IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Azure Computer Vision

Dengan Blazor Hybrid

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

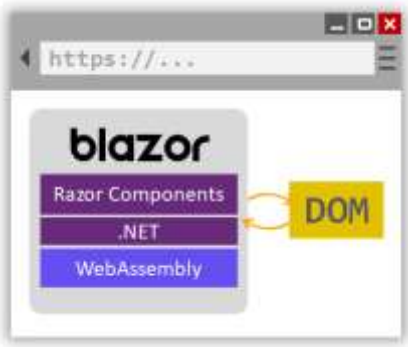
Abstrak

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.

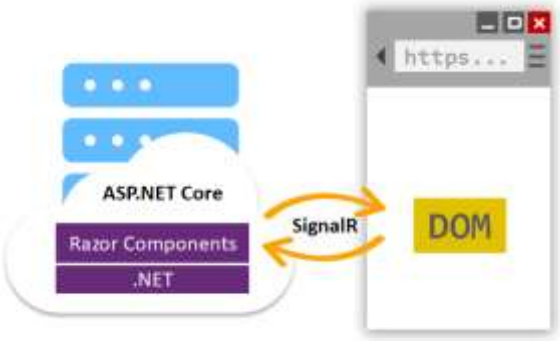
Pendahuluan

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 Computer Vision

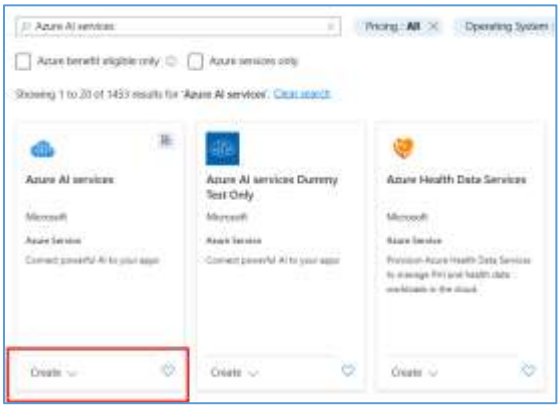
Azure Computer Vision adalah layanan dari Microsoft yang memanfaatkan kecerdasan buatan (AI) untuk menganalisis gambar atau video. Layanan ini memungkinkan komputer untuk mengenali objek-objek yang ada dalam gambar. Misalnya, ketika diberikan sebuah foto pemandangan atau gambar orang, Azure dapat mengenali berbagai elemen dalam gambar seperti pohon, mobil, atau bahkan aktivitas tertentu. Teknologi yang digunakan, seperti deep learning, dilatih dengan jutaan gambar, sehingga semakin sering digunakan, semakin akurat pula hasil analisisnya.

Selain itu, Azure Computer Vision memiliki kemampuan Optical Character Recognition (OCR), yang memungkinkan layanan ini untuk "membaca" teks yang ada di dalam gambar. Fitur ini sangat berguna ketika ingin mengekstrak teks dari gambar atau dokumen, seperti foto catatan atau poster. Tidak hanya itu, layanan ini juga dapat mengenali wajah dan bahkan mendeteksi emosi yang ada di dalam foto. Dengan demikian, Azure membantu mengubah gambar statis menjadi informasi yang bisa digunakan lebih lanjut.

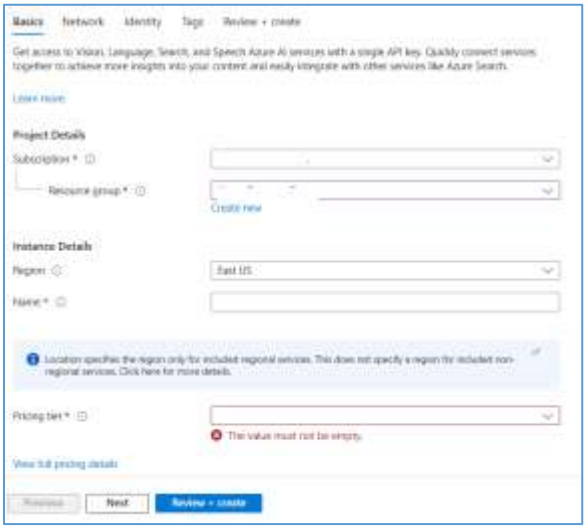
Yang lebih menarik, Azure Computer Vision dapat diaplikasikan dalam berbagai bidang. Mulai dari pengidentifikasian produk dalam foto, analisis gambar medis, hingga pembuatan aplikasi yang dapat memahami konteks gambar yang ada di media sosial. Karena menggunakan teknologi cloud, proses analisis dapat dilakukan dengan cepat dan tanpa perlu khawatir soal kapasitas penyimpanan atau pemrosesan data besar. Layanan ini memberikan kemudahan dan kecanggihan dalam memproses gambar dan video secara efisien.

