A Project/Dissertation Review-1 Report

on

Artificial Intelligence in Voice Assistant

Submitted in partial fulfillment of the
requirement for the award of the degree of

B.Tech CSE (AIML)

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Name of Supervisor :
Designation

Submitted By

Satyam Pandey – 20SCSE1180077
Mrigank Singh – 20SCSE1180082
Aditya Raj – 20SCSE1180041

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
OCT, 2022

<u>ABSTRACT</u>

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

<u>CONTENT</u>

- ❖ Declaration
- ❖ Certificate
- ❖ Acknowledgement
- ❖ Abstract
- ❖ Problem Statement
- ❖ Scope
- ❖ Technologies Stack Used:
- ❖ Docker Container
- ❖ Selenium Automation tool
- ❖ Subprocesses/Child Process
- ❖ Golang
- ❖ DevOps
- ❖ Relationship to other approaches.
- ❖ Sorensen-Dice Coefficient
- ❖ Features in Jarvis
- ❖ Future Prospectives
- ❖ Software Requirements and Specification
- ❖ DFD's of our Virtual Assistant
- ❖ Functional Requirements
- ❖ Non-Functional Requirements
- ❖ Conclusion

PROBLEM STATEMENT

We are all well aware about Cortana, Siri, Google Assistant and many other virtual assistants which are designed to aid the tasks of users in Windows, Android and iOS platforms. But to our surprise, there's no such virtual assistant available for the paradise of Developers i.e. Linux platform.

PURPOSE

This Software aims at developing a personal assistant for Linux-based systems. The main purpose of the software is to perform the tasks of the user at certain commands, provided in either of the ways, speech or text. It will ease most of the work of the user as a complete task can be done on a single command. Jarvis draws its inspiration from Virtual assistants like Cortana for Windows and Siri for iOS. Users can interact with the assistant either through voice commands or keyboard input.

PRODUCT GOALS AND OBJECTIVES

Currently, the project aims to provide the Linux Users with a Virtual Assistant that would not only aid in their daily routine tasks like searching the web, extracting weather data, vocabulary help and many others but also help in automation of various activities.

In the long run, we aim to develop a complete server assistant, by automating the entire server management process - deployment, backups, auto-scaling, logging, monitoring and make it smart enough to act as a replacement for a

general server administrator.

PRODUCT DESCRIPTION

As a personal assistant, Jarvis assists the end-user with day-to-day activities like general human conversation, searching queries in various search engines like 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.

SCOPE

Presently, Jarvis is being developed as an automation tool and virtual assistant. Among the Various roles played by Jarvis are:

1. Search Engine with voice interactions
2. Medical diagnosis with Medicine aid.
3. Reminder and To-Do application.
4. Vocabulary App to show meanings and correct spelling errors.
5. Weather Forecasting Application.

There shall be proper Documentation available on its Official Github repository for making further development easy and we aim to release our virtual assistant as an Open Source Software where modifications and contributions by the community are warmly welcomed.

Link to Github Repository:
https://github.com/Harkishen-Singh/Jarvis-personal-assistant

TECHNOLOGIES USED

FRONTEND FRAMEWORK
> ➢ AngularJS

BACKEND STACK
> ➢ GO-lang
> ➢ Machine Learning
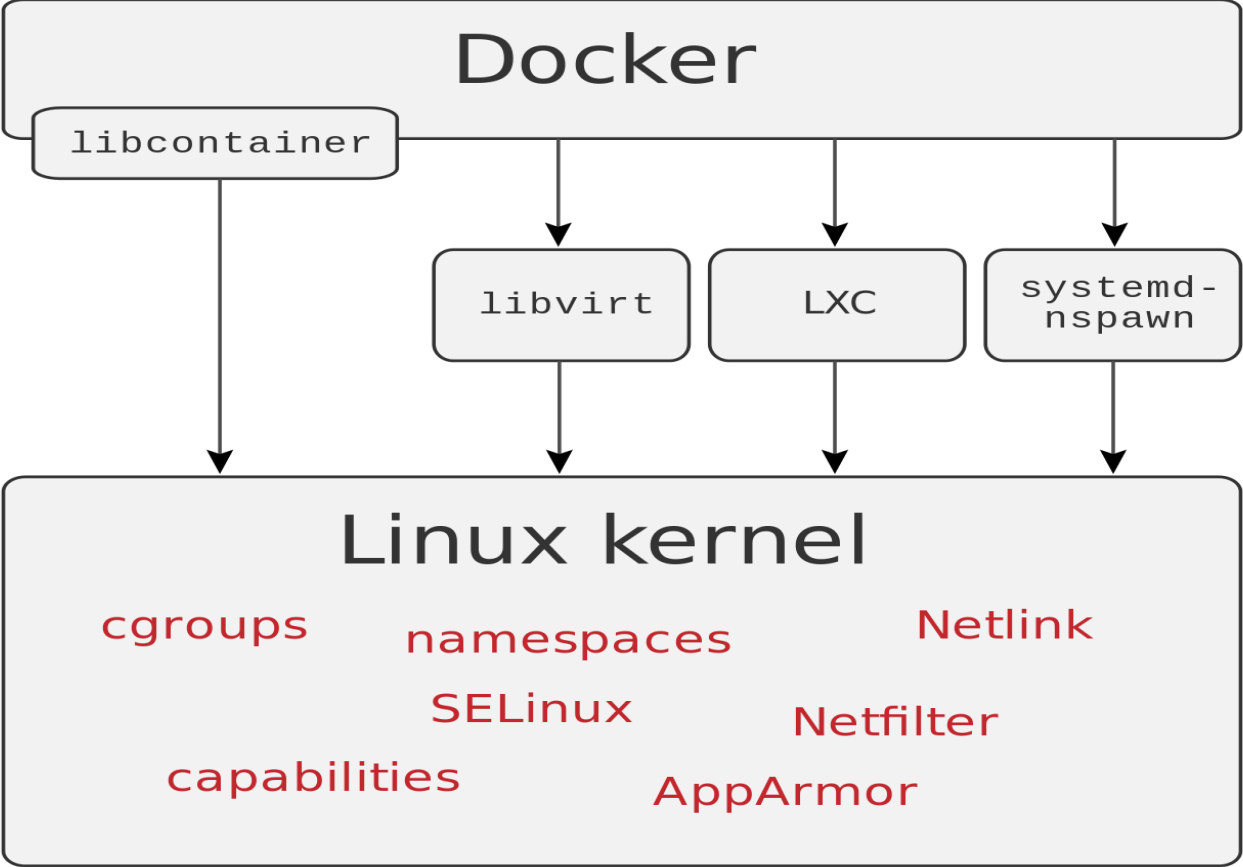> ➢ Docker Container

