**Proje Adı:** Python ile Kurumsal E-posta İmza Yönetim Sistemi

**Projenin Amacı:** Şirket çalışanlarının Outlook e-posta imzalarını merkezi olarak yöneterek kurumsal imajı standartlaştırmak, sistem yöneticisinin iş yükünü önemli ölçüde azaltmak, kullanıcı dostu bir yönetim panosu sunmak, kurum içi Active Directory (AD) ile sorunsuz entegrasyon sağlamak ve yıllık programın satın alma lisansı ile sürdürülebilir bir gelir modeli oluşturmaktır.

Amacımız Şirkette çalışan birinin(admin) tüm personelin(AD deki kullanıcılar) Outlook mail imzasını tek bir yerden oluşturup tüm kullanıcılara(belirli bir OU) bunu push etmesi.

**Hedefler:**

* Kullanıcıların Outlook e-posta imzalarını merkezi bir platformdan yönetebilmek.
* Sistem yöneticisinin imza güncelleme ve dağıtım süreçlerini otomatikleştirmek.
* Kurum genelinde tutarlı ve profesyonel bir e-posta imajı sağlamak.
* Active Directory üzerinden dinamik kullanıcı bilgilerini kullanarak imza oluşturmak.
* Farklı departmanlar veya gruplar için özelleştirilmiş imza şablonları sunmak.
* Kullanıcıların mevcut imzalarını yedekleme ve gerektiğinde geri yükleme imkanı sunmak (isteğe bağlı).
* Kullanıcıların kendi imzalarını değiştirmesini engelleme seçeneği sunmak (isteğe bağlı).
* Sistem yöneticisi için kullanıcıları ve imza süreçlerini kolayca yönetebileceği bir web tabanlı veya masaüstü yönetim panosu sunmak.
* Yönetim panosunda genel sistem durumu, lisans bilgileri ve işlem loglarını görüntülemek.
* Kullanıcıları AD'den belirli kriterlere (OU) göre filtreleyebilmek ve yönetebilmek.
* Uygulamanın kullanımını yıllık lisans ücretine bağlamak ve lisanslama sürecini uygulama üzerinden yönetmek.
* Lisans süresi dolan veya yaklaşan kullanıcıları uyarmak.
* İmza uygulama ve lisanslama süreçleri hakkında detaylı raporlar sunmak.
* Uygulamanın kararlı, güvenli ve kullanıcı dostu olmasını sağlamak.

