ITU DB2155 Documentation

ITU DB2155

FALL 2021

Contents

1 Introduction

ITUDB2155 Airline System Webpage enables that you the users to share the sample feedbacks about the airtrips that you have by evaluating the simple questions related to the Airlines. Moreover you can view the current flight schedules all across the globe. Only the authorized person's deals with the flight schedules and new collaborator airlines and airports.

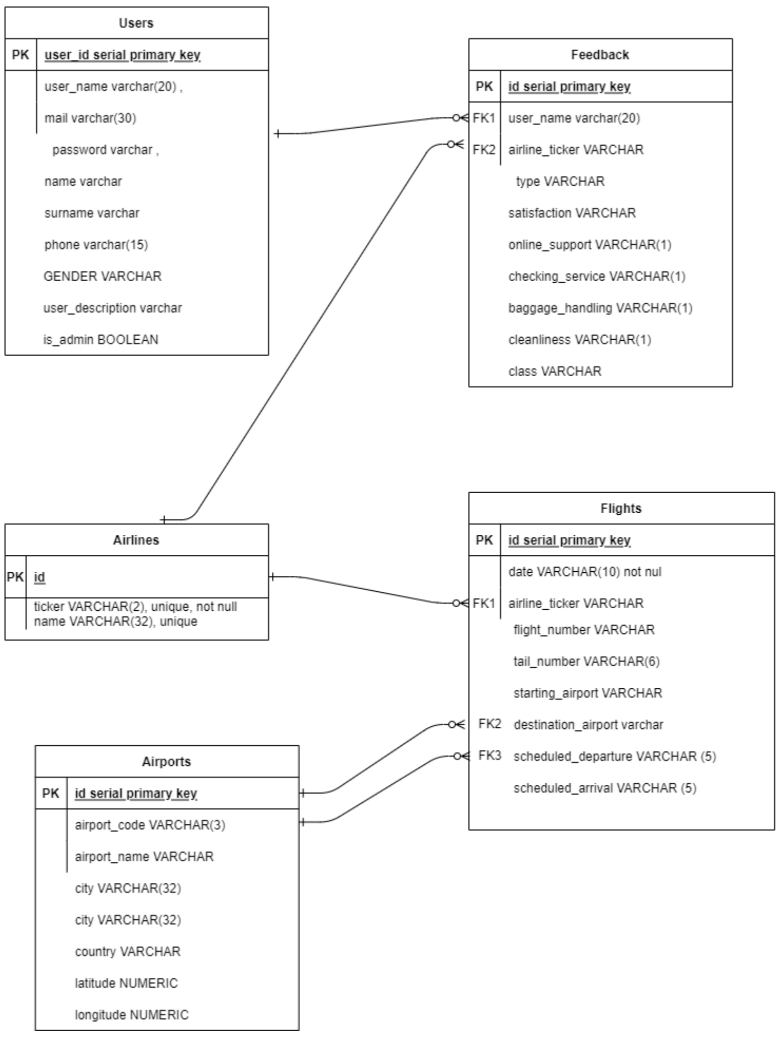
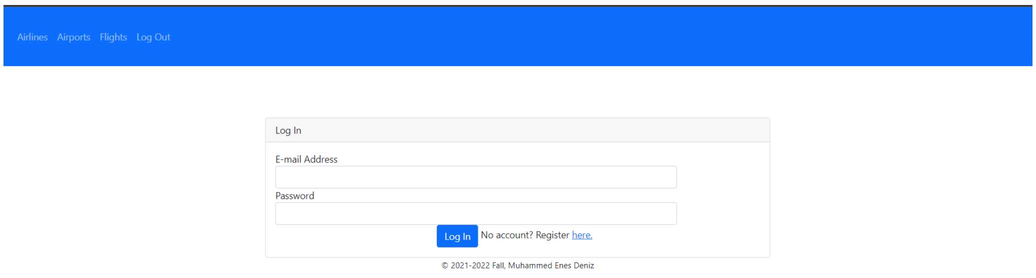


Figure 1: ER Diagram

2 USER GUIDE

2.1 Login



Log In

E-mail Address

Password

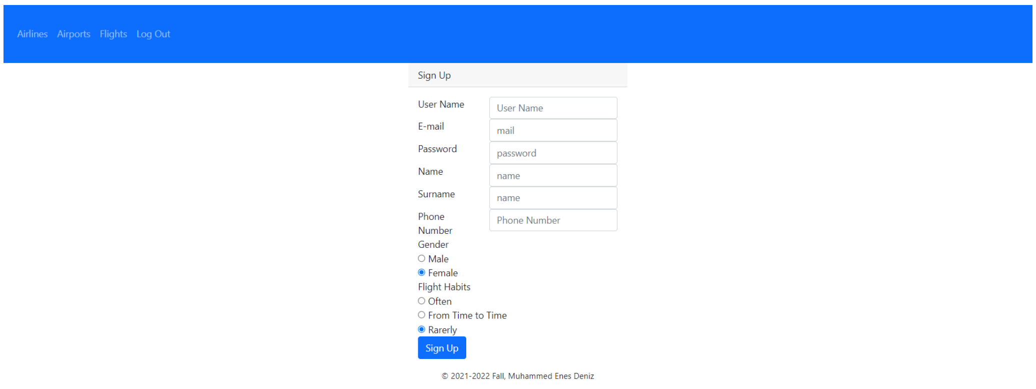
Log In No account? Register here.

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 2: Login

If the user have an existing account it can login by this page. The provided mail and password is checked and if there exists an invalid login attempt user is informed.

2.2 Register



Sign Up

User Name	User Name
E-mail	mail
Password	password
Name	name
Surname	name
Phone Number	Phone Number

Gender
- Male
- [✓ Female]

Flight Habits
- Often
- From Time to Time
- [✓ Rarerly]

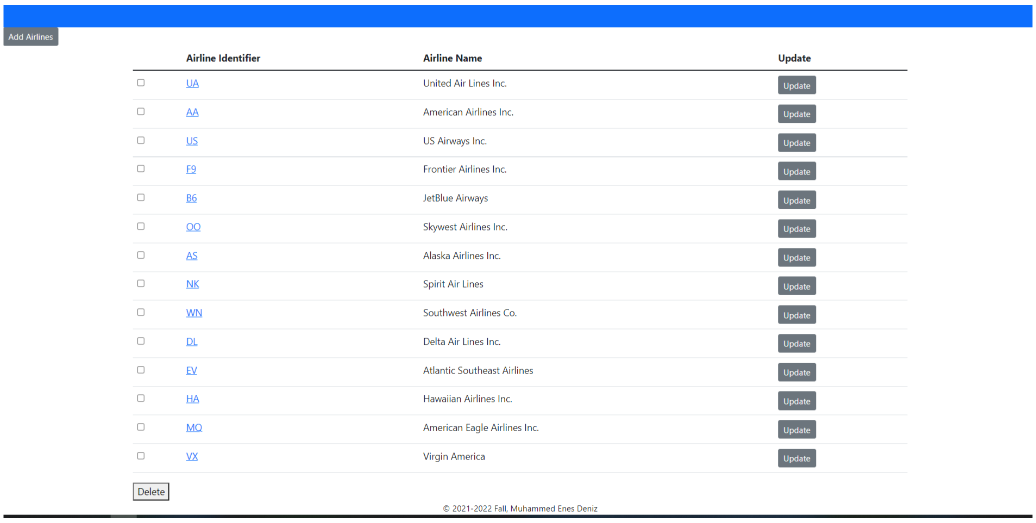
Sign Up

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 3: Register

The user may create account using this page. It must provide simple and necessary information about him. If the username is already taken it is informed about that. It also need to provide a valid password which is greater than 5 characters moreover username could not exceed 20 whereas mail could not exceed 30.

2.3 Airlines


Add Airlines

	Airline Identifier	Airline Name	Update
☐	UA	United Air Lines Inc.	Update
☐	AA	American Airlines Inc.	Update
☐	US	US Airways Inc.	Update
☐	F9	Frontier Airlines Inc.	Update
☐	B6	JetBlue Airways	Update
☐	OO	Skywest Airlines Inc.	Update
☐	AS	Alaska Airlines Inc.	Update
☐	NK	Spirit Air Lines	Update
☐	WN	Southwest Airlines Co.	Update
☐	DL	Delta Air Lines Inc.	Update
☐	EV	Atlantic Southeast Airlines	Update
☐	HA	Hawaiian Airlines Inc.	Update
☐	MQ	American Eagle Airlines Inc.	Update
☐	VX	Virgin America	Update

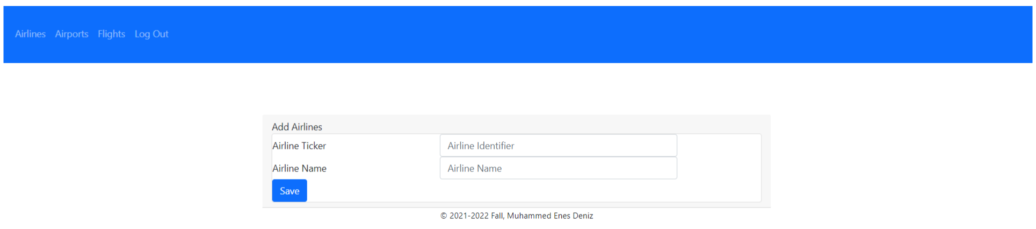
Delete

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 4: Airlines

From this page normal users only could read no further action could be taken carried forward. If the user is admin it could delete an listed airlines from the table and moreover could navigate through the update and add new airlines

2.4 Add Airlines Page



Add Airlines

Airline Ticker	Airline Identifier
Airline Name	Airline Name

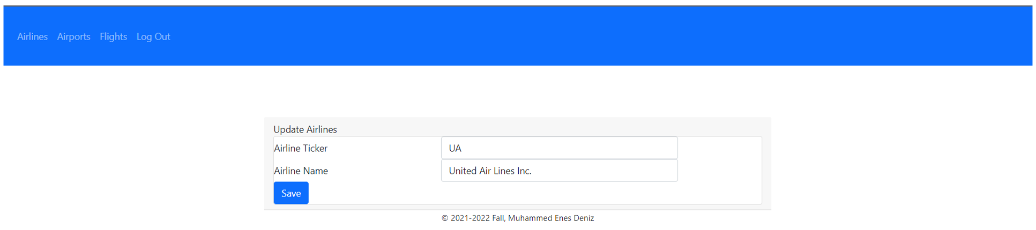
Save

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 5: Add Airlines Page

In this page the admin could add new airlines to the database. It need to provide necessary informations that is asked by the cards.

