

# WORLDLINE ASSIGNMENT JDBC TASK

Name : Daljeet Singh

Student Id:WDGET2024013

## Choice 1:Insert Passanger Details

```
Activities  Eclipse IDE for Enterprise Java and Web Developers - Latest Release  Mar 11 11:17  93 %

eclipse-workspace - AirlineTicketBookingSystem/src/main/java/AirlineTicketBookingSystem.java - Eclipse IDE

File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

Servers  Console x
AirlineTicketBookingSystem [Java Application] [pid: 15612]
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and may not be available in the future.
Connected to the database.

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 1
Enter passenger name:
DaljeetSingh
Enter passenger age:
21
Enter passenger phone number:
9265507477
Passenger details inserted successfully.

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

## Choice 2: Insert Flight Details

```
Activities  Eclipse IDE for Enterprise Java and Web Developers - Latest Release  Mar 11 11:18  93 %

eclipse-workspace - AirlineTicketBookingSystem/src/main/java/AirlineTicketBookingSystem.java - Eclipse IDE

File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

Servers  Console x
AirlineTicketBookingSystem [Java Application] [pid: 15612]
21
Enter passenger phone number:
9265507477
Passenger details inserted successfully.

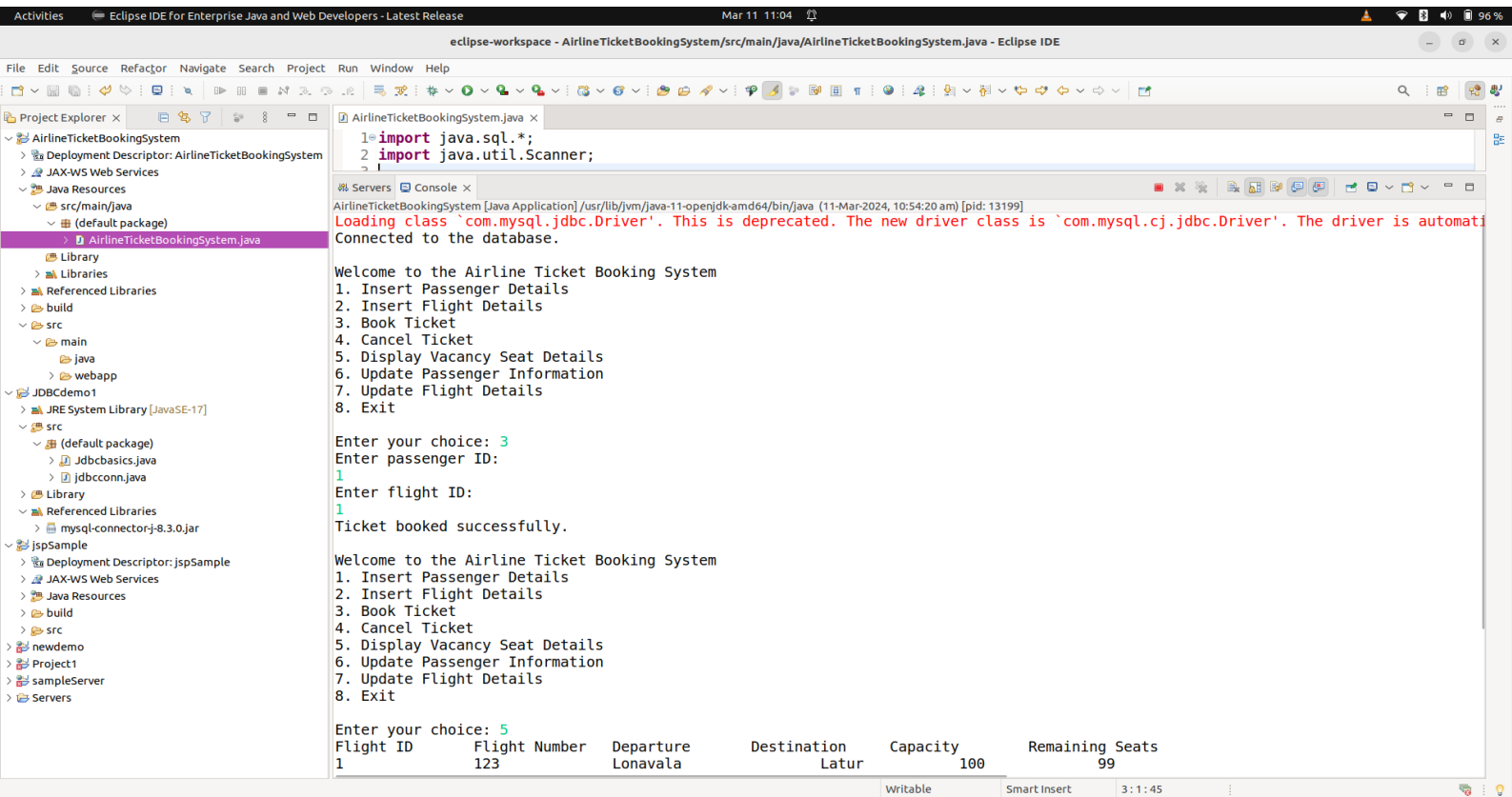
Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 2
Enter flight number:
111
Enter departure location:
Lonavala
Enter destination:
Latur
Enter capacity:
100
Flight details inserted successfully.

