IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Membangun ChatBot dengan Blazor Hybrid dan Azure Open AI

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

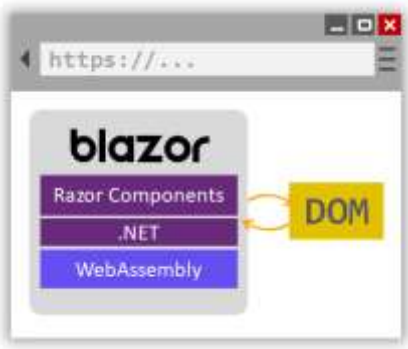
Abstrak

Blazor Hybrid adalah pendekatan inovatif yang menggabungkan framework Blazor dengan .NET MAUI (Multi-platform App UI). Yang memungkinkan kita sebagai *developer* untuk membangun aplikasi *cross platform* menggunakan teknologi web yang sudah dikenal. Dalam aplikasi Blazor Hybrid, komponen Razor berjalan secara native di perangkat dan direnderer ke kontrol Web View yang tertanam melalui local interop. Sehingga komponen ini tidak berjalan di browser dan tidak melibatkan WebAssembly.

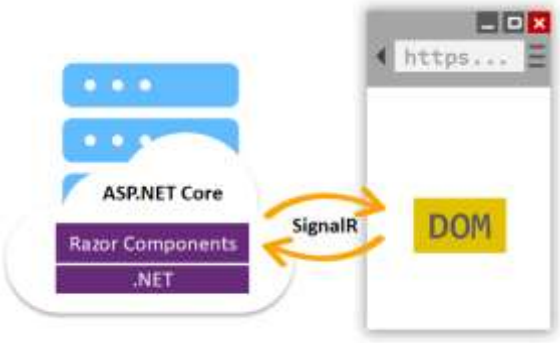
Pendahuluan

Blazor adalah Web Framework yang bersifat Open Source dimana aplikasi Web yang bersifat client-side interactive dapat dikembangkan dengan menggunakan .Net (C#) dan HTML. Pada saat ini C# biasa digunakan untuk melakukan proses back-end dari aplikasi web. Dengan menggunakan fitur baru dari ASP.NET Core yaitu Blazor, kita dapat membangun interactive WEB dengan menggunakan C# dan .NET .

Code .Net berjalan pada Web Assembly, yang artinya kita dapat menjalankan "NET" didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.



Dengan menggunakan Blazor kita dapat memilih apakah menggunakan WebAssemby (Client Side), seperti yang telah dijelaskan di atas atau Blazor yang dijalankan diatas server. Dengan menggunakan cara kedua kita memerlukan SignalR untuk menghubungkan antara client (browser) dan server app. Sebagai contoh jika user melakukan proses klik button pada browser, maka data akan dikirmkan ke Server menggunakan SignalR dan hasilnya akan dikembalikan ke client dengan mengupdate DOM pada client.



Aplikasi Blazor menggunakan runtime .NET yang didasarkan pada Web Assembly atau disingkat WASM. WASM didukung secara native oleh mayoritas browser.

Blazor Hybrid adalah pendekatan inovatif yang menggabungkan framework Blazor dengan .NET MAUI (Multi-platform App UI). Yang memungkinkan kita sebagai *developer* untuk membangun aplikasi *cross platform* menggunakan teknologi web yang sudah dikenal. Dalam aplikasi Blazor Hybrid, komponen Razor berjalan secara native di perangkat dan direnderer ke kontrol Web View yang tertanam melalui local interop. Sehingga komponen ini tidak berjalan di browser dan tidak melibatkan WebAssembly.

Disarankan untuk membaca dan menyeselesaikan latihan pada ebook dari penulis tentang Blazor Hybrid pada tautan berikut. https://tinyurl.com/4t3d9bw6

Chatbot

Chatbot atau chatterbot adalah sebuah layanan obrolan robot/tokoh virtual dengan kecerdasan buatan atau AI (Artificial Intelligent) yang menirukan percakapan manusia melalui pesan suara, obrolan teks ataupun keduanya.

Pada dasarnya bots bekerja dengan cara melihat kata kunci dalam data yang masuk dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari basis data tekstual. Artinya, jika pengguna mengirim suatu permintaan maka bots akan membalasnya dengan respon yang spesifik sesuai dengan kata kunci yang dikirim.

