Bachelor Of Technology (Computer science)



**INT-213**

Project on

Voice enable chat bot that help in

basic git commands

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll no.** | **Reg. no.** |
| Satyam Kumar | 68 | 11910483 |
| Nilanjan Majumdar | 04 | 11910126 |
| Gourav | 08 | 11910098 |

Submitted To: Mr. Sagar Pande

Date of Submission: 3rd November, 2020

**Summary**

Voice recognition Gitbot is a voice recognition tool used for basic git commands. This Gitbot can be used online by google speech recognition and also offline by pocketsphinx. We used Pyttx3 for text to speech. We have used Pyaudio as cross-platform of audio I/O library. Tkinter is also used for graphical user interface.

We used Pyttx3 for text to speech as it works offline unlike google text to speech which works online. There are two voices male and female in Pyttx3 which are in already trained to this library so we used these voices instead of training it with any different voice.

In training of the voice model, we have used Sphinxbase and Sphinxtrain. We have three files in voice model Acoustic model, Phonetic dictionary and Language model. An acoustic model is employed in automatic speech recognition to signify the connection between an audio signal and therefore the phonemes or other linguistic units that structure speech. A phonetic dictionary provides the system with a mapping of vocabulary words to sequences of phonemes. The language model provides context to distinguish between words and phrases that sound similar. In traning of default model of pocketsphinx we adapt the acoustic model to different voices so it can understand voices of different people.

We had used Tkinter because it is standard GUI library for python which provide a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

**Acknowledgement**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. Sagar Pande for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & member of this project for their kind co-operation and encouragement which help us in completion of this project.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

**List of Content**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page No.** |
| 1 | Summary | 2 |
| 2 | Acknowledgement | 3 |
| 3 | List of Contents | 4 |
| 4 | List of Figures | 5 |
| 5 | Chapter 1  Introducing Voice Recognition Gitbot | 6-7 |
| 6 | Chapter 2  How Gitbot is designed | 8-15 |
| 7 | Chapter 3  Features and Dependencies | 16-17 |
| 8 | Chapter 4  Conclusion | 18 |
| 9 | References | 19 |

**List of Figures**

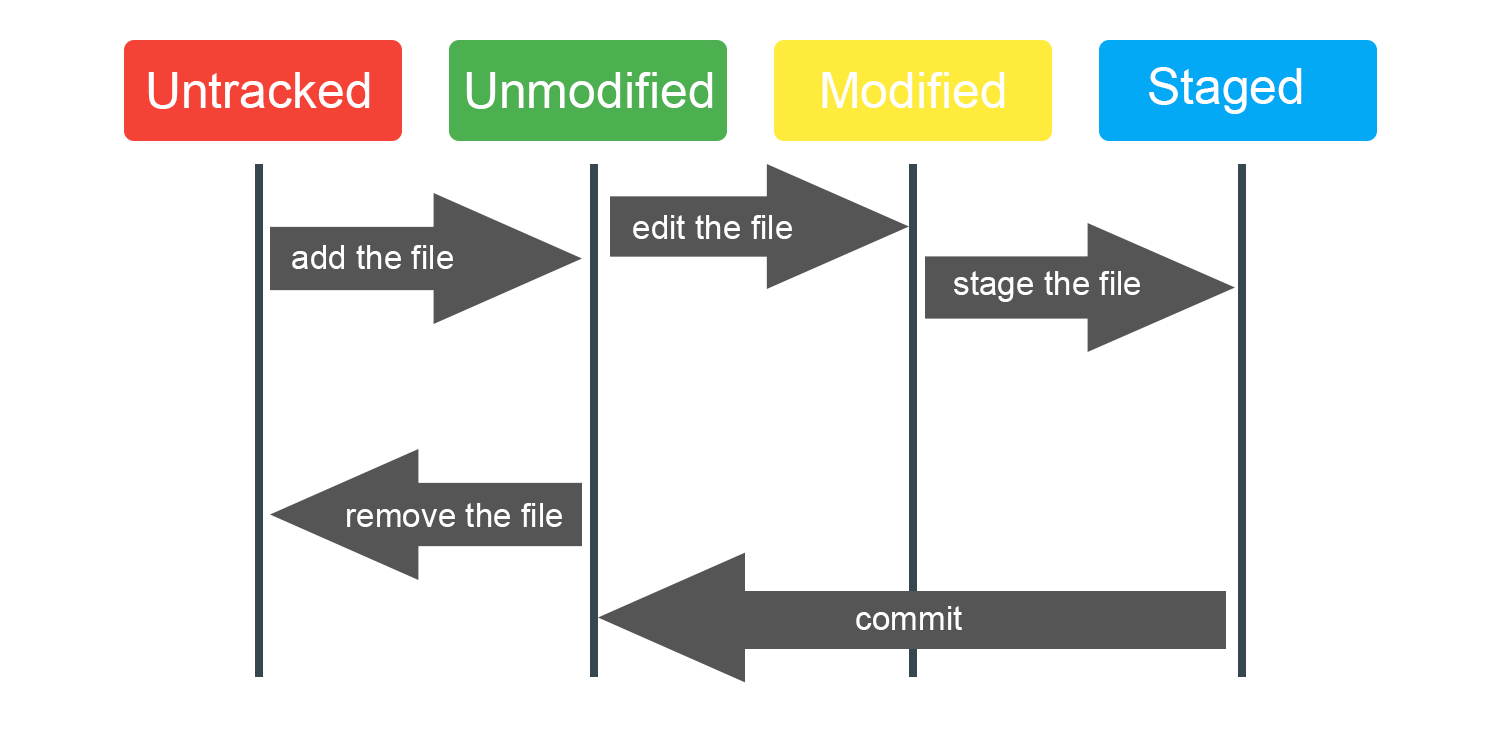
|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Figures** | **Page No.** |
| 1 | Fig 1.1: Introducing Git | 6 |
| 2 | Fig 1.2: Gitbot | 7 |
| 3 | Fig 2.1: Frontend | 9 |
| 4 | Fig 2.2: Enter Repo link | 10 |
| 5 | Fig 2.3: Error for empty Repo link | 10 |
| 6 | Fig 2.4: After setting Repo link | 11 |
| 7 | Fig 2.5: Working of Gitbot | 13 |