2.5 Update Airlines Page



Update Airlines

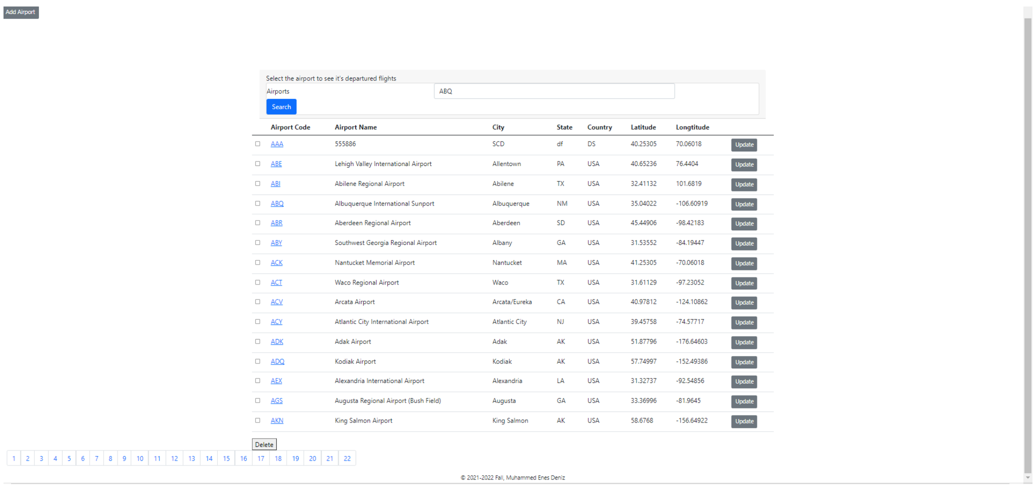
Airline Ticker	UA
Airline Name	United Air Lines Inc.

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 6: Update Airlines Page

In this page the admin could full update existing airlines record in the database.

2.6 Airports Page



Select the airport to see it's departured flights

Airports	ABQ

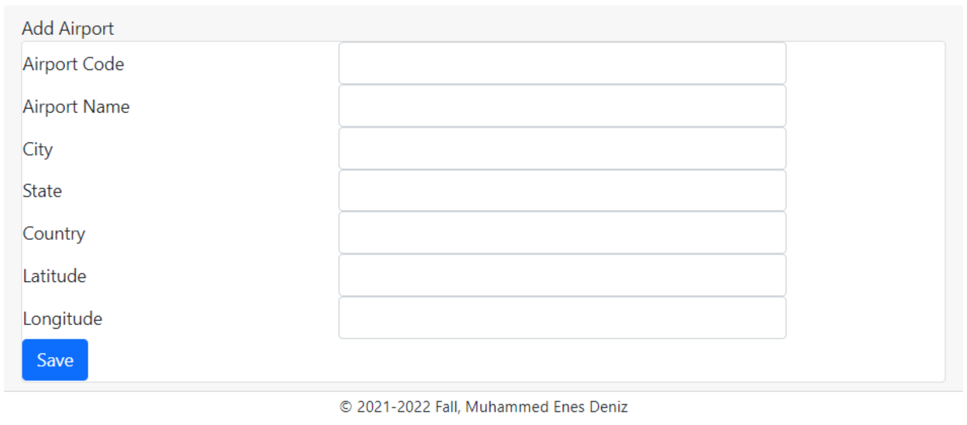
	Airport Code	Airport Name	City	State	Country	Latitude	Longtitude	
☐	AAA	555886	SCD	df	DS	40.25305	70.06018	Update
☐	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	76.4404	Update
☐	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	101.6819	Update
☐	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919	Update
☐	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183	Update
☐	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447	Update
☐	ACK	Nantucket Memorial Airport	Nantucket	MA	USA	41.25305	-70.06018	Update
☐	ACT	Waco Regional Airport	Waco	TX	USA	31.61129	-97.23052	Update
☐	ACV	Arcata Airport	Arcata/Eureka	CA	USA	40.97812	-124.10862	Update
☐	ACY	Atlantic City International Airport	Atlantic City	NJ	USA	39.45758	-74.57717	Update
☐	ADK	Adak Airport	Adak	AK	USA	51.87796	-176.64603	Update
☐	ADQ	Kodiak Airport	Kodiak	AK	USA	57.74997	-152.49386	Update
☐	AEX	Alexandria International Airport	Alexandria	LA	USA	31.32737	-92.54856	Update
☐	AGS	Augusta Regional Airport (Bush Field)	Augusta	GA	USA	33.36996	-81.9645	Update
☐	AKN	King Salmon Airport	King Salmon	AK	USA	58.6768	-156.64922	Update

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 7: Airports Page

In this page normal users only could read no further action could be taken carried forward. If the user is admin it could delete an listed airports from the table and moreover could navigate through the update and add new airlines. If it clicks an airport code it will navigate to the page which is listing the flights that are departure from this airport.

2.7 Add Airports Page



Add Airport	
Airport Code	
Airport Name	
City	
State	
Country	
Latitude	
Longitude	

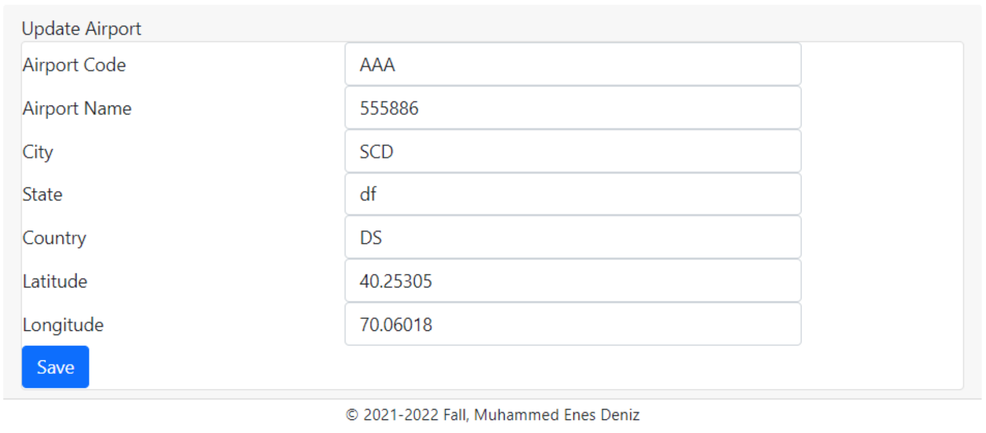
© 2021-2022 Fall, Muhammed Enes Deniz

Figure 8: Add Airports Page

In this page the admin could add new airports to the database. It need to provide necessary informations that is asked by the cards. Validation exists, so entries must be carefully selected. In the case of the invalid entry user is informed.

2.8 Update Airports Page

In this page the admin admin could full update existing airports record in the database.. It need to provide necessary informations that is asked by the cards. Validation exists in this page resulting invalid entries had been explained back to the user.

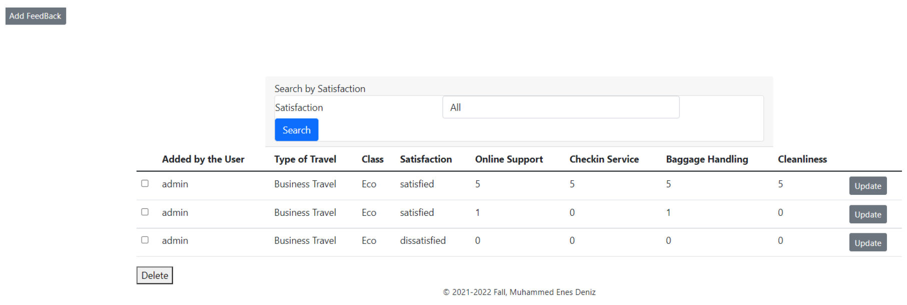


Update Airport	
Airport Code	AAA
Airport Name	555886
City	SCD
State	df
Country	DS
Latitude	40.25305
Longitude	70.06018

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 9: Update Airports Page

2.9 Feedback Page



Satisfaction	All

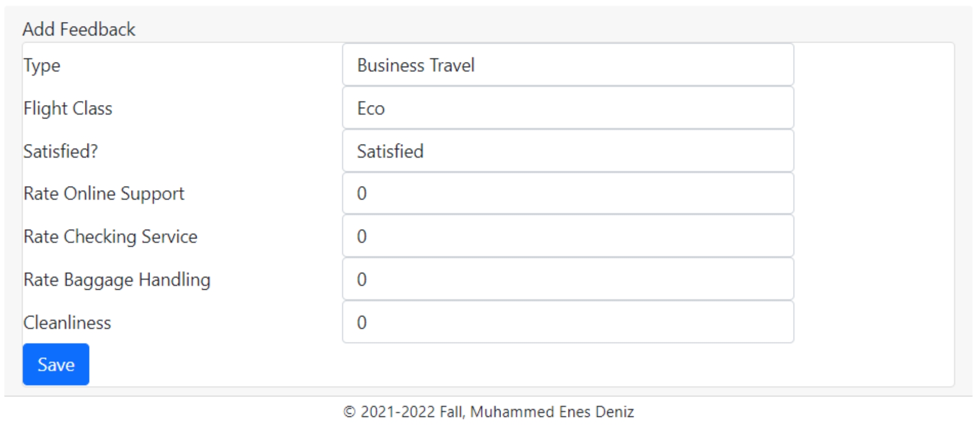
	Added by the User	Type of Travel	Class	Satisfaction	Online Support	Checkin Service	Baggage Handling	Cleanliness	
☐	admin	Business Travel	Eco	satisfied	5	5	5	5	Update
☐	admin	Business Travel	Eco	satisfied	1	0	1	0	Update
☐	admin	Business Travel	Eco	dissatisfied	0	0	0	0	Update

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 10: Feedback Page

In this page normal users have many priviliges. They are here for giving feedbacks about the airlines quality and enlighthen us. Users could filter dissatisfied users and satisfied by just filtering and having a general experience. Moreover they could delete their own previous comments but not the other people's.

2.10 Add Feedback Page



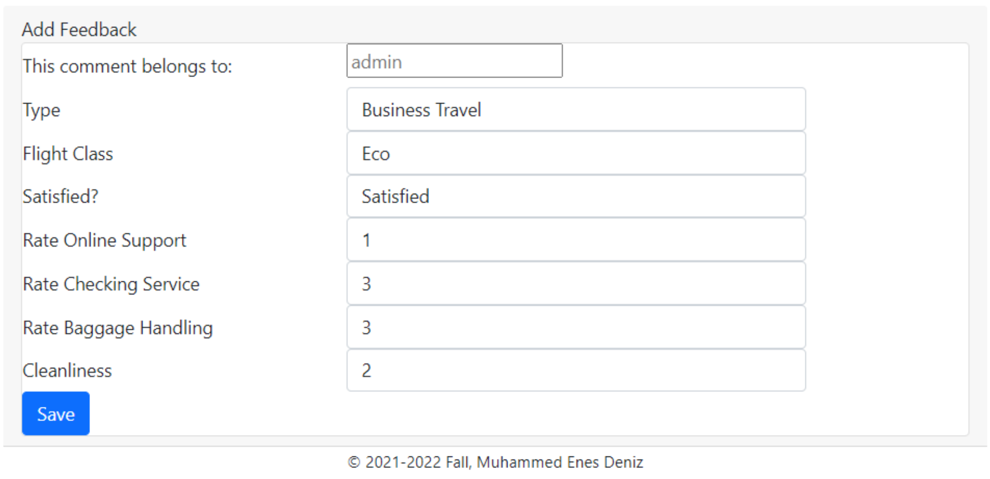
Type	Business Travel
Flight Class	Eco
Satisfied?	Satisfied
Rate Online Support	0
Rate Checking Service	0
Rate Baggage Handling	0
Cleanliness	0

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 11: Add Feedback Page

In this page every user could add new feedback to the database. It need to provide necessary informations that is asked by the cards. Validation exists. In the case of the invalid entry user is informed.

