PHASE III: Design and App

TCSS 445 - Database Systems Design (Summer 2025)

August 04, 2025

Group Name: Team 11

Project Name: Skyhub

Members: Artem Grichanichenko, Kirill Grichanichenko

Introduction

This report is phase III of the “SkyHub" Airline Terminal Booking System for the Database Systems Design course project in Summer 2025 at the University of Washington Tacoma. Phase III contains the finalized project of the application with proper normalization, documentation, and development of the front-end application. The application serves as a simulation of an airport terminal system that manages an airport system database. The application gives the user the ability to display flights filtered by airport, view flight information, book and reserve seats, and view user booking details. The application uses a backend MySQL database used to manage all features of the application through queries and data integrity logic. The front-end uses HTML, CSS, and JavaScript tech stack for managing all backend features through an intuitive UI interface. Additionally, all database designs and optimizations are included with entity-relationship and schema diagrams with normalization proofs located in the Appendix.

Objectives

SkyHub manages flights, airports, customers, tickets, and bookings. The main objective of this project was to create a simple and effective airline reservation system that makes retrieving flight and booking information both efficient and easy to use. SkyHub is designed to support airline staff, customers, and developers. However, in its current implementation, the system is only built for customers, since staff and admin functionality have not yet been added. This leaves room for future development. After logging in, a customer can perform several different tasks in the system. They are able to view all flights across different airports and locations, or filter flights by airport to see only the flights departing from that specific airport. Customers can also check available seats for a given flight and select a seat to reserve. Once a seat is reserved, the system books the user onto the flight, and it appears on the bookings page. In the booking page, the flight information is stored automatically and saved to their profile. From there, the user can cancel flights and review all details like flight times, gate numbers, and a full price breakdown with tax and seat cost.

Scope of the Project

The SkyHub application is implemented with the intent to simulate a real-world web-based airport terminal system. The scope of the application as of the due date covers basic customer functionality such as logging in, reserving seats for flights, and managing user-reserved bookings. The application does not cover all aspects of a real-world airport system and does not dynamically manage real flights across airports, operate with real pricing procedures, or involve full administration operations. However, for future work, we intend to bring the application closer toward a more realistic representation of an airport terminal with live data and flight management, as well as robust features for role-specific users such as administrators and customers.

Other Existing Work

There are many other existing web-based airport terminals today, with almost every airport having some form of flight reservation website. However, we built our application to create as realistic a simulation as possible of those already existing applications. Some difference in our application is a more robust focus, which tries to include not only a ticket reservation scheduler but also functionality for all roles within the airport domain. Within our application, we plan to include functionality for customers, such as booking and reservation management; administrators for creating and managing flights; and other staff for passenger check-in and updates.

Overview of diagrams

The relational schema represents the main idea of how our project works. We see the defined tables, primary keys, and foreign key constraints. The schema shows how each table connects to each other, like flights are linked to airports, tickets to customers, and bookings to tickets. This schema shows how the main design would be implemented in SQL. The ER diagram builds on the schema and gives a more relational look at the system. The core entities are Customer, Flight, and Ticket, which make up the main airline booking process. The Ticket is the connecting entity between the Customer and Flight relationship, and the other entities are built around those three. Each entity has relationships that show how they work together. For example, each Flight must depart from and arrive at an Airport, while Tickets act as the middle link between Flights and Customers. Tickets also connect to bookings, where a Ticket has one booking, and a Booking must have one Ticket. Inside Booking, we added a derived attribute where the Total Price is automatically calculated through a trigger. Finally, to make sure our data is consistent, we normalized each relation to BCNF. For normalization, we looked at the functional dependencies in each table and made sure all keys are superkeys. For example, in the Ticket table, both ticket_id and the pair (flight_number, seat_number) can work as keys. This keeps the design clean and helps avoid duplicate or inconsistent data when adding, updating, or deleting.

Conclusion

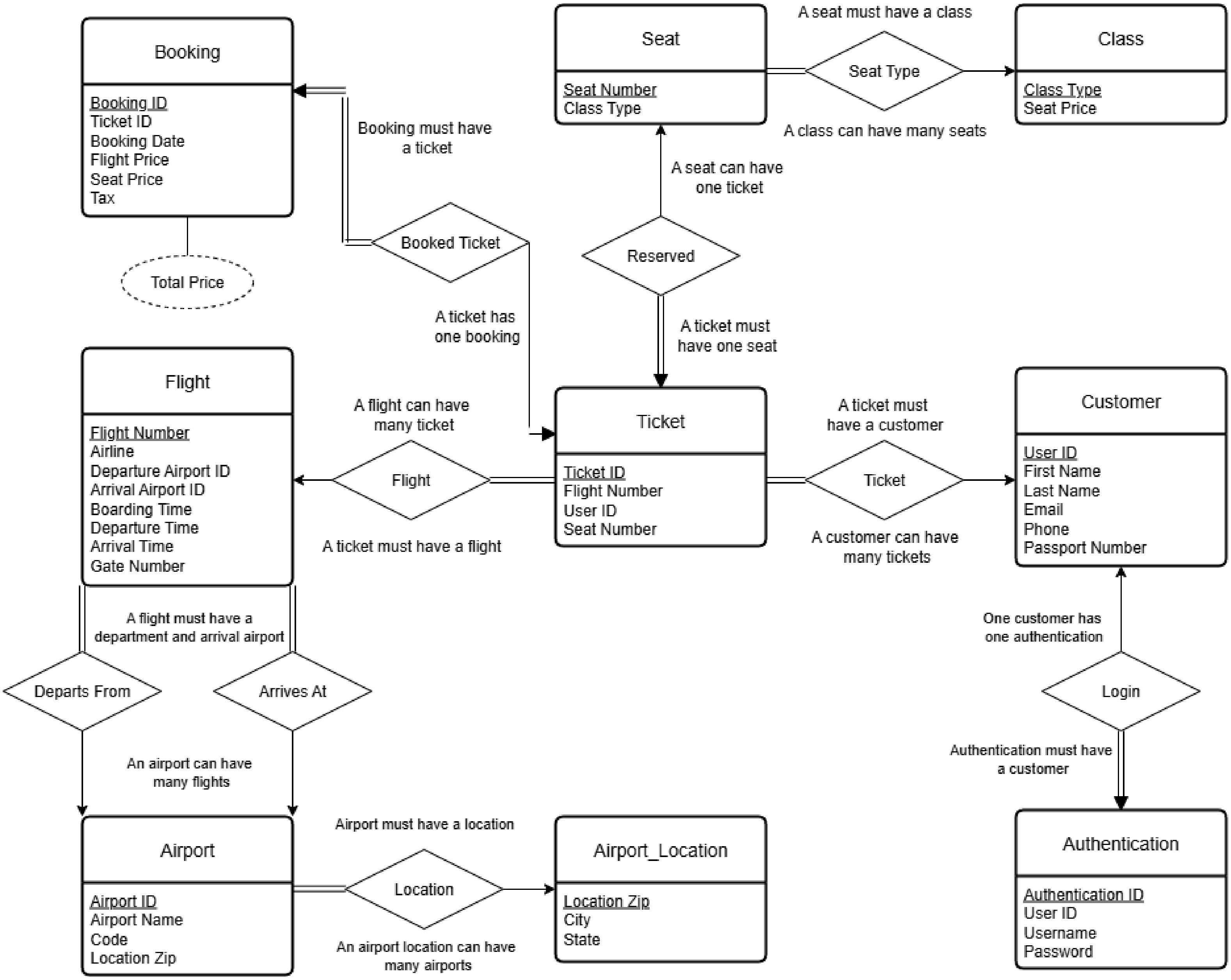
The SkyHub project gave us the chance to design and implement a working airline reservation system. Our main goal was to create a simple, user-friendly way for customers to search flights, reserve seats, and manage their bookings. Through this project, we built a relational database in MySQL, applied normalization to BCNF, and designed both schema and ER diagrams that made sure the system was consistent and not redundant. We also created a front-end using HTML, CSS, and JavaScript that connects to the database, making the system available for customers. Overall, the project showed how database concepts like constraints, triggers, and normalization can be applied in a real application. It also helped us practice connecting to backend database logic with a front-end interface. While the current version is focused only on customer features, we can continue working and add improvements like having staff and admin roles, expanding to international airports, and more realistic pricing and scheduling.

Future Work

At the current stage of the project, the application is deployed and runs on a local server with proper authentication and simple customer-role-focused features. In the future, we plan to expand the application to be more robust, with additional features, extended functionality, and enhanced roles and capabilities, as well as professional authentication, simulated payment, and flight management services. Additionally, we plan to expand the project into a microservice architecture to incorporate a separation of concerns as the project grows, thereby optimizing feature iterations. We also plan to deploy the project as a demo application at some point, either through Docker or a cloud-based platform, to provide a live demo for future employers or anyone interested in our work.

Appendix

Appendix A: ER Diagram



Appendix B: Normalization Proof to BCNF

Airport_Location

location_zip	city	state

FD1: {location_zip} -> {city, state}

Therefore: location_zip is a candidate key for Airport_Location, and it can identify the city and state.

Proof: Since FD1 determinant is a super key, this relation is in BCNF

Airport

airport_id	airport_name	code	location_zip

FD2:

{airport_id} -> {airport_name, code, location_zip}

{code} -> {airport_id, airport_name, location_zip}

Therefore: airport_id and code are both candidate keys for Airport and can identify all attributes within the table, including other keys.

