FLIGHT SCHEDULE

PROJE GEREKSİNİMLERİ:

- API Kullanımı: Dünyadaki havalimanlarını Google map apiden Places api kullanarak çekeceğim.
- Havalimanları arası uzaklığı Distance Api kullanarak dataları çekeceğim. (API key bende mevcut ve kullanıma hazır.)
- Uçuş maliyetleri: uçağın doluluk oranına ve uçuş saatine göre yakıt tüketimini göz ardı ettik. Sadece mesafeye odaklı yakıt tüketimini hesaplamak istiyoruz bunun için kullanacağımız ortalama değer aşağıdaki gibidir. (Boeing 737 uçağının ort verileri)
- Mesafeye bağlı olarak yakıt tüketimi her 1.000 km için yaklaşık 5.000 litre ve maliyet olarak 6.250 USD civarındadır.(Yakıt tüketimi oranının 5 litre/km olduğu varsayımıyla hesaplanmıştır.)
- m tane uçuşa n tane crew ataması yap. Bu atamayı yaparken crew ekibinin çalışma sürelerini düzenlemesini yap ve her uçuşa doğru crewı ata.
-Aylık uçuşlar için database'i kendimiz oluşturacağız.

YAZILIMIN AMACI

- **API'den çekilen** gerçek uçuş verilerine göre veya databaseni oluşturduğumuz uçuş bilgilerine göre (nereden nereye uçulacağı) crew çalışma saatlerine göre crew atamasını gerçekleştirsin. Ek olarak datalardaki uçuş rotasyonlarını min yakıt masrafını hesaplayarak oluştursun.
- Nereden nereye uçulacağı verilerini aldıktan sonra bu verilere göre uçuş saatini distance API'den uzaklık hesaplayıp uçuş saatini ve bu uçuş saatine bağlı yakıt tüketimini ve yakıt tüketimine bağlı olarak da yakıt tüketim masrafını hesaplasın.
- Program çıktısında optimum şekilde rotalar oluştur

AMAÇ: Min yakıt kullanarak ve doğru crew ataması yaparak bir havayolu şirketinin akıllı uçuş rotasyonunu oluşturmak.

CREW ATAMA

- Crew atamasının datasını oluştur.

Maksimum Uçuş Süresi: Genellikle pilotlar ve kabin ekibi için günlük, haftalık ve aylık uçuş süreleri sınırlandırılmıştır. Örneğin:

Aylık Uçuş Süresi: 90 saat civarında bir üst sınır vardır.

Görev Türü	Görev Süresi
Kısa Mesafeli Uçuş	2-4 saat
Uzun Mesafeli Uçuş	6-12 saat

Program Çıktısı ve Görselleştirme:

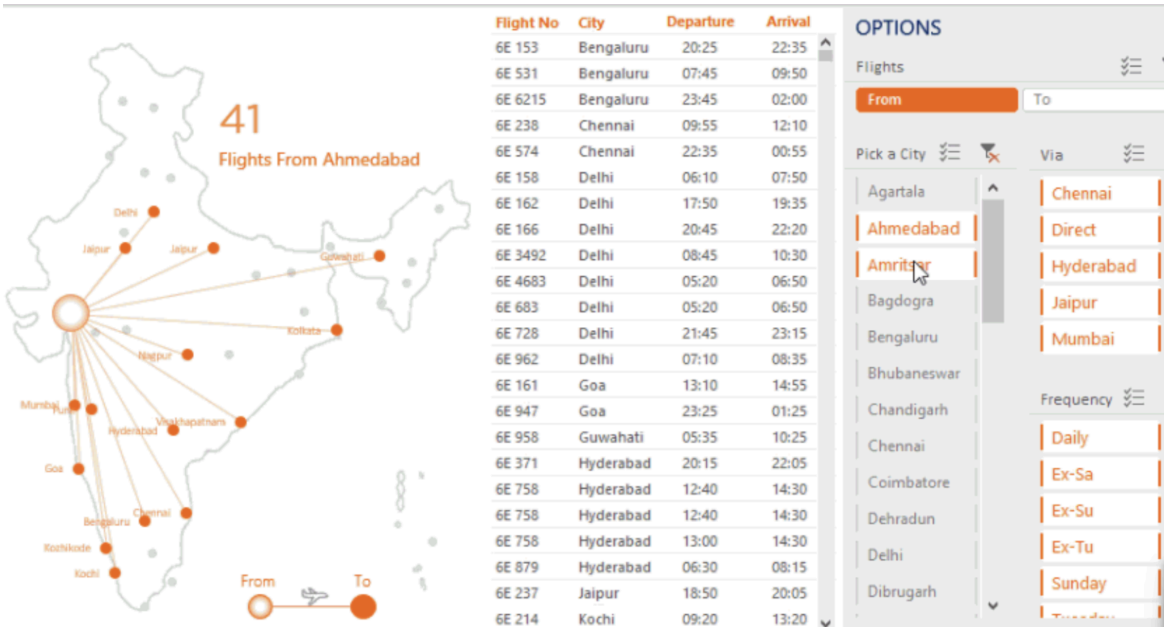
- Programı her çalıştırdığımızda kullanıcıdanaylık kaç uçuş istediğini sorsun.
- Rotalar arasında filtreleme yaparak.