Gambar

Sebagai program komputer yang dirancang untuk berinteraksi dengan manusia melalui antarmuka percakapan, seperti pesan teks atau suara. Chat bot menggunakan kecerdasan buatan dan pemrosesan bahasa alami untuk memahami pertanyaan atau perintah yang diberikan oleh pengguna, lalu memberikan respons atau melakukan tindakan yang sesuai berdasarkan pemrograman atau algoritma yang telah ditetapkan.

Chat bot dapat digunakan untuk berbagai tujuan, seperti memberikan informasi, menjawab pertanyaan pengguna, melakukan reservasi, memberikan layanan pelanggan, atau bahkan hanya untuk hiburan. Mereka dapat diimplementasikan dalam berbagai platform, termasuk situs web, aplikasi seluler, layanan pesan instan, dan banyak lagi.

Microsoft Azure

Microsoft Azure adalah platform komputasi awan yang disediakan oleh Microsoft. Ini adalah salah satu penyedia layanan komputasi awan terkemuka di dunia. Azure menyediakan berbagai layanan cloud yang mencakup infrastruktur sebagai layanan

(Infrastructure as a Service/IaaS), platform sebagai layanan (Platform as a Service/PaaS), dan perangkat lunak sebagai layanan (Software as a Service/SaaS).

Azure memungkinkan pengguna untuk menyimpan data, menjalankan aplikasi, dan mengelola sumber daya IT di lingkungan cloud, tanpa perlu membangun dan memelihara infrastruktur fisik sendiri. Platform ini menawarkan berbagai fitur dan layanan termasuk komputasi awan, penyimpanan awan, basis data awan, kecerdasan buatan, Internet of Things (IoT), analisis data, keamanan, dan banyak lagi.

Azure OpenAI Services

Azure OpenAI Services adalah sejumlah layanan yang disediakan oleh Microsoft Azure yang memungkinkan pengembang untuk mengintegrasikan teknologi kecerdasan buatan yang dikembangkan oleh OpenAI ke dalam aplikasi mereka. Ini mencakup layanan seperti Azure Cognitive Services, yang menyediakan berbagai fitur AI dan Machine Learning, serta integrasi khusus dengan model-model canggih yang dikembangkan oleh OpenAI untuk tugas-tugas seperti pemrosesan bahasa alami, pengenalan gambar, dan lainnya.

Dengan menggunakan Azure OpenAI Services, pengembang dapat dengan mudah memanfaatkan kemampuan AI dan Machine Learning tanpa perlu mengembangkan model mereka sendiri dari awal. Ini mempercepat proses pengembangan aplikasi yang cerdas dan membantu dalam menciptakan solusi yang lebih canggih untuk berbagai macam masalah bisnis dan teknis.

Azure OpenAI Service menyediakan akses REST API ke model OpenAI yang powerfull termasuk seri model GPT-4, GPT-4 Turbo dengan Vision, GPT-3.5-Turbo. Selain itu, seri model GPT-4 dan GPT-3.5-Turbo baru kini telah tersedia secara umum. Model ini dapat dengan mudah disesuaikan dengan tugas spesifik namun tidak terbatas pada pembuatan konten, ringkasan, pemahaman gambar, pencarian semantik, dan terjemahan natural language ke kode.

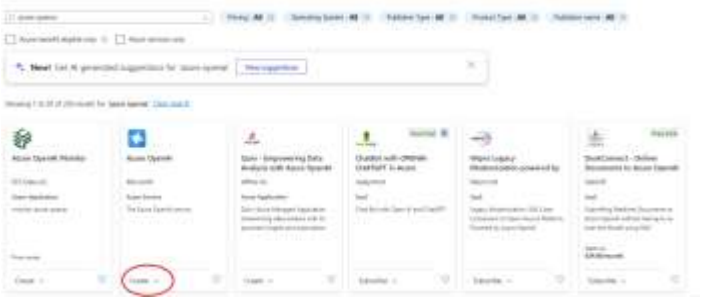
Pada artikel ini tidak menjelaskan apa itu blazor hybrid, bagaimana bekerja dengan project maupun cara-cara untuk menambahkan item pada project, karena semuanya telah dijelaskan baik pada ebook maupun artikel-artikel sebelumnya. Pastikan anda telah

menyelesaikan latihan-latihan pada artikel maupun ebook tersebut. Artikel ini akan fokus membuat chatbot dengan menggunakan Azure Open AI pada Blazor Hybrid.

