**Code Backup – 11th 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 = 13;

const MAX\_TIPS = 24;

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 ensure 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: '21',

url: 'https://dl.dropboxusercontent.com/s/kptf5nqhpi7jgxc/Birds%201.mp3?dl=1',

metadata: {

title: 'Birds',

subtitle: 'Unwind with relaxing bird sounds.',

art: {

sources: [

{

contentDescription: 'Yellow Bird on Sakura Tree',

url: 'https://dl.dropboxusercontent.com/s/bace03559m40xgp/Birds%201%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

},

],

},

backgroundImage: {

sources: [

{

contentDescription: 'Peaceful Beach Setting',

url: 'https://dl.dropboxusercontent.com/s/x09l1kz72e25sj0/Birds%201%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

},

],

},

},

},

{

token: '22',

url: 'https://dl.dropboxusercontent.com/s/d56m739pc61hr1k/Birds%202.mp3?dl=1',

metadata: {

title: 'Birds',

subtitle: 'Unwind with relaxing bird sounds',

art: {

sources: [

{

contentDescription: 'Blue, Yellow & Green Macaw Parrots',

url: 'https://dl.dropboxusercontent.com/s/yjudmzklumsed8u/Birds%202%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

},

],

},

backgroundImage: {

sources: [

{

contentDescription: 'Peaceful Beach Setting',

url: 'https://dl.dropboxusercontent.com/s/cdqalm4yndc4tl4/Birds%202%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

},

],

},

},

},

{

token: '23',

url: 'https://dl.dropboxusercontent.com/s/i58goxsymdtzu57/Fire%201.mp3?dl=1',

metadata: {

title: 'Fire',

subtitle: 'Unwind with relaxing fire sounds.',

art: {

sources: [{

contentDescription: 'Person Standing in Front of Fire',

url: 'https://dl.dropboxusercontent.com/s/yzt60blskso3o4v/Fire%201%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Campfire on a Peaceful Night',

url: 'https://dl.dropboxusercontent.com/s/fx9luegdkef41u0/Fire%201%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '24',

url: 'https://dl.dropboxusercontent.com/s/q9ls8jyoiluvtqe/Fire%202.mp3?dl=1',

metadata: {

title: 'Fire',

subtitle: 'Unwind with relaxing fire sounds.',

art: {

sources: [{

contentDescription: 'Campfire on Beach',

url: 'https://dl.dropboxusercontent.com/s/dpla0hanksge31f/Fire%202%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Large Campfire',

url: 'https://dl.dropboxusercontent.com/s/eerj64fh6efrx5l/Fire%202%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '25',

url: 'https://dl.dropboxusercontent.com/s/yk1qlx2b2dkxuyz/Ocean%20Waves%201.mp3?dl=1',

metadata: {

title: 'Ocean Waves',

subtitle: 'Unwind with relaxing ocean sounds.',

art: {

sources: [{

contentDescription: 'Underwater Coral & Fish',

url: 'https://dl.dropboxusercontent.com/s/kwby3v5ze5i5wzj/Ocean%20Waves%201%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Peaceful Beach Setting',

url: 'https://dl.dropboxusercontent.com/s/5ib3qpk7lfbv0or/Ocean%20Waves%201%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '26',

url: 'https://dl.dropboxusercontent.com/s/o5t1j40alg4hy17/Ocean%20Waves%202.mp3?dl=1',

metadata: {

title: 'Ocean Waves',

subtitle: 'Unwind with relaxing ocean sounds.',

art: {

sources: [{

contentDescription: 'Person Floating in Clear Sea',

url: 'https://dl.dropboxusercontent.com/s/rkib3k4d47sfn30/Ocean%20Waves%202%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Cathedral Cove, New Zealand',

url: 'https://dl.dropboxusercontent.com/s/9eqqtafdp49q4zk/Ocean%20Waves%202%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '27',

url: 'https://dl.dropboxusercontent.com/s/fx3lv0auqtjxzj1/Rain%20on%20Car%201.mp3?dl=1',

metadata: {

title: 'Rain on a Car',

subtitle: 'Unwind with relaxing rain sounds.',

art: {

sources: [{

contentDescription: 'Raindrops on Window',

url: 'https://dl.dropboxusercontent.com/s/7de0ugrzhr814s5/Rain%20on%20Car%201%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Raindrops on Car Window & Wingmiror',

url: 'https://dl.dropboxusercontent.com/s/4qixiqplj8brnc3/Rain%20on%20Car%201%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '28',

