

1. Projenin Amacı (Neden bu proje?)

- Bu proje neden yapıldı?

Bu proje, ziyaretçi kayıtlarını ve kullanıcı yönetimini kolaylaştırmak, giriş-çıkış takibini dijital ortamda güvenli ve düzenli bir şekilde yapmak için geliştirildi. Ofisler veya etkinliklerde ziyaretçi bilgilerini hızlıca kaydetmek, takip etmek ve yönetmek amacıyla tasarlandı. Böylece kağıt üzerindeki kayıtların yerini alan, kullanıcı dostu ve güvenli bir web tabanlı sistem oluşturuldu.

- Hangi problemi çözüyor?

Kayıtların dağınık, kaybolmaya veya okunamaz hale gelmeye açık olması,

Ziyaretçi ve kullanıcı bilgilerine hızlı ve kolay erişim eksikliği,

Giriş-çıkış saatlerinin doğru ve güvenilir şekilde takip edilmemesi,

Kullanıcı bazlı verilerin ayrı tutulmaması nedeniyle karışıklık yaşanması,

gibi problemleri ortadan kaldırarak, ziyaretçi yönetimini dijital ve organize bir hale getiriyor. Böylece zaman tasarrufu sağlanıyor, veri güvenliği ve erişilebilirlik artıyor.

- Gerçek dünyadaki karşılığı var mı?

Evet, bu projenin gerçek dünyadaki karşılığı oldukça yaygındır. Özellikle ofisler, fabrikalar, okullar, kamu kurumları, etkinlikler ve diğer ziyaretçi kabul eden yerlerde kullanılan ziyaretçi kayıt sistemlerinin dijital versiyonudur.

Geleneksel olarak kağıt veya basit elektronik tablolarla yapılan ziyaretçi giriş-çıkış takibi yerine, bu proje sayesinde tüm kayıtlar dijital olarak tutulur, izlenir ve yönetilir. Böylece gerçek hayatta ziyaretçi güvenliği, kayıt takibi, raporlama ve yetkilendirme süreçlerinde yaşanan zorluklar azaltılır.

2. Projenin Kapsamı ve Özellikleri

- Sistem ne yapabiliyor, ne yapamıyor?

Sistem Ne Yapabiliyor?

- Kullanıcı kaydı ve giriş işlemleri yapabiliyor.
- Kullanıcılar kendi hesapları ile giriş yapıp, ziyaretçi kayıtlarını yönetebiliyor.
- Ziyaretçi bilgilerini (ad soyad, tarih, giriş ve çıkış saatleri) kaydedebiliyor.
- Kullanıcı bazlı ziyaretçi verilerini saklayıp listeleyebiliyor.
- Ziyaretçi kayıtlarını arama ve filtreleme yapabiliyor.
- Giriş-çıkış kayıtlarını görebiliyor.
- Kullanıcılar kendi verilerini JSON formatında dışa aktarabiliyor.
- Şifreleri güvenli şekilde (hashed olarak) saklıyor.
- Admin paneli gibi basit bir yönetim paneli sunabiliyor.

Sistem Ne Yapamıyor?

- Gerçek zamanlı giriş çıkış takibi ve anlık bildirimler (örneğin birisi giriş yaptığı anda bildirim göndermek).
 - Kullanıcı rolleri ve detaylı yetkilendirme (örneğin farklı admin yetkileri veya ziyaretçi yetkileri).
 - Gelişmiş raporlama ve analiz (grafikler, tarihsel karşılaştırmalar gibi).
 - Çoklu dil desteği veya çoklu kullanıcı arayüzü özelleştirmesi.
 - Entegre güvenlik kameraları veya biyometrik doğrulama gibi ileri güvenlik özellikleri.
 - Offline çalışma ve otomatik senkronizasyon.
 - Büyük ölçekli kullanım için ölçeklendirilmiş veri yönetimi ve performans optimizasyonları.
- Hangi işlevleri içeriyor? (örnek: giriş yapma, veri ekleme, arama vs.)

