|  |
| --- |
| **AIRPORT MANAGEMENT SYSTEM**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **N. RITHISH BARATH [RA2311003012071]**  **B.Tech. CSE - CORE**  **GANESH.M [RA2311003012093]**  **B.Tech. CSE - CORE**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled **AIRPORT SYSTEM MANAGEMENT** is the bonafide work of REG NO: **RA2311003012071** Name: **N. RITHISH BARATH** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.  **SIGNATURE SIGNATURE**  **(GUIDE) (HEAD OF THE DEPARTMENT)** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 4 |
| 2 | Methodology / Procedure/ Algorithm | 7 |
| 3 | Flowchart | 10 |
| 4 | Coding (C/Python) | 11 |
| 5 | Front-end code (HTML, CSS, Javascript) [Optional] | 18 |
| 6 | Modules of the proposed work | 29 |
| 7 | Results/Screenshots | 32 |
| 8 | Conclusion | 40 |
| 9 | References | 41 |

1. **Problem Statement:**

**Background:** Airports are complex entities that require efficient management systems to handle a multitude of tasks, ranging from flight scheduling and passenger management to baggage handling and security. Traditional methods often involve manual processes that can lead to errors, delays, and inefficiencies. An integrated Airport Management System (AMS) leveraging modern technologies like Python, SQLite 3, and Django can significantly enhance the overall operational efficiency of an airport.

**Objectives:**

1. **User Authentication and Authorization:**
   * Implement a secure login system with multi-level authentication to ensure that only authorized personnel can access the system.
   * Define roles such as administrators, airline staff, and ground personnel, each with specific access permissions.
2. **Flight Management:**
   * Develop a module for efficient scheduling, tracking, and management of flights.
   * Allow for the addition, modification, and deletion of flight details, including departure and arrival times, gate assignments, and airline information.
3. **Passenger Information:**
   * Create a passenger management system for the registration, ticketing, and tracking of passengers.
   * Implement seat allocation and reservation features to optimize boarding processes.
4. **Baggage Tracking:**
   * Design a comprehensive baggage tracking system that links each piece of luggage to the respective passenger and flight.
   * Include functionalities for baggage check-in, tracking, and retrieval to enhance overall airport logistics.
5. **Aircraft Management:**
   * Maintain a database of aircraft with details such as specifications, maintenance schedules, and availability.
   * Enable the addition and updating of aircraft information, ensuring accuracy for operational planning.
6. **Reporting and Analytics:**
   * Develop a reporting system that generates insights into flight statistics, passenger demographics, and overall airport performance.
   * Include graphical representations of data for better decision-making by airport management.
7. **Real-time Notifications:**
   * Implement a notification system to alert users about critical events, including flight delays, gate changes, and security alerts.
   * Ensure timely communication to enhance passenger experience and operational efficiency.
8. **Security Measures:**
   * Integrate security features to safeguard sensitive data and prevent unauthorized access.
   * Implement logging mechanisms for auditing system activities and ensuring compliance with security standards.
9. **User-friendly Interface:**
   * Design an intuitive and user-friendly interface for easy navigation by airport staff, administrators, and other authorized users.
   * Ensure accessibility and responsiveness for a seamless user experience.
10. **Scalability and Maintenance:**
    * Develop the system with scalability in mind to accommodate future growth and changes in operational requirements.
    * Implement regular maintenance routines to ensure the system's reliability and address any issues promptly.

By addressing these aspects, the proposed Airport Management System aims to enhance overall operational efficiency, passenger experience, and security while providing a robust platform for future developments and expansions.

**Scope:**

The Airport Management System will encompass the design, development, and implementation of a comprehensive software solution to address the identified challenges. The system will integrate various airport functions and departments, providing a unified platform for efficient management, monitoring, and decision-making.

**Expected Outcomes:**

1. **Efficient Data Management:**
   * Centralized storage and management of airport-related data in a SQLite 3 database, providing efficient data retrieval and storage.
2. **Rapid Application Development:**
   * Leveraging the Django framework for rapid development, resulting in a scalable and maintainable airport management system.
3. **User-Friendly Interface:**
   * Development of an intuitive and user-friendly interface using Django's templating system, enhancing accessibility for airport staff and administrators.
4. **Cross-Platform Compatibility:**
   * Compatibility across various platforms and devices, allowing users to access the system seamlessly from desktops, tablets, and smartphones.
5. **Responsive Web Design:**
   * Implementation of responsive web design using Django, ensuring optimal user experience across different screen sizes and resolutions.

**Challenges and Considerations:**

* **Scalability:** Designing the system to accommodate growth and changes in operational requirements.
* **Security:** Ensuring the implementation of robust security measures to protect sensitive data.
* **Integration:** Facilitating integration with external systems for a comprehensive airport ecosystem.
* **User Training:** Providing adequate training for airport personnel to use the new system effectively.

**Conclusion:**

This Airport Management System project, utilizing Python, SQLite 3, and Django, is poised to deliver a cutting-edge solution that addresses the complexities of airport management. Through a combination of efficient coding practices, secure data management, and a user-centric interface, the system aims to set a new standard for airport operations management.

2.**Algorithm:**

**1. Define Database Models:**

* + Create models for passengers, flights, and resources using Django's Object-Relational Mapping (ORM). These models will represent the structure of the database.

1. **Passenger Check-in:**
   * Implement a function for passenger check-in.
   * Accept passenger information such as name and ticket number.
   * Use Django's ORM to create a new record in the Passenger table with the provided information.
2. **Flight Scheduling:**
   * Implement a function for flight scheduling.
   * Accept flight information including the flight number, departure time, and arrival time.
   * Use Django's ORM to create a new record in the Flight table with the provided information.
3. **Resource Allocation:**
   * Implement a function for resource allocation (e.g., gates or runways).
   * Accept the type of resource to be allocated.
   * Use Django's ORM to create a new record in the Resource table with the provided information.
4. **Sample Usage:**
   * Demonstrate the usage of the implemented functions with sample data.
   * Call the check-in function with a passenger's name and ticket number.
   * Call the flight scheduling function with a flight number, departure time, and arrival time.
   * Call the resource allocation function with the type of resource to be allocated.
5. **Database Interaction:**
   * Throughout the process, Django's ORM will interact with the SQLite 3 database to store and retrieve information.
   * The Passenger, Flight, and Resource models define the structure of the database tables, and instances of these models correspond to records in the tables.
6. **Extend Functionality (Optional):**
   * Depending on the specific requirements of the Airport Management System, additional functions can be implemented.
   * For example, functions related to security protocols, data management, user authentication, and reporting can be added.
7. **Adapt to System Requirements:**
   * Tailor the functions and features based on the unique needs and complexity of the Airport Management System.
   * Ensure that the system adheres to best practices in terms of security, efficiency, and scalability.

This algorithm provides a basic framework for passenger check-in, flight scheduling, and resource allocation, and it serves as a starting point for building a more comprehensive Airport Management System.

**Algorithm as Code:**

**# Import necessary libraries and modules**

**from django.db import models**

**# Define Django models for database representation**

**class Passenger(models.Model):**

**name = models.CharField(max\_length=100)**

**ticket\_number = models.CharField(max\_length=20)**

**# ... other passenger-related fields ...**

**class Flight(models.Model):**

**flight\_number = models.CharField(max\_length=10)**

**departure\_time = models.DateTimeField()**

**arrival\_time = models.DateTimeField()**

**# ... other flight-related fields ...**

**class Resource(models.Model):**

**resource\_type = models.CharField(max\_length=50)**

**availability = models.BooleanField(default=True)**

**# ... other resource-related fields ...**

**# Define Django views and controllers for each module**

**def check\_in\_passenger(passenger\_data):**

**# Implementation for passenger check-in**

**# Update database with passenger information**

**passenger = Passenger.objects.create(\*\*passenger\_data)**

**return f"Passenger {passenger.name} checked in successfully."**

**def schedule\_flight(flight\_data):**

**# Implementation for flight scheduling**

**# Update database with flight information**

**flight = Flight.objects.create(\*\*flight\_data)**

**return f"Flight {flight.flight\_number} scheduled successfully."**

**def allocate\_resource(resource\_data):**

**# Implementation for resource allocation**

**# Update database with resource allocation information**

**resource = Resource.objects.create(\*\*resource\_data)**

**return f"Resource {resource.resource\_type} allocated successfully."**

**# Define additional views and controllers for security, data management, etc.**

**# Sample usage of the functions**

**passenger\_data = {'name': 'John Doe', 'ticket\_number': 'ABC123'}**

**check\_in\_passenger(passenger\_data)**

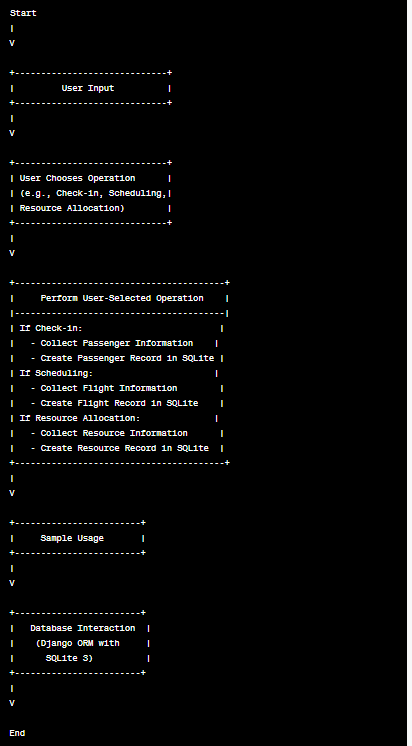
**flight\_data = {'flight\_number': 'FL123', 'departure\_time': '2023-12-01 12:00:00', 'arrival\_time': '2023-12-01 15:00:00'}**

**schedule\_flight(flight\_data)**

**resource\_data = {'resource\_type': 'Gate 1', 'availability': True}**

**allocate\_resource(resource\_data)**

**3.Flowchart:**

****

4.**Coding (Python/Django)**

**admin.py**

from django import forms

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

# Register your models here.

from .models import User, Passenger, Flight, Security, Staff

class UserCreationForm(forms.ModelForm):

    class Meta:

        model = User

        fields = ('username', 'email', 'user\_type')

    def save(self, commit=True):

        # Save the provided password in hashed format

        user = super(UserCreationForm, self).save(commit=False)

        user.set\_password(self.cleaned\_data["password"])

        if commit:

            user.save()

        return user

class CustomUserAdmin(UserAdmin):