2.11 Update Feedback Page


Add Feedback

This comment belongs to:	admin
Type	Business Travel
Flight Class	Eco
Satisfied?	Satisfied
Rate Online Support	1
Rate Checking Service	3
Rate Baggage Handling	3
Cleanliness	2

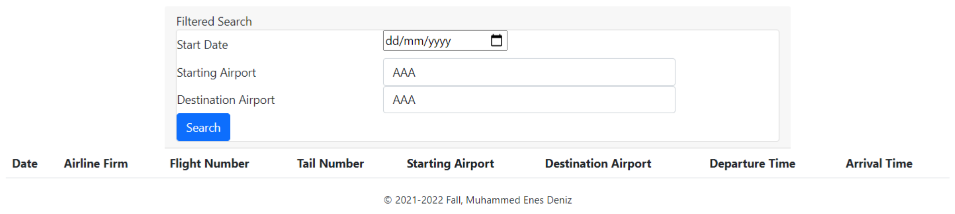
Save

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 12: Update Feedback Page

In this page every user could update their own feedback and reflect back it in the database. It need to provide necessary informations that is asked by the cards. Validation exists. In the case of the invalid entry user is informed.

2.12 Flights Page


Filtered Search

Start Date	dd/mm/yyyy
Starting Airport	AAA
Destination Airport	AAA

Search

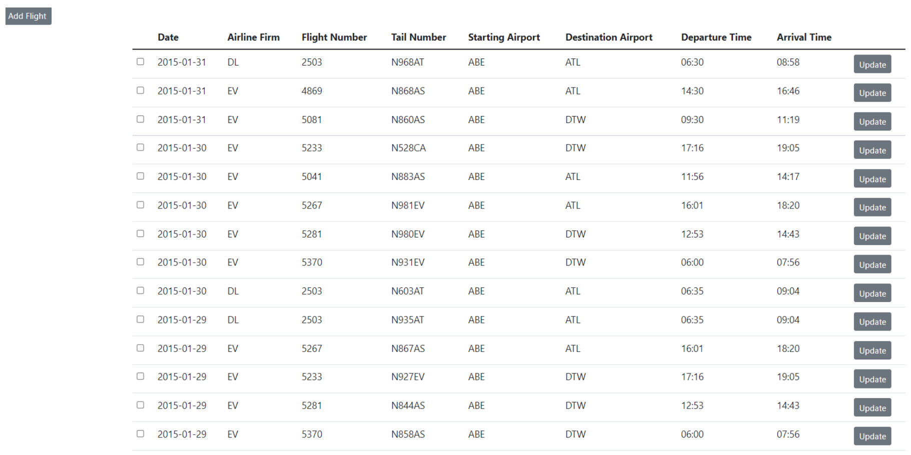
Date	Airline Firm	Flight Number	Tail Number	Starting Airport	Destination Airport	Departure Time	Arrival Time

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 13: Flights Page

In this page users could filter the flight that they want to observe in a day by also reffering the starting and destination airports. Moreover when there is not an entry exists in that time user is informed.

2.13 Flights Page By Airport

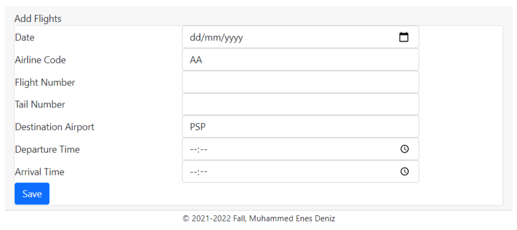


	Date	Airline Firm	Flight Number	Tail Number	Starting Airport	Destination Airport	Departure Time	Arrival Time	
☐	2015-01-31	DL	2503	N968AT	ABE	ATL	06:30	08:58	Update
☐	2015-01-31	EV	4869	N868AS	ABE	ATL	14:30	16:46	Update
☐	2015-01-31	EV	5081	N860AS	ABE	DTW	09:30	11:19	Update
☐	2015-01-30	EV	5233	N528CA	ABE	DTW	17:16	19:05	Update
☐	2015-01-30	EV	5041	N883AS	ABE	ATL	11:56	14:17	Update
☐	2015-01-30	EV	5267	N981EV	ABE	ATL	16:01	18:20	Update
☐	2015-01-30	EV	5281	N980EV	ABE	DTW	12:53	14:43	Update
☐	2015-01-30	EV	5370	N931EV	ABE	DTW	06:00	07:56	Update
☐	2015-01-30	DL	2503	N603AT	ABE	ATL	06:35	09:04	Update
☐	2015-01-29	DL	2503	N935AT	ABE	ATL	06:35	09:04	Update
☐	2015-01-29	EV	5267	N867AS	ABE	ATL	16:01	18:20	Update
☐	2015-01-29	EV	5233	N927EV	ABE	DTW	17:16	19:05	Update
☐	2015-01-29	EV	5281	N844AS	ABE	DTW	12:53	14:43	Update
☐	2015-01-29	EV	5370	N858AS	ABE	DTW	06:00	07:56	Update

Figure 14: Flights Page By Airport

In this page normal users only could read no further action could be taken carried forward. If the user is admin it could delete an listed flights from the table. Admin also have the privilige to add new records and update existing ones.

2.14 Add Flights Page



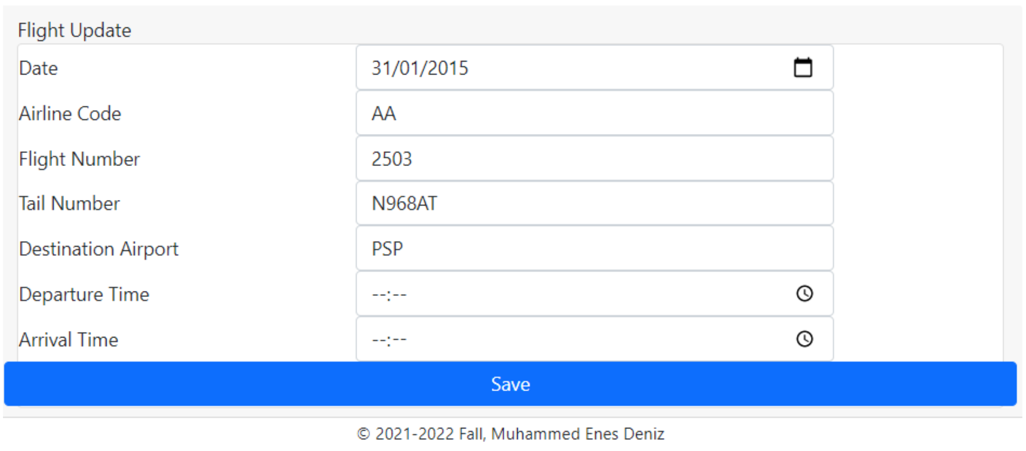
Date	dd/mm/yyyy
Airline Code	AA
Flight Number	
Tail Number	
Destination Airport	PSP
Departure Time	--:--
Arrival Time	--:--

© 2021-2022 Fall, Muhammed Enes Deniz

Figure 15: Add Flights Page

In this page the admin could add new flight to the database. It needs to provide the necessary information that is asked. If it mistakenly tries to enter a flight to the airport that he/she operates right now warning will be given.

2.15 Update Flights Page


Flight Update

Date	31/01/2015
Airline Code	AA
Flight Number	2503
Tail Number	N968AT
Destination Airport	PSP
Departure Time	--:--
Arrival Time	--:--

Figure 16: Update Flights Page

In this page the admin could update the existing flights in the database. Must remind that it is updating the flight that is departure from a airline so it could not have authority to update that information.

3 Developer Guide

First, it was tried to find a dataset with the desired features, then efforts were made to reveal the most meaningful application possible from this dataset. Two separate user types have been defined, and admin users have access to all tables, while normal users have the right to add, delete and update only with the feedback table.

The Airlines table and the Flights table were connected, thus indicating which airline the relevant flight was from.

The Airports and Flights table are linked to each other so that the departure and arrival points have a certain integrity.

User table, feedback table and airline tables are associated with each other. In this way, it is possible for normal users to give basic feedback about the relevant airline.

introduce tables

3.1 Airlines Table

Consists of 2 non-key fields. The fields are ticker and airline name. Ticker is foreign key for flights and feedbacks table. Only admins can operate on this other types of users only read from this table.

3.2 Airports Table

Consists of 7 non-key fields. The fields are Airport Name, City, State, Country, Latitude, Longitude and Airport Code. Airport code is foreign key for flights table it provides relevant information about which flight that departured from where and where it is heading to. Only admins can operate on this other types of users only read from this table.

3.3 Flights Table

Consists of 5 non-key fields. The fields are Date, Flight Number, Tail Number, Scheduled Departure, Scheduled Arrival. From this information we have grasps that the time of the flight and where it headed to. Moreover this table has 3 foreign key 1 of them comes from airlines table and one comes from the airports table. Only admins can operate on this other types of users only read from this table.

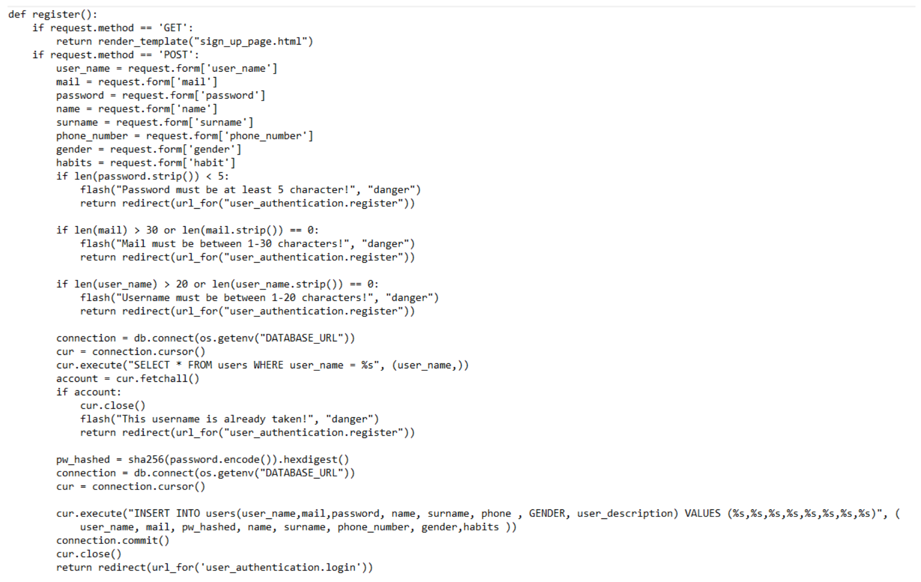
3.4 Feedback Table

Consists of 7 non-key fields. The fields are Type, Class, Satisfaction, Online Support, Checking Service, Baggage Handling and Cleanliness. Users could provide their experience of the airlines that they have beign customer of them. This table has 2 for foreign key 1 of it comes from users table and the other one comes from the airlines.

3.5 Users Table

Consists of 9 non-key fields. The fields are User Name, Mail, Password, Name, Surname, Phone, Gender, User Description and the key depicts whether user is admin or not. Admin privilege could only given by the database administrator no other user have access to that side.

