

```
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa2c37f-9c6f-4aa9-9f55-5f5e6c8ee36c", "request_id": "8dlbafc70d8f4c43-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa2c37f-9c6f-4aa9-9f55-5f5e6c8ee36c", "children": ["aaa23f5e-8156-40d0-905f-
096d7470e39a"]}, "aaa23f5e-8156-40d0-905f-096d7470e39a": {"id": "aaa23f5e-8156-40d0-
905f-096d7470e39a", "message": {"id": "aaa23f5e-8156-40d0-905f-096d7470e39a", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728784416.948072,
"update_time": null, "content": {"content_type": "text", "parts": ["how to use async
function on developer console? "]}, "status": "finished_successfully", "end_turn":
null, "weight": 1.0, "metadata": {"serialization_metadata": {"custom_symbol_offsets":
[]}}, "request_id": "8dlbb26bba134c43-MXP", "message_source": null, "timestamp_":
"absolute", "message_type": null}, "recipient": "all", "channel": null}, "parent":
"bf426d95-2252-45be-a9c6-500ad8d6c415", "children": ["558d5860-0802-434a-b927-
686d41f1515b"]}, "558d5860-0802-434a-b927-686d41f1515b": {"id": "558d5860-0802-434a-
b927-686d41f1515b", "message": {"id": "558d5860-0802-434a-b927-686d41f1515b", "author":
{"role": "assistant", "name": null, "metadata": {}}, "create_time": 1728784458.325775,
"update_time": null, "content": {"content_type": "text", "parts": ["Using `async`
functions in the Alexa Developer Console for your skill is possible, and the console
fully supports handling asynchronous code such as API calls. Here's a detailed
breakdown of how you can use `async` functions, especially when dealing with things
like HTTP requests or any asynchronous tasks like calling the OpenAI API.\n\n### Key
Concepts of `async` and `await` in JavaScript\n- async` functions: These are
functions that return a `Promise`. Using `async` allows you to write asynchronous code
that looks like synchronous code, which is more readable.\n- await`: Inside an
`async` function, you can use `await` to pause the function execution until a
`Promise` is resolved. This is commonly used when making API calls or doing any
asynchronous work.\n\n### Steps to Use `async/await` in Your Alexa Skill:\n\n1. Make
the Handler Function `async`\n To use `await` inside your intent handler, you must
declare the handler as an `async` function. This allows you to wait for the completion
of asynchronous operations (like an API call).\n\n Here's an example of how to
declare an `async` handler:\n\n ```javascript\n const GetAIResponseIntentHandler =
{\n     canHandle(handlerInput) {\n         return
Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest\n
&& Alexa.getIntentName(handlerInput.requestEnvelope) ===
GetAIResponseIntent\n     },\n     async handle(handlerInput) { //
Mark the handler function as async\n         const { intent } =
handlerInput.requestEnvelope.request;\n         // Example async operation: Check
if the slot is filled\n         if (!intent.slots.query.value) {\n
const speakOutput = "I didn't catch that. Can you please ask me a question
like, 'What is quantum mechanics?'\n         return
handlerInput.responseBuilder\n             .speak(speakOutput)\n
             .reprompt(speakOutput)\n             .getResponse();\n         }\n\n
let aiResponse = \n         try {\n             // Await the result
of an asynchronous API call\n             console.log('Sending query to
OpenAI:', intent.slots.query.value);\n             aiResponse = await
getAIResponse(intent.slots.query.value); // Await the API call result\n
console.log('AI response received:', aiResponse);\n         } catch (error)
```

```

{\n
    console.log(&#x27;Error occurred while calling OpenAI API:&#x27;,
error);\n
    aiResponse = &#x27;Sorry, I had trouble processing your
request.&#x27;;\n
    }\n\n
    // Return the AI response or fallback
message\n
    return
handlerInput.responseBuilder\n
    .speak(aiResponse)\n
    .getRe
sponse();\n
    }\n
};\n
```\n\n2. Create an `async` Helper Function\n
If
you're making an API call to OpenAI or any external service, that function needs
to be asynchronous as well. Here\u2019s how you would structure the API request using
`async/await`:\n\n
```javascript\n
const https = require(&#x27;https&#x27;);\n\n
async function getAIResponse(query) {\n
    const apiKey = &#x27;your-openai-api-
key&#x27;; // Use your OpenAI API key here\n
    const options = {\n
hostname: &#x27;api.openai.com&#x27;;\n
    path: &#x27;/v1/completions&#x27;;\n
method: &#x27;POST&#x27;;\n
    headers: {\n
&#x27;Authorization&#x27;: `Bearer ${apiKey}`, \n
    &#x27;Content-
Type&#x27;: &#x27;application/json&#x27;;\n
    }\n
};\n\n
    const body
= JSON.stringify({\n
        model: &#x27;text-davinci-003&#x27;; // Replace with
your model\n
        prompt: query,\n
        max_tokens: 100,\n
temperature: 0.7\n
    });\n\n
    // Use async/await for the API request\n
    return new Promise((resolve, reject) => {\n
        const req =
https.request(options, (res) => {\n
            let data = &#x27;&#x27;;\n
            res.on(&#x27;data&#x27;, (chunk) => {\n
                data +=
chunk;\n
            });\n
            res.on(&#x27;end&#x27;, () => {\n
                console.log(&#x27;Received response from OpenAI:&#x27;, data);\n
                const response = JSON.parse(data);\n
                resolve(response.choices[0].text.trim());\n
            });\n
        });\n
        req.on(&#x27;error&#x27;, (e) => {\n
            console.log(&#x27;API request
error:&#x27;, e);\n
            reject(e);\n
        });\n
        req.write(body);\n
        req.end();\n
    });\n
}\n
```\n\n
- async
function getAIResponse: This is the function that makes an external API call using
Node.js's `https` module. It returns a `Promise`, which is why you can use
`await` when calling it.\n
- await inside the handler: When calling
`getAIResponse`, use `await` to pause execution until the API response is
received.\n\n
3. Deploying and Testing the Skill\n
Once you've made the
changes to the skill (including adding `async` and `await`), you can:\n
- Save and
Deploy your code in the Alexa Developer Console (in the Code section).\n
- Go
to the Test tab in the Alexa Developer Console and simulate a user request to test
the full flow of `async` operations.\n\n
You should see logs and outputs in the
console to verify if the API request and responses are working as expected.\n\n
Important Points to Consider:\n
- Error Handling: Always include `try-catch` blocks
around `await` statements to handle any errors that may occur during asynchronous
operations (e.g., API failures, timeouts).\n
- Timeouts: If your external API call
takes too long, Alexa might timeout (around 8 seconds is typical), so ensure your API
responses are fast or handle timeouts gracefully.\n\n
Example Flow of Asynchronous
Handling:\n\n
1. The skill receives a user query.\n
2. The `GetAIResponseIntentHandler`
is triggered and marked `async` to handle asynchronous operations.\n
3. Inside the
handler, the API request is made asynchronously, and the function waits for the result
with `await`.\n
4. Once the API response is received, it is used to construct the Alexa
response.\n\n
Example Test Query:\n
You can now test it with queries like:\n
`"Alexa, ask [your skill name] to tell me about quantum mechanics."`

```

"Alexa, open [your skill name] and ask it what is AI."\n\nThe responses from the OpenAI API will be awaited, processed, and then sent back to Alexa.\n\nBy using `async` and `await`, you can smoothly handle external API calls in your Alexa skill, making the experience more seamless for users. Let me know if you need further clarification or help with this!"]}, "status": "finished\_successfully", "end\_turn": true

```
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id": null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o", "parent_id": "aaa23f5e-8156-40d0-905f-096d7470e39a", "request_id": "8d1bb26bba134c43-MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens": [200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent": "aaa23f5e-8156-40d0-905f-096d7470e39a", "children": ["aaa21479-5bf7-4ea6-97b5-8b2d34faff1f"]}, "aaa21479-5bf7-4ea6-97b5-8b2d34faff1f": {"id": "aaa21479-5bf7-4ea6-97b5-8b2d34faff1f", "message": {"id": "aaa21479-5bf7-4ea6-97b5-8b2d34faff1f", "author": {"role": "user", "name": null, "metadata": {}}, "create_time": 1728784835.183896, "update_time": null, "content": {"content_type": "text", "parts": ["so this is what my code looks like now: \n\n/* \n * This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v2).\n * Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,\n * session persistence, api calls, and more.\n */\n\nconst Alexa = require('ask-sdk-core');\n\nconst https = require('https'); // Required for making the OpenAI API request\n\n// Function to send the query to OpenAI API and get a response\nfunction getAIResponse(query) {\n const apiKey = '{sk-proj-bXGBo8nQV9Fij1AKg54pPGWgwp2E0QENPBonydsLFwQ5WsGaipn5tsKqqunCoCCFK6b21n38N6T3B1bkFJA vD0VF30JxG47HmdDB0msTfHvm-eHsxrYbDYluYxXyR1Bv4D40RJg1PFE5X58Ea1bzQSSW0QoA}'; // <<< FIND AND REPLACE THIS with your actual OpenAI API key\n const options = {\n hostname: 'api.openai.com';,\n path: '/v1/completions';,\n method: 'POST';,\n headers: {\n 'Authorization': `Bearer ${apiKey}`,\n 'Content-Type': 'application/json';\n }\n };\n const body = JSON.stringify({\n model: 'text-davinci-003'; // Adjust if needed for other models\n prompt: query,\n max_tokens: 100,\n temperature: 0.7\n });\n return new Promise((resolve, reject) => {\n const req = https.request(options, (res) => {\n let data = '';\n res.on('data', (chunk) => {\n data += chunk;\n });\n res.on('end', () => {\n const response = JSON.parse(data);\n resolve(response.choices[0].text.trim());\n });\n });\n req.on('error', (e) => {\n reject(e);\n });\n req.write(body);\n req.end();\n });\n}\n\n// New Intent Handler to call OpenAI API and get the response\nconst GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'\n && Alexa.getIntentName(handlerInput.requestEnvelope) === 'GetAIResponseIntent';\n },\n async handle(handlerInput) {\n const { intent } = handlerInput.requestEnvelope.request;\n // Log the received intent and slots\n console.log('Received Intent:', intent);\n // Check if the slot is filled\n if
```

```

(!intent.slots.query.value) {\n const speakOutput = "I didn't
catch that. Can you please ask me a question like, 'What is quantum
mechanics?'";\n console.log('Slot value is empty.');\n
return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n\n let
aiResponse = '';\n try {\n // Log the query being sent to
OpenAI\n console.log('Sending query to OpenAI:';
intent.slots.query.value);\n aiResponse = await
getAIResponse(intent.slots.query.value); // Call OpenAI API\n
console.log('AI response received:';, aiResponse);\n } catch (error)
{\n console.log('Error occurred while calling OpenAI API:';,
error);\n aiResponse = 'Sorry, I had trouble processing your
request.';\n }\n\n // Return the AI response or error message\n
return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'\n
&& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'GetAIResponseIntent';\n },\n async handle(handlerInput) {\n
const { intent } = handlerInput.requestEnvelope.request;\n \n // Check
if the slot is filled\n if (intent.slots.query.value === undefined ||
intent.slots.query.value === "")\n {\n const speakOutput = "I
didn't catch that. Can you please ask me a question like, 'What is quantum
mechanics?'";\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n\n let
aiResponse = '';\n try {\n aiResponse = await
getAIResponse(intent.slots.query.value); // Call OpenAI API\n
console.log('AI response:';, aiResponse); // Log AI response for
debugging\n } catch (error) {\n console.log('Error
occurred:';, error); // Log error for debugging\n aiResponse =
'Sorry, I had trouble processing your request.';\n }\n\n //
Return the AI response or error message\n return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'\n
&& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'GetAIResponseIntent';\n },\n async handle(handlerInput) {\n
const { intent } = handlerInput.requestEnvelope.request;\n \n // Check
if all slots are filled\n if (intent.slots.query.value === undefined ||
intent.slots.query.value === "")\n {\n const speakOutput = "I
didn't catch that. Can you please ask me a question like, 'What is quantum
mechanics?'";\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n\n \n //
Ensure no delegation\n handlerInput.requestEnvelope.request.dialogState =
'COMPLETED';\n let aiResponse = '';\n try {\n
aiResponse = await getAIResponse(intent.slots.query.value); // Call OpenAI API

```

```

here\n } catch (error) {\n console.log(`Error with OpenAI API call:
${error}`);\n aiResponse = 'Sorry, I had trouble processing
that.';\n }\n\n return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n};\n\n\nconst LaunchRequestHandler = {\n canHandle(handlerInput) {\n\nreturn Alexa.getRequestType(handlerInput.requestEnvelope) ===
'LaunchRequest';\n },\n handle(handlerInput) {\n const
speakOutput = 'Welcome, you can say Hello or Help. Which would you like to
try?';\n\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .reprompt(s
peakOutput)\n .getResponse();\n }\n};\n\n\nconst HelloWorldIntentHandler =
{\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';\n
& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'HelloWorldIntent';\n },\n handle(handlerInput) {\n const
speakOutput = 'Hello World!';\n\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n
//.reprompt('add a reprompt if you want to keep the session open for the user to
respond');\n .getResponse();\n }\n};\n\n\nconst HelpIntentHandler =
{\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';\n
& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'AMAZON.HelpIntent';\n },\n handle(handlerInput) {\n const
speakOutput = 'You can say hello to me! How can I help?';\n\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .reprompt(s
peakOutput)\n .getResponse();\n }\n};\n\n\nconst
CancelAndStopIntentHandler = {\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';\n
& (Alexa.getIntentName(handlerInput.requestEnvelope) ===
'AMAZON.CancelIntent';\n ||
Alexa.getIntentName(handlerInput.requestEnvelope) ===
'AMAZON.StopIntent');\n },\n handle(handlerInput) {\n const
speakOutput = 'Goodbye!';\n\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .getRespons
e();\n }\n};\n\n\n/* * * FallbackIntent triggers when a customer says something that
doesn\u2019t map to any intents in your skill\n * It must also be defined in the
language model (if the locale supports it)\n * This handler can be safely added but
will be ignored in locales that do not support it yet\n * */\n\nconst
FallbackIntentHandler = {\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';\n
& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'AMAZON.FallbackIntent';\n },\n handle(handlerInput) {\n const
speakOutput = 'Sorry, I don\\'t know about that. Please try
again.';\n\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .reprompt(s
peakOutput)\n .getResponse();\n }\n};\n\n\n/* * * SessionEndedRequest
notifies that a session was ended. This handler will be triggered when a currently
open\n * session is closed for one of the following reasons: 1) The user says
"exit" or "quit";. 2) The user does not\n * respond or says

```

something that does not match an intent defined in your voice model. 3) An error occurs

```

const SessionEndedRequestHandler = {
 canHandle(handlerInput) {
 return Alexa.getRequestType(handlerInput.requestEnvelope) ===
 SessionEndedRequest;
 },
 handle(handlerInput) {
 console.log(`~~~~ Session ended: ${JSON.stringify(handlerInput.requestEnvelope)}`);
 // Any cleanup logic goes here.
 return handlerInput.responseBuilder.getResponse(); // notice we send an empty
 response;
 }
};

// The intent reflector is used for interaction model
// testing and debugging. It will simply repeat the intent the user said. You can
// create custom handlers for your intents by defining them above, then also adding
// them to the request handler chain below
const IntentReflectorHandler = {
 canHandle(handlerInput) {
 return Alexa.getRequestType(handlerInput.requestEnvelope) ===
 IntentRequest;
 },
 handle(handlerInput) {
 const intentName = Alexa.getIntentName(handlerInput.requestEnvelope);
 const speakOutput = `You just triggered ${intentName}`;
 return handlerInput.responseBuilder
 .speak(speakOutput)
 .reprompt(speakOutput)
 .getResponse();
 }
};

// add a reprompt if you want to keep the session open for the user to
// respond
const GenericErrorHandling = {
 canHandle(handlerInput) {
 return true;
 },
 handle(handlerInput, error) {
 const speakOutput = `Sorry, I had trouble doing what you asked. Please try again.`;
 console.log(`~~~~ Error handled: ${JSON.stringify(error)}`);
 return handlerInput.responseBuilder
 .speak(speakOutput)
 .reprompt(speakOutput)
 .getResponse();
 }
};

// This handler acts as the entry point for your skill, routing all request and response
// payloads to the handlers above. Make sure any new handlers or interceptors
// you define are included below. The order matters - they're processed top to
// bottom
exports.handler = Alexa.SkillBuilders.custom()
 .addRequestHandlers(
 LaunchRequestHandler,
 HelloWorldIntentHandler,
 HelpIntentHandler,
 CancelAndStopIntentHandler,
 FallbackIntentHandler,
 GetAIResponseIntentHandler,
 SessionEndedRequestHandler,
 IntentReflectorHandler)
 .addErrorHandlers(
 ErrorHandler)
 .withCustomUserAgent('sample/hello-world/v1.2')
 .lambda();

// this my last error log thingy:
const header = {
 namespace: 'SkillDebugger',
 name: 'CaptureError',
 messageId: 'b68aaebd-9760-4a2f-8279-23127ceeaeel',
 payload: {
 skillId: 'amzn1.ask.skill.4e3cd483-a6cc-4494-b190-11c9b68be961',
 timestamp: '2024-10-13T01:57:20.175Z',
 dialogRequestId: 'a6834fa2-a3ba-4a8d-be66-7a921865fd72',
 skillRequestId: 'amzn1.echo-api.request.1884d710-5939-4070-8170-a5c021d5bdd6',
 code: 'SKILL_ENDPOINT_ERROR',
 description: 'Skill execution returned an exception for requestId amzn1.echo-api.request.1884d710-5939-4070-8170-a5c021d5bdd6',
 debuggingInfo: {}
 }
};

```

```

 "type";: "SkillExecutionInfo";, \n "content";:
 {\n "invocationRequest";: {\n
 "endpoint";: "arn:aws:lambda:eu-west-1:975050070604:function:4e3cd483-
 a6cc-4494-b190-11c9b68be961:Release_0";, \n "body";:
 {\n "version";: "1.0";, \n
 "session";: {\n "new";: false, \n
 "sessionId";: "amzn1.echo-api.session.ebd6e6c2-1980-4056-a4a0-
 5eb46e40a755";, \n "application";: {\n
 "applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
 11c9b68be961";, \n }, \n
 "attributes";: {}, \n "user";: {\n
 "userId";:
 "amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22
 CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHFVNU07KWN2ER26Z234DMXDGSQS4B4A5XMYL6FSWMB7D3BFXS7II5EI2
 KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
 WQ";, \n "permissions";: {\n
 "scopes";: {\n
 "alexa::devices:all:geolocation:read";: {\n
 "status";:
 "DENIED";, \n } \n
 } \n } \n
 }, \n "context";: {\n
 "Viewports";: [\n {\n
 "type";: "APL";, \n "id";:
 "medHub";, \n "shape";:
 "RECTANGLE";, \n "dpi";: 160, \n
 "presentationType";: "OVERLAY";, \n
 "canRotate";: false, \n
 "configuration";: {\n
 "current";: {\n "mode";:
 "HUB";, \n "video";: {\n
 "codecs";: [\n
 "H_264_42";, \n
 "H_264_41";, \n] \n
 }, \n
 "size";: {\n "type";:
 "DISCRETE";, \n
 "pixelWidth";: 1280, \n
 "pixelHeight";:
 800 \n } \n
 } \n } \n
], \n "Viewport";: {\n
 "experiences";: [\n {\n
 "arcMinuteWidth";: 221, \n
 "arcMinuteHeight";: 162, \n
 "canRotate";: false, \n
 "canResize";:
 false \n } \n], \n
 "mode";: "HUB";, \n "shape";:

```

```
"RECTANGLE"; \n\n <>pixelHeight<>;: 800, \n\n <"dpi">;: 160, \n\n <"currentPixelWidth">;:\n <"currentPixelHeight">;: 800, \n\n <"touch">;: [\n <"SINGLE">;\n <"keyboard">;: [\n <"DIRECTION">;\n <"video">;: {\n <"codecs">;: [\n <"H_264_42">;,\n <"H_264_41">;]\n }\n }, \n]\n], \n <"Extensions">;: {\n <"available">;: {\n <"aplext.backstack.10">;;\n {} \n } \n }, \n <"System">;: {\n <"application">;: {\n <"applicationId">;: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-\n 1lc9b68be961">;\n <"user">;: {\n <"userId">;:\n <"amzn1.ask.account.AMA5J5IPANGXEWCWPW3SHALU2NB4DHXU3BJAYIQQHSUIQAXPOYUACOEVDTOG22CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVUO7KWN2ER26Z234DMXDGSQS4B4A5XMVL6FSWMB7D3BFXS7II5EI2KOIZDI5FWQRR2YMT4VQHWE6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJJDZYH5Y2TL4GI7M27WQ">;,\n <"permissions">;: {\n <"scopes">;: {\n <"alexa::devices.all:geolocation:read">;: {\n <"status">;:\n <"DENIED">;\n }\n }\n }, \n <"device">;: {\n <"deviceId">;:\n <"amzn1.ask.device.AMAXRWYUCF3SDX4KFOKAJR3J2LBF4R5I5CM7OLTUQFW2JXXKD6RWEGOMWI6RG5UXMITSMS4UY6JVTA2OYW57QCAT7TGT5OUMHLI4MHFWQAHZJKLGZHJHCQ3GROHFVFZ5K5GG2FQIAI7UHBT06PYNAUSE6CYHVUGDS43ZPD2QN5IX52F545Q3W52JY5LFD43TP5K2WNYDFCAT72T">;,\n <"supportedInterfaces">;: {\n <"apiEndpoint">;: "https://api.eu.amazonalexa.com">;,\n <"accessToken">;:\n <"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZI6IjEifQ.eyJhdWQiOiJodHRwczovL2FwaS5ldS5hbWF6b25hbGV4YS5jb20iLCJpc3MiOiJBbGV4YVNraWxsS2l0Iiwic3ViIjoieyY1b6bjEuYXNrLnBraWxSLjRlM2NkNDgzLEWE2Y2MtNDQ5NC1mTkwlTEExYzlnjhizTK2MSIsImV4cCI6MTcyODc4NDY5OCwiaWF0IjoxeNzI4NmZhbnBlcnNvbGljdPK25vSHU5U3grM1IDdjgxmJJIMHJQEZEfBB1McUp5SOZPFQMtnYVnqYXJEbOFBQVYZY0IzbXBpqUVphWUV6Q1d6eHowSGhnNNVV1OTJuNEdIaUZYMVRFTxBszWLqbFlrRGJSRkhbb21URUViUDA2MUZOQ1krKZNzMlk2ZW51T1JmTHIREhNCqnZFVERjaUNLNjVtNW5jNW5NR1BnRTVuac9ZWFNUYOYvtmswcVJFWThFRkRaYk9NNmdOS0xqMFrySlNnoGVjbkpXTmq1TjFPTohlawY2WTBRofIzRWlyVkhrVMmxmWTZIEujHzVRIykdsRjh2b2ZVT1dhZGlmTFRIRZehpuhNoT0x4ekQ3Z0tkde9RDwjhUTf6NXAzUEX3MGt2WWZuwvfyc3lNNDZ5VFByQuDXWGQ4U1NUTORSU1yzUmndGJIAGpiUUUrMXJIUK16OGpiVmZDbG5OSFVJeijvm3IKRE15MXVoWhR6dkNUK2dD
```



```

HRBTVFYUysiLCJkZXZpY2VJZCI6ImFtem4xLmFzay5kZXZpY2UuQU1BWFJXWVVDJRjNTRFg0SOZPSzRKUjNKMkx
CRjRSNuk1Q003ToxUVVFGVzJKWEtYRDZSV0VHT01XSTZSRzVWWE1JVFNNuzRVWTZKV1RBMk9ZV1M3UUFDN1RHV
DVPVU1ITEk0TUhGV1FBSFpKS0xHSFpKSENRM0dST0hWR1o1SszVHRzJGUU1BSTdVSEJUTzZQWU5BVTJTRTZDWU
WVdHRFMOM1pQRDJRT041SVg1MkY1NDVRM1c1MkpZNUxGRDQzVFA1SzJXT11ERkNBVDcyVCIsInVzZXJJZCI6I
mFtem4xLmFzay5hY2NvdW50LkFNQTVKNUIQQU5HWEVXQ1BXM1NIQUxVMk5CNERIWFUzQkpBWU1RUUg1VU1RQVh
QT11VQUNPSUVWRFRPRzIyQOVOM1NDRjdLR0Y1UkpTVUNUN1pCTEs0M1RUMORIWkZOV1VPNOtXTjJFUjI2WjIzN
ERNWERHU1FTNEIOQTVYTVZMNkZTV01CNOQzQkZYUzdJSTVFSTJLT01aREk1RldRU1IyWU1UNFZRSFc2RVJRVEF
XTzJLM1VOUzJPSkMzTTJITjRFSUdQU0wyUDY3WUNNT0wOU1pCVURENFJKRFpZSDVZM1RMNEJN00yN1dRIn19.
i1op-pbgRh2ceV-F7Ryk2SCKW1pCFMIFrsGGP3Z05Jv5kMTTqwOxv6LjarDmCn9U0-
UMDftwKYMhHegXsQ90Vw_CXFXKnh7BFrfXaaoD-8PGPf5YJzyl7qn0EaFee-
6rOKd1F25r2mXkY3SLy0CF6xdgyRtFT7rVp17Ag9lr_XPRwUrJIu0sRzPxfGKXhZnbOMmpnXuJ-
eMH5gm3Y0cixMHB8hyNkM0a2hCChjrW7gcGi7luH79Ig2t1i7EWfgtCQsIr90mkcd_w8F14_iROZK_lgRtzqxL
jhxxYNcTvW8xr34G7J3gOIqFW9gyZQRpOZIzhtNjNBChke9GdoH0m4w";\n
 }\n },\n "request";: {\n
"type";: "SessionEndedRequest";,\n
"requestId";: "amzn1.echo-api.request.1884d710-5939-4070-8170-
a5c021d5bdd6";,\n "timestamp";: "2024-10-
13T01:57:19Z";,\n "locale";: "en-
US";,\n "reason";: "ERROR";,\n
"error";: {\n "type";:
"INVALID_RESPONSE";,\n "message";:
"An exception occurred while dispatching the request to the
skill.";\n }\n }\n "metrics";: {\n
"skillExecutionTimeInMilliseconds";:
450\n }\n }\n }\n }\n\nwhat can you tell me?
"]], "status": "finished_successfully", "end_turn": null, "weight": 1.0, "metadata":
{"serialization_metadata": {"custom_symbol_offsets": []}, "request_id":
"8d1bbc9f3b3e525b-MXP", "message_source": null, "timestamp": "absolute",
"message_type": null}, "recipient": "all", "channel": null}, "parent": "558d5860-0802-
434a-b927-686d41f1515b", "children": [{"d4a06054-ad14-4b8f-99f3-587fa30fa2d7"}],
"d4a06054-ad14-4b8f-99f3-587fa30fa2d7": {"id": "d4a06054-ad14-4b8f-99f3-587fa30fa2d7",
"message": {"id": "d4a06054-ad14-4b8f-99f3-587fa30fa2d7", "author": {"role":
"assistant", "name": null, "metadata": {}}, "create_time": 1728784890.786336,
"update_time": null, "content": {"content_type": "text", "parts": ["The error
you're encountering, specifically `INVALID_RESPONSE`, indicates that your Alexa
skill is returning an invalid or incomplete response, which is causing the skill to
terminate. This error can occur due to several potential issues, including problems
with the OpenAI API request, incorrect response handling, or even misconfigured
session handling.\n\n### Key Problem Areas Based on the Error Log:\n\n1. **Skill
Endpoint Error (`SKILL_ENDPOINT_ERROR`)**:\n - The log states that there was an
exception during the skill's execution, which led to the `INVALID_RESPONSE`. This
means that something went wrong with either processing the intent or calling external
services like OpenAI.\n \n2. **OpenAI API Call or Promise Issue**:\n - If the
OpenAI API call is failing, your skill might not be returning a valid response in time,
leading to this error. Ensure your API call logic is handling all errors
gracefully.\n\n3. **Response Format Issue**:\n - The response that the skill sends
back to Alexa might be incorrectly structured or missing some required data, causing
Alexa to terminate the session.\n\n### Steps to Troubleshoot:\n\n#### 1. **Check

```

OpenAI API Call and Error Handling:\*\*\nYou need to ensure that any errors during the API request are properly caught and handled, so Alexa can return a response even if the API call fails.\n\nUpdate your `getAIResponse` function to add more detailed logging and ensure proper error handling:\n\n```\njavascript\nasync function getAIResponse(query) {\n const apiKey = &#x27;your-openai-api-key&#x27;; // Replace with your actual OpenAI API key\n const options = {\n hostname: &#x27;api.openai.com&#x27;;\n path: &#x27;/v1/completions&#x27;;\n method: &#x27;POST&#x27;;\n headers: {\n &#x27;Authorization&#x27;: `Bearer \${apiKey}`,\n &#x27;Content-Type&#x27;: &#x27;application/json&#x27;;\n }\n };\n const body = JSON.stringify({\n model: &#x27;text-davinci-003&#x27;; // Adjust if needed for other models\n prompt: query,\n max\_tokens: 100,\n temperature: 0.7\n });\n return new Promise((resolve, reject) => {\n const req = https.request(options, (res) => {\n let data = &#x27;&#x27;;\n res.on(&#x27;data&#x27;, (chunk) => {\n data += chunk;\n });\n res.on(&#x27;end&#x27;, () => {\n try {\n const response = JSON.parse(data);\n console.log(&#x27;OpenAI response:&#x27;, response);\n if (response.choices && response.choices.length > 0) {\n resolve(response.choices[0].text.trim());\n } else {\n console.log(&#x27;No valid response from OpenAI&#x27;);\n resolve(&#x27;Sorry, I couldn\u2019t understand the response.&#x27;);\n }\n } catch (error) {\n console.log(&#x27;Error parsing OpenAI response:&#x27;, error);\n reject(error);\n }\n });\n req.on(&#x27;error&#x27;, (e) => {\n console.log(&#x27;Error with API request:&#x27;, e);\n reject(e);\n });\n req.write(body);\n req.end();\n });\n });\n}\n\n- \*\*Additional Logging\*\*: This will log the full response from OpenAI and catch any parsing errors.\n- \*\*Graceful Fallback\*\*: If the API call fails or returns an unexpected response, the skill will return a default fallback message (`Sorry, I couldn\u2019t understand the response.`) instead of crashing.\n\n#### 2. \*\*Handle Empty or Invalid Slot Values\*\*:\nMake sure you\u2019re checking whether the slot is filled properly before making the API call. If the slot is empty, return a prompt asking the user to clarify their input.\n\nYou already have this in place, but verify that it\u2019s working as intended:\n\n```\njavascript\nif (!intent.slots.query.value) {\n const speakOutput = &quot;I didn&#x27;t catch that. Can you please ask me a question like, &#x27;What is quantum mechanics?&#x27;&quot;;\n console.log(&#x27;Slot value is empty.&#x27;);\n return handlerInput.responseBuilder\n .speak(speakOutput)\n .reprompt(speakOutput)\n .getResponse();\n}\n\n- \*\*Ensure a Valid Response is Always Returned\*\*:\nAlexa expects a valid response even when an error occurs, so always ensure the response is returned. This is critical because if the skill does not return a response, Alexa will terminate the session with an `INVALID\_RESPONSE` error.\n\nIn your `GetAIResponseIntentHandler`, update the response handling to ensure the skill always returns a valid response, even if an error occurs:\n\n```\njavascript\nconst GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n return Alexa.getRequestType(handlerInput.requestEnvelope) === &#x27;IntentRequest&#x27;\n && Alexa.getIntentName(handlerInput.requestEnvelope) === &#x27;GetAIResponseIntent&#x27;;\n },\n async handle(handlerInput) {\n

```

const { intent } = handlerInput.requestEnvelope.request;\n\n // Check if the
slot is filled\n if (!intent.slots.query.value) {\n const
speakOutput = "I didn't catch that. Can you please ask me a question like,
'What is quantum mechanics?'";\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n let
aiResponse = '';\n try {\n // Call OpenAI API\naiResponse = await getAIResponse(intent.slots.query.value);\n } catch (error)
{\n console.log('Error occurred while calling OpenAI API:';,
error);\n aiResponse = 'Sorry, I had trouble processing your
request.';\n }\n // Return the AI response or fallback message\nreturn
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n};\n\n```\n\n#### 4. **Check for Timeouts**:\nIf your OpenAI API request
takes too long, Alexa may timeout and cause an `INVALID_RESPONSE`. Ensure that your
skill is returning responses within Alexa's 8-second timeout window.\n\nTo handle
timeouts gracefully, consider setting a maximum timeout for your OpenAI API call and
returning a default response if the API call takes too long:\n\n```\njavascript\nasync
function getAIResponse(query) {\n return Promise.race([\n // Your OpenAI API
call\n makeOpenAIRequest(query),\n // Timeout after 7 seconds (7000
milliseconds)\n new Promise((_, reject) =>{\n setTimeout(() =>{\n
reject(new Error('Timeout')), 7000)\n })\n]);\n}\n\n```\n\n#### 5.
Remove Duplicate Intent Handlers:\nIt looks like you have **duplicate intent
handlers** for `GetAIResponseIntentHandler` in your code. This can cause issues when
processing intents.\n\nRemove the redundant `canHandle` and `handle` blocks to keep
only one:\n\n```\njavascript\nconst GetAIResponseIntentHandler = {\n
canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';\n
& Alexa.getIntentName(handlerInput.requestEnvelope) ===
'GetAIResponseIntent';\n },\n async handle(handlerInput) {\n
const { intent } = handlerInput.requestEnvelope.request;\n // Ensure no
delegation\n handlerInput.requestEnvelope.request.dialogState =
'COMPLETED';\n if (!intent.slots.query.value) {\n const
speakOutput = "I didn't catch that. Can you please ask me a question like,
'What is quantum mechanics?'";\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n let
aiResponse = '';\n try {\n aiResponse = await
getAIResponse(intent.slots.query.value); // Call OpenAI API here\n } catch
(error) {\n console.log(`Error with OpenAI API call: ${error}`);\n
aiResponse = 'Sorry, I had trouble processing that.';\n }\n return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n};\n\n```\n\n#### Next Steps:\n1. **Remove duplicate intent handlers** to
avoid confusion.\n2. **Ensure proper error handling** and return valid fallback
responses in case of API failures or timeouts.\n3. **Add logging** to track API
responses and errors, and make sure to handle all errors gracefully.\n\nOnce
you've made these adjustments, deploy the code again and test the skill using the
Test tab in the Alexa Developer Console. Check the **CloudWatch logs** for more

```

[illegible]

[illegible]

```

eMH5gm3Y0ciXmHB8hYnKMOa2hCchjrW7gcGi71uH79Ig2t1i7EWfgtCQsIr90mkcd_w8F14_iROZK_lgRtzqxL
jhhxYnTvW8xr34G7J3g0IqFW9gyZQRpOZIzhtNjNBChke9GdoH0M4w"\n\t\t}\n\t}, \n\t"re
quest"; {\n\t\t"type";
"LaunchRequest";, \n\t\t"requestId";: "amzn1.echo-
api.request.6eb6f2cf-91c7-49e2-bb7d-1121d52fe3b5";, \n\t\t"locale";
"en-US";, \n\t\t"timestamp";: "2024-10-
13T01:57:18Z";, \n\t\t"shouldLinkResultBeReturned";: false\n\t}\n}\n\nThis
is the second and the response is below\n\n{\n\t"version";:
"1.0";, \n\t"session";: {\n\t\t"new";:
true, \n\t\t"sessionId";: "amzn1.echo-api.session.ebd6e6c2-1980-4056-
a4a0-5eb46e40a755";, \n\t\t"application";:
"\n\t\t\t"applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961";\n\t\t\t}, \n\t\t\t"attributes";: {}, \n\t\t\t"user";:
{\n\t\t\t\t"userId";:
"amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVU07KWN2ER26Z234DMXDGSQS4B4A5XMYL6FSWMB7D3BFXS7II5EI2
KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ";, \n\t\t\t\t"permissions";: {\n\t\t\t\t\t"scopes";:
{\n\t\t\t\t\t\t"alexa::devices:all:geolocation:read";:
{\n\t\t\t\t\t\t\t"status";:
"DENIED";\n\t\t\t\t\t\t\t}\n\t\t\t\t\t\t}\n\t\t\t\t\t}, \n\t\t\t\t"affiliatedResou
rces";: []\n\t\t}, \n\t\t"context";: {\n\t\t\t"Viewports";:
[\n\t\t\t\t{\n\t\t\t\t\t"type";: "APL";, \n\t\t\t\t\t"id";:
"medHub";, \n\t\t\t\t\t"shape";:
"RECTANGLE";, \n\t\t\t\t\t"dpi";:
160, \n\t\t\t\t\t"presentationType";:
"OVERLAY";, \n\t\t\t\t\t"canRotate";:
false, \n\t\t\t\t\t"configuration";: {\n\t\t\t\t\t\t"current";:
{\n\t\t\t\t\t\t\t"mode";: "HUB";, \n\t\t\t\t\t\t\t"video";:
{\n\t\t\t\t\t\t\t\t"codecs";:
[\n\t\t\t\t\t\t\t\t\t"H_264_42";, \n\t\t\t\t\t\t\t\t\t"H_264_41";\n\t\t\t\t\t\t\t\t\t]\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t\t\t"size";:
{\n\t\t\t\t\t\t\t\t\t"type";:
"DISCRETE";, \n\t\t\t\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t\t\t\t"pixelHeight";:
800\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"Viewport";:
{\n\t\t\t\t\t\t"experiences";: [\n\t\t\t\t\t\t\t{\n\t\t\t\t\t\t\t\t"arcMinuteWidth";:
221, \n\t\t\t\t\t\t\t\t"arcMinuteHeight";: 162, \n\t\t\t\t\t\t\t\t"canRotate";:
false, \n\t\t\t\t\t\t\t\t"canResize";:
false\n\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"mode";:
"HUB";, \n\t\t\t\t\t\t"shape";:
"RECTANGLE";, \n\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t"pixelHeight";: 800, \n\t\t\t\t\t\t"dpi";:
160, \n\t\t\t\t\t\t"currentPixelWidth";: 1280, \n\t\t\t\t\t\t"currentPixelHeight";:
800, \n\t\t\t\t\t\t"touch";:
[\n\t\t\t\t\t\t\t"SINGLE";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"keyboard";:
[\n\t\t\t\t\t\t\t"DIRECTION";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"video";:
{\n\t\t\t\t\t\t\t"codecs";:
[\n\t\t\t\t\t\t\t\t"H_264_42";, \n\t\t\t\t\t\t\t\t"H_264_41";\n\t\t\t\t\t\t\t\t\t]\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t\t\t\t"size";:
{\n\t\t\t\t\t\t\t\t\t"type";:
"DISCRETE";, \n\t\t\t\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t\t\t\t"pixelHeight";:
800\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"Viewport";:
{\n\t\t\t\t\t\t"experiences";: [\n\t\t\t\t\t\t\t{\n\t\t\t\t\t\t\t\t"arcMinuteWidth";:
221, \n\t\t\t\t\t\t\t\t"arcMinuteHeight";: 162, \n\t\t\t\t\t\t\t\t"canRotate";:
false, \n\t\t\t\t\t\t\t\t"canResize";:
false\n\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"mode";:
"HUB";, \n\t\t\t\t\t\t"shape";:
"RECTANGLE";, \n\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t"pixelHeight";: 800, \n\t\t\t\t\t\t"dpi";:
160, \n\t\t\t\t\t\t"currentPixelWidth";: 1280, \n\t\t\t\t\t\t"currentPixelHeight";:
800, \n\t\t\t\t\t\t"touch";:
[\n\t\t\t\t\t\t\t"SINGLE";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"keyboard";:
[\n\t\t\t\t\t\t\t"DIRECTION";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"video";:
{\n\t\t\t\t\t\t\t"codecs";:
[\n\t\t\t\t\t\t\t\t"H_264_42";, \n\t\t\t\t\t\t\t\t"H_264_41";\n\t\t\t\t\t\t\t\t\t]\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t\t\t\t"size";:
{\n\t\t\t\t\t\t\t\t\t"type";:
"DISCRETE";, \n\t\t\t\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t\t\t\t"pixelHeight";:
800\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"Viewport";:
{\n\t\t\t\t\t\t"experiences";: [\n\t\t\t\t\t\t\t{\n\t\t\t\t\t\t\t\t"arcMinuteWidth";:
221, \n\t\t\t\t\t\t\t\t"arcMinuteHeight";: 162, \n\t\t\t\t\t\t\t\t"canRotate";:
false, \n\t\t\t\t\t\t\t\t"canResize";:
false\n\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"mode";:
"HUB";, \n\t\t\t\t\t\t"shape";:
"RECTANGLE";, \n\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t"pixelHeight";: 800, \n\t\t\t\t\t\t"dpi";:
160, \n\t\t\t\t\t\t"currentPixelWidth";: 1280, \n\t\t\t\t\t\t"currentPixelHeight";:
800, \n\t\t\t\t\t\t"touch";:
[\n\t\t\t\t\t\t\t"SINGLE";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"keyboard";:
[\n\t\t\t\t\t\t\t"DIRECTION";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"video";:
{\n\t\t\t\t\t\t\t"codecs";:
[\n\t\t\t\t\t\t\t\t"H_264_42";, \n\t\t\t\t\t\t\t\t"H_264_41";\n\t\t\t\t\t\t\t\t\t]\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t\t\t\t"size";:
{\n\t\t\t\t\t\t\t\t\t"type";:
"DISCRETE";, \n\t\t\t\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t\t\t\t"pixelHeight";:
800\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"Viewport";:
{\n\t\t\t\t\t\t"experiences";: [\n\t\t\t\t\t\t\t{\n\t\t\t\t\t\t\t\t"arcMinuteWidth";:
221, \n\t\t\t\t\t\t\t\t"arcMinuteHeight";: 162, \n\t\t\t\t\t\t\t\t"canRotate";:
false, \n\t\t\t\t\t\t\t\t"canResize";:
false\n\t\t\t\t\t\t\t\t}\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"mode";:
"HUB";, \n\t\t\t\t\t\t"shape";:
"RECTANGLE";, \n\t\t\t\t\t\t"pixelWidth";:
1280, \n\t\t\t\t\t\t"pixelHeight";: 800, \n\t\t\t\t\t\t"dpi";:
160, \n\t\t\t\t\t\t"currentPixelWidth";: 1280, \n\t\t\t\t\t\t"currentPixelHeight";:
800, \n\t\t\t\t\t\t"touch";:
[\n\t\t\t\t\t\t\t"SINGLE";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"keyboard";:
[\n\t\t\t\t\t\t\t"DIRECTION";\n\t\t\t\t\t\t\t}, \n\t\t\t\t\t\t"video";:
{\n\t\t\t\t\t\t\t"
```

