A PROJECT REPORT
ON
JARVIS: The Personal Linux Assistant

BACHELOR OF TECHNOLOGY

6th Semester

Computer Science and Engineering

Submitted By:

Sahil Patyal

- 2121677

ABSTRACT

The project aims to develop a personal-assistant for Linux-based systems. Jarvis draws its inspiration from virtual assistants like Cortana for Windows, and Siri for iOS. It has been designed to provide a **user-friendly interface** for carrying out a variety of tasks by employing certain **well-defined commands**. Users can interact with the assistant either through **voice commands** or using keyboard input.

As a personal assistant, Jarvis assists the end-user with *day-to-day activities like general human conversation, searching queries in google, bing or yahoo, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details, health recommendations based on symptoms and reminding the user about the scheduled events and tasks*. The user statements/commands are analysed with the help of **machine learning** to give an optimal solution.

Keywords:- Personal Assistant, Linux Systems, Automation, Machine Learning , Artificial Intelligence, Windows Systems

TECHNOLOGIES USED

➢ Python

What is Python?
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

LIBRARIES USED

- **import webbrowser as wb**
- **import pyttsx3 as tts**
- **import speech_recognition as sr**
- **import random as r**
- **import pywhatkit as wk**

#1 The script webbrowser can be used as a command-line interface for the module. It accepts a URL as the argument. It accepts the following optional parameters: -n opens the URL in a new browser window, if possible; -t opens the URL in a new browser page ("tab"). The options are, naturally, mutually exclusive. Usage example:

python -m webbrowser -t "https://www.python.org"

#2 pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.

Installation
pip install pyttsx3
If you recieve errors such as No module named win32com.client, No module named win32, or No module named win32api, you will need to additionally install pypiwin32.

#3 Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. This article aims to provide an introduction to how to make use of the SpeechRecognition library of Python. This is useful as it can be used on microcontrollers such as Raspberry Pi with the help of an external microphone.

Required Installations
The following must be installed:

Python Speech Recognition module:

sudo pip install SpeechRecognition

#4 . random.random() function generates random floating numbers in the range[0.1, 1.0). (See the opening and closing brackets, it means including 0 but excluding 1). It takes no parameters and returns values uniformly distributed between 0 and 1.

Syntax : random.random()

Parameters : This method does not accept any parameter.

Returns : This method returns a random floating number between 0 and 1.

- **Software:** The Docker daemon, called dockerd, is a persistent process that manages Docker containers and handles container objects. The daemon listens for requests sent via the Docker Engine API. The Docker client program, called docker, provides a command-line interface that allows users to interact with Docker daemons.
- **Objects:** Docker objects are various entities used to assemble an application in Docker. The main classes of Docker objects are images, containers, and services.
    - A Docker container is a standardized, encapsulated environment that runs applications. A container is managed using the Docker API or CLI
    - A Docker image is a read-only template used to build containers. Images are used to store and ship applications.[34]
    - A Docker service allows containers to be scaled across multiple Docker daemons. The result is known as a *swarm*, a set of cooperating daemons that communicate through the Docker API.
- **Registries:** A Docker registry is a repository for Docker images. Docker clients connect to registries to download ("pull") images for use or upload ("push") images that they have built. Registries can be public or private. Two main public registries are Docker Hub and Docker

#5 PyWhatKit is a Python library with various helpful features. It's easy-to-use and does not require you to do any additional setup. Currently, it is one of the most popular library for WhatsApp and YouTube automation. New updates are released frequently with new features and bug fixes.

Features
1. Sending Message to a WhatsApp Group or Contact

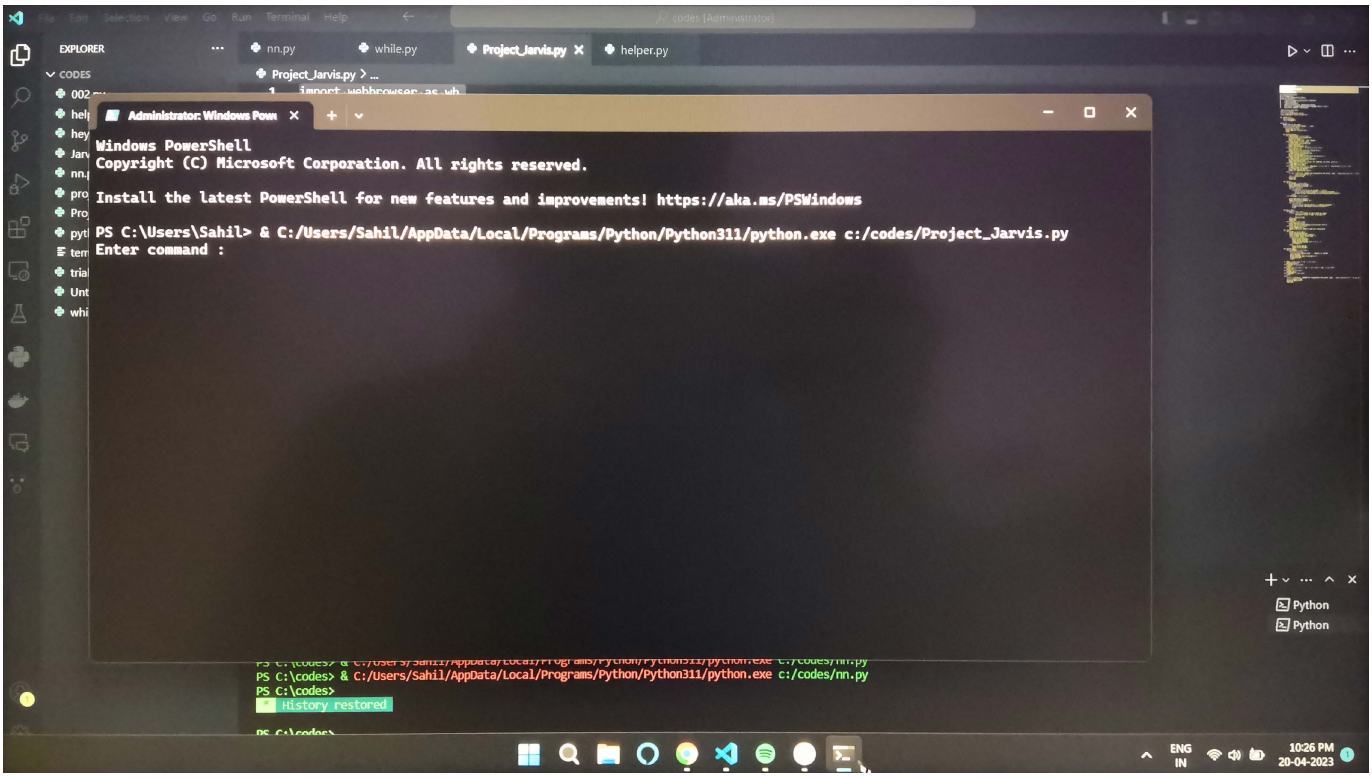
2. Sending Image to a WhatsApp Group or Contact

