Team 53 Formal Specification: AirBC

Platform to Use
The CS Ugrad Oracle installation and provided PHP/Apache.

Classes of Users
- Public
    - Can create customer account
    - Can query for routes from/to an airport
    - Can query for flights on specific routes
    - Can query for flights that have available tickets from/to specific airport
    - Cannot book tickets
- Customer
    - Can update their account information
    - Can book/cancel tickets
    - Can query for routes from/to an airport
    - Can query for flights on specific routes
    - Can query for flights that have available tickets
    - Can query for their booked tickets
- Staff
    - Can do everything that a customer can do
    - Can create staff account
    - Can update staff account
    - Can book/cancel tickets for customers
    - Can do add/remove flight
    - Can do add/remove aircraft
    - Can change aircraft's status
    - Can add/remove routes
    - Can add/remove airports
    - Can generate customer report
    - Can generate a list of high value customers
    - Can query for loyalty members on a specific flight

Data in the Application
The database will include detailed information about the accounts (both customers' and staff's), aircrafts, tickets, flights, routes, and airports.

Division of Labour
Rhys will populate the database.
Alison will work on the business logic in PHP.
Harryson will work on the UI.
Mandy will work on the query generation in SQL.

<u>Application Specifications</u>

<u>Create Customer</u>

Public can create new customer account.

User: Public

Input: name, email, password

Optional input: payment information, seat preference, travel document, billing address

Output: customer account ID or "Failed to create account"

Basic Case:

Check that email is unique and password has length >= 6. If so, return customer account ID created.

Exceptions:

- If email is already used for an account, abort transaction and return “Failed to create account”.
- If password length is < 6, abort transaction and return “Failed to create account”.

<u>Update Customer</u>

Customer can update their customer account.

User: Customer

Input: customer account ID/email, password

Optional input: payment information, seat preference, travel document, billing address

Output: “Updated” or "Failed to update account"

Basic Case:

Check that customer account id/email exists and password matches with it. If so, return “Updated”.

Exceptions:

- Customer account ID/email does not exist, abort transaction and return “Failed to update account”.
- Password does not match customer id/email, abort transaction and return “Failed to update account”.

<u>Book Ticket</u>

Customers can book tickets.

User: Customer

Input: customer account ID, flight id, quantity

Optional input:

Output: ticket id or "Failed to book"

Basic Case:

Check that customer account ID exists, the payment information is valid (ie. valid credit card information), and the flight has available seats. If so, return ticket id.

Exceptions:

- If customer account ID does not exist, cancel transaction and return “Cancelled”
- If flight is fully booked, cancel transaction and return “Cancelled”
- If the payment information is invalid, cancel transaction and return “Cancelled”

Cancel Ticket

Customers can cancel booked tickets.

User: Customer

Input: ticket id, customer account ID

Optional input: quantity (that you want to remove)

Output: "Cancelled" or “Failed to cancel”

Basic Case:

Check that the ticket id exists and corresponds to a booking made by the customer, if so, refund the customer using the original payment method and return “Cancelled”.

Exceptions:

- Inexistent ticket id, abort transaction and return “Failed to cancel”
- Inexistent customer account ID, abort transaction and return “Failed to cancel”
- ticket id and customer account ID do not match, abort transaction and return “Failed to cancel”
- Quantity is larger than initial purchase, abort transaction and return “Failed to cancel”

Staff

Book Ticket

Staff can book tickets for customers.

User: Staff

Input: customer account ID, staff account ID, flight id, quantity

Optional input:

Output: ticket id or "Failed to book"

Basic Case:

Check that customer account ID and staff account ID exist, the payment information is valid (ie. valid credit card information), and the flight has available seats. If so, return ticket id.

Exceptions:

- If customer account ID does not exist, cancel transaction and return “Cancelled”
- If staff account ID does not exist, cancel transaction and return “Cancelled”
- If flight is fully booked, cancel transaction and return “Cancelled”
- If the payment information is invalid, cancel transaction and return “Cancelled”

Cancel Ticket

Staff can cancel booked tickets for customers.

User: Staff

Input: ticket id, customer account ID, staff account ID

Optional input: quantity (that you want to remove)

Output: "Cancelled" or “Failed to cancel”

Basic Case:

Check if staff account ID exists, the ticket id exists and corresponds to a booking made by the customer. If so, refund the customer using the original payment method and return “Cancelled”.

