ISYS2120 Sem2, 2024

Assignment 3 Individual Report

SID:

530157791

(Do not edit the format of this cover sheet!)

Assignment 3 Individual Report

`routes.py`

```
## Manage airports (add, update, delete, fetch) with pagination.
@app.route('/airports/manage', methods=['GET', 'POST'])
def manage_airports():
    message = None
    message_type = "success"

    fetched_airport = None
    form_type = None

    page_num = request.args.get('page', 1, type=int)
    per_page = 10

    all_airports = database.get_all_airports()

    add_form = AddAirportForm()

    if request.method == 'POST':
        if 'add_airport' in request.form:
            form_type = 'add'
            if add_form.validate_on_submit():
                airportid = add_form.airportid.data
                name = add_form.name.data
                iatacode = add_form.iatacode.data
                city = add_form.city.data
                country = add_form.country.data

                if not session.get('isadmin', False):
                    message = 'You are not admin'
                    message_type = "error"
                else:
                    result = database.add_airport(airportid, name, iatacode,
city, country)
                    if result:
                        message = 'Airport added successfully!'
                        message_type = "success"
                    else:
                        message = 'Error adding airport.'
                        message_type = "error"
            else:
```

```
                form_type = 'add'
                message = '表单验证失败，请检查输入。'
                message_type = "error"

        elif 'update_airport' in request.form:
            form_type = 'update'
            airportid = request.form.get('airportid')
            name = request.form.get('name')
            iatacode = request.form.get('iatacode')
            city = request.form.get('city')
            country = request.form.get('country')

            if not session.get('isadmin', False):
                message = 'You are not admin'
                message_type = "error"

            elif not (airportid and name and iatacode and city and country):
                message = 'All fields are required to update an airport.'
                message_type = "error"

            else:
                result = database.update_airport(airportid, name, iatacode,
city, country)
                if result:
                    message = 'Airport updated successfully!'
                    message_type = "success"
                else:
                    message = 'Error updating airport.'
                    message_type = "error"  # Set message type to error

        elif 'delete_airport' in request.form:
            form_type = 'delete'
            airportid = request.form.get('airportid')

            if not session.get('isadmin', False):
                message = 'You are not admin'
                message_type = "error"
            elif not airportid:
                message = 'Airport ID is required to delete an airport.'
                message_type = "error"  # Set message type to error
            else:
                result = database.delete_airport(airportid)
                if result == True:
                    message = f'Airport with ID {airportid} deleted
successfully!'
                    message_type = "success"
```

```
                elif result == False:
                    message = f'Error deleting airport. Airport ID
{airportid} does not exist.'
                    message_type = "error"
                else:
                    message = f'An error occurred while deleting airport
with ID {airportid}.'
                    message_type = "error"

        elif 'fetch_airport' in request.form:
            form_type = 'fetch'
            fetch_airportid = request.form.get('fetch_airportid')
            if not fetch_airportid:
                message = 'Airport ID is required to fetch airport
information.'
                message_type = "error"  # Set message type to error
            else:
                fetched_airport =
database.get_airport_by_id(fetch_airportid)
                if fetched_airport:
                    message = 'Airport information fetched successfully.'
                    message_type = "success"
                else:
                    message = f'Airport with ID {fetch_airportid} not
found.'
                    message_type = "error"  # Set message type to error

    airports_list = database.get_airports_paginated(page_num, per_page)
    total_airports = database.count_airports()

    if airports_list is None:
        airports_list = []
        message = 'No airports found.'
        message_type = "error"

    total_pages = ceil(total_airports / per_page) if total_airports else 1

    start_page = max(page_num - 2, 1)
    end_page = min(start_page + 4, total_pages)
    start_page = max(end_page - 4, 1)

    prev_page = page_num - 1 if page_num > 1 else None
    next_page = page_num + 1 if page_num < total_pages else None

    page_info = {
        'title': 'Manage Airports',
```

```
        'current_page': page_num,
        'total_pages': total_pages,
        'prev_page': prev_page,
        'next_page': next_page,
        'start_page': start_page,
        'end_page': end_page
    }

    return render_template(
        'manage_airports.html',
        page=page_info,
        airports=airports_list,
        all_airports=all_airports,
        message=message,
        fetched_airport=fetched_airport,
        form_type=form_type,
        message_type=message_type,
        add_form=add_form
    )



@app.route('/airports/report')
def airport_report():

    if 'logged_in' not in session or not session['logged_in']:
        flash('Please log in to view this page.')
        return redirect(url_for('login'))

    # Get query parameters
    # For sorting and pagination
    try:
        page = request.args.get('page', 1, type=int)
        sort_by = request.args.get('sort_by', 'country')
        sort_dir = request.args.get('sort_dir', 'asc')
    except ValueError:
        flash('Invalid pagination or sorting parameters.')
        return redirect(url_for('airport_report'))

    # Define pagination variables
    per_page = 10
    offset = (page - 1) * per_page

    summary = database.get_airports_by_country(sort_by=sort_by,
sort_dir=sort_dir, limit=per_page, offset=offset)
```

```
    total_countries = database.count_airports_by_country()
    total_pages = ceil(total_countries / per_page) if per_page else 1

    if not summary:
        flash('No data available to display.')

    page_info = {
        'title': 'Airport Report',
        'current_page': page,
        'total_pages': total_pages,
        'prev_page': page - 1 if page > 1 else None,
        'next_page': page + 1 if page < total_pages else None,
        'sort_by': sort_by,
        'sort_dir': sort_dir
    }

    return render_template(
        'airport_report.html',
        page=page_info,
        session=session,
        summary=summary
    )
```

manage_airports.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ page.title }}</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.c
ss">
    <style>
        .container {
            margin-top: 30px;
        }

        .form-section {
            margin-bottom: 40px;
        }

        .table-container {
```

```
        margin-bottom: 20px;
        display: flex;
        flex-direction: column;
        align-items: center;
    }