Sebelum kita masuk ke coding, pertama-tama kita buat “Azure AI services multi-service account“. Ikuti langkah-langkah dibawah ini.

- Login pada Azure Portal : https://portal.azure.com/
- Pilih “Create a resource“ dan cari “Azure AI Services“.



- Lalu klik “Create“ pada Azure AI services yang ditampilkan pada hasil pencarian.



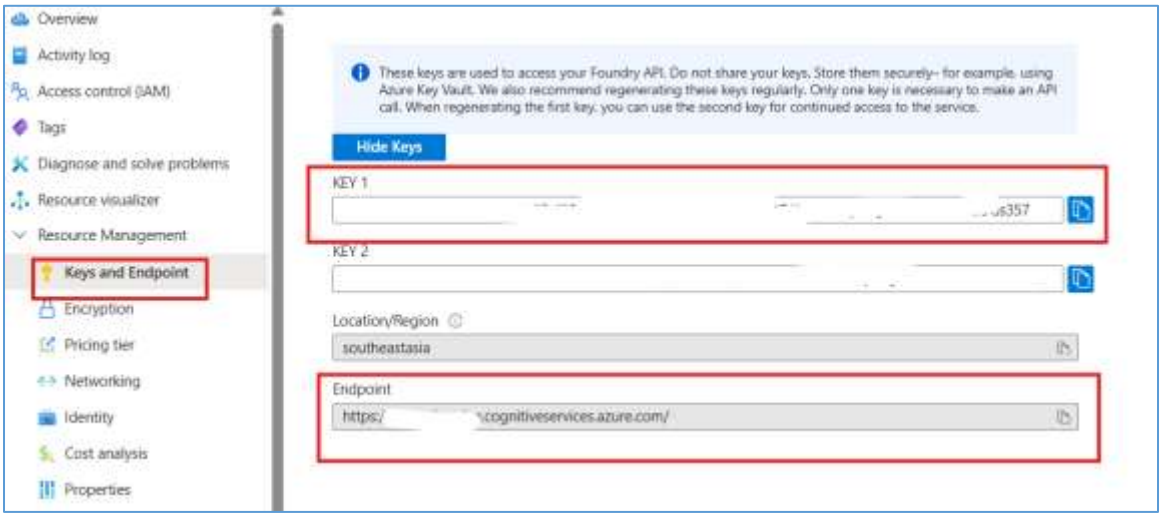
Pada halaman “Create Azure AI services“ tab Basic, masukkan informasi yang diperlukan.

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

- Jika semua informasi sudah benar klik button “Create“. Lalu tunggu hingga ada notifikasi jika resource yang dibuat telah selesai. Hasilnya seperti pada gambar dibawah ini.

Untuk mendapatkan key dan url (endpoint) cognitive services, klik “Keys and Endpoint“ pada “Resource Management“. Key dan url (endpoint) akan digunakan pada Blazor.

Untuk mendapatkan key dan url (endpoint) cognitive services, klik “Keys and Endpoint“ pada “Resource Management“. Key dan url (endpoint) akan digunakan pada Blazor.



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.

Pastikan telah membaca dan mengikuti seluruh latihan pada artikel sebelumnya.

https://junindar.blogspot.com/2025/11/sentiment-analysis-dengan-blazor-hybrid.html

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

- Tambahkan nuget " Azure.AI.TextAnalytics" pada project.

  Azure.AI.Vision.ImageAnalysis NuGet adalah paket library yang disediakan Microsoft untuk mempermudah penggunaan layanan Azure Vision Image Analysis di aplikasi berbasis .NET. Dengan NuGet ini, proses integrasi analisis gambar jadi lebih simpel karena sudah tersedia class dan fungsi siap pakai. Pengembang bisa langsung memanggil fitur seperti deteksi objek, deskripsi gambar, hingga ekstraksi informasi visual tanpa perlu membangun logika dari awal. Pendekatan ini membuat pengembangan aplikasi berbasis AI terasa lebih praktis dan efisien.