**Chapter 1**

**Introducing Voice Recognition Gitbot**

**1.1 What is Git**

Git is an open-source version control system that was started by Linus Torvalds—the same person who created Linux. It is a distributed version-control system for tracking changes in source code during software development. It is considered for managing work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.



**Fig 1.1: Introducing Git**

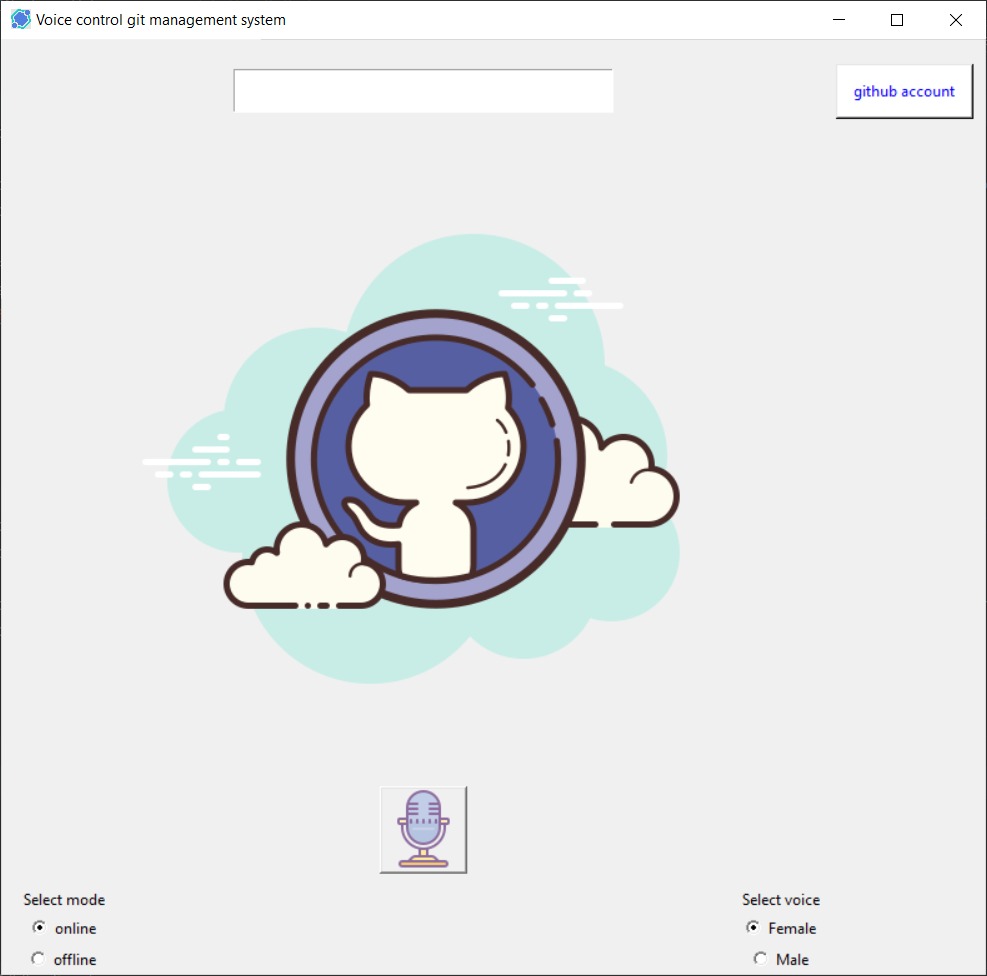
**1.2 Why is Git Important**

If Someone is learning to code, chances are his most important goal is to eventually get a job as a software developer. Learning Git and GitHub is incredibly important because 99% of the companies that can hire you will use Git and GitHub. Therefore, learning how to work with Git and GitHub make person more hireable and help you differentiate yourself from more junior developers.

So, now we know that Git is a very crucial tool but still many people don’t know how to use it. So, that’s why we have made a Voice Recognition Gitbot which can help people to use Basic Git commands with their voice.

**1.3 What is Voice Recognition Gitbot**

Voice Recognition Gitbot is a voice recognition tool used for basic git commands. This Gitbot helps the people who have never used or don’t know how to use Git. This is a voice assistant with which we can give voice commands and use Git. With this the people who dont know how to use git can have an easy time around using git and have an graphical based interaction with git.We are using tkinter to make the frontend of the gitbot which will connect with both offline and online mode and carry out the operation. At the backend for the online mode we are using Gtts( google text to speech ) to understand what the user is saying and carry out the operation and for the offline part we are using pocketsphinx Indian english adapted modal to understand and carry out the operation.



