Chat Bot interface-final test

Let's break down and ensure the complete setup, including the necessary configuration and integration steps for the AI chat interface within the NexHomeAgent application.

1. Frontend: Blazor Component for Chat Interface

Create a new Blazor component named `AiChat.razor` for the chat interface.

AiChat.razor

```
@page "/chat"
@inject IChatService ChatService

<div class="chat-container">
    <div class="chat-header" @onclick="ToggleChat">
        <img src="avatar.png" alt="Chat Avatar" class="chat-a
vatar" />
        <span>How can I assist you today?</span>
    </div>
    @if (isChatOpen)
    {
        <div class="chat-body">
            <div class="chat-messages">
                @foreach (var message in messages)
                {
                    <div class="message @(message.IsUser ? "u
ser-message" : "ai-message")">
                        @message.Text
                    </div>
                }
            </div>
            <div class="chat-input">
                <InputText @bind-Value="userInput" placeholde
r="Type your message..." class="input-field" />
                <button @onclick="SendMessage" class="send-bu
```

```
tton">Send</button>
            </div>
        </div>
    }
</div>

@code {
    private bool isChatOpen = false;
    private string userInput = string.Empty;
    private List<ChatMessage> messages = new List<ChatMessage
>();

    private void ToggleChat()
    {
        isChatOpen = !isChatOpen;
    }

    private async Task SendMessage()
    {
        if (!string.IsNullOrWhiteSpace(userInput))
        {
            messages.Add(new ChatMessage { Text = userInput,
IsUser = true });
            var response = await ChatService.SendMessageAsync
(userInput);
            messages.Add(new ChatMessage { Text = response, I
sUser = false });
            userInput = string.Empty;
        }
    }

    public class ChatMessage
    {
        public string Text { get; set; }
        public bool IsUser { get; set; }
    }
}
```

2. Backend: Integration with OpenAI API

Ensure you have configured your OpenAI API key securely. Use Azure Key Vault or environment variables to store sensitive information.

ChatService.cs

```
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;
using System.Linq;

public interface IChatService
{
    Task<string> SendMessageAsync(string message);
}

public class ChatService : IChatService
{
    private readonly HttpClient _httpClient;

    public ChatService(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }

    public async Task<string> SendMessageAsync(string messag
e)
    {
        var response = await _httpClient.PostAsJsonAsync("<ht
tps://api.openai.com/v1/engines/davinci-codex/completions>",
new
        {
            prompt = message,
            max_tokens = 150
        });

        var result = await response.Content.ReadFromJsonAsync
<OpenAiResponse>();
        return result.Choices.FirstOrDefault()?.Text.Trim();
    }

    private class OpenAiResponse
    {
        public List<Choice> Choices { get; set; }
    }

    private class Choice
    {
        public string Text { get; set; }
    }
}
```

3. Configuration

appsettings.json

```
{
  "OpenAI": {
    "ApiKey": "YOUR_API_KEY"
  }
}
```

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHttpClient<IChatService, ChatService>(client
=>
```

```
    {
        client.BaseAddress = new Uri("<https://api.openai.co
m/v1/>");
        client.DefaultRequestHeaders.Add("Authorization", "Be
arer " + Configuration["OpenAI:ApiKey"]);
    });

    services.AddRazorPages();
    services.AddServerSideBlazor();
}
```

4. Styling (CSS)

wwwroot/css/site.css

```
.chat-container {
    position: fixed;
    bottom: 20px;
    right: 20px;
    width: 300px;
    background: rgba(255, 255, 255, 0.5);
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    overflow: hidden;
}

.chat-header {
    display: flex;
    align-items: center;
    padding: 10px;
    cursor: pointer;
}

.chat-avatar {
    width: 40px;
    height: 40px;
```

```css
    border-radius: 50%;
    margin-right: 10px;
}

.chat-body {
    max-height: 400px;
    overflow-y: auto;
}

.chat-messages {
    padding: 10px;
}

.message {
    padding: 8px 10px;
    margin: 5px 0;
    border-radius: 20px;
}

.user-message {
    background: #007bff;
    color: white;
    text-align: right;
}

.ai-message {
    background: #f1f1f1;
    text-align: left;
}

.chat-input {
    display: flex;
    padding: 10px;
    border-top: 1px solid #ddd;
}
```

