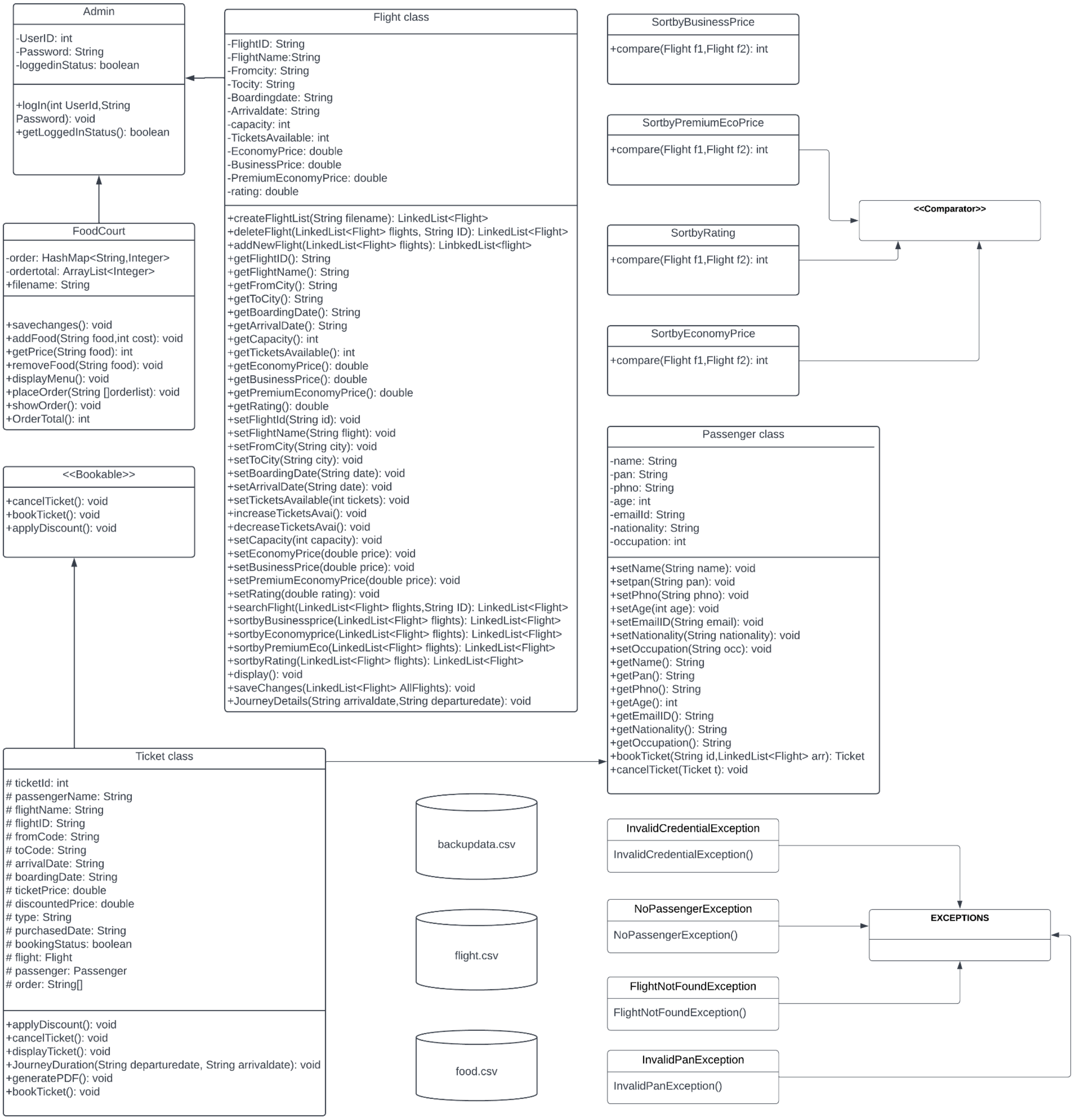
Limitations for the project include:

- Time and resources: Developing a comprehensive airline booking system can be a complex and time consuming task, and may require a significant amount of resources such as the data pertaining to the flight, tickets, passengers, etc.
- Data accuracy: The accuracy of the data in the system ,such as the flight schedules and availability may be limited to external factors such as delays, cancellations and climatic conditions.
- Scalability: The system may need to handle a large number of users and transactions, and may need to be scalable to support future growth.

Class Diagram (Old):



Class Diagram (New):