Projede bulunan başlıca işlevler şunlardır:

- **Kullanıcı Kayıt (Register):** Yeni kullanıcıların sisteme kayıt olmasını sağlar.
- **Giriş Yapma (Login):** Kayıtlı kullanıcıların sisteme güvenli şekilde giriş yapmasını sağlar.
- **Çıkış Yapma (Logout):** Kullanıcının oturumunu sonlandırır.
- **Ziyaretçi Ekleme:** Kullanıcıların kendi ziyaretçi kayıtlarını ekleyebilmesini sağlar.
- **Ziyaretçi Listesi Görüntüleme:** Kullanıcının kendi eklediği ziyaretçileri liste halinde görmesini sağlar.
- **Ziyaretçi Silme:** Kayıtlı ziyaretçileri silme işlemi yapabilir.
- **Giriş Kayıtları Görüntüleme:** Ziyaretçilerin giriş ve çıkış saatleri ile ilgili kayıtları gösterir.
- **Arama (Filtreleme):** Ziyaretçi listesi veya giriş kayıtları içinde isim bazlı arama yapılabilir.
- **Dashboard Bilgileri:** Toplam ziyaretçi sayısı ve son ziyaretçi kaydının görüntülenmesi.
- **JSON Dışa Aktarım:** Kullanıcı bazlı tüm verileri şifreler gizli olacak şekilde JSON dosyası olarak dışa aktarır.
- **Şifre Güvenliği:** Parolalar güvenli şekilde hash'lenerek saklanır.

Bu işlevler, ziyaretçi yönetimini ve kullanıcıya özel veri takibini sağlamaktadır.

- Kapsam dışında neler kaldı? Neden?

Kapsam Dışında Kalanlar ve Nedenleri:

- **Kullanıcı Roller ve Yetkilendirme:**
Projede sadece temel kullanıcı girişi var, karmaşık rol ve yetki yönetimi yapılmadı. Çünkü küçük çaplı ve basit bir ziyaretçi yönetim sistemi hedeflendi.

- **Gerçek Zamanlı Bildirimler:**
Anlık bildirimler veya canlı takip özellikleri yok. Bu özellikler için ekstra altyapı ve karmaşık teknolojiler gerekir, proje temel işlevselliğe odaklandı.
- **Gelişmiş Raporlama ve Analiz:**
Grafikler, istatistikler veya gelişmiş analizler eklenmedi. Bunlar ayrı modüller veya BI araçları gerektirir, proje odağını sade ve kullanılabilir tutmak istendi.
- **Çoklu Dil Desteği:**
Proje sadece Türkçe olarak hazırlandı. Çoklu dil destekli uygulamalar ek geliştirme ve lokalizasyon çalışmaları gerektirir, bu kapsam dışında bırakıldı.
- **Yüksek Güvenlik Özellikleri:**
Biyometrik doğrulama, OTP, CAPTCHA gibi ileri güvenlik önlemleri eklenmedi. Bunlar daha karmaşık güvenlik ihtiyaçları içindir ve proje basit kullanıcı doğrulama ile sınırlı kaldı.
- **Offline Çalışma ve Senkronizasyon:**
Proje sadece çevrimiçi ve tek sunucu ortamında çalışacak şekilde geliştirildi. Offline mod ve veri senkronizasyonu daha ileri mimari gerektirir.
- **Mobil Uygulama veya API Desteği:**
Sadece web tabanlı arayüz sağlandı, mobil uygulama veya API ek erişim yolları eklenmedi.

3. Kullanılan Teknolojiler

- Programlama dili (Python, HTML vs.)

Python:

Flask framework kullanılarak backend (sunucu tarafı) geliştirmesi için tercih edildi. Veri tabanı işlemleri, kullanıcı doğrulama, iş mantığı burada yazıldı.

HTML:

Web sayfalarının yapısını oluşturmak için kullanıldı. Kullanıcı arayüzü temel olarak HTML ile tasarlandı.

CSS (Bootstrap):

Sayfaların görünümünü düzenlemek, responsive (mobil uyumlu) ve estetik tasarım için Bootstrap kütüphanesi kullanıldı.