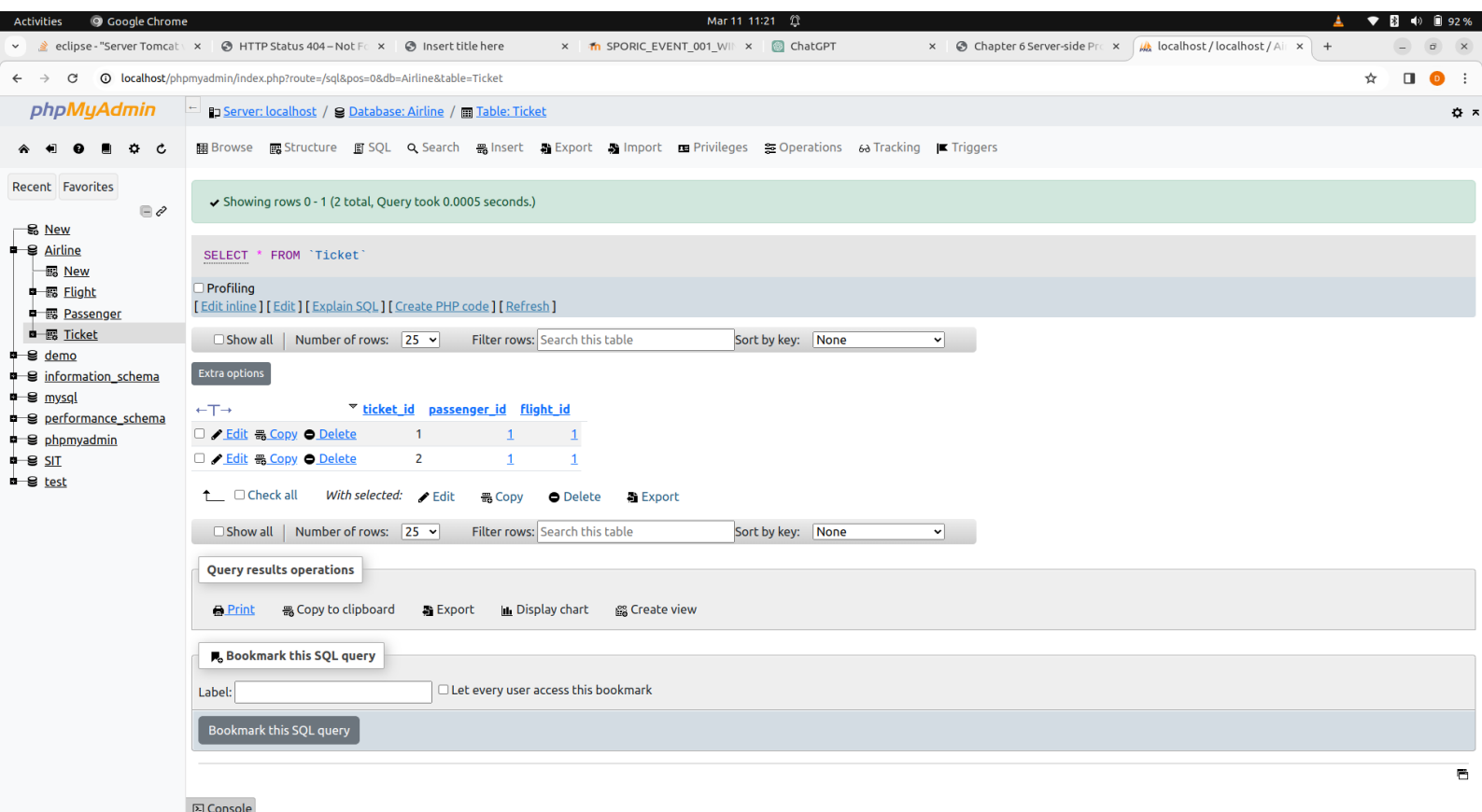
Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

## Choice 3: Book Ticket



## \* Ticket Database after updation



## \* Passenger Database Updated

The screenshot shows the phpMyAdmin interface for the 'Airline' database, specifically the 'Passenger' table. The table contains 2 rows of data. The SQL query executed is 'SELECT \* FROM `Passenger`'. The interface includes a sidebar with a tree view of the database structure, a top navigation bar with various tools, and a main content area displaying the table data and query results.