```
{\n\t\t,\n\t\t\t"Extensions"; {\n\t\t\t"available";
\n\t\t\t\t"apext:backstack:10";
{\n\t\t\t\t}\n\t\t\t,\n\t\t\t"Advertising"; {\n\t\t\t\t"advertisingId";:
"00000000-0000-0000-0000-000000000000";,\n\t\t\t\t"limitAdTracking";:
true\n\t\t\t,\n\t\t\t"System"; {\n\t\t\t\t"application";:
\n\t\t\t\t\t"applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
1lc9b68be961";\n\t\t\t\t},\n\t\t\t\t"user";: {\n\t\t\t\t\t"userId";:
"amzn1.ask.account.AMA5J5IPANGXECPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOEVDTOG22
CEN2SCF7KGF5RJ5UCT7ZBLK43TT3DZHxFNVUO7KWN2ER26Z234DMXDGSQS4B4A5XMVL6FSWWB7D3BFXS7II5EI2
KOIZDI5FWQR2YMT4VQHW6ERQTAWO2K3UNS2OJC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ";,\n\t\t\t\t\t"permissions";: {\n\t\t\t\t\t\t"scopes";:
\n\t\t\t\t\t\t\t"alexa::devices:all:geolocation:read";:
\n\t\t\t\t\t\t\t"status";:
"DENIED";\n\t\t\t\t\t\t\t\t"\n\t\t\t\t\t\t\t\t\t"\n\t\t\t\t\t\t\t\t\t"\n\t\t\t\t\t\t\t\t\t", \n\t\t\t\t\t\t\t"device
";: {\n\t\t\t\t\t\t\t"deviceId";:
"amzn1.ask.device.AMAXRWYUCF3SDX4KFOK4JR3J2LBF4R5I5CM7OLTUQFW2JXXKD6RWEGOMWI6RG5U
XMITSMS4UY6JVTA2OYWS7QC7TGT5OUMHLI4MHFWQAHZJKLGHZJHCQ3GROHVZFZ5K5GG2FQIAI7UHBT06PYNAU2
SE6CYHVWUGDS43ZPD2QN5IX52F545Q3W52JY5LFD43TP5K2WNYDFCAT72T";,\n\t\t\t\t\t\t\t"supp
ortedInterfaces";: {} \n\t\t\t\t\t\t), \n\t\t\t\t\t\t"apiEndpoint";:
"https://api.eu.amazonalexa.com";,\n\t\t\t\t\t\t"apiAccessToken";:
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJhdWQiOiJodHRwczovL2FwaS5ld
S5hbWF6b25hbGV4YS5jb20iLCJpc3MiOiJBbGV4YVNraWxsS2l0Iiwic3ViIjoieYW16bjEuYXNrnLnBraWxsLjR
lM2NkNDgzLEWEyY2MtNDQ5NC1mTkwlTEExYzlnInjiZTk2MSIsImV4cCI6MTcyODc4NDY5OCwiaWF0IjoxNzI4N
zgONjM4LClJuYmYiOiJE3Mjc3ODQzMzsInByaXZhZGVBdGFPbXBxM0nsiaXNb3JSQS16ImZhbHN1Iiwibm9uThd
hu2NvcGVzIjoieWxlLeGE60nBlcnNvb19pZDpyZWFiKiwiY29udGV4dCI6IkFBQUFBQUFBQUFBVCa2d4TXo2azZGV
y9ZWVE0a3JUMW11ZUFFQUFBQUFBURkmWRDFHFUFuOXIKWhTei81NnJkZTd5aFZBRnhDNTdicHIJclFMmZ2
5VDZOQ1fVNVVE1VURURVR3bjFpY20yeE9tbEpiRldYMFRyMO15c2VOVRSUEJxSOJZTnBhMW9mczdYNlhvVG4vT
EVNeTVSunprMIAMwxASjJHdjdpK25vsHU5U3grM11ddjkxmJJIMHJQEZFbb1McUp5S0ZPqmtNYYNQYXJEBOf
BQVYZYO1zbXpqUVphWUV6Qld6eHowSGhNNVV1OTJuNEdiAUZYMVFRTXBSZW1qbFllyrGJSRKhhb21URUViUDA2M
UZ0Q1krKZNzm1k2ZW51T1JmTHIreHNCqnZFVERjaUNLNjvtNW5jnw5NR1BnRTVuAc9ZWFNUYOYvTmswcVJFWTh
FRkraYk9NNmdOS0xqmFRysINnoGVjbkpXTmq1tJPFTOh1aWy2WTBROIzRW1yVkhrVMmxmwTZIEUJHZVRIYkdSR
jh2b2zvT1dhZG1mTFRIZEhpUHNtoTx4ekQ3ZOtkdE9rdWJUUTF6NXAzUEX3MGt2WWWzuWVFyc31NNDZ5VFBYQuD
xWGQ4U1NUTORSU1yzUmntdGJlaGpiUUNRMXIJIuk16OGpiVmZDbG5OSFVJejvm3JKRE15MXVoWHR6dkNUK2dDe
HRBTIFYUysiLCJkZXZpY2VJZCI6ImFtem4xLmFzay5kZXZpY2UuQU1BWFFJXWVDRjNTRfGOS0ZSPSZRkuJNMkx
CRjRSNUk1QOO3TOxUVVFGvzJKWeTYRDZSVOVHTOIxSTZSRzVvWE1JVFNUuzRVVTZKVIRBMk9ZV1M3UUFDNIrhV
DVPVU1ITEKOTUhGV1FBSFPkSOxHSFPkSENRM0dstohWR1oiSzVHRzJGUUBSTdvSEJUTzZQWU5BVJTJRTZDWuh
WVvdHRFMOM1pQRDJRTO41SvGlMkyINDVRM1cmkpZNUxGRDQzVFAISzJXT1IERKNBDcyVICisInvZZXJJZCI6I
mFtem4xLmFzay5hy2NdW5OLkFNQTVKNU1QQU5HWEVXQ1BXMINIUxvMk5CNERIWFUzQkpBWU1RUUG1VU1RQVh
QT1lVQUNPSUVWRFPRPrIyQOVOMINdrjdLOY1UkpTVUNUN1pCTESOM1RUMORIwkZOV1VPN0xtXJJFUjI2WjiZn
ERNWERHU1FTNEIOQTVYTVMNKztVOICNOqzQkZYUzdJSTVFSTJLT01AREKIrlDrU1IyWU1UNFZRSc2RVJRVEF
XTzJLM1VOUzJPSKmzTTJITjrFSuDQUowYUDY3WUNNTOWOU1PCVURENFJKRFpZSDVZMIRMNEdJNOOyn1drin19.
ilop-pbgRh2ceV-F7Ryk2SCKW1pCFMIFrsGGP3ZO5Jv5kMTTqw0xv6LjarDmCn9UO-
UMdfTWKMHHegXsQ90Vw_CFXFKNh7BfrfxAAD-8PGPf5YJzy17qn0EafFee-
6rOkdlF25r2mXKY3SLyOCF6xdgryrtFT7rvPl7Ag9lr_XPRwUrJIuoRsRpXFgKXHznb0MmpnXuJ-
eMH5gm3YoixMBH8hyNmMoahCcHjrW7gcGi7luH79Ig2ti7EFwgTCqsIr90mkcd_w8FI4_iROzk_lgRtzqxL
jhxyNctvW8XR34G7J3gOIqFW9gyZQRpOZIzhTNJNBChke9GdoHOm4w";\n\t\t\t}, \n\t\t\t"re
quest";: {\n\t\t\t"type";:
"LaunchRequest";,\n\t\t\t"requestId";: "amzn1.echo-
api.request.6eb6f2cf-91c7-49e2-bb7d-1121d52fe3b5";,\n\t\t\t"locale";:
```

```
"en-US"; \n\t\t"timestamp";: "2024-10-
13T01:57:18Z"; \n\t\t"shouldLinkResultBeReturned";:
false\n\t}\n\nresponse: \n\nactually there was no response both times but heres
more error log: \n\n{\n "header";: {\n "namespace";:
"Alexa.Presentation.APL";,\n "name";:
"RenderDocument";,\n "messageId";: "85e80307-9c10-49cd-
a4b9-7ca5fd2e7c2b";,\n "keys";: {\n "channel";:
"visual";,\n "isBlocking";: true,\n
"navigationType";: "NONE";\n },\n
"dialogRequestId";: "6c11a1b6-377f-4f5d-b926-
8660e2918e9c";\n },\n "payload";: {\n
"presentationToken";: "amzn1.as-
tt.v1.Domain:Application:Knowledge#TID#6ac9908d-483d-466d-9d7e-9c1764f1fcb3";,\n
">windowId";: "";,\n "timeoutType";:
"SHORT";,\n "document";: {\n "onMount";:
[\n {\n "when";:
"${environment.extension.Metrics}";,\n "type";:
"Metrics:IncrementCounter";,\n "metricId";:
"viewport.${@viewportProfile}";\n },\n {\n
"when";: "${environment.extension.Metrics &&
environment.screenReader}";,\n "type";:
"Metrics:IncrementCounter";,\n "metricId";:
"screenReaderOn";\n }\n],\n
"settings";: {\n "Metrics";: {\n
"applicationId";: "Knowledge";,\n
"experienceId";: "unified_qa";\n },\n
"AttentionSystem";: {\n
"attentionSystemStateName";:
"AttentionSystemState";\n },\n
"supportsResizing";: true\n },\n "import";:
[\n {\n "name";: "alexa-
layouts";,\n "version";:
"1.5.0";\n },\n {\n
"name";: "info-all-no-imports";,\n
"source";: "https://d27b8nvlbs5ip9.cloudfront.net/layouts/info-all-no-
imports/1.5.0.json?ver=5188559d-aa27-413b-8e19-6c5877c37dfe";,\n
"version";: "1.5.0-5188559d-aa27-413b-8e19-
6c5877c37dfe";\n }\n],\n
"mainTemplate";: {\n "item";: {\n
"layoutData";: "${payload.info.properties.layoutData}";,\n
"type";:
"${payload.info.properties.layoutType}";\n },\n
"parameters";: [\n
"payload";\n]\n },\n
"resources";: [],\n "theme";: "dark";,\n
"styles";: {},\n "graphics";: {},\n
"type";: "APL";,\n "version";:
"1.4";,\n "layouts";: {},\n
"commands";: {},\n "lang";: "en-US";,\n
```



```

 "layoutDirection":: "LTR";, \n "extensions":: [\n
 {\n "name":: "AttentionSystem";, \n
 "uri":: "aplex:attentionssystem:10";\n }, \n
 {\n "name":: "Metrics";, \n
 "uri":: "aplex:metrics:10";\n }\n], \n
 "onConfigChange":: {\n "type"::
 "Reinflate";\n }, \n "datasources":: {\n
 "info":: {\n "transformers":: [], \n
 "type":: "object";, \n "properties":: {\n
 "layoutData":: {\n "overlayDefault":: {\n
 "layoutData":: {\n "image":: {\n
 "width":: 0, \n "source"::
 "";, \n "url":: "";, \n
 "height":: 1\n }, \n
 "footer":: {\n "layoutData"::
 {}, \n "layoutType"::
 "infoFooter";\n }, \n
 "background":: {\n "layoutData"::
 {\n "default":: {\n
 "imgUrl":: "";, \n
 "source":: "";\n }, \n
 "viewportClassSmallSmall":: {\n
 "imgUrl":: "";, \n
 "source"::
 "";\n }\n
 }, \n "layoutType"::
 "infoBackgroundWrap";\n }, \n
 "header":: {\n "layoutData":: {\n
 "viewportClassSmallSmall":: {}, \n
 "title":: "get smart and tell me about the relationship between cops and
 donuts";\n }, \n
 "layoutType":: "infoHeader";\n }, \n
 "content":: {\n "layoutData"::
 {\n "text":: "According to an
 Alexa Answers contributor: Back in the days when officers walked a beat, for an entire
 shift, they would often stop for a cup of coffee and a quick snack of a donut, which
 was a very cheap and time efficient snack during that bygone era that could be eaten
 in a hurry. The walking during a shift took care of the calories.
 ";\n }, \n
 "layoutType"::
 "Text";\n }\n }, \n
 "layoutType":: "infoFallbackNew";\n }, \n
 "default":: {\n "layoutData":: {\n
 "slide":: {\n "layoutData":: {\n
 "relatedAttributes":: {\n
 "layoutData":: {\n
 "attributes":: [\n {\n
 "layoutData":: {\n
 "name":: "Shelf Life";, \n

```

```

 "value";: "2
years";\n
 "layoutType";:
"infoSlideshowVerticalAttributesRelatedAttribute";\n
 },\n
 "layoutData";: {\n
"name";: "pH";,\n
"value";:
"5";\n
 },\n
"layoutType";:
"infoSlideshowVerticalAttributesRelatedAttribute";\n
 },\n
 "layoutData";: {\n
"name";: "Closing Price";,\n
"value";:
"$2.55";\n
 },\n
"layoutType";:
"infoSlideshowVerticalAttributesRelatedAttribute";\n
 }\n
 },\n
"layoutType";:
"infoSlideshowVerticalAttributesBox";\n
 },\n
 "answer";: {\n
"layoutData";: {\n
"answerText";: "According to an Alexa Answers contributor: Back in the
days when officers walked a beat, for an entire shift, they would often stop for a cup
of coffee and a quick snack of a donut, which was a very cheap and time efficient
snack during that bygone era that could be eaten in a hurry. The walking during a
shift took care of the calories. ";,\n
"secondaryTitle";: "";,\n
"label";: "";,\n
"title";:
"Coffee";\n
 },\n
"layoutType";:
"infoSlideshowVerticalAttributesAnswerBox";\n
 },\n
 "footer";: {\n
"layoutData";: {},\n
"layoutType";:
"infoFooter";\n
 },\n
"background";: {\n
"layoutData";: {\n
"backgroundImageData";: [\n
 {\n
 "imageUrl";:
"https://m.media-amazon.com/images/S/com.evi.images-
irs/premium/57/577280ff44dc596297d83443d4f196b6._FMjpg_FP_BD1_.png";,\n
"width";: 1280,\n
"source";: "Getty Images";,\n
"sourceData";: {\n
"layoutData";: {\n

```

```

 "source";: "Getty
 Images";\n
 "layoutType";:
 "infoSlideshowVerticalAttributesImageAttribution";\n
 },\n

 "height";:
 750\n
],\n
 "layoutType";:
 "infoSlideshowVerticalAttributesPager";\n
 },\n
 "entityProperty";: {\n
 "layoutData";: {\n
 "propertyLabel";: "";,\n
 "propertyValue";:
 "";\n
 "layoutType";:
 "infoSlideshowVerticalAttributesEntityProperty";\n
 },\n
 "header";: {\n
 "layoutData";: {\n
 "title";: "get smart and tell me about the relationship between cops and
 donuts";\n
 "layoutType";:
 "infoHeader";\n
 },\n
 "infoSlideshowVerticalAttributes";\n
 },\n
 "infoSingleSlide";\n
 "layoutType";:
 "infoViewportPivot";\n
 "supportedViewports";: [\n
 "HUB";,\n
 "minWidth";: 100,\n
 "minHeight";: 100,\n
 599\n
 },\n
 "HUB";,\n
 "minWidth";: 320,\n
 "minHeight";: 240,\n
 240\n
 },\n
 "HUB";,\n
 "minWidth";: 960,\n
 "minHeight";: 100,\n
 599\n
 },\n
 "HUB";,\n
 "minWidth";: 960,\n
 "minHeight";: 600,\n
 959\n
 },\n
 "HUB";,\n
 "minWidth";: 1280,\n
 "minHeight";: 600,\n
 "shape";: "ROUND";,\n
 "maxWidth";: 599,\n
 "maxHeight";:
 {\n
 "mode";:
 "shape";: "RECTANGLE";,\n
 "maxWidth";: 320,\n
 "maxHeight";:
 {\n
 "mode";:
 "shape";: "RECTANGLE";,\n
 "maxWidth";: 1279,\n
 "maxHeight";:
 {\n
 "mode";:
 "shape";: "RECTANGLE";,\n
 "maxWidth";: 1279,\n
 "maxHeight";:
 {\n
 "mode";:
 "shape";: "RECTANGLE";,\n
 "maxWidth";: 1919,\n
 "maxHeight";:

```

```

1279\n }, \n
"HUB";, \n
"minWidth";: 1920, \n
"minHeight";: 960, \n
1279\n }, \n
"HUB";, \n
"minWidth";: 960, \n
"minHeight";: 1920, \n
2560\n }, \n
"HUB";, \n
"minWidth";: 320, \n
"minHeight";: 600, \n
959\n }, \n
"HUB";, \n
"minWidth";: 426, \n
"minHeight";: 600, \n
1279\n }, \n
"HUB";, \n
"minWidth";: 640, \n
"minHeight";: 960, \n
1279\n }, \n
"HUB";, \n
"minWidth";: 960, \n
"minHeight";: 640, \n
853\n }, \n
"MOBILE";, \n
"minWidth";: 600, \n
"minHeight";: 320, \n
1920\n }, \n
"MOBILE";, \n
"minWidth";: 960, \n
"minHeight";: 320, \n
1920\n }, \n
"MOBILE";, \n
"minWidth";: 1280, \n
"minHeight";: 320, \n
1920\n }, \n
"TV";, \n
"minWidth";: 960, \n
"minHeight";: 540, \n
540\n }, \n
"TV";, \n
"minWidth";: 960, \n
"minHeight";: 200, \n
200\n }\n], \n
"id";: "";, \n
"Domain:Application:Knowledge";, \n
[\n {\n "uri";:
"alexaext:textmessage:10";\n }, \n {\n

```

```

{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 2560, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 1279, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 425, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 639, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 853, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 1279, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 959, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 1279, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 1920, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 960, \n
 "maxHeight";:
{\n "mode";:
"shape";: "RECTANGLE";, \n
 "maxWidth";: 960, \n
 "maxHeight";:
"presentationSession";: {\n
 "skillId";:
"grantedExtensions";:
"uri";:
{\n

```

```
"uri"; "alexaext:skillconnections:10"\n },\n{\n "uri";\n"aplext:metrics:10"\n },\n{\n "uri"; "aplext:attentionsystem:10"\n },\n{\n "uri";\n"aplext:backstack:10"\n }\n]\n }\n }\n }\n}], "status": "finished_successfully", "end_turn": null, "weight": 1.0, "metadata": {\n "serialization_metadata": {\n "custom_symbol_offsets": []\n }, "request_id":\n "8d1bc50d9eaf0e65-MXP", "message_source": null, "timestamp_": "absolute",\n "message_type": null}, "recipient": "all", "channel": null}, "parent": "d4a06054-ad14-4b8f-99f3-587fa30fa2d7", "children": ["64497db0-9676-4741-af09-ef99ca5d8ffb"],\n "64497db0-9676-4741-af09-ef99ca5d8ffb": {\n "id": "64497db0-9676-4741-af09-ef99ca5d8ffb",\n "message": {\n "id": "64497db0-9676-4741-af09-ef99ca5d8ffb",\n "author": {\n "role":\n "assistant",\n "name": null,\n "metadata": {}\n },\n "create_time": 1728785216.736229,\n "update_time": null,\n "content": {\n "content_type": "text",\n "parts": [\n "It looks like the error you're encountering might be related to Alexa using a fallback or default response, instead of your skill handling the request as expected. Specifically, the skill is not returning the expected response, and Alexa is providing a generic\n \"fallback\" or pre-generated answer about \"the relationship between cops and donuts.\" This implies that Alexa is not successfully reaching your OpenAI API integration or handling the custom skill request properly.\n\nHere's what may be going wrong and how you can address it:\n\n### 1. **Skill Endpoint Error**:\nYour skill might be failing to handle the request properly, causing Alexa to fall back on a default behavior. The log shows an `INVALID_RESPONSE` error, which usually means that the skill endpoint is not returning a valid response within the required time frame or the response format is incorrect.\n\n### 2. **OpenAI API Call Issue**:\nEnsure that your skill is correctly calling the OpenAI API and handling the response or errors from that call. If there's a delay, timeout, or issue in processing the OpenAI API response, Alexa may resort to using its own fallback.\n\n### 3. **Intent Fulfillment Check**:\nMake sure the `GetAIResponseIntentHandler` is being invoked when the correct intent is triggered, and that it's handling the request and returning a proper response.\n\n### 4. **Remove Duplicate Intent Handlers**:\nAs noted earlier, you may have multiple copies of the `GetAIResponseIntentHandler`. Ensure that there is only one handler for each intent, and it's correctly implemented.\n\n### 5. **Return Proper Response**:\nYour skill must return a valid response for every request. Even if the OpenAI API call fails or returns an unexpected result, the skill should always return a valid message to Alexa.\n\nHere are specific actions to take:\n\n#### 1. **Log and Verify the Intent Handling**:\n- Make sure to add more logging to check if the correct intent handler is being called. This will help confirm that the skill is reaching the `GetAIResponseIntentHandler` and making the OpenAI API call.\n\nFor example, log the intent and any errors clearly:\n\n```\njavascript\nconst GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n console.log(`Checking if GetAIResponseIntentHandler can handle the request`);\n return Alexa.getRequestType(handlerInput.requestEnvelope) ===\n `IntentRequest`;\n },\n async handle(handlerInput) {\n console.log(`GetAIResponseIntentHandler is handling the request`);\n const { intent } = handlerInput.requestEnvelope.request;\n if
```

```

(!intent.slots.query.value) {\n const speakOutput = "I didn't
catch that. Can you please ask me a question like, 'What is quantum
mechanics?'";\n console.log('Slot value is empty.');\n
return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n let
aiResponse = '';\n try {\n console.log('Sending query
to OpenAI:', intent.slots.query.value);\n aiResponse = await
getAIResponse(intent.slots.query.value);\n console.log('Received
response from OpenAI:', aiResponse);\n } catch (error) {\n
console.log('Error occurred while calling OpenAI API:', error);\n
aiResponse = 'Sorry, I had trouble processing your request.';\n }\n\n
return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n};\n\n```\n\n#### 2. **Ensure No Duplicate Code**:\nRemove any duplicate
`GetAIResponseIntentHandler` definitions. This can lead to confusion and issues during
execution.\n\n#### 3. **Check OpenAI API Call**:\nTest the OpenAI API call separately
to ensure it's returning a valid response. If the API call is slow or returning
an unexpected error, add timeout handling and make sure the skill gracefully responds
if the call fails:\n\n```\njavascript\n// Ensure OpenAI API call resolves or times out
properly\nasync function getAIResponse(query) {\n return Promise.race([\n
makeOpenAIRequest(query),\n new Promise((_, reject) =>{\n
setTimeout(() => reject(new Error('Timeout')), 7000) // Timeout after 7
seconds\n]);\n }]);\n}\n\n```\n\n#### 4. **Check Skill JSON Response**:\nEnsure
the JSON response returned by your skill is valid and conforms to Alexa's
expected response structure.\n\n#### 5. **Handle APL (Alexa Presentation
Language)**:\nThe logs show some references to APL (visual responses), which suggests
your device supports APL but your skill might not be properly handling it. You can
either handle APL content by returning APL documents, or ensure your skill works even
without APL content.\n\n### Key Debugging Steps:\n- Add more logging to verify where
the request is breaking.\n- Check that your `GetAIResponseIntentHandler` is correctly
being triggered and completing successfully.\n- Test your OpenAI API call separately
to confirm it's functioning correctly.\n\n### Final Thought:\nIt looks like the
skill is defaulting to Alexa's built-in responses because the skill's custom
logic (especially around the API call) isn't returning a valid response. Once you
ensure your `GetAIResponseIntentHandler` is fully functional and returning a response,
the skill should behave as expected.]]], "status": "finished_successfully", "end_turn":
true
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa2477f-f05e-439c-8422-b6eda2elf0d5", "request_id": "8d1bc50d9eaf0e65-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa2477f-f05e-439c-8422-b6eda2elf0d5", "children": [{"aaa2dc27-4be9-42ff-af12-
80c5ba77c436"}], "aaa2dc27-4be9-42ff-af12-80c5ba77c436": {"id": "aaa2dc27-4be9-42ff-
af12-80c5ba77c436", "message": {"id": "aaa2dc27-4be9-42ff-af12-80c5ba77c436", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728785331.393396,
"update_time": null, "content": {"content_type": "text", "parts": ["okay give me the
complete bit of code getAIResp... should have and I'll just cut the

```

```

function/const out and replace it with the code you give me"]], "status":
"finished_successfully", "end_turn": null, "weight": 1.0, "metadata":
{"serialization_metadata": {"custom_symbol_offsets": []}, "request_id":
"8d1bc8be8a3c0e65-MXP", "message_source": null, "timestamp": "absolute",
"message_type": null, "recipient": "all", "channel": null, "parent": "64497db0-9676-
4741-af09-ef99ca5d8ffb", "children": [{"id": "5644436c-aede-4f65-b186-da7578d578e9"}],
"5644436c-aede-4f65-b186-da7578d578e9": {"id": "5644436c-aede-4f65-b186-da7578d578e9",
"message": {"id": "5644436c-aede-4f65-b186-da7578d578e9", "author": {"role":
"assistant", "name": null, "metadata": {}}}, "create_time": 1728785348.971836,
"update_time": null, "content": {"content_type": "text", "parts": ["Here is the
complete and clean `GetAIResponseIntentHandler` code that you can use to replace your
current implementation. This version includes logging, error handling, and API call to
OpenAI, ensuring it handles the slot properly and provides a fallback if the API call
fails.\n\n```\njavascript\nconst GetAIResponseIntentHandler = {\n
canHandle(handlerInput) {\n console.log('Checking if
GetAIResponseIntentHandler can handle the request');\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'.\n
'GetAIResponseIntent'.\n },\n async handle(handlerInput) {\n
console.log('GetAIResponseIntentHandler is handling the request');\n
const { intent } = handlerInput.requestEnvelope.request;\n // Check if the
slot is filled\n if (!intent.slots || !intent.slots.query
|| !intent.slots.query.value) {\n const speakOutput = "I didn't
catch that. Can you please ask me a question like, 'What is quantum
mechanics?'";\n console.log('Slot value is empty or
undefined.');\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .re
prompt(speakOutput)\n .getResponse();\n }\n }\n let
aiResponse = '';\n try {\n // Log the query being sent to
OpenAI\n console.log('Sending query to OpenAI:',\n
intent.slots.query.value);\n aiResponse = await
getAIResponse(intent.slots.query.value); // Call OpenAI API\n
console.log('Received response from OpenAI:', aiResponse);\n } catch
(error) {\n console.log('Error occurred while calling OpenAI
API:', error);\n aiResponse = 'Sorry, I had trouble processing
your request.';\n }\n // Return the AI response or error
message\n return
handlerInput.responseBuilder\n .speak(aiResponse)\n .getResponse
();\n }\n};\n\n\n### Key Points:\n1. Logging: This code logs useful
information, like whether the handler can handle the intent and what query is being
sent to the OpenAI API. This helps in debugging.\n2. Slot Checking: It checks
whether the `query` slot exists and has a value. If not, it returns a fallback
response.\n3. Error Handling: If the OpenAI API call fails for any reason, the
user is informed with a friendly error message.\n4. Response Handling: It returns
a response using Alexa's `responseBuilder`.\n\nYou can now replace the existing
`GetAIResponseIntentHandler` in your code with this version. This should handle the
OpenAI API call and process the user's query effectively."}], "status":
"finished_successfully", "end_turn": true

```

```
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa2dc27-4be9-42ff-af12-80c5ba77c436", "request_id": "8d1bc8be8a3c0e65-
MXP", "timestamp": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa2dc27-4be9-42ff-af12-80c5ba77c436", "children": ["aaa2f06f-a5c2-4480-bf14-
c70cedeeb89d"], "aaa2f06f-a5c2-4480-bf14-c70cedeeb89d": {"id": "aaa2f06f-a5c2-4480-
bf14-c70cedeeb89d", "message": {"id": "aaa2f06f-a5c2-4480-bf14-c70cedeeb89d", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728785747.879479,
"update_time": null, "content": {"content_type": "text", "parts": [{"\n
"event"; {\n "header"; {\n "namespace";
"Text";,\n "name";: "TextMessage";,\n
"messageId";: "messageId";,\n "dialogRequestId";:
"23d907c6-8dd4-4d6f-9a72-6c329f03fa32";\n },\n
"payload";: {\n "textMessage";: "tell me about the
intersection between cops and donuts";\n },\n "context";:
[\n {\n "header";: {\n "namespace";:
"System";,\n "name";: "SettingsState";,\n
"payloadVersion";: "1";\n },\n
"payload";: {\n "settings";: [\n
{\n "key";:
"com.amazon.alexa.characteristics.viewport.experiences";,\n
"value";:
"[{\"arcMinuteWidth";:\"346";,\"arcMinuteHeight";:\"216";,\"canRotate";:\">false";,\"canRe
size";:\">false";}]\";\n },\n
{\n "key";:
"com.amazon.alexa.characteristics.viewport.shape";,\n
"value";: "RECTANGLE";\n },\n
{\n "key";:
"com.amazon.alexa.characteristics.viewport.pixelWidth";,\n
"value";: "1280";\n },\n {\n
"key";: "com.amazon.alexa.characteristics.viewport.pixelHeight";,\n
"value";: "800";\n },\n {\n
"key";: "com.amazon.alexa.characteristics.viewport.dpi";,\n
"value";: "213";\n },\n {\n
"key";:
"com.amazon.alexa.characteristics.viewport.currentPixelWidth";,\n
"value";: "1280";\n },\n {\n
"key";:
"com.amazon.alexa.characteristics.viewport.currentPixelHeight";,\n
"value";: "800";\n },\n {\n
"key";: "com.amazon.alexa.characteristics.viewport.touch";,\n
"value";: "[\"SINGLE";]\";\n },\n
{\n "key";:
"com.amazon.alexa.characteristics.viewport.video";,\n
"value";: "{\"codecs";:
[\"H_264_42";,\"H_264_41";]}\";\n },\n
{\n "key";:
```





```

api.request.67f18faa-715a-4839-b66d-ec4e054e2b45"; \n "type";:
"SkillExecutionInfo"; \n "content";: {\n
"invocationRequest";: {\n "endpoint";:
"arn:aws:lambda:eu-west-1:975050070604:function:4e3cd483-a6cc-4494-b190-
11c9b68be961:Release_0"; \n "body";: {\n
"version";: "1.0"; \n "session";: {\n
"new";: false, \n "sessionId";:
"amzn1.echo-api.session.cf80e9d0-40a7-4409-87e6-1d9fb65abbe8"; \n
"application";: {\n "applicationId";:
"amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961"; \n }, \n
"attributes";: {}, \n "user";: {\n
"userId";:
"amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVO7KWN2ER26Z234DMXDGSQS4B4A5XMYL6FSWMB7D3BFXS7II5EI2
KOIZDI5FWQRR2YMT4VQHW6ERQTAW02K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ"; \n "permissions";: {\n
"scopes";: {\n
"alexa::devices:all:geolocation:read";: {\n
"status";:
"DENIED"; \n } \n
 } \n } \n } \n
 , \n "context";: {\n
"Viewports";: [\n {\n
"type";: "APL"; \n "id";:
"medHub"; \n "shape";:
"RECTANGLE"; \n "dpi";: 160, \n
"presentationType";: "OVERLAY"; \n
"canRotate";: false, \n
"configuration";: {\n "current";:
{\n "mode";: "HUB"; \n
"video";: {\n "codecs";:
[\n "H_264_42"; \n
"H_264_41"; \n] \n
 }, \n "size";:
{\n "type";:
"DISCRETE"; \n
"pixelWidth";: 1280, \n
"pixelHeight";:
800 \n } \n } \n
], \n "Viewport";: {\n
"experiences";: [\n {\n
"arcMinuteWidth";: 221, \n
"arcMinuteHeight";: 162, \n
"canRotate";: false, \n
"canResize";:
false \n } \n], \n
"mode";: "HUB"; \n "shape";:

```

