Developed by Tejinder Rai, Microsoft Azure Cloud Solutions Architect blog: https://mswebdeveloper.com

Blazor Simple AI Project

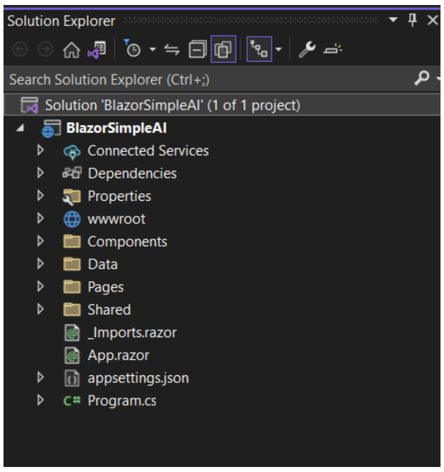
Welcome to the Blazor Simple AI Single Page App, the AI that responds to questions instantly using Microsoft Azure OpenAI Services. This document explains the project in my GitHub repository which is available here: https://github.com/tejinderrai/public/tree/main/BlazorSimpleAI.

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!



```
Solution 'BlazorSimpleAI' (1 of 1 project)
  BlazorSimpleAI
    Connected Services
    Dependencies
    Properties
    wwwroot
    Components
    Data
    Pages
    Shared
    _Imports.razor
    App.razor
    appsettings.json
    Program.cs
```

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.

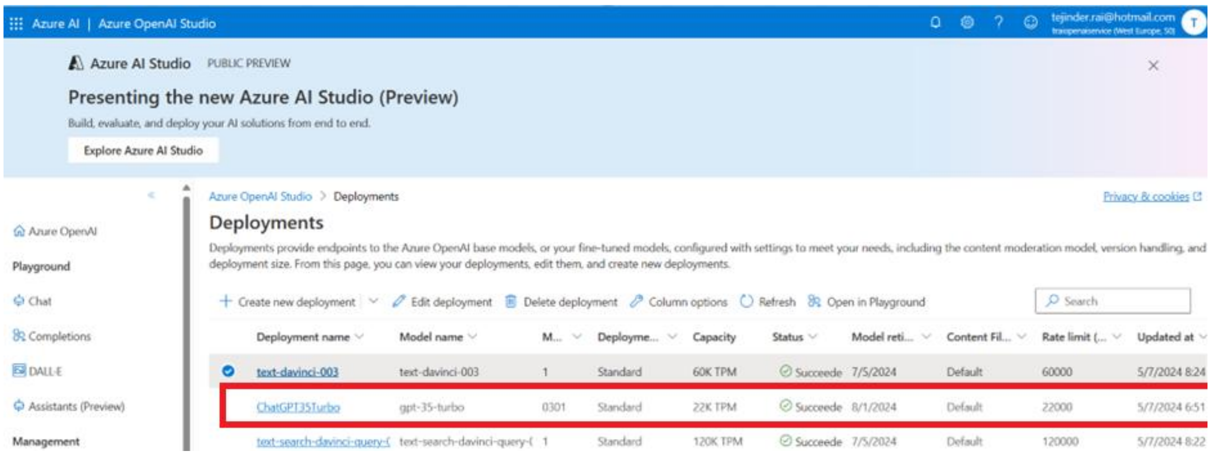
Core Blazor Template Changes

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



Deployments

Deployments provide endpoints to the Azure OpenAI base models, or your fine-tuned models, configured with settings to meet your needs, including the content moderation model, version handling, and deployment size. From this page, you can view your deployments, edit them, and create new deployments.

Deployment name	Model name	M...	Deployme...	Capacity	Status	Model reti...	Content Fil...	Rate limit (...	Updated at
text-davinci-003	text-davinci-003	1	Standard	60K TPM	Succeede	7/5/2024	Default	60000	5/7/2024 8:24
ChatGPT35Turbo	gpt-35-turbo	0301	Standard	22K TPM	Succeede	8/1/2024	Default	22000	5/7/2024 6:51
text-search-davinci-query-(	text-search-davinci-query-(	1	Standard	120K TPM	Succeede	7/5/2024	Default	120000	5/7/2024 8:22

4) Update the appsettings.json with the settings

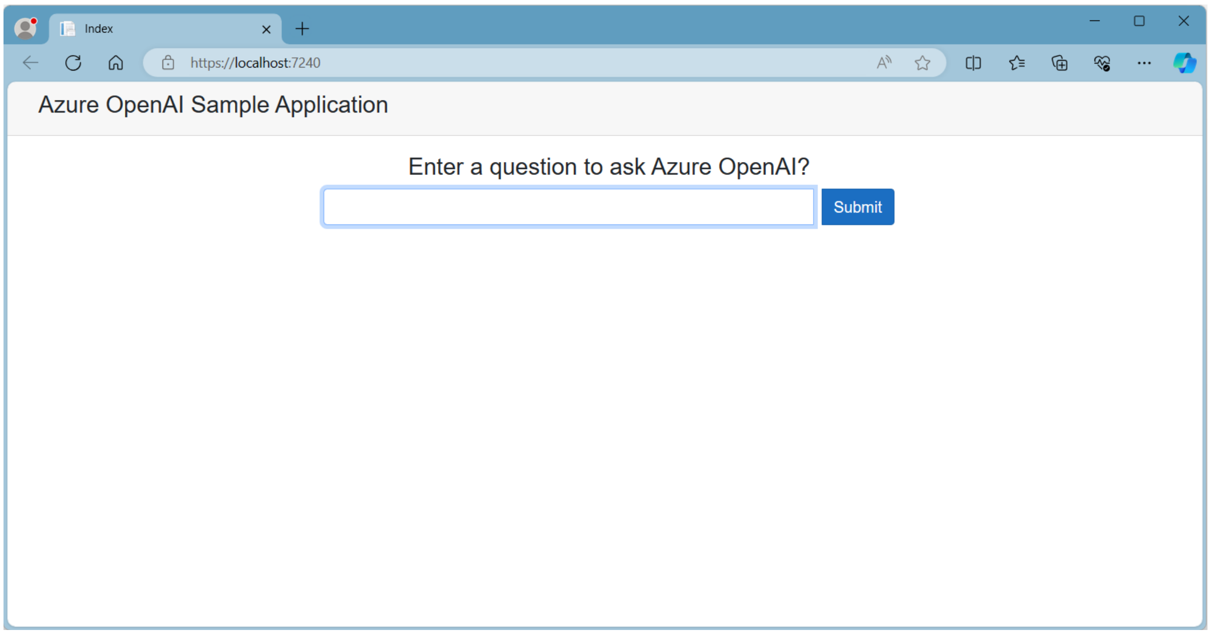
```
"AzureAIConfig": {
    "OpenAIEndpoint": "https://[You Azure OpenAI Service].openai.azure.com/",
    "OpenAIKeyCredential": "[Your Azure Open AI Key]",
    "OpenAIDeploymentName": "[Your Azure Open AI Deployment Name]"
     "RetroResponse": "true or false"

   }
```

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

The UI

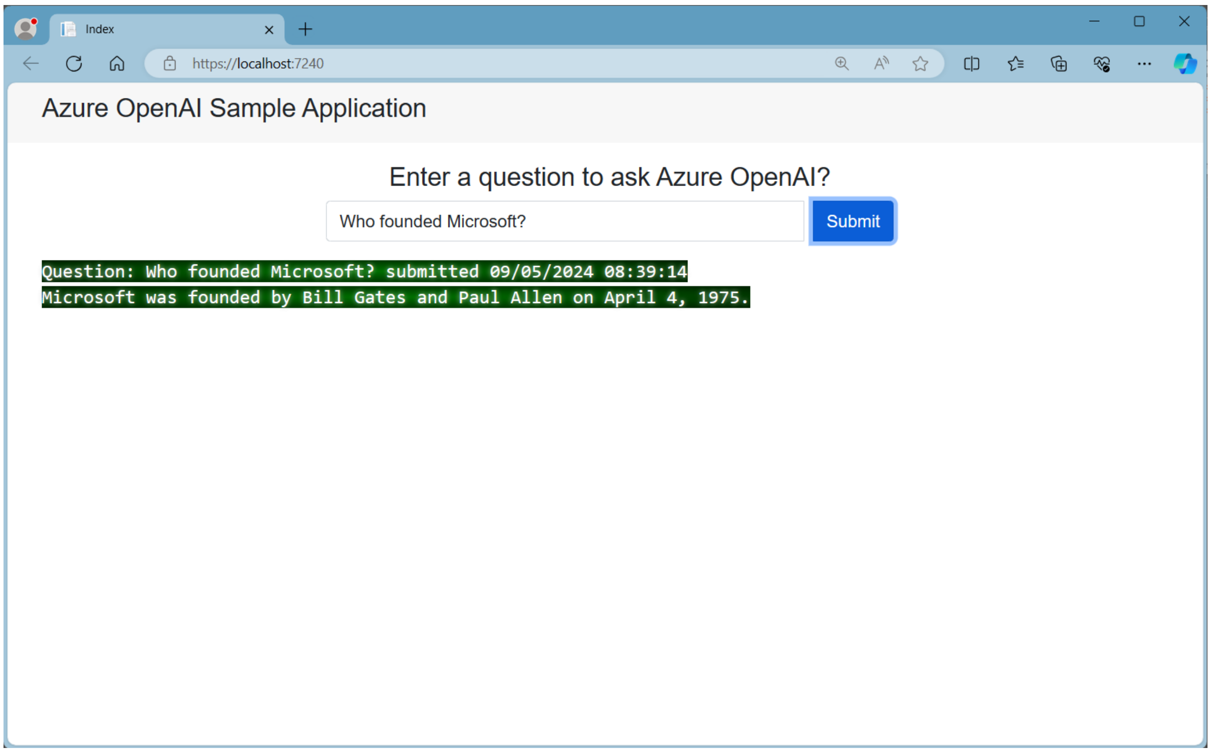
The landing page.


Index
https://localhost:7240
Azure OpenAI Sample Application
Enter a question to ask Azure OpenAI?
Submit

Sample Questions and Responses

Question 1

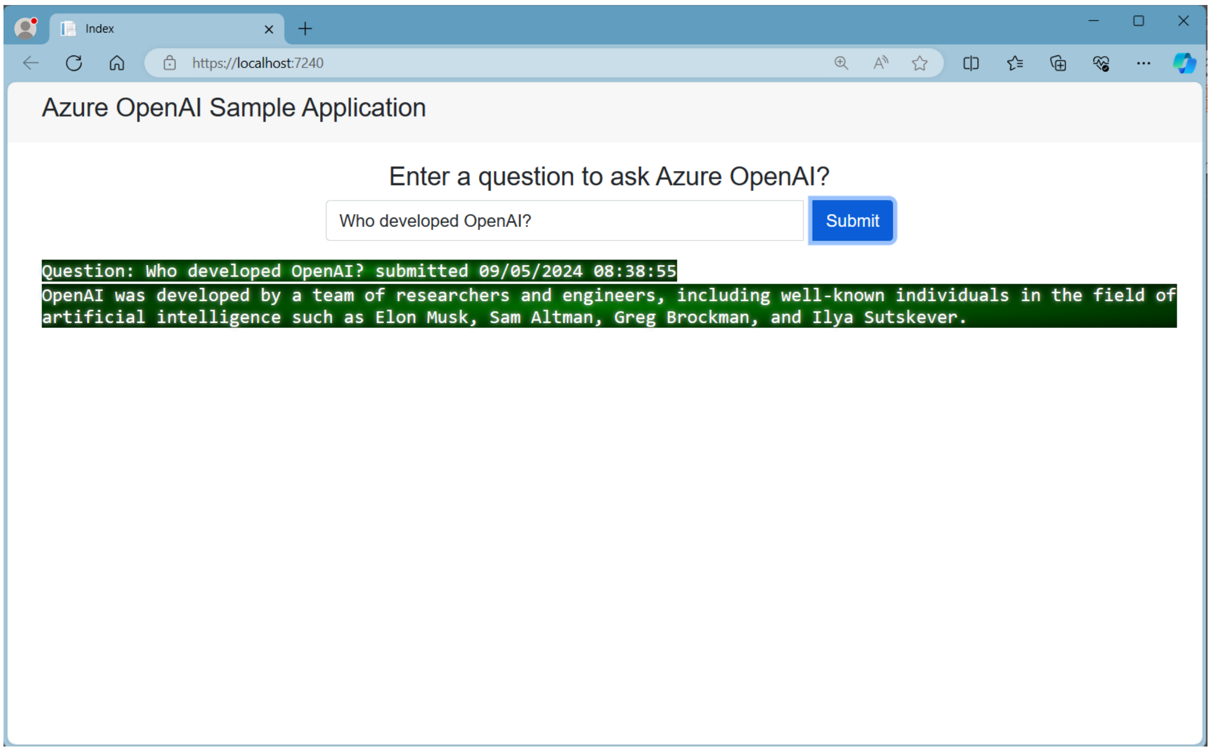
Who founded Microsoft?