3. Converting an Image to ASCII Art
Converting a String to Handwriting

4. Playing YouTube Videos

5. Sending Mails with HTML Code

Running JARVIS IN WINDOWS CLI



INTEGRATION

Docker can be integrated into various infrastructure tools, including Amazon Web Services , Ansible CFEngine, Chef Google Cloud Platform , IBM Bluemix, HPE Helion Stackato, Jelastic, Jenkins, Kubernetes, Microsoft Azure, OpenStack Nova, OpenSVC, Oracle Container Cloud Service , Puppet, ProGet, Salt,Vagrant, and VMware vSphere Integrated Containers.

The Cloud Foundry Diego project integrates Docker into the Cloud Foundry PaaS.

Nanobox uses Docker (natively and with VirtualBox) containers as a core part of its software development platform.

Red Hat's OpenShift PaaS integrates Docker with related projects (Kubernetes, Geard, Project Atomic and others) since v3 (June 2015).

The Apprenda PaaS integrates Docker containers in version 6.0 of its product.

Jelastic PaaS provides managed multi-tenant Docker containers with full compatibility to the native ecosystem.

The Tsuru PaaS integrates Docker containers in its product in 2013, the first PaaS to use Docker in a production environment.

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms. It is quite similar to HP Quick Test Pro (QTP now UFT) only that Selenium focuses on automating web-based applications. Testing done using Selenium tool is usually referred to as Selenium

#1 Performance: This could be surprising but in most of the applications as Go is faster than Python (2 and 3). The result of the Benchmarking Game, used to determine the faster programming language, clearly favours Go, because of its concurrency model and CPU scalability. Whenever we need to process some internal request we are doing it with a separate Goroutine, which is 10 times cheaper in resources than Python threads, thus saving us a lot of resources (Memory, CPU, etc.) because of the built-in language features.

#2 You don't need web framework for Go: This is the most awesome thing about the programming language. Go language creators and the community have built in so many tools natively supported by language core, that in most of the cases you don't need any 3rd party library. For example, it has HTTP, JSON, HTML templating built in language natively and you can build very complex API services without even thinking about finding the library on Github. Though there are a lot of libraries and frameworks built for Go and making web applications with Go, we will recommend building your web application or API service without any 3rd party library because in most cases they are not making your life easier than using native packages.

#3 Great IDE support and debugging: IID Support is one of the most important things when you are trying to switch your programming language. Comfortable IDE on average can save up to 80% of your coding time. We found Go Plugin For JetBrains IDEA which has support also for Webstorm, PHPStorm, etc. This plugin is giving everything that you need for project

DevOps is often viewed as an approach to applying systems administration work to cloud technology.

WinOps

WinOps is the term used for DevOps practices for a Microsoft-centric view.

Goals

The goals of DevOps span the entire delivery pipeline. They include Improved deployment frequency:

- Faster time to market;
- Less failure rate of new releases;
- Shortened lead time between fixes;
- Faster mean time to recovery (in the event of a new release crashing or otherwise disabling the current system).

Simple processes become increasingly programmable and dynamic, using a DevOps approach. DevOps aims to maximize the predictability, efficiency, security, and maintainability of operational processes. Very often, automation supports this objective.

DevOps integration targets product delivery, continuous testing, quality testing, feature development, and maintenance releases in order to improve reliability and security and provide faster development and deployment cycles. Many of the ideas (and people) involved in DevOps came from the enterprise systems management and agile software development movements.

Companies that practice DevOps have reported significant benefits, including significantly shorter time to market , improved customer satisfaction, better product quality, more reliable releases, improved productivity and efficiency, and the increased ability to build the right product by fast experimentation.

Deployment

Companies with very frequent releases may require knowledge of DevOps. For example, the company that operates an image hosting website Flickr developed a DevOps approach to support ten deployments a day. Daily deployment cycle would be much higher at organizations producing multi-focus or multi-function applications. Daily deployment is referred to as continuous deployment or continuous delivery and has been associated with the lean startup methodology. Professional associations and blogs posts have formed on the topic since 2009.

DevOps automation

DevOps automation can be achieved by repackaging platforms, systems, and applications into reusable building blocks through the use of technologies such as virtual machines and containerization.