url: 'https://dl.dropboxusercontent.com/s/ycfvszir2mpcsow/Rain%20on%20Car%202.mp3?dl=1',

metadata: {

title: 'Rain on a Car',

subtitle: 'Unwind with relaxing rain sounds.',

art: {

sources: [{

contentDescription: 'Raining in a Busy City Centre',

url: 'https://dl.dropboxusercontent.com/s/crtbd66hyl9s7h6/Rain%20on%20Car%202%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Person Holding an Umbrealla in the Rain',

url: 'https://dl.dropboxusercontent.com/s/jbtql4ziklq5zal/Rain%20on%20Car%202%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '29',

url: 'https://dl.dropboxusercontent.com/s/59z42f7ja3k5svb/Rain%201.mp3?dl=1',

metadata: {

title: 'Rain',

subtitle: 'Unwind with relaxing rain sounds.',

art: {

sources: [{

contentDescription: 'Raining in a Busy City Centre',

url: 'https://dl.dropboxusercontent.com/s/bgl88dh5ios65ox/Rain%201%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Raindrops on a House Window',

url: 'https://dl.dropboxusercontent.com/s/8neu6qhwiek3ghg/Rain%201%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '30',

url: 'https://dl.dropboxusercontent.com/s/s0ikdxcsh3b6168/Rain%202.mp3?dl=1',

metadata: {

title: 'Rain',

subtitle: 'Unwind with relaxing rain sounds.',

art: {

sources: [{

contentDescription: 'Person Holding an Umbrealla in the Rain',

url: 'https://dl.dropboxusercontent.com/s/rp81ukbi9rovkse/Rain%202%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Person Holding an Umbrealla in the Rain',

url: 'https://dl.dropboxusercontent.com/s/g92gt4gv4uuy21u/Rain%202%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '31',

url: 'https://dl.dropboxusercontent.com/s/c69uwsrizzcjoqk/Tranquility.mp3?dl=1',

metadata: {

title: 'Tranquility',

subtitle: 'Unwind with lo-fi beats.',

art: {

sources: [{

contentDescription: 'Peaceful Costal Setting',

url: 'https://dl.dropboxusercontent.com/s/gjpl9w3081xv2dy/Tranquility%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Neon Sign at Dock',

url: 'https://dl.dropboxusercontent.com/s/4zys450vwttq21d/Tranquility%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '32',

url: 'https://dl.dropboxusercontent.com/s/t70kxv5wnzy5qfa/Cruising%20Along.mp3?dl=1',

metadata: {

title: 'Cruising Along',

subtitle: 'Unwind with lo-fi beats.',

art: {

sources: [{

contentDescription: 'Person Walking Along Peaceful Road',

url: 'https://dl.dropboxusercontent.com/s/0gtje9gp927pzhz/Cruising%20Along%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'City Landscape from Afar',

url: 'https://dl.dropboxusercontent.com/s/6ybrf5u3gi4mvle/Cruising%20Along%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '33',

url: 'https://dl.dropboxusercontent.com/s/cx8ja7t3y8uwoqj/Tropical.mp3?dl=1',

metadata: {

title: 'Tropical',

subtitle: 'Unwind with lo-fi beats.',

art: {

sources: [{

contentDescription: 'Person Walking Along Peaceful Road',

url: 'https://dl.dropboxusercontent.com/s/0gtje9gp927pzhz/Cruising%20Along%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'City Landscape from Afar',

url: 'https://dl.dropboxusercontent.com/s/6ybrf5u3gi4mvle/Cruising%20Along%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '34',

url: 'https://dl.dropboxusercontent.com/s/0ntgigwq9jse668/Mindfulness%20of%20Breath.mp3?dl=1',

metadata: {

title: 'Mindfulness of Breath',

subtitle: 'Unwind with guided breathing.',

art: {

sources: [{

contentDescription: 'Person Meditating on a Beach',

url: 'https://dl.dropboxusercontent.com/s/91xamz50kw2hq07/Mindfulness%20of%20Breath%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Peaceful Mountain Setting',

url: 'https://dl.dropboxusercontent.com/s/64cx88vu84h8vgp/Mindfulness%20of%20Breath%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