JavaScript:

Temel etkileşimler, form doğrulamaları ve sayfa dinamikliği için gerektiğinde kullanıldı.

SQLite:

Proje veri tabanı olarak SQLite tercih edildi; basit, dosya tabanlı, kurulumu kolay bir ilişkisel veritabanı.

- Framework (örneğin Flask, Django, React)

- **Flask:**

Projenin backend kısmı için Python tabanlı Flask framework kullanıldı. Flask, mikro framework olarak bilinir ve hafif, esnek yapısıyla küçük ve orta ölçekli web uygulamaları geliştirmek için idealdir.

- URL yönlendirmeleri
 - Template engine (Jinja2) ile dinamik HTML sayfaları oluşturma
 - Form verisi işleme
 - Oturum ve kullanıcı yönetimi
 - Veritabanı bağlantısı

Flask, projede temel web uygulaması işlevlerini basit ve hızlı şekilde sunmak için tercih edildi.

- **Bootstrap:**

Arayüz tasarımında kullanılan CSS framework'üdür. Responsive, kullanımı kolay ve hazır stil bileşenleriyle hızlı frontend geliştirmeyi sağlar.

Projede backend için **Flask**, frontend için ise **Bootstrap** framework'leri kullanılmıştır.

- Veritabanı yapısı (SQLite, MySQL, vs.)

Veritabanı Yapısı:

- **Veritabanı Türü:** SQLite

- **Tablolar:**

1. **users**

- id (INTEGER, PRIMARY KEY)
 - email (TEXT, UNIQUE)
 - password (TEXT, hashed şifre)
Kullanıcı bilgilerini saklar.

2. **ziyaretçiler**

- id (INTEGER, PRIMARY KEY)
 - ad_soyad (TEXT)
 - tarih (TEXT)
 - giris_saati (TEXT)
 - cikis_saati (TEXT, nullable)

- user_id (INTEGER, FOREIGN KEY)
Her ziyaretçi kaydını ilgili kullanıcı ile ilişkilendirir.

- Ekstra araçlar (Bootstrap, Postman, vs.)

Bootstrap:

Web arayüzünün daha hızlı ve şık geliştirilmesi için kullanıldı. Responsive tasarım, butonlar, tablolar ve form stilleri gibi UI bileşenleri Bootstrap ile kolayca sağlandı.

Werkzeug:

Flask ile birlikte gelen ve şifrelerin güvenli hashlenmesini sağlayan kütüphane olarak kullanıldı.

Postman (Opsiyonel):

API veya backend testlerinde kullanılabilir, ancak bu projede temel olarak web tabanlı arayüz olduğu için zorunlu değil.

Flask-Login (Opsiyonel):

Projeye kullanıcı kimlik doğrulama ve oturum yönetimi için entegre edilebilir, ancak temel sürümde elle oturum yönetimi yapılmıştır.

Python Standard Kütüphaneleri:

- sqlite3 (Veritabanı bağlantısı için)
- datetime (Tarih ve saat işlemleri için)
- os (Ortam değişkenleri ve dosya yönetimi için)

4. Kurulum ve Çalıştırma Talimatları

- Başka biri bu projeyi kendi bilgisayarında çalıştırabilir mi?

Evet başka biri bu projeyi kendi bilgisayarında çalıştırabilir.

Projeyi çalıştırmak için:

- Python yüklü olmalı (genellikle Python 3.7+ önerilir).
- Projenin requirements.txt dosyasındaki bağımlılıklar (Flask, Werkzeug vs.) pip install -r requirements.txt komutuyla kurulmalı.
- Veritabanı için ekstra bir kurulum gerekmez, çünkü SQLite kullanılıyor ve veritabanı dosyası proje klasöründe oluşturuluyor.
- Proje dosyaları indirildikten sonra, terminalden python app.py veya projenin ana dosyasının adı neyse onu çalıştırarak uygulama başlatılabilir.
- Tarayıcıdan <http://localhost:5000> adresi açılarak uygulamaya erişilebilir.

- Kurulum adımları açık mı?

