

A PROJECT REPORT ON

Development of Fully Integrated Private Cloud using KVM as Virtualization

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY ,
PUNE IN THE FULFILLMENT OF THE REQUIREMENTS FOR THE
AWARD OF THE DEGREE

BACHELOR OF ENGINEERING
(Computer Engineering)

BY

Mr. Balram Pappu Chavan	(B120534223)
Mr. Aniket Anil Dalal	(B120534229)
Mr. Chatuphale Devesh Paresh	(B120534222)
Mr. Sachin Ashok Dhamal	(B120534356)

Under The Guidance of

Prof. A.V. Yenikar



DEPARTMENT OF COMPUTER ENGINEERING
ZEAL COLLEGE OF ENGINEERING AND
RESEARCH, PUNE-411 041
SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
2016 - 17



**ZES's Zeal College of Engineering and Research
DEPARTMENT OF COMPUTER ENGINEERING**

CERTIFICATE

This is to certify that the Project Entitled

**Development of Fully Integrated Private Cloud using
KVM as Virtualization**

Submitted by

Mr.Chavan Balram Pappu	(B120534223)
Mr.Dalal Aniket Anil	(B120534229)
Mr.Chatuphale Devesh Paresh	(B120534222)
Mr.Dhamal Sachin Ashok	(B120534356)

is a bonafide work carried out by students under the supervision of Prof. A. V. Yenikar and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering) Project.

Prof. A. V. Yenikar
Internal Guide
Dept. of Computer Engg.

Prof.Sunil Sangave
H.O.D
Dept. of Computer Engg.

Dr.A.M.Kate
Principal
ZES's Zeal College of Engineering and Research

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

Development of Fully Integrated Private Cloud using KVM as
Virtualization

Is successfully completed by

Mr.Balram Pappu Chavan	(B120534223)
Mr.Aniket Anil Dalal	(B120534229)
Mr.Chatuphale Devesh Paresb	(B120534222)
Mr.Sachin Ashok Dhamal	(B120534356)

at

DEPARTMENT OF COMPUTER ENGINEERING

(ZES's Zeal College of Engineering and Research)

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2016-2017

Prof. A. V. Yenikar
Internal Guide
Dept. of Computer Engg.

Prof.Sunil Sangave
H.O.D
Dept. of Computer Engg.

Abstract

Cloud computing is nowadays widely used technology. Various advanced technologies in the world are taking cloud computing very seriously as the new era for mobile as well as a steady computing environment is emerging.

Private cloud is great alternative to public cloud as it offers security of data and cost saving advantages. Automation of private cloud will save cost and human efforts. Automation can be achieved with help of opensource tools like chef and jenkins. Virtualization technologies can be utilized for efficient use of hardware. There are several Open Source hypervisors like KVM, Oracles Virtual box also offers virtualization techniques that can be used in Open Stack environment to build public and private clouds. In this system we will impliment a private cloud setup using open-stack and KVM used for virtualization to bring a new VM up and running with all the softwares installed in few minutes. This project will cover the tasks like Automating VM creation, Automating Infrastructure setup, Automating platform setup, Automating systems configuration, Continuous integration.

For the proposed system manpower and configuration time required for infrastructure setup is less as packer allows to create multiple virtual machines with same configuration.

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on 'Development of fully integrated private cloud environment using KVM as virtualization'.

*We would like to take this opportunity to thank our internal guide **Prof. A. V. Yenikar** for giving us all the help and guidance we needed. We are really grateful to her for her kind support. Her valuable suggestions were very helpful.*

*We are also grateful to **Prof. Sunil Sangve**, Head of Computer Engineering Department, Zeal College of Engineering and Research for his indispensable support and suggestions.*

*In the end our special thanks to **Mr. Ranjan Kumar, Mr. Muzeer Shiek and Mr. Santosh Daphale**(Benison Technologies Pvt. Ltd.) for providing guidelines for the project.*

Mr.Chavan Balaram Pappu

Mr.Dalal Aniket Anil

Mr.Chatuphale Devesh Paresh

Mr.Dhamal Sachin Ashok

(B.E. Computer Engineering)

Contents

1	Synopsis	1
1.1	Project Title	2
1.2	Project Option	2
1.3	Internal Guide	2
1.4	Sponsorship and External guide	2
1.5	Technical Keywords (As per ACM Keywords)	2
1.6	Problem Statement	2
1.7	Abstract	3
1.8	Goals and Objectives	3
1.9	Relevant mathematics associated with the Project	4
1.10	Names of Conferences / Journals where papers can be published	4
1.11	Plan of Project Execution	5
2	Technical Keywords	6
2.1	Area of Project	7
2.2	Technical Keywords	7
3	Introduction	8
3.1	Project Idea	9
3.2	Motivation of the Project	9
3.3	Literature Survey	10
3.4	Literature survey description	11
4	Problem Definition and scope	12
4.1	Problem Statement	13
4.1.1	Goals and objectives	13
4.1.2	Statement of scope	13
4.2	Major Constraints	13
4.3	Methodologies of Problem solving and efficiency issues	13
4.4	Outcome	14
4.5	Applications	14
4.6	Software Resources Required	14

4.7	Hardware Resources Required	14
5	Project Plan	15
5.1	Project Estimates	16
5.1.1	Reconciled Estimates	16
5.1.2	Project Resources	16
5.2	Risk Management w.r.t. NP Hard analysis	17
5.2.1	Risk Identification	17
5.2.2	Risk Analysis	17
5.2.3	Overview of Risk Mitigation, Monitoring, Management	18
5.3	Project Schedule	20
5.3.1	Project task set	20
5.3.2	Task network	20
5.3.3	Timeline Chart	21
5.4	Team Organization	21
5.4.1	Team structure	21
5.4.2	Management reporting and communication	22
6	Software requirement specification	23
6.1	Introduction	24
6.1.1	Purpose and Scope of Document	24
6.2	Usage Scenario	24
6.2.1	User profiles	24
6.2.2	Use-cases	25
6.2.3	Use Case View	26
6.3	Data Model and Description	26
6.3.1	Data Description	26
6.3.2	Data objects and Relationships	27
6.4	Functional Model and Description	27
6.4.1	Data Flow Diagram	27
6.4.2	Activity Diagram	29
6.4.3	Non Functional Requirements:	31
6.4.4	State Diagram:	31
6.4.5	Design Constraints	31
6.4.6	Software Interface Description	32
7	Detailed Design Document using Annexure A and B	33
7.1	Introduction	34
7.2	Architectural Design	34
7.3	Data design (using Annexures A and B)	34

7.3.1	Internal software data structure	34
7.3.2	Database description	35
7.4	Component Design	36
7.4.1	Class Diagram	36
8	Project Implementation	37
8.1	Introduction	38
8.2	Tools and Technologies Used	38
8.2.1	Openstack	38
8.2.2	Jenkins	38
8.2.3	Chef	39
8.3	Methodologies/Algorithm Details	39
8.3.1	The Algorithm	39
8.4	Verification and Validation for Acceptance	40
9	Software Testing	41
9.1	Type of Testing Used	42
9.1.1	Unit Testing	42
9.1.2	Integration Testing	42
9.1.3	System Testing	42
9.2	Test Cases and Test Results	43
10	Results	44
10.1	Indroduction	45
10.2	Input	45
10.2.1	Input Snap-Shot	45
10.3	Outputs	47
10.3.1	Output Snap-Shot	47
11	Deployment and Maintenance	48
11.1	Installation and un-installation	49
11.2	User help	49
12	Conclusion and Future Scope	50
12.1	Conclusion	51
12.2	Future Scope	52
12.2.1	Docker	52
	References	52
13	References	53
13.0.1	References	54

Annexure A Laboratory assignments on Project Analysis of Algorithmic Design	56
Annexure B Laboratory assignments on Project Quality and Reliability Testing of Project Design	59
B.1 Mathematical Model	60
B.1.1 System Description	60
B.1.2 UML Diagrams	61
B.2 Testing Cases performed	64
Annexure C Project Planner	66
C.1 Plan of Project Execution	67
Annexure D Reviewers Comments of Paper Submitted	68
Annexure E Plagiarism Report	70
E.1 Plagiarism report	71
Annexure F Term-II Project Laboratory Assignments	72
F.1 Assignment No. 01	73
F.1.1 Title	73
F.2 Assignment No. 02	75
F.2.1 Title	75
F.3 Assignment No. 03	79
F.3.1 Title	79
F.4 Assignment No. 04	87
F.4.1 Title	87
Annexure G Information of Project Group Members	89

List of Figures

1.1	Plan Of Execution	5
5.1	Project Planner	21
6.1	Use case diagram	26
6.2	Data flow diagram level 0	28
6.3	Data flow diagram level 1	29
6.4	Activity Diagram	30
6.5	State Transition Diagram	32
7.1	Architecture diagram	35
7.2	Class Diagram	36
10.1	Input Snap-Shot1	45
10.2	Input Snap-Shot2	46
10.3	Output Snap-Shot	47
B.1	Class Diagram	61
B.2	Activity Diagram	62
B.3	Design Flow Diagram 0	63
B.4	Design Flow Diagram 1	63
B.5	Architecture Diagram	64
C.1	Plan Of Execution	67
E.1	Plagiarism report from Paperrater.com	71
F.1	Login Window1	86
F.2	Login window2	86

List of Tables

3.1	Literature Survey	10
5.1	Time Estimate	16
5.2	Risk Table	17
5.3	Risk Probability definitions	18
5.4	Risk Impact definitions	18
6.1	Test Case 1	25
6.2	Test Case 2	25
6.3	Test Case 3	25
9.1	Test Cases	43
A.1	IDEA Matrix	57
B.1	Test Case 1	64
B.2	Test Case 2	65
B.3	Test Case 3	65
F.1	Review Paper	74
F.2	Implementation Paper	74
F.3	Review of Guide	74
F.4	Functional Testing	88
F.5	Usability Testing	88
F.6	Compatibility Testing	88
F.7	Security Testing	88
F.8	Security Testing	88

CHAPTER 1

SYNOPSIS

1.1 Project Title

Development of fully integrated private cloud environment using KVM as virtualization.

