<PROJECT 중간보고서>

Project Kalexa

alexa를 활용한 음성 인식 기반 프로그램

최영진, 김준혁, 이상훈

엄현상 교수님

윤석찬(Amazon, 테크에반젤리스트)

Table of Contents

1. Abstract
2. Introduction
   A. 프로젝트의 배경 및 중요성
   B. 보고서의 구성
3. Background Study
   A. 관련 접근방법/기술 장단점 분석
   B. 프로젝트 개발환경
4. Goal/Problem & Requirements
5. Project Architecture
   A. Architecture Diagram
   B. Architecture Description
6. Implementation Spec
   A. Input/Output Interface
   B. Inter Module Communication Interface
   C. Modules
7. Current Status
8. Future Work
9. Division & Assignment of Work
10. Schedule
11. Demo Plan

◆ [Appendix] Detailed Implementation Spec

1. Abstract

본 프로젝트는 Amazon Echo에서 동작하는 음성 인식 기반의 응용 프로그램을 제작하는 것을 목표로 한다. 현재 해외에서의 K-POP에 대한 열기는 매우 뜨겁다. K-POP에 대한 관심이 늘어남과 동시에 외국인들이 한국어를 배우고자 하는 욕구 또한 커지고 있다. 이러한 상황을 고려하여 아마존의 음성 인식 시스템인 alexa를 활용해 한국어를 학습하고, K-POP 컨텐츠와 관련된 여러 기능을 이용할 수 있도록 한다.

해당 프로그램을 제작하기 위한 개발 환경으로는 AWS(아마존 웹 서비스)내의 다양한 서비스를 활용한다. 서버의 역할을 하는 Lambda, 파일 저장소의 역할을 하는 S3, 데이터 저장소의 역할을 하는 dynamoDB 등을 융합하여 서버 로직을 구현한다. 또한 네이버의 음성합성 API를 사용하여 외국인들이 현지인의 발음을 제공받을 수 있도록 하고, 멜론의 차트, 앨범, 아티스트 등의 API를 활용하여 K-POP컨텐츠를 제공한다.

2. Introduction

A. 프로젝트의 배경 및 중요성

최근 K-POP 스타들의 많은 해외 진출과 소셜 미디어의 발달로 인해 해외에서의 K-POP에 대한 열기가 뜨겁다. 해외 팬들이 다양한 한국어 컨텐츠를 접하면서 한국어 학습에 대한 수요 또한 증가하고 있는 추세이다. 이러한 측면에서 외국인들에게 K-POP에 대한 정보를 제공해줄 뿐만 아니라 한국어를 간단하게 학습할 수 있고 올바른 발음을 들려주는 기능을 제공하는 것은 충분히 필요가치가 있다.

따라서 본 프로젝트의 목표는 크게 K-POP 정보 제공, 한국어 학습 제공 두가지로 나뉜다. 현재 Amazon Alexa가 한국어 인식/발음을 모두 지원하지 않고 있는 점이 한국어 기반의 정보를 제공하는 프로젝트에 가장 큰 한계와 문제점이다. 이 제약 조건 속에서 어떤 방식으로 문제를 해결할지 정하고 최종적으로 완성도 있는 결과물을 만들어 내는 것이 이번 과제의 궁극적인 목표이다.

B. 보고서의 구성

이후의 보고서의 구성은 다음과 같다.

- ❏ 3.Background Study - Amazon Alexa, AWS lambda, dynamoDB, S3에 대한 설명과 이번 프로젝트에서 사용할 외부 API에 대한 설명
- ❏ 4.Goal/Problem & Requirements - 본 프로젝트의 목표과 현재 Alexa의 한계점 및 요구사항
- ❏ 5.Project Architecture - 본 프로젝트에서 구현한 기능들을 실제 사용자가 사용하였을 때 동작하는 Architecture diagram을 보여주고 그에 대한 설명
- ❏ 6.Implementation Spec - 5.에서 설명한 내용을 기능 별 모듈 단위로 좀더 구체적으로 설명
- ❏ 7.Current Status - 현재까지의 각 기능 별 구현 사항에 대한 자세한 설명
- ❏ 8.Future Work - 앞으로 추가적으로 구현할 내용과 논의 해봐야될 사항들에 대한 설명
- ❏ 9.Division & Assignment of Work - 본 프로젝트에서 팀원 간 업무 분배 상황
- ❏ 10.Schedule - 프로젝트 진행 계획
- ❏ 11.Demo Plan - 프로젝트 시연 계획

❑ 12.Appendix - 각 모듈 별로 parameter와 return data type을 명시하고 각각의 기능과 내부 로직에 대한 설명

3. Background Study

A. 관련 접근방법/기술 장단점 분석

1. Amazon Alexa

아마존 알렉사(Alexa)는 2015년 출시된 일종의 음성 비서이다. 에코(Echo)라는 원통형 스피커 형태로 출시되었는데, 이 안에는 음성 인식 프로그램인 Alexa가 들어 있다. Alexa는 시각, 날씨 알림 등 기본 기능은 물론이고 주요 뉴스나 특정 정보를 브리핑하는 기능을 제공한다. 아마존은 장차 Alexa가 집 및 사무실을 자동화하는 데 핵심적인 역할을 할 것이라고 기대하고 있다.

Alexa는 현재 한국어를 지원하지 않는다. Apple 사의 Siri 등 다른 음성 인식 인공지능 역시 한국어를 지원하는 경우를 찾기 어렵다. 본 프로젝트에서는 Alexa를 타 서비스들과 연계하여 한국어 관련 기능을 제공하는 것을 목적으로 한다. 한국어를 인식하고 한국어를 구사하는 것은 가능하지도 않거니와 본 프로젝트의 범위에서 벗어난다. 그러나 영어 전용 프로그램인 Alexa에 한국어 번역 질의 응답이나 K-POP 등 인기 있는 한국 컨텐츠를 제공하는 기능을 추가하여 Alexa의 가용 범위를 늘리는 데 의의가 있다.

2. AWS Lambda

아마존 웹 서비스의 람다(Lambda)는 컴퓨터과학에서 익명 함수(Anonymous Function)을 지칭하는 용어에서 유래하였다. 프로그래밍 언어에서 익명 함수를 복잡한 선언 과정 없이 사용하듯이, 프로그래머가 개발 환경에 신경쓰지 않고 로직에만 신경쓸 수 있도록 한다. 예를 들어, 간단한 HTTP 동작을 처리하기 위해서도 일반적으로 서버를 구축하고 기본적인 웹 개발 환경을 갖추어야 한다. 그러나 아마존 람다는 해당 기능만 구현하면 서버를 관리할 필요 없이 AWS의 환경에서 실행이 가능하다. 기능에 맞는 환경을 편리하게 선택 가능하며, 호출하는 시점만 서버 자원을 사용하므로 가격 면에서도 합리적이다.

본 프로젝트는 기능 자체는 많으나 외부 인터페이스와의 연결이 큰 부분을 차지하고 있어 코드 자체의 규모는 그리 크지 않을 것으로 예상된다. 또한 서버 머신이나 운영 체제 등에 구애받을 경우도 없다고 판단되기에, 간편하게 로직을 구현하고 실행할 수 있는 람다가 적합하다고 판단하였다. EC2 등

서버를 사용하면 서버 환경 설정 및 관리에 불필요한 시간이 들어가고, 비용도 더욱 발생하는 데 반해 유리한 점이 많다고 할 수 있다.