Server: localhost / Database: Airline / Table: Passenger

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

SELECT \* FROM `Passenger`

Number of rows: 25 Filter rows: Search this table Sort by key: None

passenger_id	name	age	phone_number
1	Daljeet	21	9265507477
2	DaljeetSingh	21	9265507477

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

Bookmark this SQL query

Label: Let every user access this bookmark

Bookmark this SQL query

## \* Flight Database Updated

The screenshot shows the phpMyAdmin interface for the 'Airline' database, specifically the 'Flight' table. The table contains 2 rows of data. The SQL query executed is 'SELECT \* FROM `Flight`'. The interface includes a sidebar with a tree view of the database structure, a top navigation bar with various tools, and a main content area displaying the table data and query results.

Server: localhost / Database: Airline / Table: Flight

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

SELECT \* FROM `Flight`

Number of rows: 25 Filter rows: Search this table Sort by key: None

flight_id	flight_number	departure	destination	capacity	remaining_seats
1	123	Lonavala	Latur	100	99
2	111	Lonavala	Latur	100	100

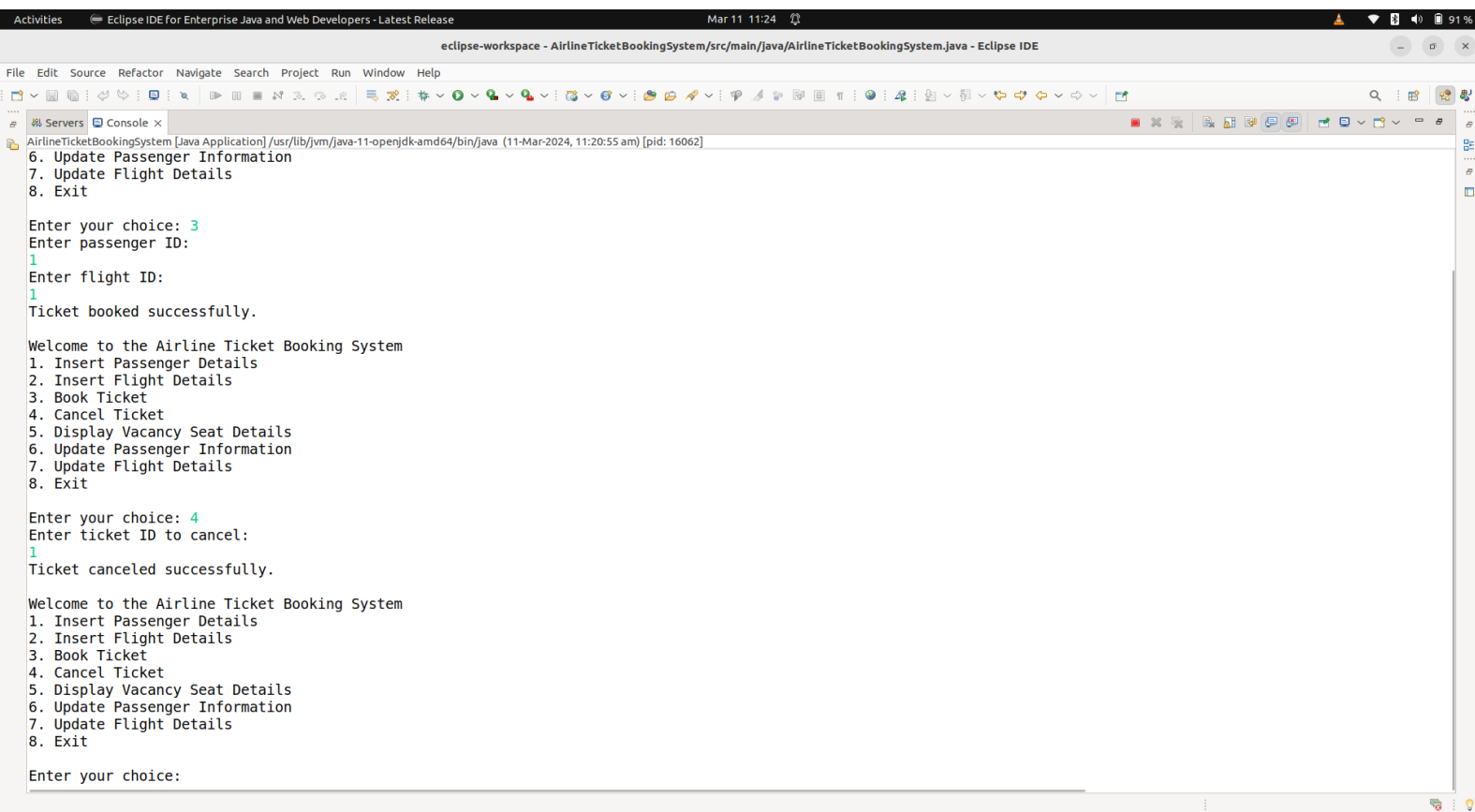
Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