    # The forms to add and change user instances

    add\_form = UserCreationForm

    list\_display = ('username', 'email', 'user\_type')  # fields to display in admin page while viewing all users

    ordering = ("email",)  # order in which displayed

    fieldsets = (

        (None, {'fields': ('email', 'password', 'first\_name', 'last\_name')}),

        )

    add\_fieldsets = (

        (None, {

            'classes': ('wide',),

            'fields': ('username', 'email', 'password', 'first\_name', 'last\_name', 'is\_superuser', 'is\_staff',

                       'is\_active', 'user\_type')}

        ),

    )

    filter\_horizontal = ()

admin.site.register(User, CustomUserAdmin)

# admin.site.register(User)#, UsersAdmin)

admin.site.register(Passenger)

admin.site.register(Flight)

admin.site.register(Security)

admin.site.register(Staff)

**apps.py**

from django.apps import AppConfig

class FlightsConfig(AppConfig):

    name = 'flights'

**models.py**

from django.db import models

from django.contrib.auth.models import AbstractUser

import datetime

import django.utils.timezone

# Create your models here.

class Flight(models.Model):

    flight\_no = models.IntegerField(primary\_key=True, default=1007)

    airline\_name = models.CharField(max\_length=50)

    no\_of\_seats = models.IntegerField(default=0)

    source = models.CharField(max\_length=50)

    source\_code = models.CharField(max\_length=3)

    destination = models.CharField(max\_length=50)

    destination\_code = models.CharField(max\_length=3)

    arrival\_time = models.DateTimeField()

    departure\_time = models.DateTimeField()

class Passenger(models.Model):

    pnr = models.CharField(max\_length=10)

    first\_name = models.CharField(max\_length=50)

    last\_name = models.CharField(max\_length=50)

    dob = models.DateField(default=1/1/1990)

    nationality = models.CharField(max\_length=50)

    gender = models.CharField(max\_length=1)

    flight\_no = models.ForeignKey(Flight, on\_delete=models.CASCADE, default=1007)

    checked\_in\_status = models.BooleanField(default=0)

    cleared\_security\_status = models.IntegerField(default=0)

class User(AbstractUser):

    USER\_TYPE\_CHOICES = (

      (1, 'flightstaff'),

      (2, 'security'),

      (3, 'admin'),

    )

    user\_type = models.PositiveSmallIntegerField(choices=USER\_TYPE\_CHOICES, default=3)

class Security(models.Model):

    user = models.OneToOneField(User, on\_delete=models.CASCADE, default=2)

    id = models.IntegerField(primary\_key=True)

class Staff(models.Model):

    user = models.OneToOneField(User, on\_delete=models.CASCADE, default=1)

    id = models.IntegerField(primary\_key=True)

    flight\_no = models.ForeignKey(Flight, on\_delete=models.CASCADE, default=1007)

**tests.py**

from django.test import TestCase

# Create your tests here.

**urls.py**

from django.urls import path, include

from . import views

from django.contrib import admin

admin.site.site\_header = 'Flight Booking Admin'

urlpatterns = [

    path('home/', views.home, name='home'),

    path('airport\_mgmt/', views.airport\_mgmt, name='airport\_mgmt'),

    path('accounts/', include('django.contrib.auth.urls')),

    path('accounts/security\_login', views.login\_security, name='security\_login'),

    path('accounts/staff\_login', views.login\_staff, name='staff\_login'),

    path('security\_clearing', views.clear\_security, name='security\_clearing'),

    path('clearing\_for\_security/<str:pk>', views.clear\_for\_security, name='clearing\_for\_security'),

    path('view\_flights', views.view\_flights, name='view\_flights'),

    path('self\_check\_in/<int:pk>', views.self\_check\_in, name='self\_check\_in'),

    path('search\_by\_source', views.search\_by\_source, name='search\_by\_source'),

    path('search\_by\_destination', views.search\_by\_destination, name='search\_by\_destination'),

    path('staff\_home/<int:flight\_no>', views.staff\_home, name='staff\_home'),

    path('view\_available\_flights/', views.view\_available\_flights, name='view\_available\_flights'),

    path('book\_flight/<int:pk>', views.book\_flight, name='book\_flight'),

    path('passenger\_home/<int:pk>', views.passenger\_home, name='passenger\_home'),

    path('create\_pdf/<int:pk>', views.create\_pdf, name='create\_pdf'),

    path('staff\_check\_in/<int:pk>', views.staff\_check\_in, name='staff\_check\_in'),

    path('generate\_report/<int:flight\_no>', views.generate\_report, name='generate\_report'),

    path('delete\_passenger/<int:flight\_no>', views.delete\_passengers, name='delete\_passengers'),

    path('view\_booking', views.view\_booking, name='view\_booking')

]

**views.py**

from django.contrib.auth import login, authenticate

from django.contrib.auth.decorators import login\_required

from django.shortcuts import render, redirect, get\_object\_or\_404

from .models import User, Passenger, Flight, Security, Staff

import datetime

import pdfkit

from django.http import HttpResponse

from django.template import loader

# Create your views here.

def login\_security(request):

    if request.method == "POST":

        username = request.POST['username']

        password = request.POST['password']

        user = authenticate(username=username, password=password)

        if user is not None:

            if user.is\_active:

                if user.user\_type == 2:

                    login(request, user)

                    return redirect('security\_clearing') #, pk=user.security.id)

                else:

                    return render(request, 'security\_login.html', {'error\_message': 'Invalid security staff credentials'})

            else:

                return render(request, 'security\_login.html', {'error\_message': 'Your account has been disabled'})

        else:

            return render(request, 'security\_login.html', {'error\_message': 'Invalid login'})

    return render(request, 'security\_login.html')

def login\_staff(request):

    if request.method == "POST":

        username = request.POST['username']

        password = request.POST['password']

        user = authenticate(username=username, password=password)

        if user is not None:

            if user.is\_active:

                if user.user\_type == 1:

                    login(request, user)

                    return redirect('staff\_home', flight\_no=user.staff.flight\_no.flight\_no)

                else:

                    return render(request, 'staff\_login.html', {'error\_message': 'Invalid flight staff credentials'})

            else:

                return render(request, 'staff\_login.html', {'error\_message': 'Your account has been disabled'})

        else:

            return render(request, 'staff\_login.html', {'error\_message': 'Invalid login'})

    return render(request, 'staff\_login.html')

def home(request):

    return render(request, 'home.html')

def airport\_mgmt(request):

    return render(request, 'airport\_mgmt.html')

def clear\_security(request):

    data = Passenger.objects.all()

    return render(request, 'security\_clearing.html', {'passengers': data})

def clear\_for\_security(request, pk):

    passenger = get\_object\_or\_404(Passenger, pk=pk)

    passenger.cleared\_security\_status = request.user.security.id

    passenger.save()

    return redirect('security\_clearing')

def staff\_home(request, flight\_no):

    # staff = get\_object\_or\_404(Staff, pk=pk)

    data = Passenger.objects.filter(flight\_no=flight\_no)

    return render(request, 'staff\_home.html', {'passengers': data, 'flight\_no': flight\_no})

def view\_flights(request):

    data = Flight.objects.all()

    return render(request, 'view\_flights.html', {'flights': data})

def self\_check\_in(request, pk):

    passenger = get\_object\_or\_404(Passenger, pk=pk)

    passenger.checked\_in\_status = True

    passenger.save()

    return redirect('passenger\_home', pk=passenger.pk)

def search\_by\_source(request):

    if request.method == "POST":

        source = request.POST['source']

        if source:

            data = Flight.objects.filter(source=source)

            return render(request, 'view\_flights.html', {'flights': data})

        else:

            return redirect('view\_flights')

    else:

        return redirect('view\_flights')

def search\_by\_destination(request):

    if request.method == "POST":

        destination = request.POST['destination']

        if destination:

            data = Flight.objects.filter(destination=destination)

            return render(request, 'view\_flights.html', {'flights': data})

        else:

            return redirect('view\_flights')

    else:

        return redirect('view\_flights')

def view\_available\_flights(request):

    if request.method == "POST":

        source = request.POST['source']

        destination = request.POST['destination']

        if source and destination:

            flights = Flight.objects.filter(source=source, destination=destination)

            if flights:

                return render(request, 'home.html', {'flights': flights})

            else:

                return render(request, 'home.html', {'error\_message\_flight': "No flights found"})

        else:

            return redirect('home')

    else:

        return redirect('home')

def book\_flight(request, pk):

    if request.method == "POST":

        flight = Flight.objects.get(flight\_no=pk)

        first\_name = request.POST['first\_name']

        last\_name = request.POST['last\_name']

        nationality = request.POST['nationality']

        gender = request.POST['gender']

        dob = request.POST['dob']

        pnr = str(flight.flight\_no) + str(flight.destination)

        passenger = Passenger(pnr=pnr, first\_name=first\_name, last\_name=last\_name, nationality=nationality,

                              flight\_no=flight, gender=gender, dob=dob)

        passenger.save()

        passenger.pnr = str(flight.flight\_no) + str(flight.destination\_code) + str(passenger.pk)

        passenger.save()

        flight.no\_of\_seats -= 1

        flight.save()

        return redirect('passenger\_home', pk=passenger.pk)

    else:

        return render(request, 'book\_flight.html', {'flight\_no': pk})

def passenger\_home(request, pk):

    passenger = Passenger.objects.get(pk=pk)

    return render(request, 'passenger\_home.html', {'passenger': passenger})

def view\_booking(request):

    if request.method == "POST":  # view existing booking

        pnr = request.POST['pnr']

        try:

            passenger = Passenger.objects.get(pnr=pnr)

        except Passenger.DoesNotExist:

            passenger = None

        if passenger:

            passenger = get\_object\_or\_404(Passenger, pnr=pnr)

            return render(request, 'passenger\_home.html', {'passenger': passenger})

        else:

            return render(request, 'home.html', {'error\_message\_booking': 'No booking found'})

def create\_pdf(request, pk):

    passenger = get\_object\_or\_404(Passenger, pk=pk)

    html = loader.render\_to\_string('passenger\_home.html', {'passenger': passenger})

    output = pdfkit.from\_string(html, output\_path=False)

    response = HttpResponse(content\_type="application/pdf")

    response.write(output)

    return response

def staff\_check\_in(request, pk):

    passenger = get\_object\_or\_404(Passenger, pk=pk)

    passenger.checked\_in\_status = True

    passenger.save()

    return redirect('staff\_home', flight\_no=passenger.flight\_no.flight\_no)

def delete\_passengers(request, flight\_no):

    flight = Flight.objects.get(flight\_no=flight\_no)

    passenger = Passenger.objects.filter(flight\_no=flight)

    passenger.delete()

    return redirect('staff\_home', flight\_no=flight.flight\_no)

def generate\_report(request, flight\_no):

    passengers = Passenger.objects.filter(flight\_no=flight\_no)

    html = loader.render\_to\_string('staff\_home.html', {'passengers': passengers, 'flight\_no': flight\_no})

    output = pdfkit.from\_string(html, output\_path=False)

    response = HttpResponse(content\_type="application/pdf")

    response.write(output)

    return response

**5.Front-end Code (HTML/CSS/JAVA)**

**airport\_mgmt.html**

{% extends 'base.html' %}

{% block title %}Airport Management System{% endblock %}

{% block body %}

<div class="jumbotron">

  <h2 class="display-3">Airport Management System</h2>

  <p class="lead">The system interfaces and integrates the majority of electronic information within the airport, assuring maximum flow of information for operations, management and security.</p>

