Blazor Simple AI Project

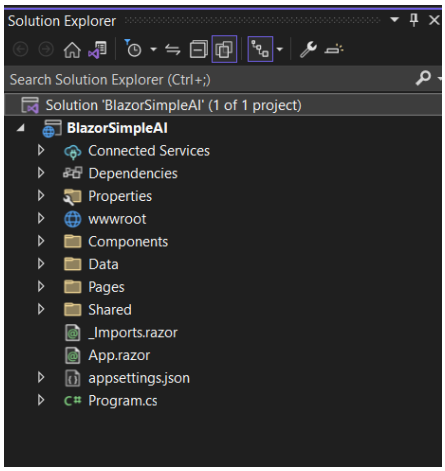
Welcome to the Blazor Simple AI Single Page App, the AI that responds to questions instantly using Microsoft Azure OpenAI Services. This document explains the project in my GitHub repository which is available here: https://github.com/tejinderrai/public/tree/main/BlazorSimpleAI.

Technologies

Blazor Simple AI is made up of the following technologies:

- Microsoft .NET Blazor (.NET 6.0 LTS release)
- Microsoft Azure.AI.OpenAI .NET Library
- Microsoft Azure AI Services – OpenAI

It's that simple!



Why Blazor?

Blazor is simply amazing, and I have been developing Blazor projects for over four years. There has been great demand for Blazor over the past few years and as a component framework and use of C# this is exactly what I need to develop solutions and concepts super-fast!

What Blazor Simple AI Does?

Blazor Simple AI is a Blazor server-side single page app which has a single page and a single component. The razor page has two basic user interface controls, a textbox and a submit button for a user to enter the question for Azure OpenAI. The component "AzureOpenAIChat.razor", has a single parameter which receives the question from the main index page. When the parameter is received by the child component, the component has OnParametersSetAsync() method which then retrieves the appsettings.json values in relation to the Azure OpenAI service AI endpoint, Azure OpenAI key and the deployment name which has the associated model, which was deployed with Azure AI Studio, then send the text to the Azure OpenAI service and retrieves and displays the response.

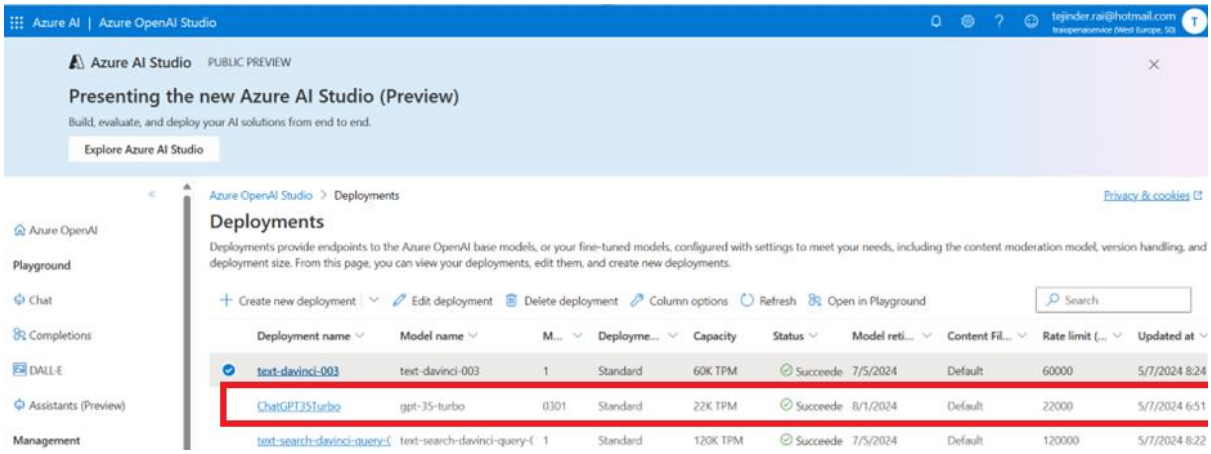
Core Blazor Template Changes

There have been some basic changes to the basic Blazor layout to accommodate the project. These are as follows:

1) The sidebar has been removed from the MainLayout.razor page
2) A new Index.razor.css style sheet has been added to centre the UI components on the page
3) A new Components folder has been added to the project
4) A new component named AzureOpenAIChat.razor has been added into the Components folder
5) A new configuration section has been added to appsettings.json to include the configuration required for the project to interact with the Azure OpenAI service
6) The title and main element have had text changes to represent the project name and description

Steps to Deploy Azure Open AI

1) Create an Azure Resource Group
2) Deploy the Azure OpenAI service in the resource group, see: How-to: Create and deploy an Azure OpenAI Service resource - Azure OpenAI | Microsoft Learn
3) Manage Deployments in Azure AI Studio and create a deployment using the gpt-35-turbo model



4) Update the appsettings.json with the settings

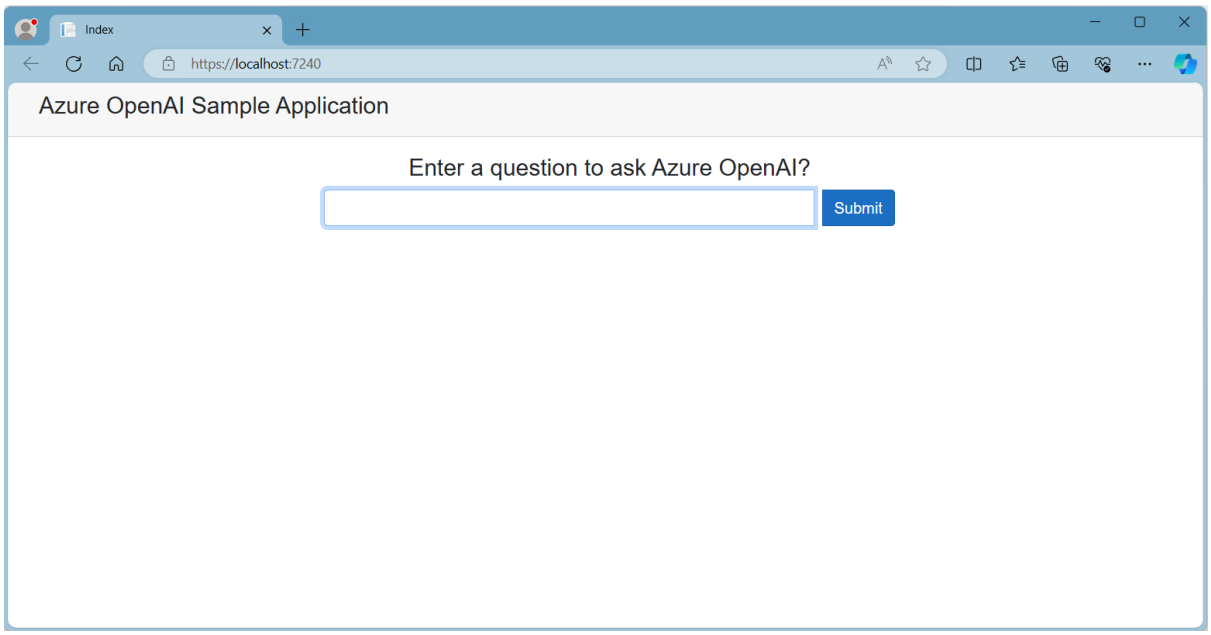
```
"AzureAIConfig": {
    "OpenAIEndpoint": "https://[You Azure OpenAI Service].openai.azure.com/",
    "OpenAIKeyCredential": "[Your Azure Open AI Key]",
    "OpenAIDeploymentName": "[Your Azure Open AI Deployment Name]"
     "RetroResponse": "true or false"

    }
```

5) Build the project and ask Azure OpenAI anything you like.

The UI

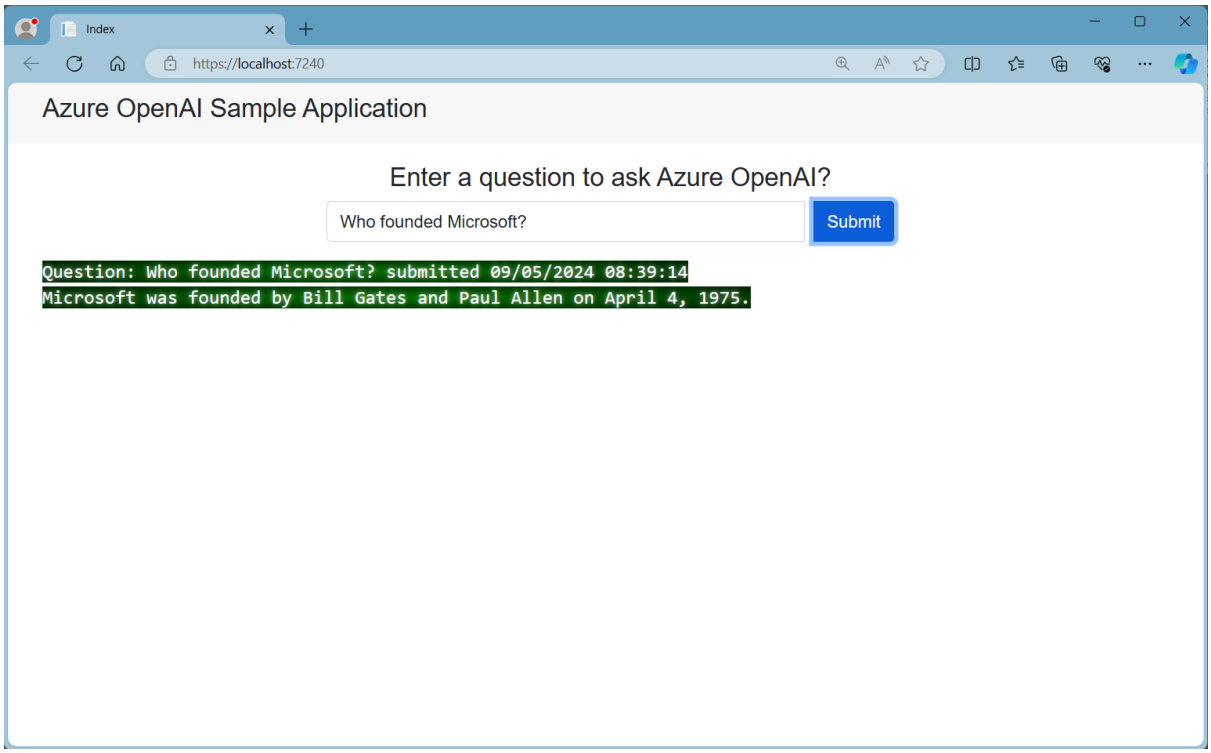
The landing page.



