Alexa In Skill Purchasing (ISP)



[This Photo](https://www.freepngimg.com/png/60724-united-money-dollar-cash-states-vector-coin) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Videos:

1. Example
2. Create an in-skill product
3. Alexa-hosted break-down of code
4. Build from scratch
5. ASK CLI version?

1. Demonstration example ISP at: https://youtu.be/h0EIVhIfD98

2. Create ISP at: https://youtu.be/c82JRithsw8

3 and 4. Walk through video at: https://youtu.be/EIp2bQJEU4A

Github repo: https://github.com/jallwork/Alexa\_ISP

**Alexa In skill purchasing (ISP)**

**Instructions on how to create an ISP and python code to use ISP.**

https://developer.amazon.com/en-US/docs/alexa/in-skill-purchase/isp-overview.html

<https://developer.amazon.com/en-US/docs/alexa/in-skill-purchase/create-isp-dev-console.html>

You can offer in-skill products with the following payment models:

**One-time purchases**: Entitlements that unlock access to features or content within a skill. One-time purchases don't expire. Examples include game expansion packs, unlocked features, extra characters, and more.

**Consumables**: Content or features that a user can purchase, deplete, and purchase again. For example, hints for a game, in-game currency, extra lives, or "day passes" for premium content.

**Subscriptions**: Offers access to premium content or features for a period of time, charged on a recurring basis until the user cancels the subscription.

We’ll use **One-time purchases**

To implement in-skill purchases in your skill:

1. Create your in-skill product(s).
2. Add support for purchasing to your skill.
3. Add code to handle the purchase requests and use your in-skill products.
4. Test your skill, then submit it for certification.
5. Create your skill and add in-skill product(s).

We'll look at a one-time purchase. You’ll pay to access extra jokes in the skill.

1. Add ISP support to your skill.

You update your interaction model to include custom intents to support user requests to:

* Buy or shop for products.
* Ask for a refund or cancel a subscription.

1. Add code.

Get a list of products that the user is eligible to purchase. Your code then passes the purchase request to Amazon's purchase flow.

Resume the skill correctly after the purchase flow completes. e.g. update the user's inventory of purchased items and keep track of those items as they are used up.

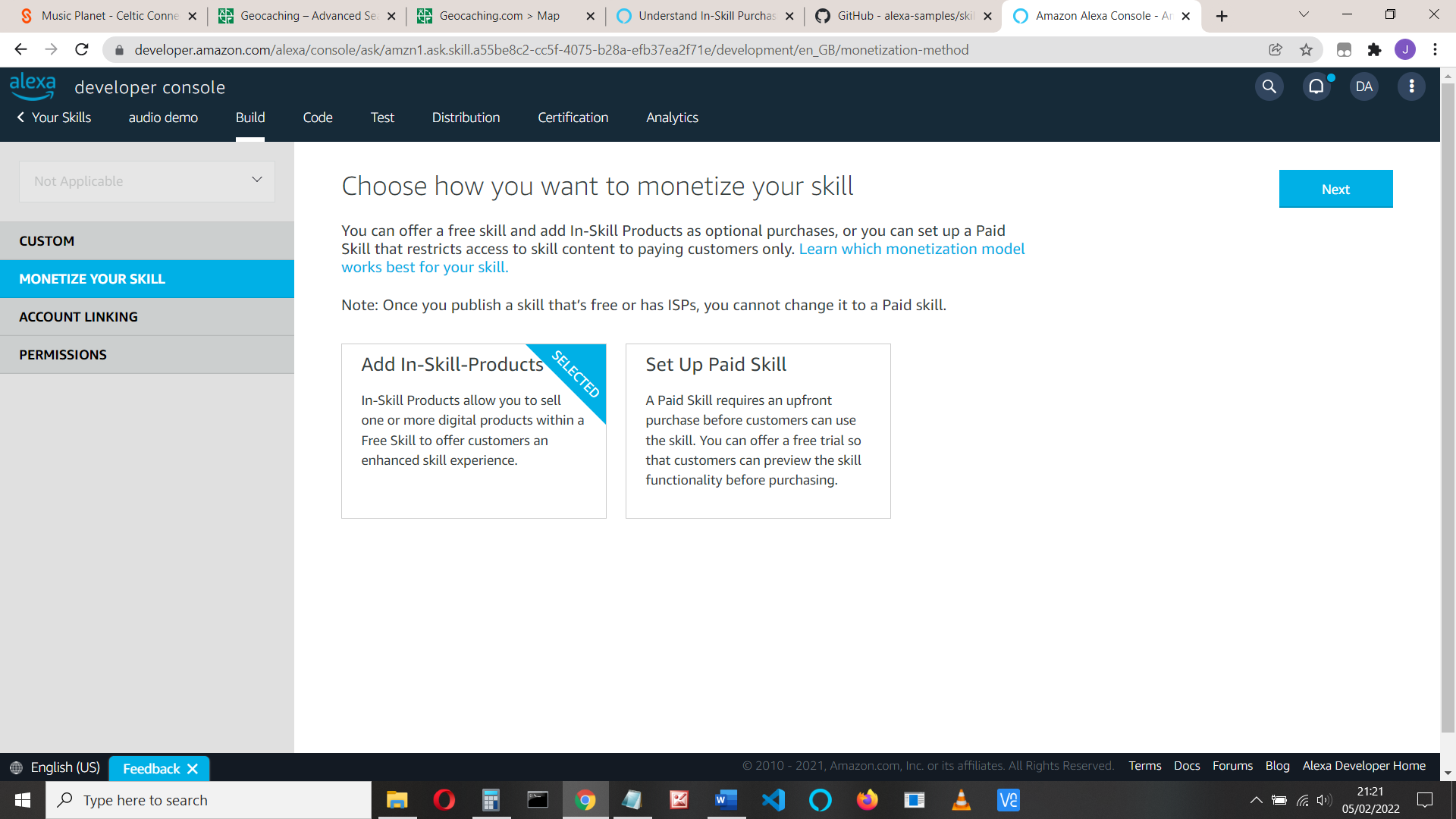
1. Test your skill and get your skill certified.

When your skill is live, you can view purchase metrics as well as earning and payments for your products.

**Create your Alexa Hosted skill** (Custom > Alexa-hosted > From scratch)

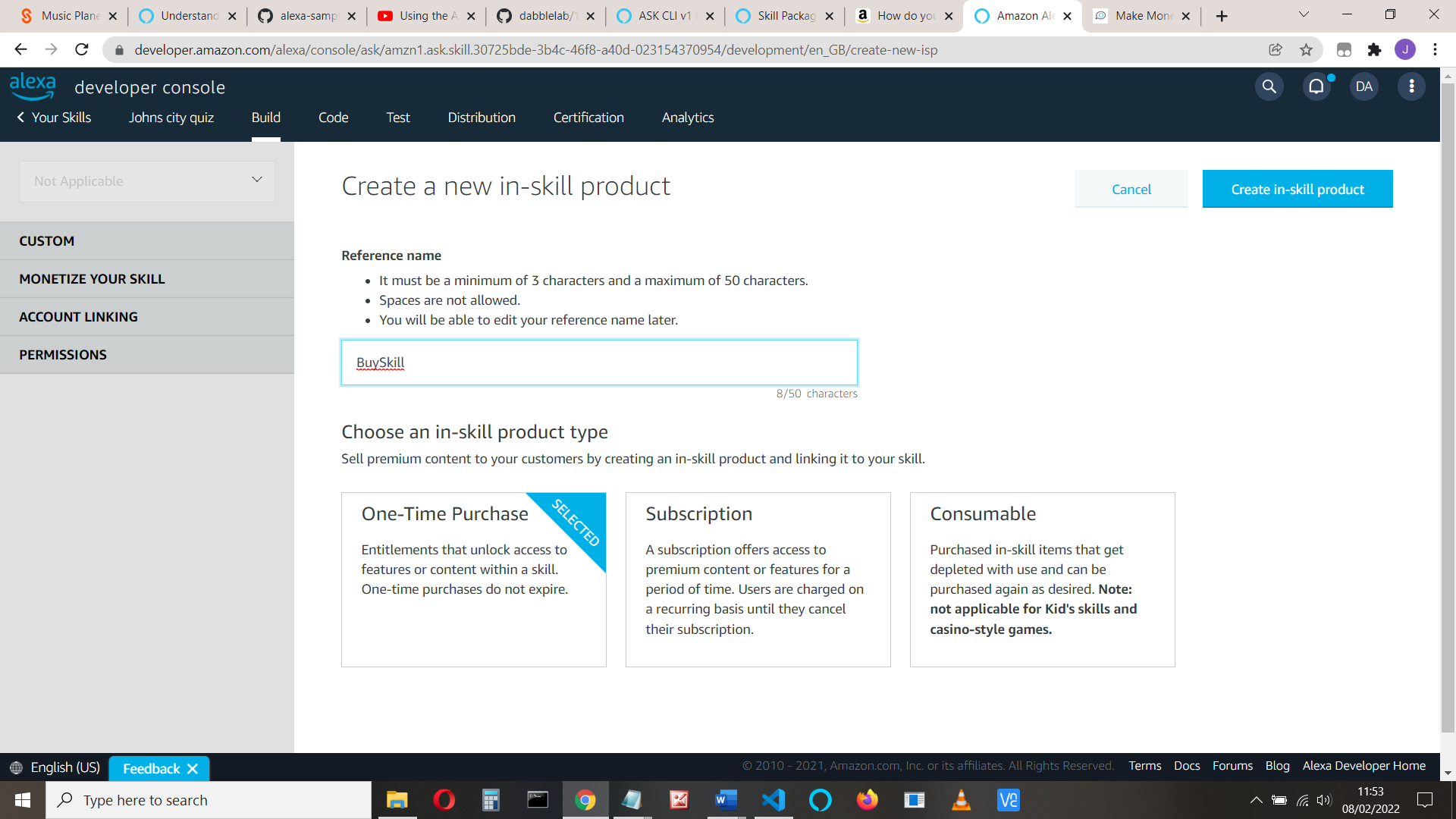
Change the invocation to **johns joke skill**

Click Build > Models > Monetize your skill, Select Add in-skill products:



See <https://developer.amazon.com/en-US/docs/alexa/in-skill-purchase/create-isp-dev-console.html>

Note the 3 types of ISP options are here.



Name it **joke** and select **One-Time Purchase**, and save

(We’ll see later that it’s best to give it a simple name)

Now we need to answer the questions about the skill (supported languages, price, icon, testing instructions, etc)

Add supported language (English(UK)) and answer the questions:

Display name –use **joke** – again use a simple name that the user can say

One sentence description – **This gives you access to more jokes**

Detailed description – **Buy this to gain access to more jokes.**

Small icon for this in-skill product – The image used for the product. \*

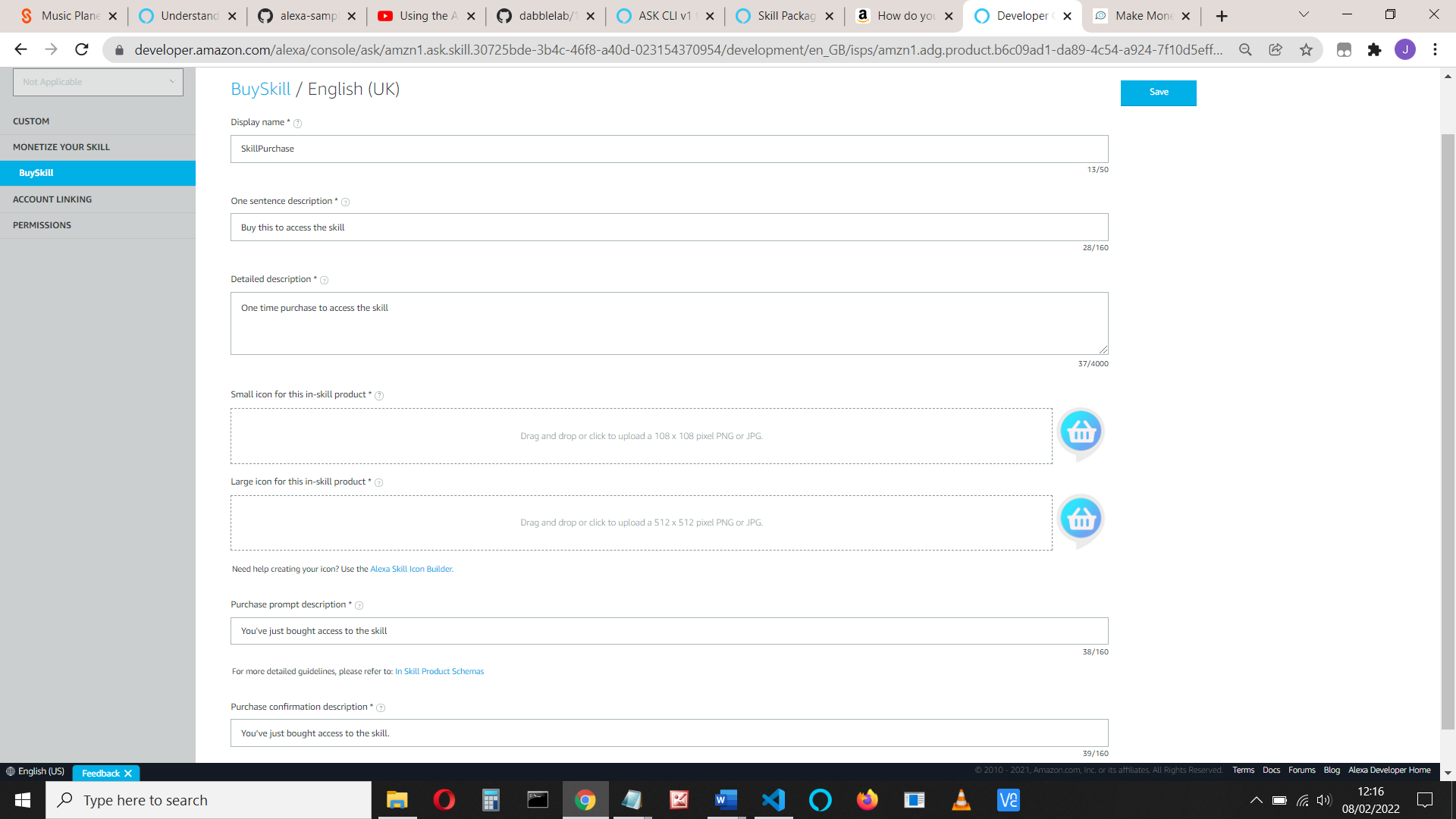
Large icon for this in-skill product – The image used for the product. \*