Admin
-UserID: int
-Password: String
-loggedinStatus: boolean
+logIn(int UserId,String Password): void
+getLoggedInStatus(): boolean
Flight class
-FlightID: String
-FlightName:String
-Fromcity: String
-Tocity: String
-Boardingdate: String
-Arrivaldate: String
-capacity: int
-TicketsAvailable: int
-EconomyPrice: double
-BusinessPrice: double
-PremiumEconomyPrice: double
-rating: double
+createFlightList(String filename): LinkedList<Flight>
+deleteFlight(LinkedList<Flight> flights, String ID): LinkedList<Flight>
+addNewFlight(LinkedList<Flight> flights): LinbkedList<flight>
+getFlightID(): String
+getFlightName(): String
+getFromCity(): String
+getToCity(): String
+getBoardingDate(): String
+getArrivalDate(): String
+getCapacity(): int
+getTicketsAvailable(): int
+getEconomyPrice(): double
+getBusinessPrice(): double
+getPremiumEconomyPrice(): double
+getRating(): double
+setFlightId(String id): void
+setFlightName(String flight): void
+setFromCity(String city): void
+setToCity(String city): void
+setBoardingDate(String date): void
+setArrivalDate(String date): void
+setTicketsAvailable(int tickets): void
+increaseTicketsAvai(): void
+decreaseTicketsAvai(): void
+setCapacity(int capacity): void
+setEconomyPrice(double price): void
+setBusinessPrice(double price): void
+setPremiumEconomyPrice(double price): void
+setRating(double rating): void
+searchFlight(LinkedList<Flight> flights,String ID): LinkedList<Flight>
+sortbyBusinessprice(LinkedList<Flight> flights): LinkedList<Flight>
+sortbyEconomyprice(LinkedList<Flight> flights): LinkedList<Flight>
+sortbyPremiumEco(LinkedList<Flight> flights): LinkedList<Flight>
+sortbyRating(LinkedList<Flight> flights): LinkedList<Flight>
+display(): void
+saveChanges(LinkedList<Flight> AllFlights): void
+JourneyDetails(String arrivaldate,String departuredate): void
SortbyBusinessPrice
+compare(Flight f1,Flight f2): int
SortbyPremiumEcoPrice
+compare(Flight f1,Flight f2): int
<<Comparator>>
SortbyRating
+compare(Flight f1,Flight f2): int
SortbyEconomyPrice
+compare(Flight f1,Flight f2): int
FoodCourt
-order: HashMap<String,Integer>
-ordertotal: ArrayList<Integer>
+filename: String
+savechanges(): void
+addFood(String food,int cost): void
+getPrice(String food): int
+removeFood(String food): void
+displayMenu(): void
+placeOrder(String []orderlist): void
+showOrder(): void
+OrderTotal(): int
<<Bookable>>
+cancelTicket(): void
+bookTicket(): void
+applyDiscount(): void
Passenger class
-name: String
-pan: String
-phno: String
-age: int
-emailId: String
-nationality: String
-occupation: int
+setName(String name): void
+setpan(String pan): void
+setPhno(String phno): void
+setAge(int age): void
+setEmailID(String email): void
+setNationality(String nationality): void
+setOccupation(String occ): void
+getName(): String
+getPan(): String
+getPhno(): String
+getAge(): int
+getEmailID(): String
+getNationality(): String
+getOccupation(): String
+bookTicket(String id,LinkedList<Flight> arr): Ticket
+cancelTicket(Ticket t): void
Ticket class
# ticketId: int
# passengerName: String
# flightName: String
# flightID: String
# fromCode: String
# toCode: String
# arrivalDate: String
# boardingDate: String
# ticketPrice: double
# discountedPrice: double
# type: String
# purchasedDate: String
# bookingStatus: boolean
# flight: Flight
# passenger: Passenger
# order: String[]
+applyDiscount(): void
+cancelTicket(): void
+displayTicket(): void
+JourneyDuration(String departuredate, String arrivaldate): void
+generatePDF(): void
+bookTicket(): void
backupdata.csv
flight.csv
food.csv
InvalidCredentialException
InvalidCredentialException()
NoPassengerException
NoPassengerException()
FlightNotFoundException
FlightNotFoundException()
InvalidPanException
InvalidPanException()
EXCEPTIONS

Identification, Implementation and Integration of Modules

Module 1 Creation of CSV files

- Separate CSV files are created to store the information of flight details, ticket details and food details that are being served in the flights.
- Flight details include flight id, flight name, boarding code, arrival code, boarding date and time, arrival date and time, capacity of the flight, business price, economy price, premium economy price, rating and available tickets for each flight
- Food details are stored in the file.csv and includes the details of food along with the cost.
- The ticket details for each passenger such as the ticket Id, Passenger name, passenger phone number and PAN are stored in ticket.csv.
- The modifications done in the program are also reflected in the CSV files which makes dynamic readability for the admin.
- These CSV files are exclusively accessible only by the admin.

Module 2 Creation of various classes

- In an Airline Booking system, various classes are created to represent the different aspects of the system. These classes are designed to store and manage the data and to provide functions and methods for interacting with the data.
- The Admin class allows the user to choose any one of the two available roles (Admin or User) where each role has special functionalities for better manipulation of data.
- Flight class is created to represent the particular flight with various attributes related to the flight and it also includes the various methods and variables for the effective manipulation of the imported data.
- The FoodCourt class is created to represent the food options available to purchase during their comfortable flight journey. The flight.csv provides the list of items and their corresponding fare which the user can purchase their loved recipes to enhance their journey in a more luxurious manner.
- The Passenger class represents the passenger on a flight with attributes related to each person. The passenger can enhance their journey by opting for any of the three level comfort zones available in the flight. These exclusive features provide the passengers to travel in a more luxurious way.

- The Ticket class represents the ticket for the flight and includes exclusive discount offers according to their occupation. This saves the money of the passenger without affecting their luxurious travel experience.