Sample Questions and Responses

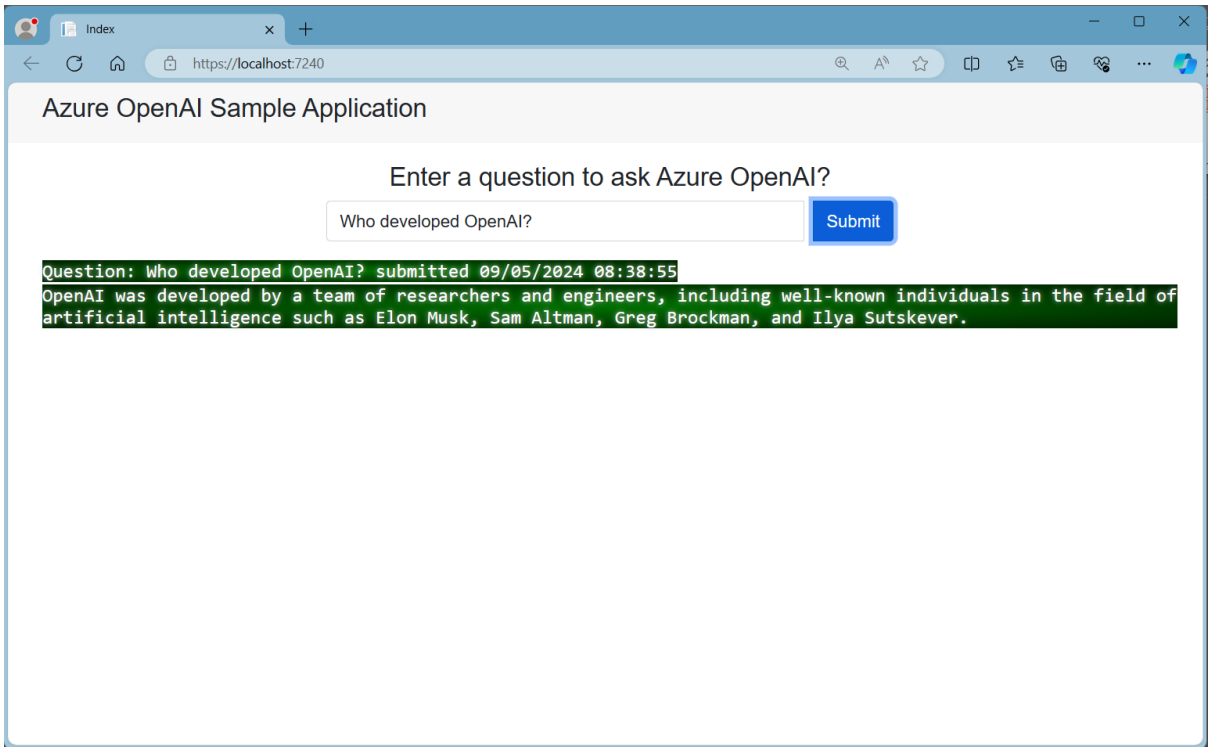
Question 1

Who founded Microsoft?



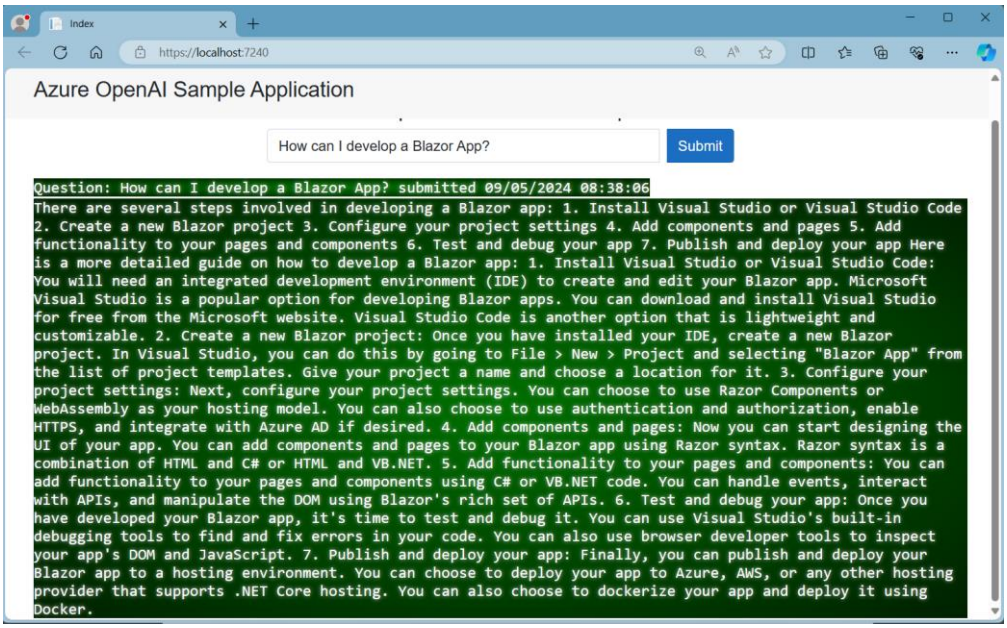
Question 2

Who developed OpenAI?



Question 3

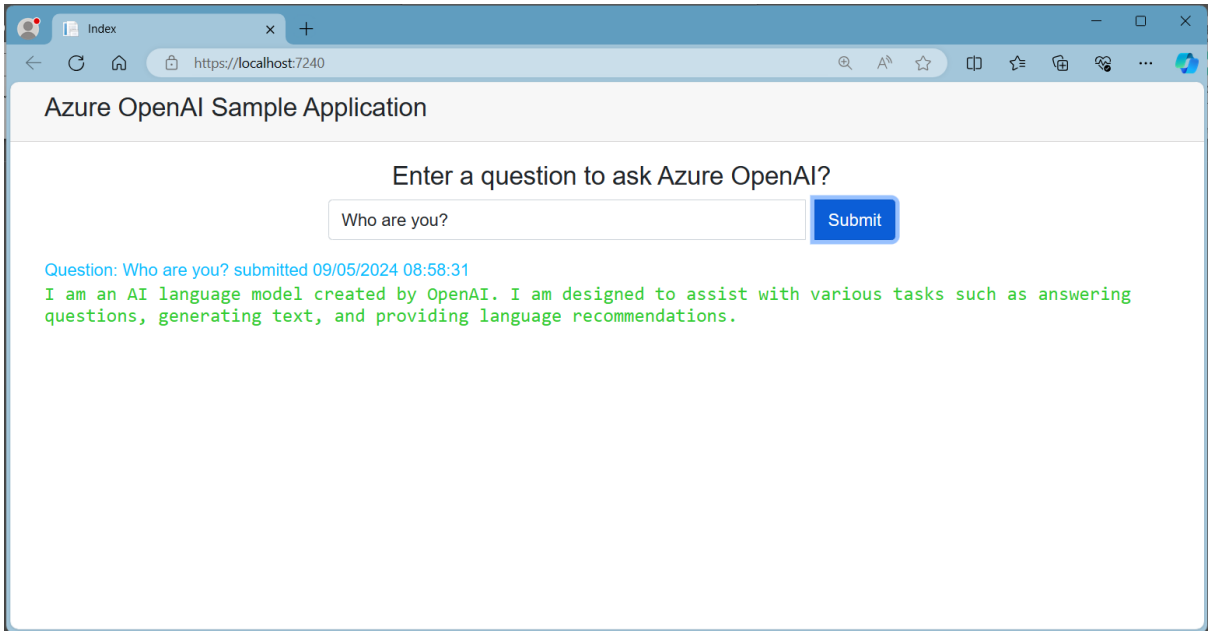
How can I develop a Blazor App?



Basic CSS

The AzureOpenAIChat.razor component has a basic CSS style sheet which allows the deployment to have a retro style response or a basic response text visualization option. If the app setting below is set to true, you will get the retro response as per the sample above. For a standard non-retro style response, you can set the value to false, example below.

```
"AzureAIConfig": {
   "RetroResponse": "false"
   }
```



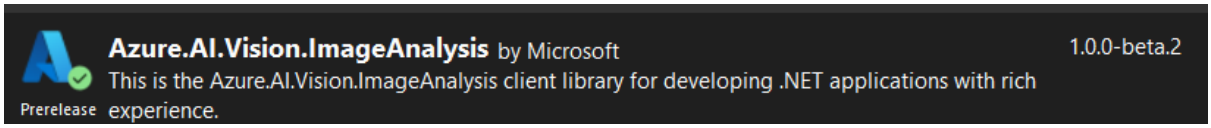
Blazor Simple AI Project (Part 2)

Image Analysis with Azure AI Vision

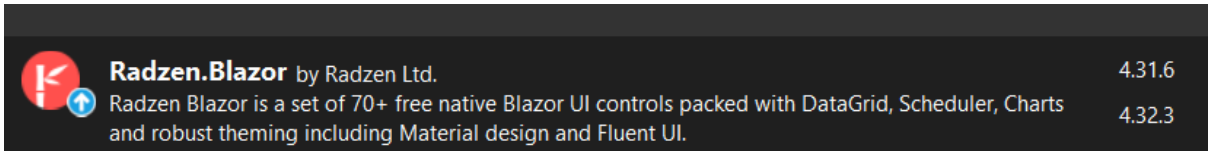
Welcome to the Blazor Simple AI Single Page App, Part 2 of the Microsoft AI services journey, which now includes image analysis utilising Microsoft Azure AI Vision. The Vision Read API is used to extract the text from an image. This document explains the project in my GitHub repository which is available here: https://github.com/tejinderrai/public/tree/main/BlazorSimpleAI.

Since part 1, the following nuget packages have been added to the project.

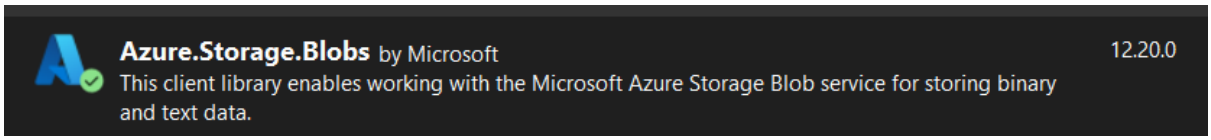
Azure AI Vision Image Analysis – for reading text and metadata from images.



Radzen Blazor – for providing an amazing UI experience.



Azure Storage Blob – for handling interactions with Azure Blob Storage.



Visual Changes

I have made some appealing improvements from the basic Blazor template and styled the UI based on a project from Martin Mogusu available here: GitHub - martinmogusu/blazor-top-navbar: A top navbar example created in blazor. This saved me a lot of time and all I had to do was apply my own visual styles after the top navigation was applied to the project in shared/NavMenu.razor. In addition, I had added a pre-built model for interactive Invoice Analysis and processing, which I will leave the full explanation until Part 3 of this post.