\* To create an icon, you can use <https://developer.amazon.com/docs/tools/icon-builder.html>

It downloads both 108 and 512 files sizes

Purchase prompt description – What the customer hears when making a purchase – **Thank you, you've now got access to more jokes**

Purchase confirmation description – **You’ve just bought access to more jokes.**



Click **Save**

Answer the other questions:

Price and Availability: minimum is 0.99

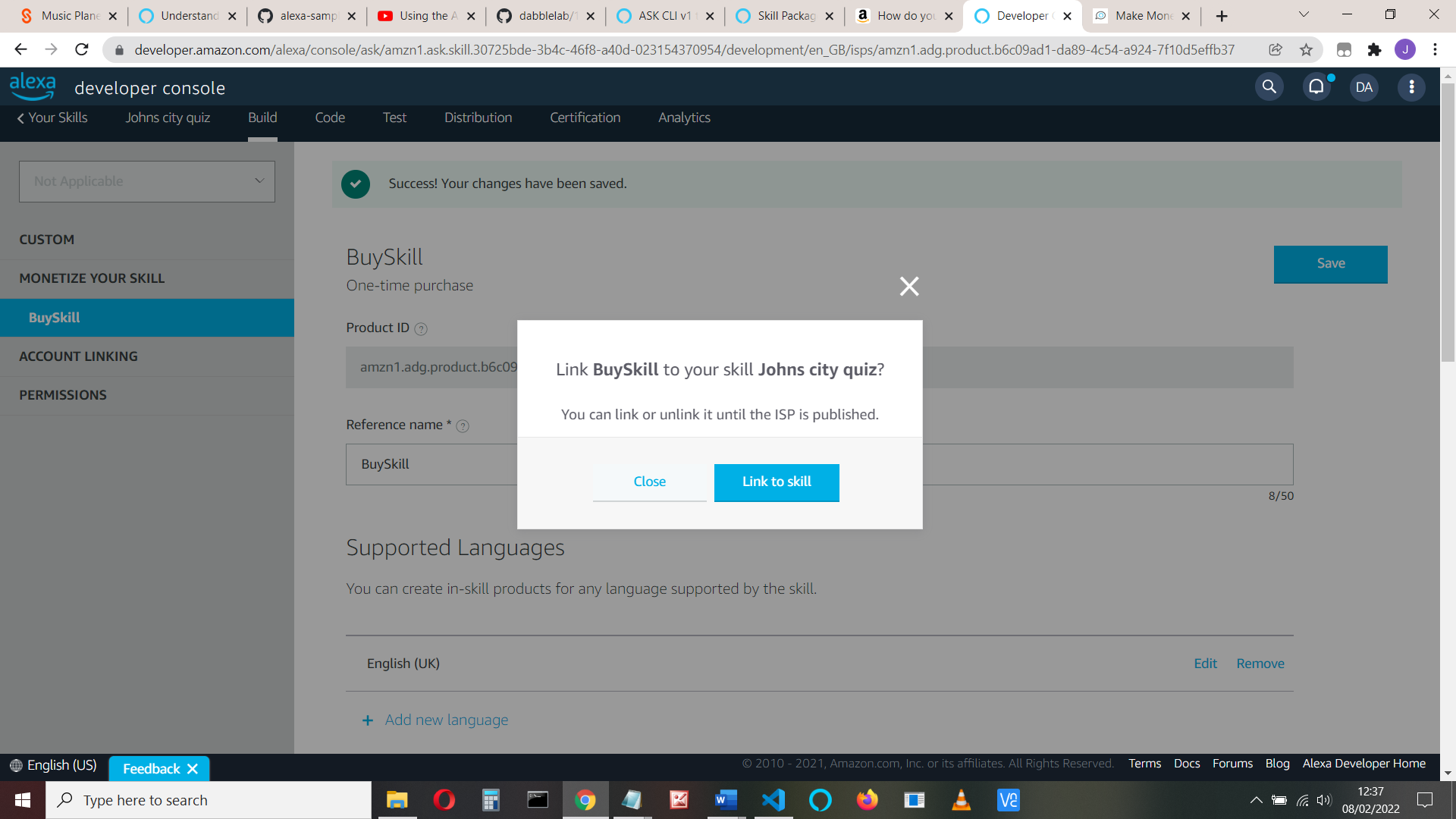
I’ve selected Amazon.com and Amazon.co.uk

Tax category: Software

For Testing, use: “Start a game. Say ‘upgrade’ to purchase the product”

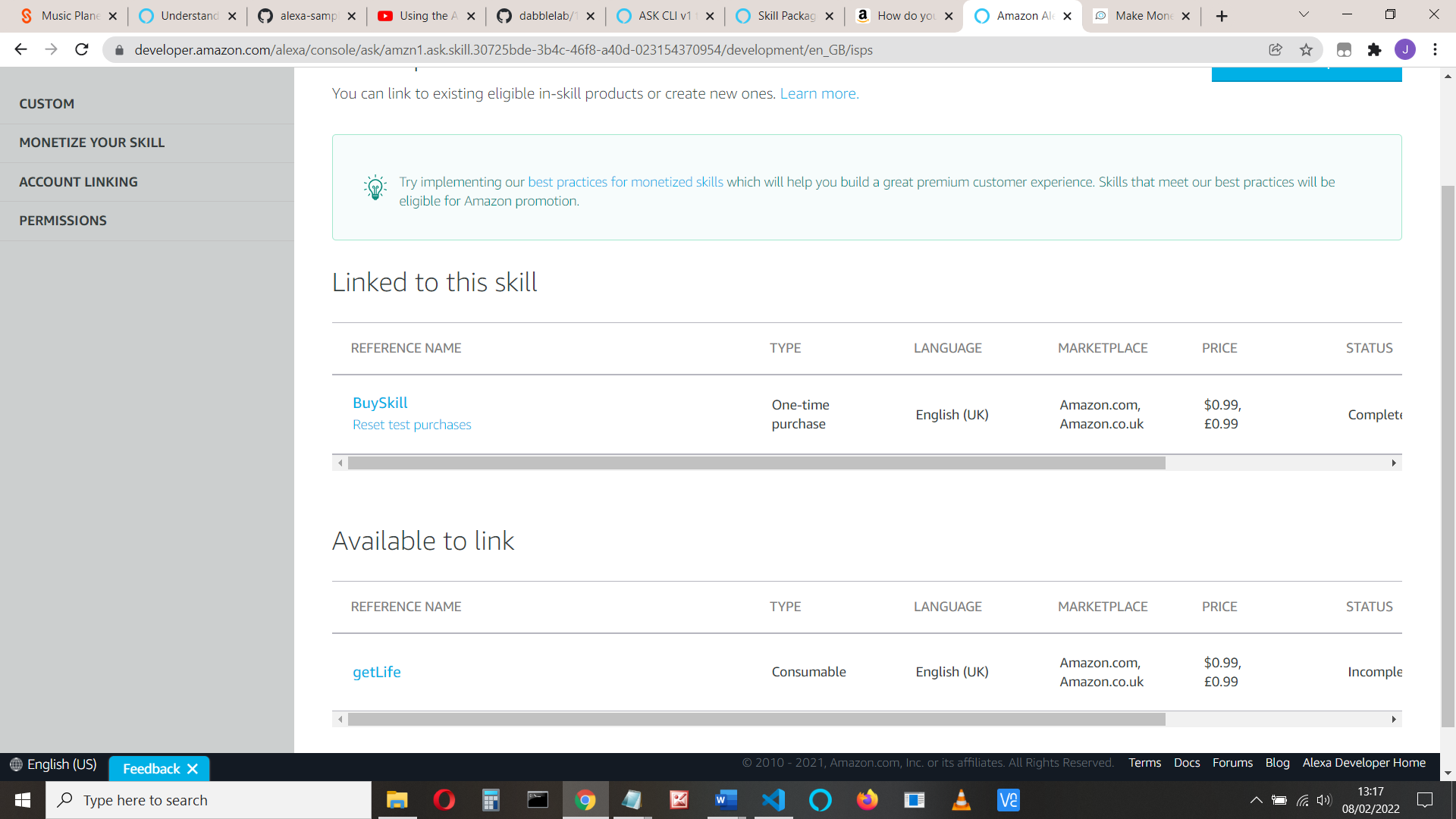
This is more important if you are going to really publish the skill

Save and link to skill



Add more languages if you want

If you return to Monetize your skill you will also see any other products you have available to link and can also add them:

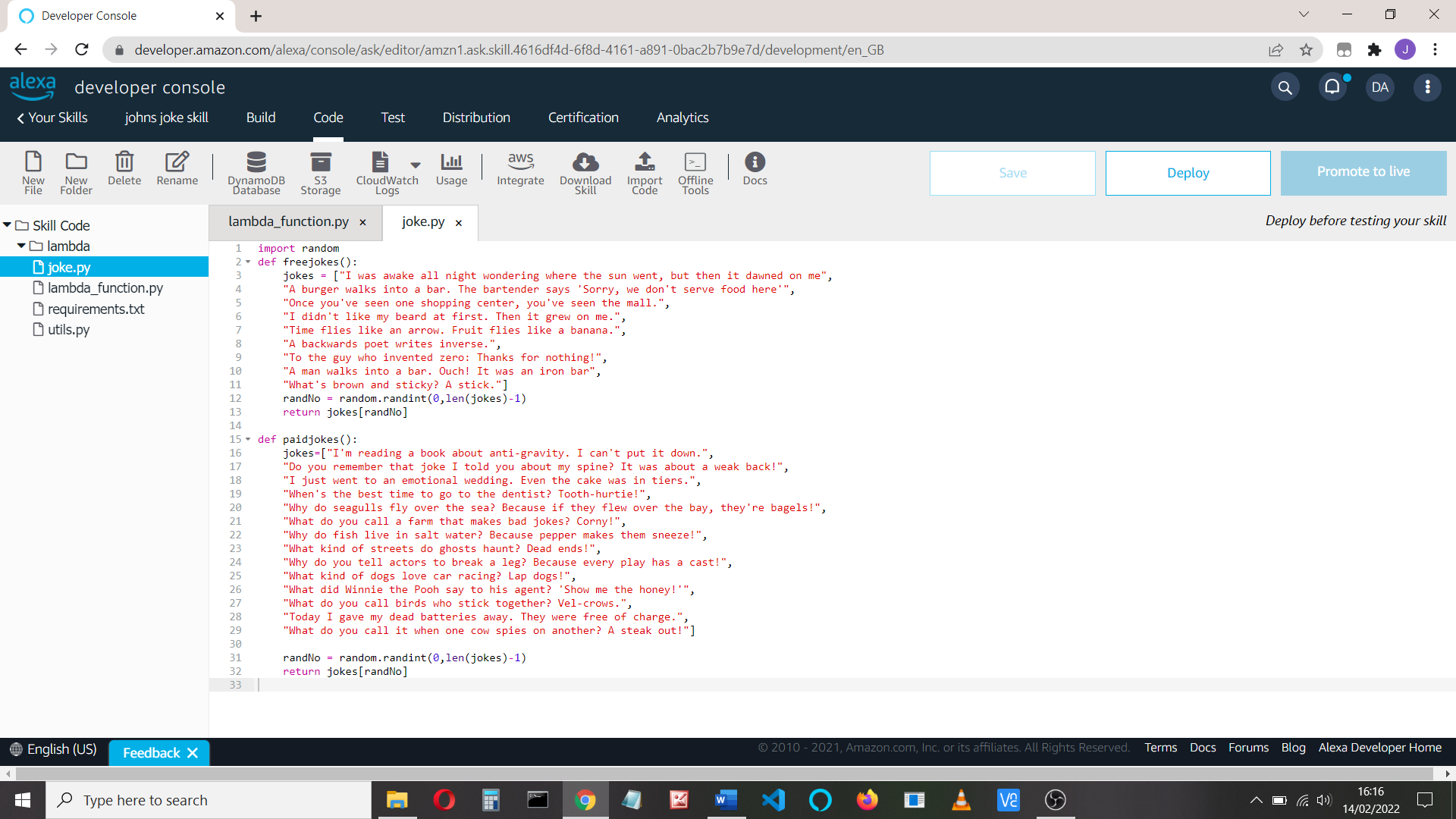


That’s it for creating the ISP. Let’s see how we access them from our code

Part c) **Add your code**

See <https://github.com/alexa-samples/skill-sample-python-fact-in-skill-purchases/blob/master/lambda/py/lambda_function.py>

Add a new file joke.py and insert the code:



Now go to lambda\_function.py

Add the code: **import joke** at the top of the code and add this line of code below the speak\_output in launch request:

speak\_output = joke.freejokes()

Deploy and test

Let’s check that free and paid jokes work in the HelloWorld handler. We’ll use session attributes to check for paid joke access:

Change the code in the HelloWorld handler to

class HelloWorldIntentHandler(AbstractRequestHandler):

"""Handler for Hello World Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("HelloWorldIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

ask\_output = "Here's a free joke: " + joke.freejokes()

speak\_output = "Here's your free joke: " + joke.freejokes() + ". You can purchase more jokes, just say 'What can I buy'"

session\_attr = handler\_input.attributes\_manager.session\_attributes

if "paid\_jokes" in session\_attr:

if session\_attr["paid\_jokes"] == True:

speak\_output = "Here's your paid joke: " +joke.paidjokes()

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(ask\_output)

.response

)

Deploy and test. Say ‘hello’ to get a free joke.