Implementation of DevOps automation in the IT-organization is heavily dependent on tools, which are required to cover different areas of the systems development lifecycle (SDLC):

1. Infrastructure as code — Ansible, Terraform, Puppet, Chef
2. CI/CD — Jenkins, TeamCity, Shippable, Bamboo, Azure DevOps
3. Test automation — Selenium, Cucumber, Apache JMeter
4. Containerization — Docker, Rocket, Unik
5. Orchestration — Kubernetes, Swarm, Mesos
6. Deployment — Elastic Beanstalk, Octopus, Vamp
7. Measurement — NewRelic, Kibana, Datadog, DynaTrace
8. ChatOps — Hubot, Lita, Cog

Source Code

```
import webbrowser as wb
import pyttsx3 as tts
import speech_recognition as sr
import random as r
import pywhatkit as wk

#sr funnction works here!
r=sr.Recognizer()
mic = sr.Microphone(device_index=1)
with mic as source:
    r.adjust_for_ambient_noise(source, duration=1)
    print("Listening...")
    r.pause_threshold=2
    audio =r.listen(source, phrase_time_limit=5)
    user_input= r.recognize_google(audio,language='eng-in').lower()
    print("Your Command :",user_input)

 #tts function works here!
engine = tts.init()
voice =engine.getProperty('voices')
engine.setProperty('voice',voice[0].id)
 def speak(audio):
    engine.say(audio)
    engine.runAndWait()

 while True:
    #takes text as user_input!
    user_input= input("Enter command : ").lower()
    if(user_input == 'exit'):
        print("see you soon...")
        speak("Bye sir!, See you soon")
        break

    def execute_browser():
        if ("open youtube" in user_input):
            wb.open("http://www.youtube.com")
            print("Executing...")
            speak("sir, Executing... your command")
        elif("open google" in user_input):
            wb.open("http://www.Google.com")
            print("Executing...")
            speak("opening google, sir")
        elif(user_input == 'hi'or 'hi jarvis' in user_input):
            print("Hi Sahil!")
            speak("Hi Sir, what would you like me to do.")
```

```
elif("open spotify" in user_input):
        wb.open(f"http://www.spotify.com")
        print("opening spotify...")
        speak("you have good taste of music by the way sir")
    elif("open chat gpt" in user_input):
        wb.open("https://chat.openai.com/chat")
        print("getting Chat GPT...")
        speak("its very difficult to say, he's good but, not better as me sir ")
    elif("open edge" in user_input):
        wb.get('Microsoft Edge')
    elif("open mail"in user_input or "open gmail" in user_input or "show my
mails" in user_input):
        wb.open("https://mail.google.com/")
        print("Getting your Mails")
        speak("sir, getting your mails")
    else:
        string = ["sorry sir, Command not recognized by the system","oops! , really
very sorry sir ","I will work on this thing, sir",
                "soon i will get it, sir"]
        say = r.choice(string)
        print(say)
        speak(say)
def search():
    if('on youtube' in user_input):
        print("Searching on Youtube")
        speak("Searching on Youtube, sir")
        searchh = user_input
        for i in searchh:
            if("search"in searchh and "on youtube" in searchh):
                search= searchh.replace("search","").replace("on youtube","")
                wb.open(f"https://www.youtube.com/results?
search_query={search}")
                break
    elif('on google' in user_input):

        print("Searching on google")
        speak("Searching on google, sir")
        searchh = user_input
        for i in searchh:
            search = searchh.replace("search","").replace("on google","")
            wb.open(f"https://www.google.com/search?q={search}")
            break

    def basic_chat():
        if('joke' in user_input):
            string = ["sir, it's harder for me to make you laugh!",
                "My sense of humour is bad, so no jokes sir"
                ]
            say = r.choice(string)
            print(say)
            speak(say)
```

```
elif("story" in user_input):
        print('Getting a story.')
        speak('you are already making a story on your life sir')
    elif("are you" in user_input):
        print("JARVIS here!")
        speak("I am Jarvis, Ready to make your things done")
  def core():
    if('system' in user_input):
        print('Verifying Identity')
        speak("please confirm your identity")
        password=input("Enter Access Code : ")
        if(password == 'karan'):
            print("Access Granted")
            speak("Access Granted")
            speak("Sir,Shuting down in 5, 4. 3, 2. 1")
            wk.shutdown(7)
        else:
            print("Access Denied")
            speak("access denied")
  def execute_play():
    if('on youtube' in user_input):
        for i in user_input:
         ip1=user_input.replace('play','').replace('on youtube','')
         print("Playing...")
         speak(f"playing {ip1} on youtube sir!")
         wk.playonyt(ip1)
         break

if('open' in user_input or 'hi' in user_input):
    execute_browser()
  elif("search" in user_input):
    search()
  elif('tell' in user_input or 'say' in user_input or 'who' in user_input):
    basic_chat()
  elif('play' in user_input):
    execute_play()
  elif('shutdown' in user_input):
    core()
  else:
    string = ["sorry sir, Command not recognized by the system","oops! , really very
sorry sir ","I will work on this thing, sir"]
    say = r.choice(string)
    print(say,"Main error")
    speak(say)
```