Components

Three components have been developed for the image analysis. These are as follows:

1) Vision.razor – The Image Analysis page
2) VisionBlobLoader.razor– This includes the capability to upload files to Azure blob storage, which also sets the content type for the blob file.
3) VisionBlobFileList.razor – This is a child component embedded into the VisionBlobLoader component, which lists the image files that have been uploaded to Azure blob storage.

Learn about Microsoft AI Vision

To learn more about the capabilities of Microsoft AI Vision, see What is Azure AI Vision? - Azure AI services | Microsoft Learn. Azure AI Vision includes more analysis capabilities, not just specifically image files.

Configuration Settings Changes

The following configuration settings were added to appsettings.json.

```
  "AzureVsionConfig": {
    "AzureAIVisionEndpoint": "https://[Your AI Vision
Service].cognitiveservices.azure.com/",
    "AzureAIVisionKeyCredential": "[AI Vision Service Key]"
  },

  "AzureStorageConfig": {
    "AzureStorageConnectionString": "[Your Storage Account Connection String",
    "AzureStorageContainer": "[Your Storage Account Container]",
    "AzureStorageAccountName": "[Your Storage Account Name]",
    "AzureStorageAccountKey": "Your Storage Account Key"
  },
```

Note: Whist this project utilises the service key, in an enterprise environment, you must consider using token based access to the service secured by Microsoft Entra ID, or if you wish to utilise the service key for any reason, utilise Azure Key Vault to protect the key used by the application with a managed identity for the application to access the service key stored in Azure Key Vault.

Components

File Upload Component (VisionBlobLoader)

The file upload component utilises Blazor InputFile for the user to select the file to upload in the application. The component reads the Azure Storage connection string from the configuration, including the container, then uploads the file to the container and also adds a blob http header for the file content type taken from the file properties. The Radzen notification service is used to notify the user of the application activities. I also included a basic spinner as part of the interaction for the upload process.

Blob List Component (VisionBlobFileList.razor)

This component reads the Azure Storage connection string from the configuration, including the container, then displays the blob file names in a Radzen DataGrid. A button is added to Analyse the image, which then calls the Radzen notification service to display the activities being taken by the application.

Data Classes

Two data classes have been created as follows:

- AzureBlobFile.cs – Azure blob file properties
- ImageDetails.cs – Image details for extraction from the AI Vision Analysis

The UI

The UI is as follows. Notice the menu control has now changed since Part 1. Invoice Analysis will be formed in Part 3, at the time of writing this blog post, I had already uploaded the code to my GitHub repo.

Home page (Chat)

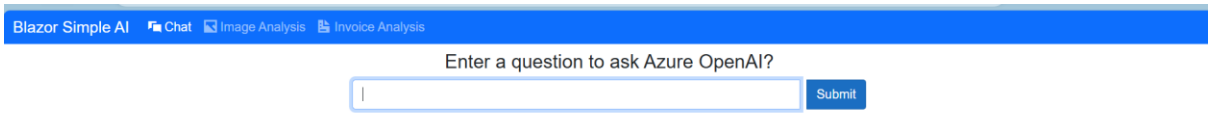
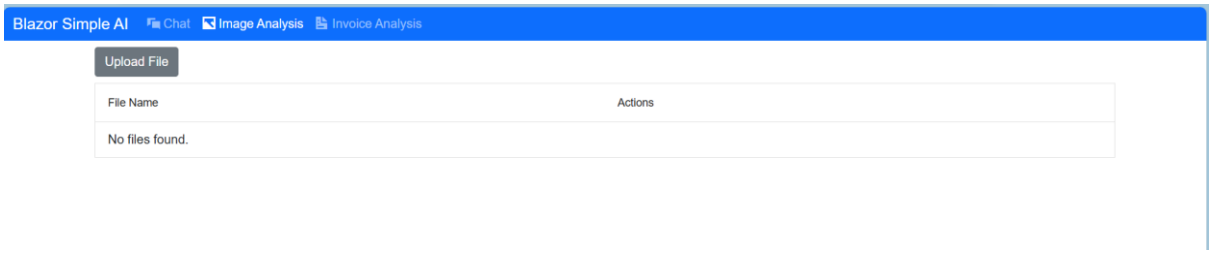
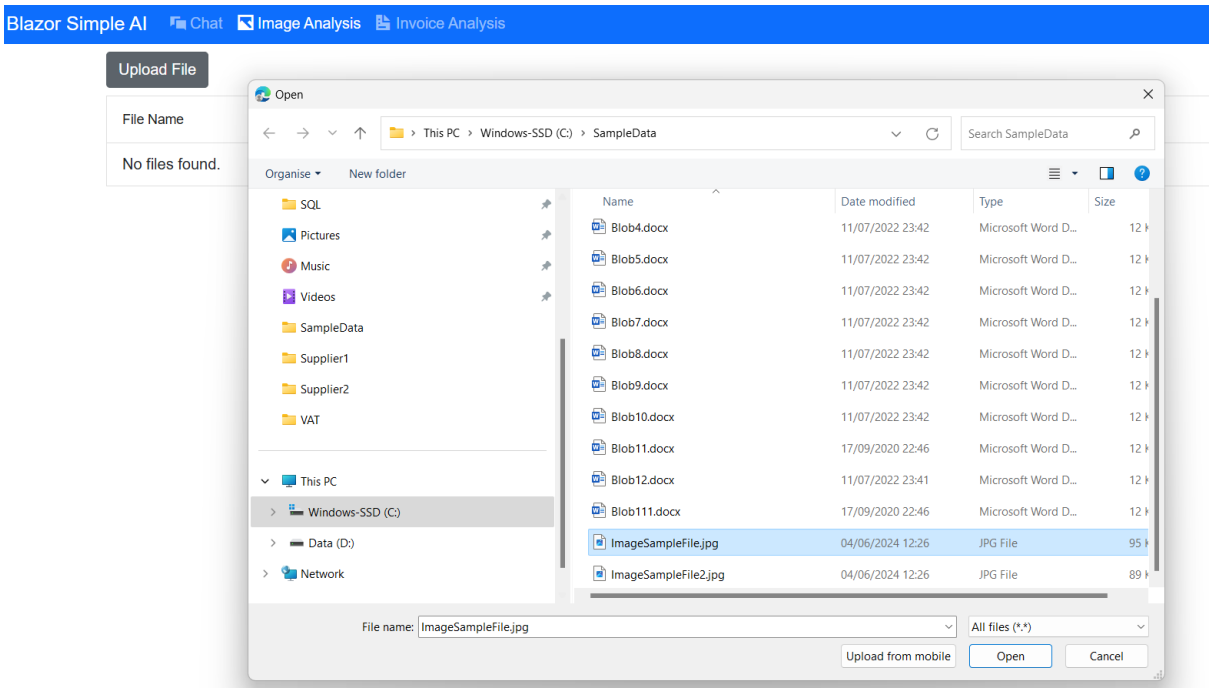


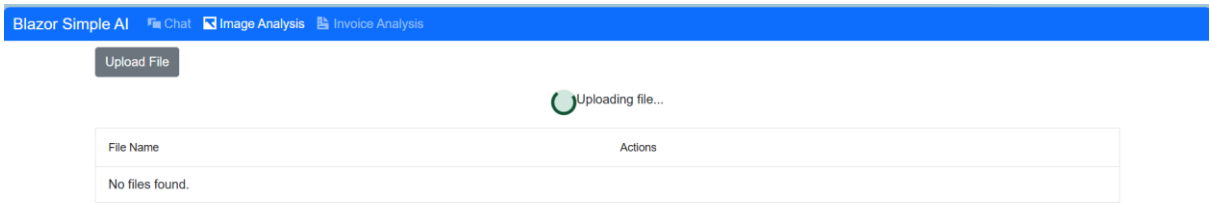
Image Analysis



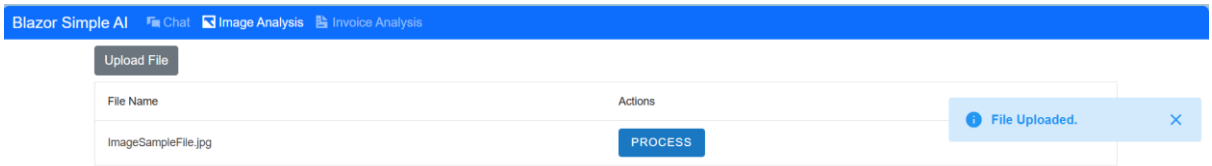
Upload File Control



Upload Action Spinner



Radzen Blazor File Uploaded Notification

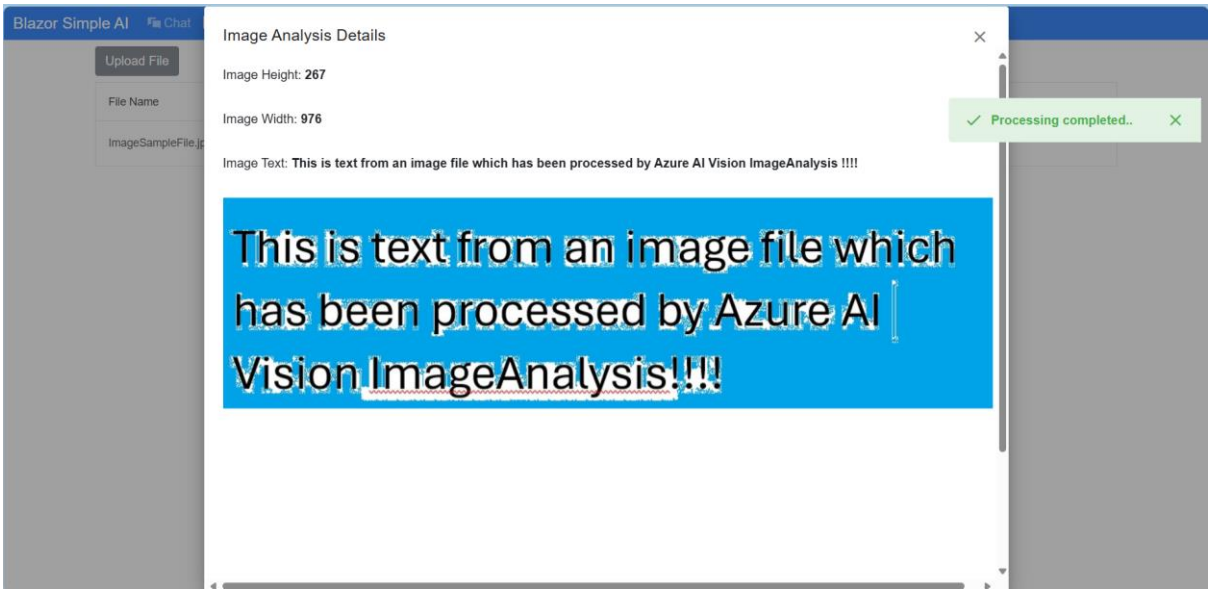


Process Button

The process button read the application configuration for the Azure AI Vision endpoint and service key, then retrieves a SAS token from Azure for the blob being processed and a URL is generated with the generated SAS token, then this is submitted to Azure AI Vision with the generated URL. The SAS token is generated by the async method `CreateServiceSASBlob(string BlobName)` in the component class. Whilst the method can be defined as a utility class, I have composed this for easier reading of code.