Exceptions:

- Inexistent staff account ID, abort transaction and return “Failed to cancel”
- Inexistent ticket id, abort transaction and return “Failed to cancel”
- Inexistent customer account ID, abort transaction and return “Failed to cancel”
- ticket id and customer account ID do not match, abort transaction and return “Failed to cancel”
- Quantity is larger than initial purchase, abort transaction and return “Failed to cancel”

<u>Create Staff</u>
Staff can create new staff account.
User: Staff
Input: name, email, password
Optional input: title
Output: account ID or "Failed to create account"
Basic Case:
Check that email is unique and password has length >= 6. If so, return account ID created.
Exceptions:

- If email is already used for an account, abort transaction and return “Failed to create account”.
- If password length is < 6, abort transaction and return “Failed to create account”.

<u>Update Staff</u>
Staff can update their staff account.
User: Staff
Input: staff ID/email, password
Optional input: name, title
Output: “Updated” or "Failed to update account"
Basic Case:
Check that customer id/email is exists and password matches with it. If so, return “Updated”.
Exceptions:

- Customer ID/email does not exist, abort transaction and return “Failed to update account”.
- Password does not match customer id/email, abort transaction and return “Failed to update account”.

<u>Add Flight</u>
Staff can add flights.
User: Staff
Input: staff account ID, flight ID, route, flight date and time, aircraft ID
Optional input:
Output: "Flight added" or "Failed to add"
Basic Case:
Check if departure and arrival airport will accept a new flight, and if AirBC has available aircraft for flight, and staff account id exists. If so, return “Flight added”.

Exceptions:

- If staff account ID does not exist, abort transaction and return “Failed to add flight”
- If aircraft already has a flight scheduled for the same time with the same aircraft return “Failed to add flight”
- If AirBC has no available aircraft, abort transaction and return “Failed to add flight”
- If departing airport does not have capacity for new flight, abort transaction and return “Failed to add flight”
- If arrival airport does not have capacity for new flight, abort transaction and return “Failed to add flight”

<u>Remove Flight</u>
Staff can remove flights.
User: Staff
Input: staff account ID, flight ID
Optional input: route, flight date and time, aircraft
Output: "Flight removed" or "Failed to remove flight"
Basic Case:
Check if flight ID is in database and staff account id exists. If it is, refund all tickets purchased by customers, remove flight from database, and return “Flight removed”.
Exceptions:

- If staff account ID does not exist, abort transaction and return “Failed to remove flight”
- If flight ID is not found in database, abort transaction and return “Failed to remove flight”

We think that this is an interesting update query because we need to use cascading deletes to ensure that there are no references to a flight which no longer exists.

<u>Add Aircraft</u>
Staff can add aircrafts.
User: Staff
Input: account ID, aircraft ID, type, first class seats, business class seats, economy seats, purchase date
Optional input: status
Output: "Aircraft added" or "Failed to add aircraft"
Basic Case:
If aircraft ID doesn’t exist in database and staff ID exists, add aircraft ID to database and return “Aircraft added”.
Exceptions:

- If staff account ID does not exist, cancel transaction and return “Failed to add aircraft”
- If aircraft ID already exists, cancel transaction and return “Failed to add aircraft”

<u>Remove Aircraft</u>
Staff can remove aircrafts.
User: Staff
Input: staff account ID, aircraft ID

Optional input: status
Output: "Aircraft removed" or "Failed to remove aircraft"
Basic Case:
If aircraft ID and staff account ID exist, remove aircraft, aircraft related information, associated flights from database and return “Aircraft removed”. Default status of “operational” can be changes with optional input.
Exceptions:

- If staff account ID does not exist, cancel transaction and return “Failed to remove aircraft”
- If aircraft ID does not exist, cancel transaction and return “Failed to remove aircraft”

<u>Change Aircraft Status</u>
Staff can change aircraft status between “in repair” and “operational”.
User: Staff
Input: account ID, aircraft ID, status to change to
Optional input:
Output: "Changed successfully" or "Failed to change"
Basic Case:
If staff ID exists, continue to check if current status is the same as desired status, return “Changed successfully” (no change); if current status != desired status, set it to desired status and return “Changed successfully”.
Exceptions:

- If staff account ID does not exist, cancel transaction and return “Failed to change”
- Aircraft ID does not exist, return “Failed to change”

<u>Add Route</u>
Staff can add routes.
User: Staff
Input: account ID, departure airport ID, arrival airport ID
Optional input:
Output: "Route added" or "Failed to add route"
Basic Case:
If staff ID exists, the departure airport ID and arrival airport ID exist, and route has not been added to database already (it doesn’t make sense to have two YVR - YYZ routes in database). If so, return “Route added”.
Exceptions:

- If staff account ID does not exist, cancel transaction and return “Failed to add route”
- If departure airport ID does not exist, cancel transaction and return “Failed to add route”
- If arrival airport ID does not exist, cancel transaction and return “Failed to add route”
- If route already exists in database, cancel transaction and return “Failed to add route”

<u>Remove Route</u>
Staff can remove routes.