#Code Ends here

Features in JARVIS

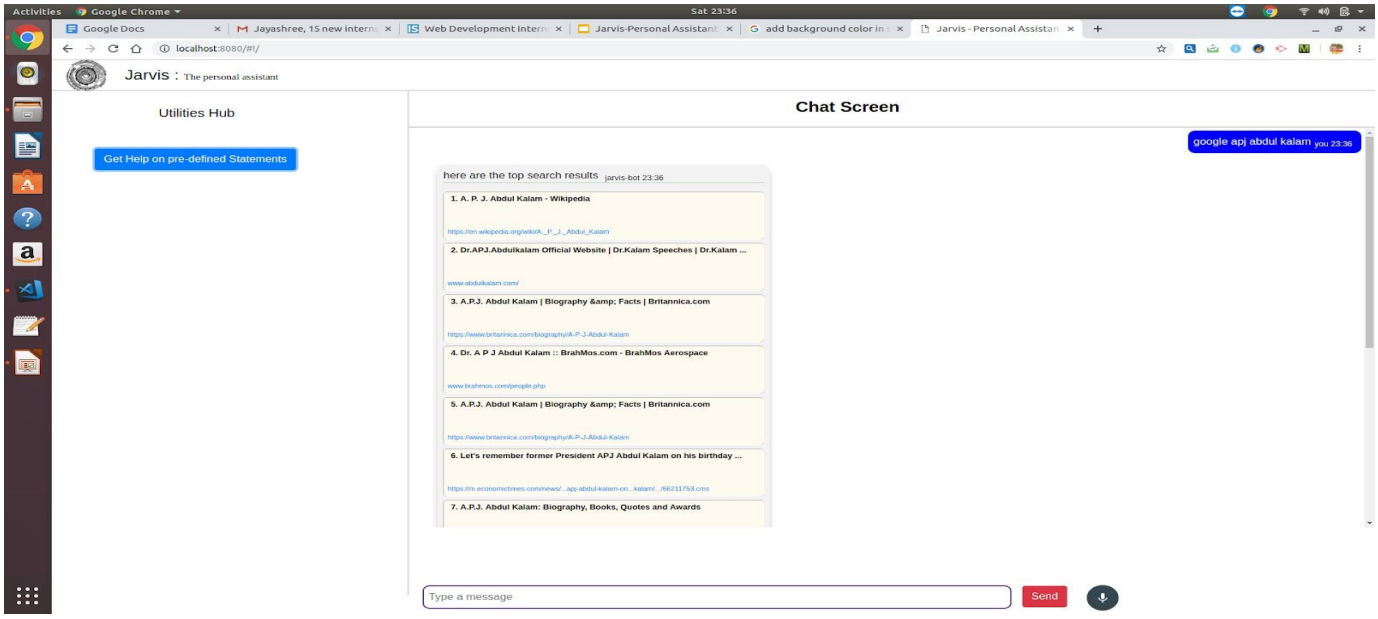
1. Queries from the web:

Making queries is an essential part of one's life, and nothing changes even for a developer working on Linux. We have addressed the essential part of a netizen's life by enabling our voice assistant to search the web. Here we have used Node JS and Selenium framework for extracting the result from the web as well as displaying it to the user. Jarvis supports a plethora of search engines like Google, Bing and Yahoo and displays the result by scraping the searched queries.

In order to make queries from different search engines, the given format should be adopted:

<search engine name> <query>

Jarvis supports Google, Bing and Yahoo, which should precede the desired query.

