A PROJECT REPORT ON

Automation of Network System Using NLP and slack API

SUBMITTED TOWARDS THE
SHRI SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

VEDANG WARTIKAR (164)
RUTUJA SHITOLE(158)
ILHAAM SHAIKH(252)

Under The Guidance of

Prof. PRADNYA MEHTA

DEPARTMENT OF COMPUTER ENGINEERING
Marathwada Mitra Mandal's College of Engineering
Karve Nagar, Pune -52
Academic Year:-2019-20

CERTIFICATE

This is to certify that the Project Entitled

Automation of Network System Using NLP and slack API

Submitted by

VEDANG WARTIKAR (164)
RUTUJA SHITOLE (158)
ILHAAM SHAIKH (252)

is a bonafide work carried out by Students under the supervision of **Prof. Pradnya Mehta** and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering).

Prof. Pradnya Mehta
Internal Guide,
Dept. of Computer Engg.

Dr. H. K. Khanuja
Head,
Dept. of Computer Engg.

Dr.S.M. Deshpande
Principal,
Marathwada Mitra Mandal's College of Engineering, Karvenagar, Pune – 411052

Place : Pune
Date :

Acknowledgment

We take this to express our deep sense of gratitude towards our esteemed guide Prof. **Pradnya Mehta** for giving us this splendid opportunity to select and present this project and also providing facilities for successful completion.
We thank Dr. Harmeet Khanuja , Head, Department of Computer Engineering, for opening the doors of the department towards the realization of the project, all the staff members, for their indispensable support, priceless suggestion and for most valuable time lent as and when required. With respect and gratitude, We would like to thank all the people, who have helped us directly or indirectly.

VEDANG WARTIKAR
RUTUJA SHITOLE
ILHAAM SHAIKH
(B.E. Computer Engg.)

Abstract

Slack is a cloud-based collaboration software tool that manages user-supplied data derived from conversational text and files such as PDFs, video, and images updating real-time across all devices. Also, Slack has an aggressive deep integration strategy with other business applications enabling an ecosystem that drives users to stay connected within the Slack platform. Its main function is to streamline and automate manual and administrative tasks while supporting other activities. We intend to make a collaborative platform that lets a person plan, track, and manage the processes, content, and individuals that are connected to the network. This will be beneficial for the individuals who manage the network system. Also this will allow all types of activity alerts to be fed directly to Slack's channels, making it easier for the network manager to remain informed and communicate with others when certain events take place in centralized system. The idealistic notion is to ease the work of the network controller and optimizing the system by automating its system seemingly improving its accuracy. The parsing for user level queries will be done through Natural Language Processing (NLP). These queries must be interpreted on respective individual PCs and sent it onto the job queue for automating the task. The detailed result of the task performed will be reverted back to the user device. Contributors' primary user flow is to submit data through a channel using simple natural language which will be interpreted to respective stream using NLP.

Keywords

- RTM - Real Time Messaging
- API - Application Programming Interface
- MST - Media Synchronicity Theory
- NLP - Natural Language Processing
- ESN - Enterprise Social Networking

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The main objective behind this project is to provide ease in team collaborations, and reduce human efforts. Our project will bring new level of comfort specially when there are number of computers connected to a server. For example, lab assistants will not have to manually shutdown each computers, rather they will just type on slack app to shut down all PCs of a particular lab and it will be done.
This provides a Different, Better and Simpler approach to deal with automating computer related tasks in a particular organization. This project provides a perfect place for any organization or a team to chat, discuss, maintain records in forums, schedule activities, and as well as chat with our Bot, which is responsible to automate each task, right from shutting down , restarting, monitoring and troubleshooting, to file handling, installations, etc. Our project will be beneficial in each and every aspect in an organization.

1.2 Problem Definition

To build a centralized Slack API based application to manage automation tasks on interconnected PCs. The parsing for user level queries will be done through Natural Language Processing(NLP). These queries must be interpreted on respective individual PCs and sent it onto the job queue for automating the task. The detailed result of the task performed will be reverted back to the user device

Chapter 2

Literature Survey

2.1 Can slack curb slacking?: Examining the importance of team communication in reducing social loafing

Social loafing is one of the biggest challenges in group projects. Previous research has pointed to the importance of communication quality in the reduction of social loafing. This study examines how the use of a team communication application, Slack, influences communication quality and social loafing. The results of the preliminary data indicate that the use of Slack significantly improved communication accuracy and appropriateness. Subsequently, the use of Slack significantly reduced social loafing. Further research with a larger sample and experimental research design should be conducted to further explore these result.[6]

2.1.1 Technique used-

IRB approval was obtained from the author's university for this study. The participants in this study included 19 undergraduate students who used Slack to communicate and 445 students who did not use Slack to communicate (the 445 students came from a previous research study conducted by the author). The 19 students were recruited from in an advanced technical communication course in the spring semester of 2016. The participants completed a team project where they were asked to research, design, and build a knowledge base on any technical topic of their choice.

2.1.2 Result -

Students found Slack to be helpful for team communication. For instance, 89% of students believed that Slack was helpful because of it allowed users to store and share files. Students were slightly less enthusiastic about Slack's direct messaging functionality as 64% of students reported the functionality was somewhat or strongly useful. Students rated Slack's potential for aiding in project coordination similarly with 58% of students somewhat or strongly agreeing that it was helpful for coordinating project events and meetings. On the other hand, 68% of students did not think Slack could completely replace email or other forms of communication.

2.1.3 Advantages-

- This paper highlights the need of proper business communication
- Gives a solid proof of usefulness of the slack platform by carrying out the proper survey
- Provides evidence of increased effectiveness in the team project

2.1.4 Disdvantages-

- Fails to cover all the features provided by Slack
- Use of Slack APIs not taken into considerations.
- Focuses on communication benefits of slack rather than technical features

2.2 Bringing Automation to the Classroom: A ChatOps-Based Approach

In this paper they present the design and implementation of a chatbot-based virtual assistant called LTKABot. Its main function is to streamline and to automate manual and administrative tasks while supporting other course-related activities. It differs from other recent approaches in that it is based on the ChatOps paradigm instead of on some AIbased schemes. LTKA-Bot introduces a case of automation and demonstrates its potentials in the area of higher education which is steadily transformed to cope with technological

progresses and administrative policy dynamics including accreditation. International accreditation body such as ABET requires fulllment of certain criteria which in turn also requires appropriate course design and quite a lot of document works. LTKA-Bot borrows the idea of automation which is very common in the modern tech companies to cope with such challenges. It facilitates more efcient course-related activities while satisfying all document requirement with minimal effort.[7]

2.2.1 Technique used-

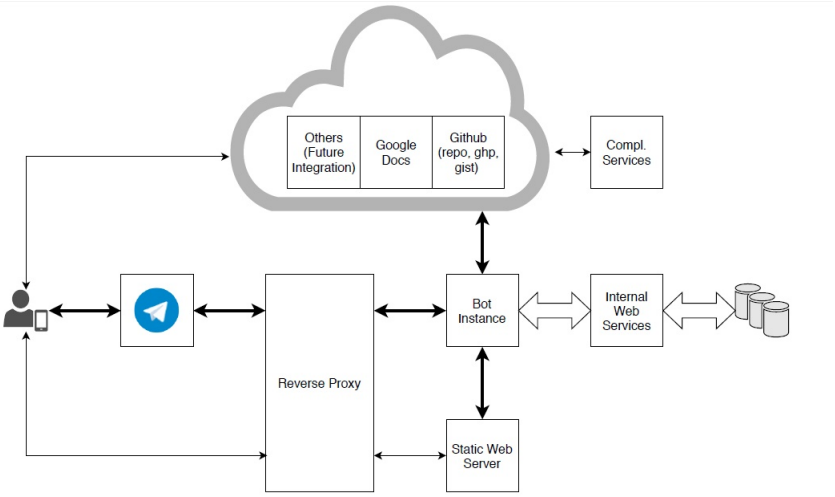

Figure 2.1: Architecture of chat-bot system

The system consists of several flexible services glued together with the bot instance as the central component. This nonmonolith approach offers flexibility to scale across multiple base infrastructure or to adapt to more complex features/ requirement in the future. The bot instance is based on Hubot

2.2.2 Result -

A beta version of LTKA-Bot has been operating on the Telegram group of LTKA class in H1 2018, following a successful prototype test in late 2017. LTKA stands for Layanan Tersambung dan Komputasi Awan (Connected Services and Cloud Computing). It is a course offered to final year students at the School of Electrical Engineering and Informatics, ITB. Content of LTKA course covers many aspects in modern connected services, so that it

fits perfectly for an application of advanced, state of the art tools such as LTKA-Bot. Many features implemented in the bot's Support module are intended for demonstrations and experience-based learning purposes.

2.2.3 Advantages-

- This paper highlights the need of automation
- Gives a solid proof of how BOTS can be used for better efficiency
- Covers almost every aspect of educational system

2.2.4 Disdvantages-

- The paper proposes a system that is built from scratch - while slack could have been used for using direct UI and it's in-built features

2.3 Faheem : Explaining URLs to people using a Slack bot

Online safety regularly depends on users' ability to know ei- ther where a URL is likely to lead or identify when they are on a site other than they expect. Unfortunately, the combination of low URL reading ability in the general population and the use of hard-to-detect approaches like look-alike letters makes the reading of URLs quite challenging for people. We design a Slack bot, named Faheem, which assists users in identifying potentially fraudulent URLs while also teaching them about URL reading and common malicious tactics. In this work, we describe the design of the bot and provide an initial evaluation. We find that Faheem does a good job of interactively help- ing users identify issues with URLs, but Faheem users show minimal retention of knowledge when they lose access to the tool. [1]

2.3.1 Technique used-

Primary design objective is to create an interactive chat bot which helps average internet users correctly read URLs and identify phishing URLs. In order to accomplish this goal they focus on two features of the bot: 1. Parsing the URLs and identifying common malicious behaviours focusing primarily on the domain issues. 2. Presenting the results to the user in a clear and easy to understand manner.