```
"RECTANGLE"; \n
"pixelHeight"; 800, \n
"currentPixelWidth"; 1280, \n
"currentPixelHeight"; 800, \n
\"touch";: \n
\"SINGLE"; \n
"keyboard"; \" \n
"DIRECTION"; \n
"video"; \" \n
"codecs"; \" \n
"H_264_42"; \n
"H_264_41"; \n
\"Extensions"; \" \n
"available"; \" \n
"aplext:backstack:10";: \n
\" \n
\"System"; \" \n
"application"; \" \n
"applicationId"; "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-11c9b68be961"; \n
"user"; \" \n
"userId"; \n
"amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVOU7KWN2ER26Z234DMXDGSS4B4A5XMYL6FSWMB7D3BFXS7II5EI2KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27WQ"; \n
"permissions"; \" \n
"scopes"; \" \n
"alexa::devices:all:geolocation:read"; \" \n
"status";: \n
"DENIED"; \n
\" \n
\" \n
"device"; \" \n
"deviceId";: \n
"amzn1.ask.device.AMAXRWYUCF3SDX4KFOK4JR3J2LBF4R5I5CM70LTUQFW2JXXKD6RWEGOMWI6RG5UXMITSMS4UY6JVTa20YWS7QAC7TGT5OUMHLI4MHFWQAHZJKLGHZJHCQ3GROHVFZ5K5GG2FQIAI7UHBTO6PYNAU2SE6CYHVWUGDS43ZPD2QON5IX52F545Q3W52JY5LFD43TP5K2WNYDFCAT72T"; \n
"supportedInterfaces"; \" \n
"apiEndpoint"; "https://api.eu.amazonalexa.com"; \n
"apiAccessToken";: \n
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjEiEifQ.eyJhdWQiOiJodHRwczovL2FwaS51dS5hbWF6b25hbGV4YS5jb20iLCJpc3MiOiJBbGV4YVNaWxsS210Iiwic3ViIjoieWV1b6bEuYXNraWxsLjRlM2NkNDgzLWE2Y2MtNDQ5NC1mTkwLTEXyZlInJhZTktM2MSIsImV4cCI6MTcyODc4NTU0NCwiaWF0IjojNzI4Nzg1NDg0LCJ1YmYiOiJlM2NkNDgzODU0ODQsInByaXZhdGVkbGFpbXMiOnsiaXNgb3JSQSI6ImZhbnHN1Iiwibm9uThdhU2NvcGVzIjoieWxleGE60nBlcnNvb19pZDpyZWFKIiwiaWF0Ij0y29udGV4dCI6IkkFBQUBFBQUBQVFCa2d4TXo2azZGVy9ZWVE0a3JUMW11ZUFFQUBQUBQUM3R1Bmbk5LUmwzVTZReThQcTVTd01ZZExEmktZcG1BWXPmeVRXL1VEeTRsNH1GemNHbytSQSt3MlZXZ3hwZm9SdDdDeX10v1kOUjNxmFNBNmhqcU10cXNYVVBhQV10cmYxZXg4S1ZZUjFLOVNYQTF3VVUrK0pZbEO3anBWNvK4SkxYmVdGbk1VYmNOYUQ4UXU5a3lMMGOzSDZwZ1FoWGRjmwZkOFNsTzRVOUNb1BSQWtMykJKSD1BNjcrZz12bW04MH02b0tRa21OKzY1SXZIbDvZu31Bd2x0VStiSHIzdkVPT29ERW1sc254bkFmY1oxUTd5YW9xb0VVRjh0VnNmOVcvcmsxVXB5aHF3aFVSZXFVXZzQkg4MUpoZVI0OF1iRXh1VmR1T3VSe1gyZX1DcWNBrdYrcXNOV0gwa3NZRXVOeEpZaFc1eFRcVklNM0xpMW1sU2gzOHA3U1ZRskIxc2xCaGQwRk4ydmNVMONBLONUS21iVH1sNkp5NzhJVvhIZmxndH1DWUtKMZlndXR3YkxTYVf2b0oyeW12dHBQOURtdUNTWnc5SkFNTWNtZDA3VmdudDZPd1BxRkt0M1NqQW1Mc2p0Vjg4bmxjUVBSY2JPZE9sQUFtTXJXTjNScFJ5U2RZVFBtVzVMTnphN0JwQUVnOU8iLCJkZXZpY2VJZCI6ImFtem4xLmFzay5kZXZpY2UuQU1BWfJXWVVRjNTRfG0SOZPSzRkuJNKMk
```

CRjRSNuk1Q003TOxUVVFGVzJKWEtYRDZSVOVHT01XSTZSRzVVWE1JVFNNUzRVWTZKV1RBMk9ZV1M3UUFDN1RHV  
DVPVU1ITEk0TUhGV1FBSFpKS0xHSFpKSENRMOdStOhWR1o1SzVHRzJGUU1BSTdVSEJUTzZQWU5BVTJTRTZDWUh  
WVVdHRFMOM1pQRDJRT041SVg1MkY1NDVRM1c1MkpZNUxGRDQzVFA1SzJXT11ERkNBVDcyVCIsInVzZXJJZCI6I  
mFtem4xLmFzay5hY2NvdW50LkFNQTVKNU1QQU5HWEVXQ1BXM1NIQUxVMk5CNERIWFUzQkpBWU1RUUg1VU1RQVh  
QT11VQUNPSUVWRFRPrzIyQOVOM1NDRjdLR0Y1UkpTVUNUN1pCTEs0M1RUMORIWKZOV1VPNOtXTjJFUjI2WjIzN  
ERNWERHU1FTNEIOQTVYTVZMNkZTV01CNOQzQkZYUzdJSTVFSTJLT01aREk1R1dRU1IyWU1UNFZRSFc2RVJRVEF  
XTzJLM1VOUzJPSkMzTTJITjRFSUdQU0wyUDY3WUNNT0wOU1pCVURENFJKRFpZSDVZM1RMNEdJN00yN1dRI19.  
jB\_7Uxi07TTMiPUjd0oYiQFC8SJuf8NAZTxoRJjJzvwrdfU-  
aKeCtr93X5K6xBZWQn7\_pN33wk39GfIG\_RKYS7Mv6PEqEsN8D-  
urIApAtLR6SPgLcmtCqHQLM7AGfRgXrbiqfystBFHhQAXnPrqF1likjXULPJPGs8hrI1iCz0xy00RZQjcowTR-  
D04Roq9cNaVEOr1fjYGQEqGAODq65Cw9ccx9UEdRuhj0EwdDEStcUYLDZ8PBnBhcisbhWh6epRjopYZhHWkOCn  
aRwpX1ZJbMIvjW-lgsjh-\_8gDLUPZQ5BSUBENCXKrvzyn0E6ZEwnZ8mN-  
gvNSRtpXTCvC07Q&quot;\n } \n }, \n  
&quot;request&quot;; { \n &quot;type&quot;;:  
&quot;SessionEndedRequest&quot;; \n &quot;requestId&quot;;:  
&quot;amzn1.echo-api.request.67f18faa-715a-4839-b66d-ec4e054e2b45&quot;; \n  
&quot;timestamp&quot;;: &quot;2024-10-13T02:11:25Z&quot;; \n  
&quot;locale&quot;;: &quot;en-US&quot;; \n &quot;reason&quot;;:  
&quot;ERROR&quot;; \n &quot;error&quot;;: { \n  
&quot;type&quot;;: &quot;INVALID\_RESPONSE&quot;; \n  
&quot;message&quot;;: &quot;An exception occurred while dispatching the request to the  
skill.&quot;; \n } \n } \n } \n  
}, \n &quot;invocationResponse&quot;;: { \n  
&quot;body&quot;;: { \n &quot;version&quot;;: &quot;1.0&quot;; \n  
&quot;response&quot;;: { \n &quot;type&quot;;:  
&quot;\_DEFAULT\_RESPONSE&quot;; \n }, \n  
&quot;sessionAttributes&quot;;: {}, \n &quot;userAgent&quot;;:  
&quot;ask-node/2.14.0 Node/v16.20.2 sample/hello-  
world/v1.2&quot;; \n } \n }, \n &quot;metrics&quot;;:  
{ \n &quot;skillExecutionTimeInMilliseconds&quot;;:  
47 \n } \n } \n } \n } \n { \n &quot;version&quot;;:  
&quot;1.0&quot;; \n &quot;session&quot;;: { \n &quot;new&quot;;:  
false, \n &quot;sessionId&quot;;: &quot;amzn1.echo-api.session.cf80e9d0-40a7-4409-  
87e6-1d9fb65abbe8&quot;; \n &quot;application&quot;;:  
{ \n &quot;applicationId&quot;;: &quot;amzn1.ask.skill.4e3cd483-a6cc-4494-b190-  
11c9b68be961&quot;; \n &quot;attributes&quot;;: {}, \n &quot;user&quot;;:  
{ \n &quot;userId&quot;;:  
&quot;amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22  
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNvU07KWN2ER26Z234DMXDGSQS4B4A5XmVL6FSWMB7D3BFXS7II5EI2  
KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27  
WQ&quot;; \n &quot;permissions&quot;;: { \n &quot;scopes&quot;;:  
{ \n &quot;alexa::devices:all:geolocation:read&quot;;:  
{ \n &quot;status&quot;;:  
&quot;DENIED&quot;; \n } \n } \n } \n } \n } \n &quot;affiliatedResou  
rces&quot;;: [] \n } \n &quot;context&quot;;: { \n &quot;Viewports&quot;;:  
[ \n &quot;type&quot;;: &quot;APL&quot;; \n &quot;id&quot;;:  
&quot;medHub&quot;; \n &quot;shape&quot;;:  
&quot;RECTANGLE&quot;; \n &quot;dpi&quot;;:  
160, \n &quot;presentationType&quot;;:

[illegible]

sNH1GemNHbytSQst3M1ZXZ3hwZm9SdDdDeXl0v1k0UjNxFNBNmhqcU10cXNYVBhQV10cmYxZXg4S1ZZUjFLO  
VNYQTF3VVUrK0pZbE03anBWNvK4SkxYmVdGbk1VYmNOYUQ4UXU5a31MMG0zSDZwZ1FoWGRjMVZk0FNsTzRVOUN  
Nb1BSQWtMYkJKSD1BNjcrZz12bW04MH02b0tRa210KzY1SXZ1bDVzU31Bd2x0VStiSHIzdkVPT29ERW1sc254b  
kFmY1oxUTd5YW9xb0VVRjh0VnNmOVcvcmsxVXB5aHF3aFVSZXFVXZZQkg4MUpoZV100F1iRXh1VmR1T3VSe1g  
yZX1DcWNBRDYrcXNOV0gwa3NZRXV0eEpZaFc1eFRCVklNm0xpMW1sU2gzOHA3U1ZRSkIxc2xCaGQwRk4ydmNVM  
ONBLONUS21iVH1sNkp5NzhJVvhIZmxndH1DWUtkMz1ndXR3YkxTYVf2b0oyeW12dHBqOURtdUNTWnc5SkFNTWN  
TZDA3VmdudDZPd1BxRktOM1NqQW1Mc2pOVjg4bmxjUVBSY2JPZE9sQUftTXJXTjNScFJ5U2RZVFBtVzVMTnphN  
0JwQUVnOU8iLCJkZXZpY2VJZCI6ImFtem4xLmFzay5kZXZpY2UuQU1BWFJXWVVDJRjNTRFgOS0ZPSzRKUjNKMkx  
CRjRSNUk1Q003TOxUVVFGVzJKWEtYRDZSV0VHT01XSTZSRzVvWE1JVFNNuzRVWTZKV1RBMk9ZV1M3UUFDN1RHV  
DVPVU1ITEk0TUhGV1FBSFpKS0xHSFpKSENRM0dST0hWR1o1SzVHRzJGUU1BSTdVSEJUTzZQWU5BVTJTRTZDWU  
WVvdHRFMOM1pQRDJRT041SVg1MkY1NDVVM1c1MkpZNUxGRDQzVFA1SzJXT11ERkNBVDcyVCIsInVzZXJJZCI6I  
mFtem4xLmFzay5hY2NvdW50LkFNQTVKNU1QQU5HWEVXQ1BXM1NIQUxVMk5CNERIWFUzQkpBWU1RUUg1VU1RQVh  
QT11VQUNPSUVWRFRPRzIyQOVOM1NDRjdLROy1UkpTVUNUN1pCTEs0M1RUM0RIWkZOV1VPNOtXtjJfUji2WjiZn  
ERNWERHU1FTNEIOQTVYTVZMNkZTV01CNOQzQkZYUzdJSTVFSTJLT01aREk1R1dRU1IyWU1UNFZRSFc2RVJRVEF  
XTzJLM1VOUzJPSkMzTTJITjRFSudQU0wyUDY3WUNNT0wOU1pCVURENFJKRFPZSDVZM1RMNEdJN00yN1dRIn19.  
jB\_7Uxi07TTMiPUjd0oYiQFC8SJuf8NAZTxoRJJjzvwrdfU-  
aKeCtr93X5K6xBZWQn7\_pN33wk39GfIG\_RKYS7Mv6PEqEsN8D-  
urIApAtLR6SPgLcmtCqHQLM7AGfRgXrbiqfystBFHhQAXnPrqF1likjXULPJPGs8hrI1iCz0xy00RZQjcowTR-  
D04Roq9cNaVE0r1fjYQGQEqGAODq65Cw9ccx9UEdRuhjOEwddESTcUYLDZ8PBnBhcisbhWh6epRjopYZhHwKOCn  
aRwpX1ZJbMIvjw-1gsjh-8gDLUPZQ5BSUBENCXKrvzynOE6ZEwnZ8mN-  
gvNSrtpXTCvC07Q&quot;\n\t\t\n\t}, \n\t&quot;request&quot;: {\n\t\t&quot;type&quot;:  
&quot;IntentRequest&quot;, \n\t\t&quot;requestId&quot;: &quot;amzn1.echo-  
api.request.2f59dede-8792-4a58-a478-d7013159a40a&quot;, \n\t\t&quot;locale&quot;:  
&quot;en-US&quot;, \n\t\t&quot;timestamp&quot;: &quot;2024-10-  
13T02:11:24Z&quot;, \n\t\t&quot;intent&quot;: {\n\t\t\t&quot;name&quot;:  
&quot;GetAIResponseIntent&quot;, \n\t\t\t&quot;confirmationStatus&quot;:  
&quot;NONE&quot;, \n\t\t\t&quot;slots&quot;: {\n\t\t\t\t&quot;query&quot;:  
{\n\t\t\t\t\t&quot;name&quot;: &quot;query&quot;, \n\t\t\t\t\t&quot;value&quot;:  
&quot;the intersection between cops and  
donuts&quot;, \n\t\t\t\t\t&quot;confirmationStatus&quot;:  
&quot;NONE&quot;, \n\t\t\t\t\t&quot;source&quot;:  
&quot;USER&quot;, \n\t\t\t\t\t&quot;slotValue&quot;: {\n\t\t\t\t\t\t&quot;type&quot;:  
&quot;Simple&quot;, \n\t\t\t\t\t\t&quot;value&quot;: &quot;the intersection between  
cops and  
donuts&quot;;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;, \n\t\t\t\t\t\t&quot;dialogState&quot;:  
&quot;STARTED&quot;;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;version&quot;: &quot;1.0&quot;, \n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t&quot;body&quot;: {\n\t\t\t\t\t\t\t&quot;version&quot;:  
&quot;1.0&quot;, \n\t\t\t\t\t\t\t&quot;response&quot;: {\n\t\t\t\t\t\t\t\t&quot;type&quot;:  
&quot;\_DEFAULT\_RESPONSE&quot;;\n\t\t\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t\t\t&quot;sessionAttributes&quot;:  
{}, \n\t\t\t\t\t\t\t\t&quot;userAgent&quot;: &quot;ask-node/2.14.0 Node/v16.20.2 sample/hello-  
world/v1.2&quot;;\n\t\t\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t\t\t&quot;session&quot;: {\n\t\t\t\t\t\t\t\t\t&quot;new&quot;:  
false, \n\t\t\t\t\t\t\t\t\t&quot;sessionId&quot;: &quot;amzn1.echo-api.session.cf80e9d0-40a7-4409-  
87e6-1d9fb65abbe8&quot;, \n\t\t\t\t\t\t\t\t\t&quot;application&quot;:  
{\n\t\t\t\t\t\t\t\t\t\t&quot;applicationId&quot;: &quot;amzn1.ask.skill.4e3cd483-a6cc-4494-b190-  
11c9b68be961&quot;;\n\t\t\t\t\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t\t\t\t\t&quot;\n\t\t\t\t\t\t\t\t\t\t&quot;attributes&quot;: {}, \n\t\t\t\t\t\t\t\t\t\t&quot;user&quot;:  
{\n\t\t\t\t\t\t\t\t\t\t\t&quot;userId&quot;:  
&quot;amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22  
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVU07KWN2ER26Z234DMXDGSQS4B4A5XWVL6FSWMB7D3BFXS7115EI2  
KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27

[illegible]

[illegible]



4f65-b186-da7578d578e9", "children": ["076788b9-e373-474a-99fe-c5b86ba299dd"]},  
 "076788b9-e373-474a-99fe-c5b86ba299dd": {"id": "076788b9-e373-474a-99fe-c5b86ba299dd",  
 "message": {"id": "076788b9-e373-474a-99fe-c5b86ba299dd", "author": {"role":  
 "assistant", "name": null, "metadata": {}}, "create\_time": 1728785779.510758,  
 "update\_time": null, "content": {"content\_type": "text", "parts": ["Yes, it seems like  
 your skill is timing out while waiting for a response. This could happen for a few  
 reasons, including delays in calling the OpenAI API or other issues related to  
 processing time. Here are a few things you can do to address this:\n\n### 1.  
 \*\*Increase the Timeout for Lambda (if you're using Lambda)\*\*\nBy default, AWS  
 Lambda functions have a timeout of 3 seconds, but you can extend this up to 15 minutes.  
 You can increase the timeout in the AWS Lambda console:\n- Navigate to the Lambda  
 function for your Alexa skill.\n- Under the \*\*General Configuration\*\* section,  
 increase the \*\*timeout\*\* value to something higher, like 10 seconds, to give your  
 OpenAI API call time to complete.\n\n### 2. \*\*Ensure Efficient API Calls\*\*\nMake sure  
 your API call is optimized to respond quickly. A couple of things you can do:\n-  
 \*\*Reduce the `max\_tokens`\*\* in your OpenAI request to minimize the time it takes to  
 process the response. For example, change `max\_tokens: 100` to something like  
 `max\_tokens: 50`.\n- \*\*Use a faster OpenAI model\*\* if possible (e.g., `gpt-3.5-turbo`  
 is faster than `text-davinci-003`).\n\n### 3. \*\*Add a Fallback Timeout for API  
 Calls\*\*\nImplement a timeout in your OpenAI API call logic, so that if the API  
 doesn't respond within a certain time, your function will gracefully handle the  
 situation. Here's an example:\n\n```\njavascript\nfunction getAIResponse(query) {\n const apiKey = &#x27;YOUR\_OPENAI\_API\_KEY&#x27;;\n const options = {\n hostname: &#x27;api.openai.com&#x27;;\n path: &#x27;/v1/completions&#x27;;\n method: &#x27;POST&#x27;;\n headers: {\n &#x27;Authorization&#x27;: &#x27;Bearer \${apiKey}&#x27;;\n &#x27;Content-Type&#x27;: &#x27;application/json&#x27;;\n }\n };\n const body = JSON.stringify({\n model: &#x27;gpt-3.5-turbo&#x27;;\n prompt: query,\n max\_tokens: 50, // Reduce token size for faster response\n temperature: 0.7\n });\n return new Promise((resolve, reject) => {\n const req = https.request(options, (res) => {\n let data = &#x27;&#x27;;\n res.on(&#x27;data&#x27;, (chunk) => {\n data += chunk;\n });\n res.on(&#x27;end&#x27;, () => {\n const response = JSON.parse(data);\n resolve(response.choices[0].text.trim());\n });\n });\n req.on(&#x27;error&#x27;, (e) => {\n reject(e);\n });\n // Set a timeout for the API request (in milliseconds)\n req.setTimeout(5000, () => {\n req.abort(); // Abort the request if it takes longer than 5 seconds\n reject(new Error(&#x27;Request timeout&#x27;));\n });\n req.write(body);\n req.end();\n });\n}\n\nThis way, if the API request takes longer than 5 seconds, it will abort and handle the situation without timing out.\n\n### 4. \*\*Pre-process Long Operations\*\*\nIf you anticipate a long processing time (like analyzing a lengthy query), you could implement a 'processing' response. For example, Alexa could say, 'Let me think about that,' and then respond with the actual result once it is available.\n\n```\njavascript\nconst GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n return Alexa.getRequestType(handlerInput.requestEnvelope) === &#x27;IntentRequest&#x27; &amp; Alexa.getIntentName(handlerInput.requestEnvelope) === &#x27;GetAIResponseIntent&#x27;;\n },\n async handle(handlerInput) {\n

```

const query = handlerInput.requestEnvelope.request.intent.slots.query.value;\n
\n let speakOutput = 'Let me think about that for a moment.';\n
\n // Give Alexa a brief response before calling the API\n
handlerInput.responseBuilder.speak(speakOutput).getResponse();\n\n try {\n
const aiResponse = await getAIResponse(query);\n speakOutput =\n
aiResponse;\n } catch (error) {\n speakOutput = 'Sorry, I had\n
trouble getting an answer.';\n console.log('Error:';\n
error);\n }\n\n return\n
handlerInput.responseBuilder.speak(speakOutput).getResponse();\n }\n};\n```\n\nThis
will give the user an immediate response while the skill fetches the OpenAI API result
in the background.\n\n### 5. **Optimize DialogState Flow**\n\nYou can simplify your
interaction by checking the `dialogState` and ensuring that the dialog is managed
efficiently, especially for long-running queries. If Alexa is struggling with slot
values, you can use dialog management to handle these cases.\n\n### Conclusion:\n\nBy
increasing the Lambda timeout, optimizing API requests, and implementing a timeout for
API calls, you can mitigate the timing issues. These changes should help reduce skill
timeout and enhance responsiveness. Let me know if you need further clarification!"]],
"status": "finished_successfully", "end_turn": true
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa2f06f-a5c2-4480-bf14-c70cedeeb89d", "request_id": "8d1bd2e77e7474ee-
FC0", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa2f06f-a5c2-4480-bf14-c70cedeeb89d", "children": [{"aaa233b0-fd75-4eb9-89d5-
1b320eb4763b"}], "aaa233b0-fd75-4eb9-89d5-1b320eb4763b": {"id": "aaa233b0-fd75-4eb9-
89d5-1b320eb4763b", "message": {"id": "aaa233b0-fd75-4eb9-89d5-1b320eb4763b", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728785913.622929,
"update_time": null, "content": {"content_type": "text", "parts": ["lets assume a long
processing time and update the intent handler function to adress that and get logic
to keep waiting for that respon for open ai"]}, "status": "finished_successfully",
"end_turn": null, "weight": 1.0, "metadata": {"serialization_metadata":
{"custom_symbol_offsets": []}, "request_id": "8d1bd6f59b515255-MXP", "message_source":
null, "timestamp_": "absolute", "message_type": null}, "recipient": "all", "channel":
null}, "parent": "076788b9-e373-474a-99fe-c5b86ba299dd", "children": [{"4f4dac8d-7d4c-
4254-8c02-98c053b29fd1"}], "4f4dac8d-7d4c-4254-8c02-98c053b29fd1": {"id": "4f4dac8d-
7d4c-4254-8c02-98c053b29fd1", "message": {"id": "4f4dac8d-7d4c-4254-8c02-98c053b29fd1",
"author": {"role": "assistant", "name": null, "metadata": {}}, "create_time":
1728785963.277636, "update_time": null, "content": {"content_type": "text", "parts":
["If you expect long processing times for the OpenAI response, you can implement a
system where Alexa first acknowledges the request, gives an immediate placeholder
response, and then continues to process the response in the background. However, Alexa
Skills must return a response within a few seconds (the maximum response window is 8
seconds). To deal with long processing times, you can simulate a "waiting"
period or break up the interaction into multiple stages using session
attributes.\n\nHere\u2019s a strategy for handling long processing times while
ensuring a smooth user experience:\n\n1. **Immediate Acknowledgment**: Give the user a
message like \u201cLet me think about that for a moment.\u201d\n2. **Session
Management**: Store the query in session attributes, then end the current session.
When the OpenAI response is ready, the user can request the result again by invoking

```

the skill.

3. **Progress Check Mechanism**: Create a new intent where the user asks Alexa for the result after some time.

### Step-by-Step Implementation

#### 1. Update the `GetAIResponseIntentHandler` to handle long processing times

This will provide an immediate response, store the query in the session, and instruct the user to come back for the result.

```

javascript
const GetAIResponseIntentHandler = {
 canHandle(handlerInput) {
 return
 Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest;
 && Alexa.getIntentName(handlerInput.requestEnvelope) ===
 GetAIResponseIntent;
 },
 handle(handlerInput) {
 const query = handlerInput.requestEnvelope.request.intent.slots.query.value;
 // Store the query in session attributes so we can use it later
 const sessionAttributes = handlerInput.attributesManager.getSessionAttributes();
 sessionAttributes.query = query;
 handlerInput.attributesManager.setSessionAttributes(sessionAttributes);
 // Respond to the user that Alexa is thinking
 const speakOutput = `Let me think about that for a moment. I'll have an answer for you soon. You can ask me for the result shortly.`;
 return handlerInput.responseBuilder
 .speak(speakOutput)
 .getResponse();
 }
};

```

#### 2. Add a new `GetResultIntentHandler` to retrieve the processed response later

This handler will be invoked by the user after waiting for a while (you can instruct them to say "Alexa, what's the result?" or something similar). It will check if the result from the OpenAI API is ready and provide it.

```

javascript
const GetResultIntentHandler = {
 canHandle(handlerInput) {
 return
 Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest;
 && Alexa.getIntentName(handlerInput.requestEnvelope) ===
 GetResultIntent;
 },
 async handle(handlerInput) {
 const sessionAttributes = handlerInput.attributesManager.getSessionAttributes();
 const query = sessionAttributes.query;
 // If no query was previously set, inform the user
 if (!query) {
 const speakOutput = `I don't have anything to look up. You can ask me something first.`;
 return handlerInput.responseBuilder
 .speak(speakOutput)
 .getResponse();
 }
 let aiResponse = null;
 try {
 aiResponse = await getAIResponse(query); // Call OpenAI API
 } catch (error) {
 console.log(`Error occurred:`, error);
 aiResponse = `Sorry, I had trouble processing your request.`;
 }
 // Return the AI response or error message
 const speakOutput = aiResponse ? aiResponse : `I'm still thinking. Please check back later.`;
 return handlerInput.responseBuilder
 .speak(speakOutput)
 .getResponse();
 }
};

```

#### 3. Example Alexa Interaction Model

You need to create a new intent in your interaction model called `GetResultIntent`. Here's a simple intent for the `GetResultIntent` to handle users asking for the response:

```

json
{
 "name": "GetResultIntent",
 "slots": [],
 "samples": [
 "What is the result?",
 "Do you have the answer?",
 "What's the result of my question?",
 "Do you have the response?",
 "Give me the answer."
]
}

```

#### 4.

Manage the OpenAI API Call

Here's a sample of the `getAIResponse` function that gets called when the user asks for the result later:

```

function
getAIResponse(query) {
 const apiKey = <YOUR_OPENAI_API_KEY>;
 const options = {
 hostname: <api.openai.com>;
 path: </v1/completions>;
 method: <POST>;
 headers: {
 <Authorization>: `Bearer ${apiKey}`,
 <Content-Type>: <application/json>;
 }
 };
 const body = JSON.stringify({
 model: <gpt-3.5-turbo>;
 prompt: query;
 max_tokens: 50, // Adjust tokens to limit processing time
 temperature: 0.7
 });
 return new Promise((resolve, reject) => {
 const req = https.request(options, (res) => {
 let data = <>;
 res.on(<data>;, (chunk) => {
 data += chunk;
 });
 res.on(<end>;, () => {
 try {
 const response = JSON.parse(data);
 resolve(response.choices[0].text.trim());
 } catch (e) {
 reject(new Error(<Failed to parse response from OpenAI>));
 }
 });
 });
 req.on(<error>;, (e) => reject(e));
 req.write(body);
 req.end();
 });
}

// 5. Add Handlers to the Skill Builder
Finally, make sure to add the new `GetResultIntentHandler` to the skill builder:
exports.handler =
 Alexa.SkillBuilders.custom()
 .addRequestHandlers([
 LaunchRequestHandler,
 GetAIResponseIntentHandler, // Add this handler
 GetResultIntentHandler, // Add this handler
 HelloWorldIntentHandler,
 HelpIntentHandler,
 CancelAndStopIntentHandler,
 FallbackIntentHandler,
 SessionEndedRequestHandler,
 IntentReflectorHandler
])
 .addErrorHandlers([
 ErrorHandler
])
 .lambda();

```

Final Interaction Flow

- User Query (GetAIResponseIntent)\*\*:** The user asks a question like "What's the intersection between cops and donuts?" - Alexa responds with something like "Let me think about that for a moment. I'll have an answer for you soon."
- Processing\*\*:** The session ends, and the backend processes the OpenAI API request in the background.
- User Requests Result (GetResultIntent)\*\*:** After a while, the user asks, "Do you have the result?" or something similar. - Alexa responds with the OpenAI-generated response (or tells the user to wait longer if the response isn't ready).

This approach will ensure that Alexa responds quickly, while giving the backend time to process more complex requests.

```

{
 "status": "finished_successfully",
 "end_turn": true,
 "weight": 1.0,
 "metadata": {
 "citations": [],
 "content_references": [],
 "gizmo_id": null,
 "message_type": null,
 "model_slug": "gpt-4o",
 "default_model_slug": "gpt-4o",
 "parent_id": "aaa233b0-fd75-4eb9-89d5-1b320eb4763b",
 "request_id": "8d1bd6f59b515255-MXP",
 "timestamp": "absolute",
 "finish_details": {
 "type": "stop",
 "stop_tokens": [200002]
 },
 "is_complete": true,
 "recipient": "all",
 "channel": null,
 "parent": "aaa233b0-fd75-4eb9-89d5-1b320eb4763b",
 "children": [
 "aaa28843-2c2b-4193-932c-c35ff7fcc552"
],
 "aaa28843-2c2b-4193-932c-c35ff7fcc552": {
 "id": "aaa28843-2c2b-4193-932c-c35ff7fcc552",
 "message": {
 "id": "aaa28843-2c2b-4193-932c-c35ff7fcc552",
 "author": {
 "role": "user",
 "name": null,
 "metadata": {}
 },
 "create_time": 1728786011.715942,
 "update_time": null,
 "content": {
 "content_type": "text",
 "parts": [
 "i only want the user to ask alexa once and then we do whatever in the background but alexa can only get

```

```

the one prompt per request"]}, "status": "finished_successfully", "end_turn": null,
"weight": 1.0, "metadata": {"serialization_metadata": {"custom_symbol_offsets": []},
"request_id": "8dlbd95abee45255-MXP", "message_source": null, "timestamp_": "absolute",
"message_type": null}, "recipient": "all", "channel": null}, "parent": "4f4dac8d-7d4c-
4254-8c02-98c053b29fd1", "children": [{"76c4f215-e092-432b-9c98-blaadcf19ce7"}],
"76c4f215-e092-432b-9c98-blaadcf19ce7": {"id": "76c4f215-e092-432b-9c98-blaadcf19ce7",
"message": {"id": "76c4f215-e092-432b-9c98-blaadcf19ce7", "author": {"role":
"assistant", "name": null, "metadata": {}}, "create_time": 1728786052.296628,
"update_time": null, "content": {"content_type": "text", "parts": ["Given that Alexa
can only respond to one request per session and cannot re-initiate interactions
without the user prompting it again, managing long-running tasks in the background is
challenging. However, here's a refined approach that keeps the interaction smooth
without forcing the user to ask again but also handles long processing time within
Alexa's constraints.\n\n### Approach: **Async Background Processing with Progress
Updates**\n\nWe will handle this by giving an initial response to the user to
acknowledge the request and inform them that processing is ongoing. Then, Alexa keeps
trying to fetch the result from OpenAI in the background until a result is available,
all within the same session.\n\nSince the session timeout for Alexa is approximately 8
seconds, you need to manage the processing within that limit or provide a
conversational progress update, so it appears as though Alexa is still
thinking.\n\n### Steps:\n\n1. **Acknowledge the Request**: Provide a quick
acknowledgment to the user that the request has been received and that Alexa is
processing it.\n\n2. **Progress Updates**: Use `reprompt()` to extend the session and
simulate waiting. This way, you can give the user progress updates or inform them that
more time is needed.\n\n3. **Final Response**: Once OpenAI's response is ready
(within the session), Alexa will deliver the result.\n\n### Updated
`GetAIResponseIntentHandler` Example:\n\n```javascript\nconst
GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest;\n },\n && Alexa.getIntentName(handlerInput.requestEnvelope) ===
IntentRequest;\n },\n async handle(handlerInput) {\n const query = handlerInput.requestEnvelope.request.intent.slots.query.value;\n // Respond to the user that Alexa is thinking\n let speakOutput =
`"Let me think about that for a moment."`;
 try {\n // Send the query to OpenAI\n const aiResponse = await
getAIResponse(query);\n // If the response arrives in time, Alexa
provides the final response\n speakOutput = `Here is what I found:
${aiResponse}`;\n } catch (error) {\n // If there's an error or
time out, inform the user\n console.log(`Error occurred while calling
OpenAI API:`, error);\n speakOutput = `"Sorry, I had trouble
processing your request."`;\n }\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .getResponse
e();\n }\n};\n```\n\n### Progressive Update Using Reprompt (if longer processing is
needed):\n\nYou can manage longer tasks with periodic responses from Alexa to simulate
waiting (though this has limitations within Alexa's
timeout).\n\n```javascript\nconst GetAIResponseIntentHandler = {\n canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest;\n },\n && Alexa.getIntentName(handlerInput.requestEnvelope) ===

```

```

 }, \n async handle(handlerInput) {\n
const query = handlerInput.requestEnvelope.request.intent.slots.query.value;\n
\n // Initial response to acknowledge\n let speakOutput = "Let me
think about that for a moment.";\n\n // Provide a reprompt to simulate
waiting\n let repromptOutput = "Still working on it...";\n\n
// Build a response to give a progress update\n const responseBuilder =
handlerInput.responseBuilder.speak(speakOutput).reprompt(repromptOutput);\n\n
// Attempt to get the AI response\n try {\n // Here, we're
simulating a slightly longer process by waiting a couple of seconds\n const
aiResponse = await getAIResponse(query);\n \n // If we have the
response in time, update the response and clear the reprompt\n speakOutput
= `Here is what I found: ${aiResponse}`;\n return
responseBuilder\n .speak(speakOutput)\n .getResponse();
\n // Final response\n } catch (error) {\n // If there's an error
or time out, inform the user\n console.log('Error occurred while
calling OpenAI API:', error);\n speakOutput = "Sorry, I had
trouble processing your request.";\n return
handlerInput.responseBuilder\n .speak(speakOutput)\n .ge
tResponse();\n }\n }\n}\n\n### Handling the OpenAI Response
(getAIResponse` Function)\n\nMake sure that your `getAIResponse` function handles API
calls efficiently and provides error handling for longer
requests:\n\n```\nfunction getAIResponse(query) {\n const apiKey =
 'YOUR_OPENAI_API_KEY';\n const options = {\n hostname:
 'api.openai.com';\n path: '/v1/completions';\n
method: 'POST';\n headers: {\n 'Authorization';:
 `Bearer ${apiKey}`, \n 'Content-Type';:
 'application/json';\n }\n }; \n\n const body = JSON.stringify({\n
model: 'gpt-3.5-turbo';\n prompt: query, \n max_tokens: 50, //
Adjust this for longer responses\n temperature: 0.7\n }); \n\n return new
Promise((resolve, reject) => {\n const req = https.request(options, (res)
=> {\n let data = '';\n res.on('data';,
(chunk) => {\n data += chunk;\n });\n
res.on('end';, () => {\n try {\n const
response = JSON.parse(data);\n
resolve(response.choices[0].text.trim());\n } catch (e) {\n
reject(new Error('Failed to parse response from
OpenAI'));\n }\n });\n
req.on('error';, (e) => {\n reject(e);\n });\n
req.write(body);\n req.end();\n });\n });\n}\n\n### Additional Notes\n\n-
Max Timeout: You can't hold the user session for too long in Alexa, so ensure
that your requests to OpenAI return within a few seconds. If you expect the request to
take longer, you may need to break up the interaction into multiple sessions (though
this goes against your preference for a single request).\n\n- **Filling Time with
Reprompt**: Alexa's reprompt mechanism is crucial for extending sessions. Use
reprompt effectively to simulate thinking or progress, but keep in mind there's
still a hard time limit (~8 seconds) to finish the conversation.\n\nThis approach
ensures that the user can give their prompt once, and Alexa will either provide a
quick response or handle a slightly longer task gracefully within its session
constraints."}], "status": "finished_successfully", "end_turn": true

```

```
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa28843-2c2b-4193-932c-c35ff7fcc552", "request_id": "8d1bd95abee45255-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa28843-2c2b-4193-932c-c35ff7fcc552", "children": ["aaa219a6-5c45-4004-8044-
609b9dd94e3d"], "aaa219a6-5c45-4004-8044-609b9dd94e3d": {"id": "aaa219a6-5c45-4004-
8044-609b9dd94e3d", "message": {"id": "aaa219a6-5c45-4004-8044-609b9dd94e3d", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728786438.246884,
"update_time": null, "content": {"content_type": "text", "parts": [{"\n
"header"; {\n
"namespace"; "SkillDebugger";\n
"name"; "CaptureDebuggingInfo";\n
"3e36fa93-4747-4a7d-9076-93e848f10120";\n
"skillId"; "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961";\n
"timestamp"; "2024-10-
13T02:23:54.388Z";\n
"dialogRequestId"; "c1e9834b-f440-
4b1b-a481-86996a014bf5";\n
"skillRequestId"; "amzn1.echo-
api.request.5f5169a3-a499-43b7-97db-39114d9be490";\n
"type";
"SkillExecutionInfo";\n
"content"; {\n
"invocationRequest"; {\n
"arn:aws:lambda:eu-west-1:975050070604:function:4e3cd483-a6cc-4494-b190-
11c9b68be961:Release_0";\n
"body"; {\n
"version"; "1.0";\n
"new"; false,\n
"amzn1.echo-api.session.531d5cca-7e3e-4084-bf31-9a804df20938";\n
"application"; {\n
"amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961";\n
"attributes"; {},\n
"userId";
"amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVU07KWN2ER26Z234DMXDGSQS4B4A5XMVL6FSWMB7D3BFXS7II5EI2
KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ";\n
"scopes"; {\n
"alexa::devices:all:geolocation:read"; {\n
"status";
"DENIED";\n
"affiliatedResources"; []\n
"context"; {\n
"id"; "medHub";\n
"RECTANGLE";\n
"presentationType"; "OVERLAY";\n
"canRotate"; false,\n
"configuration"; {\n
"video"; {\n
"type"; "APL";\n
"shape";
"dpi"; 160,\n
"current";
"mode"; "HUB";\n
"codecs";
```

```
[\\n
 "H_264_41";\\n
 },\\n
 {\\n
 "type";:
 "DISCRETE";,\\n
 "pixelWidth";: 1280,\\n
 "pixelHeight";:
 800\\n
 }\\n
 }\\n
],\\n
 "Viewport";: {\\n
 "experiences";: [\\n
 "arcMinuteWidth";: 221,\\n
 "arcMinuteHeight";: 162,\\n
 "canRotate";: false,\\n
 "canResize";:
 false\\n
]\\n
 "mode";: "HUB";,\\n
 "RECTANGLE";,\\n
 "pixelWidth";: 1280,\\n
 "pixelHeight";: 800,\\n
 "currentPixelWidth";: 1280,\\n
 "currentPixelHeight";: 800,\\n
 "touch";:
 [\\n
 "SINGLE";,\\n
 "keyboard";: [\\n
 "DIRECTION";,\\n
 "video";: {\\n
 "codecs";: [\\n
 "H_264_42";,\\n
 "H_264_41";,\\n
]\\n
 }\\n
],\\n
 "available";: {\\n
 "aplex:backstack:10";:
 {}\\n
 }\\n
 "Advertising";: {\\n
 "advertisingId";:
 "00000000-0000-0000-0000-000000000000";,\\n
 "limitAdTracking";: true\\n
 },\\n
 "System";: {\\n
 "application";: {\\n
 "applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-11c9b68be961";,\\n
 "user";: {\\n
 "userId";:
 "amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDT0G22CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHZFNVOU7KWN2ER26Z234DMXDGSQS4B4A5XMYL6FSWMB7D3BFXS7II5EI2KOIZDI5FWQRR2YMT4VQHW6ERQTAW02K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27WQ";,\\n
 }\\n
 }\\n
 },\\n
 "permissions";: {\\n
 "scopes";: {\\n
 "alexa::devices:all:geolocation:read";: {\\n
 "status";:
 "DENIED";,\\n
 }\\n
 }\\n
 },\\n
 "device";: {\\n
 "deviceId";:
```



[illegible]

```

api.request.f2eela71-80cc-4266-9302-81f668b68478"; \n "code";:
"SKILL_ENDPOINT_ERROR"; \n "description";: "Skill
execution returned an exception for requestId amzn1.echo-api.request.f2eela71-80cc-
4266-9302-81f668b68478"; \n "debuggingInfo";: {\n
"type";: "SkillExecutionInfo"; \n "content";:
{\n "invocationRequest";: {\n
"endpoint";: "arn:aws:lambda:eu-west-1:975050070604:function:4e3cd483-
a6cc-4494-b190-11c9b68be961:Release_0"; \n "body";:
{\n "version";: "1.0"; \n
"session";: {\n "new";: false, \n
"sessionId";: "amzn1.echo-api.session.531d5cca-7e3e-4084-bf31-
9a804df20938"; \n "application";: {\n
"applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961"; \n }, \n
"attributes";: {}, \n "user";: {\n
"userId";:
"amzn1.ask.account.AMA5J5IPANGXEWCPW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOIEVDTOG22
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHFVNU07KWN2ER26Z234DMXDGSQS4B4A5XMYL6FSWMB7D3BFXS7I15EI2
KOIZDI5FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ"; \n "permissions";: {\n
"scopes";: {\n
"alexa::devices:all:geolocation:read";: {\n
"status";:
"DENIED"; \n } \n
 } \n }, \n
"affiliatedResources";: [] \n }, \n
"context";: {\n "Viewports";: [\n
{\n "type";: "APL"; \n
"id";: "medHub"; \n
"shape";: "RECTANGLE"; \n
"dpi";: 160, \n
"presentationType";: "OVERLAY"; \n
"canRotate";: false, \n
"configuration";: {\n
"current";: {\n "mode";:
"HUB"; \n "video";: {\n
"codecs";: [\n
"H_264_42"; \n
"H_264_41"; \n] \n
 }, \n
"size";: {\n "type";:
"DISCRETE"; \n
"pixelWidth";: 1280, \n
"pixelHeight";:
800 \n } \n
 } \n }, \n
"experiences";: [\n "Viewport";: {\n
"arcMinuteWidth";: 221, \n

```