  <hr class="my-4">

  <p>This system would be used by multiple users who would want access to different data for their day to day operations. This includes passengers as well as security personnel and flight staff.</p>

    <button type="button" class="btn btn-outline-primary">Security Personnel</button>

    <div class="panel">

        <br>

        <p>They can easily verify the names and information of the passengers travelling across a series of flight manifests.</p>

        <p>They can clear passengers for their flight.</p>

    </div>

    <br>

    <button type="button" class="btn btn-outline-primary">Flight staff</button>

    <div class="panel">

        <br>

        <p>They can access information about passengers travelling in their respective flights.</p>

        <p>Before the flight, they generate a report of all checked in passengers who have cleared security. The report would contain details of number of passengers and their names.</p>

    </div>

    <br>

    <button type="button" class="btn btn-outline-primary">Passengers</button>

    <div class="panel">

        <br>

        <p>They can use their unique PNR to access flight information and perform self check in.</p>

        <p>They can also search and book flights.</p>

    </div>

    <br>

</div>

<style>

.panel {

    padding: 0 18px;

    background-color: white;

    max-height: 0;

    overflow: hidden;

    transition: max-height 0.2s ease-out;

}

</style>

<script>

var acc = document.getElementsByClassName("btn btn-outline-primary");

var i;

for (i = 0; i < acc.length; i++) {

  acc[i].addEventListener("click", function() {

    this.classList.toggle("active");

    var panel = this.nextElementSibling;

    if (panel.style.maxHeight){

      panel.style.maxHeight = null;

    } else {

      panel.style.maxHeight = panel.scrollHeight + "px";

    }

  });

}

</script>

{% endblock %}

**base.html**

{% load static %}

<!DOCTYPE html5>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>{% block title %}{% endblock %}</title>

    {% load staticfiles %}

{#    <link href="https://stackpath.bootstrapcdn.com/bootswatch/4.1.3/united/bootstrap.min.css" rel="stylesheet" integrity="sha384-+d4wMJSxEP3vzs2qZBElQRTZMXwgWH15Nyn2K/9XjKiHmh3sBuk1Un/IbcdMcYC4" crossorigin="anonymous">#}

{#     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">#}

        <link href="https://bootswatch.com/4/united/bootstrap.min.css" rel="stylesheet" >

{#    integrity="sha384-+d4wMJSxEP3vzs2qZBElQRTZMXwgWH15Nyn2K/9XjKiHmh3sBuk1Un/IbcdMcYC4" crossorigin="anonymous">#}

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">

  <a class="navbar-brand" href="{% url 'airport\_mgmt' %}">Airport Management System</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="Toggle navigation">

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarColor01">

    {% with url\_name=request.resolver\_match.url\_name %}

    <ul class="navbar-nav mr-auto">

      <li class="nav-item {% if url\_name == 'home' %}active{% endif %}">

        <a class="nav-link" href="{% url 'home' %}">Home <span class="sr-only">(current)</span></a>

      </li>

      <li class="nav-item {% if url\_name == 'view\_flights' %}active{% endif %}">

        <a class="nav-link" href="{% url 'view\_flights' %}">View Flights</a>

      </li>

      <li class="nav-item {% if url\_name == 'security\_login' %}active{% endif %}">

        <a class="nav-link" href="{% url 'security\_login' %}">Login as Security</a>

      </li>

      <li class="nav-item {% if url\_name == 'staff\_login' %}active{% endif %}">

        <a class="nav-link" href="{% url 'staff\_login' %}">Login as Staff</a>

      </li>

        {% if user.is\_authenticated %}

      <li class="nav-item">

        <a class="nav-link" href="{% url 'logout' %}">Logout</a>

      </li>

        {% endif %}

    </ul>

    {% endwith %}

    <!-- <form class="form-inline my-2 my-lg-0">

      <input class="form-control mr-sm-2" placeholder="Search" type="text">

      <button class="btn btn-secondary my-2 my-sm-0" type="submit">Search</button>

    </form> -->

  </div>

</nav>

<div style="margin-left:20px; margin-top:20px">

{% block body %}{% endblock %}

</div>

<footer class="footer" style="margin-left:20px; margin-top:20px">

  <small>

    Module made by <a href="https://github.com/mehnazyunus">Mehnaz Yunus</a> and <a href="https://github.com/sharanyakamath">Sharanya Kamath</a>

</small>

</footer>

</body>

</html>

**book\_flight.html**

{% extends 'base.html' %}

{% block title %}Book Flight{% endblock %}

{% block body %}

    <div class="container-fluid">

        <div class="row">

            <div class="col-sm-6 col-sm-offset-3">

                <div class="panel panel-default">

                    <div class="panel-body">

                        <form role="form" action="{% url 'book\_flight' pk=flight\_no %}" method="post" enctype="multipart/form-data">

                            {% csrf\_token %}

                          <fieldset>

                            <legend>Enter your details</legend>

                            <div class="form-group">

                              <label for="Name1">First Name</label>

                              <input type="text" class="form-control" id="Name1" name="first\_name" placeholder="Enter first name">

                            </div>

                            <div class="form-group">

                              <label for="Name2">Last Name</label>

                              <input type="text" class="form-control" id="Name2" name="last\_name" placeholder="Enter last name">

                            </div>

                            <div class="form-group">

                              <label for="gender">Gender &nbsp &nbsp</label>

                              <input type="radio"  id="Name2" name="gender" value="M" checked> Male

                              <input type="radio"  id="Name2" name="gender" value="F"> Female

                            </div>

                            <div class="form-group">

                              <label for="dob">Date of Birth</label>

                              <input type="date" class="form-control" id="Name2" name="dob" placeholder="Enter Date of Birth">

                            </div>

                            <div class="form-group">

                              <label for="nationality">Nationality</label>

                              <input type="text" class="form-control" id="nationality" name="nationality" placeholder="Enter nationality">

                            </div>

                            <button type="submit" class="btn btn-primary">Book</button>

                          </fieldset>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

{% endblock %}

**home.html**

{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block body %}

<form class="form-inline my-2 my-lg-0" method="post" action="{% url 'view\_available\_flights'%}" role="form">

{% csrf\_token %}

    <div class="form-group">

    <input class="form-control mr-sm-2" placeholder="Source" name="source" type="text">

    <input class="form-control mr-sm-2" placeholder="Destination" name="destination" type="text">

    <button class="btn btn-secondary my-2 my-sm-0" type="submit" style="margin-right:10px;">View All Available Flights</button>

    </div>

</form>

    {% if error\_message\_flight %}

        <p><strong>{{ error\_message\_flight }}</strong></p>

    {% endif %}

    {% if flights %}

    <table class="table table-hover">

  <thead>

    <tr>

      <th scope="col">Flight No.</th>

      <th scope="col">Airline</th>

      <th scope="col">No. of seats</th>

      <th scope="col">Source</th>

      <th scope="col">Destination</th>

      <th scope="col">Departure Time</th>

      <th scope="col">Arrival Time</th>

    </tr>

  </thead>

  <tbody>

    {% for flight in flights %}

        <tr>

          <td>{{flight.flight\_no}}</td>

          <td>{{flight.airline\_name}}</td>

          <td>{{ flight.no\_of\_seats }}</td>

          <td>{{ flight.source}}</td>

          <td>{{ flight.destination}}</td>

          <td>{{ flight.departure\_time}}</td>

          <td>{{ flight.arrival\_time}}</td>

          <td>

            {% if flight.no\_of\_seats > 0 %}

                <a href="{% url 'book\_flight' flight.pk%}" class="btn btn-outline-success" role="button">Book Now!</a>

            {% else %}

                <button type="button" class="btn btn-success disabled">Book Now!</button>

            {% endif %}

          </td>

        </tr>

    {% endfor %}

  </tbody>

</table>

{% endif %}

<br>

<form class="form-inline my-2 my-lg-0" method="post" action="{% url 'view\_booking' %}" role="form">

  {% csrf\_token %}

  <input class="form-control mr-sm-2" placeholder="Enter PNR" name="pnr" type="text">

  <button class="btn btn-secondary my-2 my-sm-0" type="submit" style="margin-right:10px;">View My Booking</button>

    {% if error\_message\_booking %}<p><strong>{{ error\_message\_booking }}</strong></p>{% endif %}

</form>

{% endblock %}

**passenger\_home.html**

{% extends 'base.html' %}

{% block title %}Passenger Home{% endblock %}

{% block body %}

<div class="container">

        <div class="row">

            <div class="offset-sm-3">

    <div class="card border-primary mb-3" style="max-width: 20rem;">

      <div class="card-header">Passenger Details of PNR: {{ passenger.pnr }}</div>

          <div class="card-body">

            <h6 class="card-title"><b>Name:</b> {{ passenger.first\_name }} {{ passenger.last\_name }}</h6>

