**Code Backup – 3rd November**

**Index.js**

// Lambda Function code for Alexa.

const Alexa = require("ask-sdk-core");

const https = require("https");

const invocationName = "intent fitness";

const MAX\_NUMBER = 14;

const tips = [

"to ensure good health, eat lightly, breathe deeply, live moderately, cultivate cheerfulness, and maintain an interest in life.",

"to keep the body in good health is a duty, otherwise we will not be able to keep the mind strong and clear.",

"do something today that your future self will thank you for.",

"exercising before going to bed makes your muscles burn more calories during the night.",

"a mid-day nap improves your memory and reduces the chances of developing heart diseases.",

"if you\’re feeling anxious and stressed, eat a melon. Melons help relieve anxiety and stress, plus they boost your metabolism.",

"drink two cups of cold water before a meal, as this boosts metabolism by up to 30%.",

"drinking a lot of water during the day helps you sleep better at night.",

"a cold shower can help relieve depression and also helps keep your skin and hair healthier.",

"the first day of the week defines your thinking patterns for the rest of the week. It\’s best to exercise on Mondays to emsure a healthy routine.",

"your body can stand almost anything. It\’s your mind that you have to convince.",

"fitness is like a relationship. You can\’t cheat and expect it to work.",

"do something today that your future self will thank you for.",

"life begins at the end of your comfort zone.",

"the difference between try and triumph is a little \‘umph\’.",

"don\’t count the days, make the days count.",

"when you feel like quitting, think about why you started.",

"all progress takes place outside the comfort zone.",

"success starts with self-discipline.",

"the only bad workout is the one that didn\’t happen.",

"rome wasn\’t built in a day. Work hard and good results will come.",

"motivation is what gets you started. Habit is what keeps you going.",

"fitness is not about being better than someone else. It\’s about being better than you used to be.",

"the pain you feel today will be the strength you feel tomorrow.",

"age is no barrier. It\’s a limitation you put on your mind."

]

const STREAMS = [

{

token: '1',

//url: 'https://download1327.mediafire.com/xw28a5f719ig/dea2szday442gnb/Birds.mp3',

url: 'https://download941.mediafire.com/so8m4wbvztyg/p2lvgfh4ynfkrwx/muscle-car-daniel\_simon.mp3',

metadata: {

title: 'Birds',

},

},

{

token: '2',

//url: 'https://download1494.mediafire.com/7jy1o24w3iag/odxviu1pg24wi0z/Birds+2.mp3',

url: 'https://download941.mediafire.com/so8m4wbvztyg/p2lvgfh4ynfkrwx/muscle-car-daniel\_simon.mp3',

metadata: {

title: 'Birds 2',

},

},

{

token: '3',

url: 'https://download1335.mediafire.com/3dinwb650p4g/t0rpb6khshdv8pf/Fire+1.mp3',

metadata: {

title: 'Fire',

},

},

{

token: '4',

url: 'https://download1506.mediafire.com/38ajgcmfe4tg/3aezcb75yxyv0gk/Fire+2.mp3',

metadata: {

title: 'Fire 2',

},

},

{

token: '5',

url: 'https://download1638.mediafire.com/nhwda28clyug/1t2rtlkh3ghk2hm/Forrest+1.mp3',

metadata: {

title: 'Trees',

},

},

{

token: '6',

url: 'https://download1519.mediafire.com/1822l4rgf0ug/xcldhycn48fgw85/Forrest+2.mp3',

metadata: {

title: 'Trees 2',

},

},

{

token: '7',

url: 'https://download1583.mediafire.com/jg5i83zw01tg/us7sb9fj17qd2nf/Ocean+Waves+1.mp3',

metadata: {

title: 'Ocean Waves',

},

},

{

token: '8',

url: 'https://download1501.mediafire.com/fjfd50n73vvg/4ank75v7eo4cnun/Ocean+Waves+2.mp3',

metadata: {

title: 'Ocean Waves 2',

},

},

{

token: '9',

url: 'https://download1484.mediafire.com/2t78edjdbdwg/8nk28st9oibqoaq/Rain+on+Car+1.mp3',

metadata: {

title: 'Rain on Car',

},

},

{

token: '10',

url: 'https://download855.mediafire.com/ilubi47j22yg/zc0ia23wvv9bm4w/Rain+on+Car+2.mp3',

