**İLERİ PROGRAMLAMA DERSİ**

**WEBRTC UYGULAMASI PROJE RAPORU**

**Öğrenci:** Aydın Şaşik  
**Ders:** İleri Programlama  
**Öğretim Üyesi:** Doç. Dr. Serdar Aydın

**1. PROJE GENEL BAKIŞI**

Bu proje, gerçek zamanlı iletişim teknolojisi olan WebRTC (Web Real-Time Communication) ve WebSocket protokolü kullanılarak geliştirilmiş bir görüntülü arama uygulamasıdır. Uygulama hem masaüstü hem de web tabanlı erişim imkanı sunmaktadır.

**2. PROJE MİMARİSİ VE DOSYA YAPISI**

**2.1 Ana Dizin Yapısı**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**2.2 Teknoloji Yığını**

* **Backend:** Node.js + WebSocket Server
* **Frontend:** HTML5, CSS3, JavaScript + WebSocket Client
* **Gerçek Zamanlı İletişim:** WebRTC + WebSocket
* **Tünel Servisi:** Ngrok
* **IDE Entegrasyonu:** Visual Studio
* **İletişim Protokolleri:** WebRTC, WebSocket, HTTP/HTTPS

**3. ÇALIŞMA PRENSİBİ**

**3.1 Sunucu Başlatma**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

* Node.js sunucusu port 3000'de çalışmaya başlar
* **WebSocket Server** başlatılır ve bağlantı dinlemeye başlar
* WebRTC signaling için WebSocket bağlantıları hazırlanır
* Statik dosyalar (public klasörü) sunulmaya başlar

**3.2 Ngrok Tünel Servisi**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

* Yerel sunucuyu (localhost:3000) internet üzerinden erişilebilir hale getirir
* **WebSocket bağlantılarını** da HTTPS üzerinden güvenli hale getirir
* NAT ve firewall sorunlarını bypass eder

**4. WEBSOCKET ENTEGRASYONU VE ÇALIŞMA MEKANİZMASI**

**4.1 WebSocket Server (server.js)**

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**4.2 WebSocket Client (script.js)**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**4.3 WebSocket + WebRTC İletişim Akışı**

1. WebSocket Bağlantısı: İlk handshake
2. Signaling via WebSocket: Offer/Answer değişimi
3. ICE Candidates: WebSocket üzerinden paylaşım
4. Peer Connection Kurulumu: WebRTC direkt bağlantı
5. Media Stream: Video/ses aktarımı

**5. TEKNİK DETAYLAR**

**5.1 Hibrit İletişim Mimarisi**

**WebSocket Kullanım Alanları:**

* Signaling Server: WebRTC bağlantı kurulumu
* Gerçek Zamanlı Mesajlaşma: Anında bildirimler
* Durum Senkronizasyonu: Kullanıcı durumları
* Bağlantı Yönetimi: Peer discovery ve matching

**WebRTC Kullanım Alanları:**

* Media Streaming: Video/ses aktarımı
* Peer-to-Peer İletişim: Doğrudan bağlantı
* Düşük Gecikme: Minimum latency
* Bandwidth Optimizasyonu: Efficient data transfer

**5.2 Signaling Protocol Akışı**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**6. UYGULAMA ÖZELLİKLERİ**

**6.1 Çoklu Platform Desteği**

1. **Web Tabanlı Erişim:**
   * Ngrok link üzerinden tarayıcı erişimi
   * WebSocket client otomatik bağlantı
2. **Masaüstü Uygulaması:**
   * Visual Studio entegreli Form W WEBRTC
   * Native WebSocket implementasyonu

**6.2 Gerçek Zamanlı Özellikler**

* Instant Connection: WebSocket ile anında bağlantı
* Real-time Signaling: Gerçek zamanlı sinyal değişimi
* Live Status Updates: Kullanıcı durumu güncellemeleri
* Connection Monitoring: Bağlantı durumu izleme

**7. GÜVENLİK VE PERFORMANS**

**7.1 Güvenlik Katmanı**

* WebSocket Secure (WSS): Ngrok üzerinden şifreli bağlantı
* Authentication: Şifre kontrolü (1234)
* Session Management: WebSocket session yönetimi
* Message Validation: Signaling mesaj doğrulama

**7.2 Performans Optimizasyonları**

* Connection Pooling: WebSocket bağlantı havuzu
* Message Queuing: Mesaj kuyruklama sistemi
* Bandwidth Management: Bant genişliği yönetimi
* Error Recovery: Otomatik hata kurtarma

**8. WEBSOCKET vs HTTP KARŞILAŞTIRMASI**

**8.1 WebSocket Avantajları**

* Full-Duplex Communication: Çift yönlü iletişim
* Lower Latency: Düşük gecikme
* Real-time Data: Gerçek zamanlı veri aktarımı
* Persistent Connection: Kalıcı bağlantı

**8.2 Kullanım Senaryoları**

* WebRTC Signaling: Bağlantı kurulum mesajları
* Instant Messaging: Anlık mesajlaşma
* Status Broadcasting: Durum yayını
* Connection Coordination: Bağlantı koordinasyonu

**9. GELIŞTIRME SÜRECİ VE ÖĞRENME ÇIKTILARI**

**9.1 Kullanılan Teknolojiler**

* Node.js + ws: WebSocket server implementasyonu
* JavaScript WebSocket API: Client-side WebSocket
* WebRTC API: Peer-to-peer communication
* Socket.IO (potansiyel): Gelişmiş WebSocket kütüphanesi

**9.2 Kazanılan Beceriler**

* Real-time Programming: Gerçek zamanlı programlama
* Network Programming: Ağ programlama
* Asynchronous Communication: Asenkron iletişim
* Protocol Integration: Protokol entegrasyonu

**Proje İçeriği Ekran Görüntüleri**

**metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**

**emotikon, ekran görüntüsü, gülenyüz, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**

**metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**