2.3.2 Result -

To test Faheem they compared it with URL Explainer, a simplistic web page which parses a URL for a user but focuses on a factual clear representation of the URL contents rather than helping the user identify common issues. they find that while using both tools, Faheem is better at helping a user identify URLs which have a destination other than where the user wishes to go. Additionally, they also saw some minimal learning effects with Faheem users showing an improved ability to identify phishing URLs after using the tool.

2.3.3 Advantages-

- Successfully overcomes the drawbacks of URL-explainer and achieves the task of explaining URL in a much effiecient way
- Uses NodeJs programming integrated with Slack, thereby demonstrating the vast applications of slack
- Covers almost every aspect related with URLs

2.3.4 Disdvantages-

- The URLs entered by the users are also saved in their system, and privacy policy is not provided

2.4 Slack Me If You Can! Using Enterprise Social Networking Tools in Virtual Agile Teams

Virtual teams rely on enterprise social networking tools such as Slack to collaborate efficiently. While such tools contribute to making the communication more synchronous and support distributed agile development, there are several challenges such as how to interact with each other and how to balance the communication with other types of communication mechanisms such as meetings, e-mail, and phone. In this paper, they describe and discuss how a distributed global project used Slack. Some of the challenges we identified were related to language problems, using too much direct messaging when communicating, and unbalanced activity (33% of the users accounted for

86% of the messages). The positive aspects of using the tool were increased transparency, team awareness, and informal communication. Further, Slack facilitates problem-focused communication which is essential for agile teams. Our study stresses the importance of reflecting on how virtual teams use communication tools, and they suggest that teams decide on guidelines on how to use the tools to improve their coordination. [10]

2.4.1 Technique used-

- Case Study of GeoSoft software development company
- Studied 30 virtual agile teams, their task-tracker and also their project backlogs
- Data was gathered and collected from a survey form in question-answer format

2.4.2 Result -

Co-ordination of slack can be found as VERTICAL when team-members are communicating with team-manager and HORIZONTAL when communicating with other team members.

2.4.3 Advantages-

- This paper highlights the need of proper business communication
- Gives a solid proof of usefulness of the slack platform by carrying out the proper survey
- Provides evidence of increased effectiveness in the team project

2.4.4 Disdvantages-

- Fails to cover all the features provided by Slack
- Use of Slack APIs not taken into considerations.
- Focuses on communication benefits of slack rather than technical features

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Project Scope

This application mainly consists of automation script using Slack API libraries which can parse human understandable queries into machine understandable queries and execute the desired task on multiple target PCs. This will make it possible for every network administrator to conveniently use the application. It can be utilized in client-server architecture. It can also collaborate many computer labs connected to a single server which will help in maintaining data security and integrity.

3.1.2 User Classes and Characteristics

- Admin
  - Register a team member.
  - View lab schedule.
- Assistant
  - Can edit member's details.
  - Creates tasks and schedules labs.
  - Administers a whole network.

- User
  - Read notifications.
  - Get alerts..
- Departmental Authorities
  - Analyzes log reports.
  - Arranges monthly meets for maintainance.

3.1.3 Assumptions and Dependencies

- Every lab assistant has a smartphone.
- Every assistant has a phone number by which they will recieve text notifications.

3.2 Functional Requirements

3.2.1 Automatic schedule generation

- The formatting of hard disks Installations / Uninstallation

3.2.2 Automatic shutdown process

- Shutdown/Restart is done automatically while scheduling through slack apps.

3.2.3 History

- Admins can view all the slack history.

3.2.4 Report generation

- System generates overall activity report of each machine.
- Spots malicious activities and sends alerts.

3.3 External Interface Requirements

3.3.1 User Interfaces

- For System Administrator,
- For Each team member.

3.4 Nonfunctional Requirements

- Data availability
- Response handling

3.4.1 Performance Requirements

- Fast data operations
- Efficiency
- Accuracy

3.4.2 Security Requirements

- Data encryption
- Data privacy

3.4.3 Software Quality Attributes

- Correctness
- Reliability
- Adequacy

3.5 Analysis Models: SDLC model to be applied

In order to successfully carry out this project, we have planned to use the incremental model.

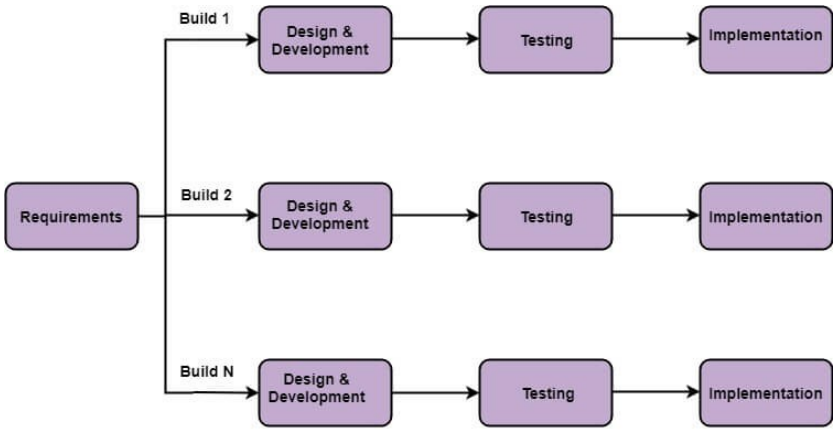

Figure 3.1: SDLC model to be followed

The system is put into production when the first increment is delivered (Connection of a slack and a CPU machine in our case). The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the end user, there is plan development for the next increment. So in this way, we first established a connection between slack and a CPU machine using Python Slack RTM API, and then once we successfully got the remote access, we started adding functionalaties like shutting down the system, restarting, installing software packages, and sending alert notifications. Once that was done for One System, same methodology was applied to a set of connected machines in a network by using Anible and Docker for deployment.

Chapter 4

System Design

4.1 System Architecture

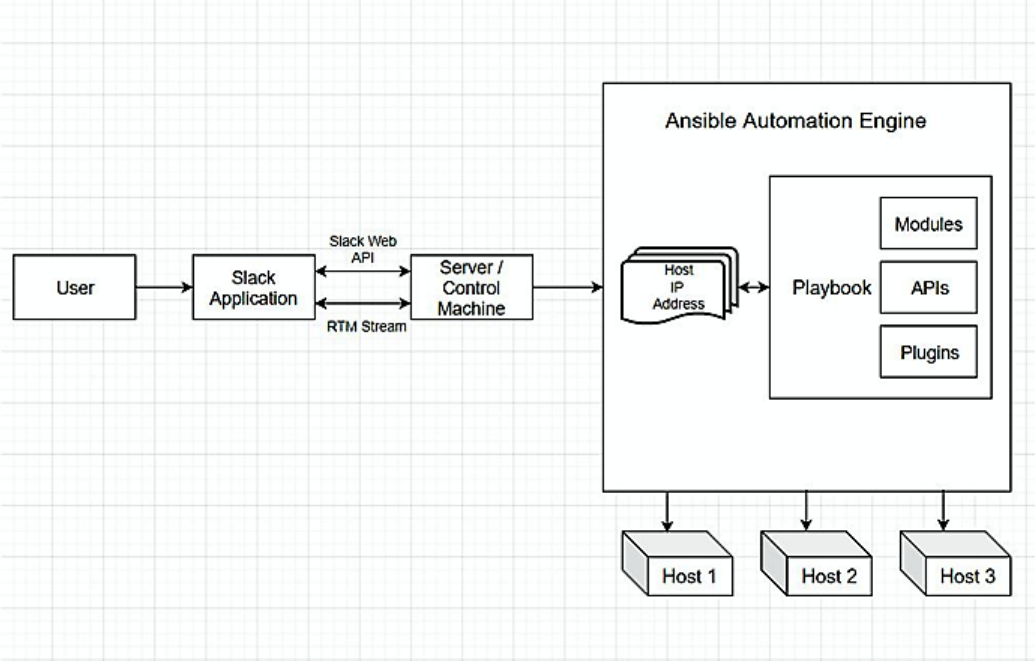

Figure 4.1: System design

The proposed architecture of the system primarily comprises of three main components:
1. User - interacting through Slack Application
2. Server / Control Machine enabled with Ansible
3. Host PCs

The user must have Slack Application installed on their device. A dedicated slack channel must be chosen that will behave as the job queue for the automation tasks. The user will interact with the Server / Control Machine through this channel. User level queries will be processed with the help of NLP and Machine Learning techniques. The slack automation scripts will be executed on multiple network systems using Docker and RedHat Ansible. Docker will contain docker images which will help in maintaining and isolating the configurations of the application script libraries from the individual machine's configuration. Ansible Playbooks can be programmed to connect multiple PCs on a network. A single host machine can interact with different guest machines and execute automation scripts through SSH scripting. The docker image and Ansible Playbooks can be visualized as:

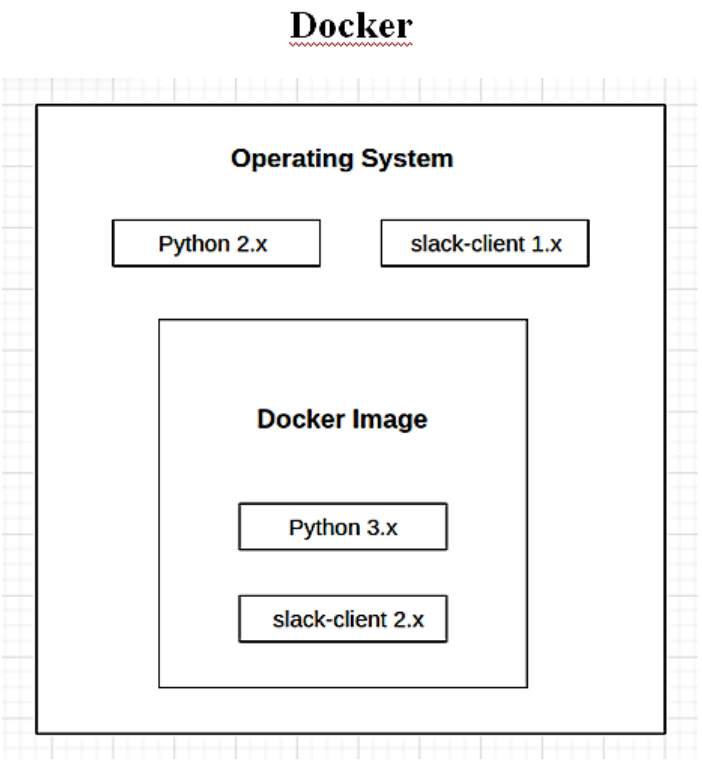

Figure 4.2: Docker

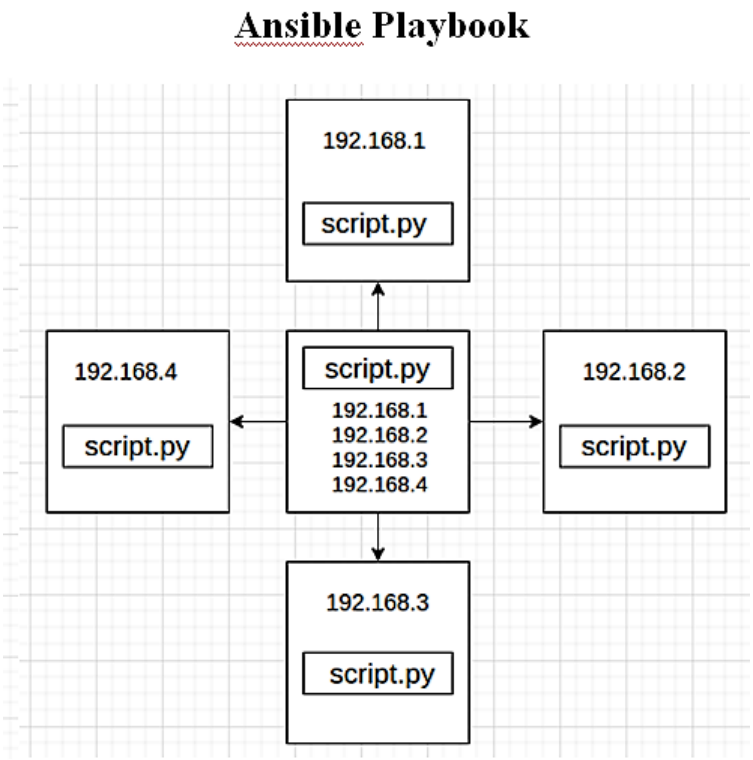

Figure 4.3: Ansible-Playbook

4.2 Docker

4.2.1 Why Docker?

Suppose we have to ship the luggage from one place to another. Luggage may be of different shape, different size, different weight, etc. Some prefer truck, or some may prefer a big trolly. But if there are MANY items to be shipped, we can use the CONTAINER.

4.2.2 Components of Docker

Server: It is the docker daemon called dockerd. It can create and manage docker images. Containers, networks, etc.
Rest API: It is used to instruct docker daemon what to do.
Command Line Interface (CLI): It is a client which is used to enter docker commands. [4]