DATABASE
> ➢ SQLite
> ➢ Cookies

DOCKER CONTAINER

Docker is a computer program that performs operating-system-level virtualization. It is used to run software packages called containers. Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

Docker is developed primarily for Linux, where it uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs). The Linux kernel's support for namespaces mostly isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU.

Docker can use various interfaces to access virtualisation features of the kernel. A Docker container, unlike a virtual machine, does not require or include a separate operating system. Instead, it relies on the kernel's functionality and uses resource isolation for CPU and memory, and separate namespaces to isolate the application's view of the operating system. Docker accesses the Linux kernel's virtualization features either directly using the libcontainer library, which is available as of Docker 0.9, or indirectly via libvirt, LXC (Linux Containers).

COMPONENTS

The Docker software is a service consisting of three components:

- **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

Cloud. Docker Hub is the default registry where Docker looks for images. Docker registries also allow the creation of notifications based on events.

Tools

- **Docker Compose** is a tool for defining and running multi-container Docker applications. It uses YAML files to configure the application's services and performs the creation and start-up process of all the containers with a single command. The docker-compose CLI utility allows users to run commands on multiple containers at once, for example, building images, scaling containers, running containers that were stopped, and more. Commands related to image manipulation, or user-interactive options, are not relevant in Docker Compose because they address one container. The **docker-compose.yml** file is used to define an application's services and includes various configuration options. For example, the build option defines configuration options such as the Dockerfile path, the command option allows one to override default Docker command, and more.]The first public version of Docker Compose (version 0.0.1) was released on December 21, 2013. The first production-ready version (1.0) was made available on October 16, 2014.
- **Docker Swarm** provides native clustering functionality for Docker containers, which turns a group of Docker engines into a single virtual Docker engine.] In Docker 1.12 and higher, Swarm mode is integrated with Docker Engine. The swarm CLI utility allows users to run Swarm containers, create discovery tokens, list nodes in the cluster, and more. The docker node CLI utility allows users to run various commands to manage nodes in a swarm, for example, listing the nodes in a swarm, updating nodes, and removing nodes from the swarm. Docker manages swarms using the Raft Consensus Algorithm. According to Raft, for an

update to be performed, the majority of Swarm nodes need to agree on the update.

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 AUTOMATION TOOL

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

Testing.

Selenium is not just a single tool but a suite of software, each catering to different testing needs of an organization.

The entire Selenium Tool Suite is comprised of four components:

- Selenium IDE, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.
- Selenium Remote Control, also known as Selenium 1, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.
- WebDriver, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.
- Selenium Grid is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.
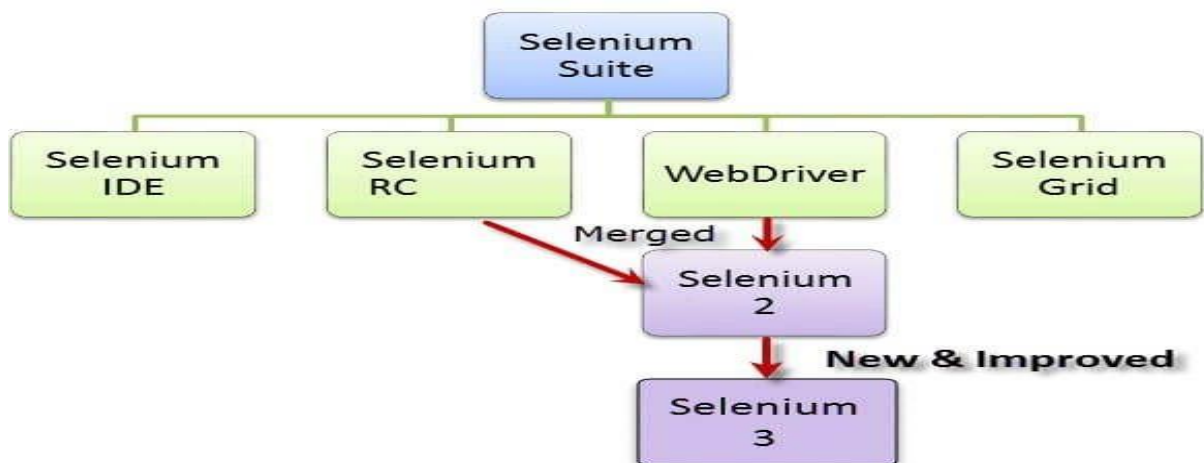
Selenium RC and WebDriver were merged to form Selenium. Selenium is more advantageous than QTP in terms of costs and flexibility. It also allows you to run tests in parallel, unlike in QTP where you are only allowed to run tests sequentially.

