1. View Public Info:

i. public search flight: render\_template('publicHome.html')

* + - 1. **get the information of an upcoming flight needed for the public home page, which is the public requests to search for**
      2. query = "SELECT flight\_number, airline\_name, departure\_airport, departure\_city, departure\_time, arrival\_airport, arrival\_city, arrival\_time, price, airplane\_id FROM flight WHERE departure\_airport = if (\'{}\' = '', departure\_airport, \'{}\') AND arrival\_airport = if (\'{}\' = '', arrival\_airport, \'{}\') AND status = 'upcoming' AND date(departure\_time) = if (\'{}\' = '',date(departure\_time), \'{}\') AND date(arrival\_time) = if (\'{}\' = '',date(arrival\_time), \'{}\') AND departure\_city = if (\'{}\' = '', departure\_city, \'{}\') AND arrival\_city = if (\'{}\' = '', arrival\_city, \'{}\') "
      3. **If not valid, an error message**
    1. Public search status: render\_template('publicHome.html')
       1. **get the information of the status of a flight needed for the public home page, which is the public requests to search for**
       2. query = "SELECT airline\_name, flight\_number, departure\_airport, departure\_city, departure\_time, arrival\_airport, departure\_city, arrival\_time, price, status, airplane\_id FROM flight WHERE flight\_number = if (\'{}\' = '', flight\_number, \'{}\') and date(departure\_time) = if (\'{}\' = '', date(departure\_time), \'{}\') and date(arrival\_time) = if (\'{}\' = '', date(arrival\_time), \'{}\') and airline\_name = if (\'{}\' = '', airline\_name, \'{}\') "

1. Login, Registration and Home pages for logged in users:
   1. Customer:
      1. Login: render\_template('cuslogin.html')
         1. **To check if it is a valid login**
         2. query = "SELECT \* FROM customer WHERE email = \'{}\' and password = md5(\'{}\')"
         3. **If so, get information needed for the home page, including customer username and purchased flight info, and render cushome page**
         4. query = "SELECT ticket\_id, airline\_name, airplane\_id, flight\_number, d.airport\_city, departure\_airport, a.airport\_city, arrival\_airport, departure\_time, arrival\_time, status FROM purchase natural join flight natural join ticket, airport as d, airport as a WHERE customer\_email = \'{}\' and status = 'upcoming' and \

d.airport\_name = departure\_airport and a.airport\_name = arrival\_airport"

**\*after finishing this part, we feel that it would be easier to include departure and arrival city into our database and made the adjustment to the database**

* + 1. Register: render\_template('cusregister.html')
       1. **To check if it is already registered**
       2. query = "SELECT \* FROM customer WHERE email = \'{}\'"
       3. **If not, insert a new row in the customer**
       4. ins = "INSERT INTO customer VALUES(\'{}\', \'{}\', md5(\'{}\'), \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\')"
       5. cursor.execute(ins.format(email, name, pwd, building\_number, street, city, state, phone\_number, passport\_number, passport\_expiration, passport\_country, dob))
       6. **Get information needed for home page same as login page**
       7. **If valid, go to homepage**
    2. Home+View My Flights: render\_template('cushome.html', email=email, emailName=email.split('@')[0], view\_my\_flights=data)
       1. **Get information needed for home page is the same as login page, including data and flights info**
       2. **If session was cleared, render\_template('404.html')**
  1. Agent:
     1. Login: render\_template('agentlogin.html')
        1. **To check if it is a valid login**
        2. query = "SELECT \* FROM booking\_agent WHERE email = \'{}\' and password = md5(\'{}\') "
        3. **If so, get information needed for the home page, including email name, agent\_id and purchased flight info within 30 days, and render agenthome page**