Module 3 Implementation of major functionalities

Admin class

- This module is used to implement the functionalities of the admin.
- Major functionalities include addition or deletion of data entry in the CSV file
- Only the admin has the privilege to make changes in the CSV files.
- The admin is given his own credentials for logging in.

Flight class

- The Flight module represents the flight with several attributes such as the Flight-Id, Flight Name, journey details such as the Boarding date, arrival date, boarding place and arrival place along with their time, capacity of the flight, available tickets for the respective flight, and various comfort options such as the economy class, Business class, Premium Economy class along with the ratings.
- Flight class has a constructor that allows the creation of a new flight object by specifying the values of these attributes.
- It also has getter and setter for each attribute which allows it to retrieve and modify the values of the attributes.
- This class also consists of methods to sort the flight by various criterias such as sorting by the rating of the flight, sorting by the fare of various comfort zones available in the flight.
- Journey duration is also calculated and sent to the passenger.
- The Flight data is manipulated by only the admin.

FoodCourt class

- Additional functionalities include the provision for ordering food.
- The FoodCourt class in the program provides the food menu for a flight. It uses the Map to store the items of the menu and their corresponding prices.

- The details of the food are read from the food.csv and the passenger has the freedom to choose any of the food from the menu.
- The add-food method is used to add the selected food along with their cost into the menu and removeFood method is used to remove the food from the menu. The passenger can select the food of their interest and can place the order and look into the menu using the showorder method.

Passenger class

- The Passenger class represents the passenger on the flight with attributes such as the passenger's name, PAN, age, phone number, email address, nationality and the occupation.
- This class has a constructor that allows the creation of a passenger object by specifying the attributes of the passenger class. This class also has the separate getters and setters methods which allows retrieval and modify the values of the attributes.
- The passenger has the privilege to book a ticket which creates a separate object for the ticket class and cancels the ticket according to the interest of the passenger.

Ticket class

- The ticket class represents the ticket for a flight with attributes such as ticket-id, passenger-name, flight-id, flight name, flight code (from and to), boarding date and arrival date, price of journey along with the discounts (if applicable), booked date and the status of the ticket.
- Ticket class has the constructor which takes the passenger and flight details along with the comfort zone selected by the passenger as the parameters.
- The passenger has the option to book the ticket and cancel the ticket, also the ticket can be viewed once the ticket has been booked and the ticket in PDF format is automatically generated at the end.
- Armed forces are provided with the discount of 10% and the school students receive a 30% discount from the total ticket price for their travel journey.
- iText and PDFBox is an open source java library used for the conversion of the ticket generated into the Portable Document Format (PDF).

Module 4 Implementation of additional functionalities

CSV files

- All the data pertaining to the flights are stored in the flight.csv such as the flight id ,flight name,boarding code,arrival code,boarding date,boarding time,arrival date, arrival time, capacity, eco price, business price,prem eco price, rating, available tickets are stored in the file and are imported into the major modules of the program.
- All the data pertaining to the food being served in the flight are stored in the food.csv file which plays a vital role in the major classes such as the FoodCourt and the Ticket.
- All the information related to the passenger and the tickets are stored in the ticket.csv file which can be accessed only by the admin.

File Handling

- File handling in Java is used to import all the data stored in the CSV files into the program for further manipulation.
- The changes or modifications done in the program are also reflected into the CSV files through the FileReader and FileWriter functions of the file handling technique.
- All the changes are auto saved by the program and can be viewed or modified according to the role in the system.

PDF Generation

- Passengers can select any of the comfort zones in the flight and have an option to order their favorite dishes online or even in person during the flight journey.
- After the required details are filled, they can select any available flights to their destination and confirm their ticket booking by entering the flightId.
- Once all the above information is done, the discounts (if applicable) are done in the Ticket class and the final ticket can be viewed by the passengers.
- The tickets are automatically downloaded as a PDF file which makes the passenger easy to view and print the generated PDF file if needed.

Module 5 Exception handling

- **NoPassengerException**

NoPassengerException is raised when there is an error related to the passenger. For example, if a passenger's booking is accessed or modified, but no passenger with the specified booking number exists, a NoPassengerException is thrown.

- **InvalidCredentialException**

An InvalidCredentialException is thrown when a user or admin attempts to login or access certain features with invalid or incorrect credentials. For example, if a user or admin tries to login with an invalid username or password, an InvalidCredentialException is thrown.

- **FlightNotFoundException**

A FlightNotFoundException is thrown when a flight cannot be found or accessed. For example, if a user searches for a flight with a given flight Id, but no flight exists with that flight Id exists and hence FlightNotFoundException is thrown.

- **InvalidPanException**

An InvalidPanException is thrown when a program is expecting a valid PAN but receives an invalid one. A PAN(Personal Account Number) is a unique identification number issued to individuals and organizations by the government to track financial transactions. For example, if the program is trying to validate the PAN and the PAN is found to be in the incorrect format or does not match the expected pattern, an InvalidPanException is thrown.