    .table-wrapper {
        width: 80%;
        max-width: 100%;
        height: 500px;
        overflow-y: auto;
        border: 1px solid #dcdcdc;
        border-radius: 5px;
        background-color: #ffffff;
    }

    table {
        width: 100%;
        margin-bottom: 0;
        table-layout: fixed;
    }

    table th,
    table td {
        padding: 10px;
        text-align: left;
        overflow: hidden;
        text-overflow: ellipsis;
        white-space: nowrap;
    }

    table th:nth-child(1),
    table td:nth-child(1) {
        width: 10%;
    }

    table th:nth-child(2),
    table td:nth-child(2) {
        width: 30%;
    }

    table th:nth-child(3),
    table td:nth-child(3) {
        width: 10%;
    }
```

```
        table th:nth-child(4),
        table td:nth-child(4) {
            width: 25%;
        }

        table th:nth-child(5),
        table td:nth-child(5) {
            width: 25%;
        }

        table thead {
            background-color: #f8f9fa;
            position: sticky;
            top: 0;
            z-index: 1;
        }

        table tbody tr:nth-child(even) {
            background-color: #f2f2f2;
        }

        .pagination {
            display: flex;
            justify-content: center;
            padding-left: 0;
            list-style: none;
            margin-top: 20px;
        }

        .pagination li {
            display: inline-block;
            margin: 0 10px;
        }

        .pagination .page-item .page-link {
            border-radius: 10px;
            padding: 10px 20px;
            font-size: 1rem;
        }

        .pagination .page-item.active .page-link {
            background-color: #007bff;
            color: white;
            border-color: #007bff;
        }
```

```css
.pagination .page-item.disabled .page-link {
    color: #6c757d;
}

.form-control {
    border-radius: 10px;
}

.btn {
    border-radius: 20px;
}

.form-group {
    margin-bottom: 20px;
}

.card-custom {
    border: 2px solid #000000;
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 30px;
    margin: 0 auto 40px auto;
    max-width: 100%;
    width: 100%;
    text-align: center;
}

.card-custom h2 {
    margin-bottom: 20px;
}

.card-custom .alert {
    margin-bottom: 20px;
}

.card-custom form {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.card-custom .form-group {
    width: 100%;
    max-width: 400px;
    margin-bottom: 20px;
    text-align: left;
```

```
}

.card-custom .form-group label {
    display: block;
    margin-bottom: 5px;
}

.card-custom .form-group input,
.card-custom .form-group select {
    width: 100%;
}

.card-custom .btn {
    width: 100%;
    max-width: 200px;
    align-self: center;
}

/* Arrange Add and Update forms side by side */
.dual-form {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: center;
}

.dual-form .card-custom,
.dual-form .fetch-form {
    flex: 1 1 300px;
    max-width: 45%;
}

.fetch-form {
    border: 2px solid #000000;
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 30px;
    margin: 0 auto 40px auto;
    max-width: 600px;
    width: 100%;
    text-align: center;
}

.fetch-form h2 {
    margin-bottom: 20px;
}
```

```
.fetch-form .alert {
    margin-bottom: 20px;
}

.fetch-form form {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.fetch-form .form-group {
    width: 100%;
    max-width: 400px;
    margin-bottom: 20px;
    text-align: left;
}

.fetch-form .form-group label {
    display: block;
    margin-bottom: 5px;
}

.fetch-form .form-group input,
.fetch-form .form-group select {
    width: 100%;
}

.fetch-form .btn {
    width: 100%;
    max-width: 200px;
    align-self: center;
}

.fetched-info {
    margin-top: 20px;
    text-align: left;
}

.info-group {
    display: flex;
    justify-content: space-between;
    margin-bottom: 15px;
}

.info-item {
```

```
        width: 48%;
    }

    .info-item.full-width {
        width: 100%;
    }

    label {
        font-weight: bold;
    }

    .success {
        color: green;
        background-color: #d4edda;
        border: 1px solid #c3e6cb;
        padding: 10px;
        border-radius: 5px;
        margin-bottom: 20px;
    }

    .error {
        color: red;
        background-color: #f8d7da;
        border: 1px solid #f5c6cb;
        padding: 10px;
        border-radius: 5px;
        margin-bottom: 20px;
    }

    .back-button {
        position: absolute;
        top: 10px;
        right: 20px;
    }


    @media (max-width: 992px) {
        .dual-form .card-custom {
            max-width: 45%;
        }
    }

    @media (max-width: 768px) {
        .table-wrapper {
            width: 100%;
        }
```

```
            .dual-form .card-custom,
            .fetch-form {
                max-width: 100%;
            }

            table th:nth-child(1),
            table td:nth-child(1),
            table th:nth-child(2),
            table td:nth-child(2),
            table th:nth-child(3),
            table td:nth-child(3),
            table th:nth-child(4),
            table td:nth-child(4),
            table th:nth-child(5),
            table td:nth-child(5) {
                width: auto;
                white-space: normal;
            }
        }
    </style>
    <script>
        const airportsData = {{ all_airports | tojson }};
    </script>
</head>

<body>
    <div class="container">
        <h1 class="text-center">{{ page.title }}</h1>

        <div class="table-container">
            <h2 class="mt-4">Airport List</h2>
            <div class="table-wrapper">
                <table class="table table-bordered table-hover">
                    <thead class="thead-light">
                        <tr>
                            <th>Airport ID</th>
                            <th>Name</th>
                            <th>IATA Code</th>
                            <th>City</th>
                            <th>Country</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for airport in airports %}
                        <tr>
```

```
                            <td>{{ airport.airportid }}</td>
                            <td title="{{ airport.name }}">{{ airport.name
}}</td>
                            <td>{{ airport.iatacode }}</td>
                            <td>{{ airport.city }}</td>
                            <td>{{ airport.country }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>

            <!-- Back to Index Button -->
            <div class="back-button">
                <a href="{{ url_for('index') }}" class="btn btn-
secondary">Back to Index</a>
            </div>