Bookmark this SQL query

Label: Let every user access this bookmark

Bookmark this SQL query

## Choice 4: Cancel Ticket



The screenshot shows the Eclipse IDE interface with the console window open. The console displays the following text:

```
Activities  Eclipse IDE for Enterprise Java and Web Developers - Latest Release  Mar 11 11:24  91 %

eclipse-workspace - AirlineTicketBookingSystem/src/main/java/AirlineTicketBookingSystem.java - Eclipse IDE

File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

Servers  Console x
AirlineTicketBookingSystem [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (11-Mar-2024, 11:20:55 am) [pid: 16062]
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 3
Enter passenger ID:
1
Enter flight ID:
1
Ticket booked successfully.

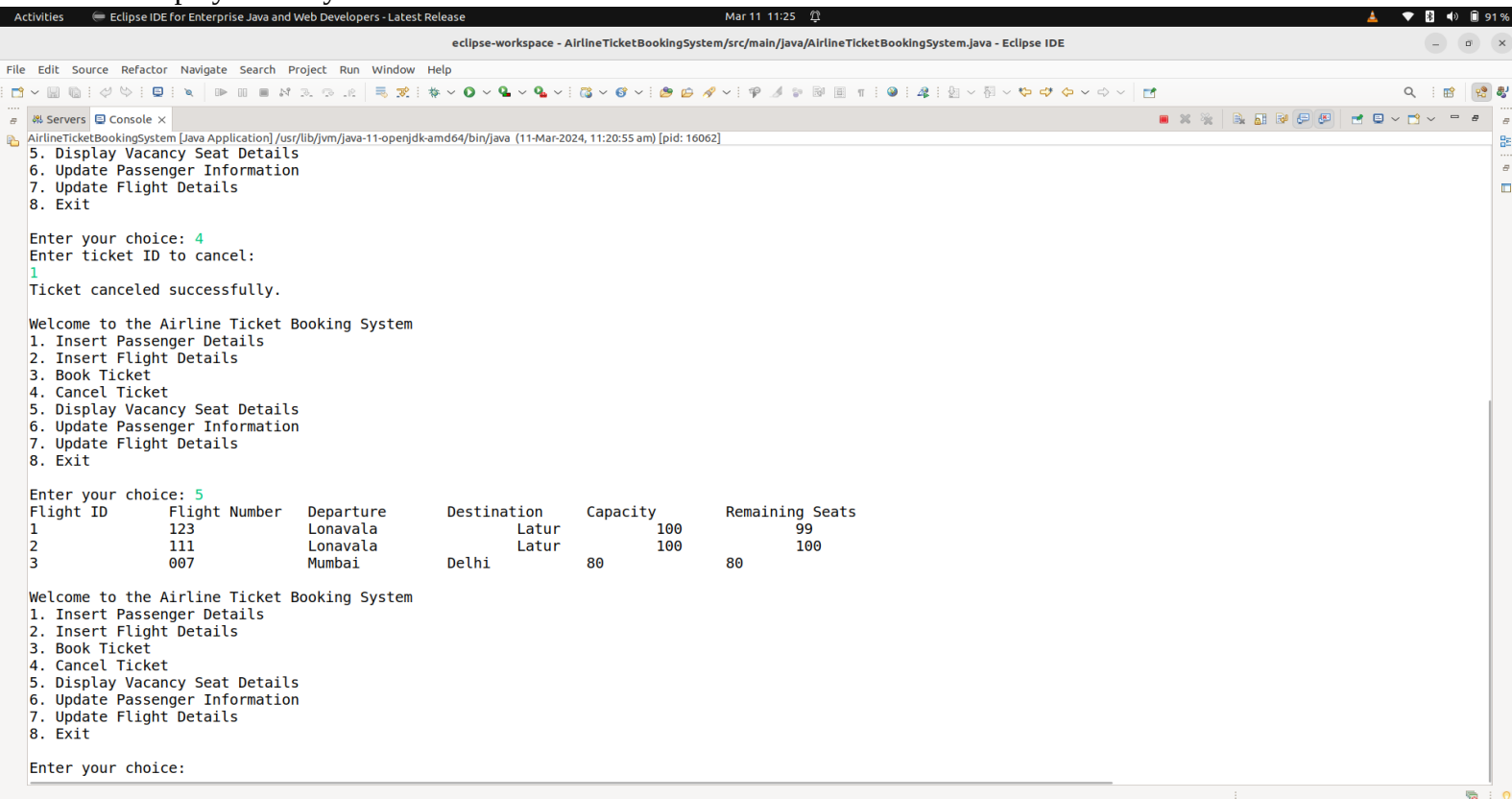
Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 4
Enter ticket ID to cancel:
1
Ticket canceled successfully.

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

## Choice 5: Display Vacancy Seat Details



The screenshot shows the Eclipse IDE interface with the console window open. The console displays the following text:

```
Activities  Eclipse IDE for Enterprise Java and Web Developers - Latest Release  Mar 11 11:25  91 %

eclipse-workspace - AirlineTicketBookingSystem/src/main/java/AirlineTicketBookingSystem.java - Eclipse IDE

File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

Servers  Console x
AirlineTicketBookingSystem [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (11-Mar-2024, 11:20:55 am) [pid: 16062]
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 4
Enter ticket ID to cancel:
1
Ticket canceled successfully.