- Proje sonucu animasyonlu en optimum rotaları Google haritalarında görselleştirsin. animasyonlu yapamazsak direkt rota lineları çıksın.
- Görselleştirdiği uçuşlardan seçip üstüne tıkladığımızda kullanıcıya yeni sayfada o uçuşa atanan mürettabatın bilgilerini listesini oluştursun.

- **SON GÖRÜNTÜ ÇIKTISI:**

Uçuşa tıkladığımızda ya da filtreleme yaptığımızda aşağıdaki çıktıları göstersin.
uçak id --- nereden-nereye uçtuğu---- Crew bilgileri-- Crew ekibi idleri(Crew ekibi ad soyad ve görevleri) ---kaç saat uçuş olduğu--toplam yakıt tüketimi--toplam yakıt tüketimi masrafı

https://goodly.co.in/flight-schedule-dashboard/



DATABASE

- CREW EKİBİ VERİLERİ: SQL'den çek:

Crew_id	Ad Soyad	Görev (pilot, kabin memuru)	Aylık max uçuş saati

- Flights: uçuş verileri bilgileri:sqlden çek

flightid	departure	arrival	duration_hours

- Airplanes: uçak bilgilerini sqlden çek.

uçak id	uçak adı	max monthly hours

- Program Çıktısı:UÇUŞ VERİLERİ: uçak kodu– uçuş id–Kalkış ve varış noktası – sqlden çek.

uçak_id	flight_id	departure	arrival	departure time	Arrival time	yakıt tüketim ne kadar	yakıt tüketim masrafı

Google Places API ve Distance API kullanarak kalkış ve varış noktası arası uzaklığı ve kaç saat uçuş olacağını sistem otomatik hesaplasın.

Proje Taslağı: Flight Schedule Optimizasyon Sistemi

1. Projenin Amacı

- Havayolu şirketlerinin uçuş rotasyonlarını optimize etmek.
- Minimum yakıt masrafıyla uçuş rotasyonları oluşturmak.
- Çalışma sürelerine uygun mürettebat (crew) atamaları yapmak.
- Kullanıcı dostu bir arayüz ve görselleştirme sunmak.

---

2. Kullanılacak Araçlar ve Teknolojiler

Amaç	Araç/ Teknoloji
API kullanımı	Google Places API, Google Distance API
Veritabanı	SQLite (Gelişim sürecinde), PostgreSQL (Dağıtımda)
Kodlama	Python

Framework	Flask (Web arayüzü), Jinja2 (HTML templating)
Görselleştirme	Folium, Matplotlib, veya Plotly Dash
Veritabanı Yönetimi	SQLAlchemy (ORM)
Kullanıcı Arayüzü	HTML, CSS, JavaScript
Ekipman ve Görev Yönetimi	Pandas, NumPy
Harita Üzerinde Görselleştirme	Google Maps API, Plotly

---

3. Modüler Yapı ve Kod Planlaması

Proje, kolay genişletilebilirlik ve düzenli geliştirme için modüler bir yapıya sahip olacak.