**Fig 1.2: Gitbot**

**Chapter 2**

**How Gitbot is designed**

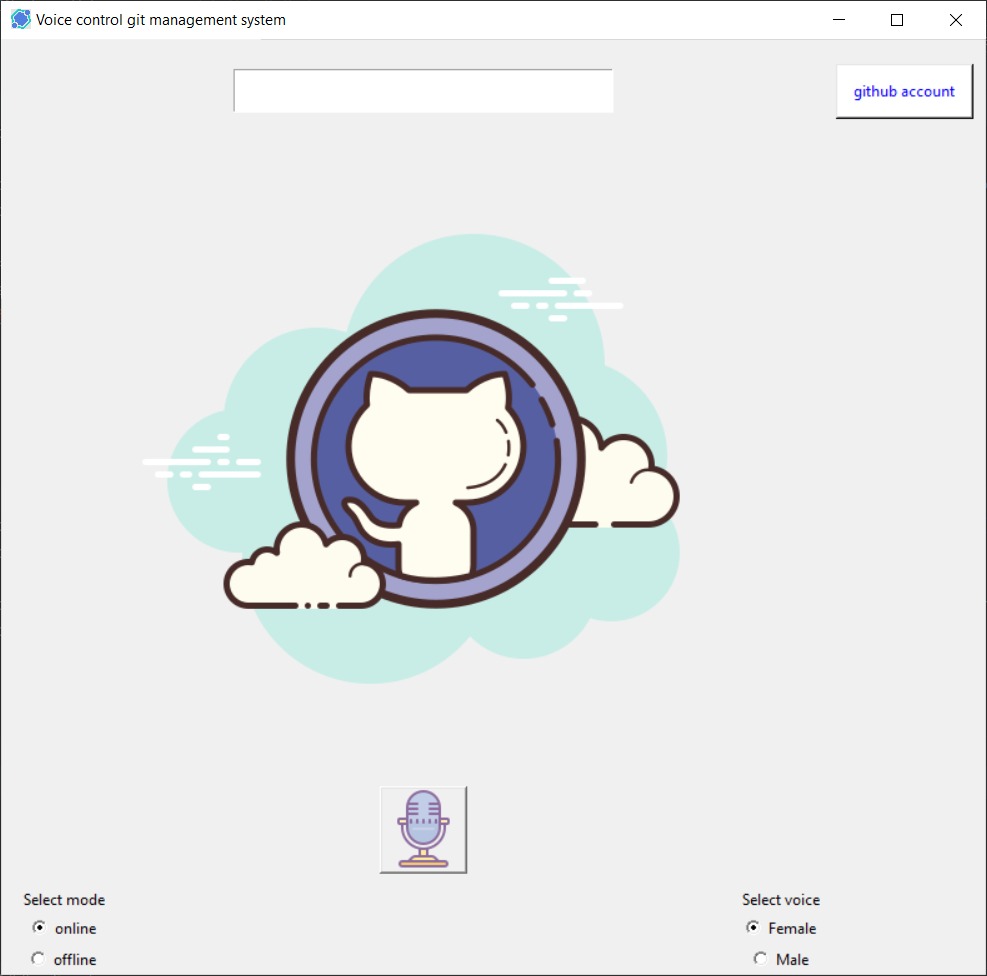
**2.1 Frontend**

We have used Tkinter for graphical user interface. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

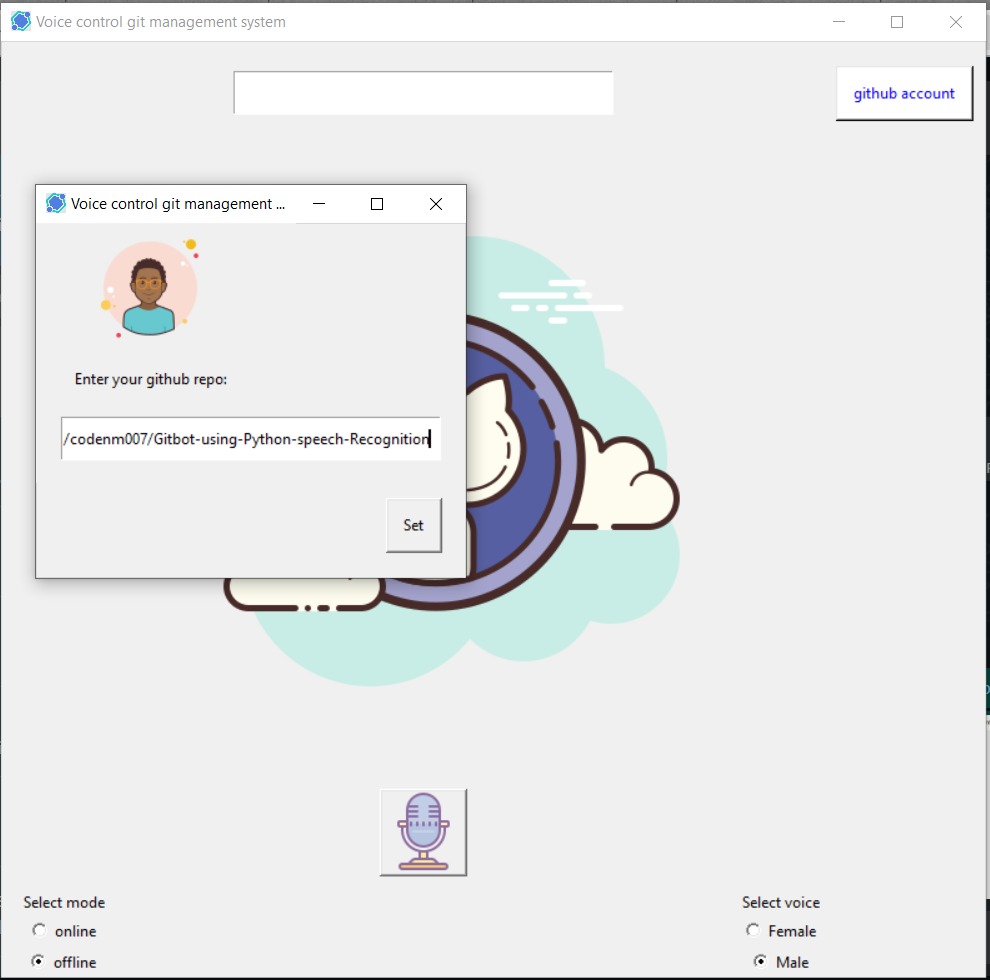
**2.1.1 Widgets used in our project**

* Label
* Entry
* Button
* Radiobutton
* PhotoImage (from importing pillow)