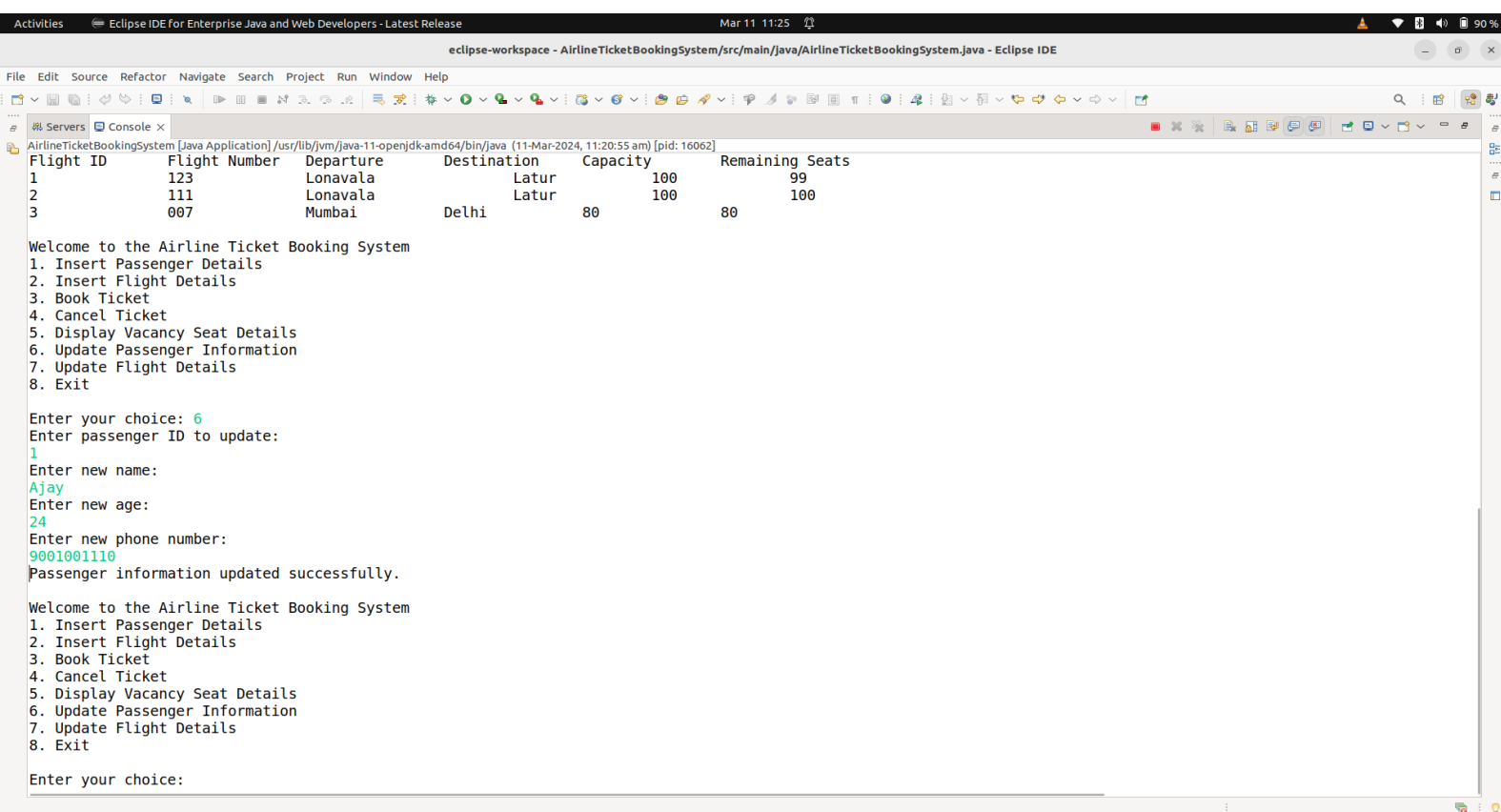
Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 5
Flight ID      Flight Number  Departure  Destination  Capacity  Remaining Seats
1             123           Lonavala  Latur        100       99
2             111           Lonavala  Latur        100       100
3             007           Mumbai    Delhi         80        80

