# Mesajlaşma Uygulaması Raporu

## 1.1 Giriş

Proje localde 3 kullanıcılı bir konsol mesajlaşma aplikasyonudur. C++ dilinde yazılmıştır.

## 1.2 Özellikler

Toplam 3 kullanıcı desteklenecektir. Gerekli bütün komutlar (işlevli özel mesajlar) başta gözükecektir. Kullanıcılar sunucuya giriş yaptığında bir menü ekranı ile karşılaşacaktır. Bu menüde:

* Mesaj modunu açabilirler.
* Mesaj gönderecekleri kullanıcıları seçebilirler.
* Görülmemiş mesajlarına bakabilirler.
* Bütün kullanıcıları görüntüleyebilirler.
* Kendi mesaj geçmişlerine bakabilirler.
* Sunucudan ayrılıp tekrar bağlanabilirler.

### 1.2.1 Giriş Süreci

Kullanıcılara başta kullanıcı adları sorulacaktır. Ardından girdikleri ismin sunucudaki başka isimler ile aynı olup olmadığını belirten “duplicated” sinyaline göre kullanıcılara isimleri tekrar sorulacak ya da kullanıcılar devam edecektir. Eğer isimleri başka isimler ile çakışıyor ise kullanıcılardan geçerli bir isim girene kadar tekrar kullanıcı adı girmeleri istenecektir. Geçerli bir isim girdikten sonra sunucudan sunucudaki aktif kişi sayısının üçü geçip geçmediğini belirten “available” sinyali gelecektir. Eğer sunucuda 3 aktif kişi yok ise gelen sinyalin ardından kullanıcıların sunucuya girişi başarıyla gerçekleşecektir. Giriş başarılı olduktan sonra sunucudaki ilgili depoda Tablo 1 ClientUser Sınıfı objesi oluşturulacaktır.

Tablo ClientUser Sınıfı

|  |  |  |
| --- | --- | --- |
| ALAN ADI | TÜR | AÇIKLAMA |
| clientSocket | SOCKET | ClientUser objesinin sunucu ile iletişiminde kullandığı soket |
| destinations | vector <int> | ClientUser objesinin mesaj hedeflerinin indexlerini içeren vektör |
| inMessageMode | bool | Mesaj modda olup olmadığını tutan ifade |
| isActive | bool | Aktif olup olmamayı tutan ifade |
| clientName | string | Kullanıcı adı |

### 1.2.2 Mesajlaşma Modu

Kullanıcılar menüde 1. seçeneği seçerse mesaj moduna girer. Kullanıcı mesajlaşma moduna girdiğinde başka kullanıcılara mesaj gönderebilir ve kendine gelen mesajları anlık bir şekilde görebilir. Kullanıcılar mesaj gönderirken önceden belirlediği hedeflere mesaj gönderir, eğer hedef belirlememişlerse mesaj gönderirken uyarı alırlar. Kullanıcıların girdileri satır olarak sunucuya string şeklinde iletilir. Bu girdiler sunucuda Tablo 2 Message Sınıfı objesi oluşturularak depolanır. Eğer hedefteki kullanıcı mesaj modunda değilse bu Message objesi tüm görülmemiş mesajların tutulduğu vektöre de koyulur. Gelen mesajlar ise konsollarında anlık olarak gönderilen saat ve gönderen kişi ile birlikte görülür.

Tablo Message Sınıfı

|  |  |  |
| --- | --- | --- |
| ALAN ADI | TÜR | AÇIKLAMA |
| msg | string | Mesajın kendisi |
| time | string | Mesaj saati |
| sender | string | Gönderen kullanıcı |
| destination | string | Gönderilen kullanıcı |

### 1.2.3 Mesaj Hedefi Seçme

Menüdeki 2. seçenek kullanıcıların mesaj hedefi seçmelerine imkân verir. Kullanıcılara önce bütün kullanıcıları sırayla listeli halde görür. Bu işlemde bütün kullanıcı adları teker teker son olup olmadığını belirten başlıklarla gönderilir ve sonra bu gelen veriler kullanıcı tarafında işlenip görüntülenir. Ardından kullanıcı hangi kullanıcıları belirlemek istediğini sayılarla girer. Eğer geçersiz bir girdi girmişlerse sunucu bunu belirleyip bir mesaj gönderir ve kullanıcıdan geçerli bir veri alınana kadar bu süreç tekrarlanır. Geçerli veri girilince sunucu, ilgili ClientUser objesinin hedefleri tutan vektörünü günceller.

