var friends = {

"saurabh" : {

"personality\_trait": "timepass",

"skill": "android"

},

"akshay" : {

"personality\_trait": "is samll kid",

"skill": "docker"

},

"piyush" : {

"personality\_trait": "is affectionate",

"skill": "amazon web services"

},

"angad" : {

"personality\_trait": "is slightly tricky",

"skill": "java"

},

"nikhil" : {

"personality\_trait" : "is handsome and easy-going",

"skill" : "good with kids"

},

"vamsi" : {

"personality\_trait" : "is affectionate",

"skill": "anything"

},

"prakash" : {

"personality\_trait": "is loud",

"skill": "electrifies others"

},

"akhil" : {

"personality\_trait": "is fast as a bolt",

"skill":"good at singing"

},

"hitesh" : {

"personality\_trait" : "is a little down on himself",

"skill" : "android developer"

},

"manoj" : {

"personality\_trait" : "admits when he's wrong",

"skill" : "good at hide-and-go-seek"

}

}

// Route the incoming request based on type (LaunchRequest, IntentRequest,

// etc.) The JSON body of the request is provided in the event parameter.

exports.handler = function (event, context) {

try {

/\*console.log("event.session.application.applicationId=" + event.session.application.applicationId);

\* Uncomment this if statement and populate with your skill's application ID to

\* prevent someone else from configuring a skill that sends requests to this function.

if (event.session.application.applicationId !== "amzn1.ask.skill.bdf7c4ec-f508-4b9b-a341-22d60fbecc17") {

context.fail("Invalid Application ID");

}\*/

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 == "myfriend") {

handleReindeerResponse(intent, session, callback)

} else if (intentName == "AMAZON.YesIntent") {

handleYesResponse(intent, session, callback)

} else if (intentName == "AMAZON.NoIntent") {

handleNoResponse(intent, session, callback)

} else if (intentName == "AMAZON.HelpIntent") {

handleGetHelpRequest(intent, session, callback)

} else if (intentName == "AMAZON.StopIntent") {

handleFinishSessionRequest(intent, session, callback)

} else if (intentName == "AMAZON.CancelIntent") {

handleFinishSessionRequest(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) {

var speechOutput = "Welcome to friend Facts! I can tell you about all the famous friends: "+

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

var reprompt = "Which friend are you interested in "

var header = "friend Facts!"

var shouldEndSession = false

var sessionAttributes = {

"speechOutput" : speechOutput,

"repromptText" : reprompt

}

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

}

function handleReindeerResponse(intent, session, callback) {

var reindeer = intent.slots.Reindeer.value.toLowerCase()

if (!friends[friends]) {

var speechOutput = "That friend isn't very famous. Try asking about another "

var repromptText = "Try asking about another friend"

var header = "Not Famous Enough"

} else {

var personality\_trait = friends[friends].personality\_trait

var skill = friends[friends].skill

var speechOutput = capitalizeFirst(friends) + " " + personality\_trait + " and " + skill + ". Do you want to hear about more reindeer?"

var repromptText = "Do you want to hear about more friend?"

var header = capitalizeFirst(friends)

}

var shouldEndSession = false

callback(session.attributes, buildSpeechletResponse(header, speechOutput, repromptText, shouldEndSession))

}

function handleYesResponse(intent, session, callback) {

var speechOutput = "Great! Which friend? "

var repromptText = speechOutput

var shouldEndSession = false

callback(session.attributes, buildSpeechletResponseWithoutCard(speechOutput, repromptText, 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 friend: " +

" 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 friend 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)

}