            <nav aria-label="Page navigation">
                <ul class="pagination">
                    <li class="page-item {% if not page.prev_page
%}disabled{% endif %}">
                        <a class="page-link" href="{{
url_for('manage_airports', page=page.prev_page) }}" aria-label="Previous">
                            <span aria-hidden="true">&laquo;</span>
                        </a>
                    </li>

                    {% for p in range(page.start_page, page.end_page + 1) %}
                    <li class="page-item {% if p == page.current_page
%}active{% endif %}">
                        <a class="page-link" href="{{
url_for('manage_airports', page=p) }}">{{ p }}</a>
                    </li>
                    {% endfor %}

                    <li class="page-item {% if not page.next_page
%}disabled{% endif %}">
                        <a class="page-link" href="{{
url_for('manage_airports', page=page.next_page) }}" aria-label="Next">
                            <span aria-hidden="true">&raquo;</span>
                        </a>
                    </li>
                </ul>
            </nav>
        </div>
```

```
        <!-- Add and Update -->
        <div class="form-section">
            <div class="dual-form">
                <!-- Add -->
                <div class="card-custom">
                    <h2>Add Airport</h2>
                    {% if form_type == 'add' and message %}
                    <div class="{{ message_type }}">
                        <p>{{ message }}</p>
                    </div>
                    {% endif %}

                    <form action="{{ url_for('manage_airports') }}"
method="POST">
                        {{ add_form.hidden_tag() }}

                        <div class="form-group">
                            {{ add_form.airportid.label(class="form-control-
label") }}
                            {{ add_form.airportid(class="form-control form-
control-lg") }}
                            {% if add_form.airportid.errors %}
                            <div class="error">
                                {% for error in add_form.airportid.errors %}
                                <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ add_form.name.label(class="form-control-
label") }}
                            {{ add_form.name(class="form-control form-
control-lg") }}
                            {% if add_form.name.errors %}
                            <div class="error">
                                {% for error in add_form.name.errors %}
                                <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ add_form.iatacode.label(class="form-control-
label") }}
```

```
                            {{ add_form.iatacode(class="form-control form-
control-lg") }}
                            {% if add_form.iatacode.errors %}
                            <div class="error">
                                {% for error in add_form.iatacode.errors %}
                                <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ add_form.city.label(class="form-control-
label") }}
                            {{ add_form.city(class="form-control form-
control-lg") }}
                            {% if add_form.city.errors %}
                            <div class="error">
                                {% for error in add_form.city.errors %}
                                <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ add_form.country.label(class="form-control-
label") }}
                            {{ add_form.country(class="form-control form-
control-lg") }}
                            {% if add_form.country.errors %}
                            <div class="error">
                                {% for error in add_form.country.errors %}
                                <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>
                        {{ add_form.add_airport(class="btn btn-success btn-
lg") }}
                    </form>
                </div>

                <div class="card-custom">
                    <h2>Update Airport</h2>
                    {% if form_type == 'update' and message %}
                    <div class="{{ message_type }}">
                        <p>{{ message }}</p>
```

```
                    </div>
                    {% endif %}

                    <form action="{{ url_for('manage_airports') }}"
method="POST">
                        <div class="form-group">
                            <label for="airportid_update">Airport ID:
</label>
                            <select id="airportid_update" name="airportid"
class="form-control form-control-lg" required>
                                <option value="" disabled selected>Select
Airport ID</option>
                                {% for airport in all_airports %}
                                <option value="{{ airport.airportid }}">{{
airport.airportid }}</option>
                                {% endfor %}
                            </select>
                        </div>
                        <div class="form-group">
                            <label for="name_update">Airport Name:</label>
                            <input type="text" id="name_update" name="name"
class="form-control form-control-lg" required>
                        </div>
                        <div class="form-group">
                            <label for="iatacode_update">IATA Code:</label>
                            <input type="text" id="iatacode_update"
name="iatacode" class="form-control form-control-lg" required maxlength="3">
                        </div>
                        <div class="form-group">
                            <label for="city_update">City:</label>
                            <input type="text" id="city_update" name="city"
class="form-control form-control-lg" required>
                        </div>
                        <div class="form-group">
                            <label for="country_update">Country:</label>
                            <input type="text" id="country_update"
name="country" class="form-control form-control-lg" required>
                        </div>
                        <button type="submit" name="update_airport"
class="btn btn-primary btn-lg">Update Airport</button>
                    </form>
                </div>
            </div>
        </div>

        <div class="form-section">
```

```
            <div class="dual-form">
                <!-- Delete Airport -->
                <div class="card-custom">
                    <h2>Delete Airport</h2>
                    {% if form_type == 'delete' and message %}
                    <div class="{{ message_type }}">
                        <p>{{ message }}</p>
                    </div>
                    {% endif %}

                    <form action="{{ url_for('manage_airports') }}"
method="POST">
                        <div class="form-group">
                            <label for="airportid_delete">Airport ID:
</label>
                            <select id="airportid_delete" name="airportid"
class="form-control form-control-lg" required>
                                <option value="" disabled selected>Select
Airport ID</option>
                                {% for airport in all_airports %}
                                <option value="{{ airport.airportid }}">{{
airport.airportid }}</option>
                                {% endfor %}
                            </select>
                        </div>
                        <button type="submit" name="delete_airport"
class="btn btn-danger btn-lg">Delete Airport</button>
                    </form>
                </div>

                <!-- Fetch Airport Information -->
                <div class="fetch-form">
                    <h2>Fetch Airport Information</h2>
                    {% if form_type == 'fetch' and message %}
                    <div class="{{ message_type }}">
                        <p>{{ message }}</p>
                    </div>
                    {% endif %}

                    <form action="{{ url_for('manage_airports') }}"
method="POST">
                        <div class="form-group">
                            <label for="fetch_airportid">Airport ID:</label>
                            <select id="fetch_airportid"
name="fetch_airportid" class="form-control form-control-lg" required>
                                <option value="" disabled selected>Select
```