Proof: Since FD2 has all the determinants as super keys, this relation is in BCNF

Customer

user_id	first_name	last_name	email	phone	passport_number

FD3:

{user_id} -> {first_name, last_name, email, phone, passport_number}

{passport_number} -> {user_id, first_name, last_name, email, phone}

{email} -> {user_id, first_name, last_name, phone, passport_number}

Therefore: user_id, email, and passport_number are all candidate keys for Customer, which can identify all attributes within the table, including other keys.

Proof: Since FD3 determinants are all superkeys, Customer is in BCNF

Authentication

authetication_id	user_id	username	password

FD:

{authentication_id} -> {user_id, username, password}

{user_id} -> {authentication_id, username, password}

{username} -> {authentication_id, user_id, password}

Therefore: user_id, username, and authentication_id are all candidate keys for Authentication, which can identify all attributes within the table, including other keys.

Proof: Since FD3 determinants are all superkeys, Authentication is in BCNF

Class

class_type	seat_price

FD4:

{class_type} -> {seat_price}

Therefore: class_type is a determinate candidate key that can acquire all attributes within Class.

Proof: Since FD4 determinate is a superkey, Class is in BCNF

Flight

flight_number	airline	departure_airport_id	arrival_airport_id	boarding_time	departure_time	arrival_time	gate_number

FD5:

{flight_number} -> {airline, departure_airport_id, arrival_airport_id, boarding_time, departure_time, arrival_time, gate_number}

{departure_airport_id, gate_number} -> {flight_number, airline, arrival_airport_id, boarding_time, departure_time, arrival_time}

Therefore: flight_number and departure_airport with gate_number are determinants that can identify all attributes within the Flight table.

Proof: Since FD4 determinants are all superkeys, Class is in BCNF.

Seat

seat_number	class_type

FD6:

{seat_number} -> {class_type}

Therefore: seat_number is a candidate key used to acquire class_type

Proof: Since FD6 {seat_number } -> {class_type}is a determinant as a candidate key Seat is in BCNF.

Ticket

ticket_id	flight_number	user_id	seat_number

FD7:

{ticket_id} -> {flight_number, user_id, seat_number}

{flight_number, seat_number} -> {ticket_id, user_id}

Therefore: ticket_id and flight_number with seat_number are super keys that can be used to determine all attributes within the table

Proof: Since all determinants are super keys, Ticket is in BCNF

Booking

booking_id	ticket_id	booking_date	flight_price	seat_price	tax	total_price

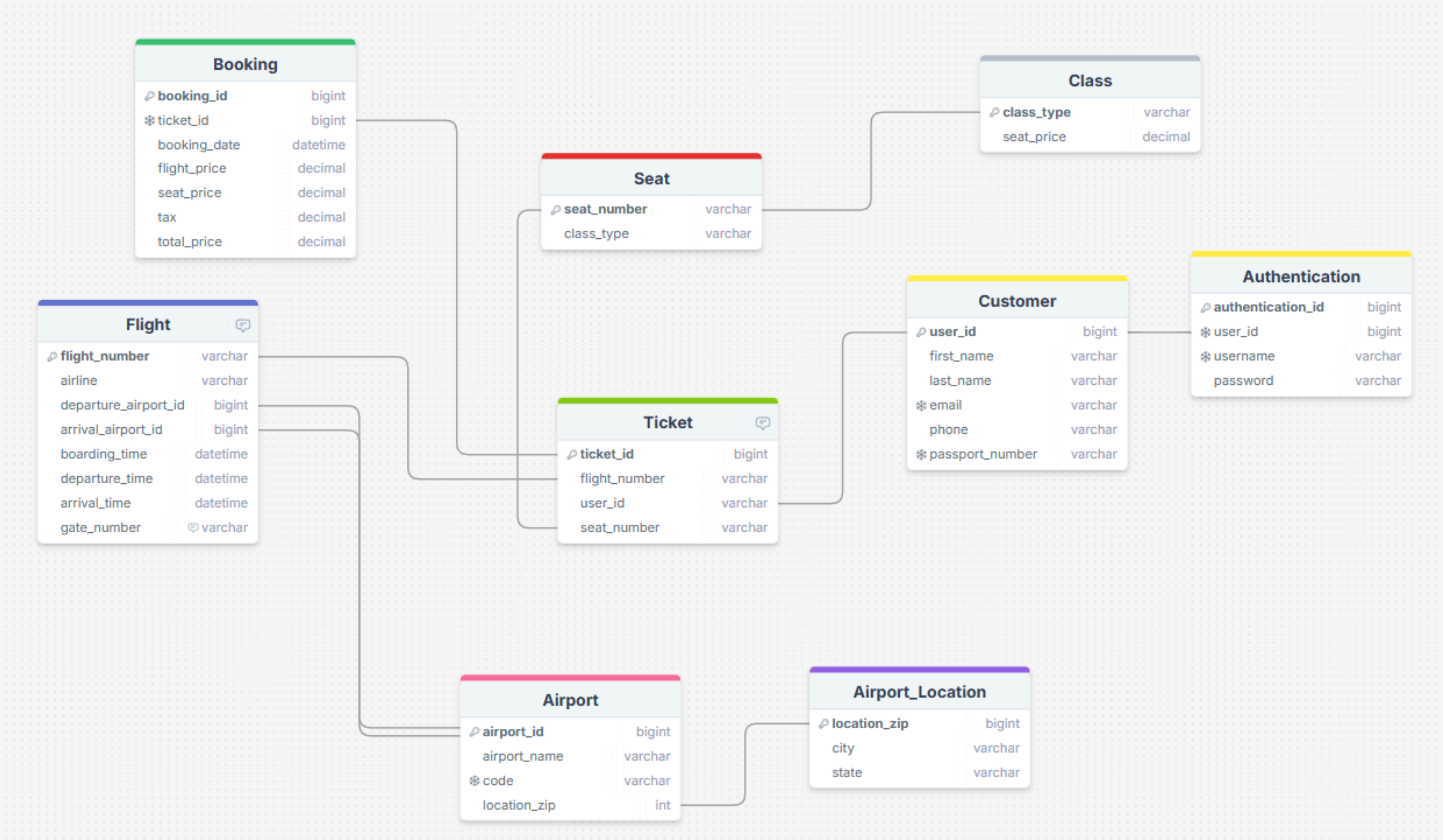
{booking_id} -> {ticket_id, booking_date, flight_price, seat_price, tax, total_price}

{ticket_id} -> {booking_id, booking_date, flight_price, seat_price, tax, total_price}

Therefore: booking_id and ticket_id are candidate keys that can be used to determine all other attributes in Booking

Proof: Since all determinants are super keys, Booking is in BCNF

Appendix C: Relational Schema Diagram



List of changes from phase 2

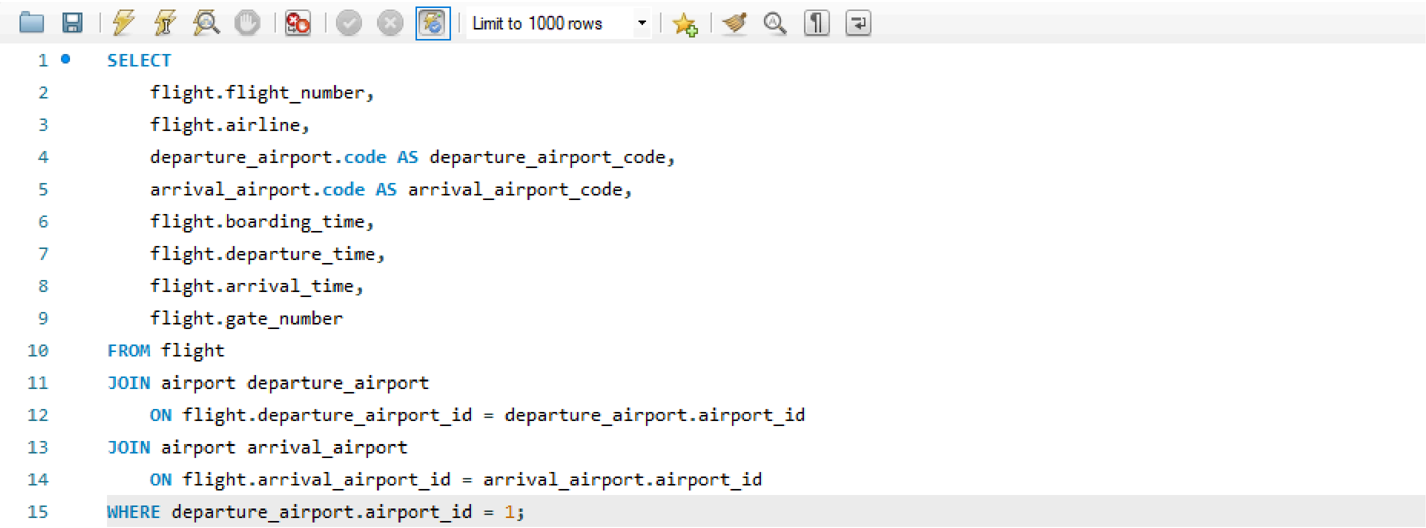
1. Added an authentication table to handle customer login and to separate customers username and password from their personal information.
2. Complete seat entity restructure.
   a. Phase 2 - Used booleans to keep track of seat availability on specific flights (required a lot of dynamic data management)
   b. Phase 3 current - includes seat table as a general representation of seats on a flight (reduces complex data management by putting the management responsibility on the ticket table).

