All Use Cases

Zetong Yang, zy891

Summary

```
## Use Cases
### General
- [x] View Public Info
- [x] Register
  - [x] Customer
  - [x] Booking Agent
  - [x] Airline Staff

- [x] Login
  - [x] Customer
  - [x] Booking Agent
  - [x] Airline Staff
### Customer
- [x] View my flight
- [x] Search for flight
- [x] Purchase Tickets
- [x] Giver ratings and comments on previous flight
- [x] Track my spending
- [x] Logout
### Booking Agent
- [x] View my flights
- [x] Search for flights
- [x] Purchase tickets
- [x] View my commission
- [x] View top customers
- [x] Logout
### Airline Staff
- [x] View flights
- [x] Create new flights
- [x] Change status of flights
- [x] Add airplane in the system
- [x] Add new airport in the system
- [x] View flight ratings
- [x] Add/delete phone number
- [x] View all the booking agents
- [x] View frequent customers
- [ ] View reports
- [ ] Comparison of revenue earned
- [x] View top destination
- [x] Logout
### Enforcing complex constraints
- [x] Prevention of http attacks
- [x] Sessions for each user and authentications each step after login
- [x] Prepared statements
- [x] Prevent cross-site scripting
```

Detail

1. View Public Info: All users, whether logged in or not, can

   a. Search for future flights based on source city/airport name, destination city/airport name, departure date for one way (departure and return dates for round trip).

By airport

```
sql = 'select distinct airline_name,flight_number,departure_time, \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport = %s and DATE(departure_time) = %s'
keys = (arrive_airport, depart_airport, return_date)
```

By city

```
sql = 'select distinct airline_name,flight_number,departure_time, \
         arrival_time, departure_airport, arrival_airport, \
         status  from flight where departure_airport =  \
         (select name from airport where city = %s) and \
         arrival_airport = (select name from airport where city = %s) \
         and DATE(departure_time) = %s'
            keys = (arrive_city, depart_city, return_date)
```

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

```
sql = 'select distinct airline_name,flight_number,departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where airline_name = %s and \
                flight_number = %s \
                and DATE(departure_time) = %s'
keys = (airline_name, flight_num, depart_date)
```

2. Register: 3 types of user registrations (Customer, Booking Agent and Airline Staff) option via forms.

Customer

```
# check if the email already exists
    sql = 'SELECT * FROM Customer WHERE email = %s'
    key = (email)
    data = fetchone(sql, key)

# insert query
sql = 'INSERT INTO Customer \
            VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'
        keys =
(email,name,password,building_num,street,city,state,phone_num,
        passport_num, passport_expiration, passport_country, dob)
```

Booking agent

```
# check if the email already exists
    sql = 'SELECT * FROM Booking_agent WHERE email = %s'
    key = (email)

# insert information
sql = 'INSERT INTO Booking_agent \
            VALUES(%s, %s, %s)'
        keys = (email,password,agent_id)
```

Airline staff

```
#check if airline exists
    sql = 'SELECT * FROM airline WHERE name = %s'
    key = (airline_name)

# check if the email already exists
    sql = 'SELECT * FROM airline_staff WHERE email = %s'
    key = (email)

# insert sql
sql = 'INSERT INTO airline_staff \
        VALUES(%s, %s, %s,%s,%s,%s)'
keys = (email,password,first_name,last_name,dob,airline_name)
```

3. Login: 3 types of user login (Customer, Booking Agent and Airline Staff).

```
#check if agent exists
sql = 'SELECT * FROM ' + usertype + ' WHERE email = %s \
                and password = %s and booking_agent_id = %s'
                keys = (username, password, agent_id)
                data = fetchone(sql,keys)

#for customer and airline staff
        sql = 'SELECT * FROM ' + usertype + \
            ' WHERE email = %s and password = %s'
        keys = (username, password)
```

a. If so, login is successful. A session is initiated with the member's username stored as a session variable

```
        session['username'] = username
        session['usertype'] = usertype
        if usertype == 'Customer':
            return redirect(url_for('customer'))
        elif usertype == 'Booking_agent':
            return redirect(url_for('agent'))
        elif usertype == 'airline_staff':
            return redirect(url_for('staff'))
```

b. If not, login is unsuccessful. A message is displayed indicating this to the user.

```
    else:
        #returns an error message to the html page
        error = 'Invalid login or username'
        return render_template('login.html', error=error)
```

Customer use cases:

4. View My flights: Provide various ways for the user to see flights information which he/she purchased.

```
sql = 'SELECT * FROM flight natural join ticket WHERE \
            ticket.customer_email = %s and departure_time > now()'
    key = (username)
```

5. Search for flights: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or return).
By airport

```
# one way
sql = 'select distinct airline_name, flight_number, departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport = %s and DATE(departure_time) = %s'
keys = (depart_airport, arrive_airport, depart_date)

#return
sql = 'select distinct airline_name,flight_number,departure_time, \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport = %s and DATE(departure_time) = %s'
            keys = (arrive_airport, depart_airport, return_date)
            return_flight = fetchall(sql, keys)
```

By city

```
# one way
sql = 'select distinct airline_name,flight_number,departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = \
                (select name from airport where city = %s) and \
                arrival_airport = (select name from airport where city
= %s) \
                and DATE(departure_time) = %s'
        keys = (depart_city, arrive_city, depart_date)
# return
sql = 'select distinct airline_name,flight_number,departure_time, \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport =  \
                (select name from airport where city = %s) and \
                arrival_airport = (select name from airport where city
= %s) \
                and DATE(departure_time) = %s'
            keys = (arrive_city, depart_city, return_date)
```

6. Purchase tickets: Customer chooses a flight and purchase ticket for this flight, providing all the needed data, via forms.

```
# choose the flight
sql = 'select * from flight where airline_name = %s and \
                flight_number = %s and departure_time = %s'
        keys = (airline_name, flight_num, departure_datetime)
# get passenger number
   sql = 'select count(*) as head from ticket where airline_name = %s \
            and flight_number = %s and departure_time = %s'
        key = (result[0]['airline_name'],result[0]['flight_number'],\
            result[0]['departure_time'])
# get total seats
        total_seats = fetchone('select seats from airplane where \
            airplane.id = %s',(result[0]['airplane_id']))['seats']
# insert sql
    sql = 'insert into ticket values \
           (%s,%s,%s,%s,%s,%s, CURDATE(), CURTIME(),%s, %s,%s,%s,NULL)'
        key = (random.randint(1,255),session['sold_price'],card_type, \
            card_number, name_on_card, expiration_date, \
                session['flight_number'], \
                session['departure_time'],session['airline_name'], \
                    session['username'])
```

6. Give Ratings and Comment on previous flights:

```
#get flight detail to comment
      sql = 'SELECT * FROM flight natural join ticket WHERE \
           ticket.customer_email = %s and departure_time < now()'
       key = (username)
#fetch ticket id
sql = 'select ticket_id from ticket where airline_name = %s and \
   flight_number = %s and departure_time = %s and customer_email = %s'
keys =(airline_name,flight_number,departure_datetime,username)
# insert comment
sql = 'insert into comment values (%s, %s, %s)'
       key = (str(ticket_id),rating,comment)
```

7.Track My Spending: Default view will be total amount of money spent in the past year and a barchart showing month wise money spent for last 6 months.

```
#get total spending
sql = 'select sum(sold_price) as total from ticket where \
customer_email = %s and purchase_date between   date_sub(curdate(),\
             interval 1 year) and curdate()'
keys = (username)
#get ticket detail
sql = 'select month(purchase_date) as month , sum(sold_price) as \
m_total from ticket where customer_email = %s and purchase_date \
     between date_sub(curdate(), interval 6 month) and curdate() \
            group by month'
keys = (username)
```

8.Logout: The session is destroyed and a “goodbye” page or the login page is displayed.

```
@app.route('/logout')
def logout():
    session.pop('username',None)
    session.pop('usertype', None)
    return redirect('/')
```

Booking agent use cases:

After logging in successfully a booking agent may do any of the following use cases:

4. View My flights: Provide various ways for the booking agents to see flights information for which he/she purchased on behalf of customers.

```
sql = 'SELECT * FROM flight natural join ticket WHERE  \
        ticket.booking_agent_id = (select booking_agent_id from
booking_agent\
             where email = %s) and departure_time > now()'
    key = (username)
```

5. Search for flights: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or arrival).
By airport

```
# one way
 sql = 'select distinct airline_name, flight_number, departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport = %s and DATE(departure_time) = %s'
        keys = (depart_airport, arrive_airport, depart_date)
#return
sql = 'select distinct airline_name,flight_number,departure_time, \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport = %s and DATE(departure_time) = %s'
            keys = (arrive_airport, depart_airport, return_date)
```