  Melalui Azure.AI.Vision.ImageAnalysis NuGet, komunikasi dengan layanan Azure berlangsung secara terstruktur dan konsisten. Proses pengiriman gambar, pengaturan parameter analisis, hingga pengolahan hasil respon dapat dilakukan dengan lebih rapi di dalam kode. Hal ini sangat membantu dalam pengembangan aplikasi modern, baik

untuk web, desktop, maupun layanan backend, karena analisis gambar dapat diintegrasikan dengan cepat tanpa mengorbankan performa dan skalabilitas aplikasi.

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

Interface

```
public interface IComputerVisionService
{
     Task<string> AnalyzeImageAsync(string imagePath);
}
```

Class

```
public interface IComputerVisionService
{
     Task<string> AnalyzeImageAsync(string imagePath);
}


public class ComputerVisionService: IComputerVisionService
{
   private readonly string _subscriptionKey = "xxxxxx";
   private readonly string _endpoint = "https://xxxx.cognitiveservices.azure.com/";

   private readonly ImageAnalysisClient _client;

   public ComputerVisionService()
   {

     _client = new ImageAnalysisClient(
      new Uri(_endpoint),
      new AzureKeyCredential(_subscriptionKey));
   }


  public async Task<string> AnalyzeImageAsync(string imagePath)
  {
     using var imageStream = new FileStream(imagePath, FileMode.Open);
     var imageData = BinaryData.FromStream(imageStream);
     var options = new ImageAnalysisOptions
     {
       Language = "en"
     };

     var result = await _client.AnalyzeAsync(
              imageData,
              VisualFeatures.Caption,
              options);
     if (!result.HasValue || result.Value.Caption == null)
     {
        return "Tidak ada deskripsi";
     }

     var caption = result.Value.Caption.Text;
     return caption;
  }
}
```

Pada sintaks di atas mendefinisikan sebuah class bernama **ComputerVisionService** yang berfungsi untuk menganalisis gambar menggunakan layanan **Azure Computer Vision**. Class ini mengimplementasikan interface **IComputerVisionService** dan menyimpan dua konfigurasi utama, yaitu subscription key (**_subscriptionKey**) dan endpoint (**_endpoint**) yang digunakan untuk mengakses layanan Azure. Selain itu, terdapat objek **ImageAnalysisClient** yang menjadi penghubung utama antara aplikasi dan layanan Computer Vision.

Pada bagian contructor, objek **ImageAnalysisClient** diinisialisasi dengan memasukkan alamat endpoint dan subscription key melalui **AzureKeyCredential**. Proses ini memastikan aplikasi memiliki izin untuk mengakses layanan analisis gambar dari Azure. Dengan pendekatan ini, client hanya dibuat satu kali saat object service dibuat, sehingga lebih efisien dan terorganisir.

Method **AnalyzeImageAsync** bertugas membaca file gambar dari path yang diberikan, lalu mengirimkannya ke layanan Computer Vision untuk dianalisis. Gambar dibuka sebagai stream, dikonversi menjadi BinaryData, kemudian dianalisis dengan fitur Caption untuk menghasilkan deskripsi gambar dalam bahasa Inggris. Jika deskripsi tidak ditemukan, method akan mengembalikan teks default. Jika berhasil, hasil caption berupa teks deskripsi gambar akan dikembalikan sebagai string.

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

```
builder.Services.AddSingleton<IComputerVisionService, ComputerVisionService>();
```

Sintaks berfungsi untuk mendaftarkan ComputerVisionService sebagai service dengan lifetime Singleton di dependency injection. Artinya, hanya akan dibuat satu instance untuk seluruh aplikasi, dan instance itu akan dipakai berulang kali setiap kali service ini dibutuhkan.

