Blazor Simple AI Project

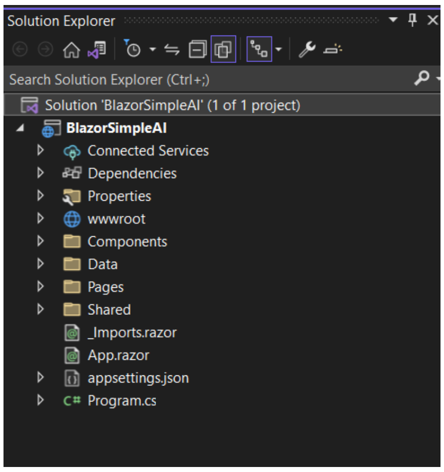
Welcome to the Blazor Simple AI Single Page App, the AI that responds to questions instantly using Microsoft Azure OpenAI Services. This document explains the project in my GitHub repository which is available here: https://github.com/tejinderrai/public/tree/main/BlazorSimpleAI.

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.

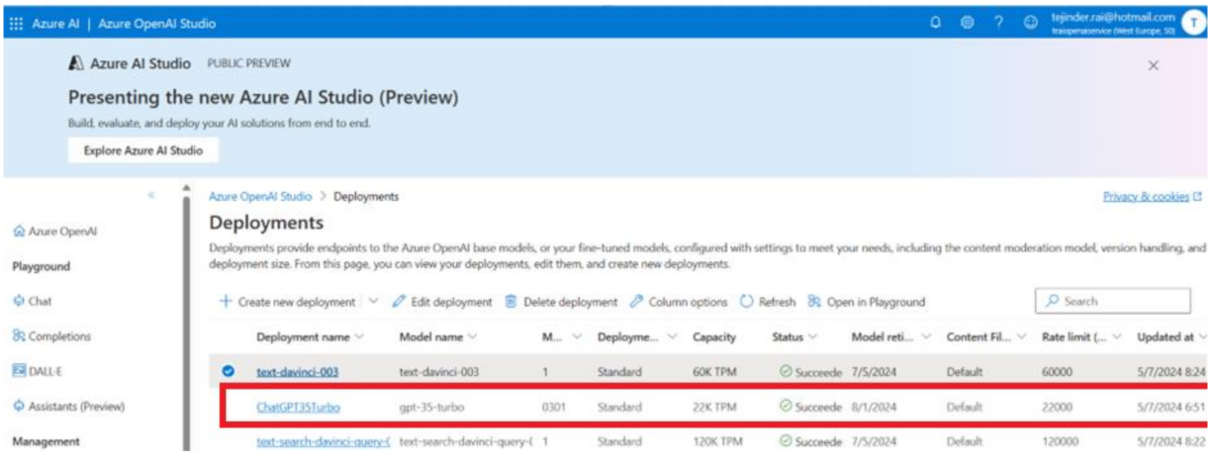
Core Blazor Template Changes

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