```

"arcMinuteHeight";: 162, \n
"canRotate";: false, \n
"canResize";:
false \n
"mode";: "HUB";, \n
"RECTANGLE";, \n
1280, \n
"dpi";: 160, \n
1280, \n
"touch";: [\n
"SINGLE";, \n
"keyboard";: [\n
"DIRECTION";, \n
"video";: { \n
"H_264_42";, \n
"H_264_41";, \n
} \n
"Extensions";: { \n
"aplext:backstack:10";:
{} \n
"Advertising";: { \n
"00000000-0000-0000-0000-000000000000";, \n
"limitAdTracking";: true \n
"System";: { \n
"applicationId";: "amzn1.ask.skill.4e3cd483-a6cc-4494-b190-
11c9b68be961";, \n
"user";: { \n
"amzn1.ask.account.AMA5J5IPANGXEWC PW3SHALU2NB4DHXU3BJAYIQQH5UIQAXPOYUACOI EVDTOG22
CEN2SCF7KGF5RJSUCT7ZBLK43TT3DHFVNU07KWN2ER26Z234DMXDGSQS4B4A5XVML6FSWMB7D3BFXS7II5EI2
KOIZD15FWQRR2YMT4VQHW6ERQTAWO2K3UNS20JC3M2HN4EIGPSL2P67YCMOL4RZBUDD4RJDZYH5Y2TL4GI7M27
WQ";, \n
"scopes";: { \n
"alexa::devices:all:geolocation:read";: { \n
"status";:
"DENIED";, \n
} \n
}, \n
"deviceId";:
"device";: { \n
"amzn1.ask.device.AMAXRWYUCF3SDX4KFOK4JR3J2LBF4R5I5CM70LTUQFW2JXKXD6RWE GOMWI6RG5U
XMITSMS4UY6JVTA20YWS7QAC7TGT50UHLI4MHFWQAHZJKLGHZJHCQ3GROHVFZ5K5GG2FQIAI7UHTO6PYNAU2
SE6CYHVVUGDS43ZPD2QON5IX52F545Q3W52JY5LFD43TP5K2WNYDFCAT72T";, \n
"supportedInterfaces";: {} \n
"apiEndpoint";: "https://api.eu.amazonalexa.com";, \n
"accessToken";:
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJhdWQiOiJodHRwczovL2FwaS5ld
S5hbWF6b25hbGV4YS5jb20iLCJpc3MiOiJBbGV4YVNaWxsS2I0Iiwic3ViIjojYW16bWJlbnRwXwLjR
lM2NkNDgzLWE2Y2MtNDQ5NC1mTkVLEtYzIiLCJmZG90b3R5b25kaWZk2MSIsImV4cCI6MTcyODc4NjMwNiwiawF0IjoXNzI4N
zg2MjQ2LjUyYmYiOiJlM2Mjg3ODYyNDYsInByaXZhdGVDbG9pbXMiOnsiaXNGB3JSQSI6ImZhbnN1Iiwibm9uThd
hU2NvcGVzIjojYWxleGE60nBlcnNvb19pZDpyZWFKIiwiaWF0Ij09dGV4dCI6IjE6bG9uZDpyZWFKIiwiaWF0Ij09dGV4dGVzZGZG

```

y9ZWVE0a3JUMW11ZUFFQUFBQUFBQUFEWTJKY0t6NWE3Wmg0V1RSZEpWdVVvWW1Gd25RaHFKbWVJakpnRnNDOUk  
1Qk9oamVrcU1BM0tGZ0JES3BFV2FqZmhJNkczTnVMRjhMSU16T0x0aE43eV1vVFRtbWxRNHgzUXh1cFFhVHpaZ  
2dOWG1oa0Y1a0xXamNSV1d0azJYcXQ4MGNnN1ZPdTJBZ01aRFBTVFg3Q3p1QnFpd1FVNjkzL1ZHa0t0TW41VKE  
1WG1pM2g1NGgwUnp5UWJSckUyM1BqaTAvCjJnTzJZ0EgrZmRSS1Vwcm1VQ1BxTud2Y29md3ExVG14Um5VTGxEQ  
zdLN2hHUXFoeGxUZWV2UV1SbGJHdDhPV1o2eW5mL1FoRGErSENXMGpKVmhRQmNuUTNTQ3UwS1d5MWpOLzJDQU5  
1S3hyL0o0ZitmdmFsSERSNGxtM1djSnA3RG1zU0JNdZ11UV1rSFFRK2VWdG9Yb1hSQ1RBWF1QSkVKSGUySmtsU  
Vo2MmQ2SDhEUnR5VTd0eDg1WXkOSjVLTUJOMOVGTOfXbTBXckhoT1dRYOpvbUwxSGUzd0hQaVAxOG0b0ZtWfD  
QM2hDe1JHMz1NSjVvKzYvMk5ncjNrU3RWZUh0dGtIK0ZkZmprWnZSZGxTZTB1SG9CMjlaelFHS3cyU31wNgluc  
3FpUkp2SUoiLCJkZXZpY2VJZCI6ImFtem4xLmFzay5kZXZpY2UuQU1BWFJXWVVRjNTRFgOS0ZPSzRKUjNKMkx  
CRjRSNuk1Q003T0xUVVFGVzJKWetYRDZSV0VHT01XSTZSRzVVWE1JVFNNuzRVVTZKV1RBMk9ZV1M3UUFDN1RHV  
DVPVU1ITEk0TUhGV1FBSFpKS0xHSFpKSENRM0dST0hWR1o1SzVHRzJGUU1BSTdVSEJUTzZQWU5BVTJTRTZDWU  
WVvdHRFMOM1pQRDJRT041SVg1MkY1NDVRM1c1MkpZNUxGRDQzVFA1SzJXT11ERkNBVDcyVCIsInVzZXJJZCI6I  
mFtem4xLmFzay5hY2NvdW50LkFNQTVKNU1QQU5HWEVXQ1BXM1NIQUxVMk5CNERIWFUzQkpBWU1RUUG1VU1RQVh  
QT11VQUNPSUVWRFRPRzIyQ0VOM1NDRjdLR0Y1UkpTVUNUN1pCTEs0M1RUM0RIWkZOV1VPNOtXTjJFUjI2WjIzN  
ERNWERHU1FTNEIOQTVYTVMNkZTV01CNOQzQkZYUzdJSTVFSTJLT01aREk1R1dRU1IyWU1UNFZRSFc2RVJRVEF  
XTzJLM1VOUzJPSkMzTTJITjRFSUdQU0wyUDY3WUNNT0wOU1pCVURENFJKRFpZSDVZM1RMNEdJN00yN1dRI19.  
Isu8w0KlBH-  
SbQyldifxTrhqeARCJ2glwOxy1xHUX8azntVHZBUHNTsRxFJaoBb01baf6FVB7i3B1cta6bXb0qT0Tqz-  
5NPnRM\_bXeK30IQNct4jYo3vyP7GzsLzzRjnd17iZuya173Z24Q6vqyGfI9M--v4wvcgi\_ijG15q3EhLR4-  
M\_53thc4oDg3yn0qQhnCy0U3kg\_N1EvJTtoqGQmuCGEorJXJeAdnKtVkVFfrQpwskcXwhGH8G1eynCIRgrCfVH  
qBYM8wDMSWysCDuKWpuG74Ustgeqvxnx9fIQb4-SV3P8NTdcEOwwE3GGFT-  
kA7B7NWhAg3122Ebodpd1w&quot;;\n } \n },  
 \n &quot;;request&quot;;: { \n  
&quot;;type&quot;;: &quot;;IntentRequest&quot;;, \n  
&quot;;requestId&quot;;: &quot;;amzn1.echo-api.request.f2eela71-80cc-4266-9302-  
81f668b68478&quot;;, \n &quot;;locale&quot;;: &quot;;en-  
US&quot;;, \n &quot;;timestamp&quot;;: &quot;;2024-10-  
13T02:24:06Z&quot;;, \n &quot;;intent&quot;;: { \n  
&quot;;name&quot;;: &quot;;GetAIResponseIntent&quot;;, \n  
&quot;;confirmationStatus&quot;;: &quot;;NONE&quot;;, \n  
&quot;;slots&quot;;: { \n &quot;;query&quot;;: { \n  
&quot;;name&quot;;: &quot;;query&quot;;, \n  
&quot;;value&quot;;: &quot;;the relationship between hearts and bonds&quot;;, \n  
&quot;;confirmationStatus&quot;;: &quot;;NONE&quot;;, \n  
&quot;;source&quot;;: &quot;;USER&quot;;, \n  
&quot;;slotValue&quot;;: { \n &quot;;type&quot;;:  
&quot;;Simple&quot;;, \n &quot;;value&quot;;:  
&quot;;the relationship between hearts and  
bonds&quot;; \n } \n  
 } \n }, \n  
&quot;;dialogState&quot;;:  
&quot;;STARTED&quot;; \n } \n } \n  
 }, \n &quot;;metrics&quot;;: { \n  
&quot;;skillExecutionTimeInMilliseconds&quot;;:  
264 \n } \n } \n } \n } \n } ] } , "status":  
"finished\_successfully", "end\_turn": null, "weight": 1.0, "metadata":  
{ "serialization\_metadata": { "custom\_symbol\_offsets": [] }, "request\_id":  
"8d1be3c439e25255-MXP", "message\_source": null, "timestamp\_": "absolute",  
"message\_type": null }, "recipient": "all", "channel": null }, "parent": "76c4f215-e092-

```

432b-9c98-blaadcf19ce7", "children": [{"b51c2bef-456f-4e4f-a7f9-d4d22f4333d3"}],
"b51c2bef-456f-4e4f-a7f9-d4d22f4333d3": {"id": "b51c2bef-456f-4e4f-a7f9-d4d22f4333d3",
"message": {"id": "b51c2bef-456f-4e4f-a7f9-d4d22f4333d3", "author": {"role":
"assistant", "name": null, "metadata": {}}, "create_time": 1728786476.368195,
"update_time": null, "content": {"content_type": "text", "parts": ["It looks like your
skill execution returned an exception with the error code `SKILL_ENDPOINT_ERROR`. This
usually indicates that the Lambda function used by Alexa is not returning a valid
response, or there was an issue in handling the request within the defined time limit.
Here's what we can do to address this issue:\n\n### Key Issues:\n1. **Timeout or
Long Execution**: Alexa may be timing out because of a long OpenAI API request.\n2.
Lambda Error: If the Lambda function encounters an error, it might not return the
correct response.\n\n### Solution: Breaking Down the Problem\n\n1. **Return an
Immediate Response from Alexa**: Alexa needs to respond quickly to user requests. We
will return an initial 'processing' message to the user while handling the
longer API call in the background.\n\n2. **Using Async Await Properly**: Ensure that
the OpenAI API call is handled asynchronously and properly awaited.\n\n3. **Error
Handling**: Add better error handling to prevent any unhandled exceptions from
breaking the skill.\n\n### Steps to Address the Timeout\n\n1. **Initial Response**:
Return a quick acknowledgment that the skill is working on the request, so Alexa
doesn't timeout while waiting for a longer response.\n\n2. **Async Processing**:
You can process the OpenAI request in the background, but you'll need to inform
the user that it's 'thinking' and manage the skill's conversation
appropriately.\n\n### Updated Code with Proper Async Handling and Initial
Response\n\n```javascript\nconst GetAIResponseIntentHandler = {\n
canHandle(handlerInput) {\n return
Alexa.getRequestType(handlerInput.requestEnvelope) === IntentRequest;\n
&& Alexa.getIntentName(handlerInput.requestEnvelope) ===
IntentName.GET_AI_RESPONSE;\n },\n async handle(handlerInput) {\n
const query = handlerInput.requestEnvelope.request.intent.slots.query.value;\n\n
// Provide a quick response first to acknowledge the request\n const
speakOutput = 'Let me think about that for a moment.';\n const
responseBuilder =
handlerInput.responseBuilder\n .speak(speakOutput)\n .getResponse
e();\n\n // Perform async operation: OpenAI API call\n try {\n
const aiResponse = await getAIResponse(query);\n\n // Log the AI response
for debugging\n console.log('AI Response: ', aiResponse);\n\n
// Now return the final response to the user, either via APL or speech\n
return handlerInput.responseBuilder\n .speak('Here is what I found:
${aiResponse}')\n .getResponse();\n } catch (error) {\n
console.error('Error fetching AI response: ', error);\n\n // In
case of error, respond with a generic error message\n return
handlerInput.responseBuilder\n .speak('Sorry, I encountered an
error processing your request.')\n .getResponse();\n }\n\n
// Return the initial 'thinking' response while async processing continues\n
return responseBuilder;\n }\n};\n\n// Example of OpenAI API call function\nasync
function getAIResponse(query) {\n const apiKey = process.env.OPENAI_API_KEY;\n
const url = 'https://api.openai.com/v1/completions';\n const body =
JSON.stringify({\n model: 'text-davinci-003',\n prompt:
query,\n max_tokens: 100,\n temperature: 0.7\n });\n const

```

```

headers = {\n
 "Content-Type": "application/json"\n
 }\n\n
 return new
 Promise((resolve, reject) => {\n
 const req = https.request(url, { method:
 'POST', headers }, (res) => {\n
 let data = ''\n
 res.on('data', (chunk) => {\n
 data +=
 chunk;\n
 });\n
 res.on('end', () => {\n
 try {\n
 const response = JSON.parse(data);\n
 resolve(response.choices[0].text.trim());\n
 } catch (error) {\n
 reject(error);\n
 }\n
 });\n
 });\n
 req.on('error', reject);\n
 req.write(body);\n
 req.end();\n
 });\n\n
 ``\n\n
 #### Explanation:\n
 1. Initial Response: Alexa will immediately tell the user, "Let me think about that for a moment," allowing us to avoid timeouts while waiting for the OpenAI response.\n
 2. Async OpenAI Request: The OpenAI API call is handled asynchronously. If the response is received quickly enough, Alexa will update the user with the final response. If it takes too long, Alexa will return the initial message.\n
 3. Error Handling: Errors (such as timeouts or parsing errors) are caught and logged, and the user is informed of any issues with a fallback error message.\n
 #### Notes:\n
 - Timeout Considerations: Alexa skills typically have a 8-second timeout for responses. If the OpenAI response consistently takes longer than this, you may want to handle that by telling the user to try again later or using an external system (e.g., sending the result to their Alexa app).\n
 - Progress Updates: Alexa can't handle asynchronous background processes effectively, so it's best to ensure your API calls are fast or provide progress updates to simulate the delay.\n
 By handling this flow asynchronously, you'll reduce timeouts and provide better error messaging when things go wrong."], "status": "finished_successfully", "end_turn": true, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id": null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o", "parent_id": "aaa219a6-5c45-4004-8044-609b9dd94e3d", "request_id": "8d1be3c439e25255-MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens": [200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent": "aaa219a6-5c45-4004-8044-609b9dd94e3d", "children": [{"id": "aaa2ac7a-4896-4d28-aad0-c37c4793b4f5"}], "aaa2ac7a-4896-4d28-aad0-c37c4793b4f5": {"id": "aaa2ac7a-4896-4d28-aad0-c37c4793b4f5", "message": {"id": "aaa2ac7a-4896-4d28-aad0-c37c4793b4f5", "author": {"role": "user", "name": null, "metadata": {}}, "create_time": 1728824199.684046, "update_time": null, "content": {"content_type": "text", "parts": ["give me a JSON TO SEND USING THE MANUAL JSON TESTING ENVIRONMENT"]}, "status": "finished_successfully", "end_turn": null, "weight": 1.0, "metadata": {"serialization_metadata": {"custom_symbol_offsets": []}, "request_id": "8d1f7dad9cd95234-MXP", "message_source": null, "timestamp_": "absolute", "message_type": null}, "recipient": "all", "channel": null}, "parent": "b51c2bef-456f-4e4f-a7f9-d4d22f4333d3", "children": [{"id": "1e5c6db6-1d53-4b17-92d7-42ab34655a47"}], "1e5c6db6-1d53-4b17-92d7-42ab34655a47": {"id": "1e5c6db6-1d53-4b17-92d7-42ab34655a47", "message": {"id": "1e5c6db6-1d53-4b17-92d7-42ab34655a47", "author": {"role": "assistant", "name": null, "metadata": {}}, "create_time": 1728824225.167214, "update_time": null, "content": {"content_type": "text", "parts": ["Sure! Below is a sample JSON that you can use to manually test the Alexa skill using the Amazon Developer Console's Manual JSON Testing Environment. This example simulates an `IntentRequest` where the user asks about the relationship between cops and donuts (for the `GetAIResponseIntent` intent).\n\n#### Sample JSON for Manual

```

[illegible]

[illegible]



```

"author": {"role": "assistant", "name": null, "metadata": {}}, "create_time":
1728824319.636195, "update_time": null, "content": {"content_type": "text", "parts":
["From the response you received:\n\n```\njson\n{\n "ssml":\n "<speak>Sorry, I encountered an error processing your\nrequest.</speak>"\n}\n\n```\n\nThis means the skill executed successfully,\nbut there was an issue or error when processing the request.\n\nHere\u2019s what this\nindicates:\n- Your Alexa skill is correctly receiving the `IntentRequest`.\n- The\nskill invocation works, but during the execution of the `GetAIResponseIntent` or when\ntrying to communicate with OpenAI, an error occurred.\n\n### Possible reasons for the\nerror:\n1. Timeout Issue: The Lambda function might be timing out, especially if\nthe process of fetching the AI response is taking too long.\n2. OpenAI API Error:\nThere might be an issue with how the request to OpenAI is being made or how the\nresponse is being processed. Ensure that the API request to OpenAI is well-formed and\nproperly handled.\n3. Unhandled Exceptions: The skill may be encountering an\nunexpected error, which wasn\u2019t caught, and is leading to a generic error\nmessage.\n\n### Steps to debug:\n1. Check CloudWatch Logs: Since this is a Lambda\nfunction, head over to AWS CloudWatch Logs to check for detailed error logs from your\nfunction. It will provide insights into the exact error.\n2. Increase Timeout: If\nyour Lambda function is waiting on a response from OpenAI, make sure the\nfunction\u2032s timeout is sufficient for longer-running tasks.\n3. Handle\nExceptions Properly: Ensure your code catches any exceptions, particularly around\nthe API call to OpenAI, and provides a meaningful error response to the user.\n4.\nTest OpenAI Integration Independently: Manually test the OpenAI API request\noutside of the Alexa skill (e.g., using Postman or a similar tool) to ensure that the\nrequest is correct and the AI model is responding.\n\nWould you like guidance on how\nto troubleshoot the API call or error handling in your Lambda function?"]}, "status":
"finished_successfully", "end_turn": true
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa24dbb-b8ed-47d1-b4b4-ad4e73d1f509", "request_id": "8d1f804be8a45234-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa24dbb-b8ed-47d1-b4b4-ad4e73d1f509", "children": ["aaa2c7ba-98af-4974-8cbd-
6b7d335ea01a"]}, "aaa2c7ba-98af-4974-8cbd-6b7d335ea01a": {"id": "aaa2c7ba-98af-4974-
8cbd-6b7d335ea01a", "message": {"id": "aaa2c7ba-98af-4974-8cbd-6b7d335ea01a", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728824399.940092,
"update_time": null, "content": {"content_type": "text", "parts": ["i\u2032m HAVING
BIG TROUBLE FINDING THOSE CLOUD WATCH LOGS"]}, "status": "finished_successfully",
"end_turn": null, "weight": 1.0, "metadata": {"serialization_metadata":
{"custom_symbol_offsets": []}, "request_id": "8d1f828e5e3f5234-MXP", "message_source":
null, "timestamp_": "absolute", "message_type": null}, "recipient": "all", "channel":
null}, "parent": "642345dc-f6c5-47b4-82f5-0338cfb8f4b2", "children": ["1dc54ee3-85a4-
4a09-a9f5-57b9f78d0c4d"]}, "1dc54ee3-85a4-4a09-a9f5-57b9f78d0c4d": {"id": "1dc54ee3-
85a4-4a09-a9f5-57b9f78d0c4d", "message": {"id": "1dc54ee3-85a4-4a09-a9f5-57b9f78d0c4d",
"author": {"role": "assistant", "name": null, "metadata": {}}, "create_time":
1728824426.278226, "update_time": null, "content": {"content_type": "text", "parts":
["Finding CloudWatch logs for your Alexa skill can be a bit tricky if you\u2032re new
to AWS. Here\u2032s a step-by-step guide to help you locate the logs for your Lambda
function:\n\n### Step 1: Ensure CloudWatch Logs are Enabled\n\nLambda functions

```

automatically generate CloudWatch logs, but you need to ensure that the right permissions are in place. Check if your Lambda function has the necessary permissions by following these steps:

- Go to AWS Lambda Console:** Visit [AWS Lambda Console] (<https://console.aws.amazon.com/lambda/home>). Select your function from the list (this will be the function linked to your Alexa skill).
- Check IAM Role Permissions:** Go to the **Configuration** tab for your Lambda function. Under **Execution role**, click the role name to open the IAM role in a new tab. Check if the role has the **AWSLambdaBasicExecutionRole** policy attached. This policy grants permission to write logs to CloudWatch. If the policy isn't attached, you can add it: Click **Add permissions** > **Attach policies**. Search for **AWSLambdaBasicExecutionRole** and attach it.

**Step 2: Go to CloudWatch Logs** Once you're sure that logging is enabled, follow these steps to access the logs:

- Go to CloudWatch Console:** Visit [AWS CloudWatch Console] (<https://console.aws.amazon.com/cloudwatch/home>).
- Access Log Groups:** In the left-hand menu, click on **Logs** > **Log groups**. Look for a log group with the name format `/aws/lambda/<lambda-function-name>`. This log group contains all logs generated by your Lambda function. If you're not sure about the function name, it should be the same as the Lambda function linked to your Alexa skill.
- View Log Streams:** Click on the log group for your Lambda function. You'll see a list of log streams. These are individual logs for each invocation of your Lambda function. The most recent log streams are at the top. Click on a log stream to view the details of each execution, including error messages, console logs, and stack traces.

**Step 3: Analyze the Logs** Once you have opened a log stream:

- Look for any lines marked as **ERROR** or containing stack traces. You'll also see the input and output of the Lambda function calls, which can help you understand where things might be going wrong.
- If OpenAI API errors are occurring, they should be logged here.

**Additional Tips:**

- Use Filters:** You can filter your logs to look for specific errors. Use the search box in the log stream view to look for terms like `ERROR`, `RequestId`, or other relevant keywords.
- Increase Logging:** If necessary, you can add more detailed logging in your Lambda function using `console.log()` to trace the execution path.

Would you like me to help with writing additional logging code in your Lambda function to better understand where the error is occurring?"]},

```

"status": "finished_successfully", "end_turn": true
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "aaa2c7ba-98af-4974-8cbd-6b7d335ea01a", "request_id": "8d1f828e5e3f5234-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"aaa2c7ba-98af-4974-8cbd-6b7d335ea01a", "children": ["aaa2fbf0-bbc5-4436-bc15-
16e4f87c3731"]}, "aaa2fbf0-bbc5-4436-bc15-16e4f87c3731": {"id": "aaa2fbf0-bbc5-4436-
bc15-16e4f87c3731", "message": {"id": "aaa2fbf0-bbc5-4436-bc15-16e4f87c3731", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728826155.746888,
"update_time": null, "content": {"content_type": "text", "parts": ["this tell you
anything important: \n\n\n "header"; {\n "namespace";
"Alexa.Presentation.APL";,\n "name";:
"RenderDocument";,\n "messageId";: "9b9d0d67-9a67-4329-
987a-5bfb915ef938";,\n "keys";: {\n "channel";:
"visual";,\n "isBlocking";: true,\n

```

```
"navigationType";: "NONE";\n },\n "dialogRequestId";: "1fe3c917-6284-4464-b9ba-f0dae3b268e5";\n },\n "payload";: {\n "presentationToken";: "amzn1.as-tt.v1.DeeAppTrafficSpeechletService#TID#DeeAppTrafficSpeechletService";,\n "timeoutType";: "SHORT";,\n "document";: {\n "type";: "APL";,\n "version";: "1.9";,\n "background";: "#2D526D";,\n "settings";: {\n "supportsResizing";: true\n },\n "theme";: "dark";,\n "import";: [\n {\n "name";: "alex-layouts";,\n "version";: "1.8.0-eifjcbfcricnvrkrhlgjgenukunfcjkifgllhlecev";\n },\n {\n "resources";: [\n {\n "dimensions";: {\n "fullWidth";: "100vw";,\n "fullHeight";: "100vh";,\n "summaryPageBodyPaddingLeft";: "@marginHorizontal";,\n "summaryPageBodyPaddingRight";: "@marginHorizontal";,\n "alternativeRoutesPageBodyPaddingLeft";: "@marginHorizontal";,\n "alternativeRoutesPageBodyPaddingRight";: "@marginHorizontal";,\n "iconDefaultWidth";: "32dp";,\n "iconDefaultHeight";: "32dp";,\n "numbers";: {\n "autoPageDelayInMs";: 10000,\n "summaryPageSegmentDetailsMaxLines";: 1,\n "travelDistanceOpacity";: 0.6,\n "travelTimePaddingLeft";: 0,\n "travelTimePartPaddingTop";: 0,\n "travelTimePartUnitPaddingTop";: 0,\n "travelTimePartUnitPaddingBottom";: 10,\n "travelTimePartStatusIconPaddingBottom";: 25,\n "travelDistancePartUnitPaddingBottom";: 0,\n "alternativeRouteSegmentDetailsMaxLines";: 1,\n "alternativeRouteTravelTimePartUnitPaddingBottom";: 7,\n "alternativeRouteTravelTimePartStatusIconPaddingBottom";: 15,\n "alternativeRouteTravelDistanceOpacity";: 0.6\n },\n "strings";: {\n "summaryPageBodyJustifyContent";: "end";,\n "summaryPageBodyTextAlign";: "left";,\n "travelTimePartValueFontSize";: "@fontSize5XLarge";,\n "travelTimePartValueFontWeight";: "@fontWeightBold";,\n "travelTimePartUnitFontSize";: "@fontSize3XLarge";,\n "travelTimePartUnitFontWeight";: "@fontWeightLight";,\n "travelTimePartUnitPaddingLeft";: "@spacing2XSmall";,\n "travelTimePartStatusIconPaddingLeft";: "@spacingXSmall";,\n "travelTimeBasePaddingRight";: "@spacingSmall";,\n "travelTimeAlignItems";: "end";,\n "travelTimeTextAlign";: "bottom";,\n "travelTimeJustifyContent";: "left";
```

```

"summaryPageSegmentDetailsFontSize";: "@fontSizeXLarge";, \n
"summaryPageSegmentDetailsFontWeight";: "@fontWeightLight";, \n
"travelDistancePartValueFontSize";: "@fontSizeXLarge";, \n
"travelDistancePartValueFontWeight";: "@fontWeightRegular";, \n
"travelDistancePartValueTextAlignVertical";: "bottom";, \n
"travelDistancePartUnitFontSize";: "@fontSizeXLarge";, \n
"travelDistancePartUnitFontWeight";: "@fontWeightRegular";, \n
"travelDistancePartUnitTextAlignVertical";: "bottom";, \n
"travelDistanceJustifyContent";: "left";, \n
"alternativeRoutesListDividerPaddintTop";: "@spacingMedium";, \n
"alternativeRoutesListDividerPaddintBottom";: "@spacingMedium";, \n
"alternativeRouteTravelTimePartValueFontSize";:
"@fontSize3XLarge";, \n
"alternativeRouteTravelTimePartValueFontWeight";:
"@fontWeightBold";, \n
"alternativeRouteTravelTimePartUnitFontSize";:
"@fontSize2XLarge";, \n
"alternativeRouteTravelTimePartUnitFontWeight";:
"@fontWeightLight";, \n
"alternativeRouteTravelTimePartUnitPaddingLeft";:
"@spacing3XSmall";, \n
"alternativeRouteTravelTimePartStatusIconPaddingLeft";:
"@spacing2XSmall";, \n
"alternativeRouteTravelTimeBasePaddingRight";: "@spacingXSmall";, \n
"alternativeRouteTravelTimeTextAlignVertical";: "bottom";, \n
"alternativeRouteTravelTimeJustifyContent";: "left";, \n
"alternativeRoutePageBodyJustifyContent";: "end";, \n
"alternativeRoutePageBodyTextAlign";: "left";, \n
"alternativeRouteSegmentDetailsFontSize";: "@fontSizeMedium";, \n
"alternativeRouteSegmentDetailsFontWeight";: "@fontWeightLight";, \n
"alternativeRouteTravelDistancePartValueFontSize";:
"@fontSizeXLarge";, \n
"alternativeRouteTravelDistancePartValueFontWeight";:
"@fontWeightRegular";, \n
"alternativeRouteTravelDistancePartValueTextAlignVertical";:
"bottom";, \n
"alternativeRouteTravelDistancePartUnitFontSize";:
"@fontSizeXLarge";, \n
"alternativeRouteTravelDistancePartUnitFontWeight";:
"@fontWeightRegular";, \n
"alternativeRouteTravelDistancePartUnitTextAlignVertical";:
"bottom";, \n
"alternativeRouteTravelDistanceJustifyContent";:
"left";\n
 }\n
 }, \n
 {\n
"when";: "${@viewportProfile == @hubRoundSmall}";, \n
"numbers";: {\n
"summaryPageSegmentDetailsMaxLines";: 3, \n
"travelTimePartUnitPaddingBottom";: 5, \n
"travelTimePartStatusIconPaddingBottom";: 20, \n

```

```

 "alternativeRouteSegmentDetailsMaxLines";: 3,\n
 "alternativeRouteTravelTimePartUnitPaddingBottom";: 5,\n
 "alternativeRouteTravelTimePartStatusIconPaddingBottom";:
20\n },\n "strings";: {\n
 "travelTimePartUnitPaddingLeft";: "@spacing3XSmall";,\n
 "travelTimePartStatusIconPaddingLeft";: "@spacing2XSmall";,\n
 "travelTimeBasePaddingRight";: "0dp";,\n
 "travelTimeJustifyContent";: "center";,\n
 "summaryPageBodyJustifyContent";: "start";,\n
 "summaryPageBodyTextAlign";: "center";,\n
 "travelTimePartValueFontSize";: "@fontSize4XLarge";,\n
 "travelDistancePartValueFontSize";: "@fontSizeMedium";,\n
 "travelDistancePartUnitFontSize";: "@fontSizeMedium";,\n
 "travelDistancePartValueFontWeight";: "@fontWeightMedium";,\n
 "travelDistanceJustifyContent";: "center";,\n
 "summaryPageSegmentDetailsFontSize";: "@fontSizeMedium";,\n
 "summaryPageSegmentDetailsFontWeight";: "@fontWeightRegular";,\n
 "alternativeRouteTravelTimePartValueFontSize";:
 "@fontSize4XLarge";,\n
 "alternativeRouteTravelTimePartValueFontWeight";:
 "@fontWeightMedium";,\n
 "alternativeRouteTravelTimePartUnitFontSize";:
 "@fontSize3XLarge";,\n
 "alternativeRouteTravelTimePartStatusIconPaddingLeft";: "0dp";,\n
 "alternativeRouteTravelTimeBasePaddingRight";: "0dp";,\n
 "alternativeRouteTravelTimeJustifyContent";: "center";,\n
 "alternativeRoutePageBodyTextAlign";: "center";,\n
 "alternativeRouteTravelDistancePartValueFontSize";:
 "@fontSizeMedium";,\n
 "alternativeRouteTravelDistancePartUnitFontSize";:
 "@fontSizeMedium";,\n
 "alternativeRouteTravelDistanceJustifyContent";:
 "center";\n }\n },\n {\n
 "when";: "${@viewportProfile == @hubLandscapeXSmall}";,\n
 "strings";: {\n "travelTimePartValueStyle";:
 "textStyleDisplay4";,\n
 "travelTimePartUnitStyle";: "textStyleDisplay4";,\n
 "summaryPageBodyJustifyContent";: "center";,\n
 "travelTimeBasePaddingRight";: "5dp";,\n
 "travelTimeAlignItems";: "center";,\n
 "travelTimeTextAlign";: "center";,\n
 "travelTimeJustifyContent";: "center";,\n
 "travelDistanceJustifyContent";: "center";,\n
 "travelDistancePartValueStyle";: "textStyleBody";,\n
 "travelDistancePartUnitStyle";:
 "textStyleBody";\n }\n },\n
 {\n "when";: "${@viewportProfile ==
 @hubLandscapeSmall}";,\n "strings";: {\n
 "travelDistancePartValueFontSize";: "@fontSizeMedium";,\n

```

```

"travelDistancePartUnitFontSize";: "@fontSizeMedium";,\n
"summaryPageSegmentDetailsFontSize";: "@fontSizeMedium";,\n
"alternativeRouteTravelDistancePartValueFontSize";:
"@fontSizeMedium";,\n
"alternativeRouteTravelDistancePartUnitFontSize";:
"@fontSizeMedium";\n
 }\n
 "when";: "${@viewportProfile ==
@hubLandscapeSmall || @viewportProfile == @hubLandscapeMedium}";,\n
"strings";: {\n
"alternativeRoutesListDividerPaddintTop";: "@spacingSmall";,\n
"alternativeRoutesListDividerPaddintBottom";:
"@spacingSmall";\n
 },\n
"numbers";: {\n
"alternativeRouteTravelTimePartUnitPaddingBottom";:
3\n
 }\n
 },\n
 {\n
"when";: "${@viewportProfile == @hubLandscapeLarge || @viewportProfile
== @hubLandscapeXLarge}";,\n
 "strings";: {\n
"alternativeRouteSegmentDetailsFontSize";:
"@fontSizeLarge";\n
 },\n
"numbers";: {\n
"travelTimePartStatusIconPaddingBottom";: 35,\n
"alternativeRouteTravelTimePartStatusIconPaddingBottom";:
25\n
 }\n
 }\n
],\n
"commands";: {\n
 "OnMountAutoPage";: {\n
"commands";: [\n
 {\n
"type";: "AutoPage";,\n
 "delay";:
"@autoPageDelayInMs";\n
 }\n
 },\n
 "layouts";: {\n
"SummaryPage";: {\n
 "parameters";: [\n
"data";\n
],\n
 "items";: [\n
 {\n
 "type";: "Container";,\n
"direction";: "column";,\n
"items";: [\n
 {\n
"when";: "${@viewportProfile != @hubLandscapeXSmall}";,\n
"type";: "AlexaBackground";,\n
"backgroundImageSource";: "${data.backgroundImage}";,\n
"backgroundAlign";: "right";,\n
"backgroundScale";: "best-
fit";\n
 },\n
 {\n
"when";: "${@viewportProfile == @hubLandscapeXSmall}";,\n
"type";: "AlexaBackground";,\n
"backgroundColor";: "black";,\n
"backgroundAlign";: "right";,\n
"backgroundScale";: "best-
fit";\n
 },\n
 {\n
"when";: "${@viewportProfile != @hubLandscapeXSmall}";,\n
"type";: "AlexaHeader";,\n
"headerTitle";: "${data.trafficSummary.trafficSummaryTitle}";,\n
"headerSubtitle";:

```

```

 "${data.trafficSummary.trafficSummarySubtitle}";\n
 },\n
 {\n
 "when";: "${@viewportProfile != @hubLandscapeXSmall}";,\n
 "type";: "SummaryPageBody";,\n
 "data";: "${data}";,\n
 "paddingLeft";: "@summaryPageBodyPaddingLeft";,\n
 "paddingRight";: "@summaryPageBodyPaddingRight";,\n
 "grow";: 1\n
 },\n
 {\n
 "when";: "${@viewportProfile ==\n
 @hubLandscapeXSmall}";,\n
 "type";:\n
 "BaklavaPageBody";,\n
 "data";:\n
 "${data}";,\n
 "grow";:\n
 1\n
 },\n
 {\n
 "when";: "${@viewportProfile != @hubRoundSmall &&\n
 @viewportProfile != @hubLandscapeXSmall}";,\n
 "type";: "AlexaFooter";,\n
 "hintText";:\n
 "${data.hintText}";\n
 }\n
]\n
 }\n
]\n
 },\n
 "BaklavaPageBody";: {\n
 "parameters";: [\n
 "data";\n
],\n
 "items";: [\n
 {\n
 "alignItems";: "center";,\n
 "type";: "Container";,\n
 "justifyContent";: "@summaryPageBodyJustifyContent";,\n
 "items";: [\n
 {\n
 "type";: "TravelTime";,\n
 "travelTime";: "${data.fastestRoute.travelTime}";,\n
 "trafficStatusColor";:\n
 "${data.fastestRoute.trafficStatusColor}";\n
 },\n
 {\n
 "type";: "TravelDistance";,\n
 "travelDistance";:\n
 "${data.fastestRoute.travelDistance}";\n
 }\n
]\n
 }\n
]\n
 },\n
 "SummaryPageBody";: {\n
 "parameters";: [\n
 "data";\n
],\n
 "items";: [\n
 {\n
 "type";: "Container";,\n
 "justifyContent";: "@summaryPageBodyJustifyContent";,\n
 "items";: [\n
 {\n
 "type";: "TravelTime";,\n
 "travelTime";: "${data.fastestRoute.travelTime}";,\n
 "statusIcon";:\n
 "${data.fastestRoute.statusIcon}";\n
 },\n
 {\n
 "type";:\n
 "SummaryPageRoute";,\n
 "routeDescriptionText";:\n
 "${data.fastestRoute.routeDescriptionText}";\n
 },\n
 {\n

```

```

 "type";: "TravelDistance";, \n
 "travelDistance";:
 "${data.fastestRoute.travelDistance}";\n
]\n
 }, \n
 "SummaryPageRoute";: {\n
"parameters";: [\n
"routeDescriptionText";\n
"items";: [\n
"maxLines";: "@summaryPageSegmentDetailsMaxLines";, \n
"fontSize";: "@summaryPageSegmentDetailsFontSize";, \n
"fontWeight";: "@summaryPageSegmentDetailsFontWeight";, \n
"text";: "${routeDescriptionText}";, \n
"textAlign";: "@summaryPageBodyTextAlign";, \n
"type";:
"Text";\n
], \n
 \n
 "TravelTime";: {\n
"parameters";: [\n
"statusIcon";, \n
"trafficStatusColor";\n
"items";: [\n
"direction";: "row";, \n
], \n
 "when";: "${travelTime.base.value != 0}";, \n
"type";: "TravelTimePart";, \n
"value";: "${travelTime.base.value}";, \n
"unit";: "${travelTime.base.unit}";, \n
"color";: "${trafficStatusColor}";, \n
"paddingRight";:
"@travelTimeBasePaddingRight";\n
 {\n
 "when";:
"${travelTime.offset.unit}";, \n
"type";: "TravelTimePart";, \n
"value";: "${travelTime.offset.value}";, \n
"unit";: "${travelTime.offset.unit}";, \n
"color";:
"${trafficStatusColor}";\n
 }, \n
 {\n
 "when";: "${@viewportProfile !=
@hubLandscapeXSmall}";, \n
 "type";:
"Container";, \n
 "paddingLeft";:
"@travelTimePartStatusIconPaddingLeft";, \n
"paddingBottom";: "@travelTimePartStatusIconPaddingBottom";, \n
"items";: [\n
"type";: "AlexaImage";, \n
"imageAspectRatio";: "round";, \n
"imageWidth";: "@iconDefaultWidth";, \n
"imageHeight";: "@iconDefaultHeight";, \n
"imageSource";: "${statusIcon}";, \n
"imageShadow";:
false\n
]

```



```

\n
 }\n
],\n
"alignItems";: "@travelTimeAlignItems";,\n
"justifyContent";: "@travelTimeJustifyContent";,\n
"shrink";: 1,\n
 "type";:\n
"Container";,\n
 "paddingLeft";:\n
"@travelTimePaddingLeft";\n
 }\n
]\n
 },\n
 "TravelTimePart";: {\n
"parameters";: [\n
 "value";,\n
"unit";,\n
 "color";,\n
],\n
"items";: [\n
 {\n
"type";: "Container";,\n
"direction";: "row";,\n
"paddingTop";: "@travelTimePartPaddingTop";,\n
"shrink";: 1,\n
 "items";: [\n
{\n
 "when";: "${@viewportProfile ==\n
@hubLandscapeXSmall}";,\n
 "type";:\n
"Text";,\n
 "text";:\n
"${value}";,\n
 "color";:\n
"${color}";,\n
 "style";:\n
"@travelTimePartValueStyle";\n
 },\n
{\n
 "when";: "${@viewportProfile ==\n
@hubLandscapeXSmall}";,\n
 "type";:\n
"Text";,\n
 "text";:\n
"${unit}";,\n
 "color";:\n
"${color}";,\n
 "style";:\n
"@travelTimePartUnitStyle";\n
 },\n
{\n
 "when";: "${@viewportProfile !=\n
@hubLandscapeXSmall}";,\n
 "type";:\n
"Text";,\n
 "fontSize";:\n
"@travelTimePartValueFontSize";,\n
"fontWeight";: "@travelTimePartValueFontWeight";,\n
"text";: "${value}";,\n
"textAlign";: "@summaryPageBodyTextAlign";,\n
"textAlignVertical";:\n
"@travelTimeTextAlign";\n
 },\n
{\n
 "when";: "${@viewportProfile !=\n
@hubLandscapeXSmall}";,\n
 "type";:\n
"Text";,\n
 "fontSize";:\n
"@travelTimePartUnitFontSize";,\n
"fontWeight";: "@travelTimePartUnitFontWeight";,\n
"text";: "${unit}";,\n
"textAlign";: "@summaryPageBodyTextAlign";,\n
"textAlignVertical";: "@travelTimeTextAlign";,\n
"paddingLeft";: "@travelTimePartUnitPaddingLeft";,\n
"paddingBottom";:\n
"@travelTimePartUnitPaddingBottom";\n
 }\n
]\n
 }\n
],\n
 "TravelDistance";: {\n
"parameters";: [\n

```