By city

```
# one way
sql = 'select distinct airline_name,flight_number,departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = \
                (select name from airport where city = %s) and \
                arrival_airport = (select name from airport where city
= %s) \
                and DATE(departure_time) = %s'
        keys = (depart_city, arrive_city, depart_date)
# return
sql = 'select distinct airline_name,flight_number,departure_time, \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport =  \
                (select name from airport where city = %s) and \
         arrival_airport = (select name from airport where city = %s) \
                and DATE(departure_time) = %s'
            keys = (arrive_city, depart_city, return_date)
```

6. Purchase tickets: Booking agent chooses a flight and purchases tickets for other customers giving customer information and payment information, providing all the needed data, via forms.

```
# select flight to purchase
        sql = 'select * from flight where airline_name = %s and \
                flight_number = %s and departure_time = %s'
        keys = (airline_name, flight_num, departure_datetime)
# get passenger number
   sql = 'select count(*) as head from ticket where airline_name = %s \
            and flight_number = %s and departure_time = %s'
        key = (result[0]['airline_name'],result[0]['flight_number'],\
            result[0]['departure_time'])
#get seat number
        total_seats = fetchone('select seats from airplane where \
            airplane.id = %s',(result[0]['airplane_id']))['seats']
#check email validity
fetchone('select * from customer where email = %s',\
            (customer_email))
#fetch booking agent id
        sql = 'select booking_agent_id from booking_agent where email
= %s'
        keys = (username)
#make sql insert
        sql = 'insert into ticket values \
            (%s,%s,%s,%s,%s,%s, CURDATE(), CURTIME(),%s, %s,%s,%s,%s)'
        key = (random.randint(1,255),session['sold_price'],card_type, \
            card_number, name_on_card, expiration_date, \
                session['flight_number'], \
                session['departure_time'],session['airline_name'], \
                    customer_email, booking_agent_id)
```

7. View my commission: Default view will be total amount of commission received in the past 30 days and the average commission he/she received per ticket booked in the past 30 days and total number of tickets sold by him in the past 30 days. He/she will also have option to specify a range of dates to view total amount of commission received and total numbers of tickets sold.

```
sql = 'select sum(sold_price * 0.1) as total, count(*) as sales from \
            ticket where booking_agent_id = %s and purchase_date
between \
            date_sub(curdate(), interval 30 day) and curdate()'
        keys = (booking_agent_id)
```

8. View Top Customers: Top 5 customers based on number of tickets bought from the booking agent in the past 6 months and top 5 customers based on amount of commission received in the last year

```
sql = 'select customer_email, count(*) as fac from ticket where \
                booking_agent_id = %s and purchase_date between \
                date_sub(curdate(), interval 6 month) and curdate()
group by \
                customer_email order by fac desc'
sql = 'select customer_email, sum(sold_price) as fac from ticket where
\
                booking_agent_id = %s and purchase_date between \
                date_sub(curdate(), interval 1 year) and curdate()
group by \
                customer_email order by fac desc'
```

9. Logout: The session is destroyed and a “goodbye” page or the login page is displayed.

Airline Staff use cases:

After logging in successfully an airline staff may do any of the following use cases:

4. View flights:
Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days.

```
#* view my flights default
    sql = 'SELECT * FROM flight WHERE airline_name = %s \
        and departure_time between now() and date_add(now(), interval
30 day)'
    key = (airline_name)
```

He/she will be able to see all the current/future/past flights operated by the airline he/she works for based range of dates, source/destination airports/city etc.
By airport

```
sql = 'select distinct airline_name, flight_number, departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = %s and \
                arrival_airport= %s and DATE(departure_time) between %s
and %s'
        keys = (depart_airport, arrive_airport, start_date,end_date)
```

By city

```
sql = 'select distinct airline_name,flight_number,departure_time , \
                arrival_time, departure_airport, arrival_airport, \
                status  from flight where departure_airport = \
                (select name from airport where city = %s) and \
                arrival_airport = (select name from airport where city
= %s) \
                and DATE(departure_time) between %s and %s'
        keys = (depart_city, arrive_city, start_date,end_date)
```

He/she will be able to see all the customers of a particular flight.

```
sql = 'select * from customer where email in (select distinct\
            customer_email from ticket where airline_name = %s and \
            flight_number = %s and departure_time = %s)'
        keys = (airline_name,flight_num,departure_datetime)
```

