

As a Microsoft Azure Specialist, I'll guide you through creating a voice-interactive solution using Azure Bot Services, integrated with Alexa, to manage customer interactions with your Quotation Management API. This solution will use Azure's Speech-to-Text and Text-to-Speech services to make the customer feel they're talking to a person. We'll use Node.js for development and the Azure Portal (no CLI) for setup. I'll also explain how to mock the solution for a demo. Since you're new to Azure, I'll provide detailed, step-by-step instructions.

Solution Overview

We're building a system where customers can:

- Call a phone number or use Alexa.
- Interact via voice, providing their first name, last name, and monthly income.
- Confirm their details.
- Have the system call your Quotation Management API and receive a response (accept, decline, or refer) for a card, communicated back via speech.

Components

- **Azure Bot Services:** Manages the conversation flow.
 - **Azure Cognitive Services (Speech):** Handles Speech-to-Text (customer input) and Text-to-Speech (bot responses).
 - **Alexa Integration:** Connects the bot to Alexa as a channel.
 - **Node.js:** Used to code the bot logic.
 - **Quotation Management API:** An existing API you provide, which we'll call with customer data.
-

Step-by-Step Solution

Step 1: Set Up Azure Bot Services in the Azure Portal

¹. Sign In to Azure Portal:

- Go to portal.azure.com and log in with your Azure account.

2 Create a New Azure Bot:

- Click **Create a resource** (the “+” icon) in the top-left corner.
- Search for **Azure Bot** in the search bar and select it.
- Click **Create**.
- Fill in the details:
 - **Bot handle:** Enter a unique name (e.g., QuotationBot).
 - **Subscription:** Select your Azure subscription.
 - **Resource group:** Click **Create new** and name it (e.g., QuotationBotGroup) or use an existing one.
 - **Location:** Choose a region close to your users (e.g., East US).
 - **Pricing tier:** Select **F0 (Free)** for testing or **S1 (Standard)** for production.
 - **Microsoft App ID:** Choose **Create new Microsoft App ID** (Azure will generate this for you).
- Click **Review + Create**, review the details, then click **Create**.
- Wait a few minutes for the bot to deploy.

3 Access Your Bot:

- Once deployed, click **Go to resource** on the deployment confirmation page.
 - You’ll see your bot’s dashboard. Note the **Messaging endpoint** under **Settings** (it’ll look like <https://<bot-name>.azurewebsites.net/api/messages>). You’ll need this later.
-

Step 2: Develop the Bot Logic Using Node.js

1 Set Up Your Development Environment:

- Download and install Node.js (version 12 or higher) on your computer.
-  

Open a terminal (e.g., Command Prompt or VS Code terminal) and install the Bot Framework SDK by running:

```
text
```

X ⌂ ⌓

```
npm install botbuilder
```

- Install the Azure Speech SDK for speech capabilities:

```
text
```

X ⌂ ⌓

```
npm install microsoft-cognitiveservices-speech-sdk
```

2 Create a Basic Bot Project:

- In your terminal, create a new folder for your project:

```
text
```

X ⌂ ⌓

```
mkdir quotation-bot  
cd quotation-bot  
npm init -y
```

- Install additional dependencies:

```
text
```

X ⌂ ⌓

```
npm install restify dotenv
```

3 Write the Bot Code:

- Create a file named `index.js` in your `quotation-bot` folder.
- Add the following code to handle the conversation flow, speech integration, and API calls:



index.js
javascript

Show inline

4 Configure Environment Variables:

- Create a `.env` file in your `quotation-bot` folder:

```
text
```

```
MICROSOFT_APP_ID=<your-app-id-from-azure>
MICROSOFT_APP_PASSWORD=<your-app-password-from-azure>
SPEECH_KEY=<your-speech-service-key>
SPEECH_REGION=<your-speech-service-region>
```

- To get these values:

- **App ID and Password:** In the Azure Portal, go to your bot > **Settings** > **Configuration** > Copy the **Microsoft App ID** and generate a password.
- **Speech Key and Region:** Create a Speech Service resource in the Azure Portal (search for **Speech** under **Create a resource**), then go to its **Keys and Endpoint** page to copy the key and region.

