CDW Documentation

Emotional support Bot with Text Analytics and OpenAI

Emotional support Bot with Text Analytics and OpenAI

This app is deployed in static webapp and uses node JS for managed Azure functions. Azure functions version 4 is used.

Below is the folder structure that needs to be deployed for Azure functions V4

screenshot_2025-07-20_at_11.08.02 pm.png

api/host.json

```
{
  "version": "2.0",
  "logging": {
    "applicationInsights": {
      "samplingSettings": {
        "isEnabled": true,
        "excludedTypes": "Request"
      }
    }
  },
  "extensionBundle": {
    "id": "Microsoft.Azure.Functions.ExtensionBundle",
    "version": "[3.*, 4.0.0)"
  },
  "functionApp": {
    "id": "emotional-support-bot"  // This is your entire app name
  }
}
```

api/package.json

```
{
  "name": "emotional-support-bot-api",
  "version": "1.0.0",
  "description": "Azure Functions for emotional support bot",
  "main": "src/functions/*.js",
  "scripts": {
    "start": "func start",
    "test": "echo \"No tests yet...\""
  },
  "dependencies": {
    "@azure/functions": "^4.0.0",
   "@azure/ai-text-analytics": "^5.1.0",
    "@azure/openai": "^2.0.0",
     "@azure/core-auth": "^1.5.0"
  },
```

```
  "devDependencies": {
    "azure-functions-core-tools": "^4.0.0"
  }
}
```

Please take a look at the section main. Main should contain the source to the node js function files which we need to run as part of static webapps

api/src/functions/analyze-sentiment.js

```
// src/functions/analyze-sentiment.js
const { app } = require('@azure/functions');

app.http('analyze-sentiment', {
    methods: ['POST'],
    authLevel: 'anonymous',
    route: 'analyze-sentiment',
    handler: async (request, context) => {
        context.log('Processing sentiment analysis request');
        try {
            const { text } = await request.json();
            if (!text) {
                return {
                    status: 400,
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ error: 'Text is required' })
                };
            }

            const endpoint = process.env.TEXT_ANALYTICS_ENDPOINT;
            const key = process.env.TEXT_ANALYTICS_KEY;
            // Logging for debugging
            context.log('Endpoint:', process.env.TEXT_ANALYTICS_ENDPOINT);
            context.log('Key exists:', !!process.env.TEXT_ANALYTICS_KEY);
            if (!endpoint || !key) {
                context.log.error('Text Analytics configuration missing');
                return {
                    status: 500,
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ error: 'Text Analytics
configuration missing' })
                };
            }

            const response = await
```

```
fetch(`${endpoint}/text/analytics/v3.1/sentiment`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'Ocp-Apim-Subscription-Key': key
                },
                body: JSON.stringify({
                    documents: [
                        {
                            id: '1',
                            text: text,
                            language: 'en'
                        }
                    ]
                })
            });

            if (!response.ok) {
                const errorText = await response.text();
                context.log.error(`Text Analytics API failed:
${response.status} - ${errorText}`);
                throw new Error(`Text Analytics API failed:
${response.status}`);
            }

            const data = await response.json();
            const sentimentResult = data.documents[0];

            return {
                status: 200,
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(sentimentResult)
            };

        } catch (error) {
            context.log.error('Error in analyze-sentiment:', error);
            return {
                status: 500,
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ error: 'Internal server error: ' +
error.message })
            };
        }
    }
});
```

api/src/functions/generate-response.js

```
// src/functions/generate-response.js
const { app } = require('@azure/functions');

function getSystemPrompt(sentiment) {
    const sentimentLabel = sentiment.sentiment;
    const confidence = sentiment.confidenceScores;

    let prompt = `You are an empathetic AI support assistant. The user's
message has been analyzed with sentiment: ${sentimentLabel} (positive:
${confidence.positive.toFixed(2)}, neutral:
${confidence.neutral.toFixed(2)}, negative:
${confidence.negative.toFixed(2)}).

Based on this sentiment analysis, provide appropriate emotional support:

`;

    if (sentimentLabel === 'positive') {
        prompt += `- The user seems to be feeling good. Acknowledge their
positive emotions and encourage them to continue.
- Be supportive and celebratory while staying genuine.
- Ask follow-up questions to understand what's going well.`;
    } else if (sentimentLabel === 'negative') {
        prompt += `- The user seems to be struggling or feeling down.
Provide compassionate support.
- Validate their feelings and offer comfort.
- Suggest coping strategies or gentle encouragement.
- Be careful not to dismiss their concerns.`;
    } else {
        prompt += `- The user's sentiment is neutral. Provide balanced
support.
- Try to understand more about their situation.
- Offer helpful guidance without making assumptions.`;
    }

    prompt += `\n\nKeep responses concise (2-3 sentences), warm, and
genuinely supportive. Focus on being helpful rather than clinical.`;

    return prompt;
}

app.http('generate-response', {
    methods: ['POST'],
    authLevel: 'anonymous',
    route: 'generate-response',
    handler: async (request, context) => {
        context.log('Processing response generation request');
        try {
            const { messages, sentiment } = await request.json();
            if (!messages || !sentiment) {
                return {
```

```
                    status: 400,
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ error: 'Messages and sentiment
are required' })
                };
            }

            const endpoint = process.env.AZURE_OPENAI_ENDPOINT;
            const key = process.env.AZURE_OPENAI_KEY;
            const deploymentName = process.env.AZURE_OPENAI_DEPLOYMENT_NAME
|| 'gpt-35-turbo';
            const apiVersion = process.env.AZURE_OPENAI_API_VERSION ||
'2024-02-01';

            // Debug logging
            context.log('Azure OpenAI Configuration:');
            context.log('Endpoint:', endpoint);
            context.log('Deployment Name:', deploymentName);
            context.log('API Version:', apiVersion);
            context.log('Key exists:', !!key);

            if (!endpoint || !key) {
                context.error('Azure OpenAI configuration missing');
                return {
                    status: 500,
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ error: 'Azure OpenAI
configuration missing' })
                };
            }

            const systemPrompt = getSystemPrompt(sentiment);
            const fullMessages = [
                { role: 'system', content: systemPrompt },
                ...messages
            ];

            context.log('Sending request to Azure OpenAI...');

            // Use REST API directly to avoid SDK issues
            const url =
`${endpoint}/openai/deployments/${deploymentName}/chat/completions?api-
version=${apiVersion}`;
            const response = await fetch(url, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
```

```
                    'api-key': key
                },
                body: JSON.stringify({
                    messages: fullMessages,
                    max_tokens: 300,
                    temperature: 0.7
                })
            });

            if (!response.ok) {
                const errorText = await response.text();
                throw new Error(`Azure OpenAI API error: ${response.status}
${response.statusText} - ${errorText}`);
            }

            const data = await response.json();

            if (!data.choices || data.choices.length === 0) {
                throw new Error('No response from Azure OpenAI');
            }

            const generatedResponse = data.choices[0].message.content;

            context.log('Successfully generated response');

            return {
                status: 200,
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ content: generatedResponse })
            };

        } catch (error) {
            context.error('Error in generate-response:', error);
            // More detailed error logging
            if (error.code) {
                context.error('Error code:', error.code);
            }
            if (error.message) {
                context.error('Error message:', error.message);
            }
            if (error.stack) {
                context.error('Error stack:', error.stack);
            }
            return {
                status: 500,
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
```

```
                    error: 'Internal server error: ' + error.message,
                    code: error.code || 'unknown'
                })
            };
        }
    }
});
```