SIGNBOT

GROUP 17

Team Members:
Aniket Umesh Kathare (20114010)
Arnav Vyas (20114015)
Deepesh Garg (20114032)
Shaurya Rana (20114086)
Shaurya Semwal (20119048)

Project Proposal

SignBot is a web-based system specially designed for hearing impaired and speech impaired people. This system takes live audio speech recording or text as input and displays the relevant Indian Sign Language (ISL) animations as output. Our app focuses on making life easier for disabled people so that they can also get involved in all the conversations and bring normalcy to the life of all the people. It will help people impaired of hearing or impaired of speaking by converting speech/text to sign language animations through an animated character.

Technology Used

- HTML, CSS and JavaScript – To create frontend for the website.
- Django – Python web-framework used for deployment of website
- SQLite – Database to store login information
- JavaScript Web Speech API – To record audio and fetch words
- Natural Language Toolkit – To process the text detected by Web speech API
- Blender – To create animations.

KEY FEATURES

The goal of the proposed project is to create a web based interactive platform for people to communicate with deaf people via animations. On running the app, the user is greeted by the home page from where he/she needs to login or signup to use all functionalities of the app. The convertor page is dedicated for the task of taking inputs from the user and displaying the animations. The user can either type the sentence he/she wants to convey or speak it into the microphone. The app figures out the words on its own and displays sign language actions for the respective words through an animated character.

1. Audio to sign conversion in very less time.
2. Easy to understand animations.
3. Users can communicate smoothly through the animations on the website.
4. The app is able to show animations for all kinds of input

DESCRIPTION OF IMPLEMENTATION

1. **Development Process -** The development process is done in accordance to the iterative waterfall model. This model is an enhanced version of the classic waterfall model which incorporates a major feature of providing feedback paths to its preceding phases. This implies that the development process can be regressed back to any of the preceding development phases if necessary. This model works well for projects where the requirements of the project aren't dynamically changing and are clearly stated before starting the project. Also, this model does not involve multiple final deliverables, thus projects following the iterative waterfall model have to deliver a single final product. Since our project is relatively small, well-defined, clearly understood and has fairly stable requirements; substantial changes to the requirements are unlikely. Due to the small size of the project, the overhead required to develop multiple revisions would not be paid off, so it will be best to deliver a final finished product. Thus, we have worked out the iterative waterfall model to be highly suitable for our project and have applied the same model for the entire development process.

2. Outline Plan-

- Milestone 1 (22nd February – 1st March) - Learning to integrate different tools for our project
- Milestone 2 (8th March – 15th March) - Feasibility Study Report
- Milestone 3 (16th March – 22nd March) - Requirements Analysis Report
- Milestone 4 (22nd March – 29th March) – Design Document
- Milestone 5 (30th March - 10th April) - Development/coding Initial 3-4 days were spent in making the front-end of the web app by the team and the server was set up by the backend team. In the next 4-5 days the JavaScript Web API was set up to record the audio and NLTK was used to process the words detected in audio. The team was simultaneously involved in making the blender animations throughout the coding phase. The remaining time was used to synchronize the animations with the text detected.
- Milestone 6 (12th April - 17th April) - Deployment and Testing of the project.

3. Coding and Development Phase-

Django framework is used to develop the website. The pre-defined templates are modified according to the need of web-app. The database management is also done using Django. The web-app development is divided into three verticals namely: front-end, back-end animations.

- **Back-end:** The main aim is to process the audio recorded and generate an animation on the screen. In order to achieve this we have to do the following tasks: record the audio input , detect sentence in the audio recorded, process the sentence in such a way to only keep the important keywords using which the meaning of the audio can be conveyed and render the animations matching the keywords on the screen. In order to achieve this functionality we would require the following libraries: nltk.tokenize, nltk.corpus, nltk.stem , nltk along with the standard Django libraries. JavaScript Web API is used to record the audio from the user and send to the server for further processing.