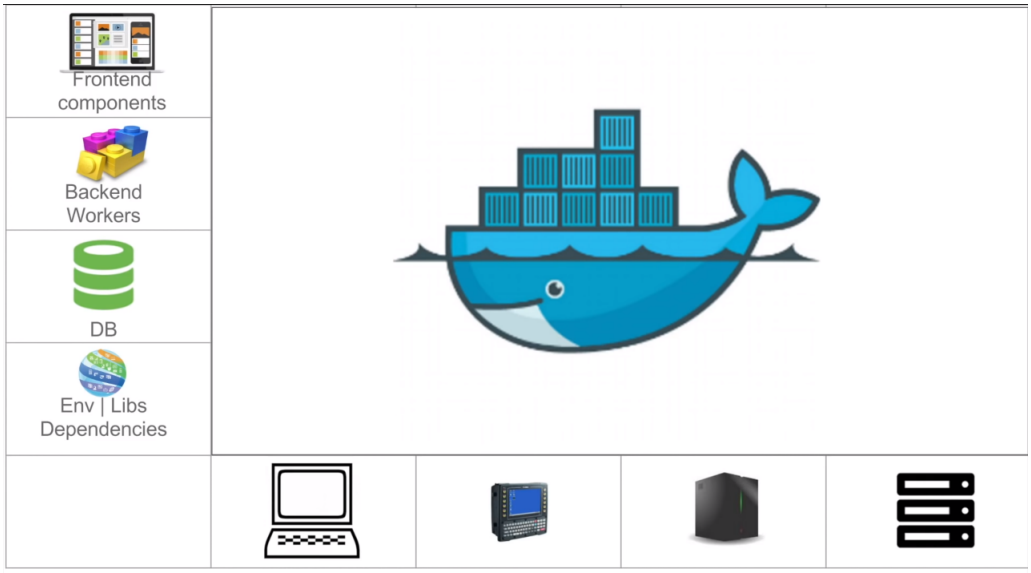

Figure 4.4: Bringing everything into one container

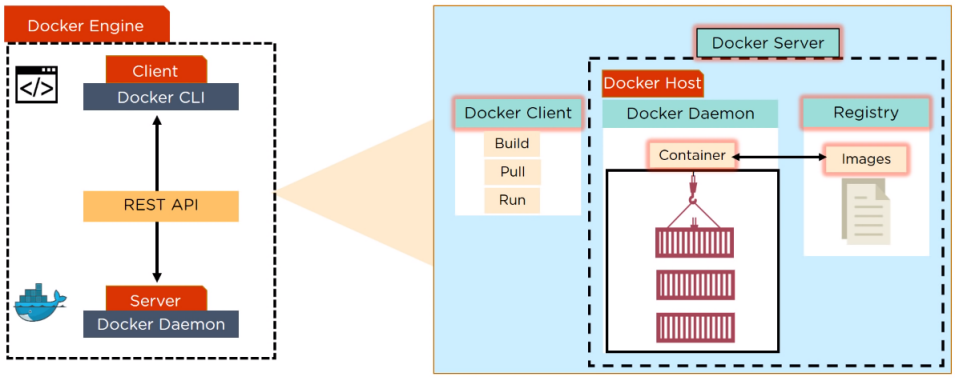

Figure 4.5: Componenets of Docker

4.2.3 Working of Docker

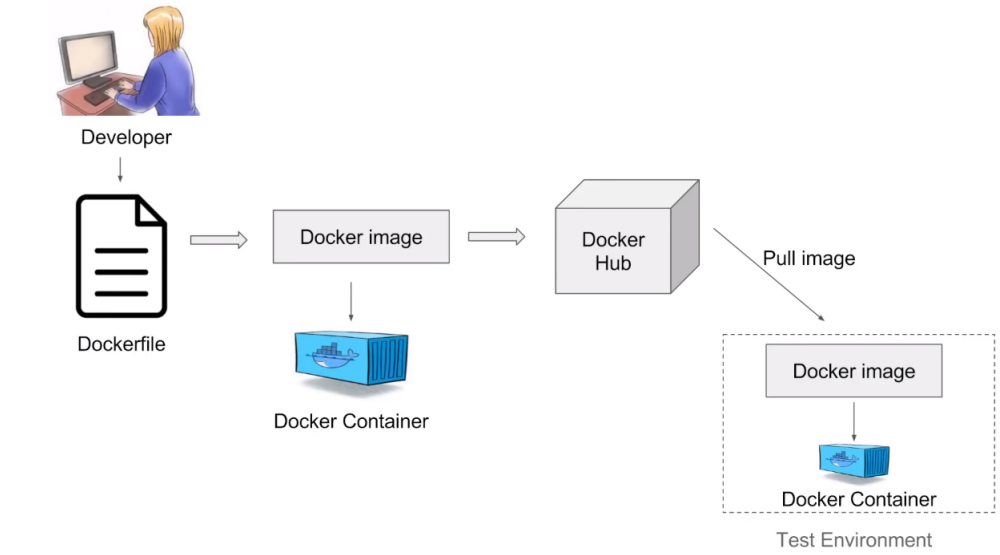

Figure 4.6: Working of docker

4.3 Ansible

4.3.1 Ansible Architecture

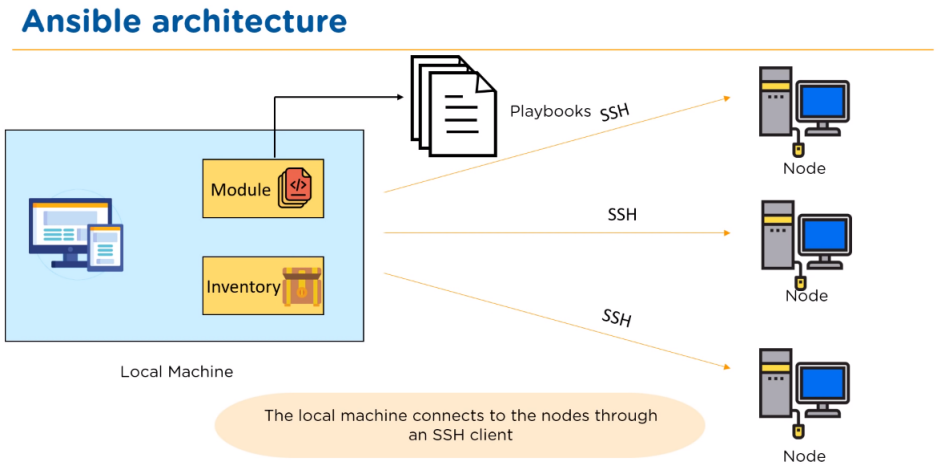

Figure 4.7: Ansible Architecture

4.3.2 Ansible Playbook

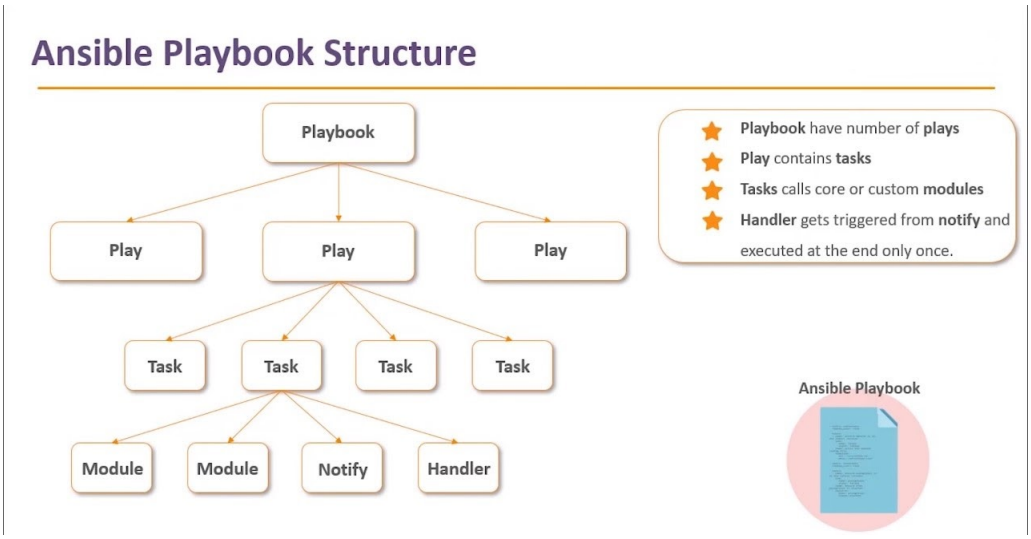

Figure 4.8: Ansible Playbook

4.3.3 Playbook Example

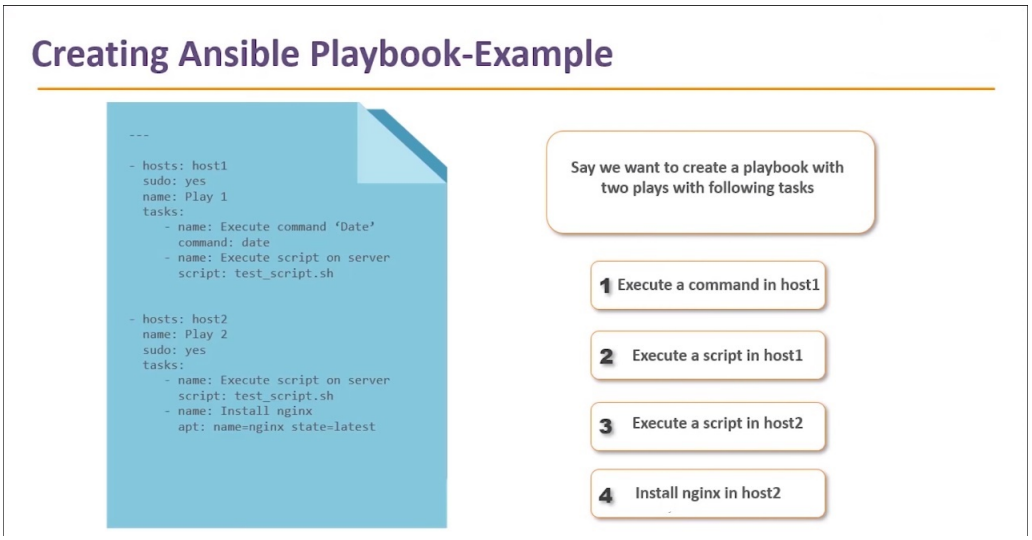

Figure 4.9: Ansible Playbook Example

Ansible is a radically simple IT automation engine that automates

cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.
Designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time.
It uses no agents and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.
Playbooks are the files where Ansible code is written. Playbooks are written in YAML format. YAML stands for Yet Another Markup Language. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks.
Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible. [3]

