A Project Report on

Autonetics and Administration for IT Laboratories

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

Bachelor of Engineering

in

Information Technology

by

Uddhabendra Maity (16104062)
Karthikeyan (16104050)
Atharv Shetty (16104061)

Guide: Dr.Sameer Nanivadekar
Co-Guide: Mr.Vishal Badgujar

Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2019-2020

Approval Sheet

This Project Report entitled ***"Autonetics and Administration for IT Laboratories"*** Submitted by ***"Uddhabendra Maity (16104062),Karthikeyan (16104050),Atharv Shetty (16104061)***is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

Mr.Vishal Badgujar
Co-Guide

Dr.Sameer Nanivadekar
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project Synopsis entitled ***"Autonetics and Administration for IT Laboratories"*** Submitted by ***"Uddhabendra Maity (16104062),Karthikeyan (16104050),Atharv Shetty (16104061)"*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***to the University of Mumbai,is a bonafide work carried out during academic year 2019-2020

Mr.Vishal Badgujar
Co-Guide

Dr.Sameer Nanivadekar
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. we understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Uddhabendra Maity (16104062)
Karthikeyan (16104050)
Atharv Shetty (16104061)

Date:

Abstract

In this modern era of Automation,it has been observed that many of the University and college Labs do not have IT automation capabilities up to the current industrial modern standards.The latest technologies aren't being implemented in colleges and thus the students can't benefit from them. The concept of automation has existed from couple of years but they typically just consisted of performing small tasks such as switching on and off of appliances automatically. In this automation project, we try to implement both the Lab as well as IT automation services together as a package.The IT automation tasks include many services such as automatic installation and uninstalling of several soft-wares whenever needed through SSH networking. The Lab administrator has the option to deliver a set of messages or referral links to the students in a live time scenario. Also an online register would be really helpful for the staff to create a record of the student's activities with timestamps.

Contents

List of Figures

List of Abbreviations

YAML:	Yet Another Markup Language
SSH:	Secure Socket Shell
FTP:	File Transfer Protocol
GUI:	Graphical User Interface
HTML:	Hyper Text Markup Language
IOT:	Internet Of Things

Chapter 1

Introduction

In this system,we try to eradicate the problem regarding the administration of labs. A powerful,systematical and efficient Lab management system is required which will resolve all these basic and generic problems with less human labor The basic technology that we implemented in our system is Ansible. It is used to automate the administration of labs. Here every lab will have its separate playbook. In that playbook a certain set of rules will be written and accordingly the automation of the labs will be done.

Therefore the main motive behind creating the system is to simplify complex orchestration and configuration management tasks. Previously it was observed that many of the PC's are remained switched on even when the labs are not in use,this results in an inefficient use of power and resources. Also the PCs have to be manually switched off by the lab assistants after the end of lab sessions if the students haven't shut it down themselves.

Ansible is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.Ansible is easy to deploy because it does not use any agents or custom security infrastructure.Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. YAML (It's a human-readable data serialization language is commonly used for configuration files)which is very easy for humans to understand, read and write.

Hence, the advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML is in human readable form). Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.The function of a play is to map a set of instructions defined against a particular host.

Mainly, there are two types of machines when we talk about deployment –
–Control machine Machine from where we can manage other machines.
–Remote machine Machines which are handled/controlled by control machine.

Unlike most Ansible modules, network modules do not run on the managed nodes. From a user's point of view, network modules work like any other modules. They work with ad-hoc commands, playbooks, and roles. Behind the scenes, however, network modules use a different methodology than the other (Linux/Unix and Windows) modules use. Ansible is written and executed in Python. Because the majority of network devices can not run Python, the Ansible network modules are executed on the Ansible control node, where Ansible or Ansible-Playbook runs.

Ansible works by connecting to your nodes and pushing out small programs, called ”Ansible modules” to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.Your library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.

Modules are executed directly on remote hosts through playbooks. The modules can control system resources, like services, packages, or files (anything really), or execute system commands. Modules do it by acting on system files, installing packages or making API calls to the service network.Plugins allows to execute Ansible tasks as a job build step. Plugins are pieces of code that augment Ansible's core functionality. Ansible ships with a number of handy plugins, which we can easily write our own plugin as well.

Chapter 2

Literature Review

Xavier Decoster [1] proposed an architecture stating the usefulness of NuGet which is an open source visual studio extension. He named the architecture as ProNuget. ProNuGet gives a solid, practical knowledge of both how to maintain the software dependencies under control and a long term reliability. Doesn't matter if you're coding entirely with .NET framework assemblies or also using HTML,CSS and JavaScript files in your various applications, the featured system shows you how to manage their requirements reliable and smoothly.

D. Palma and T.Spatzier [2] presented a meta model for stating various IT services. This meta model states both the overview of the structure as well as the managing sector. A Topology Template states the overview of the service. Plans states the various processing models that are used to build as well as kill a service and also helping in managing a particular service. Node Types are introduced differently for re usability purposes. Service Templates can be based on and built on-top of other Service Templates based on the concept of Requirements and Capabilities introduced in the previous section. This paper helped us in analyzing the different network topology which we can consider for our purpose. Also it helped us in gaining an overview of integrating cloud technologies in the system.

Pavel Mašek Martin and ŠtůsekJan Krejčí [3] proposed their design using the ansible framework where they used ansible as a core part of the design to automate the IT operations to a certain extent. However the utilization and application of this design in real life scenarios was not depicted clearly. This design supported many operating systems at the same time. Thus creating an OS friendly environment. Types of operating systems included : (i) Windows 7; (ii) Windows 10; (iii) Ubuntu 16.04; (iv) Debian 8.It also included the ability to install and uninstall various applications such as Google Chrome, Mozilla Firefox and Microsoft Visual Studio. In addition, several system operations must be enabled: (i) Displaying information and status of stations; (ii) restarting and shutting down stations; (iii) creating users; (iv) creating and deleting a file; (v) installing system updates. As this was the closest and the most relevant research related to our paper, we formed this as our base paper and carried all the further research and development over this paper.

Nishant Kumar Singh [4] from Amity University proposed a very clear and precise description of automated provisioning also called as self-service provisioning. In this paper he explained the ways of deploying IT services by using predefined procedure that are carried out electronically without requiring human intervention. In a traditional setting, provisioning is a manual process that requires the assistance of several people in several roles and involves multiple steps. It could take days or even weeks to move a request from the submission phase through the actual activation of service. Automating provisioning allows customers to set up and make changes to services themselves by using a Web browser or other client interface. It can provide a more efficient and rapid response to business requests and cut service activation or service change time down to hours or even minutes. This paper helped us in understanding the several ways of deploying the IT services and carry out the operational processes.

J.O.Benson and J. J. Prevost, P. Rad [5] presented an efficient way of taking advantage of scripting techniques. It allows the flexibility of installing all of the necessary and desired software, including updates. Likewise, scripting can be dynamic enough to distribute custom files to each of the N-nodes to ensure that necessary files work appropriately, something that is impossible when using imaging for server set up. An example of this might be distributing lists of IP addresses, custom host-names, or specific license files for software, or the system, to operate correctly. Another advantage of scripting is that it requires very little space. As we are currently designing the whole system in linux environment, we extracted a lot of knowledge about shell scripting from this paper and thus trying to utilize this technique in further operations.

Jay Shah and Dushyant Dubaria [6] in their paper "A Survey of DevOps tools for Networking" researched and stated various facts about using the Redhat's open-source technology "Ansible" . For ansible to work, a proper working version of python is required as it requires the various libraries from python. Ansible works in two modes: ad-hoc modes and playbook files. The ad-hoc mode involves passing the instructions to the remote servers via ansible commands. Using the other mode i.e. with the help of playbooks, all the tasks needed to be mentioned in the playbook yml file. Once the playbook is executed all the tasks are performed accordingly. a

Chapter 3

Project Conception

3.1 Existing System Architecture

Here in the figure shown below,the administrator will be provided with a remote access and with the help of the internet it is connected to the web server and through that web server, a web application will be hosted.

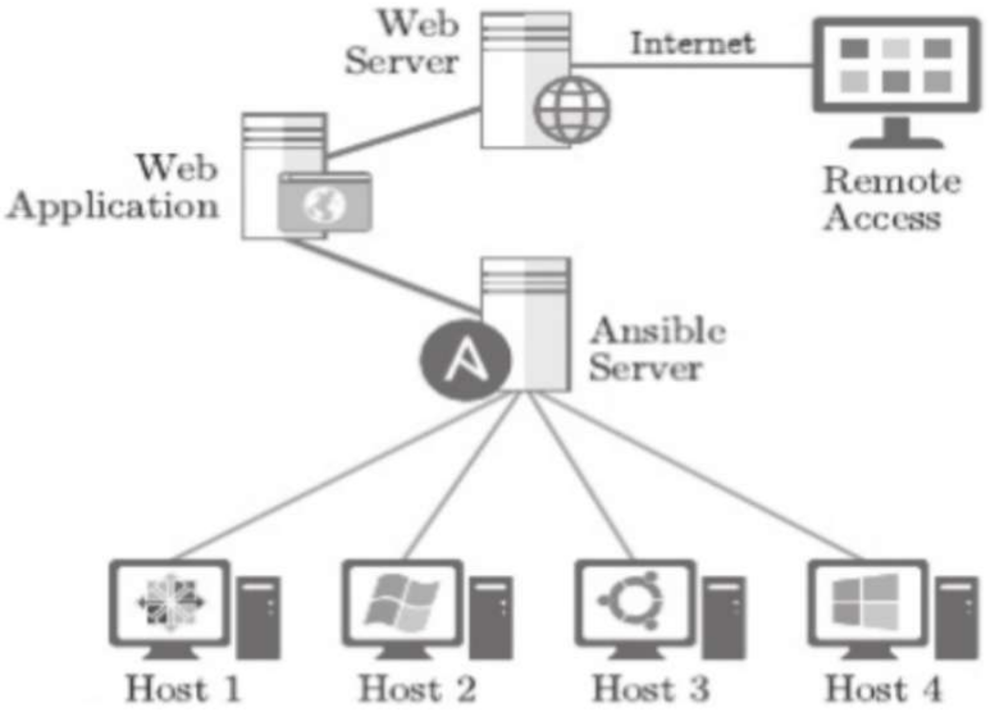


Figure 3.1: Existing Server Architecture

Remote Access:

The Administrator will be provided with the PC,where he will have access to all the pc's which are connected in network. These systems should be connected to internet via WLAN

Web Server:

Here to host a website,web server programs must be present. Apache is one of the most leading web servers used nowadays

Web Application:

Here with the help of the server a web application will be hosted. In this web application,with the help of a proper GUI the administrator will easily perform necessary tasks.