1.2 Project Option

Sponsored project

1.3 Internal Guide

Prof. A. V. Yenikar

1.4 Sponsorship and External guide

Benison Technologies Pvt. Ltd. - Mr.Ranjan Kumar

1.5 Technical Keywords (As per ACM Keywords)

- Openstack
- Virtualization
- Kernel Virtual Machine(KVM)
- Private Cloud

1.6 Problem Statement

Public cloud are being used around globe and issues are arising like security and cost. To address these issue private cloud deployment with desired functionality is one of the solution. Hence, development of fully integrated private cloud along with automation using kernel virtual machine can play a major role.

1.7 Abstract

Cloud computing is nowadays widely used technology. Various advanced technologies in the world are taking cloud computing very seriously as the new era for mobile as well as a steady computing environment is emerging.

Private cloud is great alternative to public cloud as it offers security of data and cost saving advantages. Automation of private cloud will save cost and human efforts. Automation can be achieved with help of opensource tools like chef and jenkins. Virtualization technologies can be utilized for efficient use of hardware. There are several Open Source hypervisors like KVM, Oracles Virtual box also offers virtualization techniques that can be used in Open Stack environment to build public and private clouds. In this system we will impliment a private cloud setup using open-stack and KVM used for virtualization to bring a new VM up and running with all the softwares installed in few minutes. This project will cover the tasks like Automating VM creation, Automating Infrastructure setup, Automating platform setup, Automating systems configuration, Continuous integration.

For the proposed system manpower and configuration time required for infrastructure setup is less as packer allows to create multiple virtual machines with same configuration.

1.8 Goals and Objectives

- To have a private cloud setup using open-stack and KVM used for virtualization.
- To bring a new VM up and running with all the softwares installed in few minutes.
- This project will cover the tasks like Automating VM creation, Automating infrastructure setup, Automating platform setup, Automating systems configuration, Continuous integration.

1.9 Relevant mathematics associated with the Project

System Description:

$S = s, e, X, Y, T$, success, failure

- s: Start state of the system
- e: End state of the system.
- X=input set: Dependencies and requirements of users.
- Y=output set : Fully integrated private cloud.
- T=tools used : openstack , Jenkins ,Chef
- Success : Fully integrated private cloud will be created with users requirement.
- Failure : System will fail to fulfill requirements of user due to certain parameters like connectivity failure,incompatibility.

1.10 Names of Conferences / Journals where papers can be published

- Central Universities or SPPU Conferences
- International Journal of General Science and Engineering Research
- International Journal of Advanced Research in Computer Engineering and Technology

1.11 Plan of Project Execution

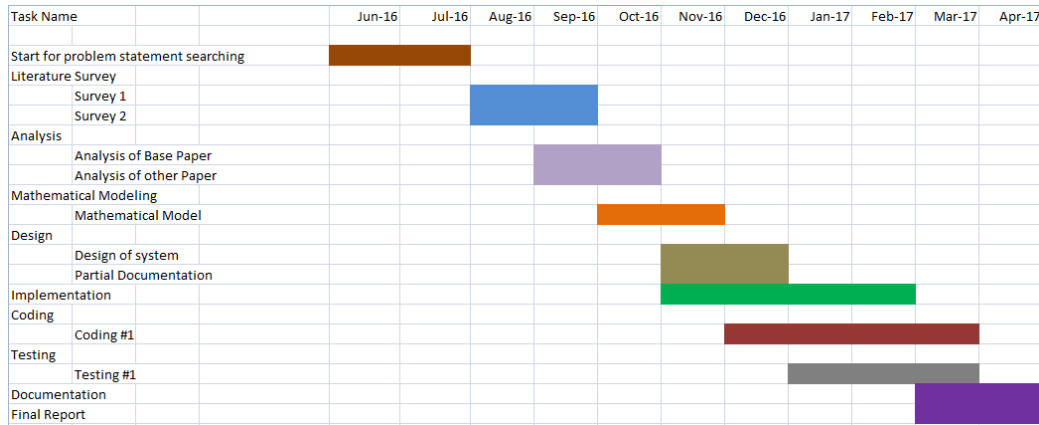


Figure 1.1: Plan Of Execution

CHAPTER 2

TECHNICAL KEYWORDS

2.1 Area of Project

Cloud computing and automation:

Cloud computing is nowadays widely used technology. Various advanced technologies in the world are taking cloud computing very seriously as the new era for mobile as well as a steady computing environment. In such a scenario many organizations are moving towards private cloud technologies and automation of infrastructure, platforms and software.

2.2 Technical Keywords

Openstack

Openstack is a free and open-source software platform which is used for cloud computing.

kernal virtual machine(KVM)

KVM is kernal based virtual machine. it is virtualization infrastructure which is acts as the hypervisor for linux kernals.

Private cloud

It is type of cloud clomputing which has same advantages as that of public cloud. It is dedicated to specific or single organization.

Virtualization

Virtualization is the process of creating virtual versions of devices, storages ,operating systems.

CHAPTER 3

INTRODUCTION

3.1 Project Idea

- Automating the tasks is the necessity of current era to reduce load and dependency on user
- Most time of user is wasted on installing supportive packages dependencies
- Automating tasks at infrastructure and application level can help reduce complexity
- We are going to provide a system having private cloud with automation.

3.2 Motivation of the Project

- Using public cloud services is costlier and more complex.
- Public cloud services also hold probability of failure,uncertainty and security issues.

3.3 Literature Survey

Sr.No	Year	Author Name	Paper Title	Paper Description
1	2016	Bruno Xavier, Tiago Ferreto, Luis Jersak	Time provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform	To Evaluate KVM, Docker and OSv(unikernel) and identify opportunities for optimization.
2	2016	Mr. Uchit Gandhi, Mr. Mitul Modi, Ms. Mitali Raval	Distributed Virtualization Manager for KVM Based Cluster	To develop a distributed, web-based manager for KVM based cluster.
3	2016	Jiuyuan Huo, Hong Qu, Ling Wu	Design and implementation of private cloud storage platform based on OpenStack	Python framework django is used.
4	2016	Marisol Garca-Valls, Tommaso Cucinotta, Chenyang Lu	Challenges in real-time virtualization and predictable cloud computing.	To identify technical challenges in supporting real-time applications in the cloud.
5	2016	Zhaojun Li Hai-jiang Li Xicheng Wang, Keqiu Li	A generic cloud platform for engineering optimization based on OpenStack Advances in Engineering Software	optimization process virtualization and cloud computing based implementation with innovative algorithms development.

Table 3.1: Literature Survey

3.4 Literature survey description

[1] The first paper Design and implementation of private cloud storage platform based on OpenStack.

This paper mainly aimed at implementation of openstack for development of private cloud platform on openstack for system feasibility. In this paper there are two methodologies used. The methodology uses openstack for the development of private cloud only using python framework django. However, implementation by this method provides services like upload and download of enterprise data only.

[2] Time provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform

This paper proposed the evaluation of KVM, Docker and OSv(unikernel) and identify opportunities for optimization. The methodologies used in this paper are,

1. Instance and operating system startup
2. Image creation
3. Openstack Overhead

[3] Distributed Virtualization Manager for KVM Based Cluster

This paper is based on the idea of developing a distributed, web-based manager for KVM based cluster. In this paper the hypervisor used is KVM and it is implemented for the distributed web based manager. Methodologies used in this paper are

1. Interacting with hypervisor libvirt
2. Web - based Manager

[4] Challenges in real-time virtualization and predictable cloud computing

This paper primarily aimed at recognising technical challenges in supporting real-time applications in the cloud environment. This paper has enlisted challenges like privacy, security, etc. However the number of challenges covered are not suffice.

CHAPTER 4

PROBLEM DEFINITION AND SCOPE

4.1 Problem Statement

Public cloud are being used around globe and issues are arising like security and cost. To address these issue private cloud deployment is one of the solution. Hence, development of fully integrated private cloud along with automation using kernel virtual machine can play a major role.

4.1.1 Goals and objectives

Goal and Objectives: To have a private cloud setup using open-stack and KVM used for virtualization to bring a new VM up and running with all the softwares installed in few minutes. This project will cover the tasks like Automating VM creation, Automating Infrastructure setup, Automating platform setup, Automating systems configuration, Continuous integration.