        - Requires only a set number of data inside the seat table
        - Uniqueness and seat management happens within the ticket table.

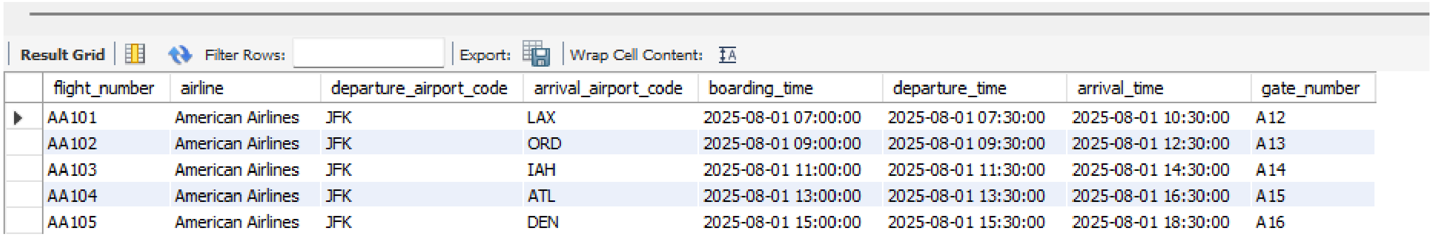
3. Combined boarding table into flight table
    a. Easier to manage boarding information of individual flights through one table.
    b. Easier to maintain uniqueness of gate id and departure airport id through one table.

Appendix D: Screenshots of functional SQL Queries (with proper documentation)

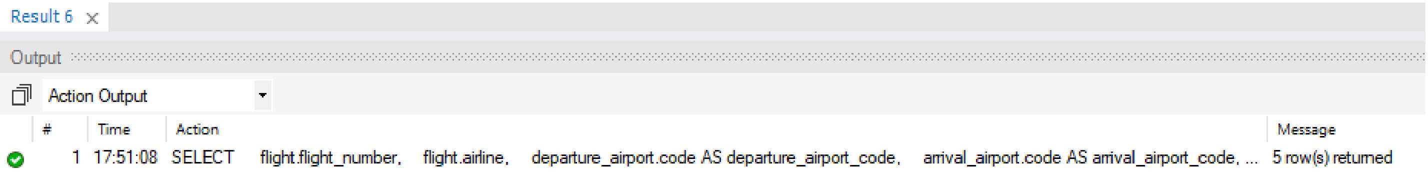
Query 1: Show all flights at an airport, including its name and code.



```
SELECT
    flight.flight_number,
    flight.airline,
    departure_airport.code AS departure_airport_code,
    arrival_airport.code AS arrival_airport_code,
    flight.boarding_time,
    flight.departure_time,
    flight.arrival_time,
    flight.gate_number
FROM flight
JOIN airport departure_airport
    ON flight.departure_airport_id = departure_airport.airport_id
JOIN airport arrival_airport
    ON flight.arrival_airport_id = arrival_airport.airport_id
WHERE departure_airport.airport_id = 1;
```

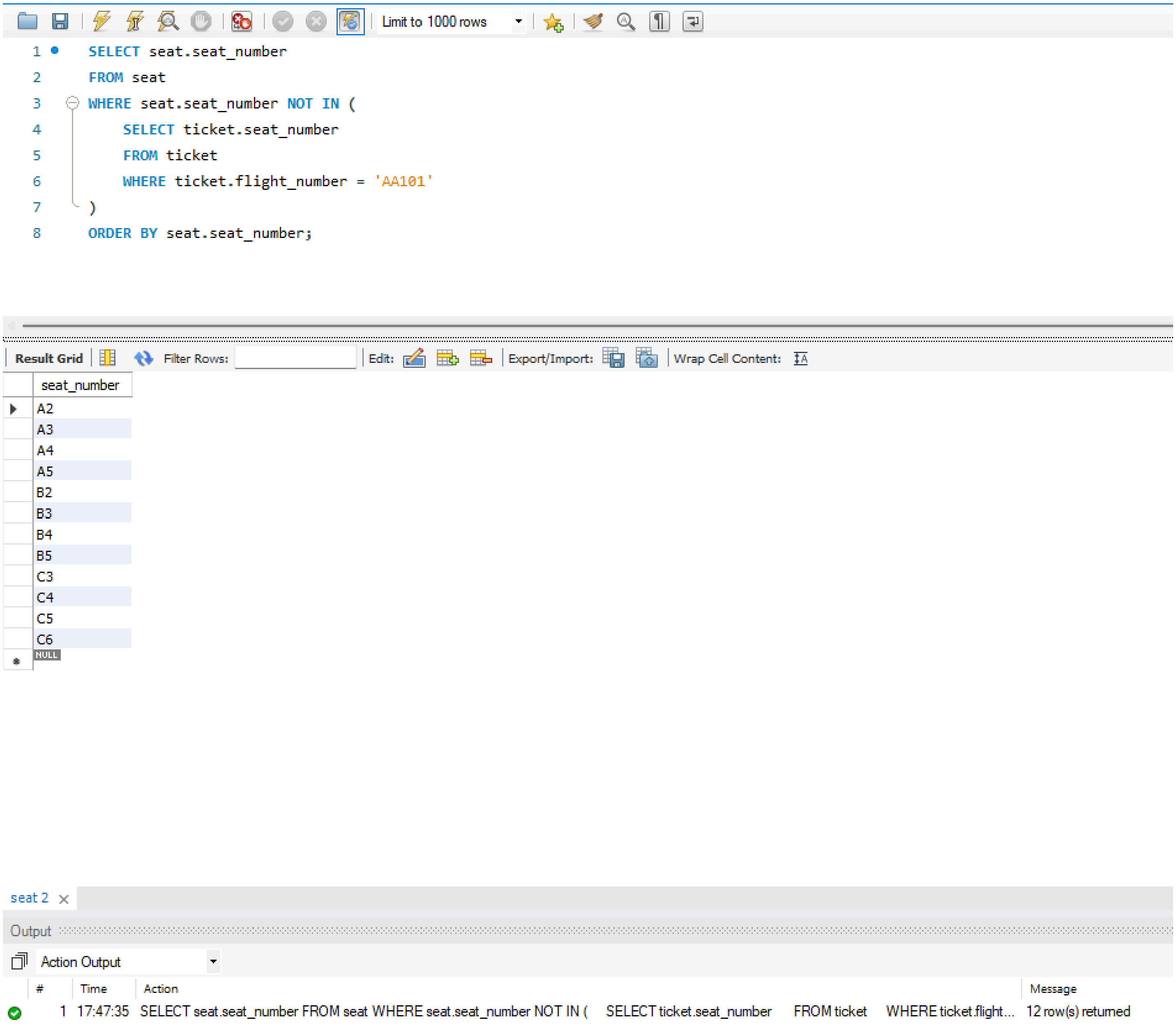


flight_number	airline	departure_airport_code	arrival_airport_code	boarding_time	departure_time	arrival_time	gate_number
AA101	American Airlines	JFK	LAX	2025-08-01 07:00:00	2025-08-01 07:30:00	2025-08-01 10:30:00	A12
AA102	American Airlines	JFK	ORD	2025-08-01 09:00:00	2025-08-01 09:30:00	2025-08-01 12:30:00	A13
AA103	American Airlines	JFK	IAH	2025-08-01 11:00:00	2025-08-01 11:30:00	2025-08-01 14:30:00	A14
AA104	American Airlines	JFK	ATL	2025-08-01 13:00:00	2025-08-01 13:30:00	2025-08-01 16:30:00	A15
AA105	American Airlines	JFK	DEN	2025-08-01 15:00:00	2025-08-01 15:30:00	2025-08-01 18:30:00	A16



```
#   Time      Action                                                                                                                              Message
1   17:51:08  SELECT   flight.flight_number,   flight.airline,   departure_airport.code AS departure_airport_code,   arrival_airport.code AS arrival_airport_code, ...   5 row(s) returned
```

Query 2: List available seat numbers for the given flight.