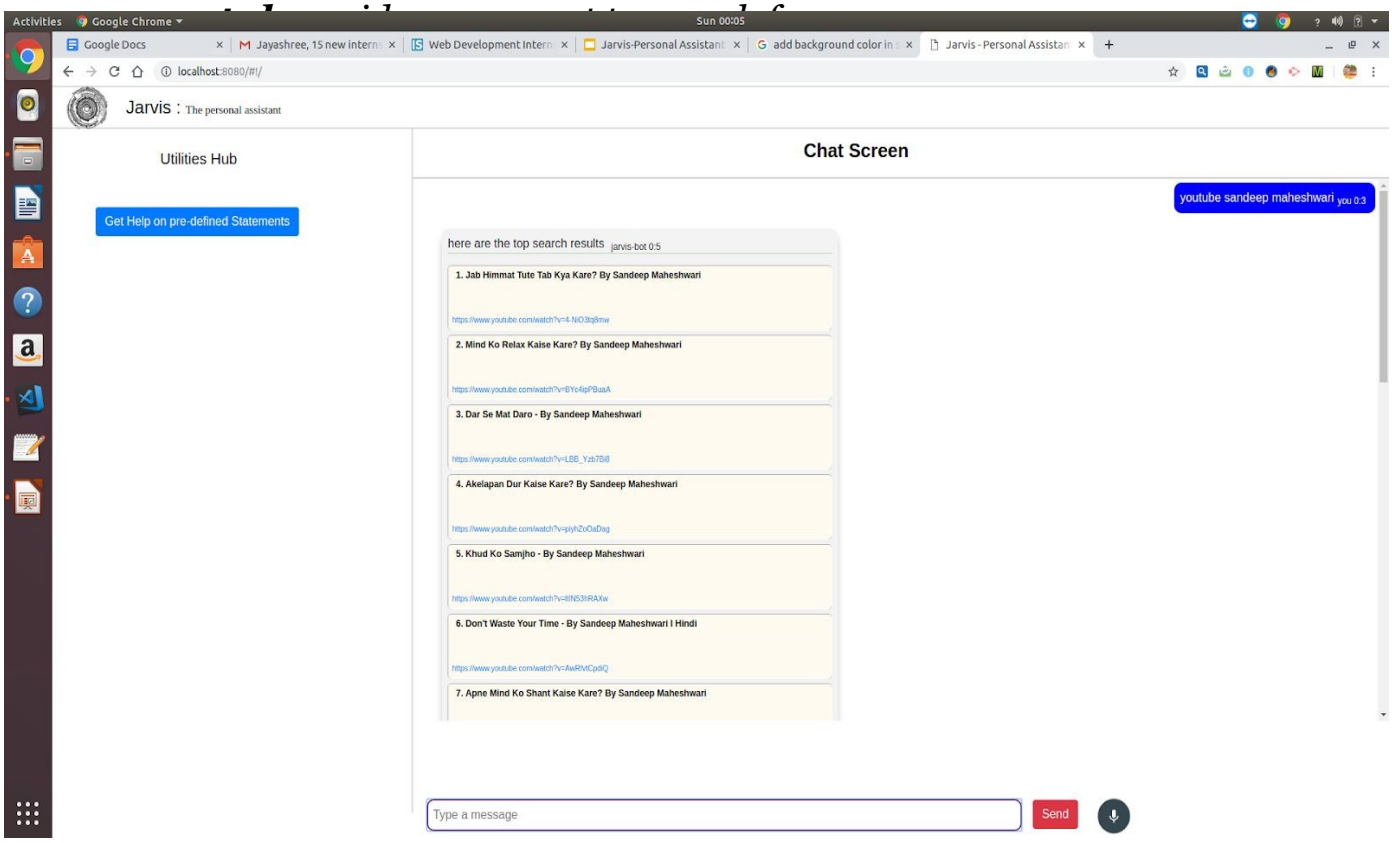


2. Accessing youtube videos

Videos have remained as a main source of entertainment, one of the most prioritized tasks of virtual assistants. They are equally important for entertainment as well as educational purposes as most teaching and research activities in present times are done through Youtube. This helps in making the learning process more practical and out of the four walls of the classroom.

Jarvis implements the feature through a subprocess module which is handled by the main Golang service. This service initiates the subprocess for Node JS which serves the Selenium WebDriver, and scraps the searched YouTube query.

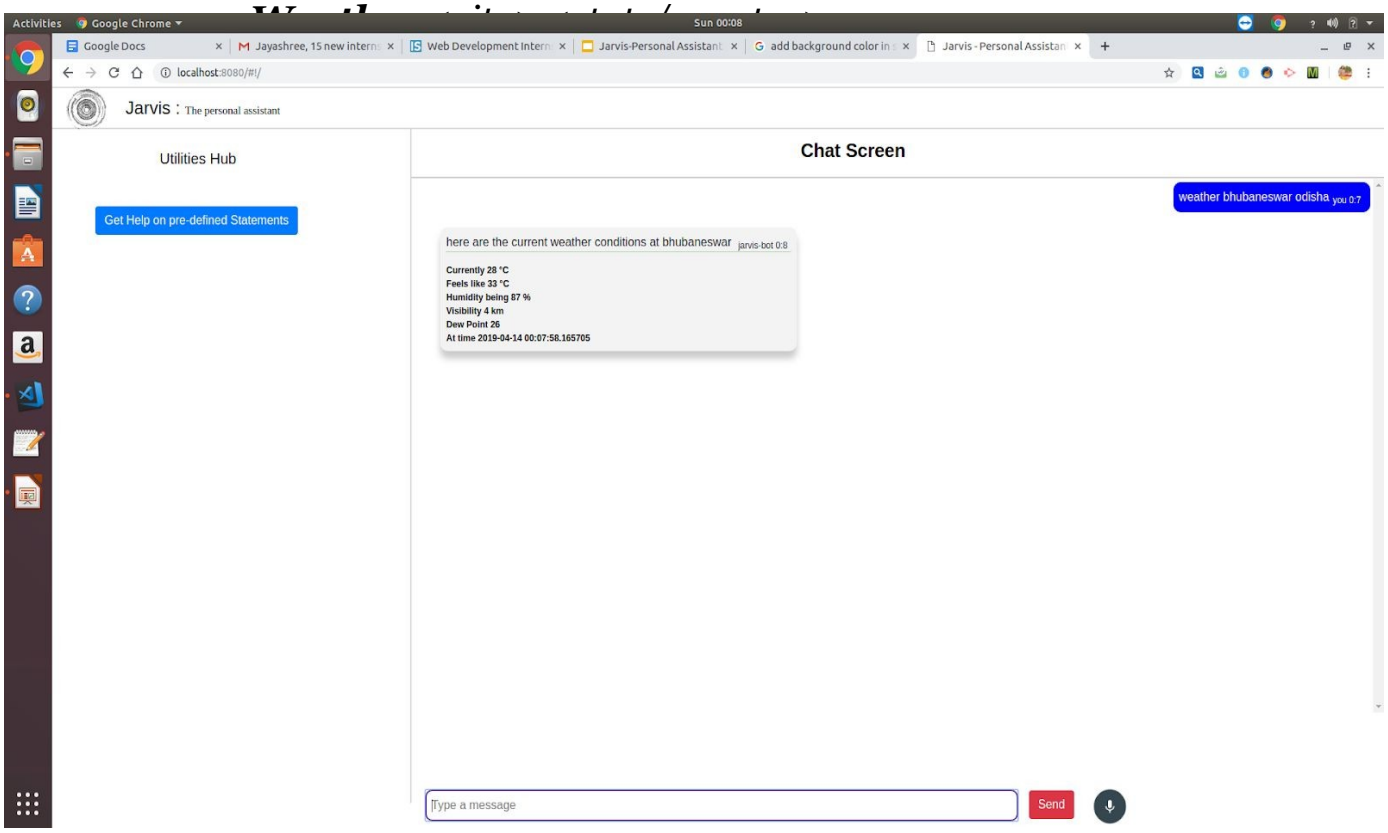
In order to access videos from youtube format is:



3. Get weather for a location

Getting live weather conditions about a place remains an important task of virtual assistants. It helps the user charter the course of their action. Jarvis addresses this issue with the help of Python.