SUB-PROCESSES/CHILD PROCESS

A subprocess is a process started by another program. There are two major procedures for creating a child process: the fork system call (preferred in Unix-like systems and the POSIX standard) and the spawn (preferred in the modern (NT) kernel of Microsoft Windows, as well as in some historical operating systems).

A child process inherits most of its attributes, such as file descriptors, from its parent. In Unix, a child process is typically created as a copy of the parent, using the fork system call. The child process can then overlay itself with a different program (using exec) as required.

Each process may create many child processes but will have at most one parent process; if a process does not have a parent this usually indicates that it was created directly by the kernel. In some systems, including Linux-based systems, the very first process (called init) is started by the kernel at booting time and never terminates (see Linux startup process); other parentless processes may be launched to carry out various daemon tasks in userspace. Another way for a process to end up without a parent is if its parent dies, leaving an orphan process; but in this case, it will shortly be adopted by init.

When a child process terminates, some information is returned to the parent process. When a child process terminates before the parent has called wait, the kernel retains some information about the process, such as its exit status, to enable its parent to call wait later. Because the child is still consuming system resources but not executing it is known as a zombie process.

Go-Lang

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software. Go is syntactically similar to C, but with memory safety, garbage collection, structural typing, and CSP-style concurrency. The main reasons why we chose Go for this project are:

#1 It Compiles Into Single Binary: Golang is built as a compiled language and Google developers did a great job with it. Using static linking it is actually combining all dependency libraries and modules into one single binary file based on OS type and architecture. This means if you are compiling your backend application on your laptop with Linux X86 CPU you can just upload compiled binary into the server and it will work, without installing any dependencies there.

#2 Static Type System: Type system is really important for large scale applications. Python is great and fun language but sometimes we get unusual exceptions because of using the variable as an integer only to find out that it's a string. Go will let you know about this issue during compile time as a compiler error, thus saving your time and the hassle.

#3 Performance: This could be surprising but in most of the application cases 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.

#4 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.

#5 Great IDE support and debugging: IDE 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

development. With the power of JetBrains IDEA, you can really boost your development.

DevOps

This project extensively uses DevOps to speed up the development process and make the entire process of delivery of code seamless by using Travis CI builds along with Heroku based deployment services

.**DevOps** is a set of software development practices that combines software development (*Dev*) and information technology operations(*Ops*) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives. It is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production while ensuring high quality.

As DevOps is intended to be a cross-functional mode of working, those that practice the methodology use different sets of tools—referred to as "toolchains"—rather than a single one. These toolchains are expected to fit into one or more of the following categories, reflective of key aspects of the development and delivery process:

1. Coding – code development and review, source code management tools, code merging
2. Building – continuous integration tools, build status
3. Testing – continuous testing tools that provide feedback on business risks
4. Packaging – artifact repository, application pre-deployment staging
5. Releasing – change management, release approvals, release automation
6. Configuring – infrastructure configuration and management, infrastructure as code tools

7. Monitoring – applications performance monitoring, end-user experience

Some categories are more essential in a DevOps toolchain than others; especially continuous integration (e.g. Jenkins) and infrastructure as code.

Relationship to other approaches

AGILE

Agile and DevOps both often utilize practices such as automated build and test, continuous integration, and continuous delivery. Agile can be viewed as addressing communication gaps between customers and developers, while DevOps addresses gaps between developers and IT operations/infrastructure. Also, DevOps has focused on the deployment of developed software, whether it is developed via Agile or other methodologies.

ArchOps

ArchOps presents an extension for DevOps practice, starting from software architecture artefacts, instead of source code, for operational deployment. ArchOps states that architectural models are first-class entities in software development, deployment, and operations.

Continuous delivery

Continuous delivery and DevOps have common goals and are often used in conjunction, but there are subtle differences.

While continuous delivery is focused on automating the processes in software delivery, DevOps also focuses on the organization change to support great collaboration between the many functions involved.[19]

DevOps and continuous delivery share a common background in agile methods and lean thinking: small and frequent changes with focused value to the end customer. They are well communicated and collaborated internally, thus helping achieve faster time to market, with reduced risks.

DataOps

The application of continuous delivery and DevOps to data analytics has been termed DataOps. DataOps seeks to integrate data engineering, data integration, data quality, data security, and data privacy with operations. It applies principles from DevOps, Agile Development and the statistical process control, used in lean manufacturing, to improve the cycle time of extracting value from data analytics.

DevSecOps

DevSecOps is another practice that rose from DevOps that includes information technology security as a fundamental aspect in all the stages of software development.

Site reliability engineering

In 2003, Google developed site reliability engineering (SRE), an approach for releasing new features continuously into large-scale high-availability systems while maintaining high-quality end-user experience. While SRE predates the development of DevOps, they are generally viewed as being related to each other. Some aspects of DevOps have taken a similar approach.