metadata: {

title: 'Rain on Car 2',

},

},

{

token: '11',

url: 'https://download1339.mediafire.com/lcrcionewpng/gix0xd945tc3zht/Rain.mp3',

metadata: {

title: 'Rain',

},

},

{

token: '12',

url: 'https://download1081.mediafire.com/p29hcafdiqxg/5a3yikc4prthxlz/Rain+2.mp3',

metadata: {

title: 'Rain 2',

},

},

{

token: '13',

url: 'https://download1483.mediafire.com/49330qa74irg/ifzy7zutfupe4o3/Tranquility.mp3',

metadata: {

title: 'Tranquility',

},

},

{

token: '14',

url: 'https://download1581.mediafire.com/i6p4vcig52zg/a26kn7s6qw2j1ym/Cruising+Along.mp3',

metadata: {

title: 'Cruising Along',

},

},

{

token: '15',

url: 'https://download1077.mediafire.com/n6z03pb96seg/nv2zhq1ccj7co8m/Tropical.mp3',

metadata: {

title: 'Tropical',

},

},

{

token: '16',

url: 'https://download1586.mediafire.com/xig9s1jgwmmg/d5v6a1jz0iw3g0u/Breathing.mp3',

metadata: {

title: 'Breathing',

},

},

{

token: '17',

url: 'https://download1475.mediafire.com/asydna9x2nyg/zzxapxc6zvadgod/Breathing+2.mp3',

metadata: {

title: 'Breathing 2',

},

},

{

token: '18',

url: 'https://download1510.mediafire.com/jb9dg74aaz2g/4m3fuajzozemyse/Breathing+3.mp3',

metadata: {

title: 'Breathing 3',

},

},

];

// 1. Intent Handlers =============================================

const AMAZON\_CancelIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'AMAZON.CancelIntent' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

const ranTips = Math.floor((Math.random() \* 24) + 0);

let say = 'Thank you for using ' + invocationName + '. Remember, ' + tips[ranTips] + ' Have a great day!';

handlerInput.responseBuilder

.addAudioPlayerClearQueueDirective('CLEAR\_ALL')

.addAudioPlayerStopDirective();

return responseBuilder

.speak(say)

.withShouldEndSession(true)

.getResponse();

},

};

const AMAZON\_HelpIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'AMAZON.HelpIntent' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

let history = sessionAttributes['history'];

let say = 'Sure thing! I can help you. ';

let previousIntent = getPreviousIntent(sessionAttributes);

if (previousIntent && !handlerInput.requestEnvelope.session.new) {

say += 'You can choose to either meditate, allowing you to listen to relaxing music or complete a guided breathing session. Otherwise, you can choose to complete a quick 10-minute workout. Which will it be?';

}

return responseBuilder

.speak(say)

.reprompt(say)

.getResponse();

},

};

const PlaybackStoppedIntentHandler = {

canHandle(handlerInput) {

return handlerInput.requestEnvelope.request.type === 'PlaybackController.PauseCommandIssued'

|| handlerInput.requestEnvelope.request.type === 'AudioPlayer.PlaybackStopped';

},

handle(handlerInput) {

handlerInput.responseBuilder

.addAudioPlayerClearQueueDirective('CLEAR\_ALL')

.addAudioPlayerStopDirective();

let sayClose = '';

return handlerInput.responseBuilder

.getResponse();

},

};

const PlaybackStartedIntentHandler = {

canHandle(handlerInput) {

return handlerInput.requestEnvelope.request.type === 'AudioPlayer.PlaybackStarted';

},

handle(handlerInput) {

handlerInput.responseBuilder

.addAudioPlayerClearQueueDirective('CLEAR\_ENQUEUED');

return handlerInput.responseBuilder

.getResponse();

},

};

const AMAZON\_StopIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'AMAZON.StopIntent' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

handlerInput.responseBuilder

.addAudioPlayerClearQueueDirective('CLEAR\_ALL')

.addAudioPlayerStopDirective();

const ranTips = Math.floor((Math.random() \* 24) + 0);

let say = 'Thank you for using ' + invocationName + '. Remember, ' + tips[ranTips] + ' Have a great day!';

return responseBuilder

.speak(say)