Index
https://localhost:7240
Azure OpenAI Sample Application
Enter a question to ask Azure OpenAI?
Who founded Microsoft?
Submit

```
Question: Who founded Microsoft? submitted 09/05/2024 08:39:14
Microsoft was founded by Bill Gates and Paul Allen on April 4, 1975.
```

Question 2

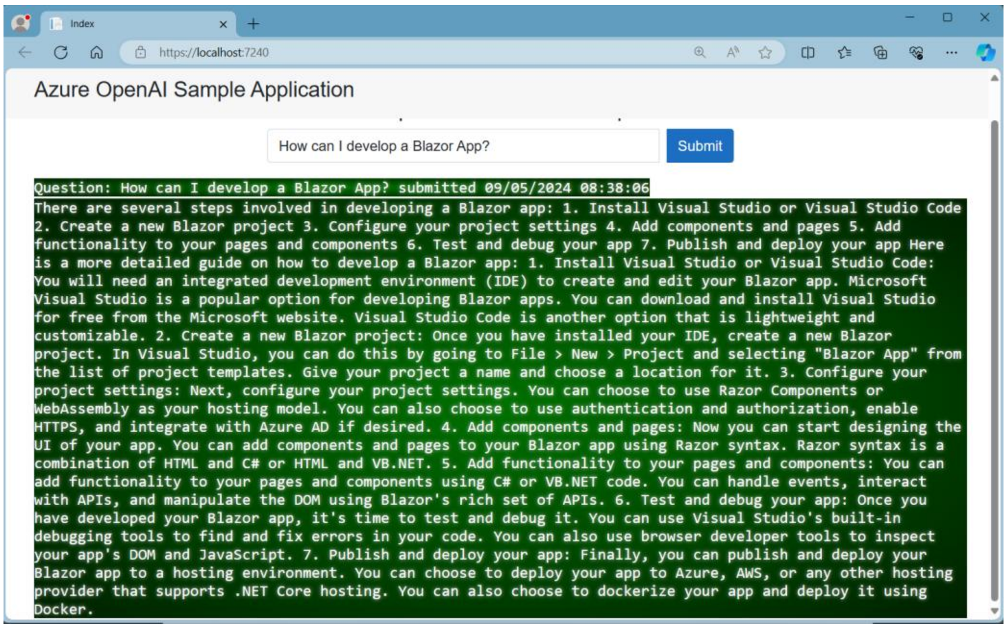
Who developed OpenAI?


Index
https://localhost:7240
Azure OpenAI Sample Application
Enter a question to ask Azure OpenAI?
Who developed OpenAI?
Submit

```
Question: Who developed OpenAI? submitted 09/05/2024 08:38:55
OpenAI was developed by a team of researchers and engineers, including well-known individuals in the field of
artificial intelligence such as Elon Musk, Sam Altman, Greg Brockman, and Ilya Sutskever.
```

Question 3

How can I develop a Blazor App?



Azure OpenAI Sample Application

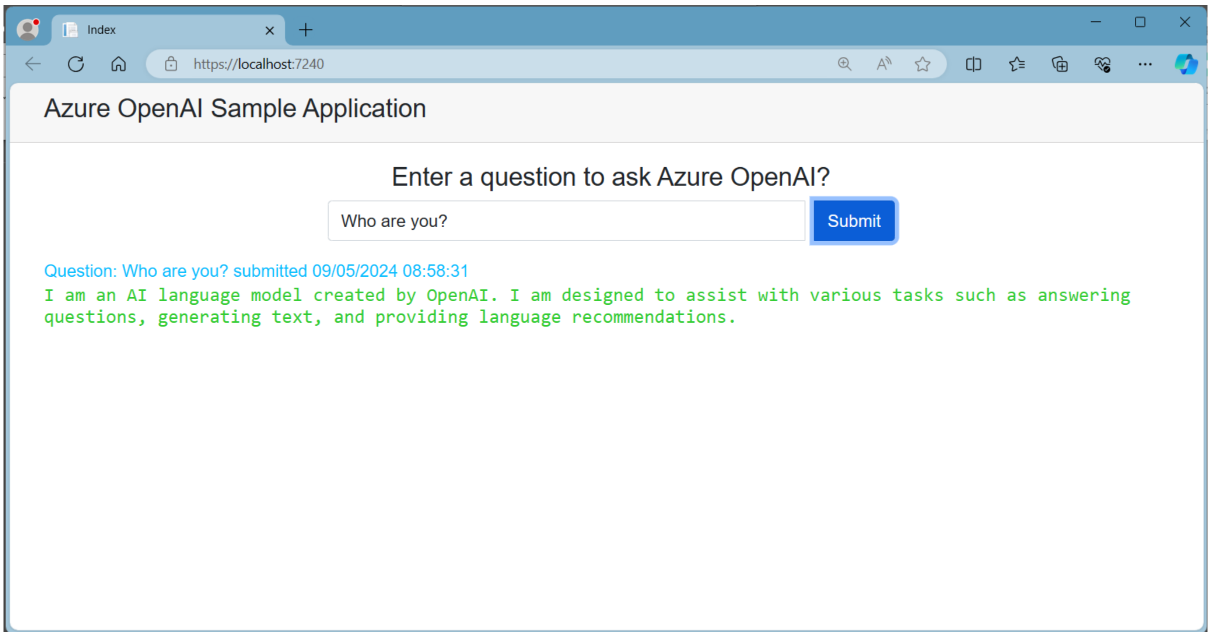
How can I develop a Blazor App?

```
Question: How can I develop a Blazor App? submitted 09/05/2024 08:38:06
There are several steps involved in developing a Blazor app: 1. Install Visual Studio or Visual Studio Code
2. Create a new Blazor project 3. Configure your project settings 4. Add components and pages 5. Add
functionality to your pages and components 6. Test and debug your app 7. Publish and deploy your app Here
is a more detailed guide on how to develop a Blazor app: 1. Install Visual Studio or Visual Studio Code:
You will need an integrated development environment (IDE) to create and edit your Blazor app. Microsoft
Visual Studio is a popular option for developing Blazor apps. You can download and install Visual Studio
for free from the Microsoft website. Visual Studio Code is another option that is lightweight and
customizable. 2. Create a new Blazor project: Once you have installed your IDE, create a new Blazor
project. In Visual Studio, you can do this by going to File > New > Project and selecting "Blazor App" from
the list of project templates. Give your project a name and choose a location for it. 3. Configure your
project settings: Next, configure your project settings. You can choose to use Razor Components or
WebAssembly as your hosting model. You can also choose to use authentication and authorization, enable
HTTPS, and integrate with Azure AD if desired. 4. Add components and pages: Now you can start designing the
UI of your app. You can add components and pages to your Blazor app using Razor syntax. Razor syntax is a
combination of HTML and C# or HTML and VB.NET. 5. Add functionality to your pages and components: You can
add functionality to your pages and components using C# or VB.NET code. You can handle events, interact
with APIs, and manipulate the DOM using Blazor's rich set of APIs. 6. Test and debug your app: Once you
have developed your Blazor app, it's time to test and debug it. You can use Visual Studio's built-in
debugging tools to find and fix errors in your code. You can also use browser developer tools to inspect
your app's DOM and JavaScript. 7. Publish and deploy your app: Finally, you can publish and deploy your
Blazor app to a hosting environment. You can choose to deploy your app to Azure, AWS, or any other hosting
provider that supports .NET Core hosting. You can also choose to dockerize your app and deploy it using
Docker.
```

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"
    }
```



Azure OpenAI Sample Application

Enter a question to ask Azure OpenAI?

Who are you?

Question: Who are you? submitted 09/05/2024 08:58:31