You can test this for paid for jokes if you set *session\_attr["paid\_jokes"] =True* in the Launch handler:

session\_attr = handler\_input.attributes\_manager.session\_attributes

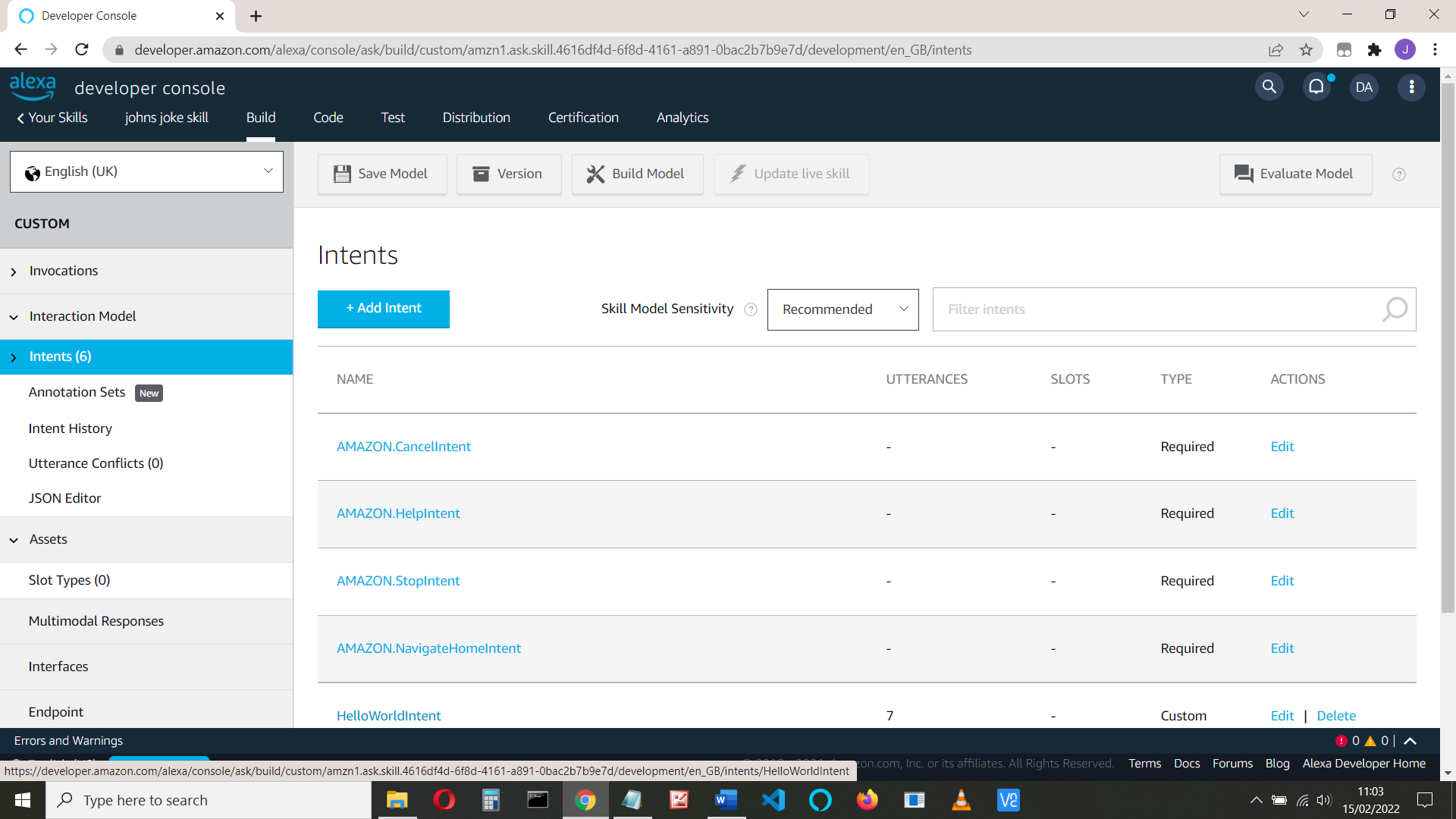
session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

Check that you get a joke from the paidjokes list.

Later, we’ll prompt for a joke, so add ‘Tell me a joke” to the HelloWorldIntent:

(Build > Intents> HelloWorldIntent)



And add “Tell me a joke” to the utterances.



**Retrieve the available in-skill products, get information on them, and purchase.**

To do this, we must add an api client.

Add the following line to requirements.txt

boto3==1.9.216

ask-sdk-core==1.11.0

**ask\_sdk**

In your code change *sb = SkillBuilder()*

To: **sb = StandardSkillBuilder()**

Add these imports at the top of the code

from ask\_sdk.standard import StandardSkillBuilder

from ask\_sdk\_model.services.monetization import (

EntitledState, PurchasableState, InSkillProductsResponse, Error,

InSkillProduct)

from ask\_sdk\_model.interfaces.monetization.v1 import PurchaseResult

from ask\_sdk\_model import Response, IntentRequest

from ask\_sdk\_model.interfaces.connections import SendRequestDirective

At the top of your code, add all these functions, which we’ll use later:

def **in\_skill\_product\_response**(handler\_input):

"""Get the In-skill product response from monetization service."""

# type: (HandlerInput) -> Union[InSkillProductsResponse, Error]

locale = handler\_input.request\_envelope.request.locale

ms = handler\_input.service\_client\_factory.get\_monetization\_service()

return ms.get\_in\_skill\_products(locale)

def **get\_all\_entitled\_products**(in\_skill\_product\_list):

"""Get list of in-skill products in ENTITLED state."""

# type: (List[InSkillProduct]) -> List[InSkillProduct]

entitled\_product\_list = [

l for l in in\_skill\_product\_list if (

l.entitled == EntitledState.ENTITLED)]

return entitled\_product\_list

def **get\_speakable\_list\_of\_products**(entitled\_products\_list):

"""Return product list in speakable form."""

# type: (List[InSkillProduct]) -> str

product\_names = [item.name for item in entitled\_products\_list]

if len(product\_names) > 1:

# If more than one, add and 'and' in the end

speech = " and ".join(

[", ".join(product\_names[:-1]), product\_names[-1]])

else:

# If one or none, then return the list content in a string

speech = ", ".join(product\_names)

return speech

skill\_name = "Johns joke skill"

**View Product**

The **in\_skill\_product\_response** provides information about the product, e.g:

{'in\_skill\_products': [{'active\_entitlement\_count': 0,

'entitled': 'NOT\_ENTITLED',

'entitlement\_reason': 'NOT\_PURCHASED',

'name': 'paid joke',

'object\_type': 'ENTITLEMENT',

'product\_id': 'amzn1.adg.product.d0aed689-6761-483c-baf9-9d282ab7b788',

'purchasable': 'PURCHASABLE'

'purchase\_mode': 'TEST',

'reference\_name': 'paidforjoke',

'summary': 'This gives you access to more jokes'}],

'is\_truncated': None,

'next\_token': None}

Note the 'entitled' and 'entitlement\_reason'. We’ll use these later. Also note the 'product\_id' we’ll need that later to buy the product (see ##)

If you want to view the products, you can add the following to your code

in\_skill\_product\_list = in\_skill\_product\_response(handler\_input)

logger.info("in\_skill\_product\_list ")

logger.info(in\_skill\_product\_list )

We have several scenarios now:

1. The user hasn’t purchased access to the paid jokes

We need to tell them what’s available and ask if they want to purchase

1. The user has access to the paid skills. Tell a paid joke when asked
2. The user asks for paid membership and either confirms or declines.
3. The user has ‘paid membership’ and wishes to cancel

See: <https://developer.amazon.com/en-US/docs/alexa/paid-skills/test-paid-skills.html#test-purchase-flows>

And: <https://developer.amazon.com/en-US/docs/alexa/paid-skills/overview.html>

Let’s do this a bit at a time. Change the launch request to:

class LaunchRequestHandler(AbstractRequestHandler):

"""Handler for Skill Launch."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("LaunchRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Welcome, you can ask for a joke or say hello"

in\_skill\_response = in\_skill\_product\_response(handler\_input)

if isinstance(in\_skill\_response, InSkillProductsResponse):

entitled\_prods = get\_all\_entitled\_products(in\_skill\_response.in\_skill\_products)

if entitled\_prods:

speak\_output = (

"Welcome to {}. You currently own {} products. "

"To hear a paid joke, say, 'Tell me a joke.").format(

skill\_name,

get\_speakable\_list\_of\_products(entitled\_prods))

else:

logger.info("No entitled products")

speak\_output = (

"Welcome to {}. To hear a joke you can say "

"'Tell me a joke', or to hear about the paid for jokes "

"for purchase, say 'What can I buy', or ask for help"

).format(skill\_name)

reprompt = "I didn't catch that. What can I help you with?"

else:

logger.info("Error calling InSkillProducts API: {}".format(

in\_skill\_response.message))

speak\_output = "Something went wrong in loading your purchase history."

reprompt = speak\_output

return (

handler\_input.response\_builder

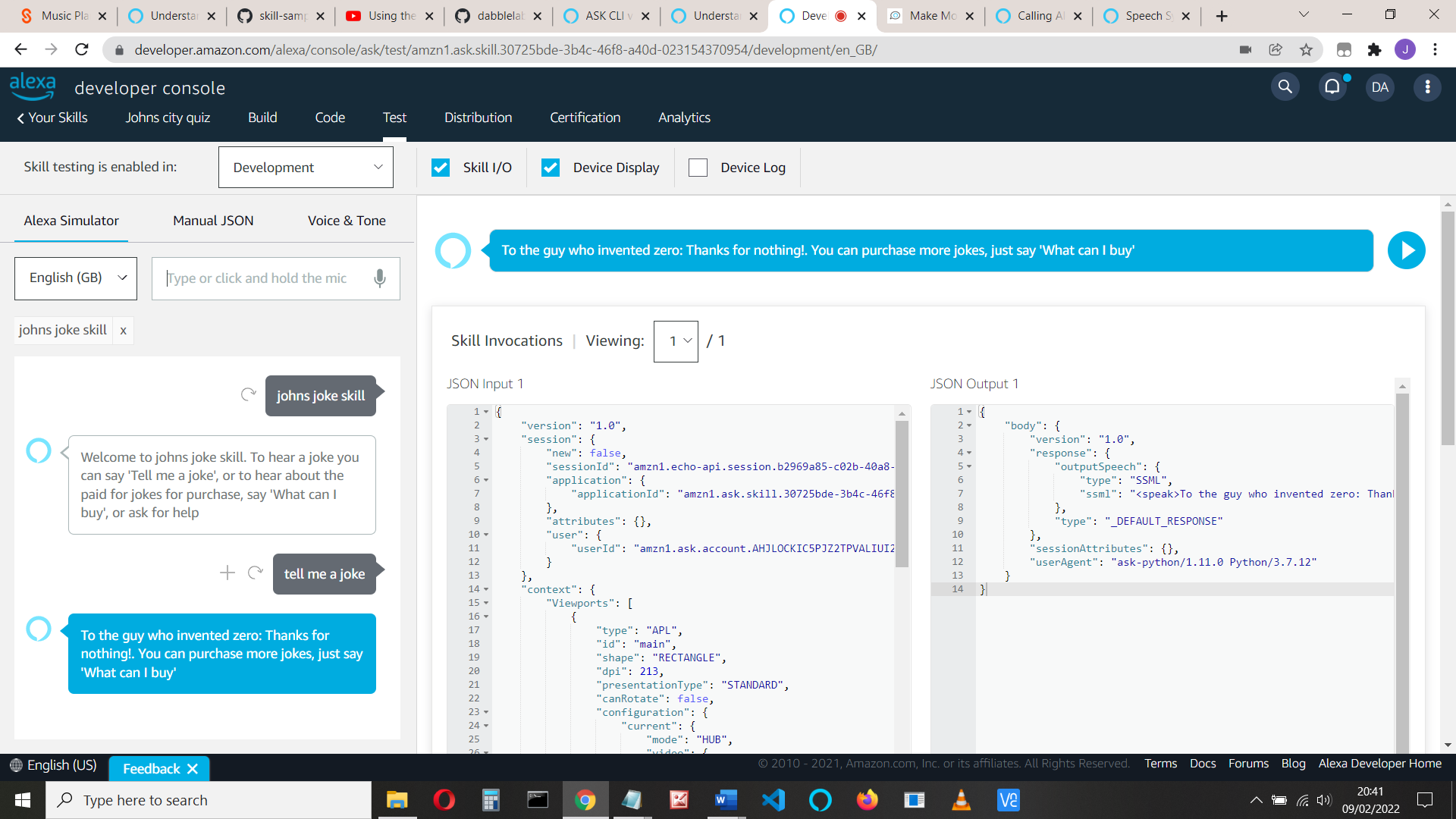
.speak(speak\_output)

.ask(speak\_output)

.response

)

Save, deploy and test. At the moment we aren’t entitled to any ISP, but you can ask for a joke.