**İzlenecek Adımlar (Fazlara Göre Detaylı Açıklamalar):**

**Faz 1: Temel Geliştirme, Mock Veri ve Basit Lisanslama Altyapısı**

1. **Planlama ve Gereksinim Analizi:**
   * Sistem yöneticileri ve ilgili paydaşlarla detaylı görüşmeler yaparak tüm gereksinimleri (imzada bulunacak zorunlu/isteğe bağlı alanlar, farklı departmanların imza gereksinimleri, güvenlik beklentileri vb.) belirlemek.
   * Projenin kapsamını (zorunlu ve isteğe bağlı özellikler) netleştirmek.
   * Kullanılacak temel teknolojileri (Python sürümü, GUI kütüphanesi, şablon motoru, veritabanı) ve geliştirme ortamını (IDE, versiyon kontrol sistemi) belirlemek.
   * Uygulama için bir yıllık lisans ücreti ve basit bir lisans anahtarı formatı (örneğin, metinsel bir kod) belirlemek.
   * Proje için gerçekçi bir zaman çizelgesi ve tahmini bütçe oluşturmak.
2. **Teknolojik Altyapı Kurulumu (Python & Mock):**
   * En güncel kararlı Python 3.x sürümünü kurmak.
   * Gerekli Python kütüphanelerini pip kullanarak yüklemek: Jinja2 (şablonlama), Tkinter veya PyQt/Kivy (GUI), configparser (yapılandırma), json veya csv (mock veri yönetimi), sqlite3 (basit lisans veri saklama).
   * Git gibi bir versiyon kontrol sistemini kurmak ve proje için bir Git deposu başlatmak.
3. **Mock Veri Yönetimi:**
   * Gerçek Active Directory yapısını taklit eden örnek kullanıcı bilgilerini içeren JSON veya CSV formatında mock veri dosyaları oluşturmak (her kullanıcı için ad, soyad, unvan, departman, e-posta adresi gibi temel alanlar).
   * Bu mock veri dosyalarını okuyabilen, filtreleyebilen ve işleyebilen Python fonksiyonlarını geliştirmek.
4. **Kullanıcı Arayüzü Geliştirme (Temel):**
   * Mock veriden okunan kullanıcıları listeleyebilen basit bir grafiksel kullanıcı arayüzü (GUI) oluşturmak (Tkinter, PyQt veya Kivy kullanarak).
   * Kullanıcı listesi üzerinde (mock düzeyde) temel filtreleme (örneğin, departmana göre) işlevselliği eklemek.
5. **İmza Şablonu Yönetimi:**
   * Kullanıcının (sistem yöneticisi) kolayca metin tabanlı veya basit bir WYSIWYG (Ne Görürsen Onu Alırsın) düzenleyici kullanarak imza şablonları oluşturabileceği bir arayüz geliştirmek.
   * Jinja2 şablon motorunu entegre ederek {ad}, {soyad}, {unvan}, {departman} gibi özel işaretleyicilerle kullanıcı bilgilerinin dinamik olarak şablona yerleştirilmesini sağlamak.
   * Oluşturulan şablonları kaydetme, yükleme ve düzenleme işlevleri eklemek.
6. **İmza Uygulama Simülasyonu:**
   * Seçilen kullanıcılar ve bir imza şablonu temelinde örnek imzalar oluşturacak bir fonksiyon geliştirmek.
   * Bu imzaların (mock düzeyde) ekranda önizlenmesini veya basit bir çıktı mekanizması ile gösterilmesini sağlamak. Gerçek bir "push" işlemi bu fazda simüle edilecektir.
7. **Basit Lisanslama Altyapısı:**
   * Uygulama için basit bir lisans anahtarı formatı (örneğin, "YYYY-AAAA-BBBB-CCCC") belirlemek.
   * Yönetim panosunda bir lisans anahtarı giriş alanı ve "Aktive Et" butonu oluşturmak.
   * Girilen lisans anahtarının belirlenen formata uygun olup olmadığını kontrol eden temel bir doğrulama fonksiyonu yazmak.
   * Lisans durumunu (aktif/süresi doldu) ve bitiş tarihini (aktivasyon tarihinden itibaren 1 yıl sonrası olarak basitçe hesaplanabilir) saklamak için basit bir dosya (örneğin, license.json) veya sqlite3 veritabanı kullanmak.
   * Uygulama her açıldığında veya yönetim panosu açıldığında lisans dosyasını/veritabanını okuyarak geçerlilik durumunu kontrol eden temel bir fonksiyon yazmak.
8. **Yönetim Panosu Temel Tasarımı:**
   * Kullanıcı yönetimi, imza şablonları yönetimi ve lisans yönetimi bölümlerini içeren temel bir yönetim panosu arayüzü oluşturmak.
   * Genel bakış bölümünde mock kullanıcı sayısı, uygulanan imza sayısı (simüle edilmiş) ve lisans durumunu (aktif/süresi doldu/yaklaşıyor) görüntülemek.
   * Lisans yönetimi bölümünde lisans anahtarı giriş alanı, aktivasyon butonu ve lisans bitiş tarihini göstermek.
   * Lisans süresi dolmaya yaklaştığında (örneğin, son 30 gün) panoda basit bir uyarı mesajı göstermek.
9. **Loglama:**
   * Uygulamanın temel olaylarını (kullanıcı listeleme, imza oluşturma, lisans kontrolü, aktivasyon girişimi vb.) zaman damgası ve kısa bir açıklama ile bir log dosyasına (.log uzantılı) kaydedecek temel bir loglama mekanizması kurmak (logging modülünü kullanarak).
10. **Temel Testler:**
    * Geliştirilen temel işlevselliği mock veriler ve basit lisanslama ile farklı senaryolarda (farklı sayıda kullanıcı, farklı şablonlar, geçerli/geçersiz lisans girişi vb.) test etmek. Kullanıcı arayüzünün temel işlevselliğini ve akışını kontrol etmek.