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

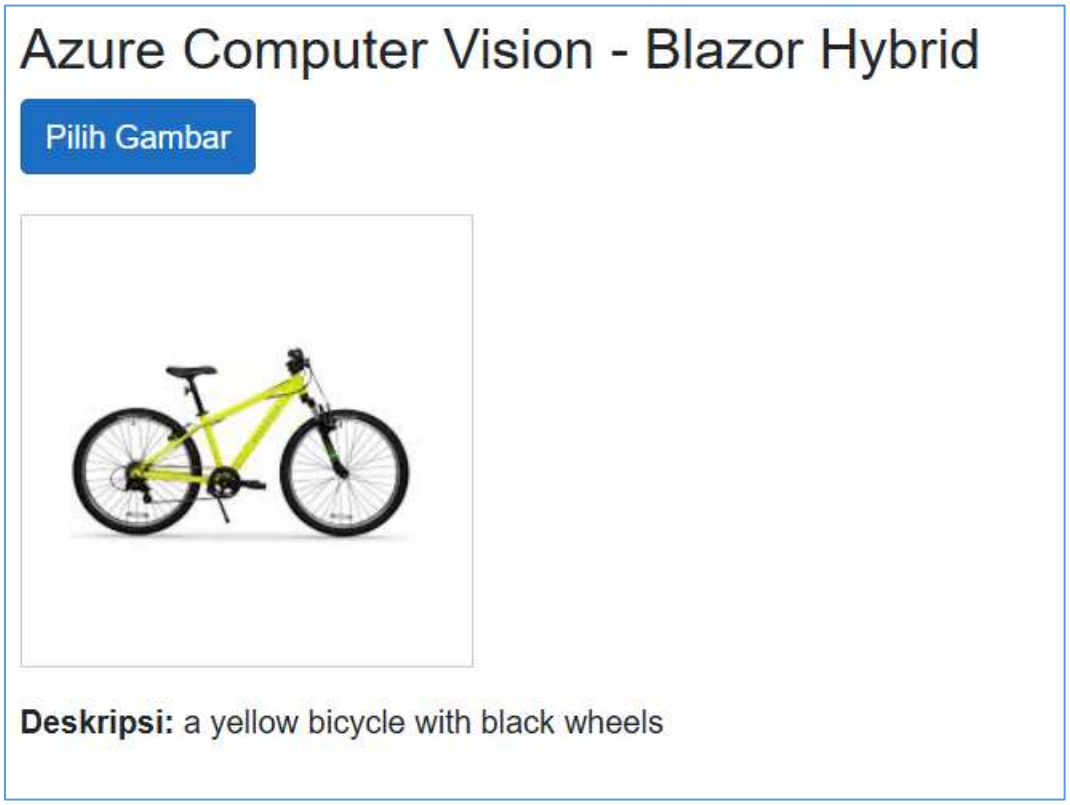
```
@page "/"
@using ComputerVisionIntro.Service
@using ComputerVisionIntro.Helper
@inject IComputerVisionService VisionService

<h3>Azure Computer Vision - Blazor Hybrid</h3>

<button class="btn btn-primary" @onclick="PickImage">
    Pilih Gambar
</button>

@if (!string.IsNullOrEmpty(ImagePreview))
{
    <div style="margin-top:20px">
        <img src="@ImagePreview"
             style="max-width:400px; border:1px solid #ccc;" />
    </div>
}

@if (!string.IsNullOrEmpty(Description))
{
    <p style="margin-top:15px">
        <b>Deskripsi:</b> @Description
    </p>
}

@code {
    private string? ImagePreview;
    private string? Description;

    private async Task PickImage()
    {
        var file = await FilePickerHelper.PickImageAsync();
        if (file == null) return;
        ImagePreview = await FileHelper.ToBase64ImageAsync(file);
        Description = await VisionService.AnalyzeImageAsync(file);
    }
}
```

Sintaks di atas merupakan halaman utama Blazor yang berfungsi sebagai antarmuka untuk fitur Azure Computer Vision. Baris **@page "/"** menandakan bahwa halaman ini diakses dari root aplikasi. Beberapa namespace dipanggil menggunakan **@using** untuk mengakses service dan helper yang dibutuhkan, sedangkan **@inject IComputerVisionService VisionService** digunakan untuk memanggil service Computer Vision agar bisa digunakan langsung di halaman ini.

Bagian markup HTML menampilkan judul, tombol Pilih Gambar, serta area tampilan gambar dan hasil deskripsi. Ketika tombol diklik, method **PickImage** akan dipanggil. Jika gambar berhasil dipilih, gambar tersebut akan ditampilkan menggunakan tag **<img>** dengan sumber data berbentuk **Base64**. Deskripsi gambar hanya akan ditampilkan jika nilai Description tidak kosong, sehingga tampilan tetap rapi dan dinamis.