3. AWS S3, DynamoDB

아마존 웹 서비스는 수많은 관련 솔루션을 제공한다. 이 중 본 프로젝트에서는 NoSQL 기반 데이터베이스인 DynamoDB와 클라우드 스토리지인 S3(Simple Storage Service)를 사용한다. 람다와 연동 SDK가 잘 되어 있을 뿐만 아니라, 효율성 및 안정성이 보장되어 있는 서비스이다.

4. 네이버 기계 번역 / 음성 합성 API

국내 유명 포털인 네이버는 개발자 저변을 확대하면서 여러 공개 API들을 제공하고 있다. 본 프로젝트에서는 그 중 기계 번역 및 음성 합성 API를 사용할 것이다. 기계 번역은 사람의 손을 거치지 않아 아직까지는 번역의 질이 떨어지나, Alexa와 같은 실시간 서비스에서 즉시 응답을 줄 수 있다는 장점이 있다. 또한 네이버의 기계 번역은 한국어 품질이 수준급이라고 알려져 있다. Alexa에서 한국어 번역 질의가 들어왔을 때, 이를 한국어 응답으로 되돌려주기 위해 사용한다.

음성 합성 API란 TTS(Text to Speech) 서비스라고도 불리는데, 텍스트를 음성으로 변환한다. Alexa는 영어 응답을 지정할 수 있으나 한국어는 발음 자체가 불가능하다. 음성 합성 사용은 이를 해결하기 위한 아이디어로, 음성 합성 API에서 반환된 MP3 파일은 언어와 관계 없이 재생이 가능하기에 Alexa가 한국어를 말하도록 할 수 있다.

5. Melon 실시간 차트 / 곡 검색 API

Melon은 국내 대형 온라인 음악서비스이다. SK 플래닛 개발자 센터에서는 Melon에 질의할 수 있는 API를 제공하여, 이를 통해 Melon의 여러 정보를 요청할 수 있다. 이 중에서 실시간 차트 정보 및 곡 검색 API를 사용할 것이다. 실시간 차트는 실시간으로 반영되는 가요 순위이다. 이를 이용하면 어디에서나 한국 가요의 실시간 순위를 그대로 받아 볼 수 있다. 차트의 순위 범위는 파라미터 값을 통해 조정할 수 있다. 곡 검색 API는 파라미터에 search keyword를 추가해서 호출하면 그 keyword를 가지고 검색된 곡의 정보를 반환한다. 본 프로젝트에서는 특정 가수의 차트 순위 곡 이외에 어떤 곡이 있는지에 대한 정보를 제공하기 위해 사용할 것이다.

B. 프로젝트 개발환경

여러 명의 조원으로 구성된 프로젝트이니만큼 협업 도구가 필수적이다. 소스 버전 관리 도구로 git을 사용하는데, 본 프로젝트의 경우 소스 파일 외에도 관리해야 할 텍스트 정보 파일들이 여럿 존재하여 이 역시 git을 통해 관리하도록 한다. 원활한 커뮤니케이션을 위해서는 현재 전 세계적으로 널리 쓰이는 업무용 메신저인 slack을 사용한다. 또한 git과 slack의 연동을 통해 실시간으로 상대의 작업내역에 대한 알림을 받을 수 있도록 하였다.

서버로는 AWS의 Lambda를 사용하였고, 개발 언어는 NodeJS(버전 4.3)이다. DB로는 AWS에서 제공하는 noSQL DB인 DynamoDB를 사용한다. 데이터 저장소로는 S3를, 추후 데이터 캐싱에는 ElastiCache 를 사용할 예정이다. Alexa는 개발 플랫폼인 Alexa Skill Set을 제공한다. 이를 통해 본 프로젝트에서 Alexa가 인식해야 하는 문자열 및 변수들을 설정할 수 있다. Alexa의 테스트는 문자열 입력 방식 및 웹 상에서의 음성 인식 방식이 모두 가능하다. 또한 네이버의 음성 합성 및 기계 번역 API를

사용한다.

Amazon Web Service는 상용 서비스이다. Lambda의 경우 사용자가 서버에 접근할 수 없으나, 내부적으로는 서버 자원을 사용하므로 이를 누적하여 과금하고, DynamoDB나 S3의 경우도 저장 공간과 이용 시간에 따라 요금이 청구된다. 본 프로젝트에서는 한국 AWS의 개발용 계정을 받는 것을 검토하였으나, 상용화 전에는 AWS Free tier 및 학생 지원 프로그램으로 충분하다고 판단되어 이를 이용할 것이다. 네이버 API의 경우도 보통 10,000글자/일에 대하여 무료로 제공되고 있다.

4. Goal/Problem & Requirements

- **프로젝트의 목적**

본 프로젝트의 목적은 한국어 및 K-POP에 관심이 있는 외국인들을 대상으로 하여, 한국어를 학습하는 기능과 다양한 K-POP 컨텐츠를 제공하는 것이다.

- **제공 언어의 한계**

기존 Alexa에서 제공하는 언어는 영어와 독일어이다. 따라서 Alexa가 한국어로 응답할 때 나타나는 한계가 있다. 한국어를 지원하지 않는 Alexa에서 한국어로 응답하도록 하기 위해서는 한국어 발음을 알파벳으로 표현하여 그것을 Alexa가 그대로 읽게 하는 방법이 있다. 하지만 이 방법의 경우 Alexa 응답의 발음이 매우 부자연스러우며 외국인이 한국어의 올바른 발음을 들을 수 없다는 문제점이 발생한다. 따라서 이와 같은 방법을 개선하기 위해 음성 합성을 사용하도록 한다. 한글 text를 알파벳으로 변환하는 것이 아니라 음성 합성을 통해 바로 mp3 파일로 만들어서 한국어 발음 그대로 재생시켜주는 방법이다. 이 방법을 사용하면 이전과 같은 부자연스러움을 줄일 수 있을 뿐만 아니라 외국인들이 올바른 한국어 발음을 들을 수 있게 해준다는 측면에서 큰 의미가 있다.

- **추천 알고리즘 구현**

아무래도 해외의 K-POP 팬은 본인이 좋아하는, 혹은 해당 언어권에서 유명한 가수나 곡 이외의 다른 한국 음악에 대한 접근성이 떨어질 것으로 사료된다. 그런 점을 고려할 때 본인의 선택 외에 시스템 상에서 자동으로 추천해주는 기능이 있다면 사용자 입장에서 매우 유용할 것으로 생각된다. 보다 취향에 맞는 곡을 추천해주기 위해 유저 기반의 데이터를 수집, 활용하여 추천하는 collaborative filtering을 적용하고자 한다. 예를 들면 현재 노래를 들은 사용자가 직전에 들은 노래, 현재 노래를 들은 사용자들이 많이 들었던 다른 노래, 나와 가장 많이 들은 곡이 겹치는 사용자의 플레이리스트 중 랜덤 추천 등이 가능할 것으로 보인다.

- **구현 시의 정량적 달성 목표**

본 프로젝트의 평가 기준으로는 두 가지 사항이 있다. 첫 째는 응답이 얼마나 빠르게 오는지, 둘째는 얼마나 정확하게 응답하는지이다.