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

Sorensen-Dice Coefficient

This method is intensively used in the project to analyse the query string by the user. This leads to the implementation of machine learning in the project, as the system could analyse the requirements of the user in a better and defined manner.

The Sørensen–Dice coefficient (see below for other names) is a statistic used to gauge the similarity of two samples. It was independently developed by the botanists Thorvald Sørensen and Lee Raymond Dice, who published in 1948 and 1945 respectively.

Formula

Sørensen's original formula was intended to be applied to discrete data. Given two sets, X and Y, it is defined as

where $|X|$ and $|Y|$ are the cardinalities of the two sets (i.e. the number of elements in each set). The Sørensen index equals twice the number of elements common to both sets divided by the sum of the number of elements in each set.

When applied to boolean data, using the definition of true positive (TP), false positive (FP), and false negative (FN), it can be written as

It is different from the Jaccard index which only counts true positives once in both the numerator and denominator. DSC is the quotient of similarity and ranges between 0 and 1. It can be viewed as a similarity measure over sets.

Similarly to the Jaccard index, the set operations can be expressed in terms of vector operations over binary vectors **a** and **b**:

which gives the same outcome over binary vectors and also gives a more general similarity metric over vectors in general terms.

For sets X and Y of keywords used in information retrieval, the coefficient may be defined as twice the shared information (intersection) over the sum of cardinalities :

When taken as a string similarity measure, the coefficient may be calculated for two strings, x and y using bigrams as follows.

where n_t is the number of character bigrams found in both strings, n_x is the number of bigrams in string x and n_y is the number of bigrams in string y. For example, to calculate the similarity between:

```
night
```

```
nacht
```

We would find the set of bigrams in each word:

```
{ni,ig,gh,ht}
```

```
{na,ac,ch,ht}
```

Each set has four elements, and the intersection of these two sets has only one element: `ht`.

Inserting these numbers into the formula, we calculate, $s = (2 \cdot 1) / (4 + 4) = 0.25$.

Applications

The Sørensen–Dice coefficient is useful for ecological community data (e.g. Looman & Campbell, 1960). The justification for its use is primarily empirical rather than theoretical (although it can be justified theoretically as the intersection of two fuzzy sets). As compared to Euclidean distance, the Sørensen distance retains sensitivity in more heterogeneous data sets and gives less weight to outliers. Recently the Dice score (and its variations, e.g. logDice taking a logarithm of it) has become popular in computer lexicography for measuring the lexical association score of two given words. It is also commonly used in

image segmentation, in particular for comparing algorithm output against reference masks in medical applications.

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.
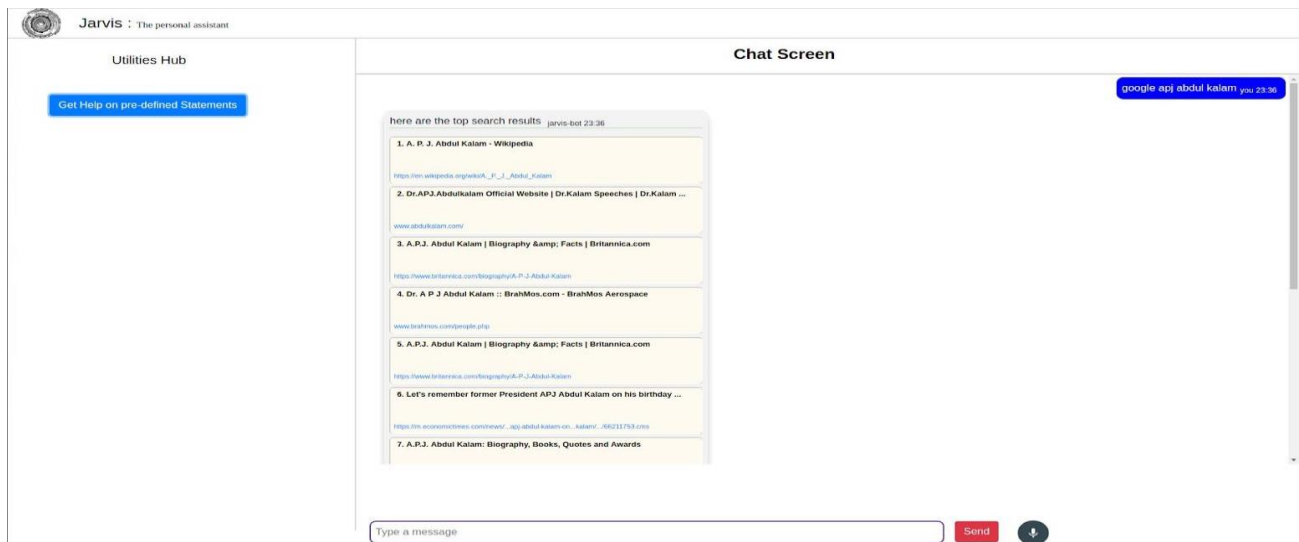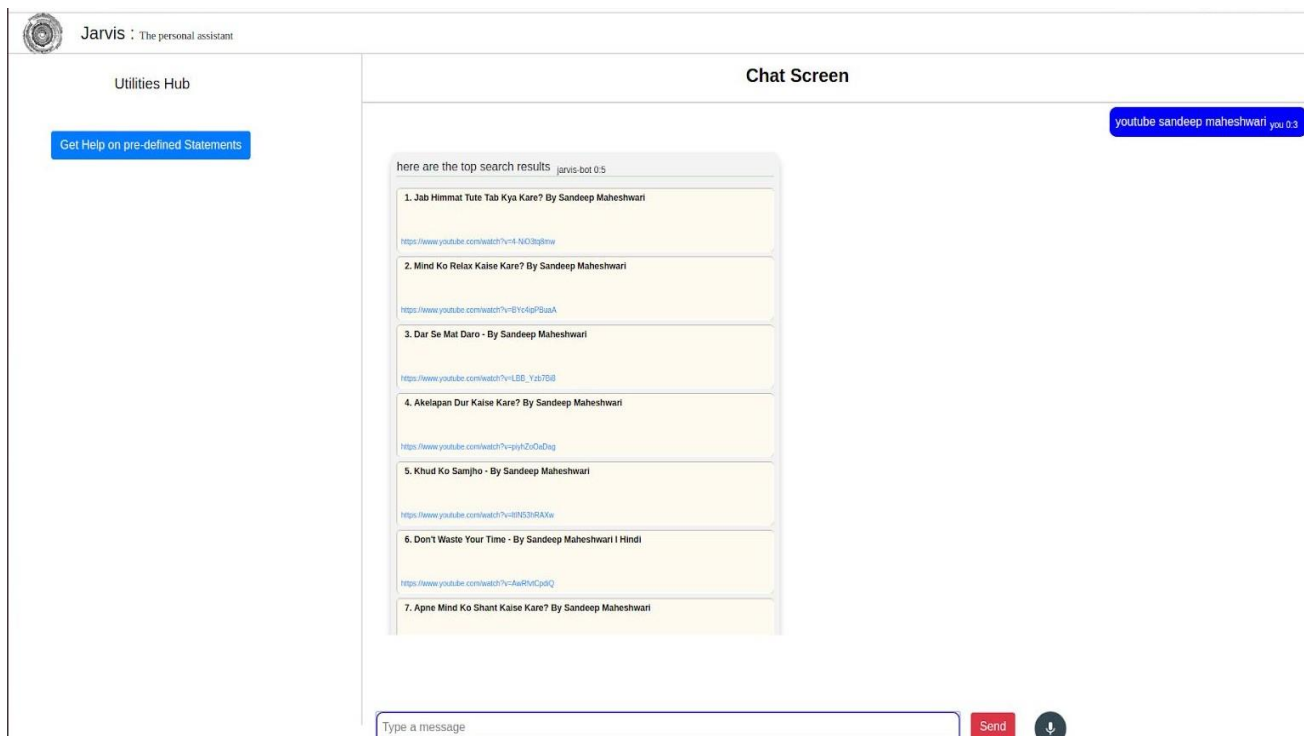
2. Accessing youtube videos