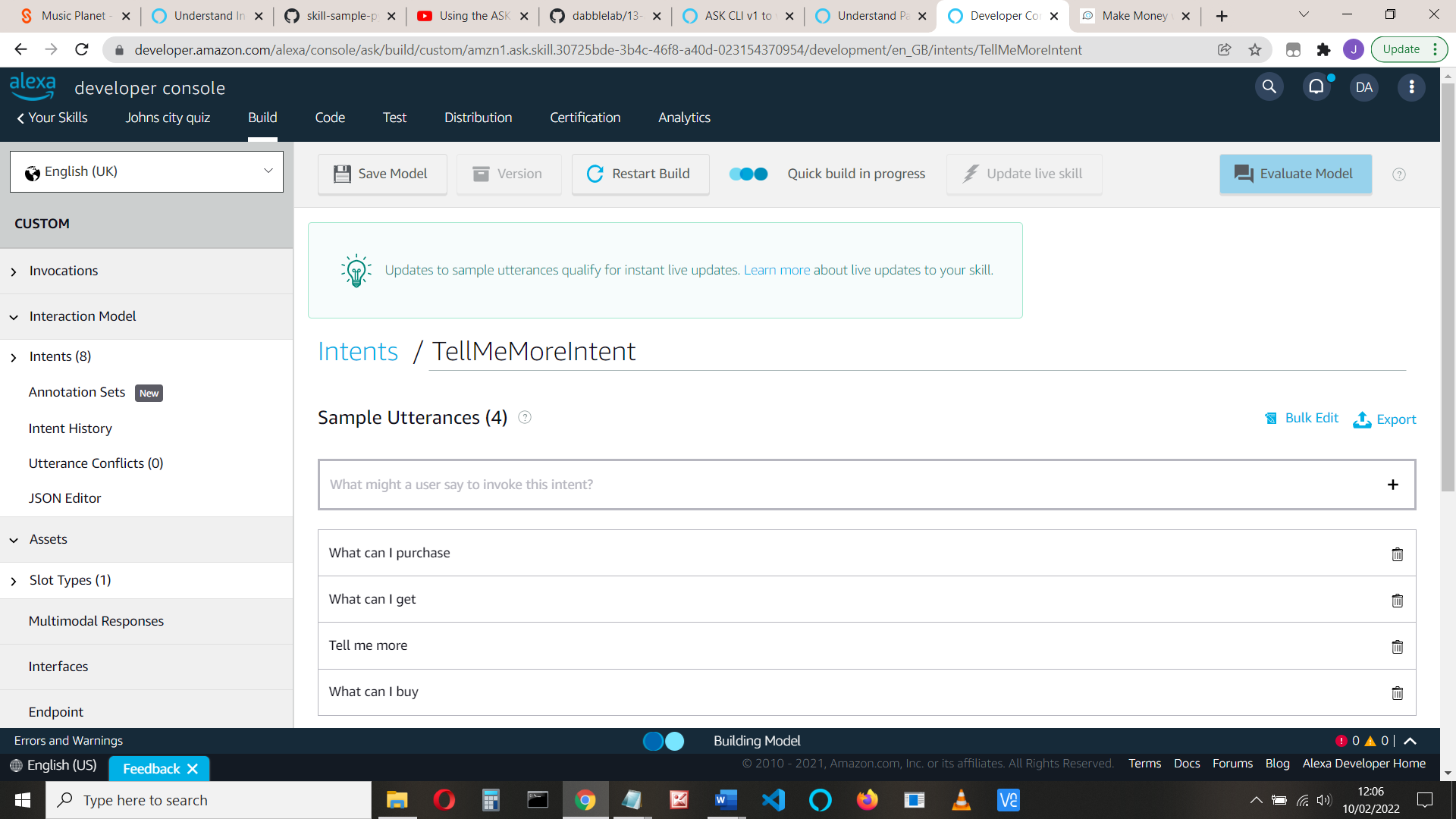


**Produce detail and purchase**

Now let’s add the purchase options – ‘tell me more’ and ‘buy’

We’ll need to add the intents. First, we’ll add a ‘tell me more’ intent that will give details about the product and ask if the user wants to buy it.

Add a TellMeMore intent, either using the console:



or edit the JSON:

{

"name": "TellMeMoreIntent",

"slots": [],

"samples": [

"What can I purchase",

"What can I get",

"Tell me more",

"What can I buy"

]

}

Save and build the model

In your code, at the top of the sb.add\_request\_handler at the end of the code, add this line:

**sb.add\_request\_handler(TellMeMoreHandler())**

Add the TellMemoreIntent code (say before class LaunchRequestHandler.. )

class TellMeMoreHandler(AbstractRequestHandler):

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("TellMeMoreIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In TellMeMoreHandler")

# Tell the user about the product(s)

speak\_output = 'You can upgrade to hear more jokes. Just say, "Upgrade" to purchase this product.'

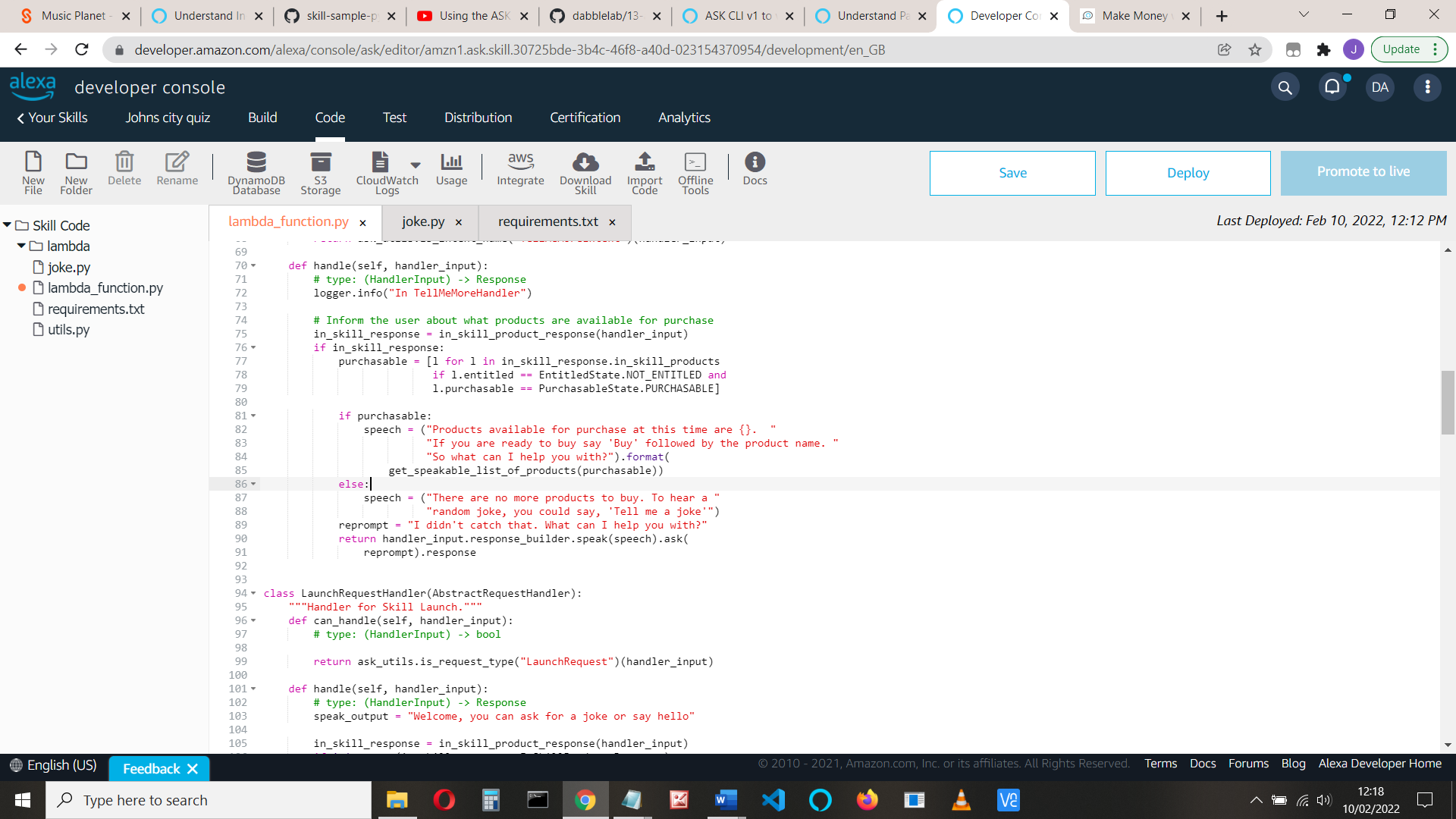
reprompt = "I didn't catch that. What can I help you with?"

return handler\_input.response\_builder.speak(speak\_output).ask(

reprompt).response

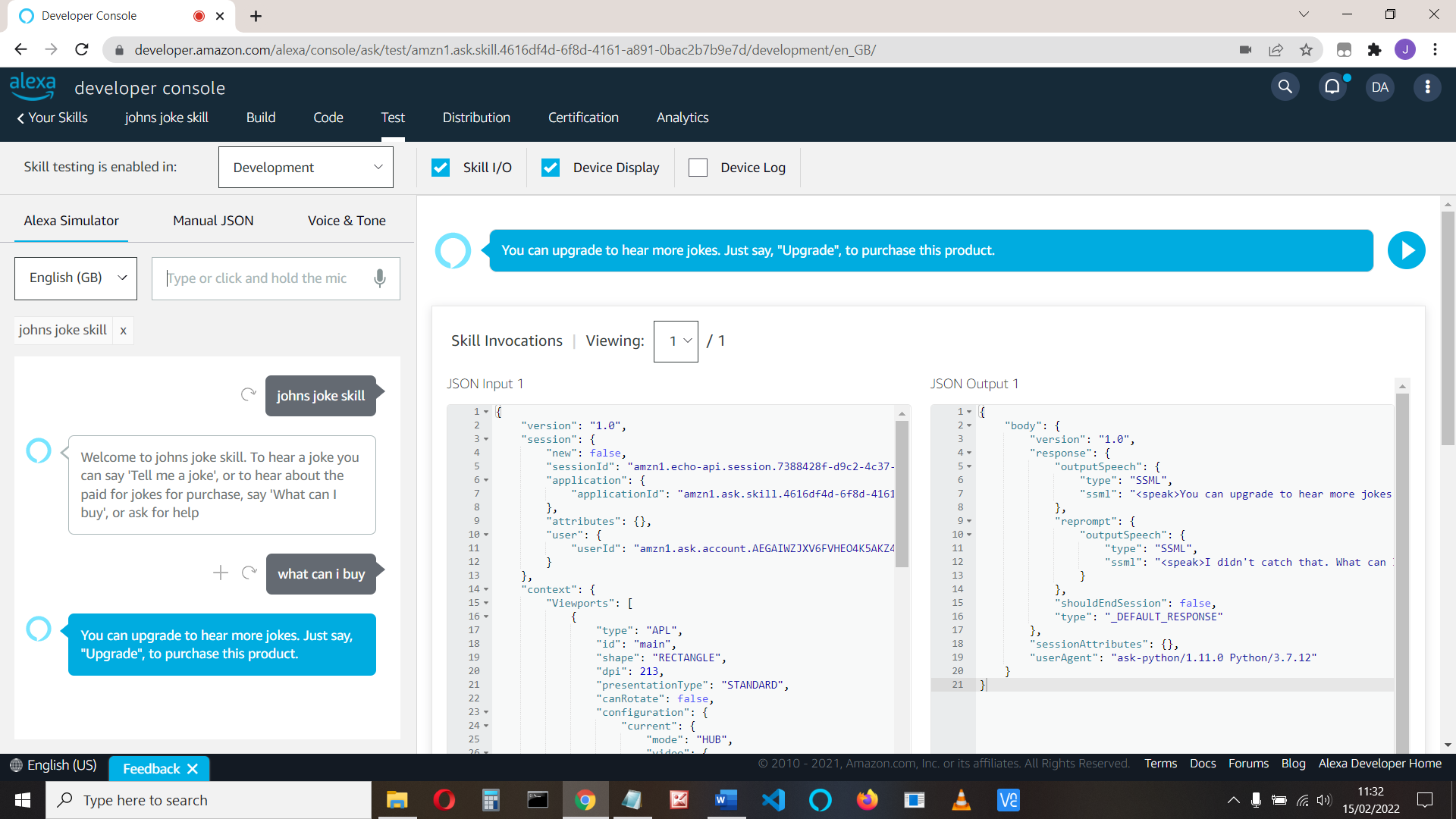
Note: If you tell the user to ‘buy’ the product, then Alexa will hear ‘Bye’ and close the skill!

