Vision Connect

Janhavi Dahihande(janhavi.dahihande@sjsu.edu)
Neha Sharma (neha.sharma@sjsu.edu)
Sarang Grover(sarang.grover@sjsu.edu)
Maahi Chatterjee (maahi.chatterjee@sjsu.edu)

San Jose State University
One Washington Square, San Jose, California -95192-0080, USA

Abstract

Vision Connect is a voice-based assistant for visually impaired people and elderly citizens. It is a mobile-based application made for and integrated with Google Home. The application aims to bridge the technological gap among the visually challenged people by providing a voice-based service to integrate with the email sending and receiving, reading verses and overall having a connection with technology which in turn will make their lives easier. Our main aim in developing the application is to provide a hassle-free and convenient service so that technology doesn't become a barrier for anyone and is accessible to anyone in spite of their visual disabilities.

KeyWords:Vision Connect, voice-based, Google Home, voice commands, email sending, email receiving, visually impaired, elderly citizens, vision impairment

1. Introduction

According to a study carried in October, 2018, Globally it is estimated that approximately, 1.3 billion people live with some kind of vision impairment. In USA itself, 25.5 million adults have reported experiencing vision loss. These are the people who have had technology deeply rooted in there lives. It becomes difficult for them to access this technology on getting affected by the impairment. Technology should always be neutral and not differentiate between normal and special people, young and elderly citizens. It should be unbiased in all aspects, in any era. Keeping this thought in mind, we made our application people-unbiased. The integration with Google Home added a touch of comfort and convenience in the lives of the elderly and the visually impaired. It became user-accessible and a known platform for all the people. For understanding the requirements of the specific group of people, we carried on the research and implemented the gained feedback in the application. The undergone research is explained in the next section.

2. Requirement Gathering

As a part of the requirement gathering process, design thinking was one of our major aspects. We visited an NGO called '' based in California which . Our experience their made us realize the vast scope of our application and drove us forward in giving us a sense of responsibility in contributing to the society through the perspective of software engineers.

In the NGO meeting, we met a lot of elderly people who have problems with their vision (e.g vision loss, weak eyesight, straining eyes, etc). We also had a discussion with people with vision impairment ranging from the age group of 18-80 years. We noted down a lot of this valuable feedback for the future prospects of our project.

In addition with meeting these people, we had an interview with the CEO of the NGO, Mr. John Christian. We put forward our idea in front of him and demonstrated the Beta version of our application. From him we understood the various problems faced by him and his officials in communicating with the special people of the community. Also, we got a positive feedback from him and an offer to actually implement the application for the NGO. Overall, this design thinking phase helped us a lot in shaping our application as a whole.

3. Architecture

A. Client side of Vision Connect

In our application, the client is the Google Assistant present in the Google Home. The

choice of the client was made taking into consideration the numerous options available for people to interact with the technology. The client uses Natural Language Processing to gauge the words from the user query. Google Assistant can be instantiated from any Android device, any iPhone device and of course from the Google Home.

DialogFlow is the backbone behind the client rendering. It build engaging voice and text-based conversational interfaces with users on your website, mobile app, the Google Assistant, Amazon Alexa, Facebook Messenger, and other popular platforms and devices. It is powered by Google's machine learning, is built on Google infrastructure and is optimized for Google Assistant. We developed various training phrases and trained the DialogFlow agent to accept the various permutations and combinations of user queries through their voice. In turn, the assistant was trained to provide optimal and accurate responses on the user interface.

The screenshots show the queries from the users fired using voice commands and the response given by the agent integrated in the Google Assistant. (Note: Google Home behaves the same. Unable to show the UI as Home operates only on voice commands and has no UI). The requests are sent by integrating the functionality of 'Fulfillment' where the server processes the request given at an HTTPS url.

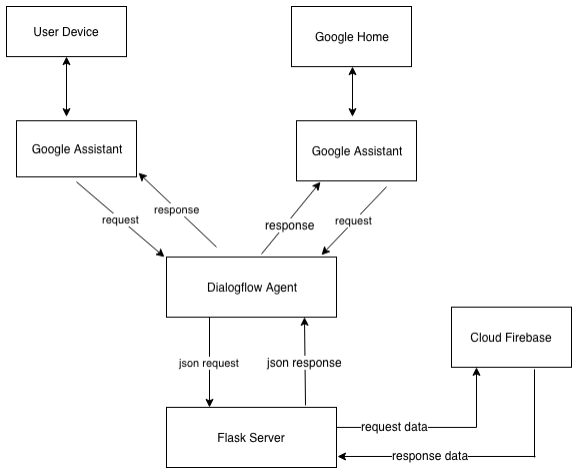


Figure -Architecture Vision Connect

B. Server side of Vision Connect

Flask server was used to develop the backend of the application. Here the user input from the DialogFlow agent is received. The data is extracted and analyzed using Python. The main libraries used in the project consist of Flask, firebase_admin, requests, flask_mail, jsonify, Mail, Message, etc.

