NumBot Geliştirici Dokümanı

Hazırlayan: Semih YİLMAZ

Özet

Bu dokumanın amacı NumBot uygulaması ve API'si ile ilgili geliştiricilere uygulamanın teknik yapısı hakkında bilgi vermektir.

NumBot uygulamasının amacı kullanıcının girdi olarak verdiği cümlede bulunan harf ile yazılmış sayıları rakamla yazılmış biçimde kullanıcıya çıkış olarak vermektir. Uygulama kullanıcı deneyiminin artırılması için uygulama ön uçu mesajlaşma uygulamaları temel alınarak tasarlanmıştır. Ayrıca NLP uygulaması için ChatGpt 3.5 Turbo'dan yararlanılmıştır.

Doküman Genel Bakış, Sınıflar, Methodlar ve Katmanlar, API Endpoints ve Postman ile Örnek İstekler, Kullanıcı Arayüzü, Geliştirici Notları başlıklarından oluşmaktadır.

Genel Bakış

Bu başlıkta NumBot uygulaması ve NumBot'un kullandığı API ile ilgili teknik anlamda genel bilgiler verilmektedir.

NumBot uygulaması MVC ile tasarlanmış olup ön uçda Bootstrap kütüphaneleri kullanan arka uçda ise RestFul yapısını kullananan ASP .Net projesidir ve C# programlama dili ile yazılmıştır. Uygulamada tasarlanan API, HTTP protokolünü kullanan ve sadece POST işlemi yapan yapıda tasarlanmıştır. Methodlar json ve parametre ile işlem yapılabilir.

Sınıflar, Methodlar ve Katmanlar

Katmanlar

NumBot Uygulaması tek katman olarak tasarlanmıştır. Bu katman ise MVC yapısında olup Model, View ve Controller alt katmanları bulunmaktadır. Model katmanında 4 adet model sınıfı, Controller katmanında 1 adet controller sınıfı ve View'de ise Home, Particials ve Shared klasörleri bulunmaktadır.

Sınıflar

Model alt katmanında 4 ana sınıf bulunmaktadır. Bu sınıflar API'den gelen isteklere ve geri dönüşlere göre tasarlanmıştır.

- `public class ChatGptGetMessageModel-> ChatGpt 3.5 Turbo API'sinden gelen json yapısındaki veriyi temsil eden sınıftır.`

```
public string id { get; set; }
public string @object { get; set; }
public int created { get; set; }
public string model { get; set; }
public List<Choice> choices { get; set; }
public Usage usage { get; set; }
```

Kod 1 ChatGptGetMessageModel sınıfının içeriği

- `public class Choice`

```
public int index { get; set; }
public Message message { get; set; }
public string finish_reason { get; set; }
```

Kod 2 Choice sınıfının içeriği

- `public class Usage`

```
public int prompt_tokens { get; set; }
public int completion_tokens { get; set;}
public int total_tokens { get; set; }
```

Kod 3 Usage sınıfının içeriği

- `public class ChatGptReqModel->` ChatGpt 3.5 Turbo API'sine gönderilen json yapısındaki veriyi temsil eden sınıftır.

```
public string model { get; set; }
public List<Message> messages { get; set; }

public ChatGptReqModel()
{
     model = "gpt-3.5-turbo";
}
```

Kod 4 ChatGptReqModel sınıfının içeriği

- `public class Message`

```
public string role { get; set; }
public string content { get; set; }
```

Kod 5 Message sınıfının içeriği

- `public class InputTextModel->` NumBot API'sine istek gönderilirken kullanılan json verisini temsil eden sınıftır.

```
public string UserText { get; set; }
```

Kod 6 InputTextModel sınıfının içeriği

- `public class OutputTextModel->` NumBot API'sinden gelen json tipindeki veriyi temsil eden sınıftır.
-

```
public string OutPut { get; set; }
```

Kod 7 OutPutTextModel sınıfının içeriği

Controller alt katmanında 1 adet Controller sınıfından kalıtım alan sınıf bulunmaktadır.

