Utopia, an Alexa Skill

Kevin Chuang (kevin.chuang@sjsu.edu), Rohit Sharma (rohit.sharma02@sjsu.edu), Viraj Upadhyay (viraj.upadhyay@sjsu.edu), and Varun Khatri (varun.khatri@sjsu.edu)

Computer Engineering Department, San Jose State University, SJSU
San Jose, California 95112

***Abstract*—This paper will describe programming an Alexa skill using Python, Flask Ask (an extension of Flask, a micro web framework), and the Alexa Skills Kit (ASK). The name of this Alexa skill is Utopia, and it has been designed to help people deal with depression. This report will describe the architecture of the Alexa skill, the reasoning behind the developing this skill, and the technologies used for developing, testing, continuous integration, and code coverage in this project. Most importantly, this paper will illustrate the features within this Alexa skill and go over the details of how those features work. Lastly, the paper will briefly describe the basic usage of the Alexa skill.**

***Index Terms* —Alexa Skills Kit (ASK), Flask, Flask Ask, Python, YAML, JSON, Continuous Integration, Travis CI, Code Coverage, Codecov.io, Google Places API, Hamilton Depression Rating Scale Survey, Jinja Templates, Speech Synthesis Markup Language (SSML)**

I. Introduction

According to the World Health Organization, 350 million people worldwide suffer from depression. Depression is the leading cause of disability.

This skill is meant to help people who are going through depression. The main feature of this skill involves taking the Hamilton Depression Rating Scale Survey to get a general idea of the severity of depression, and based on the score of the survey, the skill would recommend certain natural remedies to help. The other complementary features include giving positive and motivational quotes (other types of quotes are also supported), proposing natural solutions, listening to a collection of uplifting and powerful poems, recommendations for nearby therapists, and, if necessary, getting help from the National Suicide Prevention Lifeline.

This skill was named Utopia, because Utopia means a perfect world. With that being said, it is our hope that within this Alexa skill, a user may find his or her perfect world.

Disclaimer: Utopia is not meant to be a substitute for professional medical advice, treatment or diagnosis. It is more of a supplemental and informational helper to mitigate depression. The people who programmed this are not experts in mental illnesses such as depression, and this is a product of our subjectivity and couple months of basic research on depression.

II. Architecture

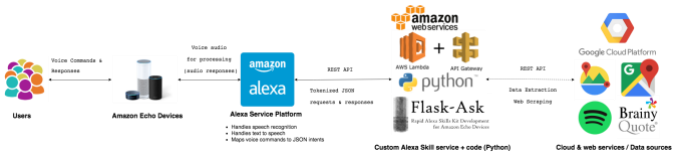

Fig. 1 Utopia Alexa Skill Architecture

For a better-quality picture, please see Reference [14].

The high level overview of how the Utopia Alexa skill works is simple:

1. A user talks to an Alexa supported device, and that speech is sent to the Alexa Service Platform.
2. The Alexa Service Platform processes the speech signal by using speech recognition and then translating the speech to text. It also maps voice commands to JSON intents, so that certain phrases/utterances trigger certain intents.
3. The Alexa Service Platform sends a JSON request via a REST API to the endpoint where our Utopia application is.
4. Our Utopia application receives the JSON request, and does some processing, such as extracting data from Google Places API or web scraping from various websites.
5. After our Utopia application is done processing, it will send a JSON response back to the Alexa Service Platform.
6. The Alexa Service Platform will translate the JSON response from text to speech, and send the speech signal to the Alexa supported device. Optionally, the Alexa Service Platform will send descriptive, informative cards to the Alexa app located on phones or tablets.
7. The Alexa supported device will play the audio back to the user.
8. Depending on whether the session is kept open or explicitly ended, the Alexa device will either await for a verbal response from the user or end gracefully.

III. Features

The features provided in Utopia are listed below:

- Hamilton Depression Rating Scale Survey consisting of 15 questions to analyze and evaluate level of depression.
  - Questionnaire is designed for adults, and asks questions regarding guilt, suicide, insomnia, agitation, anxiety, somatic symptoms and weight loss.
  - For the full list of questions, see Reference [2] below.
  - Added three additional bonus questions that use NLTK sentiment analysis to contribute to the evaluation of depression
    - Bonus Question #1: What are the first couple of words that come to your mind?
    - Bonus Question #2: What are some adjectives that describe your past week?
    - Bonus Question #3: What are some adjectives you would use to describe yourself?
- Listen to different categories of quotes from BrainyQuote.com
  - A few popular categories of quotes include positive, motivational, inspirational, family, love, and positive.
  - Implemented this via a web scraper using Python library beautifulsoup4 and Python library requests.
- Listen to advice and ideas for activities to improve mood
  - Examples include:
    - *Listen to music that makes you feel good.*
    - *Take note of all the small things you've accomplished today.*
- Listen to a collection of uplifting & powerful poems
  - Poems were carefully selected to motivate and inspire people
- Recommend therapists nearby by providing contact information, current availability, and open hours
  - Google Places API was used to help recommend therapists nearby
  - This feature will find nearby available therapists, and if there are no open therapists nearby, it will default to closed places, or places that provide no information regarding opening hours
  - The motivation behind this was to quickly find a therapist that was currently available to promptly provide assistance to depressed individuals.
- National Suicide Prevention Lifeline
  - If certain words are said, this feature will automatically trigger and provide user with contact information for the suicide hotline.
  - For a full list of trigger words, see Reference [13].
  - The trigger words were implemented as a custom slot type within the Alexa Interaction Model.

