IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Sentiment Analysis dengan Blazor Hybrid dan Azure Open AI

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

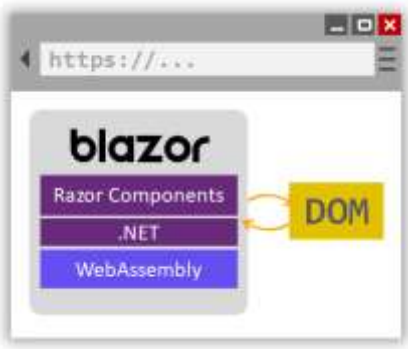
Abstrak

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.

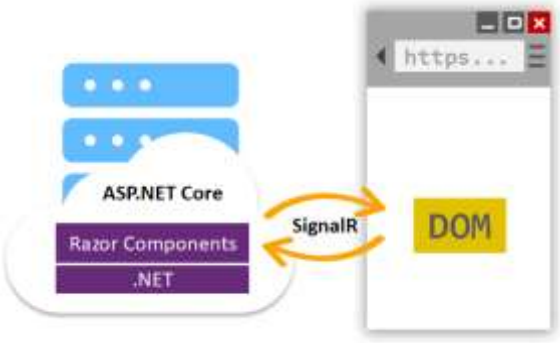
Pendahuluan

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/ .

Pastikan telah membaca dan menyelesaikan artikel-artikel sebelumnya.
https://junindar.blogspot.com/2025/10/membangun-chatbot-dengan-blazor-hybrid.html
https://junindar.blogspot.com/2025/10/pada-latihan-sebelumnya-kita-telah.html
https://junindar.blogspot.com/2025/10/membangun-chatbot-dengan-blazor-hybrid_29.html

Sentiment Analysis pada OpenAI

Sentiment Analysis pada OpenAI adalah kemampuan AI untuk memahami dan mengevaluasi emosi yang terkandung dalam sebuah teks. Dengan memanfaatkan Large Language Model / LLM seperti GPT, sistem ini mampu menentukan apakah sebuah pernyataan bersentimen positif, negatif, atau netral. Lebih dari itu, model juga dapat menangkap nuansa emosional yang lebih kompleks, seperti frustrasi, kegembiraan, kekecewaan, kekhawatiran, atau bahkan sarkasme. Hal ini membuat Sentiment Analysis sangat berguna untuk memahami opini dan respons manusia dalam bentuk teks.

Berbeda dengan metode tradisional yang hanya mengandalkan keyword atau rule-based approach, Sentiment Analysis berbasis OpenAI bekerja dengan memahami konteks penuh dari sebuah kalimat. GPT mempertimbangkan struktur kalimat, hubungan antar kalimat, intensitas emosi, serta gaya bahasa, sehingga dapat menafsirkan teks secara lebih natural. Pendekatan ini memungkinkan AI mengenali sentiment bahkan dalam teks yang ambigu, informal, atau panjang dengan berbagai macam perasaan yang bercampur.

Dalam praktiknya, Sentiment Analysis OpenAI banyak digunakan untuk berbagai aplikasi, seperti customer review analysis dan social media monitorings.Kemampuannya untuk menafsirkan emosi secara konsisten menjadikan teknologi ini sangat berharga bagi bisnis, peneliti, dan pengembang perangkat lunak. Dengan memanfaatkan Sentiment Analysis, organisasi dapat memahami perasaan pengguna secara lebih mendalam, memperbaiki layanan, serta mengambil keputusan berbasis data yang lebih tepat.

Pada latihan sebelumnya, kita telah berhasil membangun sebuah aplikasi chatbot yang memiliki beberapa fitur penting, seperti history dan session. Fitur history memungkinkan chatbot untuk menyimpan percakapan sebelumnya sehingga pengguna bisa melihat kembali interaksi yang sudah dilakukan, sedangkan fitur session membantu menjaga konteks percakapan agar chatbot dapat memberikan jawaban yang relevan sesuai dengan topik yang sedang dibahas. Dengan kedua fitur ini, pengalaman menggunakan chatbot menjadi lebih interaktif dan personal, karena sistem dapat "mengingat" percakapan pengguna dari waktu ke waktu.