Sebelum kita mulai dengan membuat project pada Blazor Hybrid, pastikan kita telah memiliki Azure Account terlebih dahulu. Jika belum memiliki Azure Account, kita dapat membuatnya terlebih dahulu disini https://azure.microsoft.com/en-us/free/ .

Setelah selesai dengan langkah-langkah diatas, kita lanjutkan dengan membuat Azure OpenAI Service Resource.

- Login pada Azure Portal : https://portal.azure.com/
- Pilih "Create a resource" dan cari "Azure OpenAI".



Gambar

- Lalu klik "Create" pada Azure OpenAI yang ditampilkan pada hasil pencarian.

Gambar

Pada halaman "Create Azure OpenAI" tab Basic, masukkan informasi yang diperlukan.

- Lalu pada Tab "Network" pilih "***All networks, including the internet, can access this resource.***" Dan klik Next button.
- Dilanjutkan dengan klik "Review+create" button.

Gambar

- Jika semua informasi sudah benar klik button “Create“. Lalu tunggu hingga ada notifikasi jika resource yang dibuat telah selesai.

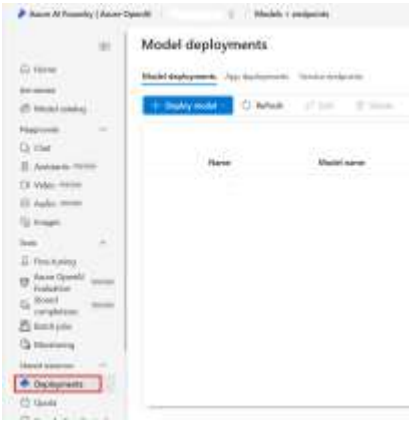
Setelah selesai membuat Resource seperti dengan langkah-langkah diatas, kita lanjutkan dengan melakukan Deploy Model pada Azure OpenAI Studio. Azure OpenAI Studio merupakan sebuah platform yang dikembangkan oleh Microsoft Azure yang memungkinkan pengguna untuk mengakses layanan dan teknologi kecerdasan buatan dari OpenAI. Ini memungkinkan pengembang dan peneliti untuk membuat, melatih, dan menerapkan model kecerdasan buatan menggunakan infrastruktur dan layanan cloud yang disediakan oleh Microsoft Azure, sambil memanfaatkan kemampuan AI yang dikembangkan oleh OpenAI. Ikuti langkah-langkah dibawah ini.

- Buka halaman Azure OpenAI Studio (https://oai.azure.com/).

Gambar

- Lalu pilih Directory dan Azure OpenAI resource yang telah kita buat sebelumnya. Dan klik button “Use resource“.

- Pada halaman Home pilih “Deployment“ pada section “Shared resources“



- Selanjutnya pilih “Deploy model >> Deploy base model“. Lalu pada popup “Select a model“, pilih model yang diinginkan dan klik button “Confirm“.

Gambar

- Lalu akan tampil popup Deploy seperti gambar dibawah lalu masukkan Deployment Name dan Deployment type. Dan klik Deploy button.

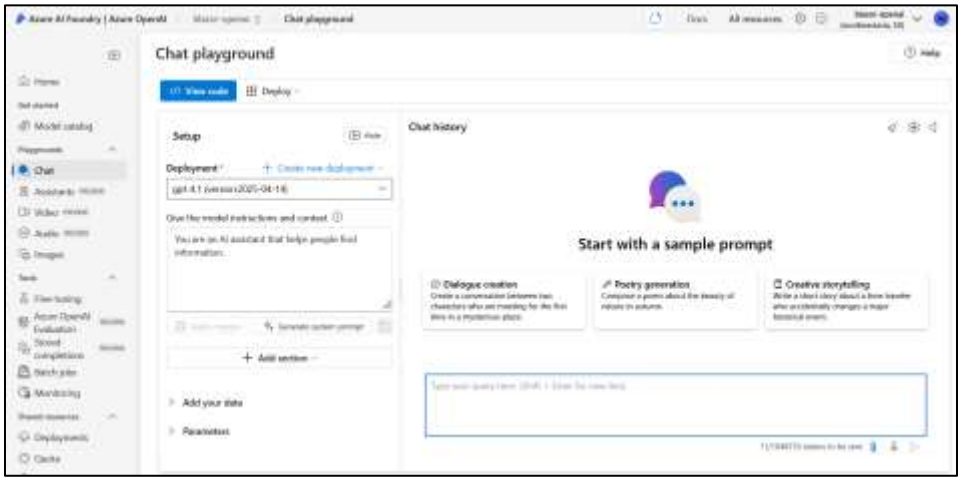
Gambar

Note:

Model : Ketersedian model bervariasi berdasarkan Region.