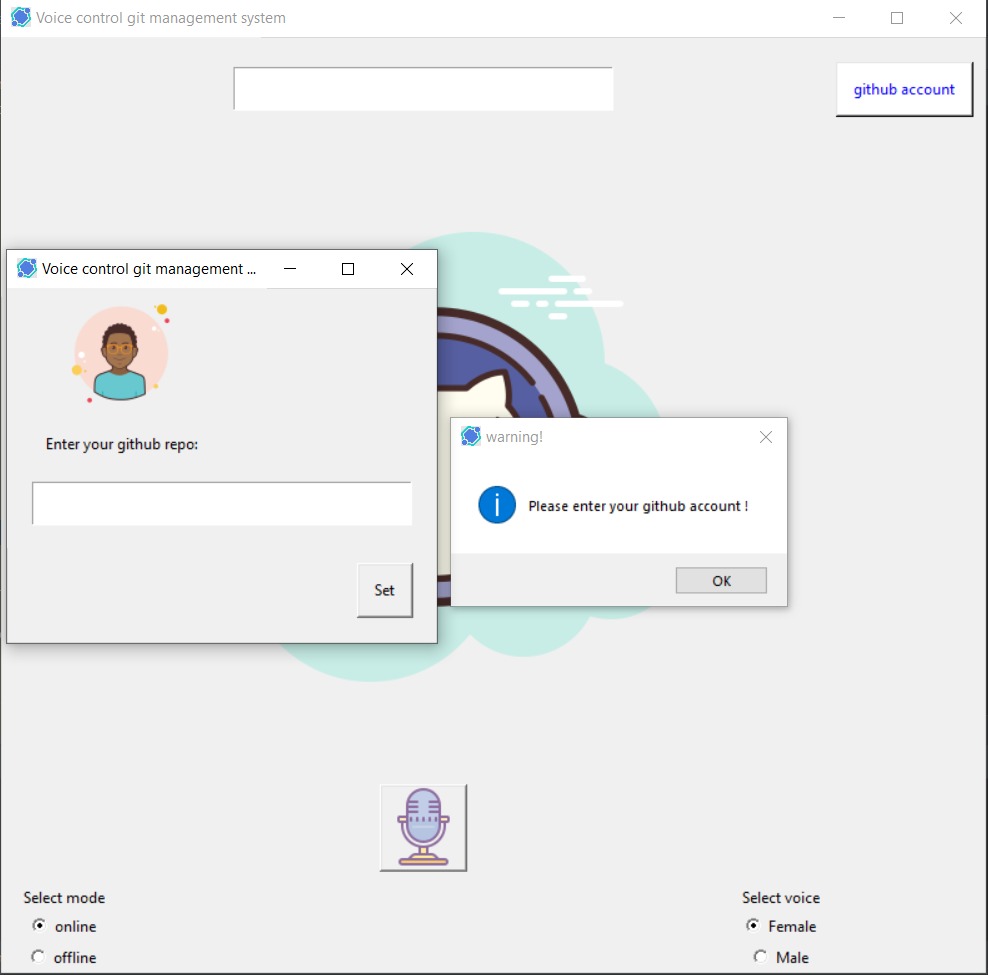
**Fig 2.1: Frontend**

Through the Ui you can connect with our backend. On the top side right corner there is a button named “github account”. This button is created using Button widget by clicking a window will open which will give user the option to enter the github link where the user can enter the repo link.