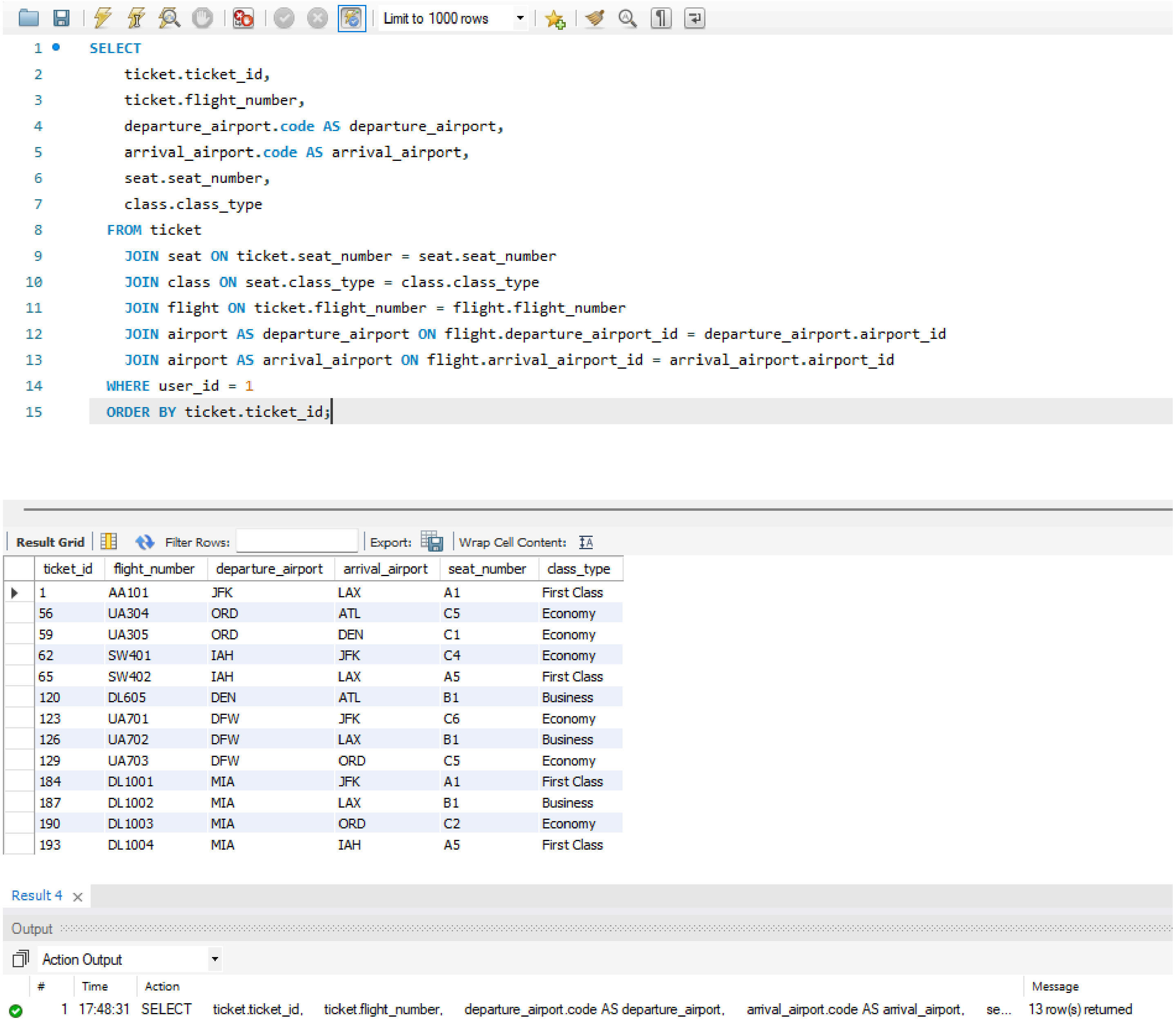


```sql
SELECT seat.seat_number
FROM seat
WHERE seat.seat_number NOT IN (
    SELECT ticket.seat_number
    FROM ticket
    WHERE ticket.flight_number = 'AA101'
)
ORDER BY seat.seat_number;
```

seat_number
A2
A3
A4
A5
B2
B3
B4
B5
C3
C4
C5
C6
NULL

```
#   Time       Action                                                                                                                              Message
1   17:47:35   SELECT seat.seat_number FROM seat WHERE seat.seat_number NOT IN (     SELECT ticket.seat_number     FROM ticket     WHERE ticket.flight...   12 row(s) returned
```

Query 3: Purpose: Show all bookings made by user.

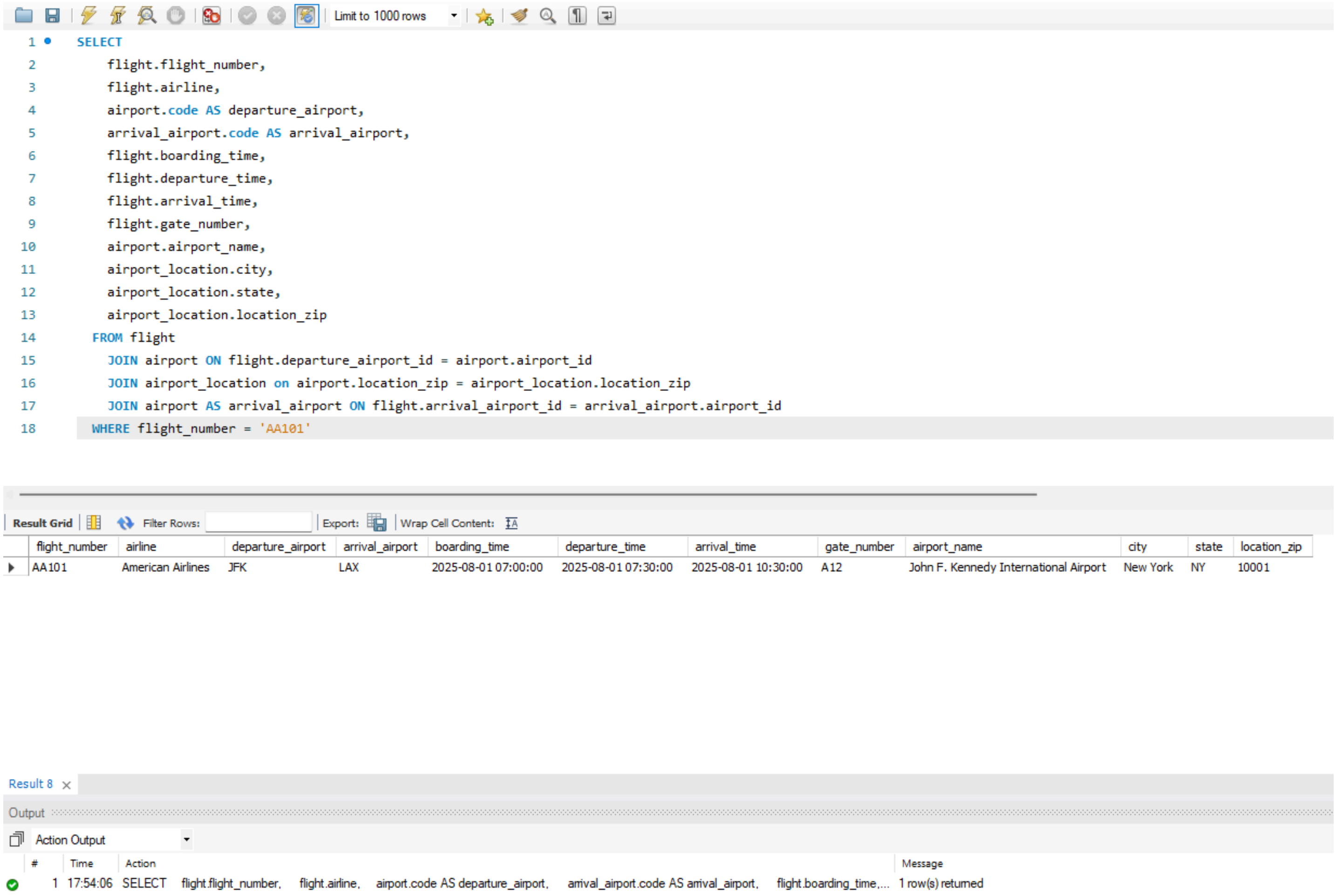


```sql
SELECT
    ticket.ticket_id,
    ticket.flight_number,
    departure_airport.code AS departure_airport,
    arrival_airport.code AS arrival_airport,
    seat.seat_number,
    class.class_type
  FROM ticket
    JOIN seat ON ticket.seat_number = seat.seat_number
    JOIN class ON seat.class_type = class.class_type
    JOIN flight ON ticket.flight_number = flight.flight_number
    JOIN airport AS departure_airport ON flight.departure_airport_id = departure_airport.airport_id
    JOIN airport AS arrival_airport ON flight.arrival_airport_id = arrival_airport.airport_id
  WHERE user_id = 1
  ORDER BY ticket.ticket_id;
```

ticket_id	flight_number	departure_airport	arrival_airport	seat_number	class_type
1	AA101	JFK	LAX	A1	First Class
56	UA304	ORD	ATL	C5	Economy
59	UA305	ORD	DEN	C1	Economy
62	SW401	IAH	JFK	C4	Economy
65	SW402	IAH	LAX	A5	First Class
120	DL605	DEN	ATL	B1	Business
123	UA701	DFW	JFK	C6	Economy
126	UA702	DFW	LAX	B1	Business
129	UA703	DFW	ORD	C5	Economy
184	DL1001	MIA	JFK	A1	First Class
187	DL1002	MIA	LAX	B1	Business
190	DL1003	MIA	ORD	C2	Economy
193	DL1004	MIA	IAH	A5	First Class

	#	Time	Action	Message
✓	1	17:48:31	SELECT ticket.ticket_id, ticket.flight_number, departure_airport.code AS departure_airport, arrival_airport.code AS arrival_airport, se...	13 row(s) returned

Query 4: Get all flight details of a given flight, including the airport and airport location of the flight