Deployment Name : Nama deployment yang akan digunakan pada code untuk memanggil Model dengan menggunakan Client Library atau REST API.

Untuk melakukan percobaan, klik Chat pada section Playground. Disini kita bisa melakukan percobaan percakapan dengan ChatGPT, seperti pada gambar dibawah ini.



Gambar

Setelah selesai dengan langkah-langkah diatas, kita lanjutkan dengan membuat sebuah aplikasi Chatbot pada Blazor Hybrid dengan menggunakan Azure OpenAI.

Buat sebuah project Blazor Hybrid dan ikuti langkah-langkah dibawah ini.

- Tambahkan nuget “Microsoft.Extensions.Http“ pada project.
- Buat sebuah folder dengan nama “Models“ dan tambahkan sebuah class dengan nama “ChatMessage“.

```
public class ChatMessage
{
    public string Role { get; set; }
    public string Content { get; set; }

    public ChatMessage(string role, string content)
    {
        Role = role;
        Content = content;
    }
}
```

- Lalu buat folder “Service“ dan tambahkan Interface dan Class masing-masing bernama “IChatService“ dan “ChatService“. Berikut sintaks detail dari dua file tersebut.

  Fungsi dari method yang ada pada class “ChatService” Adalah membuat request ke Azure Open AI dengan membawa pesan yang di ketikkan oleh pengguna lalu menunggu jawabannya dan mengirimkan kembali balasan dari AI tersebut.

  Interface

```
public interface IChatService
{
    Task<string> SendMessageAsync(List<ChatMessage> messages);
}
```

  Class

```
public class ChatService : IChatService
{
     private readonly HttpClient _httpClient;
     private readonly string _endpoint;
     private readonly string _deploymentId;
     private readonly string _apiKey;
     private readonly string _apiVersion;

     public ChatService(HttpClient httpClient)
     {
         _httpClient = httpClient;
         _endpoint = "https://xxxxxx.openai.azure.com/";
         _deploymentId = "gpt-4.1";
         _apiKey = "xxxxxxx";
         _apiVersion = "versionNo";
     }
     public async Task<string> SendMessageAsync(List<ChatMessage> messages)
     {
        try
         {
           //Sintaks detail pada lampiran project
          }
         catch (Exception ex)
         {
             return $"Exception: {ex.Message}";
         }
     }
```

  Terdapat 5 buah field readonly yang hanya bisa di-set sekali didalam constructor.

  Data untuk field-field dibawah ini bisa didapatkan pada Azure OpenAI Studio

_endpoint = URL dasar Azure OpenAI.

_deploymentId = nama model (misal: gpt-4).

_apiKey = kunci akses ke Azure OpenAI.

_apiVersion = versi API Azure.

Constructor akan dipanggil saat class dibuat. HttpClient di inject dari luar (menggunakan Dependency Injection). Sedangkan field-field yang lain untuk saat ini kita hardcode (diisi manual).

```
var url = $"{_endpoint}openai/deployments/{_deploymentId}
            /chat/completions?api-version={_apiVersion}";

_httpClient.DefaultRequestHeaders.Clear();
_httpClient.DefaultRequestHeaders.Add("api-key", _apiKey);
_httpClient.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
```

Pada sintaks diata dapat dilihat, kita membuat URL lengkap untuk memanggil API Azure OpenAI, gabungan dari:

- endpoint
- deploymentId
- path /chat/completions
- query ?api-version=

Dilanjutkan dengan menghapus semua header default dari HttpClient. Dan menambahkan header apikey, Dimana nilai dari apikey diambil dari field yang telah kita buat sebelumnya.

```
var requestBody = new
{
    messages = messages.Select(m => new { role = m.Role, content = m.Content }),
    max_tokens = 1000,
    temperature = 0.7
};
```

Sintaks diatas digunakan untuk membuat sebuah variable object (sementara) untuk dijadikan JSON. Dengan isinya sebagai berikut :