If you have more than one product you can get a list of them. Your code would be something like this:



See <https://github.com/alexa-samples/skill-sample-python-fact-in-skill-purchases/blob/master/lambda/py/lambda_function.py>

Save the model and deploy and test the code.

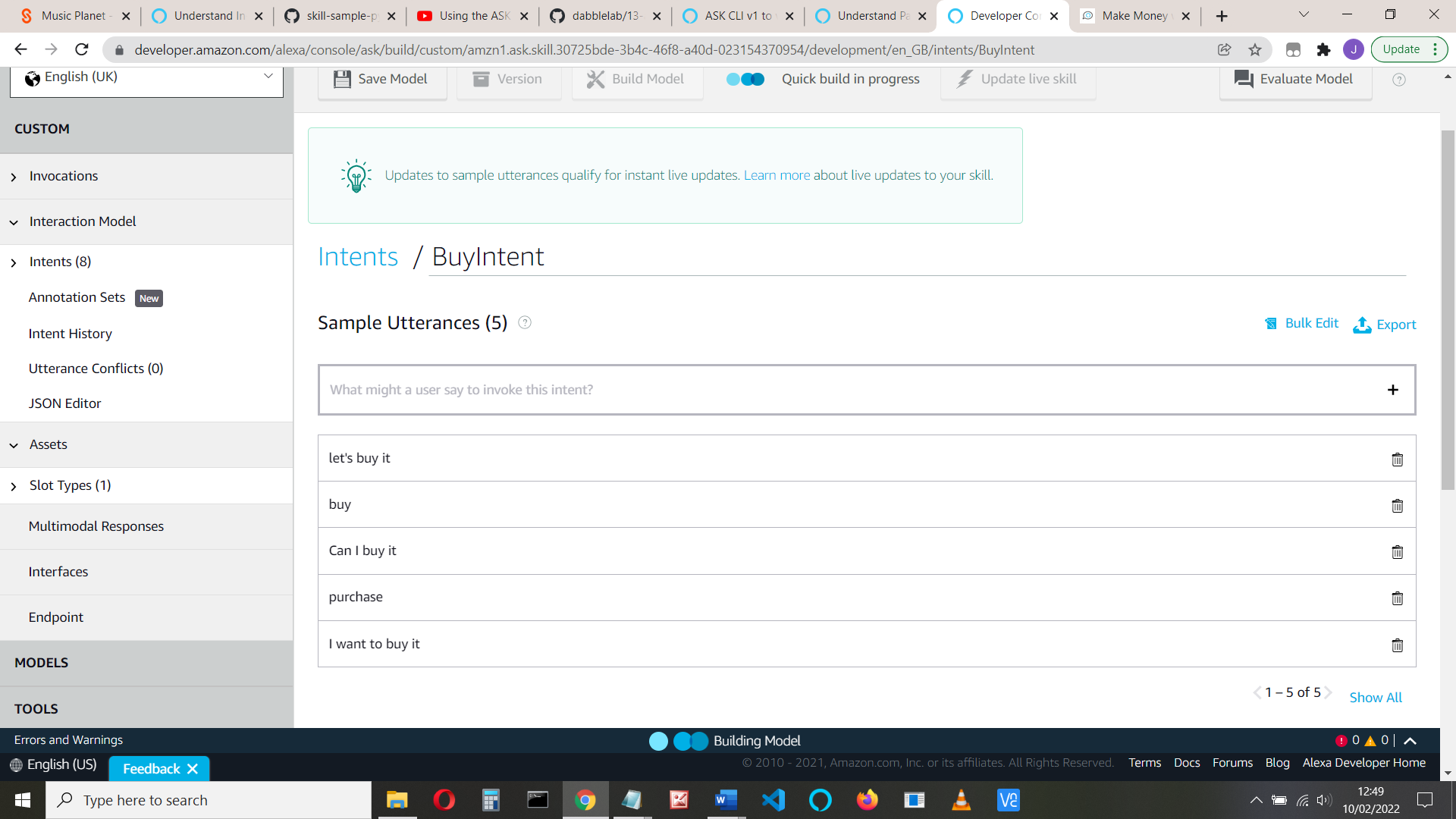


**Purchase**

Nearly there. Let’s add the buy intent.

Note: I found if I used ‘buy’ as an utterance, Alexa heard ‘Bye’ and terminated!

Console:



Or JSON:

{

"name": "BuyIntent",

"slots": [],

"samples": [

"let's buy it",

"upgrade",

"Can I buy it",

"purchase",

"I want to buy it"

]

},

If you have more than one product, your utterances will be something like:

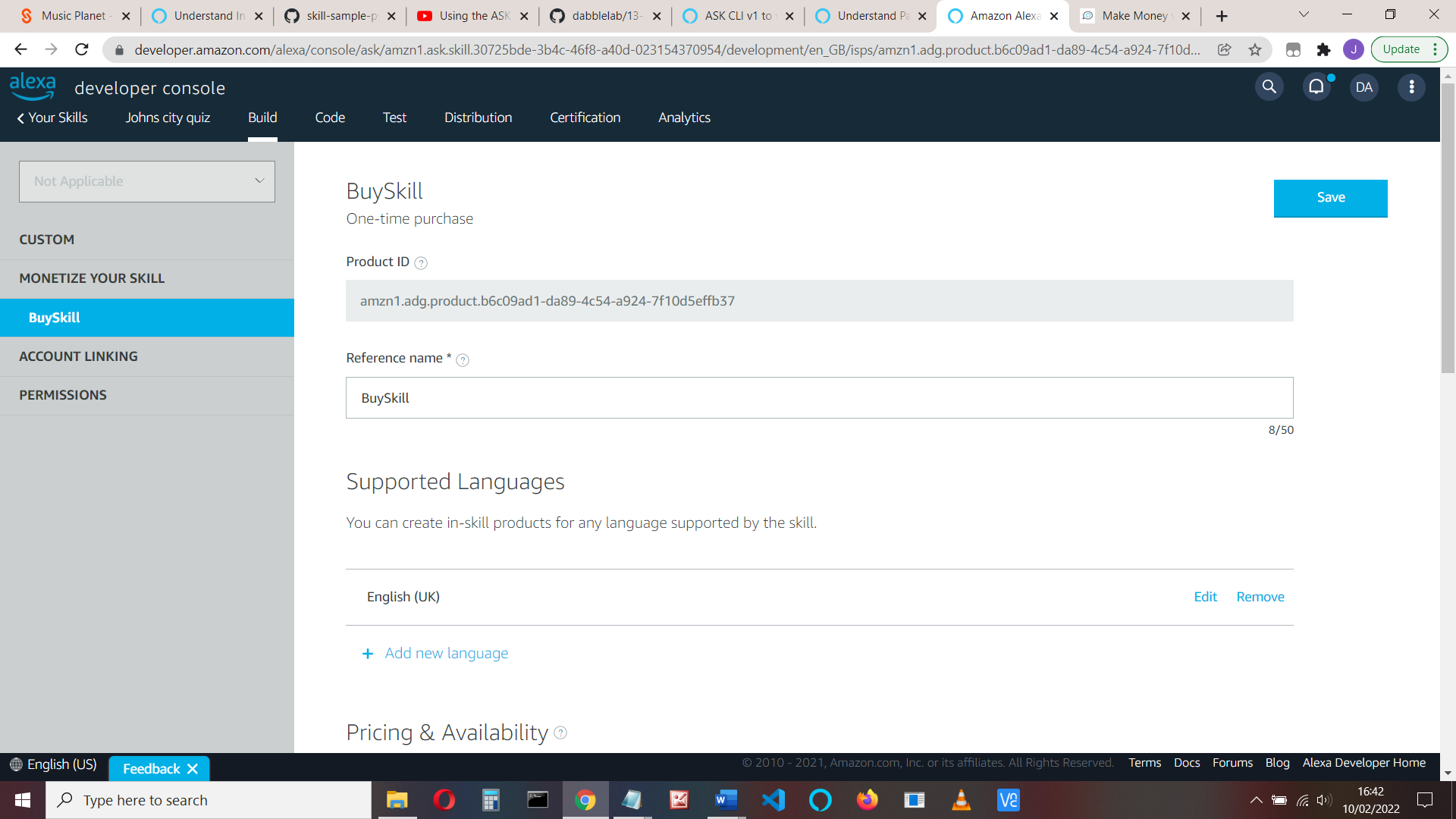
let's buy {productCategory} pack

Again, see the Alexa Sample skill for more information

Now let’s add the ‘Buy’ code. We will need two intent handlers here. One to purchase the product (which passes this to Amazon) and a second to deal with the response from Amazon with what the user says (e.g. Yes or No to ‘Buy’). We do this in two steps.

We’ll need the product ID. We saw that before (## in **View Product).**

You can also find it by returning to Build > Monetize and choose the product:



Copy the Product ID. It’s something like

amzn1.adg.product.b6c09ad1-da89-4c54-a924-7f10d5effb37

Add the sb handler as before, so that the code is like this:

sb = StandardSkillBuilder()

sb.add\_request\_handler(TellMeMoreHandler())

**sb.add\_request\_handler(BuyHandler())**

sb.add\_request\_handler(LaunchRequestHandler())

Add the following Buy intent handler code (below your TellMeMoreIntent code)

class BuyHandler(AbstractRequestHandler):

"""Handler for letting users buy the product.

User says: Alexa, upgrade

"""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("BuyIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyHandler")

product\_id = "PASTE YOUR PRODUCT ID HERE"

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Buy",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="buyToken")

).response

This code sends a ‘Buy’ request to Amazon with the product ID. Note that we use an add\_directive here. This handles the purchase, then returns control to our skill along with the purchase response.

We also need code to handle the response. We’ll add a ‘buy response handler’. This handles the **Connections.Response** event.

See <https://developer.amazon.com/en-US/docs/alexa/in-skill-purchase/add-isps-to-a-skill.html#handle-results>

We receive a **payload.purchaseResult** which is either ACCEPTED, DECLINED, ALREADY\_PURCHASED, or ERROR. We will deal with all these cases

Add the sb add\_request as before, so that it is:

sb.add\_request\_handler(TellMeMoreHandler())

sb.add\_request\_handler(BuyHandler())

**sb.add\_request\_handler(BuyResponseHandler())**

Add the code:

class BuyResponseHandler(AbstractRequestHandler):

"""This handles the Connections.Response event after a buy occurs."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Buy")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyResponseHandler")

if handler\_input.request\_envelope.request.status.code == "200":

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = "You have bought the access to more jokes, enjoy"

reprompt = "You can ask for more jokes"

elif purchase\_result in (

PurchaseResult.DECLINED.value,

PurchaseResult.ERROR.value,

PurchaseResult.NOT\_ENTITLED.value):

speech = "Thanks for your interest. Ask for another free joke"

reprompt = "Ask for a joke"

elif purchase\_result == PurchaseResult.ALREADY\_PURCHASED.value:

logger.info("Already purchased product")

speech = " You've already purchased more jokes, just ask for another joke"

reprompt = "Ask for a joke, help or exit"

else:

# Invalid purchase result value

logger.info("Purchase result: {}".format(purchase\_result))

speech = "Sorry, there was an error, please try again"

reprompt = "Ask for a joke, help or exit"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Error: {}".format(

handler\_input.request\_envelope.request.status.message))