        4. query2 = "SELECT booking\_agent\_id, ticket\_id, customer\_email, purchase.date, airline\_name, flight\_number, departure\_city, departure\_airport, departure\_time, arrival\_city, arrival\_airport, \

arrival\_time, price FROM flight NATURAL JOIN purchase NATURAL JOIN ticket NATURAL JOIN booking\_agent WHERE booking\_agent.email = \'{}\' and status = 'upcoming' and datediff(CURDATE(), DATE(departure\_time)) < 30"

* + 1. Register: render\_template('agentregister.html')
       1. **To check if it is already registered**
       2. query = "SELECT \* FROM booking\_agent WHERE email = \'{}\'"
       3. **If not, insert a new row in the booking\_agent**
       4. ins = "INSERT INTO booking\_agent VALUES(\'{}\',md5(\'{}\'), \'{}\')"
       5. cursor.execute(ins.format(email, pwd, booking\_agent\_id))
       6. **Get information needed for home page same as login page**
       7. **If valid, go to homepage**
    2. Home+View My Flights: render\_template('agenthome.html', email=email, emailName=email.split('@')[0], data2=data2, bid=data1[0][0])
       1. **Get information needed for home page is the same as login page, including data and flights info**
       2. **If session was cleared, render\_template('404.html')**
  1. Staff
     1. Login: render\_template('stafflogin.html')
        1. **To check if it is a valid login**
        2. query = "SELECT \* FROM airline\_staff WHERE username = \'{}\' and password = md5(\'{}\')"
        3. **If so, get information needed for the home page, including flight info of his/her airline within 30 days, his permission, and all his colleagues info, and render staffhome page**
        4. query1 = "SELECT username, airline\_name, airplane\_id, flight\_number, departure\_airport, arrival\_airport, departure\_time, arrival\_time, remaining\_tickets FROM flight NATURAL JOIN airline\_staff WHERE username = \'{}\' and status = 'upcoming' and datediff(CURDATE(), CURDATE()) < 30 "
        5. query2 = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
     2. Register: render\_template('staffregister.html')
        1. **To check if it is already registered**
        2. query = "SELECT \* FROM airline\_staff WHERE username = \'{}\'"
        3. **If not, insert a new row in the airline\_staff**
        4. ins = "INSERT INTO airline\_staff VALUES(\'{}\', md5(\'{}\'), \'{}\', \'{}\', \'{}\', \'{}\', 'staff')"
        5. cursor.execute(ins.format(username, password, first\_name, last\_name, date\_of\_birth, airline\_name))
        6. **Get information needed for home page same as login page**
        7. **If valid, go to homepage**
     3. Home+View My Flights+Grant Permission: render\_template('staffhome.html', username=username, posts=data,permission=data3,staffinfo=data4)
        1. **Get information needed for home page is the same as login page, including data and flights info**
        2. **If session was cleared, render\_template('404.html')**
        3. **If permission = ‘admin’, this staff can edit Permission**
        4. upd = "UPDATE airline\_staff set permission = \'{}\' WHERE username = \'{}\''"
        5. cursor.execute(upd.format(permission,staff\_username))

1. Other Customer Use Cases:
   1. Search & Purchase Ticket
      1. Search: render\_template('cusSearchPurchase.html’')
         1. **Get the information of the flight which the customer requests to search for**
         2. query1 = "SELECT airline\_name, airplane\_id, flight\_number, departure\_city, departure\_airport, arrival\_city, arrival\_airport, departure\_time, arrival\_time, price, status, remaining\_tickets from flight WHERE departure\_city = if (\'{}\' = '', departure\_city, \'{}\') AND departure\_airport = if (\'{}\' = '', departure\_airport, \'{}\') and arrival\_city = if (\'{}\' = '', arrival\_city, \'{}\') and arrival\_airport = if (\'{}\' = '', arrival\_airport, \'{}\') and date(departure\_time) = if (\'{}\' = '', date(departure\_time), \'{}\') and date(arrival\_time) = if (\'{}\' = '', date(arrival\_time), \'{}\')"