****

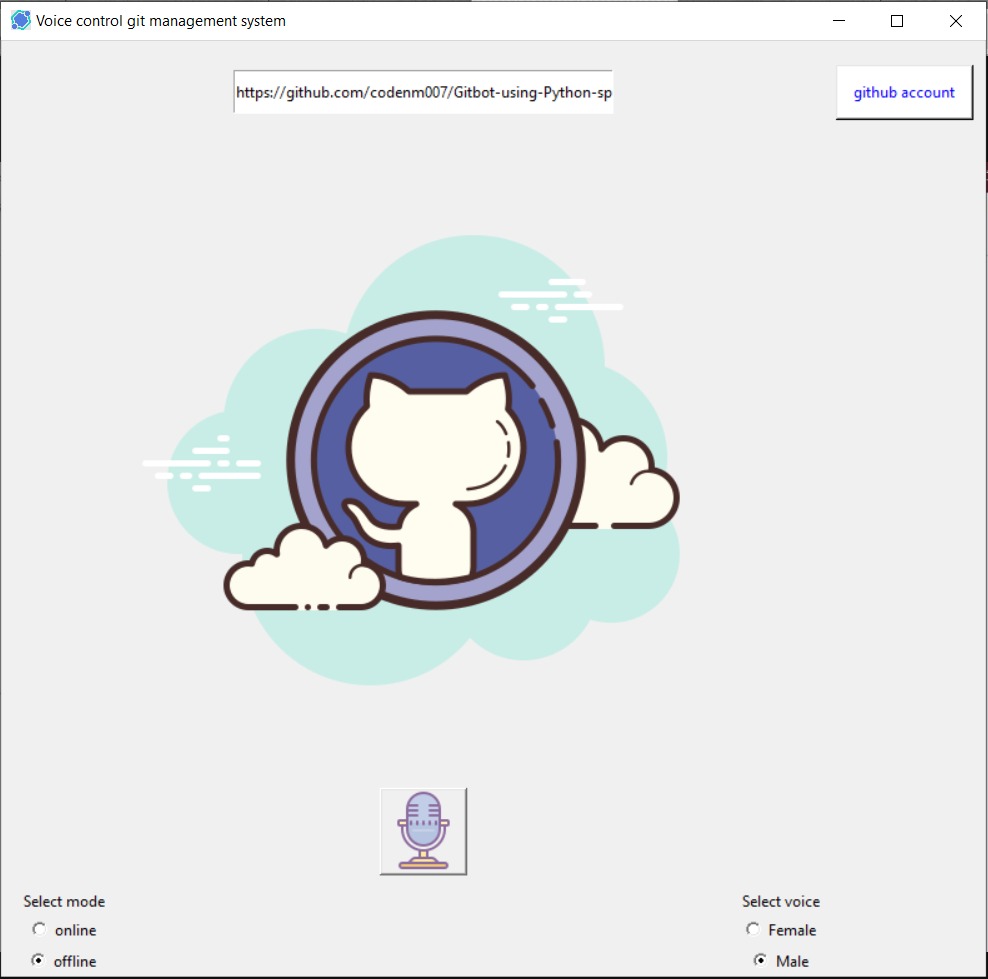
**Fig 2.2: Enter Repo link**

Then the user can click on the set button to set the repo link. If the user is not entering anything and clicking on the set button without entering the link then a message window will be displayed notifying the user to enter the repo link.

****

**Fig 2.3: Error for empty Repo link**

Then the user can select the mode and voice type and can click on the microphone button to carry out the task that the user wants to complete.



**Fig 2.4: After setting Repo link**

**2.2 Backend (Recognition Engine)**

Engine is a core essential function for other program. It coordinates the overall operation of the programs.

**2.2.1 Working**

* User first interacts with the Tkinter frontend and Tkinter contacts then engine to execute any command.
* In the recognize, first we select the mode of the voice recognition which is Online (1) or Offline (0).
* Now we select the voice of the Gitbot which is Female (1) or Male (0).

def recognize(selection,number):  
  
 engine.setProperty('voice', voices[number].id) # changing index, changes voices. 1 for female and 0 for male  
 if selection == 'google':  
 with sr.Microphone() as source:  
 print("Say something!")  
 audio = recognition.listen(source,timeout=2)  
 try:  
 voice\_data = recognition.recognize\_google(audio)  
 print(voice\_data.lower())  
 return exec\_command(voice\_data.lower())  
 except sr.UnknownValueError:  
 engine.say('Sorry, I did not get that')  
 except sr.RequestError:  
 engine.say('Sorry, googles service is down')  
  
  
 elif selection == 'offline':  
 with sr.Microphone() as source:  
 print("Say something!")  
 audio = recognition.listen(source, timeout=3)  
 voice\_data = recognition.recognize\_sphinx(audio)  
 print(voice\_data)  
 exec\_command(voice\_data)  
 else:  
 print('Invalid selection!')

