AIRLINE RESERVATION SYSTEM

Submitted by

ARUN J (RA2311026020077)

TAMIL SELVAN K (RA2311026020090)

Under the guidance of

Mr Dinesh G. T

RAMAPURAM, CHENNAI-600089

DECEMBER 2025

TABLE OF CONTENTS

ABSTRACT

The Airline Reservation System is a robust and efficient software solution designed to revolutionize the traditional ticket booking and flight management process by leveraging the structural power of Object Oriented Programming (OOP) and the data persistence capabilities of Relational Database Management Systems (RDBMS). This comprehensive platform introduces a smarter approach to aviation management through its JDBC-integrated architecture, which allows system administrators to seamlessly store, retrieve, and manage passenger records with high precision. By automatically validating seat availability in real-time and dynamically calculating fares based on specific cabin classes—such as Economy, Premium Economy, and Business—the system minimizes human error, eliminates manual calculation redundancies, and significantly accelerates the reservation transaction cycle. One of the standout features of this system is its intelligent inventory control mechanism, which interacts with the SQL database to strictly prevent overbooking scenarios by instantly updating seat counts upon every successful transaction. This not only ensures absolute data integrity but also boosts operational reliability and improves the overall user experience by providing instant booking confirmation and status updates. In addition, the platform incorporates a financial reporting module that intelligently aggregates fare data to provide real-time revenue insights, aiding in financial tracking. The integration of Microsoft SQL Server ensures continuous data persistence and security, making the system reliable for long-term record-keeping, while the modular Java design ensures that the application remains scalable and easy to maintain. This holistic approach leads to a dramatic enhancement in administrative efficiency, replacing cumbersome file-based systems with a streamlined digital interface. Overall, the Airline Reservation System transforms manual ticketing into a modern, organized, and highly automated experience that benefits the organization by fostering a more productive, error-free, and data-driven operational ecosystem.

CHAPTER 1
INTRODUCTION

1.1 Introduction to Airline Reservation System

The integration of robust database architectures into the aviation sector represents a critical operational evolution in modern travel management. The Airline Reservation System, which leverages Java Object-Oriented Programming and Microsoft SQL Server technologies, exemplifies this transformation by providing comprehensive solutions for ticket booking, inventory control, and passenger management challenges.

In an industry defined by high transaction volumes and strict scheduling, organizations face numerous hurdles in maintaining accurate seat inventories and processing reservations efficiently. Traditional manual ticketing methodologies are often labour-intensive, error-prone, and susceptible to data redundancy or loss. Meanwhile, administrators struggle to maintain real-time visibility into flight capacity and revenue streams. The Airline Reservation System addresses these pain points by creating a centralized, digital ecosystem that serves both the administration and the passengers through automated, data-driven transaction processing.

1.2 System Overview

The Airline Reservation System consists of three core components working in harmony:

1. **Automated Reservation & Fare Calculation Engine:** This component handles the core logic of the booking process, automatically validating seat availability against the database in real-time to prevent overbooking. By dynamically applying fare rules based on seat class selection (Economy, Premium, Business), it ensures financial accuracy and eliminates the calculation errors common in manual processing

2. **Passenger Data Management & Retrieval Module:**: The persistent storage layer serves as the system's memory, utilizing JDBC connectivity to securely store and manage passenger records. It allows administrators to instantly search for travelers by unique IDs or retrieve full flight manifests, facilitating seamless modifications, cancellations, and status checks without the risk of data loss associated with file-based systems.

3. **Real-Time Inventory & Revenue Analytics:** reporting component analyzes the financial and operational data within the system to provide immediate insights. By aggregating fare totals and monitoring fluctuating seat counts, it offers a snapshot of the flight’s financial performance and occupancy rates, enabling data-driven decisions .

1.3 Technological Foundation

The system's effectiveness stems from its robust technological underpinnings:

1. **JDBC & SQL Server Integration:** By leveraging the Java Database Connectivity (JDBC) API with Microsoft SQL Server, the system establishes a secure and reliable channel for data persistence. This ensures that complex transactions, such as booking confirmations and cancellations, are executed with Atomicity, Consistency, Isolation, and Durability (ACID) properties.

2. **Object-Oriented Programming (OOP) Logic:** The system employs sophisticated OOP principles—including Encapsulation, Inheritance, and Polymorphism—to model complex airline entities. This modular architecture allows for the precise handling of passenger data, seat objects, and flight schedules, ensuring code reusability and maintainability.

3. **Real-Time Transaction Management:** Unlike static file systems, this solution utilizes dynamic transaction management to handle concurrent data access. This ensures that seat inventory is updated instantly upon booking, preventing data conflicts and maintaining the integrity of the flight's status across all operations.

1.4 Impact on Airline Operations

The implementation of the Airline Reservation System fundamentally transforms traditional aviation management workflows:

1. **Efficiency Gains:** By automating the calculation of fares and the updating of seat inventories, the system dramatically reduces the time administrators spend on manual paperwork and calculation tasks, allowing them to focus on customer service.

2. **Inventory Integrity:** The system provides strict, algorithmic validation of seat availability that complements human oversight. This leads to a total elimination of overbooking errors and ensures that flight capacity is utilized optimally.

3. **Enhanced Passenger Experience:** Travelers benefit from a streamlined process that offers instant booking confirmation, quick retrieval of their travel details, and transparent updates regarding their reservation status.

4. **Financial Accuracy:** By automating fare computation based on specific seat classes rather than relying on manual entry, the system eliminates human calculation errors, ensuring precise revenue tracking and consistent financial reporting.