```

 "travelDistance";\n
 "items";: [\n
 "type";: "Container";,\n
 "direction";: "row";,\n
 "justifyContent";: "@travelDistanceJustifyContent";,\n
 "items";: [\n
 "type";: "TravelDistancePart";,\n
 "value";: "${travelDistance.distanceText}";,\n
 "unit";:\n
 "${travelDistance.distanceUnitText}";\n
]\n
 },\n
 "TravelDistancePart";: {\n
 "parameters";: [\n
 "unit";\n
 {\n
 "type";: "Container";,\n
 "direction";: "row";,\n
 [\n
 "when";: "${@viewportProfile != @hubLandscapeXSmall}";,\n
 "fontSize";: "@travelDistancePartValueFontSize";,\n
 "fontWeight";: "@travelDistancePartValueFontWeight";,\n
 "text";: "${value}";,\n
 "textAlignVertical";:\n
 "@travelDistancePartValueTextAlignVertical";,\n
 "type";: "Text";,\n
 "opacity";: "@travelDistanceOpacity";,\n
 "paddingRight";:\n
 "@travelDistancePartValuePaddingRight";\n
 {\n
 "when";: "${@viewportProfile !=\n
 @hubLandscapeXSmall}";,\n
 "@travelDistancePartUnitFontSize";,\n
 "fontWeight";: "@travelDistancePartUnitFontWeight";,\n
 "text";: "${unit}";,\n
 "textAlignVertical";: "@travelDistancePartUnitTextAlignVertical";,\n
 "type";: "Text";,\n
 "opacity";: "@travelDistanceOpacity";,\n
 "paddingBottom";:\n
 "@travelDistancePartUnitPaddingBottom";\n
 {\n
 "when";: "${@viewportProfile ==\n
 @hubLandscapeXSmall}";,\n
 "${value}";,\n
 "Text";,\n
 "@travelDistancePartValueStyle";\n
 {\n
 "when";: "${@viewportProfile ==\n
 @hubLandscapeXSmall}";,\n
 "${unit}";,\n
 "Text";,\n
 "@travelDistancePartUnitStyle";\n
],\n
],\n
],\n
 "AlternativeRoutesPage";: {\n

```

```

 "parameters";: [\n
 "data";\n
], \n
 "items";: [\n
 {\n
 "type";: "Container";, \n
 "direction";: "column";, \n
 "items";: [\n
 {\n
 "type";: "AlexaBackground";, \n
 "backgroundImageSource";: "${data.backgroundImage}";, \n
 "backgroundAlign";: "right";, \n
 "backgroundScale";: "best-
 fit";\n
 }, \n
 {\n
 "type";: "AlexaHeader";, \n
 "headerTitle";:
 "${data.alternativeRouteSummary.alternativeRouteTitle}";\n
 }, \n
 {\n
 "type";: "AlternativeRoutesPageBody";, \n
 "alternativeRoutes";:
 "${data.alternativeRouteSummary.alternativeRoutes}";, \n
 "paddingLeft";: "@alternativeRoutesPageBodyPaddingLeft";, \n
 "paddingRight";: "@alternativeRoutesPageBodyPaddingRight";, \n
 "grow";: 1\n
 }, \n
 {\n
 "when";: "${@viewportProfile !=
 @hubRoundSmall}";, \n
 "type";:
 "AlexaFooter";, \n
 "hintText";:
 "${data.hintText}";\n
 }\n
]\n
 }\n
]\n
 }, \n
 "AlternativeRoutesPageBody";: {\n
 "parameters";:
 [\n
 "alternativeRoutes";, \n
 "items";: [\n
 {\n
 "items";: [\n
 {\n
 "type";: "AlternativeRoutesList";, \n
 "alternativeRoutes";:
 "${alternativeRoutes}";, \n
 }\n
], \n
 "type";:
 "Container";\n
 }\n
]\n
 }, \n
 "AlternativeRoutesList";: {\n
 "parameters";: [\n
 "alternativeRoutes";, \n
 "items";: [\n
 {\n
 "when";: "${@viewportProfile == @hubRoundSmall || @viewportProfile ==
 @hubLandscapeSmall}";, \n
 "data";:
 "${alternativeRoutes}";, \n
 "items";: [\n
 {\n
 "type";:
 "AlternativeRoutesListItem";, \n
 "alternativeRoute";: "${data}";, \n
 "idx";:
 "${index}";, \n
 }\n
],
 \n
 "type";: "Sequence";, \n
 "id";: "alternativeRoutesList";, \n
 "grow";: 1\n
 }, \n
 {\n

```

```

 "when";: "${@viewportProfile != @hubRoundSmall &&
 @viewportProfile != @hubLandscapeSmall}";, \n
 "data";: "${alternativeRoutes}";, \n
 "items";: [\n
 "type";: "AlternativeRoutesListItem";, \n
 "alternativeRoute";: "${data}";, \n
 "idx";:
 "${index}";\n
 \n
 "type";: "Container";, \n
 "id";: "alternativeRoutesList";, \n
 "grow";: 1, \n
 "@alternativeRoutePageBodyJustifyContent";:
 "@alternativeRoutePageBodyJustifyContent";\n
]\n
 }, \n
 "AlternativeRoutesListItem";: {\n
 [\n
 "alternativeRoute";, \n
 "idx";\n
], \n
 "items";: [\n
 {\n
 "direction";: "column";, \n
 "items";: [\n
 {\n
 "when";: "${idx != 0}";, \n
 "type";: "AlexaDivider";, \n
 "width";: "100vw";, \n
 "paddingTop";: "@alternativeRoutesListDividerPaddintBottom";, \n
 "paddingBottom";:
 "@alternativeRoutesListDividerPaddintBottom";\n
 }, \n
 {\n
 "type";: "AlternativeRouteTravelTime";, \n
 "travelTime";: "${alternativeRoute.travelTime}";, \n
 "statusIcon";:
 "${alternativeRoute.statusIcon}";\n
 }, \n
 {\n
 "type";:
 "AlternativeRouteSegmentDetails";, \n
 "routeDescriptionText";:
 "${alternativeRoute.routeDescriptionText}";\n
 }, \n
 {\n
 "type";: "AlternativeRouteTravelDistance";, \n
 "travelDistance";:
 "${alternativeRoute.travelDistance}";\n
 }, \n
 {\n
 "type";:
 "Container";\n
 }, \n
 "AlternativeRouteSegmentDetails";: {\n
 "parameters";: [\n
 "routeDescriptionText";\n
], \n
 "items";: [\n
 {\n
 "maxLines";: "@alternativeRouteSegmentDetailsMaxLines";, \n
 "fontSize";: "@alternativeRouteSegmentDetailsFontSize";, \n
 "fontWeight";: "@alternativeRouteSegmentDetailsFontWeight";, \n
 "text";: "${routeDescriptionText}";, \n
 "textAlign";: "@alternativeRoutePageBodyTextAlign";, \n
 "type";:

```

```

"Text";\n }\n]\n },
\n "AlternativeRouteTravelTime";: {\n
"parameters";: [\n "travelTime";, \n
"statusIcon";\n], \n "items";:
[\n {\n "direction";:
"row";, \n "justifyContent";:
"@alternativeRouteTravelTimeJustifyContent";, \n
"items";: [\n {\n
"when";: "${travelTime.base.value != 0}";, \n
"type";: "AlternativeRouteTravelTimePart";, \n
"value";: "${travelTime.base.value}";, \n
"unit";: "${travelTime.base.unit}";, \n
"textAlign";: "@alternativeRoutePageBodyTextAlign";, \n
"paddingRight";:
"@alternativeRouteTravelTimeBasePaddingRight";\n
 }, \n {\n
"when";: "${travelTime.offset.unit}";, \n
"type";: "AlternativeRouteTravelTimePart";, \n
"value";: "${travelTime.offset.value}";, \n
"unit";: "${travelTime.offset.unit}";, \n
"textAlign";:
"@alternativeRoutePageBodyTextAlign";\n }, \n
 {\n "type";: "Container";, \n
"paddingLeft";:
"@alternativeRouteTravelTimePartStatusIconPaddingLeft";, \n
"paddingBottom";:
"@alternativeRouteTravelTimePartStatusIconPaddingBottom";, \n
"items";: [\n {\n
"type";: "AlexaImage";, \n
"imageAspectRatio";: "round";, \n
"imageWidth";: "@iconDefaultWidth";, \n
"imageHeight";: "@iconDefaultHeight";, \n
"imageSource";: "${statusIcon}";, \n
"imageShadow";:
false\n }\n]
\n }\n], \n
"alignItems";: "end";, \n "type";:
"Container";\n }\n]\n
 }, \n "AlternativeRouteTravelTimePart";: {\n
"parameters";: [\n "value";, \n
"unit";\n], \n "items";: [\n
 {\n "type";: "Container";, \n
"direction";: "row";, \n "items";:
[\n {\n
"fontWeight";: "@alternativeRouteTravelTimePartValueFontWeight";, \n
"fontSize";: "@alternativeRouteTravelTimePartValueFontSize";, \n
"text";: "${value}";, \n
"textAlign";: "@alternativeRoutePageBodyTextAlign";, \n
"textAlignVertical";:

```

```

"@alternativeRouteTravelTimeTextAlignVertical";, \n
"type";: "Text";\n }, \n
{\n "fontWeight";:
"@alternativeRouteTravelTimePartUnitFontWeight";, \n
"fontSize";: "@alternativeRouteTravelTimePartUnitFontSize";, \n
"paddingLeft";: "@alternativeRouteTravelTimePartUnitPaddingLeft";, \n
"paddingBottom";:
"@alternativeRouteTravelTimePartUnitPaddingBottom";, \n
"text";: "${unit}";, \n
"textAlign";: "@alternativeRoutePageBodyTextAlign";, \n
"textAlignVertical";:
"@alternativeRouteTravelTimeTextAlignVertical";, \n
"type";:
"Text";\n }\n]\n }, \n
}\n]\n }, \n
"AlternativeRouteTravelDistance";: {\n
"parameters";: [\n
"travelDistance";\n], \n
"items";: [\n {\n
"type";: "Container";, \n
"direction";: "row";, \n
"justifyContent";:
"@alternativeRouteTravelDistanceJustifyContent";, \n
"items";: [\n {\n
"type";: "AlternativeRouteTravelDistancePart";, \n
"value";: "${travelDistance.distanceText}";, \n
"unit";:
"${travelDistance.distanceUnitText}";\n }\n
]\n }\n]\n
}, \n "AlternativeRouteTravelDistancePart";: {\n
"parameters";: [\n "value";, \n
"unit";\n], \n "items";: [\n
{\n "type";: "Container";, \n
"direction";: "row";, \n "items";:
[\n {\n
"fontSize";: "@alternativeRouteTravelDistancePartValueFontSize";, \n
"fontWeight";:
"@alternativeRouteTravelDistancePartValueFontWeight";, \n
"text";: "${value}";, \n
"textAlignVertical";:
"@alternativeRouteTravelDistancePartValueTextAlignVertical";, \n
"type";: "Text";, \n
"opacity";:
"@alternativeRouteTravelDistanceOpacity";\n },
\n {\n
"fontSize";: "@alternativeRouteTravelDistancePartUnitFontSize";, \n
"fontWeight";:
"@alternativeRouteTravelDistancePartUnitFontWeight";, \n
"text";: "${unit}";, \n

```

```

"textAlignVertical";:
"@alternativeRouteTravelDistancePartUnitTextAlignVertical";, \n
"type";: "Text";, \n
"opacity";:
"@alternativeRouteTravelDistanceOpacity";\n
n
] \n
} \n
}, \n
"mainTemplate";: { \n
"parameters";: [\n
"payload"; \n
], \n
"items";: [\n
{ \n
"type";: "Pager";, \n
"id";: "mainPager";, \n
"navigation";:
"normal";, \n
"width";:
"@fullWidth";, \n
"height";:
"@fullHeight";, \n
"onMount";: [\n
{ \n
"type";:
"OnMountAutoPage"; \n
}, \n
], \n
"items";: [\n
{ \n
"type";: "SummaryPage";, \n
"data";: "${payload.aplData}";, \n
"when";:
"${payload.aplData.alternativeRouteSummary != null && @viewportProfile !=
@hubLandscapeXSmall}";, \n
"type";:
"AlternativeRoutesPage";, \n
"data";:
"${payload.aplData}"; \n
} \n
] \n
}, \n
"onConfigChange";: [\n
{ \n
"type";:
"Reinflate"; \n
}, \n
"payload.aplData";: { \n
"fastestRoute";: { \n
"travelTime";: { \n
"unit";:
"offset";: { \n
"value";:
"min";, \n
"value";:
26 \n
}, \n
"base";: { \n
"unit";: "h";, \n
"value";:
0 \n
}, \n
"routeDescriptionText";: "via Via Raffaele Costi and A24";, \n
"statusIcon";:
"https://d2chevgw12wd1d.cloudfront.net/condition_good.png";, \n
"travelDistance";: { \n
"distanceText";:
"14.3";, \n
"distanceUnitText";:
"km";, \n
}, \n
"trafficStatusColor";: "#00CC62";, \n
"hintText";: "";, \n
"trafficSummary";: { \n
"trafficSummarySubtitle";: "Carabinieri at Via degli Armenti, 90, 00155
Rome RM, Italy";, \n
"trafficSummaryTitle";:
"Traffic to Doce Doce at Via dei Cappellari 36";, \n
}, \n
"backgroundImage";:
"https://d2chevgw12wd1d.cloudfront.net/bg_traffic_01.jpg";, \n
"alternativeRouteSummary";: { \n
"alternativeRoutes";: [\n
{ \n

```

```

 "travelTime": {
 "unit": "min",
 "value": 28,
 "base": {
 "h": 0,
 "routeDescriptionText": "via A24 and SR5",
 "statusIcon": "https://d2chevgw12wd1d.cloudfront.net/condition_good.png",
 "travelDistance": {
 "distanceText": "15.7",
 "distanceUnitText": "km",
 "trafficStatusColor": "#00CC62",
 "travelTime": {
 "offset": {
 "min": 31,
 "base": {
 "h": 0,
 "routeDescriptionText": "via A24 and Circonvallazione Tiburtina",
 "statusIcon": "https://d2chevgw12wd1d.cloudfront.net/condition_sluggish.png",
 "travelDistance": {
 "distanceText": "15.9",
 "distanceUnitText": "km",
 "trafficStatusColor": "#FB9C03",
 "alternativeRouteTitle": "Alternate Routes",
 "supportedViewports": [
 "HUB",
 "ROUND",
 "HUB",
 "RECTANGLE",
 "TV",
 "ROUND",
 "TV",
 "RECTANGLE",
 "MOBILE",
 "ROUND",
 "MOBILE",
 "RECTANGLE",
 "PC",
 "ROUND",
 "PC",
 "RECTANGLE",
 "AUTO",
 "ROUND"
],
 "shape": "shape",
 "mode": "mode"
 }
 }
 }
 }
 }
 }
 }
],
 "mode": "mode"
}

```



```

 "AUTO"; \n "shape";:
 "RECTANGLE"; \n }, \n { \n "mode";:
 "HUB"; \n "shape";: "RECTANGLE"; \n
 "minWidth";: 960, \n "maxWidth";: 1279, \n
 "minHeight";: 600, \n "maxHeight";:
 959 \n } \n], \n "presentationSession";: { \n
 "id";: ""; \n "skillId";:
 "DeeAppTrafficSpeechletService"; \n "grantedExtensions";:
 [\n { \n "uri";:
 "aplex:backstack:10"; \n } \n] \n } \n } \n
]}, "status": "finished_successfully", "end_turn": null, "weight": 1.0, "metadata":
 {"serialization_metadata": {"custom_symbol_offsets": []}, "request_id":
 "8d1fad6ad83cba8c-MXP", "message_source": null, "timestamp_": "absolute",
 "message_type": null}, "recipient": "all", "channel": null}, "parent": "1dc54ee3-85a4-
 4a09-a9f5-57b9f78d0c4d", "children": ["db321809-0ae1-49f6-a0e5-4eb5454a3e40"]},
 "db321809-0ae1-49f6-a0e5-4eb5454a3e40": {"id": "db321809-0ae1-49f6-a0e5-4eb5454a3e40",
 "message": {"id": "db321809-0ae1-49f6-a0e5-4eb5454a3e40", "author": {"role":
 "assistant", "name": null, "metadata": {}}, "create_time": 1728826182.246997,
 "update_time": null, "content": {"content_type": "text", "parts": ["This JSON payload
 seems to be related to rendering an Alexa Presentation Language (APL) document, which
 is a system that allows you to create rich visual displays for Alexa-enabled devices
 like Echo Show. Here's a breakdown of the important elements:\n\n### Key Elements
 of the JSON:\n\n1. **Namespace**: \n - `"namespace";`
 "Alexa.Presentation.APL";` indicates that this is an APL document being
 rendered, meaning it's handling some visual output on a supported device.\n\n2.
 APL Version: \n - The `"version";: "1.9";` specifies the APL
 version being used.\n\n3. **Document Structure**: \n - The document contains a layout
 for traffic or route details, including travel time, travel distance, and different
 routes (`"fastestRoute";`, `"alternativeRouteSummary";`). These are
 presented in various containers and layouts, depending on the device's viewport
 (like size, resolution, etc.).\n\n4. **Viewports**: \n - The `supportedViewports`
 section defines how the document should render on different devices (e.g., round or
 rectangular screens, TVs, or mobile devices). The view adapts depending on the screen
 size and shape.\n\n5. **Data Sources**: \n - The `datasources` section holds dynamic
 data about the routes, such as: \n - Travel time and distance for different routes
 (e.g., `"via Via Raffaele Costi and A24";` or `"via A24 and SR5";`).\n
 - Icons for traffic conditions and other display elements.\n\n6. **APL Components**: \n
 - The document makes heavy use of APL components such as `Container`, `Text`, `Image`,
 and custom layouts like `SummaryPage`, `TravelTime`, `TravelDistance`, and others.\n
 - These components dynamically adjust to the viewport's characteristics.\n\n###
 Important Points:\n- **The primary purpose of this document**: It is rendering
 information about traffic conditions and alternative routes. \n- **Presentation
 Token**: The `"presentationToken";` seems to relate to the session tied to
 this APL rendering, potentially to identify it for a specific interaction.\n-
 Customization for Different Viewports: The document adjusts layouts and text sizes
 depending on the device, which is essential for ensuring a good user experience on
 both small and large devices.\n- **APL Commands**: There's a command section
 (`"commands";`) that auto-pages the content after a certain delay (defined by
 `@autoPageDelayInMs`), likely for showing multiple pages of content.\n\n###

```

Errors?

If you're seeing issues with this payload in your testing:

- Ensure all external resources** (like background images or icons) are accessible.
- Check supported device compatibility**: Ensure the device you're testing on supports APL rendering and matches the viewport profile.
- Timing or AutoPage Command**: If the content seems to timeout or not display properly, verify the AutoPage command timing or other commands under the ``commands`` section.

If you're trying to debug an issue related to this APL document rendering, I recommend:

- Checking CloudWatch Logs** for any errors in rendering.
- Validating all dynamic data** in the ``datasources`` section to ensure no missing or incorrectly formatted data.
- Testing on different device types** (e.g., Echo Show) to ensure the layout works as expected across different viewport profiles.

Would you like help troubleshooting or modifying this APL document?

```
{
 "status": "finished_successfully",
 "end_turn": true,
 "weight": 1.0,
 "metadata": {
 "citations": [],
 "content_references": [],
 "gizmo_id": null,
 "message_type": null,
 "model_slug": "gpt-4o",
 "default_model_slug": "gpt-4o",
 "parent_id": "aaa2fbf0-bbc5-4436-bc15-16e4f87c3731",
 "request_id": "8dlfad6ad83cba8c-MXP",
 "timestamp": "absolute",
 "finish_details": {
 "type": "stop",
 "stop_tokens": [200002]
 },
 "is_complete": true,
 "recipient": "all",
 "channel": null,
 "parent": "aaa2fbf0-bbc5-4436-bc15-16e4f87c3731",
 "children": [
 {
 "id": "aaa27996-8419-4b76-a61c-3c5ceeb8fe7b",
 "message": {
 "id": "aaa27996-8419-4b76-a61c-3c5ceeb8fe7b",
 "author": {
 "role": "user",
 "name": null,
 "metadata": {}
 },
 "create_time": 1728826265.662646,
 "update_time": null,
 "content": {
 "content_type": "text",
 "parts": [
 "and does this tell you our intent isn't being consulted?"
]
 }
 }
 }
]
 }
}
```

```
{
 "status": "finished_successfully",
 "end_turn": null,
 "weight": 1.0,
 "metadata": {
 "serialization_metadata": {
 "custom_symbol_offsets": []
 },
 "request_id": "8dlfb01e2ef9ba8c-MXP",
 "message_source": null,
 "timestamp": "absolute",
 "message_type": null,
 "recipient": "all",
 "channel": null,
 "parent": "db321809-0ae1-49f6-a0e5-4eb5454a3e40",
 "children": [
 {
 "id": "7a65fb22-8efd-46e2-b030-d7fed2b2a3a8",
 "message": {
 "id": "7a65fb22-8efd-46e2-b030-d7fed2b2a3a8",
 "author": {
 "role": "assistant",
 "name": null,
 "metadata": {}
 },
 "create_time": 1728826283.019883,
 "update_time": null,
 "content": {
 "content_type": "text",
 "parts": [
 "Yes, this JSON indicates that your intent isn't being considered or invoked. Here's why: The `ConsideredIntents` section lists the intents that Alexa is reviewing for the current session, but it shows that only intents from different skills are being evaluated (indicated by `IntentForDifferentSkill`). Your specific intent (e.g., `GetAIResponseIntent`) isn't in this list, suggesting that Alexa isn't"
]
 }
 }
 }
]
 }
}
```

considering or triggering it based on the user input or current session context.

**Possible reasons why your intent isn't being invoked:**

- \*\*Incorrect invocation\*\*:** The user might not be using the correct invocation phrase or utterance to trigger your intent.
- \*\*Skill configuration\*\*:** There might be an issue with how your skill's intents are configured in the Alexa Developer Console. For example, your intent might not be mapped to the utterance the user is providing.
- \*\*Dialog management\*\*:** If there's a dialog session or other states that are prioritizing different intents, your intent might not be getting selected.
- \*\*Skill's interaction model\*\*:** The model might not be adequately trained on utterances that correspond to your intent, causing Alexa to consider other intents instead.

**What you can do to troubleshoot:**

- \*\*Check the invocation phrases\*\*** in your Alexa Developer Console for your `GetAIResponseIntent`` and ensure they're appropriately configured for the phrases your users are saying.
- \*\*Test your skill\*\*** with a specific utterance in the Alexa Simulator and check if your intent is being invoked properly.
- \*\*Check the CloudWatch logs\*\*** or Alexa Skill Metrics for more detailed logging on how Alexa is matching user utterances to intents.
- \*\*Adjust your interaction model\*\*:** Add more sample utterances and make sure they're unique enough to avoid conflict with other intents or skills.

If the issue persists, you might need to refine the sample utterances and potentially re-deploy the updated interaction model. Would you like more guidance on how to configure this in the developer console?"]}, "status": "finished\_successfully", "end\_turn": true, "weight": 1.0, "metadata": {"citations": [], "content\_references": [], "gizmo\_id": null, "message\_type": null, "model\_slug": "gpt-4o", "default\_model\_slug": "gpt-4o", "parent\_id": "aaa27996-8419-4b76-a61c-3c5ceeb8fe7b", "request\_id": "8d1fb01e2ef9ba8c-MXP", "timestamp\_": "absolute", "finish\_details": {"type": "stop", "stop\_tokens": [200002]}, "is\_complete": true}, "recipient": "all", "channel": null}, "parent": "aaa27996-8419-4b76-a61c-3c5ceeb8fe7b", "children": ["aaa2b383-de0d-4c51-9fdb-9f8897a1b862"]}, {"id": "aaa2b383-de0d-4c51-9fdb-9f8897a1b862", "message": {"id": "aaa2b383-de0d-4c51-9fdb-9f8897a1b862", "author": {"role": "user", "name": null, "metadata": {}}, "create\_time": 1728826409.269294, "update\_time": null, "content": {"content\_type": "text", "parts": ["nd this\n\n\"status\": \"SUCCESSFUL\", \"result\": \"skillExecutionInfo\": {\"invocationRequest\": {\"endpoint\": \"arn:aws:lambda:us-east-1:975050070604:function:4e3cd483-a6cc-4494-b190-11c9b68be961:Release\_0\", \"body\": {\"version\": \"1.0\", \"session\": {\"new\": true, \"sessionId\": \"amzn1.echo-api.session.1234567890\", \"application\": {\"applicationId\": \"amzn1.ask.skill.4e3cd483-a6cc-4494-b190-11c9b68be961\", \"user\": {\"userId\": \"amzn1.ask.account.testUserId\", \"permissions\": {\"scopes\": {\"alexa::devices:all:geolocation:read\", \"status\": \"DENIED\", \"context\": {\"System\": {\"application\": {\"applicationId\": \"

```

"question":amzn1.ask.skill.4e3cd483-a6cc-4494-b190-11c9b68be961","context":{"user":{"userId":"\"","device":{"deviceId":"\"","supportedInterfaces":{"apiEndpoint":"https://api.amazonalexa.com","apiAccessToken":"testApiAccessToken","request":{"type":"IntentRequest","requestId":"amzn1.echo-api.request.testRequestId","timestamp":"2024-10-13T02:24:00Z","locale":"en-US","intent":{"name":"GetAIResponseIntent","confirmationStatus":"NONE","slots":{"query":{"name":"query","value":"the relationship between cops and donuts","confirmationStatus":"NONE","invocationResponse":{"body":"1.0","response":"outputSpeech":"SSML","ssml":"<say>Sorry, I encountered an error processing your request.</say>","userAgent":"ask-node/2.14.0 Node/v16.20.2 sample/hello-world/v1.2","sessionAttributes":{"metrics":{"skillExecutionTimeInMilliseconds":1053,"status":"finished_successfully","end_turn":null,"weight":1.0,"metadata":{"serialization_metadata":{"custom_symbol_offsets":[]},"request_id":"8d1fb39c4a0aba8c-MXP","message_source":null,"timestamp":"absolute","message_type":null,"recipient":"all","channel":null,"parent":"7a65fb22-8efd-46e2-b030-d7fed2b2a3a8","children":["52031d61-506b-4155-8fe8-f4aclad34b40"],"52031d61-506b-4155-8fe8-f4aclad34b40":{"id":"52031d61-506b-4155-8fe8-f4aclad34b40","message":{"id":"52031d61-506b-4155-8fe8-f4aclad34b40","author":{"role":"assistant","name":null,"metadata":{"create_time":1728826426.285499,"update_time":null,"content":{"content_type":"text","parts":["This response indicates that your skill's intent is being invoked, but an error occurred during its execution. Here's a breakdown of what's happening:\n\n### Key Observations:\n\n1. Intent was recognized and invoked: The 'GetAIResponseIntent' intent was triggered successfully, with the user query 'the relationship between cops and donuts'. This means your skill's invocation is working properly.\n\n2. InvocationResponse indicates an error: The skill responded with an SSML message:\n\n```\n"outputSpeech": {\n "type": "SSML",\n "ssml": "<say>Sorry, I encountered an error processing your request.</say>"\n}\n```\n\nThis indicates that an error occurred
```

within the skill's processing logic after the intent was triggered.

**Execution time:** The skill execution took approximately 1053 milliseconds, which indicates the request went through, but something went wrong in the logic before it could return a proper result.

**What could be causing the error?**

- Lambda function error:** If your skill is backed by an AWS Lambda function, there might be an issue with the code that processes the request. This could be an unhandled exception, missing API response, or any number of logical issues.
- API failure:** If your skill relies on an external API (e.g., OpenAI), the error could be related to a failure in calling that API or receiving the response.
- Slot handling or data parsing:** There could be an issue with how the query slot is being handled in your Lambda function (e.g., the value of 'query' might not be processed correctly).

**Next Steps for Debugging:**

- Check AWS CloudWatch Logs:** Look at the Lambda logs in **CloudWatch** to find details about what went wrong during the request. The logs should contain the exact error message or stack trace that caused the skill to fail.
- Error handling in Lambda:** Ensure proper error handling in your Lambda function so that errors are logged and gracefully handled. You could add a try-catch block to catch exceptions and log more details.
- Test API interactions:** If your skill calls an external API (e.g., OpenAI), test the API call separately to ensure it's functioning correctly and returning the expected response.
- Debug locally:** Test your skill's logic locally or using AWS Lambda's test feature with the same input to pinpoint where the error occurs.

This response suggests that the intent trigger is working fine, but the error lies within the skill's processing logic. Checking CloudWatch logs and handling errors more robustly in your code will help identify the root cause.

```
{
 "status": "finished_successfully",
 "end_turn": true,
 "weight": 1.0,
 "metadata": {
 "citations": [],
 "content_references": [],
 "gizmo_id": null,
 "message_type": null,
 "model_slug": "gpt-4o",
 "default_model_slug": "gpt-4o",
 "parent_id": "aaa2b383-de0d-4c51-9fdb-9f8897a1b862",
 "request_id": "8dlfb39c4a0aba8c-MXP",
 "timestamp_": "absolute",
 "finish_details": {
 "type": "stop",
 "stop_tokens": [200002]
 },
 "is_complete": true,
 "recipient": "all",
 "channel": null,
 "parent": "aaa2b383-de0d-4c51-9fdb-9f8897a1b862",
 "children": [
 "aaa25bc6-00c0-4583-81f4-bf9816a50969"
],
 "aaa25bc6-00c0-4583-81f4-bf9816a50969": {
 "id": "aaa25bc6-00c0-4583-81f4-bf9816a50969",
 "message": {
 "id": "aaa25bc6-00c0-4583-81f4-bf9816a50969",
 "author": {
 "role": "user",
 "name": null,
 "metadata": {}
 },
 "create_time": 1728826507.544074,
 "update_time": null,
 "content": {
 "content_type": "text",
 "parts": ["give me instructions on how to do this please"]
 },
 "status": "finished_successfully",
 "end_turn": null,
 "weight": 1.0,
 "metadata": {
 "serialization_metadata": {
 "custom_symbol_offsets": []
 },
 "targeted_reply": "**Test API interactions**:\n\n* If your skill calls an external API (e.g., OpenAI), test the API call separately to ensure it's functioning correctly and returning the expected response.",
 "request_id": "8dlfb602ddb70ea1-MXP",
 "message_source": null,
 "timestamp_": "absolute",
 "message_type": null,
 "recipient": "all",
 "channel": null,
 "parent": "52031d61-506b-4155-8fe8-f4aclad34b40",
 "children": [
 "bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a"
],
 "bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a": {
 "id": "bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a",
 "message": {
 "id": "bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a",
 "author": {
 "role": "system",
 "name": null,
 "metadata": {}
 },
 "create_time": 1728826507.544597,
 "update_time": null,
 "content": {
 "content_type": "text",
 "parts": [""]
 },
 "status": "finished_successfully",
 "end_turn": null,
 "weight": 1.0,
 "metadata": {
 "request_id": "8dlfb602ddb70ea1-MXP",
 "message_source": null,
 "timestamp_": "absolute",
 "message_type": null,

```

```

"is_visually_hidden_from_conversation": true}, "recipient": "all", "channel": null},
"parent": "aaa25bc6-00c0-4583-81f4-bf9816a50969", "children": ["ace688ed-796e-4e5b-9d9e-6adflfald3ad"]], "ace688ed-796e-4e5b-9d9e-6adflfald3ad": {"id": "ace688ed-796e-4e5b-9d9e-6adflfald3ad", "message": {"id": "ace688ed-796e-4e5b-9d9e-6adflfald3ad", "author": {"role": "assistant", "name": null, "metadata": {}}, "create_time": 1728826544.834038, "update_time": null, "content": {"content_type": "text", "parts": ["Here are step-by-step instructions on how to test API interactions separately** to ensure that the external API, like OpenAI's API, is functioning correctly and returning the expected response:\n\n### Step 1: Identify the API Call in Your Lambda Function\n- In your Lambda function code (or skill's backend), look for the part where you are making the API call to the external service, like OpenAI. This will typically involve:\n - An HTTP request to an API endpoint.\n - Passing parameters such as the user's input or query.\n - Receiving and handling the API's response.\n\n### Step 2: Extract the API Request Code\n- Isolate the section of your Lambda function that makes the API call.\n- For example, in Node.js, it might look something like this:\n\n```\njavascript\nconst axios =\nrequire('axios');\n\nasync function callOpenAI(query) {\n const response\n= await axios.post('https://api.openai.com/v1/completions', {\n prompt: query,\n max_tokens: 50,\n temperature: 0.7\n }, {\n headers: {\n 'Authorization': `Bearer YOUR_API_KEY`,\n 'Content-Type': 'application/json'\n }\n });\n return response.data;\n}\n```\n\n### Step 3: Test the API Call Locally\n\n#### Option 1: Use Postman\n1. Download and Install Postman (if not already installed):
[https://www.postman.com/downloads/] (https://www.postman.com/downloads/)\n2. Create a new HTTP request:
- Set the method to `POST`.\n- Enter the API endpoint (e.g., for OpenAI: `https://api.openai.com/v1/completions`).\n3. Add Authorization Header:
- Go to the `Authorization` tab or add the `Authorization` header manually.\n- For OpenAI, you'll need to provide your Bearer Token, like so:\n ```\n Authorization: Bearer YOUR_API_KEY\n ```\n4. Set the Body:
- Choose `raw` and `JSON` format for the body.\n- Input a JSON payload that matches what your Lambda function is sending, for example:\n ```\n json\n {\n "prompt": "the relationship between cops and donuts",\n "max_tokens": 50,\n "temperature": 0.7\n }\n ```\n5. Send the Request:
- Click the Send button.\n- Postman will display the API response, including status code, response body, and any error details. If you receive a valid response, it means the API is working fine.\n\n#### Option 2: Use a Local Script\nIf you prefer to use code instead of Postman:\n1. Create a new script (in JavaScript, Python, etc.) to run locally.\n2. Copy the API call logic from your Lambda function into the script.\n3. Modify the query and ensure that the headers and body are correct.\n4. Run the script locally to check the API response.\n\nFor example, in Node.js:\n\n```\njavascript\nconst axios = require('axios');\n\nasync function testAPI() {\n try {\n const response = await\naxios.post('https://api.openai.com/v1/completions', {\n prompt: "the relationship between cops and donuts",\n max_tokens: 50,\n temperature: 0.7\n }, {\n headers: {\n 'Authorization': `Bearer YOUR_API_KEY`,\n 'Content-Type': 'application/json'\n }\n });\n console.log(response.data);\n } catch (error) {\n console.error('Error

```