```
Airport ID</option>
                                {% for airport in all_airports %}
                                <option value="{{ airport.airportid }}">{{
airport.airportid }}</option>
                                {% endfor %}
                            </select>
                        </div>
                        <button type="submit" name="fetch_airport"
class="btn btn-secondary btn-lg">Fetch Airport</button>
                    </form>

                    {% if fetched_airport %}
                    <div class="fetched-info">
                        <h3>Airport Details</h3>
                        <div class="info-group">
                            <div class="info-item">
                                <label>Airport ID:</label>
                                <span>{{ fetched_airport.airportid }}</span>
                            </div>
                            <div class="info-item">
                                <label>Name:</label>
                                <span>{{ fetched_airport.name }}</span>
                            </div>
                        </div>
                        <div class="info-group">
                            <div class="info-item">
                                <label>IATA Code:</label>
                                <span>{{ fetched_airport.iatacode }}</span>
                            </div>
                            <div class="info-item">
                                <label>City:</label>
                                <span>{{ fetched_airport.city }}</span>
                            </div>
                        </div>
                        <div class="info-group">
                            <div class="info-item full-width">
                                <label>Country:</label>
                                <span>{{ fetched_airport.country }}</span>
                            </div>
                        </div>
                    </div>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
```

```
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            const updateSelect =
document.getElementById('airportid_update');
            const nameUpdate = document.getElementById('name_update');
            const iataUpdate = document.getElementById('iatacode_update');
            const cityUpdate = document.getElementById('city_update');
            const countryUpdate = document.getElementById('country_update');

            updateSelect.addEventListener('change', function() {
                const selectedId = this.value;
                const airport = airportsData.find(a => a.airportid ===
selectedId);
                if (airport) {
                    nameUpdate.value = airport.name;
                    iataUpdate.value = airport.iatacode;
                    cityUpdate.value = airport.city;
                    countryUpdate.value = airport.country;
                }
            });

            const fetchSelect = document.getElementById('fetch_airportid');
            fetchSelect.addEventListener('change', function() {
                const selectedId = this.value;
                const airport = airportsData.find(a => a.airportid ===
selectedId);
            });
        });
    </script>
</body>

</html>
```

airport_report.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ page.title }}</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.c
ss">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

```
    <style>
        .container-flex {
            display: flex;
            flex-direction: column;
            min-height: 100vh;
            position: relative;
            padding-bottom: 60px;
        }

        .back-button {
            position: absolute;
            top: 20px;
            right: 20px;
        }

        .main-content {
            flex: 1;
            margin-top: 60px;
        }

        table {
            width: 100%;
            table-layout: fixed;
            margin-bottom: 0;
        }

        /* Country */
        colgroup col:first-child {
            width: 70%;
        }

        /* Number of Airports */
        colgroup col:last-child {
            width: 30%;
        }

        /* Table*/
        table th,
        table td {
            padding: 10px;
            text-align: left;
            overflow: hidden;
            text-overflow: ellipsis;
            white-space: nowrap;
            border: 1px solid #dee2e6;
        }
```

```
        /* Sortable headers cursor */
        th.sortable {
            cursor: pointer;
        }

        th.sortable:hover {
            background-color: #f1f1f1;
        }

        .sort-indicator {
            margin-left: 5px;
            font-size: 0.8rem;
        }

        .pagination-container {
            margin-top: auto;
            padding: 20px 0;
            display: flex;
            justify-content: center; /* Centers the pagination */
        }

        /* Pie Chart*/
        .chart-container {
            position: relative;
            margin: 40px auto;
            height: 400px;
            width: 400px;
        }

        /* Responsive adjustments */
        @media (max-width: 768px) {
            table th,
            table td {
                white-space: normal; /* Allow text to wrap on smaller
screens */
            }

            .back-button {
                top: 10px;
                right: 10px;
            }

            .container-flex {
                padding-bottom: 80px; /* Increased space for mobile
pagination */
```

```
            }

            .main-content {
                margin-top: 70px; /* Adjusted for mobile back button */
            }

            .chart-container {
                width: 100%;
                height: auto;
            }
        }
    </style>
</head>
<body>
    <div class="container-flex">
        <!-- Back to Index Button -->
        <div class="back-button">
            <a href="{{ url_for('index') }}" class="btn btn-secondary">Back
to Index</a>
        </div>

        <div class="main-content container mt-5">
            <!-- Flash Messages -->
            {% with messages = get_flashed_messages() %}
                {% if messages %}
                    {% for message in messages %}
                        <div class="alert alert-warning alert-dismissible
fade show" role="alert">
                            {{ message }}
                            <button type="button" class="close" data-
dismiss="alert" aria-label="Close">
                                <span aria-hidden="true">&times;</span>
                            </button>
                        </div>
                    {% endfor %}
                {% endif %}
            {% endwith %}

            <h1 class="mb-4">{{ page.title }}</h1>

            <div class="chart-container">
                <canvas id="airportPieChart"></canvas>
            </div>

            {% if summary %}
                <div class="table-wrapper">
```

```
                    <table class="table table-striped table-bordered">
                        <colgroup>
                            <col style="width: 70%;">
                            <col style="width: 30%;">
                        </colgroup>
                        <thead class="thead-dark">
                            <tr>
                                <!-- Sortable Table Headers -->
                                <th class="sortable">
                                    <a href="{{ url_for('airport_report',
page=page.current_page, sort_by='country', sort_dir='asc' if page.sort_by !=
'country' or page.sort_dir == 'desc' else 'desc') }}" style="color: inherit;
text-decoration: none;">
                                        Country
                                        {% if page.sort_by == 'country' %}
                                            <span class="sort-indicator">
                                                {% if page.sort_dir == 'asc'
%}
                                                    &#9650; <!-- Up Arrow --
>
                                                {% else %}
                                                    &#9660; <!-- Down Arrow
-->
                                                {% endif %}
                                            </span>
                                        {% endif %}
                                    </a>
                                </th>
                                <th class="sortable">
                                    <a href="{{ url_for('airport_report',
page=page.current_page, sort_by='airport_count', sort_dir='asc' if
page.sort_by != 'airport_count' or page.sort_dir == 'desc' else 'desc') }}"
style="color: inherit; text-decoration: none;">
                                        Number of Airports
                                        {% if page.sort_by ==
'airport_count' %}
                                            <span class="sort-indicator">
                                                {% if page.sort_dir == 'asc'
%}
                                                    &#9650;
                                                {% else %}
                                                    &#9660;
                                                {% endif %}
                                            </span>
                                        {% endif %}
                                    </a>
```