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:

youtube *<video you want to search for>*

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:

Weather <city> <state/country>
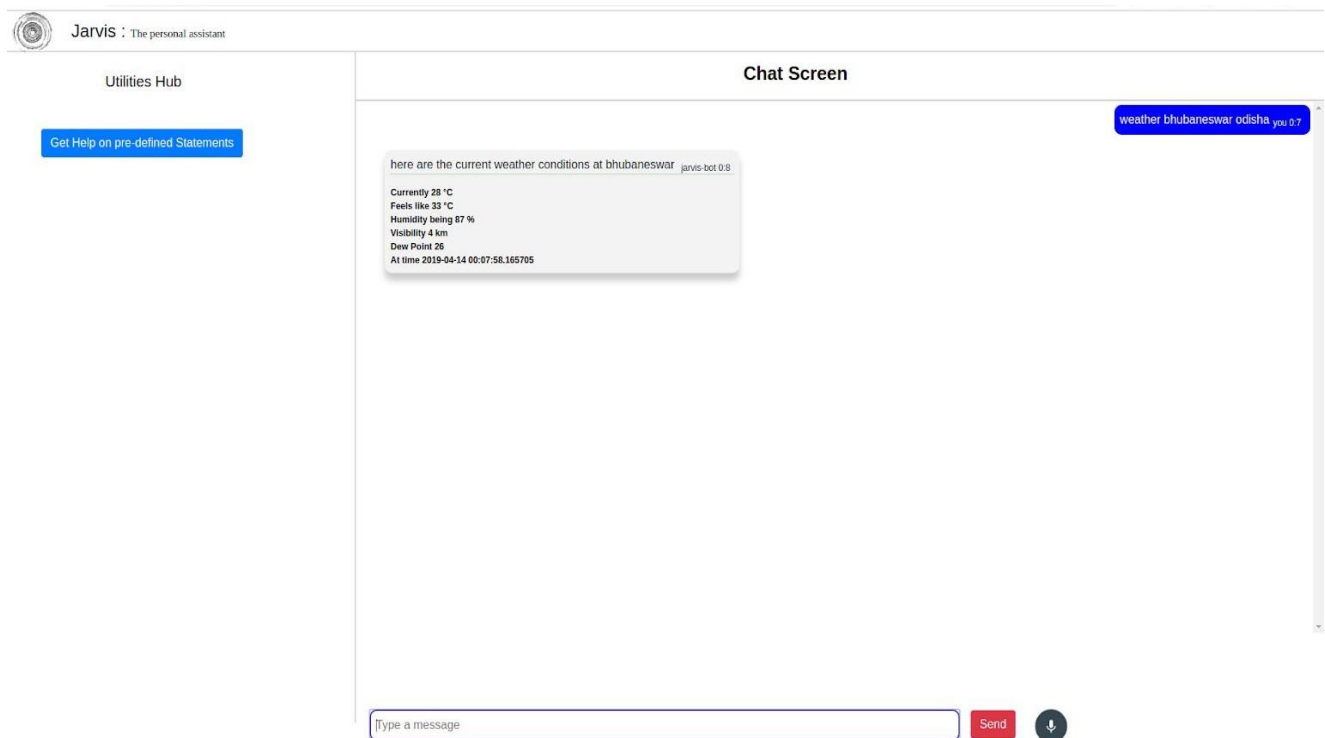
4. Retrieve images