1.5 Broader Implications for Workforce Management

Beyond the immediate benefits of efficient ticket processing, the Airline Reservation System contributes significantly to strategic airline operations planning:

1. **Route Capacity Optimization:** The system helps organizations identify demand patterns for specific flights and routes, allowing for better allocation of resources and management of seat inventories before critical shortages or over-capacity scenarios occur.

2. **Revenue Yield Management:** By aggregating data regarding class preferences (Economy vs. Business) and booking frequencies, the system can identify opportunities for dynamic pricing strategies to maximize the revenue generated per flight.

3. **Operational Intelligence:** Patterns in passenger booking behaviors, cancellation rates, and peak travel times provide valuable insights into market dynamics, informing future flight scheduling and service adjustments.

The Airline Reservation System represents a significant step forward in aviation technology, aligning with broader trends toward digital transformation in the travel industry. As airlines increasingly recognize the strategic importance of data-driven inventory control and automated financial tracking, such database-integrated systems are becoming essential components of competitive aviation operations.

The following sections will explore each component of the system in greater detail, examining implementation considerations, best practices for database integration, and metrics for evaluating operational success.

CHAPTER 2
SCOPE AND MOTIVATION

2.1 Project Scope

The Airline Reservation System project encompasses the comprehensive development of an integrated intelligent software solution designed to streamline all aspects of the flight booking and inventory management lifecycle. The automated reservation engine will employ advanced transactional logic to capture passenger details, validate data integrity, and objectively allocate seat inventory against flight capacity with high precision. Using robust Object-Oriented Programming principles, the system's enquiry module will deliver instant access to passenger records, facilitate seamless ticket cancellations, provide real-time status updates, and offer comprehensive flight manifest data to administrators throughout the operational workflow. The dynamic fare calculation engine will leverage specific algorithmic rules to associate ticket prices with cabin classes (Economy, Premium, Business), ensuring financial accuracy based on both static configurations and real-time class selection, while also maintaining strict inventory controls to prevent overbooking. A comprehensive revenue reporting dashboard will offer administrators granular insights into financial accumulation, seat occupancy rates, and overall flight performance metrics. The system architecture will implement robust data persistence measures via Microsoft SQL Server, ensuring ACID (Atomicity, Consistency, Isolation, Durability) properties for safe transactions, and scalable JDBC connectivity to support operations ranging from single-terminal interfaces to potential multi-user environments. The project scope includes the development of intuitive, accessible console-based interfaces for administrative users, comprehensive database schema documentation, and exception handling mechanisms to ensure system stability. Throughout development, careful attention will be paid to data security, input validation, and the maintenance of referential integrity across the relational database structure.

2.2 Project Motivation

The Airline Reservation System emerges as a transformative, forward-looking solution designed to address the wide array of challenges associated with traditional manual ticketing and aviation inventory management processes. In an era where operational efficiency and data accuracy are critical drivers of an airline's success, the pressure on aviation managers to process bookings instantly and error-free has never been higher. However, existing manual or file-based models are often plagued by inefficiencies, lack of scalability, and a heavy reliance on human calculation. These limitations manifest as prolonged processing times, inconsistent fare application, and a reduced organizational agility in adapting to fluctuating passenger demand. One of the most pressing issues is the manual tracking of seat inventory, which not only creates bottlenecks in the reservation pipeline but also opens the door to critical errors such as overbooking and revenue leakage. Administrators frequently find themselves overwhelmed with repetitive, clerical tasks, leaving little room for customer service excellence or strategic route planning. On the other hand, travelers often encounter a slow and uncertain booking process, with potential delays in confirmation and retrieval of travel details. This lack of automation and transparency can lead to operational friction, negative passenger experiences, and a weakened brand reputation. Moreover, the misalignment between recorded bookings and actual seat availability continues to be a significant contributor to operational failures. Inaccurate records lead to financial discrepancies, decreased customer satisfaction, and elevated administrative costs—challenges that can severely impact long-term organizational viability. There is a growing need for a smarter, scalable, and more reliable system that not only accelerates the booking process but also enhances its accuracy and financial integrity. The Airline Reservation System tackles these challenges head-on by harnessing the robust capabilities of **Java Database Connectivity (JDBC)** in conjunction with **SQL Server** data management technologies. This system leverages algorithmic logic to automate and enhance every critical stage of the reservation lifecycle—right from passenger insertion and seat validation to cancellation handling and revenue generation. It transforms ticketing from a traditionally manual and resource-heavy process into a data-driven, intelligent ecosystem that empowers both administrators and the airline organization. For the management, the system offers actionable financial insights, error reduction, and automated inventory control, ensuring faster and more accurate flight management. It enables administrative staff to focus on high-value service functions while the system handles the complex data transaction groundwork. For the operational workflow, it introduces a new level of reliability and speed by providing instant fare calculations, immediate seat updates, and secure data retrieval—ultimately modernizing the infrastructure of travel management. By integrating Java and SQL in a meaningful way, this project not only addresses existing pain points in manual reservations but also sets a new benchmark for what inventory management can look like in the digital age. The Airline Reservation System is more than a technical upgrade—it is a strategic leap toward efficiency, data security, and operational intelligence in aviation management,

CHAPTER 3

DESCRIPTION

3. System Architecture and Core Components

3.1 Technical Foundation

The Airline Reservation System is built on a robust technological framework that combines the structural integrity of Java with the data persistence power of Microsoft SQL Server to create a comprehensive aviation management solution. At its core, the system utilizes the Java Database Connectivity (JDBC) API, which provides the essential communication bridge for processing complex queries between the application layer and the relational database. This foundation is complemented by custom Object-Oriented algorithms and exception-handling mechanisms specifically designed for airline inventory logic. The system architecture follows a Modular Object-Oriented approach, with each major entity—Passenger, Flight, and Reservation—operating as an encapsulated unit with defined behaviors. This design ensures code reusability and allows for independent updates to business logic, such as fare rules or seat configurations, without disrupting the entire system. Data flows between the application and the database through secure Prepared Statement interfaces, ensuring that all information is sanitized and protected against SQL injection attacks to maintain data security compliance.