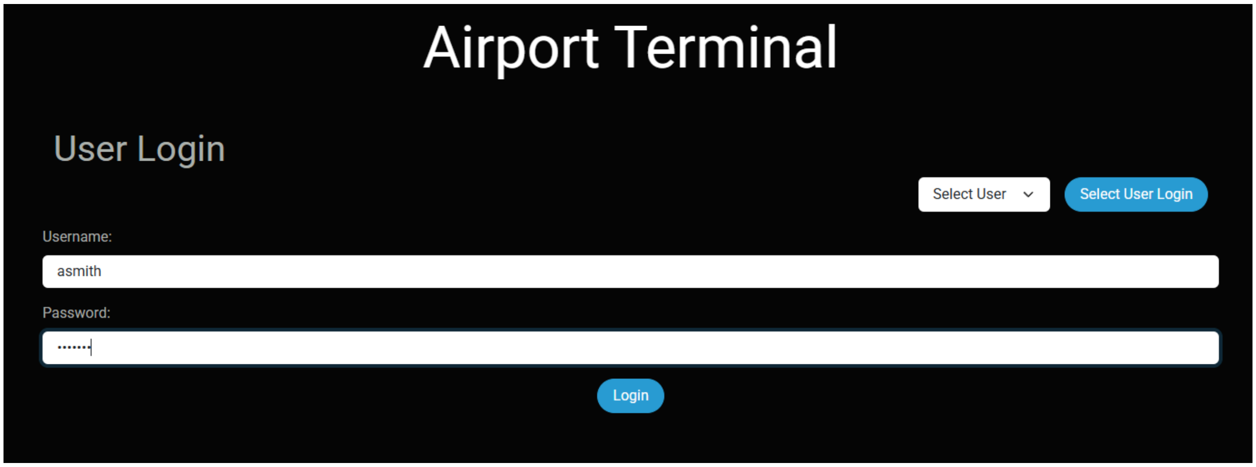
```sql
SELECT
    flight.flight_number,
    flight.airline,
    airport.code AS departure_airport,
    arrival_airport.code AS arrival_airport,
    flight.boarding_time,
    flight.departure_time,
    flight.arrival_time,
    flight.gate_number,
    airport.airport_name,
    airport_location.city,
    airport_location.state,
    airport_location.location_zip
  FROM flight
    JOIN airport ON flight.departure_airport_id = airport.airport_id
    JOIN airport_location on airport.location_zip = airport_location.location_zip
    JOIN airport AS arrival_airport ON flight.arrival_airport_id = arrival_airport.airport_id
  WHERE flight_number = 'AA101'
```

flight_number	airline	departure_airport	arrival_airport	boarding_time	departure_time	arrival_time	gate_number	airport_name	city	state	location_zip
AA101	American Airlines	JFK	LAX	2025-08-01 07:00:00	2025-08-01 07:30:00	2025-08-01 10:30:00	A12	John F. Kennedy International Airport	New York	NY	10001

```
#  Time      Action                                                                                                                          Message
1  17:54:06  SELECT  flight.flight_number,  flight.airline,  airport.code AS departure_airport,  arrival_airport.code AS arrival_airport,  flight.boarding_time,...  1 row(s) returned
```

Appendix E: Screenshots of functional Web Interface (with proper documentation)

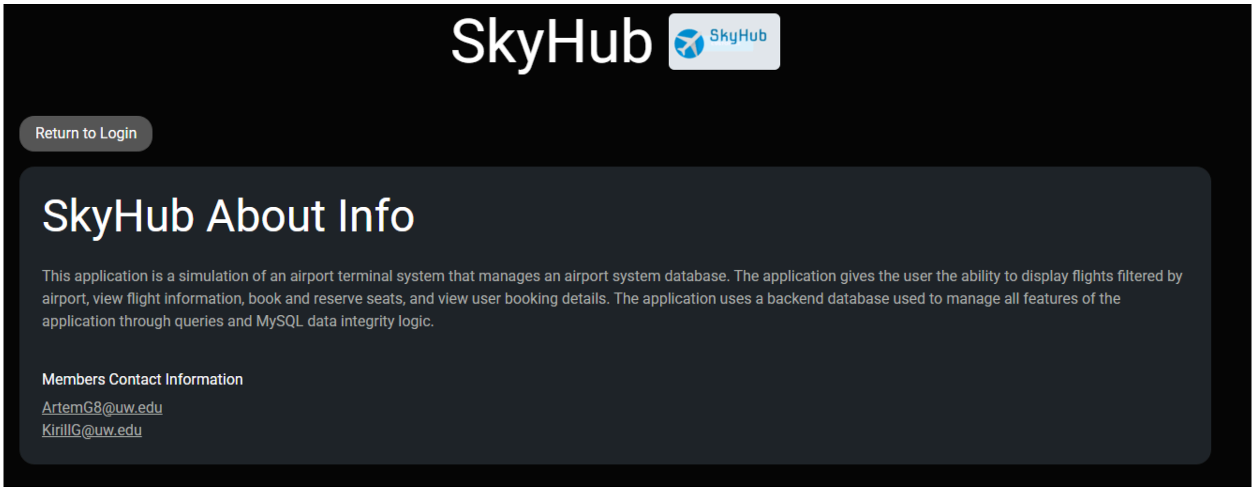
Screenshot 1: Authentication login page



Username: asmith

Password: •••••••

Screenshot 2: SkyHub About Page



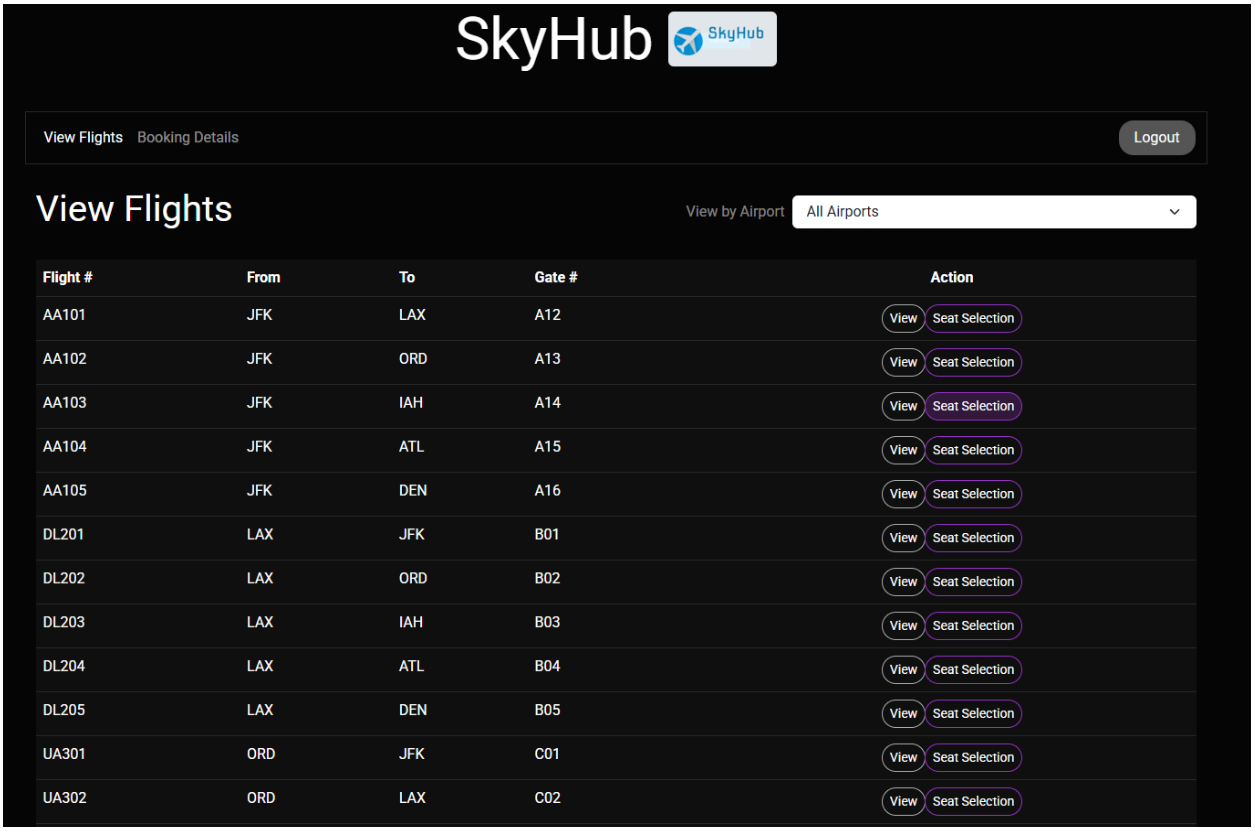
SkyHub About Info

This application is a simulation of an airport terminal system that manages an airport system database. The application gives the user the ability to display flights filtered by airport, view flight information, book and reserve seats, and view user booking details. The application uses a backend database used to manage all features of the application through queries and MySQL data integrity logic.

