CS3038: Project Writeup (Use Cases)

Group Members: Kai Shinozaki-Conefrey & Anna Tanaka Viertler

Index

No Login Use Cases

Use Case	Query	Description
View Public Info	`SELECT name, num, depTime, arrTime, base_price, status FROM lookUpFlight WHERE departureAirport = (departureAirport) AND arrivalAirport = (arrivalAirport) AND depDate = (departureDate)`	Select a list of flights from a specified airport and departure date, heading to a specified airport.
Register		
1. Customer	`query = 'INSERT INTO customer VALUES (username, password, firstName, lastName, buildingNum, street, apartmentNumber, city, state, zipCode, passportNumber, passportExpirationDate, passportCountry, DOB)`	Insert customer information into the customer table on register.

2. Staff	`'SELECT * FROM airline WHERE airline_name = (airline_name)'`
	
	`SELECT * FROM airline_staff WHERE username = (username)'`
	
	`'INSERT INTO airline_staff VALUES ((username), (airline_name), (password), (first_name), (last_name), (DOB))'`
		Check to see if that airline is registered in our system.
		
		See if that username is already taken.
		
		If all parameters look good, insert them into the table (note that passwords are hashed with SHA256 before being inserted)

Login		
3. Customer	`SELECT * FROM customer WHERE customer_email = email and password = password`	Check if the customer's username and password exists in the database.
4. Staff	`'SELECT * FROM airline_staff WHERE username = (username) and password = (password)'`	Matching a username & password with something in the database indicates it exists

Customer Use Cases

Use Case	Queries	Description
View My Flights	`SELECT name, num, depTime, arrTime, ticket_id, status, departureAirport, arrivalAirport FROM lookUpTicket WHERE customer_email = (email) AND depTime > CURRENT_TIMESTAMP() ORDER BY depTime;` `SELECT name, num, depTime, arrTime, ticket_id, status, departureAirport, arrivalAirport FROM lookUpTicket WHERE customer_email = (email) AND depTime < CURRENT_TIMESTAMP() ORDER BY depTime DESC;`	Find all flights associated with the tickets the customer has bought by checking their email. The first query checks for flights before the current day, and the second query checks for flights after.

Search for Flights	`SELECT name, num, depTime, arrTime, base_price, status FROM lookUpFlight WHERE departureAirport = (departureAirport) AND arrivalAirport = (arrivalAirport) AND depDate = (departureDate)`
	
	`'SELECT name, num, depTime, arrTime, base_price, status FROM lookUpFlight WHERE departureAirport = (arrivalAirport) AND arrivalAirport = (departureAirport) AND depDate = (returnDate)`
		Select a list of flights from a specified airport and departure date, heading to a specified airport.
		
		If a return date was specified, run a variation of the query to reverse the arrival and departure airports and look for a flight on the return date.

Purchase Tickets	`SELECT COUNT(ticket_id) FROM ticket WHERE flight_num = (flightNum) AND airline_name = (airlineName) AND departure_date_Time = (departureDateTime);`
	
	`SELECT num_seats FROM airplane, flight WHERE flight_num = (flightNum) AND flight.airline_name = (airlineName) AND departure_date_Time = (departureDateTime) AND airplane.airplane_id = flight.airplane_id`
	
	`INSERT INTO ticket_purchase VALUES`
		Find the number of tickets that have already been sold for this flight.
		
		Check how many seats are available on the flight
		
		Insert customer information into

	(ticketID, email, datetime.now(), cardType, cardNumber, cardFirstName, cardLastName, cardExpirationDate, ticketSellPrice) 'INSERT INTO ticket VALUES (ticketID, flightNum, airlineName, departureDateTime, email, firstName, lastName, DOB)	ticket_purchase table Insert customer information into ticket table..
Cancel Trip	DELETE FROM ticket WHERE ticket_id = (ticketID)	Delete ticket associated with ticket_id in order to allow a user to cancel their ticket. This doesn't delete the purchase. No refunds.
Give Rating & Comments	INSERT INTO customer_review VALUES (flightNum, departureDateTime, airlineName, email, reviewScore, reviewComment)	Insert the customer's review comment and score into the review table
Track My Spending	SELECT purchase_date_time, sold_price FROM ticket_purchase WHERE customer_email = email AND purchase_date_time between beginDate and endDate;	Select entries from ticket_purchase associated with the customer's email, and purchased between two specified dates.
Logout	session.clear()	Erase all session variables.