```
                            </th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for entry in summary %}
                            <tr>
                                <td>{{ entry.country }}</td>
                                <td>{{ entry.airport_count }}</td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        {% else %}
            <p>No data available to display.</p>
        {% endif %}
    </div>

    <!-- Pagination Controls -->
    {% if page.total_pages > 1 %}
        <div class="pagination-container">
            <nav aria-label="Page navigation">
                <ul class="pagination">
                    <!-- Previous Page Link -->
                    <li class="page-item {% if not page.prev_page
%}disabled{% endif %}">
                        <a class="page-link" href="{{
url_for('airport_report', page=page.prev_page, sort_by=page.sort_by,
sort_dir=page.sort_dir) }}" aria-label="Previous">
                            <span aria-hidden="true">&laquo;</span>
                        </a>
                    </li>

                    <!-- Page Number Links -->
                    {% for p in range(1, page.total_pages + 1) %}
                        <li class="page-item {% if p ==
page.current_page %}active{% endif %}">
                            <a class="page-link" href="{{
url_for('airport_report', page=p, sort_by=page.sort_by,
sort_dir=page.sort_dir) }}">{{ p }}</a>
                        </li>
                    {% endfor %}

                    <!-- Next Page Link -->
                    <li class="page-item {% if not page.next_page
%}disabled{% endif %}">
```

```
                            <a class="page-link" href="{{
url_for('airport_report', page=page.next_page, sort_by=page.sort_by,
sort_dir=page.sort_dir) }}" aria-label="Next">
                                <span aria-hidden="true">&raquo;</span>
                            </a>
                        </li>
                    </ul>
                </nav>
            </div>
        {% endif %}
    </div>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js">
</script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
></script>

    <!-- Extension: Pie Chart-->
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            {% if summary %}
                // Extrac name and count
                const countries = {{ summary | map(attribute='country') |
list | tojson }};
                const airportCounts = {{ summary |
map(attribute='airport_count') | list | tojson }};

                // colors
                const predefinedColors = [
                    'rgba(255, 99, 132, 0.6)',
                    'rgba(54, 162, 235, 0.6)',
                    'rgba(255, 206, 86, 0.6)',
                    'rgba(75, 192, 192, 0.6)',
                    'rgba(153, 102, 255, 0.6)',
                    'rgba(255, 159, 64, 0.6)',
                    'rgba(199, 199, 199, 0.6)',
                    'rgba(83, 102, 255, 0.6)',
                    'rgba(255, 102, 255, 0.6)',
                    'rgba(102, 255, 102, 0.6)'
                ];

                const backgroundColors = countries.map((_, index) =>
predefinedColors[index % predefinedColors.length]);
```

```
                const borderColors = backgroundColors.map(color =>
color.replace('0.6', '1'));

                const ctx =
document.getElementById('airportPieChart').getContext('2d');

                // Create pie chart
                const airportPieChart = new Chart(ctx, {
                    type: 'pie',
                    data: {
                        labels: countries,
                        datasets: [{
                            data: airportCounts,
                            backgroundColor: backgroundColors,
                            borderColor: borderColors,
                            borderWidth: 1
                        }]
                    },

                    options: {
                        responsive: false,
                        maintainAspectRatio: false,
                        plugins: {
                            legend: {
                                position: 'right',
                            },

                            tooltip: {
                                callbacks: {
                                    label: function(context) {
                                        const label = context.label || '';
                                        const value = context.parsed || 0;
                                        const total =
context.chart._metasets[context.datasetIndex].total;
                                        const percentage = ((value / total)
* 100).toFixed(2);

                                        return `${label}: ${value}
(${percentage}%)`;
                                    }
                                }
                            }

                        }
                    }
                });
```

```
            {% endif %}
        });
    </script>
</body>
</html>
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Length

class AddAirportForm(FlaskForm):
    airportid = StringField('Airport ID', validators=
[DataRequired(message="Airport ID is required.")])
    name = StringField('Airport Name', validators=
[DataRequired(message="Airport name is required.")])
    iatacode = StringField('IATA Code', validators=[
        DataRequired(message="IATA code is required."),
        Length(min=3, max=3, message="IATA code must be exactly 3 characters
long.")
    ])
    city = StringField('City', validators=[DataRequired(message="City is
required.")])
    country = StringField('Country', validators=
[DataRequired(message="Country is required.")])
    add_airport = SubmitField('Add Airport')
```

database.py

```
def get_all_airports():
    conn = database_connect()
    if conn is None:
        return None
    cur = conn.cursor()

    sql = "SELECT * FROM airports;"
    try:
        result = dictfetchall(cur, sql)
    except Exception as e:
        print("Error fetching all airports:", e)
        result = None
    finally:
        cur.close()
```

```
        conn.close()
    return result

def get_airport_by_id(airportid):
    conn = database_connect()
    if conn is None:
        return None
    cur = conn.cursor()

    sql = "SELECT * FROM airports WHERE airportid = %s;"
    try:
        cur.execute(sql, (airportid,))
        row = cur.fetchone()
        if row:
            return {
                'airportid': row[0],
                'name': row[1],
                'iatacode': row[2],
                'city': row[3],
                'country': row[4]
            }
        else:
            return None
    except Exception as e:
        print("Error fetching airport by ID:", e)
        return None
    finally:
        cur.close()
        conn.close()


def add_airport(airportid, name, iatacode, city, country):
    conn = database_connect()
    if conn is None:
        return None
    cur = conn.cursor()

    sql = """
    INSERT INTO airports (airportid, name, iatacode, city, country)
    VALUES (%s, %s, %s, %s, %s);
    """
    try:
        cur.execute(sql, (airportid, name, iatacode, city, country))
        conn.commit()
        return "Airport added successfully!"
    except Exception as e:
```