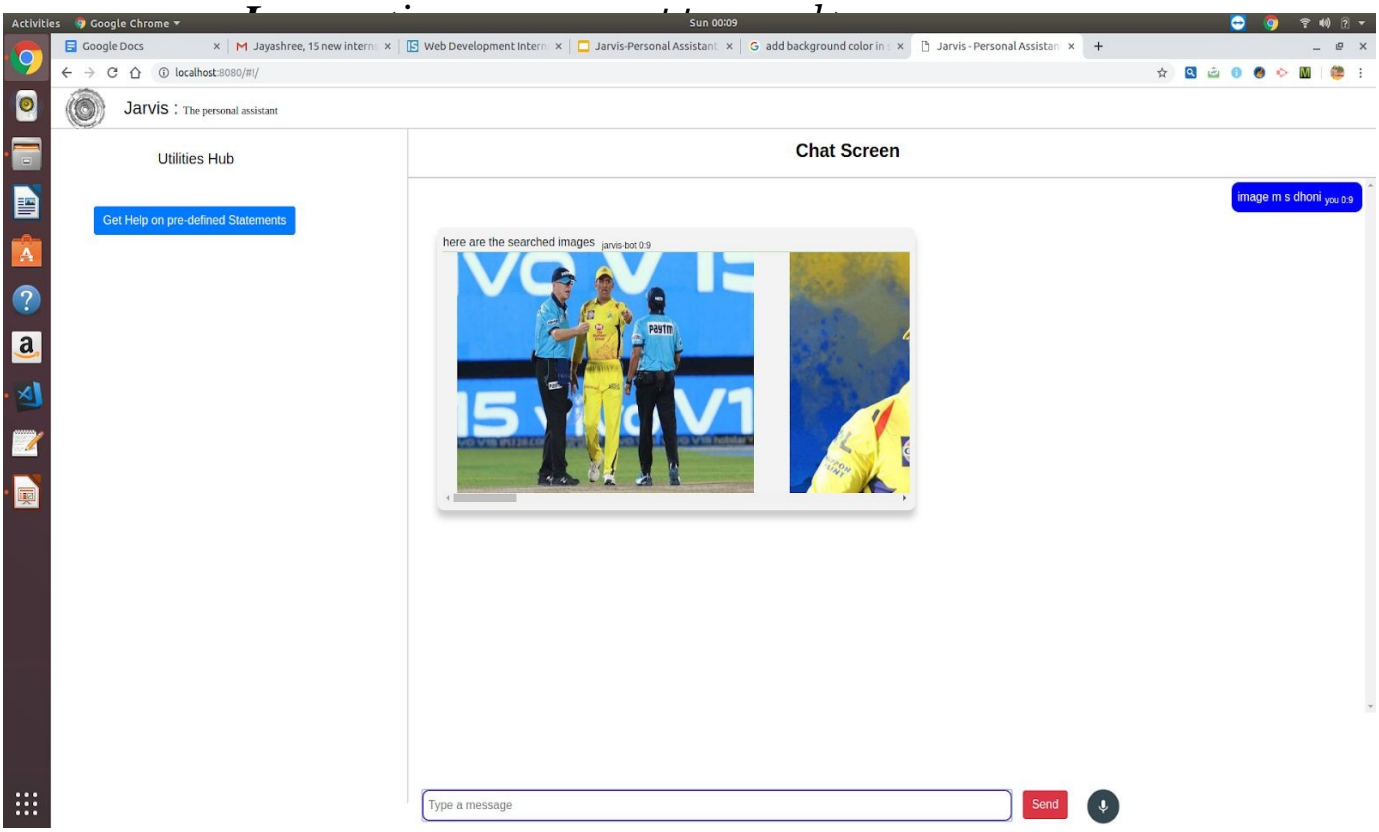
In order to access the live weather condition format is:



4. Retrieve images

Users could get images directly through the Jarvis interface. This implementation is done using the Selenium WebDriver. The images are derived as iframes from the entire web code received from Google images. These are formatted according to use and displayed in a compact manner in the Jarvis interface.