3.2 Automated Inventory System

The **Automated Reservation Engine** component represents a significant advancement over manual ticketing and file-based booking methods. Unlike static ledger systems, this component employs dynamic transactional logic to interpret and validate the availability of resources in real-time.

- **Structured Data Processing & Validation:** The system utilizes strong typing to extract and structure information from user inputs (Name, Age, Gender, Seat Preference) while maintaining data integrity. It strictly validates inputs against enumerated types (`Gender` and `SeatClass`), ensuring that only standardized, error-free data is committed to the `Passengers` database table, regardless of the sequence of entry.
- **Real-Time Inventory Validation:** Using a specialized availability algorithm, the system performs an instant cross-reference of the candidate's seat preference against the live inventory in the `Flights` table. This approach goes beyond simple record insertion; it acts as a logic gate that checks specific column values (Economy, Business) and prevents the transaction if the seat count is zero, effectively eliminating the risk of overbooking.

- **Dynamic Fare Computation:** Reservations are processed through a logic-driven pricing model that automatically assigns fares based on the selected `SeatClass`. Instead of relying on manual entry, the system utilizes the `calculateFare` method to map distinct price points (e.g., Economy vs. Business) to the passenger object, ensuring financial consistency and eliminating human calculation errors during the transaction.

3.3 User Experience and Interfaces

The system provides airline staff with a streamlined console or dashboard experience designed to optimize workflow efficiency:

- **Unified Flight Control:** Administrators can manage bookings, cancellations, and flight status from a single menu-driven interface.
- **Real-Time Analytics:** The revenue module provides an instant snapshot of the flight's financial performance without requiring complex external reporting tools.
- **Inventory Visibility:** The showAvailableSeats function gives staff an immediate view of remaining capacity in Economy, Premium, and Business classes to aid in upsell decisions.

3.4 Passenger/User Experience

While currently backend-focused, the logic supports a user-facing frontend designed to be supportive and transparent:

- **Instant Confirmation:** The transactional logic ensures that passengers receive immediate feedback on their booking status (Success vs. No Seats Available).
- **Clear Fare Transparency:** The system explicitly displays the calculated fare based on class selection before the booking is finalized.
- **Error Handling:** User-friendly error messages (e.g., "Invalid Input," "Passenger Not Found") guide the user to correct mistakes without application crash