Ansible server:

These are the pc's which will be connected to the system of the administrator. The PC's must be in connection with each other These PC's can be installed with any operating system according to their choice

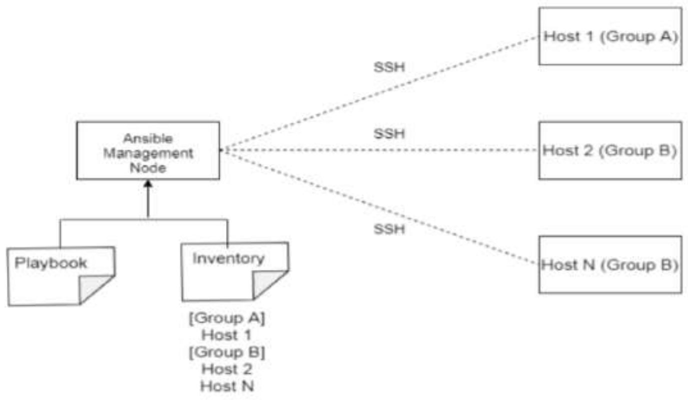


Figure 3.2: Working of ansible

3.2 Problem Statement

In current university labs most of the administrative tasks are done manually which consumes lot of time and efforts. With the help of ansible framework and a proper supporting GUI, we can unleash and maximize the full potential of the servers and many of the current lab administrative problems can be resolved easily. The major drawback of the existing architecture is that its completely restricted within the scope of ansible framework. The core functionality of this architecture is automated installation and management of software packages. Ansible is originally only a command-line interface tool and thus it lacks an elegant user interface. Only a well versed user will be able to operate on a command line tool, which means a layman user will find it difficult to operate and run the ansible commands effectively.

3.3 Objectives

- To Automate the Software installation/upgradation process.
- To Automate the PC shutdowns.
- To regulate the user identity of every PC along with time in a digital format.
- To unleash the full potential of Ansible for IT automation.
- To schedule the auto power-on of all PCs at the start of the sessions.
- To obtain the information of the installed softwares and their versions remotely for software management purposes.

3.4 Proposed Technology Stack

- Ansible- A red hat devops tool
- Python3 for running Ansible
- Semaphore-a web-based GUI interface for handling Ansible queries
- Nodes will be running on Ubuntu,Windows
- Front end developement :HTML5, Javascript,CSS

Chapter 4

Project Design

4.1 UML Diagrams

Activity Diagram:

The Lab in-charge has to enter the lab number in which he/she is conducting the lab session.The full list of the available soft-wares for the lab will be displayed and the lab in-charge has to just select the software which is needed by the students for that lab session and the soft-ware will be automatically deployed in the lab nodes.

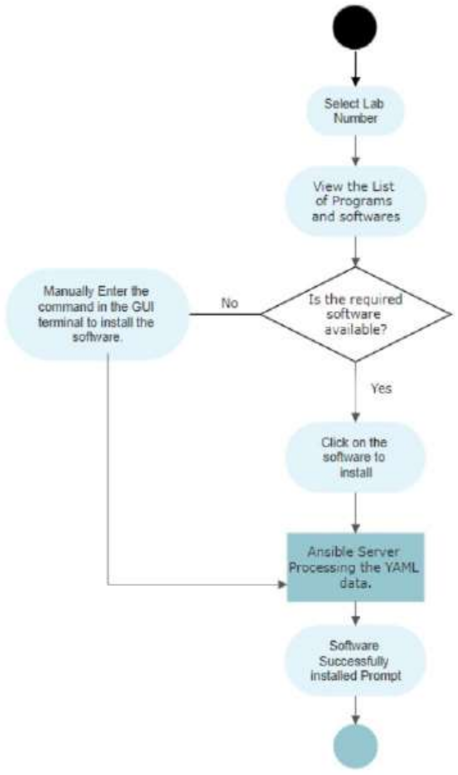

Select Lab Number
View the List of Programs and softwares
Is the required software available?
No
Manually Enter the command in the GUI terminal to install the software.
Yes
Click on the software to install
Ansible Server Processing the YAML data.
Software Successfully installed Prompt

Figure 4.1: Activity Diagram

Use Case Diagram:

1)Administrator:The administrator takes care for upgrading the list of soft-wares, make changes in the system according to the lab in-charge's feed-backs, carry out OS installations,etc.

2)Lab In-charge: He/She is responsible for conducting the labs and make the installations of soft-wares efficiently as per the required subjects and their soft-wares.

3)Students: The students will have to enter their details into the records manual and it will be stored on to our database.

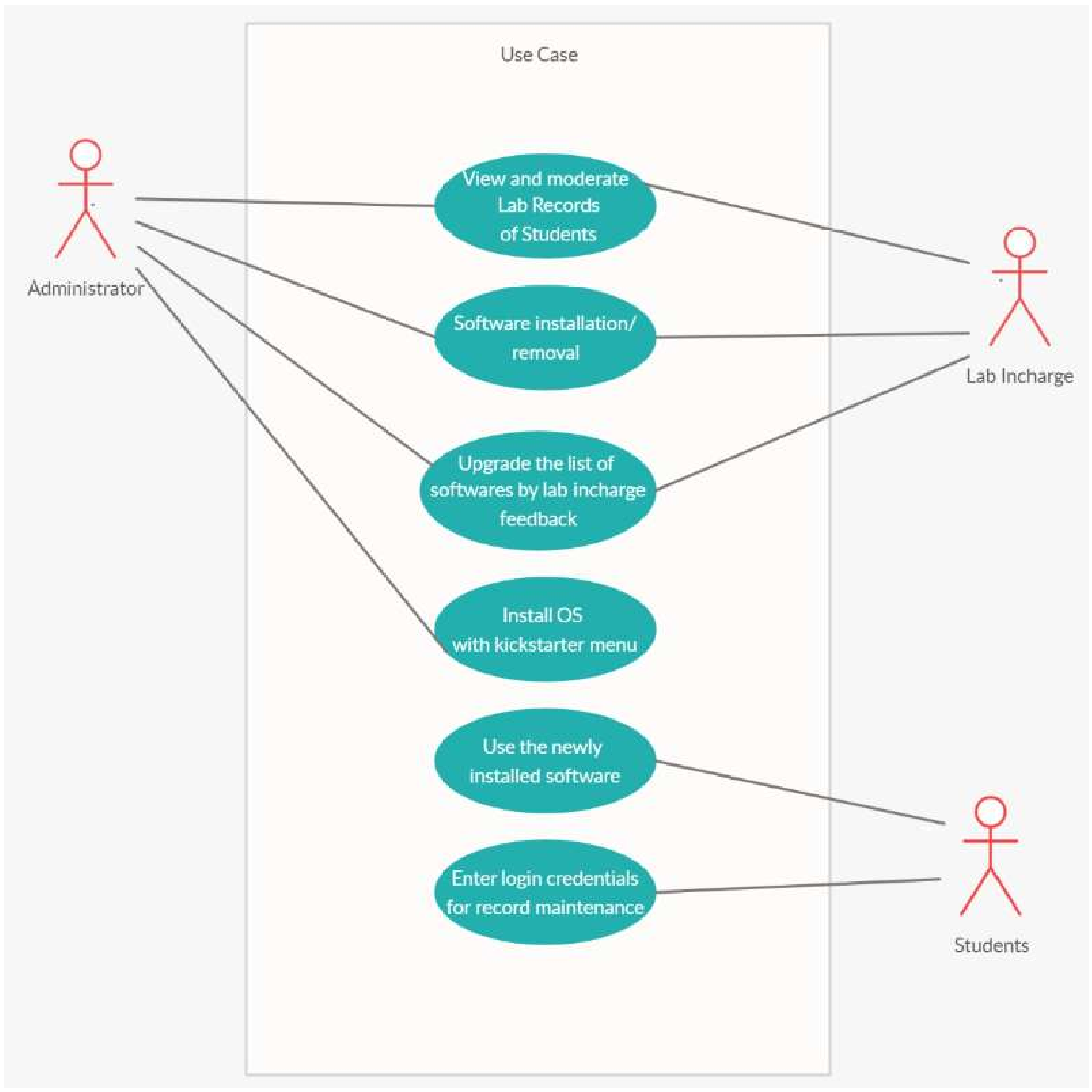

Use Case
Administrator
Lab Incharge
Students
View and moderate Lab Records of Students
Software installation/ removal
Upgrade the list of softwares by lab incharge feedback
Install OS with kickstarter menu
Use the newly installed software
Enter login credentials for record maintenance