```
        print("Error adding airport:", e)
        conn.rollback()
        return None
    finally:
        cur.close()
        conn.close()

def update_airport(airportid, name=None, iatacode=None, city=None,
country=None):
    conn = database_connect()
    if conn is None:
        return None
    cur = conn.cursor()

    set_clauses = []
    params = []

    if name:
        set_clauses.append("name = %s")
        params.append(name)
    if iatacode:
        set_clauses.append("iatacode = %s")
        params.append(iatacode)
    if city:
        set_clauses.append("city = %s")
        params.append(city)
    if country:
        set_clauses.append("country = %s")
        params.append(country)

    if not set_clauses:
        return "No fields to update."

    sql = f"UPDATE airports SET {', '.join(set_clauses)} WHERE airportid =
%s"
    params.append(airportid)

    try:
        cur.execute(sql, params)
        conn.commit()
        if cur.rowcount == 0:
            # No rows were updated, airportid does not exist
            return False
        else:
            return True
    except Exception as e:
```

```
        print(f"Error updating airport with id {airportid}:", e)
        conn.rollback()
        return None
    finally:
        cur.close()
        conn.close()


def delete_airport(airportid):
    conn = database_connect()
    if conn is None:
        return None
    cur = conn.cursor()

    sql = "DELETE FROM airports WHERE airportid = %s;"
    try:
        cur.execute(sql, [airportid])
        conn.commit()
        if cur.rowcount == 0:
            return False
        else:
            return True
    except Exception as e:
        print(f"Error deleting airport with id {airportid}:", e)
        conn.rollback()
        return None
    finally:
        cur.close()
        conn.close()


def get_airports_by_country(sort_by='country', sort_dir='asc', limit=10,
offset=0):

    allowed_sort_columns = {'country': 'country', 'airport_count':
'airport_count'}
    sort_column = allowed_sort_columns.get(sort_by, 'country')
    sort_direction = 'ASC' if sort_dir.lower() == 'asc' else 'DESC'

    conn = database_connect()
    if conn is None:
        return []
    cur = conn.cursor()

    sql = f"""
    SELECT country, COUNT(*) as airport_count
```

```
    FROM airports
    GROUP BY country
    ORDER BY {sort_column} {sort_direction}
    LIMIT %s OFFSET %s;
    """
    try:
        cur.execute(sql, (limit, offset))
        rows = cur.fetchall()
        # Convert to list of dictionaries
        result = [{'country': row[0], 'airport_count': row[1]} for row in
rows]
    except Exception as e:
        print(f"Error fetching airport report by country: {e}")
        result = []
    finally:
        cur.close()
        conn.close()
    return result

def count_airports_by_country():
    conn = database_connect()
    if conn is None:
        return 0
    cur = conn.cursor()
    sql = """
    SELECT COUNT(DISTINCT country)
    FROM airports;
    """
    try:
        cur.execute(sql)
        count = cur.fetchone()[0]
    except Exception as e:
        print(f"Error counting countries: {e}")
        count = 0
    finally:
        cur.close()
        conn.close()
    return count



def get_airports_paginated(page, per_page):
    offset = (page - 1) * per_page
    query = "SELECT * FROM airports LIMIT %s OFFSET %s"

    conn = database_connect()
```

```
    if conn is None:
        return None
    cur = conn.cursor()

    try:
        cur.execute(query, (per_page, offset))
        result = cur.fetchall()
        columns = [desc[0] for desc in cur.description]
        result = [dict(zip(columns, row)) for row in result]
    except Exception as e:
        print("Error fetching paginated airports:", e)
        result = None
    finally:
        cur.close()
        conn.close()

    return result

def count_airports():
    query = "SELECT COUNT(*) FROM airports"

    conn = database_connect()
    if conn is None:
        return 0
    cur = conn.cursor()

    try:
        cur.execute(query)
        result = cur.fetchone()
    except Exception as e:
        print("Error counting airports:", e)
        result = None
    finally:
        cur.close()
        conn.close()