- `public class HomeController->` API methodlarını barındıran Controller türündeki sınıftır.

Uygulamada bu katmanlar dışında ise 1 adet sınıf Bulunmaktadır.

- `public class Keys->` ChatGpt API anahtarını içeren sınıftır. Bu sınıfı oluşturmaktaki amaç geliştirmeyi paylaşırken .gitignore ile API anahtarını saklamaktır.

```
public string apiKey = "sk7pc0…Kw";
```

Kod 8 Keys sınıfının içeriği

Methodlar

ASP .Net ile standart şekilde gelmiş methodlar dışında HomeController sınıfında geliştirme için yapılmış 3 adet method bulunmaktadır. Bunlardan 1 tanesi private, 2 tanesi public olarak ayarlanmış ve sınıf kapsüle edilmiştir.

- `private async Task<string> AskYourQuestionAsync(string InputText) ->` 1 adet parametre alan ve ChatGpt 3.5 API'sine istek gönderen methoddur. InputText adındaki parametre kullanıcıdan gelen cümle girişini temsil eder. Methodun içeriği aşağıdaki şekildedir.

```
try
{
    using (HttpClient httpClient = new HttpClient())
    {
        ChatGptReqModel model = new ChatGptReqModel();
        ChatGptGetMessageModel  messageModel = new ChatGptGetMessageModel();
        keys keys = new keys();
        string apiUrl = "https://api.openai.com/v1/chat/completions";

        string apiKey = keys.apiKey;

        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);

        Message message = new Message();
        message.role = "user";
        message.content = "' "+ InputText + " '" + " cümlesindeki okuduğun tüm sayıları rakam olarak yazarak cümleyi bana söyle. " +
                                                                                    "Sadece cümle ile cevap ver";
        List<Message> messages = new List<Message>();
        messages.Add(message);

        model.messages = messages;
        string requestBody = JsonConvert.SerializeObject(model);

        StringContent content = new StringContent(requestBody, Encoding.UTF8, "application/json");
        HttpResponseMessage response = await httpClient.PostAsync(apiUrl, content);
        response.EnsureSuccessStatusCode();
        string responseData = await response.Content.ReadAsStringAsync();

        messageModel = JsonConvert.DeserializeObject<ChatGptGetMessageModel>(responseData);
        if (messageModel != null)
            return messageModel.choices[0].message.content;
        else
            return "Analiz başarısız! Lütfen yeni bir cümle ile deneyiniz.";
    }
}
catch (HttpRequestException ex)
{
    return $"Hata oluştu: {ex.Message}";

}
```

Kod 9 AskYourQuestionAsync methodunun içeriği

Methodun kaba kodu aşağıdaki şekildedir.

1. HTTPClient oluştur.
2. ChatGptReqModel nesnesini oluştur.
3. ChatGptGetMessageModel nesnesini oluştur.
4. apiURL değişkenine API URL'sini ata.

5. apiKey değişkenine API Key'ini ata.
6. Yetkilendirme için apiKey tipini Bearer olarak ayarla.
7. Message nesnesi oluştur.
8. Message.Role için "User" stringini ata.
9. Message.content için chatGpt'ye sorulacak soru cümlesi ile doldur.
10. API'ye body olarak json göndermek için requestBody değişkenine ChatGptReqModel tipi ile json serialize işlemi yap.

```
{
  "model": "gpt-3.5-turbo",
  "messages": [
    {
      "role": "user",
      "content": "'Yüz bin Lira borcum var' cümlesindeki okuduğun tüm sayıları
 rakam olarak yazarak cümleyi bana söyle. Sadece cümle ile cevap ver"
    }
  ]
}
```

Kod 10 ChatGpt API'sine gönderilen json

11. API'ye json olan body'i gönder ve dönüş değerini responseData değişkenine ata.
12. responseData değişkenini deserialize ederek json'dan ChatGptGetMessageModel tipindeki messageModel nesnesine dönüştür.

```
{
    "id": "chatcmpl-7imeB72pVZoPtlMY0tGYUpgBW07DM",
    "object": "chat.completion",
    "created": 1690907787,
    "model": "gpt-3.5-turbo-0613",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": "100.000 Lira borcum var'"
            },
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "prompt_tokens": 56,
        "completion_tokens": 51,
        "total_tokens": 107
    }
}
```