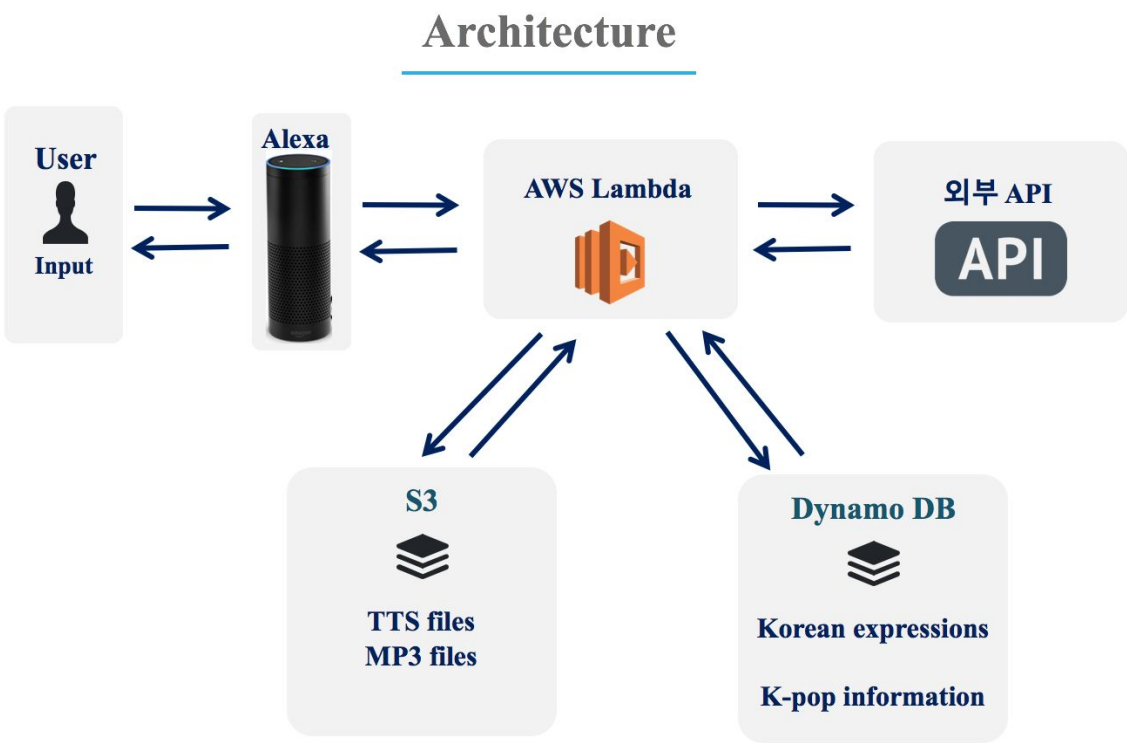
응답 시간의 경우 Alexa에서 설정해 놓은 타임아웃이 있는 것으로 확인되는데, 무엇보다 그 시간을 넘지 않아야 할 것이고, 구체적으로는 일반 네트워크 환경에서 5초 이내에 응답이 오는 것을 목표로 한다. Alexa에서는 현재 미국 동부의 버지니아 지역의 AWS람다만 지원하는데 국내의 다양한 서비스 API와 통신이 수반되어야 한다. 그리고 본 프로젝트를 평가하는 환경은 한국이라는 점을 통틀어 고려해 봤을 때 다수의 대륙간 통신이 필수불가결 하므로 5초는 납득 가능한 시간이라 판단된다.

정확도 부분에서는 본 프로젝트에서 alexa자체의 인식률을 높일 수는 없다. 따라서 낮은 인식률로 인한 오작동은 본 프로젝트에서는 손쓸 수 있는 부분이 없다. 따라서 기본적으로는,

alexa콘솔에서 제공하는 텍스트 입력 기반의 테스트 툴(음성 입력이 아닌 텍스트를 입력 받아 응답 형태를 확인할 수 있고, 실제로 재생할 수 있음)에서는 항상 정상적으로 기능이 동작하게 할 것이다. 그리고 실제로 음성 입력이 들어왔을 때는 utterance간 간섭이 최대한 일어나지 않도록 설정을 하여 원하는 기능이 70%이상의 성공률로 동작하도록 하고자 한다. 또, 기능별로 utterance의 충돌은 없되, 다양한 방식의 문장을 지원해야하므로 각 기능 당 5개 이상의 utterance를 설정하는 것을 목표로 한다.

5. Project Architecture

A. Architecture Diagram



B. Architecture Description

위의 그림은 본 프로젝트의 기본 구조를 보여준다.

사용자가 Alexa에 말을 하면, Alexa는 서버 역할을 하는 AWS람다에 입력된 문장을 전달한다. Lambda함수는 모든 로직을 처리하는 역할을 한다. 기능에 따라 세부 흐름은 상이하지만, lambda가 다른 lambda 혹은 외부의 API를 호출하여 로직을 처리한다.

S3는 파일들을 저장하는 파일 서버이다. 본 프로젝트에서는 한국어 문장을 TTS를 이용하여 만든 mp3파일, K-POP 음악 mp3파일들을 저장한다. 이는 추후에 alexa에서 url기반으로 음성 파일을 재생하는데 사용된다.

DynamoDB는 noSQL형태의 데이터 저장소이다. 가장 최근에 요청된 차트 정보 혹은 곡 정보

등이 필요에 따라 저장되어 상태를 저장하여 유용하게 쓰이거나, 중복된 API호출을 방지한다. 또한, API호출을 하지 않음으로써 속도의 개선효과도 기대할 수 있다.

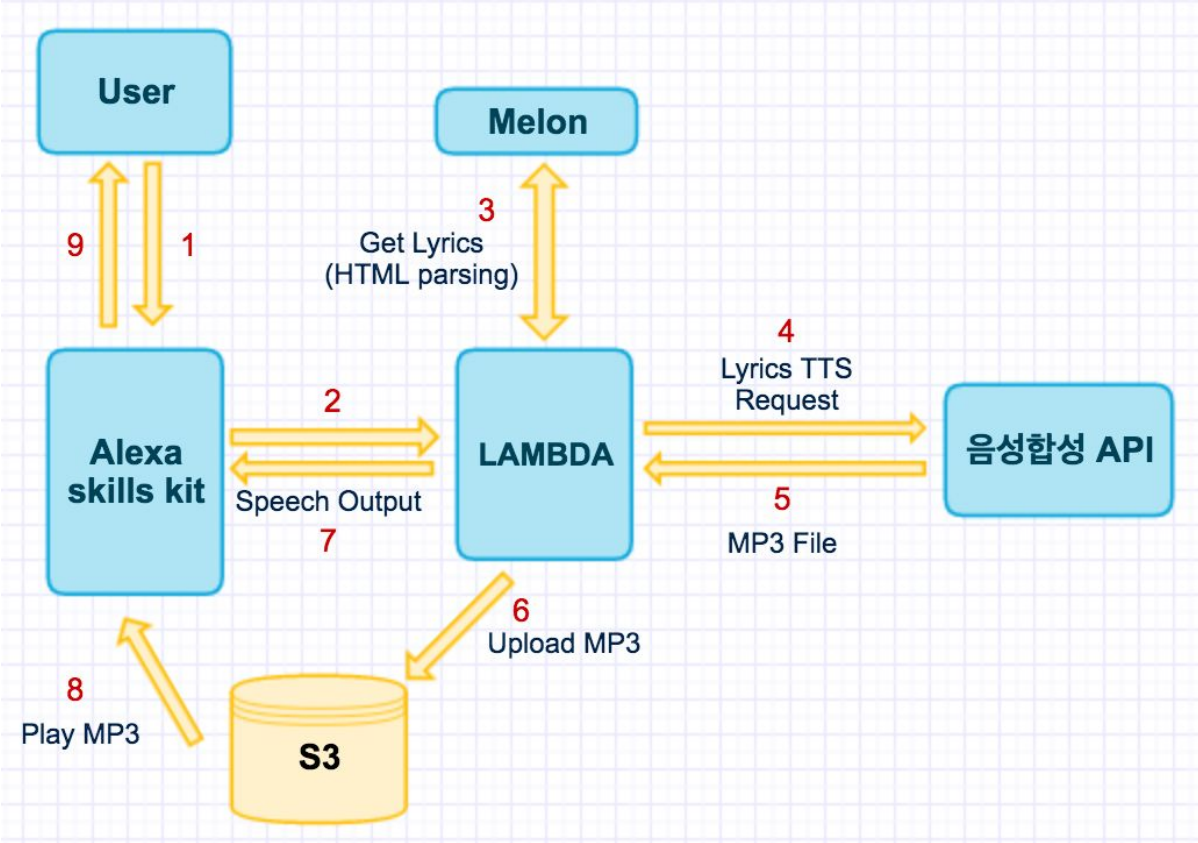
외부 API 또한 빠질 수 없는 요소이다. 본 프로젝트에서 제공되는 컨텐츠인 한국어 학습 컨텐츠에서 네이버의 기계 번역 API와 음성합성 API가 사용된다. 그리고 K-POP관련 컨텐츠는 melon API로부터 관련 데이터를 가져온다.