IV. Technology

Utopia is powered by Python, Flask-Ask (an extension of Flask, a micro web framework), and Alexa Skills Kit.

A. Flask-Ask

For this project's backend, the 3rd party Python package Flask-Ask is used. Flask-Ask is an extension of Flask, the micro framework based on Werkzeug and Jinja 2. Flask implements the WSGI (Web Service Gateway Interface) application protocol. WSGI is a standard for web servers to forward requests to web applications or frameworks written in Python (Reference [15]). The way it handles a request is by an environment dictionary variable that describes the request data, and returns a Response object. It does not care where the request came from or where there response goes. Those details are handled by WSGI server protocol. The Werkzeug servers handle the WSGI server protocol. Flask abstracts away all the complications of web development, and packages it nicely into an easy-to-use library. See Reference [16] for more information on Flask.

Flask-Ask handles a lot of things and is built on top of Flask. It provides a simple, intuitive way to develop Alexa skills using Flask by handling the boilerplate, so users can focus on writing clean code. Flask-Ask is used to route and map Alexa requests and intents to view functions. It also helps construct ask and tell responses, re prompts and cards. Session management is also made simpler. With support for Jinja templates, users can easily separate code and speech. The speech is located in a YAML file called templates.yaml for all speech interactions and uses SSML to enhance speech interactions. SSML is a markup language that provides a standard way to markup text for generation of synthetic speech (See Reference[18]). It allows more control over how Alexa generates the speech from the text by including effects such as natural breaks, whispering, emphasis on certain words, and explicit pronunciations of certain phonemes. Alexa is able to understand this language and translate it to produce a certain effect. Lastly, Flask-Ask verifies Alexa request signatures.

B. Alexa Skills Kit

In the Alexa Skills Kit (ASK), this is where you generate the custom interaction model, which is a model that describes how interactions between the user and Alexa occur.

First, there is an invocation name, which is Utopia in this case. The invocation name is how you invoke and start the skill by saying a simple command like 'Alexa, start <invocation name>'. In our case, we would say 'Alexa, start Utopia'.

Then, we have the custom intents, another core part in the interaction model. An intent represents an action that fulfills a user's spoken request and can have slots associated with them. We implement each feature as an intent.

Another essential aspect of the interaction model are slot types. There are two types of slots, built in and custom. Built in slot types are always prefixed with AMAZON, and include

AMAZON.US_CITY, AMAZON.DATE, etc. For a full list of the built-in slot types, please see Reference [7]. Custom slot types are created when the built-in AMAZON slots are not sufficient in storing certain crucial arguments. For example, there is no slot type for the suicide trigger words, which justified creating a new custom slot type for these words.

An optional part of the interaction model is the interfaces. In this project, we only use the Audio player interface to stream audio for the poem feature. Other supported interfaces include display for displaying templates on Echo Shows or video apps for streaming videos.

Lastly, to complete the custom interaction model, an endpoint is necessary. Alexa sends JSON requests to this endpoint, and the endpoint should be where your program is. There are many options for picking endpoints. You can use ngrok for secure local tunneling. This option is only good for when you are developing the skill to locally test your program. See Reference [8] for more information. When you are ready to deploy the Alexa skill program to production, you can use a Python library called zappa to seamlessly package up your application and local virtual environment into a Lambda compatible archive, pushing the code into the AWS Lambda function, creating a new API gateway, and finally linking that to the Lambda function. The AWS Lambda function is an event driven, serverless computing platform, and is production ready. See Reference [9] for more information.

V. Testing and Code Coverage

There are many reasons why one should test their code, and it is always a good practice to test code. Some of the reasons include a higher confidence that the main application code is production ready, and regression testing where new features do not break functionality of old features.

For our project, we implemented unit tests to test functionality of each core feature of our Alexa skill program. We used Python's built-in library called unittest to write the unit tests for our main application code, and a Python 3rd party package called pytest as our test runner. We also used a plugin for pytest called pytest-cov to test for code coverage. Code coverage is measured by how much of our test cases cover the main application code. In addition, we used some testing modules in Flask's library to set up testing environments and to generate proper environment dictionaries.