    return result[0] if result else 0
```

Part B

Extensions

Pagination

On the Airport Manage page, end-users can view all airport information through the buttons located below the table. Similarly, on the Airport Report page, end-users can access statistics by country using the buttons provided. This functionality is implemented through Flask's routing and template rendering.

Each time a page is requested, the backend retrieves the corresponding data from the database but only returns a limited number of records. To control pagination, a `page` parameter is passed to specify the current page number. In the backend, the query to the database uses a pagination function to limit the number of records returned per request.

On the frontend, a pagination navigation bar displays page number links, allowing users to click buttons to switch between different pages. The data for the current page is passed to the template and displayed in the HTML table.

Airport List

Airpo...	Name	IATA ...	City	Country
11	Amsterdam Airport Schiphol	AMS	Amsterdam	Netherlands
12	Dallas-Fort Worth International ...	DFW	Dallas	United States
13	Guangzhou Baiyun Internationa...	CAN	Guangzhou	China
14	Frankfurt am Main International...	FRA	Frankfurt	Germany
15	Istanbul Atatürk Airport	IST	Istanbul	Turkey
16	Indira Gandhi International Airp...	DEL	Delhi	India
17	Soekarno-Hatta International A...	CGK	Jakarta	Indonesia
18	Singapore Changi Airport	SIN	Singapore	Singapore
19	Incheon International Airport	ICN	Seoul	South Korea

« 1 2 3 4 5 »

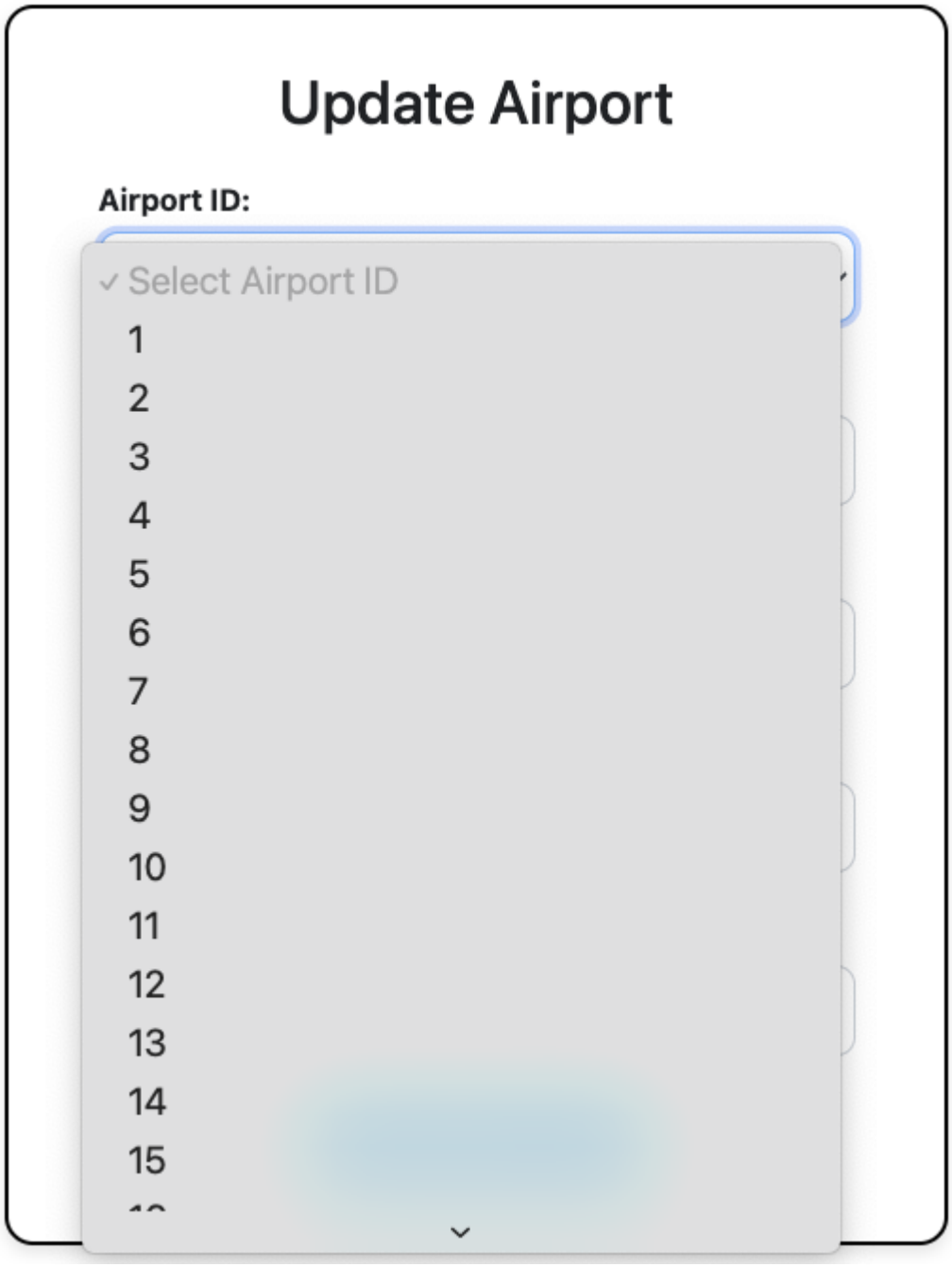
Country ▲	Number of Airports
addd	1
Australia	3
Austria	1
Belgium	1
Brazil	1
Canada	2
China	16
Colombia	1
Denmark	1
France	1

« 1 2 3 4 »

Dropdown menu

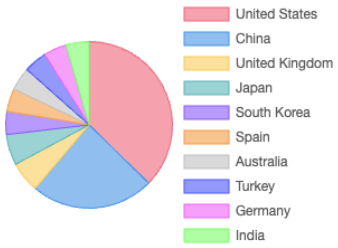
For update, delete, or fetch operations, a dropdown menu is used to let the user select an airport ID. This is implemented by creating a `<select>` element in the HTML form, where the options are populated with all the airport data from the database. Each time the page loads, the

server passes the list of all airports to the template, which then generates the dropdown options for the user to choose the corresponding airport.

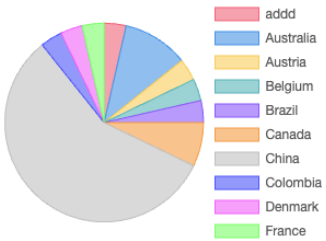


Sort And Pie Chart

On the airport report page, the records in the table can be sorted in ascending or descending order by country name or the number of airports. To achieve this, airport data is first retrieved from the database, and then Python's sorting functionality is used to sort by country name or airport count. The corresponding pie chart is generated using the JavaScript library `Chart.js` to display the distribution of airports across different countries.



Country	Number of Airports ▼
United States	25
China	16
United Kingdom	4
Japan	4
South Korea	3
Spain	3
Australia	3
Turkey	3
Germany	3
India	3

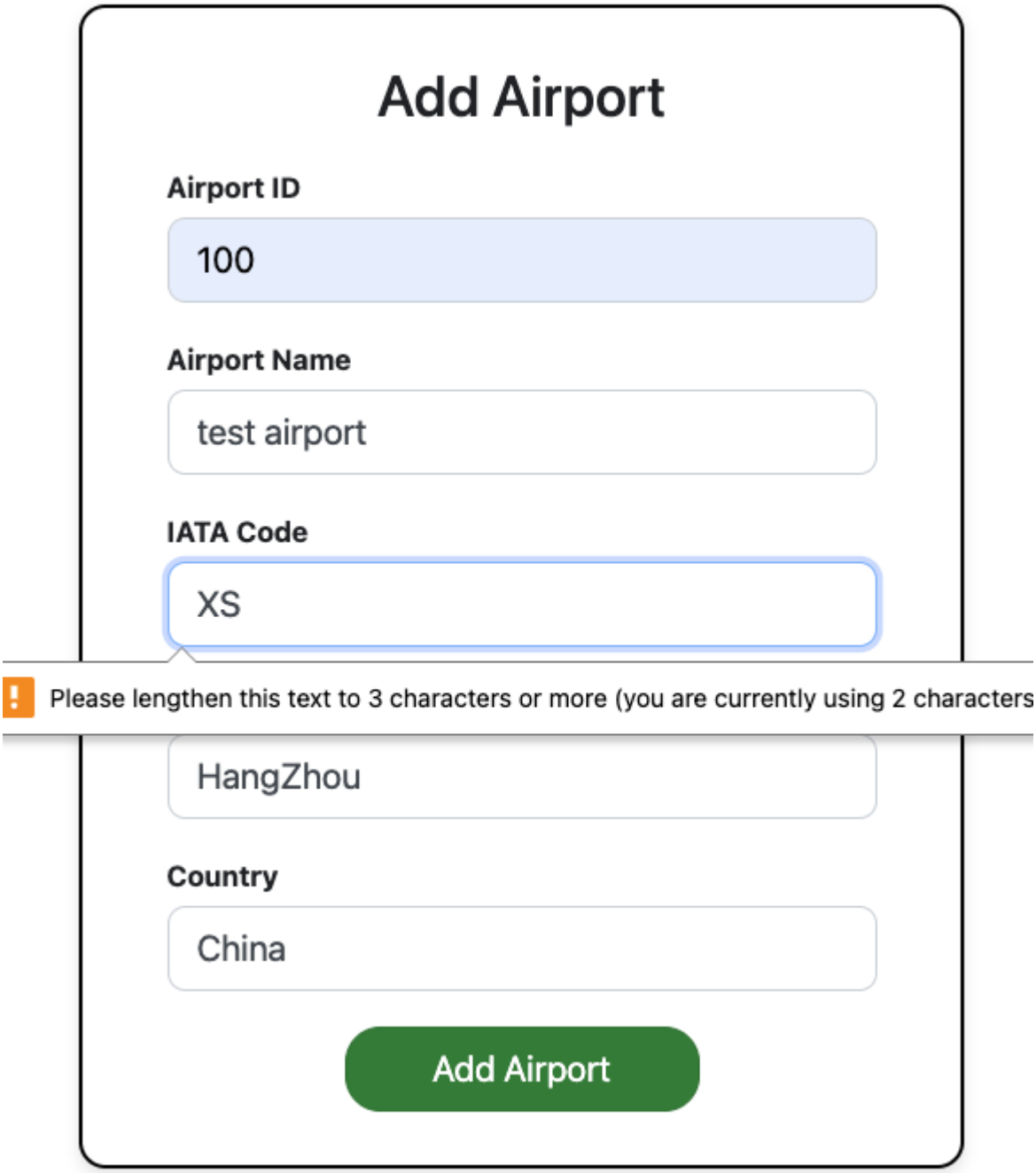


Country ▲	Number of Airports
addd	1
Australia	3
Austria	1
Belgium	1
Brazil	1
Canada	2
China	16
Colombia	1
Denmark	1
France	1

Input Validation

When the Admin tries to add an airport, the app will block invalid inputs, such as ensuring the IATA code must be exactly 3 characters long. This is achieved using Flask-WTF by adding validators (like DataRequired and Length) to the form fields. Flask-WTF automatically checks if

