User Cases

Frontend: Zihao Ren Backend: Zhanghao Chen
December 2018

General User Cases **FINISHED**

1. View Public Info

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Additional information:

- Consistency of date will be checked (arrival_date < departure_date is not allowed)

Search for Upcoming Flights

Action:

upcoming

Provide:

- departure_date
- arrival_date
- departure_airport
- arrival_airport

Return:

- result_upcoming
- message_upcoming

SQL query:

```
SELECT *
FROM flight
WHERE departure_airport = { departure_airport } AND
    DATE(departure_time) = { departure_date} AND
    arrival_airport = { arrival_airport } AND
    DATE(arrival_time) = { arrival_date }
```

Check Flight Status

Action:

status

Provide:

- flight_num
- departure_date
- arrival_date

Return:

- result_status
- message_status

SQL query:

SELECT *
FROM flight
WHERE flight_num = { flight_num } AND DATE(departure_time) = { departure_date}
AND DATE(arrival_time) = { arrival_date }

2. Register

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

registerAuth

Return:

- error (a string of error information)

Additional Information:

- The password is hashed with passlib.hash.pbkdf2_sha256 password hashing schemes, and the hashed value is stored in the database.

Customer

Provide:

- name
- email
- password
- building_number
- street
- city
- state
- phone_number
- passport_number
- passport_expiration
- passport_country
- date_of_birth

SQL query:

- Check whether the email already exists:
  SELECT * FROM customer WHERE email = { email }
- Insert into database
  INSERT INTO customer VALUES({ name }, { email }, { password }, { building_number },
  { street }, { city }, { state }, { phone_number }, { passport_number },
  { passport_expiration }, { passport_country }, { date_of_birth })

Booking Agent

Provide:

- email
- password
- booking_agent _id

SQL query:

- Check whether the email already exists:
  SELECT * FROM booking_agent WHERE email = { email }
- Insert into database
  INSERT INTO booking_agent VALUES({ email }, { password }, { booking_agent_id })

Airline Staff

Provide:

- username
- password
- first_name
- last_name
- date_of_birth
- airline_name

SQL query:

- Check whether the email already exists:
  SELECT * FROM airline_staff WHERE username = { username }
- Insert into database
  INSERT INTO airline_staff VALUES({ username }, { password }, { first_name },
  { last_name }, { date_of_birth }, { airline_name })

3. Login

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

loginAuth

Provide:

- usertype
- username
- password

Returns:

- error (a string of error information)

SQL query:

- SELECT * FROM { usertype } WHERE { username } = { password }

Additional information:

- Username and usertype will be recorded in the session. In addition, booking_agent_id will be recorded in the session for booking agent, same for airline_name for airline staff
- The password is verified with passlib.hash.pbkdf2_sha256 password hashing schemes

Customer User Cases **FINISHED**

1. View My Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Return:
- result_viewMyFlights
- message_viewMyFlights

Default View

Action:

viewMyFlights

Provide:
- customer_email (stored in session)
- current_time (got from MySQL server)

SQL query:

```
SELECT *
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE customer_email = { customer_email } AND departure_time > NOW()
ORDER BY departure_time
```

Option View (by specifying a range of departure date)

Action:

viewMyFlightsOption

Provide:
- customer_email (stored in session)
- start_date
- end_date

Additional information:
- Consistency of date will be checked (end_date < start_date is not allowed)

SQL query:

```
SELECT *
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE customer_email = { customer_email } AND
        DATE(departure_time) BETWEEN { start_date } AND { end_date }
```

2. Purchase Tickets

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

purchaseTickets

Provide:

- customer_email (stored in session)
- airline_name
- flight_num
- purchase_date (got from MySQL server)

Return:

- message_purchaseTickets

SQL query:

- Check seat availability (seat available only if current count_ticket < seats)
  SELECT COUNT(*) as count, seats
  FROM ticket NATURAL JOIN flight NATURAL JOIN airplane
  WHERE airline_name = { airline_name } AND flight_num = { flight_num }
  GROUP BY airline_name, flight_num
- Generates ticket id (total number of tickets in the current system + 1)
  ticket_id = { SELECT COUNT(*) FROM ticket } + 1
- Insert into the database
  INSERT INTO ticket VALUES({ ticket_id }, { airline_name }, { flight_num })
  INSERT INTO purchases VALUES({ ticket_id }, { customer_email }, NULL, CURDATE())

3. Search for Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

searchFlights

Provide:

- departure_date
- arrival_date
- departure_airport
- arrival_airport

Return:

- result_searchFlights
- message_searchFlights

Additional information:

- Consistency of date will be checked (arrival_date < departure_date is not allowed)

SQL query:

SELECT *

FROM flight
WHERE departure_airport = { departure_airport } AND
DATE(departure_time) = { departure_time } AND
arrival_airport = { arrival_airport } AND
DATE(arrival_time) = { arrival_time }

4. Track My Spending

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

SQL query: applied multiple times to generate the data for the bar chart

SELECT SUM(price) as total
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE customer_email = { customer email } AND purchase_date >= { start_date } AND
purchase_date < { end_date }

Default View:

Action:

trackMySpendingDefault

Provide:

- customer_email (stored in session)

Return:

- total
- monthwise_label
- monthwise_total

Optional View

Action:

trackMySpendingOptional

Provide:

- customer_email (stored in session)
- start_month
- end_month

Return:

- total_option
- monthwise_label_option
- monthwise_total_option

Additional information:

Consistency of date will be checked (start_month > end_month is not allowed)

5. Logout

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

logout

Booking Agent User Cases **FINISHED**

1. View My Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Return:

- result_viewMyFlights
- message_viewMyFlights

Default View

Action:

viewMyFlights

Provide:

- booking_agent_email (stored in session)
- current_time (got from MySQL server)

SQL query:

```
SELECT *
FROM booking_agent NATURAL JOIN purchases NATURAL JOIN
        ticket NATURAL JOIN flight
WHERE email = { booking_agent_email } AND departure_time > NOW()
ORDER BY departure_time
```

Option View (by specifying a range of departure date)

Action:

viewMyFlightsOption

Provide:

- booking_agent_email (stored in session)
- start_date
- end_date

Additional information:

- Consistency of date will be checked (end_date < start_date is not allowed)

SQL query:

```
SELECT *
FROM booking_agent NATURAL JOIN purchases NATURAL JOIN
        ticket NATURAL JOIN flight
WHERE email = { booking_agent_email } AND
        DATE(departure_time) BETWEEN { start_date } AND { end_date }
```

2. Purchase Tickets

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

purchaseTickets

Provide:

- booking_agent_id (stored in session)
- customer_email
- airline_name
- flight_num
- purchase_date (got from MySQL server)

Return:

- message_purchaseTickets

SQL query:

- Check seat availability (seat available only if current count_ticket < seats)

  ```
  SELECT COUNT(*) as count, seats
  FROM ticket NATURAL JOIN flight NATURAL JOIN airplane
  WHERE airline_name = { airline_name } AND flight_num = { flight_num }
  GROUP BY airline_name, flight_num
  ```

- Generates ticket id (total number of tickets in the current system + 1)

  ```
  ticket_id = { SELECT COUNT(*) FROM ticket } + 1
  ```

- Insert into the database

  ```
  INSERT INTO ticket VALUES({ ticket_id }, { airline_name }, { flight_num })
  INSERT INTO purchases VALUES({ ticket_id }, { customer_email },
          { booking_agent_id }, CURDATE())
  ```

3. Search for Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

searchFlights

Provide:

- departure_date
- arrival_date

- departure_airport
- arrival_airport

Return:

- result_searchFlights
- message_searchFlights

Additional information:

- Consistency of date will be checked (arrival_date < departure_date is not allowed)

SQL query:

```
SELECT *
FROM flight
WHERE departure_airport = { departure_airport } AND
    DATE(departure_time) = { departure_time } AND
    arrival_airport = { arrival_airport } AND
    DATE(arrival_time) = { arrival_time }
```

4. View My Commission

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Default View:

Action:

- commission_default

Provide:

- booking_agent_id (stored in session)

Return:

- result_commission_default
- message_commission

SQL query:

```
SELECT
    SUM(price) * 0.1 as sum_commission,
    AVG(price) * 0.1 as avg_commission,
    COUNT(*) as num_tickets
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE booking_agent_id = { booking_agent_id } AND
    purchase_date >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
```

Option View:

Action:

- commssion_option

Provide:

- booking_agent_id (stored in session)

- start_date
- end_date

Return:

- result_commission_option
- message_commission

Additional information:

- Consistency of date will be checked (end_date < start_date is not allowed)

SQL query:

```
SELECT
        SUM(price) * 0.1 as sum_commission,
        AVG(price) * 0.1 as avg_commission,
        COUNT(*) as num_tickets
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE booking_agent_id = { booking_agent_id } AND
        purchase_date BETWEEN { start_date } AND { end_date }
```

5. View Top Customers

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

viewTopCustomers

Provide:

- booking_agent_id (stored in session)

Return:

- top5_by_count (customer_email, count)
- top5_by_commission (customer_email, commission)
- message_viewTopCustomers