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

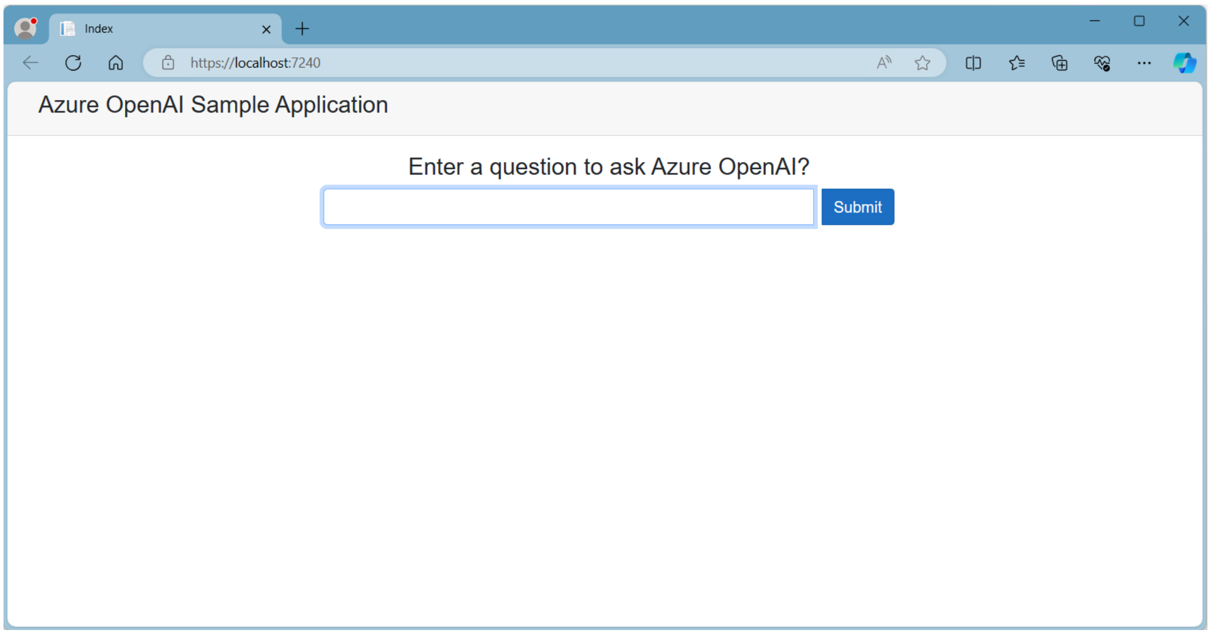
```
"AzureAIConfig": {
    "OpenAIEndpoint": "https://[You Azure OpenAI Service].openai.azure.com/",
    "OpenAIKeyCredential": "[Your Azure Open AI Key]",
    "OpenAIDeploymentName": "[Your Azure Open AI Deployment Name]"
     "RetroResponse": "true or false"

  }
```

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

The UI

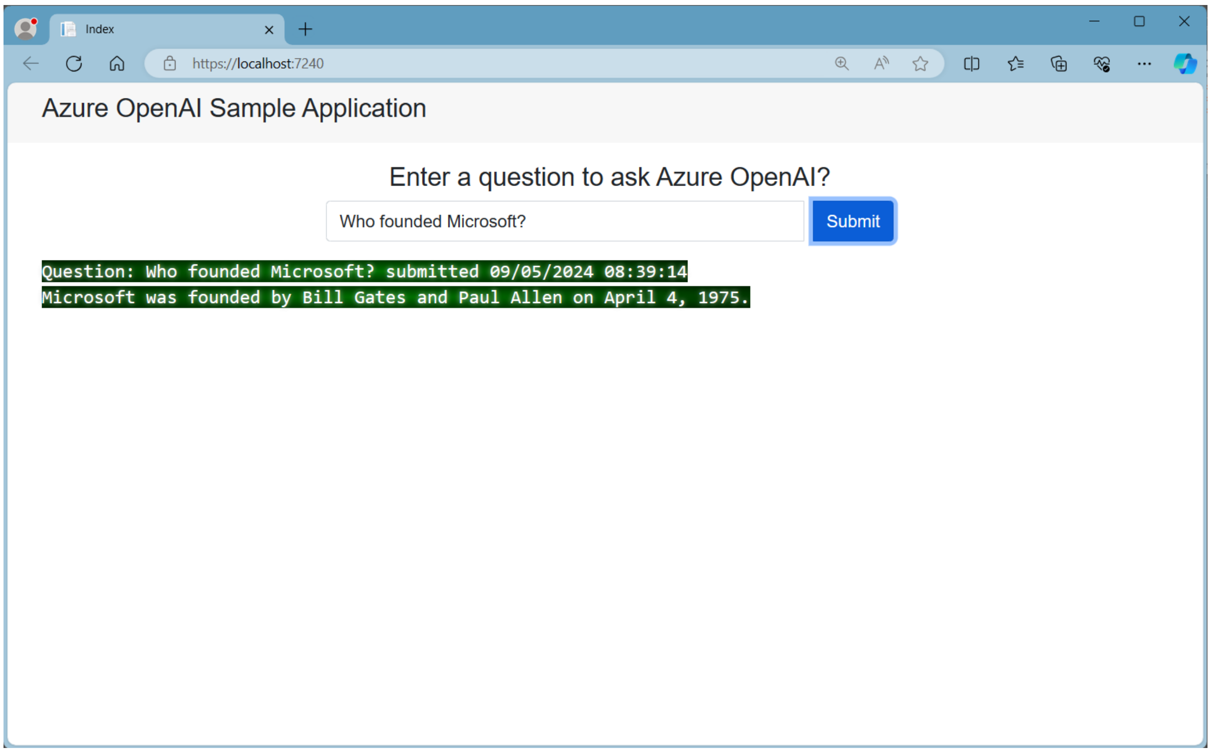
The landing page.


