Software Detail Design Report – Rough Draft

Team LER: Amazon Alexa O&M Skill

Eric Schulze, Ryan Gosling, Larome Dickerson

Software Detail Design Report – Rough Draft

Team LER: Amazon Alexa O&M Skill

Eric Schulze, Ryan Gosling, Larome Dickerson

2017

**Table of Contents**

1. Overview …………………………………………………………………………………………………2

2. Data Design …………………………………………………………………………………………2

3. Architecture Design ……………………………………………………………………2

4. Interface Design ……………………………………………………………………………3

5. Procedural Design …………………………………………………………………………3

**1. Overview**

The American Printing House for the Blind (APH) is the world’s largest company devoted to manufacturing, researching, and developing products for people who are blind or visually impaired. APH designs and manufactures textbooks and magazines in braille, large print, and digital formats, as well as other educational, recreational, and daily living products, one of which we have been tasked to develop. APH has requested our services to develop a skill for Amazon’s Alexa voice interface platform.

This skill will consist of a trivia game in which the subject of Orientation and mobility (O&M) will be the focus. The O&M trivia game will be used in concurrence with the training, and education provided to the visually impaired through all stages of life. Blindness or visual impairment can happen to an individual through different stages of their life, from birth to old age, and learning through engagement will solidify the skills they will need to survive in a visual based world, which this Alexa skill will provide.

1a. Glossary

APH: American Print House for the Blind, our project sponsor

O&M: Orientation and Mobility, a subject of instructional lessons for the visually impaired that APH works with

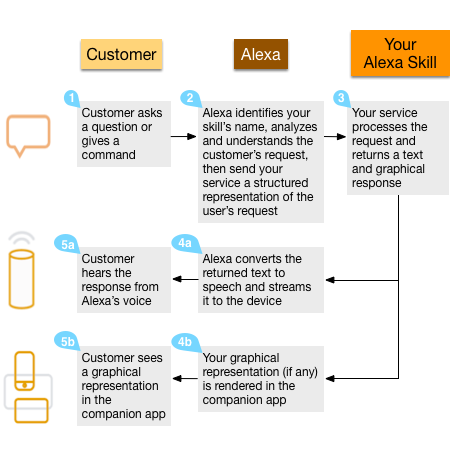
Skill: A capability of Alexa, the Amazon cloud based service that handles speech recognition, machine learning, and natural language understanding, that defines how you can interact with Alexa.

Interaction Model: The Voice User Interface (VUI) of the skill, it defines what functionalities or behaviors the skill is able to handle

Hosted Service: The programming logic of the skill, hosted by Amazon Lambda Services, that responds to a user’s requests and phrases

Utterance: A phrase from the user that the skill can understand and map to an intent

Intent: A representation of functions that the skill is capable of performing. Multiple utterances can map to one intent that then gets passed on to the Hosted Service.

1b. System Overview

The Alexa skill will consist of two individual parts working in tandem with each other, the Interaction Model and the Hosted Service. The Interaction Model is a voice user interface (VUI) for the Alexa skill. The Interaction Model is to the skill, what a graphical user interface is to a mobile application; it defines how a user is able to interact with our skill. This Interaction model includes the intents and utterances that a user can use. The intents and utterances trigger responses based on the second part of the skill, the Hosted Service. The hosted service is the brains behind how Alexa handles responses and phrases from the user. The hosted service takes as input intents, deciphered from the user utterance by the VUI, and outputs response phrases.

1c. References

Amazon Alexa Skills Kit (<https://developer.amazon.com/alexa-skills-kit>)

Alexa Skills Documentation (<https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/overviews/understanding-custom-skills>)

Codecademy – Build Alexa Skills (<https://www.codecademy.com/learn/learn-alexa>)