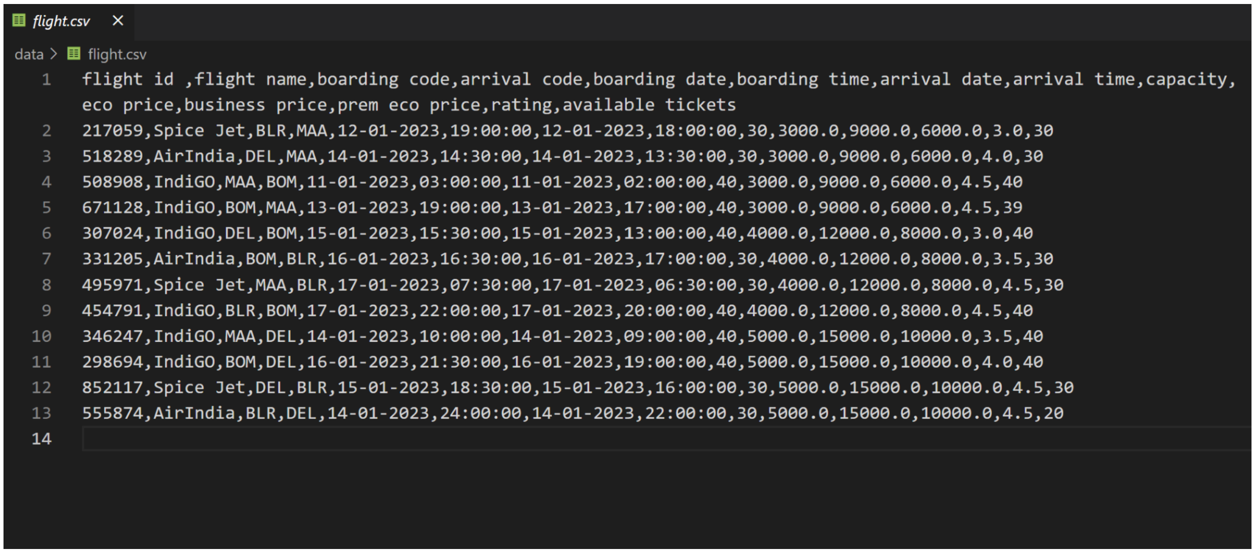
Object Oriented Concepts Used:

1. Polymorphism
2. Inheritance
3. Collection Frameworks
4. Data Encapsulation
5. Coupling
6. File Handling
7. Abstraction