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

## Choice 6: Update Passanger Details



The screenshot shows the Eclipse IDE interface with the console output of a Java application. The application is titled "AirlineTicketBookingSystem" and is running on a Java 11 application. The console output displays a menu of options, and the user has chosen option 6, "Update Passenger Information". The user has entered the passenger ID 1, the new name "Ajay", the new age 24, and the new phone number 9001001110. The output confirms that the passenger information was updated successfully. The console also shows the initial state of the flight details table and the menu options.

```
Flight ID      Flight Number  Departure      Destination      Capacity      Remaining Seats
1              123           Lonavala       Latur            100           99
2              111           Lonavala       Latur            100           100
3              007           Mumbai         Delhi             80            80

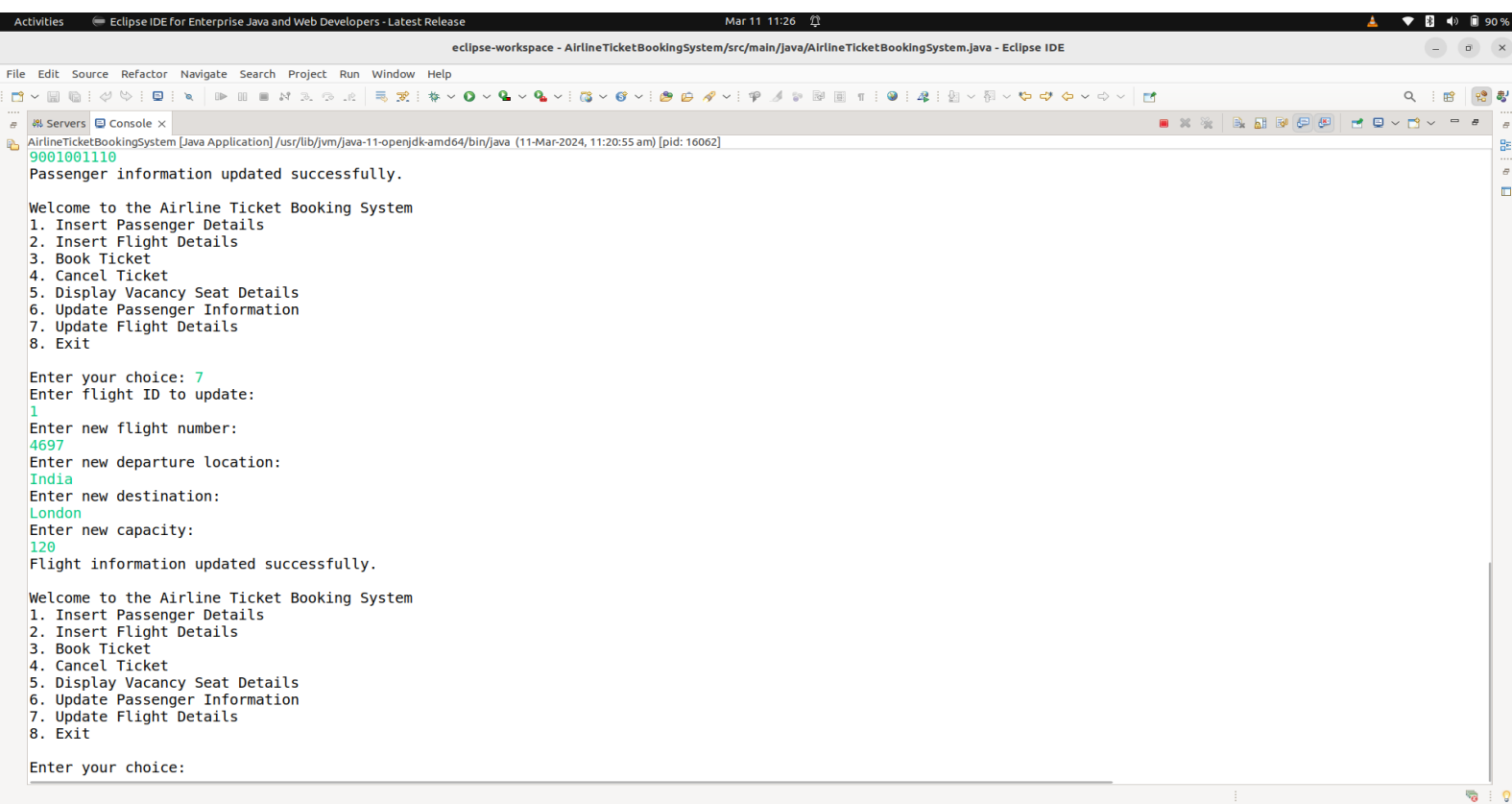
Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 6
Enter passenger ID to update:
1
Enter new name:
Ajay
Enter new age:
24
Enter new phone number:
9001001110
Passenger information updated successfully.

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

## Choice 7: Update Flight Details



The screenshot shows the Eclipse IDE interface with the console output of a Java application. The application is titled "AirlineTicketBookingSystem" and is running on a Java 11 application. The console output displays a menu of options, and the user has chosen option 7, "Update Flight Details". The user has entered the flight ID 1, the new flight number 4697, the new departure location "India", the new destination "London", and the new capacity 120. The output confirms that the flight information was updated successfully. The console also shows the initial state of the flight details table and the menu options.

```
Flight ID      Flight Number  Departure      Destination      Capacity      Remaining Seats
1              123           Lonavala       Latur            100           99
2              111           Lonavala       Latur            100           100
3              007           Mumbai         Delhi             80            80

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice: 7
Enter flight ID to update:
1
Enter new flight number:
4697
Enter new departure location:
India
Enter new destination:
London
Enter new capacity:
120
Flight information updated successfully.