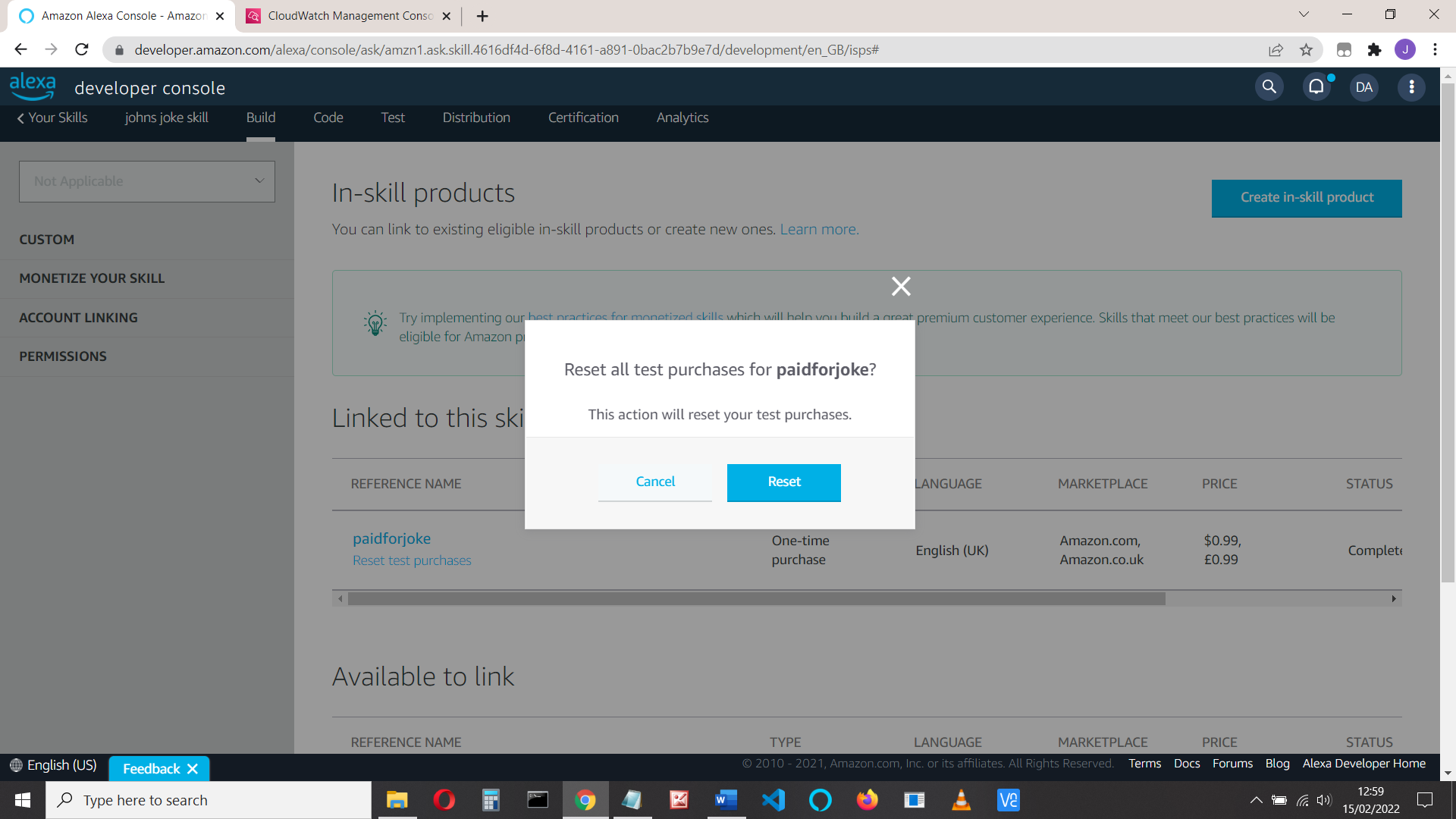
return handler\_input.response\_builder.speak(

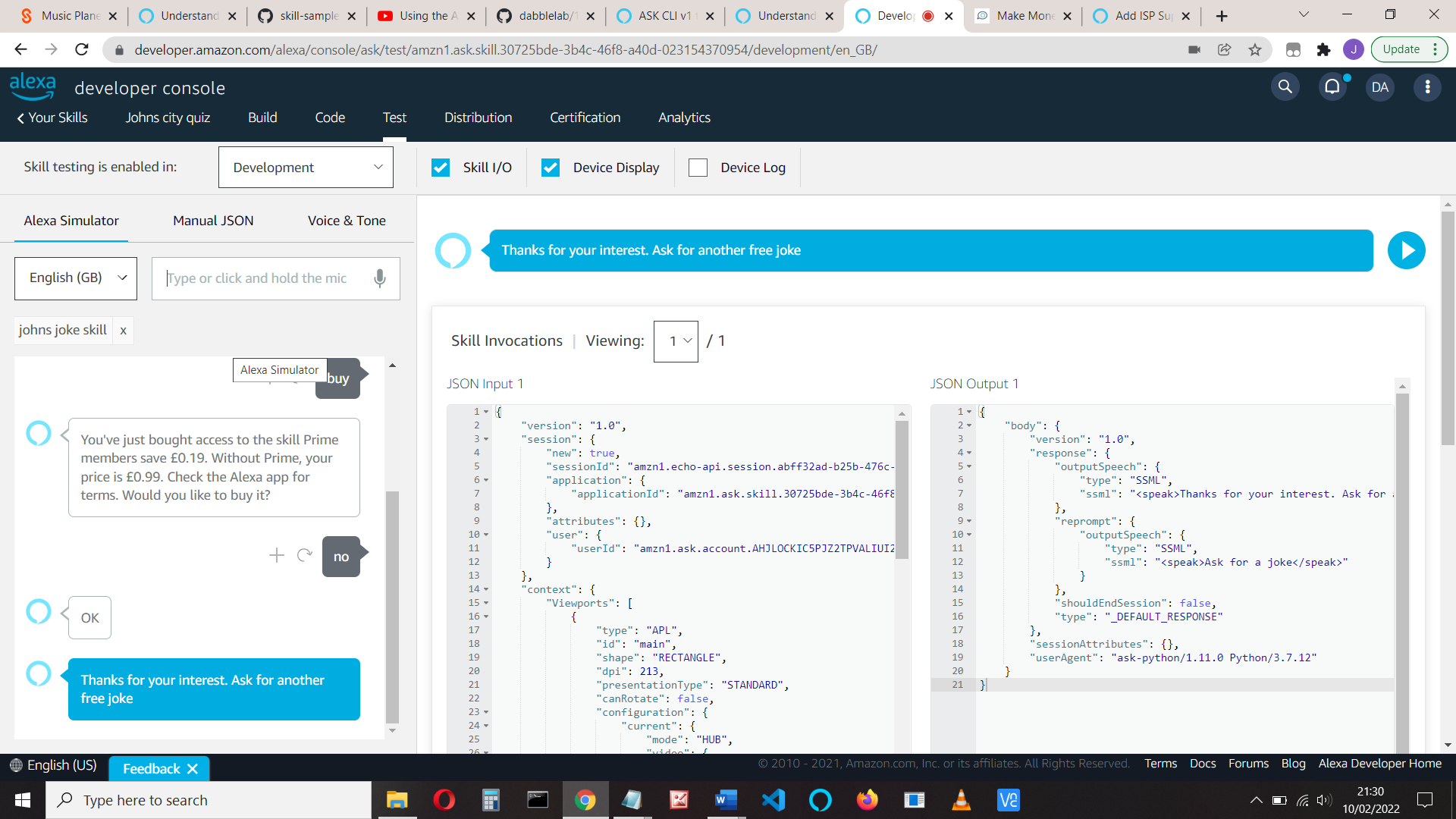
"There was an error handling your purchase request. "

"Please try again or ask for help")

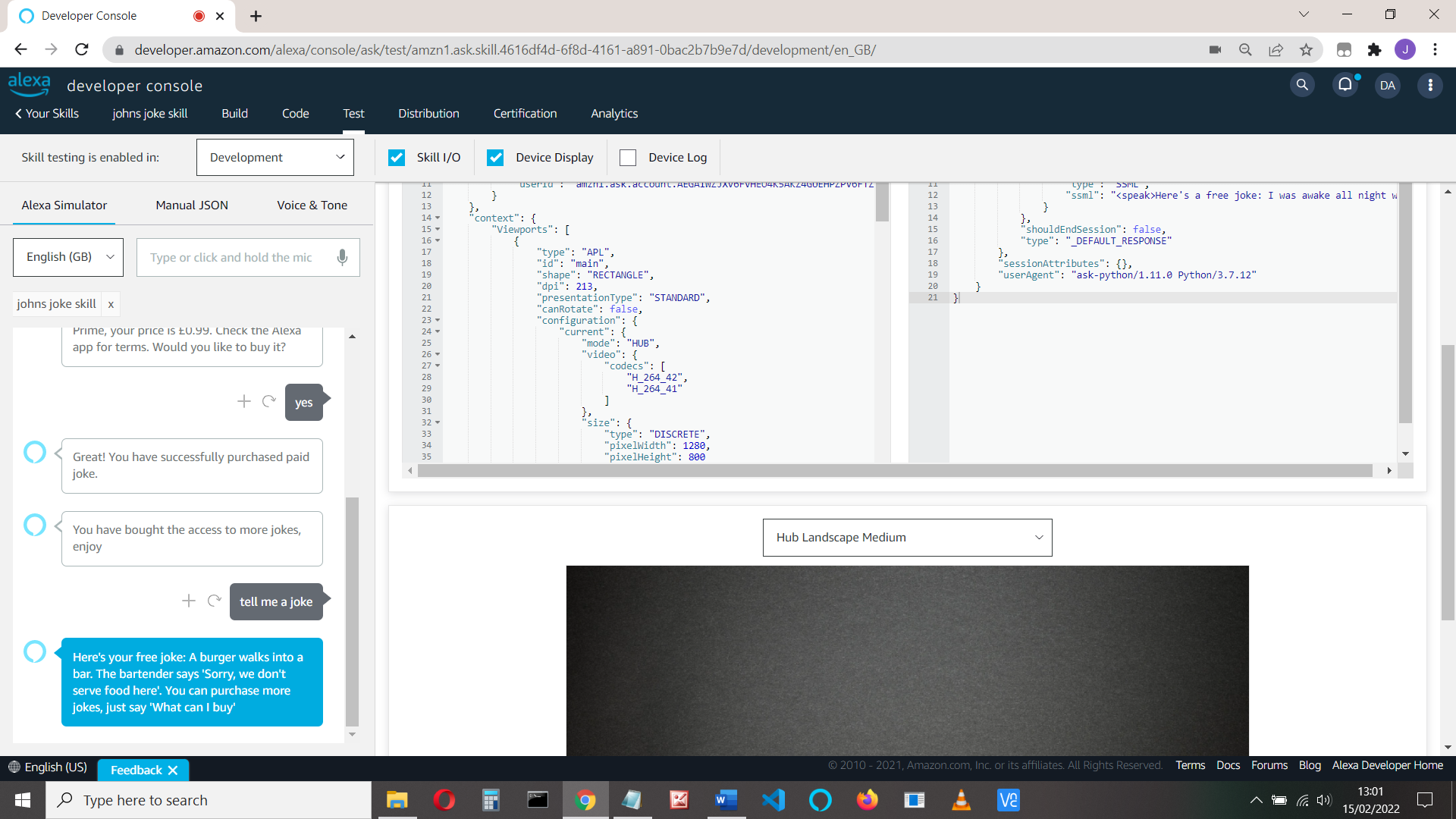
Note that we update the session attribute.

If necessary, reset the purchases (build > monetize> reset) and test the skill:

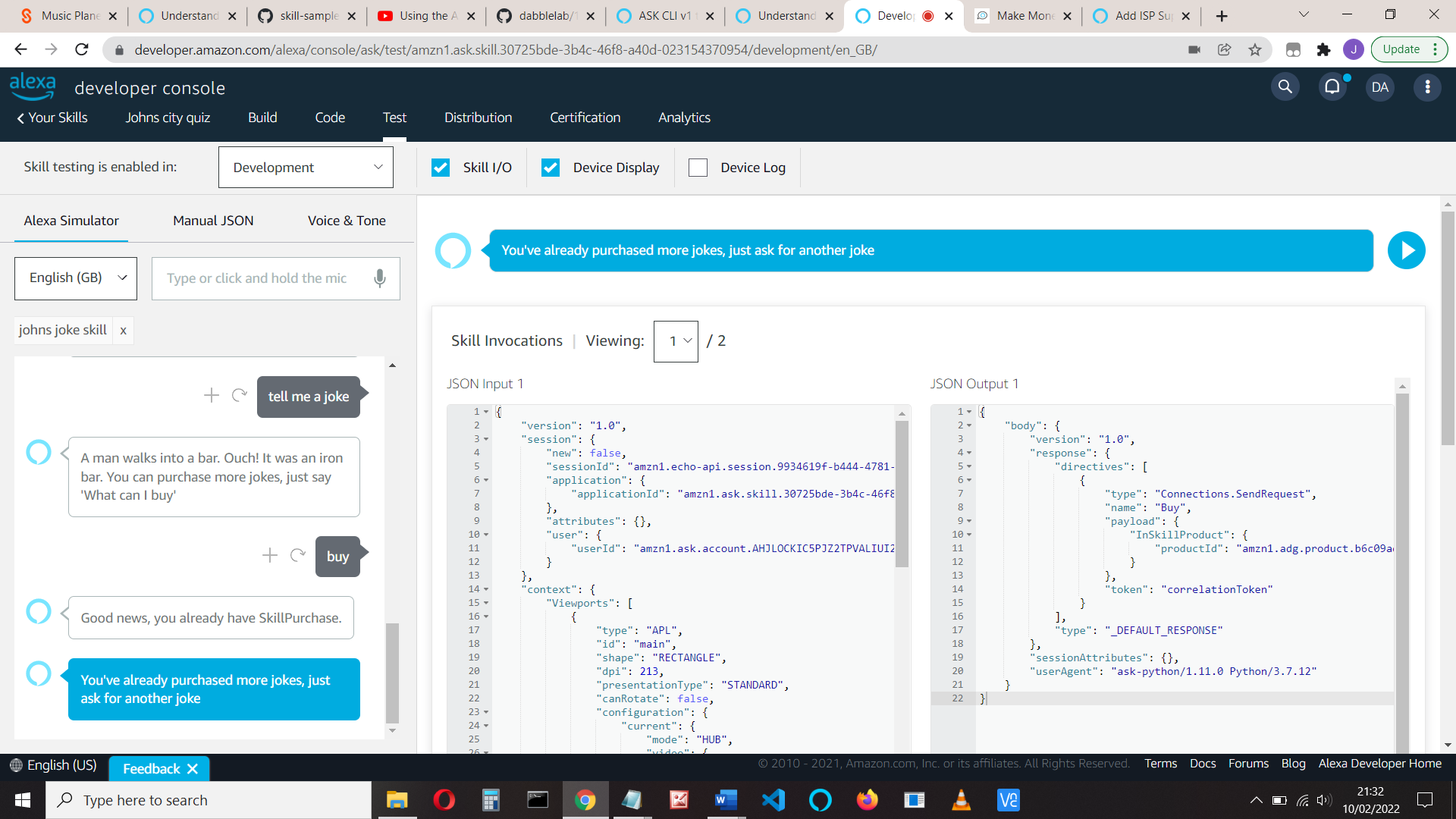


So ‘No’ to purchase:

Say ‘Yes’ to purchase



And if we’ve already got it



But if you try this, you see that we are still only getting free jokes.