Output snapshot

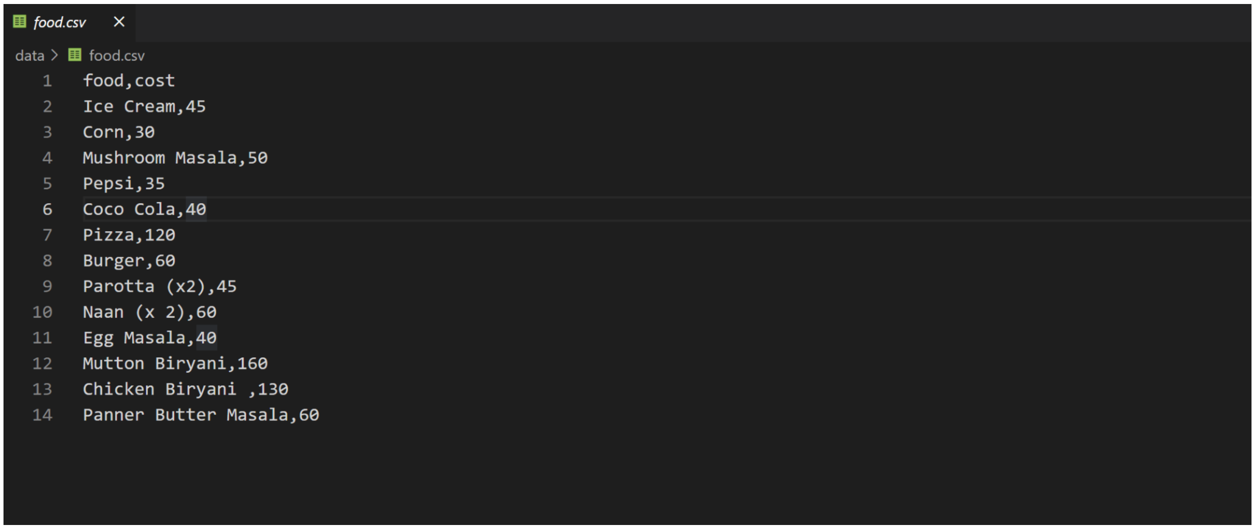
CSV files

flight.csv



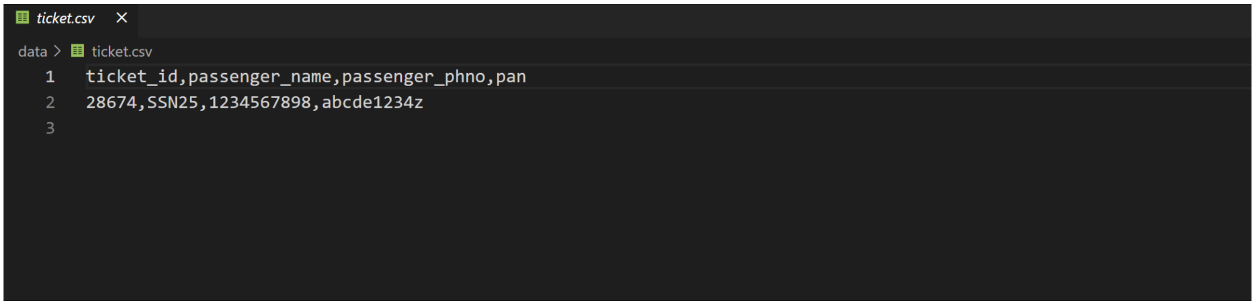
```
flight id ,flight name,boarding code,arrival code,boarding date,boarding time,arrival date,arrival time,capacity,eco price,business price,prem eco price,rating,available tickets
217059,Spice Jet,BLR,MAA,12-01-2023,19:00:00,12-01-2023,18:00:00,30,3000.0,9000.0,6000.0,3.0,30
518289,AirIndia,DEL,MAA,14-01-2023,14:30:00,14-01-2023,13:30:00,30,3000.0,9000.0,6000.0,4.0,30
508908,IndiGO,MAA,BOM,11-01-2023,03:00:00,11-01-2023,02:00:00,40,3000.0,9000.0,6000.0,4.5,40
671128,IndiGO,BOM,MAA,13-01-2023,19:00:00,13-01-2023,17:00:00,40,3000.0,9000.0,6000.0,4.5,39
307024,IndiGO,DEL,BOM,15-01-2023,15:30:00,15-01-2023,13:00:00,40,4000.0,12000.0,8000.0,3.0,40
331205,AirIndia,BOM,BLR,16-01-2023,16:30:00,16-01-2023,17:00:00,30,4000.0,12000.0,8000.0,3.5,30
495971,Spice Jet,MAA,BLR,17-01-2023,07:30:00,17-01-2023,06:30:00,30,4000.0,12000.0,8000.0,4.5,30
454791,IndiGO,BLR,BOM,17-01-2023,22:00:00,17-01-2023,20:00:00,40,4000.0,12000.0,8000.0,4.5,40
346247,IndiGO,MAA,DEL,14-01-2023,10:00:00,14-01-2023,09:00:00,40,5000.0,15000.0,10000.0,3.5,40
298694,IndiGO,BOM,DEL,16-01-2023,21:30:00,16-01-2023,19:00:00,40,5000.0,15000.0,10000.0,4.0,40
852117,Spice Jet,DEL,BLR,15-01-2023,18:30:00,15-01-2023,16:00:00,30,5000.0,15000.0,10000.0,4.5,30
555874,AirIndia,BLR,DEL,14-01-2023,24:00:00,14-01-2023,22:00:00,30,5000.0,15000.0,10000.0,4.5,20
```

food.csv



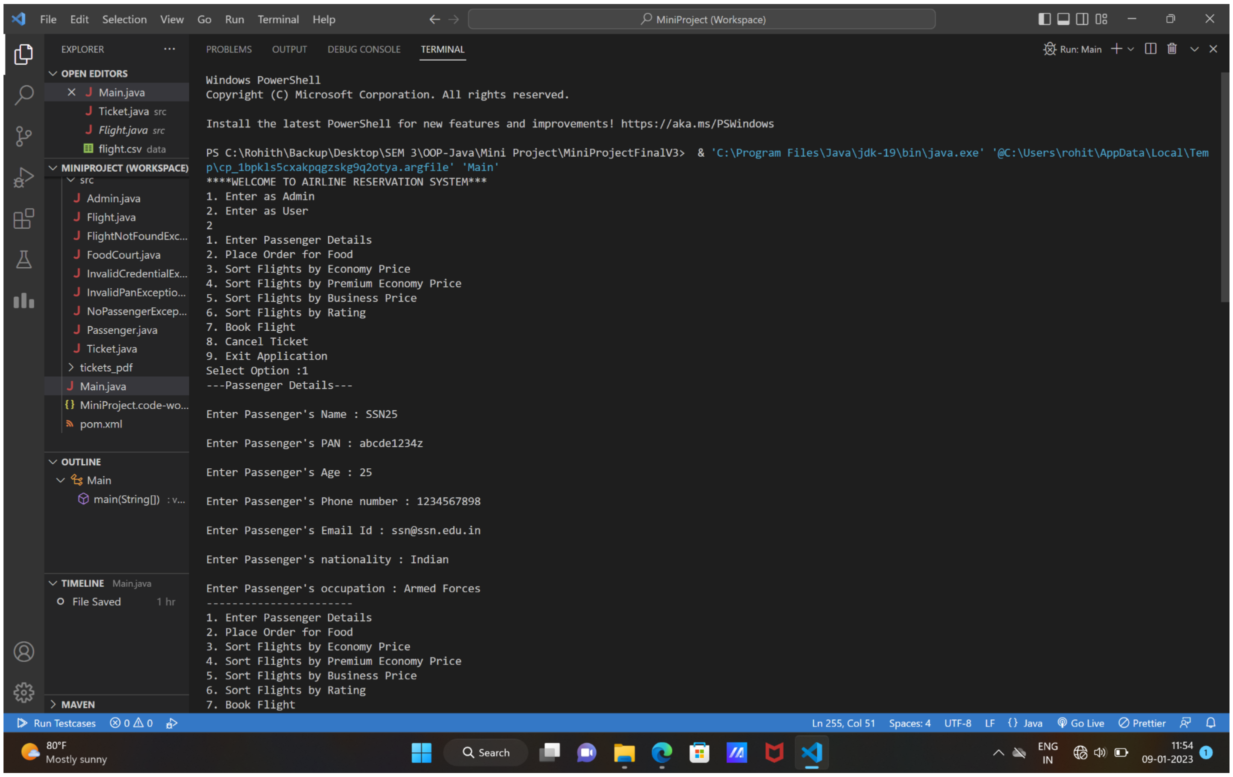
```
food,cost
Ice Cream,45
Corn,30
Mushroom Masala,50
Pepsi,35
Coco Cola,40
Pizza,120
Burger,60
Parotta (x2),45
Naan (x 2),60
Egg Masala,40
Mutton Biryani,160
Chicken Biryani ,130
Panner Butter Masala,60
```