```

with API call:"; error.response ? error.response.data :
error.message);\n }\n}\n\ntestAPI();\n```\n\nRun this script locally and check the
console output for the API response.\n\n### Step 4: Debug the API Response\n- If the
API responds successfully, check what is returned (e.g., the expected text completion
for OpenAI).\n- If there is an error, the response should contain details such as:\n
- Invalid API key\n
- Malformed request body\n
- Too many requests (rate-limited)\n
\nThis information will help you debug the API integration in your Lambda
function.\n\n### Step 5: Implement API Call in Lambda Function\n- After verifying the
API works as expected, return to your Lambda function and ensure that:\n
- The API
call is structured correctly.\n
- The data being sent to the API matches the required
format.\n
- You handle errors appropriately (e.g., catch and log any errors during
the API call).\n\n### Step 6: Monitor Logs in AWS CloudWatch\nAfter testing locally,
you can re-deploy your Lambda function and then monitor the logs in AWS CloudWatch
to ensure the API call is functioning correctly when triggered by your Alexa
skill.\n\n#### How to View Logs in CloudWatch:\n1. Go to the AWS Management
Console and open CloudWatch.\n2. Select Logs from the left-hand menu.\n3.
Find the log group for your Lambda function (e.g.,
`/aws/lambda/YOUR_LAMBDA_FUNCTION_NAME`).\n4. Open the most recent log stream to view
details of the function's execution.\n\nLook for errors related to API calls or
unhandled exceptions in the log output.\n\n---\n\nThis process should help you verify
if the API is working correctly, and once confirmed, you can adjust your Lambda code
accordingly to handle any errors and integrate the API smoothly.]]}, "status":
"finished_successfully", "end_turn": true
, "weight": 1.0, "metadata": {"citations": [], "content_references": [], "gizmo_id":
null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug": "gpt-4o",
"parent_id": "bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a", "request_id": "8d1fb602ddb70ea1-
MXP", "timestamp_": "absolute", "finish_details": {"type": "stop", "stop_tokens":
[200002]}, "is_complete": true}, "recipient": "all", "channel": null}, "parent":
"bb9c0aa6-4d09-4564-9e74-95eff2ab4b0a", "children": ["aaa24021-2e84-4833-832a-
0ab4f3ae16e9"]}, {"id": "aaa24021-2e84-4833-832a-0ab4f3ae16e9", "author":
{"role": "user", "name": null, "metadata": {}}, "create_time": 1728826660.324852,
"update_time": null, "content": {"content_type": "text", "parts": ["hey, just thought
of something. other people have already probably done this if it could be done, can
you check the web and github for any sort of opensource premade code blocks to
accomplish just this task? ones where I just plug in my open api key and go"]},
"status": "finished_successfully", "end_turn": null, "weight": 1.0, "metadata":
{"serialization_metadata": {"custom_symbol_offsets": []}, "request_id":
"8d1fb9c0887abadf-MXP", "message_source": null, "timestamp_": "absolute",
"message_type": null}, "recipient": "all", "channel": null}, {"parent": "ace688ed-796e-
4e5b-9d9e-6adflfald3ad", "children": ["e422cd6f-1a07-4dc4-a7fd-49034838d0ee"]},
{"id": "e422cd6f-1a07-4dc4-a7fd-49034838d0ee", "author": {"role":
"assistant", "name": null, "metadata": {}}, "create_time": 1728826675.304464,
"update_time": null, "content": {"content_type": "code", "language": "unknown",
"response_format_name": null, "text": "search("open source code alexa skill
openai integration github")"}, "status": "finished_successfully", "end_turn":
false, "weight": 1.0, "metadata": {"citations": [], "content_references": [],
"gizmo_id": null, "message_type": null, "model_slug": "gpt-4o", "default_model_slug":

```

"gpt-4o", "parent\_id": "aaa24021-2e84-4833-832a-0ab4f3ael6e9", "finish\_details": {"type": "stop", "stop\_tokens": [200007]}, "is\_complete": true, "request\_id": "8d1fb9c0887abadf-MXP", "timestamp\_": "absolute", "recipient": "browser", "channel": null}, "parent": "aaa24021-2e84-4833-832a-0ab4f3ael6e9", "children": ["3ce6b292-5009-4dd7-9d5a-6cf552384fbf"], "3ce6b292-5009-4dd7-9d5a-6cf552384fbf": {"id": "3ce6b292-5009-4dd7-9d5a-6cf552384fbf", "message": {"id": "3ce6b292-5009-4dd7-9d5a-6cf552384fbf", "author": {"role": "tool", "name": "browser", "metadata": {}}, "create\_time": 1728826675.304649, "update\_time": null, "content": {"content\_type": "tether\_browsing\_display", "result": "# \u0020\u0020\u0020GitHub - PedroHaupenthal/alexa-generative-ai: Alexa Skill integrated ... \u0020github.com\u0020\u0020Node.js backend for an AWS Lambda function used by the 'Generative AI' Alexa Skill. The function utilizes OpenAI's Generative Artificial Intelligence API to generate responses in a natural conversation with the device. The goal is to create an interactive and intelligent experience, allowing users ... \u0020 \u0020\u0020GitHub - k4llsh/alexa-gpt: A tutorial on how to use ChatGPT in Alexa \u0020github.com\u0020\u0020\u0020Fund open source developers The ReadME Project. GitHub community articles ... This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the ChatGPT model. ... Go to 'Code' section and add 'openai' to requirements.txt. Your requirements.txt should look like this:\n# \u0020\u0020\u0020alexa-skill \u0020\u0020\u0020 GitHub Topics \u0020\u0020\u0020 GitHub \u0020github.com\u0020\u0020\u0020The Alexa Skills Kit SDK for Python helps you get a skill up and running quickly, letting you focus on skill logic instead of boilerplate code. ... An open source Ruby framework for text and voice chatbots. \u0020\u0020\u0020 ... An Alexa Skill to interface with chat.openai.com. openai alexa-skill alexa-sdk voice-assistant chatgpt Updated Mar 10, 2023 ... \u0020 \u0020\u0020Hey ChatGPT on Amazon Alexa - Community - OpenAI Developer Forum \u0020community.openai.com\u0020\u0020\u0020gpt-4, chatgpt, gpt-4o. binweili July 26, 2024, 9:20pm 1. Hey everyone! I\u0020\u0020\u0020 excited to share that I\u0020\u0020\u0020ve got a working Alexa skill that connects to the GPT-4o and GPT-4o-mini language models. You can now chat with Alexa about any topic you choose. These skills are already certified by Amazon. Check them out at the following links:\n# \u0020\u0020\u0020Let\u0020\u0020\u0020s bring chatGPT to Amazon Alexa! - OpenAI API Community Forum \u0020community.openai.com\u0020\u0020\u0020sono\_matteo February 12, 2023, 3:26pm 1. Let\u0020\u0020\u0020s bring chatGPT to Amazon Alexa! Hi everyone, I have several working skills for Amazon Alexa connecting to the text-davinci-003 language model. You can engage Alexa in a conversation on a topic of your choice. They are already certified by Amazon. You can find them at the following links:\n# \u0020\u0020\u0020\u0020\u0020[Plugin & Source Code] Introducing the ChatGPT OpenAI API Plugin ... \u0020community.openai.com\u0020\u0020\u0020The ChatGPT Plugin for OpenAI API is a powerful tool that seamlessly integrates ChatGPT with the OpenAI API, enabling users to leverage the capabilities of OpenAI's language models in creative and interactive ways inside the ChatGPT interface.. This plugin acts as an intelligent API caller, allowing ChatGPT to generate text completions, engage in dynamic conversations, fine-tune language ... \u0020 \u0020\u0020\u0020Open Source at Alexa \u0020alexa.github.io\u0020\u0020\u0020Code snippets and full demos for building Alexa Skills in Node.js, Python, Java, and C#, or interacting with devices using the Alexa Voice Service (AVS) or Smart Home APIs. Featured tags: C# Java Node.js Python Smart Home In-Skill Purchasing Alexa Conversations Alexa Presentation Language Video Skills Kit\n# \u0020\u0020\u0020\u0020Import an Alexa-Hosted Skill from a Public Git Repository \u0020developer.amazon.com\u0020\u0020\u0020Click Import skill. The Import skill



dialog appears. Enter the .git link to the Git repository that contains the skill that you want to import. Click Continue. The message Creating your Alexa-hosted skill appears. If Alexa validates that the Git repository contains an Alexa skill that it can import, Alexa creates your Alexa-hosted skill.

[\u0008\u2020Alexa Skill with ChatGPT - GitHub\u2020github.com\u0011](#)  
This is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training. ... Fund open source developers The README Project. GitHub community articles Repositories. ... This repository contains an example of how to ...

[\u0009\u2020Try out OpenAI \u0001 in GitHub Copilot and Models\u2020github.blog\u0011](#)  
OpenAI \u0001 is a new series of AI models equipped with advanced reasoning capabilities, trained to think through complex tasks using an internal thought process. During our exploration of using \u0001-preview with GitHub Copilot, we found the model's reasoning capability allows for a deeper understanding of code constraints and edge cases produced ...

[\u0010\u2020First Look: Exploring OpenAI \u0001 in GitHub Copilot\u2020github.blog\u0011](#)  
Here's a first look at where we think it can add value to your day to day. Today, OpenAI released OpenAI \u0001, a new series of AI models equipped with advanced reasoning capabilities to solve hard problems. Like you, we are excited to put the new \u0001 model through its paces and have tested integrating \u0001-preview with GitHub Copilot.

[\u0011\u2020Sign up for OpenAI \u0001 access on GitHub - GitHub Changelog - The GitHub Blog\u2020github.blog\u0011](#)  
You can now join the waitlist for early access to OpenAI \u0001 for use in GitHub Copilot in Visual Studio Code and GitHub Models. The waitlist is currently available to all Copilot users. Join the waitlist for access to OpenAI \u0001 on GitHub.. In Visual Studio Code, you can choose to use \u0001-preview or \u0001-mini to power GitHub Copilot Chat in place of the current default model, GPT-4o.

[\u0012\u20202024 GitHub Accelerator: Meet the 11 projects shaping open source AI\u2020github.blog\u0011](#)  
Founder, Alex Combessie, CEO, and machine learning R&D engineer, Weixuan XIAO, have built an open source library for testing and evaluating large language models (LLMs). Giskard raises the bar for open source AI model quality, advancing overall adoption, research, transparency, and accountability. Designed for data scientists and developers ...

[\u0013\u2020Learn with Our Alexa Python Programming Tutorial - Alexa Skills Kit ... \u2020developer.amazon.com\u0011](#)  
The Alexa Skills Kit SDK for Python simplifies the development of the back-end cloud service for your Alexa skill. A part of the Alexa Skills Kit, the SDK reduces the amount of code you need to write to process Alexa requests and responses and to handle other common skill tasks.. Visit our GitHub repository to find everything you need to build more quickly, including samples listed below.

[\u0014\u2020Integrate Amazon Alexa + OpenAI \(GPT-4, DALL-E, Whisper ... - Zapier\u2020zapier.com\u0011](#)  
Connect Amazon Alexa and OpenAI (GPT-4, DALL-E, Whisper) to integrate crucial parts of your business. With Zapier, you can integrate everything from basic data entry to end-to-end processes. Here are some of the business-critical workflows that people automate with Zapier. Lead management. Marketing campaigns.

[\u0015\u2020GitHub - whyhow-ai/knowledge-table: Knowledge Table is an open-source ... \u2020github.com\u0011](#)  
Knowledge Table is an open-source package designed to simplify extracting and exploring structured data from unstructured documents. It enables the creation of structured knowledge representations, such as tables and graphs, using a natural language query interface. With customizable extraction ...

[\u0016\u2020GitHub - bsab/Alexa-AI-Skill\u2020github.com\u0011](#)  
Azure OpenAI GPT-4.0

will be integrated, and the project is Alexa AI: Boost your Alexa by enhancing its responses. This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the model. If you have directly imported the skill from ... \n# \u301017\u2020Realtime API example for local development environment #1453 - GitHub\u2020github.com\u2020\u3011\nSummary This example will guide developers through the process of integrating the OpenAI Realtime platform into a Node.js application that runs locally on mac (or other device) for development. It ... \n# \u301018\u2020Boosting developer efficiency with GitHub Copilot and AI code ... \u2020allthingsopen.org\u2020\u3011\nopen source, open tech, ... October 11, 2024. Boosting developer efficiency with GitHub Copilot and AI code generation Learn to navigate AI tools with these tips and tricks from Visual Studio's product manager. By ATO Team. Dalia Abo Sheasha, a product manager on the Visual Studio team, sat down with the All Things Open team and shared insights ... \n# \u301019\u2020OpenAI Codex\u2020openai.com\u2020\u3011\nGPT-3\u20202019s main skill is generating natural language in response to a natural language prompt, meaning the only way it affects the world is through the mind of the reader. OpenAI Codex has much of the natural language understanding of GPT-3, but it produces working code\u2020\u2020meaning you can issue commands in English to any piece of software with an ... \nVisible: 0% - 100%", "summary": null, "assets": [], "tether\_id": null}, "status": "finished\_successfully", "end\_turn": null, "weight": 0.0, "metadata": {"\_cite\_metadata": {"citation\_format": {"name": "tether\_og"}, "metadata\_list": [{"type": "webpage", "title": "GitHub - PedroHaupenthal/alexa-generative-ai: Alexa Skill integrated ...", "url": "https://github.com/PedroHaupenthal/alexa-generative-ai", "text": "\nNode.js backend for an AWS Lambda function used by the 'Generative AI' Alexa Skill. The function utilizes OpenAI's Generative Artificial Intelligence API to generate responses in a natural conversation with the device. The goal is to create an interactive and intelligent experience, allowing users ... \n", "pub\_date": "2024-09-21T01:21:00.0000000Z", "extra": null}, {"type": "webpage", "title": "GitHub - k4llsh/alexa-gpt: A tutorial on how to use ChatGPT in Alexa", "url": "https://github.com/k4llsh/alexa-gpt", "text": "\nFund open source developers The README Project. GitHub community articles ... This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the ChatGPT model. ... Go to 'Code' section and add 'openai' to requirements.txt. Your requirements.txt should look like this:\n", "pub\_date": "2024-10-12T04:16:00.0000000Z", "extra": null}, {"type": "webpage", "title": "alexa-skill \u00b7 GitHub Topics \u00b7 GitHub", "url": "https://github.com/topics/alexa-skill", "text": "\nThe Alexa Skills Kit SDK for Python helps you get a skill up and running quickly, letting you focus on skill logic instead of boilerplate code. ... An open source Ruby framework for text and voice chatbots. \ud83e\udd16 ... An Alexa Skill to interface with chat.openai.com. openai alexa-skill alexa-sdk voice-assistant chatgpt Updated Mar 10, 2023 ... \n", "pub\_date": "2024-09-30T04:11:00.0000000Z", "extra": null}, {"type": "webpage", "title": "Hey ChatGPT on Amazon Alexa - Community - OpenAI Developer Forum", "url": "https://community.openai.com/t/hey-chatgpt-on-amazon-alexa/882670", "text": "\ngpt-4, chatgpt, gpt-4o. binweili July 26, 2024, 9:20pm 1. Hey everyone! I\u2020m excited to share that I\u2020ve got a working Alexa skill that connects to the GPT-4o and GPT-4o-mini language models. You can now chat with Alexa about any topic you choose. These skills are already certified by Amazon. Check them out at the following links:\n", "pub\_date": "2024-10-12T12:30:00.0000000Z", "extra": null}, {"type": "webpage", "title": "Let\u2020s bring chatGPT to Amazon Alexa! -

OpenAI API Community Forum", "url": "https://community.openai.com/t/chatgpt-on-top-of-amazon-alexa/58515", "text": "\nsono\_matteo February 12, 2023, 3:26pm 1. Let\u2019s bring chatGPT to Amazon Alexa! Hi everyone, I have several working skills for Amazon Alexa connecting to the text-davinci-003 language model. You can engage Alexa in a conversation on a topic of your choice. They are already certified by Amazon. You can find them at the following links:\n", "pub\_date": "2024-10-11T14:38:00.000000Z", "extra": null}, {"type": "webpage", "title": "[Plugin & Source Code] Introducing the ChatGPT OpenAI API Plugin ...", "url": "https://community.openai.com/t/plugin-source-code-introducing-the-chatgpt-openai-api-plugin-enabling-recursive-workflow-diverse-output-generation-and-model-fine-tuning-capabilities/173397", "text": "\nThe ChatGPT Plugin for OpenAI API is a powerful tool that seamlessly integrates ChatGPT with the OpenAI API, enabling users to leverage the capabilities of OpenAI\u2019s language models in creative and interactive ways inside the ChatGPT interface.. This plugin acts as an intelligent API caller, allowing ChatGPT to generate text completions, engage in dynamic conversations, fine-tune language ... \n", "pub\_date": "2024-10-09T05:04:00.000000Z", "extra": null}, {"type": "webpage", "title": "Open Source at Alexa", "url": "https://alexa.github.io/", "text": "\nCode snippets and full demos for building Alexa Skills in Node.js, Python, Java, and C#, or interacting with devices using the Alexa Voice Service (AVS) or Smart Home APIs. Featured tags: C# Java Node.js Python Smart Home In-Skill Purchasing Alexa Conversations Alexa Presentation Language Video Skills Kit\n", "pub\_date": "2024-10-11T11:55:00.000000Z", "extra": null}, {"type": "webpage", "title": "Import an Alexa-Hosted Skill from a Public Git Repository", "url": "https://developer.amazon.com/en-US/docs/alexa/hosted-skills/alexa-hosted-skills-git-import.html", "text": "\nClick Import skill. The Import skill dialog appears. Enter the .git link to the Git repository that contains the skill that you want to import. Click Continue. The message Creating your Alexa-hosted skill appears. If Alexa validates that the Git repository contains an Alexa skill that it can import, Alexa creates your Alexa-hosted skill.\n", "pub\_date": "2024-10-06T08:45:00.000000Z", "extra": null}, {"type": "webpage", "title": "Alexa Skill with ChatGPT - GitHub", "url": "https://github.com/Joao208/alexa-chatgpt", "text": "\nThis is an Alexa bot developed with the help of OpenAI\u2019s ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training. ... Fund open source developers The README Project. GitHub community articles Repositories. ... This repository contains an example of how to ... \n", "pub\_date": "2024-10-11T23:00:00.000000Z", "extra": null}, {"type": "webpage", "title": "Try out OpenAI o1 in GitHub Copilot and Models", "url": "https://github.blog/news-insights/product-news/try-out-openai-o1-in-github-copilot-and-models/", "text": "\nOpenAI o1 is a new series of AI models equipped with advanced reasoning capabilities, trained to think through complex tasks using an internal thought process. During our exploration of using o1-preview with GitHub Copilot, we found the model\u2019s reasoning capability allows for a deeper understanding of code constraints and edge cases produced ... \n", "pub\_date": "2024-10-12T19:04:00.000000Z", "extra": null}, {"type": "webpage", "title": "First Look: Exploring OpenAI o1 in GitHub Copilot", "url": "https://github.blog/news-insights/product-news/openai-o1-in-github-copilot/", "text": "\nHere\u2019s a first look at where we think it can add value to your day to day. Today, OpenAI released OpenAI o1, a new series of AI models equipped with advanced reasoning capabilities to solve hard problems. Like you, we are excited to put the new o1 model through its paces and have tested integrating o1-preview with GitHub Copilot.\n", "pub\_date": "2024-10-11T13:25:00.000000Z", "extra": null}

null}, {"type": "webpage", "title": "Sign up for OpenAI o1 access on GitHub - GitHub Changelog - The GitHub Blog", "url": "https://github.blog/changelog/2024-09-19-sign-up-for-openai-o1-access-on-github/", "text": "\nYou can now join the waitlist for early access to OpenAI o1 for use in GitHub Copilot in Visual Studio Code and GitHub Models. The waitlist is currently available to all Copilot users. Join the waitlist for access to OpenAI o1 on GitHub.. In Visual Studio Code, you can choose to use o1-preview or o1-mini to power GitHub Copilot Chat in place of the current default model, GPT-4o.\n", "pub\_date": "2024-10-12T10:11:00.0000000Z", "extra": null}, {"type": "webpage", "title": "2024 GitHub Accelerator: Meet the 11 projects shaping open source AI", "url": "https://github.blog/news-insights/company-news/2024-github-accelerator-meet-the-11-projects-shaping-open-source-ai/", "text": "\nFounder, Alex Combessie, CEO, and machine learning R&D engineer, Weixuan XIAO, have built an open source library for testing and evaluating large language models (LLMs). Giskard raises the bar for open source AI model quality, advancing overall adoption, research, transparency, and accountability. Designed for data scientists and developers ...\n", "pub\_date": "2024-10-10T07:08:00.0000000Z", "extra": null}, {"type": "webpage", "title": "Learn with Our Alexa Python Programming Tutorial - Alexa Skills Kit ...", "url": "https://developer.amazon.com/en-US/alexa/alexa-skills-kit/get-deeper/tutorials-code-samples/alexa-skill-python-tutorial", "text": "\nThe Alexa Skills Kit SDK for Python simplifies the development of the back-end cloud service for your Alexa skill. A part of the Alexa Skills Kit, the SDK reduces the amount of code you need to write to process Alexa requests and responses and to handle other common skill tasks.. Visit our GitHub repository to find everything you need to build more quickly, including samples listed below.\n", "pub\_date": "2024-10-10T18:03:00.0000000Z", "extra": null}, {"type": "webpage", "title": "Integrate Amazon Alexa + OpenAI (GPT-4, DALL-E, Whisper ... - Zapier", "url": "https://zapier.com/apps/amazon-alexa/integrations/openai", "text": "\nConnect Amazon Alexa and OpenAI (GPT-4, DALL-E, Whisper) to integrate crucial parts of your business. With Zapier, you can integrate everything from basic data entry to end-to-end processes. Here are some of the business-critical workflows that people automate with Zapier. Lead management. Marketing campaigns.\n", "pub\_date": "2024-10-10T21:22:00.0000000Z", "extra": null}, {"type": "webpage", "title": "GitHub - whyhow-ai/knowledge-table: Knowledge Table is an open-source ...", "url": "https://github.com/whyhow-ai/knowledge-table", "text": "\nKnowledge Table is an open-source package designed to simplify extracting and exploring structured data from unstructured documents. It enables the creation of structured knowledge representations, such as tables and graphs, using a natural language query interface. With customizable extraction ...\n", "pub\_date": "2024-10-11T21:22:00.0000000Z", "extra": null}, {"type": "webpage", "title": "GitHub - bsab/Alexa-AI-Skill", "url": "https://github.com/bsab/Alexa-AI-Skill", "text": "\nAzure OpenAI GPT-4.0 will be integrated, and the project is Alexa AI: Boost your Alexa by enhancing its responses. This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the model. If you have directly imported the skill from ...\n", "pub\_date": "2024-10-12T16:12:00.0000000Z", "extra": null}, {"type": "webpage", "title": "Realtime API example for local development environment #1453 - GitHub", "url": "https://github.com/openai/openai-cookbook/pull/1453/%7B%7B%20revealButtonHref%20%7D%7D", "text": "\nSummary This example will guide developers through the process of integrating the OpenAI Realtime platform into a Node.js application that runs locally on mac (or other device) for

development. It ... \n", "pub\_date": "2024-10-12T22:49:00.000000Z", "extra": null}, {"type": "webpage", "title": "Boosting developer efficiency with GitHub Copilot and AI code ...", "url": "https://allthingsopen.org/articles/developer-efficiency-ai-code-generation", "text": "\nopen source, open tech, ... October 11, 2024. Boosting developer efficiency with GitHub Copilot and AI code generation Learn to navigate AI tools with these tips and tricks from Visual Studio's product manager. By ATO Team. Dalia Abo Sheasha, a product manager on the Visual Studio team, sat down with the All Things Open team and shared insights ... \n", "pub\_date": "2024-10-11T22:45:00.000000Z", "extra": null}, {"type": "webpage", "title": "OpenAI Codex", "url": "https://openai.com/index/openai-codex/", "text": "\nGPT-3\u2019s main skill is generating natural language in response to a natural language prompt, meaning the only way it affects the world is through the mind of the reader. OpenAI Codex has much of the natural language understanding of GPT-3, but it produces working code\u2014meaning you can issue commands in English to any piece of software with an ... \nVisible: 0% - 100%", "pub\_date": "2024-10-11T14:13:00.000000Z", "extra": null}, {"type": "webpage", "title": "Search results for query: &#x27;open source code alexa skill openai integration github&#x27;;", "url": "", "text": "# \u30100\u2020GitHub - PedroHauptenthal/alexa-generative-ai: Alexa Skill integrated ... \u2020github.com\u3011\u2020Node.js backend for an AWS Lambda function used by the &quot;Generative AI&quot; Alexa Skill. The function utilizes OpenAI&#x27;s Generative Artificial Intelligence API to generate responses in a natural conversation with the device. The goal is to create an interactive and intelligent experience, allowing users ... \n# \u30101\u2020GitHub - k4llsh/alexa-gpt: A tutorial on how to use ChatGPT in Alexa\u2020github.com\u3011\u2020Fund open source developers The README Project. GitHub community articles ... This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the ChatGPT model. ... Go to &quot;Code&quot; section and add &quot;openai&quot; to requirements.txt. Your requirements.txt should look like this:\n# \u30102\u2020alexa-skill \u20207 GitHub Topics \u20207 GitHub\u2020github.com\u3011\u2020The Alexa Skills Kit SDK for Python helps you get a skill up and running quickly, letting you focus on skill logic instead of boilerplate code. ... An open source Ruby framework for text and voice chatbots. \u202083e\u2020dd16 ... An Alexa Skill to interface with chat.openai.com. openai alexa-skill alexa-sdk voice-assistant chatgpt Updated Mar 10, 2023 ... \n# \u30103\u2020Hey ChatGPT on Amazon Alexa - Community - OpenAI Developer Forum\u2020community.openai.com\u3011\u2020ngpt-4, chatgpt, gpt-4o. binweili July 26, 2024, 9:20pm 1. Hey everyone! I\u2019m excited to share that I\u2019ve got a working Alexa skill that connects to the GPT-4o and GPT-4o-mini language models. You can now chat with Alexa about any topic you choose. These skills are already certified by Amazon. Check them out at the following links:\n# \u30104\u2020Let\u2019s bring chatGPT to Amazon Alexa! - OpenAI API Community Forum\u2020community.openai.com\u3011\u2020sono\_matteo February 12, 2023, 3:26pm 1. Let\u2019s bring chatGPT to Amazon Alexa! Hi everyone, I have several working skills for Amazon Alexa connecting to the text-davinci-003 language model. You can engage Alexa in a conversation on a topic of your choice. They are already certified by Amazon. You can find them at the following links:\n# \u30105\u2020[Plugin & Source Code] Introducing the ChatGPT OpenAI API Plugin ... \u2020community.openai.com\u3011\u2020The ChatGPT Plugin for OpenAI API is a powerful tool that seamlessly integrates ChatGPT with the OpenAI API, enabling users to leverage the capabilities of OpenAI\u2019s language models in creative and

interactive ways inside the ChatGPT interface.. This plugin acts as an intelligent API caller, allowing ChatGPT to generate text completions, engage in dynamic conversations, fine-tune language ...  
Alexa\alexagithub.io\Code snippets and full demos for building Alexa Skills in Node.js, Python, Java, and C#, or interacting with devices using the Alexa Voice Service (AVS) or Smart Home APIs. Featured tags: C# Java Node.js Python Smart Home In-Skill Purchasing Alexa Conversations Alexa Presentation Language Video Skills Kit  
Import an Alexa-Hosted Skill from a Public Git Repository\developer.amazon.com\Click Import skill. The Import skill dialog appears. Enter the .git link to the Git repository that contains the skill that you want to import. Click Continue. The message Creating your Alexa-hosted skill appears. If Alexa validates that the Git repository contains an Alexa skill that it can import, Alexa creates your Alexa-hosted skill.  
Alexa Skill with ChatGPT - GitHub\github.com\This is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training. ...  
Fund open source developers The README Project. GitHub community articles Repositories. ... This repository contains an example of how to ...  
Try out OpenAI o1 in GitHub Copilot and Models\github.blog\OpenAI o1 is a new series of AI models equipped with advanced reasoning capabilities, trained to think through complex tasks using an internal thought process. During our exploration of using o1-preview with GitHub Copilot, we found the model's reasoning capability allows for a deeper understanding of code constraints and edge cases produced ...  
First Look: Exploring OpenAI o1 in GitHub Copilot\github.blog\Here's a first look at where we think it can add value to your day to day. Today, OpenAI released OpenAI o1, a new series of AI models equipped with advanced reasoning capabilities to solve hard problems. Like you, we are excited to put the new o1 model through its paces and have tested integrating o1-preview with GitHub Copilot.  
Sign up for OpenAI o1 access on GitHub - GitHub Changelog - The GitHub Blog\github.blog\You can now join the waitlist for early access to OpenAI o1 for use in GitHub Copilot in Visual Studio Code and GitHub Models. The waitlist is currently available to all Copilot users. Join the waitlist for access to OpenAI o1 on GitHub.. In Visual Studio Code, you can choose to use o1-preview or o1-mini to power GitHub Copilot Chat in place of the current default model, GPT-4o.  
2024 GitHub Accelerator: Meet the 11 projects shaping open source AI\github.blog\Founder, Alex Combessie, CEO, and machine learning engineer, Weixuan XIAO, have built an open source library for testing and evaluating large language models (LLMs). Giskard raises the bar for open source AI model quality, advancing overall adoption, research, transparency, and accountability. Designed for data scientists and developers ...  
Learn with Our Alexa Python Programming Tutorial - Alexa Skills Kit ...\developer.amazon.com\The Alexa Skills Kit SDK for Python simplifies the development of the back-end cloud service for your Alexa skill. A part of the Alexa Skills Kit, the SDK reduces the amount of code you need to write to process Alexa requests and responses and to handle other common skill tasks.. Visit our GitHub repository to find everything you need to build more quickly, including samples listed below.  
Integrate Amazon Alexa + OpenAI (GPT-4, DALL-E, Whisper ... - Zapier\zapier.com\Connect Amazon Alexa and OpenAI (GPT-4, DALL-E, Whisper) to integrate crucial parts of your business.

With Zapier, you can integrate everything from basic data entry to end-to-end processes. Here are some of the business-critical workflows that people automate with Zapier. Lead management. Marketing campaigns.

[\u0301015\u2020GitHub - whyhow-ai/knowledge-table: Knowledge Table is an open-source ...\u2020github.com\u03011\u2020Knowledge Table is an open-source package designed to simplify extracting and exploring structured data from unstructured documents. It enables the creation of structured knowledge representations, such as tables and graphs, using a natural language query interface. With customizable extraction ...\u0301016\u2020GitHub - bsab/Alexa-AI-Skill\u2020github.com\u03011\u2020Azure OpenAI GPT-4.0 will be integrated, and the project is Alexa AI: Boost your Alexa by enhancing its responses. This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the model. If you have directly imported the skill from ...\u0301017\u2020Realtime API example for local development environment #1453 - GitHub\u2020github.com\u03011\u2020Summary This example will guide developers through the process of integrating the OpenAI Realtime platform into a Node.js application that runs locally on mac \(or other device\) for development. It ... \u0301018\u2020Boosting developer efficiency with GitHub Copilot and AI code ... \u2020allthingsopen.org\u03011\u2020open source, open tech, ... October 11, 2024. Boosting developer efficiency with GitHub Copilot and AI code generation Learn to navigate AI tools with these tips and tricks from Visual Studio's product manager. By ATO Team. Dalia Abo Sheasha, a product manager on the Visual Studio team, sat down with the All Things Open team and shared insights ... \u0301019\u2020OpenAI Codex\u2020openai.com\u03011\u2020GPT-3\u20202019s main skill is generating natural language in response to a natural language prompt, meaning the only way it affects the world is through the mind of the reader. OpenAI Codex has much of the natural language understanding of GPT-3, but it produces working code\u2020meaning you can issue commands in English to any piece of software with an ... \nVisible: 0% - 100%", "pub\\_date": null, "extra": null}\], "original\\_query": null, "command": "search", "args": \["open source code alexa skill openai integration github"\], "status": "finished", "is\\_complete": true, "message\\_type": null, "model\\_slug": "gpt-4o", "default\\_model\\_slug": "gpt-4o", "parent\\_id": "e422cd6f-1a07-4dc4-a7fd-49034838d0ee", "request\\_id": "8d1fb9c0887abadf-MXP", "timestamp\\_": "absolute", "recipient": "all", "channel": null, "parent": "e422cd6f-1a07-4dc4-a7fd-49034838d0ee", "children": \[{"id": "663a2df5-b0c7-4590-b32e-c7c75df1df8c"}\], "663a2df5-b0c7-4590-b32e-c7c75df1df8c": {"id": "663a2df5-b0c7-4590-b32e-c7c75df1df8c", "message": {"id": "663a2df5-b0c7-4590-b32e-c7c75df1df8c", "author": {"role": "assistant", "name": null, "metadata": {}}, "create\\_time": 1728826675.304808, "update\\_time": null, "content": {"content\\_type": "code", "language": "unknown", "response\\_format\\_name": null, "text": "mclick\(\[0, 1, 8, 16\]\)"}, "status": "finished\\_successfully", "end\\_turn": false, "weight": 1.0, "metadata": {"citations": \[\], "content\\_references": \[\], "gizmo\\_id": null, "message\\_type": null, "model\\_slug": "gpt-4o", "default\\_model\\_slug": "gpt-4o", "parent\\_id": "3ce6b292-5009-4dd7-9d5a-6cf552384fbf", "finish\\_details": {"type": "stop", "stop\\_tokens": \[200007\]}, "is\\_complete": true, "request\\_id": "8d1fb9c0887abadf-MXP", "timestamp\\_": "absolute", "recipient": "browser", "channel": null, "parent": "3ce6b292-5009-4dd7-9d5a-6cf552384fbf", "children": \[{"id": "d67606d4-7173-434b-a8cc-9e14bca72f93"}\], "d67606d4-7173-434b-a8cc-9e14bca72f93": {"id": "d67606d4-7173-434b-a8cc-9e14bca72f93", "message": {"id": "d67606d4-7173-434b-a8cc-9e14bca72f93", "author": {"role": "tool", "name": "browser", "metadata": {}}, "create\\_time": 1728826675.304961, "update\\_time": null, "content": {"content\\_type": "tether\\_quote", "url":](#)

"https://github.com/PedroHaupenthal/alexa-generative-ai", "domain": "github.com",  
"text": " \* \u301017\u2020 Financial services \u3011\n \* \u301018\u2020  
Manufacturing \u3011\n\nBy use case\n \* \u301019\u2020 CI/CD & Automation  
\u3011\n \* \u301020\u2020 DevOps \u3011\n \* \u301021\u2020 DevSecOps \u3011\n\n \* Resources \n\nTopics\n \* \u301022\u2020 AI \u3011\n \* \u301023\u2020 DevOps  
\u3011\n \* \u301024\u2020 Security \u3011\n \* \u301025\u2020 Software  
Development \u3011\n \* \u301026\u2020 View all \u3011\n\nExplore\n \*  
\u301027\u2020 Learning Pathways \u2020resources.github.com\u3011\n \*  
\u301028\u2020 White papers, Ebooks, Webinars \u2020resources.github.com\u3011\n \*  
\u301029\u2020 Customer Stories \u3011\n \* \u301030\u2020 Partners  
\u2020partner.github.com\u3011\n\n \* Open Source \n\n \* \u301031\u2020 GitHub  
Sponsors Fund open source developers \u3011\n\n \* \u301032\u2020 The README Project  
GitHub community articles \u3011\n\nRepositories\n \* \u301033\u2020 Topics \u3011\n\n \* \u301034\u2020 Trending \u3011\n \* \u301035\u2020 Collections \u3011\n\n \*  
Enterprise \n\n \* \u301013\u2020 Enterprise platform AI-powered developer platform  
\u3011\n\n\nAvailable add-ons\n \* \u301036\u2020 Advanced Security Enterprise-grade  
security features \u3011\n \* \u301037\u2020 GitHub Copilot Enterprise-grade AI  
features \u3011\n \* \u301038\u2020 Premium Support Enterprise-grade 24/7 support  
\u3011\n\n \* \u301039\u2020 Pricing\u3011 \n\nSearch or jump to...\n\n# Search code,  
repositories, users, issues, pull requests...\n\nSearch \n\nClear \n\n\n\u301040\u2020  
Search syntax tips \u2020docs.github.com\u3011 \n\n# Provide feedback \n\nWe read  
every piece of feedback, and take your input very seriously.\n\nInclude my email  
address so I can be contacted\n\nCancel Submit feedback \n\n# Saved searches \n\n##  
Use saved searches to filter your results more quickly\n\nName\n\nQuery\n\nTo see all  
available qualifiers, see our \u301040\u2020documentation\u2020docs.github.com\u3011.  
\n\nCancel Create saved search \n\n\n\u30101\u2020 Sign in \u3011 \n\n\n\u301041\u2020  
Sign up \u3011 Reseting focus\n\nYou signed in with another tab or window. Reload to  
refresh your session. You signed out in another tab or window. Reload to refresh your  
session. You switched accounts on another tab or window. Reload to refresh your  
session. Dismiss alert\n\n{{ message }}\n\n\n\u301042\u2020 PedroHaupenthal \u3011 /  
\u301043\u2020 alexa-generative-ai\u3011 Public\n\n \* \u301044\u2020 Notifications  
\u3011 You must be signed in to change notification settings\n \* \u301044\u2020 Fork  
2 \u3011 \n \* \u301044\u2020 Star 3 \u3011\n\n\nAlexa Skill integrated with OpenAI for  
dynamic AI-generated user interactions \n\n### License\n\n\n\u301045\u2020 Apache-2.0  
license \u3011 \n\n\n\u301046\u2020 3 stars \u3011 \u301047\u2020 2 forks \u3011  
\u301048\u2020 Branches \u3011 \u301049\u2020 Tags \u3011 \u301050\u2020 Activity  
\u3011 \n\n\n\u301044\u2020 Star \u3011\n\n\n\u301044\u2020 Notifications \u3011 You must  
be signed in to change notification settings\n\n \* \u301043\u2020 Code \u3011\n\n \*  
\u301051\u2020 Issues 0 \u3011\n\n \* \u301052\u2020 Pull requests 0 \u3011\n\n \*  
\u301053\u2020 Actions \u3011\n\n \* \u301054\u2020 Projects 0 \u3011\n\n \*  
\u301055\u2020 Security \u3011\n\n \* \u301056\u2020 Insights \u3011\n\n\nAdditional  
navigation options \n\n \* \u301043\u2020 Code \u3011 \n\n \* \u301051\u2020 Issues  
\u3011 \n\n \* \u301052\u2020 Pull requests \u3011 \n\n \* \u301053\u2020 Actions \u3011  
\n\n \* \u301054\u2020 Projects \u3011 \n\n \* \u301055\u2020 Security \u3011 \n\n \*  
\u301056\u2020 Insights \u3011 \n\n\n# PedroHaupenthal/alexa-generative-ai\n\nThis  
commit does not belong to any branch on this repository, and may belong to a fork  
outside of the  
repository.\n\n\n0a0main\n\n\n\u301048\u2020 Branches \u3011 \u301049\u2020 Tags \u3011\n\n\nGo  
to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message



\n\nLast commit date    \n\n## Latest commit\n\n## History\n\n\u0305\u020222  
 Commits\u030511  
 \n\n\u0305\u0202assets\u030511\n\n\u0305\u0202assets\u030511\n\n\u0305\u0202interactionModels/  
 custom\u030511\n\n\u0305\u0202lambda\u030511\n\n\u0305\u0202lambda\u030511\n\n\u0305\u02022020.gitignore\u030511\n\n\u0305\u02022020.gitignore\u030511\n\n\u0305\u02022020.prettierrc\u030511\n\n\u0305\u02022020.prettierrc\u030511\n\n\u0305\u0202LICENSE\u030511\n\n\u0305\u0202LICENSE\u030511\n\n\u0305\u0202README.md\u030511\n\n\u0305\u0202README.md\u030511\n\n\u0305\u0202skill.json\u030511\n\n\u0305\u0202skill.json\u030511\n\nView all files  
 \n\n## Repository files navigation \n\n \* README \n \* Apache-2.0 license \n\n# Alexa  
 Generative AI Skill \n\n[Image 0: Architectural Diagram]\n\n## About \n\nNode.js  
 backend for an AWS Lambda function used by the "Generative AI" Alexa Skill.  
 The function utilizes OpenAI's Generative Artificial Intelligence API to generate  
 responses in a natural conversation with the device.\n\nThe goal is to create an  
 interactive and intelligent experience, allowing users to receive dynamic responses to  
 a variety of questions and commands.\n\n## Architecture \ufe0f\n\nUsing the simple  
 diagram above that illustrates the architecture\n\n \* Alexa Skill: Skill that  
 receives voice commands from users.\n \* AWS Lambda: Lambda function that processes  
 the received commands and interacts with the OpenAI API.\n \* OpenAI API: API that  
 generates responses using generative AI based on the sent conversation.\n\n## Setup  
 \n\n### Prerequisites\n\n \* AWS account with access to AWS Lambda and Alexa Skills  
 Kit.\n \* OpenAI account with access to the OpenAI API.\n\n### Configuring the Alexa  
 Skill\n\n1. Create a new Alexa skill using the Alexa Skills Kit (ASK). Define the  
 intentions and interaction models as needed.\n2. Implement the skill logic that will  
 process user inputs and interact with the Lambda function.\n\n### Configuring AWS  
 Lambda\n\n1. Upload the code from this repository to your Lambda function.\n2. Set  
 up Lambda trigger as Alexa/IoT and link your Skill ID.\n3. Copy the Lambda  
 function's ARN and set it as the endpoint for your Alexa Skill.\n\n###  
 Integration with OpenAI API\n\n1. Obtain an API key from OpenAI.\n2. Configure the  
 OpenAI API credentials in the Lambda function code.\n\n## Testing and Deployment \n\n  
 \* Test the skill using the Alexa test console or an actual Alexa device.\n\n##  
 Limitations and Warnings \u026a\ufe0f\n\n \* Developer Mode: This skill is only  
 available in Developer mode, meaning it is not visible or accessible in the Skills  
 Store and is only available for testing.\n \* Amazon Account: If you intend to test  
 with a physical Alexa device (e.g., Amazon Echo, Echo Dot, Echo Show), the email of  
 the Amazon account must be the same as that of the Alexa Skills Kit.\n \* Conversation  
 History: The current version does not support conversation history.\n\n##  
 About\n\nAlexa Skill integrated with OpenAI for dynamic AI-generated user interactions  
 \n\n### Topics\n\n\u0305\u0202 nodejs \u030511 \u0305\u0202 alexa \u030511  
 \u0305\u0202 ai \u030511 \u0305\u0202 aws-lambda \u030511 \u0305\u0202 openai \u030511  
 \u0305\u0202 alexa-skill \u030511 \u0305\u0202 gpt-4 \u030511 \u0305\u0202  
 generative-ai \u030511 \n\n", "title": "GitHub - PedroHaupenthal/alexagenerative-ai:  
 Alexa Skill integrated with OpenAI for dynamic AI-generated user interactions",  
 "tether\_id": null}, "status": "finished\_successfully", "end\_turn": null, "weight": 0.0,  
 "metadata": {"\_cite\_metadata": {"citation\_format": {"name": "tether\_log"},  
 "metadata\_list": [{"type": "webpage", "title": "GitHub - PedroHaupenthal/alexagenerative-ai:  
 Alexa Skill integrated with OpenAI for dynamic AI-generated user  
 interactions", "url": "https://github.com/PedroHaupenthal/alexagenerative-ai", "text":  
 " \* \u0305\u0202 Financial services \u030511\n \* \u0305\u0202 Manufacturing

\u3011\n\nBy use case\n \* \u301019\u2020 CI/CD & Automation \u3011\n \*  
\u301020\u2020 DevOps \u3011\n \* \u301021\u2020 DevSecOps \u3011\n\n \* Resources  
\n\nTopics\n \* \u301022\u2020 AI \u3011\n \* \u301023\u2020 DevOps \u3011\n \*  
\u301024\u2020 Security \u3011\n \* \u301025\u2020 Software Development \u3011\n\n \* \u301026\u2020 View all \u3011\n\nExplore\n \* \u301027\u2020 Learning Pathways  
\u2020resources.github.com\u3011\n \* \u301028\u2020 White papers, Ebooks, Webinars  
\u2020resources.github.com\u3011\n \* \u301029\u2020 Customer Stories \u3011\n \*  
\u301030\u2020 Partners \u2020partner.github.com\u3011\n\n \* Open Source \n\n \*  
\u301031\u2020 GitHub Sponsors Fund open source developers \u3011\n\n \*  
\u301032\u2020 The README Project GitHub community articles \u3011\n\nRepositories\n \* \u301033\u2020 Topics \u3011\n \* \u301034\u2020 Trending \u3011\n \*  
\u301035\u2020 Collections \u3011\n\n \* Enterprise \n\n \* \u301013\u2020  
Enterprise platform AI-powered developer platform \u3011\n\nAvailable add-ons\n \*  
\u301036\u2020 Advanced Security Enterprise-grade security features \u3011\n \*  
\u301037\u2020 GitHub Copilot Enterprise-grade AI features \u3011\n \*  
\u301038\u2020 Premium Support Enterprise-grade 24/7 support \u3011\n\n \*  
\u301039\u2020Pricing\u3011 \n\nSearch or jump to...\n\n# Search code, repositories,  
users, issues, pull requests...\n\nSearch \n\nClear \n\n\u301040\u2020 Search syntax  
tips \u2020docs.github.com\u3011 \n\n# Provide feedback \n\nWe read every piece of  
feedback, and take your input very seriously.\n\nInclude my email address so I can be  
contacted\n\nCancel Submit feedback \n\n# Saved searches \n\n## Use saved searches  
to filter your results more quickly\n\nName\n\nQuery\n\nTo see all available  
qualifiers, see our \u301040\u2020documentation\u2020docs.github.com\u3011. \n\nCancel  
Create saved search \n\n\u30101\u2020 Sign in \u3011 \n\n\u301041\u2020 Sign up \u3011  
Resetting focus\n\nYou signed in with another tab or window. Reload to refresh your  
session. You signed out in another tab or window. Reload to refresh your session. You  
switched accounts on another tab or window. Reload to refresh your session. Dismiss  
alert\n\n{{ message }}\n\n\u301042\u2020 PedroHauptenthal \u3011 /  
\u301043\u2020alexa-generative-ai\u3011 Public\n\n \* \u301044\u2020 Notifications  
\u3011 You must be signed in to change notification settings\n\n \* \u301044\u2020 Fork  
2 \u3011 \n\n \* \u301044\u2020 Star 3 \u3011\n\nAlexa Skill integrated with OpenAI for  
dynamic AI-generated user interactions \n\n### License\n\n\u301045\u2020 Apache-2.0  
license \u3011 \n\n\u301046\u2020 3 stars \u3011 \u301047\u2020 2 forks \u3011  
\u301048\u2020 Branches \u3011 \u301049\u2020 Tags \u3011 \u301050\u2020 Activity  
\u3011 \n\n\u301044\u2020 Star \u3011\n\n\u301044\u2020 Notifications \u3011 You must  
be signed in to change notification settings\n\n \* \u301043\u2020 Code \u3011\n\n \*  
\u301051\u2020 Issues 0 \u3011\n\n \* \u301052\u2020 Pull requests 0 \u3011\n\n \*  
\u301053\u2020 Actions \u3011\n\n \* \u301054\u2020 Projects 0 \u3011\n\n \*  
\u301055\u2020 Security \u3011\n\n \* \u301056\u2020 Insights \u3011\n\n\nAdditional  
navigation options \n\n \* \u301043\u2020 Code \u3011 \n\n \* \u301051\u2020 Issues  
\u3011 \n\n \* \u301052\u2020 Pull requests \u3011 \n\n \* \u301053\u2020 Actions \u3011  
\n\n \* \u301054\u2020 Projects \u3011 \n\n \* \u301055\u2020 Security \u3011 \n\n \*  
\u301056\u2020 Insights \u3011 \n\n# PedroHauptenthal/alexa-generative-ai\n\nThis  
commit does not belong to any branch on this repository, and may belong to a fork  
outside of the  
repository.\n\n\u00a0main\n\n\u301048\u2020Branches\u3011\u301049\u2020Tags\u3011\n\nnGo  
o to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message  
\n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301057\u20202022  
Commits\u3011

\n\n\u301058\u2020assets\u3011\n\n\u301058\u2020assets\u3011\n\n\u301059\u2020interactionModels/ custom\u3011\n\n\u301059\u2020interactionModels/ custom\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301061\u2020.gitignore\u3011\n\n\u301061\u2020.gitignore\u3011\n\n\u301062\u2020.prettierrc\u3011\n\n\u301062\u2020.prettierrc\u3011\n\n\u301045\u2020LICENSE\u3011\n\n\u301045\u2020LICENSE\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301064\u2020skill.json\u3011\n\n\u301064\u2020skill.json\u3011\n\nView all files

\n\n## Repository files navigation \n\n \* README \n \* Apache-2.0 license \n\n# Alexa Generative AI Skill \n\n[Image 0: Architectural Diagram]\n\n## About \n\nNode.js backend for an AWS Lambda function used by the "Generative AI" Alexa Skill. The function utilizes OpenAI's Generative Artificial Intelligence API to generate responses in a natural conversation with the device.\n\nThe goal is to create an interactive and intelligent experience, allowing users to receive dynamic responses to a variety of questions and commands.\n\n## Architecture \ufe0f\n\nUsing the simple diagram above that illustrates the architecture\n\n \* Alexa Skill: Skill that receives voice commands from users.\n \* AWS Lambda: Lambda function that processes the received commands and interacts with the OpenAI API.\n \* OpenAI API: API that generates responses using generative AI based on the sent conversation.\n\n## Setup \n\n### Prerequisites\n\n \* AWS account with access to AWS Lambda and Alexa Skills Kit.\n \* OpenAI account with access to the OpenAI API.\n\n### Configuring the Alexa Skill\n\n 1. Create a new Alexa skill using the Alexa Skills Kit (ASK). Define the intentions and interaction models as needed.\n 2. Implement the skill logic that will process user inputs and interact with the Lambda function.\n\n### Configuring AWS Lambda\n\n 1. Upload the code from this repository to your Lambda function.\n 2. Set up Lambda trigger as Alexa/IoT and link your Skill ID.\n 3. Copy the Lambda function's ARN and set it as the endpoint for your Alexa Skill.\n\n### Integration with OpenAI API\n\n 1. Obtain an API key from OpenAI.\n 2. Configure the OpenAI API credentials in the Lambda function code.\n\n## Testing and Deployment \n\n \* Test the skill using the Alexa test console or an actual Alexa device.\n\n## Limitations and Warnings \u26a0\ufe0f\n\n \* Developer Mode: This skill is only available in Developer mode, meaning it is not visible or accessible in the Skills Store and is only available for testing.\n \* Amazon Account: If you intend to test with a physical Alexa device (e.g., Amazon Echo, Echo Dot, Echo Show), the email of the Amazon account must be the same as that of the Alexa Skills Kit.\n \* Conversation History: The current version does not support conversation history.\n\n## About\n\nAlexa Skill integrated with OpenAI for dynamic AI-generated user interactions\n\n### Topics\n\n\u301065\u2020 nodejs \u3011 \u301066\u2020 alexa \u3011 \u301067\u2020 ai \u3011 \u301068\u2020 aws-lambda \u3011 \u301069\u2020 openai \u3011 \u301070\u2020 alexa-skill \u3011 \u301071\u2020 gpt-4 \u3011 \u301072\u2020 generative-ai \u3011\n\n", "pub\_date": null, "extra": null}], "original\_query": null, "command": "mclick", "args": "[0, 1, 8, 16]", "status": "finished", "is\_complete": true, "message\_type": null, "model\_slug": "gpt-4o", "default\_model\_slug": "gpt-4o", "parent\_id": "663a2df5-b0c7-4590-b32e-c7c75df1df8c", "request\_id": "8d1fb9c0887abadf-MXP", "timestamp\_": "absolute", "recipient": "all", "channel": null, "parent": "663a2df5-b0c7-4590-b32e-c7c75df1df8c", "children": ["6036f1ab-60a7-46d1-9199-2754a3bfcf39"], "6036f1ab-60a7-46d1-9199-2754a3bfcf39": {"id": "6036f1ab-60a7-46d1-9199-2754a3bfcf39", "message": {"id": "6036f1ab-60a7-46d1-9199-2754a3bfcf39", "author": {"role": "tool", "name": "browser", "metadata": {}}, "create\_time": 1728826675.305079, "update\_time": null, "content": {"content\_type":