Image Analysis Dialog

When the image processing has completed, a Radzen notification is displayed to the user, with a Radzen dialog popping up to show basic metadata (height and width) of the image, including the text the AI Vision service has extracted as well as the image itself.



That is AI Vision and Image Analysis wrapped up. Part 3 will focus on processing invoices using the pre-built AI model “prebuilt-invoice” part of Microsoft Azure AI Document Intelligence.

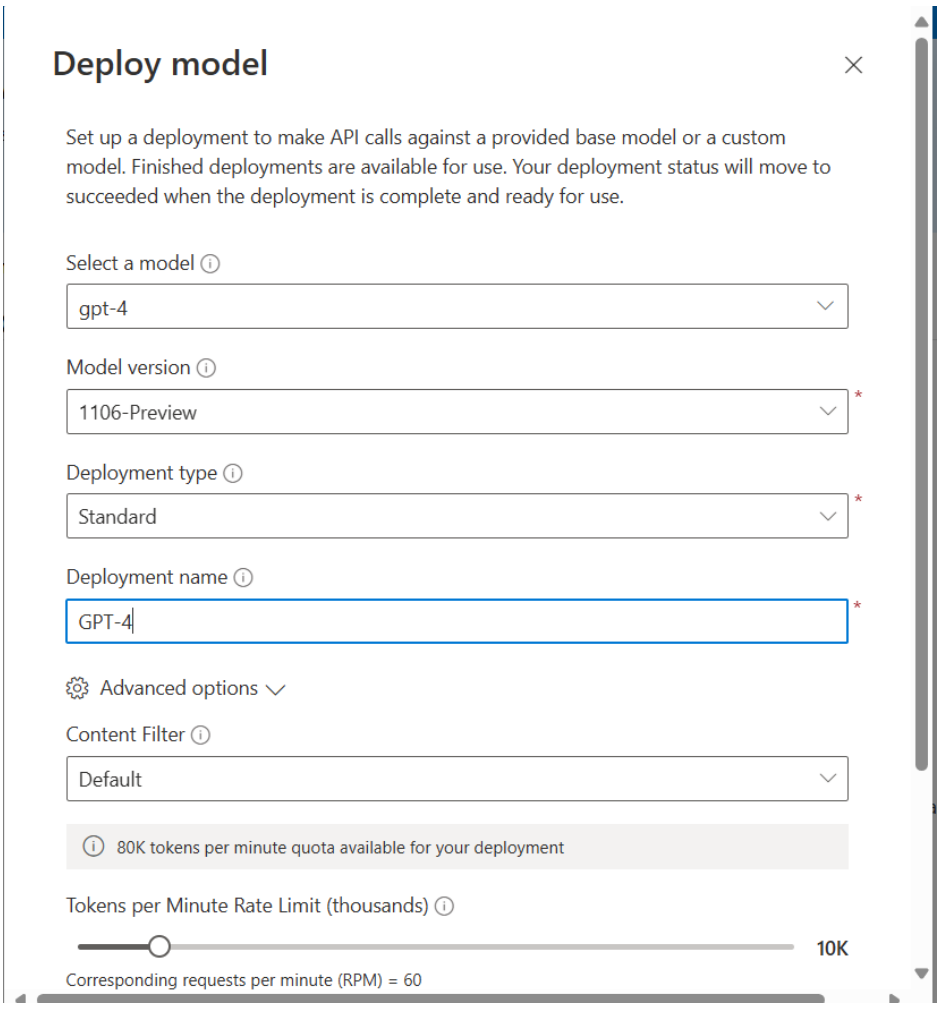
Blazor Simple AI Project (Part 3)

Upgrading to GPT-4

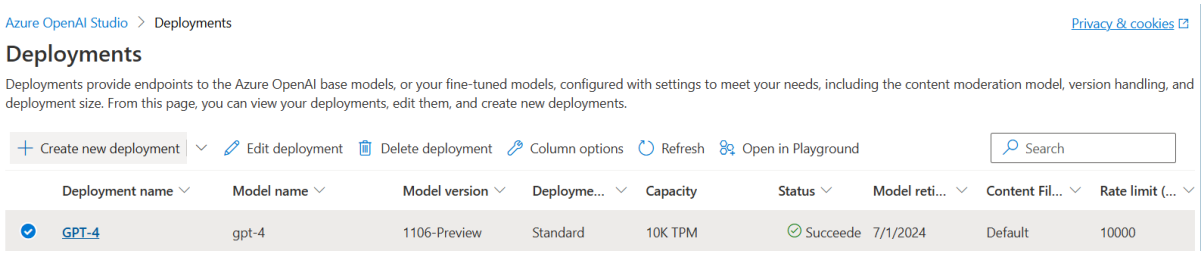
Many models in the Azure Open AI service are being deprecated on June 14th 2024. All Microsoft Azure Open AI service model retirement dates can be found on Microsoft learn here. It's time to deploy GPT-4 to Blazor Simple AI and make the minor changes in appsettings.json to utilise the a deployment based on GPT-4. Follow the steps below.

Deploy a new Model with Azure AI Studio

1) Launch and authenticate to AI Studio https://oai.azure.com/
2) Click **Deployments**
3) Click **Deploy a new model**
4) Set the model version, deployment type, I have chosen standard, enter the name of the deployment and the number of required tokens minute and click **Create**.



Your model will be deployed.



Update the configuration settings in the application

In the configuration section below, update the Open AI deployment name setting, in my case the deployment name I had chosen is “GPT-4”.

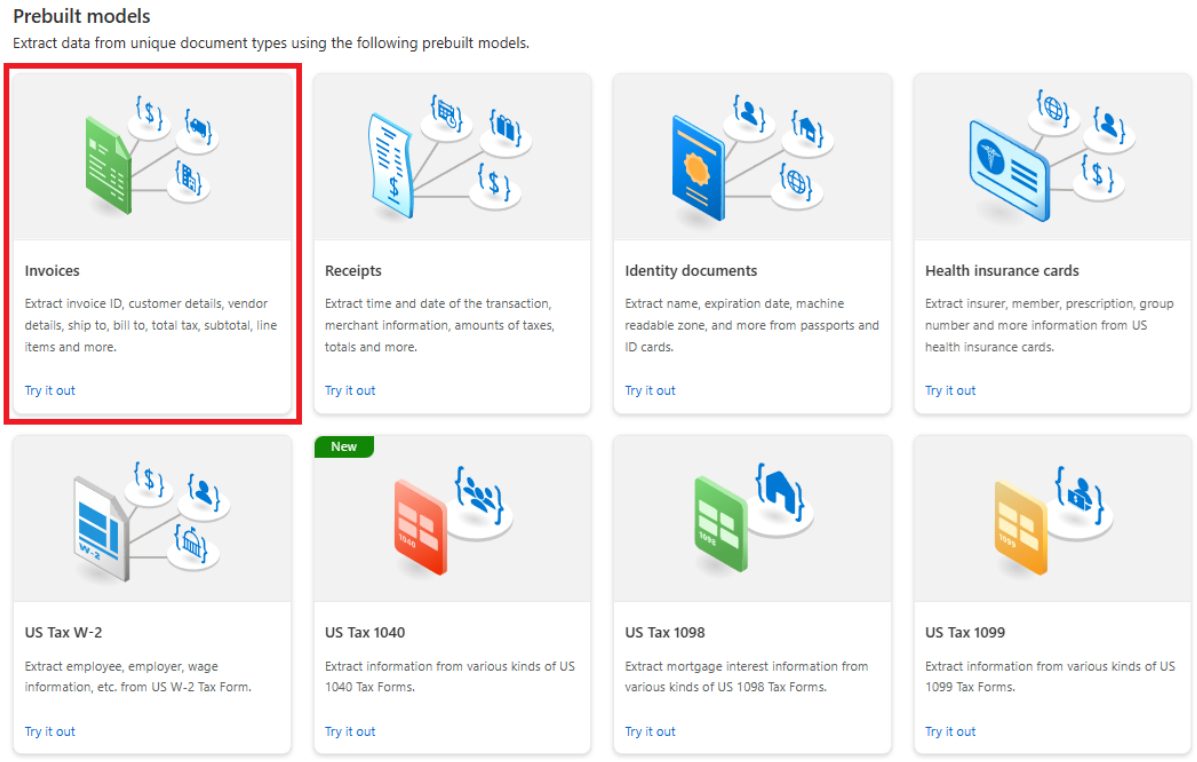
```
"AzureAIConfig": {
      "OpenAIDeploymentName": "GPT-4",
}
```

That’s all you need to do!

Blazor Simple AI Project (Part 4)

Invoice Analysis

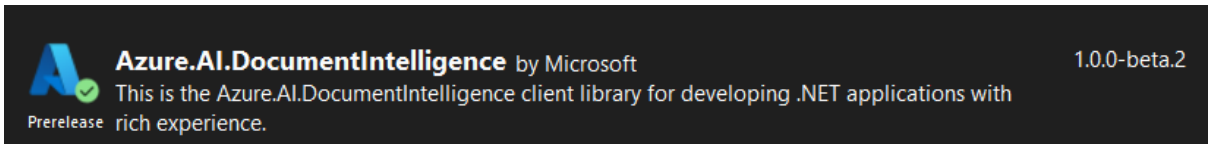
Welcome to the Blazor Simple AI Single Page App, Part 4 of the Microsoft AI services journey, which now includes invoice analysis which utilises Microsoft Azure AI Document Intelligence service. The document Intelligence service is used to extract the text from an invoice using a pre-built model. A sample of some of the models is shown below in document intelligence studio.



This document explains the project in my GitHub repository which is available here: https://github.com/tejinderrai/public/tree/main/BlazorSimpleAI.

Since part 3, the following nuget packages have been added to the project.