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:

Image *<image you want to search>*
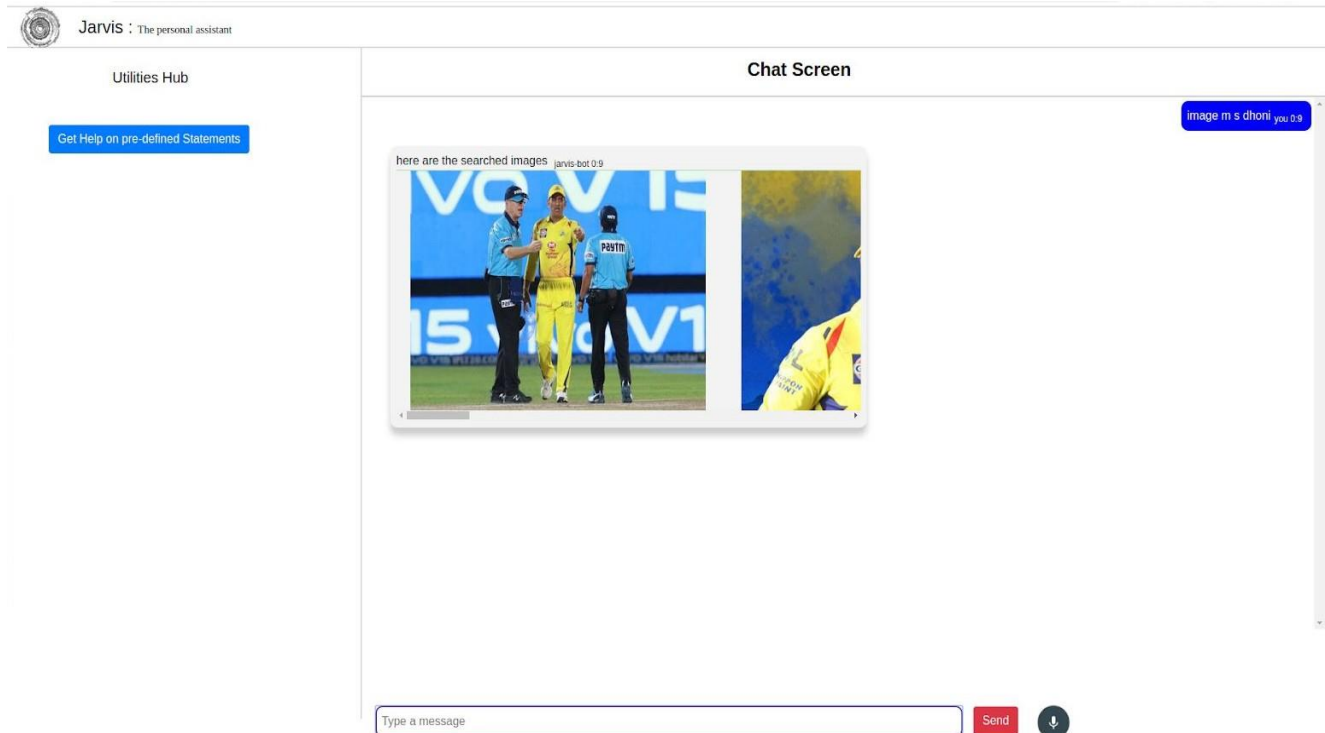
5. Dictionary meaning