Kod 11ChatGpt API'sinden json tipindeki geri dönüş

13. Eğer messageModel boş değilse content değişkenini döndür.
14. Eğer messageModel boş ise hata mesajı döndür.

- public async Task<IActionResult>GetResponse([FromBody] InputTextModel inputText) -> [FromBody] InputTextModel tipinde 1 adet parametre alan ve kullanıcıdan gelen cümle girdisini json olarak alıp, giriş verisini AskYourQuestionAsync methoduna gönderip oradan dönen değeri json olarak gönderen NumBot Uygulamasının POST tipinde async methodudur.

```
[HttpPost]
0 references
public async Task<IActionResult> GetResponse([FromBody] InputTextModel inputText)
{
    string result = inputText.UserText; ;

    result = await AskYourQuestionAsync(result);

    OutputTextModel response = new OutputTextModel() {
        OutPut = result
    };

    return Json(response);
}
```

Kod 12 GetResponse methodu

Methodun kaba kodu aşağıdaki gibidir.

1. String tipindeki result değişkeni inputText.UserText ile doldurulur.
2. Result değişkeni AskYourQuestionAsync methoduna yollanır ve geri dönen değer result değişkenine eşitlenir.
3. OutPutTextModel sınıfından OutPut değişkeninin result olacak biçimde response adında nesne üretilir.
4. response değişkeni json olarak gönderilir.

- public async Task<IActionResult> GetResponseWithParam (string inputText) -> GetResponse methodunun API'de parametre ile istek gönderilmek istendiğinde kullanılan tipidir. İki method arasındaki tek fark GetResponse methodu json olarak parametre alırken bu method string tipinde inputText adında bir parametre alarak işlem yapılır.

```
[HttpPost]
0 references
public async Task<IActionResult> GetResponseWithParam(string inputText)
{
    string result = inputText;

    result = await AskYourQuestionAsync(inputText);

    var response = new { Output = result };

    return Json(response);
}
```

