Since you're new to Azure and need a detailed step-by-step guide to implement your requirement, I'll walk you through building a system where a customer calls a number, interacts with a voice that feels human (using Azure's text-to-speech and speech-to-text services), provides their first name, last name, and monthly income, and receives a quotation status based on a Java API response. For this proof of concept (POC), we'll simulate the Java API with an Azure Function. I'll also explain how to test it thoroughly.

Overview of the Solution

- **Telephony**: Azure Communication Services (ACS) will handle incoming calls.
- **Speech Services**: Azure Cognitive Services will enable speech-to-text (to capture customer responses) and text-to-speech (to respond naturally).
- **Conversation Logic**: Azure Bot Framework will manage the interaction, collect data, and call the API.
- **API**: A mock Azure Function will simulate your Java quotation management API.

Step-by-Step Implementation

Step 1: Set Up Azure Resources

You'll need to create resources in the Azure portal.

1. **Create a Speech Resource**:
   - Go to the Azure portal.
   - Click **Create a resource** > Search for "Speech" > Select **Speech** (under Azure AI

Click **Create a resource** > Search for "Speech" > Select **Speech** (under Azure AI services).

- Fill in:
  - Subscription: Your Azure subscription.
  - Resource group: Create a new one (e.g., `VoiceQuotationPOC`) or use an existing one.
  - Region: Choose a region close to you (e.g., East US).
  - Name: `SpeechServicePOC`.
  - Pricing tier: Select **Free F0** for the POC.
- Click **Review + create** > **Create**.
- After deployment, go to the resource > **Keys and Endpoint**. Note the **Key 1** and **Region** (e.g., `eastus`).

2 **Create an Azure Communication Services Resource**:

- In the Azure portal, click **Create a resource** > Search for "Communication Services" > Select **Communication Services**.
- Fill in the details and create the resource.
- After deployment, go to the resource > **Phone numbers** > **Get**.
- Acquire a phone number (e.g., a toll-free number). Note this number (e.g., `+1-800-555-1234`).

3 **Create an Azure Bot**:

- In the Azure portal, click **Create a resource** > Search for "Azure Bot" > Select **Azure Bot**.

- Fill in:
  - Bot handle: `QuotationBot`.
  - Subscription and resource group: Same as above.
  - Template: Choose **Echo Bot** (we'll modify it later).
  - Microsoft App ID: Select **Create new**.
- Click **Review + create** > **Create**.
- After deployment, go to the bot > **Settings** > Note the **Microsoft App ID** and generate a **Password**.

4 **Create an Azure Function (Mock API)**:

- In the Azure portal, click **Create a resource** > Search for "Function App" > Select **Function App**.
- Fill in:
  - App name: `QuotationMockAPI`.
  - Runtime stack: **.NET**.
- Click **Review + create** > **Create**.
- After deployment, go to the Function App > **Functions** > **+ Create** > Select **HTTP trigger**.
- Name it `QuotationAPI`, set authorization to **Anonymous**, and create.
- Replace the default code with:

```csharp
using System.IO;
```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;

namespace QuotationMockAPI
{
    public static class QuotationAPI
    {
        [FunctionName("QuotationAPI")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = null
        {
            string requestBody = await new StreamReader(req.Body).ReadToEnd
            // For POC, always return "accepted" as a sample response
            var response = new { status = "accepted" };
            return new OkObjectResult(response);
        }
    }
}
```

Show in sidebar

- Save and click **Get function URL**. Note the URL (e.g., `https://quotationmockapi.azurewebsites.net/api/QuotationAPI`).

Step 2: Configure the Bot for Speech

1. **Enable Direct Line Speech**:
   - Go to your Azure Bot resource > **Channels** > Select **Direct Line Speech**.
   - Connect it to your Speech resource using the **Key** and **Region** from Step 1.
   - Save the configuration.

Step 3: Implement the Bot Logic

1. **Download and Modify the Bot Code**:

- In the Azure Bot resource, go to **Build** > **Download Bot source code**.
- Open the project in Visual Studio.
- Update `EchoBot.cs` (or equivalent) to handle the conversation flow:
  - Add state management to track responses.
  - Call the mock API with collected data.
- Example code (simplified):

csharp