4.4 DFD diagrams

4.4.1 DFD level 0

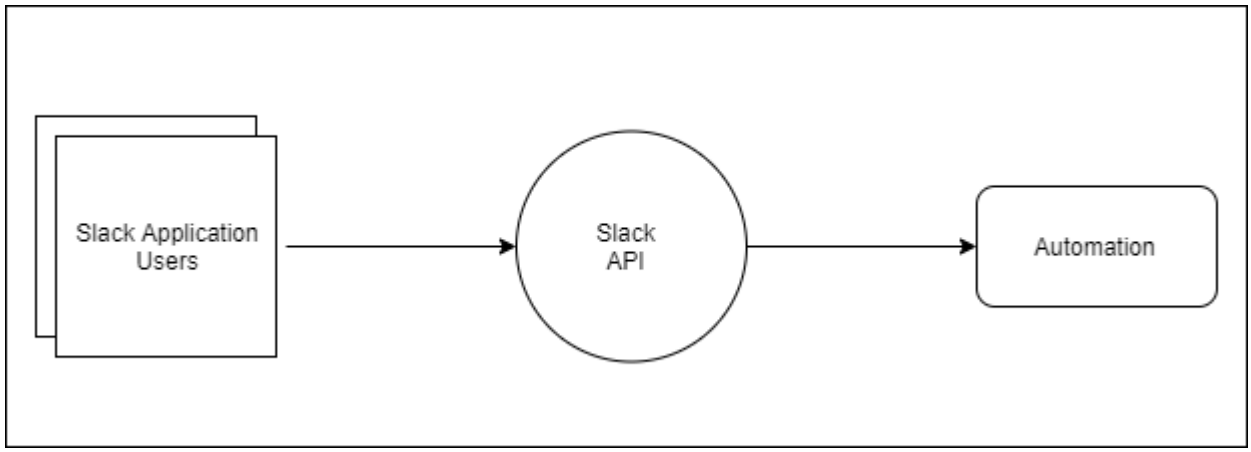

Figure 4.10: DFD level 0

4.4.2 DFD level 1

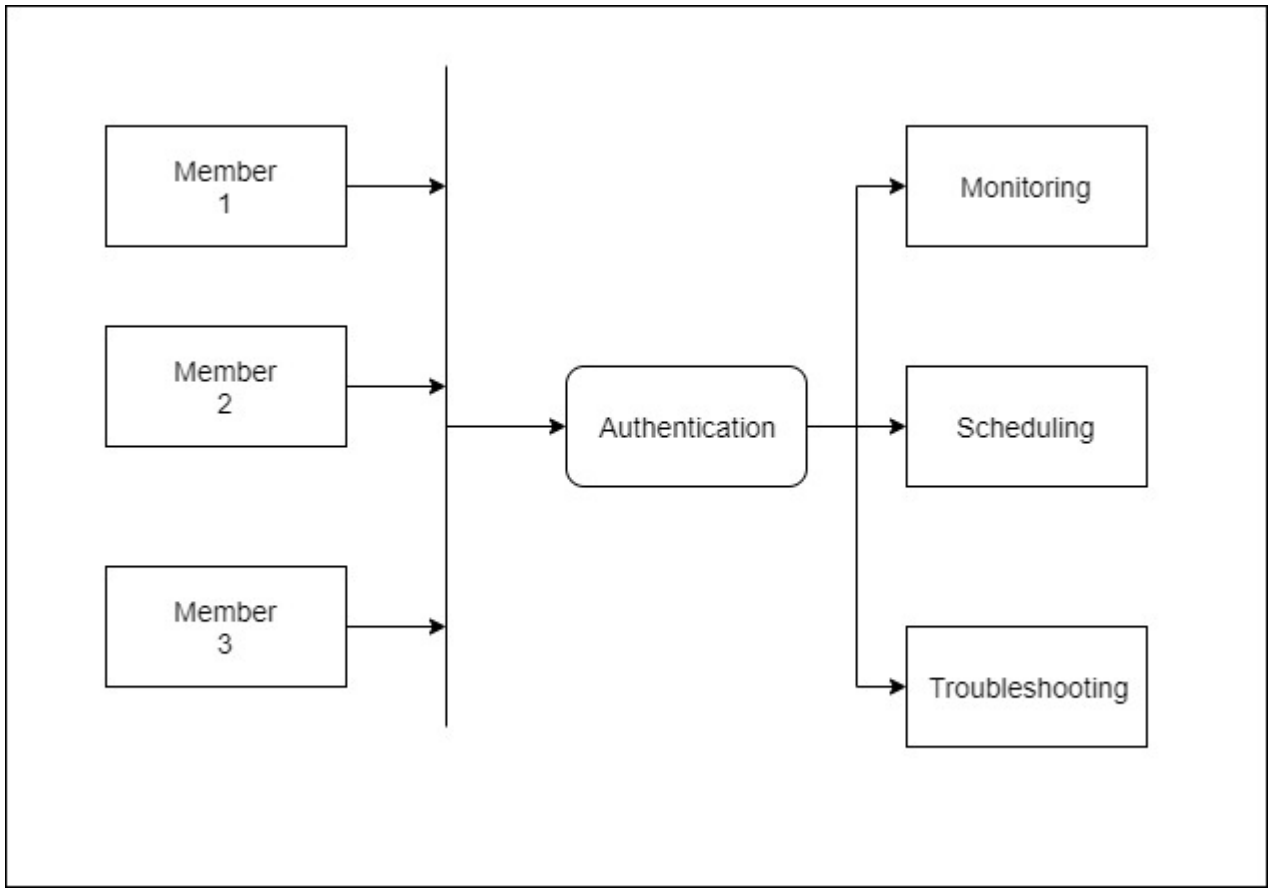

Figure 4.11: DFD level 1

4.4.3 DFD level 2

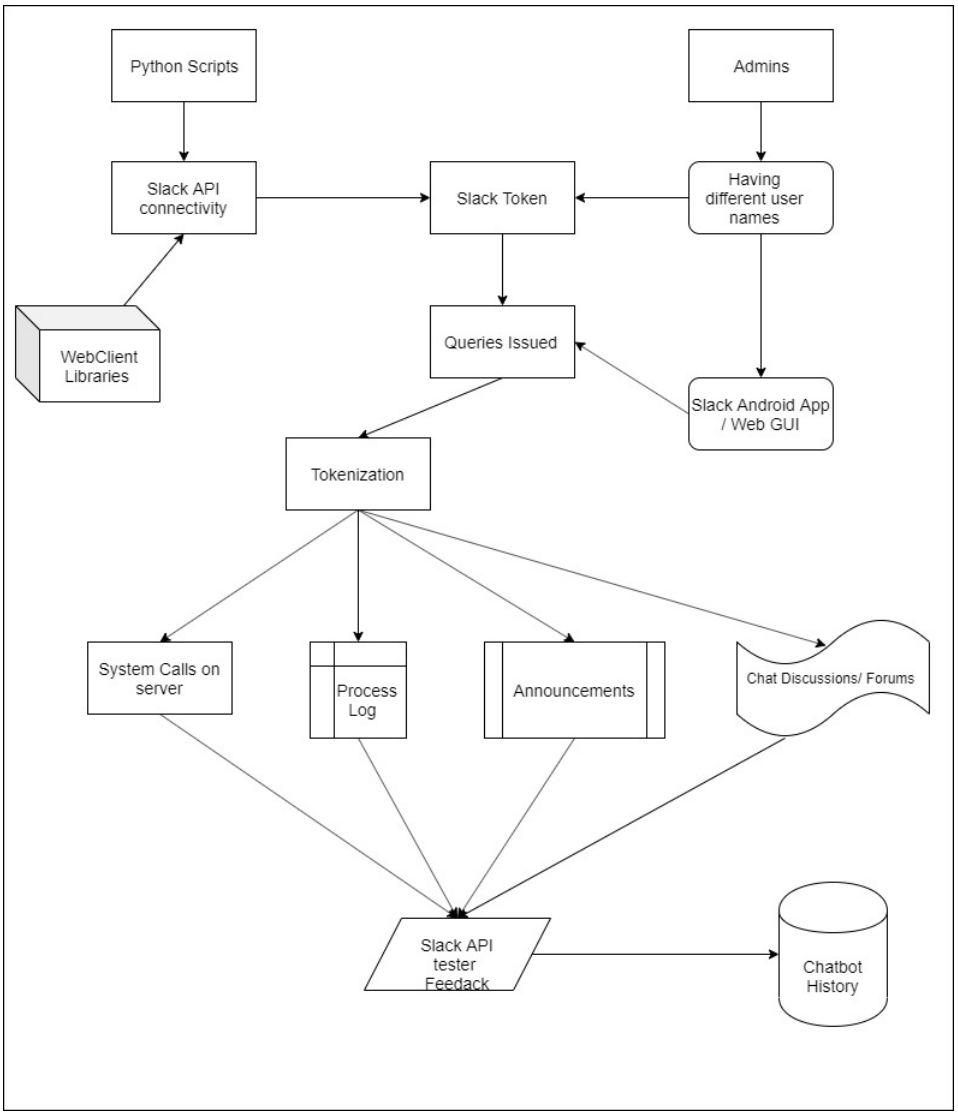

Figure 4.12: DFD level 2

4.5 UML diagrams

4.5.1 Use-Case Diagram

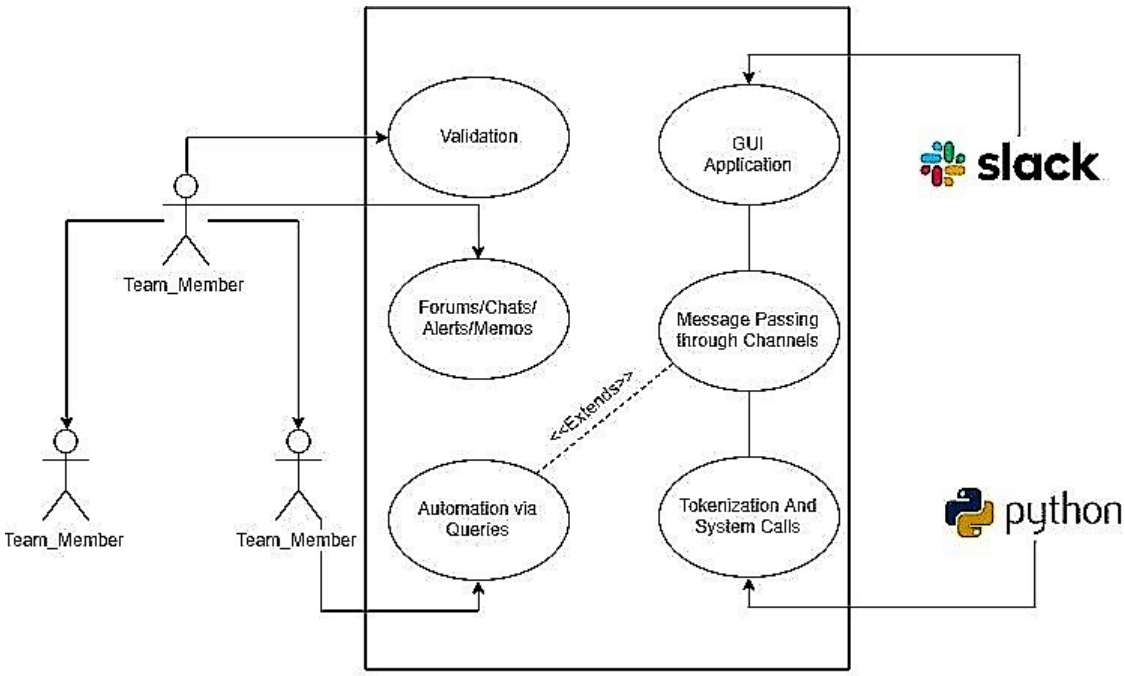

Figure 4.13: Use-Case Diagram

4.5.2 Activity Diagram

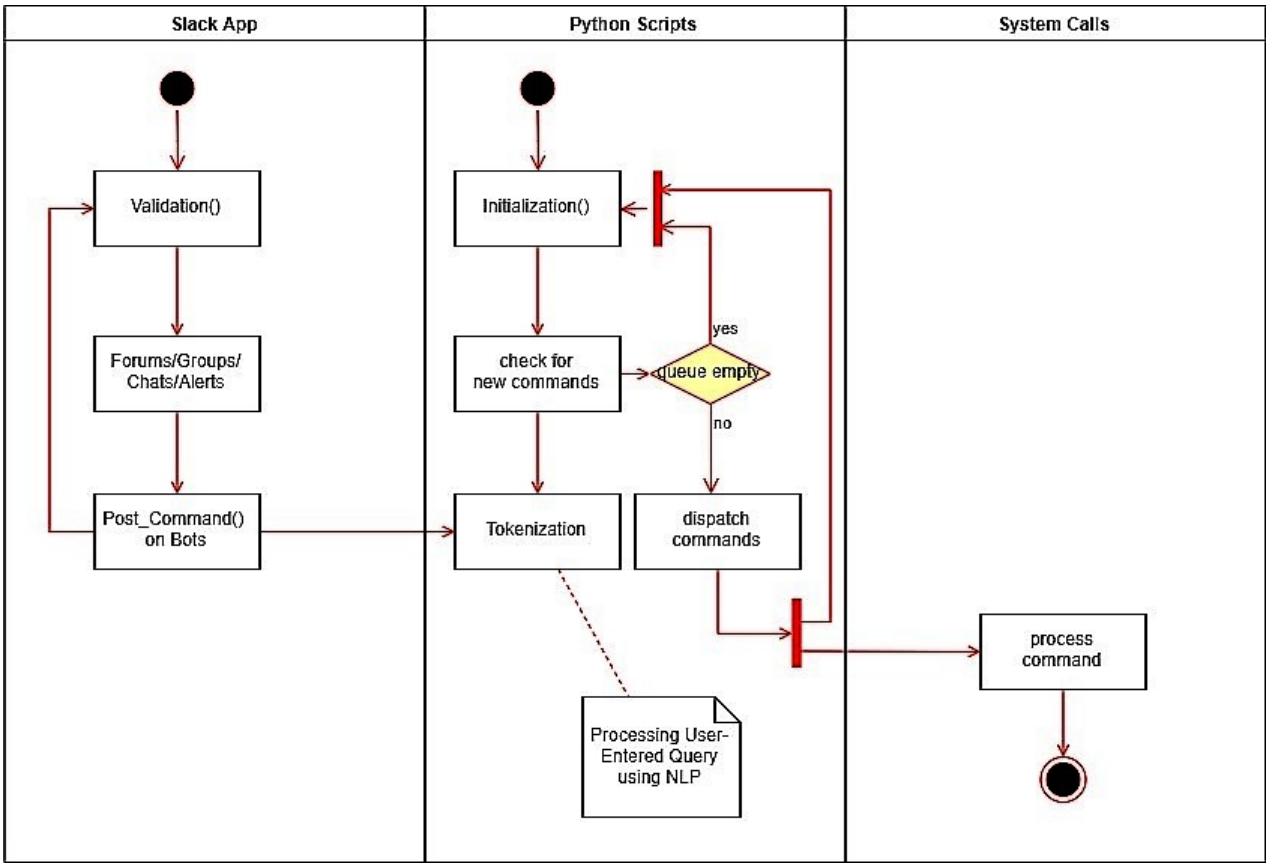

Figure 4.14: Activity Diagram

4.5.3 Sequence Diagram

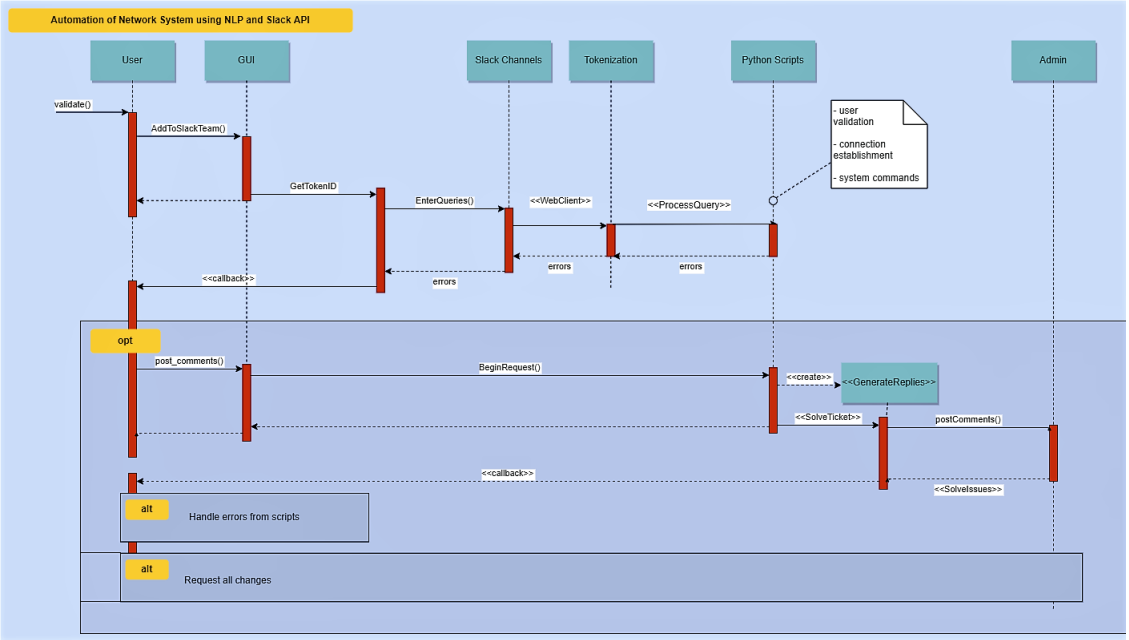

Figure 4.15: Sequence Diagram

Chapter 5

Project Plan

5.1 Project Estimate

5.1.1 Reconciled Estimates

The cost of the project will be the cost of work that is put in COCOMO model. Number of code lines:-
We estimate our project will have 1000 Delivered Source instructions.

Efforts = a*loc b

Duration = c*effort d
Staffing = effort/duration
Where, E is the effort applied in person-months, D is the development time in chronological months, KLOC is the estimated number of delivered lines of code for the project (express in thousands).
Effort: The Effort is calculated by formula. E=2.4*(KLOC)Î1.05 E=2.4*(1)Î1.05 E=2.5 appx

Time Estimates - 10 to 11 months

5.1.2 Project Resources

Stake holders: 3 people in team, 2 guides
Development Platform: Slack
Technologies: Python , Docker , Ansible

5.2 Risk Management

5.2.1 Risk Identification

- Have top software and customer managers formally committed to support the project? Yes
- Are end-users (lab assistants in our case) enthusiastically committed to the project and the system/product to be built? Yes
- Are requirements fully understood by the software engineering team? Yes
- Do end-users have realistic expectations? Yes
- Are project requirements stable? Yes
- Is the number of people on the project team adequate to do the job (3 in our case)? Yes
- Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built? Yes

5.2.2 Risk Analysis

Risk analysis is the systematic study of uncertainties and risks we encounter in business, engineering, public policy, and many other areas. Risk analysts seek to identify the risks faced by an institution or business unit, understand how and when they arise, and estimate the impact (financial or otherwise) of adverse outcomes. Risk managers start with risk analysis, then seek to take actions that will mitigate or hedge these risks. Various risk identified for our project are :

			Impact		
ID	Risk Description	Probability	Schedule	Quality	Overall
1	System Failure	Low	Low	Medium	Medium
2	Database crashed	Low	Low	High	High
3	Wrong Output	Low	Low	High	High

Table 5.1: Risk table

Probability	Value	Description
High	Probability of occurrence is	>75%
Medium	Probability of occurrence is	26 75%
Low	Probability of occurrence is	<25%

Table 5.2: Risk Probability definitions

Impact	Value	Description
High	>10%	Schedule impact or Unacceptable quality
Medium	5 10%	Schedule impact or Some parts of the project have low quality
Low	<5%	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

Table 5.3: Risk Impact definitions