6. Implementation Spec

A. Input/Output Interface

입력과 출력은 모두 마이크와 스피커가 내장되어 있는 아마존 에코에 의해 이루어진다. 음성으로 입력이 들어오면 에코내의 음성인식 모듈이 text로 변환하여 API에게 요청을 한다. 출력은 두 가지 형식이 가능한데, 일반 평문과 SSML태그를 이용한 음성 파일 재생이 가능하다. 실제 음성의 출력은 아마존 에코 내의 TTS엔진에 의해 동작한다.

B. Inter Module Communication Interface



본 프로젝트를 구성하는 다양한 모듈은 기능에 따라 사용되기도 하고, 사용되지 않기도 한다. 또한 순서도 조금씩을 다를 수 있는데, 대부분의 모듈을 사용하는 하나의 기능을 예로 들어 설명하고자 한다. 위의 그림은 가사를 재생하는 기능의 순서도를 나타낸다.

위의 그림에서 1번은 사용자가 Amazon Echo에 말을 하여 입력이 들어오는 부분이다. Alexa는 음성을 텍스트로 변환하여 텍스트를 람다 함수에 가사를 들려달라고 요청한다(순서 2). 람다는 서버의

역할을 하는 모듈로 대부분의 로직을 처리한다. 우선 람다는 해당 곡의 가사를 Melon으로부터 받아온다(순서 3). 이렇게 가져온 가사는 대부분이 한글로 이루어져 있을것이다. 하지만 alexa는 한국어를 지원하지 않기 때문에 한국어 텍스트를 음성으로 변환하는 작업이 필요하다. 이 역할은 NAVER의 음성합성 API가 담당한다. 람다는 음성합성 API에 요청을 보내고(순서 4) 응답으로 MP3 파일을 받는다(순서 5). 람다는 다시 응답으로 반환받은 MP3 파일을 S3에 업로드한다(순서 6). 람다는 S3에 파일이 올라간 것을 확인하고 alexa에게 그 파일을 읽어주라고 응답을 보낸다(순서 7). alexa는 건네 받은 MP3 파일 주소를 통해 그 파일을 로드하여(순서 8) 사용자에게 들려준다(순서 9).

C. Modules

- RequestHandler
  - AlexaSkillsKit과의 통신에서 모든 입력과 출력을 관리하는 람다이다.
- PlayLyrics
  - 특정 곡 ID를 인자로 받아 해당 곡의 가사를 재생해준다.
- EmotionBasedRecommendation
  - 감정 상태를 인자로 받아 기분에 따라 적절한 곡을 랜덤으로 추천 재생한다.
- Translate
  - 인자를 받아 네이버 번역 API를 호출하는 람다이다.
- TTS
  - 한국어 인자를 받아 네이버 음성 합성 API를 호출하는 람다이다.
  - MP3 파일을 Alexa Playable Format으로 변환한다.
  - MP3 파일을 S3에 업로드한다.
- TranslateWrapper
  - Translate 및 TTS를 Wrapping한 람다이다.
  - Translate 및 TTS는 여러 기능에서 복합적으로 쓰이기 때문에 따로 구현하였으나 한국어 번역 질의에서는 이 Wrapper를 사용한다.
- KpopChart
  - 멜론 실시간 차트 API를 호출하는 람다이다.
  - 1~10위 까지의 차트 정보를 받아온다.
  - 차트 정보는 DynamoDB에 업데이트 한다.
- SpeakKpopChart
  - 차트 정보에서 음성합성할 곡 제목과 가수 이름을 정리하고 재생할 MP3 파일을 만들기 위해 TTS를 호출한다.
- GetOtherSongs
  - 가수 이름을 인자로 받아 멜론 곡 검색 API를 호출한다.
  - 차트에 없는 해당 가수의 곡 4개를 리턴한다.
- ArtistBasedRecommendation
  - 가수 이름을 인자로 받아 GetOtherSongs를 호출하고 받은 곡 정보에서 음성합성할 곡 제목과 가수이름을 정리하고 재생할 MP3 파일을 만들기 위해 TTS를 호출한다.

7. Current Status

현재까지 구현된 상황은 다음과 같다.

- **Alexa Skills Kit**

Alexa Skills Kit에서 미리 설정해 두어야 할 것은 크게 3종류로 나뉜다.

첫째는 Intent이다. Intent란 alexa에 들어온 입력이 어떤 종류의 기능에 대한 입력인지 구별할 수 있도록 해주는 String 값이다. 예를 들어, 가사를 재생해 주는 기능은 PlayLyricsIntent, 감정 기반의 곡 추천 기능은 EmotionBasedRecommendationIntent등으로 등록이 가능하다. 또 각 Intent에서 추후에 설명될 slot이 쓰인다면 그 부분도 명시가 되어야 한다. 이러한 Intent설정을 json형태로 만들어 둔다.

둘째는 슬롯(Slot)이라는 개념이다. 슬롯은 여러 가지의 어휘 혹은 문장을 하나의 카테고리로 묶는 것이다. 예를 들어, 사용자가 본인의 감정상태를 말하고자 할 때 사용될 수 있는 어휘는 수백 혹은 수천가지에 달할 것이다. 이러한 수백가지의 감정들을 'Emotion'이라는 슬롯으로 등록해 둘 수 있다. 어휘 리스트는 개행문자로 구분된 텍스트 형태로 저장된다.

마지막으로는 utterances다. 이는 각 기능별로, 즉 Intent별로 입력으로 들어와 매칭될 수 있는 모든 문장을 저장해둔다. 예를 들어, 현재 슬프다고 감정표현을 한다고 하면 'I am sad', 'I feel sad'등이 가능할 것이다. 이러한 문장들을 Intent에 매칭 시켜 저장해두는 설정이 utterances설정이다. 위의 예에서 sad대신에 들어올 수 있는 수많은 감정들은 {Emotion}과 같은 파싱 구분자로 모든 입력에 대해 일괄 처리가 가능하다.