Pada bagian @code, terdapat dua properti, yaitu **ImagePreview** untuk menyimpan data gambar yang akan ditampilkan dan **Description** untuk menyimpan hasil analisis gambar. Method **PickImage** berjalan secara asynchronous, dimulai dengan memilih gambar dari perangkat, lalu mengonversinya ke format Base64 untuk ditampilkan di layar. Setelah itu, gambar dikirim ke service Computer Vision melalui **AnalyzeImageAsync** untuk menghasilkan deskripsi yang kemudian ditampilkan di halaman.

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.

Azure AI Computer Vision OCR adalah layanan dari Microsoft Azure yang digunakan untuk membaca dan mengenali teks yang terdapat di dalam gambar secara otomatis. Layanan ini mampu mendeteksi berbagai jenis teks, baik cetak maupun tulisan tangan, lalu mengubahnya menjadi teks digital yang dapat diproses oleh aplikasi. Teknologi ini mendukung banyak bahasa dan dapat bekerja pada berbagai sumber gambar seperti foto, hasil scan dokumen, atau tangkapan layar.

Dalam pengembangan aplikasi, Azure AI Computer Vision OCR berperan sebagai solusi praktis untuk ekstraksi teks tanpa proses manual. Gambar cukup dikirim ke layanan Azure, kemudian sistem akan mengembalikan hasil pembacaan teks dalam bentuk data terstruktur. Fitur ini sangat berguna untuk kebutuhan seperti digitalisasi dokumen, pembacaan struk, pengolahan formulir, dan otomatisasi alur kerja berbasis teks.

Latihan selanjutnya akan berfokus pada pembacaan struk dari sebuah restoran. Pada proses ini, pengguna memasukkan gambar struk ke dalam aplikasi, kemudian sistem akan memproses gambar tersebut menggunakan teknologi OCR.

Aplikasi akan membaca informasi penting yang terdapat pada struk, seperti daftar pesanan, harga masing-masing item, serta total pembayaran. Hasil pembacaan ini kemudian dapat ditampilkan atau diolah lebih lanjut sesuai kebutuhan aplikasi.

Ikuti Langkah-langkah dibawah ini.

- Buka file IComputerVisionService dan tambahkan sebuah method seperti dibawah ini.

```
Task<(List<string> Lines, string Total, string Date, string Merchant, byte[] ImageData)>
AnalyzeReceiptAsync(
     Stream imageStream);
```

Lalu lanjutkan dengan membuat implementasi untuk method diatas pada ComputerVisionService.

```
public async Task<(List<string> Lines, string Total, string Date, string Merchant, byte[]
ImageData)> AnalyzeReceiptAsync(Stream imageStream)
 {
     byte[] imageBytes;
     using (var ms = new MemoryStream())
     {
         await imageStream.CopyToAsync(ms);
         imageBytes = ms.ToArray();
     }
     var imageData = BinaryData.FromBytes(imageBytes);

     var options = new ImageAnalysisOptions
     {
         Language = "en"
     };


     var result = await _client.AnalyzeAsync(imageData, VisualFeatures.Read, options);

     if (!result.HasValue || result.Value?.Read?.Blocks == null)
     {
         return (new List<string>(), "", "", "", imageBytes);
     }


     var lines = result.Value.Read.Blocks
         .SelectMany(block => block.Lines)
         .Select(line => line.Text.Trim())
         .ToList();

     int totalIndex = lines.FindIndex(l => l.Trim().Equals("Total:"));

     string total = totalIndex >= 0 && totalIndex + 1 < lines.Count
         ? lines[totalIndex + 1]
         : "";


     string date = lines
         .FirstOrDefault(l => l.StartsWith("Tanggal:"))
         ?.Replace("Tanggal:", "")
         .Trim();



     string merchant = lines
         .FirstOrDefault(l => !string.IsNullOrWhiteSpace(l) && !Regex.IsMatch(l, @"[\d.,]+")) ?? "";

     return (lines, total, date, merchant, imageBytes);
 }
```

Method **AnalyzeReceiptAsync** digunakan untuk membaca dan menganalisis gambar struk belanja menggunakan fitur OCR dari Azure Computer Vision. Method ini menerima input berupa Stream gambar dan mengembalikan beberapa data sekaligus, yaitu daftar teks hasil OCR, total pembayaran, tanggal, nama merchant, serta data gambar dalam bentuk byte array. Pendekatan ini memudahkan proses lanjutan tanpa perlu membaca ulang gambar.