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Risk mitigation planning is the process of developing options and actions to enhance opportunities and reduce threats to project objectives. Risk mitigation implementation is the process of executing risk miti- gation actions. Risk mitigation progress monitoring includes tracking identified risks, identifying new risks, and evaluating risk process effec- tiveness throughout the project. Risks can be monitored on a continuous basis to check if any change is made. New risks can be identified through the constant monitoring and assessing mechanisms. Mitigation for identified Risks are:

Risk ID	1
Risk Description	System Failure
Category	Requirements
Source	identified during early development and testing
Probability	Low
Impact	High
Response	Accept
Strategy	Better testing will solve this
Risk Status	Occurred

Table 5.4: Risk ID 1

Risk ID	2
Risk Description	Server not responding
Category	Project cost risk
Source	Excess load on server
Probability	Low
Impact	High
Response	Mitigate
Strategy	Check and recovery
Risk Status	Identified

Table 5.5: Risk ID 2

Risk ID	3
Risk Description	Wrong Output
Category	Testing
Source	This was identified during early development and testing
Probability	Low
Impact	High
Response	Mitigate
Strategy	Logical error can result in execution of some other command
Risk Status	Occurred

Table 5.6: Risk ID 3

5.3 Project Schedule

5.3.1 Project Task Set

- Task 1: Research: Searching for IEEE and other publication papers related to Slack
- Task 2: Software Requirement specification. For understanding customer requirements and analyzing objectives to be achieved.
- Task 3: Synopsis.
- Task 4: Designing of system architecture, DFD, gathering slack APIs for executions
- Task 6: Implementation of python code.
- Task 7: Testing of system by using different test cases.
- Task 8: Report

Month	Task	Date of Execution
July	Paper Selection	12/07/2019
July	Problem Formation	17/07/2019
August	Literature Survey	27/08/2019
September	Synopsis Submission	01/09/2019
October	Project GUI Design	26/10/2019
November	Project Coding	18/11/2019
December	Suggested Modification	09/12/2019
January	Project Coding	03/01/2020
February	Suggested Modification	25/02/2020
March	Project Testing	15/02/2020
April	Report Generation	25/04/2020

Table 5.7: Project Task Set

Schedule of the project:
Start Date: 25th June 2019
End Date: April 2020
Duration: Approx 11 Months

5.3.2 Task Network

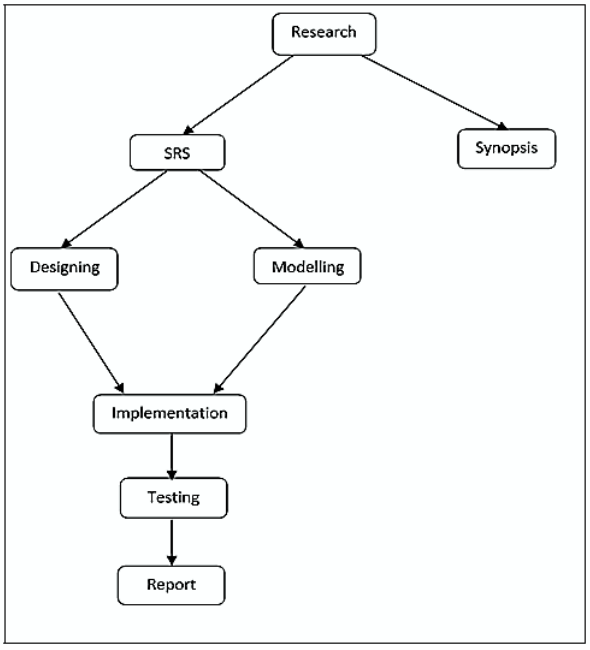

Figure 5.1: Task Network

5.3.3 Timeline

Schedule		Project Activity
July	1st Week	Formation Of Project Group
	2nd Week	Project Topic Selection
	3rd Week	Synopsis Submission
August	1st Week	Presentation On Project Ideas
	2nd Week	Submission Of Literature Survey
	3rd Week	Feasibility Assessment
September	1st Week	Middle Semester Presentation
	3rd Week	Design Of Mathematical Model
	4th Week	End Semester Presentation.
October	1st Week	Report Preparation And Submission
December	3rd Week	1st module presentation
	4th Week	Implementation of 2nd module
January	1st Week	Preparation for paper publishing
	2nd Week	Study of implementation of Docker and Ansible
	3rd Week	Modification to Improve in output.
	4th Week	1st and 2nd module presentation
	5th Week	Discussion on flow of project
February	1st Week	Modification of modules.
	2nd Week	Designed test cases for our module.
	3rd Week	Worked on user interface.
March	1st Week	Integration of all modules.
	3rd Week	Final Report and presentation.
April	1st Week	Final Project Submission