4.1.2 Statement of scope

Making private cloud automated and developing strong substitute to public cloud services.

4.2 Major Constraints

1. System should satisfy minimum hardware requirements.
2. The language with which user will interact is constraint to only one language that is english.

4.3 Methodologies of Problem solving and efficiency issues

Following steps explain the process,

1. Define the problem: Automation of private cloud.
2. Determine the root cause of the problem: To save time of user by automating tasks like adding dependencies and management, save cost of user by using existing hardware.

4.4 Outcome

Private cloud will be generated where user requirements and dependencies added implicitly using users existing hardware and leveraging virtualization provided by KVM multiple instances using existing hardware are created without spending more on hardware requirements.

4.5 Applications

1. Creating private cloud as per need using own hardware resources
2. Automating the tasks of configuration and installation.
3. Provide a flexible virtualization platform for optimum utilization of hardware resources

4.6 Software Resources Required

1. Operating System : GNU/Linux based distribution
2. Software tools :Openstack,Chef,Jenkins
3. Programming Language : Ruby
4. UI/front end : AngularJS, HTML

4.7 Hardware Resources Required

1. Processor - Intel i5 core
2. Hard disk -200 GB
3. RAM - 8 GB

CHAPTER 5

PROJECT PLAN

5.1 Project Estimates

Agile software development is a group of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.

5.1.1 Reconciled Estimates

5.1.1.1 Cost Estimate

This project is developed using open source tools which are freely available in the market. Only hardware used as per hardware requirements cost 55,000 in rupees.

5.1.1.2 Time Estimates

Task	Duration
Literature survey	2 months
Analysis and Mathematical Model	3 months
Design	3 months
Implementation and coding	3 months
Testing	1 month

Table 5.1: Time Estimate

5.1.2 Project Resources

1. Team of four people for development
2. Software tools - Chef, Openstack, Jenkins
3. Front end development framework-AngularJS, HTML

5.2 Risk Management w.r.t. NP Hard analysis

This section discusses Project risks and the approach to managing them. The risks for the Project can be analyzed within the constraints of time and quality

5.2.1 Risk Identification

For risks identification, review of scope document, requirements specifications and schedule is done.

1. The connectivity to be maintained among the frameworks is affected by disturbances in connectivity. Hence stable and constant connectivity is required.
2. Frameworks generally support almost all stable linux distribution releases but system incompatibility may occur in certain cases.

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Connectivity	Low	Low	High	High
2	unrealistic expectetions	medium	Low	High	High
3	System incompatibility	medium	Low	High	High

Table 5.2: 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 5.3: Risk Probability definitions

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

Table 5.4: Risk Impact definitions

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk.

Risk ID	1
Risk Description	Connectivity Failure
Category	Development Environment.
Source	Software requirement Specification document.
Probability	Low
Impact	High
Response	Mitigate
Strategy	Using defect free connection medium
Risk Status	Identified

Risk ID	2
Risk Description	Unrealistic expectations by end user
Category	Requirements
Source	Software Design Specification documentation review.
Probability	Low
Impact	High
Response	Mitigate
Strategy	Providing better knowledge to user using precised documents,manuals to resolve issue.
Risk Status	Identified

Risk ID	3
Risk Description	System incompatiblity
Category	Technology
Source	software requirement specification
Probability	Low
Impact	Very High
Response	Accept
Strategy	Verifying and checking software versions and compatibility features
Risk Status	Identified

5.3 Project Schedule

5.3.1 Project task set

Major Tasks in the Project stages are:

- Task 1: Documentation for end user
- Task 2: Developing front-end user interface with login
- Task 3: Configuring and coding for infrastructure automation on frameworks
- Task 4: Perform testing at respective stages of development

5.3.2 Task network

The Task 3 is dependent on the Task 2. The User interfaces will depend on the states that are placed in the flowchart and data flow diagrams in Task 2. The task 4 is dependent on tasks 2 and 3.

5.3.3 Timeline Chart

A project timeline chart is presented.

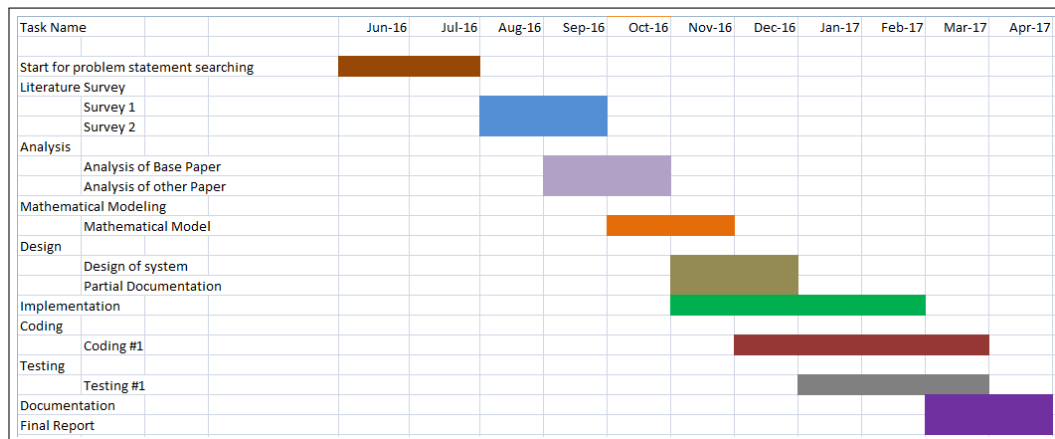


Figure 5.1: Project Planner

5.4 Team Organization

5.4.1 Team structure

Team consisting of four students as, 1. Balaram Chavan 2. Aniket Dalal 3. Devesh Chatuphale 4. Sachin Dhamal

Role of each in team is as follows,

1. Balaram Chavan- Worked on Jenkins for code deployment for automatic building.

2. Aniket Dalal- Worked on Openstack platform for creation of instances and management.

3. Devesh Chatuphale- Worked on Chef automation platform for infrastructure automation.

4. Sachin Dhamal- Worked on Development of front end UI using AngularJS and nodeJS.

5.4.2 Management reporting and communication

Mechanisms for progress reporting and inter/intra team communication are identified as per assessment sheet and lab time table. Progress points regarding to Project are maintained in Log Book. Remarks are taken from internal guide. Suggestions are noted given by internal guide.

CHAPTER 6

SOFTWARE REQUIREMENT SPECIFICATION

6.1 Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete system by defining the problem statement in detail. It also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features. The detailed requirements of the project are provided in this document

6.1.1 Purpose and Scope of Document

Purpose of the project is to develop private cloud using open source tools like openstack, jenkins, chef along with kernel virtual machine as hypervisor for virtualization so the document provides details about input, output, operations, probable risks and constraints

6.2 Usage Scenario

- 1) User will provide credentials (Login Id and password)
- 2) Server will verify credentials
- 3) User will request dependencies
- 4) Server will verify and if dependencies available or not
- 5) If dependencies available server will provide dependencies
- 6) Platform will be ready to run

6.2.1 User profiles

User or Actor: User can be technical person. User is that one entity which actually interacts with the software.

6.2.2 Use-cases

Test Case ID	1
Test Case Description	Generation of instance
Steps	create instance through jenkins
Test Case Result	Jenkins triggers to create instance of linux and returns an IP.
Expected Result	Instance should be up and running
Status	Pass

Table 6.1: Test Case 1

Test Case ID	2
Test Case Description	Installation of dependencies by chef
Steps	Chef will install dependencies on the instance
Test Case Result	Dependencies are installed
Expected Result	Dependencies should get installed without error
Status	Pass.

Table 6.2: Test Case 2

Test Case ID	3
Test Case Description	Application should be running on instance
Steps	Jenkins should trigger an application to run on the instance
Test Case Result	Application is running on instance
Expected Result	Application should be running on instance with desired IP
Status	Pass.

