var https = require('https');

var querystring = require("querystring");

exports.handler = function (event, context) {

try {

if (event.session.new) {

onSessionStarted({requestId: event.request.requestId}, event.session);

}

if (event.request.type === "LaunchRequest") {

onLaunch(event.request,

event.session,

function callback(sessionAttributes, speechletResponse) {

context.succeed(buildResponse(sessionAttributes, speechletResponse));

});

} else if (event.request.type === "IntentRequest") {

onIntent(event.request,

event.session,

function callback(sessionAttributes, speechletResponse) {

context.succeed(buildResponse(sessionAttributes, speechletResponse));

});

} else if (event.request.type === "SessionEndedRequest") {

onSessionEnded(event.request, event.session);

context.succeed();

}

} catch (e) {

context.fail("Exception: " + e);

}

};

/\*\*

\* Called when the session starts.

\*/

function onSessionStarted(sessionStartedRequest, session) {

// add any session init logic here

}

/\*\*

\* Called when the user invokes the skill without specifying what they want.

\*/

function onLaunch(launchRequest, session, callback) {

getWelcomeResponse(callback)

}

/\*\*

\* Called when the user specifies an intent for this skill.

\*/

function onIntent(intentRequest, session, callback) {

var intent = intentRequest.intent

var intentName = intentRequest.intent.name;

// dispatch custom intents to handlers here

if (intentName == "mailing") {

handleReindeerResponse(intent, session, callback)

} else {

throw "Invalid intent"

}

}

/\*\*

\* Called when the user ends the session.

\* Is not called when the skill returns shouldEndSession=true.

\*/

function onSessionEnded(sessionEndedRequest, session) {

}

// ------- Skill specific logic -------

function getWelcomeResponse(callback) {

console.log('Loading function');

var speechOutput = "welcome to the email service"

var reprompt = "welcome to the email service"

var header = "welcome to the email service"

var shouldEndSession = false

var sessionAttributes = {

"speechOutput" : speechOutput,

"repromptText" : reprompt

}

callback(sessionAttributes, buildSpeechletResponse(header, speechOutput, reprompt, shouldEndSession))

}

function handleReindeerResponse(intent, session, callback) {

console.log('Loading function');

var eParams = {

Destination: {

ToAddresses: ["layeequerehman786@gmail.com"]

},

Message: {

Body: {

Text: {

Data: "Hey! What is up?"

}

},

Subject: {

Data: "Email Subject!!!"

}

},

Source: "layeequerehman786@gmail.com"

};

console.log('===SENDING EMAIL===');

var email = ses.sendEmail(eParams, function(err, data){

if(err) console.log(err);

else {

console.log("===EMAIL SENT===");

console.log(data);

console.log("EMAIL CODE END");

console.log('EMAIL: ', email);

//context.succeed(event);

callback(null,"success");

}

});

var speechOutput = "your mail has been sent"

var reprompt = "your mail has been sent"

var header = "your mail has been sent!"

var shouldEndSession = false

var sessionAttributes = {

"speechOutput" : speechOutput,

"repromptText" : reprompt

}

callback(sessionAttributes, buildSpeechletResponse(header, speechOutput, reprompt, shouldEndSession))

}

function handleNoResponse(intent, session, callback) {

handleFinishSessionRequest(intent, session, callback)

}

function handleGetHelpRequest(intent, session, callback) {

// Ensure that session.attributes has been initialized

if (!session.attributes) {

session.attributes = {};

}

var speechOutput = "I can tell you facts about all the famous reindeer: " +

"Dasher, Dancer, Prancer, Vixen, Comet, Cupid, Blitzen, Rudolph, and Olive." +

" Which reindeer are you interested in? Remember, I can only give facts about one reindeer at a time."

var repromptText = speechOutput

var shouldEndSession = false

callback(session.attributes, buildSpeechletResponseWithoutCard(speechOutput, repromptText, shouldEndSession))

}

function handleFinishSessionRequest(intent, session, callback) {

// End the session with a "Good bye!" if the user wants to quit the game

callback(session.attributes,

buildSpeechletResponseWithoutCard("Good bye! Thank you for using Reindeer Facts!", "", true));

}

// ------- Helper functions to build responses for Alexa -------

function buildSpeechletResponse(title, output, repromptText, shouldEndSession) {

return {

outputSpeech: {

type: "PlainText",

text: output

},

card: {

type: "Simple",

title: title,

content: output

},

reprompt: {

outputSpeech: {

type: "PlainText",

text: repromptText

}

},

shouldEndSession: shouldEndSession

};

}

function buildSpeechletResponseWithoutCard(output, repromptText, shouldEndSession) {

return {

outputSpeech: {

type: "PlainText",

text: output

},

reprompt: {

outputSpeech: {

type: "PlainText",

text: repromptText

}

},

shouldEndSession: shouldEndSession

};

}

function buildResponse(sessionAttributes, speechletResponse) {

return {

version: "1.0",

sessionAttributes: sessionAttributes,

response: speechletResponse

};

}

function capitalizeFirst(s) {

return s.charAt(0).toUpperCase() + s.slice(1)

}