- messages= daftar pesan, hanya ambil role + content.
- max_tokens = panjang jawaban maksimal.
- Temperature= kreativitas jawaban.

```
var json = JsonSerializer.Serialize(requestBody);
var content = new StringContent(json, Encoding.UTF8, "application/json");

var response = await _httpClient.PostAsync(url, content);

if (!response.IsSuccessStatusCode)
{
    var errorText = await response.Content.ReadAsStringAsync();
    return $"Error: {response.StatusCode} - {errorText}";
}
using var responseStream = await response.Content.ReadAsStreamAsync();
using var jsonDoc = await JsonDocument.ParseAsync(responseStream);

var completion = jsonDoc.RootElement
    .GetProperty("choices")[0]
    .GetProperty("message")
    .GetProperty("content")
    .GetString();

return completion?.Trim() ?? "(no response)";
```

Dan terakhir kita kirimkan request ke API dan tunggu hasilnya. Lalu cek hasilnya apakah Sukses (Status Code = 200). Jika tidak sukses, maka ambil isi error dan kembalikkan. Jika berhasil/sukses ambil isi dari jawaban dalam bentuk stream. Dan terakhir ambil teks jawaban dari AI dan kembalikkan nilainya.

Lalu buka “MauiProgram.cs” dan tambahkan sintak berikut.

```
builder.Services.AddHttpClient<IChatService, ChatService>();
```

Sintaks berfunsgi untuk mendaftarkan ChatService sebagai implementasi dari interface IChatService, serta menyediakan HttpClient yang otomatis dikonfigurasi dan dikelola oleh sistem Blazor/.NET.

Lalu kita lanjutkan dengan membukan file Home.razor, lalu ganti sintaks-nya seperti dibawah ini.

```
@page "/"
@inject IChatService chatService
@using ChatbotAI.Models

<h3>🤖 AI ChatBot</h3>

<input @bind="userInput" placeholder="Ask anything..." class="form-control" />
<button @onclick="SendMessage" class="btn btn-primary mt-2">Send</button>

<div class="mt-4">
    @if (string.IsNullOrEmpty(reply) == false)
    {
        <div class="alert alert-secondary">@reply</div>
    }
</div>
```

Input teks digunakan untuk memasukkan pesan user, dan kita binding dengan userInput (@bind="userInput") yang artinya nilai input otomatis tersimpan di field userInput, dan perubahan userInput juga akan mempengaruhi tampilan.

Untuk button digunakan untuk mengeksekusi method SendMessage ketika diklik, Dimana Event handler Blazor menerima method Task (async). Setelah selesai dengan code pada razor, kita lanjutkan dengan code pada C# (behind razor)

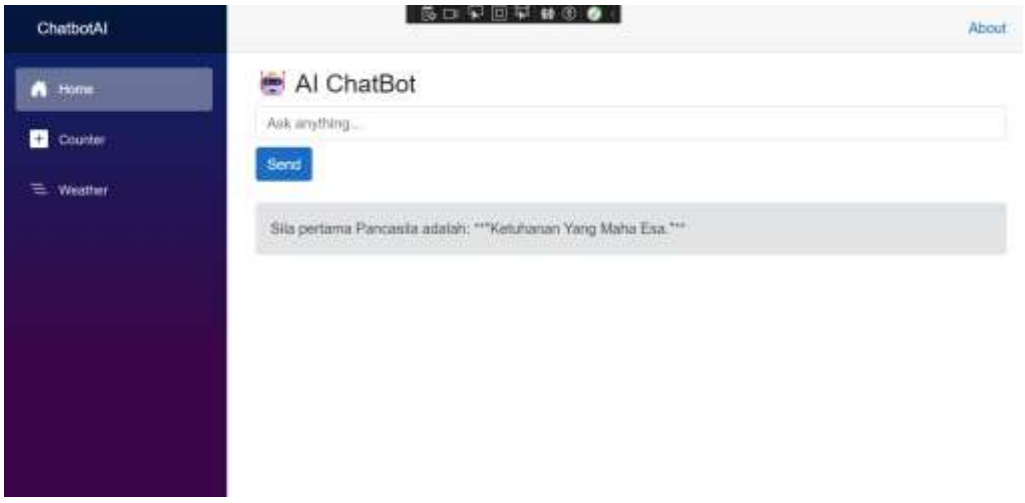
```
@code {

    private string userInput;
    private string reply;
    private List<ChatMessage> messages = new();


    private async Task SendMessage()
    {
        if (string.IsNullOrWhiteSpace(userInput)) return;

        messages.Add(new ChatMessage("user", userInput));

        var response = await chatService.SendMessageAsync(messages);

        if (response != null)
        {

            reply = response;
        }

        userInput = string.Empty;
    }

}
```

userInput: menyimpan teks yang diketik user (di-bind ke <input>).