Table 6.3: Test Case 3

6.2.3 Use Case View

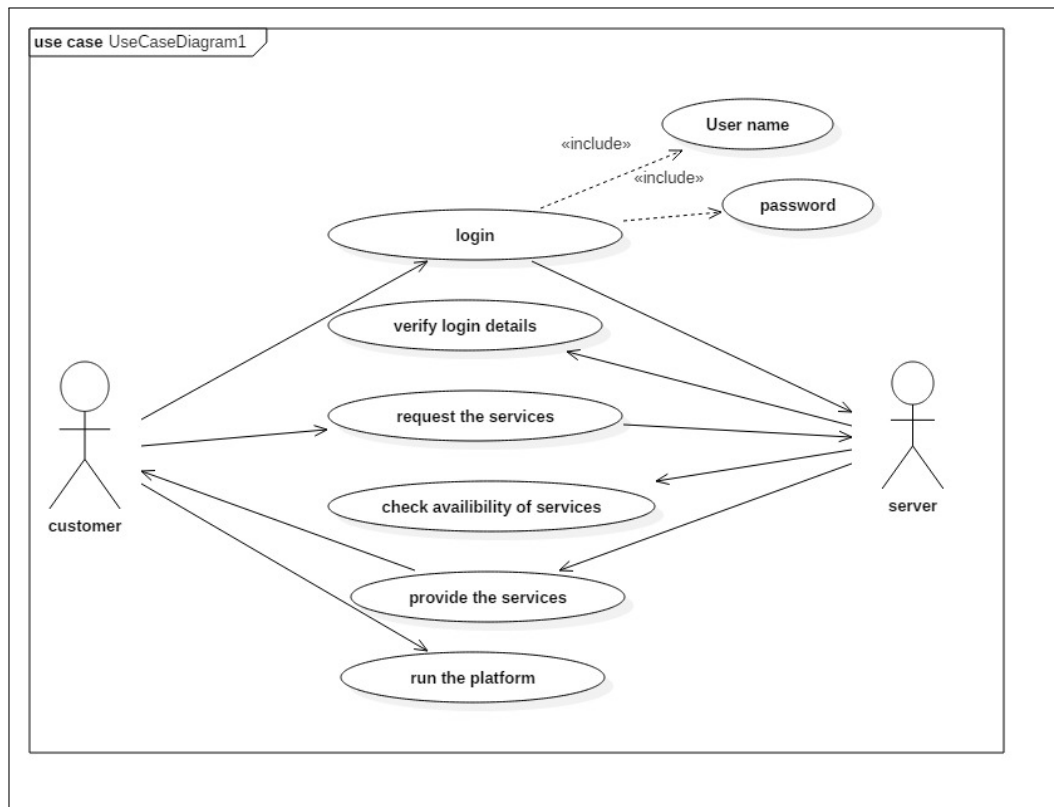


Figure 6.1: Use case diagram

6.3 Data Model and Description

6.3.1 Data Description

Data objects that will be managed/manipulated by the software are described in this section. The database entities or files or data structures required to be described. For data objects details can be given as below

6.3.2 Data objects and Relationships

Every Product consist of Practical Data Objects as well as Assumed Data Objects. Both of them put same impact on execution flow of product.

6.4 Functional Model and Description

The application has major classes that are specified below along with their attributes:

- Openstack class: It will create multiple instances as per the the necessity.
- Chef class :Chef will install dependencies as per the requirement
- Jenkins class :Jenkins will get the code from repository and deploy it on the instance