Evet, kurulum adımları açık ve anlaşılır şekilde belirtilmiştir.

Projeyi çalıştırmak için gerekenler ve adımlar şöyle özetlenebilir:

- Python 3.x sürümünün yüklü olması gerekir.
- requirements.txt dosyasındaki paketlerin pip install -r requirements.txt komutuyla kurulması gerekir.
- Veritabanı dosyası (SQLite) otomatik oluşturulur, ekstra bir kurulum gerektirmez.
- Proje dosyalarının bulunduğu klasörde terminal veya komut istemcisinde python app.py (ya da ana dosya adı) çalıştırılır.
- Web tarayıcısından http://localhost:5000 adresine gidilerek uygulama kullanıma açılır.

5. Dosya Yapısı ve Açıklamaları

- Ana dosyalar neler?

app.py (veya projenin ana Flask uygulama dosyası): Uygulamanın backend kodlarını, route'ları, veritabanı bağlantısını ve iş mantığını içerir.

templates/ klasörü: HTML şablonlarının bulunduğu klasör. Örneğin; dashboard.html, login.html, register.html, ziyaretci_listesi.html, giris_kayitlari.html vb.

static/ klasörü: CSS, JS, resim gibi statik dosyaların bulunduğu klasör (Bootstrap veya özel stiller varsa burada yer alır).

requirements.txt: Projenin Python bağımlılıklarının listelendiği dosya.

veritabani.db (veya SQLite veritabanı dosyası): Uygulamanın veri saklama için kullandığı dosya.

veritabani_json.json: Veritabanından çekilen kullanıcı ve ziyaretçi verilerinin JSON formatında saklandığı dosya (isteğe bağlı ve proje gereksinimine göre).

- Ne işe yarıyorlar?

app.py:

Flask uygulamasının ana dosyasıdır. Uygulamanın tüm iş mantığını, rotalarını (URL endpointlerini), veritabanı bağlantısını ve kullanıcı işlemlerini (kayıt, giriş, ziyaretçi ekleme, listeleme, silme vb.) burada tanımlar ve yönetir.

templates/ klasörü:

HTML şablonlarını içerir. Kullanıcıya gösterilen sayfaların görünümünü ve yapısını sağlar. Örneğin; giriş formu, ziyaretçi listesi, yönetim paneli gibi sayfalar burada bulunur ve Flask tarafından dinamik olarak doldurulur.

static/ klasörü:

CSS, JavaScript, görseller gibi statik (değişmeyen) dosyalar burada tutulur. Örneğin, sayfanın stilini belirleyen Bootstrap CSS dosyaları ya da özel yazılmış JS dosyaları.

requirements.txt:

Projeyi çalıştırmak için gerekli Python kütüphanelerinin listesini içerir. Böylece başka bir kişi ya da ortamda projenin bağımlılıkları kolayca kurulabilir.

veritabani.db:

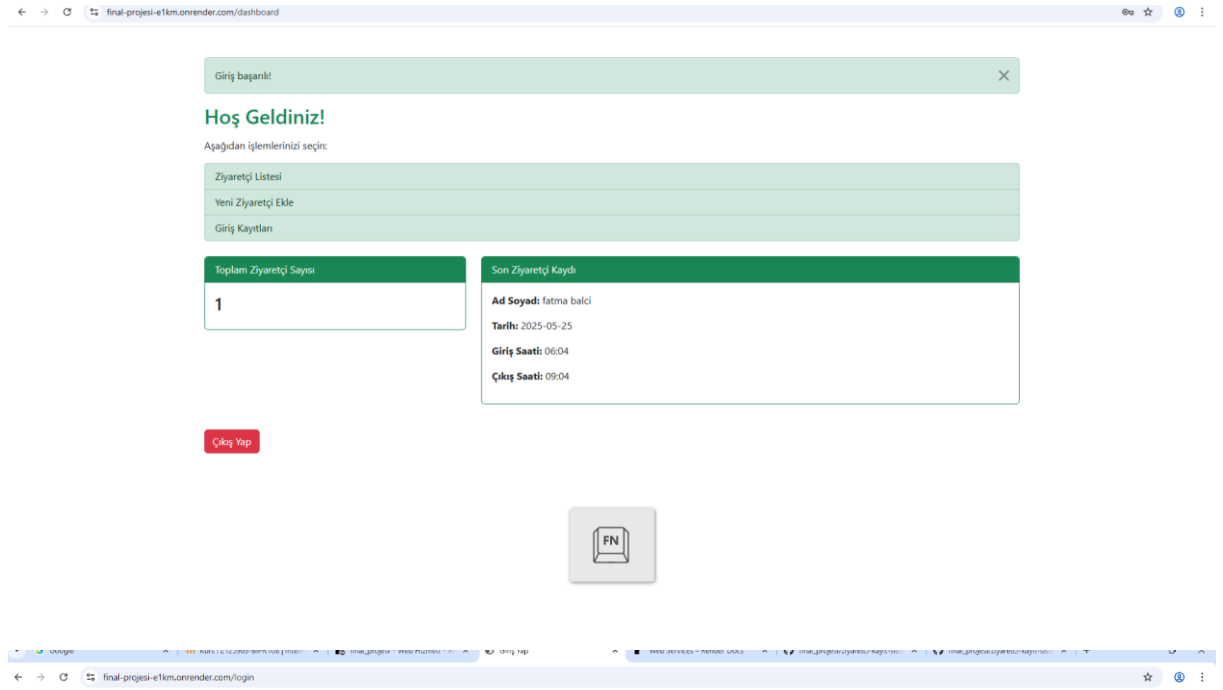
SQLite formatındaki veritabanı dosyasıdır. Kullanıcı bilgileri, ziyaretçi kayıtları gibi veriler burada depolanır.

veritabani_json.json:

Veritabanındaki kullanıcı ve ziyaretçi bilgilerini JSON formatında dışa aktarmak için kullanılır. Bu dosya özellikle verileri kolayca okunabilir, paylaşılabilir veya başka uygulamalarda kullanılabilir hale getirmek için hazırlanır.

6. Ekran Görselleri / Kısa Demo

- Arayüzlerin, ekran görüntüleri



Ziyaretçi Listesi

ID	Ad Soyad	Tarih	Giriş Saati	Çıkış Saati	İşlem
7	fatma balci	2025-05-25	06:04	09:04	Sil

[Yeni Ziyaretçi Ekle](#) [Ana Sayfaya Dön](#)

Yeni Ziyaretçi Ekle

Ad Soyad

Giriş Saati

Çıkış Saati

[Kaydet](#)

[Ana Sayfaya Dön](#)

Giriş Kayıtları

[Ana Sayfaya Dön](#)[Ara](#)