CHAPTER 4
IMPLEMENTATION

```
import java.sql.*;
import java.util.*;

class DBConnection {
  public static Connection
getConnection() throws Exception {
      String url =
        "jdbc:sqlserver://localhost:1433;"
       +
"databaseName=AirlineReservationDB;"
       + "integratedSecurity=true;"
       + "encrypt=true;"
       + "trustServerCertificate=true;";
      return
DriverManager.getConnection(url);
  }
}

enum Gender {
  MALE, FEMALE, OTHERS
}

enum SeatClass {
  ECONOMY,
PREMIUM_ECONOMY, BUSINESS,
STANDBY, WAITING
}

class Passenger {
  private String name;
  private int age;
  private Gender gender;
  private SeatClass seat;
  private double fare;

  public Passenger(String name, int age,
Gender gender) {
      this.name = name;
      this.age = age;
      this.gender = gender;
  }

  public String getName() { return name;
}
  public int getAge() { return age; }
  public Gender getGender() { return
gender; }
  public SeatClass getSeat() { return seat;
}
  public void setSeat(SeatClass seat) {
this.seat = seat; }
  public double getFare() { return fare; }
  public void setFare(double fare) {
this.fare = fare; }
}

abstract class ReservationSystem {

  protected String flightCode;
  public ReservationSystem(String
flightCode) {
      this.flightCode = flightCode;
  }

  public abstract void
bookTicket(Passenger p, SeatClass
preference);
  public abstract void cancelTicket(int
passengerId);

  public void showPassengers() {
      String sql = "SELECT * FROM
Passengers";
      try (Connection conn =
DBConnection.getConnection();
             Statement st =
conn.createStatement();
             ResultSet rs =
st.executeQuery(sql)) {

            if (!rs.next()) {
                System.out.println("No
Passenger Booked");
                return;
            }

            do {
                System.out.println("---------------
---------------");
                System.out.println("Passenger
ID : " + rs.getInt("PassengerID"));
```

```
            System.out.println("Name        :
" + rs.getString("Name"));
            System.out.println("Age         : "
+ rs.getInt("Age"));
            System.out.println("Gender      :
" + rs.getString("Gender"));
            System.out.println("Seat Class  :
" + rs.getString("SeatClass"));
            System.out.println("Fare        : ₹"
+ rs.getDouble("Fare"));
            System.out.println("---------------
---------------");
        } while (rs.next());

      } catch (Exception e) {
        e.printStackTrace();
      }
   }

   public void searchPassenger(int
passengerId) {
      String sql = "SELECT * FROM
Passengers WHERE PassengerID=?";
      try (Connection conn =
DBConnection.getConnection();
          PreparedStatement ps =
conn.prepareStatement(sql)) {

        ps.setInt(1, passengerId);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
                System.out.println("---------------
---------------");
                System.out.println("Passenger
ID : " + rs.getInt("PassengerID"));
                System.out.println("Name        :
" + rs.getString("Name"));
                System.out.println("Age         : "
+ rs.getInt("Age"));
                System.out.println("Gender      :
" + rs.getString("Gender"));
                System.out.println("Seat Class  :
" + rs.getString("SeatClass"));
                System.out.println("Fare        :
Rs. " + rs.getDouble("Fare"));
                System.out.println("---------------
---------------");
            } else {
                System.out.println("Passenger
not found");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void showTotalRevenue() {
        String sql = "SELECT SUM(Fare)
AS Total FROM Passengers";
        try (Connection conn =
DBConnection.getConnection();
             Statement st =
conn.createStatement();
          ResultSet rs =
st.executeQuery(sql)) {

          if (rs.next()) {
             System.out.println("Total
Revenue: Rs. " + rs.getDouble("Total"));
          }

       } catch (Exception e) {
          e.printStackTrace();
       }
    }
}

class FlightReservation extends
ReservationSystem {

    public FlightReservation(String
flightCode) {
       super(flightCode);
    }

    private int getSeat(String column) {
       String sql = "SELECT " + column +
" FROM Flights WHERE FlightCode=?";
       try (Connection conn =
DBConnection.getConnection();
          PreparedStatement ps =
conn.prepareStatement(sql)) {

          ps.setString(1, flightCode);
```

```
            ResultSet rs = ps.executeQuery();
            if (rs.next()) return
rs.getInt(column);

        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }

    private void updateSeat(String column,
int value) {
        String sql = "UPDATE Flights SET "
+ column + "=? WHERE FlightCode=?";
        try (Connection conn =
DBConnection.getConnection();
             PreparedStatement ps =
conn.prepareStatement(sql)) {

            ps.setInt(1, value);
            ps.setString(2, flightCode);
            ps.executeUpdate();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void insertPassenger(Passenger
p) {
        String sql =
            "INSERT INTO Passengers
```

```
(Name, Age, Gender, SeatClass, Fare) " +
          "VALUES (?, ?, ?, ?, ?)";
      try (Connection conn =
DBConnection.getConnection();
           PreparedStatement ps =
conn.prepareStatement(sql)) {

          ps.setString(1, p.getName());
          ps.setInt(2, p.getAge());
          ps.setString(3,
p.getGender().toString());
          ps.setString(4,
p.getSeat().toString());
          ps.setDouble(5, p.getFare());
          ps.executeUpdate();

      } catch (Exception e) {
          e.printStackTrace();
      }
   }

   private double calculateFare(SeatClass
seat) {
      switch (seat) {
          case ECONOMY: return 3500;
          case PREMIUM_ECONOMY:
return 5500;
          case BUSINESS: return 9000;
          default: return 0;
      }
   }
```

```
  @Override
  public void bookTicket(Passenger p,
SeatClass pref) {

    SeatClass allotted;

    if (pref == SeatClass.ECONOMY
&& getSeat("Economy") > 0) {
      allotted = SeatClass.ECONOMY;
      updateSeat("Economy",
getSeat("Economy") - 1);

    } else if (pref ==
SeatClass.PREMIUM_ECONOMY &&
getSeat("PremiumEconomy") > 0) {
      allotted =
SeatClass.PREMIUM_ECONOMY;
      updateSeat("PremiumEconomy",
getSeat("PremiumEconomy") - 1);

    } else if (pref ==
SeatClass.BUSINESS &&
getSeat("Business") > 0) {
      allotted = SeatClass.BUSINESS;
      updateSeat("Business",
getSeat("Business") - 1);

    } else {
      System.out.println("No seats
available.");
      return;
    }
```