- **RequestHandler람다 생성**

alexa skills kit의 모든 입력은 하나의 람다가 받아서 해당 람다가 기능에 따라 다른 람다를 호출하는 형식으로 설계하였다. 이러한 람다를 'ReqeustHandler(이하 핸들러)'라고 명명하였는데 Alexa는 응답으로 평문 혹은 SSML태그, 두가지의 형식을 지원한다. 평문은 단순히 람다함수에서 텍스트를 리턴하면 그대로 읽어주므로 큰 문제는 없다. 본 프로젝트에서는 Naver음성합성 API를 사용하여 생성된 mp3파일 혹은 K-POP관련 mp3파일들을 재생해야 하는 경우가 꽤 있었다. 이를 위해서 SSML 태그를 사용해야만 한다. 이런 출력 형식은 모든 람다에서 쓰이지만 중복된 코드가 모든 람다에 있는 것은 비효율적이므로, 핸들러에서 각 Intent별로 다른 형식의 응답을 반환할 수 있도록 설계하였다.

- **네이버 API Wrapping**
  - **a. 기계 번역**

네이버 기계 번역 API를 람다 함수에서 호출할 수 있도록 구현하였다. 네이버 비로그인 API의 경우 복잡한 인증 과정 없이 Client-ID, Client-Secret 상수값만 파라미터로 전달하면 동작하도록 되어 있다. Node.js의 HTTPS 모듈을 사용하여 Lambda가 Trigger될 때 그 인자값과 함께 기계 번역 API를 호출하고, 응답을 받아 콜백 함수에서 그대로 리턴한다. 인자와 리턴 모두 Plain Text로 큰 어려움은 없었다.

  - **b. 음성 합성**

음성 합성의 경우도 호출 자체는 기계 번역과 같이 쉽게 구현되었다. 그러나 음성 합성 API는 응답값이 MP3 바이너리 파일로, 이를 처리하는 데 있어 여러 기술적 이슈가 발생하였다.

우선 Lambda에서 MP3 파일을 받기 때문에, 로컬 환경처럼 즉시 파일의 무결성을 확인할 수 없었다. 그래서 파일 처리 메커니즘이 모두 구현된 뒤에야 디버깅을 할 수 있었는데, 버그 발생 시 어느

부분의 문제인지 확신할 수 없는 상황이 많았다.

Lambda는 서버가 아니므로 State를 가질 수 없고, 파일도 영구적으로 저장할 수 없다. 따라서 위에서 서술한 S3에 Upload하여야만 했다. 초기에는 MP3 스트림을 즉시 S3으로 돌림으로써 시간과 자원을 절약하였으나, 추후 Alexa Research가 진행되면서 Alexa에서 MP3을 재생하기 위해서는 특별한 변환 과정이 필요함을 알게 되었다.

Lambda에서 유일하게 접근 가능한 디렉토리는 /tmp/이다. (물론 State를 가지지 않으며 한 번 호출된 후 원칙적으로 사라질 수 있다.) 이곳에 MP3 파일을 저장한 후 ffmpeg라는 미디어 처리 라이브러리를 활용하여 변환 과정을 거쳤다. Lambda에서도 ffmpeg 같은 External Library를 사용할 수 있는데, 패키지에 같이 묶어서 업로드한 후 특정 경로로 접근하는 방식이었다. 물론 그 라이브러리에서 참조하는 라이브러리들은 Static으로 함께 빌드해야 한다.

그런데 팀원들의 Mac/Linux 환경에서는 잘 변환되는 MP3 파일이 Lambda에서는 간헐적으로 변환 실패하는 현상이 나타났다. 서버에 직접 접근할 수 있었으면 문제 해결이 손쉬웠을텐데, Lambda의 로그를 통해서만 간접적으로 상황을 추측할 수 있어 굉장히 오랜 시간이 소요되었다. 이 부분은 Lambda 사용의 단점이라고 할 수 있겠다. 결과적으로 드러난 문제는 크게 두 가지였는데, Output file이 이미 존재할 경우 무한정 Hang이 걸리는 현상 및 정상 파일 변환에서 ffmpeg crash가 발생하고, Lambda 실행도 멈추는 현상이었다.

두 가지 모두 로컬 환경에서는 전혀 발생하지 않는 이슈이고, ffmpeg 역시 다양한 미디어 플레이어에서 사용되는 유서 깊은 라이브러리라 예상치 못한 상황이었다. 그러나 당장 Amazon Linux에서 디버깅을 진행할 수 없었기에 이 부분은 우회하여 해결한 뒤 Future Plan으로 남기고자 한다. Output file이 이미 존재하는 경우 지워주었고, ffmpeg이 crash가 난 경우에는 반복해서 실행하도록 하였다. 결과적으로 거의 모든 경우에 1-2번의 변환 과정 안에 해결되었다.

```
25 function uploadAndClean(tempName, fileName, callback) {
26     var s3 = new aws.S3();
27     var file = fs.createReadStream(DEFAULT_PATH + fileName);
28     var s3param = {
29         Bucket: 'koreantts',
30         Key: fileName,
31         Body: file,
32         ACL: 'public-read'
33     };
34     s3.upload(s3param, function(err, data) {
35         if (err) console.log('S3 upload error : ' + err, err.stack);
36         else console.log('S3 upload succeed : ' + data);
37 
38         cp.exec('rm -rf ' + DEFAULT_PATH  + tempName + ' ' + DEFAULT_PATH + fileName,
39             function(error, stdout, stderr) {
40                 if(error)
41                     console.log('deleting temp error : ' + error);
42                 else
43                     console.log('deleting temp succeed');
44                 callback(null, 'https://s3.amazonaws.com/koreantts/' + fileName);
45             }
46         );
47     });
48 }
49 
50 function convertMP3(tempName, fileName, callbackAfterUpload) {
51     cp.exec(
52         DEFAULT_PATH + 'ffmpeg -i ' + DEFAULT_PATH + tempName
53         + ' -analyzeduration 10000000 -probesize 100000000 '
54         + ' -ac 2 -codec:a libmp3lame -b:a 48k -ar 16000 -ss 00:00:00 -t 00:01:25 '
55         + DEFAULT_PATH + fileName,
56         function(error, stdout, stderr) {
57             if(error) {
58                 console.log('executing ffmpeg error : RETRY');
59                 convertMP3(tempName, fileName, callbackAfterUpload);
60             } else {
61                 console.log('executing ffmpeg');
62                 uploadAndClean(tempName, fileName, callbackAfterUpload);
63             }
64         }
65     );
<s.js   CWD: /Users/yeongjin/Dropbox/SNU/2016F/Project/kalexa   Line: 58   Column: 4   49%
```

- **멜론 API**
  - **a. 실시간 차트**

멜론 실시간 차트 API를 람다 함수에서 호출할 수 있도록 구현하였다. SK플래닛 개발자 센터에서 제공받은 App key 로 인증이 가능하고 파라미터로 version(현재 1 고정값), page(받을 page 번호), count(받을 곡의 수) 를 전달하여 API를 호출하면 그에 따른 실시간 차트 정보를 응답받는다. 프로젝트에서 제공할 차트범위는 1위부터 10위까지(page = 1, count = 10)로 정했다. 응답받는 파라미터에는 곡에 대한 여러가지 정보가 모두 포함되어 있지만 그중에서 이후 기능 구현에 필요한 songId, songName, artistId, artistName을 담은 song object의 list로 정리하여 JSON 형태로 리턴하도록 했다. 또한 이번 프로젝트에서 K-pop 관련 기능을 구현할 때 실시간 차트를 이용하는 부분이 상대적으로 많이 발생한다. 따라서 실시간 차트는 실시간으로 순위가 반영이 되는 data지만 실제 현실에서 차트 순위의 변경이 실시간으로 계속 뒤바뀌지 않고 일정 기간은 순위정보가 유지된다는 것을 고려하여 매번 실시간 차트 API를 호출하지 않고 DB에 저장된 차트의 정보를 활용할 수 있도록 차트 정보를 리턴하기 전에 실시간 차트 정보를 DynamoDB에 업데이트 하도록 구현하였다.