Azure.AI.DocumentIntelligence (Pre-release)



New Components

Three components have been developed for the image analysis. These are as follows:

1) InvoiceLoader.Razor – A component which includes the child component (InvoiceFileList.Razor), which uploads invoices to Azure blog storage container

2) InvoiceFileList.Razor – A component which lists the invoices that a present in the invoice upload container
3) InvoiceViewer.Razor – A component which allows the user to view the uploaded invoice in a dialog

Provisioning a Microsoft AI Document Intelligence Service

To provision a Microsoft AI Document Intelligence Service resource, follow the instructions in the article below.

Create a Document Intelligence (formerly Form Recognizer) resource - Azure AI services | Microsoft Learn

Learn about Microsoft AI Document Intelligence

To learn more about the capabilities of Microsoft AI Document Intelligence capabilities, see. Document Intelligence documentation - Quickstarts, Tutorials, API Reference - Azure AI services | Microsoft Learn. Microsoft Azure AI Document Intelligence includes more analysis capabilities, not just specifically an invoice model.

Configuration Settings Changes

The following configuration settings were added to appsettings.json.

```
  "AzureDocumentIntelligenceConfig": {
    "AzureDocumentIntelligenceKey": "[Your Azure Document Intelligence Key]",
    "AzureDocumentIntelligenceEndpoint": "[Your document intelligence endpoint
https://[Resource Name].cognitiveservices.azure.com/"
  },
"AzureStorageConfig": {
    "AzureStorageInvoiceContainer": "[Your Invoice Upload Container Name]",
    "AzureStorageInvoiceProcessedContainer": "[Your Invoice Processed Container
Name]",
  },
```

The invoice processed container is the output interface file that is generated from the text extracted from the original invoice file. It is an output of JSON which utilises the `InvoiceAnalysisData data` type.

Note: Whist this project utilises the service key, in an enterprise environment, you must consider using token based access to the service secured by Microsoft Entra ID, or if you wish to utilise the service key for any reason, utilise Azure Key Vault to protect the key used by the application with a managed identity for the application to access the service key stored in Azure Key Vault.

Components

Invoice Loader Component (InvoiceLoader.Razor)

The invoice upload component utilises Blazor InputFile for the user to select the file to upload in the application. The component reads the Azure Storage connection string from the configuration, including the container, then uploads the file to the container and also adds a blob http header for the file content type taken from the file properties. The Radzen notification service is used to notify the user of the application activities. I also included a basic spinner as part of the interaction for the upload process.

Invoice File List Component (InvoiceFileList.Razor)

This component reads the Azure Storage connection string from the configuration, including the container, then displays the invoice blob file names in a Radzen DataGrid. A button is added to view the invoice, or process the invoice, which then calls the Radzen notification service to display the activities being taken by the application.

Invoice Viewer Component (InvoiceViewer.Razor)

This component is a child component displayed in a Radzen dialog box which displays the original uploaded invoice directly from the Azure blob storage invoice upload container. A storage SAS key is generated which provides time limited access to the user in order for the invoice to be displayed in the dialog.

Data Classes

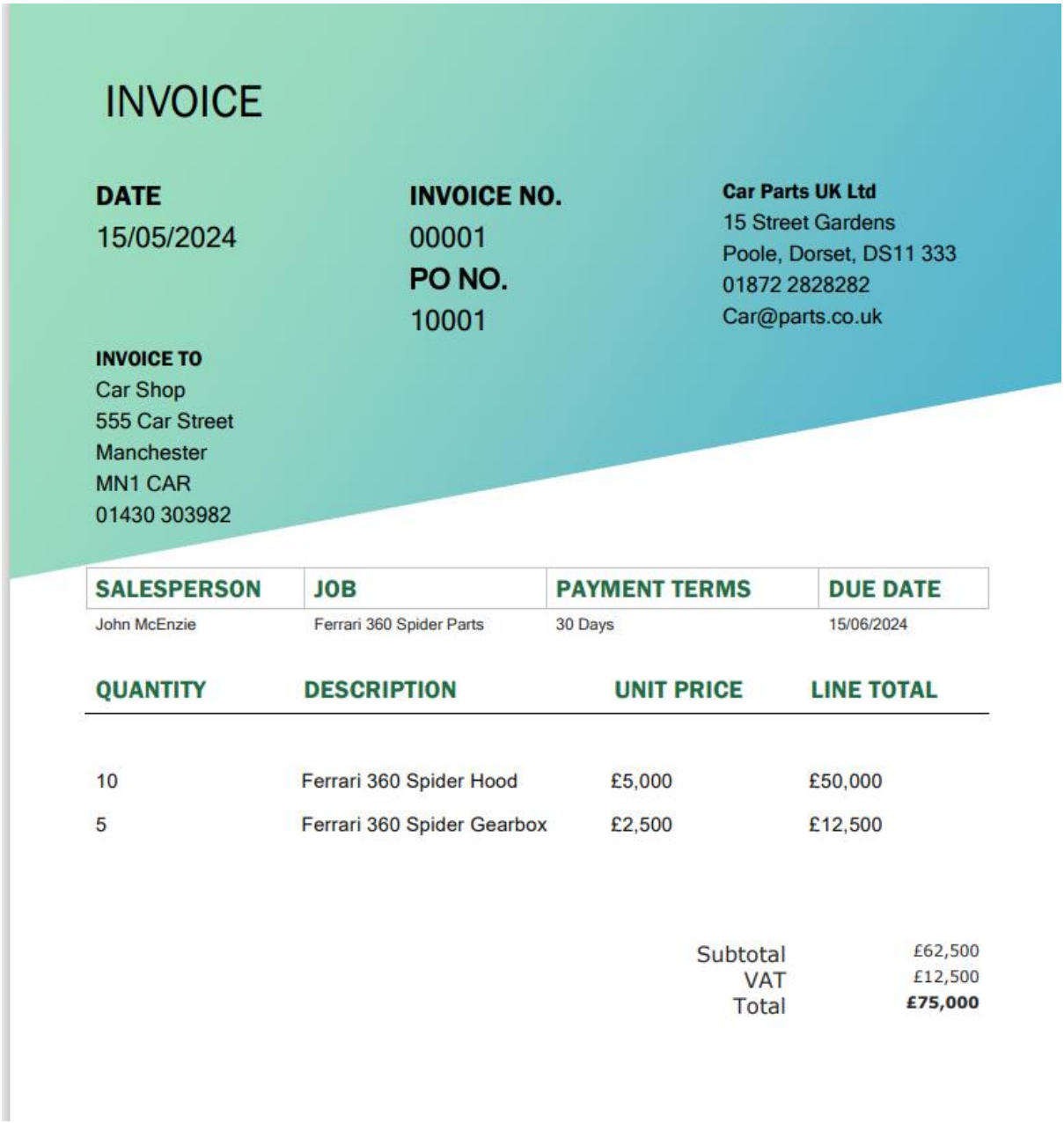
InvoiceAnalysisData.cs – The class for the invoice.

InvoiceItem.cs - The class for the invoice items.

Invoice Sample

I have created an invoice samples to test the pre-built invoice model from Microsoft Azure Document Intelligence.

Supplier 1 Invoice (PDF)



INVOICE

DATE
15/05/2024

INVOICE NO.
00001

PO NO.
10001

Car Parts UK Ltd
15 Street Gardens
Poole, Dorset, DS11 333
01872 2828282
Car@parts.co.uk

INVOICE TO
Car Shop
555 Car Street
Manchester
MN1 CAR
01430 303982

SALESPERSON	JOB	PAYMENT TERMS	DUE DATE
John McEnzie	Ferrari 360 Spider Parts	30 Days	15/06/2024

QUANTITY	DESCRIPTION	UNIT PRICE	LINE TOTAL
10	Ferrari 360 Spider Hood	£5,000	£50,000
5	Ferrari 360 Spider Gearbox	£2,500	£12,500

Subtotal	£62,500
VAT	£12,500
Total	**£75,000**

I created two additional sample invoices, both of which were tested and successfully processed. I have not covered the upload of these in my blog post.

Supplier 2 – Jpeg image

(Missing Quantity)

Porsche Parts Specialist

INVOICE

The best parts, all the time, on time!

911 Porsche Avenue,
Silverstone,
SL11 POS
Phone 04556 282811
porsche@partsspecialist.com

INVOICE NO. 00002
DATE 21/05/2024

TO:

CAR SHOP
555 CAR STREET
MANCHESTER
MN1 CAR
01430 303982

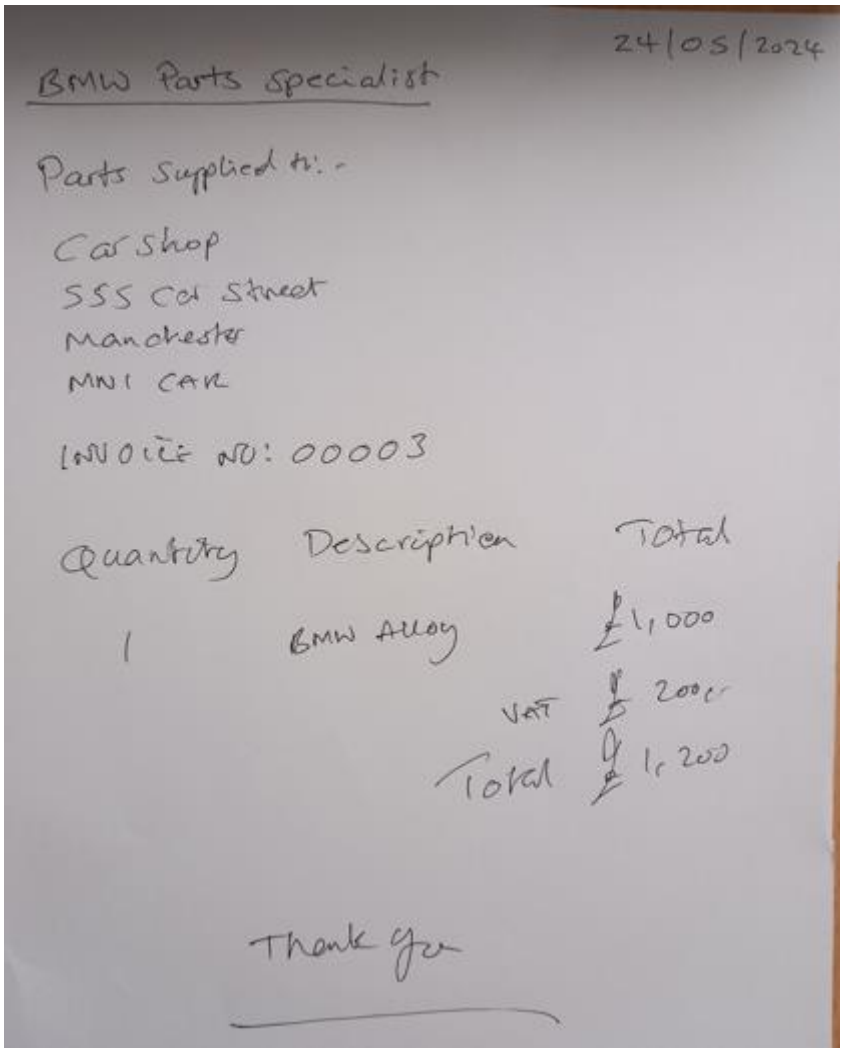
FOR Porsche 911 Refresh Project
PO NO. 10002

Description	Amount
Porsche 911 993 Engine casing	£3,000
Porsche 911 993 IMS Bearing	£2,000
Porsche 911 993 Rear Spoiler	£2,000
Porsche 911 993 Suspension Kit	£8,000
VAT 20%	£3,000
Total	**£18,000**

Make all cheques payable to Porsche Parts Specialist
Payment is due within 30 days.
If you have any questions concerning this invoice, contact Rob | 04456 282812 | rob@porschespecialist.com

THANK YOU FOR YOUR BUSINESS.