Figure 4.2: Use Case Diagram

Class Diagram:

The different entities in this environment are:
1. Student
2. PCs in the lab
3. Lab Instructor
4. Ansible Server
5. Softwares
6. System Admin

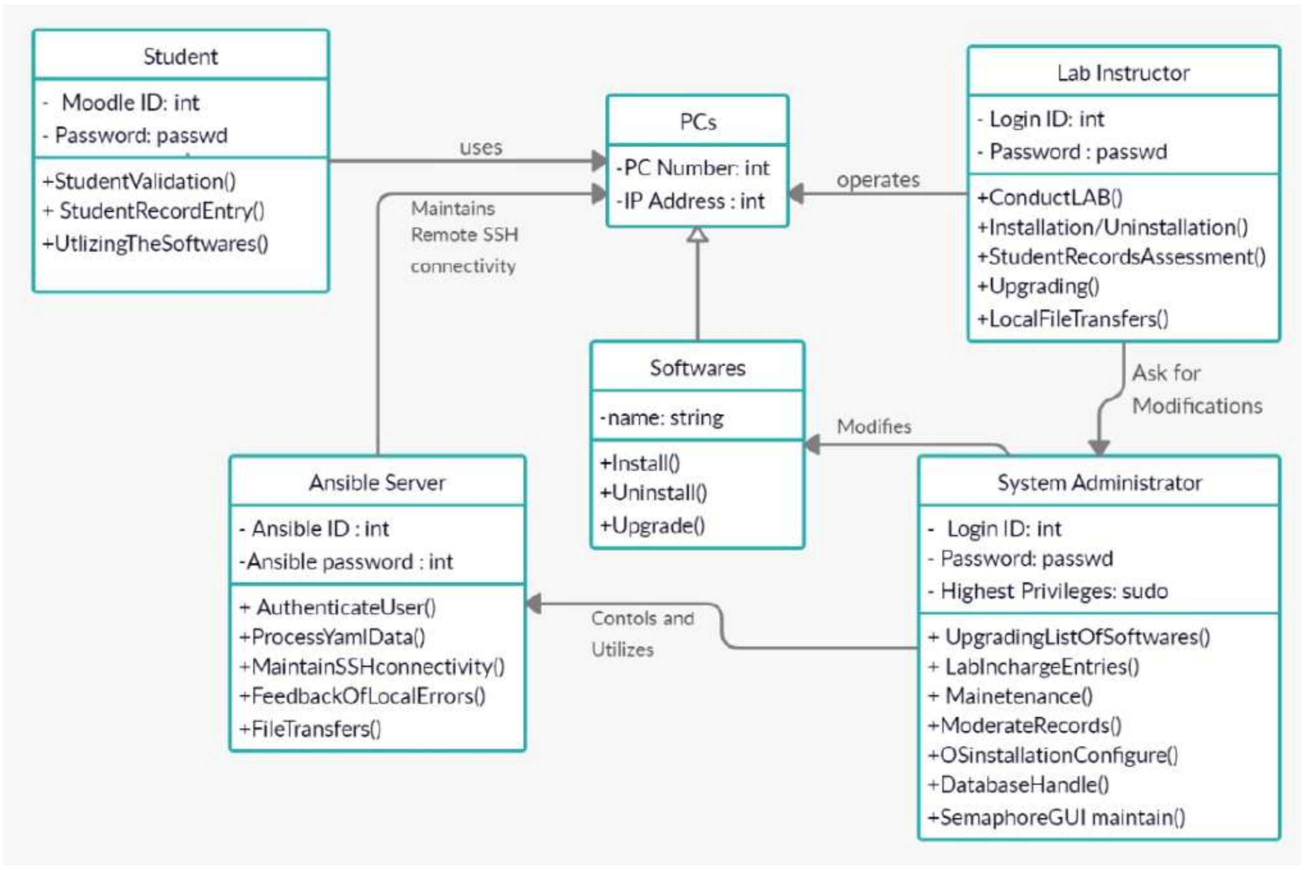


Figure 4.3: Class Diagram

4.2 Proposed System Architecture

The project comprises of the following modules:

Remote Devices: Desktops and mobiles will be useful for managing the lab which will be further connected to the main web server via internet medium. Here both web based GUI as well as an android application will be developed for more flexibility.

Web Server: Web server shall be used for hosting the whole automated environment. Web server will supervise important characteristics like time zone, host names, user information data and domain Names.

Application: The web server will host the required Web application which will be connected to all the lab systems and can be easily managed and controlled via a user-friendly GUI.

Ansible Server: Ansible server will be used for hosting the ansible environment throughout the whole network via the LAN connection and thus all ansible related queries and commands for lab automation can be easily carried throughout.

Client PCs: No matter what operating system the client PC's are running on , any activity can be easily carried out irrespective of what the Operating system is, throughout the network. As we are using a free open-source platform for our system, many labs can be automated using the same architecture at a very cost-effective valuation.

FTP server: This server will contain all the large files for easy deployment to the local machines in the LAN instead of downloading them from internet individually.