To run tests and check code coverage, run the following command in the root project folder:

```
$ pytest tests/test_utopia_unit.py -v –cov utopia
```

A. *Travis CI*

This project uses Travis-CI, a hosted, distributed continuous integration service that builds and tests software hosted on GitHub. Every commit that is pushed to a GitHub repo automatically triggers Travis CI to run, build and test your software.

The continuous integration configuration is specified in travis.yml file, and specifies the programming language used, desired building and testing environments, and various other parameters. For this project, the configuration is set up to create environment variables, install necessary dependencies, run the test suite, and after a successful test run, report code coverage.

Travis CI generates a badge with the current build status. The Travis CI repository for this project is located in Reference [11].

B. *Codecov*

This project uses Codecov.io to generate code coverage reports. Codecov.io is a free, open source code coverage reporting tool that integrates seamlessly with GitHub and Travis CI. It calculates and measures code coverage and delivers the coverage metrics in a clear, understandable way. This is all configured in the .travis.yml configuration file. Mentioned above, code coverage is measured by how much of the test cases cover the main application code. Codecov shows code coverage metrics as a percentage of main application code covered by test cases.

Similar to Travis CI, Codecov also generates a clickable badge with the current code coverage metrics. The Codecov repository for this project is located in Reference [12].

VI. Usage of Utopia

To start the skill, say a simple invocation phrase, such as the following listed below:

Starting Phrase	Example
<invocation name>	Alexa, Utopia
Ask <invocation name>	Alexa, Ask Utopia
Open <invocation name>	Alexa, Open Utopia
Start <invocation name>	Alexa, Start Utopia

Fig. 2 Usage of Utopia

To start and hear the available features and how to invoke them, you can say the following:

```
Alexa, ask Utopia for available features
```

VII. Conclusion

Depression is a serious and common mental disorder and affects millions of people. Creating a personalized, private platform where individuals going through depression can be more aware of their level of depression and providing ways to help whether it be natural remedies or listening to motivational poems was the purpose of this project. If this skill helps at least one person, then the goal of this project would be meant.

Again, this project was not meant to be a substitute for professional help. However, this Alexa skill could be used as a supplement to professional help. With a certain user's information, such as the Hamilton Depression Rating survey score, a licensed therapist, psychologist, or psychiatrist can then medically and professionally determine the help a user needs. If and when necessary, Utopia could provide this information to a professional to streamline and accelerate help to an individual going through depression.

Acknowledgment

Thanks to Professor Rakesh Ranjan for his insightful, constructive feedback and guidance for this project. His insight and exposure to industry and modern technologies has enabled me to understand software development and requirements in a more professional, practical, and business-oriented way. Also, thanks to Manika Kapoor for her prompt help with defining the requirements for this project and in general for this course.

References

[1] Depression Fact Sheet – World Health Organization (WHO). [Online]. Available: http://www.who.int/en/news-room/fact-sheets/detail/depression

[2] Hamilton Depression Rating Scale Survey [Online]. Available: https://www.psychcongress.com/saundras-corner/scales-screeners/depression/hamilton-depression-rating-scale-ham-d

[3] NLTK Documentation [Online]. Available: http://bridges.brooklyn.cuny.edu/index.php/home

[4] BrainyQuote [Online]. Available: https://www.brainyquote.com/

[5] Google Places API. [Online]. Available: https://cloud.google.com/maps-platform/

[6] National Suicide Prevention Lifeline. [Online]. Available: https://suicidepreventionlifeline.org/

[7] Amazon Alexa Skills Kit [Online]. Available: https://developer.amazon.com/docs/

[8] ngrok [Online]. Available: https://ngrok.com/

[9] zappa [Online]. Available: https://github.com/Miserlou/Zappa

[10] Utopia Alexa Skill GitHub Repo [Online]. Available: https://github.com/SJSU272LabSP18/utopia

[11] Utopia Travis CI [Online]. Available: https://travis-ci.org/k-chuang/utopia-alexa-skill

[12] Utopia Codecov [Online]. Available: https://codecov.io/gh/k-chuang/utopia-alexa-skill

[13] Suicide Trigger Words [Online]. Available: https://github.com/SJSU272LabSP18/utopia/blob/master/speech_assets/custom_slot_types/TriggerWords.txt

[14] Utopia Alexa Skill Architecture [Online]. Available: https://github.com/k-chuang/utopia-alexa-skill/blob/master/images/Utopia-alexa-skill-architecture.png.

[15] WSGI (Web Service Gateway Interface) [Online]. Available: https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface.

[16] Flask Documentation [Online]. Available: http://flask.pocoo.org/docs/1.0/.

[17] Flask-Ask Documentation [Online]. Available: http://flask-ask.readthedocs.io/en/latest/ .

[18] Speech Synthesis Markup Language (SSML) Reference [Online]. Available: https://developer.amazon.com/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html