Dalam artikel ini, kita akan membahas langkah-langkah secara bertahap untuk membangun aplikasi sentiment analysis menggunakan teknologi Open AI. Kita akan mempelajari cara mengirim teks ke model AI, bagaimana model tersebut menilai sentimen, dan cara menampilkan hasil analisis secara mudah dipahami. Dengan mengikuti tutorial ini, pembaca tidak hanya belajar membuat aplikasi analisis sentimen, tetapi juga memahami konsep dasar bagaimana AI dapat membaca dan memahami emosi manusia melalui teks. Hasil akhirnya adalah sebuah aplikasi yang interaktif, informatif, dan bisa digunakan untuk berbagai keperluan analisis data teks.

Note : Untuk memudahkan pemahaman terhadap kode atau sintaks yang akan kita gunakan, sangat disarankan untuk membuat proyek baru terlebih dahulu dan menyalin seluruh kode dari artikel sebelumnya. Pastikan proyek tersebut berjalan dengan baik sebelum melanjutkan, agar kita dapat fokus pada penambahan fitur baru tanpa terganggu oleh kesalahan dari konfigurasi sebelumnya.

Mari kita mulai latihan ini dengan mengikuti langkah-langkah yang akan dijelaskan secara bertahap.

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 " SentimentResult".

```
public class SentimentResult
{
    public string? Language { get; set; }
    public string Sentiment { get; set; }
    public double Confidence { get; set; }
    public string Explanation { get; set; }
}
```

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

  Interface

```
public interface IChatService
{
        Task<SentimentResult> AnalyzeSentimentAsync(string text);
}
```

  Class

```
public class OpenAiService: IOpenAiService
{
     private readonly HttpClient _httpClient;
     private readonly string _endpoint;
     private readonly string _deploymentId;
     private readonly string _apiKey;
     private readonly string _apiVersion;

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

  Sintaks di atas mendefinisikan sebuah kelas **OpenAiService** yang mengimplementasikan interface **IOpenAiService**, yang bertujuan untuk menyediakan layanan komunikasi dengan API OpenAI, khususnya untuk fitur sentiment analysis. Di dalam kelas ini terdapat beberapa properti penting seperti **_httpClient** untuk melakukan permintaan HTTP, **_endpoint** yang menyimpan URL layanan Azure OpenAI, **_deploymentId** yang menentukan model AI yang digunakan (misalnya gpt-4.1), **_apiKey** untuk otentikasi ke Open AI, dan **_apiVersion** untuk versi API yang dipanggil. Constructor kelas ini menerima HttpClient sebagai parameter, sehingga komunikasi HTTP dapat dilakukan secara efisien dengan menggunakan dependency injection pada aplikasi .NET.

Metode **AnalyzeSentimentAsync** adalah fungsi asinkron yang menerima teks sebagai input dan bertugas untuk mengembalikan hasil analisis sentimen. Di dalam metode ini, logika utama untuk memanggil API OpenAI ditempatkan di blok try, yang kemungkinan besar akan membangun permintaan HTTP, mengirim teks ke model AI, dan memproses hasil respons menjadi objek **SentimentResult**. Blok catch menangkap semua kemungkinan pengecualian yang terjadi selama proses komunikasi dengan API dan mengembalikan pesan error sebagai string, sehingga aplikasi tetap aman dan tidak crash ketika terjadi kesalahan jaringan, otentikasi, atau masalah internal lainnya.

```
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"));

var prompt = $@"
You are a sentiment analysis assistant.
Analyze the sentiment of the following text and respond ONLY in strict JSON format:
{{
  ""Sentiment"": ""Positive|Negative|Neutral"",
  ""Confidence"": 0.0,
  ""Explanation"": ""Short reason""
}}
Text: ""{text}""";