  - **b. 곡 검색**

멜론 곡 검색 API를 호출하는 람다함수를 구현하였다. 실시간차트와 호출방법은 유사하며 다른점은 search key 파라미터가 추가되어야 한다는 것이다. search key의 값은 임의의 String 값이

들어갈 수 있다. 이후 목차 F에서 설명할 특정 가수의 다른 노래는 어떤 것들이 있는지에 대한 정보를 제공하는 기능을 구현하기 위해 해당 API를 사용한다. count는 5로 설정하여 검색된 목록에서 5개의 곡의 정보를 응답 받도록 구현하였다.

- **차트 정보 제공**

실시간 1~10위 차트에 올라온 곡의 제목, 가수 이름을 alexa가 응답 해주도록 한다. 곡의 제목과 가수의 이름은 대부분 한글로 되어 있으며 이를 alexa가 응답해주기 위해서는 음성합성을 이용하여 곡 제목과 해당 가수의 이름을 mp3 파일로 저장하여 이를 재생하는 방법으로 진행되어야 한다.

곡 제목과 가수이름에는 부연 설명을 위해 특수문자가 포함되어 있거나 중복되는 정보가 있는 경우도 있다. 예를 들면 “돌아오지마 (Feat. 용준형 Of 비스트)”와 같은 곡 제목 또는 “헤이즈 (Heize)”와 같은 가수이름이 있다. 이런 경우 그대로 음성 합성을 하면 실제 곡 제목인 “돌아오지마” 뿐만아니라 뒷 내용모두 음성합성이 되어 이를 출력할 경우 의아한 곡 제목으로 들릴 수 있다. 후자의 경우 음성합성을 하면 같은 내용이 두번 반복되어 출력되므로 부자연스러워 진다. 따라서 음성 합성을 하기 전에 곡 제목과 가수 이름 String에 대해 적절한 subsrting을 만드는 작업이 필요하다. 이에 대한 현재 구현 방법은 “(“가 포함되어 있을 경우 그 전까지의 String을 받아들이도록 구현하였다. 이러면 중복되는 정보는 제거할 수 있다. 하지만 피처링을 한 가수 정보가 괄호 안에 있었다면 그 정보는 손실된다. 보통 어떤 곡에 여러 가수가 참여할 경우 가수들의 정보가 배열로 들어있다. 하지만 위에 예시와 같이 곡 제목에만 그 정보를 써두고 가수정보에는 없는 경우도 있다. 이럴 경우 현재 까지는 ‘피처링을 누가 했느냐에 대한 정보는 차트 정보 제공의 큰 틀에서 보았을 때 생략할 만한 정보이다’라는 가정을 하고 구현하였다. 하지만 추후에 가수정보 배열에 곡 제목에 있는 정보를 추가하여 기존 data를 수정하는 방안 등 보완할 수 있는 방안들을 논의해볼 예정이다.

곡 제목과 가수 이름을 음성합성 할 때 이미 S3에 저장된 정보가 있는지 판별하기 위해 mp3 파일 제목인 key값을 정한다. songId 와 artistId 를 활용하여 unique한 파일 이름을 만든다. 따라서 음성합성을 하기 전에 Id값으로 mp3파일 이름을 도출하고 S3에 파일 존재여부를 확인한 뒤 존재 하지 않을 경우에만 음성합성 람다 함수를 호출하여 음성변환 후 S3에 새로운 mp3 파일을 업로드 한다. 이미 존재하는 파일이라면 음성 합성을 호출할 필요없이 바로 S3에 접근하여 mp3파일을 가져온다.

처음에는 alexa에게 ‘차트를 알려줘’와 같은 미리 설정해놓았던 말을 하면 alexa가 1위부터 10위까지 순서대로 곡 제목과 해당 곡의 가수이름을 순차적으로 응답하도록 구현하였다. 하지만 alexa는 한 응답에 최대 5개의 mp3 파일만 재생시킬 수 있다는 한계가 있다. 따라서 곡 제목 10개, 가수이름 10개 총 20개의 mp3 파일을 재생해야 하는 차트 정보 제공 기능은 불가능하다. 따라서 응답하는 종류를 세분화 하여 재구현 하였다. {fromSixth}라는 slot과 {Nth}라는 slot을 추가하고 사용자가 요청한 문장에 두 slot이 없을 경우는 1~5위의 곡 제목만 출력하도록하고 {fromSixth} slot이 있을 경우 6~10위의 곡 제목을 출력한다. {Nth}라는 slot이 있으면 해당 순위에 있는 곡 제목과 가수이름을 출력하도록 설계하였다.

- **차트 순위에 있는 가수의 순위곡 이외의 다른 노래 정보 제공**

사용자가 ‘몇번째 가수의 다른 곡 알려줘’와 같은 형태의 요청을 하면 alexa는 해당 가수의 노래 최대 4개를 제공한다. 노래가 최대 4개 인 이유는 이전 항목에서 설명한 alexa의 재생가능한 최대 mp파일 갯수가 5개라는 점이기 때문이다. 곡 제목 4개와 나머지 1개는 해당 가수의 이름이다.

이를 구현하기 위해서는 곡 검색 API 호출 람다 함수를 사용한다. Intent에 {Nth} slot 옵션을 추가하여 사용자가 몇번 째를 원하는 지에 대한 정보를 파라미터로 받고 DynamoDB에 저장되어 있는 차트 정보를 불러와서 해당 순위의 곡 정보를 가져온다. 곡 정보를 곡 검색 API 람다 함수로 넘겨주고 artistName을 search key로 하여 곡 검색 API 호출을 한다. 곡 검색 API호출 람다 함수에서 최대 5개의 곡을 가져오는 이유는 차트에 있는 곡도 검색 목록에 포함될 수 있기 때문에 넘겨받은 songId와의 비교를

통해서 이를 제외하고 4개 목록을 가져오기 위함이다. 이렇게 해당 가수의 곡 4개의 정보를 받으면 각 노래의 제목을 음성합성한다. 이 때에도 마찬가지로 S3에 존재하지 않는 파일일 경우에만 업로드하도록 구현했다. 만일 해당 가수의 검색 결과가 없을 경우 alexa는 '다른곡이 없다'와 같은 응답을 하고 그렇지 않을 경우 가수이름과 4개의 곡 제목을 순차적으로 응답해준다.