**Faz 2: Active Directory Entegrasyonu ve Gelişmiş Lisanslama**

1. **Active Directory Entegrasyonu:**
   * ldap3 kütüphanesini kullanarak şirket Active Directory sunucusuna güvenli bir şekilde bağlantı kurmak için gerekli yapılandırma ayarlarını (sunucu adresi, port, kullanıcı adı, parola) uygulama ayarlarında saklamak.
   * Belirtilen Organizasyonel Birim (OU) altındaki kullanıcıları filtreleyerek ad, soyad, unvan, departman, e-posta adresi gibi temel bilgileri çekebilen fonksiyonlar geliştirmek.
   * Kullanıcı listeleme ve filtreleme işlevselliğini gerçek AD verilerine göre uyarlamak.
2. **Gelişmiş Lisanslama Mekanizması:**
   * Lisans anahtarını ve bitiş tarihini uygulamanın kolayca kurcalanamayacağı bir şekilde (örneğin, basit bir şifreleme yöntemiyle) saklamak.
   * Kullanıcının girdiği lisans anahtarını okuyup, geçerlilik formatını kontrol ettikten sonra, geçerli bir bitiş tarihi belirleyip güvenli bir şekilde saklamak.
   * Uygulama her açıldığında veya belirli aralıklarla lisans bilgilerini okuyarak geçerlilik durumunu kontrol eden daha sağlam bir mekanizma uygulamak.
   * Lisans süresi dolmuşsa, imza uygulama, kullanıcı yönetimi gibi temel uygulama işlevlerini engelleyecek mekanizmalar eklemek.
   * Lisans süresi dolmaya yaklaştığında yönetim panosunda belirgin uyarılar göstermek ve (isteğe bağlı olarak) sistem yöneticisine e-posta bildirimleri gönderme işlevselliği eklemek (SMTP ayarları gerektirir).
3. **Yönetim Panosu Geliştirme:**
   * Gerçek AD verilerini kullanarak kullanıcı listesini ve ilgili bilgileri panoda dinamik olarak göstermek.
   * Lisans yönetimi bölümünde lisans bilgilerini (anahtar, bitiş tarihi, durum) daha detaylı bir şekilde görüntülemek ve (gerekirse dikkatli kullanılması gereken) manuel güncelleme seçenekleri sunmak.
   * Kullanıcıları AD'deki departman veya diğer özelliklere göre filtreleme seçeneklerini geliştirmek.
   * Uygulama ayarlarının (AD bağlantı bilgileri, lisans ayarları vb.) yapılandırılabileceği bir bölüm eklemek.
4. **Gerçek İmza Uygulama Mekanizması:**
   * Seçilen kullanıcılara veya gruplara oluşturulan imzaların uygulanmasını sağlayacak fonksiyonlar geliştirmek. Bu, kullanıcının Outlook imza klasörlerine (genellikle %appdata%\Microsoft\Signatures) programatik olarak HTML ve RTF formatında imza dosyaları oluşturup kopyalamayı içerebilir. Kullanıcı bilgisayarlarına ağ üzerinden erişim veya uzaktan komut çalıştırma yetkileri gerekebilir.
5. **Gelişmiş Raporlama:**
   * İmza uygulama tarihleri, uygulanan kullanıcılar, lisans durumları gibi kriterlere göre raporlar oluşturma ve görüntüleme işlevleri eklemek.
   * Oluşturulan raporları tablo veya grafik formatında görüntüleme seçenekleri sunmak.
   * Raporları CSV veya Excel gibi formatlarda dışa aktarma işlevselliği eklemek.
6. **Gelişmiş Log Yönetimi:**
   * Log kayıtlarını zaman aralığına, seviyeye (bilgi, uyarı, hata) veya belirli olaylara göre filtreleme ve sıralama seçenekleri eklemek.
   * Log kayıtlarını bir dosyaya dışa aktarma işlevselliği sunmak.
7. **Kapsamlı Testler (AD ve Lisanslama ile):**
   * Uygulamanın gerçek Active Directory ortamında doğru çalıştığını, kullanıcı bilgilerini doğru çektiğini ve filtrelediğini test etmek.
   * Gelişmiş lisanslama mekanizmasının tüm senaryolarda (geçerli/geçersiz anahtar, süre dolumu, uyarılar, engelleme) doğru çalıştığını titizlikle test etmek.
   * İmza uygulama mekanizmasının farklı Outlook sürümlerinde ve farklı kullanıcı yapılandırmalarında doğru çalıştığını doğrulamak.
   * Yönetim panosunun tüm işlevlerinin beklendiği gibi çalıştığını test etmek.
8. **Dağıtım ve Belgeleme (Son Sürüm):**
   * Uygulamanın son sürümünü sistem yöneticisinin kolayca kurup çalıştırabileceği bir şekilde paketlemek (örneğin, bir kurulum dosyası oluşturmak).
   * Uygulamanın kurulumu, yapılandırılması (AD bağlantısı, lisanslama), kullanımı, yönetim panosu özellikleri ve olası sorun giderme adımlarını içeren detaylı bir kullanıcı kılavuzu hazırlamak.
9. **Bakım ve Destek Planlaması:**
   * Uygulamanın gelecekteki bakımı, olası hataların giderilmesi, güncellemeler ve kullanıcı desteği için bir plan oluşturmak. Kullanıcı geri bildirimlerini toplama ve değerlendirme süreçlerini belirlemek.

Bu bilgiler ışığında readme.md ve todo.md sayfamızı oluşturalım. Her aşamada todo sayfamızı kontrol ederek devam edelim. Tamamlanan aşamalara tik atalım.

Github da python-kurumsal-eposta adında reponun içine majör değişikliklerden sonra mutlaka push et.

Github bilgilerim:

Username:mbirinci9321

Email:murat@muratbirinci.com.tr