Use cases & Queries

Net ID: bk2930

Name: Bomin Kim

Home page when not logged-in:

1. View Public Info

a. search based on dates, source/detination airport

if they do not input return date, it automatically shows only outbound flights.

```
@app.route('/search',methods=['GET','POST'])
def search():
    # grabs information from the forms
    dep_airport = request.form.get('dep_airport')
    arr_airport = request.form.get('arr_airport')
    dep_date = request.form.get('dep_date')
    return_date = request.form.get('return_date',None)
    print(return_date)

    cursor = conn.cursor()
    query1 = '''
            SELECT Airline_name, flight.ID as flight_num, dep_date, dep_time, arr_date,arr_time
            FROM flight, airplane
            WHERE Airplane_ID = airplane.ID
            AND dep_airport = %s
            AND arr_airport = %s
            AND dep_date = %s;'''
    cursor.execute(query1, (dep_airport, arr_airport, dep_date))
    data1 = cursor.fetchall()
```

Else if they input return date, there will be two tables shown.

```
if return_date and return_date != '':
    query2 = '''
    SELECT  flight.ID as flight_num, dep_date, dep_time, arr_date, arr_time, Airline_name
    FROM flight, airplane
    WHERE Airplane_ID = airplane.ID
    AND dep_airport = %s
    AND arr_airport = %s
    AND dep_date = %s;'''
    cursor.execute(query2, (arr_airport, dep_airport , return_date))
    data2 = cursor.fetchall()
    cursor.close()
    return render_template('index.html', dep_airport=dep_airport, arr_airport=arr_airport, dep_da
                           return_date=return_date, flights=data1, flights2=data2, username=session[
```

b. Will be able to see the flights status based on airline name, flight number, arrival/departure date.

```
query = '''
        SELECT Airline_name, flight.ID as flight_num, dep_date, dep_time, arr_date,arr_time, sta
        FROM flight, airplane
        WHERE Airplane_ID = airplane.ID
        AND dep_airport = %s
        AND arr_airport = %s
        AND dep_date = %s;'''
```

They also can check flight status(stored in 'flight' table), searching by dates and airports.

2. Register

1) customer

```
query = 'SELECT * FROM customer WHERE email = %s'
cursor.execute(query, (email))
#stores the results in a variable
data = cursor.fetchone()
#use fetchall() if you are expecting more than 1 data row
error = None
if(data):
    #If the previous query returns data, then user exists
    error = "This user already exists"
    return render_template('register_customer.html', error = error)
else:
    ins = 'INSERT INTO customer VALUES(%s, %s, %s, %s, %s)'
    cursor.execute(ins, (fname, lname, email, password, dob))
    conn.commit()
    cursor.close()
    return render_template('index.html')
```

If they register as customer, query checks if there is the same email used already. If there is not same data, then insert registration information into 'customer' table .

2) staff

```
query = 'SELECT * FROM airlinestaff WHERE username = %s'
cursor.execute(query, (username))
#stores the results in a variable
data = cursor.fetchone()
#use fetchall() if you are expecting more than 1 data row
error = None
if(data):
    #If the previous query returns data, then user exists
    error = "This user already exists"
    return render_template('register_staff.html', error = error)
else:
    ins = 'INSERT INTO airlinestaff VALUES(%s, %s, %s)'
    cursor.execute(ins, (username, password, airline))
```

3. Login

```
cursor.execute("SELECT * FROM customer WHERE email=%s AND password=%s", (username, password))
customer = cursor.fetchone()

if customer:
    session['username'] = username
    return redirect('/customer_dashboard')  # 고객용 대시보드로 리다이렉트
else:
    # 직원 테이블에서 사용자명과 비밀번호 확인
    cursor.execute("SELECT * FROM airlinestaff WHERE username=%s AND password=%s", (username, password))
    staff = cursor.fetchone()
```

If table 'customers' has corresponding data with email and password, it redirect pages to customer page. Else if table 'airlinestaff' has one, it leads logging in as staff and redirect to staff page.

Customer Use Cases

1. View My flights

```
today = datetime.today().strftime('%Y-%m-%d')
future_date = (datetime.today() + timedelta(days = 180)).strftime('%Y-%m-%d')

start_date = request.form.get('start_date',today)
end_date = request.form.get('end_date',future_date)

cursor = conn.cursor()
query = '''
    SELECT  flight.ID as flight_num, dep_date, dep_time, arr_date, arr_time, status
    FROM flight INNER JOIN ticket
    where customer_email = %s
    and ticket.flight_id = flight.id
    and dep_date between %s and %s;

'''
cursor.execute(query, (session.get('username'),start_date, end_date))
```

They can view all the tickets they've booked and specify a date range to search for their tickets. By default, I've set the start date as today and the end date as 180 days (6 months) ahead from today, allowing for an initial display of these flights for basic reference.