reply: menyimpan balasan dari service/AI yang akan ditampilkan.

messages: berisi objek ChatMessage dan List ini dikirim ke chatService.

Lalu jalankan program untuk melihat hasilnya. Pastikan mendapatkan hasil seperti pada gambar dibawah.



Dapat dilihat pada contoh diatas kita hanya menampilkan nilai “reply” kedalam halaman. Yang artinya kita tidak memiliki “history” dari pertanyaan atau jawaban sebelumnya. Untuk membuat agar aplikasi kita dapat menampilkan pertanyaan atau jawaban kita sebelumnya, mari ikuti Langkah-langkah dibawah ini.

```
private bool isLoading = false;
```

Tambahkan sebuah private field dengan tipenya adalah Boolean. Lalu pada method “SendMessage”, sebelum memanggil method “SendMessageAsync” dari class ChatService, isikan nilai “isLoading” menjadi true. Dan set kembali menjadi false setelah selesai/berhasil menjalankan method “SendMessageAsync”.

Lalu tambahkan method seperti dibawah.

```
private async Task ScrollToBottom()
{
        await Task.Delay(100);
}
```

Dan panggil method diatas setelah kita ganti nilai “isLoading” menjadi false pada method “SendMessage”.

```
userInput = string.Empty;
isLoading = false;

await ScrollToBottom();
```

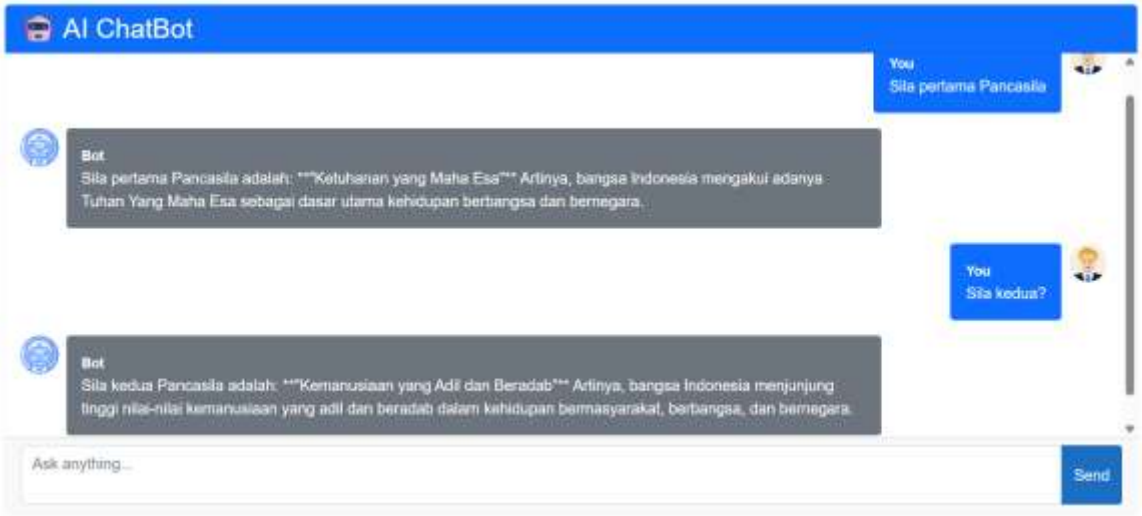
Method ini digunakan untuk menunggu render UI selesai, dalam hal ini komponen baru saja menambahkan pesan baru. Blazor perlu waktu sejenak untuk merender elemen DOM baru. Setelah elemen tersebut muncul di DOM, baru bisa dilakukan scroll otomatis ke bagian bawah.