```
tether_quote", "url": "https://github.com/k4llsh/alexa-gpt", "domain": "github.com",
"text": "\n Additional navigation options \n\n * \u002F\u002F Code \u002F\u002F Issues \u002F\u002F Pull requests \u002F\u002F Actions \u002F\u002F Projects \u002F\u002F Security \u002F\u002F Insights \u002F\u002F\nk4llsh/alexa-gpt\n\nThis commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.\n\n\u002F\u002F\u002F\u002F Branches \u002F\u002F\u002F\u002F Tags \u002F\u002F\u002F\u002F\no to file \u002F\u002F\u002F\u002F Folders and files \u002F\u002F\u002F\u002F Name Name \u002F\u002F\u002F\u002F Last commit message \u002F\u002F\u002F\u002F Latest commit \u002F\u002F\u002F\u002F History \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F Commits \u002F\u002F\u002F\u002F\n\n\n\u002F\u002F\u002F\u002F ask \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F ask \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F images \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F images \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F lambda \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F lambda \u002F\u002F\u002F\u002F models \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F models \u002F\u002F\u002F\u002F LICE NSE \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F LICENSE \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F README.md \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F README.md \u002F\u002F\u002F\u002F ask-resources.json \u002F\u002F\u002F\u002F ask-resources.json \u002F\u002F\u002F\u002F json_editor.json \u002F\u002F\u002F\u002F json_edi tor.json \u002F\u002F\u002F\u002F skill.json \u002F\u002F\u002F\u002F skill.json \u002F\u002F\u002F\u002F\nView all files \u002F\u002F\u002F\u002F Repository files navigation \u002F\u002F\u002F\u002F * README \u002F\u002F\u002F\u002F * MIT license \u002F\u002F\u002F\u002F Alexa GPT \u002F\u002F\u002F\u002F [Image 0: License: MIT] \u002F\u002F\u002F\u002F Boost your Alexa by making it respond as ChatGPT. \u002F\u002F\u002F\u002F This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the ChatGPT model. \u002F\u002F\u002F\u002F [Image 1] \u002F\u002F\u002F\u002F Prerequisites \u002F\u002F\u002F\u002F * An \u002F\u002F\u002F\u002F Amazon Developer account \u002F\u002F\u002F\u002F developer.amazon.com \u002F\u002F\u002F\u002F * An \u002F\u002F\u002F\u002F OpenAI API key \u002F\u002F\u002F\u002F platform.openai.com \u002F\u002F\u002F\u002F Step-by-step tutorial \u002F\u002F\u002F\u002F 1. \u002F\u002F\u002F\u002F Log in to your Amazon Developer account and navigate to the \u002F\u002F\u002F\u002F Alexa Developer Console \u002F\u002F\u002F\u002F developer.amazon.com \u002F\u002F\u002F\u002F 2. \u002F\u002F\u002F\u002F Click on "Create Skill" and name the skill "Chat". Choose the primary locale according to your language. \u002F\u002F\u002F\u002F [Image 2: name your skill] \u002F\u002F\u002F\u002F 3. \u002F\u002F\u002F\u002F Choose "Other" and "Custom" for the model. \u002F\u002F\u002F\u002F [Image 3: type of experience] \u002F\u002F\u002F\u002F [Image 4: choose a model] \u002F\u002F\u002F\u002F 4. \u002F\u002F\u002F\u002F Choose "Alexa-hosted (Python)" for the backend resources. \u002F\u002F\u002F\u002F [Image 5: hosting services] \u002F\u002F\u002F\u002F 5. \u002F\u002F\u002F\u002F You now have two options: \u002F\u002F\u002F\u002F * Click on "Import Skill", paste the link of this repository (\u002F\u002F\u002F\u002F https://github.com/k4llsh/alexa-gpt.git \u002F\u002F\u002F\u002F) and click on "Import". [Image 6: template] \u002F\u002F\u002F\u002F Or if you want to create the skill manually \u002F\u002F\u002F\u002F * Select "Start from Scratch" and click on "Create Skill" \u002F\u002F\u002F\u002F [Image 7: template] \u002F\u002F\u002F\u002F 6. \u002F\u002F\u002F\u002F In the "Build" section, navigate to the "JSON Editor" tab. \u002F\u002F\u002F\u002F 7. \u002F\u002F\u002F\u002F If you have directly imported the skill from this repository, just change the "invocationName" to "chat" or another preferred word for activation and proceed to step 12. \u002F\u002F\u002F\u002F However, if you chose to manually create the skill, replace the existing JSON content with the \u002F\u002F\u002F\u002F provided JSON content \u002F\u002F\u002F\u002F \u002F\u002F\u002F\u002F {\n "interactionModel":\n {\n "languageModel":\n {\n "invocationName":\n "chat",\n "intents":\n [\n {\n "name":\n "GptQueryIntent",\n "slots":\n [\n {\n "name":\n "query",\n "type":\n "AMAZON.Person"\n }\n],\n
```

```

"samples";: [\n
"{query}";\n]\n }, \n
{\n "name";: "AMAZON.CancelIntent";, \n
"samples";: []\n }, \n {\n
"name";: "AMAZON.HelpIntent";, \n
"samples";: []\n }, \n {\n
"name";: "AMAZON.StopIntent";, \n
"samples";: []\n }, \n {\n
"name";: "AMAZON.NavigateHomeIntent";, \n
"samples";: []\n }\n], \n
"types";: []\n }\n }\n }\n\n[Image 8:
json_editor]\n\n### 8.\n\nSave the model and click on "Build Model".\n\n###
9.\n\nGo to "Code" section and add "openai" to requirements.txt.
Your requirements.txt should look like this:\n\n ask-sdk-core==1.11.0\n
boto3==1.9.216\n requests>=2.20.0\n\n### 10.\n\nCreate an OpenAI API key on the
\u301068\u2020API keys page\u2020platform.openai.com\u3011 by clicking "+ Create
new secret key".\n\n### 11.\n\nReplace your lambda_functions.py file with the
\u301071\u2020provided lambda_function.py\u3011.\n\n from
ask_sdk_core.dispatch_components import AbstractExceptionHandler\n from
ask_sdk_core.dispatch_components import AbstractRequestHandler\n from
ask_sdk_core.skill_builder import SkillBuilder\n from ask_sdk_core.handler_input
import HandlerInput\n from ask_sdk_model import Response\n import
ask_sdk_core.utils as ask_utils\n import requests\n import logging\n import
json\n\n # Set your OpenAI API key\n api_key = "YOUR_API_KEY"\n\n
logger = logging.getLogger(__name__)\n logger.setLevel(logging.INFO)\n\n class
LaunchRequestHandler(AbstractRequestHandler):\n """Handler for
Skill Launch."""\n def can_handle(self, handler_input):\n
type: (HandlerInput) -> bool\n return
ask_utils.is_request_type("LaunchRequest")(handler_input)\n\n def
handle(self, handler_input):\n # type: (HandlerInput) -> Response\n
speak_output = "Chat G.P.T. mode activated"\n\n session_attr =
handler_input.attributes_manager.session_attributes\n
session_attr["chat_history"] = []\n\n return (\n
handler_input.response_builder\n .speak(speak_output)\n
 .ask(speak_output)\n .response\n)\n\n class
GptQueryIntentHandler(AbstractRequestHandler):\n """Handler for
Gpt Query Intent."""\n def can_handle(self, handler_input):\n
type: (HandlerInput) -> bool\n return
ask_utils.is_intent_name("GptQueryIntent")(handler_input)\n\n def
handle(self, handler_input):\n # type: (HandlerInput) -> Response\n
query =
handler_input.request_envelope.request.intent.slots["query"].value\n\n
session_attr = handler_input.attributes_manager.session_attributes\n if
"chat_history" not in session_attr:\n
session_attr["chat_history"] = []\n\n response =
generate_gpt_response(session_attr["chat_history"], query)\n
session_attr["chat_history"].append((query, response))\n\n return
(\n
handler_input.response_builder\n .speak(response)\n

```

```

 .ask("Any other
questions?"")\n
 .response\n
)\n\n class
CatchAllExceptionHandler(AbstractExceptionHandler):\n """Generic
error handling to capture any syntax or routing errors."""\n def
can_handle(self, handler_input, exception):\n # type: (HandlerInput,
Exception) -> bool\n return True\n\n def handle(self,
handler_input, exception):\n # type: (HandlerInput, Exception) ->
Response\n logger.error(exception, exc_info=True)\n\n speak_output = "Sorry, I had trouble doing what you asked. Please try
again."\n\n return (\n
handler_input.response_builder\n .speak(speak_output)\n
 .ask(speak_output)\n
 .respons", "title": "GitHub -
k4llsh/alexa-gpt: A tutorial on how to use ChatGPT in Alexa", "tether_id": null},
"status": "finished_successfully", "end_turn": null, "weight": 0.0, "metadata":
{"_cite_metadata": {"citation_format": {"name": "tether_og"}, "metadata_list":
[{"type": "webpage", "title": "GitHub - k4llsh/alexa-gpt: A tutorial on how to use
ChatGPT in Alexa", "url": "https://github.com/k4llsh/alexa-gpt", "text": "\n
Additional navigation options \n\n * \u30104\u2020 Code \u3011 \n * \u30105\u2020
Issues \u3011 \n * \u301053\u2020 Pull requests \u3011 \n * \u301054\u2020 Actions
\u3011 \n * \u301055\u2020 Projects \u3011 \n * \u301056\u2020 Security \u3011 \n *
\u301057\u2020 Insights \u3011 \n\n# k4llsh/alexa-gpt\n\nThis commit does not belong
to any branch on this repository, and may belong to a fork outside of the
repository.\n\n\u00a0main\n\n\u301049\u2020Branches\u3011\u301050\u2020Tags\u3011\n\nGo
to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message
\n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301058\u202072
Commits\u3011
\n\n\u301059\u2020. ask\u3011\n\n\u301059\u2020. ask\u3011\n\n\u301060\u2020images\u3011
\n\n\u301060\u2020images\u3011\n\n\u301061\u2020lambda\u3011\n\n\u301061\u2020lambda\u
3011\n\n\u301062\u2020models\u3011\n\n\u301062\u2020models\u3011\n\n\u301064\u2020LICE
NSE\u3011\n\n\u301064\u2020LICENSE\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301063\u
2020README.md\u3011\n\n\u301064\u2020ask-resources.json\u3011\n\n\u301064\u2020ask-
resources.json\u3011\n\n\u301065\u2020json_editor.json\u3011\n\n\u301065\u2020json_edi
tor.json\u3011\n\n\u301066\u2020skill.json\u3011\n\n\u301066\u2020skill.json\u3011\n\n
View all files \n\n## Repository files navigation \n\n * README \n * MIT license
\n\n# Alexa GPT\n\n[Image 0: License: MIT]\n\nBoost your Alexa by making it respond as
ChatGPT.\n\nThis repository contains a tutorial on how to create a simple Alexa skill
that uses the OpenAI API to generate responses from the ChatGPT model.\n\n[Image
1]\n\n## Prerequisites\n\n * An \u301067\u2020Amazon Developer
account\u2020developer.amazon.com\u3011\n * An \u301068\u2020OpenAI API
key\u2020platform.openai.com\u3011\n\n## Step-by-step tutorial\n\n### 1. \n\nLog in to
your Amazon Developer account and navigate to the \u301069\u2020Alexa Developer
Console\u2020developer.amazon.com\u3011.\n\n### 2. \n\nClick on "Create Skill"
and name the skill "Chat". Choose the primary locale according to your
language.\n\n[Image 2: name your skill]\n\n### 3. \n\nChoose "Other" and
"Custom" for the model.\n\n[Image 3: type of experience]\n\n[Image 4: choose
a model]\n\n### 4. \n\nChoose "Alexa-hosted (Python)" for the backend
resources.\n\n[Image 5: hosting services]\n\n### 5. \n\nYou now have two options:\n\n
* Click on "Import Skill", paste the link of this repository
(\u301070\u2020https://github.com/k4llsh/alexa-gpt.git\u3011) and click on

```

```
<code>"Import";\n\n\n * Select "Start from Scratch", and click on "Create Skill"\n\n[Image 7: template]\n\n### 6.\n\nIn the "Build" section, navigate to the "JSON Editor" tab.\n\n### 7.\n\nIf you have directly imported the skill from this repository, just change the "invocationName" to "chat" or another preferred word for activation and proceed to step 12.\n\nHowever, if you chose to manually create the skill, replace the existing JSON content with the \u003d\u003dprovided JSON content\u003d\u003d:\n\n {\n \"interactionModel\": {\n \"name\": \""chat"\", \n \"intentNames\": [\n \""GptQueryIntent"\", \n \""AMAZON.Person"\"],\n \"samples\": [\n { \"query\": \"ask_sdk_core\" },\n { \"query\": \"cancel intent\" },\n { \"query\": \"help intent\" },\n { \"query\": \"stop intent\" },\n { \"query\": \"navigate home intent\" }],\n \"types\": [] }\n }\n\n[Image 8: json_editor]\n\n### 8.\n\nSave the model and click on "Build Model".\n\n### 9.\n\nGo to "Code" section and add "openai" to requirements.txt. Your requirements.txt should look like this:\n\nask-sdk-core==1.11.0\nboto3==1.9.216\nrequests==2.20.0\n\n### 10.\n\nCreate an OpenAI API key on the \u003d\u003dAPI keys page\u003d\u003dplatform.openai.com by clicking "+ Create new secret key".\n\n### 11.\n\nReplace your lambda_functions.py file with the \u003d\u003dlambda_function.py\u003d\u003d from ask_sdk_core.dispatch_components import AbstractExceptionHandler\nfrom ask_sdk_core.dispatch_components import AbstractRequestHandler\nfrom ask_sdk_core.skill_builder import SkillBuilder\nfrom ask_sdk_core.handler_input import HandlerInput\nfrom ask_sdk_model import Response\nimport logging\nimport requests\nimport sys\n\n# Set your OpenAI API key\napi_key = "YOUR_API_KEY"\nlogger = logging.getLogger(__name__)\nlogger.setLevel(logging.INFO)\n\nclass LaunchRequestHandler(AbstractRequestHandler):\n def can_handle(self, handler_input):\n # type: (HandlerInput) -> bool\n return True\n\n def handle(self, handler_input):\n # type: (HandlerInput) -> Response\n speak_output = "Chat G.P.T. mode activated"\n\n session_attr = handler_input.attributes_manager.session_attributes\n session_attr[\"chat_history\"] = []\n\n return (\n handler_input.response_builder\n .speak(speak_output)\n .with_message(\"I'm still learning how to best respond to questions about the Amazon Alexa Skills Kit. Please try asking me something else.\")\n .build()\n)\n\nlambda_handler = SkillBuilder().add_request_handler(LaunchRequestHandler()).build_lambda_handler()\n\nif __name__ == '__main__':\n lambda_handler.invoke(sys.argv[1:])</code>
```

```

 .ask(speak_output)\n .response\n)\n\n class
GptQueryIntentHandler(AbstractRequestHandler):\n """Handler for
Gpt Query Intent."""\n def can_handle(self, handler_input):\n
type: (HandlerInput) -> bool\n return
ask_utils.is_intent_name("GptQueryIntent")(handler_input)\n\n def
handle(self, handler_input):\n # type: (HandlerInput) -> Response\n
query =
handler_input.request_envelope.request.intent.slots["query"].value\n\n
session_attr = handler_input.attributes_manager.session_attributes\n if
"chat_history" not in session_attr:\n
session_attr["chat_history"] = []\n response =
generate_gpt_response(session_attr["chat_history"], query)\n
session_attr["chat_history"].append((query, response))\n\n return
(\n
handler_input.response_builder\n .speak(response)\n
 .ask("Any other
questions?")\n .response\n)\n\n class
CatchAllExceptionHandler(AbstractExceptionHandler):\n """Generic
error handling to capture any syntax or routing errors."""\n def
can_handle(self, handler_input, exception):\n # type: (HandlerInput,
Exception) -> bool\n return True\n\n def handle(self,
handler_input, exception):\n # type: (HandlerInput, Exception) ->
Response\n logger.error(exception, exc_info=True)\n\n
speak_output = "Sorry, I had trouble doing what you asked. Please try
again."\n\n return (\n
handler_input.response_builder\n .speak(speak_output)\n
 .ask(speak_output)\n .respons", "pub_date": null, "extra":
null]], "original_query": null}, "command": "mclick", "args": "[0, 1, 8, 16]",
"status": "finished", "is_complete": true, "message_type": null, "model_slug": "gpt-
4o", "default_model_slug": "gpt-4o", "parent_id": "d67606d4-7173-434b-a8cc-
9e14bca72f93", "request_id": "8d1fb9c0887abadf-MXP", "timestamp_": "absolute",
"recipient": "all", "channel": null}, "parent": "d67606d4-7173-434b-a8cc-9e14bca72f93",
"children": [{"id": "5b2e0d3a-84a3-4e2d-af97-a4bc13f12f95"}], "5b2e0d3a-84a3-4e2d-af97-
a4bc13f12f95": {"id": "5b2e0d3a-84a3-4e2d-af97-a4bc13f12f95", "message": {"id":
"5b2e0d3a-84a3-4e2d-af97-a4bc13f12f95", "author": {"role": "tool", "name": "browser",
"metadata": {}}, "create_time": 1728826675.305175, "update_time": null, "content":
{"content_type": "tether_quote", "url": "https://github.com/Joao208/alexa-chatgpt",
"domain": "github.com", "text": "
 * \u301012\u2020 GitHub Skills
\u2020skills.github.com\u3011\n
 * \u301013\u2020 Blog \u2020github.blog\u3011\n\n
 * Solutions \n\nBy size\n
 * \u301014\u2020 Enterprise \u3011\n
 * \u301015\u2020 Teams \u3011\n
 * \u301016\u2020 Startups \u3011\n\nBy industry\n
 * \u301017\u2020 Healthcare \u3011\n
 * \u301018\u2020 Financial services \u3011\n
 * \u301019\u2020 Manufacturing \u3011\n\nBy use case\n
 * \u301020\u2020 CI/CD &
Automation \u3011\n
 * \u301021\u2020 DevOps \u3011\n
 * \u301022\u2020 DevSecOps
\u3011\n\n
 * Resources \n\nTopics\n
 * \u301023\u2020 AI \u3011\n
 * \u301024\u2020 DevOps \u3011\n
 * \u301025\u2020 Security \u3011\n
 * \u301026\u2020 Software Development \u3011\n
 * \u301027\u2020 View all
\u3011\n\nExplore\n
 * \u301028\u2020 Learning Pathways
\u2020resources.github.com\u3011\n
 * \u301029\u2020 White papers, Ebooks, Webinars

```



\u2020resources.github.com\u3011\n \* \u301030\u2020 Customer Stories \u3011\n \*  
\u301031\u2020 Partners \u2020partner.github.com\u3011\n\n \* Open Source \n\n \*  
\u301032\u2020 GitHub Sponsors Fund open source developers \u3011\n\n \*  
\u301033\u2020 The README Project GitHub community articles \u3011\n\nRepositories\n\* \u301034\u2020 Topics \u3011\n \* \u301035\u2020 Trending \u3011\n \*  
\u301036\u2020 Collections \u3011\n\n \* Enterprise \n\n \* \u301014\u2020  
Enterprise platform AI-powered developer platform \u3011\n\nAvailable add-ons\n \*  
\u301037\u2020 Advanced Security Enterprise-grade security features \u3011\n \*  
\u301038\u2020 GitHub Copilot Enterprise-grade AI features \u3011\n \*  
\u301039\u2020 Premium Support Enterprise-grade 24/7 support \u3011\n\n \*  
\u301040\u2020Pricing\u3011 \n\nSearch or jump to...\n\n# Search code, repositories,  
users, issues, pull requests...\n\nSearch \n\nClear \n\n\u301041\u2020 Search syntax  
tips \u2020docs.github.com\u3011 \n\n# Provide feedback \n\nWe read every piece of  
feedback, and take your input very seriously.\n\nInclude my email address so I can be  
contacted\n\nCancel Submit feedback \n\n# Saved searches \n\n## Use saved searches  
to filter your results more quickly\n\nName\n\nQuery\n\nTo see all available  
qualifiers, see our \u301041\u2020documentation\u2020docs.github.com\u3011. \n\nCancel  
Create saved search \n\n\u30101\u2020 Sign in \u3011 \n\n\u301042\u2020 Sign up \u3011  
Resetting focus\n\nYou signed in with another tab or window. Reload to refresh your  
session. You signed out in another tab or window. Reload to refresh your session. You  
switched accounts on another tab or window. Reload to refresh your session. Dismiss  
alert\n\n{{ message }}\n\n\u301043\u2020 Joao208 \u3011 / \u301044\u2020alexa-  
chatgpt\u3011 Public\n\n \* \u301045\u2020 Notifications \u3011 You must be signed in  
to change notification settings\n \* \u301045\u2020 Fork 48 \u3011 \n \*  
\u301045\u2020 Star 190 \u3011\n\nThis is an Alexa bot developed with the help of  
OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa,  
answering questions and performing tasks based on its language training.  
\n\n\u301046\u2020bit.ly/3wukdiz\u2020bit.ly\u3011 \n\n\u301047\u2020 190 stars \u3011  
\u301048\u2020 48 forks \u3011 \u301049\u2020 Branches \u3011 \u301050\u2020 Tags  
\u3011 \u301051\u2020 Activity \u3011 \n\n\u301045\u2020 Star \u3011\n\n\u301045\u2020  
Notifications \u3011 You must be signed in to change notification settings\n\n \*  
\u301044\u2020 Code \u3011\n \* \u301052\u2020 Issues 4 \u3011\n \* \u301053\u2020  
Pull requests 0 \u3011\n \* \u301054\u2020 Actions \u3011\n \* \u301055\u2020 Projects  
0 \u3011\n \* \u301056\u2020 Security \u3011\n \* \u301057\u2020 Insights \u3011\n\nAdditional navigation options \n\n \* \u301044\u2020 Code \u3011 \n \* \u301052\u2020  
Issues \u3011 \n \* \u301053\u2020 Pull requests \u3011 \n \* \u301054\u2020 Actions  
\u3011 \n \* \u301055\u2020 Projects \u3011 \n \* \u301056\u2020 Security \u3011 \n \*  
\u301057\u2020 Insights \u3011 \n\n# Joao208/alexa-chatgpt\n\nThis commit does not  
belong to any branch on this repository, and may belong to a fork outside of the  
repository.\n\n\u00a0main\n\n\u301049\u2020Branches\u3011\u301050\u2020Tags\u3011\n\nGo  
to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message  
\n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301058\u2020\u202010  
Commits\u3011  
\n\n\u301059\u2020.vscode\u3011\n\n\u301059\u2020.vscode\u3011\n\n\u301060\u2020.gitig  
nore\u3011\n\n\u301060\u2020.gitignore\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301062\u2020index.js\u3011\n\n\u301062\u2020index.js\u3011  
11\n\n\u301063\u2020local-debugger.js\u3011\n\n\u301063\u2020local-  
debugger.js\u3011\n\n\u301064\u2020package.json\u3011\n\n\u301064\u2020package.json\u3011  
011\n\n\u301065\u2020util.js\u3011\n\n\u301065\u2020util.js\u3011\n\n\u301066\u2020yar

lock\u0301\u0301\u0301\n\nlock\u0301\u0301\u0301\n\nView all files \n\n## Repository files navigation \n\n \* README \n\n# Alexa Skill with ChatGPT\n\nThis repository contains an example of how to use OpenAI's ChatGPT language model to create an Alexa Skill.\n\n## How it works\n\nThis code connects to the OpenAI ChatGPT API and sends Alexa's questions to the model, which processes them and returns the answers. Then, these responses are transmitted back to Alexa and presented to the user.\n\n## Configuration\n\nYou will need to obtain an API key from OpenAI to use the ChatGPT model. More information on how to do this can be found in the OpenAI API documentation.\n\nOnce you have your API key, simply insert it into the index.js file in the designated location.\n\n## Running the example\n\nThis example was developed using Amazon's ASK CLI. To run it, you will need to install the ASK CLI and configure your Amazon Developer account.\n\nOnce configured, simply run the following commands in the terminal:\n\nask init\nask deploy\n\nThis will deploy your Skill to your Amazon Developer account and make it available for use with Alexa.\n\n## Final considerations\n\nThis is just a basic example of how to use ChatGPT with Alexa. You can expand this implementation to create more complex and personalized skills. Enjoy!\n\n## About\n\nThis is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training.

\n\nbit.ly/3wUKdIz\n\n Topics\n\njavascript \n\n chat \n\n alexa \n\n axios \n\n chat-gpt \n\n Resources\n\nReadme\n\nActivity \n\n Stars\n\n190 stars \n\n Watchers\n\n7 watching \n\n Forks\n\n48 forks \n\n Report repository \n\n \n\n Releases \n\nNo releases published\n\n Packages 0 \n\n packages published \n\n Contributors 2 \n\n \* \n\n \"title\": \"GitHub - Joao208/alexa-chatgpt: This is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training.\" , \"tether\_id\": null}, \"status\": \"finished\_successfully\", \"end\_turn\": null, \"weight\": 0.0, \"metadata\": {\"\_cite\_metadata\": {\"citation\_format\": {\"name\": \"tether Og\"}, \"metadata\_list\": [{\"type\": \"webpage\", \"title\": \"GitHub - Joao208/alexa-chatgpt: This is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training.\" , \"url\": \"https://github.com/Joao208/alexa-chatgpt\", \"text\": \" \* \n\n GitHub Skills \n\n Blog \n\n Solutions \n\n By size \n\n Enterprise \n\n Teams \n\n Startups \n\n By industry \n\n Healthcare \n\n Financial services \n\n Manufacturing \n\n By use case \n\n CI/CD & Automation \n\n DevOps \n\n DevSecOps \n\n Resources \n\n Topics \n\n AI \n\n DevOps \n\n Security \n\n Software Development \n\n View all \n\n Explore \n\n Learning Pathways \n\n resources.github.com \n\n White papers, Ebooks, Webinars \n\n Customer Stories \n\n

\u301031\u2020 Partners \u2020partner.github.com\u3011\n\n \* Open Source \n\n \*  
 \u301032\u2020 GitHub Sponsors Fund open source developers \u3011\n\n \*  
 \u301033\u2020 The README Project GitHub community articles \u3011\n\nRepositories\n
 \* \u301034\u2020 Topics \u3011\n \* \u301035\u2020 Trending \u3011\n \*  
 \u301036\u2020 Collections \u3011\n\n \* Enterprise \n\n \* \u301014\u2020  
 Enterprise platform AI-powered developer platform \u3011\n\nAvailable add-ons\n \*  
 \u301037\u2020 Advanced Security Enterprise-grade security features \u3011\n \*  
 \u301038\u2020 GitHub Copilot Enterprise-grade AI features \u3011\n \*  
 \u301039\u2020 Premium Support Enterprise-grade 24/7 support \u3011\n\n \*  
 \u301040\u2020Pricing\u3011 \n\nSearch or jump to...\n\n# Search code, repositories,  
 users, issues, pull requests...\n\nSearch \n\nClear \n\n\u301041\u2020 Search syntax  
 tips \u2020docs.github.com\u3011 \n\n# Provide feedback \n\nWe read every piece of  
 feedback, and take your input very seriously.\n\nInclude my email address so I can be  
 contacted\n\nCancel Submit feedback \n\n# Saved searches \n\n## Use saved searches  
 to filter your results more quickly\n\nName\n\nQuery\n\nTo see all available  
 qualifiers, see our \u301041\u2020documentation\u2020docs.github.com\u3011. \n\nCancel  
 Create saved search \n\n\u30101\u2020 Sign in \u3011 \n\n\u301042\u2020 Sign up \u3011  
 Resetting focus\n\nYou signed in with another tab or window. Reload to refresh your  
 session. You signed out in another tab or window. Reload to refresh your session. You  
 switched accounts on another tab or window. Reload to refresh your session. Dismiss  
 alert\n\n{{ message }}\n\n\u301043\u2020 Joao208 \u3011 / \u301044\u2020alexa-  
 chatgpt\u3011 Public\n\n \* \u301045\u2020 Notifications \u3011 You must be signed in  
 to change notification settings\n \* \u301045\u2020 Fork 48 \u3011 \n \*  
 \u301045\u2020 Star 190 \u3011\n\nThis is an Alexa bot developed with the help of  
 OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa,  
 answering questions and performing tasks based on its language training.  
 \n\n\u301046\u2020bit.ly/3wukdiz\u2020bit.ly\u3011 \n\n\u301047\u2020 190 stars \u3011  
 \u301048\u2020 48 forks \u3011 \u301049\u2020 Branches \u3011 \u301050\u2020 Tags  
 \u3011 \u301051\u2020 Activity \u3011 \n\n\u301045\u2020 Star \u3011\n\n\u301045\u2020  
 Notifications \u3011 You must be signed in to change notification settings\n\n \*  
 \u301044\u2020 Code \u3011\n \* \u301052\u2020 Issues 4 \u3011\n \* \u301053\u2020  
 Pull requests 0 \u3011\n \* \u301054\u2020 Actions \u3011\n \* \u301055\u2020 Projects  
 0 \u3011\n \* \u301056\u2020 Security \u3011\n \* \u301057\u2020 Insights \u3011\n\n  
 Additional navigation options \n\n \* \u301044\u2020 Code \u3011 \n \* \u301052\u2020  
 Issues \u3011 \n \* \u301053\u2020 Pull requests \u3011 \n \* \u301054\u2020 Actions  
 \u3011 \n \* \u301055\u2020 Projects \u3011 \n \* \u301056\u2020 Security \u3011 \n \*  
 \u301057\u2020 Insights \u3011 \n\n# Joao208/alexa-chatgpt\n\nThis commit does not  
 belong to any branch on this repository, and may belong to a fork outside of the  
 repository.\n\nu00a0main\n\n\u301049\u2020Branches\u3011\u301050\u2020Tags\u3011\n\nG  
 o to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message  
 \n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301058\u2020\u202010  
 Commits\u3011  
 \n\n\u301059\u2020 vscode\u3011\n\n\u301059\u2020 vscode\u3011\n\n\u301060\u2020.gitignore\u3011\n\n\u301060\u2020.gitignore\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301062\u2020index.js\u3011\n\n\u301062\u2020index.js\u3011  
 \u3011\n\n\u301063\u2020local-debugger.js\u3011\n\n\u301063\u2020local-  
 debugger.js\u3011\n\n\u301064\u2020package.json\u3011\n\n\u301064\u2020package.json\u3011  
 \u3011\n\n\u301065\u2020util.js\u3011\n\n\u301065\u2020util.js\u3011\n\n\u301066\u2020yarn  
 .lock\u3011\n\n\u301066\u2020yarn.lock\u3011\n\nView all files \n\n## Repository