.withShouldEndSession(true)

.getResponse();

},

};

const AMAZON\_NavigateHomeIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'AMAZON.NavigateHomeIntent' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

let say = 'Hello from AMAZON.NavigateHomeIntent. ';

return responseBuilder

.speak(say)

.reprompt('try again, ' + say)

.getResponse();

},

};

const AMAZON\_FallbackIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'AMAZON.FallbackIntent' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

let previousSpeech = getPreviousSpeechOutput(sessionAttributes);

return responseBuilder

.speak('Sorry I didnt catch what you said, ')

.reprompt('You can get started with a quick 10 minute workout, unwind with 10 minutes of meditation or follow a 10 minute guided breathing session. Which will it be?')

.getResponse();

},

};

const getSounds\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'getSounds' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

const ranTips = Math.floor((Math.random() \* 24) + 0);

const ranSong = Math.floor((Math.random() \* MAX\_NUMBER) + 0);

const ranBirds = Math.floor((Math.random() \* 1) + 0);

const ranFire = Math.floor((Math.random() \* 3) + 2);

const ranTrees = Math.floor((Math.random() \* 5) + 4);

const ranOcean = Math.floor((Math.random() \* 7) + 6);

const ranRainonCar = Math.floor((Math.random() \* 9) + 8);

const ranRain = Math.floor((Math.random() \* 11) + 10);

const ranLoFi = Math.floor((Math.random() \* 14) + 12);

const sBirds = STREAMS[ranBirds];

const sFire = STREAMS[ranFire];

const sTrees = STREAMS[ranTrees];

const sOcean = STREAMS[ranOcean];

const sRainonCar = STREAMS[ranRainonCar];

const sRain = STREAMS[ranRain];

const sLoFi = STREAMS[ranLoFi];

const sRandom = STREAMS[ranSong];

// delegate to Alexa to collect all the required slots

const currentIntent = request.intent;

if (request.dialogState && request.dialogState !== 'COMPLETED') {

return handlerInput.responseBuilder

.addDelegateDirective(currentIntent)

.getResponse();

}

let say = '';

let slotValues = getSlotValues(request.intent.slots);

let sayClose = 'Thank you for using ' + invocationName + '. Remember, ' + tips[ranTips] + ' Have a great day!';

if(slotValues && slotValues.sound){

if (slotValues.sound.heardAs == 'birds' || slotValues.sound.heardAs == 'bird'){

say = 'Great choice! I will play bird sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sBirds.url, sBirds.token, 0, null, sBirds.metadata);

}

else if(slotValues.sound.heardAs == 'fire' || slotValues.sound.heardAs == 'crackling'){

say = 'Great choice! I will play fire sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sFire.url, sFire.token, 0, null, sFire.metadata);

}

else if (slotValues.sound.heardAs == 'trees' || slotValues.sound.heardAs == 'tree'){

say = 'Great choice! I will play tree sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sTrees.url, sTrees.token, 0, null, sTrees.metadata);

}

else if(slotValues.sound.heardAs == 'ocean waves' || slotValues.sound.heardAs == 'ocean' || slotValues.sound.heardAs == 'waves' || slotValues.sound.heardAs == 'ocean wave'){

say = 'Great choice! I will play ocean sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sOcean.url, sOcean.token, 0, null, sOcean.metadata);

}

else if(slotValues.sound.heardAs == 'rain on car' || slotValues.sound.heardAs == 'raining on a car' || slotValues.sound.heardAs == 'rain on a car' || slotValues.sound.heardAs == 'raining on car'){

say = 'Great choice! I will play rain on a car sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sRainonCar.url, sRainonCar.token, 0, null, sRainonCar.metadata);

}

else if(slotValues.sound.heardAs == 'rain' || slotValues.sound.heardAs == 'rain drop' || slotValues.sound.heardAs == 'rain drops'){

say = 'Great choice! I will play rain sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sRain.url, sRain.token, 0, null, sRain.metadata);

}

else if(slotValues.sound.heardAs == 'chill' || slotValues.sound.heardAs == 'chilled' || slotValues.sound.heardAs == 'relaxing'){

say = 'Great choice! I will play chilled lo-fi sounds for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sLoFi.url, sLoFi.token, 0, null, sLoFi.metadata);