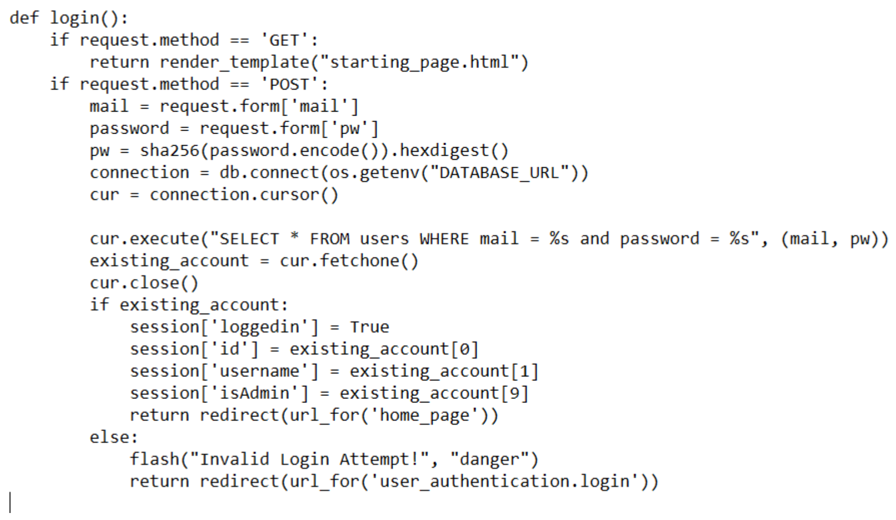
3.6 Authentication



```
def register():
    if request.method == 'GET':
        return render_template("sign_up_page.html")
    if request.method == 'POST':
        user_name = request.form['user_name']
        mail = request.form['mail']
        password = request.form['password']
        name = request.form['name']
        surname = request.form['surname']
        phone_number = request.form['phone_number']
        gender = request.form['gender']
        habits = request.form['habit']
        if len(password.strip()) < 5:
            flash("Password must be at least 5 character!", "danger")
            return redirect(url_for("user_authentication.register"))

        if len(mail) > 30 or len(mail.strip()) == 0:
            flash("Mail must be between 1-30 characters!", "danger")
            return redirect(url_for("user_authentication.register"))

        if len(user_name) > 20 or len(user_name.strip()) == 0:
            flash("Username must be between 1-20 characters!", "danger")
            return redirect(url_for("user_authentication.register"))

        connection = db.connect(os.getenv("DATABASE_URL"))
        cur = connection.cursor()
        cur.execute("SELECT * FROM users WHERE user_name = %s", (user_name,))
        account = cur.fetchall()
        if account:
            cur.close()
            flash("This username is already taken!", "danger")
            return redirect(url_for("user_authentication.register"))

        pw_hashed = sha256(password.encode()).hexdigest()
        connection = db.connect(os.getenv("DATABASE_URL"))
        cur = connection.cursor()

        cur.execute("INSERT INTO users(user_name,mail,password, name, surname, phone , GENDER, user_description) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)", (
            user_name, mail, pw_hashed, name, surname, phone_number, gender,habits ))
        connection.commit()
        cur.close()
        return redirect(url_for('user_authentication.login'))
```

Figure 17: Register Page

Users fill the request form and their response comes to the backend. In backend necessary validations performed and password hashed to store. After that if there is no obstacle to create such a user user created.

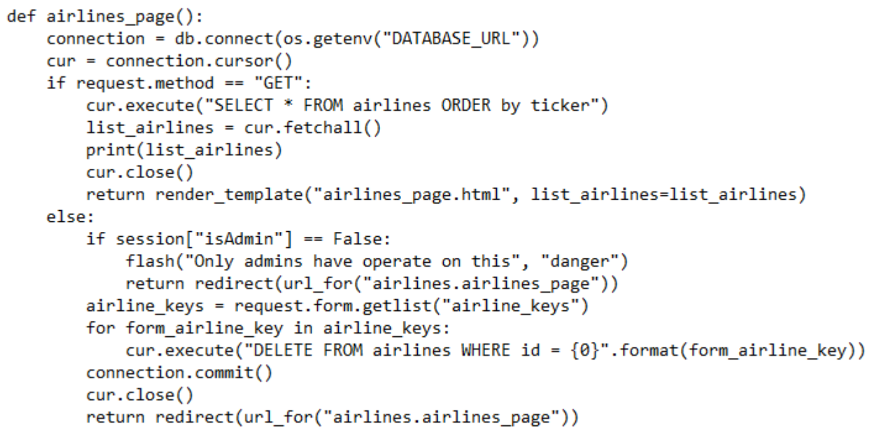


```
def login():
    if request.method == 'GET':
        return render_template("starting_page.html")
    if request.method == 'POST':
        mail = request.form['mail']
        password = request.form['pw']
        pw = sha256(password.encode()).hexdigest()
        connection = db.connect(os.getenv("DATABASE_URL"))
        cur = connection.cursor()

        cur.execute("SELECT * FROM users WHERE mail = %s and password = %s", (mail, pw))
        existing_account = cur.fetchone()
        cur.close()
        if existing_account:
            session['loggedin'] = True
            session['id'] = existing_account[0]
            session['username'] = existing_account[1]
            session['isAdmin'] = existing_account[9]
            return redirect(url_for('home_page'))
        else:
            flash("Invalid Login Attempt!", "danger")
            return redirect(url_for('user_authentication.login'))
```

Figure 18: Update Flights Page

Users enter their mail and password. If they enter invalid entry they will inform with the proper message.

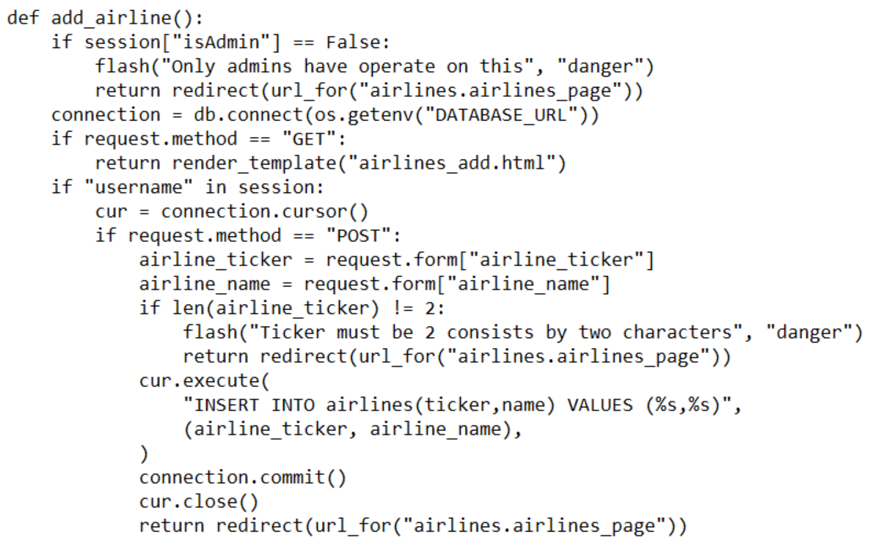
3.7 Airlines Operation



```python
def airlines_page():
    connection = db.connect(os.getenv("DATABASE_URL"))
    cur = connection.cursor()
    if request.method == "GET":
        cur.execute("SELECT * FROM airlines ORDER by ticker")
        list_airlines = cur.fetchall()
        print(list_airlines)
        cur.close()
        return render_template("airlines_page.html", list_airlines=list_airlines)
    else:
        if session["isAdmin"] == False:
            flash("Only admins have operate on this", "danger")
            return redirect(url_for("airlines.airlines_page"))
        airline_keys = request.form.getlist("airline_keys")
        for form_airline_key in airline_keys:
            cur.execute("DELETE FROM airlines WHERE id = {0}".format(form_airline_key))
        connection.commit()
        cur.close()
        return redirect(url_for("airlines.airlines_page"))
```

Figure 19: Airline Delete and Read

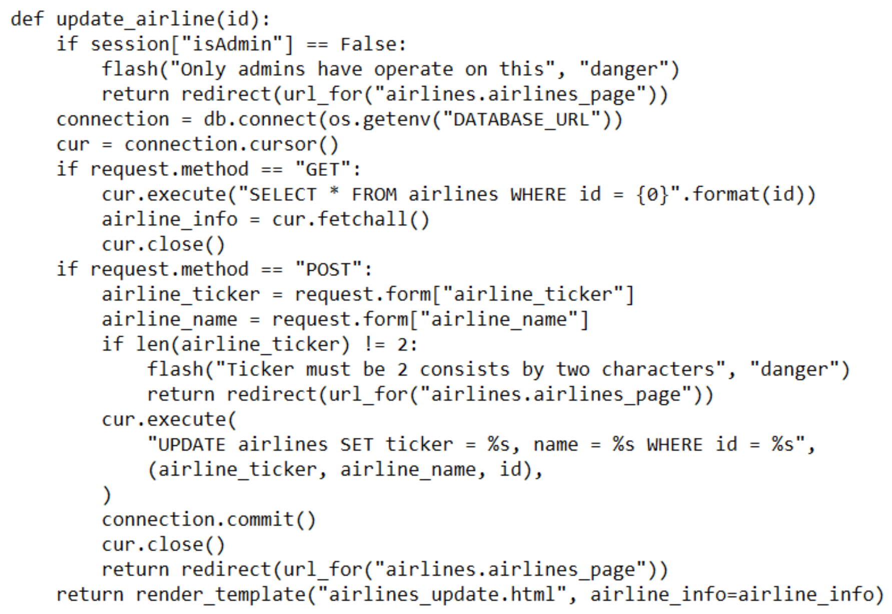
In the respective page admin users could see the check boxes at the entries and if they wish the delete an entry they will checkmark all of the entries and delete operation will be executed.



```python
def add_airline():
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("airlines.airlines_page"))
    connection = db.connect(os.getenv("DATABASE_URL"))
    if request.method == "GET":
        return render_template("airlines_add.html")
    if "username" in session:
        cur = connection.cursor()
        if request.method == "POST":
            airline_ticker = request.form["airline_ticker"]
            airline_name = request.form["airline_name"]
            if len(airline_ticker) != 2:
                flash("Ticker must be 2 consists by two characters", "danger")
                return redirect(url_for("airlines.airlines_page"))
            cur.execute(
                "INSERT INTO airlines(ticker,name) VALUES (%s,%s)",
                (airline_ticker, airline_name),
            )
            connection.commit()
            cur.close()
            return redirect(url_for("airlines.airlines_page"))
```

Figure 20: Airline Insert

Admin users could insert new entries to the database and the validation exists for this operation. Airline ticker must be 2 character both frontend and backend validation had been executed for this case.