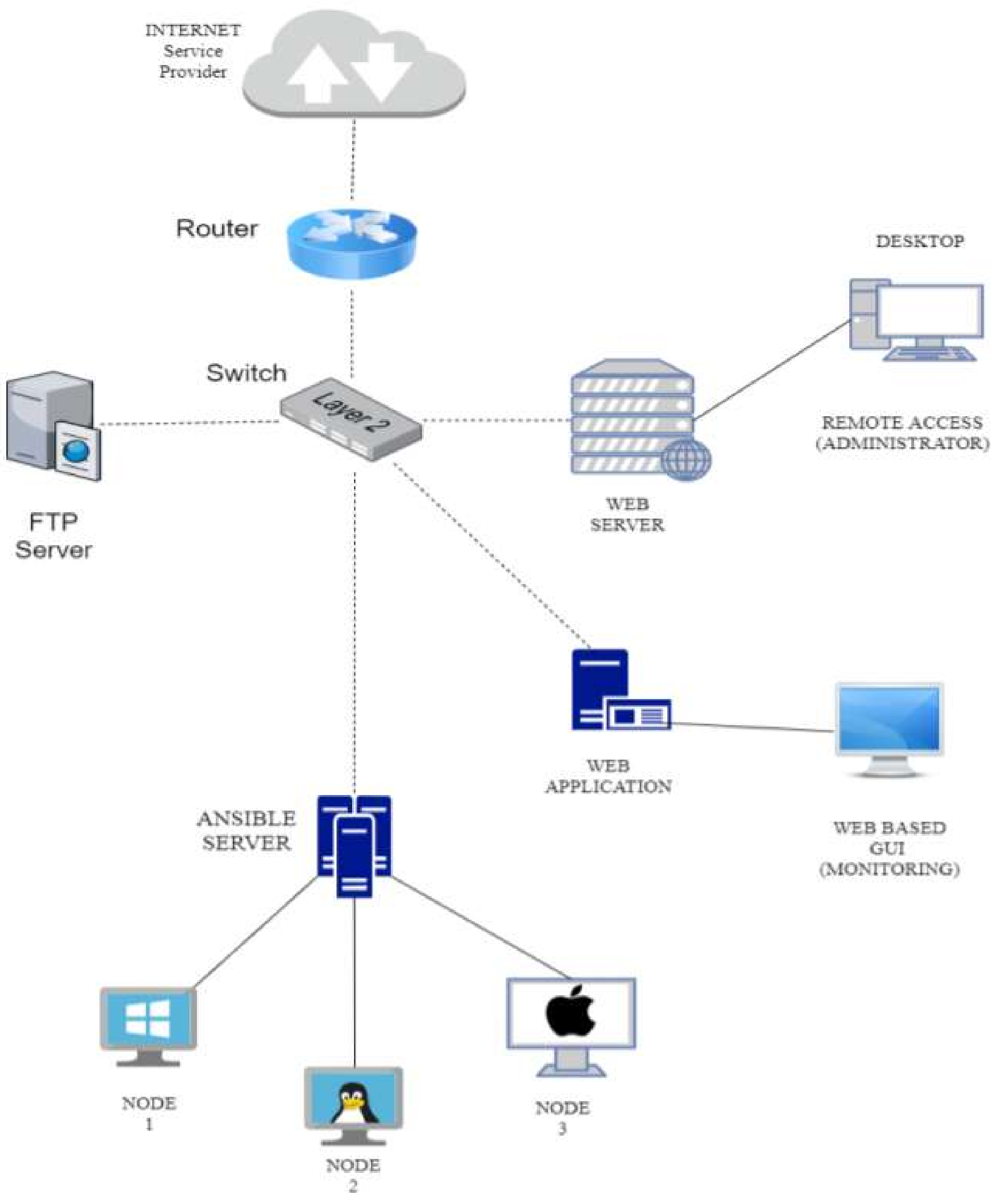


Figure 4.4: Project Design

Chapter 5

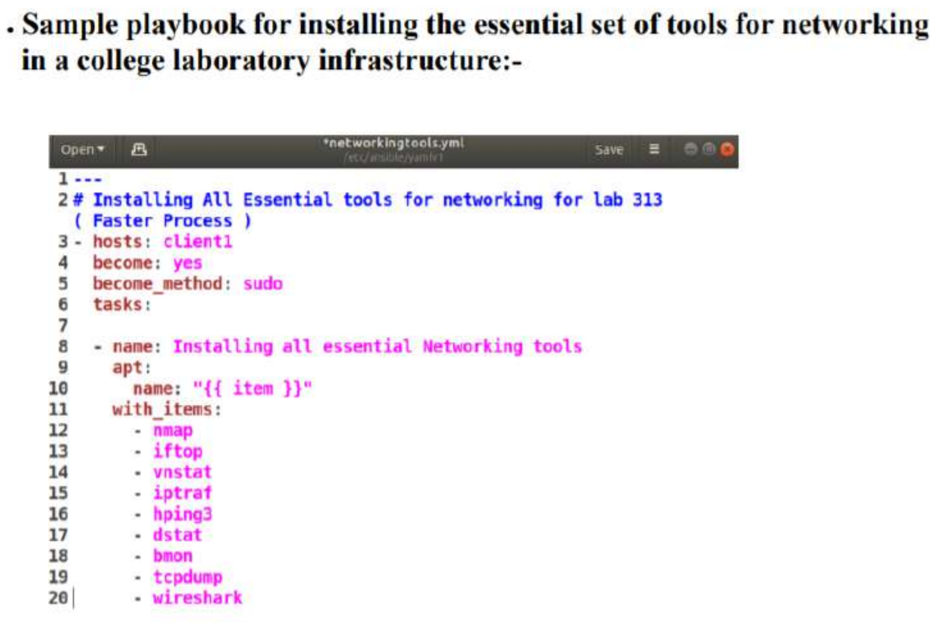
Project Implmentation

5.1 Implementation

1.Here is a sample playbook which includes various tasks:

2.This sample playbook contains all the essential softwares which is required for that particular lab

3.The Script which is written in the YMl file is displayed below:



. Sample playbook for installing the essential set of tools for networking in a college laboratory infrastructure:-

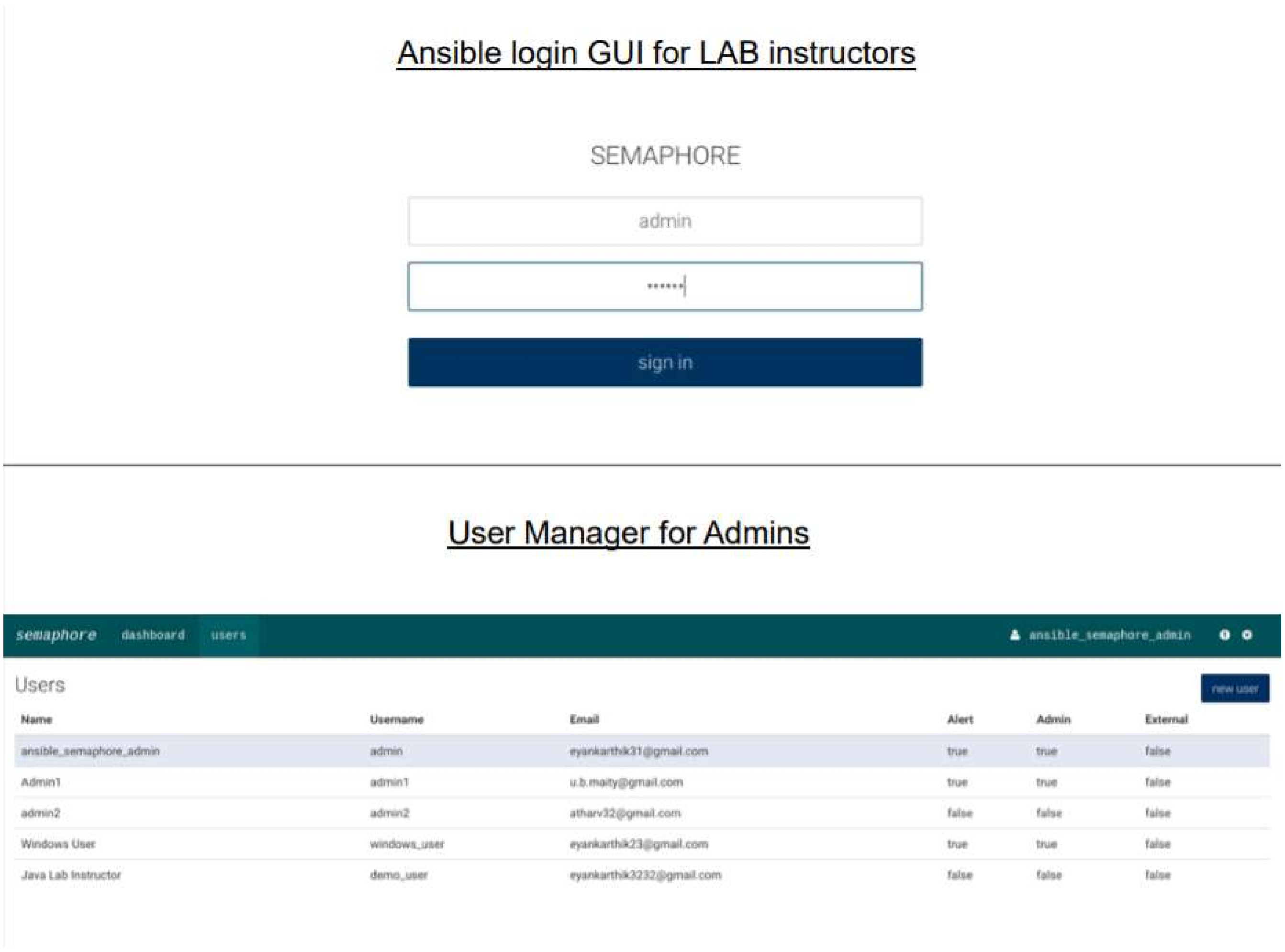
```
---
# Installing All Essential tools for networking for lab 313
( Faster Process )
- hosts: client1
  become: yes
  become_method: sudo
  tasks:

  - name: Installing all essential Networking tools
    apt:
      name: "{{ item }}"
    with_items:
      - nmap
      - iftop
      - vnstat
      - iptraf
      - hping3
      - dstat
      - bmon
      - tcpdump
      - wireshark
```

Figure 5.1: Sample playbook

Semaphore is a great hosted CI/deployment solution.The great part though is that because Semaphore gives you a functioning Ubuntu 12.04 LTS instance you don't have to deal with booting up a new node.

1.Here we wil get to see the ansible GUI which is hosted by the semaphore,where the admin will be provided with the username and credentials

2.Here the admin can add users and grant the privileges according to the needs



Users

Name	Username	Email	Alert	Admin	External
ansible_semaphore_admin	admin	eyankarthik31@gmail.com	true	true	false
Admin1	admin1	u.b.maity@gmail.com	true	true	false
admin2	admin2	atharv32@gmail.com	false	false	false
Windows User	windows_user	eyankarthik23@gmail.com	true	true	false
Java Lab Instructor	demo_user	eyankarthik3232@gmail.com	false	false	false