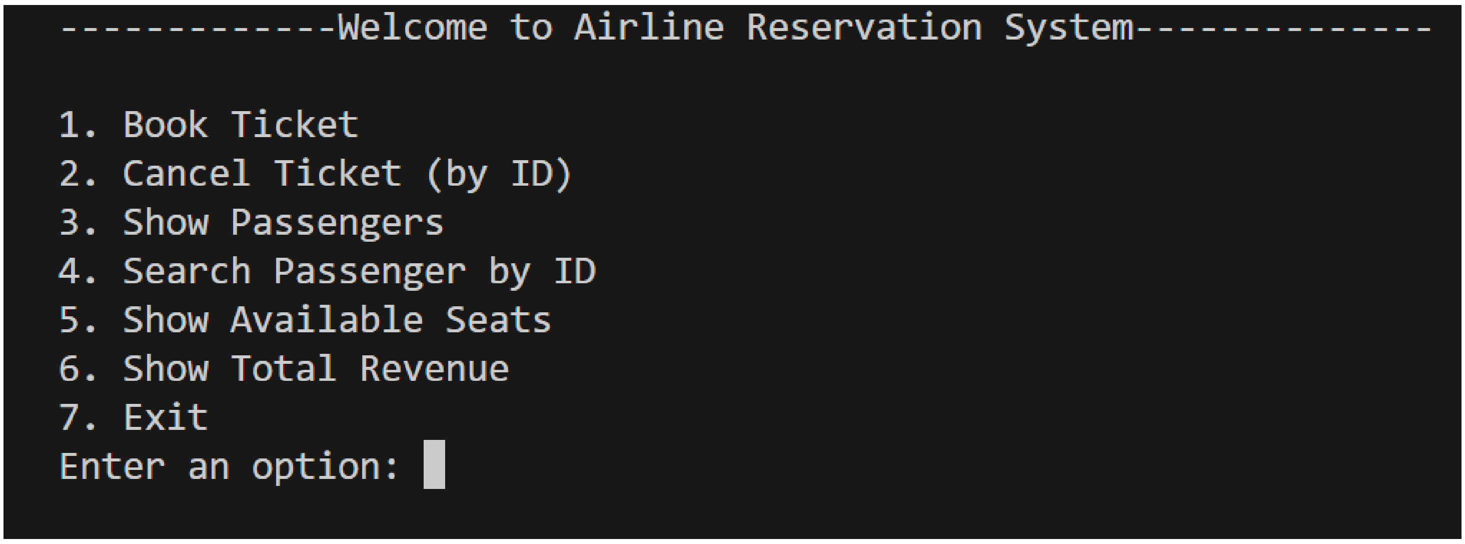
```
        p.setSeat(allotted);
        double fare = calculateFare(allotted);
        p.setFare(fare);

        insertPassenger(p);

        System.out.println("Ticket booked on
flight " + flightCode);
    }

    @Override
    public void cancelTicket(int
passengerId) {
        String sql = "DELETE FROM
Passengers WHERE PassengerID=?";
        try (Connection conn =
DBConnection.getConnection();
             PreparedStatement ps =
conn.prepareStatement(sql)) {

            ps.setInt(1, passengerId);

            if (ps.executeUpdate() > 0)
                System.out.println("Ticket
cancelled successfully.");
            else
                System.out.println("Passenger
not found.");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void showAvailableSeats() {
        String sql = "SELECT * FROM
Flights WHERE FlightCode=?";
        try (Connection conn =
DBConnection.getConnection();
             PreparedStatement ps =
conn.prepareStatement(sql)) {

            ps.setString(1, flightCode);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                System.out.println("\nFlight: " +
flightCode);
                System.out.println("Economy
    : " + rs.getInt("Economy"));
                System.out.println("Premium
Economy : " +
rs.getInt("PremiumEconomy"));
                System.out.println("Business
 : " + rs.getInt("Business"));
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
public class AirlineReservationSystem {

   public static void main(String[] args) {

       System.out.println("------------
Welcome to Airline Reservation System--
-----------");
       System.out.print("How can I help
You?");

       Scanner sc = new
Scanner(System.in);
       ReservationSystem flight = new
FlightReservation("6E-204");

       while (true) {

          System.out.println("\n1. Book
Ticket");
          System.out.println("2. Cancel
Ticket (by ID)");
          System.out.println("3. Show
Passengers");
          System.out.println("4. Search
Passenger by ID");
          System.out.println("5. Show
Available Seats");
          System.out.println("6. Show Total
Revenue");
          System.out.println("7. Exit");

          System.out.print("Enter an option:
```

```
");
        int choice = sc.nextInt();
        sc.nextLine();

        switch (choice) {

          case 1:
            System.out.print("Name: ");
            String name = sc.nextLine();

            System.out.print("Age: ");
            int age = sc.nextInt();
            sc.nextLine();

            System.out.print("Gender
(MALE/FEMALE/OTHERS): ");
            Gender g =
Gender.valueOf(sc.nextLine().toUpperCa
se());

            System.out.print("Seat
Preference: ");
            SeatClass s =
SeatClass.valueOf(sc.nextLine().toUpper
Case());

            Passenger p = new
Passenger(name, age, g);
            flight.bookTicket(p, s);
            break;

          case 2:
            System.out.print("Passenger
ID: ");
            flight.cancelTicket(sc.nextInt()
);
            sc.nextLine();
            break;

          case 3:
            flight.showPassengers();
            break;

          case 4:
            System.out.print("Passenger
ID: ");
            flight.searchPassenger(sc.nextI
nt());
            sc.nextLine();
            break;

          case 5:
            ((FlightReservation)
flight).showAvailableSeats();
            break;

          case 6:
            flight.showTotalRevenue();
            break;

          case 7:
            System.out.println("---------
Thank You for Choosing Airline
Reservation System---------");
```