```
I am an AI language model created by OpenAI. I am designed to assist with various tasks such as answering
questions, generating text, and providing language recommendations.
```

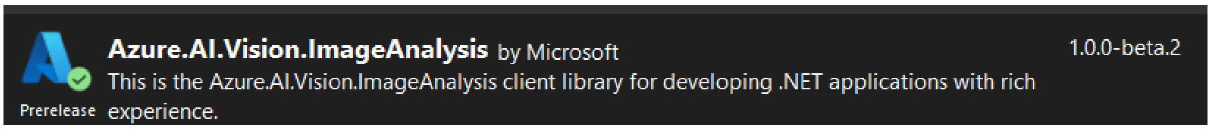
Blazor Simple AI Project (Part 2)

Image Analysis with Azure AI Vision

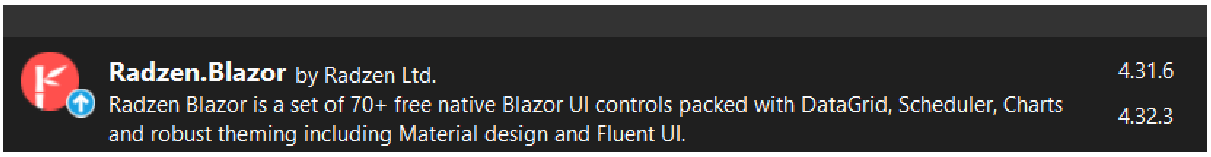
Welcome to the Blazor Simple AI Single Page App, Part 2 of the Microsoft AI services journey, which now includes image analysis utilising Microsoft Azure AI Vision. The Vision Read API is used to extract the text from an image. This document explains the project in my GitHub repository which is available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

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

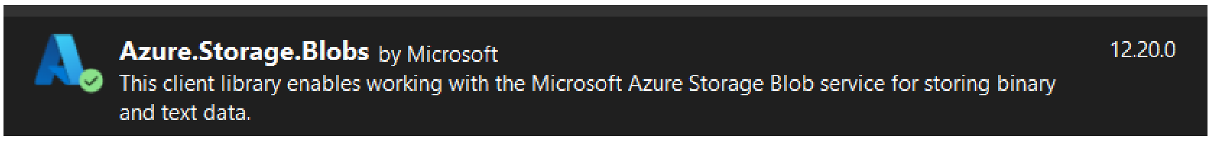
Azure AI Vision Image Analysis – for reading text and metadata from images.


Azure.AI.Vision.ImageAnalysis by Microsoft 1.0.0-beta.2
This is the Azure.AI.Vision.ImageAnalysis client library for developing .NET applications with rich experience.
Prerelease

Radzen Blazor – for providing an amazing UI experience.


Radzen.Blazor by Radzen Ltd. 4.31.6 4.32.3
Radzen Blazor is a set of 70+ free native Blazor UI controls packed with DataGrid, Scheduler, Charts and robust theming including Material design and Fluent UI.

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


Azure.Storage.Blobs by Microsoft 12.20.0
This client library enables working with the Microsoft Azure Storage Blob service for storing binary and text data.

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

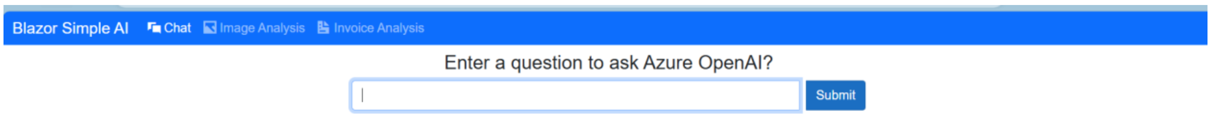
Two data classes have been created as follows:

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

The UI

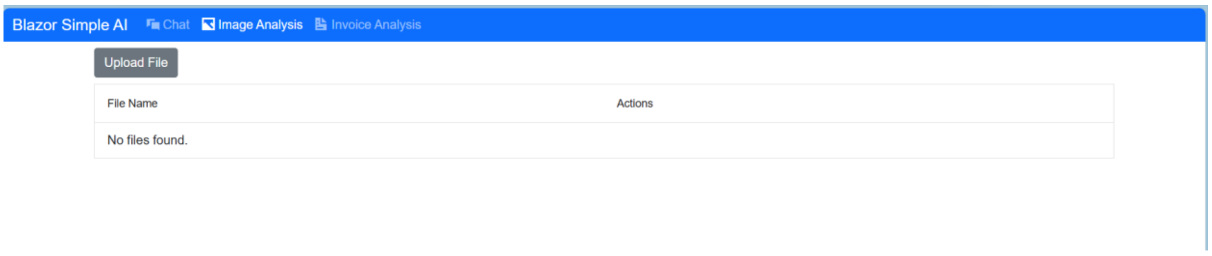
The UI is as follows. Notice the menu control has now changed since Part 1. Invoice Analysis will be formed in Part 3, at the time of writing this blog post, I had already uploaded the code to my GitHub repo.

Home page (Chat)



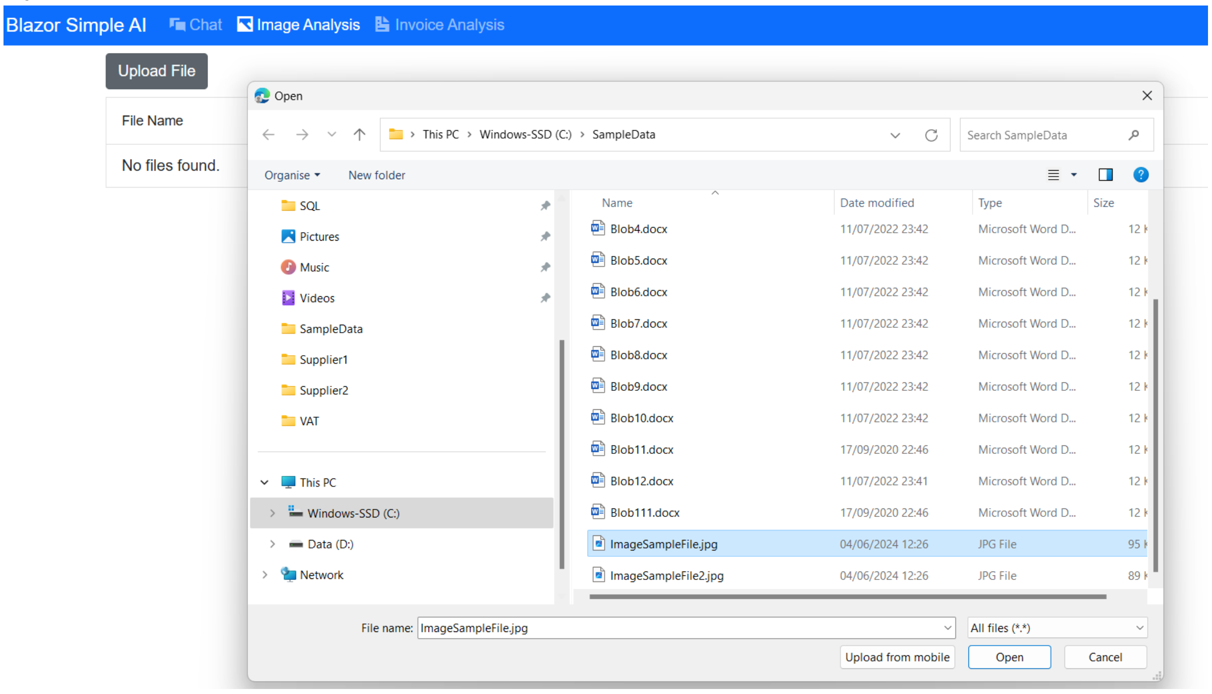
Enter a question to ask Azure OpenAI?

Image Analysis



File Name	Actions
No files found.	

Upload File Control

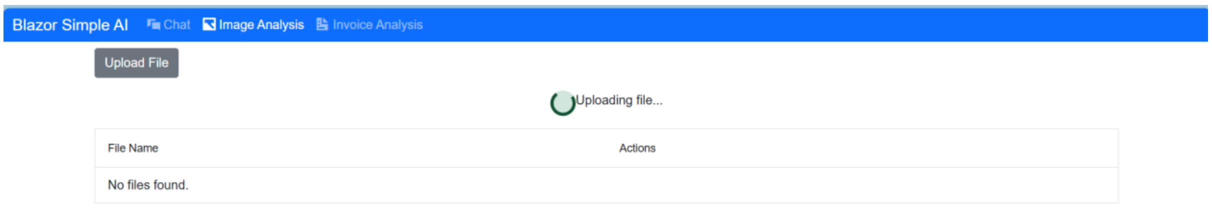


File Name
No files found.

Name	Date modified	Type	Size
Blob4.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob5.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob6.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob7.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob8.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob9.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob10.docx	11/07/2022 23:42	Microsoft Word D...	12
Blob11.docx	17/09/2020 22:46	Microsoft Word D...	12
Blob12.docx	11/07/2022 23:41	Microsoft Word D...	12
Blob111.docx	17/09/2020 22:46	Microsoft Word D...	12
ImageSampleFile.jpg	04/06/2024 12:26	JPG File	95
ImageSampleFile2.jpg	04/06/2024 12:26	JPG File	89

File name: ImageSampleFile.jpg
All files (*.*)

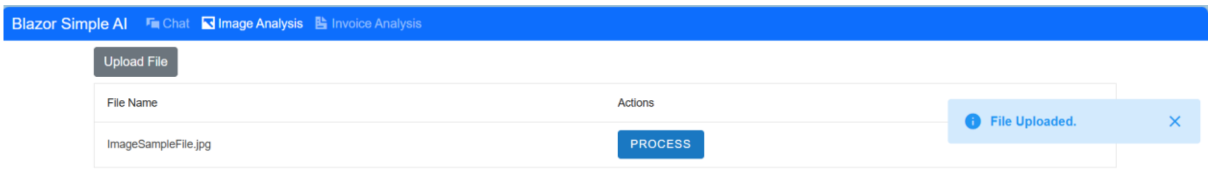
Upload Action Spinner



Uploading file...

File Name	Actions
No files found.	

Radzen Blazor File Uploaded Notification



File Name	Actions
ImageSampleFile.jpg	PROCESS

File Uploaded.

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.

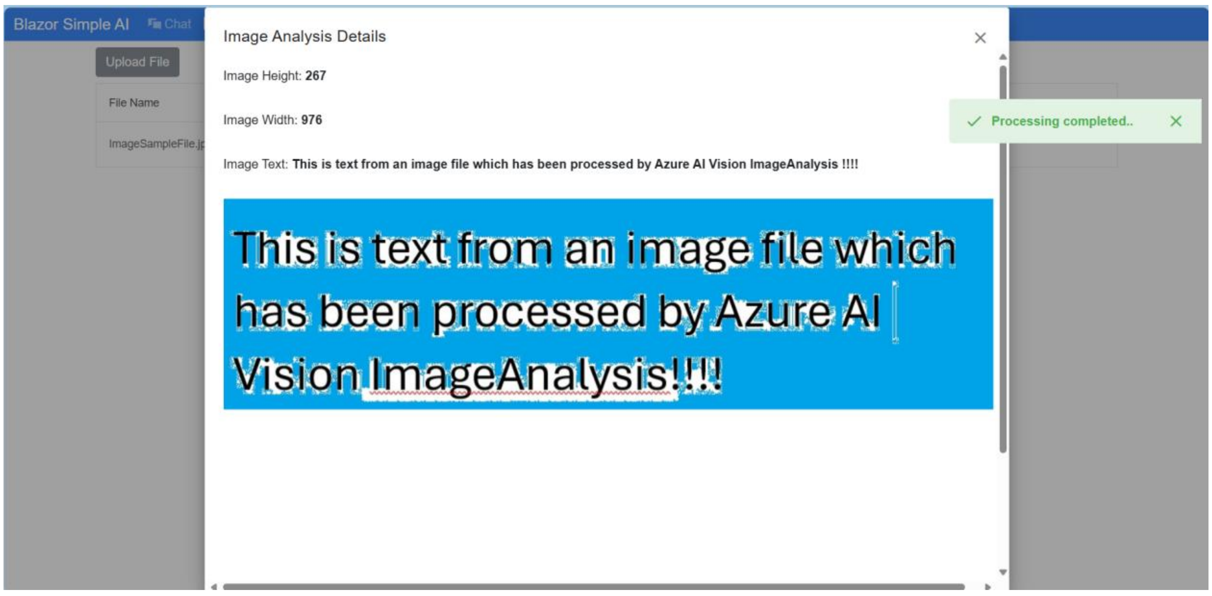


Image Analysis Details

Image Height: **267**

Image Width: **976**

Image Text: **This is text from an image file which has been processed by Azure AI Vision ImageAnalysis !!!!**

This is text from an image file which has been processed by Azure AI Vision ImageAnalysis!!!!

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.

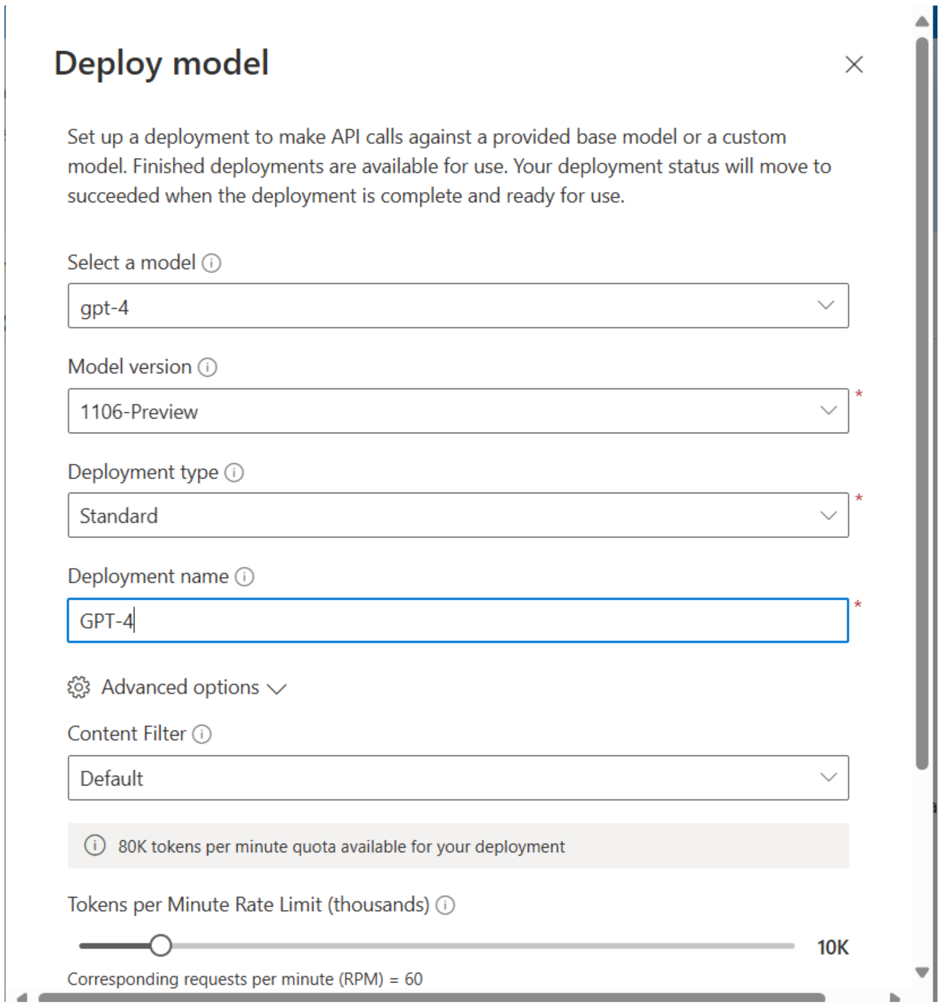
Blazor Simple AI Project (Part 3)

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**.



Deploy model

Set up a deployment to make API calls against a provided base model or a custom model. Finished deployments are available for use. Your deployment status will move to succeeded when the deployment is complete and ready for use.

Select a model: gpt-4

Model version: 1106-Preview

Deployment type: Standard

Deployment name: GPT-4

Advanced options

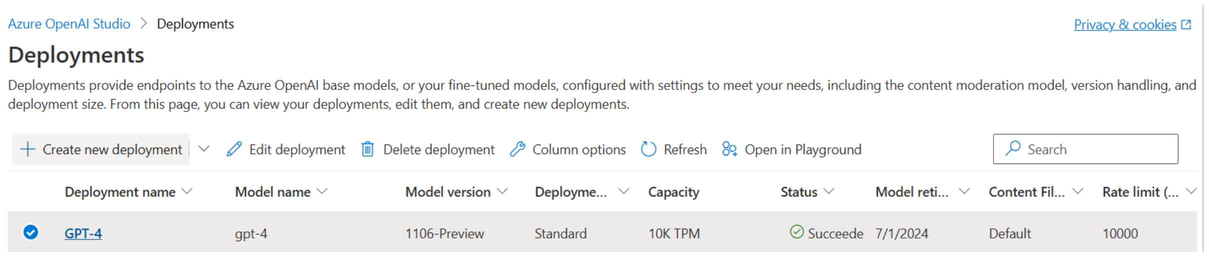
Content Filter: Default

80K tokens per minute quota available for your deployment

Tokens per Minute Rate Limit (thousands): 10K

Corresponding requests per minute (RPM) = 60

Your model will be deployed.



Azure OpenAI Studio > Deployments

Privacy & cookies

Deployments

Deployments provide endpoints to the Azure OpenAI base models, or your fine-tuned models, configured with settings to meet your needs, including the content moderation model, version handling, and deployment size. From this page, you can view your deployments, edit them, and create new deployments.

	Deployment name	Model name	Model version	Deployme...	Capacity	Status	Model reti...	Content Fil...	Rate limit (...
✓	GPT-4	gpt-4	1106-Preview	Standard	10K TPM	✓ Succeede	7/1/2024	Default	10000

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”.

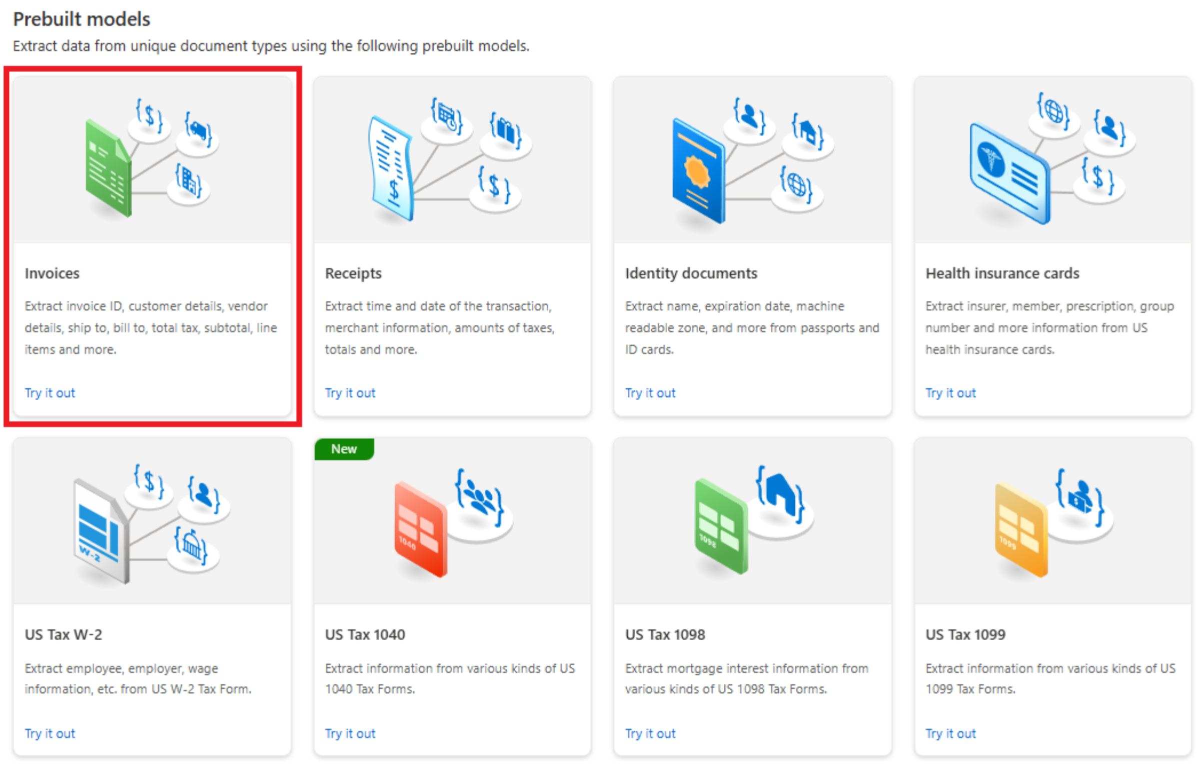