Pada bagian awal, stream gambar disalin ke dalam **MemoryStream** lalu diubah menjadi array byte. Data ini kemudian dikonversi menjadi BinaryData agar dapat diproses oleh **ImageAnalysisClient**. Opsi analisis disiapkan melalui **ImageAnalysisOptions** dengan bahasa Inggris sebagai bahasa pengenalan teks, sehingga hasil OCR lebih konsisten.

Proses analisis dilakukan dengan memanggil **AnalyzeAsync** menggunakan fitur **VisualFeatures.Read** untuk membaca teks dari gambar. Jika hasil analisis tidak tersedia atau tidak mengandung data OCR, method langsung mengembalikan nilai kosong namun tetap menyertakan data gambar. Jika berhasil, seluruh baris teks diambil dari setiap blok OCR, dirapikan, lalu disimpan ke dalam list string.

Setelah semua baris teks terkumpul, sistem mulai mengekstrak informasi penting. Nilai total diambil dengan mencari posisi teks “Total:” lalu membaca baris setelahnya, sedangkan tanggal diambil dari baris yang diawali dengan kata “Tanggal:”. Nama merchant ditentukan dari baris pertama yang bukan kosong dan tidak mengandung angka. Seluruh data tersebut kemudian dikembalikan sebagai satu kesatuan hasil analisis.

- Tambahkan sebuah class dengan nama “MyHelper”, dan ketikkan sintaks seperti dibawah.

  Class **MyHelper** adalah helper statis yang berisi beberapa method untuk memproses teks hasil OCR dari struk atau dokumen. Method **Normalize** membersihkan string dari spasi, titik, dan koma agar angka dalam format seperti 1.500 atau 2,000 menjadi mudah diolah, sedangkan IsNumber memeriksa apakah string yang sudah dinormalisasi bisa dikonversi menjadi angka (int). Method IsPrice lebih spesifik untuk mendeteksi angka yang kemungkinan merupakan harga, dengan syarat string bisa dikonversi menjadi angka dan

memiliki panjang minimal 4 karakter setelah dinormalisasi. Semua method ini memudahkan aplikasi dalam mengekstrak dan memvalidasi data numerik dari hasil OCR secara akurat dan konsisten.

```
public static class MyHelper
{
    public static string Normalize(string input)
    {
         return input
             .Trim()
             .Replace(".", "")
             .Replace(",", "");
    }

   public static bool IsNumber(string input)
   {
         return int.TryParse(Normalize(input), out _);
   }

   public static bool IsPrice(string input)
   {
         return int.TryParse(Normalize(input), out _)
                && Normalize(input).Length >= 4;
   }
}
```

- Buka file Home.razor, ganti sintaksnya seperti dibawah ini.

```
@page ""
@using ComputerVisionIntro.Service
@using ComputerVisionIntro.Helper
@using ComputerVisionIntro.Models
@inject IComputerVisionService VisionService
@using System.Text.RegularExpressions
<h3>Scan Struk Belanja</h3>

<InputFile OnChange="OnFileSelected" accept="image/*" />
@if (ImagePreview != null)
{
    <img src="@ImagePreview" style="max-width:300px" />
}

@if (Lines != null)
{
    <h4>Hasil OCR:</h4>
    <p><b>Merchant:</b> @Merchant</p>
    <p><b>Tanggal:</b> @Date</p>
    <p><b>Total:</b> @Total</p>



    <table class="table table-bordered">
        <thead class="table-light">
        <tr>
            <th>No</th>
            <th>Nama Makanan</th>
            <th>Harga</th>
            <th>Jumlah</th>
            <th>Total</th>
        </tr>
        </thead>
        <tbody>
        @foreach (var item in items)
        {
            <tr>
                <td>@item.No</td>
                <td>@item.Nama</td>
                <td>@item.Harga.ToString("N0")</td>
                <td>@item.Jumlah</td>
                    <td>@item.Total.ToString("N0")</td>
            </tr>
        }

        <tr>
            <td colspan="4" class="text-end"><strong>Sub Total</strong></td>
                <td><strong>@subTotal.ToString("N0")</strong></td>
        </tr>
        <tr>
            <td colspan="4" class="text-end"><strong>Pajak (10%)</strong></td>
                <td><strong>@pajak.ToString("N0")</strong></td>
        </tr>
        <tr>
            <td colspan="4" class="text-end"><strong>Total</strong></td>
                <td><strong>@total.ToString("N0")</strong></td>
        </tr>
        </tbody>
    </table>


}
```

Sintaks di atas adalah halaman Blazor yang digunakan untuk membaca struk belanja menggunakan teknologi OCR. Halaman ini menyediakan tombol untuk memilih file gambar melalui **<InputFile>**, dan setelah gambar dipilih, preview

gambar akan ditampilkan di layar. Variabel **ImagePreview** menyimpan data gambar dalam format yang bisa langsung ditampilkan di browser, sehingga pengguna bisa melihat gambar yang akan dianalisis.