var requestBody = new
{
    messages = new[]
    {
        new { role = "system", content = "You are a helpful AI assistant specialized
        in sentiment analysis." },
        new { role = "user", content = prompt }
    },
    temperature = 0.0,
    max_tokens = 200
};
```

Potongan sintaks diatas menunjukkan bagaimana aplikasi menyiapkan dan mengirim permintaan ke Azure OpenAI API untuk melakukan analisis sentimen. Pertama-tama, URL endpoint dibangun menggunakan **_endpoint**, **_deploymentId**, dan **_apiVersion**, sehingga request diarahkan ke model tertentu (misalnya gpt-4.1) yang telah dideploy. Setelah itu, header HTTP disiapkan, semua header sebelumnya dibersihkan (**Clear()),** kemudian ditambahkan api-key untuk autentikasi.

Langkah-langkah ini untuk memastikan komunikasi dengan API aman, terotentikasi, dan respons dapat diproses dengan mudah oleh aplikasi.

Selanjutnya, prompt untuk AI dibuat menggunakan string interpolasi. AI diminta untuk menganalisis sentimen teks dan hanya memberikan jawaban dalam format JSON.

JSON ini mencakup tiga field: Sentiment (Positive/Negative/Neutral), Confidence (tingkat keyakinan AI terhadap prediksi), dan Explanation (alasan singkat). Lalu, request body dibuat sebagai objek anonim yang berisi array messages, mengikuti format Chat API OpenAI. Di sini, ada dua peran system untuk memberi instruksi umum tentang kemampuan AI, dan user untuk menyertakan prompt analisis sentimen. Parameter temperature diatur ke 0.0 untuk membuat respons lebih deterministik dan max_tokens dibatasi 200 agar output tetap ringkas. Dengan konfigurasi ini, aplikasi siap mengirim permintaan ke model AI dan mendapatkan hasil analisis sentimen secara terstruktur.

```
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 new SentimentResult
    {
        Sentiment = "InvalidFormat",
        Explanation = errorText
    };
}

using var responseStream = await response.Content.ReadAsStreamAsync();
using var jsonDoc = await JsonDocument.ParseAsync(responseStream);


var textOut = jsonDoc.RootElement
    .GetProperty("choices")[0]
    .GetProperty("message")
    .GetProperty("content")
    .GetString()?
    .Trim();
if (string.IsNullOrWhiteSpace(textOut))
     return null;
```

Di bagian ini, objek **requestBody** yang sudah dibuat sebelumnya diubah menjadi string JSON menggunakan **JsonSerializer.Serialize**, lalu dibungkus dalam objek **StringContent** agar bisa dikirim melalui HTTP POST dengan tipe konten **application/json**. Kemudian request dikirim ke OpenAI API dengan **_httpClient.PostAsync(url, content)**. Setelah request dikirim, ada pengecekan **IsSuccessStatusCode** untuk memastikan respons dari server berhasil. Kalau server merespons dengan status gagal, misalnya karena API key salah atau format request

tidak sesuai, kode akan membaca pesan error dari server (ReadAsStringAsync) dan mengembalikan objek **SentimentResult** dengan **Sentiment** = "InvalidFormat" serta **Explanation** berisi detail error. Dengan cara ini, aplikasi tetap bisa menangani kegagalan request tanpa crash.

Kalau respons dari server berhasil, kode akan membaca stream respons (**ReadAsStreamAsync**) dan mem-parse-nya menjadi JSON document menggunakan **JsonDocument.ParseAsync**. Dari JSON ini, kode menavigasi ke properti **choices[0].message.content** untuk mendapatkan output teks hasil analisis sentimen, lalu melakukan Trim() untuk menghapus spasi yang tidak perlu. Jika output kosong atau hanya berisi spasi, metode akan mengembalikan null, menandakan tidak ada data valid yang diterima. Dengan cara ini, kode memastikan bahwa hanya hasil analisis yang valid yang diproses lebih lanjut, sementara respons yang kosong atau gagal tetap ditangani dengan aman.

```
try
{
    var result = JsonSerializer.Deserialize<SentimentResult>(textOut,
        new JsonSerializerOptions { PropertyNameCaseInsensitive = true });
    return result;
}
catch
{
    return new SentimentResult
    {
        Sentiment = "InvalidFormat",
        Explanation = textOut
    };
}
```

Dan terakhir pada bagian ini, sintaks digunakan untuk mengubah string **textOut** yang diterima dari OpenAI menjadi objek **SentimentResult** menggunakan **JsonSerializer.Deserialize**, dengan opsi PropertyNameCaseInsensitive = true agar deserialisasi tidak sensitif terhadap huruf kapital pada nama properti. Kalau proses deserialisasi berhasil, objek **SentimentResult** yang valid langsung dikembalikan. Tapi kalau terjadi error, misalnya karena format JSON yang diterima tidak sesuai atau string kosong, blok catch akan menangkap exception tersebut dan mengembalikan objek SentimentResult baru dengan Sentiment = "InvalidFormat" dan Explanation berisi textOut asli, sehingga aplikasi tetap aman dan kita bisa melihat apa yang salah dari respons AI.

- Selanjutnya buka file Home.razor, lalu ganti sintaksnya seperti dibawah ini.