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,

> ***meaning*** *<word>*
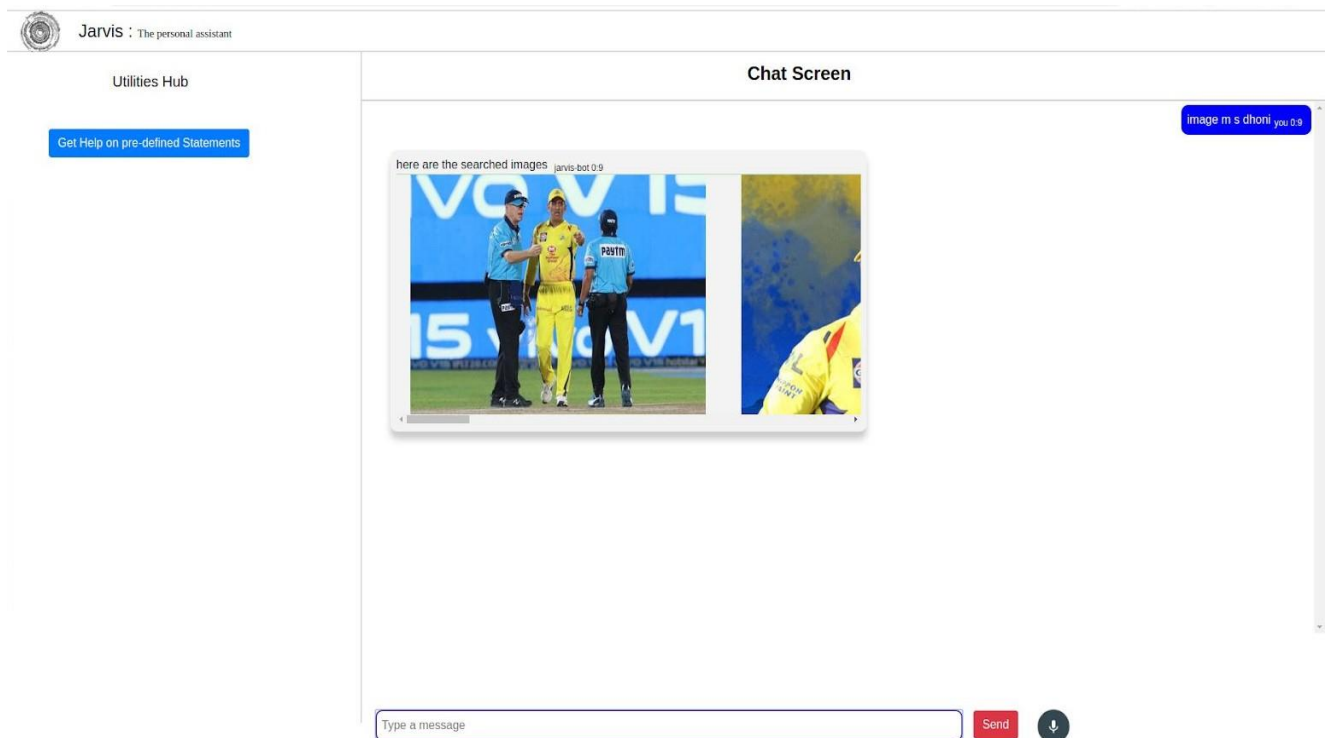
6. Medicine Details

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

you query about the symptoms while Jarvis lists various diseases one is likely to be affected along with their medicine. This is helpful for people who are quite busy with their life and find trouble visiting the doctor immediately, thus relying on the web to find the best result for short term cause.

Here we use Node JS framework along with Selenium to scrap the required data from the web and display it to the user. We have a huge database of various medicines and symptoms which helps Jarvis respond to the queries of the user at ease. The syntax to be used for querying the necessary are:

In order to get details about medicine format is,

Medicine *<medicine name>*

In order to re-track the causes of symptoms format is,

Symptoms *<disease/ailment>*

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.
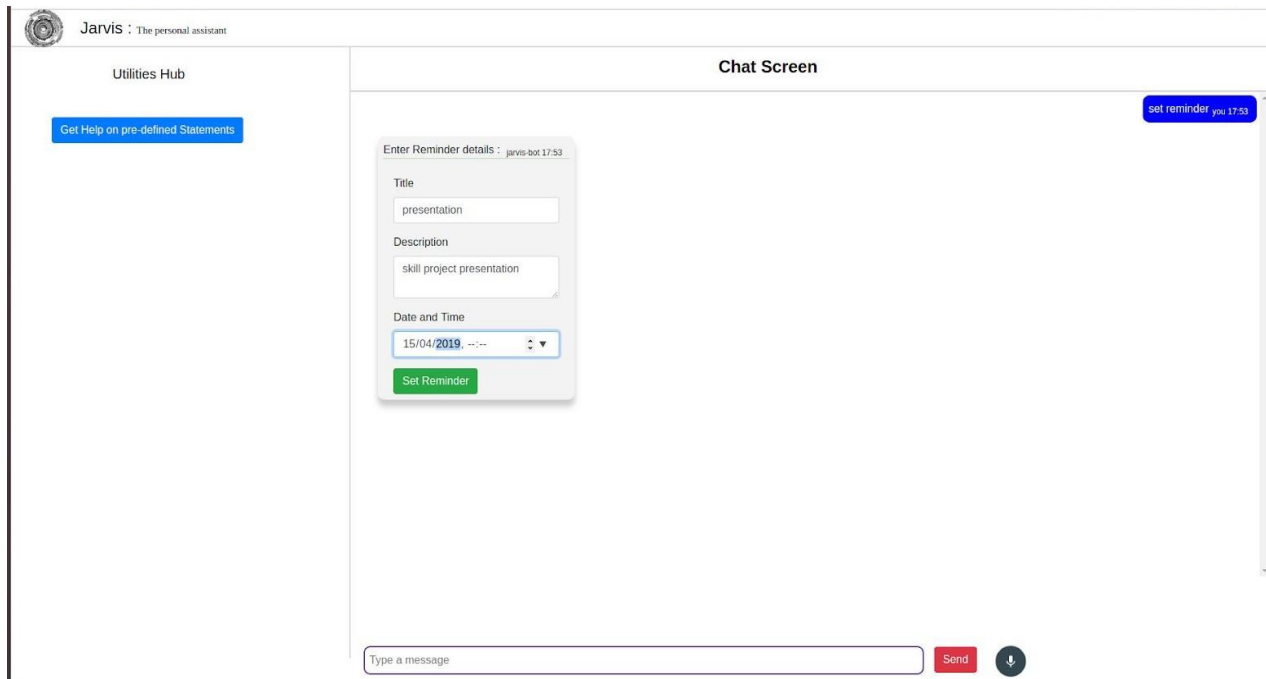
8. Sending Emails

