DOKÜMAN 1 – GENEL TEKNİK GEREKSİNİMLER

1. Amaç

Bu doküman, üniversite öğrencilerinin geliştireceği:

1. Finans Portalı Uygulaması
2. IT Service Management (Ticket) Uygulaması

için ortak teknik altyapı ve mimari gereksinimleri tarif eder. Tüm ekiplerin benzer bir “full-stack” mimariyi kullanması beklenmektedir.

---

2. Genel Mimari

- Uygulamalar **container tabanlı mimari** üzerinde çalışmalıdır.
- Tercihen:
    - Docker imajları oluşturulmalı
    - Docker Compose veya Kubernetes gibi bir orkestrasyon aracı ile yönetilebilir olmalıdır (en azından Compose seviyesinde).
- Uygulama bileşenleri (backend, frontend, veritabanı, log/monitoring bileşenleri) birbirinden ayrık container’lar olarak tasarlanmalıdır.

---

3. Teknoloji Yığını (Tech Stack)

3.1 Backend

- Dil: **Java**
- Framework: **Spring Boot**
- Temel beklentiler:
    - RESTful API’ler
    - Katmanlı mimari (Controller, Service, Repository vb.)
    - Configuration yönetimi (application.yml / profiles)
    - Exception handling ve standart API response formatı

3.2 Web Frontend

- Framework: **ReactJS**
- Beklentiler:
    - Component tabanlı yapı

- Routing (React Router vb.)
- State management (Context API veya basit state yönetimi; Redux zorunlu değil ama tercih edilebilir)
- API çağrılarının backend'e REST üzerinden yapılması

3.3 Mobil Uygulama

- Framework: **React Native**
- Beklentiler:
  - Temel ekranlar web ile fonksiyonel olarak uyumlu olmalı (login, listeleme, detay vb.)
  - Backend ile REST API üzerinden haberleşme

---

4. Veritabanı

- Veritabanı: **PostgreSQL**
- Beklentiler:
  - Tabloların ve ilişkilerin normalleştirilmiş bir şekilde tasarlanması
  - ORM kullanımı (Spring Data JPA / Hibernate)
  - Migration aracı kullanılması tercih edilir (Flyway, Liquibase vb.)

---

5. Loglama ve İzleme

5.1 Loglama

- Log kütüphanesi: **log4j** (tercihen log4j2)
- Backend uygulamasında:
  - Request/response, hata ve iş akışı logları tutulmalı
  - Log seviyeleri (INFO, WARN, ERROR, DEBUG) anlamlı kullanılmalı

5.2 Logların OpenSearch'e Aktarılması

- Loglar **OpenSearch**'e akıtılmalıdır.
- Logların iletimi:
  - Doğrudan veya
  - Arada **Kafka** kullanılarak (log4j → Kafka → Log tüketici servisi → OpenSearch) yapılabilir.
- Log formatı:

- Mümkünse JSON formatında standardize edilmelidir (timestamp, level, serviceName, message, exception vb.)

---

6. Gözlemlenebilirlik ve Performans (OpenTelemetry)

- Uygulama, **OpenTelemetry** ile izlenebilir hale getirilmelidir.
- Beklentiler:
    - Temel metriklerin toplanması (istek sayısı, response süreleri, hata oranları vb.)
    - Trace / span bilgisi ile isteklerin uçtan uca izlenebilmesi
    - Bu verilerin **OpenSearch** veya uyumlu bir arayüze aktarılması
    - Örnek dashboard'ların hazırlanması
        - Örnek: "API Response Time", "Error Rate", "Request Volume", "Service Health"

---

7. Güvenlik ve Kimlik Yönetimi

- Kimlik yönetimi için: **Keycloak** kullanılmalıdır.
- Keycloak arkasında:
    - **OpenLDAP** veya **Apache Directory Server** benzeri bir directory server kullanılabilir.
- Beklenen özellikler:
    - Kullanıcı yönetimi ve rol bazlı yetkilendirme
    - **2-Factor / 2-Phase Authentication** desteği
        - Örneğin: Authenticator uygulaması (Google Authenticator vb.) ile OTP
    - **"Beni hatırla"** (remember me) özelliği
    - Hem web frontend hem mobil uygulamada SSO / token tabanlı oturum yönetimi (JWT veya OAuth2/OpenID Connect akışı)