6.4.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

6.4.1.1 Level 0 Data Flow Diagram

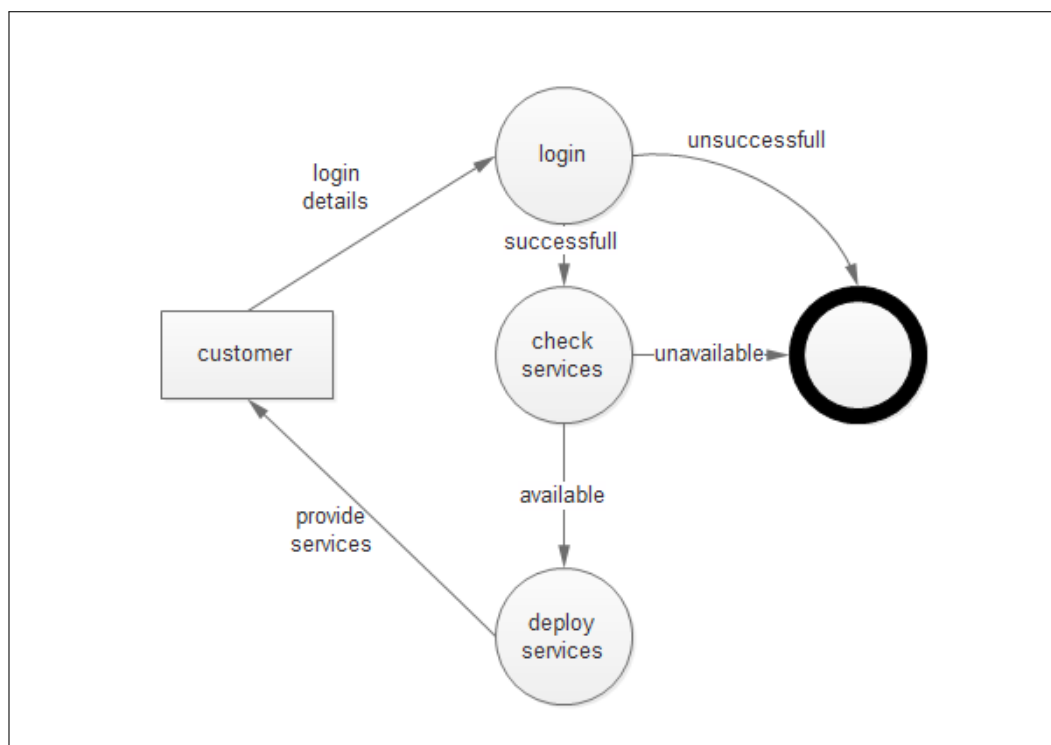


Figure 6.2: Data flow diagram level 0

6.4.1.2 Level 1 Data Flow Diagram

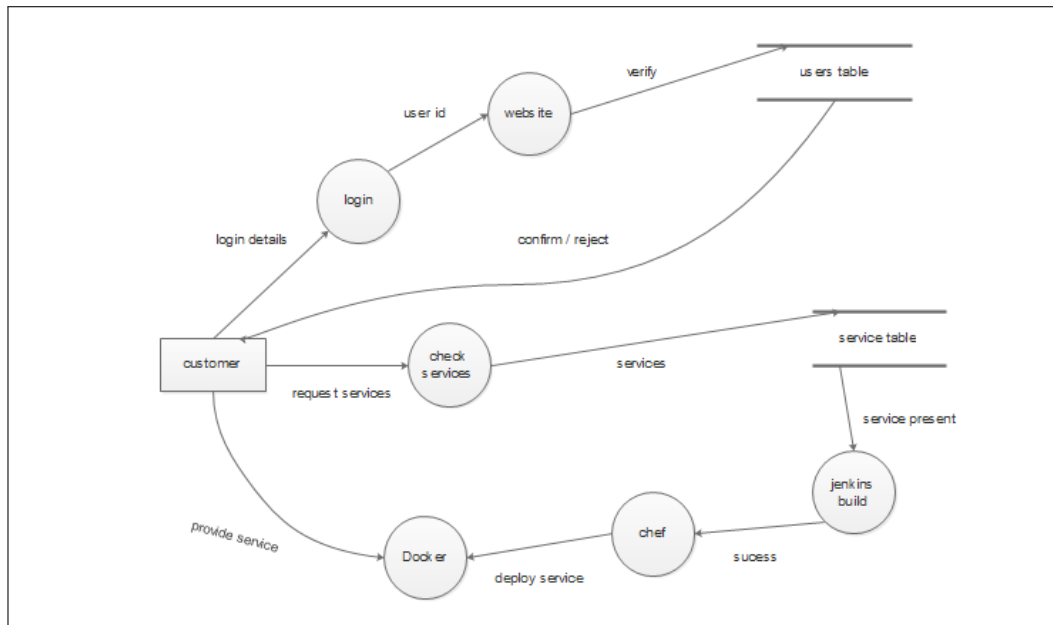


Figure 6.3: Data flow diagram level 1

6.4.2 Activity Diagram

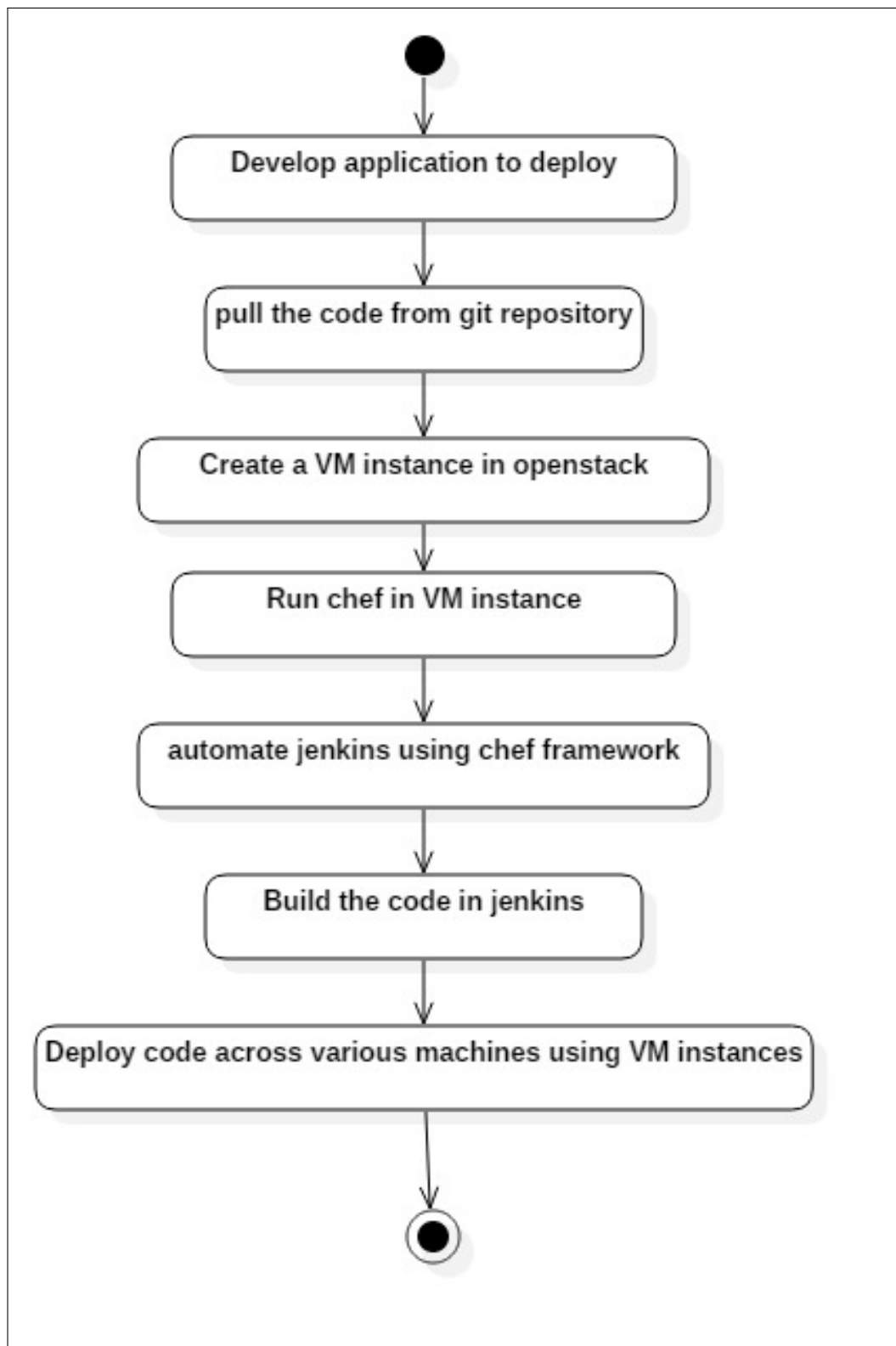


Figure 6.4: Activity Diagram

6.4.3 Non Functional Requirements:

- Interface Requirements

- 1) User Interface

- i. User Authentication
- ii. Material Design Interface

- 2) Hardware Interface

It will be a system with minimum required configuration through which user will interact with application.

- ii. It also requires good network connectivity.

- 3) Software Interface

Framework user interface with authentication

- Performance Requirements

The system should not get affected by the size of instance. The system should execute in minimum time.

- Safety Requirements

User data should be saved safely and should not be vulnerable.

- Security Requirements

Credentials should not be disclosed to anyone in any case.

- Software Quality Requirements

Interface should be excellent with respect to display resolution.

6.4.4 State Diagram:

State Transition Diagram

Diagram shows the state transition diagram of instance generation along with along with installing dependencies and deploying application.

6.4.5 Design Constraints

The server on which our system will be running should be compatible with the frameworks.

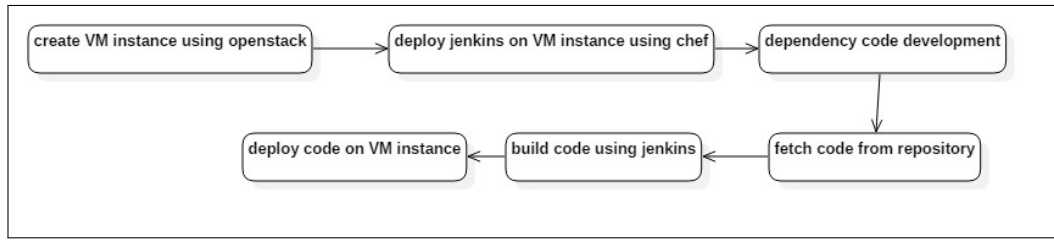


Figure 6.5: State Transition Diagram

6.4.6 Software Interface Description

Framework application: This feature allows the user to login with credentials and in return the IP address of the instance will be returned to user.

CHAPTER 7
DETAILED DESIGN DOCUMENT
USING ANNEXURE A AND B

7.1 Introduction

The private cloud with automated features can be a strong substitute to public clouds with soar pricings. This automation can be obtained by implementing step by step automation using frameworks Jenkins and Chef. User can develop such integrated private cloud on his or her existing hardware with much more security and flexibility.

7.2 Architectural Design

The flow of control passes through following blocks

- Jenkins: This framework first triggers openstack to create instances if linux operating system
- Openstack: Openstack creates the instances of linux with necessary hardware resources allocated.
- Chef: Chef will install all necessities on openstack instance that are required for end point application to run.

7.3 Data design (using Annexures A and B)

The description of all data structures are as follows:

7.3.1 Internal software data structure

- Array: Arrays works as collections of items, for instance strings. You can use them to gather items in a single group, and perform various operations on them, e.g. sorting. Besides that, several methods within the framework work on arrays, to make it possible to accept a range of items instead of just one. This fact alone makes it important to know a bit about arrays.
- Stack: Whenever a procedure is executed, its activation record is stored on the stack, also known as control stack. The main task behind the algorithm converting the stack-based code is to identify dependencies among operations. And it is conditional and unconditional jumps that make hard to figure these dependencies. So the code without them can be transformed into the three-address code in a straightforward way.

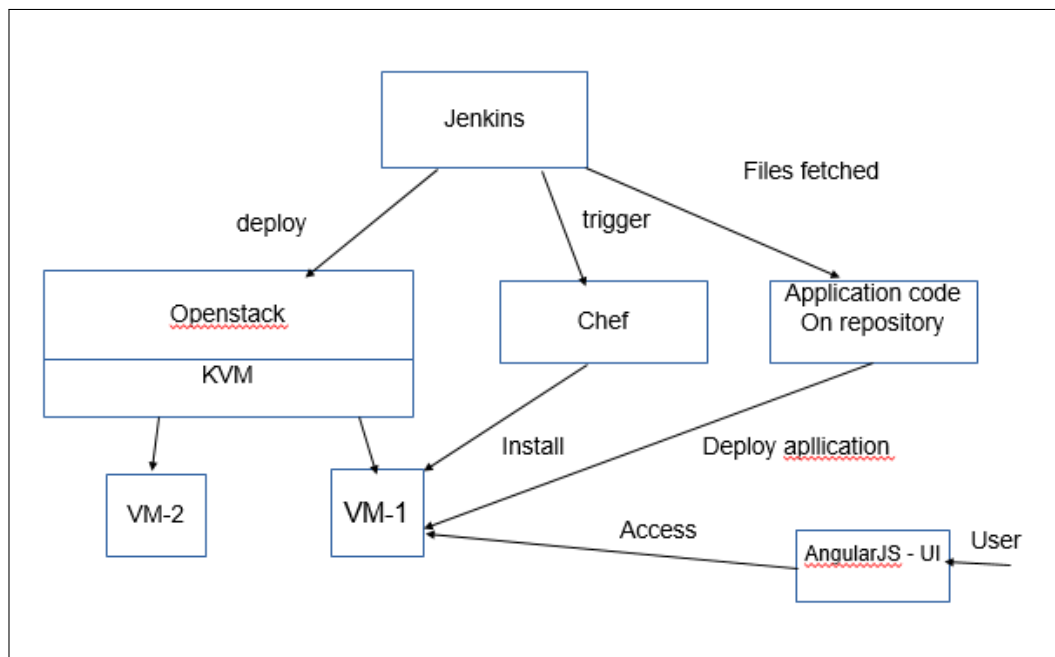


Figure 7.1: Architecture diagram

7.3.2 Database description

User has to login before accessing the instance. Hence database such as sqllite is used to store user credentials. The database is also utilised by application running on the instance

7.4 Component Design

7.4.1 Class Diagram

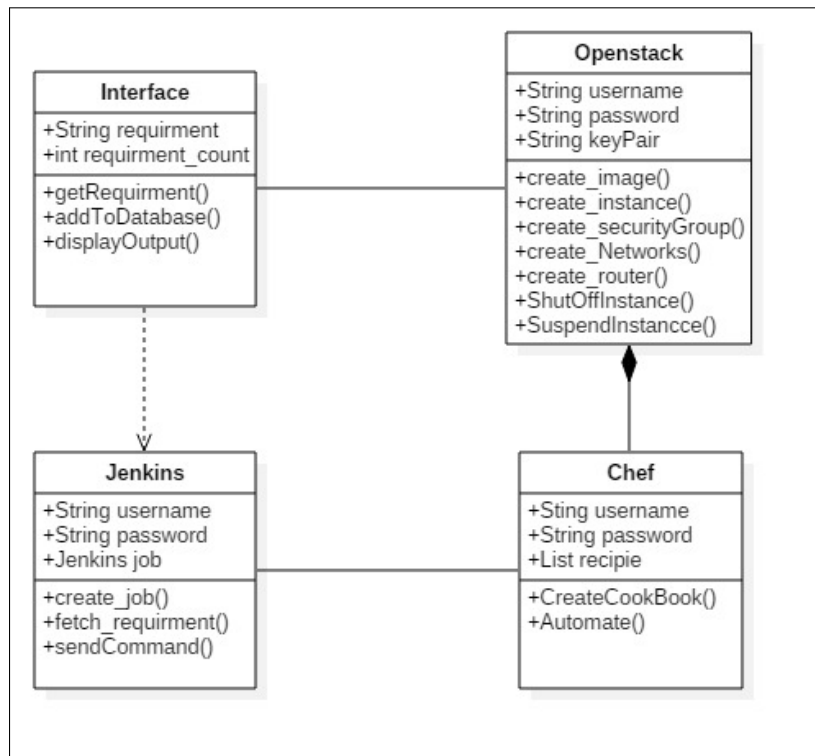


Figure 7.2: Class Diagram

CHAPTER 8

PROJECT IMPLEMENTATION

8.1 Introduction

This project is developed to create private cloud instances within few clicks and with less efforts to user. For this several tools are used which bring automation in certain stages and at the end a coherent framework that performs the functionalities mentioned earlier.