}

else if (slotValues.sound.heardAs == 'random'){

say = 'Great choice! I will choose a random relaxing sound to play for you.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sRandom.url, sRandom.token, 0, null, sRandom.metadata);

}

}

return responseBuilder

.speak(say)

.getResponse();

// .speak(sayClose);

},

};

const getMode\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && request.intent.name === 'getMode' ;

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

const ranBreathing = Math.floor((Math.random() \* 17) + 15);

const sBreathing = STREAMS[ranBreathing];

// delegate to Alexa to collect all the required slots

const currentIntent = request.intent;

if (request.dialogState && request.dialogState !== 'COMPLETED') {

return handlerInput.responseBuilder

.addDelegateDirective(currentIntent)

.getResponse();

}

let say = '';

let slotValues = getSlotValues(request.intent.slots);

if(slotValues && slotValues.mode){

if (slotValues.mode.heardAs == 'meditation' ||slotValues.mode.heardAs == 'meditate'||slotValues.mode.heardAs == 'meditating'||slotValues.mode.heardAs == 'mindfulness' ){

say = 'Sure! We will get started with a meditation session. You can choose to listen to sounds such as birds, rain and fire. You can also say random and I will choose a sound for you. Which will it be?';

}

else if(slotValues.mode.heardAs == 'workout' || slotValues.mode.heardAs == 'exercise'){

say = 'Sure! We will get started with a workout session. The rest of this section has still to be implemented';

}

else if(slotValues.mode.heardAs == 'guided breathing' || slotValues.mode.heardAs == 'breathing'){

say = 'Sure! We will get started with a guided breathing session.';

handlerInput.responseBuilder

.addAudioPlayerPlayDirective('REPLACE\_ALL', sBreathing.url, sBreathing.token, 0, null, sBreathing.metadata);

}

}

return responseBuilder

.speak(say)

.reprompt('I didn\'t quite get that. You can get started with a quick 10 minute workout, unwind with 10 minutes of meditation or follow a 10 minute guided breathing session. Which will it be?')

.getResponse();

},

};

const LaunchRequest\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'LaunchRequest';

},

handle(handlerInput) {

const responseBuilder = handlerInput.responseBuilder;

let say = 'Hello' + ' and welcome to ' + invocationName + '! You can get started with a quick 10 minute workout, unwind with 10 minutes of meditation or follow a 10 minute guided breathing session. Which will it be?';

return responseBuilder

.speak(say)

.reprompt('I didn\'t quite get that. You can get started with a quick 10 minute workout, unwind with 10 minutes of meditation or follow a 10 minute guided breathing session. Which will it be?')

.getResponse();

},

};

const SessionEndedHandler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'SessionEndedRequest';

},

handle(handlerInput) {

console.log(`Session ended with reason: ${handlerInput.requestEnvelope.request.reason}`);

return handlerInput.responseBuilder.getResponse();

}

};

const ErrorHandler = {

canHandle() {

return true;

},

handle(handlerInput, error) {

const request = handlerInput.requestEnvelope.request;

console.log(`Error handled: ${error.message}`);

return handlerInput.responseBuilder

.speak(`Sorry, your skill got this error. ${error.message} `)

.reprompt(`Sorry, your skill got this error. ${error.message} `)

.getResponse();

}

};

function getPreviousSpeechOutput(attrs) {

if (attrs.lastSpeechOutput && attrs.history.length > 1) {

return attrs.lastSpeechOutput;

} else {

return false;

}

}

function getPreviousIntent(attrs) {

if (attrs.history && attrs.history.length > 1) {

return attrs.history[attrs.history.length - 2].IntentRequest;

} else {

return false;

}

}

function getSlotValues(filledSlots) {

const slotValues = {};

Object.keys(filledSlots).forEach((item) => {

const name = filledSlots[item].name;

if (filledSlots[item] &&

filledSlots[item].resolutions &&

filledSlots[item].resolutions.resolutionsPerAuthority[0] &&

filledSlots[item].resolutions.resolutionsPerAuthority[0].status &&

filledSlots[item].resolutions.resolutionsPerAuthority[0].status.code) {

switch (filledSlots[item].resolutions.resolutionsPerAuthority[0].status.code) {

case 'ER\_SUCCESS\_MATCH':

slotValues[name] = {

heardAs: filledSlots[item].value,

resolved: filledSlots[item].resolutions.resolutionsPerAuthority[0].values[0].value.name,