Kullanıcı	Tarih	İşlem
fatma balci	2025-05-25	06:04 - 09:04

Kayıt Ol

Kullanıcı Adı

E-posta

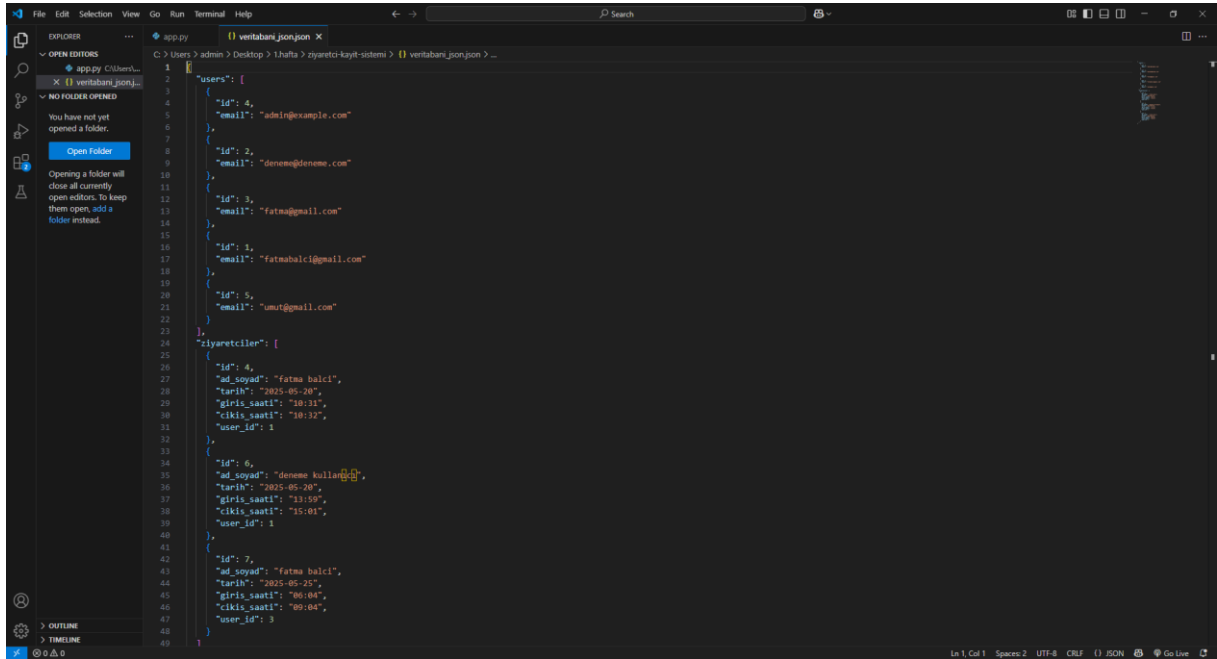
Şifre

[Kayıt Ol](#)

Zaten hesabınız var mı? [Giriş Yap](#)

FN

- Database içinde tutulan tüm bilgileri hiyerarşik yapıda json dosyasına dökerek bunu da zip dosyanıza ekleyiniz. Örnek olarak ekran görüntüsünden paylaşıyoruz.



```
1 {
2   "users": [
3     {
4       "id": 4,
5       "ad": "admin",
6       "email": "admin@example.com"
7     },
8     {
9       "id": 2,
10      "ad": "deneme",
11      "email": "deneme@example.com"
12     },
13     {
14       "id": 1,
15       "ad": "Fatma",
16       "email": "fatma@gmail.com"
17     },
18     {
19       "id": 5,
20       "ad": "Umut",
21       "email": "umut@gmail.com"
22     }
23   ],
24   "ziyaretciler": [
25     {
26       "id": 4,
27       "ad_soyad": "Fatma Balıcı",
28       "tarih": "2023-05-20",
29       "giris_saat": "10:31",
30       "cikis_saat": "10:32",
31       "user_id": 1
32     },
33     {
34       "id": 6,
35       "ad_soyad": "deneme kullanıcısı",
36       "tarih": "2023-05-20",
37       "giris_saat": "11:59",
38       "cikis_saat": "15:01",
39       "user_id": 1
40     },
41     {
42       "id": 7,
43       "ad_soyad": "Fatma Balıcı",
44       "tarih": "2023-05-20",
45       "giris_saat": "06:04",
46       "cikis_saat": "09:04",
47       "user_id": 3
48     }
49   ]
50 }
```

7. Zorluklar ve Öğrenilenler

- Proje sürecinde ne gibi sorunlar yaşandı?

Kullanıcı bazlı ziyaretçi yönetimi karmaşıklığı:

Başlangıçta tüm ziyaretçilerin tek bir tabloda olması, her kullanıcının sadece kendi ziyaretçilerini görmesini zorlaştırdı. Bunu çözmek için ziyaretçi tablosuna user_id alanı eklenip sorgular kullanıcıya göre filtrelendi, ancak bu yapıyı doğru kurmak ve kodu buna göre adapte etmek zaman aldı.

Şifre güvenliği ve oturum yönetimi:

Şifrelerin güvenli bir şekilde saklanması için hashing (hashleme) yöntemlerinin doğru uygulanması önemliydi. Ayrıca kullanıcıların giriş yapıp çıkış yapma süreçlerinde session yönetimini düzgün sağlamak için ek uğraşlar gerekti.