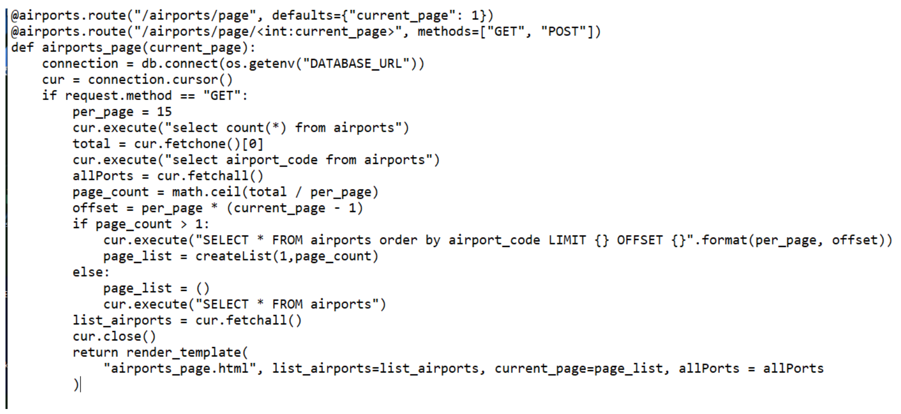


```
def update_airline(id):
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("airlines.airlines_page"))
    connection = db.connect(os.getenv("DATABASE_URL"))
    cur = connection.cursor()
    if request.method == "GET":
        cur.execute("SELECT * FROM airlines WHERE id = {0}".format(id))
        airline_info = cur.fetchall()
        cur.close()
    if request.method == "POST":
        airline_ticker = request.form["airline_ticker"]
        airline_name = request.form["airline_name"]
        if len(airline_ticker) != 2:
            flash("Ticker must be 2 consists by two characters", "danger")
            return redirect(url_for("airlines.airlines_page"))
        cur.execute(
            "UPDATE airlines SET ticker = %s, name = %s WHERE id = %s",
            (airline_ticker, airline_name, id),
        )
        connection.commit()
        cur.close()
        return redirect(url_for("airlines.airlines_page"))
    return render_template("airlines_update.html", airline_info=airline_info)
```

Figure 21: Airline Update

Admin users could update existing entries through the form and it reflects in the database and the validation exists for this operation.

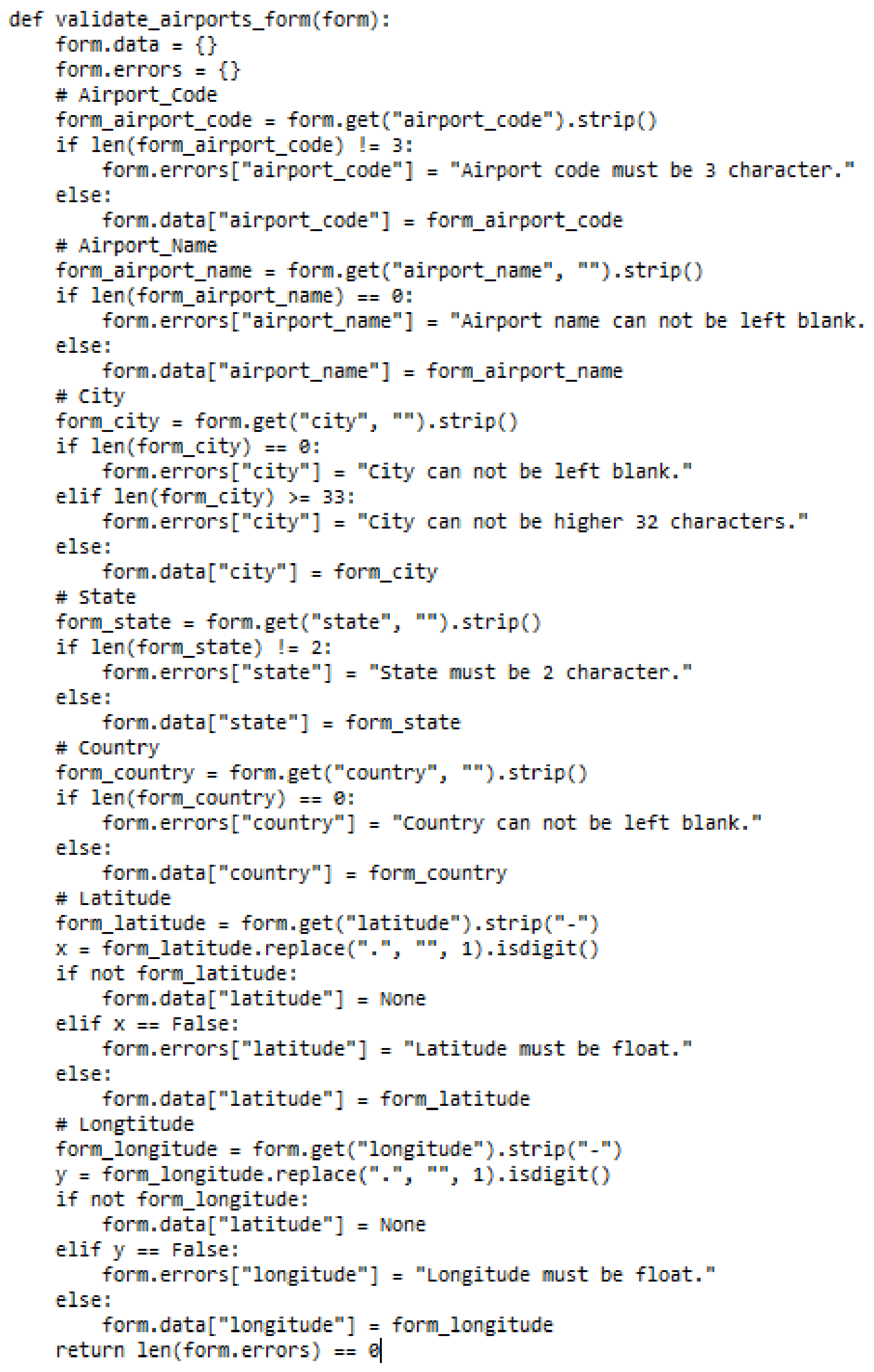
3.8 Airports Operation



```
@airports.route("/airports/page", defaults={"current_page": 1})
@airports.route("/airports/page/<int:current_page>", methods=["GET", "POST"])
def airports_page(current_page):
    connection = db.connect(os.getenv("DATABASE_URL"))
    cur = connection.cursor()
    if request.method == "GET":
        per_page = 15
        cur.execute("select count(*) from airports")
        total = cur.fetchone()[0]
        cur.execute("select airport_code from airports")
        allPorts = cur.fetchall()
        page_count = math.ceil(total / per_page)
        offset = per_page * (current_page - 1)
        if page_count > 1:
            cur.execute("SELECT * FROM airports order by airport_code LIMIT {} OFFSET {}".format(per_page, offset))
            page_list = createList(1,page_count)
        else:
            page_list = ()
            cur.execute("SELECT * FROM airports")
        list_airports = cur.fetchall()
        cur.close()
        return render_template(
            "airports_page.html", list_airports=list_airports, current_page=page_list, allPorts = allPorts
        )
```

Figure 22: Airports Read and Pagination

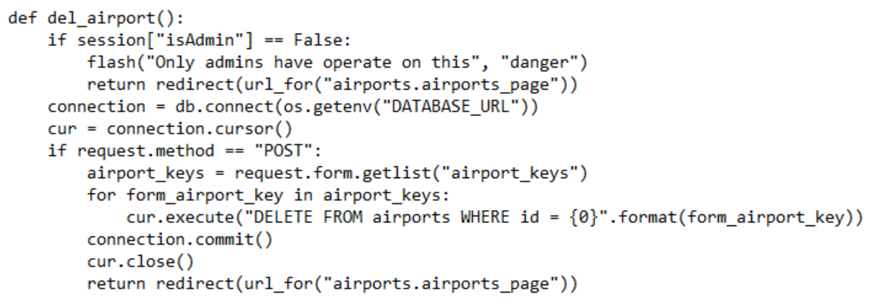
In the airports page there could be potential of huge entries thus, pagination needs to be implemented. In each page 15 entries exist and the select query had been written accordance to that.



```
def validate_airports_form(form):
    form.data = {}
    form.errors = {}
    # Airport_Code
    form_airport_code = form.get("airport_code").strip()
    if len(form_airport_code) != 3:
        form.errors["airport_code"] = "Airport code must be 3 character."
    else:
        form.data["airport_code"] = form_airport_code
    # Airport_Name
    form_airport_name = form.get("airport_name", "").strip()
    if len(form_airport_name) == 0:
        form.errors["airport_name"] = "Airport name can not be left blank.
    else:
        form.data["airport_name"] = form_airport_name
    # City
    form_city = form.get("city", "").strip()
    if len(form_city) == 0:
        form.errors["city"] = "City can not be left blank."
    elif len(form_city) >= 33:
        form.errors["city"] = "City can not be higher 32 characters."
    else:
        form.data["city"] = form_city
    # State
    form_state = form.get("state", "").strip()
    if len(form_state) != 2:
        form.errors["state"] = "State must be 2 character."
    else:
        form.data["state"] = form_state
    # Country
    form_country = form.get("country", "").strip()
    if len(form_country) == 0:
        form.errors["country"] = "Country can not be left blank."
    else:
        form.data["country"] = form_country
    # Latitude
    form_latitude = form.get("latitude").strip("-")
    x = form_latitude.replace(".", "", 1).isdigit()
    if not form_latitude:
        form.data["latitude"] = None
    elif x == False:
        form.errors["latitude"] = "Latitude must be float."
    else:
        form.data["latitude"] = form_latitude
    # Longtitude
    form_longitude = form.get("longitude").strip("-")
    y = form_longitude.replace(".", "", 1).isdigit()
    if not form_longitude:
        form.data["latitude"] = None
    elif y == False:
        form.errors["longitude"] = "Longitude must be float."
    else:
        form.data["longitude"] = form_longitude
    return len(form.errors) == 0
```

Figure 23: Airport Validate

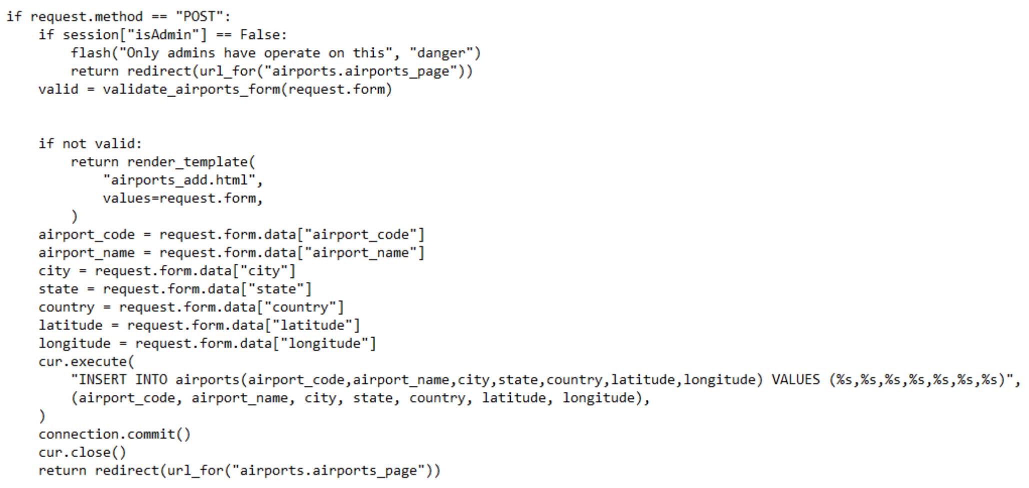
The form that makes the update and insert operations to the database is validated through all of it's inputs so it obeys the general integrity. Moreover in the frontend side minor preventetions had been taken in accordance to this policy.



```
def del_airport():
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("airports.airports_page"))
    connection = db.connect(os.getenv("DATABASE_URL"))
    cur = connection.cursor()
    if request.method == "POST":
        airport_keys = request.form.getlist("airport_keys")
        for form_airport_key in airport_keys:
            cur.execute("DELETE FROM airports WHERE id = {0}".format(form_airport_key))
        connection.commit()
        cur.close()
        return redirect(url_for("airports.airports_page"))
```