SQL query:

- Top 5 by number of tickets in the last 6 months

```
SELECT customer_email, COUNT(*) as count
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE booking_agent_id = { booking_agent_id } AND
        purchase_date >= DATE_SUB(NOW(), INTERVAL 6 MONTH)
GROUP BY customer_email
ORDER BY count DESC
LIMIT 5
```

- Top 5 by number of commission in the last year

```
SELECT customer_email, SUM(price) * 0.1 as commission
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
WHERE booking_agent_id = { booking_agent_id } AND
        purchase_date >= DATE_SUB(NOW(), INTERVAL 1 YEAR)
GROUP BY customer_email
```

ORDER BY commission DESC
LIMIT 5

6. Logout

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:
logout

Airline Staff User Cases **FINISHED**

1. View My Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Default View (flights of my airline in the next 30 days)

Action:
viewMyFlights
Provide:
- airline_name (stored in session)

Returns:
- result_viewMyFlights
- message

SQL query:

```
SELECT *
FROM flight
WHERE airline_name = %s AND
    departure_time BETWEEN CURDATE() AND
    DATE_ADD(NOW(), INTERVAL 30 DAY)
ORDER BY departure_time
```

Option View (specifying departure/arrival airport and a range of departure date)

Action:
viewMyFlightsOption
Provide:
- airline_name (stored in session)
- departure_airport

- arrival_airpot
- start_date
- end_date

Returns:

- result_viewMyFlights
- message

Additional information:

Consistency of date will be checked (start_date > end_date is not allowed)

SQL query:

```
SELECT *
FROM flight
WHERE airline_name = { airline_name }  AND departure_airport = { departure_airport }
        AND arrival_airport = { arrival_airport } AND
        departure_time BETWEEN { start_date } AND { end_date }
ORDER BY departure_time DESC
```

Sub-action (view all customers of the selected flight)

Action:

viewAllCustomers

Provide:

- airline_name (stored in session)
- flight_num (self-provided by the form)

Returns:

- airline_name
- flight_num
- result_viewAllCustomers
- message

SQL query:

```
SELECT ticket_id, customer_email, booking_agent_id, purchase_date
FROM ticket NATURAL JOIN purchases
WHERE airline_name = { airline_name } AND flight_num = { flight_num }
ORDER by purchase_date DESC
```

2. Create New Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

createNewFlights

Provide:

- airline_name (stored in session)
- flight_num

- departure_airport
- departure_time
- arrival_airport
- arrival_time
- price
- status
- airplane_id

Return:

- result (string of system info)

Additional information:

- Operation status will be checked with MySQL error provided

SQL query:

```
INSERT INTO flight VALUES({ airline_name }, { flight_num }, { departure_airport },
    { departure_time }, { arrival_airport }, { arrival_time}, { price }, { status },
    { airplane_id })
```

3. Change Status of Flights

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

changeFlightStatus

Provide:

- airline_name (stored in session)
- flight_num
- status

Return:

- result (string of system info)

Additional information:

- Operation status will be checked with MySQL error provided

SQL query:

```
UPDATE flight
SET status = { status }
WHERE airline_name = { airline_name } AND flight_num = { flight_num }
```

4. Add New Airplane in the System

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

addNewAirplane

Provide:

- airline_name (stored in session)
- airplane_id
- seats

Return:

- result (string of system info)

Additional information:

- Operation status will be checked with MySQL error provided

SQL query:

INSERT INTO airplane VALUES({ airline_name }, { airplane_id }, { seats })

5. Add New Airport in the System

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

addNewAirport

Provide:

- airport_name
- airport_city

Return:

- result (string of system info)

Additional information:

- Operation status will be checked with MySQL error provided

SQL query:

INSERT INTO airport VALUES({ airport_name }, { airport_city })

6. View Top5 Booking Agents

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

viewTop5BookingAgent

Provide:

- airline_name (stored in session)

Return:

- top5bycount_past_month
- top5bycount_past_year
- top5bycommission_past_year
- message_viewTop5BookingAgent

SQL query:

- Top5 by ticket count in the past month
  SELECT booking_agent_id, COUNT(ticket_id) as count
  FROM ticket NATURAL JOIN purchases

WHERE airline_name = { airline_name } AND booking_agent_id IS NOT NULL AND
purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 MONTH) AND
CURDATE()
GROUP BY booking_agent_id
ORDER by count DESC
LIMIT 5