The server hosts many functionalities related to the application. For example, email receiving, sending email and reading verses. Depending on the client intent, (i.e. DialogFlow agent Intent), the data differentiation and processing is done. The response is sent back to the Agent in the form of a Fulfillment text which is then displayed on the client. The fulfillment text can have various forms such as normal text, lists, cards, etc. We have majorly used text responses for the client side UI.

The major feature of email sending is archived by importing the Python library of SendGrid. An API key is generated for the library by creating an account for the same. The number of requests and responses are logged in on the SendGrid portal and the account holder can gain various insights from these numbers.

C. Cloud Filestore

Cloud Firestore is a flexible, scalable NoSQL cloud database to store and sync data for client- and server-side development. As of now it is still in its Beta release and has its own set of limitations and quotas. But it was the perfect candidate for our application and we decided to go ahead with it. It showed promising results in the whole development process and the deployment phase as well.

A Python file was created on the Flask server for setting the various certificates and the initialization process of the application. A reference to this file store was then sent to the actual server implementation functions which in turn processed the data(added, updated and deleted) using the provided functions of push(), update(), set(), delete(), etc.

The main aim of going with the Cloud Firestore is its efficiency, ease of integration with various platforms and its strong set of functionalities. It must be brought to attention that even it being in Beta version, Cloud Firestore has an almost stable functioning and thus is a perfect candidate for future enhancements if to be made.

4. Services offered by the Application

Once the proper setup of the server and the client is done, we can get to experiment with the various services offered by the application. The various services are as follows:

A. Email Sending functionality

This functionality lets the user send emails to a receiver using voice based commands from the Google Home. Once the user says “Talk to my Email App”, our DialogFlow agent gets invoked. The user can initiate the email sending process by giving a voice command saying “Send an email”. The email id and the message to be sent are the entities required for this intent. These entities are satisfied by the agent by gaining the data from the user as seen in the screenshot.

Once the entities are achieved, the call goes to the server and calls theSendGrid API client with the provided API key. Also, the data gets stored in the Cloud Firestore in the form go key-value pairs. Once this data gets saved in the cloud, it can be retrieved any time in the future.

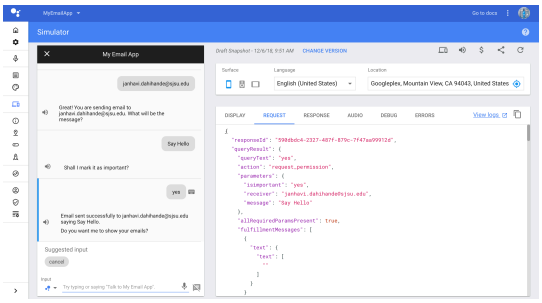


Fig. 1 Email sending functionality shown in Actions on Google Simulator

B. Email Receiving functionality

This functionality lets the user see its received emails from other users using the same agent. The system identifies the user logged in user in the Google Assistant. For this, the user has to give “Show my Emails” voice command to the Google Home. Once this command is given, DialogFlow maps the command with one of the training phrases and provided the name of the intent. The intent then calls our backend with the logged in user id.

On the server-side, the Cloud Firestore data is queried wherein the receiver email id of all the emails is the logged in user id. These data entries are retrieved one by one and sent to the client in the form of fulfillment text. The DialogFlow agent identifies the text and sends the response to the Google Assistant of the Google Home. All the emails are read out to the user by the Google Assistant.

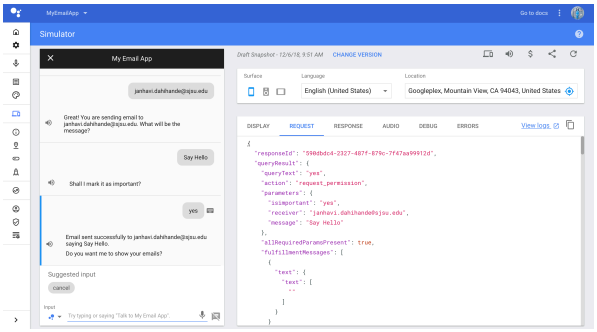


Fig. 2 Email receiving functionality shown in Actions on Google Simulator

C. Mark emails as important

Whenever the user intimates the process of sending an email, he/she is queried whether to mark the email as important(The ‘starred’ functionality in the Gmail application). If the user replies as ‘Yes’, the database entry is changed to store a boolean value of ‘is_important’ as True. Once the entry is set as true, the system can differentiate between the important and not important emails.