8.2 Tools and Technologies Used

8.2.1 Openstack

Openstack is open source software for creating private and public cloud. OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the Openstack API. Any organization or individual can use openstack for building their own cloud environment (IaaS). Openstack use modularized design in which core modules are Swift (Object storage service), Nova (Computing service), Neutron (Networking services) and Keystone (Database service). Storage service supports storage of object, replication of data, archieving data and massive data access service. Computation service provide managment tool for running instances, user and network. Mirror service allow storage and managment of virtual machines.

As per today for developing a private cloud using openstack 3-tier platform design was used in which Python Django was used for interface, Web server layer provides middleware to handle user access and swift service provide API to programming language for operations such as creation, modification, deletion and access to stored object.

8.2.2 Jenkins

Jenkins is an cross-platform, continuous integration and delivery application that increases productivity. Continuous Integration is the practice of running your tests on a non-developer machine automatically every time someone pushes new code into the source repository. This has advantage of always knowing if all tests work and getting fast feedback. The fast feedback is important so we always know right after you broke the build . what we did that failed and how to revert it. If we only run your tests occasionally the problem is that a lot of code changes may have happened since the last time and it is rather hard to figure out which change introduced the problem.

When it is run automatically on every push then it is always pretty obvious what and who introduced the problem. Built on top of Continuous Integration are Continuous Deployment/Delivery where after a successful test run instantly and automatically release the latest version of codebase. Makes deployment a non-issue and helps to speed up development.

8.2.3 Chef

Chef server enables to manage different server on cloud. chef framework is used for purpose of automation along with Jenkins. CHEF technology has got three different working modes: 1) traditional chef-client to chef-server connection 2) chef-client to chef-server connection simulation, by connecting to a local process (chef zero) that reads from file system all required information (cookbooks, recipes associations with nodes, environments, roles, data bags) 3) chef-solo client execution that does not connect to any kind of server with some limitation respect to previous modes.

8.3 Methodologies/Algorithm Details

8.3.1 The Algorithm

- 1 Start Jenkins server.
- 2 Connect to OpenStack using plugin and credentials.
- 3 Launch Linux instance with required hardware resources.
- 4 Connect to Chef server using Jenkins plugin.
- 5 Install all necessary dependencies and softwares on running instance using Chef server.
- 6 Fetch application source code from Bitbucket using Jenkins
- 7 Deploy application on instance using Jenkins
- 8 Provide IP address of the system to end user
- 9 User access the application using the IP of instance running on cloud

8.4 Verification and Validation for Acceptance

Verification and validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "independent", indicating that the verification and validation is to be performed by a disinterested third party. "Independent verification and validation" can be abbreviated as "IVV".

CHAPTER 9

SOFTWARE TESTING

9.1 Type of Testing Used

9.1.1 Unit Testing

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output.

9.1.2 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated IT) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

9.1.3 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

9.2 Test Cases and Test Results

Test case ID	Test scenario	Expected Results	Actual results	Status
01	Check User Login with valid data	User should Login into application	Login Successful	Pass
02	Check User Login with invalid Data	User should not Login into application	Message : Invalid username or password.	Pass
03	Check openstack ip is accessible from browser	IP should be accesible	IP accesible	Pass
04	Check if chef cookbooks uploaded on server	Cookbooks should be displayed TextBox	Cookbooks displayed in verification command	Pass
05	Check instance get launced on openstack	Instance should get launched	Instance created and lauched successfully	Pass
06	Check connectivity between Jenkins and Openstack	Connection should be accesible	Connection established successfully	Pass
07	Check if Jenkins jobs can be created	Jobs should get created	Jobs created Successfully	Pass
08	Check connectivity between Jenkins and Chef	Connection should get established	Connection established successfully	Pass

Table 9.1: Test Cases

CHAPTER 10

RESULTS

10.1 Indroduction

The end user will have an application running on the cloud which can be accessible anywhere within the organization wiht effective as well as flexible virtualization. User has to input credentials for login to access instances.

