FINAL PROJESI (RAPOR)

Kişisel projemle ilgili raporu sizlere sunmaktan mutluluk duyuyorum: A'dan Z'ye tasarlayıp geliştirdiğim web tabanlı bir personel yönetim uygulaması. Amacım, bir kullanıcının kendi personel dosyalarını yönetmesi için basit, güvenli ve etkili bir araç yaratmaktı.

Bu Proje Neden ve Hangi Teknolojileri Kullandım

Başından beri küçük ölçekte de olsa gerçek bir ihtiyaca cevap verecek somut bir şey inşa etmek istiyordum. Flask, URL yönlendirmesinden HTML sayfa oluşturmaya kadar Python web geliştirmenin temellerini öğrenmeme yardımcı oldu.

Özellikle kimlik doğrulamada güvenliğe önem verildi. Bu yüzden kullanıcı oturumlarını, girişlerini ve çıkışlarını yönetmek için Flask-Login'i entegre ettim. Ve şifreleri asla düz metin olarak saklamamak (altın kural!), şifrelerin güvenli bir şekilde saklanmasını ve doğru bir şekilde doğrulanmasını sağlayan Werkzeug.security'nin karma araçlarını kullandım.

Uygulamam Ne Yapabilir (ve Bunu Nasıl Yaptım)

Hedefime ulaşmak için bir dizi temel özelliği uygulamaya koydum:

Tam Kimlik Doğrulama Yönetimi:

Kayıt: Yeni kullanıcılar kendilerine özel bir e-posta adresi ve benim hemen belirlediğim bir şifre ile hesap oluşturabilirler. Zaten kullanımda olan bir e-posta adresiyle kayıt olamayacağınızdan emin oldum.
Giriş: Mevcut kullanıcılar güvenli bir şekilde giriş yapabilirler. Şifrelerinin karma halini kontrol ediyorum, gerçekten onlara ait olduğundan emin olmak için.
Çıkış: Çok basit, tek tıklamayla oturum sona erer.
Korunmuş Erişim: Kullanıcı oturum açtıktan sonra bir gösterge panosuna erişir. Bağlantı olmadan personel yönetim özelliklerine erişim mümkün değildir. Bu, önemli rotalarıma yerleştirdiğim @login_required dekoratörünün rolüdür.
Personel Yönetimi (Ünlü CRUD operasyonları):

Personel Ekleme: Her kullanıcı yeni personel kayıtları (soyadı, adı, pozisyonu, departmanı) ekleyebilir. Eklenen bonus ise her dosyanın otomatik olarak onu oluşturan kullanıcıya bağlanmasıdır.
Personelimi Görüntüle: Panoda, bir kullanıcının eklediği tüm personeli listeliyorum. Daha net bir genel bakış için özel bir sayfa da mevcut.
Personel Düzenle: Mevcut bir kaydın bilgilerini düzenlemek için bir fonksiyon uyguladım. Önemli nokta: Kullanıcının yalnızca kendi dosyalarını değiştirebildiğini, başkalarının dosyalarını değiştiremediğini kontrol ettim. Bu, veri güvenliği açısından esastır.
Bir Personeli Silme: Silme işleminde de aynı şeyi yaptım, bir kullanıcının yalnızca kendi kayıtlarını silebileceğinden emin oldum.

Gelişmiş Kullanıcı Deneyimi: Kullanıcıya geri bildirim vermek için flaş mesajlar kullandım. Örneğin, başarılı bir kayıt işleminin ardından veya giriş hatası durumunda sizi bilgilendirmek için küçük bir mesaj görüntülenir.

Yaklaşımım ve Ondan Öğrendiklerim

Kodumu oldukça açık bir şekilde yapılandırdım; veri modellerini, rotaları ve yönetim fonksiyonlarını ayırdım. Benim için okunabilir ve bakımı kolay kodlar önemliydi.

Bu proje web geliştirme konusundaki bilgilerimi pekiştirmemi sağladı. ORM ile veritabanlarını daha sezgisel olarak nasıl yöneteceğimi, kimlik doğrulamanın nasıl güvenli hale getirileceğini ve verilerin nasıl korunacağını öğrendim. En büyük zorluk, her kullanıcının yalnızca kendi verileriyle etkileşime girebilmesini sağlamaktı ve bunu etkili bir şekilde başardığımı düşünüyorum.

Peki projemin yapısı nasıl olacak?

static/

Bu dizin, tarayıcı tarafından doğrudan sunulan uygulamamın statik dosyalarına ayrılmıştır.

image/ : Web sayfalarımda kullandığım görselleri (logolar, ikonlar, vb.) saklamak için bir alt klasör.

styles.css: Ana CSS stil sayfam. Uygulamamın görsel görünümünü (renkler, yazı tipleri, düzen) burada tanımlıyorum.

şablonlar/

Bu dizin kullanıcı arayüzüm için olmazsa olmazdır. Flask tarafından oluşturulan tüm HTML dosyalarını (Jinja2 şablonları) içerir. Her dosya bir sayfayı veya sayfa bileşenini temsil eder:

add_personnel.html: Yeni personel eklemek için kullanılan form.

dashboard.html: Giriş yaptıktan sonra erişilebilen ana sayfa, muhtemelen bir özet veya personel listesi görüntülüyor.

edit_personnel.html: Mevcut bir personel üyesinin bilgilerini düzenlemek için kullanılan form.

home.html: Uygulamanın ana sayfası (giriş yapmadan önce).

login.html: Giriş formu.

personnel_list.html: Kullanıcının tüm personel listesini görüntülemek için ayrılmış bir sayfa.

register.html: Yeni kullanıcılar için kayıt formu.

Python ve json :

all2json.py ve db2json.py: Bu Python betikleri, verileri JSON formatlarına aktarmak veya JSON formatlarından içe aktarmak, yedekleme, transfer veya test amaçlarıyla kullanmak üzere veri işleme yardımcı programları önerir.

app.py: Bu Flask uygulamamın kalbidir. Tüm uygulama mantığını içerir: Flask yapılandırması, SQLAlchemy model tanımı (Kullanıcı, Personel), görünümler (rotalar), kimlik doğrulama mantığı (giriş, kayıt, çıkış) ve personel için CRUD işlemleri.

personnel.json ve users.json: Bu JSON dosyaları muhtemelen *2json.py betikleri tarafından veya ilk geliştirme için kullanılan personel ve kullanıcılar için örnek veya test verileri içerir.

personnel.py: Bu dosya, personel yönetimiyle ilgili fonksiyonları veya sınıfları içeren ayrı bir Python modülü olabilir veya app.py'ye entegre edilmeden önce Personel modeli tanımının daha eski bir sürümü olabilir.

Kısacası, güvenli bir temele ve açık bir mantığa sahip, işlevsel bir personel yönetim sistemi kurdum. Bu, Python ile tam yığın geliştirmeyi anlamamda önemli bir adım ve sonuçtan gurur duyuyorum.