Step 3: Integrate with Alexa Using the Azure Portal

1. Add Alexa Channel:

- In the Azure Portal, go to your bot's resource page.
- In the left menu, click **Channels and Content**.

In the left menu, click **Channels** under **Settings**.

- Click the **Alexa** icon in the list of available channels.
- Follow the prompts:
 - You'll need an Amazon Developer account. Sign in or create one at developer.amazon.com.
 - Copy the **Messaging endpoint** from your bot's **Settings** page (e.g., `https://<bot-name>.azurewebsites.net/api/messages`).
 - In the Alexa configuration window, paste this endpoint and complete the setup as guided.

2 Configure Alexa Skill:

- Go to the [Alexa Developer Console](#).
- Click **Create Skill**:
 - Name: e.g., Quotation Assistant.
 - Choose **Custom model** and **Provision your own hosting**.
 - Click **Create Skill**.
- Under **Endpoint**, select **HTTPS** and paste your bot's messaging endpoint.
- Save and build the skill.

3 Test Alexa Integration:

- In the Alexa Developer Console, go to the **Test** tab.
- Enable testing and say, "Open Quotation Assistant" to simulate the interaction.

Step 4: Mock the Solution for Demo Purposes

1. Mock the Quotation Management API:

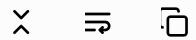
- For demo purposes, the `callQuotationApi` function in the `index.js` code above is already mocked to return `accept` or `reject` based on income. Replace it with your

already mocked to return `'accept'` or `'refer'` based on income. Replace it with your real API call later.

- Alternatively, create a simple Node.js server:

- Create a file `mock-api.js`:

text



```
const express = require('express');
const app = express();
app.use(express.json());
app.post('/api/quotation', (req, res) => {
  const income = req.body.monthlyIncome;
  res.json({ status: income > 5000 ? 'accept' : income > 3000 ? 'refer' });
});
app.listen(3000, () => console.log('Mock API running on port 3000'));
```

- Install Express: `npm install express`.
 - Run it: `node mock-api.js`.
 - Update `callQuotationApi` in `index.js` to call `http://localhost:3000/api/quotation`.

2 Test Locally:

- Run your bot locally: `node index.js`.
- Use the Bot Framework Emulator to test the conversation flow (connect to `http://localhost:3978/api/messages`).
- Use the Alexa Developer Console's simulator for voice testing.

Step 5: Deploy and Test

1 Deploy the Bot to Azure:

- In your bot folder, create a zip file of all files (`index.js`, `package.json`, etc.).
- In the Azure Portal, create an App Service.

IN THE AZURE PORTAL, CREATE AN APP SERVICE.

- Click **Create a resource** > Search for **App Service** > **Create**.
- Fill in details (e.g., name: `QuotationBotApp`, runtime: `Node.js`).
- Deploy your zip file via the **Deployment Center** (select **Local Git or FTP**).
- Update your bot's **Messaging endpoint** in the Azure Portal to the App Service URL (e.g., `https://QuotationBotApp.azurewebsites.net/api/messages`).

2 Test the Entire Flow:

- Use Alexa: Say, "Open Quotation Assistant" in the simulator or on an Alexa device.
- Verify that:
 - Speech-to-Text captures your voice input correctly.
 - Text-to-Speech responds naturally.
 - The bot asks for and confirms your details, calls the API, and provides the response.

Final Notes

- Replace the mock API call with your real Quotation Management API endpoint when ready.
- For phone number integration, consider adding a telephony channel (e.g., Twilio) via Azure Bot Services' **Channels** section, though this requires additional setup.
- Explore the Azure Bot Services documentation and Speech SDK documentation for advanced features.

This solution provides a fully functional, voice-interactive system using Azure and Node.js, tailored to your needs as a beginner! Let me know if you need further clarification.