{

token: '35',

url: 'https://dl.dropboxusercontent.com/s/ns4qqy36rzolnvq/Mountain%20Meditation.mp3?dl=1',

metadata: {

title: 'Mountain Meditation',

subtitle: 'Unwind with guided breathing.',

art: {

sources: [{

contentDescription: 'Person Looking Out Onto Mountains',

url: 'https://dl.dropboxusercontent.com/s/3hgvgt7p993i4hl/Mountain%20Meditation%20Art.jpg?dl=1',

widthPixels: 512,

heightPixels: 512,

}, ],

},

backgroundImage: {

sources: [{

contentDescription: 'Peaceful Mountain Setting',

url: 'https://dl.dropboxusercontent.com/s/p2gziee71smrb6a/Mountain%20Meditation%20Back.jpg?dl=1',

widthPixels: 1200,

heightPixels: 800,

}, ],

},

},

},

];

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

const AMAZON\_CancelIntent\_Handler = {

canHandle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

return request.type === 'IntentRequest' && (request.intent.name === 'AMAZON.CancelIntent' || request.intent.name === 'AMAZON.StopIntent');

},

handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

const ranTips = Math.floor(Math.random() \* MAX\_TIPS);

let say = 'Thank you for using ' + invocationName + '. You can schedule a reminder to workout or meditate at this time tomorrow by saying "remind me to exercise with Intent Fitness". 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, I can help. This app allows you to 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()

.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();

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 CreateReminderIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest' &&

Alexa.getIntentName(handlerInput.requestEnvelope) === 'getReminder';

},

async handle(handlerInput) {

const request = handlerInput.requestEnvelope.request;

const responseBuilder = handlerInput.responseBuilder;

let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

// 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.reminder) {

const reminderApiClient = handlerInput.serviceClientFactory.getReminderManagementServiceClient(),

{ permissions } = handlerInput.requestEnvelope.context.System.user;

if (!permissions) {

return handlerInput.responseBuilder

.speak("Please go to the Alexa mobile app to grant reminders permissions.")

.withAskForPermissionsConsentCard(['alexa::alerts:reminders:skill:readwrite'])

.getResponse()

}

const reminderRequestWorkout = {

"trigger": {

"type": "SCHEDULED\_RELATIVE",

"offsetInSeconds": "15"

},

"alertInfo": {

"spokenInfo": {

"content": [{

"locale": "en-GB",

"text": "Intent Fitness: Remember to exercise today!",

"ssml": "<speak> Remember to exercise today with Intent Fitness!</speak>"

}]

}

},

"pushNotification": {

"status": "ENABLED"

}

}

const reminderRequestMeditate = {

"trigger": {

"type": "SCHEDULED\_RELATIVE",

"offsetInSeconds": "15"

},

"alertInfo": {

"spokenInfo": {

"content": [{

"locale": "en-GB",

"text": "Intent Fitness: Remember to meditate today!",

"ssml": "<speak> Remember to meditate today with Intent Fitness!</speak>"

}]

}

},

"pushNotification": {

"status": "ENABLED"

}

}

if (slotValues.reminder.heardAs == 'meditate' || slotValues.reminder.heardAs == 'meditation' || slotValues.reminder.heardAs == 'meditating' || slotValues.reminder.heardAs == 'relax') {

try {

await reminderApiClient.createReminder(reminderRequestMeditate)

}

catch (error) {

console.log(`~~~ Error: ${error}`)

return handlerInput.responseBuilder

.speak('There was an error scheduling your reminder. Please try again later.')

.getResponse();

}

const speakOutput = 'You have successfully scheduled a reminder to meditate at this time tomorrow!';

return handlerInput.responseBuilder

.speak(speakOutput)

//.reprompt('add a reprompt if you want to keep the session open for the user to respond')

.getResponse();

}

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

try {

await reminderApiClient.createReminder(reminderRequestWorkout)

}

catch (error) {

console.log(`~~~ Error: ${error}`)

return handlerInput.responseBuilder

.speak('There was an error scheduling your reminder. Please try again later.')

.getResponse();

}

const speakOutput = 'You have successfully scheduled a reminder to exercise at this time tomorrow!';

return handlerInput.responseBuilder

.speak(speakOutput)

//.reprompt('add a reprompt if you want to keep the session open for the user to respond')

.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() \* MAX\_TIPS);

let say = 'Thank you for using ' + invocationName + '. You can schedule a reminder to workout or meditate at this time tomorrow by saying "remind me to exercise with Intent Fitness". 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();

return responseBuilder

.speak('Sorry, 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?')

.reprompt()

.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 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?')