ERstatus: 'ER\_SUCCESS\_MATCH'

};

break;

case 'ER\_SUCCESS\_NO\_MATCH':

slotValues[name] = {

heardAs: filledSlots[item].value,

resolved: '',

ERstatus: 'ER\_SUCCESS\_NO\_MATCH'

};

break;

default:

break;

}

} else {

slotValues[name] = {

heardAs: filledSlots[item].value || '', // may be null

resolved: '',

ERstatus: ''

};

}

}, this);

return slotValues;

}

const RequestPersistenceInterceptor = {

process(handlerInput) {

if(handlerInput.requestEnvelope.session['new']) {

return new Promise((resolve, reject) => {

handlerInput.attributesManager.getPersistentAttributes()

.then((sessionAttributes) => {

sessionAttributes = sessionAttributes || {};

sessionAttributes['launchCount'] += 1;

handlerInput.attributesManager.setSessionAttributes(sessionAttributes);

handlerInput.attributesManager.savePersistentAttributes()

.then(() => {

resolve();

})

.catch((err) => {

reject(err);

});

});

});

}

}

};

const ResponsePersistenceInterceptor = {

process(handlerInput, responseOutput) {

const ses = (typeof responseOutput.shouldEndSession == "undefined" ? true : responseOutput.shouldEndSession);

if(ses || handlerInput.requestEnvelope.request.type == 'SessionEndedRequest') { // skill was stopped or timed out

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

sessionAttributes['lastUseTimestamp'] = new Date(handlerInput.requestEnvelope.request.timestamp).getTime();

handlerInput.attributesManager.setPersistentAttributes(sessionAttributes);

return new Promise((resolve, reject) => {

handlerInput.attributesManager.savePersistentAttributes()

.then(() => {

resolve();

})

.catch((err) => {

reject(err);

});

});

}

}

};

// 4. Exports handler function and setup ===================================================

const skillBuilder = Alexa.SkillBuilders.custom();

exports.handler = skillBuilder

.addRequestHandlers(

AMAZON\_CancelIntent\_Handler,

AMAZON\_HelpIntent\_Handler,

AMAZON\_StopIntent\_Handler,

AMAZON\_NavigateHomeIntent\_Handler,

AMAZON\_FallbackIntent\_Handler,

getSounds\_Handler,

getMode\_Handler,

LaunchRequest\_Handler,

SessionEndedHandler

)

.addErrorHandlers(ErrorHandler)

.lambda();

**JSON File**

{

"interactionModel": {

"languageModel": {

"invocationName": "intent fitness",

"modelConfiguration": {

"fallbackIntentSensitivity": {

"level": "LOW"

}

},

"intents": [

{

"name": "AMAZON.CancelIntent",

"samples": []

},

{

"name": "AMAZON.HelpIntent",

"samples": []

},

{

"name": "AMAZON.StopIntent",

"samples": []

},

{

"name": "AMAZON.NavigateHomeIntent",

"samples": []

},

{

"name": "AMAZON.FallbackIntent",

"samples": []

},

{

"name": "getSounds",

"slots": [

{

"name": "sound",

"type": "SOUNDS",

"samples": [

"play {sound} sounds for me",

"i want to listen to {sound} sounds",

"{sound}",

"i want to listen to {sound}",

"i would like to listen to {sound}",

"play {sound} for me",

"play {sound} sounds",

"play {sound}"

]

}

],

"samples": [

"play {sound} sounds for me",

"play {sound} sounds",

"play {sound} ",

"play {sound} for me",

"I would like to listen to {sound}",

"relax me",

"I want to listen to {sound}",

"{sound}",

"I want to listen to {sound} sounds"

]

},

{

"name": "PlayStreamIntent",

"slots": [],

"samples": [

"to play",

"to start playing",

"to begin playing"

]

},

{

"name": "AboutIntent",

"slots": [],

"samples": [

"about",

"what is this",

"tell me about this"

]

},

{

"name": "AMAZON.PauseIntent",

"samples": []

