Justin Cheok

Professor Dey

Introduction to Databases

2 December 2023

Application Use Cases

1. View all flights without logging in:
    a. ‘SELECT * FROM flight’. Select all flights from flight table, display the data about them.
2. Register, self-explanatory: Insert the associated values into
    a. User Register:
        i. 'SELECT * FROM customer WHERE email = %s', to check if customer account already exists
        ii. 'INSERT INTO customer VALUES(email, md5(password), firstName, lastName, buildingNum, street, aptNum, city, state, zipcode, passportNumber, passportExpiration, passportCountry, dateOfBirth), to insert values into customer table
    b. Staff Register:
        i. 'SELECT * FROM airlinestaff WHERE username = %s and password = %s', to check if the staff already exists in the table
        ii. 'INSERT INTO airlinestaff VALUES(username, md5(password), airlineName, firstName, lastName, dateOfBirth), to insert values into airlinestaff table
3. Login
    a. User Login:
        i. 'SELECT * FROM customer WHERE email = %s and password = md5(%s)', to check if account exists
        ii. 'GRANT SELECT ON airplane_system.* TO user@localhost', 'GRANT DELETE ON airplane_system.purchase TO user@localhost', 'GRANT INSERT ON airplane_system.rating TO user@localhost', 'GRANT INSERT ON airplane_system.purchase TO user@localhost', give specific SQL permissions to users
    b. Staff Login:
        i. 'SELECT * FROM airlinestaff WHERE username = %s and password = md5(%s)', to check if account exists
        ii. 'GRANT ALL PRIVILEGES ON airplane_system TO user@localhost', give all SQL permissions to staff

Customer Use Cases

1. View my flights:
    a. 'SELECT * FROM flight as f, ticket as t, purchase as p WHERE p.email = email AND p.TicketID = t.TicketID AND t.FlightNum = f.FlightNum AND f.departureTime > NOW()', to select all upcoming flights that the user has purchased
2. Search for flights:

a.

```
query = 'SELECT * FROM flight WHERE departureTime > NOW()'
params = []
if departureAirport:
    query += ' AND departureAirport = %s'
    params.append(departureAirport)
if arrivalAirport:
    query += ' AND arrivalAirport = %s'
    params.append(arrivalAirport)
if departureTime:
    query += ' AND DATE(departureTime) = %s'
    params.append(departureTime)
if arrivalTime:
    query += ' AND DATE(arrivalTime) = %s'
    params.append(arrivalTime)
cursor.execute(query, params)
```

Note: query will change based on what the user adds as additional field requirements, however the base query involves selecting all future flights.

3. Purchase tickets:
    a. 'SELECT * FROM ticket WHERE FlightNum = %s AND TicketID NOT IN (SELECT TicketID FROM purchase)', to select the available tickets, in which we will use cursor.fetchone() to obtain the first available ticket for that flight
    b. 'SELECT basePrice FROM flight WHERE FlightNum = %s', to obtain the base price of the ticket
    c. ‘INSERT INTO purchase VALUES(ticketID, email, cardType, cardNum, cardExp, cardName, purchaseTime, price, ticketFirstName, lastName, dateOfBirth)’, to log the purchase into the table
4. Cancel trip:
    a. 'DELETE FROM purchase WHERE TicketID = %s AND email = %s', to remove purchase from table
5. Give ratings & comment on previous flights:
    a. 'SELECT * FROM flight as f, ticket as t, purchase as p WHERE p.email = email AND p.TicketID = t.TicketID AND t.FlightNum = f.FlightNum AND f.arrivalTime < NOW()', to show all of their previous flights that they can review
    b. ‘INSERT INTO rating VALUES(email, rating, flightNum, comment)', insert rating into table
6. Track my spending:
    a. 'SELECT SUM(calculatedPrice) FROM ticket as t, purchase as p WHERE t.ticketID = p.TicketID AND p.email = %s AND p.purchaseTime BETWEEN DATE_SUB(NOW(), INTERVAL 6 MONTH) AND NOW()', by default show the sum of the prices of the tickets from the last 6 months
    b. 'SELECT SUM(calculatedPrice) FROM ticket as t, purchase as p WHERE t.ticketID = p.TicketID AND p.email = email AND p.purchaseTime BETWEEN dateFrom AND dateTo', calculate sum of all prices of tickets from whatever dates that they want to specify.
7. Logout:
    a. No SQL query here