```
@page "/"
@using TextAnalyticsIntro.Models
@using TextAnalyticsIntro.Service
@inject IOpenAiService OpenAiService
<h3>Sentiment Analysis</h3>

<textarea @bind="InputText" rows="6" style="width:100%"></textarea>
<div class="mt-2">
    <button class="btn btn-primary" @onclick="Analyze">Analyze</button>
</div>

@if (Result != null)
{
    <div class="card mt-3 p-3">
        <h5>Sentiment: @Result.Sentiment</h5>
        <p>Confidence: @Result.Confidence</p>
        <p><b>Explanation:</b> @Result.Explanation</p>
    </div>
}

@code {
    string InputText = string.Empty;
    SentimentResult? Result;

    async Task Analyze()
    {
        if (string.IsNullOrWhiteSpace(InputText)) return;
        Result = await OpenAiService.AnalyzeSentimentAsync (InputText);
    }
}
```

Sintaks pada Home ini adalah halaman Blazor sederhana yang digunakan untuk melakukan sentiment analysis secara interaktif langsung. Di bagian atas, halaman mengimpor namespace untuk model dan service, serta menggunakan dependency injection pada **IOpenAiService** agar metode **AnalyzeSentimentAsync** bisa dipanggil. Pada tampilan UI, terdapat textarea untuk input teks pengguna yang diikat (bind) ke variabel **InputText**, serta sebuah tombol “Analyze” yang akan mengeksekusi method Analyze ketika diklik. Method ini memeriksa apakah InputText tidak kosong, lalu memanggil service OpenAI untuk menganalisis sentimen teks yang dimasukkan, dan menyimpan hasilnya di variabel Result.

Bagian bawah halaman menampilkan hasil analisis jika Result tidak null, dengan informasi seperti Sentiment, Confidence, dan Explanation, semuanya dibungkus dalam card agar mudah dibaca. Dengan struktur ini, pengguna dapat mengetik teks, menekan tombol analisis, dan langsung melihat hasil sentiment analysis secara real-time. Halaman ini memberikan pengalaman interaktif yang sederhana

tapi efektif untuk memahami bagaimana AI menilai perasaan atau opini dari teks yang diberikan.

Lalu setelah selesai dengan Home.razor, kita lakukan register interface dan class diatas pada “MauiProgram.cs”, seperti pada gambar dibawah ini.

```
#if DEBUG
        builder.Services.AddBlazorWebViewDeveloperTools();
        builder.Logging.AddDebug();
#endif

        builder.Services.AddHttpClient<IOpenAiService, OpenAiService>();
```

Selanjutnya jalankan program untuk mendapatkan hasil seperti pada gambar dibawah.

Setelah tahap sebelumnya berhasil diterapkan, kita dapat melanjutkan dengan menambahkan fitur pendeteksian bahasa. Fitur ini memungkinkan sistem OpenAI untuk secara otomatis mengenali bahasa apa yang digunakan oleh pengguna ketika mereka memasukkan teks ke dalam textbox. Dengan kemampuan deteksi bahasa ini,

aplikasi dapat menjadi lebih fleksibel dan responsif terhadap berbagai jenis pengguna tanpa perlu pengaturan manual.

Selain itu, hasil keluaran dari OpenAI juga akan menyesuaikan dengan bahasa yang terdeteksi. Jika pengguna menulis dalam bahasa Indonesia, maka respons yang diberikan akan berbahasa Indonesia, begitu pula jika pengguna menggunakan bahasa lainnya. Dengan demikian, pengalaman pengguna menjadi lebih natural, intuitif, dan efisien karena sistem secara otomatis menyesuaikan konteks.

Ikuti Langkah-langkah dibawah ini.

- Buka file SentimentResult.cs dan tambahkan property dengan nama “Language”

```
public string? Language { get; set; }
```

- Lalu pada method “AnalyzeSentimentAsync” di class “OpenAiService” ganti nilai dari variable “prompt” menjadi seperti berikut.