Kod 13 GetResponseWithParam methodu

API Endpoints ve Postman ile Örnek İstekler

API Endpoints

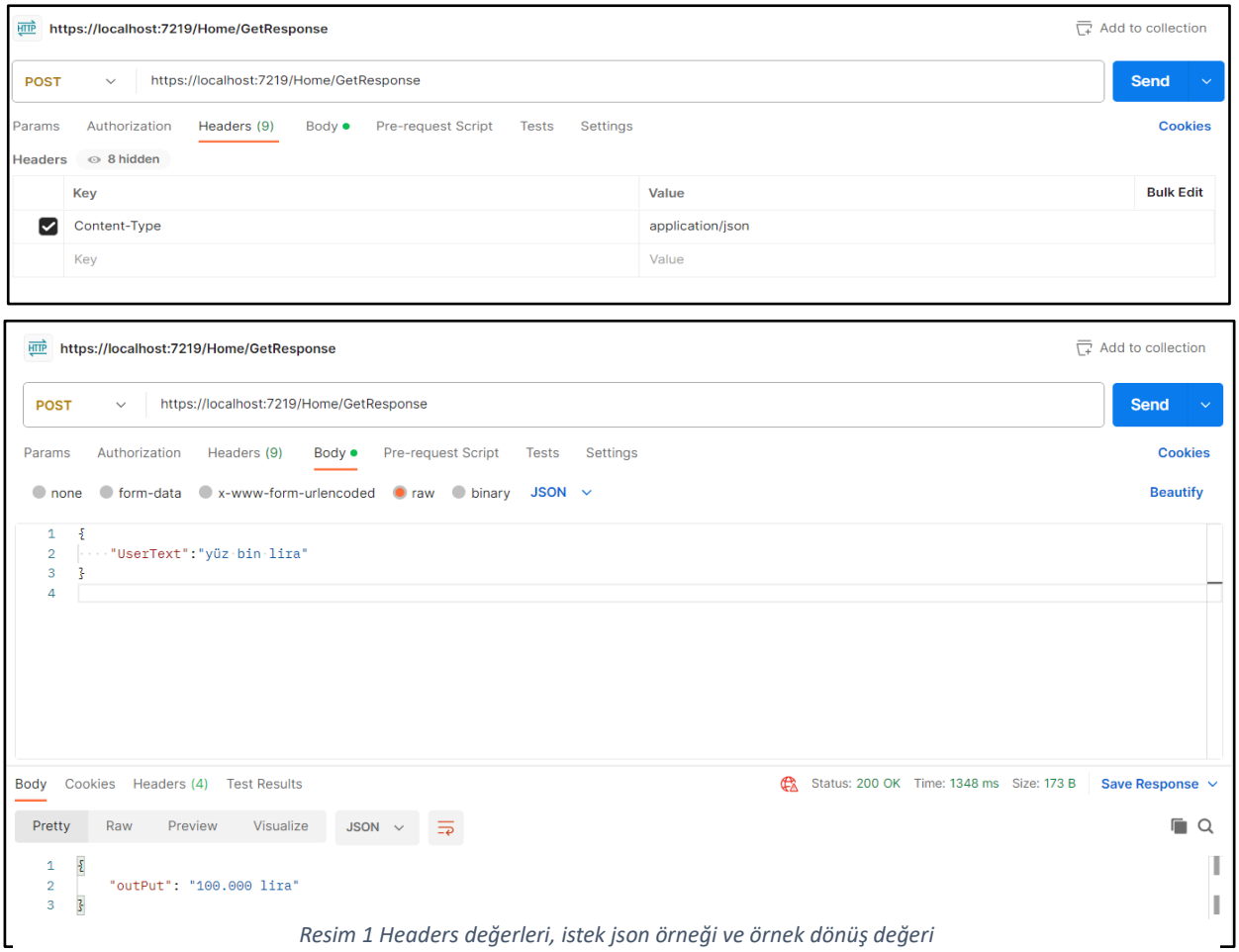
NumBot uygulamasının API'sinde iki adet EndPoint bulunmaktadır. API'deki EndPoint'ler için herhangi bir yetkilendirme bulunmamaktadır. Bunun sebebi projenin, açık kaynak kodlu ve test projesi olarak amaçlanması sebebiyledir. API, herhangi bir sunucuda yayınlanmadığı için şu an local bir sunucuda çalışmaktadır.

EndPoint Adresi	Headers	Body/Parametre	Response
https://localhost:XXXX/Home/GetResponse	Content-Type = application/json	{ "UserText":"" }	{ "output": "" }
https://localhost:XXXX/Home/GetResponseWithParam	-	inputText = ""	{ "output": "" }