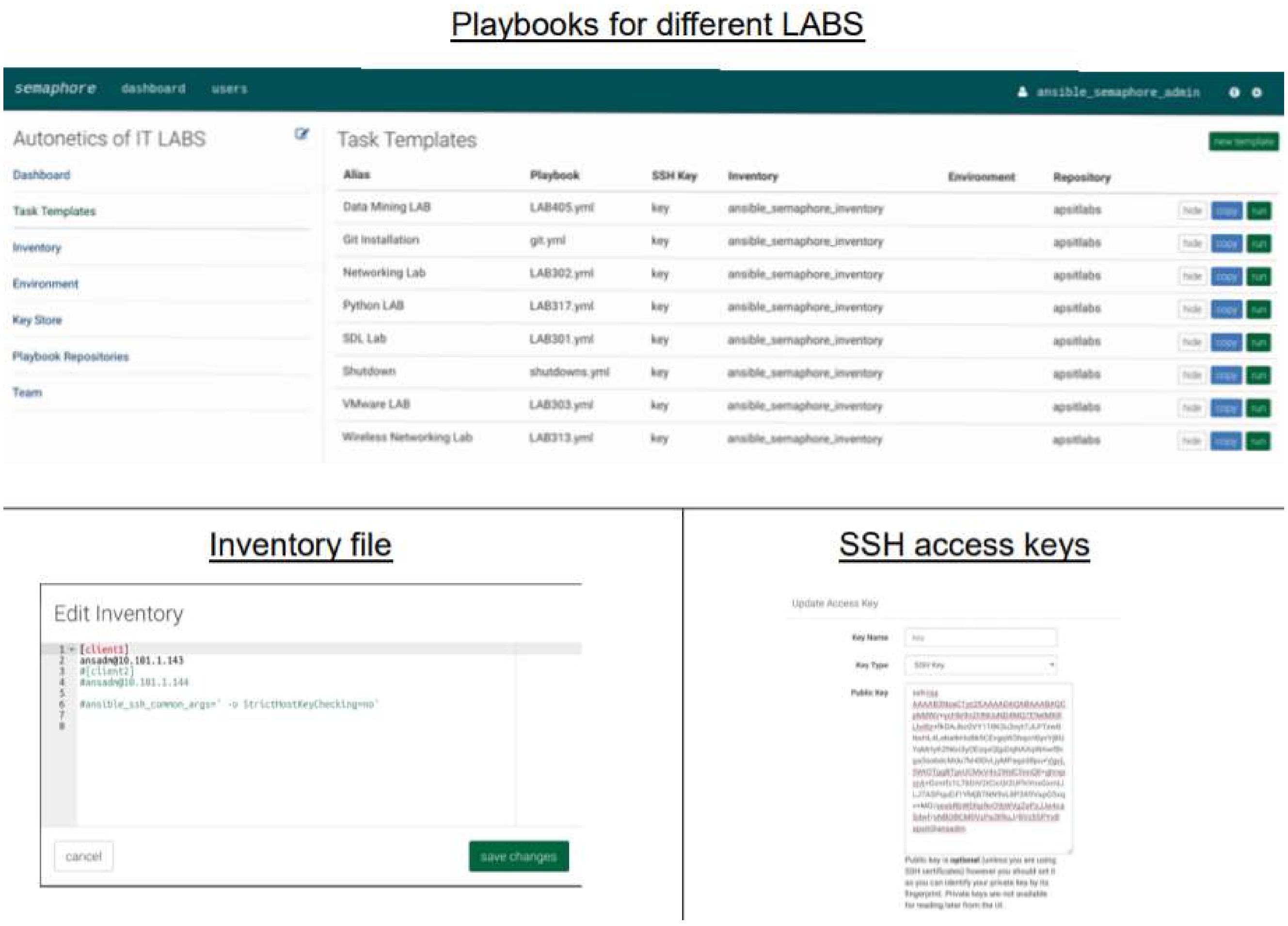
Figure 5.2: Ansible GUI

1.Here as shown below there are various playbooks created for different labs as per their needs.

2.The playbooks will be executed once the the user presses the button.

3.Inventory file includes the client ip addresses.

4.SSH keys contain the public key of the host.



Task Templates

Alias	Playbook	SSH Key	Inventory	Environment	Repository	
Data Mining LAB	LAB405.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
Git Installation	git.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
Networking Lab	LAB302.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
Python LAB	LAB317.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
SDL Lab	LAB301.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
Shutdown	shutdowns.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
VMware LAB	LAB303.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run
Wireless Networking Lab	LAB313.yml	key	ansible_semaphore_inventory		apsitlabs	hide copy run

Edit Inventory

```
[client1]
ansadm@10.101.1.143
#[client2]
#ansadm@10.101.1.144

#ansible_ssh_common_args=' -o StrictHostKeyChecking=no'


```

cancel   save changes

Update Access Key

Key Name	key
Key Type	SSH Key
Public Key	ssh-rsa [illegible]

Public key is **optional** (unless you are using SSH certificates) however you should set it as you can identify your private key by its fingerprint. Private keys are not available for reading later from the UI

Figure 5.3: Playbooks for diffrent files

1.Here is the Student Lab Utilization Record

2.The Lab utilization Record contains the admin login and the faculty login

3.Here the student have to enter the credentials such as Username,Password,Batch, Subject and Lab Number

4.Once all the credentials are entered the student information will be stored successfully in the record

Student Lab Utilization Record

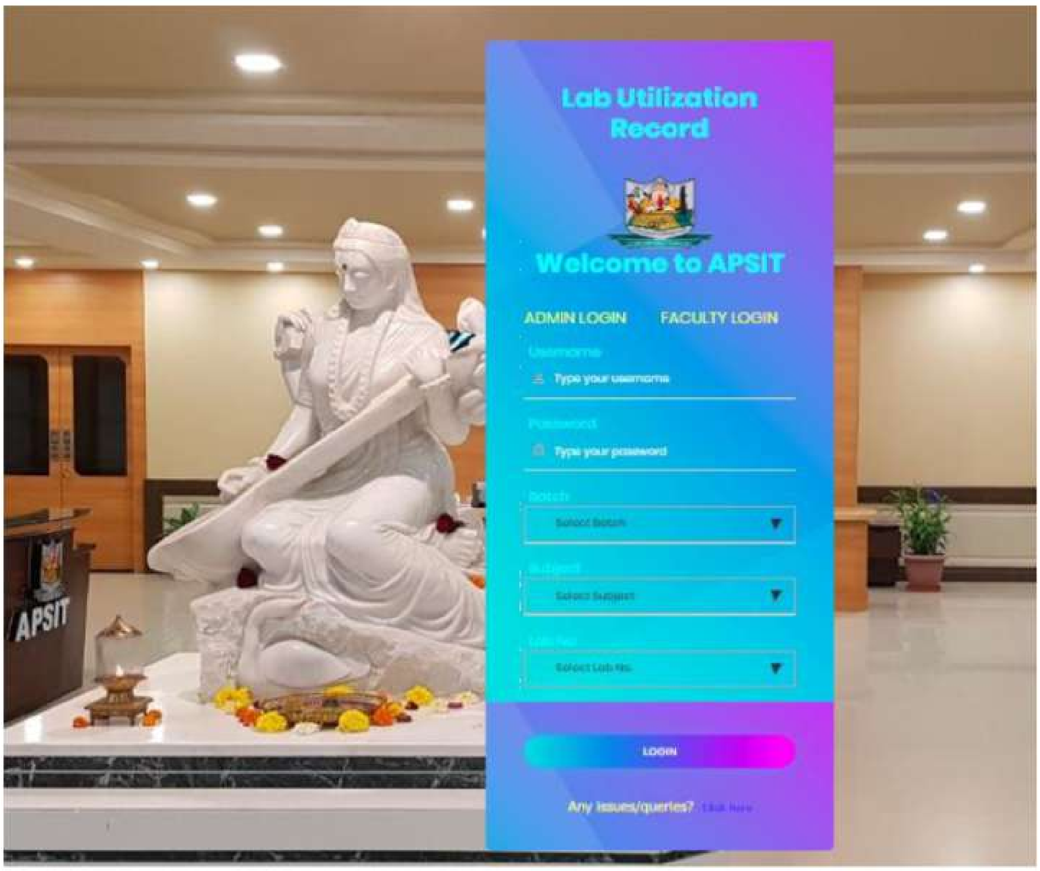


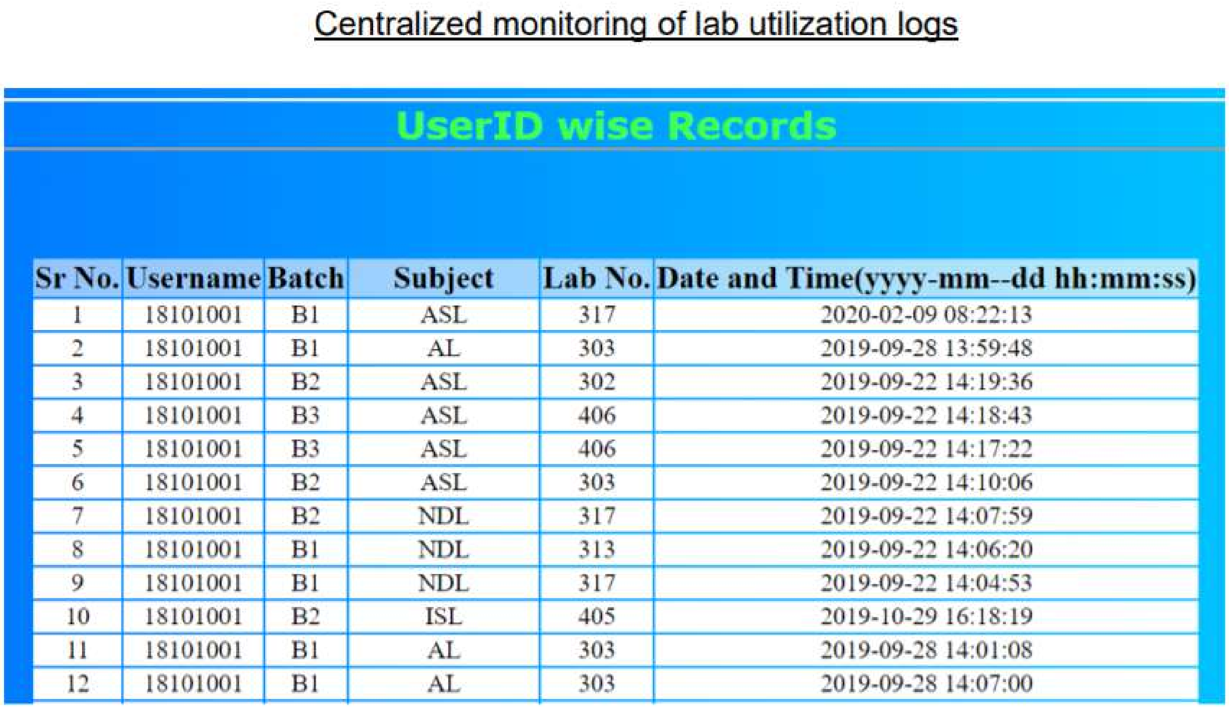
Figure 5.4: Student Lab Utilization Record

1.Here are the logs generated of various students who have logged in

2.The logs contains SrNo,Username,Batch,Subject,Lab No.and Date and Time

3.This will be stored in the database securely.

4.This information will be stored along with the exact time and date.