```

file navigation \n\n * README \n\n# AlexaSkill with ChatGPT\n\nThis repository
contains an example of how to use OpenAI's ChatGPT language model to create an
Alexa Skill.\n\n## How it works\n\nThis code connects to the OpenAI ChatGPT API and
sends Alexa's questions to the model, which processes them and returns the
answers. Then, these responses are transmitted back to Alexa and presented to the
user.\n\n## Configuration\n\nYou will need to obtain an API key from OpenAI to use the
ChatGPT model. More information on how to do this can be found in the OpenAI API
documentation.\n\nOnce you have your API key, simply insert it into the index.js file
in the designated location.\n\n## Running the example\n\nThis example was developed
using Amazon's ASK CLI. To run it, you will need to install the ASK CLI and
configure your Amazon Developer account.\n\nOnce configured, simply run the following
commands in the terminal:\n\n ask init\n ask deploy\n\nThis will deploy your
Skill to your Amazon Developer account and make it available for use with Alexa.\n\n##
Final considerations\n\nThis is just a basic example of how to use ChatGPT with Alexa.
You can expand this implementation to create more complex and personalized skills.
Enjoy!\n\n## About\n\nThis is an Alexa bot developed with the help of OpenAI's
ChatGPT language model. With it, you can converse naturally with Alexa, answering
questions and performing tasks based on its language training.
\n\n\u030101046\u0301020bit.ly/3wUKdIz\u0301020bit.ly\u0301011 \n\n#### Topics\n\n\u030101067\u0301020
javascript \u0301011 \u030101068\u0301020 chat \u0301011 \u030101069\u0301020 alexa \u0301011
\u030101070\u0301020 axios \u0301011 \u030101071\u0301020 chat-gpt \u0301011 \n\n#### Resources\n\nReadme
\n\n\u030101051\u0301020 Activity \u0301011 \n\n#### Stars\n\n\u030101047\u0301020 190 stars \u0301011
\n\n#### Watchers\n\n\u030101072\u0301020 7 watching \u0301011 \n\n#### Forks\n\n\u030101048\u0301020
48 forks \u0301011 \n\n\u030101073\u0301020 Report repository \u0301011 \n\n## \u030101074\u0301020
Releases \u0301011\n\nNo releases published\n\n## \u030101075\u0301020 Packages 0 \u0301011\n\nNo
packages published \n\n## \u030101076\u0301020 Contributors 2 \u0301011\n\n *
\u030101043\u0301020 \u0301011 \u03010104", "pub_date": null, "extra": null}], "original_query":
null}, "command": "mclick", "args": "[[0, 1, 8, 16]]", "status": "finished",
"is_complete": true, "message_type": null, "model_slug": "gpt-4o",
"default_model_slug": "gpt-4o", "parent_id": "6036flab-60a7-46d1-9199-2754a3bfcf39",
"request_id": "8d1fb9c0887abadf-MXP", "timestamp": "absolute", "recipient": "all",
"channel": null}, "parent": "6036flab-60a7-46d1-9199-2754a3bfcf39", "children":
["f4dfc757-99ea-4881-bd4b-bc2bdae243a8"]}, "f4dfc757-99ea-4881-bd4b-bc2bdae243a8":
{"id": "f4dfc757-99ea-4881-bd4b-bc2bdae243a8", "message": {"id": "f4dfc757-99ea-4881-
bd4b-bc2bdae243a8", "author": {"role": "tool", "name": "browser", "metadata": {}},
"create_time": 1728826675.305252, "update_time": null, "content": {"content_type":
"tether_quote", "url": "https://github.com/bsab/Alexa-AI-Skill", "domain":
"github.com", "text": " * \u030101053\u0301020 Pull requests 0 \u0301011\n * \u030101054\u0301020
Actions \u0301011\n * \u030101055\u0301020 Projects 0 \u0301011\n * \u030101056\u0301020 Security
\u0301011\n * \u030101057\u0301020 Insights \u0301011\n\n Additional navigation options \n\n *
\u030101044\u0301020 Code \u0301011 \n * \u030101052\u0301020 Issues \u0301011 \n * \u030101053\u0301020
Pull requests \u0301011 \n * \u030101054\u0301020 Actions \u0301011 \n * \u030101055\u0301020 Projects
\u0301011 \n * \u030101056\u0301020 Security \u0301011 \n * \u030101057\u0301020 Insights \u0301011 \n\n#
bsab/Alexa-AI-Skill\n\nThis commit does not belong to any branch on this repository,
and may belong to a fork outside of the
repository. \n\n\u030100a0main\n\n\u030101049\u0301020Branches\u0301011\u030101050\u0301020Tags\u0301011\n\nG
o to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message
\n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u030101058\u03010205
Commits\u0301011

```

\n\n\u301059\u2020images\u3011\n\n\u301059\u2020images\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301061\u2020models\u3011\n\n\u301061\u2020models\u3011\n\n\u301062\u2020.gitignore\u3011\n\n\u301062\u2020.gitignore\u3011\n\n\u301046\u2020LICENSE\u3011\n\n\u301046\u2020LICENSE\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301064\u2020ask-resources.json\u3011\n\n\u301064\u2020ask-resources.json\u3011\n\n\u301065\u2020json\_editor.json\u3011\n\n\u301065\u2020json\_editor.json\u3011\n\n\u301066\u2020skill.json\u3011\n\n\u301066\u2020skill.json\u3011\n\nView all files \n\n## Repository files navigation \n\n \* README \n \* MIT license \n\n# Alexa AI\n\n[Image 0: License: MIT]\n\nAzure OpenAI GPT-4.0 will be integrated, and the project is Alexa AI: Boost your Alexa by enhancing its responses. This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the model.\n\n[Image 1]\n\n## Prerequisites\n\n\* An \u301067\u2020Amazon Developer account\u2020developer.amazon.com\u3011\n\n\* An \u301068\u2020OpenAI API key\u2020platform.openai.com\u3011\n\n## Step-by-step tutorial\n\n### 1. \n\nLog in to your Amazon Developer account and navigate to the \u301069\u2020Alexa Developer Console\u2020developer.amazon.com\u3011.\n\n### 2. \n\nClick on "Create Skill" and name the skill "Chat". Choose the primary locale according to your language.\n\n[Image 2: name your skill]\n\n### 3. \n\nChoose "Other" and "Custom" for the model.\n\n[Image 3: type of experience]\n\n[Image 4: choose a model]\n\n### 4. \n\nChoose "Alexa-hosted (Python)" for the backend resources.\n\n[Image 5: hosting services]\n\n### 5. \n\nYou now have two options:\n\n\* Click on "Import Skill", paste the link of this repository (\u301070\u2020https://github.com/k4l1sh/alexa-gpt.git\u3011) and click on "Import".\n\n[Image 6: template]\n\nOr if you want to create the skill manually\n\n\* Select "Start from Scratch" and click on "Create Skill".\n\n[Image 7: template]\n\n### 6. \n\nIn the "Build" section, navigate to the "JSON Editor" tab.\n\n### 7. \n\nIf you have directly imported the skill from this repository, just change the "invocationName" to "chat" or another preferred word for activation and proceed to step 12.\n\nHowever, if you chose to manually create the skill, replace the existing JSON content with the \u301065\u2020provided JSON content\u3011:\n\n
 {\n
 "interactionModel": {\n
 "languageModel": {\n
 "invocationName": "chat",\n
 "intents": [\n
 {\n
 "name": "GptQueryIntent",\n
 "slots": [\n
 {\n
 "name": "query",\n
 "type": "text",\n
 "AMAZON.Person": {\n
 "samples": [\n
 {\n
 "name": "AMAZON.CancelIntent",\n
 "samples": [\n
 {\n
 "name": "AMAZON.HelpIntent",\n
 "samples": [\n
 {\n
 "name": "AMAZON.StopIntent",\n
 "samples": [\n
 {\n
 "name": "AMAZON.NavigateHomeIntent",\n
 "samples": [\n

```

"types": []\n }\n }\n\n[Image 8:

json_editor]\n\n### 8.\n\nSave the model and click on "Build Model".\n\n###

9.\n\nGo to "Code" section and add "openai" to requirements.txt.

Your requirements.txt should look like this:\n\n ask-sdk-core==1.11.0\n boto3==1.9.216\n requests==2.20.0\n\n### 10.\n\nCreate an OpenAI API key on the

\u0026#x002F;1068\u0026#x002F;API keys page\u0026#x002F;platform.openai.com\u0026#x002F;11 by clicking "+ Create

new secret key".\n\n### 11.\n\nReplace your lambda_functions.py file with the

\u0026#x002F;1071\u0026#x002F;provided lambda_function.py\u0026#x002F;11.\n\n from

ask_sdk_core.dispatch_components import AbstractExceptionHandler\n from

ask_sdk_core.dispatch_components import AbstractRequestHandler\n from

ask_sdk_core.skill_builder import SkillBuilder\n from ask_sdk_core.handler_input

import HandlerInput\n from ask_sdk_model import Response\n import

ask_sdk_core.utils as ask_utils\n import requests\n import logging\n import

json\n\n # Set your OpenAI API key\n api_key = "YOUR_API_KEY"\n\n logger = logging.getLogger(__name__)\n logger.setLevel(logging.INFO)\n\n class

LaunchRequestHandler(AbstractRequestHandler):\n """Handler for

Skill Launch."""\n def can_handle(self, handler_input):\n\n # type: (HandlerInput) -> bool\n return

ask_utils.is_request_type("LaunchRequest")(handler_input)\n\n def

handle(self, handler_input):\n # type: (HandlerInput) -> Response\n speak_output = "Chat G.P.T. mode activated"\n\n session_attr =

handler_input.attributes_manager.session_attributes\n session_attr["chat_history"] = []\n\n return (\n handler_input.response_builder\n .speak(speak_output)\n .response\n)\n\n class

GptQueryIntentHandler(AbstractRequestHandler):\n """Handler for

Gpt Query Intent."""\n def can_handle(self, handler_input):\n\n # type: (HandlerInput) -> bool\n return

ask_utils.is_intent_name("GptQueryIntent")(handler_input)\n\n def

handle(self, handler_input):\n # type: (HandlerInput) -> Response\n query =

handler_input.request_envelope.request.intent.slots["query"].value\n\n session_attr = handler_input.attributes_manager.session_attributes\n if

"chat_history" not in session_attr:\n session_attr["chat_history"] = []\n\n response =

generate_gpt_response(session_attr["chat_history"], query)\n\n session_attr["chat_history"].append((query, response))\n\n return

(\n handler_input.response_builder\n .speak(response)\n .ask("Any other

questions?")\n .response\n)\n\n class

CatchAllExceptionHandler(AbstractExceptionHandler):\n """Generic

error handling to capture any syntax or routing errors."""\n def

can_handle(self, handler_input, exception):\n # type: (HandlerInput,

Exception) -> bool\n return True\n\n def handle(self,

handler_input, "title": "GitHub - bsab/Alexa-AI-Skill", "tether_id": null}, "status":

"finished_successfully", "end_turn": null, "weight": 0.0, "metadata":

{"_cite_metadata": {"citation_format": {"name": "tether_og"}, "metadata_list":

["type": "webpage", "title": "GitHub - bsab/Alexa-AI-Skill", "url":

```

"https://github.com/bsab/Alexa-AI-Skill", "text": " \* \u301053\u2020 Pull requests 0 \u3011\n \* \u301054\u2020 Actions \u3011\n \* \u301055\u2020 Projects 0 \u3011\n \* \u301056\u2020 Security \u3011\n \* \u301057\u2020 Insights \u3011\n\n Additional navigation options \n\n \* \u301044\u2020 Code \u3011\n \* \u301052\u2020 Issues \u3011\n \* \u301053\u2020 Pull requests \u3011\n \* \u301054\u2020 Actions \u3011\n \* \u301055\u2020 Projects \u3011\n \* \u301056\u2020 Security \u3011\n \* \u301057\u2020 Insights \u3011\n\n# bsab/Alexa-AI-Skill\n\nThis commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.\n\n\u00a0main\n\n\u301049\u2020Branches\u3011\u301050\u2020Tags\u3011\n\nG o to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message \n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301058\u2020205 Commits\u3011

\n\n\u301059\u2020images\u3011\n\n\u301059\u2020images\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301060\u2020lambda\u3011\n\n\u301061\u2020models\u3011\n\n\u301061\u2020mode ls\u3011\n\n\u301062\u2020.gitignore\u3011\n\n\u301062\u2020.gitignore\u3011\n\n\u301046\u2020LICENSE\u3011\n\n\u301046\u2020LICENSE\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301063\u2020README.md\u3011\n\n\u301064\u2020ask-resources.json\u3011\n\n\u301064\u2020ask-resources.json\u3011\n\n\u301065\u2020json\_editor.json\u3011\n\n\u301065\u2020json\_editor.json\u3011\n\n\u301066\u2020skill.json\u3011\n\n\u301066\u2020skill.json\u3011\n\nView all files \n\n## Repository files navigation \n\n \* README \n \* MIT license \n\n# Alexa AI\n\n[Image 0: License: MIT]\n\nAzure OpenAI GPT-4.0 will be integrated, and the project is Alexa AI: Boost your Alexa by enhancing its responses. This repository contains a tutorial on how to create a simple Alexa skill that uses the OpenAI API to generate responses from the model.\n\n[Image 1]\n\n## Prerequisites\n\n \* An \u301067\u2020Amazon Developer account\u2020developer.amazon.com\u3011\n \* An \u301068\u2020OpenAI API key\u2020platform.openai.com\u3011\n\n## Step-by-step tutorial\n\n### 1. \n\nLog in to your Amazon Developer account and navigate to the \u301069\u2020Alexa Developer Console\u2020developer.amazon.com\u3011.\n\n### 2. \n\nClick on "Create Skill"; and name the skill "Chat";. Choose the primary locale according to your language.\n\n[Image 2: name your skill]\n\n### 3. \n\nChoose "Other"; and "Custom"; for the model.\n\n[Image 3: type of experience]\n\n[Image 4: choose a model]\n\n### 4. \n\nChoose "Alexa-hosted (Python)"; for the backend resources.\n\n[Image 5: hosting services]\n\n### 5. \n\nYou now have two options:\n\n \* Click on "Import Skill";, paste the link of this repository (\u301070\u2020https://github.com/k411sh/alexa-gpt.git\u3011) and click on "Import";. [Image 6: template]\n\nOr if you want to create the skill manually\n\n \* Select "Start from Scratch"; and click on "Create Skill";\n\n[Image 7: template]\n\n### 6. \n\nIn the "Build"; section, navigate to the "JSON Editor"; tab.\n\n### 7. \n\nIf you have directly imported the skill from this repository, just change the "invocationName"; to "chat"; or another preferred word for activation and proceed to step 12.\n\nHowever, if you chose to manually create the skill, replace the existing JSON content with the \u301065\u2020provided JSON content\u3011:\n\n{\n "interactionModel": {\n "languageModel": {\n "invocationName": "chat",\n "intents": [\n {\n "name": "GptQueryIntent",\n "slots": [\n {\n "name": "query"\n }\n ]\n }\n ]\n }\n }\n}

```

 "type";
 "AMAZON. Person";\n
 "samples";: []\n
 "{query}";\n
 {\n
 "name";: "AMAZON. CancelIntent";,\n
 "samples";: []\n
 "name";: "AMAZON. HelpIntent";,\n
 "samples";: []\n
 "name";: "AMAZON. StopIntent";,\n
 "samples";: []\n
 "name";: "AMAZON. NavigateHomeIntent";,\n
 "samples";: []\n
 "types";: []\n
 }\n
 }\n
 }\n\n[Image 8:
json_editor]\n\n### 8.\n\nSave the model and click on "Build Model".\n\n###
9.\n\nGo to "Code" section and add "openai" to requirements.txt.
Your requirements.txt should look like this:\n\n ask-sdk-core==1.11.0\n
boto3==1.9.216\n requests>=2.20.0\n\n### 10.\n\nCreate an OpenAI API key on the
\u301068\u2020API keys page\u2020platform.openai.com\u3011 by clicking "+ Create
new secret key".\n\n### 11.\n\nReplace your lambda_functions.py file with the
\u301071\u2020provided lambda_function.py\u3011.\n\n from
ask_sdk_core.dispatch_components import AbstractExceptionHandler\n from
ask_sdk_core.dispatch_components import AbstractRequestHandler\n from
ask_sdk_core.skill_builder import SkillBuilder\n from ask_sdk_core.handler_input
import HandlerInput\n from ask_sdk_model import Response\n import
ask_sdk_core.utils as ask_utils\n import requests\n import logging\n import
json\n\n # Set your OpenAI API key\n api_key = "YOUR_API_KEY"\n\n logger = logging.getLogger(__name__)\n logger.setLevel(logging.INFO)\n\n class
LaunchRequestHandler(AbstractRequestHandler):\n """Handler for
Skill Launch."""\n def can_handle(self, handler_input):\n
type: (HandlerInput) -> bool\n return
ask_utils.is_request_type("LaunchRequest")(handler_input)\n\n def
handle(self, handler_input):\n # type: (HandlerInput) -> Response\n
speak_output = "Chat G.P.T. mode activated"\n\n session_attr =
handler_input.attributes_manager.session_attributes\n
session_attr["chat_history"] = []\n\n return (\n
handler_input.response_builder\n .speak(speak_output)\n
 .ask(speak_output)\n .response\n)\n\n class
GptQueryIntentHandler(AbstractRequestHandler):\n """Handler for
Gpt Query Intent."""\n def can_handle(self, handler_input):\n
type: (HandlerInput) -> bool\n return
ask_utils.is_intent_name("GptQueryIntent")(handler_input)\n\n def
handle(self, handler_input):\n # type: (HandlerInput) -> Response\n
query =
handler_input.request_envelope.request.intent.slots["query"].value\n\n
session_attr = handler_input.attributes_manager.session_attributes\n if
"chat_history" not in session_attr:\n
session_attr["chat_history"] = []\n response =
generate_gpt_response(session_attr["chat_history"], query)\n
session_attr["chat_history"].append((query, response))\n\n return

```



```
(\n
handler_input.response_builder\n
 .ask("Any other
questions?")\n
 .response\n
)\n\n
class
CatchAllExceptionHandler(AbstractExceptionHandler):\n
 """Generic
error handling to capture any syntax or routing errors."""\n
 def
can_handle(self, handler_input, exception):\n
 # type: (HandlerInput,
Exception) -> bool\n
 return True\n\n
 def handle(self,
handler_input, *, pub_date: null, extra: null}], original_query: null}, command":
"mclick", args": [[0, 1, 8, 16]], status": "finished", is_complete": true,
"message_type": null, model_slug": "gpt-4o", default_model_slug": "gpt-4o",
"parent_id": "5b2e0d3a-84a3-4e2d-af97-a4bc13f12f95", request_id": "8d1fb9c0887abadf-
MXP", timestamp_": "absolute", recipient": "all", channel": null}, parent":
"5b2e0d3a-84a3-4e2d-af97-a4bc13f12f95", children": ["cc71f4b3-541b-40e9-9a3f-
e386a87c95d6"], "cc71f4b3-541b-40e9-9a3f-e386a87c95d6": {"id": "cc71f4b3-541b-40e9-
9a3f-e386a87c95d6", message": {"id": "cc71f4b3-541b-40e9-9a3f-e386a87c95d6", author":
{"role": "assistant", name": null, metadata": {}}, create_time": 1728826675.305326,
"update_time": null, content": {"content_type": "text", parts": ["Yes, there are
open-source repositories that can help you integrate OpenAI with Alexa quickly. Here
are a couple of options:\n\n1. Joao208/alexa-chatgpt: This project connects Alexa
with OpenAI's ChatGPT to allow natural conversations. You can set it up by
modifying the existing code to include your OpenAI API key. The repository provides
all the necessary components to get started with creating an Alexa Skill that can
leverage OpenAI's responses. You can find more about this project [here on
GitHub] (https://github.com/Joao208/alexa-chatgpt)\u301089\u2020source\u3011.\n\n2.
bsab/Alexa-AI-Skill: This repository demonstrates how to build an Alexa skill that
uses OpenAI's GPT models. It includes step-by-step instructions on setting up the
skill in the Alexa Developer Console and integrating OpenAI API calls within the
lambda function. You just need to plug in your API key and follow the setup process.
Check out the repository [here] (https://github.com/k4llsh/alexa-
gpt)\u301090\u2020source\u3011.\n\nBoth of these solutions provide templates where you
can just insert your OpenAI API key and deploy the Alexa skill using the Amazon Skills
Kit (ASK). They are customizable to suit more complex use cases as well."}], "status":
"finished successfully", end_turn": true
, "weight": 1.0, "metadata": {"citations": [{"start_ix": 549, "end_ix": 560,
"citation_format_type": "tether_og", "metadata": {"type": "webpage", "title": "GitHub
- Joao208/alexa-chatgpt: This is an Alexa bot developed with the help of OpenAI's
ChatGPT language model. With it, you can converse naturally with Alexa, answering
questions and performing tasks based on its language training.", "url":
"https://github.com/Joao208/alexa-chatgpt", "text": "
 * \u301012\u2020 GitHub
Skills \u2020skills.github.com\u3011\n
 * \u301013\u2020 Blog
\u2020github.blog\u3011\n\n
 * Solutions \u3011\n\n
 * \u301014\u2020 Enterprise
\u3011\n
 * \u301015\u2020 Teams \u3011\n
 * \u301016\u2020 Startups \u3011\n\n
 * \u301017\u2020 Healthcare \u3011\n
 * \u301018\u2020 Financial
services \u3011\n
 * \u301019\u2020 Manufacturing \u3011\n\n
 * \u301020\u2020 CI/CD & Automation \u3011\n
 * \u301021\u2020 DevOps \u3011\n
 * \u301022\u2020 DevSecOps \u3011\n\n
 * Resources \u3011\n\n
 * \u301023\u2020 AI
\u3011\n
 * \u301024\u2020 DevOps \u3011\n
 * \u301025\u2020 Security \u3011\n
 * \u301026\u2020 Software Development \u3011\n
 * \u301027\u2020 View all
```

\u3011\n\nExplore\n \* \u301028\u2020 Learning Pathways  
\u2020resources.github.com\u3011\n \* \u301029\u2020 White papers, Ebooks, Webinars  
\u2020resources.github.com\u3011\n \* \u301030\u2020 Customer Stories \u3011\n \*  
\u301031\u2020 Partners \u2020partner.github.com\u3011\n\n \* Open Source \n\n \*  
\u301032\u2020 GitHub Sponsors Fund open source developers \u3011\n\n \*  
\u301033\u2020 The README Project GitHub community articles \u3011\n\nRepositories\n\* \u301034\u2020 Topics \u3011\n \* \u301035\u2020 Trending \u3011\n \*  
\u301036\u2020 Collections \u3011\n\n \* Enterprise \n\n \* \u301014\u2020  
Enterprise platform AI-powered developer platform \u3011\n\nAvailable add-ons\n \*  
\u301037\u2020 Advanced Security Enterprise-grade security features \u3011\n \*  
\u301038\u2020 GitHub Copilot Enterprise-grade AI features \u3011\n \*  
\u301039\u2020 Premium Support Enterprise-grade 24/7 support \u3011\n\n \*  
\u301040\u2020Pricing\u3011 \n\nSearch or jump to...\n\n# Search code, repositories,  
users, issues, pull requests...\n\nSearch \n\nClear \n\n\u301041\u2020 Search syntax  
tips \u2020docs.github.com\u3011 \n\n# Provide feedback \n\nWe read every piece of  
feedback, and take your input very seriously.\n\nInclude my email address so I can be  
contacted\n\nCancel Submit feedback \n\n# Saved searches \n\n## Use saved searches  
to filter your results more quickly\n\nName\n\nQuery\n\nTo see all available  
qualifiers, see our \u301041\u2020documentation\u2020docs.github.com\u3011. \n\nCancel  
Create saved search \n\n\u30101\u2020 Sign in \u3011 \n\n\u301042\u2020 Sign up \u3011  
Resetting focus\n\nYou signed in with another tab or window. Reload to refresh your  
session. You signed out in another tab or window. Reload to refresh your session. You  
switched accounts on another tab or window. Reload to refresh your session. Dismiss  
alert\n\n{{ message }}\n\n\u301043\u2020 Joao208 \u3011 / \u301044\u2020alexa-  
chatgpt\u3011 Public\n\n \* \u301045\u2020 Notifications \u3011 You must be signed in  
to change notification settings\n \* \u301045\u2020 Fork 48 \u3011 \n \*  
\u301045\u2020 Star 190 \u3011\n\nThis is an Alexa bot developed with the help of  
OpenAI&#x27;s ChatGPT language model. With it, you can converse naturally with Alexa,  
answering questions and performing tasks based on its language training.  
\n\n\u301046\u2020bit.ly/3wukdiz\u2020bit.ly\u3011 \n\n\u301047\u2020 190 stars \u3011  
\u301048\u2020 48 forks \u3011 \u301049\u2020 Branches \u3011 \u301050\u2020 Tags  
\u3011 \u301051\u2020 Activity \u3011 \n\n\u301045\u2020 Star \u3011\n\n\u301045\u2020  
Notifications \u3011 You must be signed in to change notification settings\n\n \*  
\u301044\u2020 Code \u3011\n \* \u301052\u2020 Issues 4 \u3011\n \* \u301053\u2020  
Pull requests 0 \u3011\n \* \u301054\u2020 Actions \u3011\n \* \u301055\u2020 Projects  
0 \u3011\n \* \u301056\u2020 Security \u3011\n \* \u301057\u2020 Insights \u3011\n\nAdditional navigation options \n\n \* \u301044\u2020 Code \u3011 \n \* \u301052\u2020  
Issues \u3011 \n \* \u301053\u2020 Pull requests \u3011 \n \* \u301054\u2020 Actions  
\u3011 \n \* \u301055\u2020 Projects \u3011 \n \* \u301056\u2020 Security \u3011 \n \*  
\u301057\u2020 Insights \u3011 \n\n# Joao208/alexa-chatgpt\n\nThis commit does not  
belong to any branch on this repository, and may belong to a fork outside of the  
repository.\n\n0main\n\n\u301049\u2020Branches\u3011\u301050\u2020Tags\u3011\n\nG  
o to file \n\nCode\n\n## Folders and files \n\nName Name \n\nLast commit message  
\n\nLast commit date \n\n## Latest commit\n\n## History\n\n\u301058\u20202010  
Commits\u3011  
\n\n\u301059\u2020.vscode\u3011\n\n\u301059\u2020.vscode\u3011\n\n\u301060\u2020.gitig  
nore\u3011\n\n\u301060\u2020.gitignore\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301061\u2020README.md\u3011\n\n\u301062\u2020index.js\u3011\n\n\u301062\u2020index.js\u3011  
11\n\n\u301063\u2020local-debugger.js\u3011\n\n\u301063\u2020local-

```

debugger.js\u0011\n\n\u001064\u0020package.json\u0011\n\n\u001064\u0020package.json\u0011\n\n\u001065\u0020util.js\u0011\n\n\n\u001065\u0020util.js\u0011\n\n\n\u001066\u0020yarn.lock\u0011\n\n\n\u001066\u0020yarn.lock\u0011\n\n\nView all files \n\n\n## Repository files navigation \n\n * README \n\n# Alexa Skill with ChatGPT\n\n\nThis repository contains an example of how to use OpenAI's ChatGPT language model to create an Alexa Skill.\n\n\n## How it works\n\n\nThis code connects to the OpenAI ChatGPT API and sends Alexa's questions to the model, which processes them and returns the answers. Then, these responses are transmitted back to Alexa and presented to the user.\n\n\n## Configuration\n\n\nYou will need to obtain an API key from OpenAI to use the ChatGPT model. More information on how to do this can be found in the OpenAI API documentation.\n\n\nOnce you have your API key, simply insert it into the index.js file in the designated location.\n\n\n## Running the example\n\n\nThis example was developed using Amazon's ASK CLI. To run it, you will need to install the ASK CLI and configure your Amazon Developer account.\n\n\nOnce configured, simply run the following commands in the terminal:\n\n\n ask init\n ask deploy\n\n\nThis will deploy your Skill to your Amazon Developer account and make it available for use with Alexa.\n\n\n## Final considerations\n\n\nThis is just a basic example of how to use ChatGPT with Alexa. You can expand this implementation to create more complex and personalized skills. Enjoy!\n\n\n## About\n\n\nThis is an Alexa bot developed with the help of OpenAI's ChatGPT language model. With it, you can converse naturally with Alexa, answering questions and performing tasks based on its language training.\n\n\n\u001046\u0020bit.ly/3wUKdIz\u0020bit.ly\u0011 \n\n\n### Topics\n\n\n\u001067\u0020javascript \u0011 \u001068\u0020chat \u0011 \u001069\u0020alexa \u0011 \u001070\u0020axios \u0011 \u001071\u0020chat-gpt \u0011\n\n\n### Resources\n\n\nReadme \u0011 \u001051\u0020Activity \u0011\n\n\n### Stars\n\n\n\u001047\u0020190 stars \u0011\n\n\n### Watchers\n\n\n\u001072\u00207 watching \u0011\n\n\n### Forks\n\n\n\u001048\u002048 forks \u0011\n\n\n\u001073\u0020Report repository \u0011\n\n\n \u001074\u0020Releases \u0011\n\n\nNo releases published\n\n\n \u001075\u0020Packages 0 \u0011\n\n\nNo packages published \n\n\n \u001076\u0020Contributors 2 \u0011\n\n * \u001043\u0020\u0011 \u00104\u0020, "pub_date": null, "extra": {"cited_message_idx": 89, "search_result_idx": null, "evidence_text": "source", "cloud_doc_url": null}}}, {"start_ix": 969, "end_ix": 980, "citation_format_type": "tether_og", "metadata": {"type": "webpage", "title": "GitHub - bsab/Alexa-AI-Skill", "url": "https://github.com/bsab/Alexa-AI-Skill", "text": " * \u001053\u0020Pull requests 0 \u0011\n * \u001054\u0020Actions \u0011\n * \u001055\u0020Projects 0 \u0011\n * \u001056\u0020Security \u0011\n * \u001057\u0020Insights \u0011\n\n Additional navigation options \n\n * \u001044\u0020Code \u0011\n * \u001052\u0020Issues \u0011\n * \u001053\u0020Pull requests \u0011\n * \u001054\u0020Actions \u0011\n * \u001055\u0020Projects \u0011\n * \u001056\u0020Security \u0011\n * \u001057\u0020Insights \u0011\n\n# bsab/Alexa-AI-Skill\n\n\nThis commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.\n\n\n\u0000main\n\n\n\u001049\u0020Branches\u0011\u001050\u0020Tags\u0011\n\n\nGo to file \n\n\nCode\n\n\n## Folders and files \n\n\nName Name \n\n\nLast commit message \n\n\nLast commit date \n\n\n## Latest commit\n\n\n## History\n\n\n\u001058\u00202020 Commits\u0011\n\n\n\u001059\u0020images\u0011\n\n\n\u001059\u0020images\u0011\n\n\n\u001060\u0020lambda\u0011\n\n\n\u001060\u0020lambda\u0011\n\n\n\u001061\u0020models\u0011\n\n\n\u001061\u0020mode ls\u0011\n\n\n\u001062\u0020.gitignore\u0011\n\n\n\u001062\u0020.gitignore\u0011\n\n\n\u001063\u0020LICENSE\u0011\n\n\n\u001063\u0020LICENSE\u0011\n\n\n\u001064\u0020LICENSE\u0011\n\n\n\u001064\u0020LICENSE\u0011\n\n\n\u001065\u0020README.md\u0011\n\n\n\u001065\u0020README.md\u0011\n\n\n\u001066\u0020index.js\u0011\n\n\n\u001066\u0020index.js\u0011\n\n\n\u001067\u0020package.json\u0011\n\n\n\u001067\u0020package.json\u0011\n\n\n\u001068\u0020util.js\u0011\n\n\n\u001068\u0020util.js\u0011\n\n\n\u001069\u0020yarn.lock\u0011\n\n\n\u001069\u0020yarn.lock\u0011\n\n\n\u001070\u0020yarn.lock\u0011\n\n\n\u001070\u0020yarn.lock\u0011\n\n\n\u001071\u0020yarn.lock\u0011\n\n\n\u001071\u0020yarn.lock\u0011\n\n\n\u001072\u0020yarn.lock\u0011\n\n\n\u001072\u0020yarn.lock\u0011\n\n\n\u001073\u0020yarn.lock\u0011\n\n\n\u001073\u0020yarn.lock\u0011\n\n\n\u001074\u0020yarn.lock\u0011\n\n\n\u001074\u0020yarn.lock\u0011\n\n\n\u001075\u0020yarn.lock\u0011\n\n\n\u001075\u0020yarn.lock\u0011\n\n\n\u001076\u0020yarn.lock\u0011\n\n\n\u001076\u0020yarn.lock\u0011\n\n\n\u001077\u0020yarn.lock\u0011\n\n\n\u001077\u0020yarn.lock\u0011\n\n\n\u001078\u0020yarn.lock\u0011\n\n\n\u001078\u0020yarn.lock\u0011\n\n\n\u001079\u0020yarn.lock\u0011\n\n\n\u001079\u0020yarn.lock\u0011\n\n\n\u001080\u0020yarn.lock\u0011\n\n\n\u001080\u0020yarn.lock\u0011\n\n\n\u001081\u0020yarn.lock\u0011\n\n\n\u001081\u0020yarn.lock\u0011\n\n\n\u001082\u0020yarn.lock\u0011\n\n\n\u001082\u0020yarn.lock\u0011\n\n\n\u001083\u0020yarn.lock\u0011\n\n\n\u001083\u0020yarn.lock\u0011\n\n\n\u001084\u0020yarn.lock\u0011\n\n\n\u001084\u0020yarn.lock\u0011\n\n\n\u001085\u0020yarn.lock\u0011\n\n\n\u001085\u0020yarn.lock\u0011\n\n\n\u001086\u0020yarn.lock\u0011\n\n\n\u001086\u0020yarn.lock\u0011\n\n\n\u001087\u0020yarn.lock\u0011\n\n\n\u001087\u0020yarn.lock\u0011\n\n\n\u001088\u0020yarn.lock\u0011\n\n\n\u001088\u0020yarn.lock\u0011\n\n\n\u001089\u0020yarn.lock\u0011\n\n\n\u001089\u0020yarn.lock\u0011\n\n\n\u001090\u0020yarn.lock\u0011\n\n\n\u001090\u0020yarn.lock\u0011\n\n\n\u001091\u0020yarn.lock\u0011\n\n\n\u001091\u0020yarn.lock\u0011\n\n\n\u001092\u0020yarn.lock\u0011\n\n\n\u001092\u0020yarn.lock\u0011\n\n\n\u001093\u0020yarn.lock\u0011\n\n\n\u001093\u0020yarn.lock\u0011\n\n\n\u001094\u0020yarn.lock\u0011\n\n\n\u001094\u0020yarn.lock\u0011\n\n\n\u001095\u0020yarn.lock\u0011\n\n\n\u001095\u0020yarn.lock\u0011\n\n\n\u001096\u0020yarn.lock\u0011\n\n\n\u001096\u0020yarn.lock\u0011\n\n\n\u001097\u0020yarn.lock\u0011\n\n\n\u001097\u0020yarn.lock\u0011\n\n\n\u001098\u0020yarn.lock\u0011\n\n\n\u001098\u0020yarn.lock\u0011\n\n\n\u001099\u0020yarn.lock\u0011\n\n\n\u001099\u0020yarn.lock\u0011\n\n\n\u001100\u0020yarn.lock\u0011\n\n\n\u001100\u0020yarn.lock\u0011\n\n\n\u001101\u0020yarn.lock\u0011\n\n\n\u001101\u0020yarn.lock\u0011\n\n\n\u001102\u0020yarn.lock\u0011\n\n\n\u001102\u0020yarn.lock\u0011\n\n\n\u001103\u0020yarn.lock\u0011\n\n\n\u001103\u0020yarn.lock\u0011\n\n\n\u001104\u0020yarn.lock\u0011\n\n\n\u001104\u0020yarn.lock\u0011\n\n\n\u001105\u0020yarn.lock\u0011\n\n\n\u001105\u0020yarn.lock\u0011\n\n\n\u001106\u0020yarn.lock\u0011\n\n\n\u001106\u0020yarn.lock\u0011\n\n\n\u001107\u0020yarn.lock\u0011\n\n\n\u001107\u0020yarn.lock\u0011\n\n\n\u001108\u0020yarn.lock\u0011\n\n\n\u001108\u0020yarn.lock\u0011\n\n\n\u001109\u0020yarn.lock\u0011\n\n\n\u001109\u0020yarn.lock\u0011\n\n\n\u001110\u0020yarn.lock\u0011\n\n\n\u001110\u0020yarn.lock\u0011\n\n\n\u001111\u0020yarn.lock\u0011\n\n\n\u001111\u0020yarn.lock\u0011\n\n\n\u001112\u0020yarn.lock\u0011\n\n\n\u001112\u0020yarn.lock\u0011\n\n\n\u001113\u0020yarn.lock\u0011\n\n\n\u001113\u0020yarn.lock\u0011\n\n\n\u001114\u0020yarn.lock\u0011\n\n\n\u001114\u0020yarn.lock\u0011\n\n\n\u001115\u0020yarn.lock\u0011\n\n\n\u00
```

```
n\n\u0301063\u03012020README.md\u030103011\n\n\u03010301064\u03012020ask-
resources.json\u030103011\n\n\u03010301064\u03012020ask-
resources.json\u030103011\n\n\u03010301065\u03012020json_editor.json\u030103011\n\n\u03010301065\u03012020json_edit
tor.json\u030103011\n\n\u03010301066\u03012020skill.json\u030103011\n\n\u03010301066\u03012020skill.json\u030103011\n\nView all files \n\n## Repository files navigation \n\n * README \n * MIT license
\n\n# Alexa AI\n\n[Image 0: License: MIT]\n\nAzure OpenAI GPT-4.0 will be integrated,
and the project is Alexa AI: Boost your Alexa by enhancing its responses. This
repository contains a tutorial on how to create a simple Alexa skill that uses the
OpenAI API to generate responses from the model.\n\n[Image 1]\n\n## Prerequisites\n\n* An \u03010301067\u03012020Amazon Developer account\u03012020developer.amazon.com\u030103011\n * An
\u03010301068\u03012020OpenAI API key\u03012020platform.openai.com\u030103011\n\n## Step-by-step
tutorial\n\n#### 1. \n\nLog in to your Amazon Developer account and navigate to the
\u03010301069\u03012020Alexa Developer Console\u03012020developer.amazon.com\u030103011.\n\n####
2. \n\nClick on "Create Skill"; and name the skill "Chat". Choose
the primary locale according to your language.\n\n[Image 2: name your skill]\n\n####
3. \n\nChoose "Other"; and "Custom"; for the model.\n\n[Image 3: type
of experience]\n\n[Image 4: choose a model]\n\n#### 4. \n\nChoose "Alexa-hosted
(Python)" for the backend resources.\n\n[Image 5: hosting services]\n\n####
5. \n\nYou now have two options:\n\n * Click on "Import Skill", paste the
link of this repository (\u03010301070\u03012020https://github.com/k4ll1sh/alexa-gpt.git\u030103011)
and click on "Import". [Image 6: template]\n\nOr if you want to create the
skill manually\n\n * Select "Start from Scratch"; and click on "Create
Skill"\n\n[Image 7: template]\n\n#### 6. \n\nIn the "Build"; section,
navigate to the "JSON Editor"; tab.\n\n#### 7. \n\nIf you have directly
imported the skill from this repository, just change the "invocationName"; to
"chat"; or another preferred word for activation and proceed to step
12.\n\nHowever, if you chose to manually create the skill, replace the existing JSON
content with the \u03010301065\u03012020provided JSON content\u030103011:\n\n{\n "interactionModel"; {\n "languageModel"; {\n "invocationName"; "chat";,\n "intents"; [\n {\n "name";\n "GptQueryIntent";,\n "slots"; [\n {\n "name"; "query";,\n "type";\n }\n],\n "AMAZON.Person";\n },\n "samples"; [\n "{query}";\n],\n {\n "name"; "AMAZON.CancelIntent";,\n "samples"; [\n "";\n],\n "name"; "AMAZON.HelpIntent";,\n "samples"; [\n "";\n],\n "name"; "AMAZON.StopIntent";,\n "samples"; [\n "";\n],\n "name"; "AMAZON.NavigateHomeIntent";,\n "samples"; [\n "";\n],\n "types"; [\n "";\n]\n }\n]\n }\n }\n}\n\n[Image 8:
json_editor]\n\n#### 8. \n\nSave the model and click on "Build Model".\n\n####
9. \n\nGo to "Code"; section and add "openai"; to requirements.txt.
Your requirements.txt should look like this:\n\n ask-sdk-core==1.11.0\n
```