* Now after this user speak something through mic and then the engine will fetch data from comprehensions.json file to cheak if there exist any command spoken by the user. And if there exist the command then it will check for its appropriate execution in readout.json and give the output correspondent to the command.

def exec\_command(voice\_data):  
 if there\_exists(comprehensions\_data["what is your name"],voice\_data):  
 engine.say(read\_out\_data["name"])  
 engine.runAndWait()  
 return read\_out\_data["name"]  
 if there\_exists(comprehensions\_data["time"],voice\_data):  
 engine.say("The time is:")  
 engine.say(datetime.now().strftime("%I:%M %p"))  
 engine.runAndWait()  
 return datetime.now().strftime("%I:%M %p")

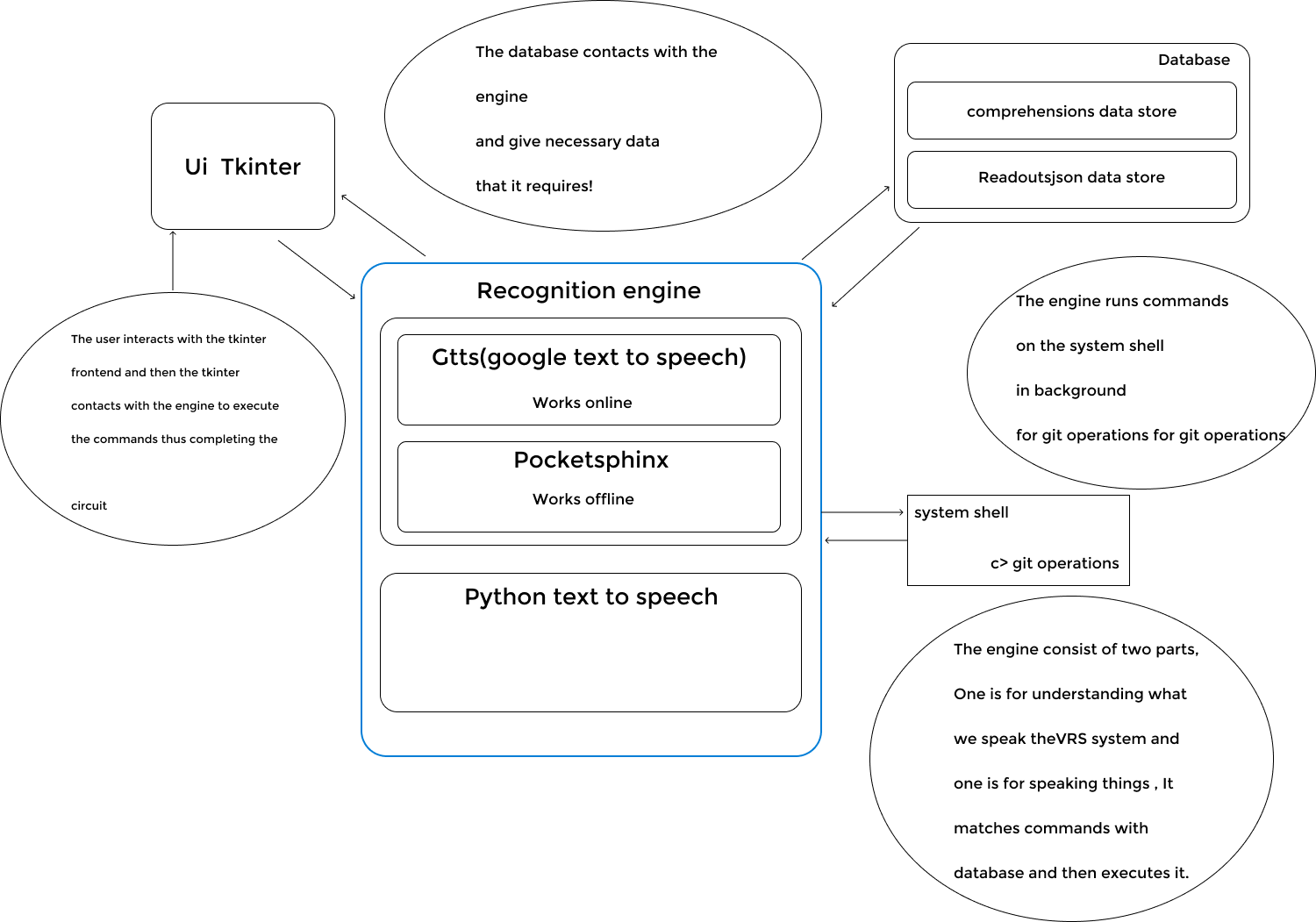
**comprehension.json:**

{ "what is your name" :["name","your name","tell me your name","hi","hello"],  
 "time" : ["time","whats the time","tell me the time"]}

**readout.json:**

{"name": "Hi, my name is Viki ! How can I help you?"}

* And for the Git commands the engine will run commands on the system shell in background for git operations.