Centralized monitoring of lab utilization logs

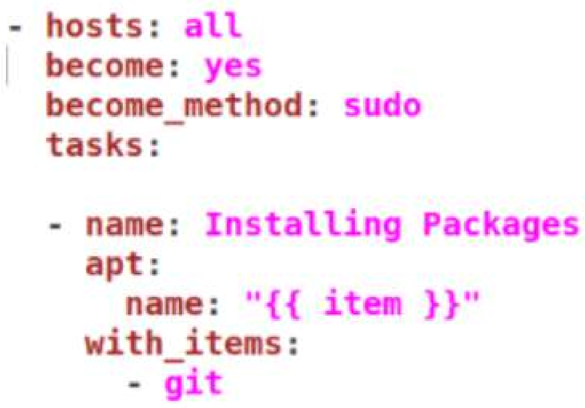
UserID wise Records

Sr No.	Username	Batch	Subject	Lab No.	Date and Time(yyyy-mm--dd hh:mm:ss)
1	18101001	B1	ASL	317	2020-02-09 08:22:13
2	18101001	B1	AL	303	2019-09-28 13:59:48
3	18101001	B2	ASL	302	2019-09-22 14:19:36
4	18101001	B3	ASL	406	2019-09-22 14:18:43
5	18101001	B3	ASL	406	2019-09-22 14:17:22
6	18101001	B2	ASL	303	2019-09-22 14:10:06
7	18101001	B2	NDL	317	2019-09-22 14:07:59
8	18101001	B1	NDL	313	2019-09-22 14:06:20
9	18101001	B1	NDL	317	2019-09-22 14:04:53
10	18101001	B2	ISL	405	2019-10-29 16:18:19
11	18101001	B1	AL	303	2019-09-28 14:01:08
12	18101001	B1	AL	303	2019-09-28 14:07:00

Figure 5.5: Centralized minitoring of lab utilization logs

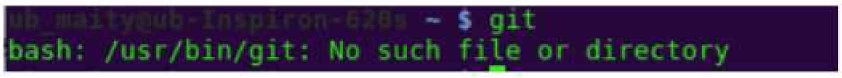
5.2 Results

1.Running git Playbook File in Ansible Server



```
- hosts: all
  become: yes
  become_method: sudo
  tasks:

  - name: Installing Packages
    apt:
      name: "{{ item }}"
    with_items:
      - git
```

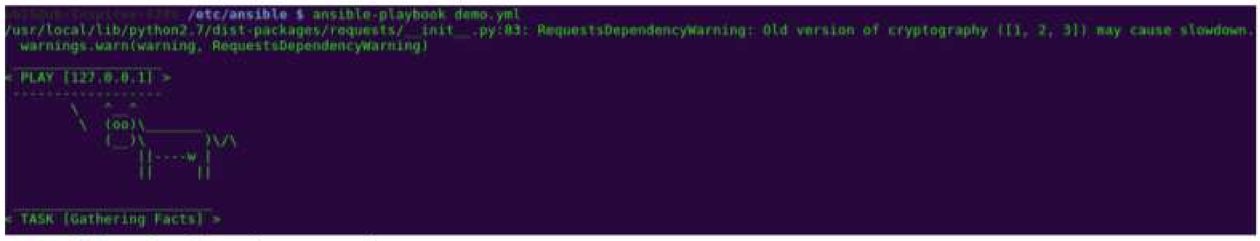
2.Checking on the client machine whether git is already installed or not



```
ub_maity@ub-Inspiron-620s ~ $ git
bash: /usr/bin/git: No such file or directory
```

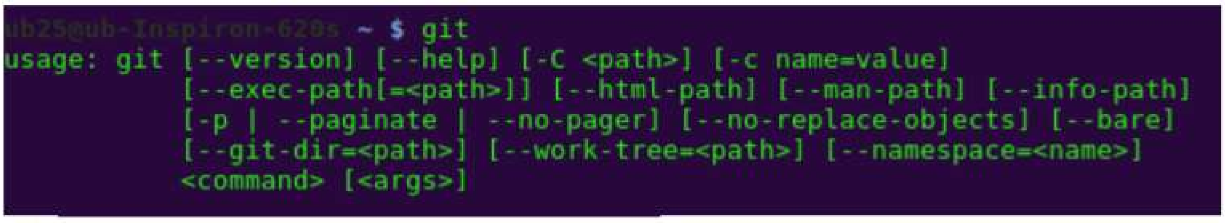
3.Executing the playbook with command

$ anible-playbook git.yml



```
[illegible] /etc/ansible $ ansible-playbook demo.yml
/usr/local/lib/python2.7/dist-packages/requests/__init__.py:83: RequestsDependencyWarning: Old version of cryptography ([1, 2, 3]) may cause slowdown.
  warnings.warn(warning, RequestsDependencyWarning)
 ___________________
< PLAY [127.0.0.1] >
 -------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

 _________________________
< TASK [Gathering Facts] >
```

4.Checking whether git gets installed or not.



```
ub25@ub-Inspiron-620s ~ $ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

Chapter 6

Testing

For this project, we opted to go with the following testing methods:
1. Functional Testing
2. Usability Testing
3. Graphical User Interface Testing

6.1 Functional Testing

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. Functional testing is normally performed during the levels of System Testing and Acceptance Testing.

Functional testing was important in our project in order to examine the command line output produced by Ansible when executing a playbook. It included examining the module arguments, tasks, log files and the execution status of Playbooks. Functional testing also helped us in assessing the real-time logs generated by Semaphore (Ansible GUI), sorting Lab Utilization as per Moodle ID, Year and Batch.

6.2 Usability Testing

Usability testing is a way to see how easy to use something is by testing it with real users. Usability testing is a method used to evaluate how easy a website is to use. The tests take place with real users to measure how 'usable' or 'intuitive' a website is and how easy it is for users to reach their goals. The key difference between usability testing and traditional testing (bug testing, acceptance testing etc.) is that usability testing takes place with actual users or customers of the product. Whilst traditional testing might be undertaken by a developer, designer or project manager, usability testing removes any bias by collecting feedback direct from the end user. Usability Testing was necessary in evaluating the overall ease-of-use of operating with Ansible environment like: a) Accessing and Editing Ansible hosts, inventory files. b) Working with Ansible Modules. c) Configuring and executing Ansible Playbooks. d) Viewing Lab Utilization Records on the website. Usability Testing was essential in evaluating the feasibility and intuition feature of Semaphore (Ansible GUI).

6.3 Graphical User Interface Testing

GUI Testing is a software testing type that checks the Graphical User Interface of the Application Under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes, and windows, etc. The purpose of Graphical User Interface (GUI) Testing is to ensure UI functionality works as per the specification. GUI is what the user sees. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

Graphical User Interface Testing helped us in evaluating the UI functionality of Semaphore (Ansible GUI) and Lab Utilization Record Website. It helped us in adjusting the fonts, images and buttons used in our website to make them more accessible and readable, and positioning website elements in an optimal width alignment. GUI Testing was essential in ensuring that the colours of the font and warning messages are aesthetically pleasing. It helped us to make sure that the users could execute the intended functionality of Ansible using Semaphore (Ansible GUI).

Chapter 7

Conclusion and Future Scope

The main motive of our work is to create a trustworthy, efficient and real-time system for administration of IT labs in universities. Now all the administrative tasks inside the lab can be executed at a very minimal time and effort with our system. The overall purpose was to minimize the efforts and ensure rapid deliveries of the needed soft-wares through automation. These objectives have been checked successfully and we hope to enhance the system furthermore and increase the advancements in our system. Thus we are making an effort to implement this system in the current university labs and modernize the IT labs methodically.

Integration of IOT :

We plan to integrate IOT interfaces in our system for controlling all the electrical appliances throughout the lab remotely.

Docker Container :

Using Docker containers for easy deployment of applications in real time.

Enhancing centralized administration :

Providing real time log generation to the system administrators for moderating the student's usage during exams or placements.

Bibliography

[1] Xavier Decoster and Maarten Balliauw“Automated Delivery in Pro Nuget” October 2016.

[2] D.Palma and T.Spatzier. December 2016 “Topology and orchestration specification for cloud applications (TOSCA) ” November 2013

[3] Pavel MasekMartin and ŠtůsekJan Krejčí, “Unleashing Full Potential of ansible Framework: University Labs Administration” May 2018

[4] Nishant Kumar Singh , Amity University, “Automated Provisioning of Application” January 2016

[5] J.O.Benson, J. J. Prevost, P. Rad, “Survey of automated software deployment for computational and engineering research,” in System Conference (Sys Con), 2016 Annual IEEE, pp.1–6, IEEE, 2016.

Appendices

Appendix-A: Ansible Download and Installation

1. Go to terminal and run the following commands

```
sudo apt-get update
sudo apt install software-properties-common
sudo apt-add-repository –yes –update ppa:ansible/ansible
sudo apt install ansible
```

2. Check whether the ansible is installed

```
$ ansible –version
```

3. Editing the /etc/hosts file with server and client ip addresses.

```
$ /etc/hosts
```

4. Appending the sudoers file in /etc/sudoers.d

```
$/etc/sudoers.d
```

5. Editing the /ansible/hosts file i.e adding the ip addresses of the clients..