10.2 Input

10.2.1 Input Snap-Shot

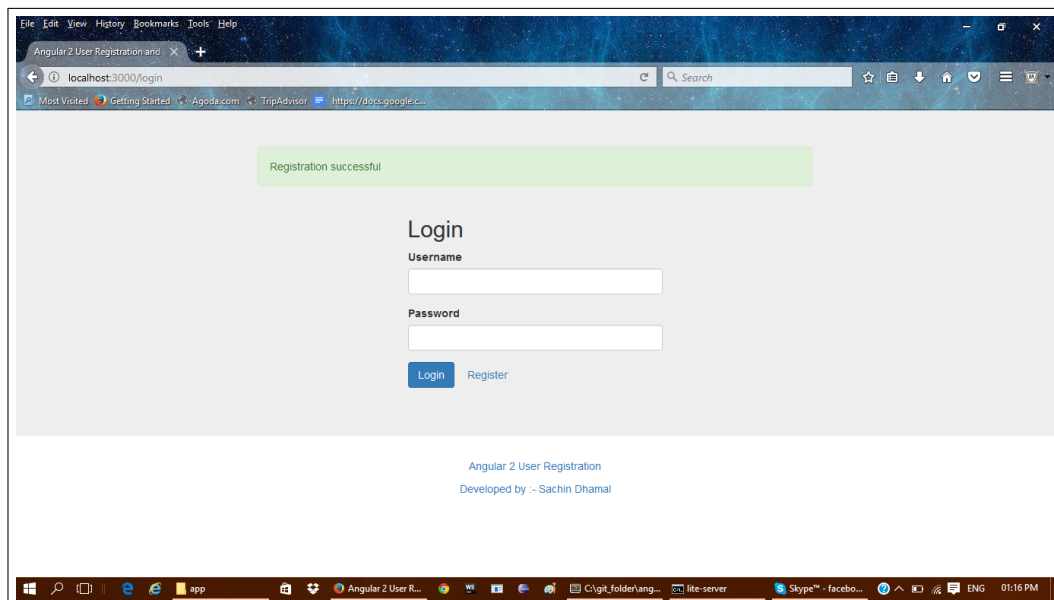


Figure 10.1: Input Snap-Shot1

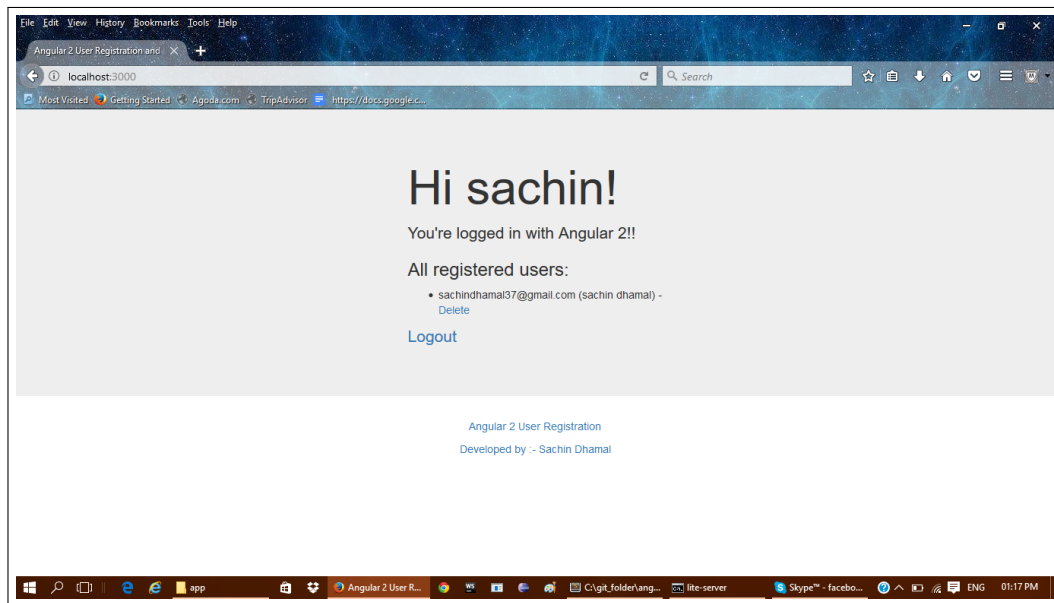


Figure 10.2: Input Snap-Shot2

10.3 Outputs

10.3.1 Output Snap-Shot

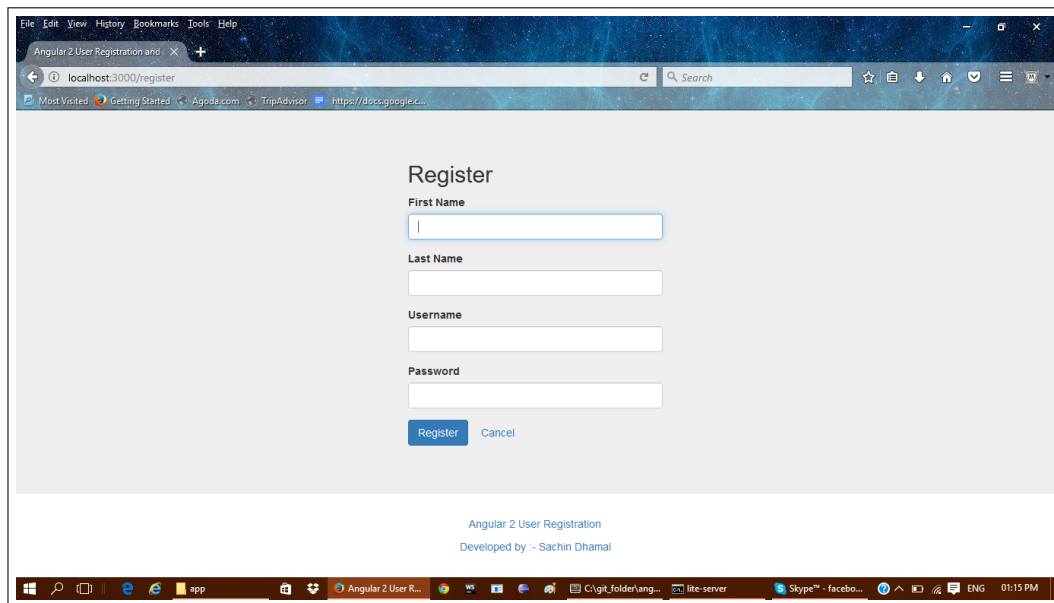


Figure 10.3: Output Snap-Shot

CHAPTER 11

DEPLOYMENT AND MAINTENANCE

11.1 Installation and un-installation

User does not need to install anything on his side. User's local machine should have a web browser which is generally installed by default on operating systems.

11.2 User help

When user will enter his or her credential and get signed in ,he or she has to enter hardware requirements as well as specifications of softwares to be installed or configuration details etc.

After this when openstack cloud instances along with specifications will get created and user will get an IP address at which the openstack services are residing . User can access openstack instances through console provided by openstack after logging in.

CHAPTER 12
CONCLUSION AND FUTURE
SCOPE

12.1 Conclusion

Cloud services offer flexible services such as PaaS, IaaS and SaaS. Use of cloud has been increased as it provides on demand services and it is highly scalable. Many organization use public cloud to obtain cloud services. Users of public cloud are facing issues such as privacy and cost. This issues can be addressed by implementing private cloud.

Automation of private cloud will play a important role in reducing cost and human efforts. User will get rid of hectic tasks like cloud instance creation, allocating resources etc. Automation can be implemented using open source automation tools and private cloud development softwares. For automation purpose tools like chef and jenkins can be used. Virtualization techniques will allow effective utilization of hardware resources. Virtualization can be effectively archived using KVM(kernel virtual machine) for utilization of existing hardware.

12.2 Future Scope

The instances created required to be compatible with public cloud like Amazon Web Services(AWS).Live migration of instances should be possible with ease.Also the instances should be loosely coupled for effective flexibility.The system can be enhanced by extending it by making use of container tools like Docker.

12.2.1 Docker

Docker is an open-source platform for management and development of Linux Containers, XEN and KVM. In Docker, containers are built on top of images decoupled in a multi-layer lesystem model. Docker allows to build and share disk images. This architecture consists on multiple levels, one upon another in a design that resembles a version control system. It uses a technique called copy-on-write. From the point of view of a container, all layers are unied in a single vision. However, the write operation is allowed only on the top layer.

CHAPTER 13

REFERENCES

13.0.1 References

- [1] Antonio Corradi, Mario Fanelli , Luca Foschini, VM consolidation: A real case based on OpenStack Cloud, ScienceDirect, Future Generation Computer Systems 32 ,Vol.32, No. , pp. 118127,2014.
- [2] Bruno Xavier,Tiago Ferreto ,Luis Jersak ,Time provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform , IEEE, International Symposium on Cluster Cloud and Grid Computing,Vol.16, No., pp.277-280, 2016.
- [3] M. Boschetti, Vito.baglio, P. Ruiiu, O. Terzo, A Cloud automation platform for flexibility in applications and resources Provisioning, Ninth International Conference on Complex, Intelligent and Software Intensive Systems, Vol., No. , pp 204 208, 2015.
- [4] Mr. Uchit Gandhi, Distributed Virtualization Manager for KVM Based Cluster, International Conference on Communication Computing and Virtualization ,Procedia Computer Science 79, Vol. ,No., pp.182 189, 2016.
- [5] Marisol Garca-Valls, , Tommaso Cucinotta, Chenyang Lu,Challenges in real-time virtualization and predictable cloud computing, Journal of Systems Architecture 60 ,Vol., No., pp.726740,2014.
- [6] Zhaojun Li ,Haijiang Li ,Xicheng Wang,Keqiu Li, A generic cloud platform for engineering optimization based on OpenStack, Advances in Engineering Software 75,Vol., No., pp.4257, 2014.
- [7] Jiuyuan Huo, Hong Qu, Ling Wu, Design and implementation of private cloud storage platform based on OpenStack, IEEE International Conference on Smart City/SocialCom/SustainCom together with DataCom 2015 and SC2 2015,Vol., No., pp.1098 1101, 2015.
- [8] Jlio Proa no Orellana, Maria Blanca Caminero, Carmen Carrion, On the provision of SaaS-level Quality of Service within Heterogeneous Private Clouds, IEEE/ACM 7th International Conference on Utility and Cloud Computing,Vol., No., pp. 146 155, 2014.
- [9] Abdulrahman Azab, Diana Domanska, Software Provisioning Inside a Secure Environment as Docker Containers using STROLL File-system, 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Comput-

ing, Vol., No., pp .674 683, 2016.

[10] Anshu Awasthi, Ravi Gupta, Multiple hypervisor based Open Stack cloud and VM migration, 6th International Conference - Cloud System and Big Data Engineering, Vol., No., pp .130 134, 2016.

ANNEXURE A
LABORATORY ASSIGNMENTS ON
PROJECT ANALYSIS OF
ALGORITHMIC DESIGN

- To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.
For IDEA Matrix and Knowledge canvas model. Case studies are given in this book. IDEA Matrix is represented in the following form. Knowledge canvas represents about identification of opportunity for product. Feasibility is represented w.r.t. business perspective.

	I		D		E		A
I	Increase: Security in cloud	D	Drive: Automaton of cloud	E	Educate: Virtualization for hardware utilization	A	Accelerate: Deployment, installation of VMs
I	Improve: User experience & ease	D	Deliver: Application on cloud	E	Evaluate: Use of frameworks in development process	A	Associate: Virtualization & cloud
I	Ignore:Public cloud	D	Decrease: Development cost & Time	E	Eliminate: uncertainty in cloud computing	A	Avoid:using public clouds

Table A.1: IDEA Matrix

- Project problem statement feasibility assessment using NP-Hard, NP-Complete or satisfy ability issues using modern algebra and/or relevant mathematical models.
- Project problem statement feasibility assessment using NP-Hard, NP-Complete
or satisfy ability issues using modern algebra and/or relevant mathematical models.
Our Problem is an P-type problem that can be proved by following points:
 1. Private cloud can be created using the existing hardware resources using virtualization.

2. Every problem state is a deterministic state.
3. Even at the occurrences of error, the output generated will be definite, based upon the nature of error occurring.
4. There won't be any situation where decision making has to be done by the system and any significant change is not dependent on it. Hence, it is a P-type problem.

- Mathematical Model:

- Input : User credentials for login
- Output: An IP address of the instance with an application running on it.
- Functions :
 - Launch() - Launch the instance of linux on existing hardware.
 - Install() - Install dependencies on the system using chef.
 - Display() - Display the IP address of the instance on which the desired application is running.
- Success Conditions: The instance with IP address will get created with an application running on it.
- Failure Conditions: Instance will fail to launch.

ANNEXURE B
LABORATORY ASSIGNMENTS
ON PROJECT QUALITY AND
RELIABILITY TESTING OF
PROJECT DESIGN

B.1 Mathematical Model

B.1.1 System Description

- Input : User credentials for login
- Output: An IP address of the instance with an application running on it.
- Functions :
 - Launch() - Launch the instance of linux on existing hardware.
 - Install() - Install dependencies on the system using chef.
 - Display() - Display the IP address of the instance on which the desired application is running.
- Success Conditions: The instance with IP address will get created with an application running on it.
- Failure Conditions: Instance will fail to launch.