Members Contact Information

ArtemG8@uw.edu
KirillG@uw.edu

Screenshot 3: View flights page with all flights

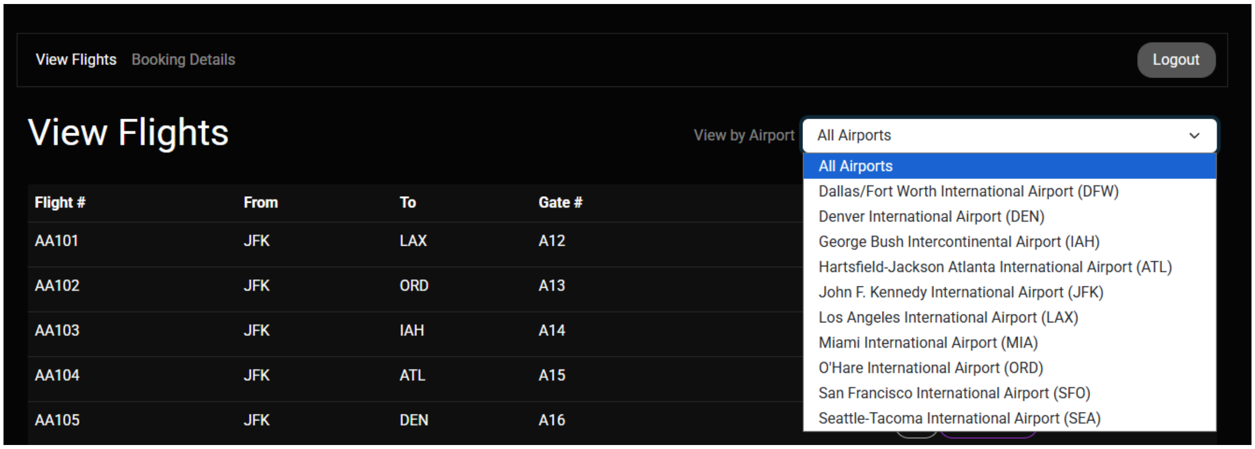


View Flights

View by Airport: All Airports

Flight #	From	To	Gate #	Action
AA101	JFK	LAX	A12	View Seat Selection
AA102	JFK	ORD	A13	View Seat Selection
AA103	JFK	IAH	A14	View Seat Selection
AA104	JFK	ATL	A15	View Seat Selection
AA105	JFK	DEN	A16	View Seat Selection
DL201	LAX	JFK	B01	View Seat Selection
DL202	LAX	ORD	B02	View Seat Selection
DL203	LAX	IAH	B03	View Seat Selection
DL204	LAX	ATL	B04	View Seat Selection
DL205	LAX	DEN	B05	View Seat Selection
UA301	ORD	JFK	C01	View Seat Selection
UA302	ORD	LAX	C02	View Seat Selection

Screenshot 4: Query to filter flights by an airport

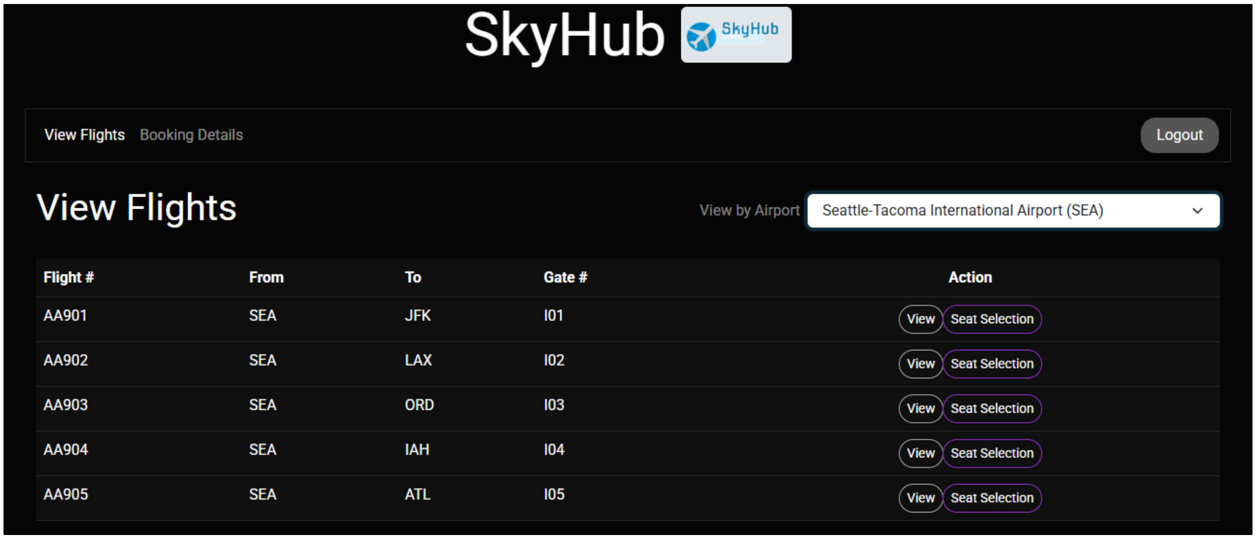

View Flights
Booking Details
Logout
All Airports
Dallas/Fort Worth International Airport (DFW)
Denver International Airport (DEN)
George Bush Intercontinental Airport (IAH)
Hartsfield-Jackson Atlanta International Airport (ATL)
John F. Kennedy International Airport (JFK)
Los Angeles International Airport (LAX)
Miami International Airport (MIA)
O'Hare International Airport (ORD)
San Francisco International Airport (SFO)
Seattle-Tacoma International Airport (SEA)

View Flights

View by Airport: All Airports

Flight #	From	To	Gate #
AA101	JFK	LAX	A12
AA102	JFK	ORD	A13
AA103	JFK	IAH	A14
AA104	JFK	ATL	A15
AA105	JFK	DEN	A16

Screenshot 5: Query to filter flights by an airport in action

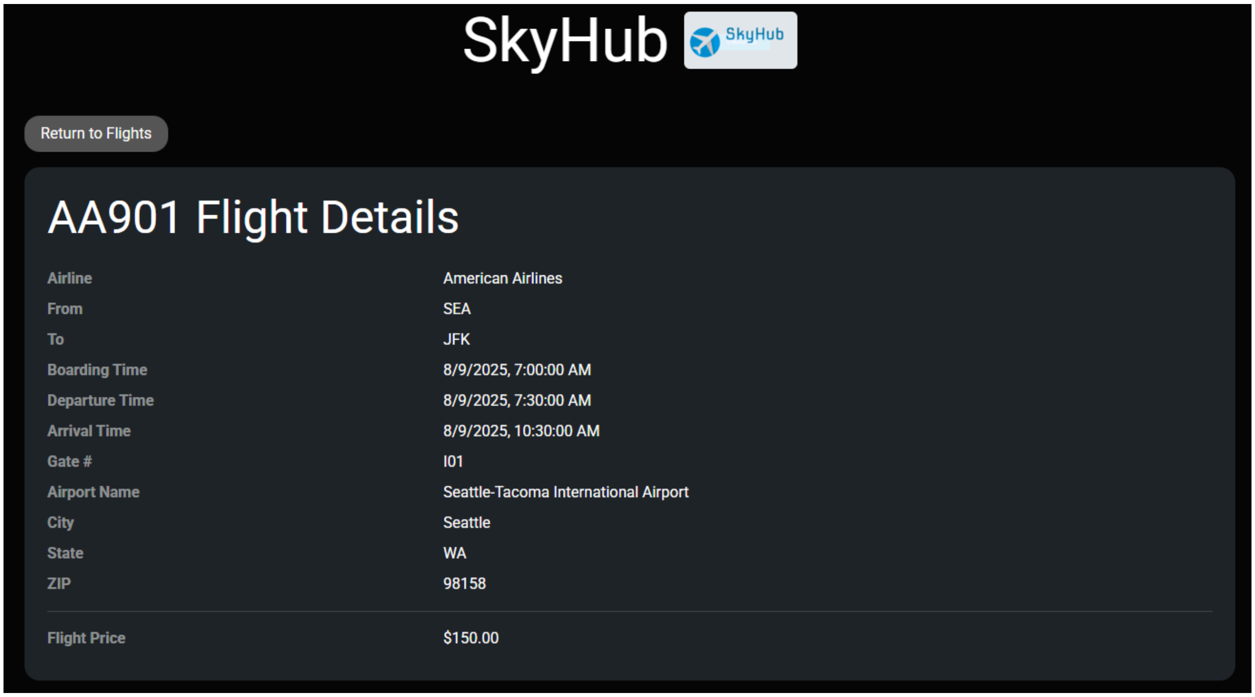


View Flights

View by Airport: Seattle-Tacoma International Airport (SEA)

Flight #	From	To	Gate #	Action
AA901	SEA	JFK	I01	View Seat Selection
AA902	SEA	LAX	I02	View Seat Selection
AA903	SEA	ORD	I03	View Seat Selection
AA904	SEA	IAH	I04	View Seat Selection
AA905	SEA	ATL	I05	View Seat Selection