```
$ /ansible/hosts
```

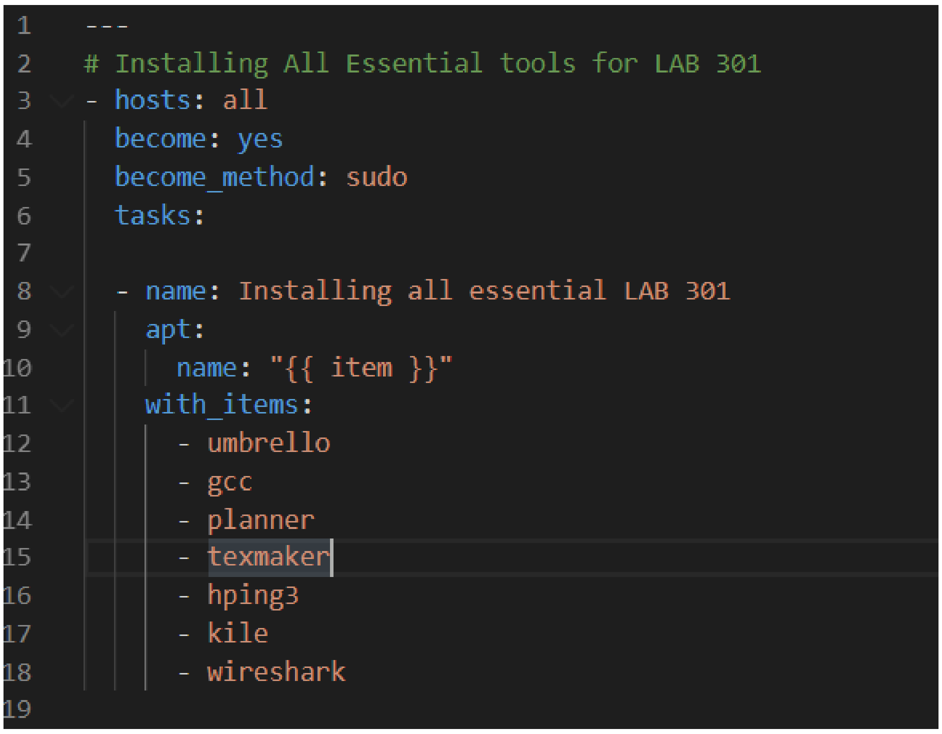
6. Creating a secure path with SSH connectivity between client and server.

```
$ssh-keygen -t rsa
```

Once all the installation and configuration process is done . Now Ansibe server is ready to run a playbook.

Appendix-B: Running a playbook in the ansible server

1. Write an YAML file consisting of the required tasks.
Example:



```yaml
---
# Installing All Essential tools for LAB 301
- hosts: all
  become: yes
  become_method: sudo
  tasks:

  - name: Installing all essential LAB 301
    apt:
      name: "{{ item }}"
    with_items:
      - umbrello
      - gcc
      - planner
      - texmaker
      - hping3
      - kile
      - wireshark

```

This is an YAML file which includes the host options and also the various tasks which are going to be performed or deployed with the help of the ansible server.

hosts: This attribute consists of the list of the clients to whom the ansible server should interact with.

become and become method: This attribute deals with the sudo privileges of the system and ensures that the ansible server has the privileges of performing the given task.

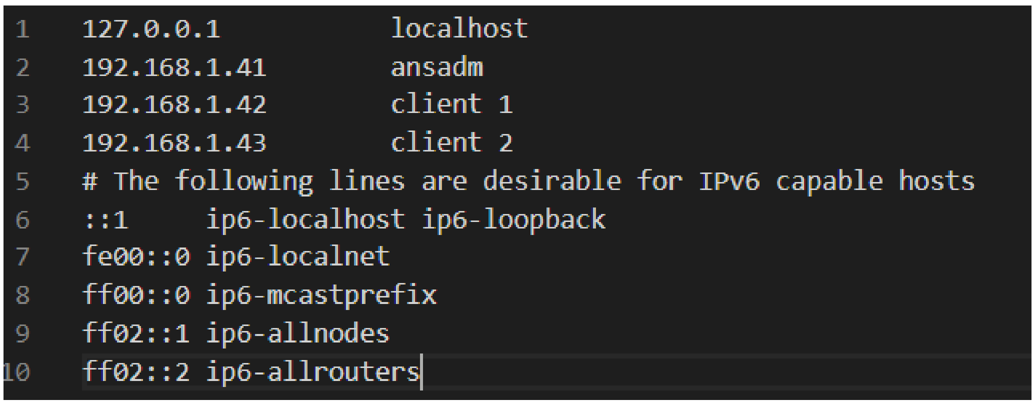
tasks: This attribute consists of the various required tasks.

All the mentioned tasks in a YAML file gets executed in a sequential manner in a top to bottom approach. In this way all the high priority tasks can be performed earlier than the others with the low priority.

Appendix-C: Editing the Host files

1.Go to /etc/hosts and open the hosts file with sudo privileges.

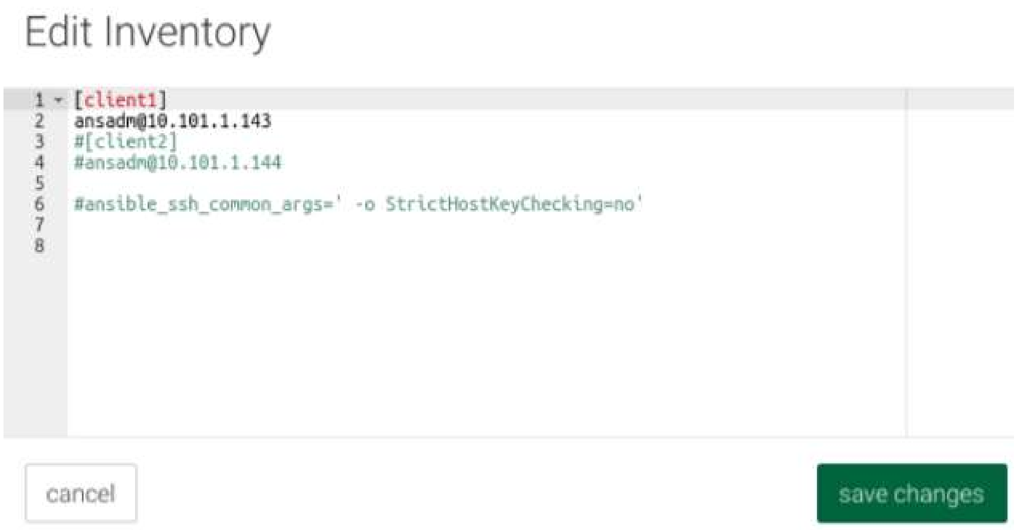
2.Add the IP addresses of the client machines in the network.



```
127.0.0.1           localhost
192.168.1.41        ansadm
192.168.1.42        client 1
192.168.1.43        client 2
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Here, ansadm is the host-name of the master-node i.e the ansible server and the rest of the machines are the client machines in the network.

3.Next, go to the inventory file of ansible located at /etc/ansible/hosts.



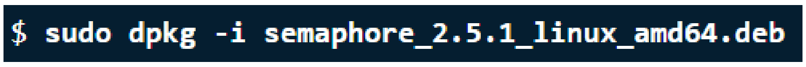
```
[client1]
ansadm@10.101.1.143
#[client2]
#ansadm@10.101.1.144

#ansible_ssh_common_args=' -o StrictHostKeyChecking=no'
```

This inventory file contains the names and IP addresses of the client machines which is secured through SSH tunnelling and is authenticated with the master-node i.e the ansible server machine at the earlier stage itself.

Appendix-D: Setting up the Ansible GUI Semaphore

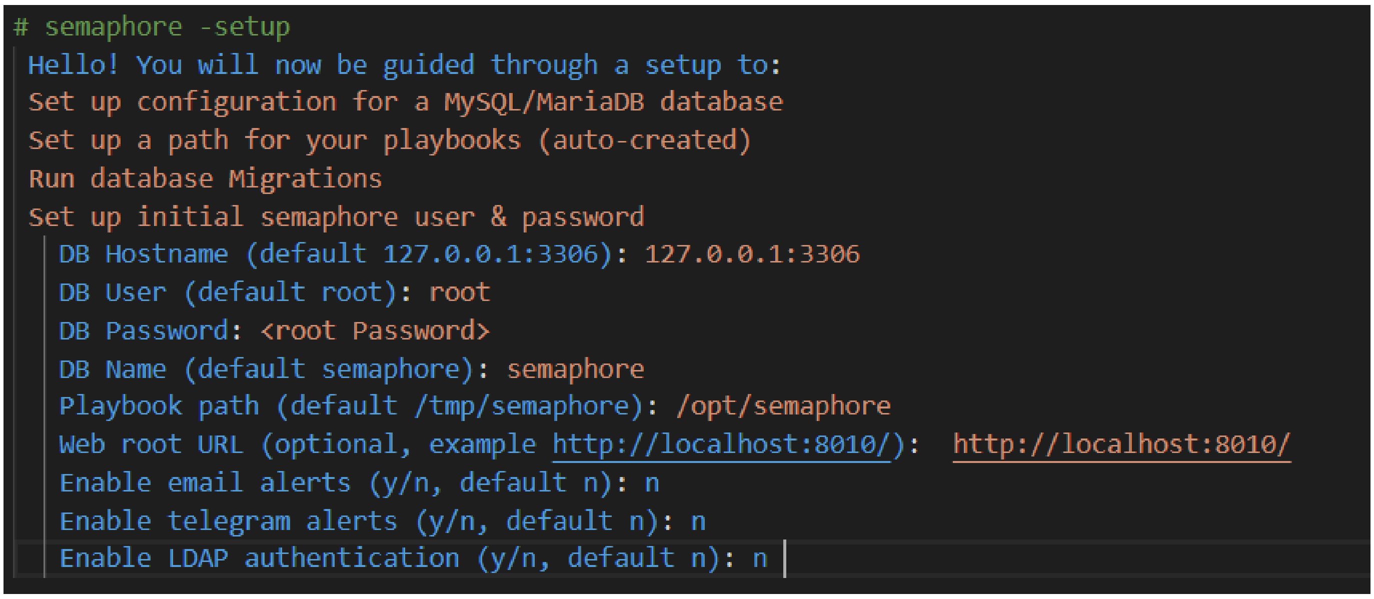
1.Download Semaphore from https://github.com/ansible-semaphore/semaphore/releases