### 1.2.4 Görülmemiş Mesajlara Bakma

Kullanıcı 3.seçenek ile görmediği mesajlara bakar. Sunucu bütün görülmemiş mesajları tarayıp hedef olarak ilgili kullanıcının geçtiği mesajları kullanıcıya iletir. Ve ardından bu mesajı görülmemiş mesajlar deposundan siler. Eğer ilgili kullanıcının hiç görülmemiş mesajı yoksa kullanıcı ona göre bir uyarı mesajı ile karşılaşır.

### 1.2.4.1 Görülmemiş Mesajlara Bakma İşleminin Gerçekleşmesi

Önce sunucu bütün görülmemiş mesajları depodan tarar ve mesajların ilgili kullanıcıya gönderilip gönderilmeyeceğine göre bir sayaç tutar. Ardından bu depoyu tekrar tarar ve önceden saymış olduğu sayaca göre eğer taranan mesaj ilgili kullanıcıya iletilecekse ve iletilecek son mesaj değilse sonuna “1” konur ve mesaj iletilir. Eğer mesaj iletilen son mesaj ise sonuna “0” konur ve iletilir. Bu şifreleme yöntemi kullanıcının aldığı mesajların son mu değil mi olduğunu algılamasına imkân sağlayıp ona mesaj alma döngüsünden çıkmaya yardımcı olacaktır. İletilen her mesajdan sonra sunucu kullanıcıdan bir okundu sinyali alır. Bu geri dönüşün tercih edilme sebebi TCP protokolünün veri almadan art arda veri yollarken paketleri birleştirmesidir. Bunun önüne geçmek için bu geri dönüş yöntemi tercih edilmiştir.

### 1.2.5 Kullanıcıları Görüntüleme

Sunucu her bir kullanıcı ismini sonuna son olup olmamasına göre sonlandırma başlığı ekleyip gönderir. Kullanıcı gelen verileri başlığa göre son olup olmadığını algılayarak görüntüler. Son olduğunu algıladığında döngüden çıkar.