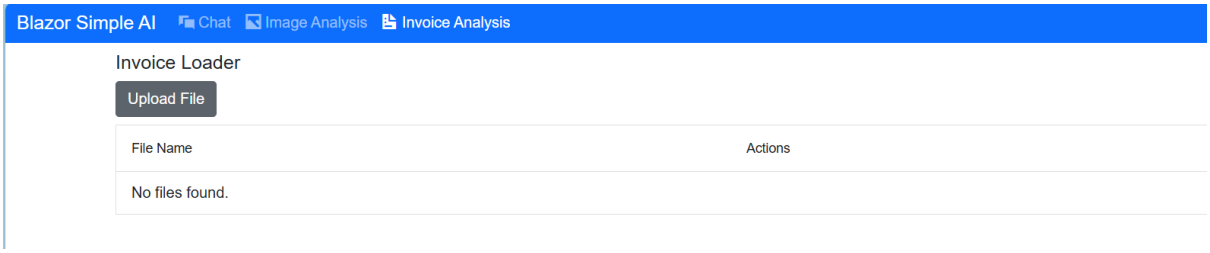
Invoice 3 – Handwritten Invoice – jpeg image



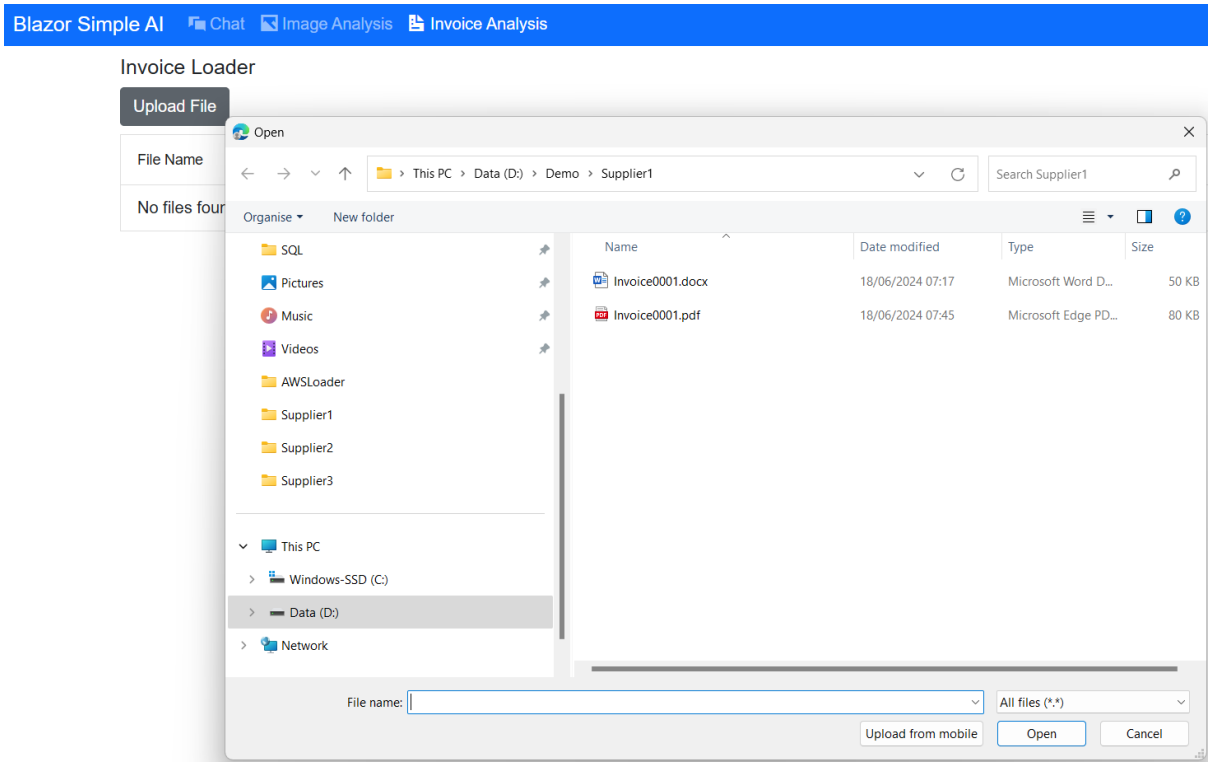
The UI

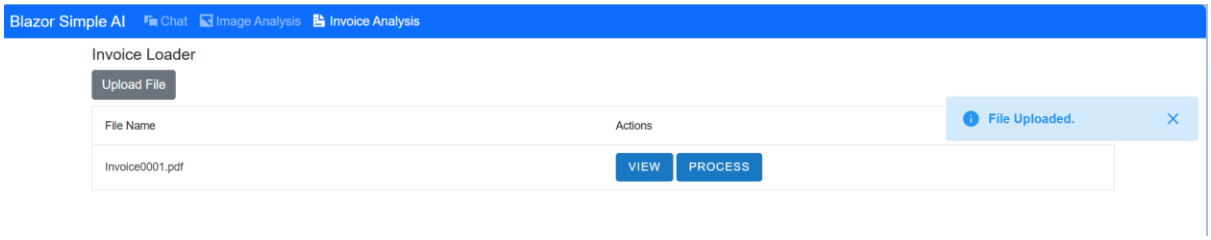
The UI for invoice analysis is as follows.