- **노래 재생 기능**

차트 정보나 가수의 다른 곡을 alexa가 재생시켜주는 기능이다. '차트 몇번째 노래 재생해줘' 또는 '몇번째 순위 가수의 다른 곡 재생해줘'와 같은 요청을 하면 alexa는 그에 대한 곡을 재생한다. 이 기능에서는 {Nth} slot 과 {NotinChart} slot 옵션을 가진다. {Nth} slot 정보만 있으면 사용자가 몇번 째를 요청하는건지에 대한 정보를 받고 DynamoDB에 저장되 있는 차트 정보를 가져와서 해당 순위 songId를 이용해 곡을 재생한다. {Nth} ,{NotinChart} slot 정보가 모두 있는 경우 차트 정보에서 해당 순위 가수이름에 대해 곡 검색을 한 후 제일 검색된 곡중 제일 상위 곡의 songId를 받아 곡을 재생시킨다.

- **사용자의 감정 상태에 따른 추천**

사용자가 현재 자신의 감정 상태를 입력했을 때 alexa가 감정 상태에 따라 지정된 곡들을 추천하는 기능이다. 약 200여 가지의 감정을 지원하고, 실제 처리부분에서는 4가지(희, 로, 애, 락)로 구분한다. 선곡은 이미 선별해서 만들어 둔 리스트에서 랜덤 재생된다. 또한, 해당 추천곡이 사용자의 마음에 들어, 가사 재생 기능을 요청할 수 있는 상황을 대비하여 마지막으로 재생된 곡의 정보를 DB에 저장해둔다. 만약 사용자가 '이 곡의 가사를 알려줘'와 같은 말을 Alexa에게 한다면 가사를 재생한다.

- **가사 재생 기능**

특정 곡을 골라 가사를 재생하는 기능이다. 곡의 고유 ID인 songId가 입력으로 들어오면 해당 곡의 가사 정보를 멜론 홈페이지에서 크롤링한다. 그렇게 가져온 가사를 음성합성 API를 호출하여 mp3파일로 만들어 내고 Alexa에게는 해당 파일을 재생한다. 다만 이미 가사mp3파일이 있는데 음성합성 API를 호출하게 되면 불필요하게 중복된 파일이 S3에 쌓이게 되고, 속도 또한 느릴 수 밖에 없다. 이를 방지하기 위해 해당 곡 가사 재생 최초 요청시에 songId와 url 정보를 dynamoDB에 저장한다. 추후에 같은 요청이 들어오게 되면 DB에 저장된 정보가 있는지 확인하고 없을 경우에만 음성 합성 API를 호출한다.

- **한국어 번역**

한국어 번역은 Alexa에서 영어 표현의 한국어 번역을 질의할 수 있는 기능으로, 현재 부분적으로 완료되었다. Alexa에게 어떤 표현의 한국어 번역이 무엇인지 물어보면, 내부적으로는 해당하는 람다 함수가 호출된다. Lambda에서는 네이버 기계 번역을 통해 문자열을 한국어로 번역하고, 이를 다시 음성 합성으로 한국어가 재생되는 MP3을 생성한다. Alexa에 이 MP3 파일을 전달하여 마침내 유저는 한국어 번역 결과를 들을 수 있다.

기계 번역 및 음성 합성 API 부분을 모듈화하여 따로 Lambda 함수로 만들어 놓았기에, 이를 가져다 쓰는 것은 간단한 일이었다. 기본 동작을 구현한 뒤 Alexa가 알아들어야 하는 질의들을 추가했다. Alexa Interaction Model에서 질의에 해당하는 Intent와, 그 내부 번역 대상에 해당하는 Slot들을 채워주었다.

그러나 현재 Alexa는, 미리 정의된 문자들만을 변수로 받을 수 있다. 즉 한국어 번역을 위해서는, 사용자가 물어볼 만한 단어와 문장들을 '모두' 정리해서 저장해놓고 있어야 하는 것이다. 심지어 단어들을 저장한 후 이에 기반한 문장들을 받는 형식도 지원하지 않는데, 이론적으로 이 서비스를 상용화하기 위해서는 천문학적인 리스트가 필요할 것이다. 이 부분은 본 프로젝트의 난점으로, Future Plan에서

아마존 측에 제안해볼 수 있는 내용이다.

지금은 동작 테스트를 위한 기초 영어 단어 1000여 개로 이루어져 있다.

- **기타 설정**

람다가 아마존에서 제공하는 기본 패키지 외의 패키지를 사용하고자 하는 경우에는 소스파일과 해당 패키지를 같이 압축해서 람다를 생성해야 한다. 이와 같은 상황을 고려하여 하나의 람다별로 하나의 폴더를 갖는 구조를 갖게 하였다. 이 과정에서 생성되는 패키지나, 압축파일은 공유될 필요가 없으므로 gitignore에 추가하여 버전 관리가 되지 않도록 하였다.

8. Future Work

추가적인 구현 계획은 다음과 같다.

- 한국어 퀴즈 기능 구현
- 캐싱 적용으로 성능 향상
- K-Pop추천 알고리즘 구현
- 성능 개선 및 중간 prototype과 향상도 비교 및 분석
- 구현 시 발생한 문제 및 개선점 해결
    - FFMPEG 모듈 분리
    - FFMPEG Crash 이슈
- Alexa의 한계점 분석 및 제안
    - 임의의 음성을 모두 써 주어야 인식할 수 있는 문제

9. Division & Assignment of Work

항목	담당자
Alexa Skills Kit intent 및 utterance 설정	팀 전원
기계 번역 API 호출 Lambda function 구현	최영진
음성 합성 API 호출 Lambda function 구현	최영진
DB modeling	팀 전원
Learning Korean 서비스 구현	최영진
차트 상위	이상훈
해당 곡 아티스트의 다른 앨범 및 노래	이상훈
가사	김준혁
감정 상태에 따른 곡 추천	김준혁
추천 알고리즘 구현	김준혁, 이상훈
Data cache 구현	팀 전원
기능별 성능 개선	팀 전원

Prototype과 결과물의 성능 비교 및 분석	팀 전원

10. Schedule

내용	9월			10월				11월				12월	
사전지식 이해 및 개발 환경 구축			9/30 스펙 발표										
Protoype 개발 Alexa Skills Kit intent 구현													
기계 번역, 음성 합성 API 호출 Lambda function 구현 DB modeling													
K-pop 컨텐츠 구현													
Learning Korean 컨텐츠 구현							10/28 중간 발표						
한국어 퀴즈 구현													
추천 알고리즘 구현													
Data caching 구현													
소스 리팩토링													
Test 및 데모													
최종 발표 준비													

11. Demo Plan

Alexa skills kit에서는 실제 amazon echo디바이스를 대신하여 웹 환경에서 테스트할 수 있도록 페이지를 제공한다. 현재까지 개발된 기능은 K-pop관련 컨텐츠가 대부분인데 그 중 일부와 한국어 번역 기능을 중간 발표 중에 간단히 시연해볼 예정이다.

최종 시연시에는 실물 아마존 에코를 활용하여 시연을 해볼 예정이다. 중간 발표 때 이미 시연한 기능을 제외하더라도 시간 관계상 모든 기능을 시연할 수 있을지는 의문이 들어 우선순위를 설정해 둘

예정이다. 우선 한국어 학습과 관련된 기능을 먼저 시연코자 한다. 한국어 번역 기능, 한국어 퀴즈 기능을 모두 시연할 것이다. K-pop컨텐츠 관련 기능 중에서는 기본 차트 정보 제공 기능, 차트내 특정 곡 재생 기능, 가사 제공 기능, 임의 곡 추천 기능, 아티스트의 다른 곡 정보 제공 기능, 감정 기반 추천 곡 재생 기능 순으로 시간이 허락하는 선에서 시연을 할 계획이다.