```
.input-field {
    flex-grow: 1;
    border: none;
    padding: 10px;
    border-radius: 20px;
    margin-right: 10px;
}

.send-button {
    background: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 20px;
    cursor: pointer;
}
```

5. Azure Configuration

Azure Key Vault

1. **Create Azure Key Vault:**

   ```
   az keyvault create --name YourKeyVaultName --resource-grou
   p YourResourceGroup --location YourLocation
   ```

2. **Store OpenAI API Key:**

   ```
   az keyvault secret set --vault-name YourKeyVaultName --nam
   e OpenAI-ApiKey --value "YOUR_OPENAI_API_KEY"
   ```

3. **Configure Azure App Service to use Key Vault:**

   ```
   az webapp identity assign --resource-group YourResourceGro
   up --name NexHomeAgentApp
   ```

```
az keyvault set-policy --name YourKeyVaultName --resource-group YourResourceGroup --object-id <identityPrincipalId> --secret-permissions get
```

4. **Configure App Settings in Azure App Service:**

```
az webapp config appsettings set --resource-group YourResourceGroup --name NexHomeAgentApp --settings "KeyVaultName=YourKeyVaultName"
```

6. Conclusion

This setup provides a modern, user-friendly AI chat interface integrated with a lifelike avatar, using OpenAI for intelligent responses and incorporating the latest UI trends. This ensures that the NexHomeAgent application offers a state-of-the-art user experience. If you have any further requirements or need additional features, feel free to ask!

To run the NexHomeAgent application with the AI chat interface, you'll need to install several dependencies. These dependencies include libraries and packages for .NET, Blazor, and Azure integration. Here's a comprehensive list of the dependencies and libraries you'll need to install:

.NET Core SDK

First, ensure you have the .NET Core SDK installed. You can download it from the official .NET website.

NuGet Packages for Blazor

Install the necessary NuGet packages for Blazor and ASP.NET Core.

```
dotnet add package Microsoft.AspNetCore.Components.WebAssembly
dotnet add package Microsoft.AspNetCore.Components.WebAssembly.Authentication
dotnet add package Microsoft.AspNetCore.Components.WebAssembly.DevServer
dotnet add package Microsoft.Extensions.Http
dotnet add package Microsoft.Extensions.Configuration
dotnet add package Microsoft.Extensions.Configuration.Json
dotnet add package Microsoft.Extensions.Configuration.AzureKe
yVault
dotnet add package Microsoft.Azure.KeyVault
dotnet add package Microsoft.Azure.Services.AppAuthentication
dotnet add package System.Net.Http.Json
dotnet add package BlazorInputFile
dotnet add package Microsoft.AspNetCore.SignalR.Client
dotnet add package Microsoft.Identity.Web
```

Azure CLI

Install the Azure CLI to manage Azure resources and configure your application.

```
# Windows
Invoke-WebRequest -Uri <https://aka.ms/installazurecliwindows
> -OutFile .\\AzureCLI.msi; Start-Process msiexec.exe -Argume
ntList '/I AzureCLI.msi /quiet' -Wait

# macOS
brew update && brew install azure-cli

# Ubuntu
curl -sL <https://aka.ms/InstallAzureCLIDeb> | sudo bash
```

Setting up the Environment

Configure environment variables for your application. Create a `.env` file in the root directory of your project to store sensitive information like your OpenAI API key and Azure Key Vault details.

.env

```
OpenAI_ApiKey=YOUR_OPENAI_API_KEY
Azure_KeyVaultName=YourKeyVaultName
```

```
Azure_ClientId=YourAzureClientId
Azure_ClientSecret=YourAzureClientSecret
Azure_TenantId=YourAzureTenantId
```

Middleware Setup

Make sure your `Startup.cs` file includes the necessary middleware and services configuration:

Startup.cs

```
public class Startup
{
    public IConfiguration Configuration { get; }

    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public void ConfigureServices(IServiceCollection service
s)
    {
        services.AddHttpClient<IChatService, ChatService>(cli
ent =>
        {
            client.BaseAddress = new Uri("<https://api.opena
i.com/v1/>");
            client.DefaultRequestHeaders.Add("Authorization",
"Bearer " + Configuration["OpenAI:ApiKey"]);
        });

        services.AddAzureKeyVault(
            $"https://{Configuration["Azure_KeyVaultName"]}.v
ault.azure.net/",
            Configuration["Azure_ClientId"],
            Configuration["Azure_ClientSecret"]
        );

        services.AddRazorPages();
        services.AddServerSideBlazor();
    }

    public void Configure(IApplicationBuilder app, IWebHostEn
vironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Error");
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();
        app.UseRouting();

