**Problem Tanımı**

Sender ve Receiver "thread"ler barındıran bir sistemin görselleştirilmesi ve yönetimi yapılacaktır. Kullanıcı frontend'de Sender ve Receiver thread sayılarını seçip, backend'de send ve receive işlemler yapan thread'ler oluşturacaktır. Sender lar ortak bir queue veri yapısına saniyede 1 frekansla ekleme yapacaktır, Receiver larda gelen veriyi aynı frekansta queue’dan tüketecektir. Kullanıcılar frontend üzerinden Sender/Receiver Thread’ler arası veri paylaşımında kullanılan queue’nun anlık güncel durumunu, tüm thread'lerin önceliklerini değiştirebilmeli, anlık durumlarını görebilmeli, seçilen thread'leri durdurup başlatabilmelidir. Backend kısmında, Thread'lerin oluşturulması, durumu, önceliği ve yaşam döngüsü yönetimi yapılmalıdır. Backend Frontend arası iletişim için API oluşturulması beklenmektedir.

**Requirements (Gereksinimler)**

**Frontend (React)**

1. **Thread Sayısını Seçme ve Başlatma**
   * Kullanıcı, frontend üzerinde **thread sayısını** seçebilir (örneğin: 5, 10, 20, vb.). Bu seçim backend'e gönderilir.
   * Backend, bu sayıya göre **send ve receive işlemler yapan** thread'ler başlatır.
2. **Thread Gösterimi ve Durum İzleme**
   * Thread'ler frontend'de gösterilmeli ve her birinin durumu (aktif/pasif vb) anlık gösterilmelidir.
   * Kullanıcı, thread'lerin durumunu **başlatma**, **durdurma** ve **öncelik değiştirme** gibi işlemlerle kontrol edebilmelidir.
3. **Thread’lerin Durumunun Arkayüzde Loglanması**
   * Backend Tarafında Thread lerin tüm durum değişikliklerinin Console da loglanması ve güncel durumlarının listelenmesi sağlanmalıdır.
4. **UI/UX**
   * UI, kullanıcı dostu olmalı ve thread’lerin durumu net bir şekilde kullanıcıya sunulmalıdır.
   * Thread’lerin durumu canlı bir şekilde UI da takip edilebilmelidir.

**Backend (Spring Boot)**

1. **Thread Yönetimi API'si**
   * **Thread oluşturma API’si**: Kullanıcı belirli bir sayıda thread başlatmak istediğinde, backend bu kadar sayıda **send/receive işlemi yapan thread** oluşturmalıdır.
2. **Thread Durumu Güncelleme ve İzleme**
   * Her thread'in durumu backend tarafından izlenmeli ve frontend'e iletilmelidir. Durumlar, başlatılabilir, durdurulabilir ve öncelikleri değiştirilebilir.

**Deliverables (Teslimatlar)**

1. **Frontend Kodu (React)**
   * Kullanıcıların thread sayısını seçebileceği ve bu sayıya göre backend'in thread lerinin başlatılacağı bir UI
   * Thread’lerin durumu ve işlemlerini izlemeyi sağlayan dinamik bir görselleştirme.
2. **Backend Kodu (Spring Boot)**
   * **Thread başlatma** API’si, belirli sayıda thread'in başlatılmasını sağlayacak
   * **Thread durumu** API'si, thread'lerin aktif/pasif durumlarının kontrolü.
3. **Testler**
   * **Backend testleri**: API’lerin doğru çalıştığını test eden birim testleri.
   * **Frontend testleri**: UI’nin doğru şekilde thread’leri ve durumları görüntülediğini test etmek.
4. **Dokümantasyon**
   * Projenin nasıl çalıştığına dair açıklamalar ve geliştirme ortamının kurulumu.
   * API kullanım talimatları, thread’lerin nasıl başlatılacağı, durdurulacağı ve önceliğin nasıl değiştirileceğine dair açıklamalar.

# DEĞERLENDİRME KRİTERLERİ

1. **Dökümantasyon**
2. **Proje Yapısı/Mimarisi**
3. **Kod Kalitesi**
   * Tasarım prensiplerine/kalıplarına uyum
   * En iyi pratiklerin uygulanması
4. **Tasarım ve İş Mantığı**
   * Anlamlı değişken ve method isimleri
   * Mimari tasarım
   * Nesne yönelimli programlama prensiplerine uyum
5. **Test**
   * Birim test
6. **Uygulama Fonksiyonalitesi**
7. **Versiyon Kontrol**
   * Git Kullanımı ve Commit Tarihçesi
8. **Bonus**
   * Advanced Java Concepts
   * Documentation of API Endpoints (Swagger ya da OpenAPI)
9. **Sunum**