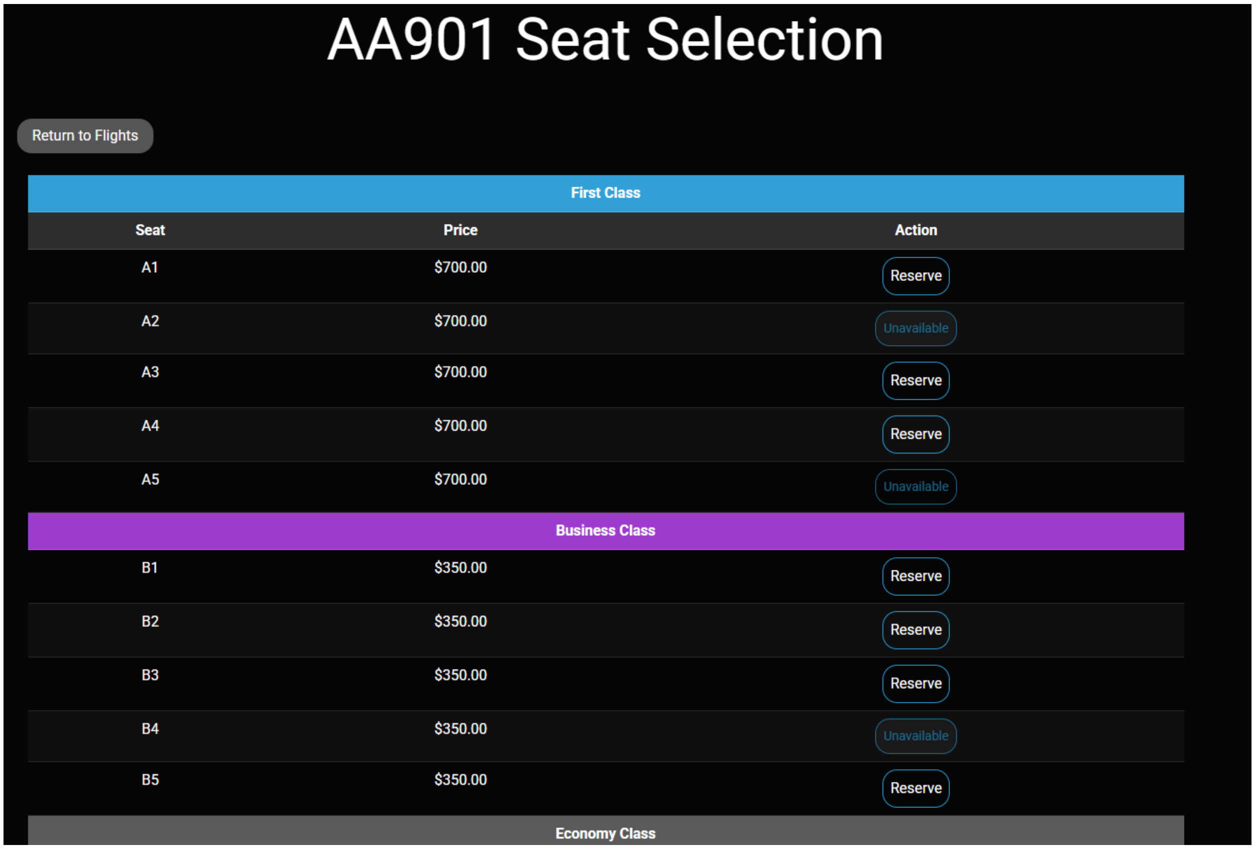
Screenshot 6: Query to show all flight details



AA901 Flight Details

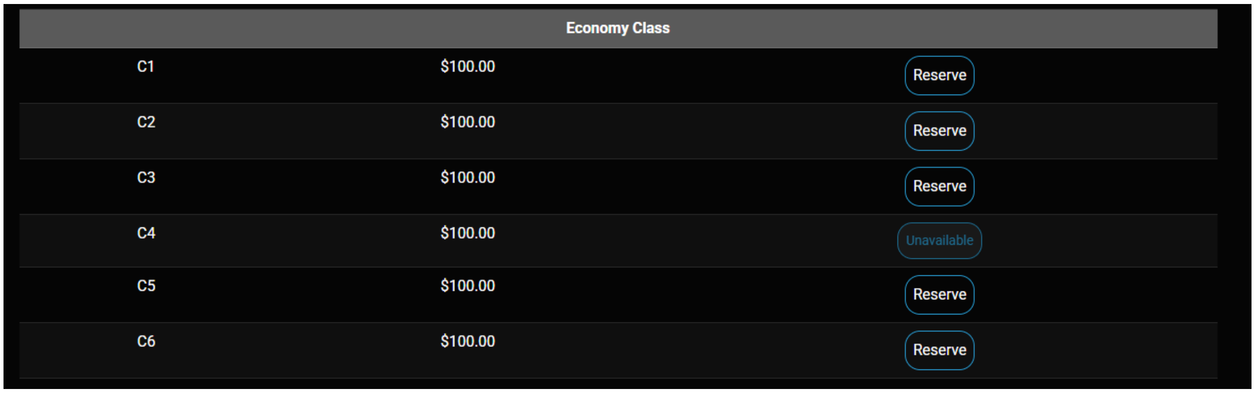
Airline	American Airlines
From	SEA
To	JFK
Boarding Time	8/9/2025, 7:00:00 AM
Departure Time	8/9/2025, 7:30:00 AM
Arrival Time	8/9/2025, 10:30:00 AM
Gate #	I01
Airport Name	Seattle-Tacoma International Airport
City	Seattle
State	WA
ZIP	98158
Flight Price	$150.00

Screenshot 7, Query to show all available seats for a given flight



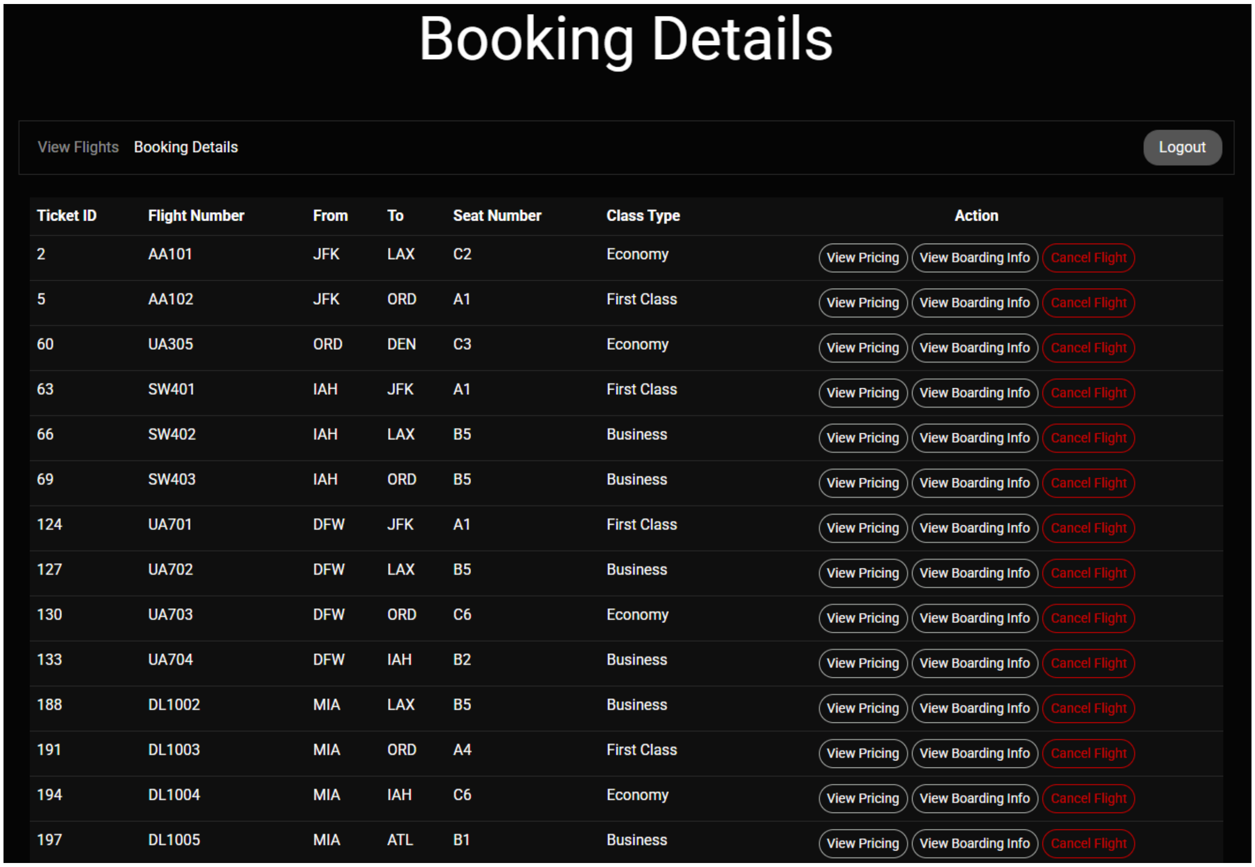
AA901 Seat Selection

First Class		
Seat	**Price**	**Action**
A1	$700.00	Reserve
A2	$700.00	Unavailable
A3	$700.00	Reserve
A4	$700.00	Reserve
A5	$700.00	Unavailable
Business Class		
B1	$350.00	Reserve
B2	$350.00	Reserve
B3	$350.00	Reserve
B4	$350.00	Unavailable
B5	$350.00	Reserve
Economy Class		



Economy Class		
C1	$100.00	Reserve
C2	$100.00	Reserve
C3	$100.00	Reserve
C4	$100.00	Unavailable
C5	$100.00	Reserve
C6	$100.00	Reserve

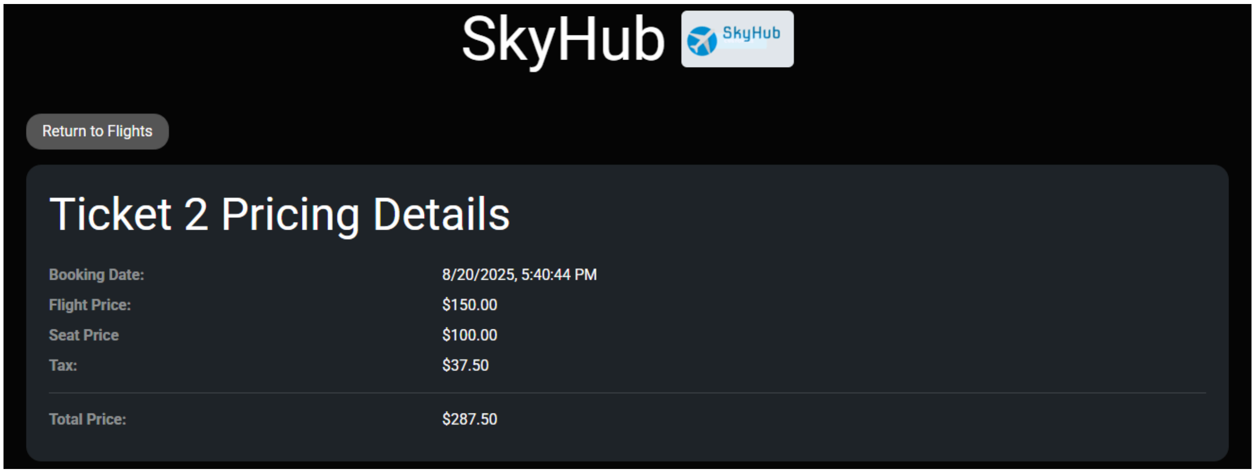
Screenshot 8, Query to show all bookings made by user