        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapBlazorHub();
            endpoints.MapFallbackToPage("/_Host");
        });
    }
}
```

Azure Key Vault Configuration

Ensure you have the Azure Key Vault properly configured to securely store and access sensitive data.

1. **Create Azure Key Vault:**

   ```
   az keyvault create --name YourKeyVaultName --resource-group YourResourceGroup --location YourLocation
   ```

2. **Store OpenAI API Key:**

   ```
   az keyvault secret set --vault-name YourKeyVaultName --name OpenAI-ApiKey --value "YOUR_OPENAI_API_KEY"
   ```

3. **Configure Azure App Service to use Key Vault:**

   ```
   az webapp identity assign --resource-group YourResourceGroup --name NexHomeAgentApp
   ```

   ```
   az keyvault set-policy --name YourKeyVaultName --resource-group YourResourceGroup --object-id <identityPrincipalId> --secret-permissions get
   ```

4. **Configure App Settings in Azure App Service:**

   ```
   az webapp config appsettings set --resource-group YourResourceGroup --name NexHomeAgentApp --settings "KeyVaultName=YourKeyVaultName"
   ```

Running the Application

Once all dependencies are installed and the environment is configured, you can run the application using the following command:

```
dotnet run
```

README File

Create a `README.md` file in your project root directory to provide instructions for setting up and running the application.

README.md

~~~markdown
# NexHomeAgent Application

## Overview

This project is a web application built using Blazor and .NET Core, integrated with OpenAI for an AI chat interface. The application leverages Azure services for secure key management and deployment.

## Prerequisites

- .NET Core SDK
- Azure CLI
- NuGet Packages (listed below)

## NuGet Packages

```bash
dotnet add package Microsoft.AspNetCore.Components.WebAssembly
dotnet add package Microsoft.AspNetCore.Components.WebAssembly.Authentication
dotnet add package Microsoft.AspNetCore.Components.WebAssembly.DevServer
dotnet add package Microsoft.Extensions.Http
dotnet add package Microsoft.Extensions.Configuration
dotnet add package Microsoft.Extensions.Configuration.Json
dotnet add package Microsoft.Extensions.Configuration.AzureKeyVault
dotnet add package Microsoft.Azure.KeyVault
~~~

```
dotnet add package Microsoft.Azure.Services.AppAuthentication
dotnet add package System.Net.Http.Json
dotnet add package BlazorInputFile
dotnet add package Microsoft.AspNetCore.SignalR.Client
dotnet add package Microsoft.Identity.Web
```

Environment Configuration

Create a `.env` file in the root directory with the following content:

```
OpenAI_ApiKey=YOUR_OPENAI_API_KEY
Azure_KeyVaultName=YourKeyVaultName
Azure_ClientId=YourAzureClientId
Azure_ClientSecret=YourAzureClientSecret
Azure_TenantId=YourAzureTenantId
```

Azure Key Vault Setup

```
# Create Azure Key Vault
az keyvault create --name YourKeyVaultName --resource-group Y
ourResourceGroup --location YourLocation

# Store OpenAI API Key
az keyvault secret set --vault-name YourKeyVaultName --name O
penAI-ApiKey --value "YOUR_OPENAI_API_KEY"

# Assign Managed Identity to App Service
az webapp identity assign --resource-group YourResourceGroup
--name NexHomeAgentApp

# Set Key Vault Policy
az keyvault set-policy --name YourKeyVaultName --resource-gro
up YourResourceGroup --object-id <identityPrincipalId> --secr
et-permissions get
```

```
# Configure App Settings in Azure App Service
az webapp config appsettings set --resource-group YourResourc
eGroup --name NexHomeAgentApp --settings "KeyVaultName=YourKe
yVaultName"
```

Running the Application

```
dotnet run
```

Contact

For any inquiries, please contact [Your Contact Information].

```

By following these instructions and installing the necessary
dependencies, you can successfully run the NexHomeAgent appli
cation on your local machine and ensure a seamless developmen
t experience.
```