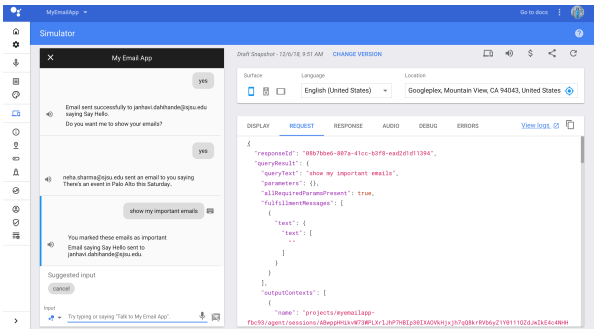


Fig. 3 Marking emails as important functionality shown in Actions on Google Simulator

D. Read Verses from Bible

This functionality lets the user command the

system to show random verses from the Bible. Here, the user says ‘Read a verse from the Bible’. This input is matched to one of the training parades and the intent is identified. This intent then calls the server and once there it queires the database for any random verse. The selected verse is then read out aloud for the user through the Google Home. The user can request for another one, and it responds with another verse for the user through voice on Google Home.

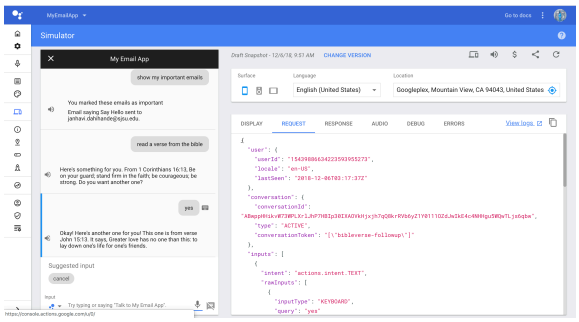


Fig. 4 Reading verses from Bible functionality shown in Actions on Google Simulator

4. Process flow

The basic process flow of the application is shown below

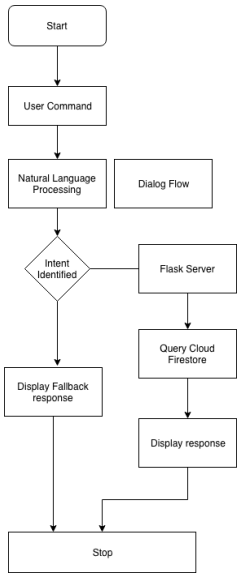
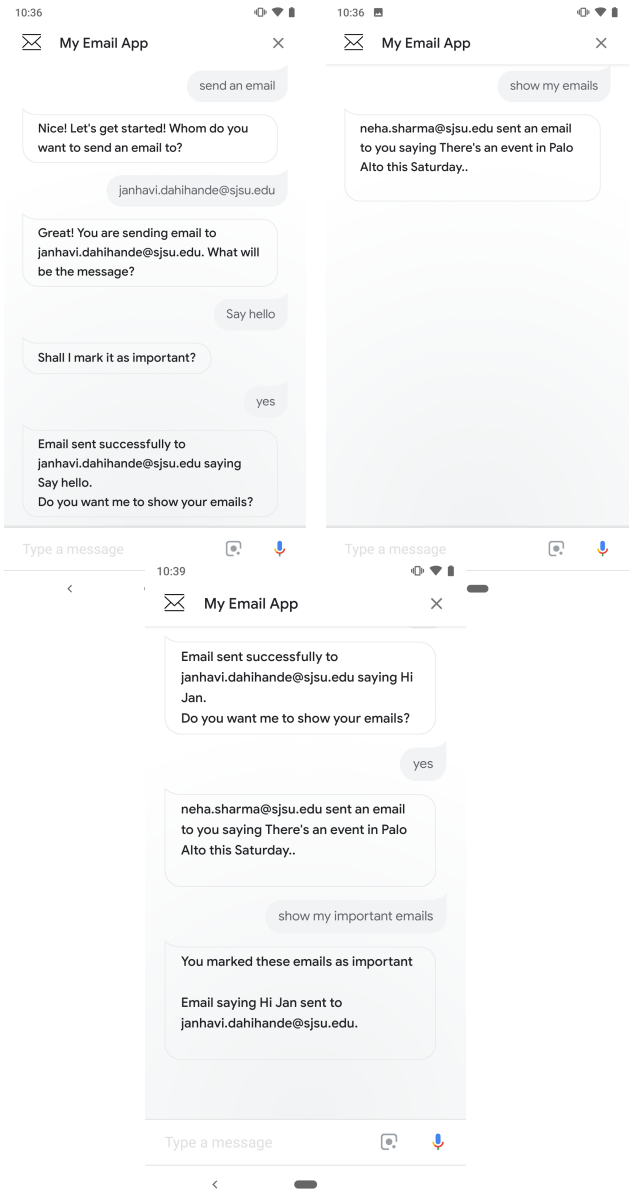


