**Fun Translate:**

The “Fun Translate” Alexa skill uses Amazon Polly, DynamoDB and Amazon S3 and the FunTranslations API to translate a phrase the user says, to the target language that has been pre-set by the user. The basic architecture of the skill is shown in the architecture diagram below and the flow of the skill is as follows: first the skill is launched, and Alexa asks the user to select a language as the target translation language. The user can choose between Dothraki, Shakespearean and Pig Latin as their target language. Once they have chosen their language, the user will then be able to translate any phrase or sentence they want by saying “translate {phrase}”. The user can also repeat the translation if they wish or ask what the target language has been set to. If the user tries to translate something before a target language has been set, the response will prompt the user to first select the target language before trying to translate something.

The entirety of the skill’s design was based around the API, as the API was the first decision made regarding the skill. The FunTranslation API was selected as it seemed like a fun API to work with. Once this was selected, I referred to the existing Alexa skills tutorials to start building the invocation model and the basic handlers e.g. LaunchRequestHandler, SessionEndedRequestHandler, HelpIntentHandler etc. The invocation model contains (alongside the standard intents): TranslateIntent (invoked when users wish to translate a phrase), SetLanguageIntent (invoked when users wish to set a target language), AskSetLanguageIntent (invoked when users need to enquire which target language is set) and the overridden standard RepeatIntent (invoked when users ask to repeat the previously spoken phrase).

A Lambda function was chosen as the endpoint for the Alexa skill as Lambda can easily host the code for the handlers and be plugged into the Alexa Developer Console as an endpoint for the Alexa skill. The process utilises Amazon Polly for the text-to-speech conversion, Amazon S3 to store the audio files and DynamoDB to store the S3 URL’s and translation, corresponding to the original sentence the user asked to be translated and the language they chose to translate it to.

The Amazon products were chosen as they were easily accessible through the boto3 library and thus could integrate easily with the Lambda function. Though particularly, the choice to use Amazon Polly was made as the service provided a variety of voices the output audio could have. This meant that the voice could be adjusted for each of the target translation language. Additionally, the decision to add in the DynamoDB table to store the URL for the S3 audio file was made to avoid having to run through the steps required for translation every time. This was also necessary to overcome the limitation posed by the API, which could only be called 5 times in the space of an hour. As previously translated phrases could be accessed through the table instead, there would be no need to call the API repeatedly.

A close up of a map

Description automatically generated

**Basic structure of skill:**

1. User speaks to VUI
2. VUI picks up utterance and triggers AWS Lambda function
3. Lambda function calls FunTranslations API and API sends back translated phrase
4. Lambda provides translated phrase to Amazon Polly to covert from text to speech, and Polly outputs the audio file
5. The audio file is uploaded to the S3 bucket
6. The S3 file location is uploaded to DynamoDB**\***, alongside the translated phrase
7. The audio file is then accessed from the S3 bucket
8. The response is sent back to the Alexa Skill VUI
9. The response is repeated back to the user

**\*** If the phrase has already been translated previously, then steps 3-5 are skipped and the audio file location is accessed directly from the DynamoDB table instead