         3. cursor.execute(query1.format(departure\_city,departure\_city,departure\_airport,departure\_airport, arrival\_city, arrival\_city, arrival\_airport, arrival\_airport, departure\_date, departure\_date, arrival\_date,arrival\_date))
         4. **If no such flight exists, show an error message.**
      2. Purchase ticket: render\_template(render\_template('cusSearchPurchase.html’')
         1. **Get the information of the flight which the customer requests to buy**
         2. query = "SELECT \* FROM flight WHERE airline\_name = \'{}\' AND flight\_number = \'{}\' AND remaining\_tickets > 0 "
         3. **If not valid, error message**
         4. **If valid, get the corresponding ticket id of the flight**
         5. query\_id = "SELECT ticket\_id FROM ticket ORDER BY ticket\_id DESC LIMIT 1"
         6. **Then, insert a new row in the ticket table**
         7. ins = "INSERT INTO ticket VALUES (\'{}\', \'{}\', \'{}\')"
         8. **Also, insert a new row in the purchase table**
         9. ins = "INSERT INTO purchase VALUES (\'{}\', \'{}\', NULL, CURDATE())"
         10. **Finally, reduce the remaining tickets by 1**
         11. upd = "UPDATE flight set remaining\_tickets = \'{}\' WHERE flight\_number = \'{}\'"
   2. Track My Spending
      1. Total spending+month-wise spending: render\_template('cusSpending.html’')
         1. **Get the total spending of a period of time which the customer requests to track**
         2. query = "select sum(price) from purchase natural join ticket natural join flight where customer\_email = \'{}\' and (date between DATE\_ADD(NOW(), INTERVAL - \'{}\' DAY) and NOW())"
         3. **Get the month-wise spending of a period of time which the customer requests to track**
         4. query = "select year(date) as year, month(date) as month, sum(price) as monthly\_spending from purchase natural join ticket natural join flight where customer\_email = \'{}\' and (date between date(\'{}\') and NOW()) group by year(date), month(date)"
         5. cursor.execute(upd.format(int(data[0][-1])-1, flight\_num))
2. Other Agent Use Cases:
   1. Search & Purchase Ticket
      1. Search: render\_template('agentSearchPurchase.html’')
         1. **Get the flights info of the airlines he works for**
         2. query = "SELECT airline\_name, airplane\_id, flight\_number, departure\_airport, departure\_city, arrival\_airport, arrival\_city, departure\_time, arrival\_time, status, price, remaining\_tickets from flight WHERE departure\_city = if (\'{}\' = '', departure\_city, \'{}\') AND departure\_airport = if (\'{}\' = '', departure\_airport, \'{}\') and arrival\_city = if (\'{}\' = '', arrival\_city, \'{}\') and arrival\_airport = if (\'{}\' = '', arrival\_airport, \'{}\') and date(departure\_time) = if (\'{}\' = '', date(departure\_time), \'{}\') and date(arrival\_time) = if (\'{}\' = '', date(arrival\_time), \'{}\') and airline\_name in (select airline\_name FROM works\_for WHERE email = \'{}\')"
         3. cursor.execute(query.format(departure\_city, departure\_city,departure\_airport,departure\_airport, arrival\_city, arrival\_city, arrival\_airport, arrival\_airport, departure\_date, departure\_date, arrival\_date, arrival\_date,email))
         4. **If no such flight exists, show an error message.**
      2. Purchase:
         1. **Get the information of the flight which the agent requests to buy**
         2. query = "SELECT \* FROM flight WHERE airline\_name = \'{}\' AND flight\_number = \'{}\'"
         3. **If not valid, error message**
         4. **If valid, get the corresponding ticket id of the flight**
         5. query\_id = "SELECT ticket\_id FROM ticket ORDER BY ticket\_id DESC LIMIT 1"
         6. **Then, insert a new row in the ticket table**
         7. ins = "INSERT INTO ticket VALUES (\'{}\', \'{}\', \'{}\')"
         8. **Also, insert a new row in the purchase table**
         9. ins = "INSERT INTO purchase VALUES (\'{}\', \'{}\', \'{}\', CURDATE())"