2.Install semaphore



```
$ sudo dpkg -i semaphore_2.5.1_linux_amd64.deb
```

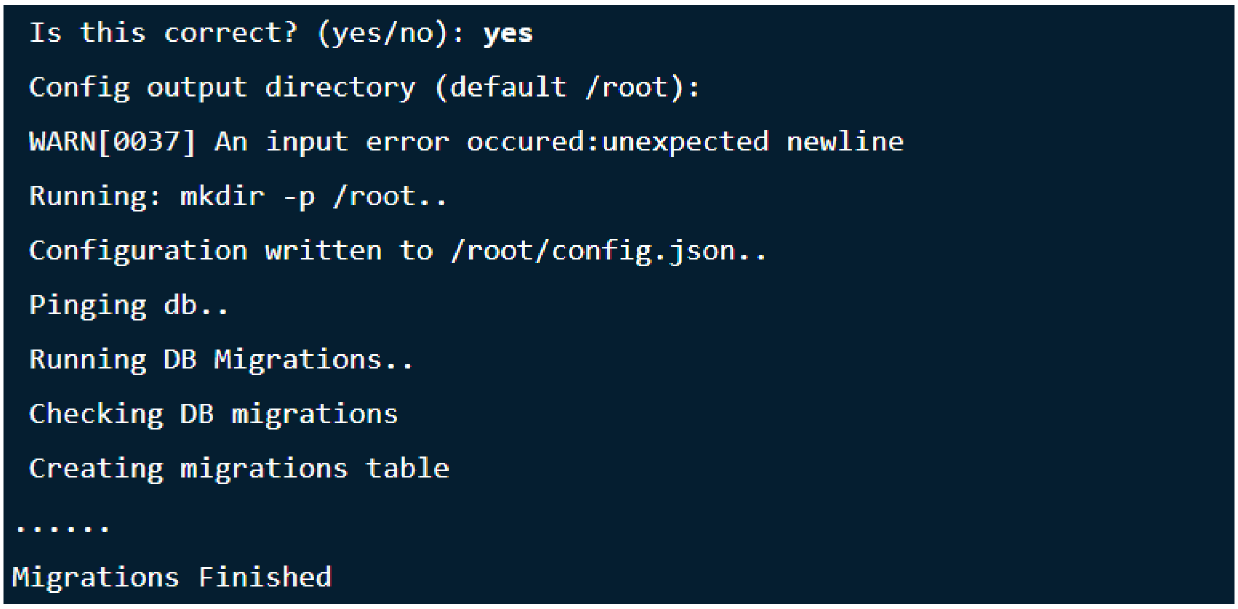
3.Setup Semaphore



```
# semaphore -setup
 Hello! You will now be guided through a setup to:
 Set up configuration for a MySQL/MariaDB database
 Set up a path for your playbooks (auto-created)
 Run database Migrations
 Set up initial semaphore user & password
   DB Hostname (default 127.0.0.1:3306): 127.0.0.1:3306
   DB User (default root): root
   DB Password: <root Password>
   DB Name (default semaphore): semaphore
   Playbook path (default /tmp/semaphore): /opt/semaphore
   Web root URL (optional, example http://localhost:8010/):  http://localhost:8010/
   Enable email alerts (y/n, default n): n
   Enable telegram alerts (y/n, default n): n
   Enable LDAP authentication (y/n, default n): n
```

4.Confirm generated configuration and Agree to start installation.

5.Confirm these values are correct to initiate setup.

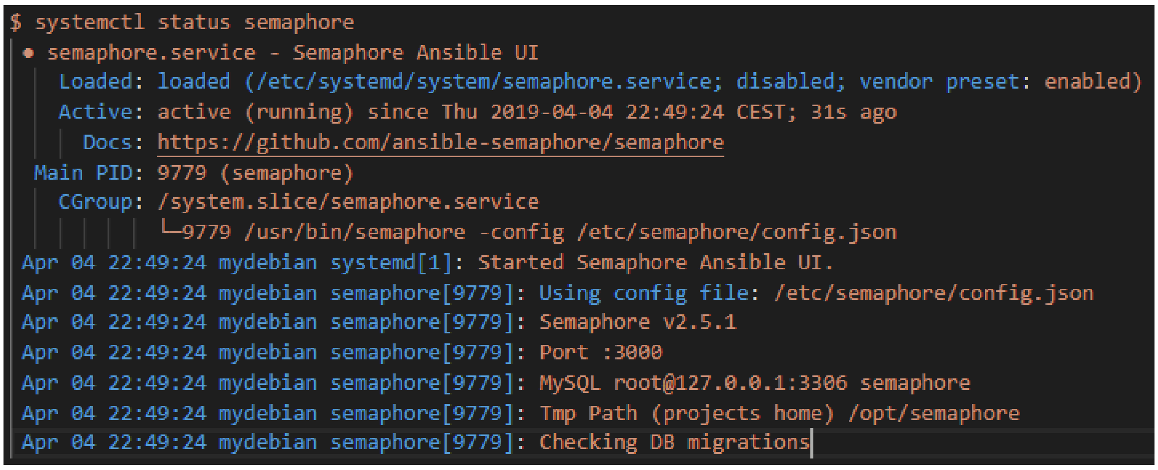


```
 Is this correct? (yes/no): yes
 Config output directory (default /root):
 WARN[0037] An input error occured:unexpected newline
 Running: mkdir -p /root..
 Configuration written to /root/config.json..
 Pinging db..
 Running DB Migrations..
 Checking DB migrations
 Creating migrations table
......
Migrations Finished
```

6.Start Semaphore Service

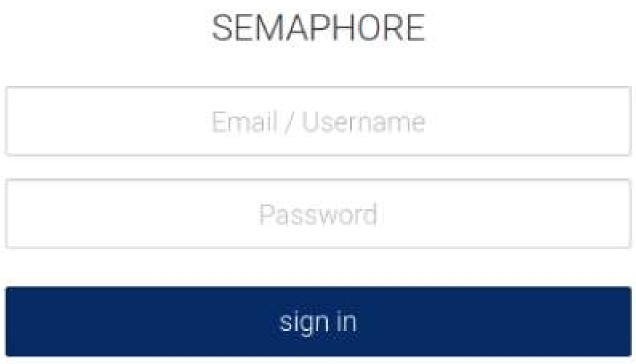
```
semaphore -config /root/config.json &
```

7.Check status to see if running



```
$ systemctl status semaphore
 ● semaphore.service - Semaphore Ansible UI
    Loaded: loaded (/etc/systemd/system/semaphore.service; disabled; vendor preset: enabled)
    Active: active (running) since Thu 2019-04-04 22:49:24 CEST; 31s ago
      Docs: https://github.com/ansible-semaphore/semaphore
  Main PID: 9779 (semaphore)
    CGroup: /system.slice/semaphore.service
            └─9779 /usr/bin/semaphore -config /etc/semaphore/config.json
 Apr 04 22:49:24 mydebian systemd[1]: Started Semaphore Ansible UI.
 Apr 04 22:49:24 mydebian semaphore[9779]: Using config file: /etc/semaphore/config.json
 Apr 04 22:49:24 mydebian semaphore[9779]: Semaphore v2.5.1
 Apr 04 22:49:24 mydebian semaphore[9779]: Port :3000
 Apr 04 22:49:24 mydebian semaphore[9779]: MySQL root@127.0.0.1:3306 semaphore
 Apr 04 22:49:24 mydebian semaphore[9779]: Tmp Path (projects home) /opt/semaphore
 Apr 04 22:49:24 mydebian semaphore[9779]: Checking DB migrations
```

8.Access Semaphore Web interface
On your web browser, open semaphore Server IP on port 3000.



Acknowledgement

We have great pleasure in presenting the report on **Autonetics and Administration for IT Laboratories.** We take this opportunity to express our sincere thanks towards our guide **Dr. Sameer Nanivadekar** & Co-Guide **Mr.Vishal Badgujar** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Mr. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Mr. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Student Name1:Uddhabendra Maity
Student ID1:16104062

Student Name2:Karthikeyan
Student ID2:16104050

Student Name3:Atharv Shetty
Student ID3:16104061

Publication

Paper entitled **“Autonetics and Administration for IT Laboratories”** is presented in **“International Conference on Convergence to Digital World (ICCDW- 2020)”** by **“Atharv Shetty,Karthikeyan Venkatachalam, Uddhabendra Maity, Dr. Sameer Nanivadekar and Mr. Vishal Badgujar”.**