Booking Details

Ticket ID	Flight Number	From	To	Seat Number	Class Type	Action
2	AA101	JFK	LAX	C2	Economy	View Pricing View Boarding Info Cancel Flight
5	AA102	JFK	ORD	A1	First Class	View Pricing View Boarding Info Cancel Flight
60	UA305	ORD	DEN	C3	Economy	View Pricing View Boarding Info Cancel Flight
63	SW401	IAH	JFK	A1	First Class	View Pricing View Boarding Info Cancel Flight
66	SW402	IAH	LAX	B5	Business	View Pricing View Boarding Info Cancel Flight
69	SW403	IAH	ORD	B5	Business	View Pricing View Boarding Info Cancel Flight
124	UA701	DFW	JFK	A1	First Class	View Pricing View Boarding Info Cancel Flight
127	UA702	DFW	LAX	B5	Business	View Pricing View Boarding Info Cancel Flight
130	UA703	DFW	ORD	C6	Economy	View Pricing View Boarding Info Cancel Flight
133	UA704	DFW	IAH	B2	Business	View Pricing View Boarding Info Cancel Flight
188	DL1002	MIA	LAX	B5	Business	View Pricing View Boarding Info Cancel Flight
191	DL1003	MIA	ORD	A4	First Class	View Pricing View Boarding Info Cancel Flight
194	DL1004	MIA	IAH	C6	Economy	View Pricing View Boarding Info Cancel Flight
197	DL1005	MIA	ATL	B1	Business	View Pricing View Boarding Info Cancel Flight

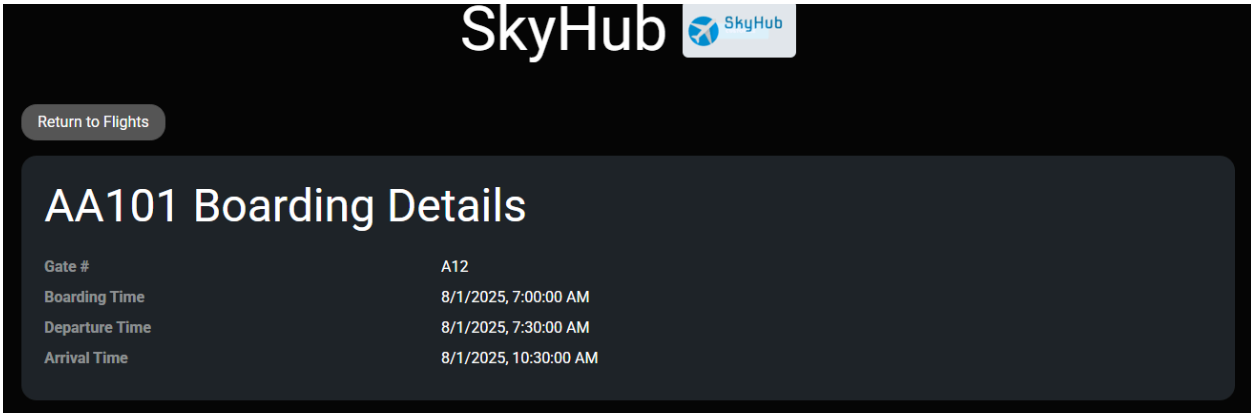
Screenshot 9, View pricing page



Ticket 2 Pricing Details

Booking Date:	8/20/2025, 5:40:44 PM
Flight Price:	$150.00
Seat Price	$100.00
Tax:	$37.50
Total Price:	$287.50

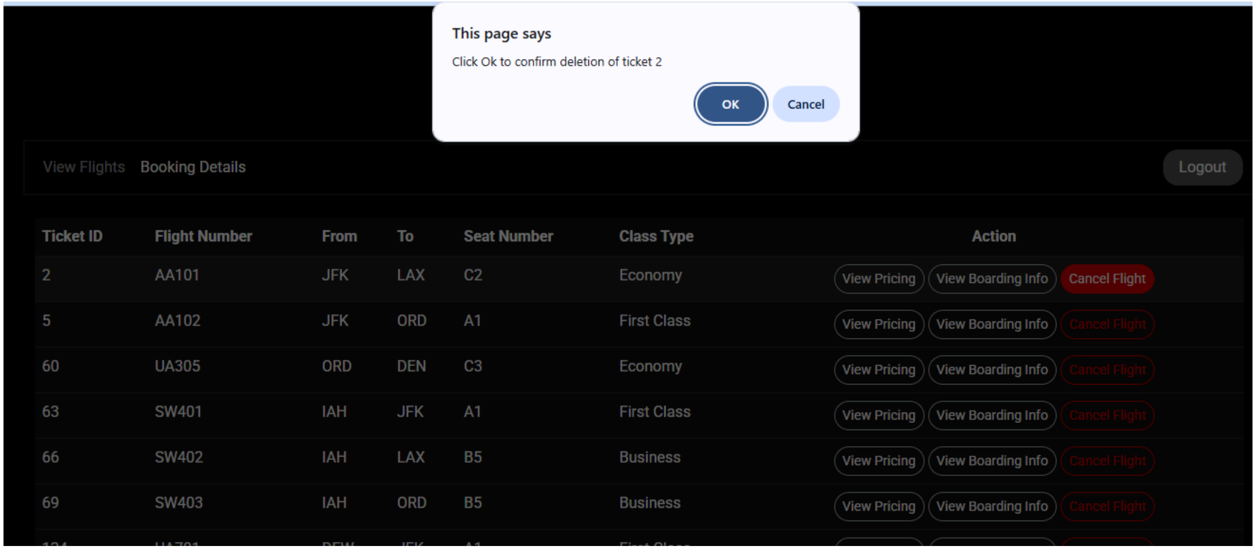
Screenshot 10, View boarding details page



AA101 Boarding Details

Gate #	A12
Boarding Time	8/1/2025, 7:00:00 AM
Departure Time	8/1/2025, 7:30:00 AM
Arrival Time	8/1/2025, 10:30:00 AM

Screenshot 11, Cancel confirmation message if user selects *Cancel Flight*



This page says

Click Ok to confirm deletion of ticket 2

Ticket ID	Flight Number	From	To	Seat Number	Class Type	Action
2	AA101	JFK	LAX	C2	Economy	View Pricing View Boarding Info Cancel Flight
5	AA102	JFK	ORD	A1	First Class	View Pricing View Boarding Info Cancel Flight
60	UA305	ORD	DEN	C3	Economy	View Pricing View Boarding Info Cancel Flight
63	SW401	IAH	JFK	A1	First Class	View Pricing View Boarding Info Cancel Flight
66	SW402	IAH	LAX	B5	Business	View Pricing View Boarding Info Cancel Flight
69	SW403	IAH	ORD	B5	Business	View Pricing View Boarding Info Cancel Flight

References

[1] W3Schools, "HTML Reference," [Online]. Available: https://www.w3schools.com/tags/. [Accessed: 20-Aug-2025].

[2] W3Schools, "CSS Tutorial," [Online]. Available: https://www.w3schools.com/css/default.asp. [Accessed: 20-Aug-2025].

[3] W3Schools, "JavaScript and HTML DOM Reference," [Online]. Available: https://www.w3schools.com/jsrEF/default.asp. [Accessed: 20-Aug-2025].

[4] Mozilla Developer Network (MDN), "JavaScript Reference," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference. [Accessed: 20-Aug-2025].

[5] jQuery Foundation, "jQuery," [Online]. Available: https://jquery.com/. [Accessed: 20-Aug-2025].

[6] W3Schools, "jQuery Tutorial," [Online]. Available: https://www.w3schools.com/jquery/. [Accessed: 20-Aug-2025].

[7] Bootstrap, "Bootstrap," [Online]. Available: https://getbootstrap.com/. [Accessed: 20-Aug-2025].

[8] Bootswatch, "Cyborg Theme," [Online]. Available: https://bootswatch.com/cyborg/. [Accessed: 20-Aug-2025].

[9] W3Schools, "MySQL Tutorial," [Online]. Available: https://www.w3schools.com/mysql/default.asp. [Accessed: 20-Aug-2025].

[10] University of Washington, "TCSS460: Web Programming," [Online]. Available: https://courses.washington.edu/tcss460/. [Accessed: 20-Aug-2025].

[11] YouTube, "MySQL Node.js Express," [Online]. Available: https://youtube/Hej48pi_lOc. [Accessed: 20-Aug-2025].

[12] ChatGPT. OpenAI. Accessed: Jul. 30, 2025. [Online]. Prompt: "Generate sample data for this database script:" Available: https://chat.openai.com/chat