Figure 24: Delete Airport

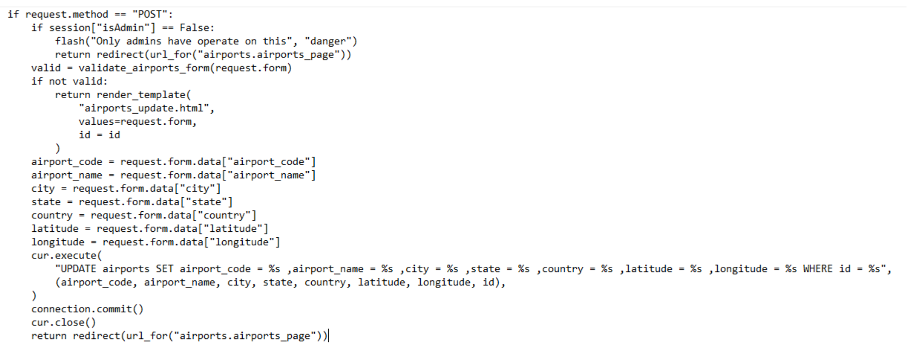
Admin users have the privilege to delete an existing airport entry from the data table.



```
if request.method == "POST":
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("airports.airports_page"))
    valid = validate_airports_form(request.form)


    if not valid:
        return render_template(
            "airports_add.html",
            values=request.form,
        )
    airport_code = request.form.data["airport_code"]
    airport_name = request.form.data["airport_name"]
    city = request.form.data["city"]
    state = request.form.data["state"]
    country = request.form.data["country"]
    latitude = request.form.data["latitude"]
    longitude = request.form.data["longitude"]
    cur.execute(
        "INSERT INTO airports(airport_code,airport_name,city,state,country,latitude,longitude) VALUES (%s,%s,%s,%s,%s,%s,%s)",
        (airport_code, airport_name, city, state, country, latitude, longitude),
    )
    connection.commit()
    cur.close()
    return redirect(url_for("airports.airports_page"))
```

Figure 25: Airports Add

Only admin user have the right to insert new airport to the database. From the validation airports the entries had been checked and if there is an error exists the entered entries and problem message re-delievered to the page.

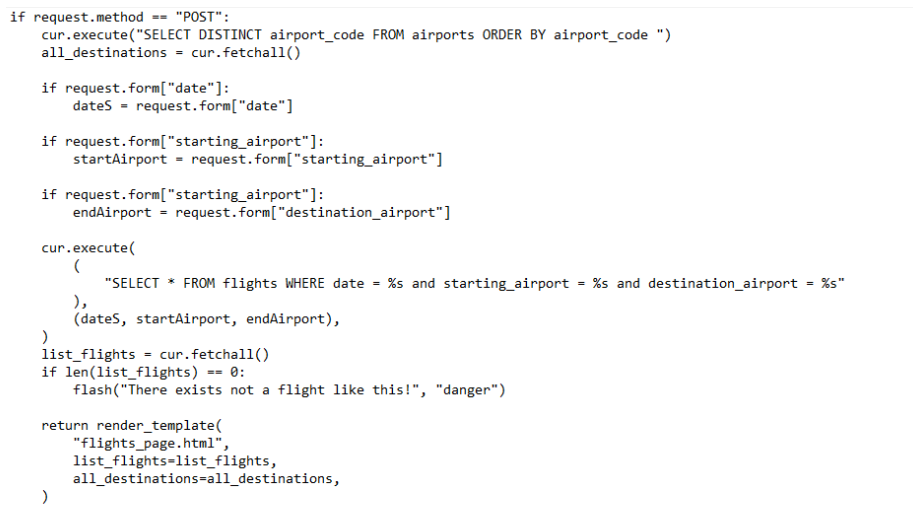


```
if request.method == "POST":
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("airports.airports_page"))
    valid = validate_airports_form(request.form)
    if not valid:
        return render_template(
            "airports_update.html",
            values=request.form,
            id = id
        )
    airport_code = request.form.data["airport_code"]
    airport_name = request.form.data["airport_name"]
    city = request.form.data["city"]
    state = request.form.data["state"]
    country = request.form.data["country"]
    latitude = request.form.data["latitude"]
    longitude = request.form.data["longitude"]
    cur.execute(
        "UPDATE airports SET airport_code = %s ,airport_name = %s ,city = %s ,state = %s ,country = %s ,latitude = %s ,longitude = %s WHERE id = %s",
        (airport_code, airport_name, city, state, country, latitude, longitude, id),
    )
    connection.commit()
    cur.close()
    return redirect(url_for("airports.airports_page"))
```

Figure 26: Airports Update

To update an existing airport you must have an admin privilige. Admins could update the whole information about the existing airport entry.

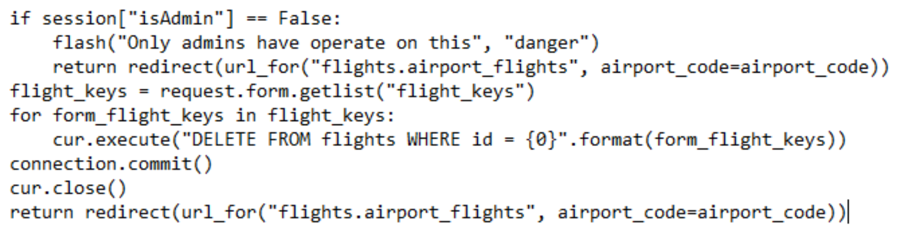
3.9 Flights Operation



```
if request.method == "POST":
    cur.execute("SELECT DISTINCT airport_code FROM airports ORDER BY airport_code ")
    all_destinations = cur.fetchall()

    if request.form["date"]:
        dateS = request.form["date"]

    if request.form["starting_airport"]:
        startAirport = request.form["starting_airport"]

    if request.form["starting_airport"]:
        endAirport = request.form["destination_airport"]

    cur.execute(
        (
            "SELECT * FROM flights WHERE date = %s and starting_airport = %s and destination_airport = %s"
        ),
        (dateS, startAirport, endAirport),
    )
    list_flights = cur.fetchall()
    if len(list_flights) == 0:
        flash("There exists not a flight like this!", "danger")

    return render_template(
        "flights_page.html",
        list_flights=list_flights,
        all_destinations=all_destinations,
    )
```

Figure 27: Filter and Search Flights

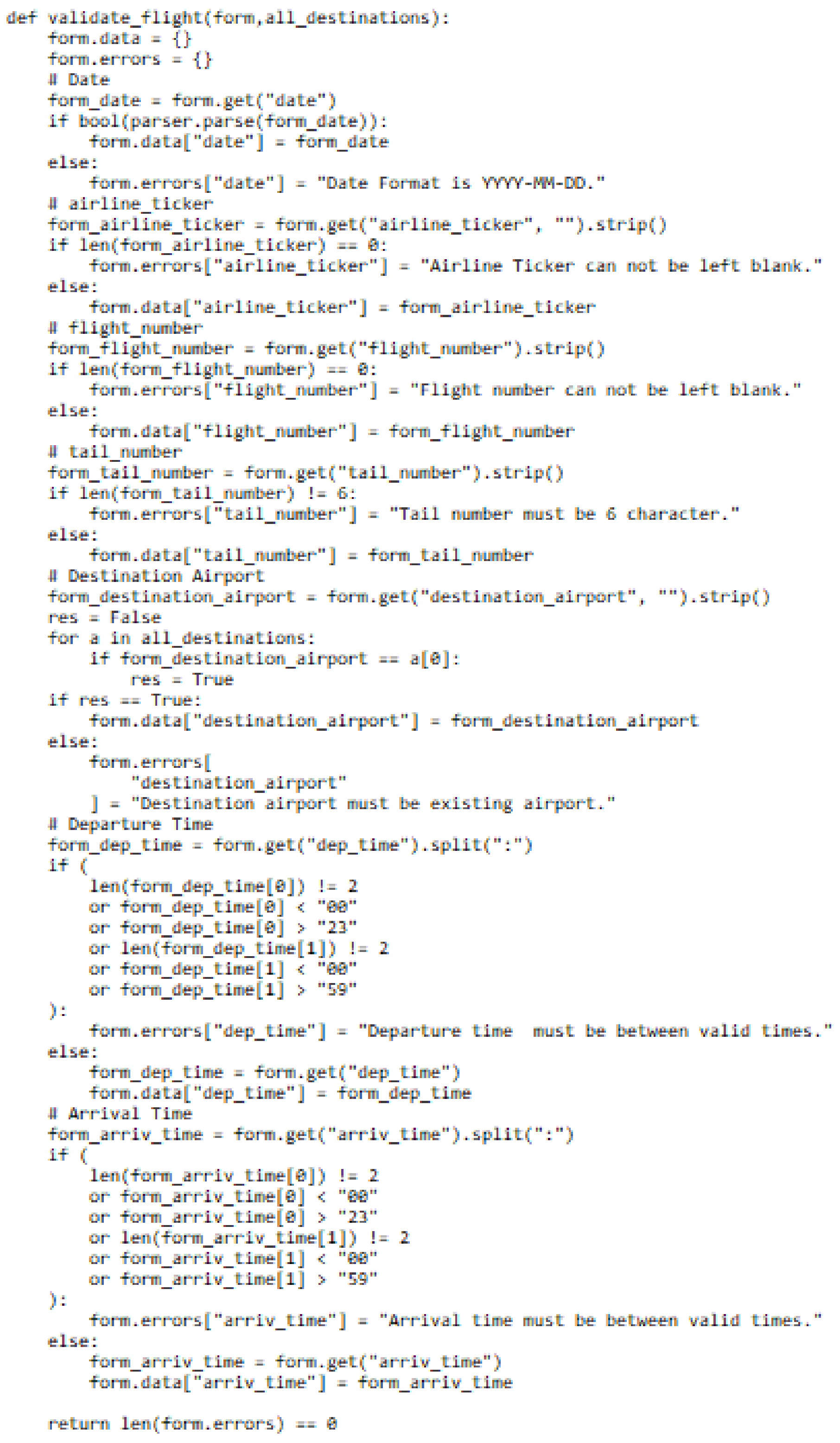
Users and viewers could search and filter flights from here by selecting appropriate date and starting and destination airport. If there exists not a flight like that user will informed about that.



```
if session["isAdmin"] == False:
    flash("Only admins have operate on this", "danger")
    return redirect(url_for("flights.airport_flights", airport_code=airport_code))
flight_keys = request.form.getlist("flight_keys")
for form_flight_keys in flight_keys:
    cur.execute("DELETE FROM flights WHERE id = {0}".format(form_flight_keys))
connection.commit()
cur.close()
return redirect(url_for("flights.airport_flights", airport_code=airport_code))
```