```
"AzureAIConfig": {
        "OpenAIDeploymentName": "GPT-4",
}
```

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.


Prebuilt models
Extract data from unique document types using the following prebuilt models.

Invoices
Extract invoice ID, customer details, vendor details, ship to, bill to, total tax, subtotal, line items and more.
Try it out

Receipts
Extract time and date of the transaction, merchant information, amounts of taxes, totals and more.
Try it out

Identity documents
Extract name, expiration date, machine readable zone, and more from passports and ID cards.
Try it out

Health insurance cards
Extract insurer, member, prescription, group number and more information from US health insurance cards.
Try it out

US Tax W-2
Extract employee, employer, wage information, etc. from US W-2 Tax Form.
Try it out

New
US Tax 1040
Extract information from various kinds of US 1040 Tax Forms.
Try it out

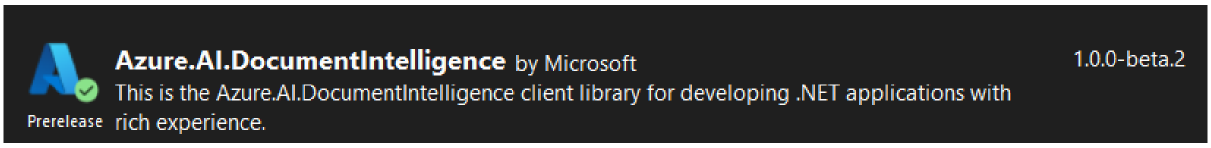
US Tax 1098
Extract mortgage interest information from various kinds of US 1098 Tax Forms.
Try it out

US Tax 1099
Extract information from various kinds of US 1099 Tax Forms.
Try it out

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)


Azure.AI.DocumentIntelligence by Microsoft 1.0.0-beta.2
This is the Azure.AI.DocumentIntelligence client library for developing .NET applications with rich experience.
Prerelease

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

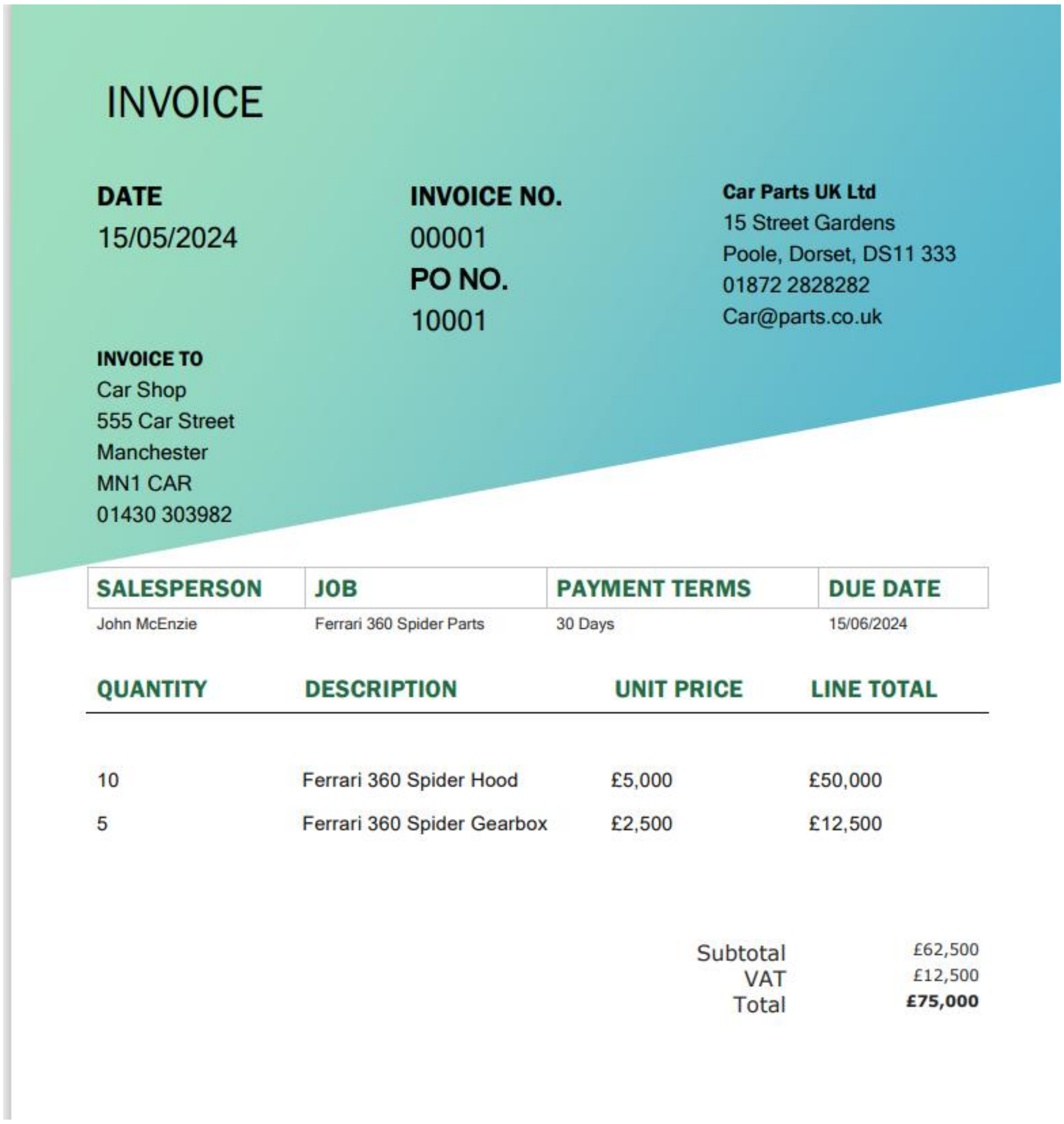
InvoiceAnalysisData.cs – The class for the invoice.

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

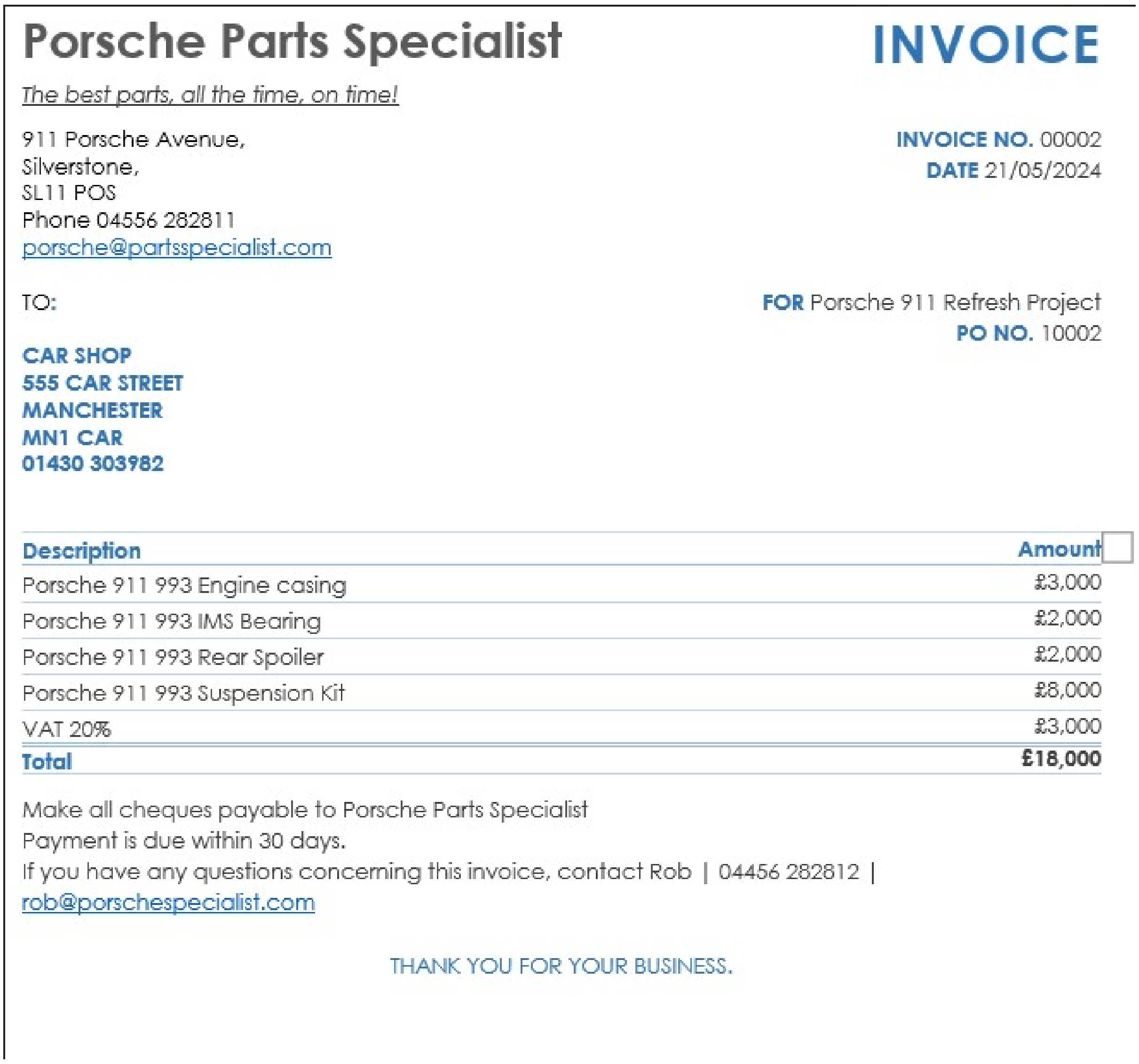
QUANTITY	DESCRIPTION	UNIT PRICE	LINE TOTAL
10	Ferrari 360 Spider Hood	£5,000	£50,000
5	Ferrari 360 Spider Gearbox	£2,500	£12,500

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

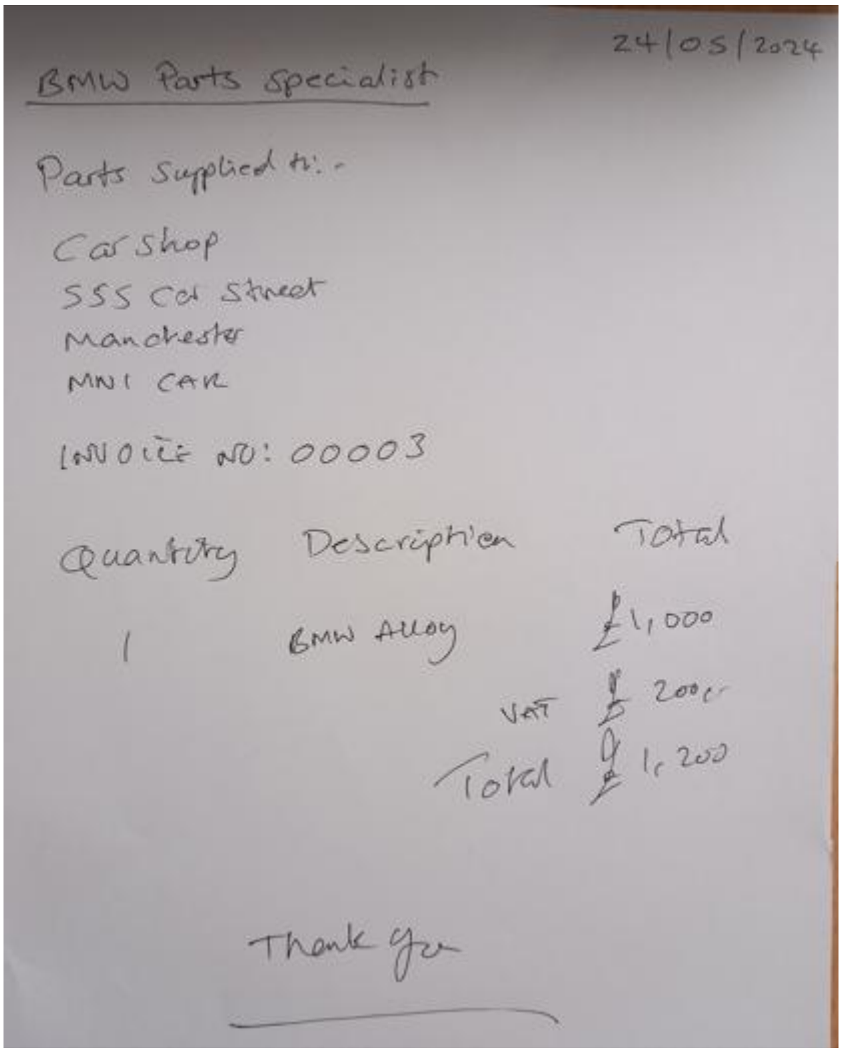
FOR Porsche 911 Refresh Project
PO NO. 10002

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



24/05/2024

BMW Parts specialist

Parts supplied to:-

Car Shop
555 Car Street
Manchester
MN1 CAR

INVOICE NO: 00003

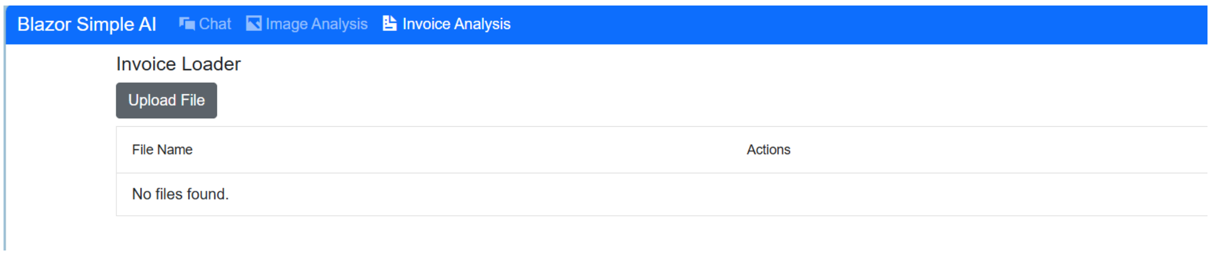
Quantity	Description	Total
1	BMW Alloy	£1,000
	VAT	£ 200
	Total	£ 1,200

Thank you

The UI

The UI for invoice analysis is as follows.

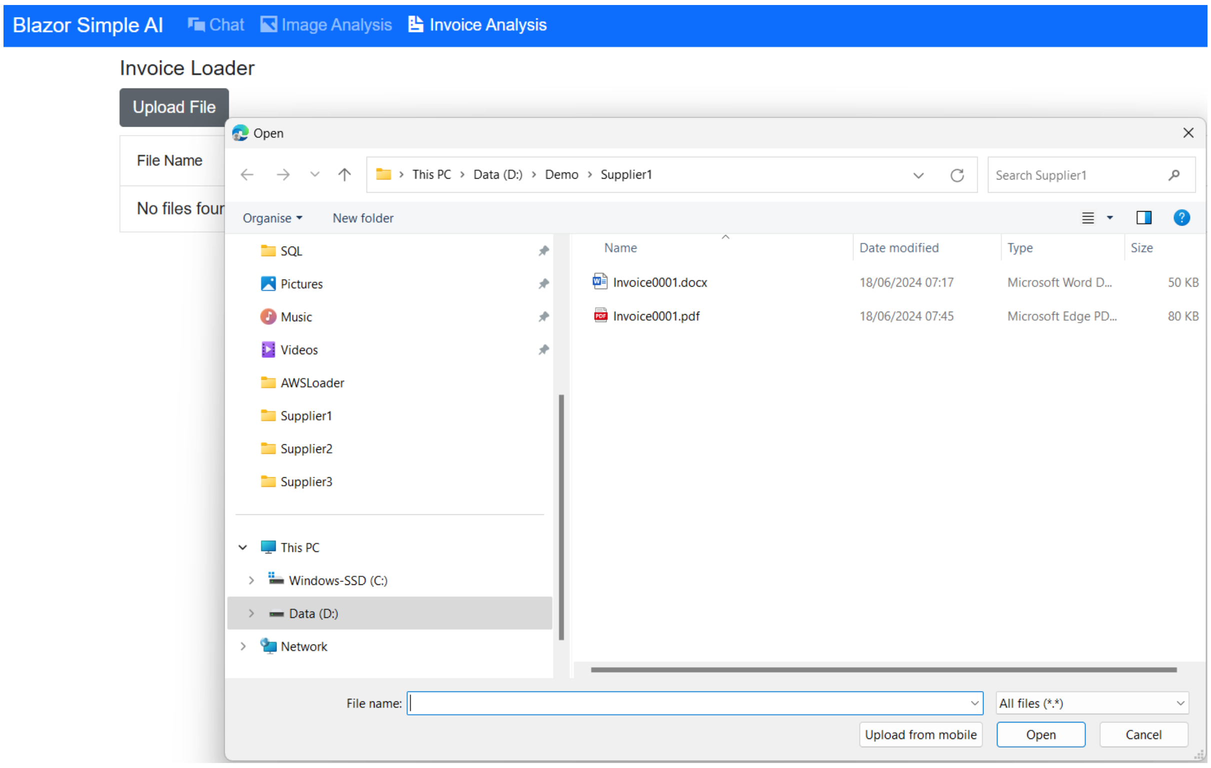
Invoice Analysis



Invoice Loader

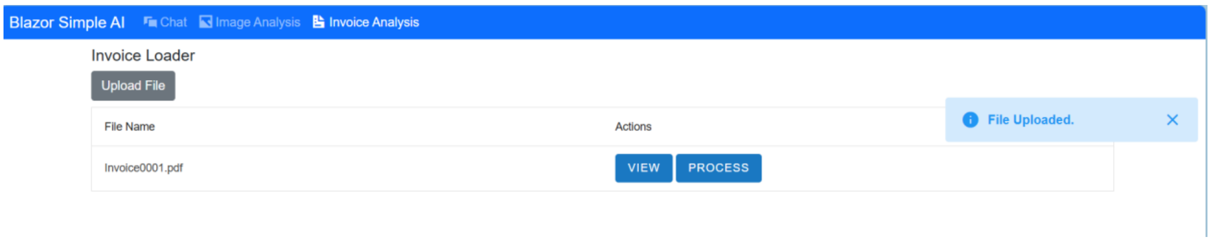
File Name	Actions
No files found.	

Upload File



Invoice Loader

Name	Date modified	Type	Size
Invoice0001.docx	18/06/2024 07:17	Microsoft Word D...	50 KB
Invoice0001.pdf	18/06/2024 07:45	Microsoft Edge PD...	80 KB

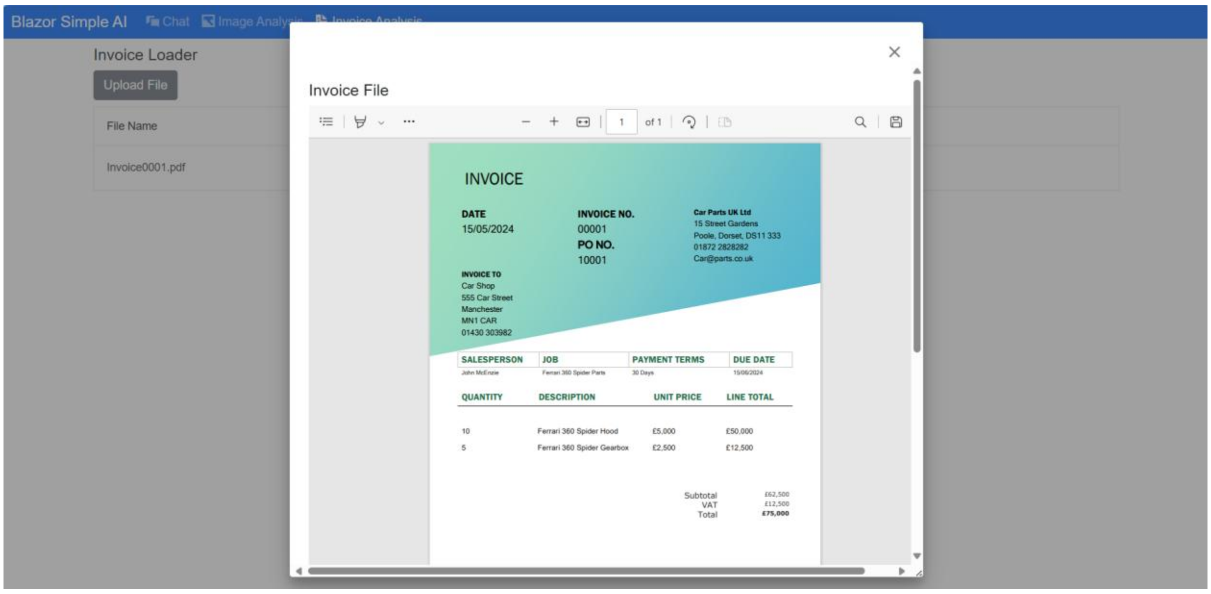


Invoice Loader

File Uploaded.

File Name	Actions
Invoice0001.pdf	VIEW PROCESS

View Button – Opens the PDF in a dialog box



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

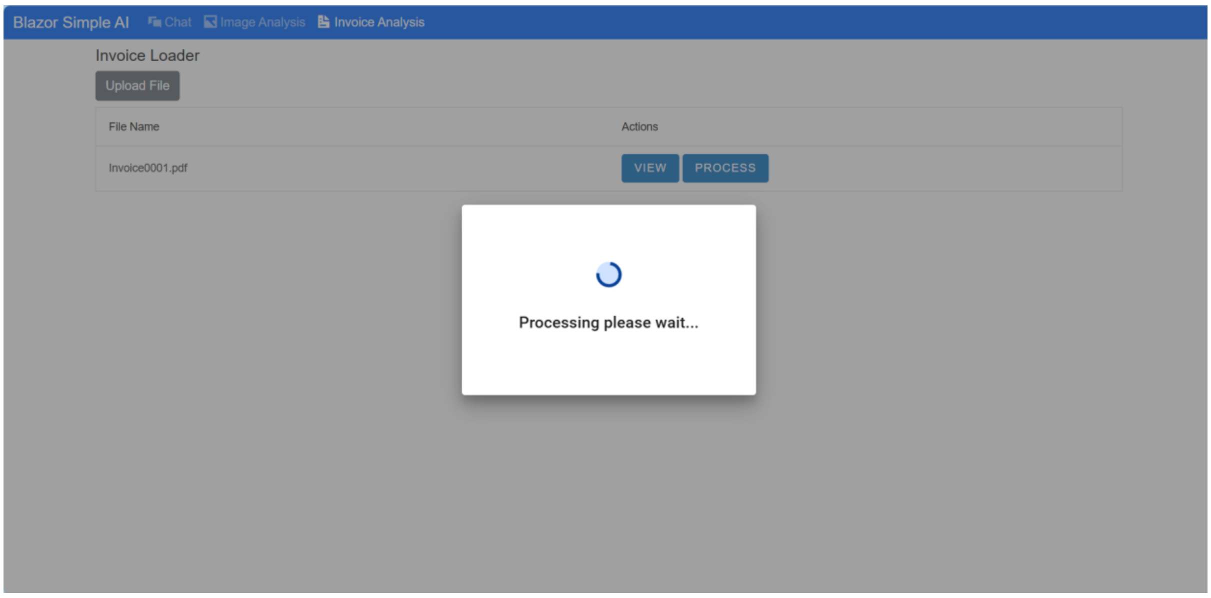
INVOICE TO
Car Shop
555 Car Street
Manchester
MN1 CAR
01430 303982

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

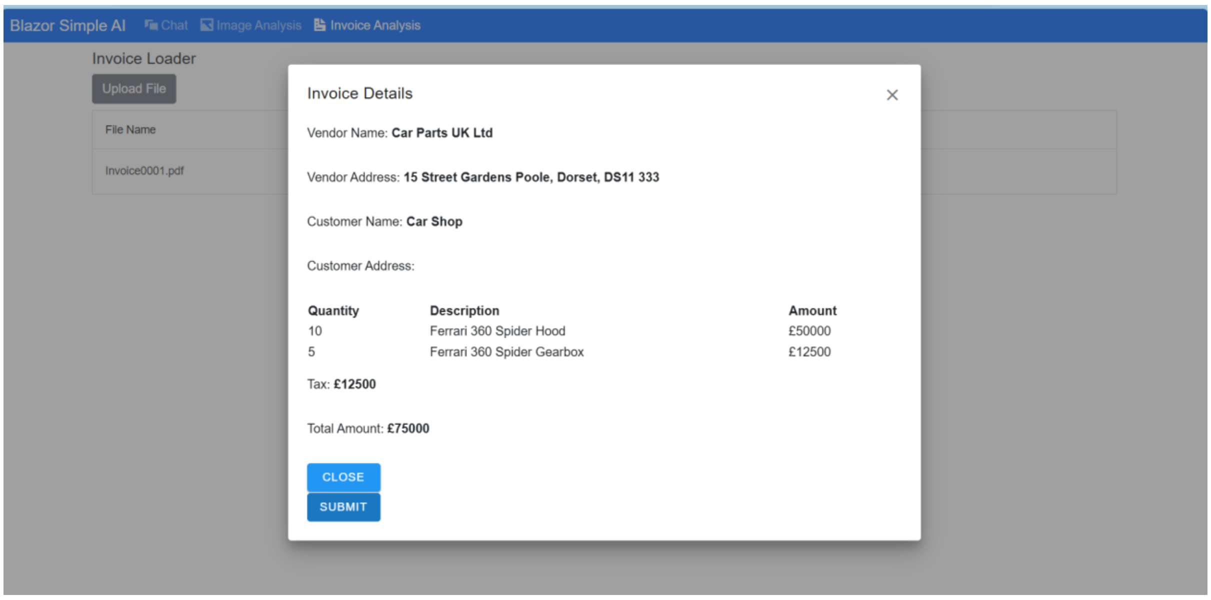
QUANTITY	DESCRIPTION	UNIT PRICE	LINE TOTAL
10	Ferrari 360 Spider Hood	£5,000	£50,000
5	Ferrari 360 Spider Gearbox	£2,500	£12,500

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

Process Button – Interactive Dialog box



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



Invoice Details

Vendor Name: **Car Parts UK Ltd**

Vendor Address: **15 Street Gardens Poole, Dorset, DS11 333**

Customer Name: **Car Shop**

Customer Address:

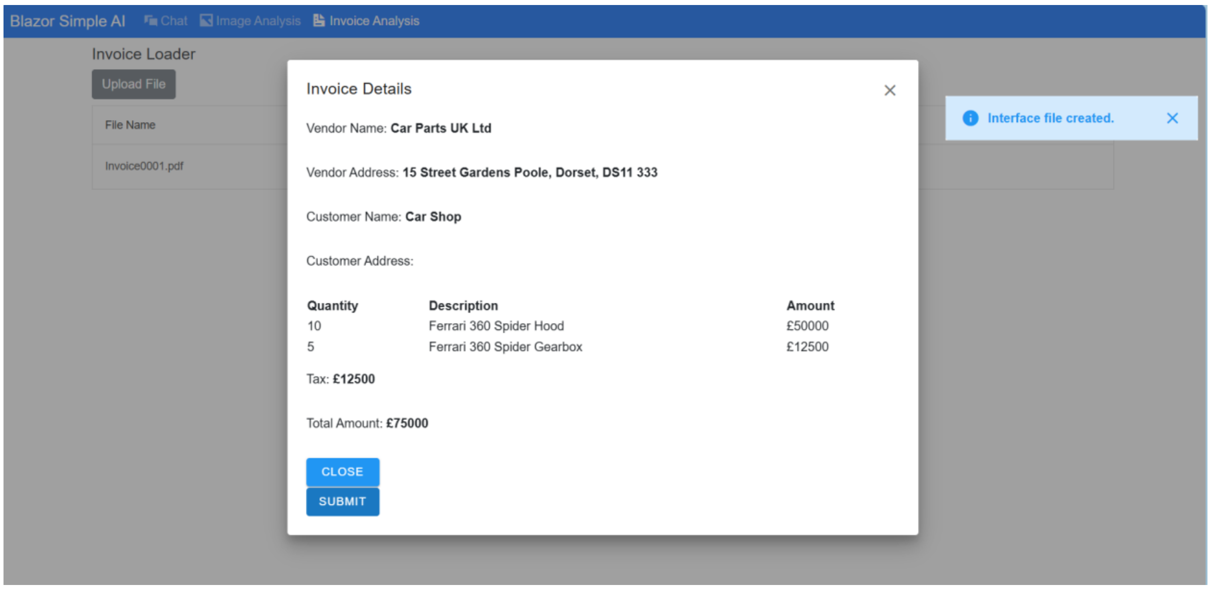
Quantity	Description	Amount
10	Ferrari 360 Spider Hood	£50000
5	Ferrari 360 Spider Gearbox	£12500

Tax: **£12500**

Total Amount: **£75000**

CLOSE
SUBMIT

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



Invoice Details

Vendor Name: **Car Parts UK Ltd**

Vendor Address: **15 Street Gardens Poole, Dorset, DS11 333**

Customer Name: **Car Shop**

Customer Address:

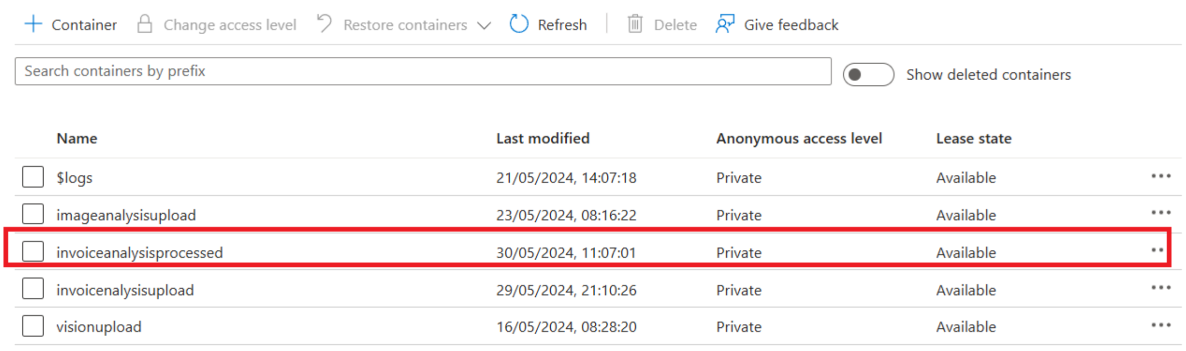
Quantity	Description	Amount
10	Ferrari 360 Spider Hood	£50000
5	Ferrari 360 Spider Gearbox	£12500

Tax: **£12500**

Total Amount: **£75000**

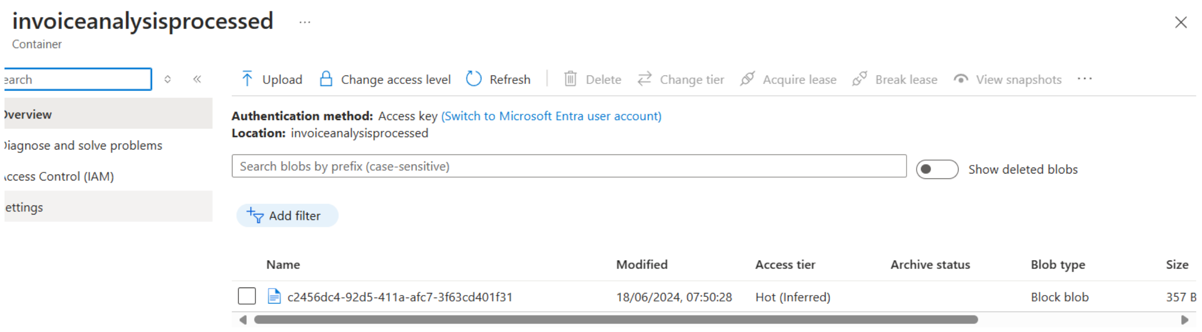
CLOSE
SUBMIT

Azure Storage – (invoiceanalysisupload container)



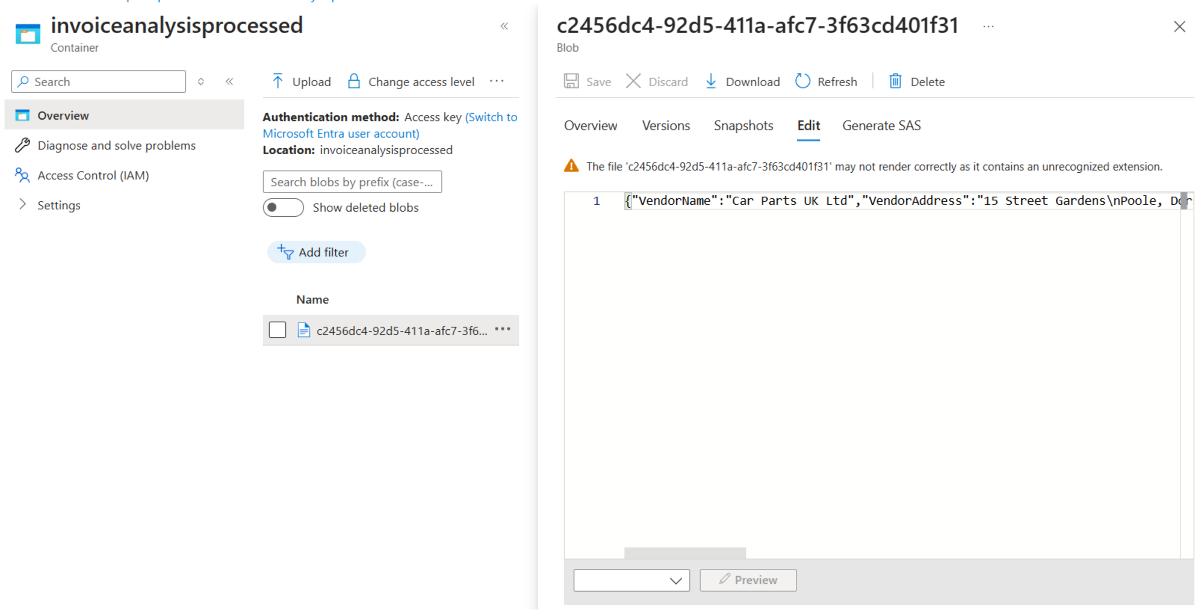
Name	Last modified	Anonymous access level	Lease state
$logs	21/05/2024, 14:07:18	Private	Available
imageanalysisupload	23/05/2024, 08:16:22	Private	Available
invoiceanalysisprocessed	30/05/2024, 11:07:01	Private	Available
invoicenalysisupload	29/05/2024, 21:10:26	Private	Available
visionupload	16/05/2024, 08:28:20	Private	Available

Processed Output file


invoiceanalysisprocessed
Container

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: invoiceanalysisprocessed

Name	Modified	Access tier	Archive status	Blob type	Size
c2456dc4-92d5-411a-afc7-3f63cd401f31	18/06/2024, 07:50:28	Hot (Inferred)		Block blob	357 B


c2456dc4-92d5-411a-afc7-3f63cd401f31
Blob

The file 'c2456dc4-92d5-411a-afc7-3f63cd401f31' may not render correctly as it contains an unrecognized extension.