Postman ile Örnek İstekler

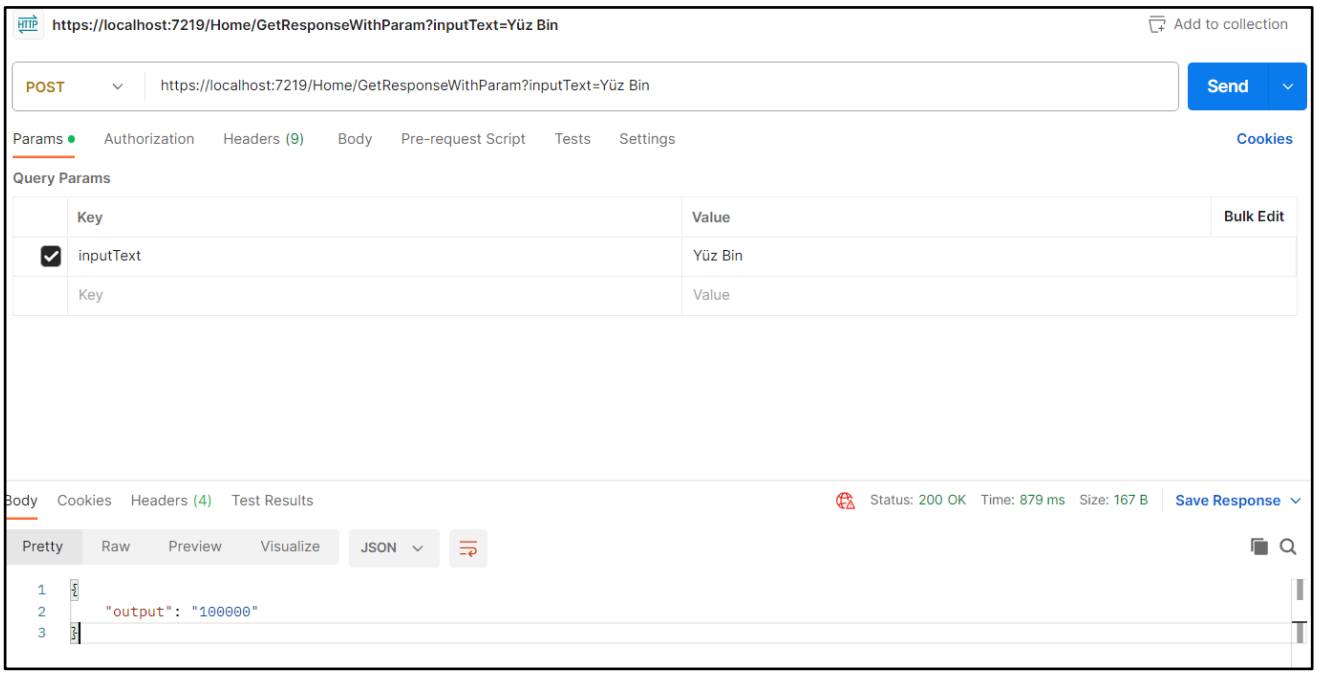
Bu başlık Postman uygulaması ile örnek istekleri göstermektedir.