---

8. İş Akışı Yönetimi (jBPM)

- Özellikle **Ticket / Servis Yönetimi Uygulaması** için arka planda iş akışları **jBPM** ile yönetilebilir:
    - Ticket yaşam döngüsü
    - SLA süreçleri
    - Onay / eskalasyon süreçleri
- jBPM kullanımı minimum:

- Basit bir “process definition” hazırlanması
- Ticket statü geçişlerinin bu süreç üzerinden yönetilmesi

---

9. Ortak Fonksiyonel Olmayan İsterler

- **Performans**: Normal yük altında cevap süreleri makul olmalı (örn. < 2 sn hedeflenebilir).
- **Dokümantasyon**:
  - Kısa bir mimari doküman
  - API endpoint listesi (endpoint, metot, parametreler, response örneği)
- **Test**:
  - Temel unit test / integration test örnekleri
- **DevOps (Tercihen)**:
  - Basit bir CI/CD pipeline tasarımı (örnek doküman olarak bile olabilir)

DOKÜMAN 2 – UYGULAMA ÖZEL İSTERLERİ

2.1 Finans Portalı Uygulaması

2.1.1 Amaç

Kullanıcılara finansal piyasa verilerini, haberleri ve kendi portföylerini yönetebilecekleri, temel analizler yapabilecekleri bir web ve mobil finans portalı sunmak.

2.1.2 Temel Fonksiyonel Modüller

1. **Haber Modülü**
   - Açık kaynaklardan:
     - Güncel haberler
     - Özellikle finans ile ilgili haberler çekilmeli
   - Haberler kategorilere ayrılabilmeli (genel ekonomi, hisse, döviz, tahvil vb.)
   - Haber detay sayfası olmalı (başlık, tarih, kaynak, içerik özeti)
2. **Piyasa Verileri Modülü**
   - **Kur Bilgileri:**
     - TCMB kurları
     - Çeşitli bankaların kurları
     - Diğer açık kaynak kurlar

- **Hisse Senedi Bilgileri**
- **Tahvil / Bono Bilgileri**
- **VIOP Enstrümanları**
- **Fon Bilgileri** (yatırım fonları vb.)
- Veriler en azından “okuma” amaçlı güncel olarak görüntülenebilmelidir.

3. **Tarihsel Veri ve Analiz**
    - Kullanıcılar:
        - Belirli bir enstrüman için tarih aralığı seçebilmelidir (örn. son 1 ay, 3 ay, 1 yıl).
        - Tarihsel fiyat verilerini grafik olarak görebilmelidir.
    - Basit **karşılaştırma**:
        - Aynı grafikte birden fazla enstrümanın performansı gösterilebilmelidir (örn. USD/TRY vs EUR/TRY, veya belirli iki hisse senedi).
    - **Temel teknik analiz** (basit seviyede yeterli):
        - Hareketli ortalama (MA)
        - Basit trend analizi (örn. yükselen/düşen trendin görsel gösterimi)
        - Yeterlilik kriteri: Öğrencilerin uygun gördüğü basit indikatörler
4. **Portföy Yönetimi**
    - Kullanıcılar kendi portföylerini oluşturabilmelidir:
        - Enstrüman seçimi (hisse, döviz, fon vb.)
        - Miktar / alış fiyatı bilgisi girişi
    - Portföy ekranında:
        - Güncel değer
        - Kar/Zarar (TL ve/veya yüzde)
    - Basit portföy analizi:
        - Enstrüman bazlı dağılım (örneğin grafik / pie chart)
        - Toplam getiri
5. **Kullanıcı Yönetimi**
    - Login / logout Keycloak üzerinden
    - Profil bilgisi görüntüleme
    - Basit rol modeli (normal kullanıcı, admin gibi)

2.1.3 Teknik İsterler (Finans Portalına Özel)

- Veri kaynakları yapılandırılabilir olmalıdır (örneğin config'den API URL'leri).
- Veri çekme işleri:
    - Zamanlanmış görevler (scheduler) ile periyodik olarak cache'lenebilir.
- Performans için:
    - Sık kullanılan veriler için basit cache mekanizması (in-memory veya Redis vb. – zorunlu değil ama artı puan).