Welcome to the Airline Ticket Booking System
1. Insert Passenger Details
2. Insert Flight Details
3. Book Ticket
4. Cancel Ticket
5. Display Vacancy Seat Details
6. Update Passenger Information
7. Update Flight Details
8. Exit

Enter your choice:
```

-----CodeSegment-----

Create Database:

```
CREATE TABLE Ticket (  
    ticket_id INT PRIMARY KEY AUTO_INCREMENT,  
    passenger_id INT,  
    flight_id INT,  
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id),  
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)  
);
```

```
CREATE TABLE Passenger (  
    passenger_id INT PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(100),  
    age INT,  
    phone_number VARCHAR(20)  
);
```

```
CREATE TABLE Flight (  
    flight_id INT PRIMARY KEY AUTO_INCREMENT,  
    flight_number VARCHAR(10),  
    departure VARCHAR(100),  
    destination VARCHAR(100),  
    capacity INT,  
    remaining_seats INT  
);
```

AirlineTicketBookingSystem.java

```
import java.sql.*;  
import java.util.Scanner;  
  
public class AirlineTicketBookingSystem {  
  
    private static Connection connection;  
    private static Scanner scanner = new Scanner(System.in);  
  
    public static void main(String[] args) {  
        try {  
            Class.forName("com.mysql.jdbc.Driver");  
            connection =  
DriverManager.getConnection("jdbc:mysql://localhost:4306/Airline",  
"root", "");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

    if (connection != null) {
        System.out.println("Connected to the database.");
        displayMenu();
    } else {
        System.out.println("Failed to connect to the database.");
    }
}

private static void displayMenu() {
    boolean exit = false;

    while (!exit) {
        System.out.println("\nWelcome to the Airline Ticket
Booking System");
        System.out.println("1. Insert Passenger Details");
        System.out.println("2. Insert Flight Details");
        System.out.println("3. Book Ticket");
        System.out.println("4. Cancel Ticket");
        System.out.println("5. Display Vacancy Seat Details");
        System.out.println("6. Update Passenger Information");
        System.out.println("7. Update Flight Details");
        System.out.println("8. Exit");

        System.out.print("\nEnter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                insertPassengerDetails();
                break;
            case 2:
                insertFlightDetails();
                break;
            case 3:
                bookTicket();
                break;
            case 4:
                cancelTicket();
                break;
            case 5:
                displayVacancySeatDetails();
                break;
            case 6:
                updatePassengerInformation();

```

```

        break;
    case 7:
        updateFlightDetails();
        break;
    case 8:
        exit = true;
        break;
    default:
        System.out.println("Invalid choice. Please enter a
number between 1 and 8.");
    }
}

// Close the scanner and database connection before exiting
the program
scanner.close();
try {
    if (connection != null) {
        connection.close();
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private static void insertPassengerDetails() {
    try {
        System.out.println("Enter passenger name:");
        String name = scanner.next();
        System.out.println("Enter passenger age:");
        int age = scanner.nextInt();
        System.out.println("Enter passenger phone number:");
        String phoneNumber = scanner.next();

        String query = "INSERT INTO Passenger (name, age,
phone_number) VALUES (?, ?, ?)";
        PreparedStatement preparedStatement =
connection.prepareStatement(query);
        preparedStatement.setString(1, name);
        preparedStatement.setInt(2, age);
        preparedStatement.setString(3, phoneNumber);
        preparedStatement.executeUpdate();
        System.out.println("Passenger details inserted
successfully.");
    }
}

```



```

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

private static void insertFlightDetails() {
    try {
        System.out.println("Enter flight number:");
        String flightNumber = scanner.next();
        System.out.println("Enter departure location:");
        String departure = scanner.next();
        System.out.println("Enter destination:");
        String destination = scanner.next();
        System.out.println("Enter capacity:");
        int capacity = scanner.nextInt();

        String query = "INSERT INTO Flight (flight_number,
departure, destination, capacity, remaining_seats) VALUES (?, ?, ?, ?,
?)";

        PreparedStatement preparedStatement =
connection.prepareStatement(query);
        preparedStatement.setString(1, flightNumber);
        preparedStatement.setString(2, departure);
        preparedStatement.setString(3, destination);
        preparedStatement.setInt(4, capacity);
        preparedStatement.setInt(5, capacity); // Initially, all
seats are available
        preparedStatement.executeUpdate();
        System.out.println("Flight details inserted
successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void bookTicket() {
    // Implement method to book a ticket using JDBC
    try {
        System.out.println("Enter passenger ID:");
        int passengerId = scanner.nextInt();
        System.out.println("Enter flight ID:");
        int flightId = scanner.nextInt();

        // Check if the selected flight has available seats
    }
}

```

```

        PreparedStatement checkAvailabilityStatement =
connection.prepareStatement("SELECT remaining_seats FROM Flight WHERE
flight_id = ?");
        checkAvailabilityStatement.setInt(1, flightId);
        ResultSet resultSet =
checkAvailabilityStatement.executeQuery();
        if (resultSet.next()) {
            int remainingSeats =
resultSet.getInt("remaining_seats");
            if (remainingSeats > 0) {
                // Book the ticket
                PreparedStatement bookTicketStatement =
connection.prepareStatement("INSERT INTO Ticket (passenger_id,
flight_id) VALUES (?, ?)");
                bookTicketStatement.setInt(1, passengerId);
                bookTicketStatement.setInt(2, flightId);
                bookTicketStatement.executeUpdate();