GetResponse



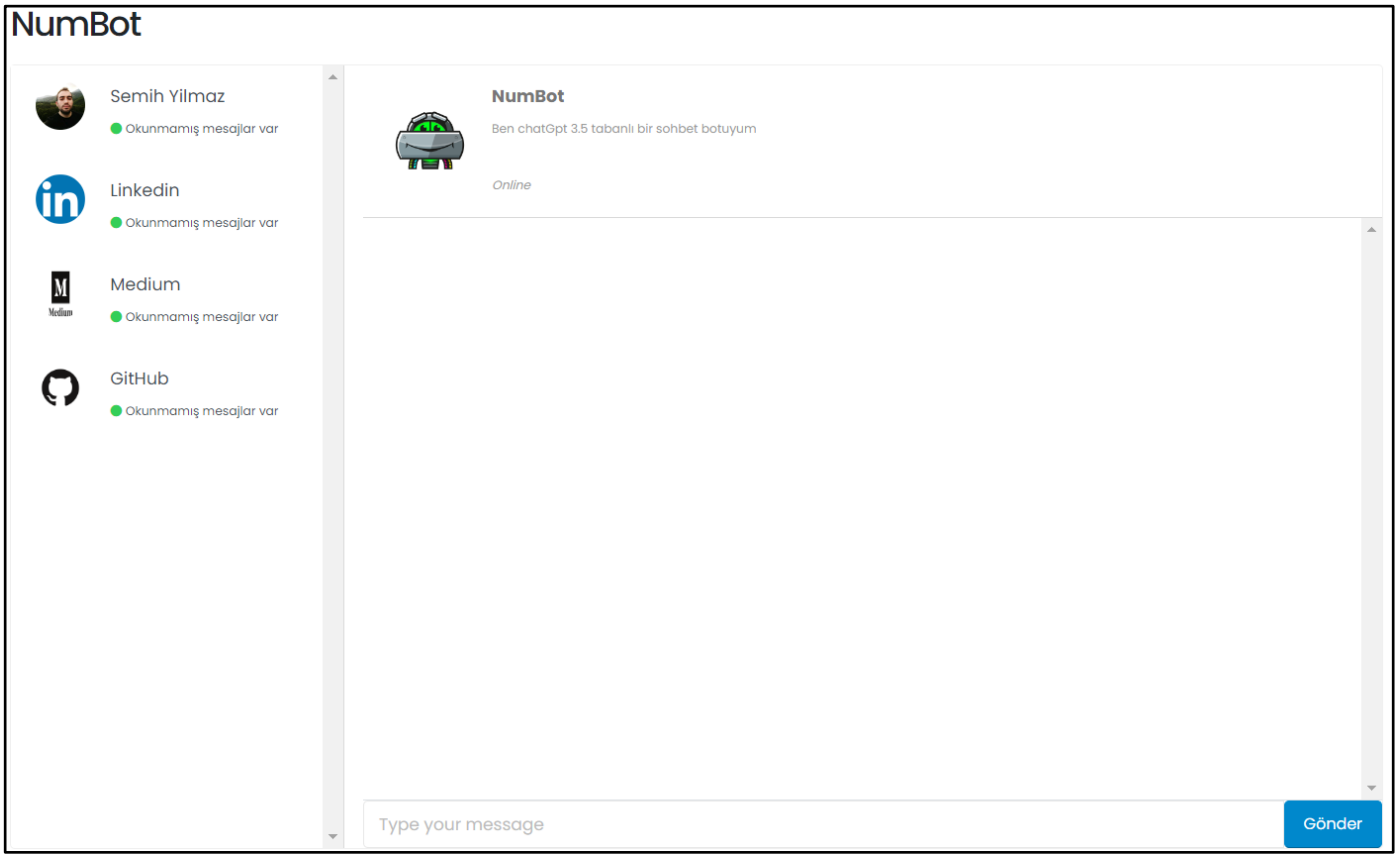
Resim 1 Headers değerleri, istek json örneği ve örnek dönüş değeri