Sedangkan untuk perubahan code (sintaks detail) pada razor dapat dilihat pada project lampiran.

```
@if (isLoading)
 {
     <div class="text-center mt-3">
         <div class="spinner-grow text-primary" role="status">
             <span class="visually-hidden">Loading...</span>
         </div>
     </div>
 }
```

@if (isLoading) { ... } adalah conditional rendering pada Razor, dimana “spinner“ hanya muncul ketika isLoading bernilai “true“. Dipakai bersama kode C# yang mengganti nilai isLoading sebelum dan sesudah operasi async sehingga pengguna tahu aplikasi sedang memproses tugas.

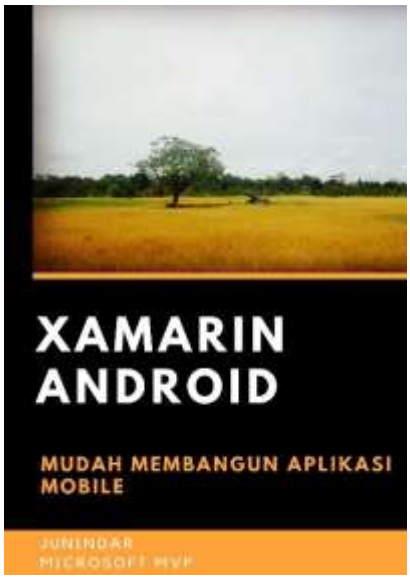
Jalankan program, dan coba beberapa pertanyaan kedalam Chatbot yang telah kita modifikasi. Dan pastikan mendapatkan hasil seperti dibawah.



Penutup

Sedangkan untuk memudahkan dalam memahami isi artikel, maka penulis juga menyertakan dengan full source code project latihan ini, dan dapat di download disini https://junindar.blogspot.com/2025/10/membangun-chatbot-dengan-blazor-hybrid.html

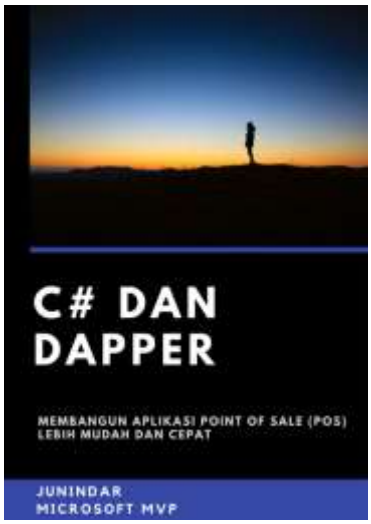
Referensi



https://play.google.com/store/books/details?id=G4tFDgAAQBAJ

https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

https://play.google.com/store/books/details/Junindar_Xamarin_Forms?id=6Wg-DwAAQBAJ



https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

https://play.google.com/store/books/details/Junindar_ASP_NET_MVC_Membangun_Aplikasi_Web_Lebih?id=XLlyDwAAQBAJ

https://play.google.com/store/books/details/Junindar_ASP_NET_CORE_MVC?id=xEe5DwAAQBAJ

Biografi Penulis.

Junindar Lahir di Tanjung Pinang, 21 Juni 1982. Menyelesaikan Program S1 pada jurusan Teknik Inscreenatika di Sekolah Tinggi Sains dan Teknologi Indonesia (ST-INTEN-Bandung). Junindar mendapatkan Award Microsoft MVP VB pertanggal 1 oktober 2009 hingga saat ini. Senang mengutak-atik computer yang berkaitan dengan bahasa pemrograman. Keahlian, sedikit mengerti beberapa bahasa pemrograman seperti : VB.Net, C#, SharePoint, ASP.NET, VBA. Reporting: Crystal Report dan Report Builder. Database: MS Access, MY SQL dan SQL Server. Simulation / Modeling Packages: Visio Enterprise, Rational Rose dan Power Designer. Dan senang bermain gitar, karena untuk bisa menjadi pemain gitar dan seorang programmer sama-sama membutuhkan seni. Pada saat ini bekerja di salah satu Perusahaan Consulting dan Project Management di Malaysia sebagai Senior Consultant. Memiliki beberapa sertifikasi dari Microsoft yaitu Microsoft Certified Professional Developer (MCPD – SharePoint 2010), MOS (Microsoft Office Specialist) dan MCT (Microsoft Certified Trainer) Mempunyai moto hidup: "**Jauh lebih baik menjadi Orang Bodoh yang giat belajar, dari pada orang Pintar yang tidak pernah mengimplementasikan ilmunya".**