“Zenginleştirilmiş Bulut/SaaS Yönetimi için Prototip Entegrasyon ve Analitik Aracı” projesini eksiksiz, tam ve adım adım olacak şekilde detaylı bir liste halinde bulacaksınız. Bu liste, sıfırdan başlayarak proje fikrinin olgunlaşması, teknik temellerin atılması, geliştirme, test, dokümantasyon, demo ve sunum hazırlıklarına kadar tüm aşamaları kapsamaktadır.

**Hazırlık ve Planlama Aşamaları**

1. **Proje Hedefinin Netleştirilmesi**
   * (1.1) Projenin amacını açıkça tanımla: Bulut/SaaS lisans, kullanım, maliyet analizleri ve optimizasyon önerileri sunan bir prototip.
   * (1.2) Kapsam belirle: İlk aşamada tek bir SaaS sağlayıcısı (örn. Microsoft 365) ile entegrasyon yap. İleride başka SaaS platformları eklenebilir.
   * (1.3) Başarı kriterlerini belirle: Prototipin göstereceği metrikler (kullanılmayan lisans yüzdesi, potansiyel maliyet tasarrufu, kullanım trendleri) ve hedef çıktılar (raporlar, grafikler, optimizasyon önerileri).
2. **Teknik Gereksinimlerin ve Kaynakların Belirlenmesi**
   * (2.1) Yazılım dili ve çerçeveleri seç: Backend için Python (FastAPI) + veri işleme (Pandas), Frontend için React, veritabanı için PostgreSQL.
   * (2.2) Donanım/Geliştirme ortamı: Kendi bilgisayarında Docker kullanarak yerel geliştirme ortamı oluştur.
   * (2.3) API bilgisi toplama: Seçilen SaaS sağlayıcısının (Microsoft Graph API) dokümantasyonunu indir, OAuth2 kimlik doğrulama detaylarını incele.
3. **Mimari ve Veri Modeli Tasarımı**
   * (3.1) Yüksek seviye mimari diyagramı oluştur.
   * (3.2) Veritabanı şeması tasarla: raw\_data tablosu (ham JSON), normalized\_data tablosu (kullanıcı, lisans, tarih, maliyet alanları) ve olası ek tablolar (ör. departments, license\_types).
   * (3.3) Analitik katmanda kullanacağın veri dönüşümleri, zaman serisi analizi, filtreleme, grouping kriterlerini belirle.

**Geliştirme Aşamaları**

1. **Geliştirme Ortamının Kurulumu**
   * (4.1) Gerekli araçları yükle: Python 3.x, Node.js, npm/yarn, PostgreSQL, Docker.
   * (4.2) Git repo oluştur ve temel proje yapısını (backend, frontend, scripts, docs klasörleri) hazırla.
2. **API Entegrasyon Katmanının Geliştirilmesi (Backend - Ingestion Layer)**
   * (5.1) SaaS API için kimlik doğrulama (OAuth2 token alımı) kodla.
   * (5.2) Bir cron job veya scheduled task mantığı ekle: Örneğin her gün saat 00:00’da API’dan veri çek.
   * (5.3) Veri çekme fonksiyonunu yaz: API endpoint’lerine istek atıp kullanıcı, lisans, maliyet verilerini JSON olarak al.
   * (5.4) Gelen veriyi raw\_data tablosuna kaydeden bir fonksiyon ekle.
3. **Veri Normalizasyon ve Zenginleştirme Katmanı**
   * (6.1) Ham veriyi raw\_data’dan okuyup, kullanıcı kimlikleri, lisans tipleri, tarih metrikleri gibi bilgileri temizle ve standardize et.
   * (6.2) Kullanılmayan lisansları belirlemek için iş kuralları: Örn. 30 gündür aktif olmayan kullanıcıyı inaktif say.
   * (6.3) Normalized\_data tablosuna işlenen veriyi yaz. Tarihe, kullanıcıya, lisans tipine göre indeksleri ayarla.
4. **Analitik Fonksiyonların Geliştirilmesi**
   * (7.1) Trend analizi: Son X gün/ay kullanım grafiği oluşturmak için verileri gruplandır ve zaman serisi analizi yap.
   * (7.2) Lisans optimizasyon önerileri: Kullanım sıfır olan veya düşük olan lisansları tespit ederek “Daha düşük maliyetli plana geç” önerisi üret.
   * (7.3) Maliyet projeksiyonu: Mevcut trendlere bakarak gelecek ayın tahmini maliyetini hesapla.
   * (7.4) Gerekirse basit bir ML modeli ekle: K-means ile benzer kullanım profiline sahip kullanıcıları gruplandır.
5. **Web Dashboard (Front-end) Geliştirme**
   * (8.1) React uygulaması oluştur.
   * (8.2) Ana sayfa: Özet KPI’lar (toplam lisans, kullanılmayan lisans yüzdesi, potansiyel tasarruf).
   * (8.3) Grafik sayfası: Zaman serisi grafikler (line chart), maliyet dağılımı (bar chart), lisans tipine göre kullanım (pie chart).
   * (8.4) Filtreler: Departman veya lisans tipine göre sonuçları filtrele.
   * (8.5) Öneri sekmesi: Tespit edilen optimizasyon fırsatlarını listele.
6. **API ve Front-end Entegrasyonu**
   * (9.1) Backend tarafında analitik sonuçları JSON olarak sunan endpoint’ler yaz.
   * (9.2) Frontend tarafında Axios veya Fetch API ile backend’e istek atarak verileri çek.
   * (9.3) Hata yönetimi, loading spinner, responsif tasarım gibi kullanıcı deneyimi iyileştirmelerini yap.