              <h6><b>Airline:</b> {{ passenger.flight\_no.airline\_name }}</h6>

              <h6><b>Departure:</b> {{ passenger.flight\_no.departure\_time }}</h6>

              <h6><b>Arrival:</b> {{ passenger.flight\_no.arrival\_time }}</h6>

              <h6><b>From:</b> {{ passenger.flight\_no.source }}</h6>

              <h6><b>To:</b> {{ passenger.flight\_no.destination }}</h6>

          </div>

    </div>

    {% if passenger.checked\_in\_status == False %}

      <a href="{% url 'self\_check\_in' passenger.pk %}" class="btn btn-outline-success" role="button">Perform Self Check-In</a>

    {% else %}

      <a href="#" class="btn btn-outline-success disabled" role="button">Checked In</a>

    {% endif %}

    <a href="{% url 'create\_pdf' passenger.pk %}" class="btn btn-outline-success" role="button">Download Ticket</a>

  </div>

</div>

</div>

{% endblock %}

**security\_clearing.html**

{% extends 'base.html' %}

{% block title %}Security Clearing{% endblock %}

{% block body %}

    <h3>Security</h3>

    <table class="table table-hover">

  <thead>

    <tr>

      <th scope="col">PNR</th>

      <th scope="col">Name</th>

      <th scope="col">Flight No.</th>

      <th scope="col">Cleared Security Status</th>

    </tr>

  </thead>

  <tbody>

    {% for passenger in passengers %}

        <tr>

          <td>{{ passenger.pnr}}</td>

          <td>{{ passenger.first\_name}} {{ passenger.last\_name}}</td>

          <td>{{ passenger.flight\_no.flight\_no }}</td>

          <td>{{ passenger.cleared\_security\_status }}</td>

          <td>

            {% if passenger.cleared\_security\_status == 0 %}

                <a href="{% url 'clearing\_for\_security' passenger.pk%}" class="btn btn-outline-success" role="button">Clear for security</a>

            {% else %}

                <button type="button" class="btn btn-success disabled">Clear for security</button>

            {% endif %}

          </td>

        </tr>

    {% endfor %}

  </tbody>

</table>

{% endblock %}

**security\_login.html**

{% extends 'base.html' %}

{% block title %}Security Log In{% endblock %}

{% block body %}

    <div class="container-fluid">

        <div class="row">

            <div class="col-sm-6 offset-sm-3">

                <div class="panel panel-default">

                    <div class="panel-body">

                        {% if error\_message %}

                        <p><strong>{{ error\_message }}</strong></p>

                        {% endif %}

                        <form role="form" action="{% url 'security\_login'%}" method="post" enctype="multipart/form-data">

                            {% csrf\_token %}

                          <fieldset>

                            <legend>Security Staff Login</legend>

                            <div class="form-group">

                              <label for="InputUsername1">Username</label>

                              <input type="text" class="form-control" id="InputUsername1" name="username" aria-describedby="emailHelp" placeholder="Enter username">

                            </div>

                            <div class="form-group">

                              <label for="exampleInputPassword1">Password</label>

                              <input type="password" class="form-control" id="exampleInputPassword1" name="password" placeholder="Password">

                            </div>

                            <button type="submit" class="btn btn-primary">Submit</button>

                          </fieldset>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

{% endblock %}

**staff\_home.html**

{% extends 'base.html' %}

{% block title %}Staff Report Generation{% endblock %}

{% block body %}

    <h3>Staff</h3>

    <table class="table table-hover">

  <thead>

    <tr>

      <th scope="col">Flight No.</th>

      <th scope="col">PNR</th>

      <th scope="col">Name</th>

      <th scope="col">Date of Birth</th>

      <th scope="col">Nationality</th>

      <th scope="col">Gender</th>

      <th scope="col">Checked In Status</th>

      <th scope="col">Cleared Security Status</th>

    </tr>

  </thead>

  <tbody>

    {% for passenger in passengers %}

        <tr>

          <td>{{ passenger.flight\_no.flight\_no}}</td>

          <td>{{ passenger.pnr}}</td>

          <td>{{ passenger.first\_name}} {{ passenger.last\_name}}</td>

          <td>{{ passenger.dob}}</td>

          <td>{{ passenger.nationality}}</td>

          <td>{{ passenger.gender}}</td>

          <td>

            {% if passenger.checked\_in\_status == 0 %}

                <a href="{% url 'staff\_check\_in' passenger.pk%}" class="btn btn-outline-danger" role="button">Check In</a>

            {% else %}

                <button type="button" class="btn btn-success disabled">Checked In</button>

            {% endif %}

          </td>

          <td>{{ passenger.cleared\_security\_status}}</td>

        </tr>

    {% endfor %}

  </tbody>

</table>

<a href="{% url 'generate\_report' flight\_no %}" class="btn btn-outline-success" role="button">Generate Flight Report</a>

  <!-- Trigger the modal with a button -->

  <button type="button" class="btn btn-danger" data-toggle="modal" data-target="#myModal">CLEAR FLIGHT FOR TAKE OFF</button>

  <!-- Modal -->

  <div class="modal" id="myModal" role="dialog">

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

        <h5 class="modal-title">Confirmation</h5>

        <button type="button" class="close" data-dismiss="modal" aria-label="Close">

            <span aria-hidden="true">&times;</span>

        </button>

        </div>

        <div class="modal-body">

        <p>Are you sure you want to clear flight for take off?</p>

        </div>

        <div class="modal-footer">

            <a href="{% url 'delete\_passengers' flight\_no %}" class="btn btn-danger" role="button">Yes, clear</a>

{#        <button type="button" class="btn btn-danger">Yes, clear</button>#}

        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

        </div>

      </div>

    </div>

  </div>

{% endblock %}

**staff\_login.html**

{% extends 'base.html' %}

{% block title %}Staff Log In{% endblock %}

{% block body %}

<div class="container-fluid">

        <div class="row">

            <div class="col-sm-6 offset-sm-3">

                <div class="panel panel-default">

                    <div class="panel-body">

                        {% if error\_message %}

                        <p><strong>{{ error\_message }}</strong></p>

                        {% endif %}

                        <form role="form" action="{% url 'staff\_login'%}" method="post" enctype="multipart/form-data">

                            {% csrf\_token %}

                          <fieldset>

                            <legend>Flight Staff Login</legend>

                            <div class="form-group">

                              <label for="InputUsername1">Username</label>

                              <input type="text" class="form-control" id="InputUsername1" name="username" aria-describedby="emailHelp" placeholder="Enter username">

                            </div>

                            <div class="form-group">

                              <label for="exampleInputPassword1">Password</label>

                              <input type="password" class="form-control" id="exampleInputPassword1" name="password" placeholder="Password">

                            </div>

                            <button type="submit" class="btn btn-primary">Submit</button>

                          </fieldset>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

{% endblock %}

**view\_flights.html**

{% extends 'base.html' %}

{% block title %}View Flights{% endblock %}

{% block body %}

<h3>Flights</h3>

<div>

<form class="form-inline my-2 my-lg-0" method="post" action="{% url 'search\_by\_source'%}" role="form">

  {% csrf\_token %}

  <input class="form-control mr-sm-2" placeholder="Search by Source" name="source" type="text">

  <button class="btn btn-secondary my-2 my-sm-0" type="submit" style="margin-right:10px;">Search</button>

</form>

<br>

<form class="form-inline my-2 my-lg-0" method="post" action="{% url 'search\_by\_destination'%}" role="form">

  {% csrf\_token %}

  <input class="form-control mr-sm-2" placeholder="Search by Destination" name="destination" type="text">

  <button class="btn btn-secondary my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

<br>

    <table class="table table-hover">

  <thead>

    <tr>

      <th scope="col">Flight No.</th>

      <th scope="col">Airline</th>

      <th scope="col">No. of seats</th>

      <th scope="col">Source</th>

      <th scope="col">Destination</th>

      <th scope="col">Departure Time</th>

      <th scope="col">Arrival Time</th>

    </tr>

  </thead>

  <tbody>

    {% for flight in flights %}

        <tr>

          <td>{{ flight.flight\_no}}</td>

          <td>{{ flight.airline\_name}}</td>

          <td>{{ flight.no\_of\_seats }}</td>

          <td>({{ flight.source\_code}}) {{ flight.source}}</td>

          <td>({{ flight.destination\_code}}) {{ flight.destination}}</td>

          <td>{{ flight.departure\_time}}</td>

          <td>{{ flight.arrival\_time}}</td>

          <td>

            {% if flight.no\_of\_seats > 0 %}

                <a href="{% url 'book\_flight' flight.pk%}" class="btn btn-outline-success" role="button">Book Now!</a>

            {% else %}

                <button type="button" class="btn btn-success disabled">Book Now!</button>

            {% endif %}

          </td>

        </tr>

    {% endfor %}

  </tbody>

</table>

{% endblock %}

**6.Modulus of proposed work**

An airport management system is a complex software system that involves various modules to handle different aspects of airport operations. Below are some key modules that could be part of an airport management system developed using Python. Each module corresponds to a specific functionality:

1. **User Authentication and Authorization:**
   * Manage user accounts for airport staff.
   * Define roles and permissions for different users (e.g., administrators, air traffic controllers, security personnel).
2. **Flight Scheduling:**
   * Schedule and manage flights.
   * Assign gates, runways, and parking spots for arrivals and departures.
3. **Passenger Management:**
   * Handle passenger check-ins and boarding.
   * Manage passenger information and reservations.
4. **Baggage Handling:**
   * Track and manage baggage from check-in to retrieval.
   * Ensure accurate baggage handling and routing.
5. **Air Traffic Control:**
   * Monitor and manage air traffic within the airport's airspace.
   * Coordinate with pilots for takeoffs and landings.
6. **Security:**
   * Implement security measures and protocols.
   * Monitor and control access to secure areas.
7. **Maintenance and Repairs:**
   * Schedule and track maintenance activities for runways, terminals, and other infrastructure.
   * Manage repairs and inspections.
8. **Financial Management:**
   * Handle financial transactions, such as ticket sales and airport fees.
   * Generate financial reports and analytics.
9. **Emergency Response:**
   * Implement emergency response protocols.
   * Coordinate with relevant authorities during emergencies.
10. **Reporting and Analytics:**
    * Generate reports on various aspects of airport operations.
    * Provide analytics for performance monitoring and optimization.
11. **Communication:**
    * Facilitate communication between airport staff, airlines, and other stakeholders.
    * Implement notifications and alerts for critical events.
12. **Aircraft and Fleet Management:**
    * Manage the airport's fleet of vehicles and ground support equipment.
    * Track aircraft movements and availability.
13. **Environmental Monitoring:**
    * Monitor environmental factors that could affect operations (e.g., weather conditions).
    * Plan for and respond to environmental challenges.