Figure 28: Delete existing Flight

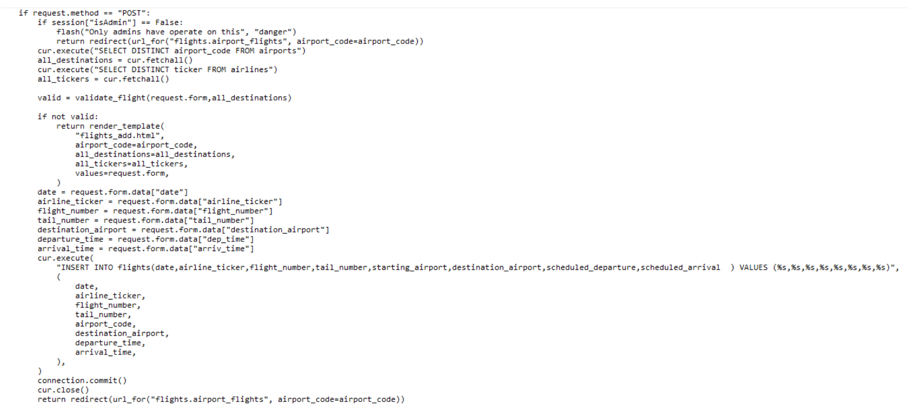
Only admin could operate on this page and it could delete the flights that it find redundant.



```python
def validate_flight(form,all_destinations):
    form.data = {}
    form.errors = {}
    # Date
    form_date = form.get("date")
    if bool(parser.parse(form_date)):
        form.data["date"] = form_date
    else:
        form.errors["date"] = "Date Format is YYYY-MM-DD."
    # airline_ticker
    form_airline_ticker = form.get("airline_ticker", "").strip()
    if len(form_airline_ticker) == 0:
        form.errors["airline_ticker"] = "Airline Ticker can not be left blank."
    else:
        form.data["airline_ticker"] = form_airline_ticker
    # flight_number
    form_flight_number = form.get("flight_number").strip()
    if len(form_flight_number) == 0:
        form.errors["flight_number"] = "Flight number can not be left blank."
    else:
        form.data["flight_number"] = form_flight_number
    # tail_number
    form_tail_number = form.get("tail_number").strip()
    if len(form_tail_number) != 6:
        form.errors["tail_number"] = "Tail number must be 6 character."
    else:
        form.data["tail_number"] = form_tail_number
    # Destination Airport
    form_destination_airport = form.get("destination_airport", "").strip()
    res = False
    for a in all_destinations:
        if form_destination_airport == a[0]:
            res = True
    if res == True:
        form.data["destination_airport"] = form_destination_airport
    else:
        form.errors[
            "destination_airport"
        ] = "Destination airport must be existing airport."
    # Departure Time
    form_dep_time = form.get("dep_time").split(":")
    if (
        len(form_dep_time[0]) != 2
        or form_dep_time[0] < "00"
        or form_dep_time[0] > "23"
        or len(form_dep_time[1]) != 2
        or form_dep_time[1] < "00"
        or form_dep_time[1] > "59"
    ):
        form.errors["dep_time"] = "Departure time  must be between valid times."
    else:
        form_dep_time = form.get("dep_time")
        form.data["dep_time"] = form_dep_time
    # Arrival Time
    form_arriv_time = form.get("arriv_time").split(":")
    if (
        len(form_arriv_time[0]) != 2
        or form_arriv_time[0] < "00"
        or form_arriv_time[0] > "23"
        or len(form_arriv_time[1]) != 2
        or form_arriv_time[1] < "00"
        or form_arriv_time[1] > "59"
    ):
        form.errors["arriv_time"] = "Arrival time must be between valid times."
    else:
        form_arriv_time = form.get("arriv_time")
        form.data["arriv_time"] = form_arriv_time

    return len(form.errors) == 0
```

Figure 29: Validation of Flights Form

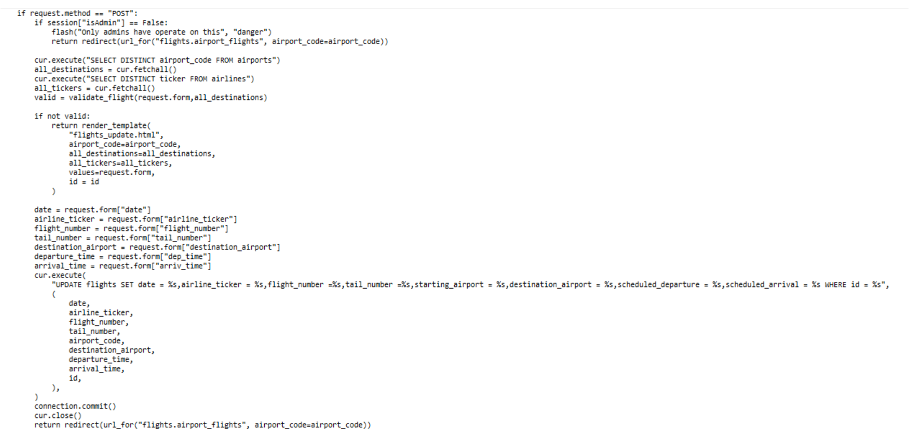
Flights form validation exists in here. Date and time validation had been done and admin could not destinate the current and destination airports as the same



```python
if request.method == "POST":
    if session["isAdmin"] == False:
        flash("Only admins have operate on this", "danger")
        return redirect(url_for("flights.airport_flights", airport_code=airport_code))
    cur.execute("SELECT DISTINCT airport_code FROM airports")
    all_destinations = cur.fetchall()
    cur.execute("SELECT DISTINCT ticker FROM airlines")
    all_tickers = cur.fetchall()

    valid = validate_flight(request.form,all_destinations)

    if not valid:
        return render_template(
            "flights_add.html",
            airport_code=airport_code,
            all_destinations=all_destinations,
            all_tickers=all_tickers,
            values=request.form,
        )
    date = request.form.data["date"]
    airline_ticker = request.form.data["airline_ticker"]
    flight_number = request.form.data["flight_number"]
    tail_number = request.form.data["tail_number"]
    destination_airport = request.form.data["destination_airport"]
    departure_time = request.form.data["dep_time"]
    arrival_time = request.form.data["arriv_time"]
    cur.execute(
        "INSERT INTO flights(date,airline_ticker,flight_number,tail_number,starting_airport,destination_airport,scheduled_departure,scheduled_arrival  ) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)",
        (
            date,
            airline_ticker,
            flight_number,
            tail_number,
            airport_code,
            destination_airport,
            departure_time,
            arrival_time,
        ),
    )
    connection.commit()
    cur.close()
    return redirect(url_for("flights.airport_flights", airport_code=airport_code))
```

Figure 30: Add Flights

Admin could insert new flight from the specific airports flights page. Flights could only inserted from the airport page. Validation exists.

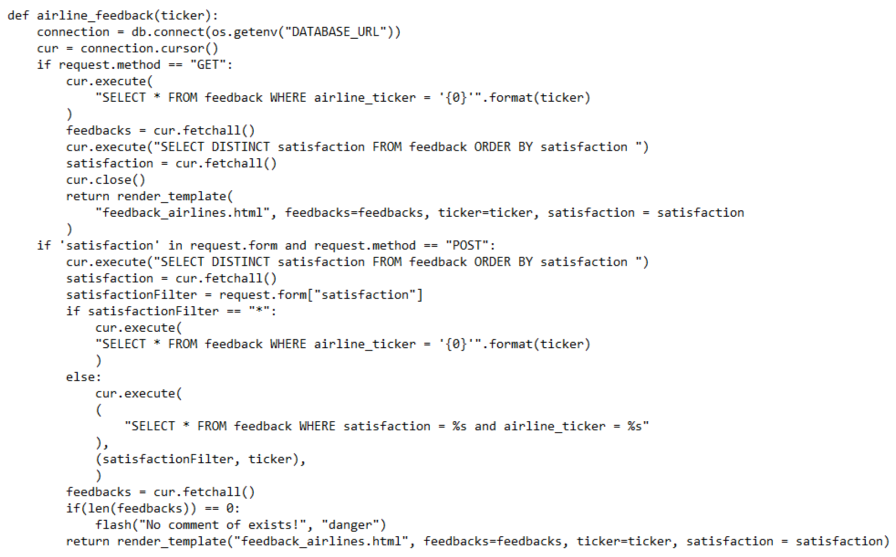


```python
if request.method == "POST":
   if session["isAdmin"] == False:
       flash("Only admins have operate on this", "danger")
       return redirect(url_for("flights.airport_flights", airport_code=airport_code))

   cur.execute("SELECT DISTINCT airport_code FROM airports")
   all_destinations = cur.fetchall()
   cur.execute("SELECT DISTINCT ticker FROM airlines")
   all_tickers = cur.fetchall()
   valid = validate_flight(request.form,all_destinations)

   if not valid:
       return render_template(
           "flights_update.html",
           airport_code=airport_code,
           all_destinations=all_destinations,
           all_tickers=all_tickers,
           values=request.form,
           id = id
       )

   date = request.form["date"]
   airline_ticker = request.form["airline_ticker"]
   flight_number = request.form["flight_number"]
   tail_number = request.form["tail_number"]
   destination_airport = request.form["destination_airport"]
   departure_time = request.form["dep_time"]
   arrival_time = request.form["arriv_time"]
   cur.execute(
       "UPDATE flights SET date = %s,airline_ticker = %s,flight_number =%s,tail_number =%s,starting_airport = %s,destination_airport = %s,scheduled_departure = %s,scheduled_arrival = %s WHERE id = %s",
       (
           date,
           airline_ticker,
           flight_number,
           tail_number,
           airport_code,
           destination_airport,
           departure_time,
           arrival_time,
           id,
       ),
   )
   connection.commit()
   cur.close()
   return redirect(url_for("flights.airport_flights", airport_code=airport_code))
```

Figure 31: Update Flights

If there exists some schedule updates or necessary change in the destination flights could be updated from here.