         10. cursor.execute(ins.format(new\_ticket\_id, customer\_email, booking\_agent\_id))
         11. **Finally, reduce the remaining tickets by 1**
         12. upd = "UPDATE flight set remaining\_tickets = \'{}\' WHERE flight\_number = \'{}\'"
         13. cursor.execute(upd.format(int(data[0][-1])-1, flight\_num))
   2. View My Commission
      1. search: render\_template('agentCommission.html’')
         1. **Get the commission data for a period of time which the agent requests to view, including the total amount of commission he/she received, the average commission he/she received per ticket booked in a period of time, and total number of tickets sold by him/her in a period of time**
         2. query = 'select sum(price \* 0.1), avg(price \* 0.1), count(price \* 0.1) from flight natural join ticket natural join purchase natural join booking\_agent where email = \'{}\' and (date between DATE\_ADD(NOW(), INTERVAL -\'{}\' DAY) and NOW())'
   3. Top Customers: return render\_template('agentTopCustomers.html', email=email, emailName=email.split('@')[0], ppl1=ppl1, ppl2=ppl2, tickets=ticket\_data, commissions=commission\_data)
      1. **Get top 5 customers based on number of tickets bought last 6 months**
      2. query = "select customer\_email, count(ticket\_id) from flight natural join ticket natural join purchase natural join booking\_agent where email = \'{}\' and datediff(CURDATE(), DATE(date)) < 182 group by customer\_email order by count(ticket\_id) desc"
      3. cursor.execute(query.format(email))
      4. **Get top 5 customers based on amount of commission received last year**
      5. query2 = "select customer\_email, sum(price) \* 0.1 from flight natural join ticket natural join purchase natural join booking\_agent where email = \'{}\' and datediff(CURDATE(), DATE(date)) < 365 group by customer\_email order by sum(price) desc"
      6. cursor.execute(query2.format(email))
3. Other Staff Use Cases: render\_template('staffflight.html', username=username, upcoming\_flights=data1, posts = data2)
   1. Change Status of flights
      1. **Search flight that the staff want to edit status**
      2. query = "SELECT airline\_name, airplane\_id, flight\_number, departure\_airport, departure\_city, arrival\_airport, arrival\_city, departure\_time, arrival\_time, status, price from flight WHERE departure\_city = if (\'{}\' = '', departure\_city, \'{}\') AND departure\_airport = if (\'{}\' = '', departure\_airport, \'{}\') and arrival\_city = if (\'{}\' = '', arrival\_city, \'{}\') and arrival\_airport = if (\'{}\' = '', arrival\_airport, \'{}\') and date(departure\_time) = if (\'{}\' = '', date(departure\_time), \'{}\') and date(arrival\_time) = if (\'{}\' = '', date(arrival\_time), \'{}\') and airline\_name = \'{}\' "
      3. cursor.execute(query.format(departure\_city, departure\_city,departure\_airport,departure\_airport, arrival\_city, arrival\_city, arrival\_airport, arrival\_airport, departure\_date, departure\_date, arrival\_date,arrival\_date,data2[0][1]))
      4. **Get the permission and check whether it is ‘operator’**
      5. query\_permission = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
      6. cursor.execute(query\_permission.format(username))
      7. **If is ‘operation’, commit change**
      8. upd = "UPDATE flight set status = \'{}\' WHERE flight\_number = \'{}\'"
      9. cursor.execute(upd.format(status, flight\_num))
      10. **If not, show an error message.**
   2. Add New Items (flights, airplane, airport): render\_template('staffaddinfo.html')
      1. Create flights
         1. **Check whether departure & arrival airport, exists**
         2. query = "SELECT airport\_name FROM airport WHERE airport\_name = \'{}\'"
         3. **Check whether number of tickets is smaller than the seats**
         4. num = "SELECT seats FROM airplane NATURAL JOIN airline\_staff WHERE username = \'{}\' and airplane\_id = \'{}\'"
         5. **Check whether airplane exists**