Table 5.8: Project Timeline

5.4 Team Organization

5.4.1 Team structure

- Vedang Wartikar- Lead, Paper publishing, Python Code
- Rutuja Shitole- Synopsis, Ansible Docker implementation
- Shaikh ilhaam- UMLs and DFDs, documentaion

5.4.2 Management reporting and communication

- Weekly reporting to our guide.
- Continuous updating and reviewing of SRS.
- External guidance (sponsorhip)

Chapter 6

Project Implementation

6.1 Overview of Project Modules

- Shutdown - shutting down all machines in a network at once
- Restart - restarting all machines in a network at once
- Installation - Installing softwares/libraries in all the machines in a network at once
- Process Log - Getting the process log for each machine
- Notification - Sending exam alerts / popups etc to all the machines.

6.2 Tools and Technologies Used

The technology stack include

- Python - Programming language used
- Docker - Creating a container image
- Ansible - Automation of deployment
- Slack - Open Source App used

6.3 Algorithm Details

Slack RTM
Slack provide a Web API that gives you the ability to build applications

that interact with Slack in a variety of ways. This Development Kit is a module based wrapper that makes interaction with that API easier. We have a basic example here with some of the more common uses but a full list of the available methods are available here. More detailed examples can be found in our Basic Usage guide
Sending a message to Slack: One of the most common use-cases is sending a message to Slack. If you want to send a message as your app, or as a user, this method can do both. In our examples, we specify the channel name [5], however it is recommended to use the channel-id where possible.

```
import os
import slack

client = slack.WebClient(
    token=os.environ['SLACK_API_TOKEN'])

response = client.chat_postMessage(
    channel='#random',
    text="Hello world!")
assert response["ok"]
assert response["message"]["text"] == "Hello world!"
```

Here we also ensure that the response back from Slack is a successful one and that the message is the one we sent by using the assert statement.

Uploading files to Slack:[2] We've changed the process for uploading files to Slack to be much easier and straight forward. You can now just include a path to the file directly in the API call and upload it that way.

```
import os
import slack

client = slack.WebClient
(token=os.environ['SLACK_API_TOKEN'])

response = client.files_upload(
    channels='#random',
    file="my_file.pdf")
assert response["ok"]
```

Basic Usage of the RTM Client: [8]

The Real Time Messaging (RTM) API is a WebSocket-based API that

allows you to receive events from Slack in real time and send messages as users.

If you prefer events to be pushed to you instead, we recommend using the HTTP-based Events API instead. Most event types supported by the RTM API are also available in the Events API. You can check out our Python Slack Events Adaptor if you want to use this API instead.

An RTMClient allows apps to communicate with the Slack Platform's RTM API.

The event-driven architecture of this client allows you to simply link callbacks to their corresponding events. When an event occurs this client executes your callback while passing along any information it receives. We also give you the ability to call our web client from inside your callbacks.

In our example below, we watch for a message event that contains "Hello" and if its received, we call the say-hello() function. We then issue a call to the web client to post back to the channel saying "Hi" to the user.

```
import os
import slack

@slack.RTMClient.run_on(event='message')
def say_hello(**payload):
    data = payload['data']
    web_client = payload['web_client']
    rtm_client = payload['rtm_client']
    if 'Hello' in data.get('text', []):
        channel_id = data['channel']
        thread_ts = data['ts']
        user = data['user']

        web_client.chat_postMessage(
            channel=channel_id,
            text=f"Hi <@{user}>!",
            thread_ts=thread_ts
        )

slack_token = os.environ["SLACK_API_TOKEN"]
rtm_client = slack.RTMClient(token=slack_token)
rtm_client.start()
```

Async usage slackclient v2 and higher uses aiohttp and asyncio to enable async functionality.

Normal usage of the library does not run it in async, hence a kwarg of run-async=True is needed.

When in async mode its important to remember to await or run/run-until-complete the call.

Slackclient as a script

```
import os
import slack


client = slack.WebClient(
    token=os.environ['SLACK_API_TOKEN'],
    run_async=True
    )

async def send_async_message(
     channel='#random', text='')
    response = await client.chat_postMessage(
            channel=channel,
            text=text
            )
    assert response["ok"]
    assert response["message"]["text"] == "Hello world!"
```

```
import os
import slack
import asyncio

loop = asyncio.get_event_loop()

client = slack.WebClient(
    token=os.environ['SLACK_API_TOKEN'],
    run_async=True
    )

response = loop.run_until_complete(
        client.chat_postMessage(
        channel='#random',
        text="Hello world!"
        )
```

```
        )
assert response["ok"]
assert response["message"]["text"] == "Hello world!"
```

6.4 Software Testing

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest.
Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.

6.4.1 Type of Testing

Manual Testing: It is a type of Software Testing where Testers manually execute test cases without using any automation tools. Manual Testing is the most primitive of all testing types and helps find bugs in the software system.
Any new application must be manually tested before its testing can be automated. Manual testing requires more effort but is necessary to check automation feasibility.Manual Testing does not require knowledge of any testing tool.

6.4.2 Test cases and Test Results

Id	Input	Expected Result	Actual Result	Status
1	Enter wrong Command	Help provided	Directs to Help Menu	Pass
2	Enter Correct Command	Executes on all machines in a network	Executed	Pass
3	Enter wrong IP in SSH	Access denied	Access denied	Pass
4	Send file command which isn't present	File not found message	File not found message	Pass
5	File Sharing	Uploads required file	Uploaded	Pass
6	Shutdown command when process are running	Work should be saved	Work isn't saved	Fail

Table 6.1: Test Cases

Chapter 7

Results

7.1 Outcomes

- Slack is an App for inter-communication of an organization
- That can be integrated with many services with Slack API
- To automate the organizational activities
- By using SLACK BOTS
- And Python Scripts on the server machine
- Environment Deployed on ALL of the client machines in the lab using Docker
- Commands sent to clients using Ansible Playbooks
- That requires SSH to establish connection between server and the clients.