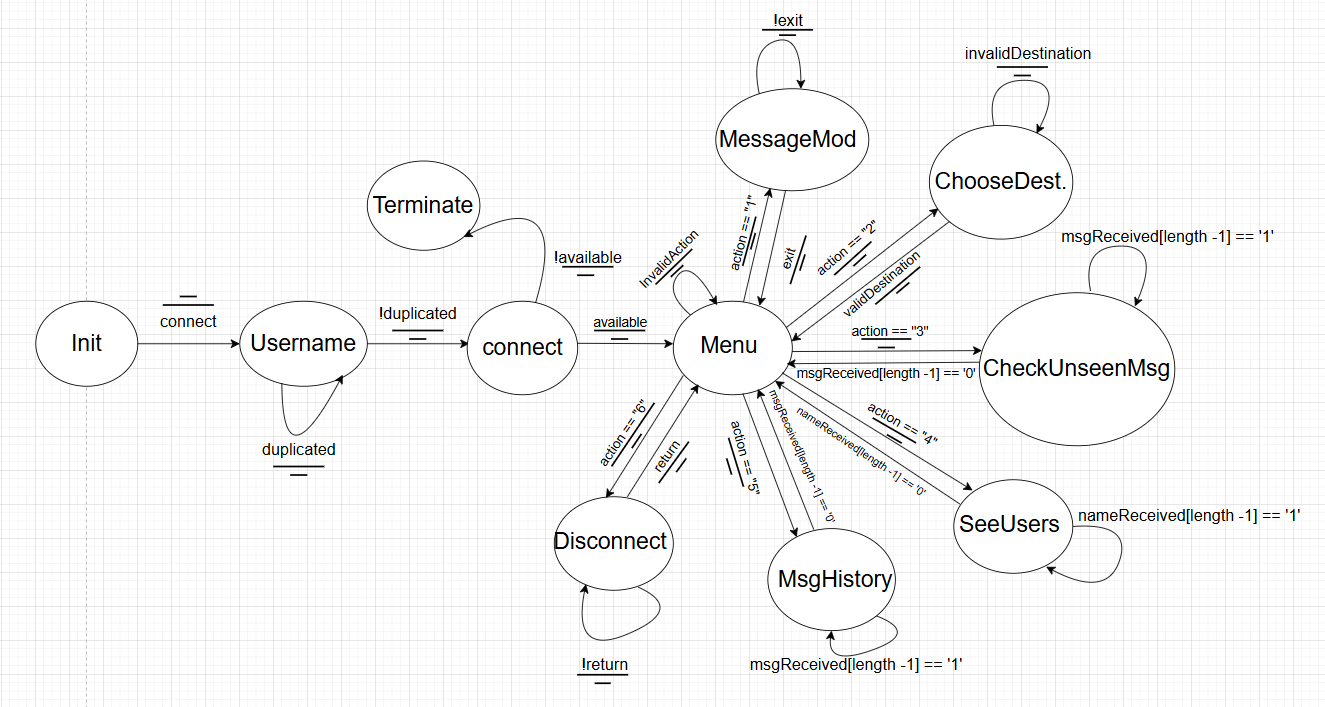
### 1.2.6 Mesaj Geçmişini Görüntüleme

Sunucu ilgili kullanıcının gönderen veya gönderilen olduğu her mesajı saat ve kişi bilgileriyle ilgili kullanıcıya görüntülenmek üzere, sonuna başlık ekleyerek gönderir. Kullanıcı son mesaj gelene kadar bütün mesajları görüntüler.

### 1.2.7 Sunucudan Ayrılma

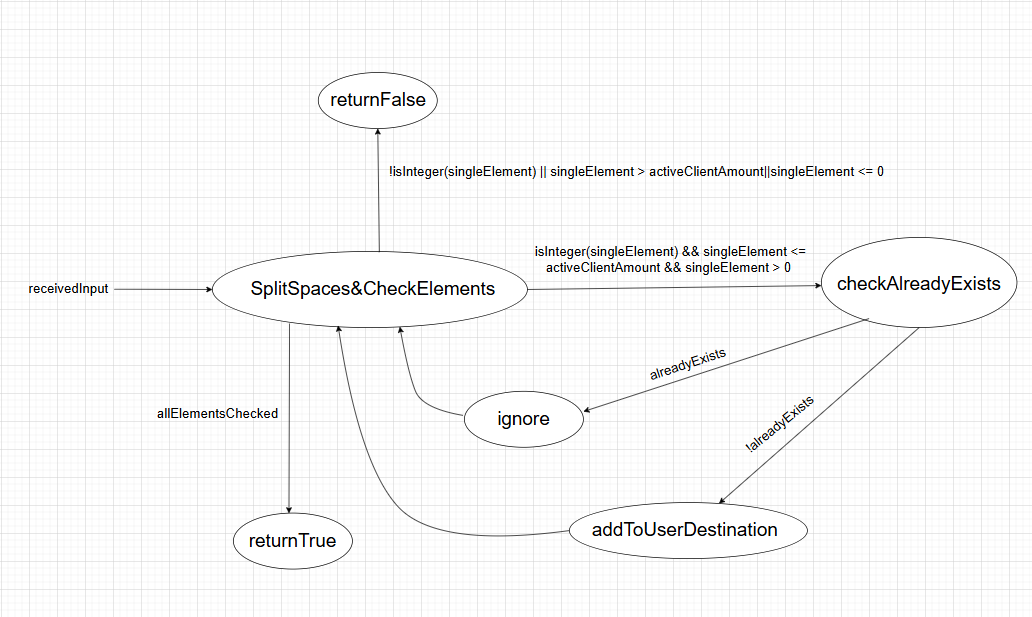
Kullanıcı altıncı seçenek ile sunucudan ayrılabilir. Kullanıcı bir bekleme ekranına girer ve yeniden bağlanma komutunu girene kadar o bekleme ekranında kalır. Bu durumda kullanıcının depodaki ClientUser objesi pasif olarak gözükür.

### 2.1 Kullanıcı FSM



### 2.2 validDestination-invalidDestination Sinyalinin Belirlenmesi

Client mesaj hedefi seçme ekranındayken girdiği girdi sunucuda kontrol edilir ve geçerli olup olmadığına göre sunucu bir kontrol sinyali üretir. Bu sinyal tekrar client’a iletilip onun döngüde kalmasına ya da döngüden çıkmasına yön verir.



### 3.1 Ek Özellikler

-Sonradan GUI yapılması düşünülmektedir.

-Kodun FSM mantığına göre yeniden tasarlanması düşünülmektedir.