Processing the words:

The sentence which is detected is converted to lower case for further processing. We tokenize the sentence to detect all the words. Next using nltk.pos_tag() we generate a pair of the words and corresponding part of speech. The next step follows to detect the tense of the sentence identified. We loop through the words of the sentence and according to the tagger of the word, we store them in future, present, past or present_continuous. We have defined some words which we want to ignore in the sentence. These words are stored in stop_words. The reason behind doing this is to convey the correct meaning using minimum words. We ignore words like articles, verbs like is, are etc. The next step is to process the words using WordNetLemmatizer(). Before using this function we run a loop where we check the tagger in case it is a verb or adjective we call the WordNetLemmatizer() function and send the word and respective part of speech (verb or adjective) as the arguments. In the remaining cases we only send the word as an argument. In order to convey the appropriate meaning of the sentence we check for the number of entries in a particular tense. If the tense is past we add “Before” , in case of future we add “Will” and in case of present_continuous we add “Now”. The only purpose behind this is to accurately convey the meaning of the recorded audio. After successfully detecting and modifying the sentence we check if there is an animation corresponding to our sentence. If we find the animation corresponding to the word, we store it in filtered text array else we break the word in alphabets and store it the array. Finally we pass this filtered text to the animation.html page to render the animation on screen.

- **Front-end:** Minimalistic brown-themed design is used for the frontend which looks more elegant. Design is kept in such a way that new users will have no difficulty in using the website in one go without any guiding. Landing page shows sign language animation of the word welcome. Google Fonts Poppins has bren imported and used in all the pages. The entire process of conversion of users input audio/text to the sign language animations gets displayed. Which word’s animation is being shown gets highlighted. All these frontend features add to the user experience.

- **Animations:** The animations are made using Blender 3D tool which is free and open-source 3D computer graphic software. A customized character has been used and rigged which is a technique in computer animation in which a character is represented in two parts: a surface representation used to draw the character (called the mesh or skin) and a hierarchical set of interconnected parts (called bones/skeleton/rig), a virtual armature used to animate (pose and keyframe) the mesh. Keyframing technique is used for animation, based on the notion, the object has a beginning state or condition and will be changing over time, in position, form, colour, orientation, or any other property, to some different final form. We only need to show the initial and final "key" frames, or conditions, that describe the movement of the character, all other intermediate positions are figured out automatically. Sign language videos are referred for posing the character in 3D space accurately with the help of different coordinates. Finally, the animation is rendered in .mp4 format. Our application contains around 150 animations which include sign language of alphabets, numbers, nouns, verbs, and some other words. More animations will be added time to time.

4. Integrating server, frontend, database, API and libraries-

- Our frontend is made entirely using HTML, CSS and JavaScript.
- The web-server is accessed through Django.
- The database used is sqlite.
- We have used the JavaScript Web-Speech API which is used to record audio and fetch words.
- From NLTK multiple libraries have been imported for tokenizing, removing stopwords and to apply lemmatizing process to the words.

The server processes all the requests made by the frontend. Django helps in integrating the whole web-app from providing database file to providing templates for all the files(views, url etc). JavaScript web speech API is used to record the audio from the user and identify the words in audio. Using NLTK, pre-processing of the words is done where identifying the important keywords is done. Further the animations are rendered on the screen with the help of Django.

5. Password Authentication-

The Django authentication system is used for both authentication and authorization. In simple words authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Django uses a flexible password storage system and uses PBKDF2 by default. PKDF2(Password based key derivation function 2) is a key function with a sliding computational cost, used to reduce vulnerabilities of brute-force attacks.

APPLICATION OF THE PROJECT

According to a World Health Organization report, around 63 million people in India suffer from either complete or partial deafness, and of these, at least 50 lakhs are children which hold for 20% of the hearing and speech impaired world population. This population lacks the amenities which a normal person should own. The big reason behind this is the lack of communication

The goal of the web app is to bridge the communication gap between disabled and able-bodied people. The user can interact with the website to view signs for the audio input. Our app focuses on making life easier for disabled people so that they can also get involved in all the conversations and bring normalcy to the life of all the people. It can be used in live seminars, news reporting, in the education sector to communicate efficiently and lift the differential barriers.

FUTURE SCOPE

The first release of our application was intended towards building a simple application with efficient login system which displays Indian Sign language animations corresponding to the input. The necessary features mentioned above were implemented successfully in this web-app. In further releases, to improve our application we would like to add these new features:

- Add newer animations to possibly cover all distinct words in the English dictionary and associate the synonyms of such words to them.
- Add newer animation sets for other sign languages other than the Indian Sign Language.
- As the number of users tend to increase we would like to deploy our application on Digital Ocean rather than running it on a remote server.

THANK YOU!