ticket.csv

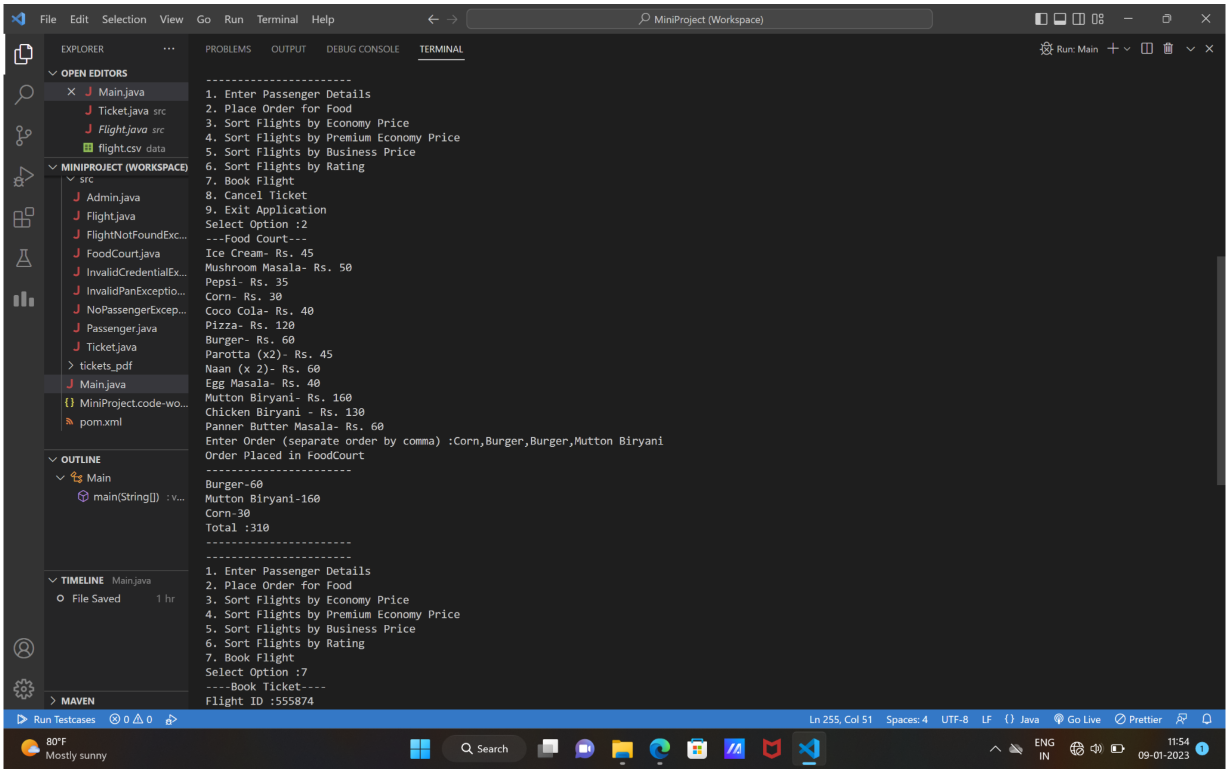


```
ticket_id,passenger_name,passenger_phno,pan
28674,SSN25,1234567898,abcde1234z