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.
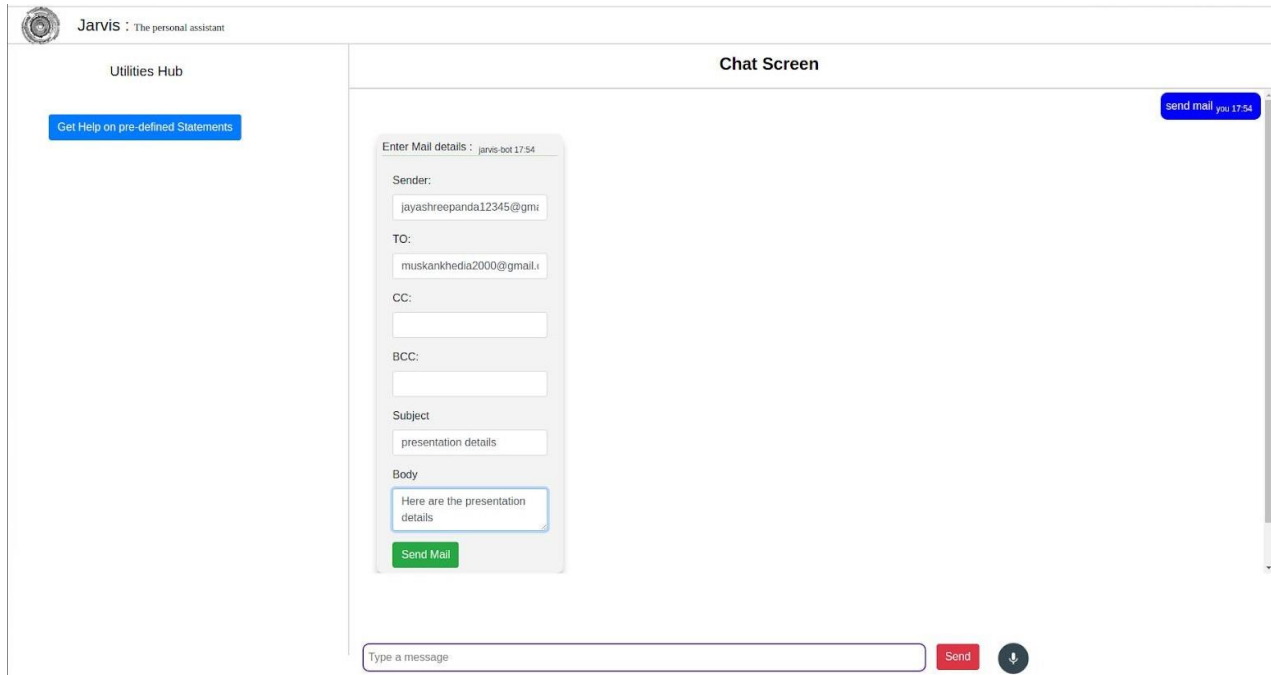
Why to use Jarvis?

1. It fulfils the lack of a virtual assistant in Linux systems.
2. It has an easy to install and use interface.
3. It accepts inputs even through voice or keyboard.
4. It automates tedious tasks like deployment, unit testing through a single command.
5. It gives live weather updates.
6. It gives advice on health.

CODE SCREENSHOTS


```python
import pyttsx3
import pywhatkit
from pywikihow import WikiHow,search_wikihow
import os
import webbrowser as web
import smtplib
import wikipedia
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
def Speak(audio):
    print(f"Karen:{audio}")
    engine.say(audio)
    engine.runAndWait()
def youtubesearch(term):
    result="https://www.youtube.com/results?search_query="+term
    web.open(result)
    Main.Speak("This is what I found for you")
    pywhatkit.playonyt(term)
def sendEmail(to, content):
    f = open('C:\\Users\\Asus\\Documents\\a.txt', 'r')
    a = f.read()
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('shradulyasingh96@gmail.com', password= a)
    server.sendmail('shradulyasingh96@gmail.com', to, content)
    server.close()
    f.close()
def GoogleSearch(query):
    query = query.replace("karen","")
    query = query.replace("what is","")
    query = query.replace("how to","")
    query = query.replace("what do you mean by","")
    query=query.replace("google search","")

    writeab = str(query)

    oooooo = open('E:\\karen\\Database\\database.txt','a')
    oooooo.write(writeab)
    oooooo.close()

    pywhatkit.search(query)

    if 'how to' in query:

        max result = 1
```



```python
import pyttsx3
import speech_recognition as sr
import os
from Feature import *
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
def Speak(audio):
    print(f"Karen:{audio}")
    engine.say(audio)
    engine.runAndWait()
def takecommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('Listening...')
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f'User said: {query}\n')
    except Exception as e:
        #print(e)
        print("Say that again please...")
        return "None"
    return query
def takecommand_hindi():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('Listening...')
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='hi')
        print(f'User said: {query}\n')
    except Exception as e:
        #print(e)
        print("Say that again please...")
        return "None"
    return query
def Taskexe():
    while True:
        query=takecommand().lower()
        if 'Youtube search' in query:
            youtubesearch(query)
        elif 'search' in query:
```

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
- Heroku CLI
- Mplayer for voice support (Text-to-Speech)
- Chromium-based browser, like Chrome, Edge
- Heroku Credentials
- Node JS with npm

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.