Airline Staff Use Cases:

1. View flights:

a. By default, the query will select all upcoming flights for the next 30 days from the same airline that the staff is working for. Otherwise, it will use whatever fields are provided.

```
# Build the base query
query = 'SELECT * FROM flight WHERE AirlineName = %s'

# Create a list to store the query parameters
params = [airlineName]

# Check if startDate and endDate are provided
if startDate and endDate:
    query += ' AND DepartureTime BETWEEN %s AND %s'
    params.extend([startDate, endDate])

# Check if departureAirport is provided
if departureAirport:
    query += ' AND DepartureAirport = %s'
    params.append(departureAirport)

# Check if arrivalAirport is provided
if arrivalAirport:
    query += ' AND ArrivalAirport = %s'
    params.append(arrivalAirport)

# If none of these fields are provided, show flights for the next 30 days
if not startDate and not endDate and not departureAirport and not arrivalAirport:
    query += ' AND DepartureTime BETWEEN NOW() AND DATE_ADD(NOW(), INTERVAL 30 DAY)'
# Add the ORDER BY clause
query += ' ORDER BY DepartureTime'
```

2. Create new flights:
    a. 'INSERT INTO flight VALUES(airlineName, flightNum, departureTime, departureAirport, arrivalTime, arrivalAirport, basePrice, status, planeID)', insert flight and its values into table
3. Change status of flights:
    a. 'UPDATE flight SET status = status WHERE AirlineName = airlineName AND flightNum = flightNum', change the status of the associated flight.
4. Add airplane in the system:
    a. 'INSERT INTO airplane VALUES(planeID, airlineName, numSeats, manufacturer, modelNum, manufactureDate, age)', add new airplane to table
5. Add new airport in the system:
    a. ‘INSERT INTO airport VALUES(airportCode, airportName, airportCity, airportCountry, numTerminals, type)', add new airport to table
6. View flight ratings:
    a. Note: A button
    b. 'SELECT * FROM rating as r, flight as f WHERE r.FlightNum = f.flightNum AND AirlineName = airlineName', select all of the ratings for flights that are from the same airline that the staff is working for
7. Schedule maintenance:

    a. 'INSERT INTO maintenance VALUES(planeID, maintenanceID, startTime, endTime)', insert maintenance and values into table
8. View most frequent customer:
    a. 'SELECT * FROM customer as c, ticket as t, purchase as p, flight as f WHERE c.email = p.email AND p.TicketID = t.ticketID AND t.FlightNum = f.FlightNum AND f.AirlineName = airlineName GROUP BY c.email ORDER BY COUNT(*) DESC LIMIT 1', to obtain the customer that has bought the most tickets from the airline that the staff is working at
    b. ‘SELECT * FROM flight as f, ticket as t, purchase as p WHERE p.email = email AND p.TicketID = t.TicketID AND t.FlightNum = f.FlightNum', to obtain all of the flights that the most frequent customer ahs taken
9. View earned revenue:
    a. 'SELECT SUM(calculatedPrice) FROM ticket as t, purchase as p, flight as f WHERE t.ticketID = p.TicketID AND t.FlightNum = f.FlightNum AND f.AirlineName = airlineName AND p.purchaseTime BETWEEN DATE_SUB(NOW(), INTERVAL 30 DAY) AND NOW()', obtain the sum of the price of all tickets that the airline has sold in the last 30 days
    b. 'SELECT SUM(calculatedPrice) FROM ticket as t, purchase as p, flight as f WHERE t.ticketID = p.TicketID AND t.FlightNum = f.FlightNum AND f.AirlineName = airlineName AND p.purchaseTime BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND NOW()', same query except for the last year, instead of the last 30 days
10. Logout
    a. No SQL query heres