Staff Use Cases

Use Case	Query	Description
View Flights	`SELECT name, num, depTime, arrTime, status, departureAirport, arrivalAirport FROM lookUpFlight WHERE name = airline_name AND depTime BETWEEN CURRENT_TIMESTAMP() AND DATE_ADD(CURRENT_TIMESTAMP(), INTERVAL 30 DAY) ORDER BY depTime;`	Select information from flights associated with the admin's airline, from the present up to 30 days in the future.
	`SELECT name, num, depTime, arrTime, departureAirport, arrivalAirport FROM lookUpFlight WHERE name = airline_name AND depTime <= CURRENT_TIMESTAMP() ORDER BY depTime DESC;'`	Select information from flights associated with the admin's airline that occurred in the past.
Create New Flights	`'SELECT * FROM flight WHERE flight_num = (flight_num) AND departure_date_time = (departure_date) AND airline_name = (airline_name)'`	Finding a unique flight ID within an airline
	`'SELECT * FROM airport WHERE code = (airport_code)'`	Finding codes for airports to enforce domestic-only or international-only airports can only host those kinds of flights
	`'SELECT * FROM maintenance WHERE airplane_id = (airplane_id) AND airline_name = (airline_name) AND (start_date/end_date) BETWEEN start_date and end_date'`	Ensuring that a flight doesn't overlap with any scheduled maintenances
	`'SELECT * FROM flight_arrival NATURAL JOIN flight WHERE airplane_id = (airplane_id) AND airline_name = (airline_name) AND flight_arrival.departure_date_time BETWEEN (departure_date) and (arrival_date)'`	Ensuring flights don't overlap.
	`INSERT INTO flight VALUES ((flight_num), (departure_date), (airline_name), (airplane_id), (base_price), "On-Time")'`	If everything's fine, then we'll add an entry to the flights table
	`'INSERT INTO flight_arrival VALUES ((arrival_airport), (flight_num),`	Also add an entry to flight_arrival table

	`(departure_date), (arrival_date), (airline_name))'`
	
	`'INSERT INTO flight_departure VALUES ((departure_airport), (flight_num), (departure_date), (airline_name))'`
		Also add an entry to flight_departure table

Change Status of Flights	`'UPDATE flight SET status = (status) WHERE flight_num = (flight_num) AND departure_date_time = (departure_date) AND airline_name = (airline_name)'`	Update a flight's current status based on flight number, departure date, and airline

Add Airplane	`'SELECT * FROM airplane WHERE airplane_id = (airplane_id) AND airline_name = (airline_name)'`
	
	`'INSERT INTO airplane VALUES ((airplane_id), (airline_name), (num_seats), (manufacturing_company), (model_num), (manufacturing_date), 0)'`
	
	`'UPDATE airplane SET age = (DATEDIFF(NOW(), manufacturing_date)) / 365 WHERE airline_name = (airline_name) AND airplane_id = (airplane_id)'`
		Ensure each airplane has a unique ID within the same airlines
		
		Insert the new airplane into the table
		
		Update its age based on the manufacturing date

Add New Airport	`'SELECT * FROM airport WHERE code = (code)'`
	
	`'INSERT INTO airport VALUES ((code), (airport_name), (city), (country), (num_terminals), (type))'`
		Check if the airport code already exists (meaning the airport already exists)
		
		If everything else looks good, we'll add our new airport to our airport table

View Flight Ratings	`'SELECT customer_email, rating, comment FROM customer_review NATURAL JOIN flight WHERE departure_date_time = (departure_date) AND flight_num = (flight_id) AND airline_name = (airline_name)'`
	
	`'SELECT round(avg(rating), 1) as average FROM customer_review WHERE flight_num = (flight_id) AND departure_date_time = (departure_date) AND airline_name = (airline_name(;'`
		Find all reviews from a particular flight based on departure date, flight ID, and airline name
		
		Get average rating from all the flight's reviews based on flight ID, departure date, and airline name

Schedule Maintenance	`'SELECT * FROM airplane WHERE airplane_id = (airplane_id)'`
	
	`SELECT * FROM maintenance WHERE airplane_id = (airplane_id) AND airline_name = (airline_name) AND ((start_date BETWEEN (maintenance_start) AND (maintenance_end)) OR (end_date BETWEEN (maintenance_start) AND (maintenance_end)))`
	
	`'SELECT * FROM ticket WHERE ticket_id = (ticket_id)'`
	
	`'INSERT INTO maintenance VALUES ((maintenance_id), (airplane_id), (airline_name), (maintenance_start), (maintenance_end))'`
		Check to see if the plane we're trying to schedule maintenance for actually exists
		
		Check if there's already maintenance during the given times.
		
		This query is repeated until we have a unique ticket ID (the ticket ID is randomly generated)
		
		If everything looks good, we insert the maintenance into the system

View Frequent Fliers	`'SELECT * FROM ticket WHERE airline_name = (airline_name) AND flight_num = (flight_id) AND departure_date_time = (departure_date) ORDER BY ticket_id'`
	
	`'SELECT * FROM ticket WHERE customer_firstname = (customer_firstname) AND customer_lastname = (customer_lastname) AND airline_name = (airline_name)'`
		Get a tally and order all customers by how many tickets they're bought (and kept)
		
		See all flights a customer is in/has been in within your own airline based on their first and last name

View Earned Revenue	`SELECT SUM(sold_price) FROM ticket NATURAL JOIN ticket_purchase WHERE airline_name = airlineName AND purchase_date_time BETWEEN beginDate and endDate;`	Sum up the prices of tickets sold between two dates. This query is executed twice, the first time with a month between the two dates and the second time with a year in between.
Logout	`session.clear()`	Erase all session variables.