         6. query = "SELECT airplane\_id FROM airplane WHERE airline\_name = \'{}\' and airplane\_id = \'{}\'"
         7. **Check whether the flight with same primary key already exists**
         8. query = "SELECT airline\_name, flight\_number FROM flight WHERE airline\_name = \'{}\' and flight\_number = \'{}\'"
         9. **Get permission and check whether it is ‘admin’**
         10. query\_permission = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
         11. cursor.execute(query\_permission.format(username))
         12. **If so, create flight**
         13. ins = "INSERT INTO flight VALUES(\'{}\', \'{}\', \'{},{}\', \'{},{}\',\'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\',\'{}\')"
         14. cursor.execute(ins.format(airline\_name, flight\_num, departure\_date, departure\_time,arrival\_date,arrival\_time, price, status, airplane\_id, departure\_airport, arrival\_airport,departure\_city,arrival\_city,number))
         15. **If not, show an error message**
      2. Add airplane:
         1. **Check whether the airplane already exists**
         2. query = "SELECT airline\_name, airplane\_id FROM airplane WHERE airline\_name = \'{}\' and airplane\_id = \'{}\'"
         3. cursor.execute(query.format(airline\_name, airplane\_id))
         4. **Get staff permission and check whether it is ‘admin’**
         5. query\_permission = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
         6. cursor.execute(query\_permission.format(username))
         7. **If so, create new airplane**
         8. ins = "INSERT INTO airplane VALUES(\'{}\', \'{}\', \'{}\')"
         9. cursor.execute(ins.format(airline\_name, airplane\_id, seats))
         10. **If not, show an error message**
      3. Add airport:
         1. **Check whether the airport already exists**
         2. airport = "SELECT airport\_name FROM airport WHERE airport\_name = \'{}\'"
         3. cursor.execute(airport.format(airport\_name))
         4. **Get the staff permission and check whether it is ‘admin’**
         5. query\_permission = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
         6. cursor.execute(query\_permission.format(username))
         7. **If so, add airport**
         8. ins = "INSERT INTO airport VALUES(\'{}\', \'{}\')"
         9. cursor.execute(ins.format(airport\_name, airport\_city))
         10. **If not, show an error message**
   3. Agent (view top agents, add booking agents): render\_template('staffagent.html')
      1. **Get top 5 booking agents based on the amount of commission received for the last year**
      2. query1 = "SELECT email, booking\_agent\_id, sum(price) \* 0.1 as commission FROM booking\_agent NATURAL JOIN purchase NATURAL JOIN flight NATURAL JOIN ticket AS T, airline\_staff WHERE username = \'{}\' and airline\_staff.airline\_name = T.airline\_name and datediff(CURDATE(), DATE(date)) < 365 GROUP BY email, booking\_agent\_id ORDER BY commission DESC LIMIT 5 "
      3. cursor.execute(query1.format(username))
      4. **Get top 5 booking agents based on the number of tickets sales for the past month and past year**
      5. query2 = "SELECT booking\_agent.email, booking\_agent\_id, count(ticket\_id) as ticket FROM booking\_agent NATURAL JOIN purchase NATURAL JOIN ticket AS T, airline\_staff WHERE username = \'{}\' and airline\_staff.airline\_name = T.airline\_name and datediff(CURDATE(), DATE(date)) < 30 GROUP BY email, booking\_agent\_id ORDER BY ticket DESC LIMIT 5 "
      6. cursor.execute(query2.format(username))
      7. query3 = "SELECT email, booking\_agent\_id, count(ticket\_id) as ticket FROM booking\_agent NATURAL JOIN purchase NATURAL JOIN ticket AS T, airline\_staff WHERE username = \'{}\' and airline\_staff.airline\_name = T.airline\_name and datediff(CURDATE(), DATE(date)) < 365 GROUP BY email, booking\_agent\_id ORDER BY ticket DESC LIMIT 5 "
      8. cursor.execute(query3.format(username))
      9. Add new agents