When developing an airport management system, it's essential to consider security, scalability, and reliability, given the critical nature of airport operations. Additionally, compliance with aviation regulations and standards is crucial for the system's success. Each of these modules can be implemented as Django apps within the overall Django project, ensuring a modular and scalable architecture.

**Using Django apps:**

1. **User Authentication and Authorization:**
   * Django app: **authentication**
   * Features: User registration, login, and role-based access control.
2. **Flight Scheduling:**
   * Django app: **flight scheduling**
   * Features: Flight creation, scheduling, gate assignment, runway assignment.
3. **Passenger Management:**
   * Django app: **passenger management**
   * Features: Passenger check-in, boarding, reservation management.
4. **Baggage Handling:**
   * Django app: **baggage handling**
   * Features: Baggage tracking, handling, and routing.
5. **Air Traffic Control:**
   * Django app: **air traffic control**
   * Features: Real-time monitoring of air traffic, coordination with pilots.
6. **Security:**
   * Django app: **security**
   * Features: Access control, monitoring secure areas, security protocols.
7. **Maintenance and Repairs:**
   * Django app: **maintenance**
   * Features: Schedule maintenance activities, track repairs and inspections.
8. **Financial Management:**
   * Django app: **financial management**
   * Features: Ticket sales, airport fees, financial reports.
9. **Emergency Response:**
   * Django app: **emergency response**
   * Features: Emergency protocols, coordination with authorities.
10. **Reporting and Analytics:**
    * Django app: **analytics**
    * Features: Generate reports on airport operations, provide analytics.
11. **Communication:**
    * Django app: **communication**
    * Features: Facilitate communication between staff and stakeholders, notifications.
12. **Aircraft and Fleet Management:**
    * Django app: **fleet management**
    * Features: Manage the airport's fleet, track aircraft movements.
13. **Environmental Monitoring:**
    * Django app: **environmental monitoring**
    * Features: Monitor environmental factors affecting operations, plan for challenges.

Each Django app can have its models, views, templates (HTML files), and static files. SQLite will be the default database for this example, but in a production environment, you might consider using a more robust database like PostgreSQL.