- Top5 by ticket count in the past year
  SELECT booking_agent_id, COUNT(ticket_id) as count
  FROM ticket NATURAL JOIN purchases
  WHERE airline_name = { airline_name } AND booking_agent_id IS NOT NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND
  CURDATE()
  GROUP BY booking_agent_id
  ORDER by count DESC
  LIMIT 5
- Top5 by commission in the past year
  SELECT booking_agent_id, SUM(price) * 0.1 as commission
  FROM ticket NATURAL JOIN purchases NATURAL JOIN flight
  WHERE airline_name = { airline_name } AND booking_agent_id IS NOT NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND
  CURDATE()
  GROUP BY booking_agent_id
  ORDER by commission DESC
  LIMIT 5

7. View Frequent Customers

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Main Action

Action:
viewFrequentCustomers
Provide:

- airline_name (stored in session)

Return:

- result_viewFrequentCustomers
- message_viewFrequentCustomers

SQL_query:
SELECT customer_email, COUNT(ticket_id) AS count
FROM ticket NATURAL JOIN purchases
WHERE airline_name = { airline_name } AND
purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND

CURDATE()
GROUP BY customer_email
ORDER by count DESC

Sub-action (View all the flights under the airline taken by a customer)

Action:
viewFlightsTaken
Provide:
- airline_name (stored in session)
- Customer_email (self-provided by the form)

Return:
- result_viewFlightsTaken (customer_email, flight_num, purchase_date)

SQL_query:
```
SELECT customer_email, flight_num, purchase_date
FROM ticket NATURAL JOIN purchases
WHERE airline_name = { airline_name } AND
    customer_name = { customer_name )
ORDER by purchase_date DESC
```

8. View Reports

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

SQL query: applied multiple times to generate the data for the bar chart
```
SELECT COUNT(ticket_id) as total
FROM purchases NATURAL JOIN ticket
WHERE airline_name = { airline_name } AND purchase_date >= { start_date } AND
    purchase_date < { end_date }
```

Action:
viewReports
Provide:
- airline_name (stored in session)
- start_month
- end_month

Return:
- total
- monthwise_label
- monthwise_total

Additional information:
Consistency of date will be checked (start_month > end_month is not allowed)

9. Comparison of Revenue Earned

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

compareRevenue

Provide:

- airline_name (stored in session)

Return:

- revenue_direct_sale_last_month
- revenue_indirect_sale_last_month
- revenue_direct_sale_last_year
- revenue_indirect_sale_last_year
- message

SQL query:

- Revenue form direct sales in the last month
  SELECT SUM(price) as revenue
  FROM flight NATURAL JOIN ticket NATURAL JOIN purchases
  WHERE airline_name = { airline_name } AND booking_agent_id IS NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 MONTH) AND CURDATE()
- Revenue form indirect sales in the last month
  SELECT SUM(price) as revenue
  FROM flight NATURAL JOIN ticket NATURAL JOIN purchases
  WHERE airline_name = { airline_name } AND booking_agent_id IS NOT NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 MONTH) AND CURDATE()
- Revenue form direct sales in the last year
  SELECT SUM(price) as revenue
  FROM flight NATURAL JOIN ticket NATURAL JOIN purchases
  WHERE airline_name = { airline_name } AND booking_agent_id IS NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND CURDATE()
- Revenue form indirect sales in the last year
  SELECT SUM(price) as revenue
  FROM flight NATURAL JOIN ticket NATURAL JOIN purchases
  WHERE airline_name = { airline_name } AND booking_agent_id IS NOT NULL AND
  purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND CURDATE()

10. View Top3 Destinations

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

viewTop3Destitnations

Provide:

- airline_name (stored in session)

Return:

- top3_past3month
- top3_past1year
- message_viewTop3Destitnations

SQL_query:

- Top3 in the past 3 months

```
SELECT arrival_airport, airport_city, COUNT(ticket_id) as count
FROM flight NATURAL JOIN ticket NATURAL JOIN purchases, airport
WHERE airline_name = { airline_name } AND arrival_airport = airport_name AND
        purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 3 MONTH) AND
        CURDATE()
GROUP BY arrival_airport
ORDER BY count DESC
LIMIT 3
```

- Top3 in the past 1 year

```
SELECT arrival_airport, airport_city, COUNT(ticket_id) as count
FROM flight NATURAL JOIN ticket NATURAL JOIN purchases, airport
WHERE airline_name = { airline_name } AND arrival_airport = airport_name AND
        purchase_date BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND
        CURDATE()
GROUP BY arrival_airport
ORDER BY count DESC
LIMIT 3
```

11. Logout

Backend: **FINISHED** Frontend: **FINISHED** Review: **FINISHED**

Action:

logout