```

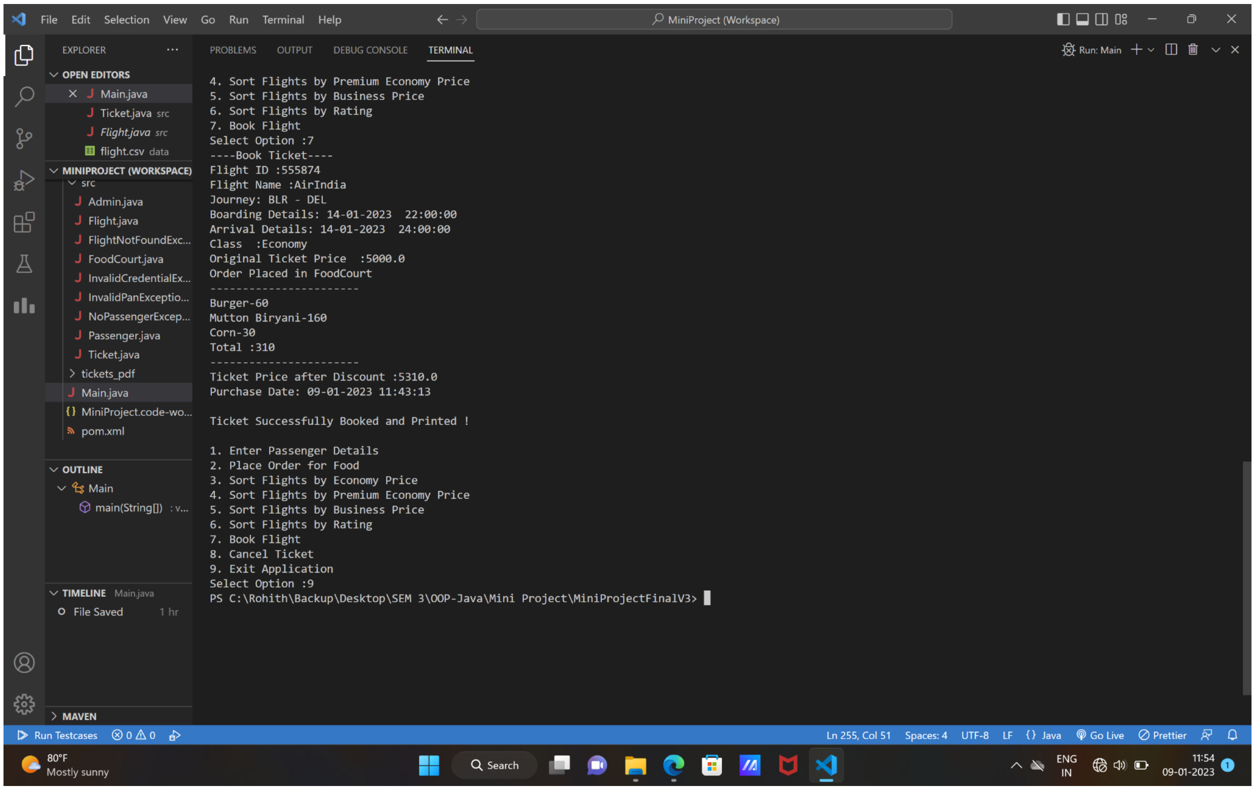
Console



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Mini Project\MiniProjectFinalV3>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '@C:\Users\rohit\AppData\Local\Tem
p\cp_1bpkls5cxakpqgzskg9q2otya.argfile' 'Main'
****WELCOME TO AIRLINE RESERVATION SYSTEM***
1. Enter as Admin
2. Enter as User
2
1. Enter Passenger Details
2. Place Order for Food
3. Sort Flights by Economy Price
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
8. Cancel Ticket
9. Exit Application
Select Option :1
---Passenger Details---

Enter Passenger's Name : SSN25

Enter Passenger's PAN : abcde1234z

Enter Passenger's Age : 25

Enter Passenger's Phone number : 1234567898

Enter Passenger's Email Id : ssn@ssn.edu.in

Enter Passenger's nationality : Indian

Enter Passenger's occupation : Armed Forces
----------------------
1. Enter Passenger Details
2. Place Order for Food
3. Sort Flights by Economy Price
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
```

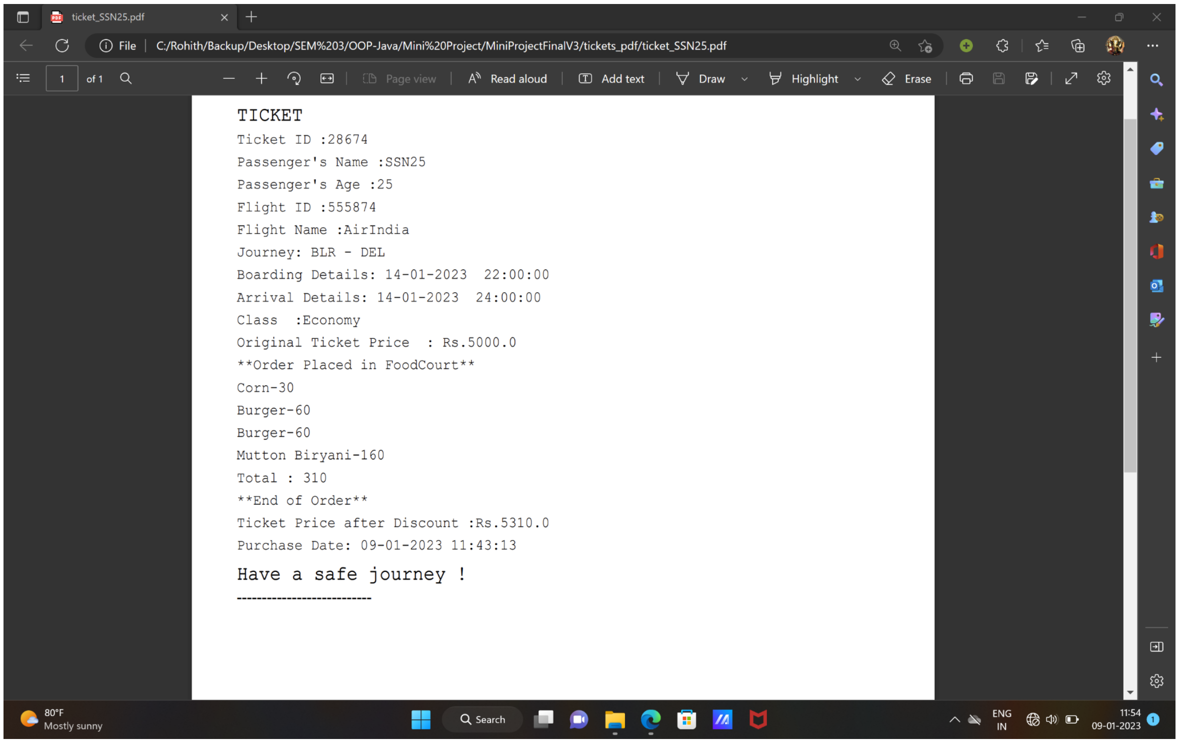