         1. **Check whether this agent already works for your airline**
         2. query5 = "select \* from works\_for where email=\'{}\' and airline\_name=\'{}\'"
         3. cursor.execute(query5.format(agent\_email,airline\_name\_staff[0][0]))
         4. **Get staff permission and check whether it is ‘admin’**
         5. query\_permission = "select permission, airline\_name from airline\_staff WHERE username = \'{}\' "
         6. cursor.execute(query\_permission.format(username))
         7. **If so, add agent**
         8. ins = "INSERT INTO works\_for VALUES(\'{}\', \'{}\')"
         9. cursor.execute(ins.format(agent\_email, airline\_name\_staff[0][0]))
         10. **If not, show an error message**
   4. Customer (view frequent customers, see flights bought by specific customer): render\_template('staffcus.html')
      1. **view frequent customers last year**
      2. query1 = "SELECT email, name, count(ticket\_id) as ticket FROM customer, purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE email = customer\_email AND username = \'{}\' and datediff(CURDATE(), DATE(date)) < 365 GROUP BY email, name ORDER BY ticket DESC LIMIT 1"
      3. **List all flights a particular Customer has taken of that particular airline**
      4. query2 = "SELECT DISTINCT airplane\_id, flight\_number, departure\_airport, arrival\_airport, departure\_time, arrival\_time, status FROM customer, purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE email = \'{}\' and email = customer\_email and username = \'{}\'"
      5. cursor.execute(query2.format(email, username))
   5. View Top Destinations: render\_template('staffDest.html')
      1. **Find the top 3 most popular destinations last 3 months and last year**
      2. query1 = "SELECT airport\_city, count(ticket\_id) AS ticket FROM purchase NATURAL JOIN ticket NATURAL JOIN flight, airport WHERE airport\_name = arrival\_airport and datediff(CURDATE(), DATE(date)) < 90 GROUP BY airport\_city ORDER BY ticket DESC LIMIT 3"
      3. query2 = "SELECT airport\_city, count(ticket\_id) AS ticket FROM purchase NATURAL JOIN ticket NATURAL JOIN flight, airport WHERE airport\_name = arrival\_airport and datediff(CURDATE(), DATE(date)) < 365 GROUP BY airport\_city ORDER BY ticket DESC LIMIT 3"
   6. Comparison of Revenue earned: render\_template('staffReve.html')
      1. **Get direct sales last month**
      2. query3 = "SELECT sum(price) FROM purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE username = \'{}\' AND booking\_agent\_id is NULL AND datediff(CURDATE(), DATE(date)) < 30 GROUP BY airline\_name"
      3. **Get indirect sales last month**
      4. query4 = "SELECT sum(price) FROM purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE username = \'{}\' AND booking\_agent\_id is NOT NULL AND datediff(CURDATE(), DATE(date)) < 30 GROUP BY airline\_name"
      5. **Get direct sales last year**
      6. query5 = "SELECT sum(price) FROM purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE username = \'{}\' AND booking\_agent\_id is NULL AND datediff(CURDATE(), DATE(date)) < 365 GROUP BY airline\_name"
      7. **Get indirect sales last year**
      8. query6 = "SELECT sum(price) FROM purchase NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline\_staff WHERE username = \'{}\' AND booking\_agent\_id is NOT NULL AND datediff(CURDATE(), DATE(date)) < 365 GROUP BY airline\_name"
   7. View Tickets Reports: render\_template('staffTickets.html’)
      1. **Get the number of tickets sold during selected time zone**
      2. ticket = "SELECT YEAR(date) AS year, MONTH(date) AS month, count(ticket\_id) FROM purchase NATURAL JOIN airline\_staff NATURAL JOIN flight NATURAL JOIN ticket WHERE date > \'{}\' and date < \'{}\' AND username = \'{}\' GROUP BY year, month ORDER BY year, month"
      3. **Then organize those data by month**
4. Logout: redirect('/cuslogin')
   1. **Clear session:** session.clear()
   2. **Goes back cuslogin page:**  return redirect('/cuslogin')