**Test, Güvenlik, Performans ve Kalite Aşamaları**

1. **Test Süreci**
   * (10.1) Birim testleri: Veri çekme fonksiyonları, normalizasyon fonksiyonları, analitik hesaplama fonksiyonları için yaz.
   * (10.2) Entegrasyon testleri: API’dan çekilen verinin veri tabanına sorunsuz kaydedildiğini, frontend’in doğru veriyi gösterdiğini test et.
   * (10.3) Performans testleri: Büyük bir sahte veri setiyle sorgu sürelerini ölç, indekslerin doğru çalıştığından emin ol.
2. **Güvenlik ve Erişim Kontrolleri**
   * (11.1) SaaS API erişimi için gerekli tokenların güvenli saklanması.
   * (11.2) Frontend’de rol tabanlı erişim: Belki yöneticiler maliyet önerilerini görebilirken, normal kullanıcılar sadece kendi departmanının kullanımını görebilir.
   * (11.3) Tüm API isteklerinde HTTPS zorunluluğu ve veritabanı bağlantısında SSL kullanımı.
3. **Dokümantasyon ve Sürüm Yönetimi**
   * (12.1) README dosyasında kurulum, çalıştırma, bağımlılıklar, örnek konfigurasyon dosyaları gibi bilgileri yaz.
   * (12.2) OpenAPI/Swagger dökümanları ile backend endpoint’lerini otomatik belgelemeye çalış.
   * (12.3) Mimari diyagramlar, veri modeli şeması, örnek sorgular ve ekran görüntüleri içeren bir teknik doküman hazırla.
4. **CI/CD ve DevOps Uygulamaları**
   * (13.1) Git repo ve branch stratejisi oluştur.
   * (13.2) Basit bir CI pipeline ile kod push edildiğinde testleri otomatik çalıştır.
   * (13.3) Dockerfile hazırlayarak hem backend hem frontend’i konteynerize et.
   * (13.4) İster lokal, ister bir bulut VM üzerinde kolay dağıtım için docker-compose veya Kubernetes manifestleri ekle.

**Demo, Sunum ve Geliştirme Sonrası Aşamalar**

1. **Demo Verisi ve Senaryoları Hazırlama**
   * (14.1) Küçük bir demo veri seti oluştur: 50 kullanıcı, 3 lisans planı, 90 günlük kullanım verisi.
   * (14.2) Bazı kullanıcıların inaktif kaldığı, bazılarının pahalı lisansa sahip olduğu durumlar oluştur, böylece demo sırasında optimizasyon önerilerini gösterebil.
2. **Sunum Hazırlığı**
   * (15.1) PowerPoint veya benzeri bir sunumla projeyi özetle: Sorun, çözüm, mimari, ekran görüntüleri, sağlanan faydalar.
   * (15.2) Canlı demo: Terminalde veri çekme sürecini, sonrasında dashboard’da kullanım trendlerini göster.
   * (15.3) Sorulara karşı hazırlıklı ol: Metriklerin nasıl hesaplandığı, başka SaaS platformlarına nasıl ekleneceği, güvenlik önlemleri gibi konulara cevap hazırla.
3. **Gelecek Geliştirmeler ve Yol Haritası Belirleme**
   * (16.1) Daha fazla SaaS entegrasyonu: Salesforce, Atlassian ürünleri vb.
   * (16.2) Makine öğrenimi tabanlı tahmin ve segmentasyon modelleri ekleme.
   * (16.3) Lisans değişikliklerini otomatik tetikleyen iş akışları ekleme (proaktif aksiyonlar).

**Kontrol Listesi ve Son Rötuşlar**

1. **Kontrol Listesi**
   * (17.1) Kod kalitesini artır: Linter, formatter kullan.
   * (17.2) Gereksiz bağımlılıkları, veritabanı tablolarını, endpoint’leri gözden geçir.
   * (17.3) Ortam değişkenleri (.env dosyası) ile konfigürasyon ver. Demo sırasında tüm ayarlar net olsun.
2. **Final Test ve Gösterim Öncesi Hazırlık**
   * (18.1) Tüm testleri son kez çalıştır.
   * (18.2) Demo veri setiyle canlı deneme yap: Göstereceğin sayfalar hızlıca yüklensin mi, grafikler doğru mu gösteriyor, raporlar anlamlı mı?
   * (18.3) Sunum sırasında internet bağlantısı veya API erişim sorunlarına karşı bir “offline demo” seçeneği (önceden kaydedilmiş ekran görüntüleri) hazırla.