**2. Data Design**

{"questions":{[

"question":{

"id": "id\_value",

"difficulty": "difficulty\_value",

"question\_text": "text\_value",

"correct\_answer": "answer\_value",

"question\_type": "type\_value",

"answers": [

{"answer\_text": "text\_value"},

{"answer\_text": "text\_value"},

{"answer\_text": "text\_value"},

{"answer\_text": "text\_value"},

],

"additional\_answer\_info": "text\_value",

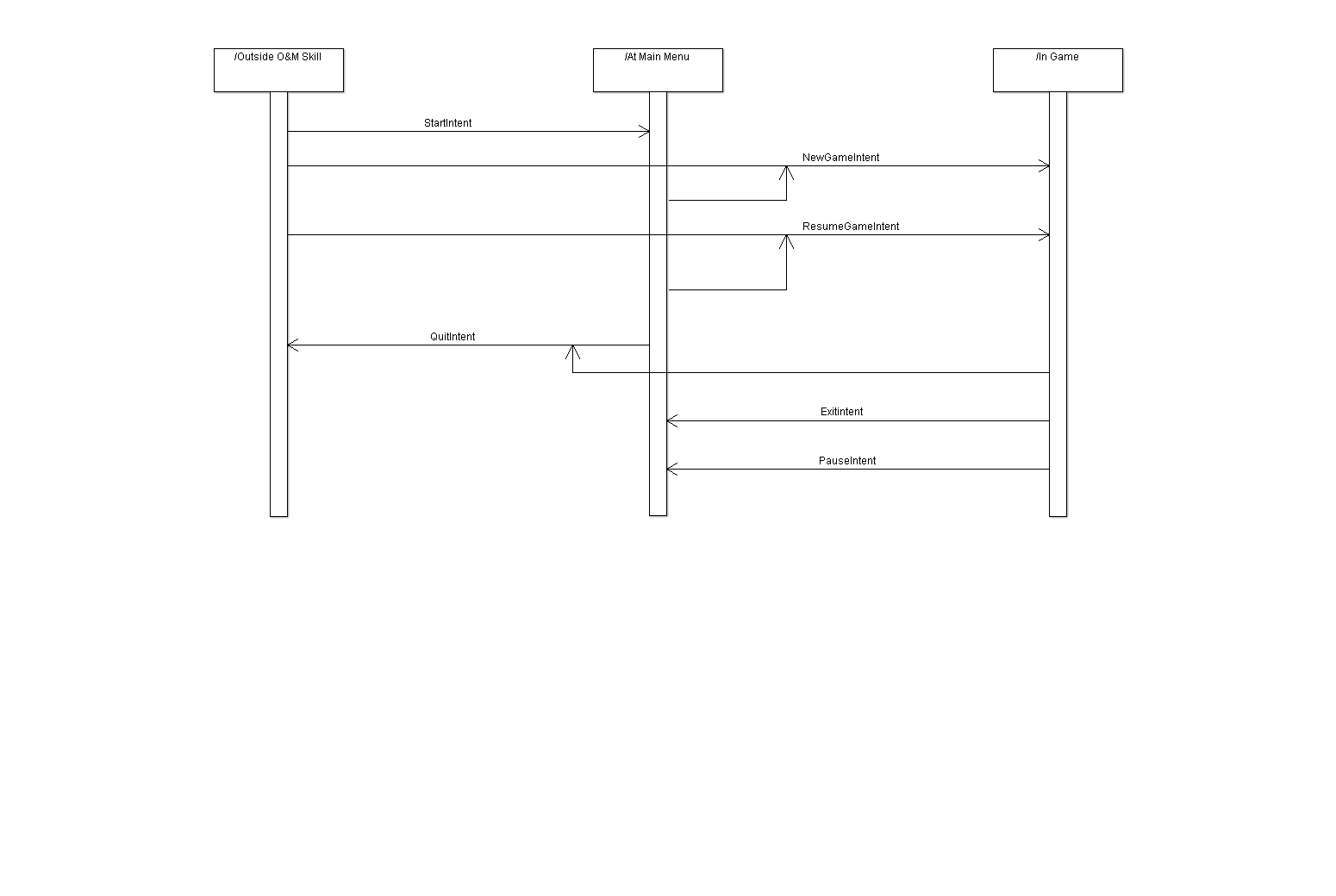
}

]}

}

**3. Architecture Design**

3a. Stage 1: Outside APH O&M Skill

For Stage 1 and Stage 2, the only control that will need to be handled will be the utterance-intent-response loop. These actions will navigate between all of the stages. (See diagram for intent actions)

3b. Stage 2: At Main Menu

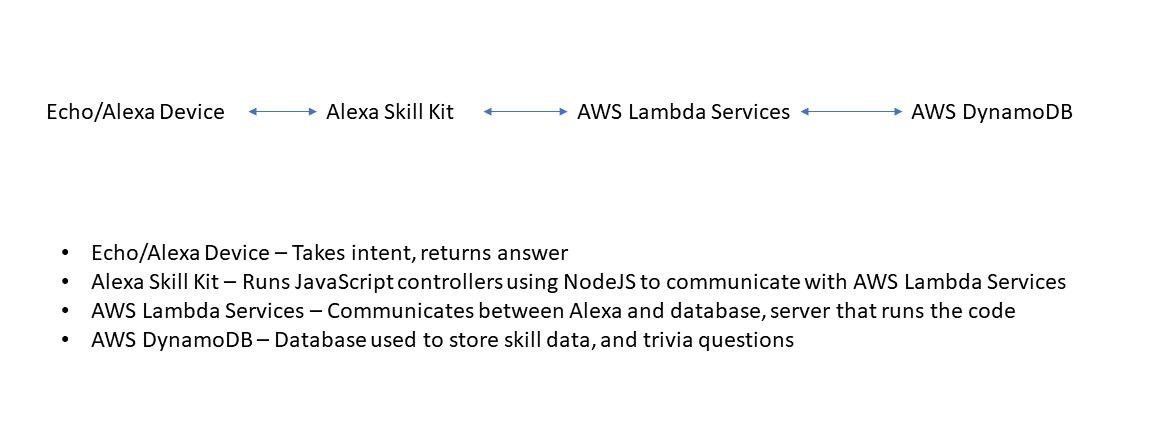
For Stage 1 and Stage 2, the only control that will need to be handled will be the utterance-intent-response loop. These actions will navigate between all of the stages. (See diagram for intent actions)

3c. Stage 3: In Game

A screenshot of a cell phone

Description generated with high confidenceFor Stage 3, there will be a repeated loop through each question. This loop will terminate either if there is no response after three attempts to garner a response, or if the list of questions has been completed (i.e. the game is finished). See the following UML diagram for the loop actions.