◆ [Appendix] Detailed Implementation Spec

A. KalexaRequestHandler

AlexaSkillsKit과의 기능별 람다 사이에서 게이트웨이 역할을 하는 함수이다. AlexaSkillsKit에서 보내는 request를 파싱하여 각 Intent별로 람다 호출을 한다. 또한, 기능별 람다가 리턴한 결과를 AlexaSkillsKit의 형식에 맞게 응답하는 wrapping함수를 포함한다.

B. PlayLyrics

Parameter	@int : 재생하고자 하는 곡의 songId
Return	String : 가사 파일의 url
Implementation	songId에 해당하는 곡의 가사 파일이 이미 만들어있는지 DB확인. - 있을 경우 url 그대로 리턴 - 없을 경우 songId로 멜론 홈페이지에서 HTML파싱하여 가사 획득. 획득한 가사로 TTS람다 호출 결과로 리턴 받은 mp3파일url DB에 저장 후 해당 url 리턴
Additional Info	

C. EmotionBasedRecommendation

Parameter	@text string : 감정을 나타내는 어휘 혹은 표현
Return	String : mp3 file url
Implementation	입력된 감정의 상위 분류(희로애락)의 선곡 리스트 중 랜덤 곡 리턴
Additional Info	

D. Translate

Parameter	@text string English string to translate into Korean
Return	Translated Korean string

Implementation	네이버 기계 번역 API 호출 (source: en, target: ko, text: @text) UTF-8 인코딩 사용
Additional Info	

E. TTS

Parameter	@text string Korean string to speech @fileName string (optional) S3 upload file name Default: temporary name by datetime
Return	URL of MP3 which reads given string parameter (Playable in Alexa)
Implementation	네이버 음성 합성 API 호출 (speaker: Mijin, speed: 0, text: @text) UTF-8 인코딩 사용 FFMPEG 라이브러리 사용 Alexa 플레이 가능한 포맷으로 변환 <FFMPEG Options> -ac 2 -codec:2 libmp3lame -b:a 48k -ar 16000 -ss 00:00:00 -t 00:01:25
Additional Info	내부적으로 디렉토리 처리 등 부가 로직 FFMPEG 실패 시 재시도

F. 한국어 번역

Parameter	What is {sentence} in Korean? 등의 한국어 번역 질의
Return	한국어 번역 결과 재생
Implementation	Alexa에서 번역 질의를 받으면 번역 해당 부분 파싱하여 전달 Translate, TTS를 거쳐 한국어 MP3 파일 생성 MP3 파일 URL Alexa에 전달하여 재생
Additional Info	{sentence}는 Predefined Set

G. KpopChart

Parameter	
Return	차트 정보 - list of song objects { songId(int) , songName(string) , artists(list) } artists : list of artist objects { artistId(int), artistName(string) }
Implementation	멜론 실시간 차트 API 호출

	(version: 1, page: 1, count: 10) 차트정보 DynamoDB에 update
Additional Info	

H. SpeakKpopChart

Parameter	
Return	차트 정보
Implementation	KpopChart 호출하여 실시간 차트 정보 받음 차트 정보에서 songName 과 artistName을 음성 변환에 적절한 String으로 변경 TTS에 넘겨줄 @fileName 값을 “songId” + “1”, “artistId” + “1”로 설정 TTS 람다 호출
Additional Info	S3에 해당 MP3 파일이 없는 경우에만 TTS 람다 호출

I. 차트 정보 제공

Parameter	1. I want to know the chart 등의 1~5위 차트 정보 요구 질의 2. Let me know the chart {fromSixth} 등의 6~10위 차트 정보 요구 질의 3. Who is the {Nth} song’s artist 등의 가수 정보 요구 질의
Return	1. 차트 1~5위 곡 제목 재생 2. 차트 6~10위 곡 제목 재생 3. 해당 순위 곡 제목과 가수 이름 재생
Implementation	SpeakKpopChart 람다 함수를 호출하여 차트 곡 제목과 가수 이름 S3에 갱신 리턴 받은 차트 정보로 곡 제목과 가수 이름 MP3 파일의 URL 도출 질의 종류에 따라서 재생할 MP3 URL을 Alexa에 전달
Additional Info	{fromSixth}: 6위부터 라는 의미를 가진 표현 {Nth}: 몇번째인지를 나타내는 표현

J. GetOtherSongs

Parameter	@songId (int), @artistId (int), @artistName(string)
Return	list of at most 4 songs
Implementation	멜론 곡 검색 API 호출 (version: 1, page: 1, count: 5, searchKeyword: @artistName) API호출로 받은 5개의 곡 리스트에서 해당 곡 songId와 @songId의 비교를 통해 차트에 있지 않은 노래 구별하고 4개만 리턴 4개 이하일 경우 songId와 @songId 비교 후 모두 리턴

Additional Info	

K. ArtistBasedRecommendation

Parameter	@nth (int)
Return	list of at most 4 songs
Implementation	DynamoDB에서 차트 정보 받아서 @nth index에 있는 곡 정보 획득 GetOtherSongs 람다 함수 호출하여 추천 곡 list 받음 곡 정보에서 songName 과 artistName을 음성 변환에 적절한 String으로 변경 TTS에 넘겨줄 @fileName 값을 “songId” + “1”, “artistId” + “1”로 설정 TTS 람다 호출
Additional Info	S3에 해당 MP3 파일이 없는 경우에만 TTS 람다 호출

L. 순위권 가수의 차트에 없는 다른 곡 정보 제공

Parameter	I want to know other song of {Nth} artist 등의 다른 곡 추천 요구 질의
Return	해당 가수의 이름과 최대 4개의 곡 제목 재생
Implementation	ArtistBasedRecommendation 람다 함수 호출 하여 곡 리스트 획득 곡 정보에서 songId와 artistId를 이용하여 해당 MP3 URL 도출 MP3 URL을 Alexa에게 전달
Additional Info	{Nth}: 몇번째인지를 나타내는 표현

M. I, L에서 제공된 노래 재생

Parameter	1. Play {Nth} song in the chart 등의 차트 해당 순위 곡 재생 질의 2. Play {NotinChart} song of {Nth} artist in the chart 등의 해당 가수의 차트에 없는 곡 재생 질의
Return	1. 해당 차트 순위 곡 재생 2. 해당 차트 순위 가수의 또다른 곡 재생
Implementation	{NotinChart}가 없는 경우 DynamoDB에서 차트 정보 받아서 {Nth} 번째 곡 songId로 음원 MP3 URL 도출 후 Alexa에 전달 {NotinChart}가 있는 경우 {Nth} 정보를 해당 int 값으로 변환 후 ArtistBasedRecommendation 람다 함수 호출하여 곡 list 획득 곡 list에서 임의의 songId를 선택하여 음원 MP3 URL 도출 후 Alexa에 전달
Additional Info	{Nth}: 몇번째인지를 나타내는 표현 {NotinChart}: 차트에 없는 다른 곡을 의미하는 표현