2.2 IT Service Management (Ticket) Uygulaması

2.2.1 Amaç

Müşterilerin teknik sorunlarını bildirebildiği, durumunu takip edebildiği ve destek ekibinin bu talepleri SLA kurallarıyla yönettiği bir **ticket yönetim sistemi** geliştirmek.

2.2.2 Temel Fonksiyonel Modüller

1. **Müşteri Portalı**
    - Müşteriler:
        - Portal üzerinden login olur.
        - Yeni ticket açabilir.
        - Mevcut ticket'larının listesini görebilir.
        - Ticket detayını ve durumunu takip edebilir.
    - Ticket açarken:
        - Ürün seçimi (dropdown vb.)
        - Özet ve detay açıklama
        - Öncelik (low/medium/high) tercihen
2. **Log Yükleme ve Validasyon**
    - Müşteri ticket açarken:
        - Log dosyası veya ekran görüntüsü gibi ek dosya yükleyebilmelidir.
    - Log validasyonu:
        - Basit kurallar yeterlidir (örn. dosya tipi kontrolü, boyut sınırı, text dosyası ise içinde belirli anahtar kelimeler aranması vb.)
3. **Ticket Yaşam Döngüsü Yönetimi**
    - Ticket statüleri (örnek):

        - New → In Progress → Waiting for Customer → Resolved → Closed
    - Support ekibi:
        - Ticket atama (assignee)
        - Statü değiştirme
        - Çözüm notu ekleme
    - **Worklog**:
        - Destek ekibi ticket üzerinde harcadığı süreyi kaydedebilmelidir (tarih, süre, açıklama).
    - **SLA Yönetimi**:
        - Ticket tipine/önceliğine göre hedef çözüm süreleri tanımlanabilir.
        - SLA'ya yaklaşan veya ihlal eden ticket'ların işaretlenmesi (renk, ikon vb.)
    - **Notification**:
        - Temel bildirim mekanizması:
            - Ticket açıldığında
            - Statü değiştiğinde
            - SLA ihlal riski olduğunda
              e-posta benzeri bildirim simülasyonu (gerçek mail zorunlu değil, UI üstünden "notifications" alanı da olabilir).

4. **İç / Dış İletişim**
    - Ticket detay ekranında:
        - "Internal" (sadece destek ekibinin gördüğü) notlar
        - "External" (müşterinin de gördüğü) yorumlar
    - Bu sayede:
        - Müşteri ile yazışma ticket üzerinde tutulur.
        - Destek ekibi kendi aralarında iç not tutabilir.
5. **Raporlama**
    - Örnek raporlar:
        - Açık ticket sayısı (duruma göre dağılım)
        - Kapalı ticket sayısı (tarih aralığına göre)
        - Kişi / ekip bazlı performans (kaç ticket çözüldü)
        - SLA uyum oranı (zamanında kapanan vs geç kapanan ticket'lar)
    - Raporlar:
        - Basit tablo ve/veya grafikler ile gösterilebilir.

2.2.3 Teknik / İş Akışı İsterleri (jBPM)

- Ticket yaşam döngüsü **jBPM** süreçleri ile yönetilebilir:
    - Her ticket için bir process instance oluşturulması
    - Statü geçişlerinin bu süreç üzerinden ilerlemesi
- SLA kontrolleri:
    - jBPM içinde timer veya benzeri mekanizma kullanılarak takip edilebilir (tasarım seviyesi bile yeterli olabilir).
- Süreç modelinin (BPMN) doküman olarak sunulması artı puan sağlar.

2.2.4 Güvenlik ve Yetkilendirme

- Aynı Keycloak / LDAP altyapısı kullanılır.
- Rol örnekleri:
    - CUSTOMER (müşteri)
    - AGENT (destek personeli)
    - MANAGER (raporları ve SLA’yı gören yönetici)
- Erişim kontrolleri:
    - Müşteri yalnızca kendi ticket’larını görebilir.
    - Destek ekipleri tüm ticket’ları veya atandığı ticket’ları görebilir.