```
{"VendorName":"Car Parts UK Ltd","VendorAddress":"15 Street Gardens\nPoole, Do
```

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"
}
```

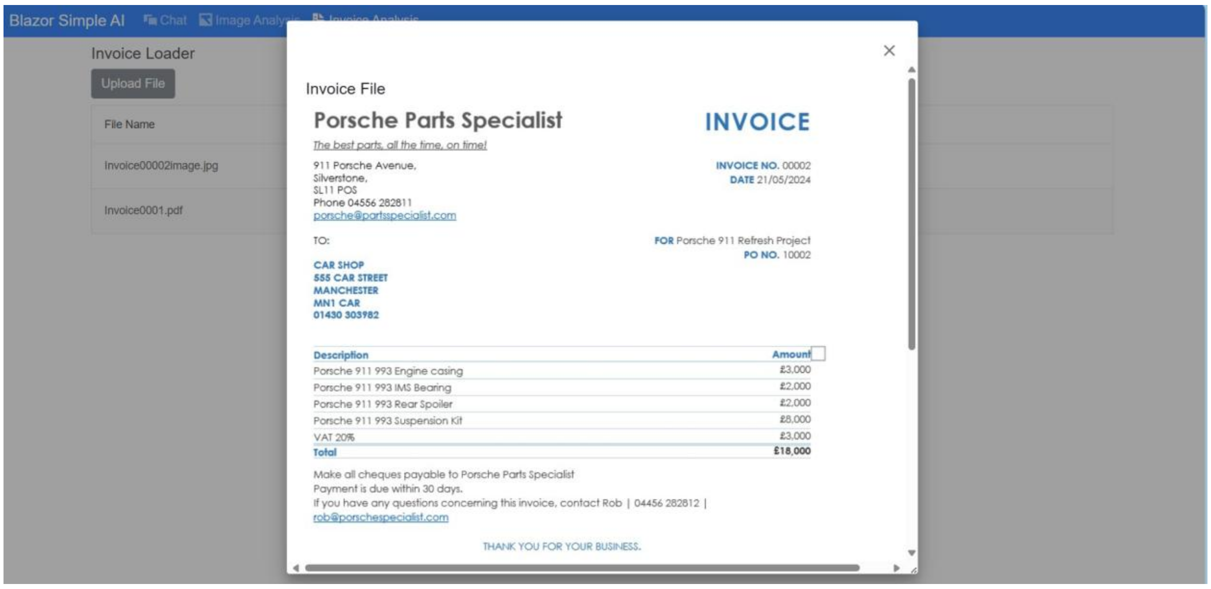
Note: The code does not extract the customer address, but this is in fact possible.