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

Sample Questions and Responses

Question 1

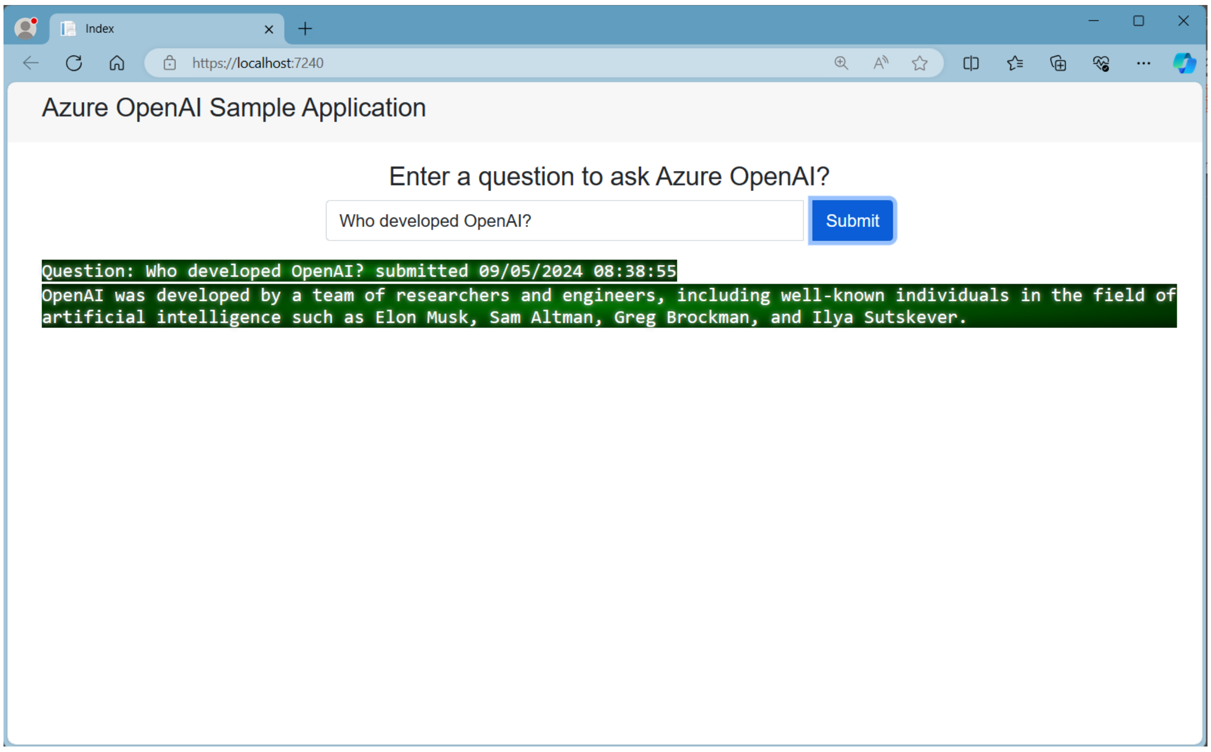
Who founded Microsoft?


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?