7.2 Screen Shots

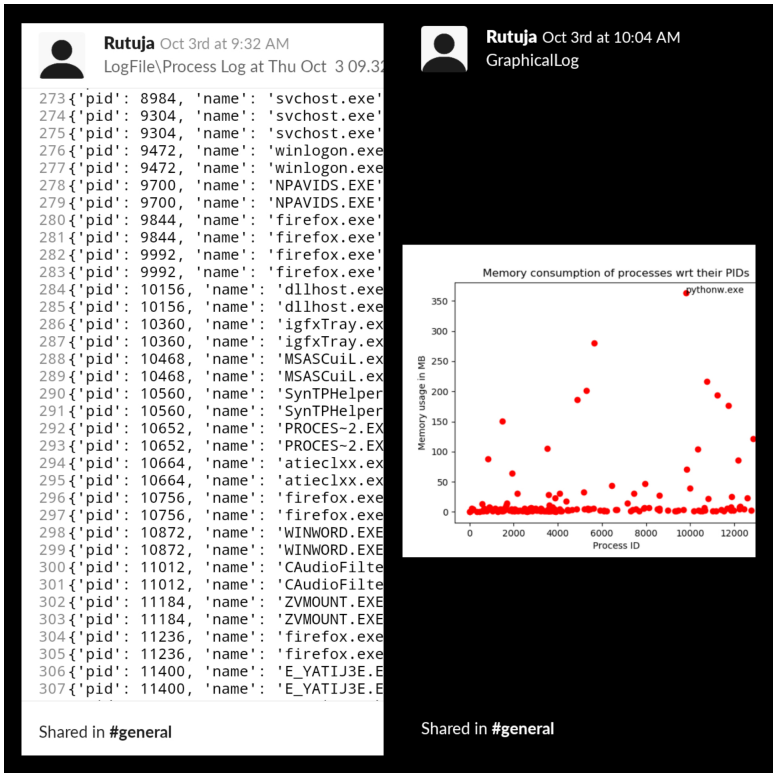

Figure 7.1: Monitoring using process log

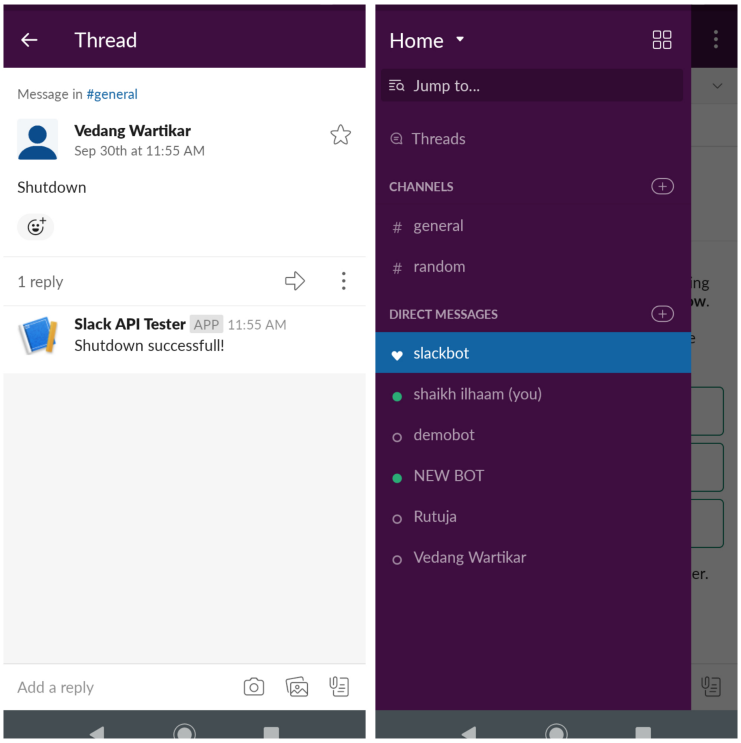

Figure 7.2: Slack App

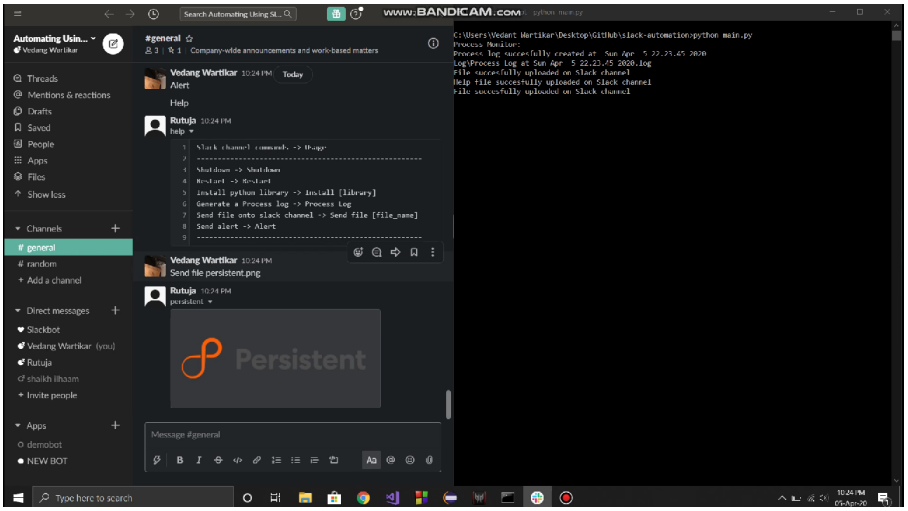

Figure 7.3: File Sharing

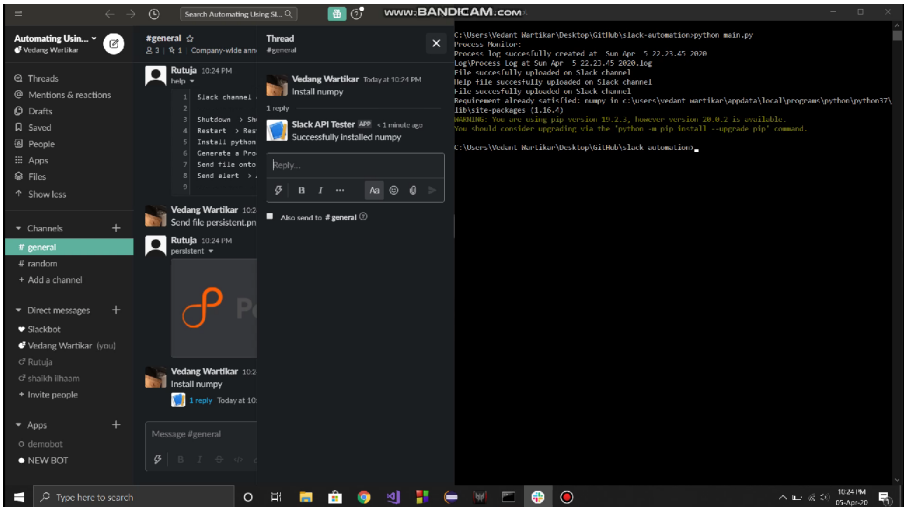

Figure 7.4: Remote installation of softwares/libraries

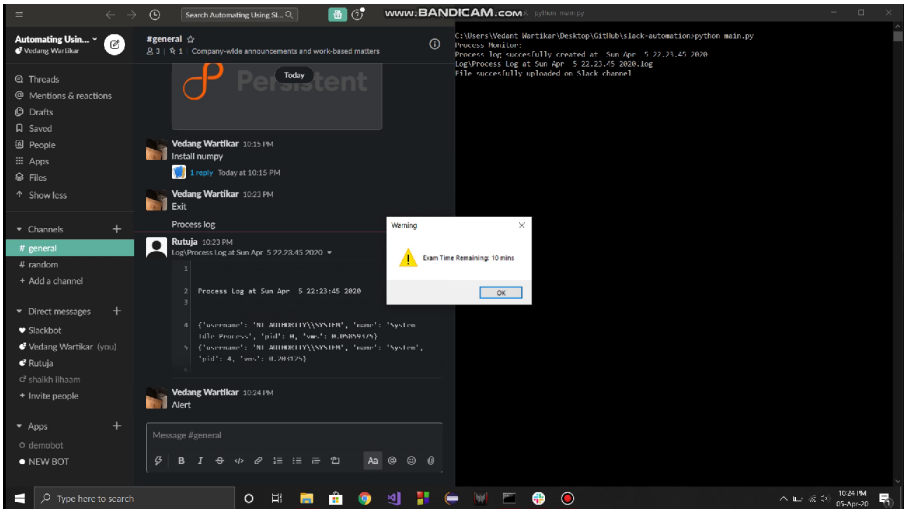

Figure 7.5: Sending alerts/pop-ups to each machine in the network

Chapter 8

Conclusions

8.1 Conclusion

Slack is a powerful, customisable, and intuitive tool for improving team collaboration and productivity. Slack integrates with over 300 other services including Dropbox, Asana, Google Drive, Skype, and more. Slack makes it easy to work in one place, without needing to switch between applications. Slack is highly customisable - you can make it your own in so many ways. Slack is free for an unlimited number of users, and even its paid plans are very affordable (starting at around 22 cents per user per day). Slack is easy and fun to use. Slack has heaps of different bots that can be added, allowing you to put various tasks on autopilot. All content in Slack is easily searchable, including users, messages and files. Slack has a range of slash commands available, making it easy to quickly carry out a task or enquiry.

8.2 Future Work

- Deploying in college labs
- Bringing Ability to command the specific IP from a set of network
- Applying Natural Language Processing to interpret the commands better and give a robotic-like experience to end-user
- Developing an accurate chatbot

8.3 Benefits:

1. Less e-mail
2. Conversations and issues can be rapidly resolved without meetings or without the delay of long e-mail chains.
3. Can have a pulse on all the notifications from all your other tools and systems from inside Slack. It's like a notification dashboard.
4. Can create a workplace culture where people aren't interrupted while working to answer in-person questions.
5. All Team Communication is in One Place
6. Integration with Services We Already Use
7. File Sharing
8. Being able to share and test code snippets across our team in an instant is a great feature of Slack that shouldn't go unmentioned. This alone has increased our productivity by leaps and bounds.
9. Take Advantage of the Open API [9] This essentially means that Slack is made for developers. The tool makes it easy to add other tools to a developer's user stream so developer can use Slack for payroll, Intranet, advertising, structure management, and more. 10. Create a Channel for Blog Posts to Discuss Social Promotion This is usually where businesses start. Sometimes it's tough to get everyone in company on the same page when it come to content, but even IT professionals can benefit from sharing and knowing what's happening with the blog. Can use this Channel as a way to add links to most recent content that has been published and ask others to help promote on social.

8.4 Downsides:

1. High volume of 'urgent' messages.
2. Using of a dedicated Slack channel is a must
3. Manual entry of newly added Host is needed
4. Server failure will cause improper functioning of the system

Chapter 9

References

[1] Kholoud Althobaiti, Kami Vaniea, and Serena Zheng. Faheem: Explaining urls to people using a slack bot. 04 2018.

[2] Help Centre. Add files to Slack. `https://slack.com/intl/en-in/help/articles/201330736-Add-files-to-Slack`, 2009. [Online; File Sharing in Slack].

[3] Ansible Documentation. Everything you need to know about Ansible. `https://docs.ansible.com/`. [The official documentation of Ansible].

[4] Docker Documentation. Everything you need to know about Docker. `https://docs.docker.com/`. [The official documentation of Docker].

[5] Channel Forums. Slack Channel Help. `https://slack.com/intl/en-in/help/articles/360017938993-What-is-a-channel`, 2009. [Online; Help POrtal].

[6] Chris Lam. Can slack curb slacking?: Examining the importance of team communication in reducing social loafing. In *2016 IEEE International Professional Communication Conference (IPCC)*. IEEE, oct 2016.

[7] Eueung Mulyana, Rifqy Hakimi, and Hendrawan. Bringing automation to the classroom: A ChatOps-based approach. In *2018 4th International Conference on Wireless and Telematics (ICWT)*. IEEE, jul 2018.

[8] Community of Slack. Slack RTM API. `https://api.slack.com/rtm`, 2009. [Online; Real Time Messaging].

[9] Slack. Slack API documentaion. `https://api.slack.com/`, 2009. [Online; accessed 19-August-2019].

[10] V. Stray, N. B. Moe, and M. Noroozi. Slack me if you can! using enterprise social networking tools in virtual agile teams. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pages 111–121, May 2019.