**4. Interface Design**

****

A screenshot of a cell phone

Description generated with high confidence**5. Procedural Design**

GetIntent(Intent i){

Listen for incoming intent; //Always running on alexa devices.

Switch(i){ //This may be covered by the Control method

Case “Game + Players + Difficulty”: StartGame(NumPlayers, Difficulty, SaveState);

Break;

Case “Main Menu”: MainMenu(SaveState);

Break;

Case “Quit”: QuitGame(SaveState);

Break;

Default;

Break;

}

}

StartGame(NumPlayers, Difficulty, SaveState){

Int[] ListOfQuesIDs;

If(SaveState.isEmpty()){

ListOfQuesIDs = GetNewQuestionSet(Difficulty); //loads 20 question ids into an array

}

Else{

ListOfQuesIDs = SaveState.getListOfQuesIDs();

}

Question q;

Boolean correctAns;

Int playerID = 1;

Int[] score;

Int questionNum = 0;

While(intent == Game && questionNum != ListOfQuesIDs.length){

ReadIntro(questionNum, playerID);

q = GetQuestion(ListOfQuesIDs[questionNum])); //gets question from database by ID

ReadQuestion(q.getQues());

ReadAnswers(q.getAnswers());

\*Listen for answer.

isCorrectAns(q.getCorrectAns(), Player’s given answer);

For(int a = 0; a < 3; a++){

If(Player answers are invalid twice){

ReadOptions();

}

If(Player answers are invalid thrice){

correctAns = false;

}

}

addScore(playerID, score[playerID – 1], correctAns);

if(playerID == NumPlayers){

readScore(score);

playerID = 1; //player 1 goes again, new round starts.

}

Else{

playerID++;

}

}

Congratulate(player with highest score);

}

StartMainMenu(Savestate){

If(Intent comes from ‘Out of game’ state){

ReadMenuOptions();

\*Listen for answer.

}

Else if(Intent come from ‘In game’ state){

SaveGame(SaveState);

ReadMenuOptions();

\*Listen for answer.

}

}

SaveGame(SaveState){

Upload each variable of the save state to the server:

• AlexaID //unique alexa ID for the device

• NumPlayers //The number of players in this game

• Difficulty //The difficulty of this game

• ListOfQuesIDs //The list of question IDs, in order to have the same questions

• CurrentQues //Which question the current game is on

• Scores //The scores for the players

}

QuitGame(SaveState){

SaveGame(SaveState);

\*”Alexa, quit the game”;

}

ReadIntro(QuestionNum, PlayerNum)

This procedure will create a statement for a question introduction for Alexa to read at the beginning of each question. As inputs, it will take the question order number and the player number, both integers. The output will be the string introduction message. An example of a possible output would be “Question Number 6, for Player 2”. There will be a standard introduction statement with two variables for the question number and player number. This standard introduction will then be returned with the correct variables supplied from the input.

Process:

Input: int questionNum, int playerNum

Output: “Question Number “ + questionNum + “, for Player Number “ + playerNum

ReadQuestion(QuestionText)

This procedure will create a statement for a question for Alexa to read for each question. As input, it will take the question text as a string. The output will be the string question reading.

Process:

Input: Question Text

Output: Question Text string

ReadAnswers(Answers[])

This procedure will create a statement Alexa to read all of the answers to a question. As inputs, it will take a string array of answers. The output will be the string message with the answers randomized.

Process:

Input: string[] answers

Randomize array

Output: “Is the answer a “ + answers[1] + “ or b “ + answers[2] + “ or c “ …

ReadOptions()

This procedure will create a statement Alexa to read of the different options a user has while in the trivia game. This statement will be read if Alexa has not received input within a given time. It will take no inputs. The output will be the string message listing the different options a user can say to Alexa.

Process:

Input: none

Output: Options message

GetQuestions(ListOfQuestionIDs)

This procedure will go out to the database and bring back a list of questions, with all of the information associated with those questions, in the form of a JSON object. The input will be a list of questionIds. This will be the list of questions that are returned, in the same order. The output will be a list of question objects.

Process:

Input: list questionIds

Convert questionIds to a string list of ids.

Pass the string to a database call that takes the ids and returns the JSON object for each question.

Parse each JSON object into a question object and put it into the questionObjects list.

Output: list questionObjects

IsAnswerCorrect(CorrectAns, GivenAns)

This procedure simply compare. This statement will be read if Alexa has not received input within a given time. It will take no inputs. The output will be the string message listing the different options a user can say to Alexa.

Process:

Input: string correctAns, string givenAns

Output: Boolean correct