the user's input meets the requirements.



Learning Outcome

Flask-WTF Form Handling

In the project, I learned how to use Flask-WTF to simplify form handling. By defining the `AddAirportForm` class in `forms.py`, I was able to add validators to the form fields and render them in the frontend using templates. This made the entire form handling process much simpler, avoiding the manual creation of forms. In `forms.py`, each field is defined using WTForms field types, like `StringField`, and paired with validators to enforce input rules. For

instance, I used the `DataRequired` validator to ensure that required fields are not left empty and the `Length` validator to ensure that the IATA code is exactly 3 characters long.

Form Validation Methods

The `DataRequired` in validator ensures that a field is not left empty. This is useful for mandatory fields like airport ID, name, etc.The `Length` validator restricts the length of input strings. In the project, I used it to ensure the IATA code is exactly 3 characters long.When the user submits the form, Flask-WTF automatically checks the validators on all fields. If the input is invalid, the form will not pass, and error messages will be returned to the user.

HTTP Methods Usage

When handling HTTP requests, I learned how to use `GET` and `POST` methods to distinguish different operational logic. By combining Flask's `routes`, I can use the GET method to load webpage content and the POST method to handle user-submitted form data.
For the `@app.route('/manage_airports', methods=['GET', 'POST'])` in `routes.py`, the `GET` method is used to load the page and display existing airport data, allowing users to view the list of airports. The `POST` method handles form submission, adding new airport information to the system.

Session Management

In the project, I learned how to use Flask's Session to track login status. Through Flask's session management, I can track a user's login status on the server side. In my project, the Session is used to determine whether a user is logged in and, based on their login status, control their access to certain pages or actions. For example, only logged-in users can access the airport management page (`manage_airports`). For each request, I check the `user_logged_in` flag in the session to determine if the user has access permissions.

`render_template`

In the project, I learned how to use Flask's `render_template` to render templates and pass parameters to the frontend HTML files. In each route, we use `render_template` to render the corresponding page and pass data to the template for dynamic display. For example, in the code below, I use `render_template` to pass airport data, pagination information, and session data to the `manage_airports.html` template, allowing the frontend to display the list of airports and show different content based on the user's login status.

```
return render_template('manage_airports.html', page=page, session=session,
airports=airport_list, add_form=add_form)
```

`flash`

In my Flask project, I learned how to use the `flash` function to display temporary messages to users. Flash messages are typically used to provide feedback to users, such as success or failure of form submission. These messages are displayed to the user after a request and are only shown once.