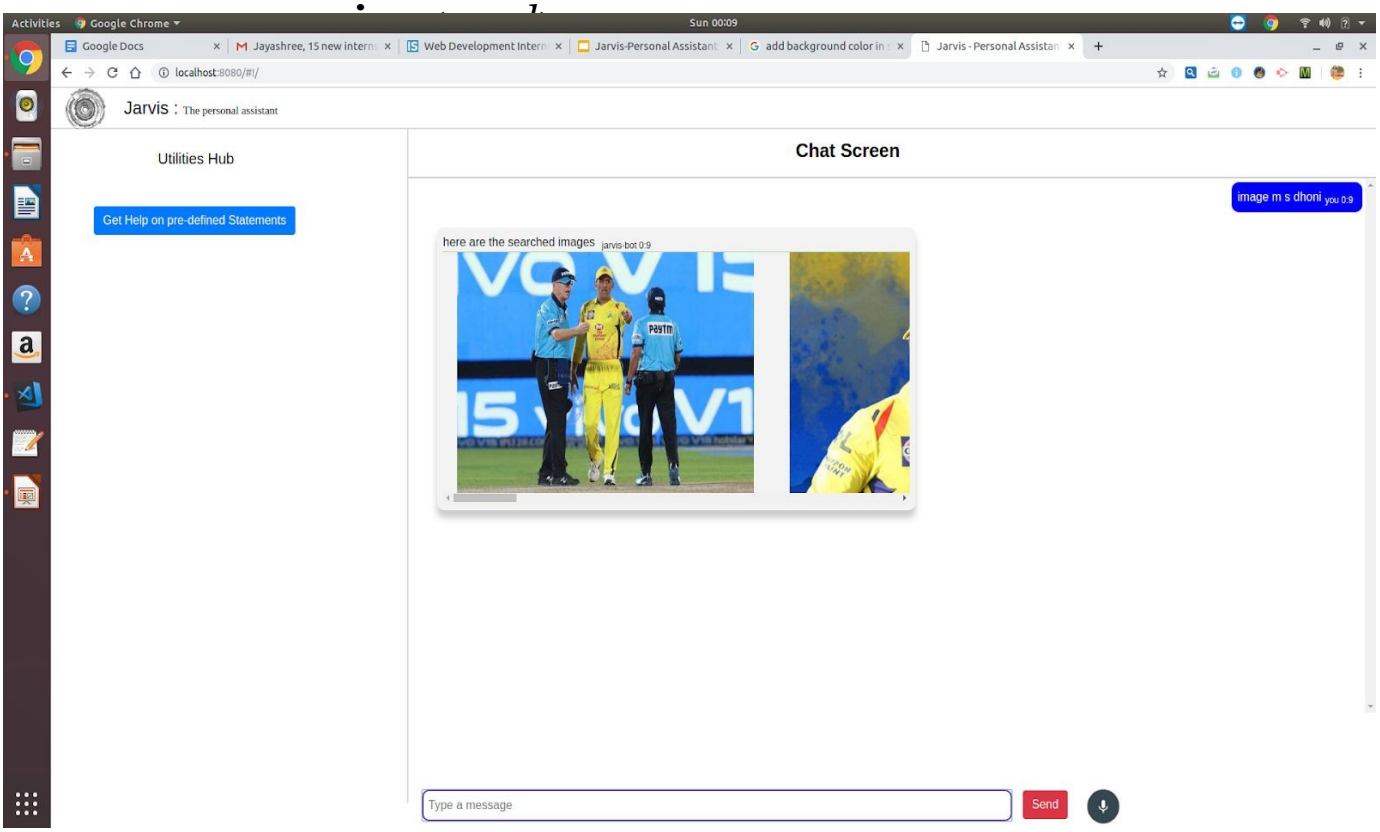
In order to retrieve image format is:



5. Dictionary meaning

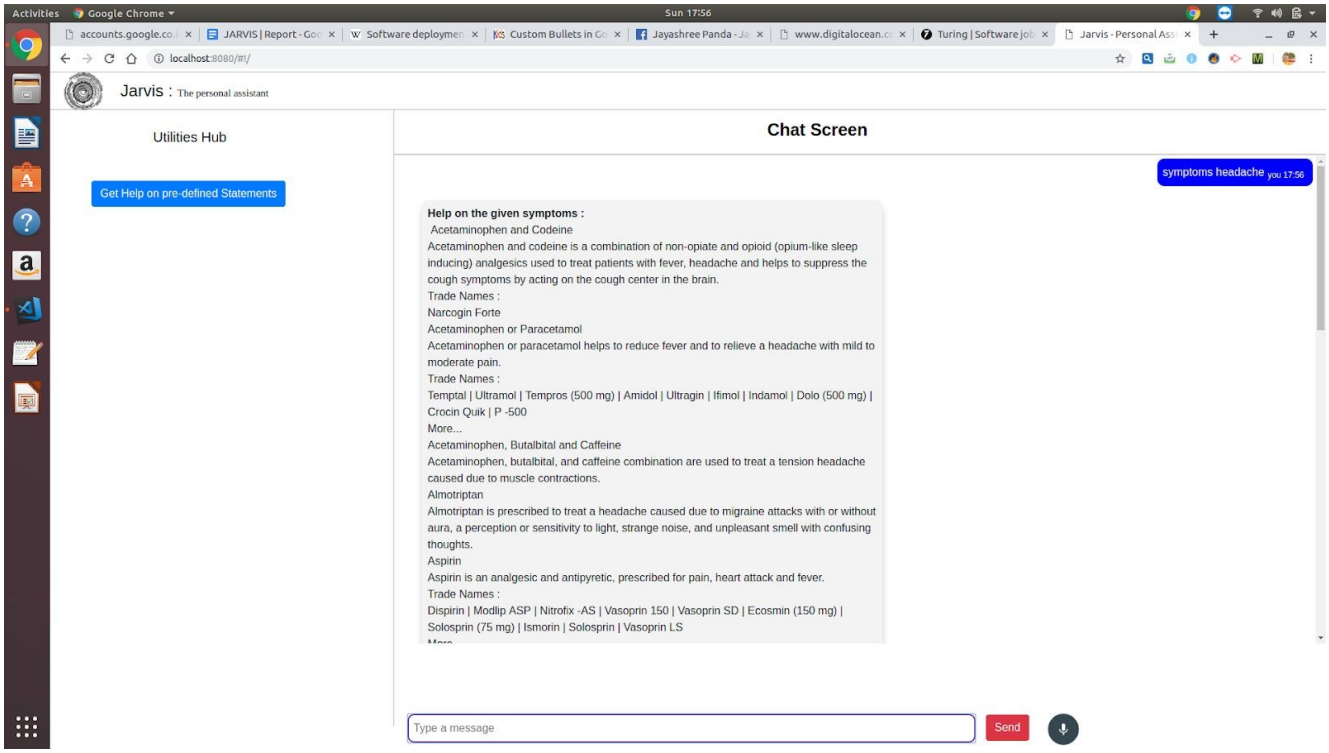
One of the usages of the web is to find word meaning and its usage in our day to day life. Instead of going through the bulky books, our users can simply search for it using the voice assistant and get the meaning within a fraction of seconds.