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 File

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:

FOR Porsche 911 Refresh Project
PO NO. 10002

CAR SHOP
555 CAR STREET
MANCHESTER
MN1 CAR
01430 303982

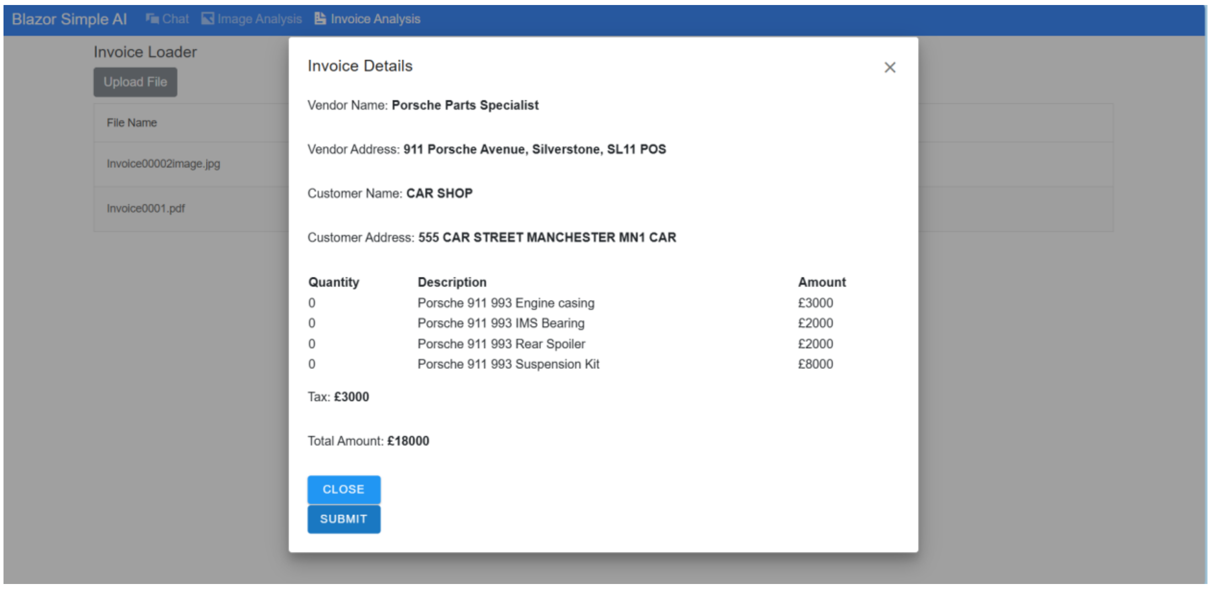
Description	Amount
Porsche 911 993 Engine casing	£3,000
Porsche 911 993 IMS Bearing	£2,000
Porsche 911 993 Rear Spoiler	£2,000
Porsche 911 993 Suspension Kit	£8,000
VAT 20%	£3,000
Total	**£18,000**

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 2 – Processed

(Note: Missing Quantity in the file)