```
using Microsoft.Bot.Builder;
using Microsoft.Bot.Schema;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

public class QuotationBot : ActivityHandler
{
    private readonly IStatePropertyAccessor<Dictionary<string, string>> _s
    private readonly BotState _conversationState;

    public QuotationBot(ConversationState conversationState)
    {
        _conversationState = conversationState;
        _stateAccessor = conversationState.CreateProperty<Dictionary<strin
    }

    protected override async Task OnMessageActivityAsync(ITurnContext<IMes
    {
        var state = await _stateAccessor.GetAsync(turnContext, () => new D

        if (!state.ContainsKey("firstName"))
        {
            await turnContext.SendActivityAsync("Hello, thank you for call
            state["firstName"] = turnContext.Activity.Text;
        }
        else if (!state.ContainsKey("lastName"))
```

```
        {
            await turnContext.SendActivityAsync($"Thank you, {state["first
            state["lastName"] = turnContext.Activity.Text;
        }
        else if (!state.ContainsKey("income"))
        {
            await turnContext.SendActivityAsync($"Great, {state["firstName
            state["income"] = turnContext.Activity.Text;
        }
        else
        {
            var jsonRequest = new { firstName = state["firstName"], lastNa
            var response = await CallQuotationAPI(jsonRequest);
            var status = response["status"];
            var message = status == "accepted" ? "Your quotation has been
                          status == "referred" ? "Your quotation has been
                          "Unfortunately, your quotation has been declined
            await turnContext.SendActivityAsync(message, cancellationToken
            state.Clear();
        }

        await _conversationState.SaveChangesAsync(turnContext, false, canc
    }

    private async Task<Dictionary<string, string>> CallQuotationAPI(object
    {
        using var client = new HttpClient();
        var content = new StringContent(Json.SerializeObject(jsonRequest),
        var response = await client.PostAsync("https://quotationmockapi.az
        var responseString = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject<Dictionary<string, string>>(r
    }
}
```

- Replace the URL in `CallQuotationAPI` with your Azure Function URL.

2 **Deploy the Bot**:

- In Visual Studio, right-click the project > **Publish** > Select your Azure Bot resource > Publish.

Step 4: Connect the Bot to ACS

1. **Route Calls to the Bot**:
   - In your ACS resource, go to **Phone numbers** > Select your number.
   - Configure it to route calls to your bot (follow ACS documentation for bot integration).

Step 5: Test the Solution

1. **Preparation**:
   - Ensure all resources are deployed and connected.
   - Use a phone to call the ACS phone number.

Use a phone to call the ACS phone number.

2 **Test Interaction**:

- **Step 1**: Call the number.
- **Step 2**: When prompted, say your first name (e.g., "John").
- **Step 3**: Say your last name (e.g., "Doe").
- **Step 4**: Say your monthly income (e.g., "5000").
- **Step 5**: Listen for the response (e.g., "Your quotation has been accepted.").

3 **Gather POC Evidence**:

- **Logs**: Add logging to the bot (e.g., using `Console.WriteLine` or Application Insights) to capture transcribed inputs and responses.
- **Recording**: Use a softphone app (e.g., Zoom Phone) to record the call if possible.
- **Verification**: Check logs in Azure (via Application Insights) to confirm:
  - Inputs were captured correctly (e.g., "John", "Doe", "5000").
  - API was called with the correct JSON (e.g., `{"firstName": "John", "lastName": "Doe", "income": "5000"}`).
  - Response was spoken back accurately.

4 **Document Results**:

- Save logs as text files or screenshots.
- Note any issues (e.g., speech recognition errors) and how they were handled.

Tips for Success

- **Speech Accuracy**: Test with clear speech. For POC, assume standard inputs.
- **Natural Voice**: Use a neural voice in the Speech resource settings (e.g., "en-US-JennyNeural") for a human-like feel.
- **Privacy**: Don’t store data; process it in memory.

This guide should help you build and test your POC successfully! Let me know if you need further clarification.