GetResponseWithParam



Resim 2 GetResponseWithParam örnek istek ve döüş değerleri

Kullanıcı Arayüzü

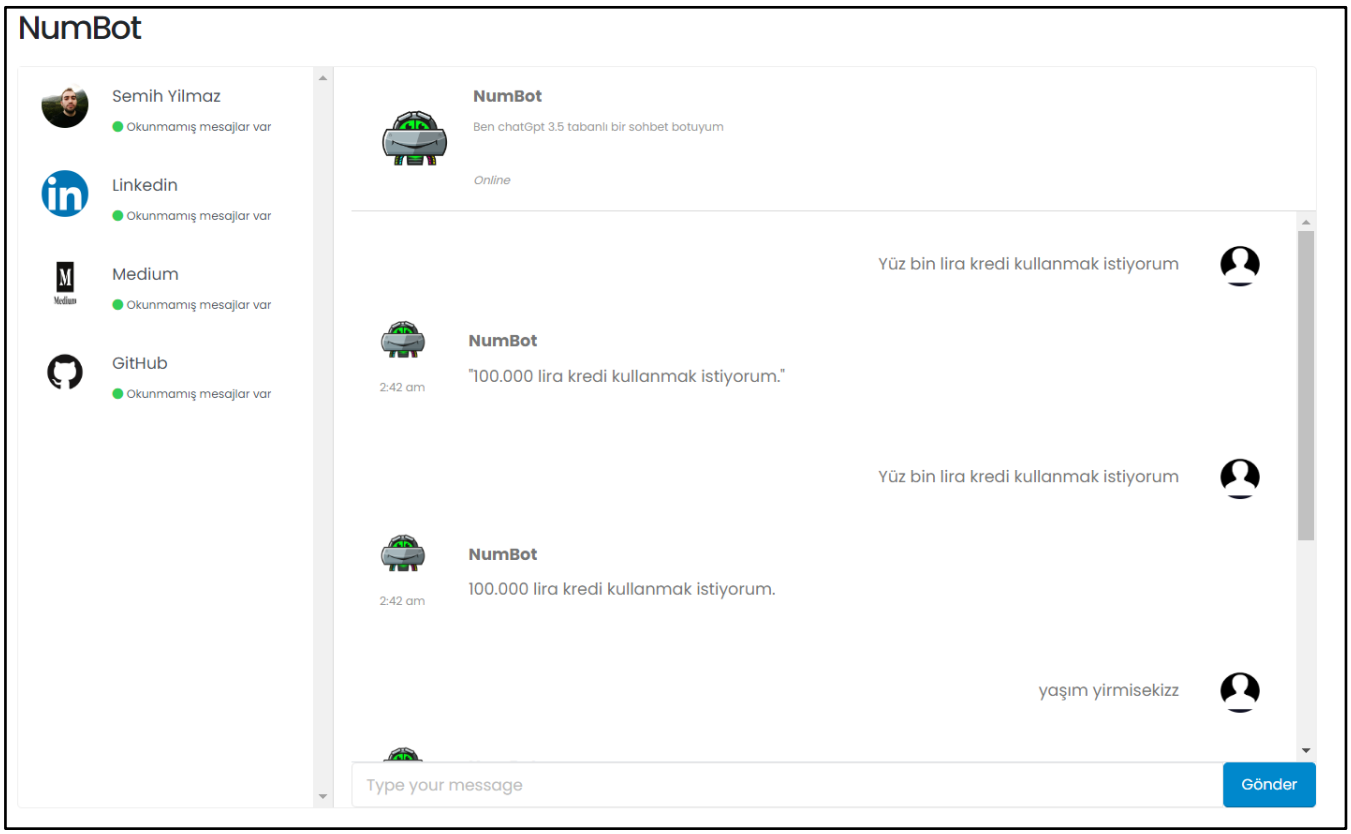


Resim 3 NumBot Uygulaması ana ekran

NumBot uygulamasının kullanıcı tarafında, kullanıcı deneyimini artırmak için kullanıcı arayüzü mesajlaşma arayüzü olarak tasarlanmıştır. Tasarlanırken Bootstrap kütüphanesi kullanılmış olup ek olarak CSS, Javascript ve Jquery/Ajax methodlarından yaralanılmıştır.

Arayüz tek bir Index.cshtml ile tasarlanmış olup, mesajlaşma işlemlerinin gösterilmesi için Particials adında bir View klasörü eklenmiş olup gönderen ve alıcıyı temsil eden iki adet .cshtml dosyalar oluşturularak Index.cshtml view'inde bu dosyalar Partials olarak gösterilmiştir.

Kullanıcı arayüzünde sağ tarafta mesajlaşma arayüzü bulunurken sol tarafında geliştirici ile ilgili bilgilerin bulunduğu linkler bulunmaktadır.



Resim 4 Kullanım sırasında NumBot kullanıcı arayüzü

Geliştirici Notları

Geliştirme OpenAI'ın ürünü olan ChatGpt 3.5 turbo ürününden yardım alınarak geliştirilmiştir. Bu ürünün kullanılmasının sebebi proje isterlerinden olan NLP ile entegre olacak bir .Net projesinin istenmesidir. Baştan sona sıfırdan türkçe NLP uygulaması yapmak hem zaman açısından hem de Türkçe dilinde bulunan NLP verilerinin az olmasından dolayı tercih edilmemiştir. Bunun yerine gayet türkçe NLP olarak gayet güçlü ve ulaşılması kolay olan ChatGpt 3.5 Turbo ürününden faydalanılmıştır.

Kullanılan Teknolojiler

Geliştirmede ASP. Net framework'ü ile MVC tasarımı kullanılmış olup, Ön uçda Bootstrap kütüphanesi, HTML, CSS ve Javascript programala dili ve Jquery teknolojilerinden yararlanılmıştır. Arka uçda ise C# programa dili teknolojisinden yararlanılmıştır.