Invoice Details

Vendor Name: **Porsche Parts Specialist**

Vendor Address: **911 Porsche Avenue, Silverstone, SL11 POS**

Customer Name: **CAR SHOP**

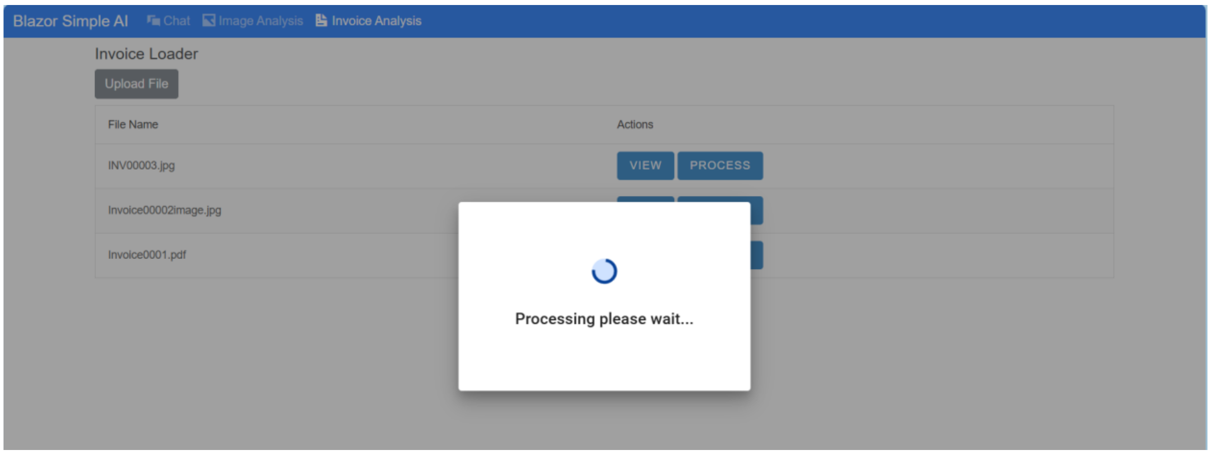
Customer Address: **555 CAR STREET MANCHESTER MN1 CAR**

Quantity	Description	Amount
0	Porsche 911 993 Engine casing	£3000
0	Porsche 911 993 IMS Bearing	£2000
0	Porsche 911 993 Rear Spoiler	£2000
0	Porsche 911 993 Suspension Kit	£8000

Tax: **£3000**

Total Amount: **£18000**

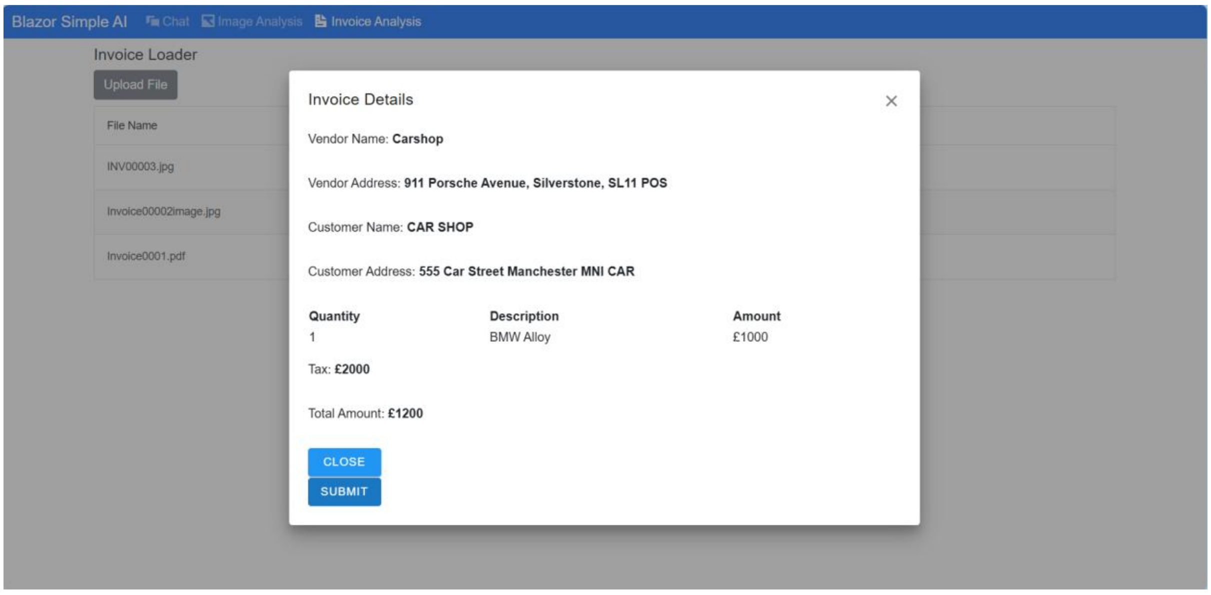
Invoice 3 – Handwritten jpeg



Invoice Loader

File Name	Actions
INV00003.jpg	VIEW PROCESS
Invoice00002image.jpg	
Invoice0001.pdf	

Processing please wait...



Invoice Loader

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**

Blazor Simple AI Project (Part 5)

Azure Open AI Chat Audio Recoding Button

Welcome to the Blazor Simple AI Single Page App, Part 5 of the Microsoft AI services journey, which now includes an audio recording button in the Open AI Chat component.

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

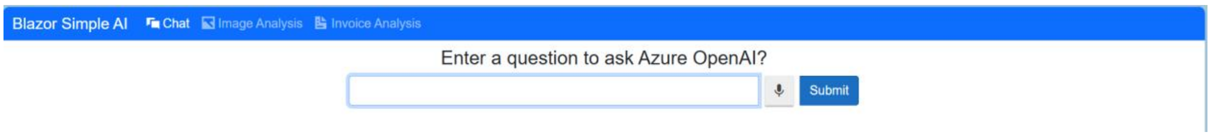
Visual Changes

The audio button has been added to the index.razor page as this is the main landing page. The audio button component is part of Radzen Blazor and simple to interact with which is the `RadzenSpeechToTextButton`. This utilises JavaScript as a component itself, there is an API to get user media.

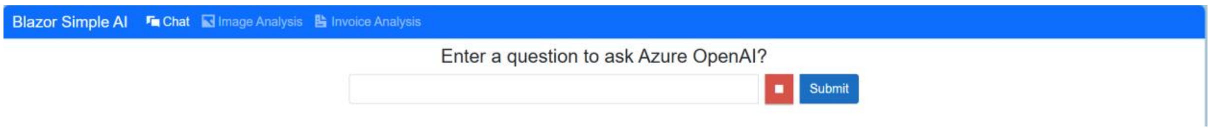
For further information on the Razden Blazor Speech To Text Button, see: Blazor SpeechToTextButton Component | Free UI Components by Radzen.

Landing Page

The new landing page has the audio button added next to the chat text box.

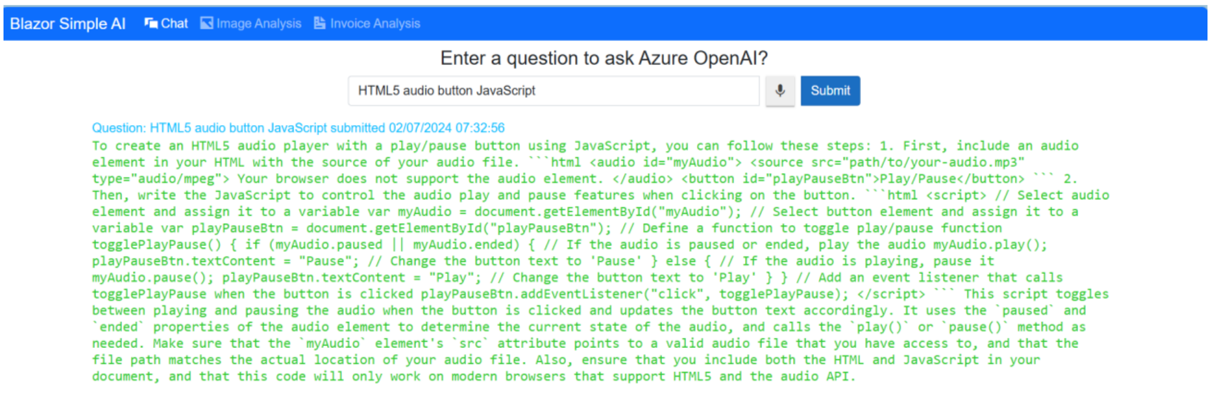


When you click on the audio button, the first instance will provide a prompt requesting access from the site to the devices microphone, then recording has started as shown below.



When you are finished speaking, you click the button to stop recording and the text is submitted to the Question string and a OnChange event occurs and the Question value is set, then the state is changed for the component. Since the Question string is a bound field to the child component, AzureOpenAIChat, which then executes the component code to call the Microsoft Azure Open AI service with the text that was bound to the Question string.

An example of the recorded audio text and Azure Open AI response is shown below.


Blazor Simple AI
Chat
Image Analysis
Invoice Analysis
Enter a question to ask Azure OpenAI?
HTML5 audio button JavaScript
Submit

Question: HTML5 audio button JavaScript submitted 02/07/2024 07:32:56

```
To create an HTML5 audio player with a play/pause button using JavaScript, you can follow these steps: 1. First, include an audio
element in your HTML with the source of your audio file. ```html <audio id="myAudio"> <source src="path/to/your-audio.mp3"
type="audio/mpeg"> Your browser does not support the audio element. </audio> <button id="playPauseBtn">Play/Pause</button> ``` 2.
Then, write the JavaScript to control the audio play and pause features when clicking on the button. ```html <script> // Select audio
element and assign it to a variable var myAudio = document.getElementById("myAudio"); // Select button element and assign it to a
variable var playPauseBtn = document.getElementById("playPauseBtn"); // Define a function to toggle play/pause function
togglePlayPause() { if (myAudio.paused || myAudio.ended) { // If the audio is paused or ended, play the audio myAudio.play();
playPauseBtn.textContent = "Pause"; // Change the button text to 'Pause' } else { // If the audio is playing, pause it
myAudio.pause(); playPauseBtn.textContent = "Play"; // Change the button text to 'Play' } } // Add an event listener that calls
togglePlayPause when the button is clicked playPauseBtn.addEventListener("click", togglePlayPause); </script> ``` This script toggles
between playing and pausing the audio when the button is clicked and updates the button text accordingly. It uses the `paused` and
`ended` properties of the audio element to determine the current state of the audio, and calls the `play()` or `pause()` method as
needed. Make sure that the `myAudio` element's `src` attribute points to a valid audio file that you have access to, and that the
file path matches the actual location of your audio file. Also, ensure that you include both the HTML and JavaScript in your
document, and that this code will only work on modern browsers that support HTML5 and the audio API.
```

Code Changes

The following code changes were made in index.razor.

Added audio recoding button and spacing.

```
<Radzen.Blazor.RadzenSpeechToTextButton class="padding-right:10px;"
Change="@(args => OnSpeechCaptured(args, "SpeechToTextButton"))" />
<div style="padding-left:10px;" />
```

Added the OnSpeechCaptured method.

Note: I removed the question marks and period from the string return from the Radzen Speech To Text button as the characters were automatically to the returned text string value from the component.

```
private void OnSpeechCaptured(string speechValue, string name)
{
    speechValue = speechValue.Trim(new Char[] { '.', '?' });
    RecordedSpeech = speechValue;
    Question = RecordedSpeech;
    this.StateHasChanged();
}
```

For my next post, I will be utilizing the `RadzenSpeechToTextButton` for a different purpose in the Blazor Simple AI project.

Blazor Simple AI Project (Part 6)

Azure Open AI Image Generation

Welcome to the Blazor Simple AI Single Page App, Part 6 of the Microsoft AI services journey, which now includes Microsoft Azure Open AI image generation.

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

Since part 5, the following changes to the project have been implemented.

Project Changes

The following changes have been made to the project in this version.

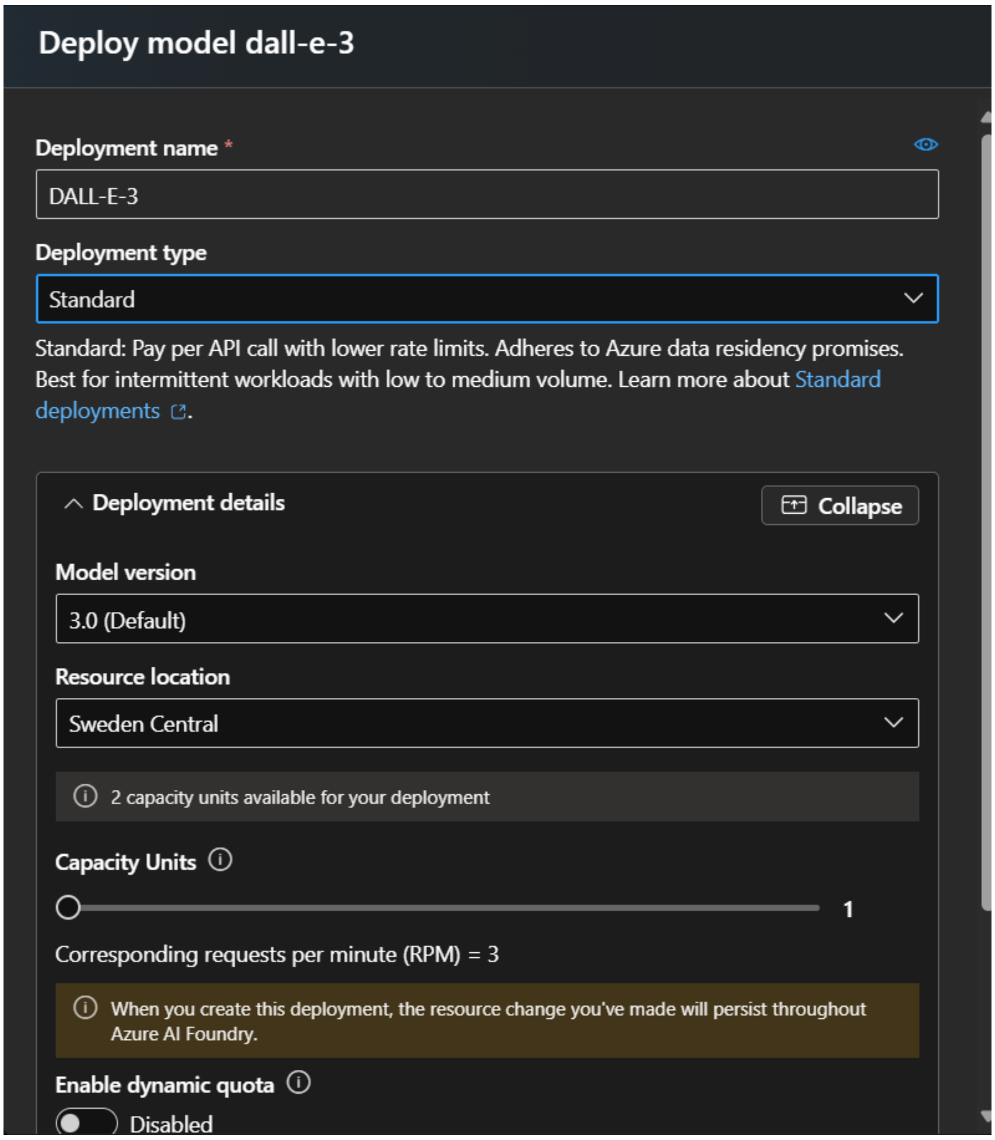
1) ImageGen.razor page has been added to the project Pages folder. This is a page hosting the image generation component and necessary code
2) AzureOpenAIImageGeneration.razor component has been added to the project components folder which handles the user prompt, then displays the image viewer dialogue with the Azure Open AI generated image
3) ImageViewer.razor component has been added to the project components folder. This displays the image dialogue
4) The following configuration changes have been added to AppSettings.Json for the DALL-E deployment name

```
"AzureAIConfig"

{

 "OpenAIDALLEEndpoint": "[You Azure Open AI endpoint which is hosting the
DALL-E deployment]",
 "OpenAIKeyDALLECredential": "[Your Azure Open AI key] ",
 "OpenAIDALLEDeploymentName": "[Your DALL-E deployment name]"
}
```

5) The following base model was added to the Open AI Service.