Jika hasil OCR berhasil, teks yang diekstrak dari struk disimpan dalam list Lines dan ditampilkan di halaman. Informasi penting seperti merchant, tanggal, dan total pembayaran ditampilkan secara ringkas di atas tabel. Tabel ini menampilkan detail setiap item, termasuk nomor, nama makanan, harga, jumlah, dan total per item, serta menghitung sub total, pajak, dan total akhir. Pendekatan ini membuat hasil OCR mudah dibaca dan rapi, sehingga pengguna bisa langsung melihat ringkasan struk secara visual dan terstruktur.

```
private List<string>? Lines;
 private string? Total;
 private string? Date;
 private string? Merchant;
 private string? ImagePreview;

 private  List<ReceiptItem> items = new();

 private int subTotal;
 private int pajak;
 private int total;

 private async Task OnFileSelected(InputFileChangeEventArgs e)
 {
     var file = e.File;
     if (file == null) return;

     using var stream = file.OpenReadStream();
     var result = await VisionService.AnalyzeReceiptAsync(stream);

     Lines = result.Lines;
     Total = result.Total;
     Date = result.Date;
     Merchant = result.Merchant;

     items.Clear();
     ParseItems();

     subTotal = ExtractValue("sub");
     pajak = ExtractValue("pajak");
     total = ExtractValue("total");

     ImagePreview = $"{Convert.ToBase64String(result.ImageData)}";
 }
```

Kode di atas adalah method **OnFileSelected** yang berjalan ketika pengguna memilih file gambar struk. File dibaca sebagai stream dan dikirim ke service **VisionService.AnalyzeReceiptAsync** untuk diproses dengan OCR. Hasil analisis berupa daftar teks (Lines), total pembayaran (Total), tanggal (Date),

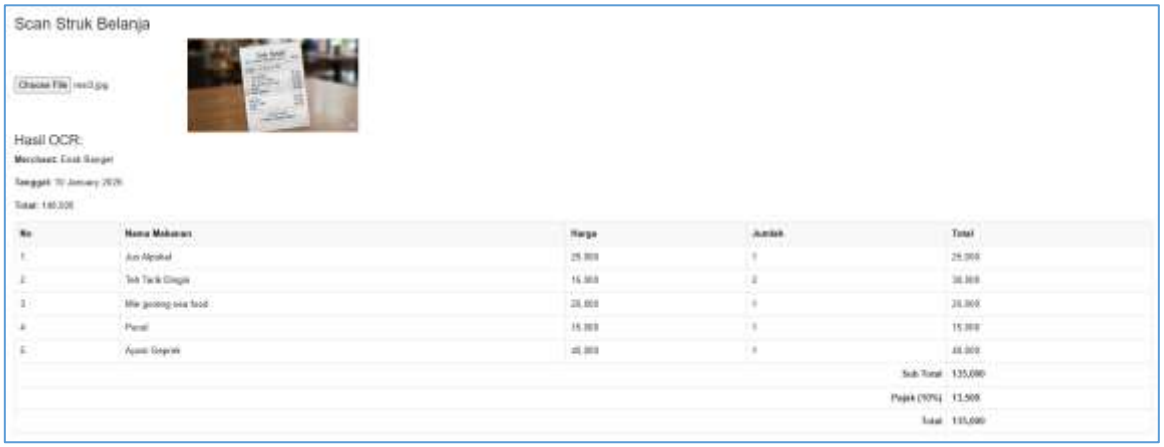
merchant (Merchant), dan data gambar dalam bentuk byte disimpan ke variabel yang sesuai, sehingga aplikasi bisa menampilkan informasi struk secara otomatis.