**Simplified structure:**

**airport\_management/**

**|-- authentication/**

**|-- flightscheduling/**

**|-- passengermanagement/**

**|-- baggagehandling/**

**|-- airtrafficcontrol/**

**|-- security/**

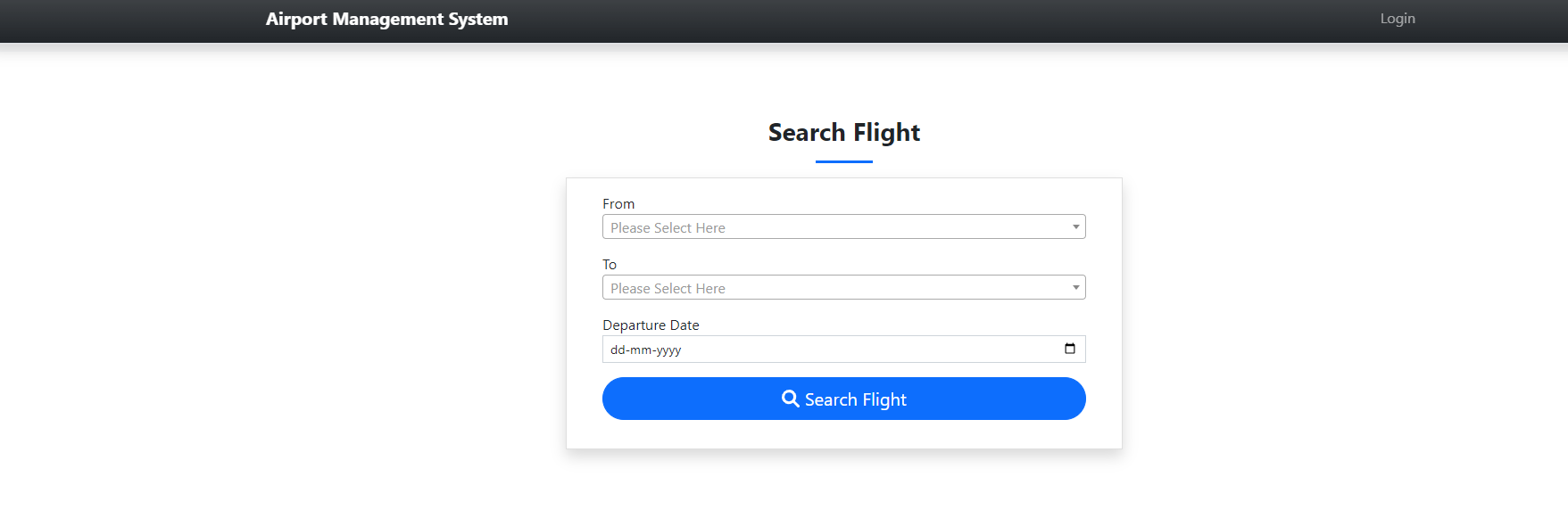
**|-- maintenance/**

**|-- financialmanagement/**

**|-- emergencyresponse/**

**7.Result**

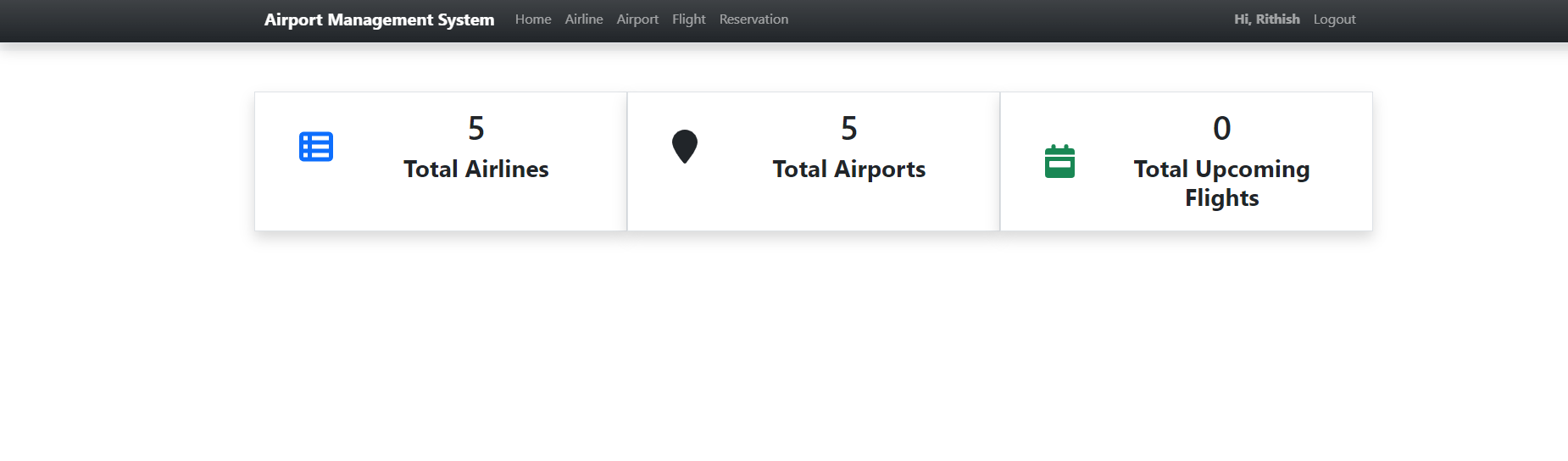
**Main Page**



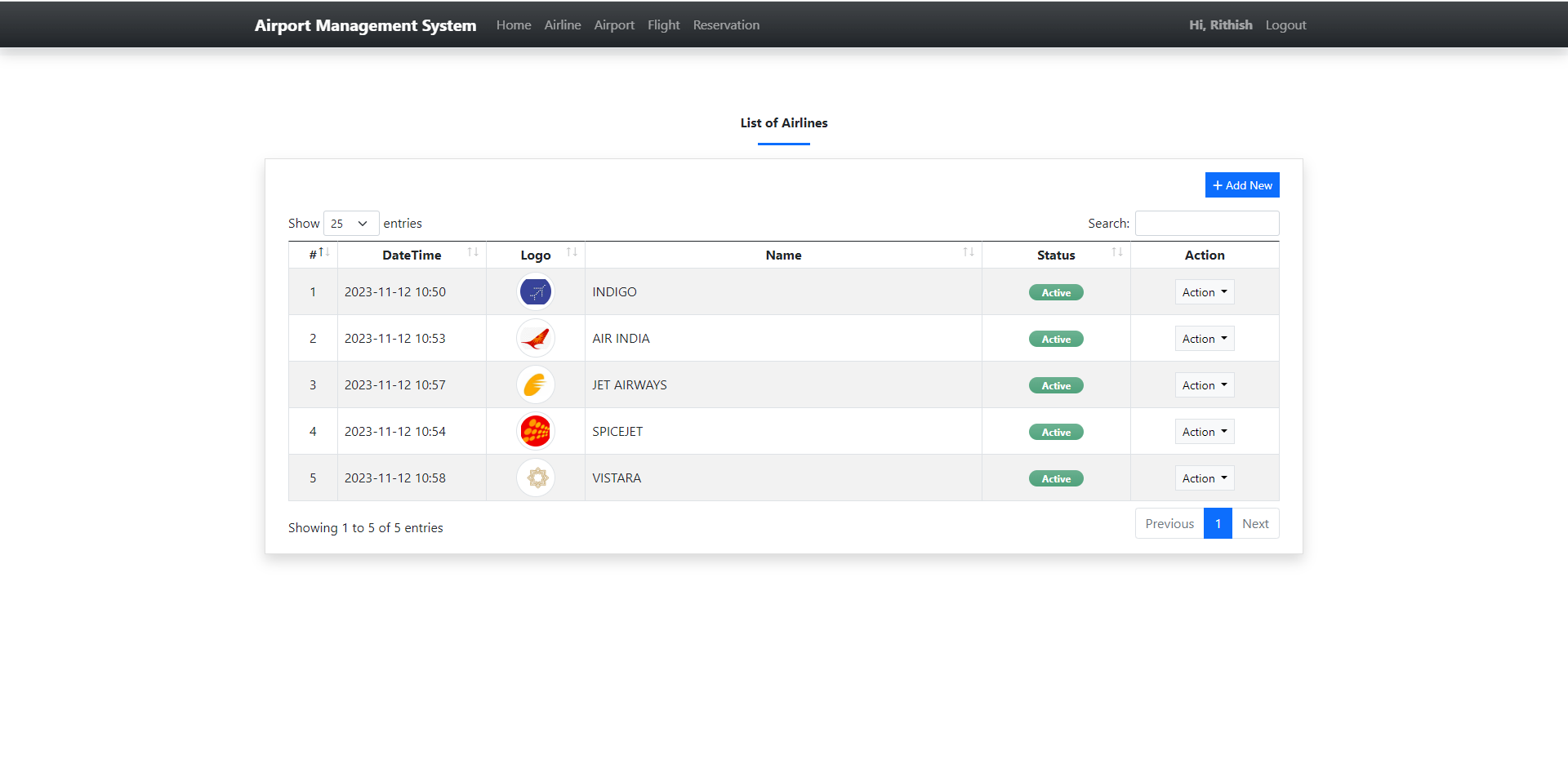
Login Page



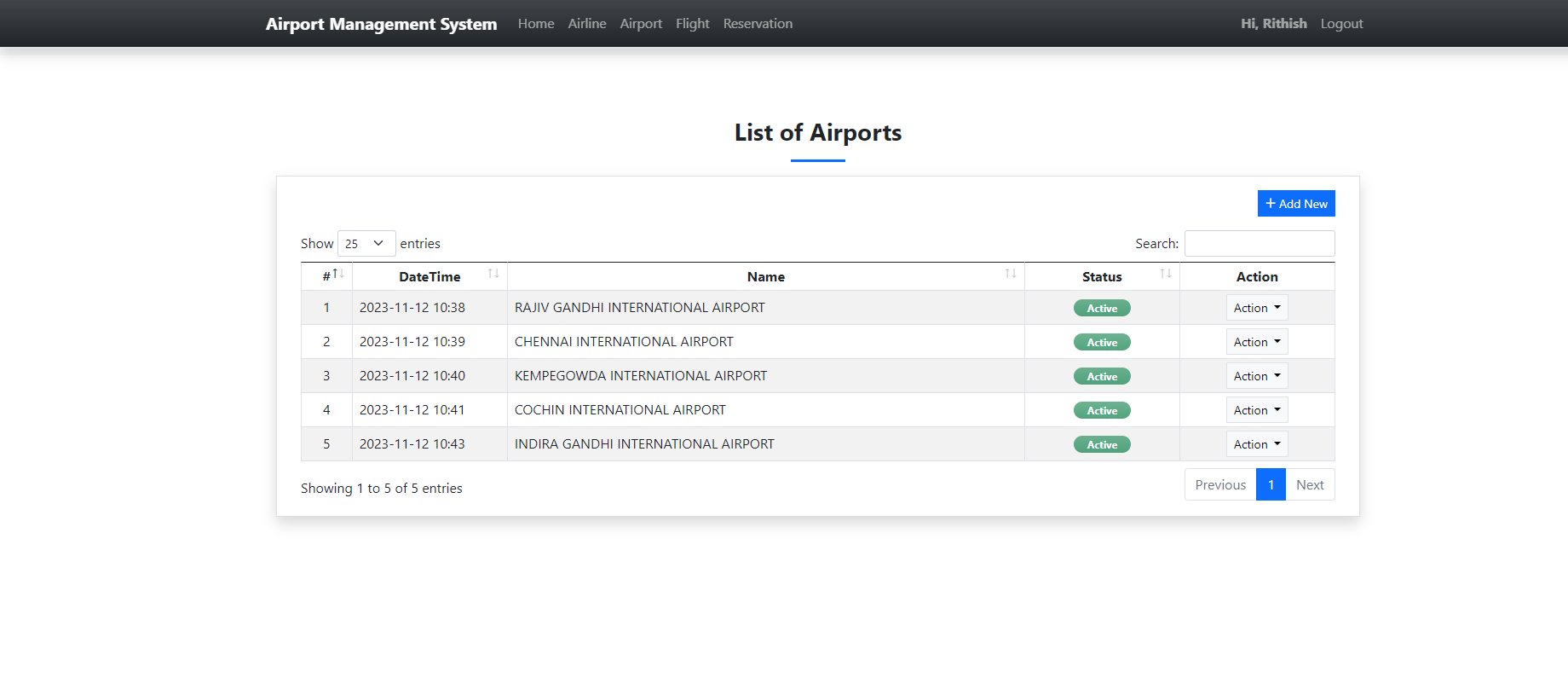
Home page



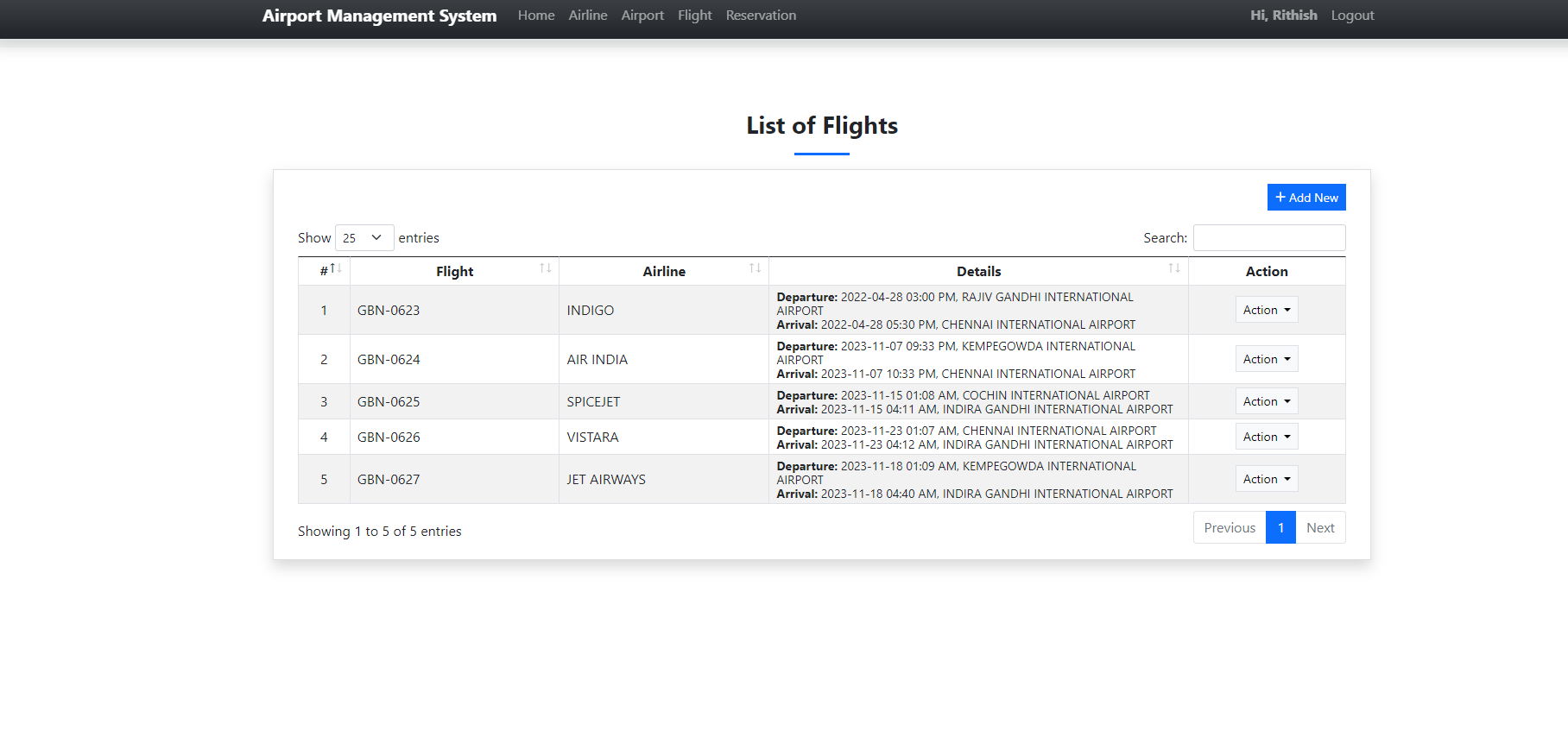
Airlines



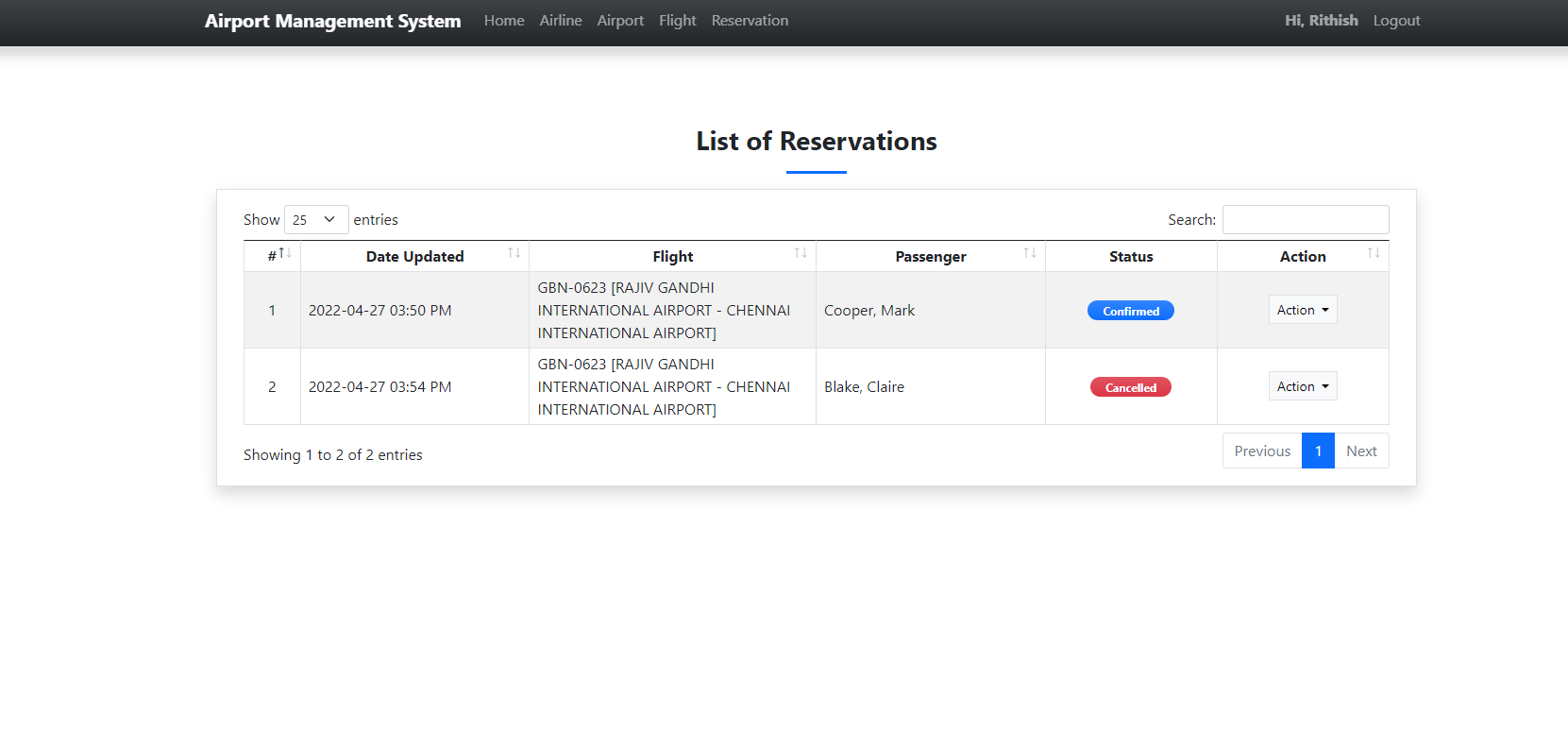
Airports



Flights



Reservations

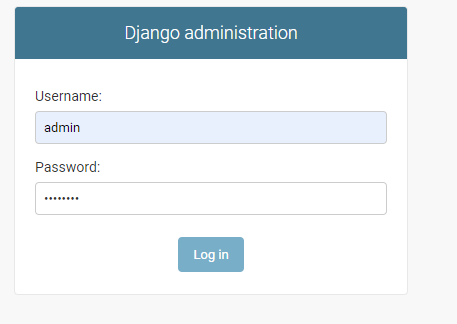


Logout

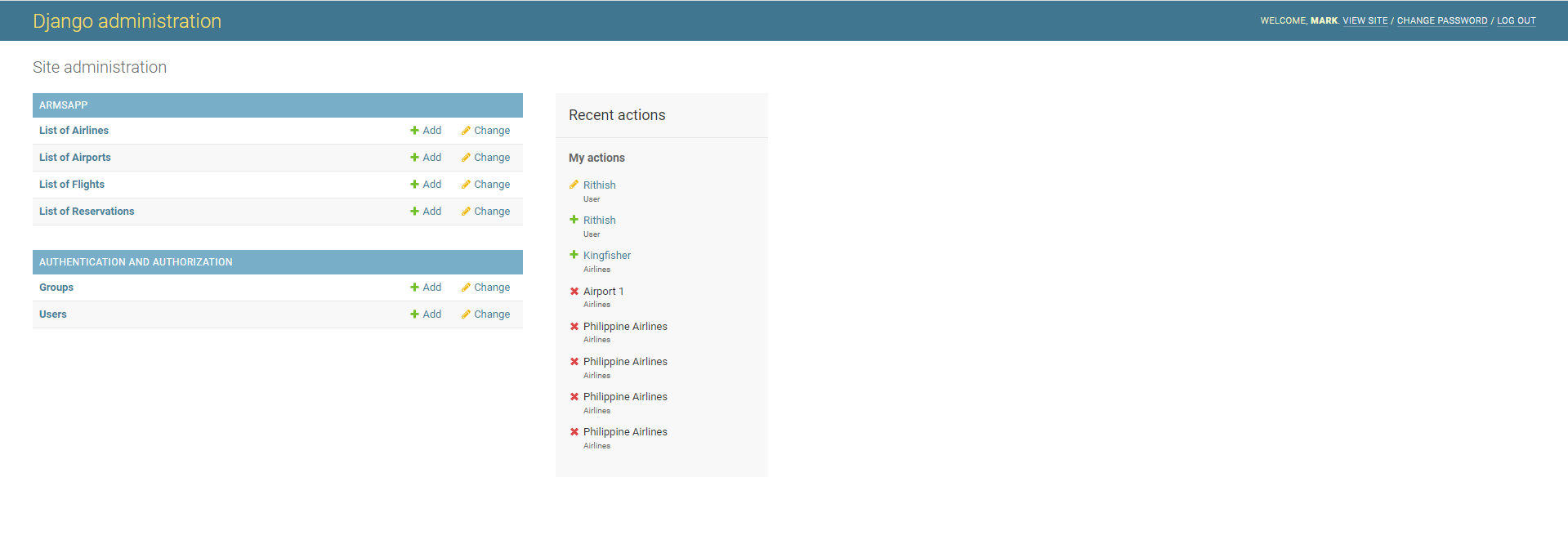


Admin

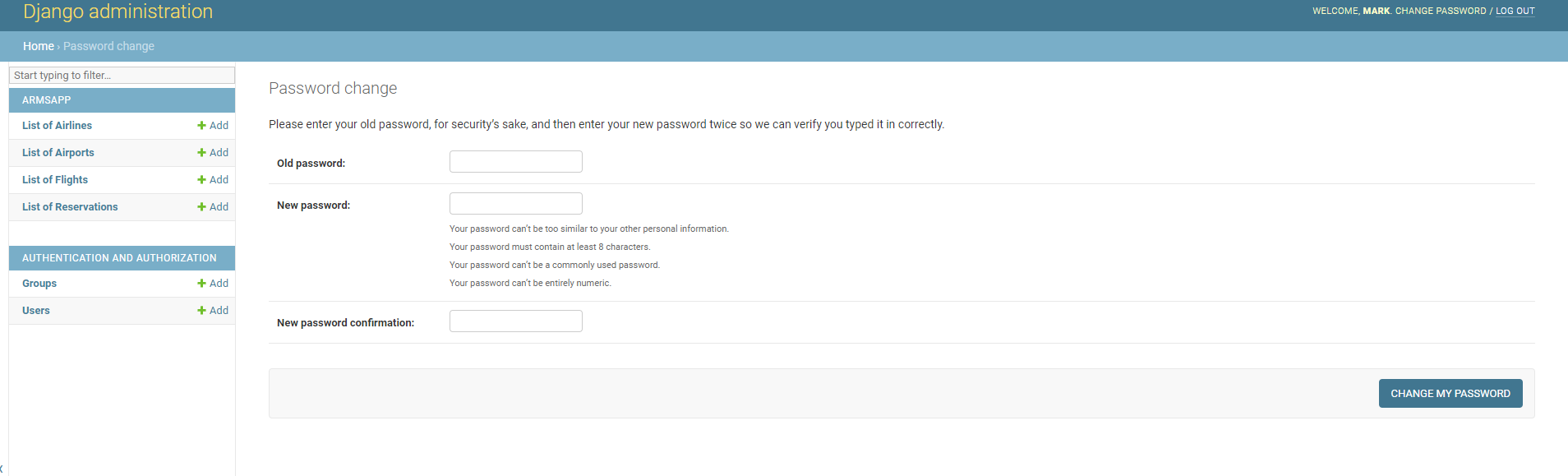
Login



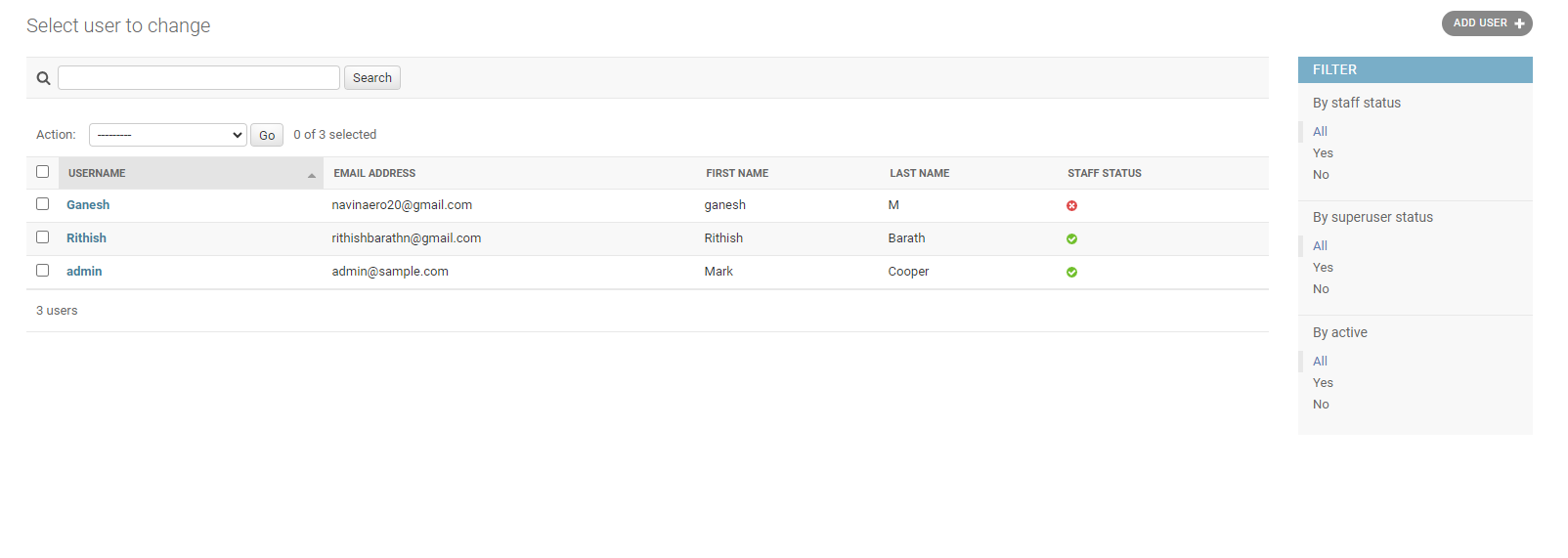
Home page

****

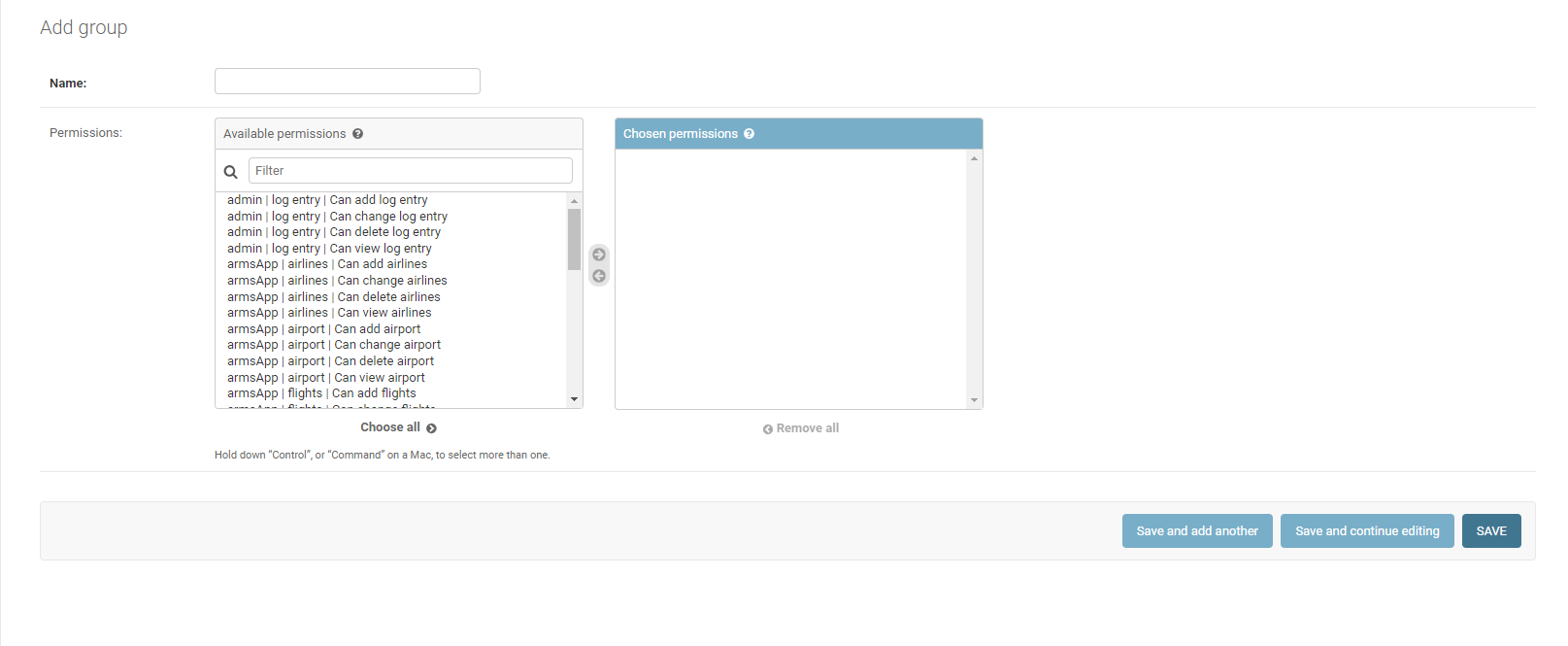
**Password settings**

****

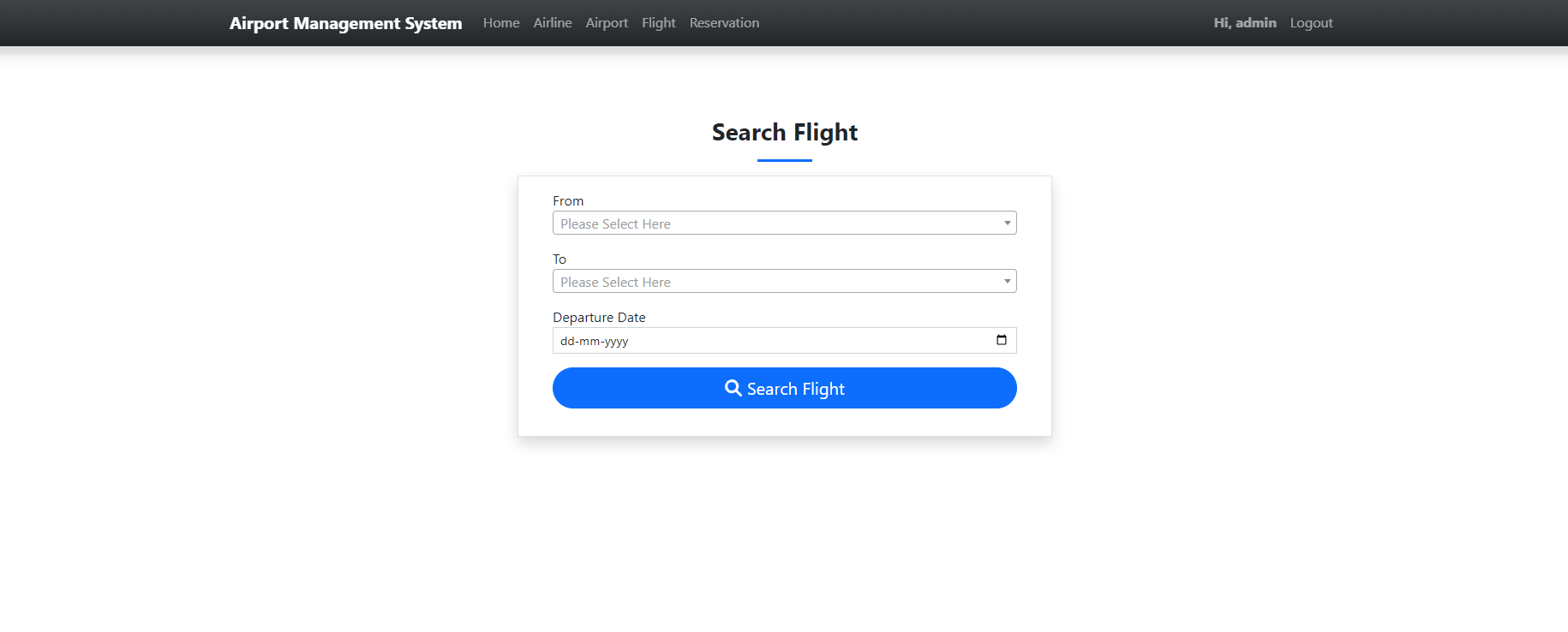
**Lists of users**

****

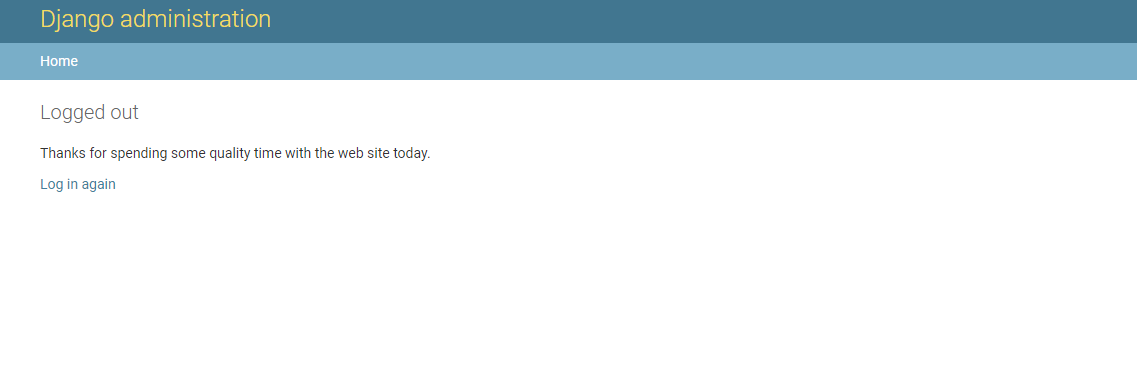
**Groups settings**

****

View



Logout



**8.conclusion**

The development of an airport management system using Python, Django, HTML, and SQLite represents a sophisticated undertaking with the potential to streamline and enhance various aspects of airport operations. The modular structure, organized into Django apps, enables a scalable and