User: Staff
Input: account ID, route ID
Optional input: Departure airport ID, Arrival airport ID
Output: “Route removed” or "Failed to remove route"
Basic Case:
If staff account id exists, the route ID, or the pairing of departure airport ID and arrival airport ID, exist in the database, remove all the associated flights and return “Route removed”.
Exceptions:
- If staff account ID does not exist, cancel transaction and return “Failed to remove route”
- The route does not exist in the database, cancel transaction and return “Failed to remove route”.

Add Airport
Staff can add airports.
User: Staff
Input: account ID, airport code, location
Optional input: city airport is located in
Output: “Airport added” or "Failed to add airport"
Basic Case:
Airport does not exist in the database, add airport to database and return “Airport added”.
Exceptions:
- If staff account ID does not exist, cancel transaction and return “Failed to add airport”
- If airport code exists in database, cancel transaction and return “Failed to add airport”.

Remove Airport
Staff can remove airports.
User: Staff
Input: account ID, airport code
Optional input:
Output: “Airport removed” or "Failed to remove airport"
Basic Case:
If airport code exists in the database, delete it from database and return “Airport removed”.
Exceptions:
- If staff account ID does not exist, cancel transaction and return “Failed to remove airport”
- Airport code does not exist in database, cancel transaction and return “Failed to remove airport”

Routes from/to Airport Query
Everyone can search for routes from/to a specific airport.
User: Public
Input: airport code, from/to option
Optional input:
Output: List of routes, “No routes”

Basic Case:
If airport code exists, return a list of routes from/to the specific airport based on user's from/to option.
Exceptions:
- Airport code does not exist in database, return "No routes".

<u>Get Flights with empty seats from/to Specific Airport</u>
Everyone can search for flights that have empty seats from/to a specific airport.
User: Public
Input: airport code, from/to option
Optional input: dates, only economy seats, only business seats
Output: List of flights, "No flights"
Basic Case:
If airport code exists, return a list of flights from/to the specific airport based on user's from/to option. Optionally, filter results within a given date range or seating preferences.
Exceptions:
- Airport code does not correspond to airport in database, return "No flights"

We think that this is a complex query because it uses an aggregation (count the number of tickets sold for each seat type) and compares that to the number of seats available on the aircraft.

<u>Flights on a specific route</u>
Everyone can search for flights on a specific route.
User: Public
Input: airport code from, airport code to
Optional input: dates, only economy seats, only business seats
Output: List of flights, "No flights"
Basic Case:
- A user will place a request for all flights on a specific airport. Optionally, filter results within a given date range or seating preferences.

Exceptions:
- Route does not correspond to airport in database.

<u>Generate Customer Report</u>
Staff can generate a report for a customer or a customer can generate for themselves which includes the tickets purchased between specified dates.
User: Customer, Staff
Input: Customer account ID, from date, to date
Optional input:
Output: Report(a list of tickets purchased and revenue from customer), "Failed to generate report"
Basic Case:

If customer account ID and staff account ID exist, from date is earlier than to date, return a report of a list the tickets purchased by the customer within given dates and calculated revenue from the customer for the given period.
Exceptions:

- If customer account ID does not exist, return “Failed to generate report”.
- If staff account ID does not exist, return “Failed to generate report”.
- If from date is later than to date, return “Failed to generate report”

<u>Generate List of High Value Customers</u>
Staff can generate a list of customers who have spent a large amount on the airline.
User: Staff
Input: Purchased revenue
Optional input: from date, to date
Output: List of customers “Failed to find customers”
Basic Case:
If staff account ID exists, return a list of high value customers who have spent more than a specified revenue amount on the airline. Optionally, restrict ticket sales to a specified range.
Exceptions:

- If purchased revenue is a negative number, return “Failed to find customers”
- If staff account ID does not exist, return “Failed to find customers”
- If from date is later than to date, return “Failed to find customers”

We think that this is an interesting select query as it uses aggregate functions.

<u>Get loyalty members on specific flight</u>
Staff can find all loyalty members on a flight.
User: Staff
Input: Staff account ID, flight ID
Optional input:
Output: List of customers, “Failed to find loyalty members onboard”
Basic Case:
If staff account ID and flight ID exist, return list of loyalty members on a flight.
Exceptions:

- If flight ID does not exist, return “Failed to find loyalty members onboard”
- If staff account ID does not exist, return “Failed to find loyalty members onboard”

We think that this is an interesting select query as it needs to check if a customer is a loyalty member or not.