Fig. 5
Complete Process flow of the system

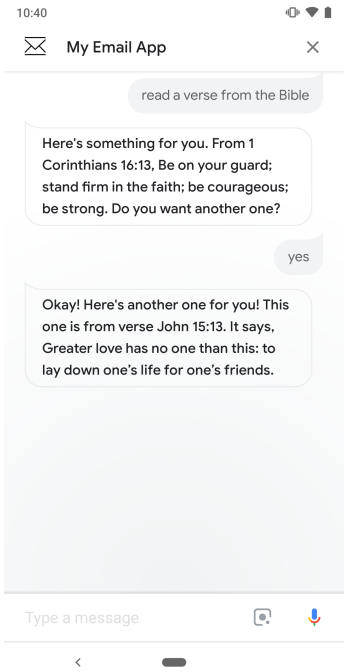
1. The user sends a command through voice to the Google Assistant which in turn forwards it to the DialogFlow agent..
2. DialogFlow uses Natural Language Processing to identify the intent of the user command
3. Once the intent is matched, it forwards the command to Flask server in the form of json request.
4. The server queries the Cloud firestore database and displays the response on the Google Assistant

5. Application User Interface on Android App

1. User command and system response for a) sending emails given user input as sender email and email message and receiving emails



2. User command and system response for reading verses

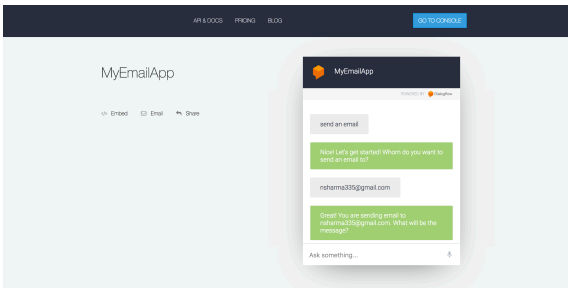


6. Varied Integrations

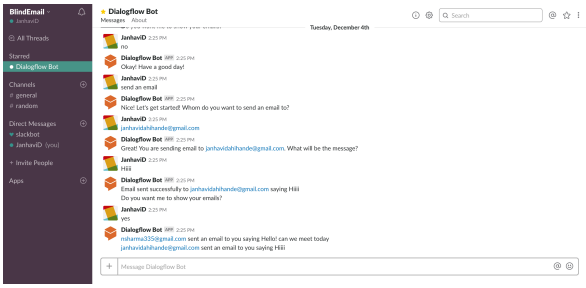
The Vision Connect app can be integrated with varied platforms. As of now, we have integrated it with web and Slack. The web ui can be embedded into any website using just a few lines of code. The UI interfaces are shown in the screenshots.

Furthermore, the integration can take place with any other system. For example, in the future, if someone invented a Braille device system, our app can be easily integrated in it and the user will be able to access the same functionalities.

A. ***Web integration***



B. ***Slack integration***



7. Future Enhancements

We are looking forward to develop Vision connect into a complete virtual personal assistant for visually impaired people. We are looking to implement all the use cases like uploading books, documents ,event information and take the application to production stage and deliver the product to the organization.

8. Conclusion

We analyzed the problems faced by visually impaired people and provided a solution to enable people who once had complete access to technology to continue using and also encourage people who don't use technology and smartphones to use the platform.

9. Project Repository

GitHub URL:
https://github.com/SJSU272LabF18/Project-Team-28

10. References

Dialog Flow: https://dialogflow.com/

https://cloud.google.com/dialogflow-enterprise/

Flask:
http://flask.pocoo.org/docs/1.0/