maintainable architecture, fostering easier development and future expansions. Each module addresses specific functionalities crucial to efficient airport management, ranging from user authentication and flight scheduling to security, financial management, and environmental monitoring.

The user authentication module establishes a secure foundation, allowing for role-based access control and management of user accounts. Flight scheduling ensures the seamless coordination of arrivals and departures, including gate and runway assignments. Passenger management and baggage handling modules enhance the passenger experience, providing tools for check-in, boarding, and efficient baggage tracking.

Air traffic control, security, and emergency response modules contribute to the safety and security of airport operations. Maintenance and repairs, along with fleet management, address the need for proactive infrastructure upkeep and efficient handling of the airport's vehicles and aircraft.

Financial management facilitates revenue generation through ticket sales and efficient management of airport fees. Reporting and analytics offer insights into operations, aiding in decision-making and optimization. Communication and environmental monitoring modules enhance overall coordination and resilience in the face of emergencies or environmental challenges.

By utilizing SQLite as the default database, the system maintains a lightweight yet functional database management system suitable for initial development. However, in a production environment, consideration may be given to employing a more robust database solution, such as PostgreSQL, to handle the complexity and scale of airport data.

Throughout the development process, adherence to security practices, regulatory compliance, and a user-centric design philosophy are paramount. The implementation of HTML templates ensures a visually intuitive and responsive user interface, contributing to an overall positive user experience.

In summary, the proposed airport management system, structured around Python, Django, HTML, and SQLite, offers a comprehensive solution to the multifaceted challenges of airport operations. Its modular design and use of modern technologies position it well for adaptability, scalability, and successful deployment in real-world airport environments.

**References**

**Sumitha Arora – python book**

**The python workbook**

**Python for beginners book**

[**https://youtu.be/HcOc7P5BMi4?si=1e8xIqI4WXuisGS7**](https://youtu.be/HcOc7P5BMi4?si=1e8xIqI4WXuisGS7)

[**https://youtu.be/xErUnOKQbFw?si=2bygJwPc8O0SKzJu**](https://youtu.be/xErUnOKQbFw?si=2bygJwPc8O0SKzJu)

[**https://youtu.be/HXV3zeQKqGY?si=Aguw-d3LBvvQms40**](https://youtu.be/HXV3zeQKqGY?si=Aguw-d3LBvvQms40)