5. Create new flights: He or she creates a new flight, providing all the needed data, via forms.

```
#check if flight number exists
fetchone('select * from flight where flight_number = %s',\
            (flight_num))

#check if airplane id exists
fetchone('select * from airplane where id = %s',\
            (airplane_id))

#check if airport exists
fetchone('select * from airport where %s in \
            (select name from airport) and %s in (select name from
airport)',\
            (depart_airport,arrive_airport))
#insert flight detail
sql = 'insert into flight values (%s,%s,%s,%s,%s,%s,%s,%s,%s)'
        keys = (flight_num,depart_datetime,arrive_datetime,price, \

status,airline_name,airplane_id,depart_airport,arrive_airport)
```

6. Change Status of flights: He or she changes a flight status (from on-time to delayed or vice versa) via forms.

```
sql = 'update flight set status = %s where flight_number = %s and \
            departure_time = %s and airline_name = %s'
        keys = (status, flight_num, depart_datetime, airline_name)
```

7. Add airplane in the system: He or she adds a new airplane, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. In the

confirmation page, she/he will be able to see all the airplanes owned by the airline he/she works for.

```
#get the last airplane id
        sql = 'select max(id) as id_max from airplane where 
airline_name = %s '
sql = 'insert into airplane values (%s, %s, %s)'
        keys = (str(airplane_id), seats, airline_name)
```

8. Add new airport in the system: He or she adds a new airport, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action.

```
sql = 'insert into airport values (%s, %s)'
        keys = (airport_name, airport_city)
```

9. View flight ratings: Airline Staff will be able to see each flight's average ratings and all the comments and ratings of that flight given by the customers.

```
sql = 'select * from comment natural join ticket where \
            airline_name = %s and flight_number = %s and departure_time 
= %s'
        keys = (airline_name,flight_num,departure_datetime)
```

10. View all the booking agents: Top 5 booking agents based on number of tickets sales for the past month and past year. Top 5 booking agents based on the amount of commission received for the last year.

```
sql = 'select email, booking_agent_id, count(*) as sales from \
            booking_agent natural join ticket where purchase_date 
between \
            date_sub(curdate(), interval 1 ' + timespan + ') and 
curdate() \
            group by email, booking_agent_id order by sales desc limit 
5'
sql = 'select email, booking_agent_id, sum(sold_price * 0.1) as \
            commission from booking_agent natural join ticket where \
            purchase_date between date_sub(curdate(), interval 1 year) 
and \
            curdate() group by email, booking_agent_id order by 
commission \
            desc limit 5'
```

11. View frequent customers: Airline Staff will also be able to see the most frequent customer within the last year. In addition, Airline Staff will be able to see a list of all flights a particular Customer has taken only on that particular airline.

```
# view the most frequent customer

sql = 'select customer_email, count(*) as frequency from ticket where \
            airline_name = %s and purchase_date between
date_sub(curdate(), \
            interval 1 year) and curdate() group by customer_email
order by \
            frequency desc limit 1'
        keys =(airline_name)
# view flight by customer
sql = 'select distinct airline_name, flight_number, departure_time, \
             departure_airport, arrival_airport from ticket natural
join \
                flight where customer_email = %s and airline_name = %s'
        keys =(customer_email,airline_name)
```

12. View reports: Total amounts of ticket sold based on range of dates/last year/last month etc. Month wise tickets sold in a bar chart.

```
sql = 'select month(purchase_date) as month, count(*) as m_total from \
            ticket where airline_name = %s and purchase_date between %s
and %s\
                 group by month'
        keys = (airline_name, start_date,end_date)
```

13. Comparison of Revenue earned:

```
sql = 'select sum(sold_price) as rev from ticket where airline_name = \
            %s and purchase_date between date_sub(curdate(), interval 1
\
                '+ option + ') and curdate()'
```

14. View Top destinations: Find the top 3 most popular destinations for last 3 months and last year (based on tickets already sold).

```
sql = 'select arrival_airport, count(*) as frequency from ticket \
            natural join flight where airline_name = %s and
departure_time \
            between date_sub(curdate(), interval '+ timespan +  ') and
\
            curdate() group by arrival_airport order by frequency desc
limit 3'
        keys =(airline_name)
```

15. Logout: The session is destroyed and a “goodbye” page or the login page is displayed.

Additional Requirements:

Enforcing complex constraints: Your air ticket reservation system implementation should prevent users from doing actions they are not allowed to do.

```python
def doorman(lock):
    username = session.get('username')
    usertype = session.get('usertype')

    if not username or not usertype :
        return False
    if usertype != lock:
        return False
    sql = 'select * from '+ usertype + ' where email = %s'
    key = (username)
    data = fetchone(sql,key)
    if not data:
        return False
    else:
        return True
```

You must use prepared statements if your programming language supports them.
All query are executed with pymysql prepared statement feature.