Setelah data diperoleh, method memanggil **ParseItems()** untuk mengekstrak detail setiap item belanja dari daftar teks, seperti nama, jumlah, dan harga per item, lalu menghitung total per item. Nilai sub total, pajak, dan total akhir diambil menggunakan method **ExtractValue()**. Akhirnya, gambar struk dikonversi ke format Base64 agar bisa langsung ditampilkan di halaman melalui ImagePreview, membuat proses OCR dan tampilan hasil menjadi otomatis dan interaktif.

```
  private void ParseItems()
  {
      int no = 1;
      for (int i = 0; i < Lines.Count; i++)
      {
          string line = Lines[i].Trim();

          if (TryParseCase1(line, ref i, ref no)) continue;
          if (TryParseCase2(i, ref i, ref no)) continue;
      }
  }


  private bool TryParseCase1(string line, ref int i, ref int no)
  {
      if (!Regex.IsMatch(line, @"^\d+\s+.+")) return false;

      var parts = line.Split(' ', 2);
      int qty = int.Parse(parts[0]);
      string nama = parts[1];

      if (i + 1 < Lines.Count && MyHelper.IsPrice(Lines[i + 1]))
      {
          double harga = double.Parse(MyHelper.Normalize(Lines[i + 1])) / qty;
          items.Add(new ReceiptItem { No = no++, Nama = nama, Harga = harga, Jumlah = qty
});
          i++;
      }
      return true;
  }
```

```
private bool TryParseCase2(int index, ref int i, ref int no)
  {
      if (!int.TryParse(Lines[index].Trim(), out int jumlah)) return false;
      if (index + 2 >= Lines.Count || !MyHelper.IsPrice(Lines[index + 2])) return false;

      string nama = Lines[index + 1].Trim();
      double harga = double.Parse(MyHelper.Normalize(Lines[index + 2])) / jumlah;
      items.Add(new ReceiptItem { No = no++, Nama = nama, Harga = harga, Jumlah = jumlah });
      i += 2;
      return true;
  }

  private int ExtractValue(string keyword)
  {
      for (int i = 0; i < Lines.Count; i++)
      {
          string line = Lines[i].Trim().ToLower();
          if (!line.Contains(keyword)) continue;


          if (i + 1 < Lines.Count && MyHelper.IsNumber(Lines[i + 1]))
              return int.Parse(MyHelper.Normalize(Lines[i + 1]));


          var match = Regex.Match(Lines[i], @"([\d.,]+)$");
          if (match.Success) return int.Parse(MyHelper.Normalize(match.Value));
      }
      return 0;
  }
```

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.

Scan Struk Belanja

Choose File res3.jpg

Gambar Struk

Hasil OCR

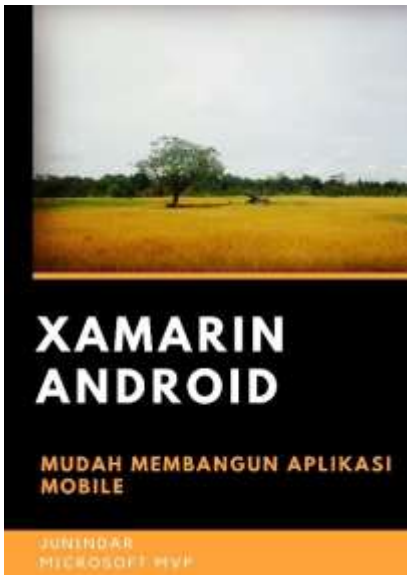
Merchant	Tanggal	Total
Enak Banget	10 January 2026	148,500

No	Nama Makanan	Harga	Jumlah	Total
1	Jus Alpukat	25,000	1	25,000
2	Teh Tarik Dingin	15,000	2	30,000
3	Mie goreng sea food	25,000	1	25,000
4	Pocal	15,000	1	15,000
5	Ayam Geprek	40,000	1	40,000
			Sub Total	135,000
			Pajak (10%)	13,500
			Total	135,000

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/2026/09/azure-computer-vision-dengan-blazor.html

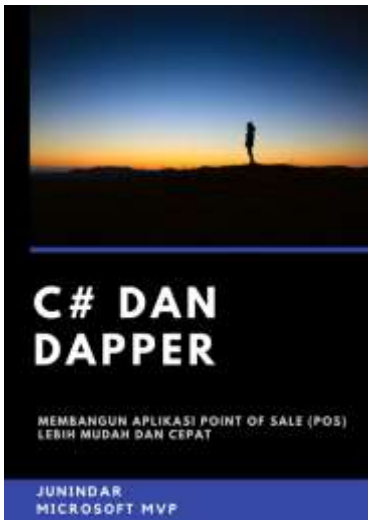
Referensi



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".**