```
var prompt = $@"
You are a multilingual sentiment analysis expert.
Your task:
1. Detect the language of the text automatically.
2. Determine if the sentiment is Positive, Negative, or Neutral.
3. Estimate a confidence score (0 to 1).
4. Give a short explanation in the same language as the text.
Return strictly valid JSON, no extra words.

Example format:
{{
  ""Language"": ""Indonesian"",
  ""Sentiment"": ""Negative"",
  ""Confidence"": 0.93,
  ""Explanation"": ""Kalimat ini menunjukkan ketidakpuasan terhadap layanan
pelanggan.""
}}

Text to analyze:""{text}""";
```

Sintaks ini sebenarnya berfungsi untuk membuat sebuah prompt yang akan dikirim ke model AI. Di dalamnya sudah dijelaskan bahwa model bertugas mendeteksi bahasa, menentukan sentimen, memberi skor dari Confidence, dan menjelaskan hasilnya dalam bahasa yang sama dengan teks yang dianalisis. Format jawabannya juga sudah diatur harus dalam bentuk JSON yang rapi, jadi aplikasi yang menerima responsnya bisa mengolahnya dengan baik.Lalu pada bagian akhirnya, ada penyisipan variabel **text** yang merupakan input asli dari pengguna.

- Dan terakhir buka Home.razor, lalu tambahkan sintaks seperti pada gambar dibawah ini.

```
<div class="card mt-3 p-3">
    <h5>Language: @Result.Language</h5>
    <h5>Sentiment: @Result.Sentiment</h5>
    <p>Confidence: @Result.Confidence</p>
    <p><b>Explanation:</b> @Result.Explanation</p>
</div>
```

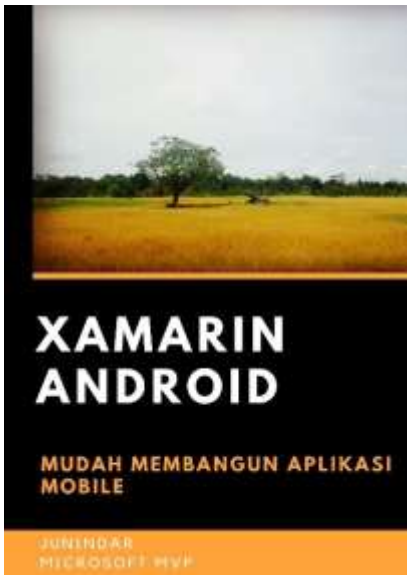
Selanjutnya jalankan program untuk mendapatkan hasil seperti pada gambar dibawah.

Jika ingin mendapatkan hasil seperti gambar dibawah, agar kelihatan lebih profesional. Sintaksnya dapat dilihat pada project lampiran. Yang banyak berubah adalah dari sisi UI-nya saja. Sedangkan back-end atau C#-nya hanya beberapa bagian saja yang berubah.

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/2025/11/sentiment-analysis-dengan-blazor-hybrid.html

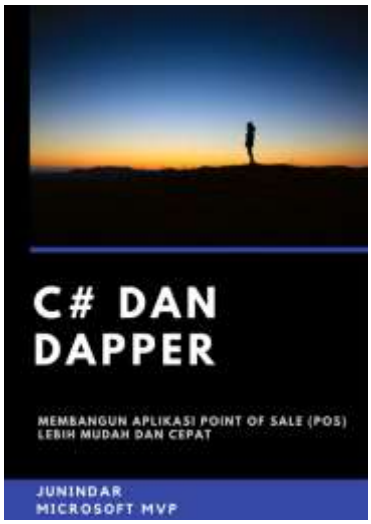
Referensi



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

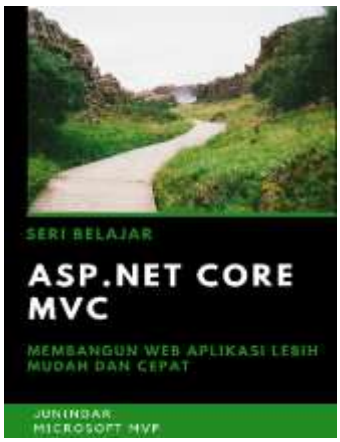
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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".**