```
----------------------
1. Enter Passenger Details
2. Place Order for Food
3. Sort Flights by Economy Price
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
8. Cancel Ticket
9. Exit Application
Select Option :2
---Food Court---
Ice Cream- Rs. 45
Mushroom Masala- Rs. 50
Pepsi- Rs. 35
Corn- Rs. 30
Coco Cola- Rs. 40
Pizza- Rs. 120
Burger- Rs. 60
Parotta (x2)- Rs. 45
Naan (x 2)- Rs. 60
Egg Masala- Rs. 40
Mutton Biryani- Rs. 160
Chicken Biryani - Rs. 130
Panner Butter Masala- Rs. 60
Enter Order (separate order by comma) :Corn,Burger,Burger,Mutton Biryani
Order Placed in FoodCourt
----------------------
Burger-60
Mutton Biryani-160
Corn-30
Total :310
----------------------
----------------------
1. Enter Passenger Details
2. Place Order for Food
3. Sort Flights by Economy Price
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
Select Option :7
----Book Ticket----
Flight ID :555874
```



```
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
Select Option :7
----Book Ticket----
Flight ID :555874
Flight Name :AirIndia
Journey: BLR - DEL
Boarding Details: 14-01-2023  22:00:00
Arrival Details: 14-01-2023  24:00:00
Class  :Economy
Original Ticket Price  :5000.0
Order Placed in FoodCourt
----------------------
Burger-60
Mutton Biryani-160
Corn-30
Total :310
----------------------
Ticket Price after Discount :5310.0
Purchase Date: 09-01-2023 11:43:13

Ticket Successfully Booked and Printed !

1. Enter Passenger Details
2. Place Order for Food
3. Sort Flights by Economy Price
4. Sort Flights by Premium Economy Price
5. Sort Flights by Business Price
6. Sort Flights by Rating
7. Book Flight
8. Cancel Ticket
9. Exit Application
Select Option :9
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Mini Project\MiniProjectFinalV3>
```

Ticket.pdf


ticket_SSN25.pdf
C:/Rohith/Backup/Desktop/SEM%203/OOP-Java/Mini%20Project/MiniProjectFinalV3/tickets_pdf/ticket_SSN25.pdf

```
TICKET
Ticket ID :28674
Passenger's Name :SSN25
Passenger's Age :25
Flight ID :555874
Flight Name :AirIndia
Journey: BLR - DEL
Boarding Details: 14-01-2023  22:00:00
Arrival Details: 14-01-2023  24:00:00
Class  :Economy
Original Ticket Price  : Rs.5000.0
**Order Placed in FoodCourt**
Corn-30
Burger-60
Burger-60
Mutton Biryani-160
Total : 310
**End of Order**
Ticket Price after Discount :Rs.5310.0
Purchase Date: 09-01-2023 11:43:13
Have a safe journey !
----------------------------
```

Conclusion

- An Airline Booking System in Java could consist of several classes such as Flight, FoodCourt, Passenger, Admin and Ticket classes to represent the different aspects of the system. The ‘Flight’ class could store information about the particular flight, such as the flight number, departure and arrival cities, date and time, capacity, fare for each comfort zone etc. The ‘FoodCourt’ class could store a list of items available for purchase on the flight and the corresponding prices. The ‘Passenger’ class could store the information of the passenger, such as the name, email address, phone number, PAN, age, occupation, nationality, etc. The ticket class could store the information about a ticket, such as the passenger, flight, price and discounts if applicable.
- Together, these classes could be used to manage the booking and reservation process for an airline. The ‘Flight’ class can be used to keep track of the flight. The ‘FoodCourt’ class is used to store and display the food options available in the flight, the ‘Passenger’ class can be used to store the information of the passenger and the flights. The ‘Ticket’ class is used to generate the ticket for the passenger. The CSV files are used to import the stored information of the flight and food details into the program.
- Overall, the Airline Booking System in Java provides a convenient and efficient way for users to book and manage their flights.

References

- Java The Complete Reference 9th Edition - Herbert Schildt
- Head First Java - Kathy Sierra and Bert Bates
- Java for Dummies - Barry A Burd
- www.youtube.com
- www.quora.com