Azure OpenAI Sample Application

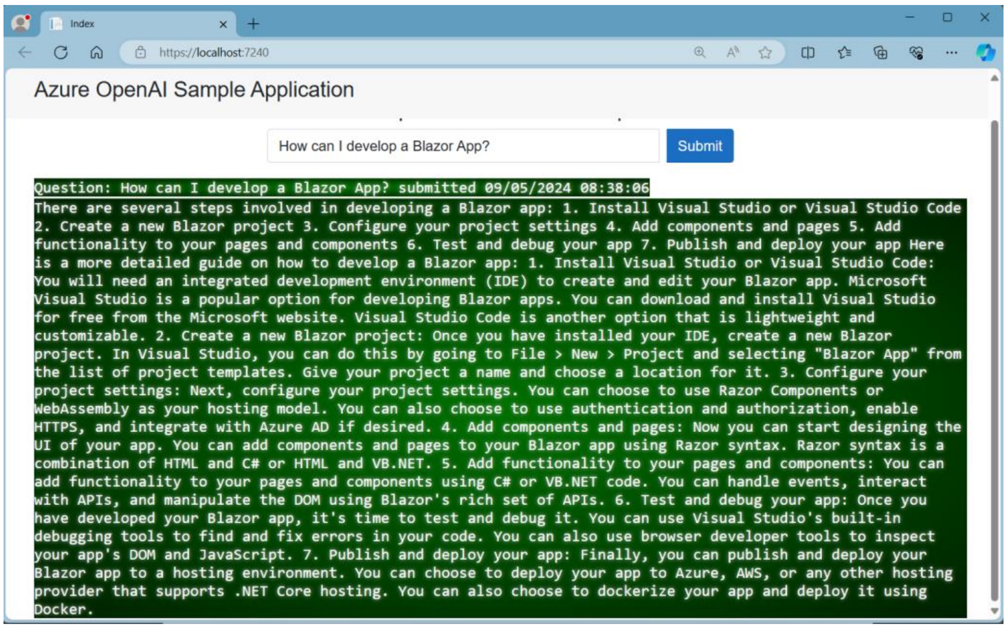
Enter a question to ask Azure OpenAI?

Who developed OpenAI?

```
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?

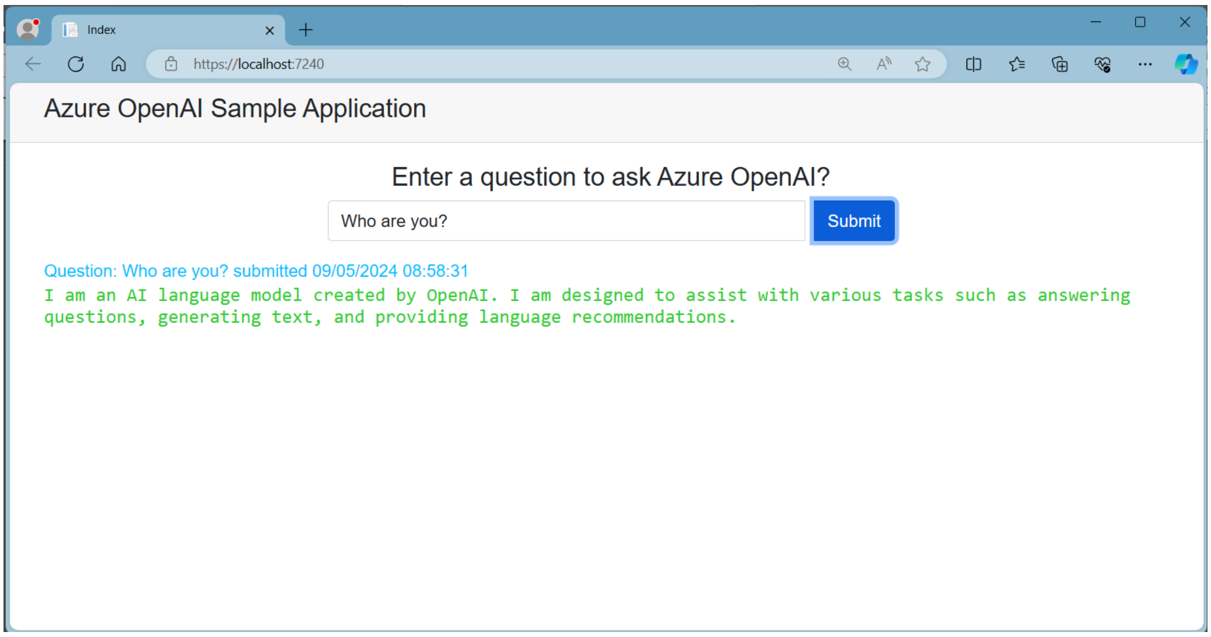

Index
https://localhost:7240
Azure OpenAI Sample Application
How can I develop a Blazor App?
Submit

```
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"
    }
```


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

```
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.
```