****

**Fig 2.5: Working of Gitbot**

**2.3 Online speech recognition**

For best accuracy we have used Google Speech Recognition which works online.

**2.3.1 Requirements:**

* Python 2.6, 2.7, 3.3+
* Pyaudio 0.2.11+

**2.4 Offline speech recognition**

For offline speech recognition we have used Pocketsphinx.

Pocketsphinx is a part of the CMU Sphinx Open Source Toolkit for Speech Recognition. This package provides a python interface to CMU Sphinxbase and Pocketsphinx libraries created with SWIG and Setuptools.

In Pocketsphinx we have a model of three files acoustic model, phonetic dictionary and language model. From all the three files the acoustic model is the one which need training to understand different voices. So, we trained the acoustic model to understand the voices of different people.

So, for training we need to have:

* a list of sentences (fileids file)
* a dictionary describing the pronunciation of all the words in that list of sentences (transcription file)
* a recording of you speaking each of those sentences (WAV audio recordings)

Before training the model, we needed Sphinxbase and Sphinxtrain. After that follow the following:

* First of all the mdef file which is in binary form in the acoustic model has to changed to plain text used by Sphinxtrain tools. This can be done by pocketsphinx\_mdef\_convert tool present in Pocketsphinx. For this run command:

./pocketsphinx\_mdef\_convert -text en-us/mdef en-us/mdef.txt

* In order to run the adaptation tools, you must generate a set of acoustic model feature files from these WAV audio recordings. This can be done with the sphinx\_fe tool from Sphinxbase. For this run command:

./sphinx\_fe -argfile en-us/feat.params -samprate 16000 -c arctic20.fileids -di . -do . -ei wav -eo mfc -mswav yes

* Now, the next step is to collect statistics from from the adaptation data. Thi can be done by a tool named bw which is present in sphinxtrain. For this run command:

./bw -hmmdir en-us -moddeffn en-us/mdef.txt -ts2cbfn .cont. -feat 1s\_c\_d\_dd -lda en-us/feature-transform -cmn current -agc none -dictfn cmudict-en-us.dict -ctlfn arctic20.fileids -lsnfn arctic20.transcription -accumdir .

* Now time to adapt the model, for this will use first mllr\_solve tool and then map\_adapt tool which are present in Sphinxtrain. For this run command:

./mllr\_solve -meanfn en-us/means -varfn en-us/variances -outmllrfn mllr\_matrix -accumdir .

* For saving space we create a sendump file by mk\_s2sendump tool present in Sphinxtrain. For this run command:

./mk\_s2sendump -pocketsphinx yes -moddeffn en-us/mdef.txt -mixwfn en-us/mixture\_weights -sendumpfn en-us/sendump

Now we can delete mixture\_weights file and mdef.text file as these are not used by the decoder and we can save some space.

**Chapter 3**

**Features and Dependencies**

**3.1 Features of Gitbot**

* Online recognition
* Offine recognition (using pocketsphinx)
* Basic git commands
* Voice interaction
* Friendly UI

**3.2 Supported Git Commands**

|  |  |
| --- | --- |
| Git commands | Voice commands |
| git status | what's the status |
| git add . | backup my code |
| git commit -m "commited by viki the gitbot" | commit my code |
| git push origin master | upload to cloud |
| git pull origin master | update |

**3.3 Fun with gitbot**

* what is your name, tell me your name, hi, hello
* what's the time, time
* tell me about yourself

**3.4 Dependencies used in project (Must install):**

* Pyaudio
* Pocketsphinx
* Speechrecognition
* Tkinter
* Pillow

**Chapter 4**

**Conclusion**

In conclusion, Git provides a way of keeping track of past versions of software and papers, making collaboration between various authors easy, and provides backup for your software. It has proven very useful to the open-source community and in academia as well.

So, our Voice Recognition Gitbot provides an easy graphical interface to the user to use the git and git hub efficiently. It will help them to use and understand the git and its importance.

**References**

* <https://pypi.org/project/SpeechRecognition/>
* <https://pypi.org/project/pocketsphinx/>
* <https://github.com/cmusphinx/pocketsphinx>
* <https://pypi.org/project/Pillow/>
* <https://github.com/cmusphinx/sphinxbase>
* <https://github.com/cmusphinx/sphinxtrain>
* <https://cmusphinx.github.io/wiki/tutorial/>