2. Search for flights

```
query1 = '''
        SELECT  flight.ID as flight_num, dep_date, dep_time, arr_date,arr_time, Airline_name
        FROM flight, airplane
        WHERE Airplane_ID = airplane.ID
        AND dep_airport = %s
        AND arr_airport = %s
        AND dep_date = %s;'''
cursor.execute(query1, (dep_airport, arr_airport, dep_date))
data1 = cursor.fetchall()


if return_date and return_date != '':
    query2 = '''
    SELECT  flight.ID as flight_num, dep_date, dep_time, arr_date, arr_time, Airline_name
    FROM flight, airplane
    WHERE Airplane_ID = airplane.ID
    AND dep_airport = %s
    AND arr_airport = %s
    AND dep_date = %s;'''
    cursor.execute(query2, (arr_airport, dep_airport , return_date))
    data2 = cursor.fetchall()
```

Same with the case for all users

3. Purchase tickets

```
cursor = conn.cursor()
ins = '''INSERT INTO ticket (flight_id, customer_email, fname, lname, dob, card_type, card_num, exp_date, purchase_date)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)'''

cursor.execute(ins, (flight_id, customer_email, fname, lname, dob, card_type, card_num, exp_date, purchase_date) )
```

As they checkout the ticket, it stores the purchase information into table 'ticket'.

4. Cancel Trip

```
query = """SELECT  ticket.id as ticket_id, flight.ID as flight_num, dep_date, dep_time, arr_date, arr_time, fname, lname
    FROM flight INNER JOIN ticket
    where customer_email = %s
    and ticket.flight_id = flight.id
    and TIMESTAMP(dep_date, dep_time) > NOW() + INTERVAL 24 HOUR;"""
```

On the cancel trip page, flight that will take place more than 24 hours will be shown. I used now() function and compared with the dep date and dep_time of the flights.

```
query = "delete from ticket where id = %s;"
```

When customer click cancel, the row is deleted from the table.

5.Logout

```
@app.route('/logout')
def index():
    session['username']=None
    return render_template('login.html')
```

Once they log out, the page moves to the login page.

Airline Staff use cases

1. View flights

```
cursor = conn.cursor()
query = '''
    SELECT  flight.ID as flight_num, dep_date, dep_time, arr_date, arr_time, status, Airline_name
    FROM flight
    INNER JOIN airplane ON flight.Airplane_ID = airplane.ID
    WHERE Airline_name IN (
        SELECT airline_name
        FROM airlinestaff
        WHERE username = %s and
        dep_date BETWEEN %s AND %s
    )
'''
params = (session.get('username'),start_date, end_date,)

if dep_airport and arr_airport :
    query += 'AND dep_airport = %s AND arr_airport = %s;'
```

It filters the results based on the user's associated airline and a specified departure date range. And then show the flights information.

```
query = """SELECT fname, lname, email FROM customer
            WHERE email IN ( SELECT Customer_email FROM ticket
                                WHERE flight_id = %s);"""
```

They can view all the customers' name and email address which bought the tickets for corresponding flight.

2. Create new flights

```
ins = 'INSERT INTO flight VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'
cursor.execute(ins, (flight_num,dep_airport,dep_date,dep_time,
                    arr_airport,arr_date,arr_time,base_price,status,airplane))
```

3. Change Status of flights

```
ins = '''
    UPDATE flight
    SET status =
        CASE
            WHEN status = 'delayed' THEN 'on-time'
            ELSE 'delayed'
        END
    WHERE ID = %s;'''
```

If the status of flight is 'delayed', it changes into ' on- time' and apply into flight table.

4. Add airplane in the system

```
ins = 'INSERT INTO airplane VALUES(%s, %s, %s, %s, %s, %s, %s)'
cursor.execute(ins, (id,seats,airline_name,manufacturing_company,model_num,manufacturing_date,age))
```

5. Add new airport

```
cursor = conn.cursor()
ins = 'INSERT INTO airport (code, name, city, country, terminal, type) VALUES (%s, %s, %s, %s, %s, %s)'
cursor.execute(ins, (code, name, city, country, terminal, airport_type))
```

6.Logout

```
@app.route('/logout')
def index():
    session['username']=None
    return render_template('login.html')
```