Add this code to the LaunchRequest, after

entitled\_prods = get\_all\_entitled\_products(in\_skill\_response.in\_skill\_products)

if entitled\_prods and len(entitled\_prods) > 0:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

**Failed Purchase**

You should check for a failed purchase. We do that in the BuyResponseHandler code:

else:

# Invalid purchase result value

logger.info("Purchase result: {}".format(purchase\_result))

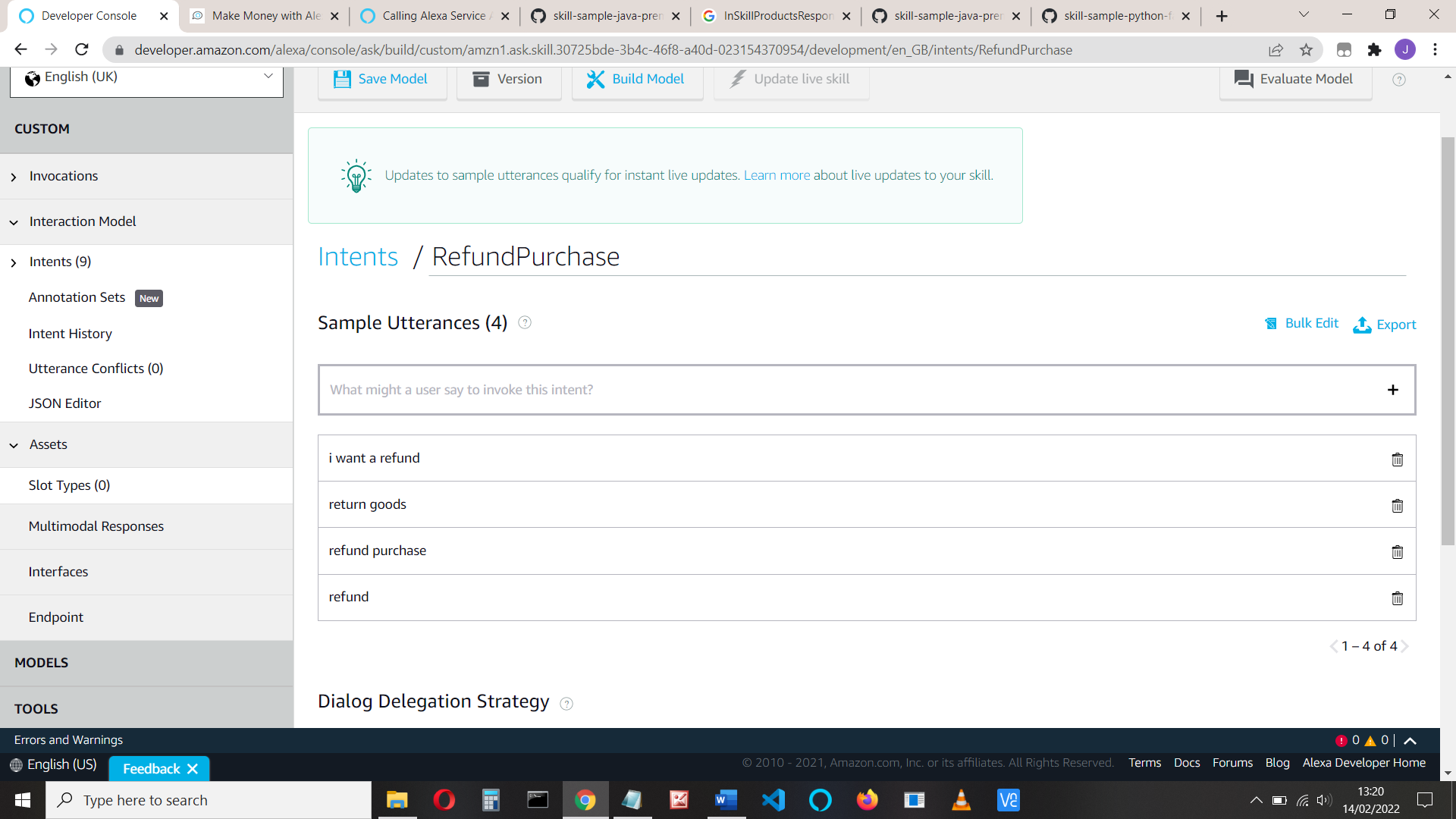
speech = "Sorry, there was an error, please try again"

reprompt = "Ask for a joke, help or exit"

**Refunds**

You must implement a refund / returns policy.

As previously, add a new Intent. Call it RefundPurchase, and add utterances: “Refund”, Refund purchase”, “return goods”, “I want a refund” etc.:



Or add the JSON code:

{

"name": "RefundPurchase",

"slots": [],

"samples": [

"i want a refund",

"return goods",

"refund purchase",

"refund"

]

}

Now add the refund code. Again, like the purchase, we will have two handlers: one will use an add\_directive to send a SendRequestDirective with type “Cancel" and the second to handle the response from Amazon/Alexa (remember the error we had before: Unable to find a suitable request handler)

Add the sb handlers

sb.add\_request\_handler(TellMeMoreHandler())

sb.add\_request\_handler(BuyHandler())

sb.add\_request\_handler(BuyResponseHandler())

**sb.add\_request\_handler(RefundPurchaseHandler())**

**sb.add\_request\_handler(CancelResponseHandler())**

And the RefundPurchase code:

class RefundPurchaseHandler(AbstractRequestHandler):

#

#Deal with refund requests

#User says: Alexa, refund

#

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("RefundPurchase")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In RefundPurchaseHandler")

product\_id= ‘ADD YOUR PRODUCT ID HERE’

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Cancel",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="correlationToken")

).response

This simply sends a Cancel RequestDirective with the product ID. We need to know the product ID for the item being refunded. If you have more than one, refer to the Alexa code, about line 550 at:

[hhttps://github.com/alexa-samples/skill-sample-python-fact-in-skill-purchases/blob/136b6cd0a78fc7268f38d31a69e08cbb292257d4/lambda/py/lambda\_function.py](https://github.com/alexa-samples/skill-sample-python-fact-in-skill-purchases/blob/136b6cd0a78fc7268f38d31a69e08cbb292257d4/lambda/py/lambda_function.py#L220)

We also need the Cancel response handler.

Add the request handler. You could just set the paid\_joke session attribute to False:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = False

handler\_input.attributes\_manager.session\_attributes = session\_attr

But this code handles with other possibilities and gracefully, checking purchase ACCEPTED, and DECLINED.

class CancelResponseHandler(AbstractRequestHandler):

#This handles the Connections.Response event after a cancel occurs.

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Cancel")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In CancelResponseHandler")

in\_skill\_response = in\_skill\_product\_response(handler\_input)

product\_id = handler\_input.request\_envelope.request.payload.get("productId")

if in\_skill\_response:

product = [l for l in in\_skill\_response.in\_skill\_products

if l.product\_id == product\_id]

if handler\_input.request\_envelope.request.status.code == "200":

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

purchasable = product[0].purchasable

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = False

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = ("You have successfully cancelled your paid joke access.")

reprompt = "Ask for a free joke"

if purchase\_result == PurchaseResult.DECLINED.value:

if purchasable == PurchasableState.PURCHASABLE:

speech = ("You don't currently have paid joke access.")

else:

speech = "Ask for a free joke"

reprompt = "Ask for a free joke"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Connections.Response indicated failure. "

"Error: {}".format(

handler\_input.request\_envelope.request.status.message))

return handler\_input.response\_builder.speak(

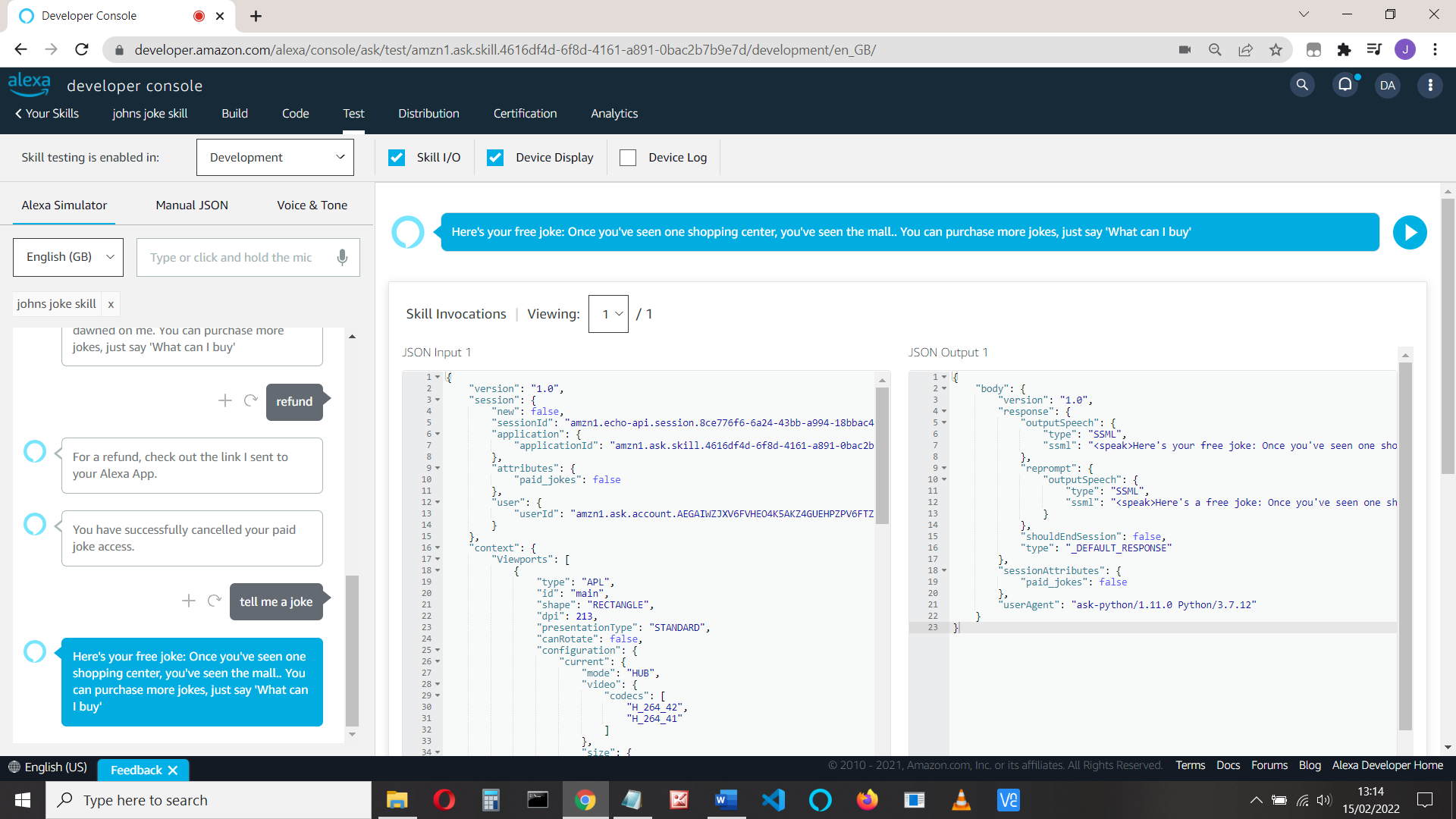
"There was an error handling your cancellation "