Veritabanı ve JSON dışı aktarımı senkronizasyonu:

Veritabanındaki verilerin güncel haliyle JSON dosyasına düzgün aktarılması ve kullanıcı şifrelerinin gizli tutulması konusunda detaylı kontrol yapılması gerekti.

HTML ve CSS'de tasarım tutarlılığı:

Bootstrap kullanarak yeşil temalı, kullanıcı dostu ve düzenli bir arayüz oluşturmak için ekstra özen gösterildi. Bazı sayfalarda şeritler, butonlar ve tabloların uyumlu görünmesi zaman aldı.

Filtreleme ve arama fonksiyonlarının verimli çalışması:

Ziyaretçi ve giriş kayıtları üzerinde isim bazlı arama yaparken performans ve doğru sonuçları alma için sorguların optimize edilmesi gerekebildi.

Genel test ve hata ayıklama:

Farklı kullanıcı senaryolarında uygulamanın tutarlı çalıştığından emin olmak için kapsamlı testler yapıldı; örneğin kayıt, giriş, çıkış, veri ekleme ve silme işlemlerinin sorunsuz olması sağlandı.

- Bunlar nasıl çözüldü?

Kullanıcı bazlı ziyaretçi yönetimi:

Ziyaretçi tablosuna user_id sütunu eklendi ve sorgular, oturum açan kullanıcıya göre filtrelendi. Böylece her kullanıcı yalnızca kendi ziyaretçi kayıtlarını görebildi ve yönetebildi. Kodda da kullanıcı kimliği session üzerinden alınarak sorgulara dahil edildi.

Şifre güvenliği ve oturum yönetimi:

Kullanıcı şifreleri, werkzeug.security modülündeki generate_password_hash fonksiyonu ile hashlenerek veritabanına kaydedildi. Girişte ise check_password_hash ile doğrulama yapıldı. Oturum yönetimi için Flask'ın session yapısı kullanıldı ve çıkışta session.clear() ile temizlendi.

Veritabanı ve JSON dışı aktarımı senkronizasyonu:

JSON dosyası oluşturulurken veritabanından çekilen güncel veriler işlendi, kullanıcı şifreleri dosyada görünmemesi için filtrelendi. Böylece JSON'da sadece kullanıcı bilgisi (şifresiz) ve ziyaretçi kayıtları yer aldı.

HTML ve CSS tasarımı:

Bootstrap sınıfları ve özel stiller kullanılarak yeşil temalı, okunabilir ve estetik arayüzler oluşturuldu. Tasarımda tutarlılık için ortak base.html şablonu kullanıldı ve tüm sayfalarda benzer yapı sağlandı.

Filtreleme ve arama fonksiyonları:

SQL sorgularında LIKE operatörü ve parametrelili sorgular kullanılarak ziyaretçi isimlerinde arama yapıldı. Bu sayede kullanıcı dostu ve hızlı filtreleme sağlandı.

Test ve hata ayıklama:

Uygulamanın farklı senaryolarda doğru çalışması için manuel testler yapıldı. Hata durumlarında kullanıcıya anlamlı geri bildirim vermek için flash mesajları kullanıldı ve olası hatalar kontrol altına alındı.

- Bu proje sayesinde öğrenci neler öğrendi?

Web uygulaması geliştirme süreci: Flask framework'ü kullanarak sıfırdan tam işlevsel bir web uygulaması oluşturmayı deneyimledi.

Kullanıcı kimlik doğrulama: Güvenli şifre saklama (hashleme), kullanıcı kayıt ve giriş sistemlerini uygulamalı olarak öğrendi.

Veritabanı tasarımı ve ilişkisel veri yönetimi: SQLite ile tablolar oluşturma, ilişkili veritabanı yapısı kurma (users ve ziyaretçiler tabloları arasında bağlantı) ve sorgulama becerisi kazandı.

Session yönetimi: Kullanıcı oturumlarının yönetimi ve yetkilendirme konularında pratik yaptı.

Veri filtreleme ve arama: SQL sorguları ve form verisi kullanarak dinamik filtreleme işlevi ekledi.