3.1. Dosya Organizasyonu

```
Flight_Schedule/
│
├── app/
│   ├── main.py             # Ana dosya
│   ├── routes.py           # Flask route tanımlamaları
│   ├── utils.py            # Yardımcı fonksiyonlar
│   ├── crew_assignment.py  # Mürettebat atama algoritmaları
│   ├── optimization.py     # Optimizasyon algoritmaları
│   ├── api_handler.py      # Google API entegrasyonları
│   ├── db/
│      ├── models.py        # SQLAlchemy modelleri
│      ├── database.db      # SQLite Veritabanı (geliştirme süreci için)
│      ├── seed_data.sql    # Veritabanı başlangıç verileri
│   ├── templates/
│      ├── index.html       # Ana HTML şablonu
│      ├── flight_info.html # Uçuş detayları için şablon
│   ├── static/
```

```
│       ├── css/            # CSS dosyaları
│       ├── js/             # JavaScript dosyaları
│       ├── images/         # Statik görseller
├── tests/
│   ├── test_api.py         # API birim testleri
│   ├── test_optimization.py # Optimizasyon birim testleri
│   ├── test_routes.py      # Route birim testleri
├── requirements.txt        # Gerekli Python kütüphaneleri
├── README.md               # Proje açıklaması
```

3.2. Modüller

1. **API Handler (`api_handler.py`)**
    - Google Places API ve Distance API ile entegrasyon sağlar.
    - Havalimanları arasındaki mesafeyi ve uçuş saatlerini hesaplar.
2. **Crew Assignment (`crew_assignment.py`)**
    - Mürettebat atama algoritmalarını içerir.
    - Çalışma saatlerine ve uçuş türüne uygun ekip ataması yapar.
3. **Optimization (`optimization.py`)**
    - Minimum yakıt masrafı için uçuş rotasyonlarını optimize eder.
    - En kısa mesafeleri ve yakıt tüketimini hesaplar.
4. **Database (`db/models.py`)**
    - SQLAlchemy kullanarak uçuş ve mürettebat veritabanını oluşturur.
5. **Web Routes (`routes.py`)**
    - Kullanıcıdan giriş alır ve sonuçları görselleştirir.
    - Veritabanına ve optimizasyon sonuçlarına erişim sağlar.

---

4. Proje İçeriği ve Adımları

4.1. Veritabanı Tasarımı

- **Crew Table**

    - `crew_id`: Primary key
    - `name`: İsim ve soyisim
    - `role`: Görev (pilot, kabin memuru)
    - `max_hours`: Aylık maksimum uçuş saati
- **Flight Table**
    - `flight_id`: Primary key
    - `plane_code`: Uçak kodu
    - `origin`: Kalkış noktası
    - `destination`: Varış noktası
    - `distance`: Uzaklık (API'den alınacak)

4.2. Kullanıcı Akışı

1. Kullanıcı uçuş verilerini yükler (haftalık kaç uçuş yapılacağı).
2. Sistem, uçuş rotalarını Distance API ile optimize eder.
3. Yakıt masrafları hesaplanır ve toplam maliyet çıkartılır.
4. Mürettebat ataması yapılır.
5. Sonuçlar Google Maps üzerinde görselleştirilir.
6. Uçuş detaylarına tıklayarak mürettebat bilgileri ve diğer detaylar gösterilir.

4.3. Çıktı Örneği

Uçuş Listesi:

Uçak ID: ABC123
Kalkış: İstanbul
Varış: Berlin
Uzaklık: 2,000 km
Yakıt Tüketimi: 10,000 litre
Yakıt Masrafı: $12,500
Crew: [John Doe (Pilot), Jane Smith (Kabin Memuru)]

-
- **Harita Görselleştirme:**
    - Folium/Plotly ile uçuş rotalarının animasyonlu gösterimi.
    - Uçuşa tıklandığında detayların bir pop-up ile açılması.

---

5. Test Planı

- API bağlantı testleri (Distance API, Places API).
- Veritabanı CRUD işlemleri testleri.
- Optimizasyon algoritması testleri.
- Mürettebat atama algoritması testleri.
- Kullanıcı arayüzü (UI/UX) testleri.

---

6. Zaman Çizelgesi

1. **Hafta 1-2:** API ve Veritabanı entegrasyonu.
2. **Hafta 3-4:** Mürettebat atama algoritmaları.
3. **Hafta 5:** Uçuş rotası optimizasyonu.
4. **Hafta 6:** Kullanıcı arayüzü ve görselleştirme.
5. **Hafta 7:** Test ve hata ayıklama.
6. **Hafta 8:** Proje raporlaması ve sunum.