},

{

"name": "AMAZON.ResumeIntent",

"samples": []

},

{

"name": "AMAZON.LoopOnIntent",

"samples": []

},

{

"name": "AMAZON.NextIntent",

"samples": []

},

{

"name": "AMAZON.PreviousIntent",

"samples": []

},

{

"name": "AMAZON.RepeatIntent",

"samples": []

},

{

"name": "AMAZON.ShuffleOnIntent",

"samples": []

},

{

"name": "AMAZON.StartOverIntent",

"samples": []

},

{

"name": "AMAZON.LoopOffIntent",

"samples": []

},

{

"name": "AMAZON.ShuffleOffIntent",

"samples": []

},

{

"name": "getMode",

"slots": [

{

"name": "mode",

"type": "MODE",

"samples": [

"i want to {mode}",

"i would like to {mode}",

"{mode}"

]

}

],

"samples": [

"i want to {mode}",

"{mode}",

"I would like to {mode}"

]

}

],

"types": [

{

"name": "SOUNDS",

"values": [

{

"name": {

"value": "trees",

"synonyms": [

"tree"

]

}

},

{

"name": {

"value": "chill",

"synonyms": [

"chilled",

"relaxing"

]

}

},

{

"name": {

"value": "random"

}

},

{

"name": {

"value": "rain on car",

"synonyms": [

"raining on a car",

"rain on a car",

"raining on car"

]

}

},

{

"name": {

"value": "rain",

"synonyms": [

"rain drop",

"rain drops"

]

}

},

{

"name": {

"value": "ocean waves",

"synonyms": [

"ocean wave",

"waves",

"ocean"

]

}

},

{

"name": {

"value": "fire",

"synonyms": [

"crackling"

]

}

},

{

"name": {

"value": "birds",

"synonyms": [

"bird"

]

}

}

]

},

{

"name": "MODE",

"values": [

{

"name": {

"value": "guided breathing",

"synonyms": [

"breathing"

]

}

},

{

"name": {

"value": "meditate",

"synonyms": [

"mindfulness",

"meditating",

"meditation"

]

}

},

{

"name": {

"value": "workout",

"synonyms": [

"exercise"

]

}

}

]

}

]

},

"dialog": {

"intents": [

{

"name": "getSounds",

"confirmationRequired": false,

"prompts": {},

"slots": [

{

"name": "sound",

"type": "SOUNDS",

"confirmationRequired": false,

"elicitationRequired": true,

"prompts": {

"elicitation": "Elicit.Slot.1518147732038.1393534478204"

},

"validations": [

{

"type": "hasEntityResolutionMatch",

"prompt": "Slot.Validation.1059701369172.183985370.529286722304"

}

]

}

]

},

{

"name": "getMode",

"confirmationRequired": false,

"prompts": {},

"slots": [

{

"name": "mode",

"type": "MODE",

"confirmationRequired": false,

"elicitationRequired": true,

"prompts": {

"elicitation": "Elicit.Slot.757103645059.1413855593511"

},

"validations": [

{

"type": "hasEntityResolutionMatch",

"prompt": "Slot.Validation.1357825314305.831302278205.1243342547567"

}

]

}

]

}

],

"delegationStrategy": "ALWAYS"

},

"prompts": [

{

"id": "Slot.Validation.1059701369172.183985370.529286722304",

"variations": [

{

"type": "PlainText",

"value": "I'm sorry, I don't know that sound. You can ask for things such as birds, rain or fire sounds."

}

]

},

{

"id": "Slot.Validation.1357825314305.831302278205.1243342547567",

"variations": [

{

"type": "PlainText",

"value": "I'm sorry, I don't know that. You can choose to either meditate or workout. Which will it be?"

}

]

},

{

"id": "Elicit.Slot.1518147732038.1393534478204",

"variations": [

{

"type": "PlainText",

"value": "You can choose to listen to sounds such as birds, rain and fire. You can also say random and I will choose a sound for you. Which will it be?"

}

]

},

{

"id": "Elicit.Slot.757103645059.1413855593511",

"variations": [

{

"type": "PlainText",

"value": "You can get started with a quick ten minute workout, unwind with ten minutes of meditation or follow a ten minute guided breathing session. Which will it be?"

}

]

}

]

}

}