For retrieving the meaning of a word format is,



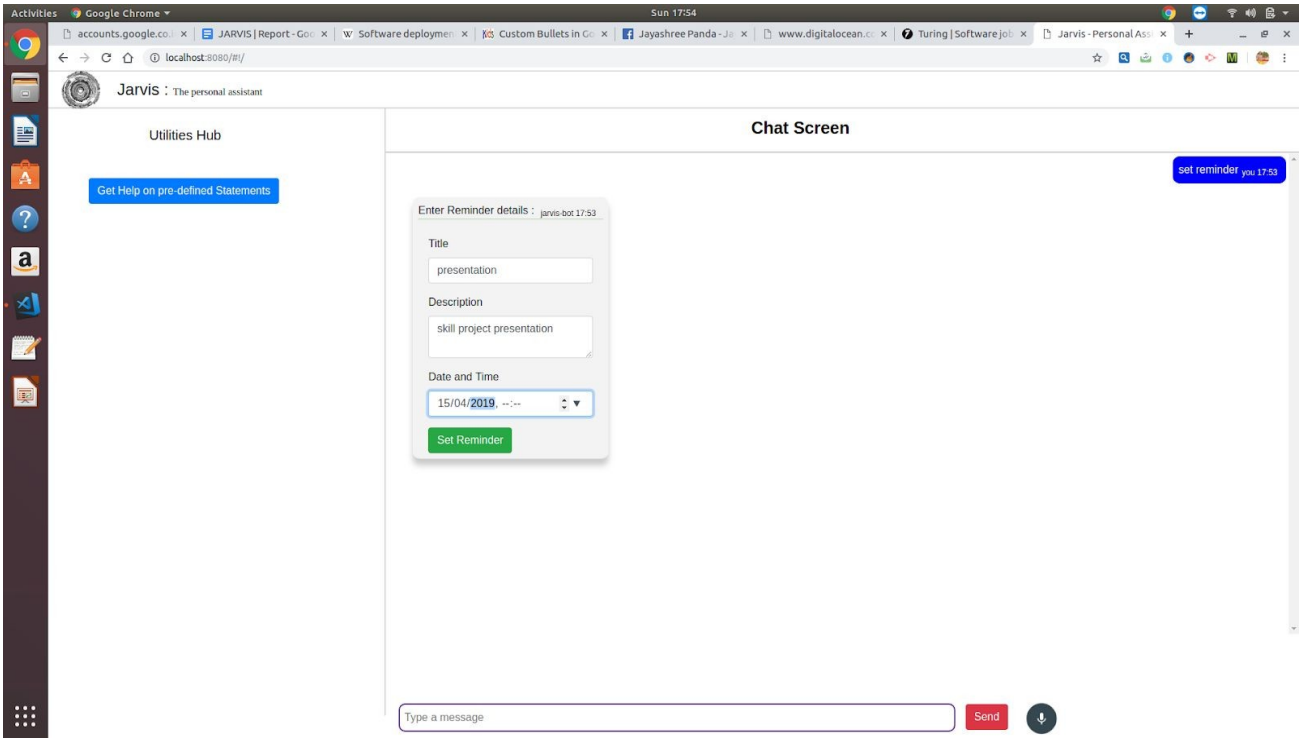
6. Medicine Details

One of the important issue Jarvis addresses is of healthcare, and medicine in general. The user can query either the medicine or the symptoms. The former lets you know the complete details of the medicine, like indications, contradictions, trade or brand names, dosage, the process of consumption, warning and precautions, storage conditions, etc. On the other hand, the symptom feature lets



7. Set Reminders

One of the main features of a voice assistant is to set a reminder for the user accordingly. Jarvis is no different when it comes to this. The user can set reminders to be notified about a task at a particular time. This will help users, especially developers to schedule their time and resources easily. All the user have to do is to input ***Set reminder*** to the assistant**.** A form will be displayed. Fill the form with the required details and click on **set reminder** button.



8. Sending Emails

Integrating mailing features to Jarvis eases the job of mailing, which otherwise would have to be done by opening the concerned email address. With Jarvis, you do not need to go for another tab to do one of the major task of your day to day affairs. The user can send emails to the desired receiver. He should input ***Send mail***, after which a form will be displayed. Fill the form with the required details and click on the **send mail** button.

FUTURE PROSPECTIVE

We plan to Integrate Jarvis with mobile using react native, to provide a synchronized experience between the two connected devices.

Further, in the long run, Jarvis is planned to feature auto deployment supporting elastic beanstalk, backup files, and all operations which a general Server Administrator does. The functionality would be seamless enough to replace the Server Administrator with Jarvis.

Functional Requirements:

- Linux Distribution
- Proper Internet Connection
- Github Credentials
- Docker installed
- Python 2.7
- Windows CLI
- Mplayer for voice support (Text-to-Speech)
- Chromium-based browser, like Chrome, Edge

Non-Functional Requirements:

The non-functional requirements of the system include:

- The system ensures safety, security and usability, which are observable during operation (at run time).
- The system is adaptable to different situations.
- The project has good and compact UI using AngularJS with responsive interface.
- The project is light on resources.

CONCLUSION

Through this voice assistant, we have automated various services using a single line command. It eases most of the tasks of the user like searching the web, retrieving weather forecast details, vocabulary help and medical related queries. We aim to make this project a complete server assistant and make it smart enough to act as a replacement for a general server administration. The future plans include integrating Jarvis with mobile using React Native to provide a synchronised experience between the two connected devices. Further, in the long run, Jarvis is planned to feature auto deployment supporting elastic beanstalk, backup files, and all operations which a general Server Administrator does. The functionality would be seamless enough to replace the Server Administrator with Jarvis.