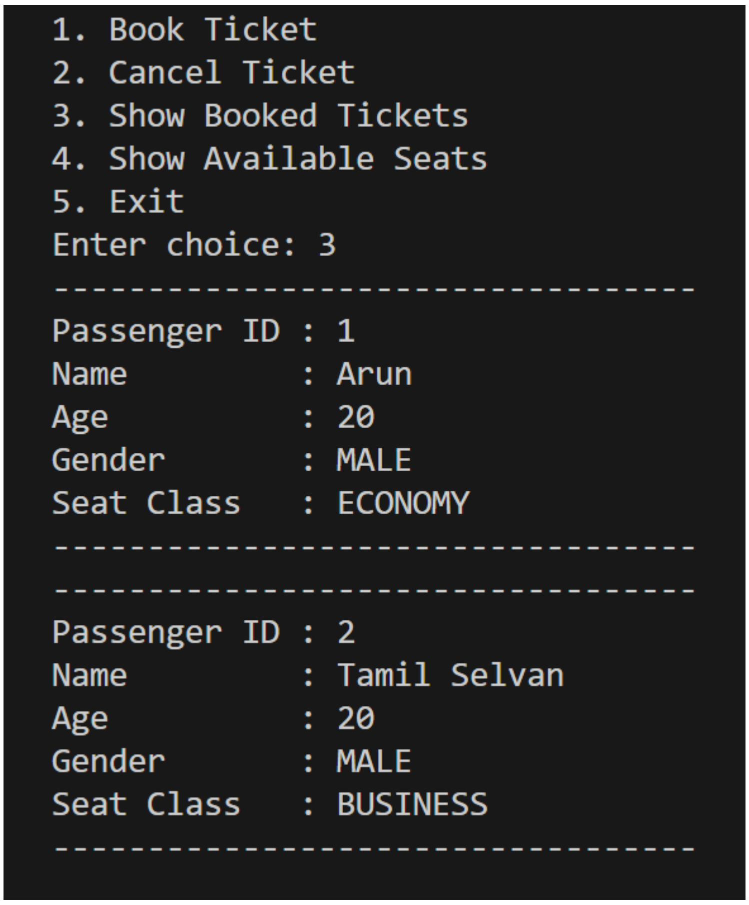
```
                sc.close();
                return;
            }
        }
    }
}
```

CHAPTER 5
OUTPUT



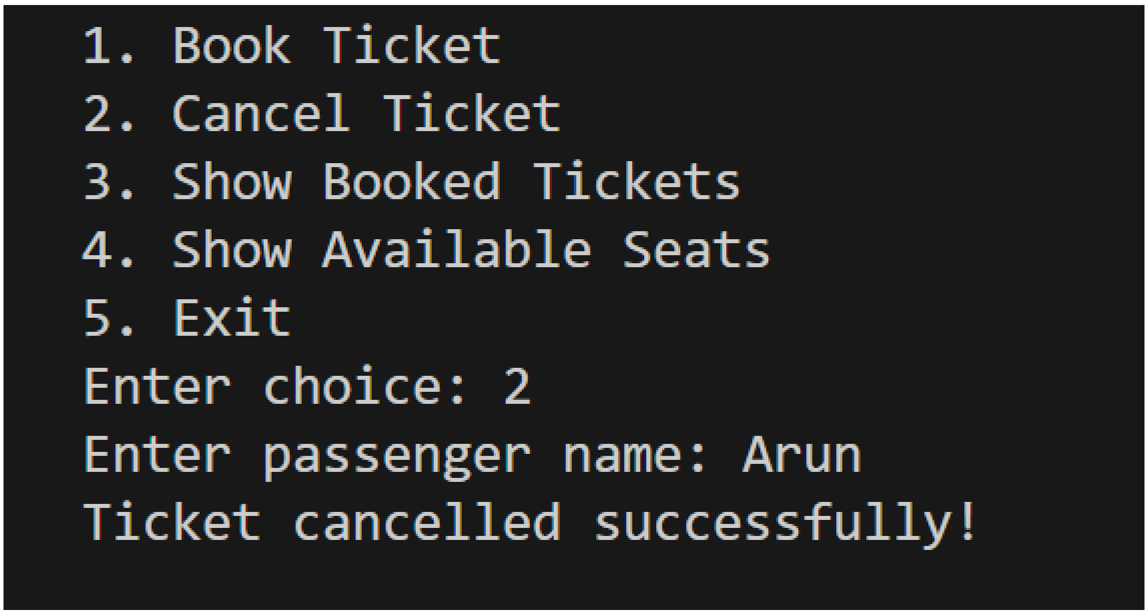
```
-------------Welcome to Airline Reservation System--------------

1. Book Ticket
2. Cancel Ticket (by ID)
3. Show Passengers
4. Search Passenger by ID
5. Show Available Seats
6. Show Total Revenue
7. Exit
Enter an option: 
```