Deploy model dall-e-3

Deployment name *

DALL-E-3

Deployment type

Standard

Standard: Pay per API call with lower rate limits. Adheres to Azure data residency promises. Best for intermittent workloads with low to medium volume. Learn more about Standard deployments.

Deployment details

Model version

3.0 (Default)

Resource location

Sweden Central

2 capacity units available for your deployment

Capacity Units

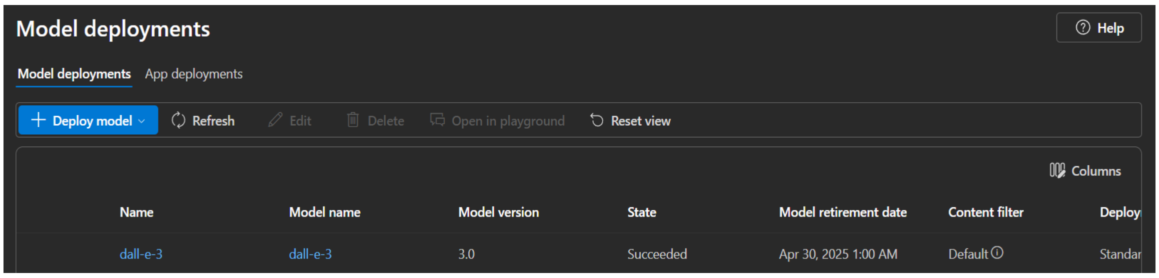
1

Corresponding requests per minute (RPM) = 3

When you create this deployment, the resource change you've made will persist throughout Azure AI Foundry.

Enable dynamic quota

Disabled



Model deployments

Name	Model name	Model version	State	Model retirement date	Content filter	Deploy
dall-e-3	dall-e-3	3.0	Succeeded	Apr 30, 2025 1:00 AM	Default	Standar

Components

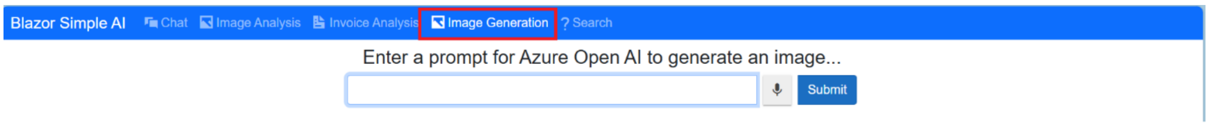
ImageGen.razor (page)
The ImageGen.razor page is used to host the prompt for the user to generate the image. This is distinctively similar to the Open AI Chat index page, which follows a similar pattern to accept prompt or audio recordings and then the text is passed to the child component, AzureOpenAIImageGeneration, to process the text and generate the image from the Azure Open AI service.

AzureOpenAIImageGeneration.razor
A component which accepts the text from the prompt and then calls the Azure Open AI service to generate the image.

ImageView.Razor
This component displays the output, the image, generated from the Azure Open AI service which is the template for the image dialogue box. This is called from the Image Generation child component.

The UI

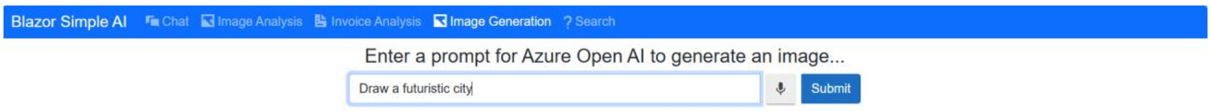
The landing page. I have added a Image Generation navigation link.



Sample Questions and Responses

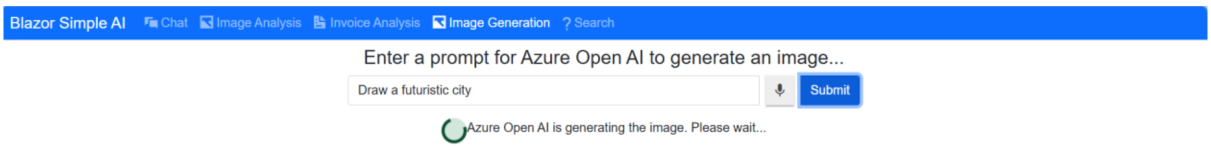
Question 1

“Draw a futuristic city”

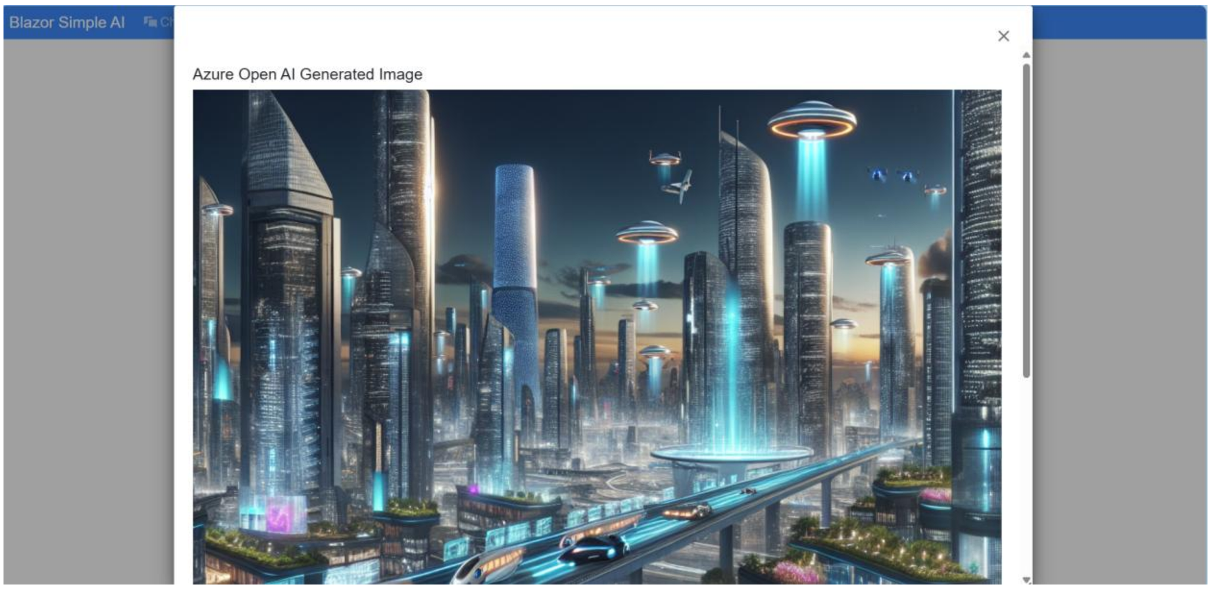


Output for question 1:

The process takes a few seconds for the image generation to complete, so I have displayed a spinning wheel and a prompt for the user to wait for the result.


Blazor Simple AI
Chat
Image Analysis
Invoice Analysis
Image Generation
Search
Enter a prompt for Azure Open AI to generate an image...
Draw a futuristic city
Submit
Azure Open AI is generating the image. Please wait...

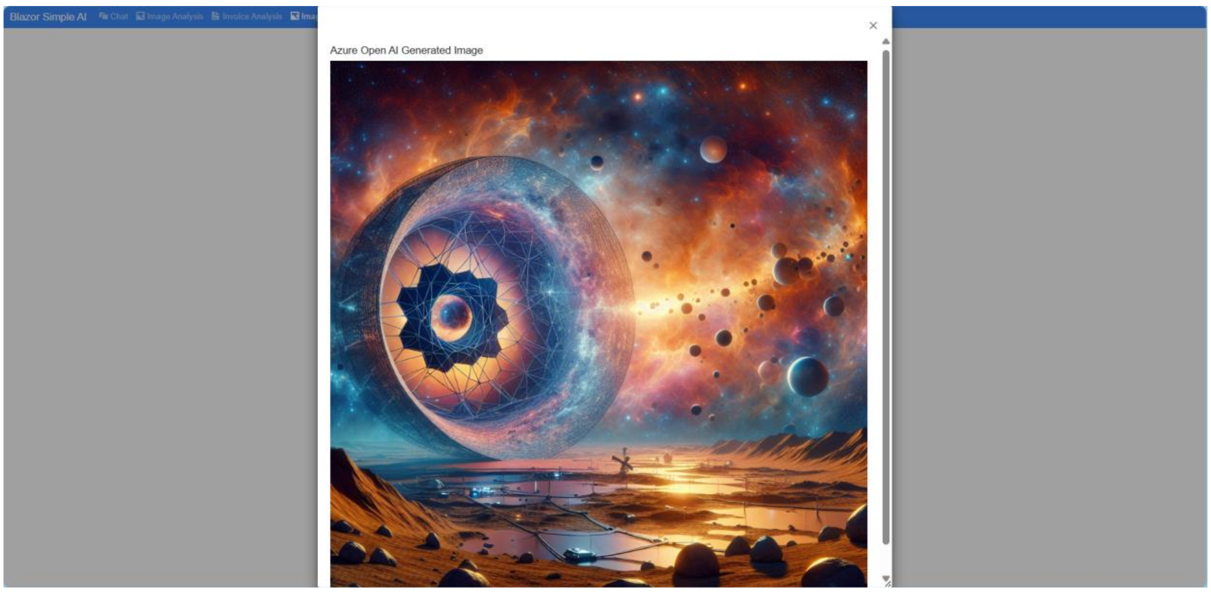
The output is displayed as follows:



Question 2

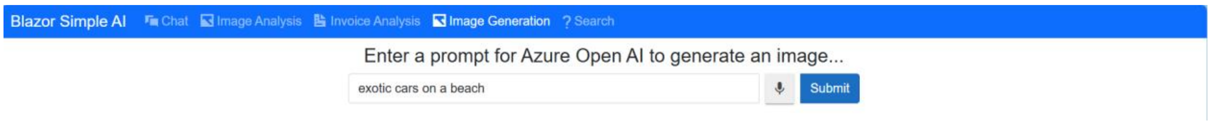
“Origins of the universe by the James Webb telescope”

The output is displayed as follows:



Question 3

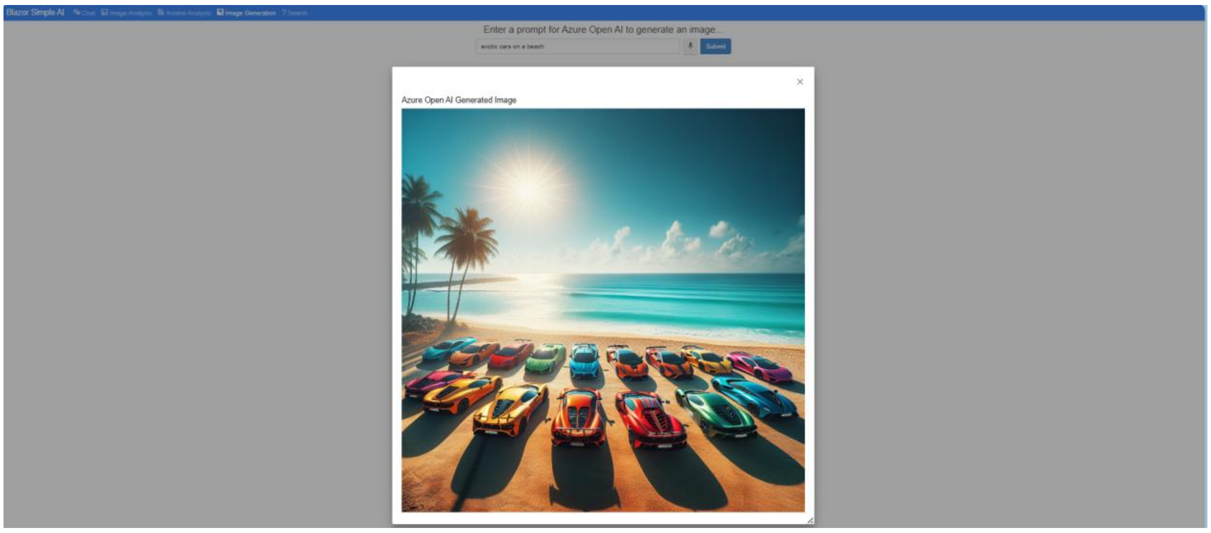
"exotic cars on a beach"



Enter a prompt for Azure Open AI to generate an image...

exotic cars on a beach

The output is displayed as follows:



Enter a prompt for Azure Open AI to generate an image...

exotic cars on a beach

Azure Open AI Generated Image

That's it!

This shows how simple it is to integrate a Blazor Web application with Azure Open AI image generation.