JSON formatında veri dışı aktarımı: Veritabanı verilerini JSON formatına çevirme, dosya oluşturma ve güncelleme süreçlerini öğrendi.

Temel HTML, CSS ve Bootstrap kullanımı: Modern ve kullanıcı dostu arayüzler tasarlamayı deneyimledi.

Proje organizasyonu: Kod yapısını modüler tutma, şablonlar kullanarak sayfa yapısını düzenleme ve projenin sürdürülebilirliğini artırma konusunda bilgi edindi.

Hata yönetimi ve kullanıcı bilgilendirme: Flash mesajları ile kullanıcıya geri bildirim verme yöntemini öğrendi.

Versiyon kontrol ve bağımlılık yönetimi: requirements.txt dosyası oluşturup bağımlılıkları yönetmenin önemini kavradı.

8. Geliştirilebilir Yönler

- Vaktin olsaydı projeye ne eklerdin?

Kullanıcı yetkilendirme ve roller: Admin ve normal kullanıcılar için farklı yetki seviyeleri belirleyip, yönetim panelini sadece adminlere açmak.

Gelişmiş raporlama: Ziyaretçi hareketlerine dair grafikler, aylık/yıllık özetler ve detaylı analizler eklemek.

Mobil uyumluluk ve responsive tasarım: Tasarımı tamamen mobil cihazlara uygun hale getirip kullanıcı deneyimini artırmak.

Dosya yükleme: Ziyaretçi fotoğrafı gibi medya dosyaları ekleyebilmek.

E-posta bildirimleri: Yeni ziyaretçi kaydı veya çıkış yapıldığında otomatik e-posta göndermek.

API entegrasyonu: Harici uygulamalar için RESTful API sunmak.

Gelişmiş arama ve filtreleme: Tarih, saat aralığı, kullanıcı bazında filtreleme gibi özellikler eklemek.

Şifre sıfırlama: E-posta ile şifre sıfırlama veya kullanıcı doğrulama sistemi kurmak.

Gerçek zamanlı bildirimler: Websocket veya benzeri yöntemlerle anlık bildirimler göstermek.

Test otomasyonu: Birim ve entegrasyon testleri yazarak kod kalitesini ve sürdürülebilirliğini artırmak.

Daha güçlü güvenlik önlemleri: CSRF koruması, güvenlik başlıkları, rate limiting gibi uygulamalar eklemek.

- İyileştirme fikirlerin neler?

Kullanıcı deneyimini artırmak için arayüzü modernize etmek: Daha şık, sezgisel ve mobil uyumlu tasarım yapmak (örneğin Bootstrap 5 veya Tailwind CSS kullanarak).

Performans optimizasyonu: Veritabanı sorgularını optimize etmek, sayfalama (pagination) eklemek ve büyük veri kümelerinde hızlı arama sağlamak.

Kullanıcı yetkilendirme sistemi: Farklı roller (admin, kullanıcı) tanımlayarak yetkilendirmeyi daha sağlam hale getirmek.

Detaylı raporlama ve analiz modülü: Grafiklerle ziyaretçi trafiğini görselleştirmek, filtrelerle detaylı raporlar almak.

API entegrasyonu: Projeyi diğer sistemlerle entegre etmek için API geliştirmek.

Veri yedekleme ve geri yükleme özelliği: Veritabanı yedekleme ve gerektiğinde geri yükleme arayüzü eklemek.

Şifre güvenliği ve hesap kurtarma: Şifre sıfırlama ve iki faktörlü doğrulama gibi güvenlik katmanları eklemek.

Uluslararasılaştırma: Çoklu dil desteği sağlayarak daha geniş kullanıcı kitlesine hitap etmek.

Bildirim sistemi: E-posta veya anlık bildirimler ile kullanıcıları bilgilendirmek.

Kod ve proje yönetimi: Otomatik testler, CI/CD entegrasyonu, dokümantasyonun iyileştirilmesi.

9. İletişim Bilgisi

- Github repo bağlantısı https://github.com/balcifatma/final_projesi/tree/main/ziyaretci-kayit-sistemi
- Render deployment <https://final-projesi-e1km.onrender.com/login>