Fig 6.1: Home Page



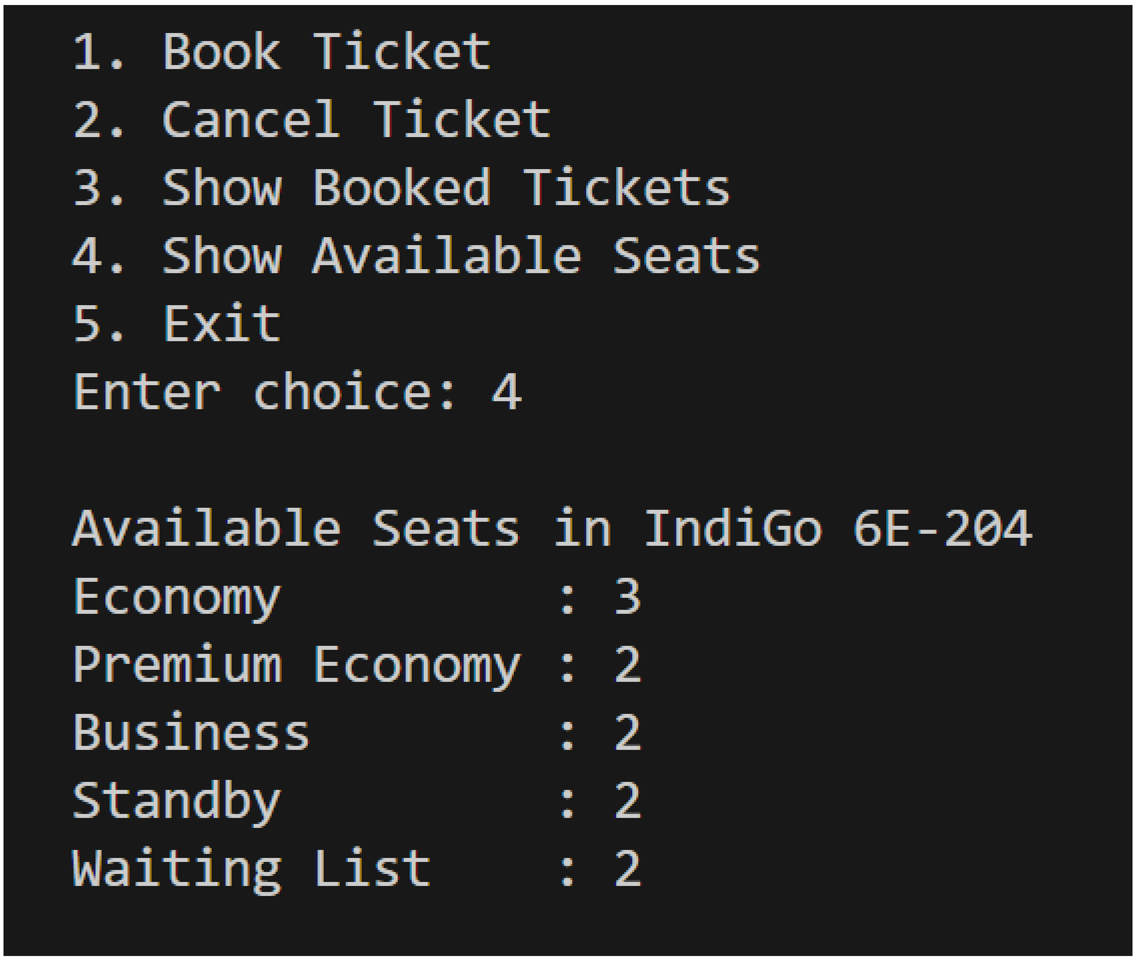
```
1. Book Ticket
2. Cancel Ticket
3. Show Booked Tickets
4. Show Available Seats
5. Exit
Enter choice: 3
----------------------------------
Passenger ID : 1
Name         : Arun
Age          : 20
Gender       : MALE
Seat Class   : ECONOMY
----------------------------------
----------------------------------
Passenger ID : 2
Name         : Tamil Selvan
Age          : 20
Gender       : MALE
Seat Class   : BUSINESS
----------------------------------
```

Fig 6.2: Booked tickets Page



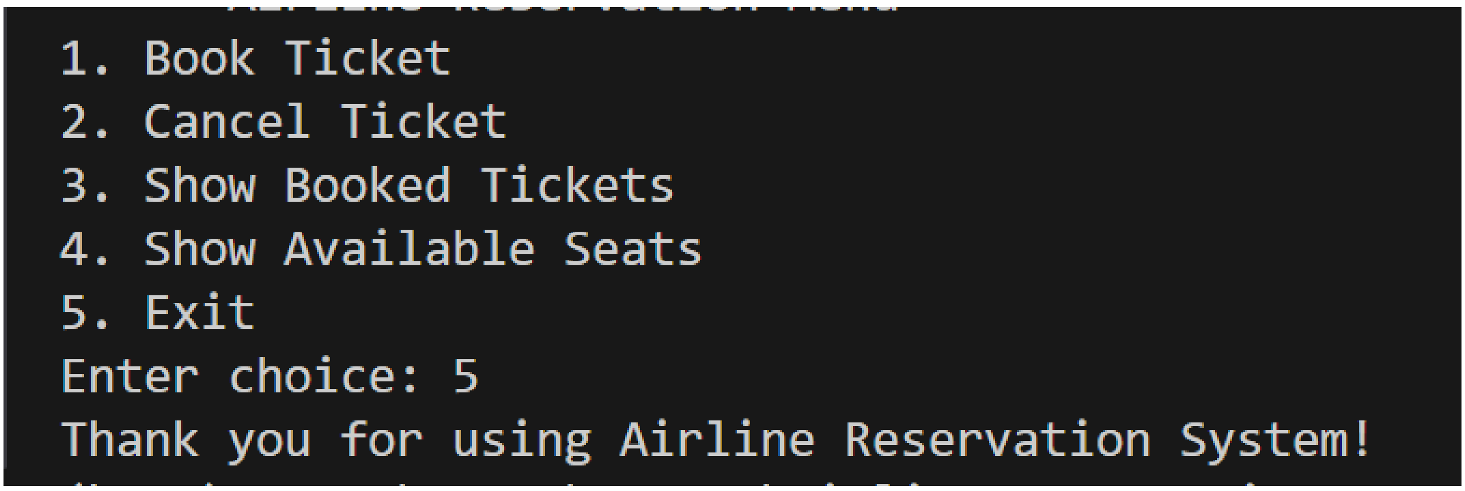
```
1. Book Ticket
2. Cancel Ticket
3. Show Booked Tickets
4. Show Available Seats
5. Exit
Enter choice: 2
Enter passenger name: Arun
Ticket cancelled successfully!
```