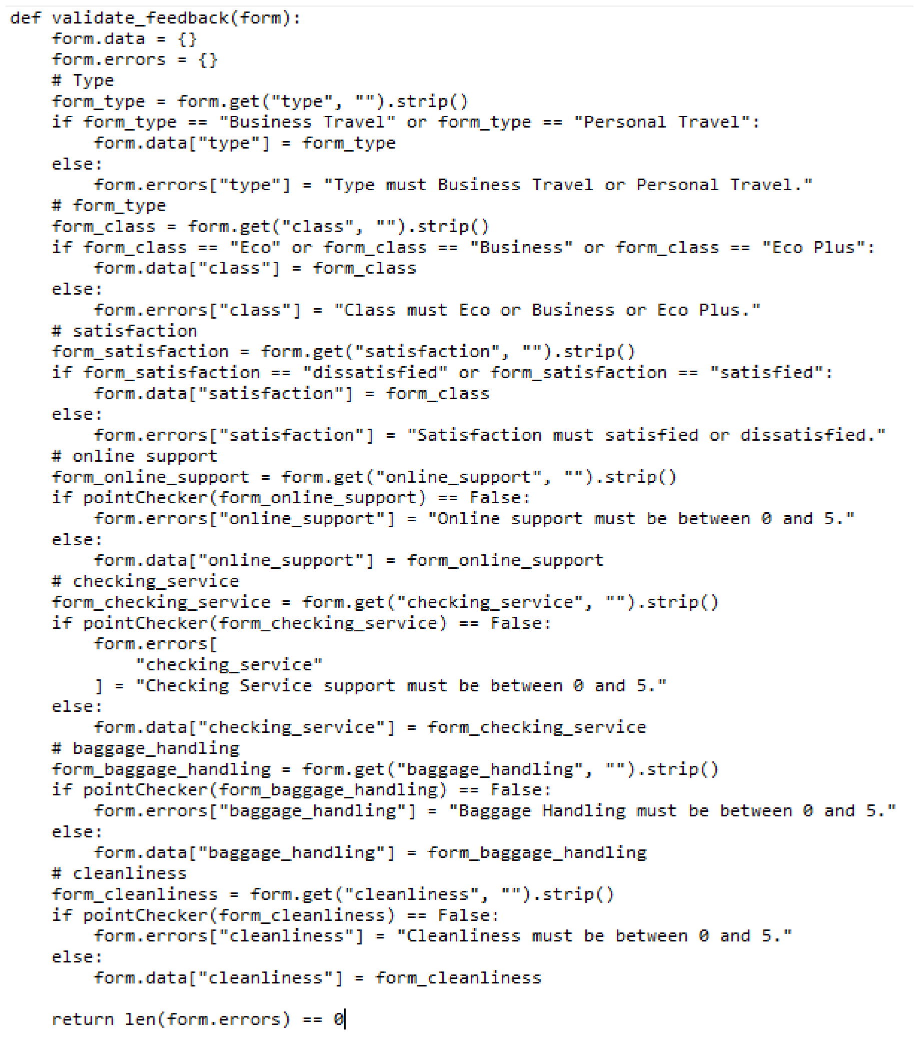
3.10 Feedback Operation



```
def airline_feedback(ticker):
    connection = db.connect(os.getenv("DATABASE_URL"))
    cur = connection.cursor()
    if request.method == "GET":
        cur.execute(
            "SELECT * FROM feedback WHERE airline_ticker = '{0}'".format(ticker)
        )
        feedbacks = cur.fetchall()
        cur.execute("SELECT DISTINCT satisfaction FROM feedback ORDER BY satisfaction ")
        satisfaction = cur.fetchall()
        cur.close()
        return render_template(
            "feedback_airlines.html", feedbacks=feedbacks, ticker=ticker, satisfaction = satisfaction
        )
    if 'satisfaction' in request.form and request.method == "POST":
        cur.execute("SELECT DISTINCT satisfaction FROM feedback ORDER BY satisfaction ")
        satisfaction = cur.fetchall()
        satisfactionFilter = request.form["satisfaction"]
        if satisfactionFilter == "*":
            cur.execute(
            "SELECT * FROM feedback WHERE airline_ticker = '{0}'".format(ticker)
            )
        else:
            cur.execute(
            (
                "SELECT * FROM feedback WHERE satisfaction = %s and airline_ticker = %s"
            ),
            (satisfactionFilter, ticker),
            )
        feedbacks = cur.fetchall()
        if(len(feedbacks)) == 0:
            flash("No comment of exists!", "danger")
        return render_template("feedback_airlines.html", feedbacks=feedbacks, ticker=ticker, satisfaction = satisfaction)
```

Figure 32: Filtered Search Feedback

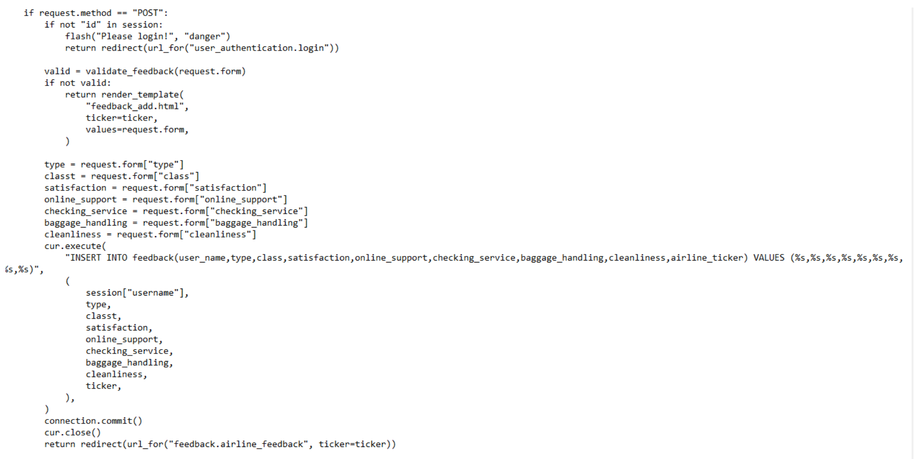
Users could fliter the feedbacks by the satisfaction and observe the others experience of the same airlines. Moreover the guests could grasps some knowledge about that specific airline.



```
def validate_feedback(form):
    form.data = {}
    form.errors = {}
    # Type
    form_type = form.get("type", "").strip()
    if form_type == "Business Travel" or form_type == "Personal Travel":
        form.data["type"] = form_type
    else:
        form.errors["type"] = "Type must Business Travel or Personal Travel."
    # form_type
    form_class = form.get("class", "").strip()
    if form_class == "Eco" or form_class == "Business" or form_class == "Eco Plus":
        form.data["class"] = form_class
    else:
        form.errors["class"] = "Class must Eco or Business or Eco Plus."
    # satisfaction
    form_satisfaction = form.get("satisfaction", "").strip()
    if form_satisfaction == "dissatisfied" or form_satisfaction == "satisfied":
        form.data["satisfaction"] = form_class
    else:
        form.errors["satisfaction"] = "Satisfaction must satisfied or dissatisfied."
    # online support
    form_online_support = form.get("online_support", "").strip()
    if pointChecker(form_online_support) == False:
        form.errors["online_support"] = "Online support must be between 0 and 5."
    else:
        form.data["online_support"] = form_online_support
    # checking_service
    form_checking_service = form.get("checking_service", "").strip()
    if pointChecker(form_checking_service) == False:
        form.errors[
            "checking_service"
        ] = "Checking Service support must be between 0 and 5."
    else:
        form.data["checking_service"] = form_checking_service
    # baggage_handling
    form_baggage_handling = form.get("baggage_handling", "").strip()
    if pointChecker(form_baggage_handling) == False:
        form.errors["baggage_handling"] = "Baggage Handling must be between 0 and 5."
    else:
        form.data["baggage_handling"] = form_baggage_handling
    # cleanliness
    form_cleanliness = form.get("cleanliness", "").strip()
    if pointChecker(form_cleanliness) == False:
        form.errors["cleanliness"] = "Cleanliness must be between 0 and 5."
    else:
        form.data["cleanliness"] = form_cleanliness

    return len(form.errors) == 0
```

Figure 33: Feedback Validation

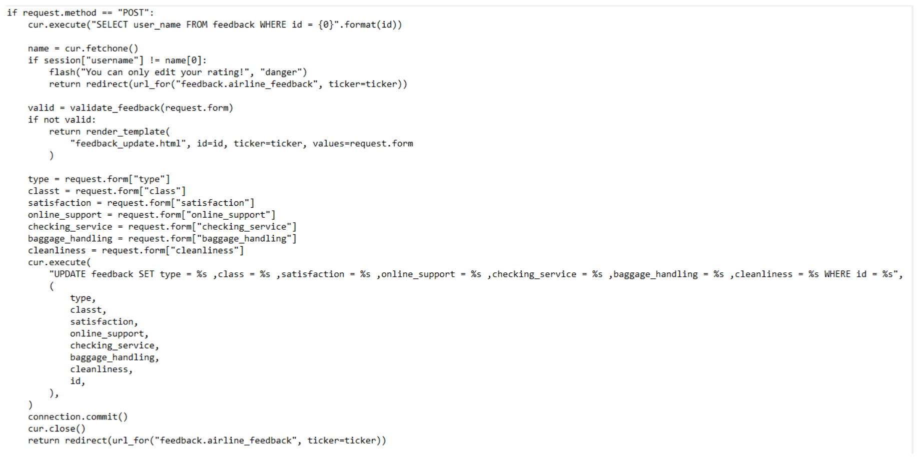
Validation of the feedback had been executed. Only the user could delete it's own comments nobody could delete it's comment.



```
    if request.method == "POST":
        if not "id" in session:
            flash("Please login!", "danger")
            return redirect(url_for("user_authentication.login"))

        valid = validate_feedback(request.form)
        if not valid:
            return render_template(
                "feedback_add.html",
                ticker=ticker,
                values=request.form,
            )

        type = request.form["type"]
        classt = request.form["class"]
        satisfaction = request.form["satisfaction"]
        online_support = request.form["online_support"]
        checking_service = request.form["checking_service"]
        baggage_handling = request.form["baggage_handling"]
        cleanliness = request.form["cleanliness"]
        cur.execute(
            "INSERT INTO feedback(user_name,type,class,satisfaction,online_support,checking_service,baggage_handling,cleanliness,airline_ticker) VALUES (%s,%s,%s,%s,%s,%s,%s,
%s,%s)",
            (
                session["username"],
                type,
                classt,
                satisfaction,
                online_support,
                checking_service,
                baggage_handling,
                cleanliness,
                ticker,
            ),
        )
        connection.commit()
        cur.close()
        return redirect(url_for("feedback.airline_feedback", ticker=ticker))
```

Figure 34: Feedback Add

Users could only insert one comment to the specific airlines no more comment from the existing user to that airport is excluded. If it’s want to change it’s view he/she could use the update.



```python
if request.method == "POST":
    cur.execute("SELECT user_name FROM feedback WHERE id = {0}".format(id))

    name = cur.fetchone()
    if session["username"] != name[0]:
        flash("You can only edit your rating!", "danger")
        return redirect(url_for("feedback.airline_feedback", ticker=ticker))

    valid = validate_feedback(request.form)
    if not valid:
        return render_template(
            "feedback_update.html", id=id, ticker=ticker, values=request.form
        )

    type = request.form["type"]
    classt = request.form["class"]
    satisfaction = request.form["satisfaction"]
    online_support = request.form["online_support"]
    checking_service = request.form["checking_service"]
    baggage_handling = request.form["baggage_handling"]
    cleanliness = request.form["cleanliness"]
    cur.execute(
        "UPDATE feedback SET type = %s ,class = %s ,satisfaction = %s ,online_support = %s ,checking_service = %s ,baggage_handling = %s ,cleanliness = %s WHERE id = %s",
        (
            type,
            classt,
            satisfaction,
            online_support,
            checking_service,
            baggage_handling,
            cleanliness,
            id,
        ),
    )
    connection.commit()
    cur.close()
    return redirect(url_for("feedback.airline_feedback", ticker=ticker))
```

Figure 35: Feedback Update

Since the users could only add a one feedback to the specific airline their view might be changed so they have to chance to update their rating. They could not change other people’s ratings.