Invoice Analysis

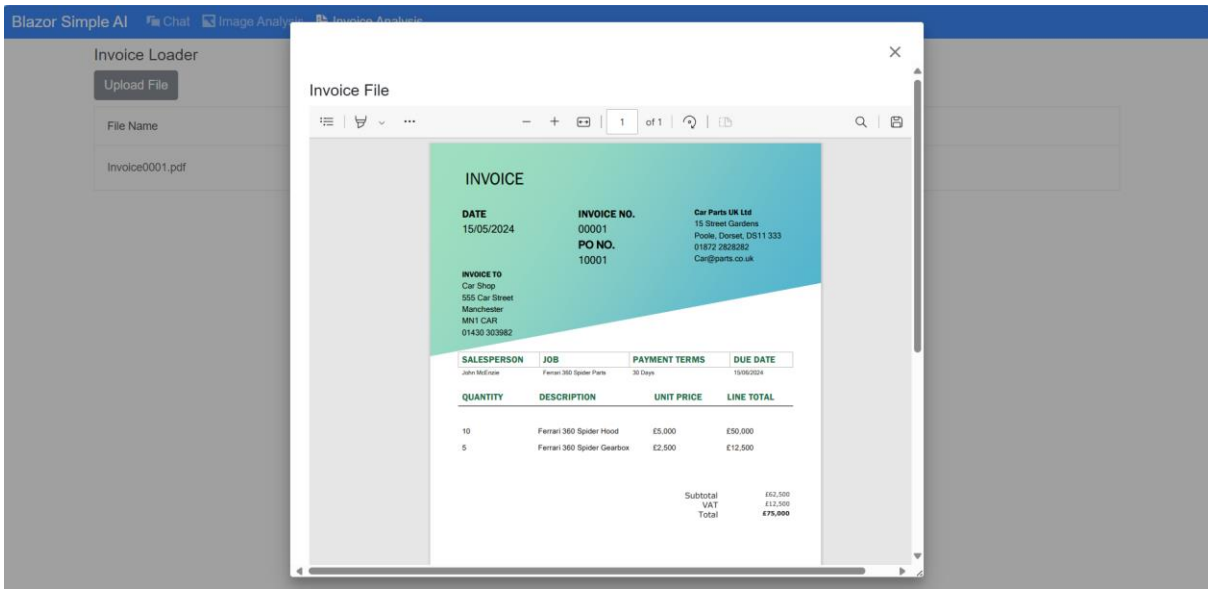


Upload File

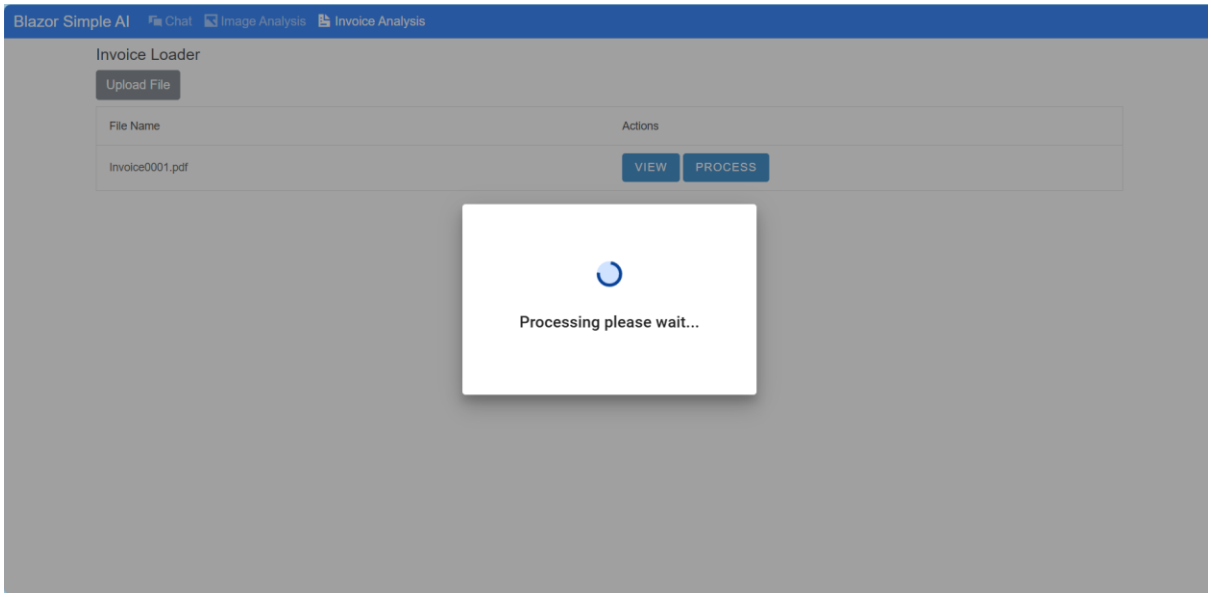




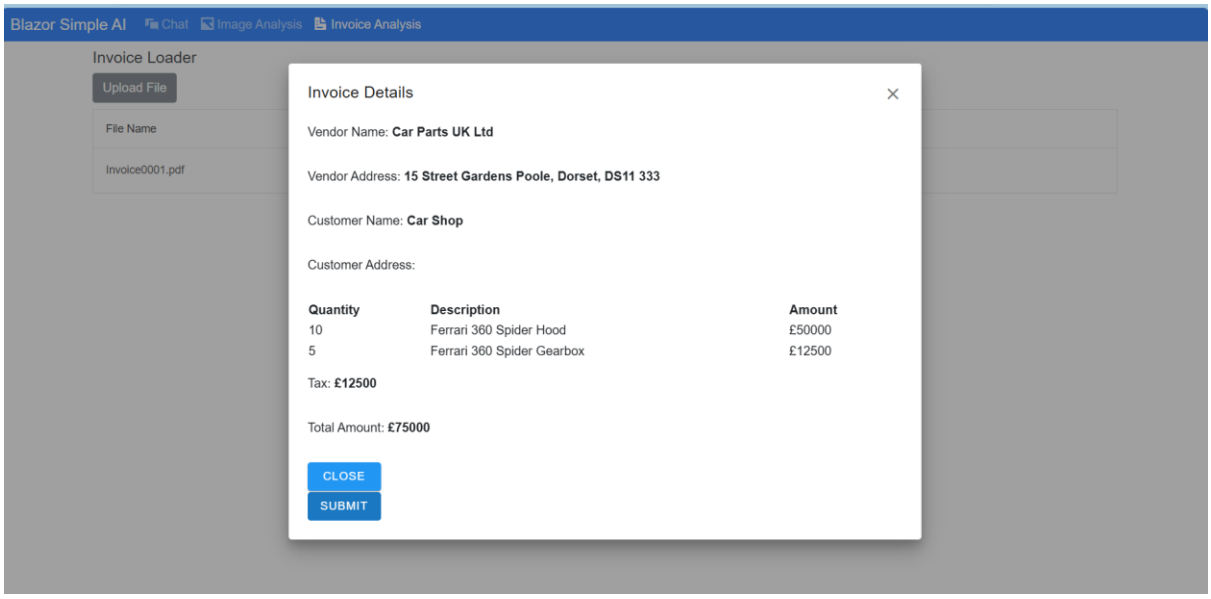
View Button – Opens the PDF in a dialog box



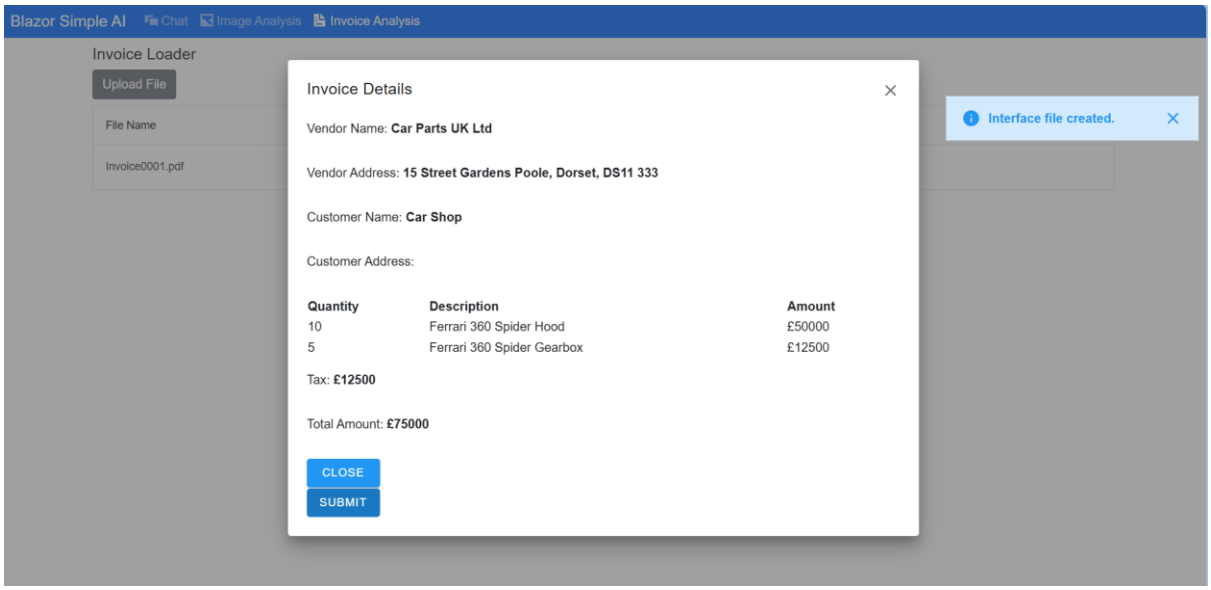
Process Button – Interactive Dialog box



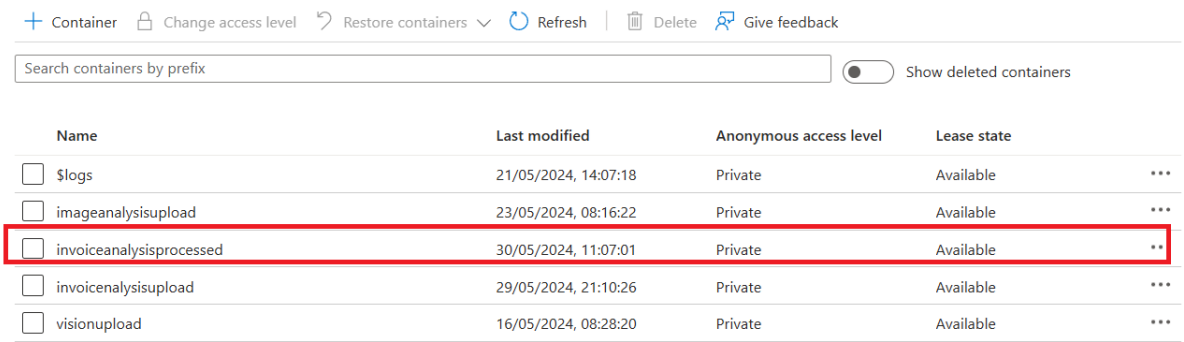
Processing Completed – Invoice details – text extracted into InvoiceAnalysis object.



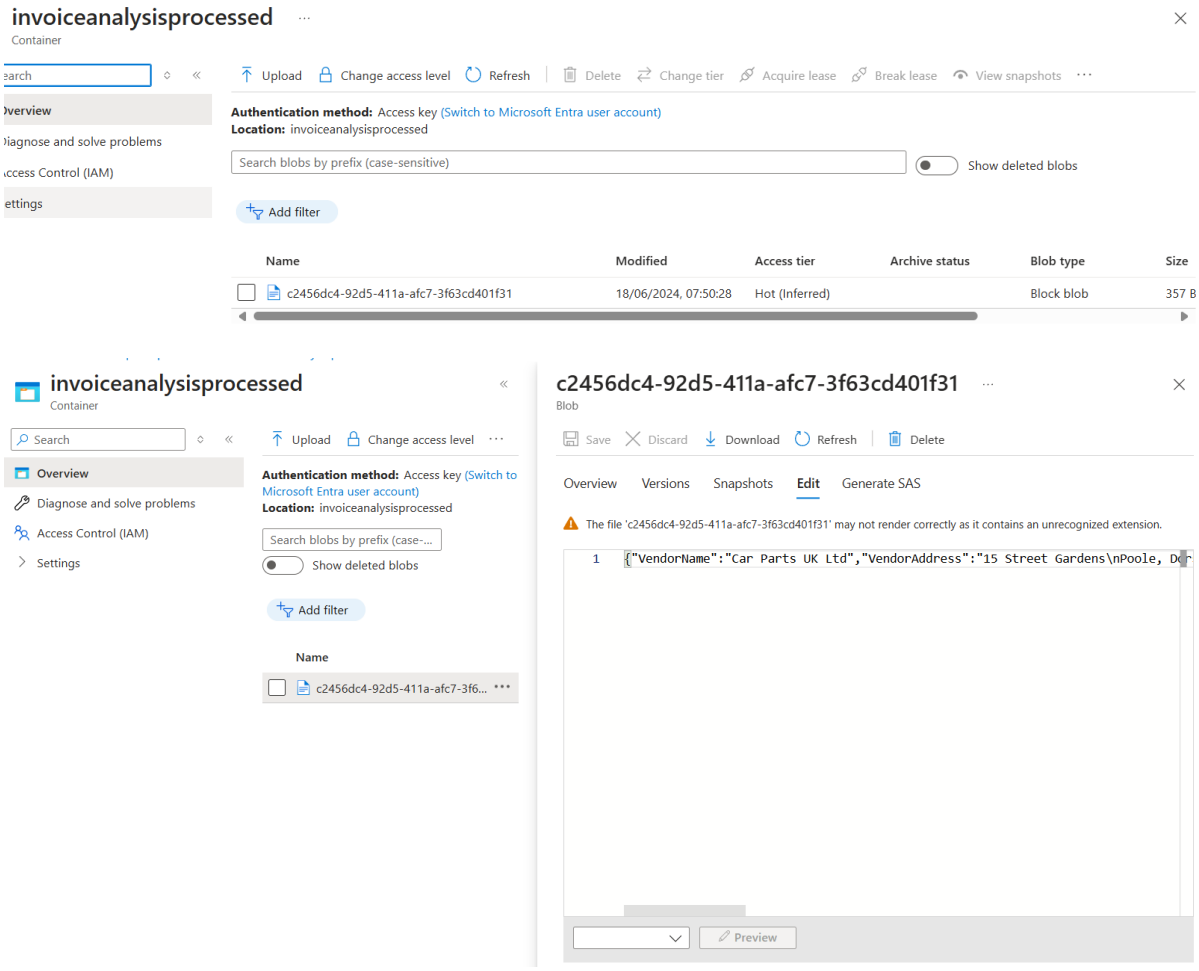
Submit Button – Create an output interface file in JSON format.