Fig 6.3: Cancel Ticket page



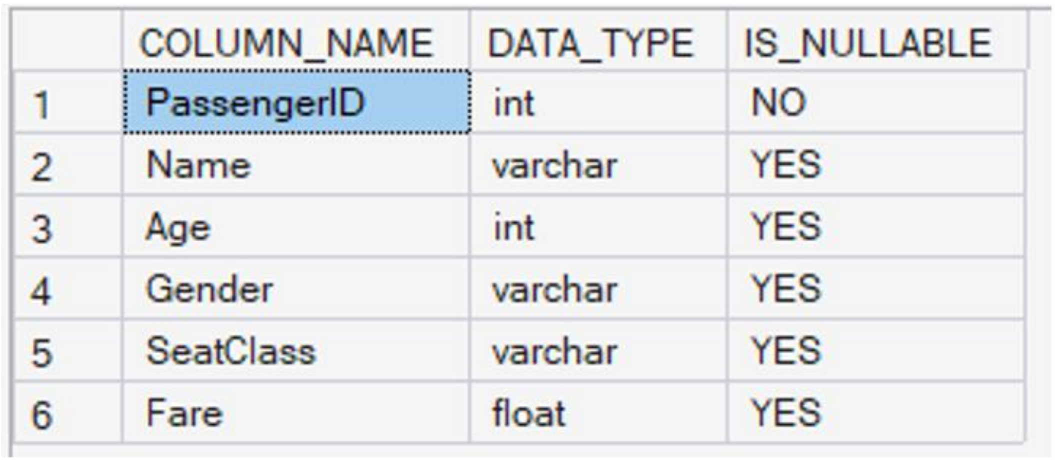
```
1. Book Ticket
2. Cancel Ticket
3. Show Booked Tickets
4. Show Available Seats
5. Exit
Enter choice: 4

Available Seats in IndiGo 6E-204
Economy         : 3
Premium Economy : 2
Business        : 2
Standby         : 2
Waiting List    : 2
```

Fig 6.4: Available tickets

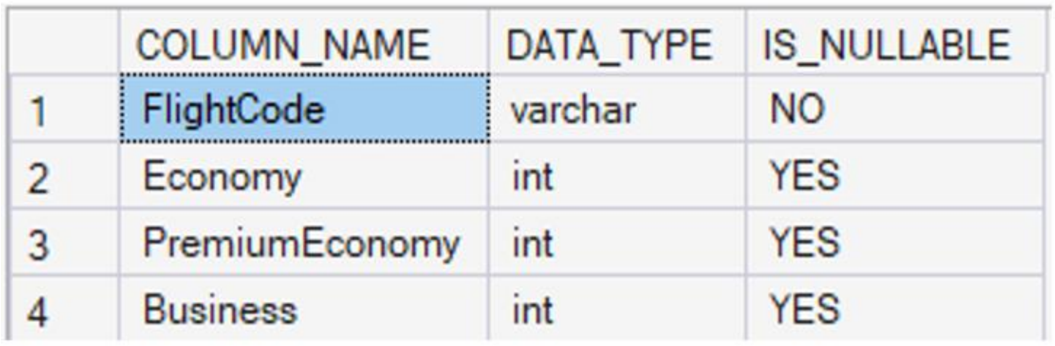


```
1. Book Ticket
2. Cancel Ticket
3. Show Booked Tickets
4. Show Available Seats
5. Exit
Enter choice: 5
Thank you for using Airline Reservation System!
```

Fig 6.5: Exit page



	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	PassengerID	int	NO
2	Name	varchar	YES
3	Age	int	YES
4	Gender	varchar	YES
5	SeatClass	varchar	YES
6	Fare	float	YES



	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	FlightCode	varchar	NO
2	Economy	int	YES
3	PremiumEconomy	int	YES
4	Business	int	YES

Fig 6.6: Database parameters

CHAPTER 6
CONCLUSION

The development of the Airline Reservation System represents a successful integration of robust software engineering principles with practical database management solutions. By bridging the logic-driven capabilities of Java with the persistent storage power of Microsoft SQL Server, the project has effectively addressed the critical inefficiencies inherent in manual aviation management. The system successfully automates complex transaction cycles—from real-time inventory validation to dynamic fare computation—thereby eliminating human error and ensuring absolute data integrity.

Through its modular Object-Oriented architecture, the system not only streamlines the booking interface for administrators but also provides a scalable foundation for future expansion. The implementation of strict ACID properties within the database connectivity layer guarantees that the airline's operational data remains consistent, secure, and retrievable at all times. Ultimately, this project demonstrates that moving from legacy file-based methods to a centralized, automated database ecosystem significantly enhances operational throughput, financial accuracy, and overall service reliability.

8.1 Future Scope

The Airline Reservation System is designed as an evolving platform. While the current iteration establishes a solid core for inventory and passenger management, the following enhancements are planned to extend its capabilities and market viability:

- **Graphical User Interface (GUI) Migration:** Transitioning from the current console-based command line interface to a modern, intuitive GUI (using JavaFX or a Web-based React framework). This will enhance usability for non-technical staff and enable visual features such as interactive seat maps where users can click to select specific seats.
- **Secure Payment Gateway Integration:** Implementing live API connections with financial processors (e.g., Stripe, PayPal, or Banking APIs). This will allow the system to handle real-time credit card transactions, verify payments before confirming bookings, and generate automated digital invoices.
- **Dynamic Pricing Algorithms:** Upgrading the calculateFare logic to include "Yield Management" AI. This would automatically adjust ticket prices based on variables such as remaining seat inventory, time until departure, and historical demand trends, maximizing revenue per flight.
- **Global Distribution System (GDS) Connectivity:** Expanding the backend to interface with global travel networks (like Amadeus or Sabre). This would allow third-party travel agents and booking aggregators (like Expedia) to view and book the airline's inventory in real-time.