                // Update remaining seats in the Flight table
                PreparedStatement updateSeatsStatement =
connection.prepareStatement("UPDATE Flight SET remaining_seats =
remaining_seats - 1 WHERE flight_id = ?");
                updateSeatsStatement.setInt(1, flightId);
                updateSeatsStatement.executeUpdate();

                System.out.println("Ticket booked successfully.");
            } else {
                System.out.println("Sorry, no available seats for
the selected flight.");
            }
        } else {
            System.out.println("Flight not found.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void cancelTicket() {
    // Implement method to cancel a ticket using JDBC
    try {
        System.out.println("Enter ticket ID to cancel:");
        int ticketId = scanner.nextInt();
    }
}

```

```

        // Get flight ID associated with the ticket
        PreparedStatement getFlightIdStatement =
connection.prepareStatement("SELECT flight_id FROM Ticket WHERE
ticket_id = ?");
        getFlightIdStatement.setInt(1, ticketId);
        ResultSet resultSet = getFlightIdStatement.executeQuery();
        int flightId = -1;
        if (resultSet.next()) {
            flightId = resultSet.getInt("flight_id");
        }

        if (flightId != -1) {
            // Cancel the ticket
            PreparedStatement cancelTicketStatement =
connection.prepareStatement("DELETE FROM Ticket WHERE ticket_id = ?");
            cancelTicketStatement.setInt(1, ticketId);
            cancelTicketStatement.executeUpdate();

            // Update remaining seats in the Flight table
            PreparedStatement updateSeatsStatement =
connection.prepareStatement("UPDATE Flight SET remaining_seats =
remaining_seats + 1 WHERE flight_id = ?");
            updateSeatsStatement.setInt(1, flightId);
            updateSeatsStatement.executeUpdate();

            System.out.println("Ticket canceled successfully.");
        } else {
            System.out.println("Ticket not found.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void displayVacancySeatDetails() {
    // Implement method to display vacancy seat details for all
    flights using JDBC
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT *
FROM Flight");
        System.out.println("Flight ID\tFlight Number\tDeparture\t
tDestination\tCapacity\tRemaining Seats");
        while (resultSet.next()) {

```

```

        int flightId = resultSet.getInt("flight_id");
        String flightNumber =
resultSet.getString("flight_number");
        String departure = resultSet.getString("departure");
        String destination =
resultSet.getString("destination");
        int capacity = resultSet.getInt("capacity");
        int remainingSeats =
resultSet.getInt("remaining_seats");
        System.out.println(flightId + "\t\t" + flightNumber +
"\t\t" + departure + "\t\t" + destination + "\t\t" + capacity + "\t\t"
+ remainingSeats);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private static void updatePassengerInformation() {
    // Implement method to update passenger information using JDBC
    try {
        System.out.println("Enter passenger ID to update:");
        int passengerId = scanner.nextInt();
        System.out.println("Enter new name:");
        String newName = scanner.next();
        System.out.println("Enter new age:");
        int newAge = scanner.nextInt();
        System.out.println("Enter new phone number:");
        String newPhoneNumber = scanner.next();

        PreparedStatement updateStatement =
connection.prepareStatement("UPDATE Passenger SET name = ?, age = ?,
phone_number = ? WHERE passenger_id = ?");
        updateStatement.setString(1, newName);
        updateStatement.setInt(2, newAge);
        updateStatement.setString(3, newPhoneNumber);
        updateStatement.setInt(4, passengerId);
        updateStatement.executeUpdate();

        System.out.println("Passenger information updated
successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

}

private static void updateFlightDetails() {
    // Implement method to update flight details using JDBC
    try {
        System.out.println("Enter flight ID to update:");
        int flightId = scanner.nextInt();
        System.out.println("Enter new flight number:");
        String newFlightNumber = scanner.next();
        System.out.println("Enter new departure location:");
        String newDeparture = scanner.next();
        System.out.println("Enter new destination:");
        String newDestination = scanner.next();
        System.out.println("Enter new capacity:");
        int newCapacity = scanner.nextInt();

        PreparedStatement updateStatement =
connection.prepareStatement("UPDATE Flight SET flight_number = ?,
departure = ?, destination = ?, capacity = ?, remaining_seats = ?
WHERE flight_id = ?");
        updateStatement.setString(1, newFlightNumber);
        updateStatement.setString(2, newDeparture);
        updateStatement.setString(3, newDestination);
        updateStatement.setInt(4, newCapacity);
        updateStatement.setInt(5, newCapacity); // Initially, all
seats are available
        updateStatement.setInt(6, flightId);
        updateStatement.executeUpdate();

        System.out.println("Flight information updated
successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```