.reprompt()

.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() \* MAX\_TIPS);

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

const ranBirds = Math.floor(Math.random() \* 2); // 0 and 1

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

const ranOcean = Math.floor((Math.random() \* 2) + 4); // 4 and 5

const ranRainonCar = Math.floor((Math.random() \* 2) + 6); // 6 and 7

const ranRain = Math.floor((Math.random() \* 2) + 8); // 8 and 9

const ranLoFi = Math.floor((Math.random() \* 3) + 10); //10, 11 and 12

const sBirds = STREAMS[ranBirds];

const sFire = STREAMS[ranFire];

const sOcean = STREAMS[ranOcean];

const sRainonCar = STREAMS[ranRainonCar];

const sRain = STREAMS[ranRain];

const sLoFi = STREAMS[ranLoFi];

const sRandom = STREAMS[ranSong];

const sMindofBreath = STREAMS[13];

const sMountBreathing = STREAMS[14];

// 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 me";

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 == '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);

}

else if (slotValues.sound.heardAs == 'mountain' || slotValues.sound.heardAs == 'mountain meditation' || slotValues.sound.heardAs == 'session one') {

say = 'Great choice!';

handlerInput.responseBuilder

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

}

else if (slotValues.sound.heardAs == 'mindfulness of breath' || slotValues.sound.heardAs == 'mindfulness' || slotValues.sound.heardAs == 'breath' || slotValues.sound.heardAs == 'session one') {

say = 'Great choice!';

handlerInput.responseBuilder

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

}

}

return responseBuilder

.speak(say)

.getResponse();

},

};

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();

// 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') {

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

}

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. You can choose either Mindfulness of Breath or Mountain Meditation to start different sessions. Which will it be?';

}

}

return responseBuilder

.speak(say)

.reprompt('I don\'t know that one. You can choose to listen to sounds such as birds, rain, fire and waves. You can also say random and I will choose a sound for you. What would you like to listen to?')

.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('Sorry, 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();

}

};

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,

CreateReminderIntentHandler,

SessionEndedHandler

)

.withApiClient(new Alexa.DefaultApiClient())

.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}"

]

},

{

"name": "getReminder",

"slots": [

{

"name": "reminder",

"type": "REMINDER",

"samples": [

"remind me to {reminder}",

"set up a reminder to {reminder}",

"remember me to {reminder}",

"i would like to {reminder}",

"{reminder}"

]

}

],

"samples": [

"remember me",

"remember me to {reminder}",

"remind me",

"set up a reminder to {reminder}",

"remind me to {reminder}"

]

}

],

"types": [

{

"name": "SOUNDS",

"values": [

{

"name": {

"value": "mindfulness of breath",

"synonyms": [

"session one",

"breath",

"mindfulness"

]

}

},

{

"name": {

"value": "mountain",

"synonyms": [

"session 2",

"mountain meditation"

]

}

},

{

"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": [

"meditating",

"meditation"

]

}

},

{

"name": {

"value": "workout",

"synonyms": [

"exercise"

]

}

}

]

},

{

"name": "REMINDER",

"values": [

{

"name": {

"value": "meditate",

"synonyms": [

"relax",

"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"

}

]

}

]

},

{

"name": "getReminder",

"confirmationRequired": false,

"prompts": {},

"slots": [

{

"name": "reminder",

"type": "REMINDER",

"confirmationRequired": false,

"elicitationRequired": true,

"prompts": {

"elicitation": "Elicit.Slot.1308217539947.242246796500"

}

}

]

}

],

"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, fire and waves. You can also say random and I will choose a sound for you. What would you like to listen to?"

}

]

},

{

"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?"

}

]

},

{

"id": "Elicit.Slot.61803317158.47237465252",

"variations": [

{

"type": "PlainText",

"value": "What would you like to be reminded of?"

},

{

"type": "PlainText",

"value": "Great. What message would you like in the reminder?"

}

]

},

{

"id": "Confirm.Slot.61803317158.47237465252",

"variations": [

{

"type": "PlainText",

"value": "The message. {message} will be sent as a reminder"

}

]

},

{

"id": "Confirm.Intent.1350208148512",

"variations": [

{

"type": "PlainText",

"value": "Sure, I will remind you to {reminder}"

}

]

},

{

"id": "Elicit.Slot.1308217539947.242246796500",

"variations": [

{

"type": "PlainText",

"value": "I can remind you to exercise or meditate at this time tomorrow. Which will it be?"

}

]

}

]

}

}