"request. Please try again or contact us for "

"help").response

Don’t forget to remove the paid\_joke session attribute

And here’s the response:



See the Alexa code if you have more than one purchased product (line 526):

<https://github.com/alexa-samples/skill-sample-python-fact-in-skill-purchases/blob/master/lambda/py/lambda_function.py>

For a NodeJS version, see:

<http://alexaskillstutorials.com/2020/01/23/make-money-with-alexa-in-skill-purchasing-2020/>

That’s it!

import random

def freejokes():

jokes = ["I was awake all night wondering where the sun went, but then it dawned on me",

"A burger walks into a bar. The bartender says 'Sorry, we don't serve food here'",

"Once you've seen one shopping center, you've seen the mall.",

"I didn't like my beard at first. Then it grew on me.",

"Time flies like an arrow. Fruit flies like a banana.",

"A backwards poet writes inverse.",

"To the guy who invented zero: Thanks for nothing!",

"A man walks into a bar. Ouch! It was an iron bar",

"What's brown and sticky? A stick."]

randNo = random.randint(0,len(jokes)-1)

return jokes[randNo]

def paidjokes():

jokes=["I'm reading a book about anti-gravity. I can't put it down.",

"Do you remember that joke I told you about my spine? It was about a weak back!",

"I just went to an emotional wedding. Even the cake was in tiers.",

"When's the best time to go to the dentist? Tooth-hurtie!",

"Why do seagulls fly over the sea? Because if they flew over the bay, they're bagels!",

"What do you call a farm that makes bad jokes? Corny!",

"Why do fish live in salt water? Because pepper makes them sneeze!",

"What kind of streets do ghosts haunt? Dead ends!",

"Why do you tell actors to break a leg? Because every play has a cast!",

"What kind of dogs love car racing? Lap dogs!",

"What did Winnie the Pooh say to his agent? 'Show me the honey!'",

"What do you call birds who stick together? Vel-crows.",

"Today I gave my dead batteries away. They were free of charge.",

"What do you call it when one cow spies on another? A steak out!"]

randNo = random.randint(0,len(jokes)-1)

return jokes[randNo]

**Change lines 89 and 163**

**product\_id = "amzn1.adg.product.b6c09ad1-da89-4c54-a924-7f10d5effb37"**

# -\*- coding: utf-8 -\*-

# This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK for Python.

# Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,

# session persistence, api calls, and more.

# This sample is built using the handler classes approach in skill builder.

import logging

import ask\_sdk\_core.utils as ask\_utils

import joke

from ask\_sdk\_core.skill\_builder import SkillBuilder

from ask\_sdk\_core.dispatch\_components import AbstractRequestHandler

from ask\_sdk\_core.dispatch\_components import AbstractExceptionHandler

from ask\_sdk\_core.handler\_input import HandlerInput

from ask\_sdk\_model import Response

logger = logging.getLogger(\_\_name\_\_)

logger.setLevel(logging.INFO)

from ask\_sdk.standard import StandardSkillBuilder

from ask\_sdk\_model.services.monetization import (

EntitledState, PurchasableState, InSkillProductsResponse, Error,

InSkillProduct)

from ask\_sdk\_model.interfaces.monetization.v1 import PurchaseResult

from ask\_sdk\_model import Response, IntentRequest

from ask\_sdk\_model.interfaces.connections import SendRequestDirective

skill\_name = "johns joke skill"

def in\_skill\_product\_response(handler\_input):

"""Get the In-skill product response from monetization service."""

# type: (HandlerInput) -> Union[InSkillProductsResponse, Error]

locale = handler\_input.request\_envelope.request.locale

ms = handler\_input.service\_client\_factory.get\_monetization\_service()

return ms.get\_in\_skill\_products(locale)

def get\_all\_entitled\_products(in\_skill\_product\_list):

"""Get list of in-skill products in ENTITLED state."""

# type: (List[InSkillProduct]) -> List[InSkillProduct]

entitled\_product\_list = [

l for l in in\_skill\_product\_list if (

l.entitled == EntitledState.ENTITLED)]

return entitled\_product\_list

def get\_speakable\_list\_of\_products(entitled\_products\_list):

"""Return product list in speakable form."""

# type: (List[InSkillProduct]) -> str

product\_names = [item.name for item in entitled\_products\_list]

if len(product\_names) > 1:

# If more than one, add and 'and' in the end

speech = " and ".join(

[", ".join(product\_names[:-1]), product\_names[-1]])

else:

# If one or none, then return the list content in a string

speech = ", ".join(product\_names)

return speech

class TellMeMoreHandler(AbstractRequestHandler):

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("TellMeMoreIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In TellMeMoreHandler")

# Tell the user about the product(s)

speak\_output = 'You can buy an upgrade to hear more jokes. Just say "Buy jokes" to purchase them.'

reprompt = "I didn't catch that. What can I help you with?"

return handler\_input.response\_builder.speak(speak\_output).ask(

reprompt).response

class BuyHandler(AbstractRequestHandler):

"""Handler for letting users buy the product.

User says: Alexa, purchase/buy <category>.

"""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("BuyIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyHandler")

product\_id = "amzn1.adg.product.b6c09ad1-da89-4c54-a924-7f10d5effb37"

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Buy",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="correlationToken")

).response

class BuyResponseHandler(AbstractRequestHandler):

"""This handles the Connections.Response event after a buy occurs."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Buy")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyResponseHandler")

#in\_skill\_response = in\_skill\_product\_response(handler\_input)

#product\_id = handler\_input.request\_envelope.request.payload.get("productId")

if handler\_input.request\_envelope.request.status.code == "200": # 200 is OK

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = "You have bought the access to more jokes, enjoy"

reprompt = "You can ask for more jokes"

elif purchase\_result in (

PurchaseResult.DECLINED.value,

PurchaseResult.ERROR.value,

PurchaseResult.NOT\_ENTITLED.value):

speech = "Thanks for your interest. Ask for another free joke"

reprompt = "Ask for a joke"

elif purchase\_result == PurchaseResult.ALREADY\_PURCHASED.value:

logger.info("Already purchased product")

speech = " You've already purchased more jokes, just ask for another joke"

reprompt = "Ask for a joke, help or exit"

else:

# Invalid purchase result value

logger.info("Purchase result: {}".format(purchase\_result))

speech = "Sorry, there was an error, please try again"

reprompt = "Ask for a joke, help or exit"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Error: {}".format(

handler\_input.request\_envelope.request.status.message))

return handler\_input.response\_builder.speak(

"There was an error handling your purchase request. "

"Please try again or ask for help")

class RefundPurchaseHandler(AbstractRequestHandler):

"""

Deal with reund requests

User says: Alexa, refund

"""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("RefundPurchase")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In RefundPurchaseHandler")

product\_id= 'amzn1.adg.product.b6c09ad1-da89-4c54-a924-7f10d5effb37'

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Cancel",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="correlationToken")

).response

class CancelResponseHandler(AbstractRequestHandler):

"""This handles the Connections.Response event after a cancel occurs."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Cancel")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In CancelResponseHandler")

in\_skill\_response = in\_skill\_product\_response(handler\_input)

product\_id = handler\_input.request\_envelope.request.payload.get("productId")

if in\_skill\_response:

product = [l for l in in\_skill\_response.in\_skill\_products

if l.product\_id == product\_id]

if handler\_input.request\_envelope.request.status.code == "200":

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

purchasable = product[0].purchasable

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = False

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = ("You have successfully cancelled your paid joke access.")

reprompt = "Ask for a free joke"

if purchase\_result == PurchaseResult.DECLINED.value:

if purchasable == PurchasableState.PURCHASABLE:

speech = ("You don't currently have paid joke access.")

else:

speech = "Ask for a free joke"

reprompt = "Ask for a free joke"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Connections.Response indicated failure. "

"Error: {}".format(

handler\_input.request\_envelope.request.status.message))

return handler\_input.response\_builder.speak(

"There was an error handling your cancellation "

"request. Please try again or contact us for "

"help").response

class LaunchRequestHandler(AbstractRequestHandler):

"""Handler for Skill Launch."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("LaunchRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Welcome, you can ask for a joke or say hello"

in\_skill\_response = in\_skill\_product\_response(handler\_input)

if isinstance(in\_skill\_response, InSkillProductsResponse):

entitled\_prods = get\_all\_entitled\_products(in\_skill\_response.in\_skill\_products)

if entitled\_prods and len(entitled\_prods) > 0:

session\_attr = handler\_input.attributes\_manager.session\_attributes

logger.info("session\_attr")

logger.info(session\_attr)

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

if entitled\_prods:

speak\_output = (

"Welcome to {}. You currently own {} products. "

"To hear a paid joke, say, 'Tell me a joke.'").format(

skill\_name,

get\_speakable\_list\_of\_products(entitled\_prods))

else:

logger.info("No entitled products")

speak\_output = (

"Welcome to {}. To hear a joke you can say "

"'Tell me a joke', or to hear about the paid for jokes "

"for purchase, say 'What can I buy'"

).format(skill\_name)

reprompt = "Sorry, please try again, say 'Tell me a joke'"

else:

logger.info("Error calling InSkillProducts API: {}".format(

in\_skill\_response.message))

speak\_output = "Something went wrong in loading your purchase history."

reprompt = speak\_output

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

class HelloWorldIntentHandler(AbstractRequestHandler):

"""Handler for Hello World Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("HelloWorldIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

ask\_output = "Here's a free joke: " + joke.freejokes()

speak\_output = "Here's your free joke: " + joke.freejokes() + ". You can purchase more jokes, just say 'What can I buy'"

session\_attr = handler\_input.attributes\_manager.session\_attributes

if "paid\_jokes" in session\_attr:

if session\_attr["paid\_jokes"] == True:

speak\_output = "Here's your paid joke: " +joke.paidjokes()

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(ask\_output)

.response

)

class HelpIntentHandler(AbstractRequestHandler):

"""Handler for Help Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("AMAZON.HelpIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "You can say hello to me! How can I help?"

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

class CancelOrStopIntentHandler(AbstractRequestHandler):

"""Single handler for Cancel and Stop Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return (ask\_utils.is\_intent\_name("AMAZON.CancelIntent")(handler\_input) or

ask\_utils.is\_intent\_name("AMAZON.StopIntent")(handler\_input))

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Goodbye!"

return (

handler\_input.response\_builder

.speak(speak\_output)

.response

)

class FallbackIntentHandler(AbstractRequestHandler):

"""Single handler for Fallback Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("AMAZON.FallbackIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In FallbackIntentHandler")

speech = "Hmm, I'm not sure. You can say Hello or Help. What would you like to do?"

reprompt = "I didn't catch that. What can I help you with?"

return handler\_input.response\_builder.speak(speech).ask(reprompt).response

class SessionEndedRequestHandler(AbstractRequestHandler):

"""Handler for Session End."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("SessionEndedRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

# Any cleanup logic goes here.

return handler\_input.response\_builder.response

class IntentReflectorHandler(AbstractRequestHandler):

"""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.

"""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("IntentRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

intent\_name = ask\_utils.get\_intent\_name(handler\_input)

speak\_output = "You just triggered " + intent\_name + "."

return (

handler\_input.response\_builder

.speak(speak\_output)

# .ask("add a reprompt if you want to keep the session open for the user to respond")

.response

)

class CatchAllExceptionHandler(AbstractExceptionHandler):

"""Generic error handling to capture any syntax or routing errors. If you receive an error

stating the request handler chain is not found, you have not implemented a handler for

the intent being invoked or included it in the skill builder below.

"""

def can\_handle(self, handler\_input, exception):

# type: (HandlerInput, Exception) -> bool

return True

def handle(self, handler\_input, exception):

# type: (HandlerInput, Exception) -> Response

logger.error(exception, exc\_info=True)

speak\_output = "Sorry, I had trouble doing what you asked. Please try again."

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

# The SkillBuilder object 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've

# defined are included below. The order matters - they're processed top to bottom.

#sb = SkillBuilder()

sb = StandardSkillBuilder()

sb.add\_request\_handler(TellMeMoreHandler())

sb.add\_request\_handler(BuyHandler())

sb.add\_request\_handler(BuyResponseHandler())

sb.add\_request\_handler(RefundPurchaseHandler())

sb.add\_request\_handler(CancelResponseHandler())

sb.add\_request\_handler(LaunchRequestHandler())

sb.add\_request\_handler(HelloWorldIntentHandler())

sb.add\_request\_handler(HelpIntentHandler())

sb.add\_request\_handler(CancelOrStopIntentHandler())

sb.add\_request\_handler(FallbackIntentHandler())

sb.add\_request\_handler(SessionEndedRequestHandler())

sb.add\_request\_handler(IntentReflectorHandler()) # make sure IntentReflectorHandler is last so it doesn't override your custom intent handlers

sb.add\_exception\_handler(CatchAllExceptionHandler())

lambda\_handler = sb.lambda\_handler()

References:

<https://github.com/alexa-samples/skill-sample-java-premium-hello-world/tree/master/lambda/custom/handlers>

<http://alexaskillstutorials.com/2020/01/23/make-money-with-alexa-in-skill-purchasing-2020/>