Azure Storage – (invoiceanalysisupload container)



Processed Output file



File Contents


```
{
    "VendorName":"Car Parts UK Ltd",
    "VendorAddress":"15 Street Gardens\nPoole, Dorset, DS11 333",
    "CustomerName":"Car Shop",
    "CustomerAddress":null,
    "InvoiceItems":[
        {"Quantity":10.0,"ItemDescription":"Ferrari 360 Spider Hood","Amount":"50000"},
        {"Quantity":5.0,"ItemDescription":"Ferrari 360 Spider Gearbox","Amount":"12500"}
    ],
    "Tax":"12500",
    "InvoiceTotal":"75000"
}
```

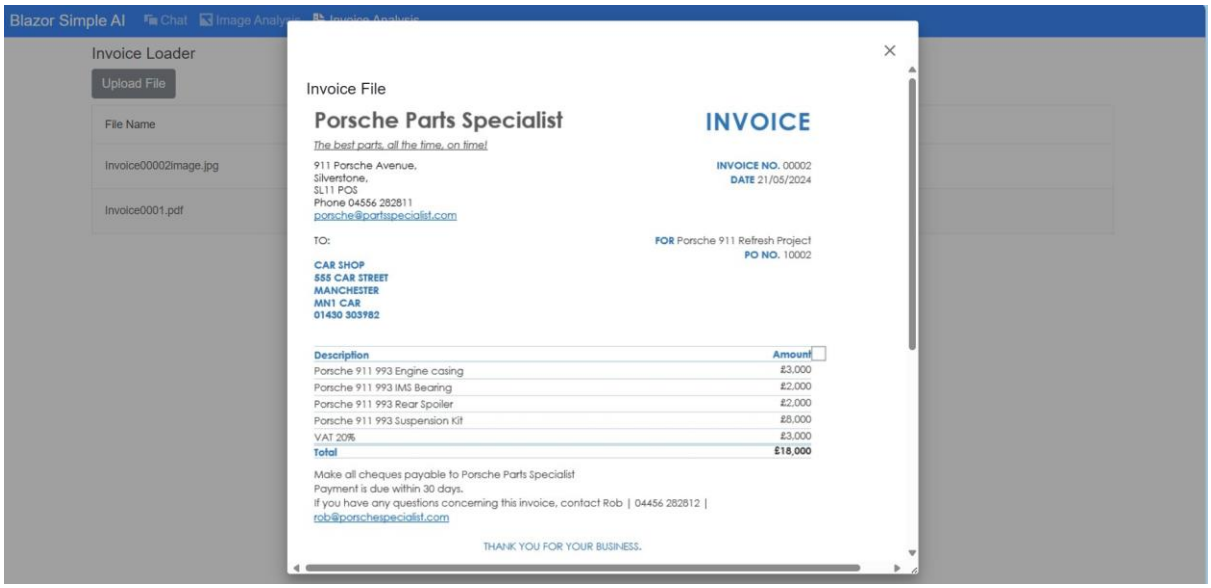

Note: The code does not extract the customer address, but this is in fact possible.

The handwritten jpeg image, the second invoice as a jpeg image and the PDF all proved to have 100% extraction using the Microsoft AI Document Intelligence service. That's just amazing!

It is as simple as that!

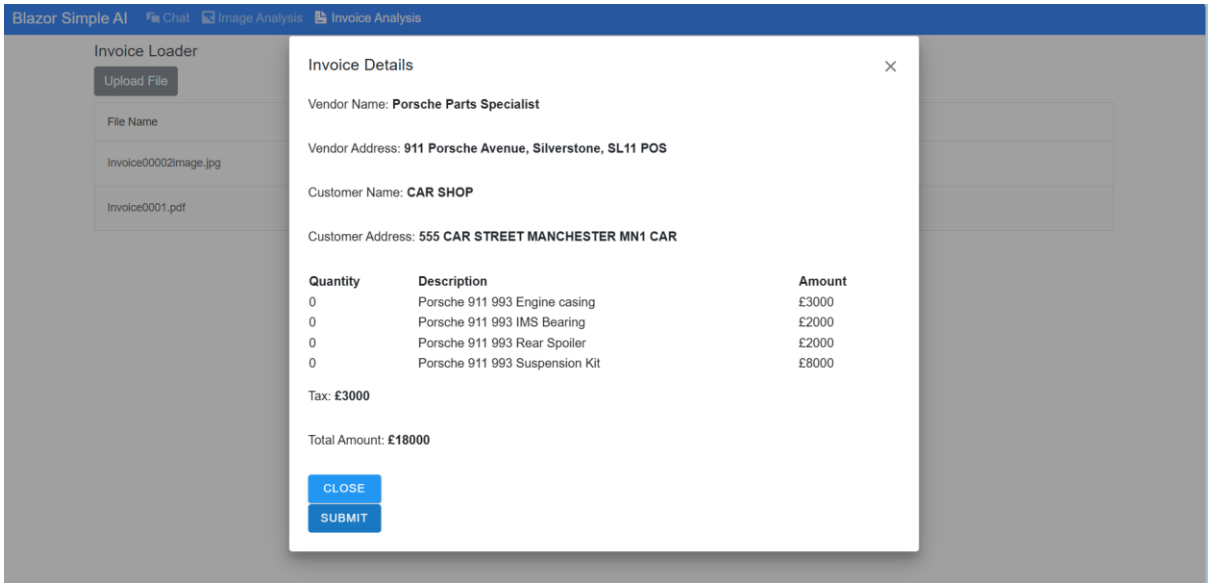
The reason for creating a interactive SPA as a sample app, is to demonstrate the features. The same code can be used in event driven architectures, or scheduled triggers. That will be something I will post next.

Invoice 2 - jpeg output

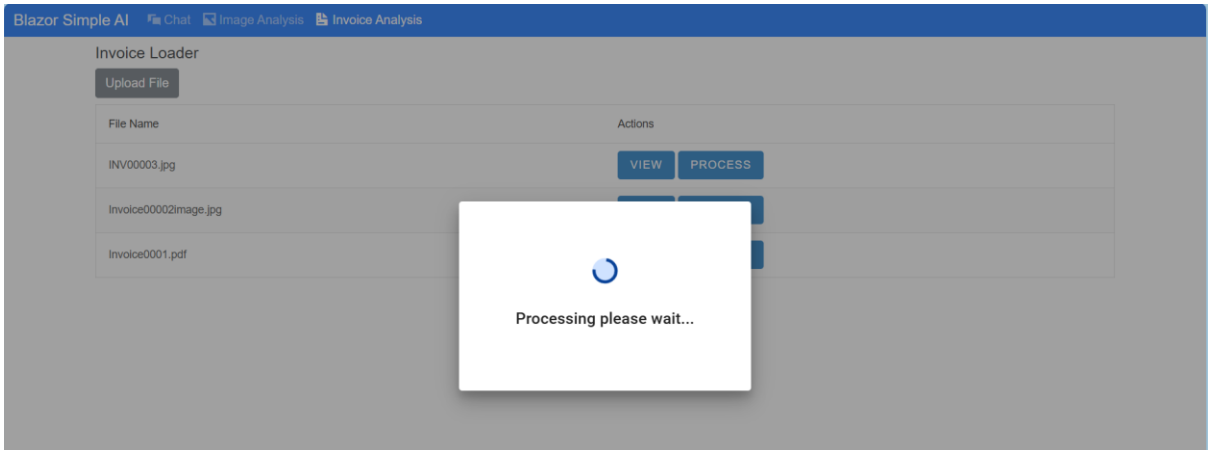


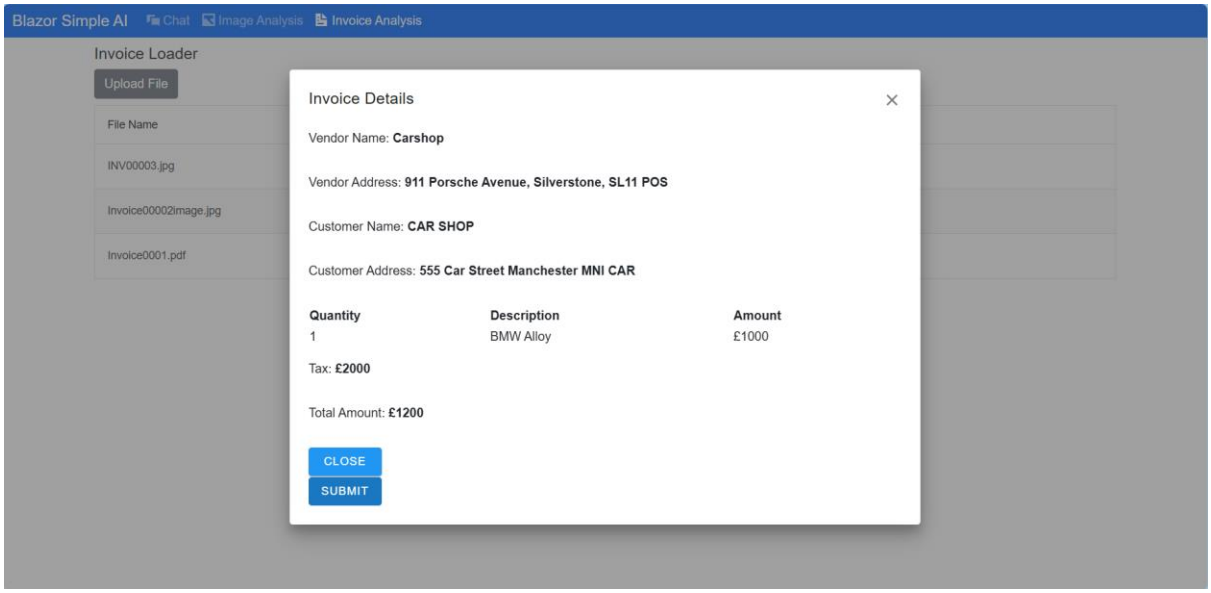
Invoice 2 – Processed

(Note: Missing Quantity in the file)



Invoice 3 – Handwritten jpeg


Blazor Simple AI
Chat
Image Analysis
Invoice Analysis
Invoice Loader
Upload File
File Name
Actions
INV00003.jpg
VIEW
PROCESS
Invoice00002Image.jpg
Invoice0001.pdf
Processing please wait...


Blazor Simple AI
Chat
Image Analysis
Invoice Analysis
Invoice Loader
Upload File
File Name
INV00003.jpg
Invoice00002Image.jpg
Invoice0001.pdf
Invoice Details
Vendor Name: Carshop
Vendor Address: 911 Porsche Avenue, Silverstone, SL11 POS
Customer Name: CAR SHOP
Customer Address: 555 Car Street Manchester MNI CAR
Quantity
Description
Amount
1
BMW Alloy
£1000
Tax: £2000
Total Amount: £1200
CLOSE
SUBMIT