B.1.2 UML Diagrams

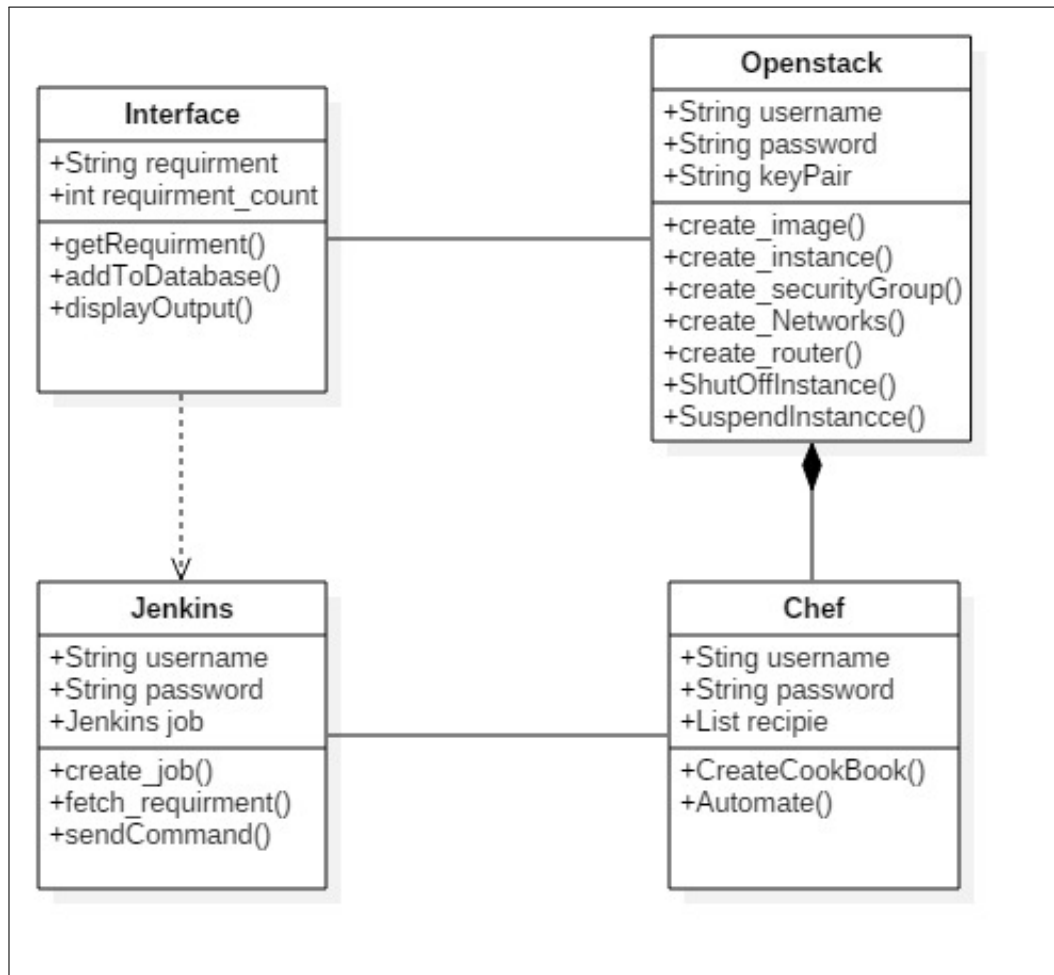


Figure B.1: Class Diagram

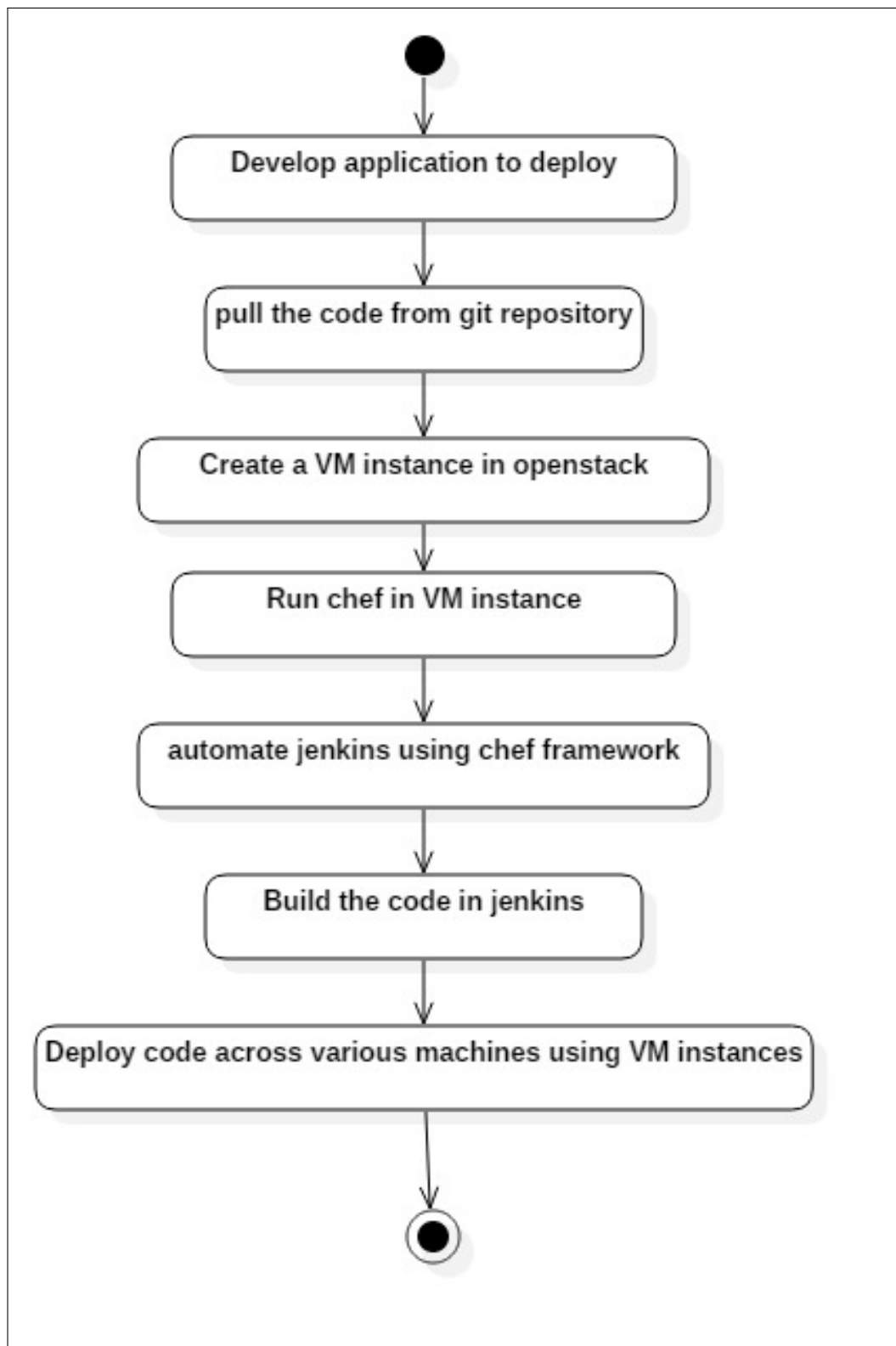


Figure B.2: Activity Diagram

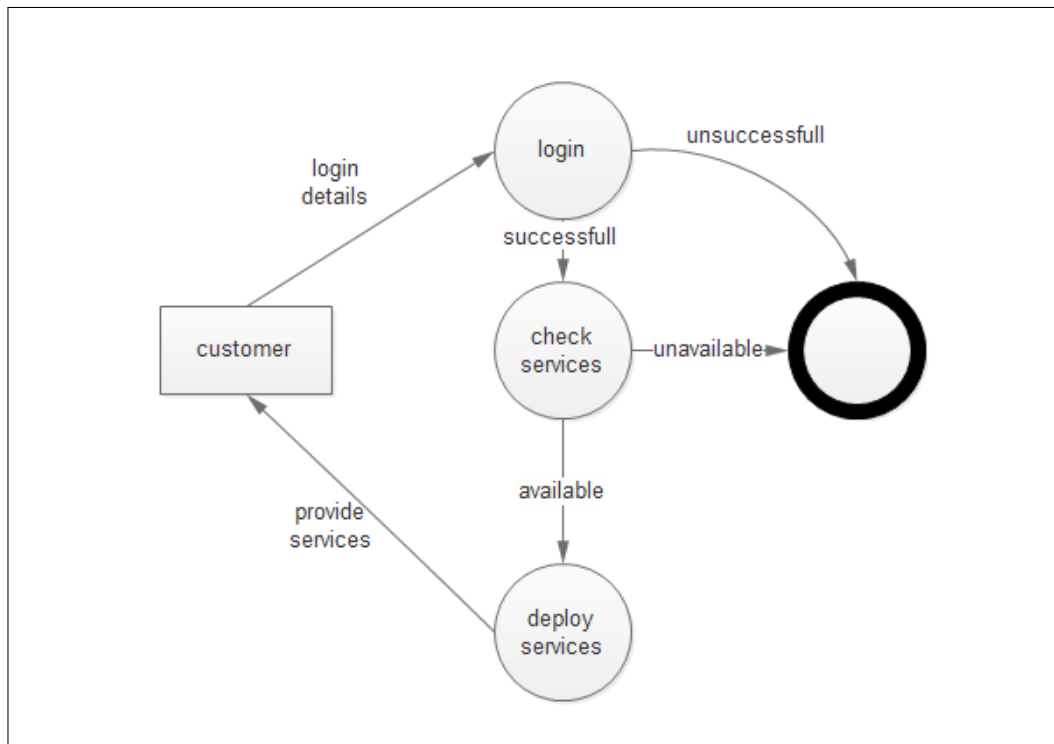


Figure B.3: Design Flow Diagram 0

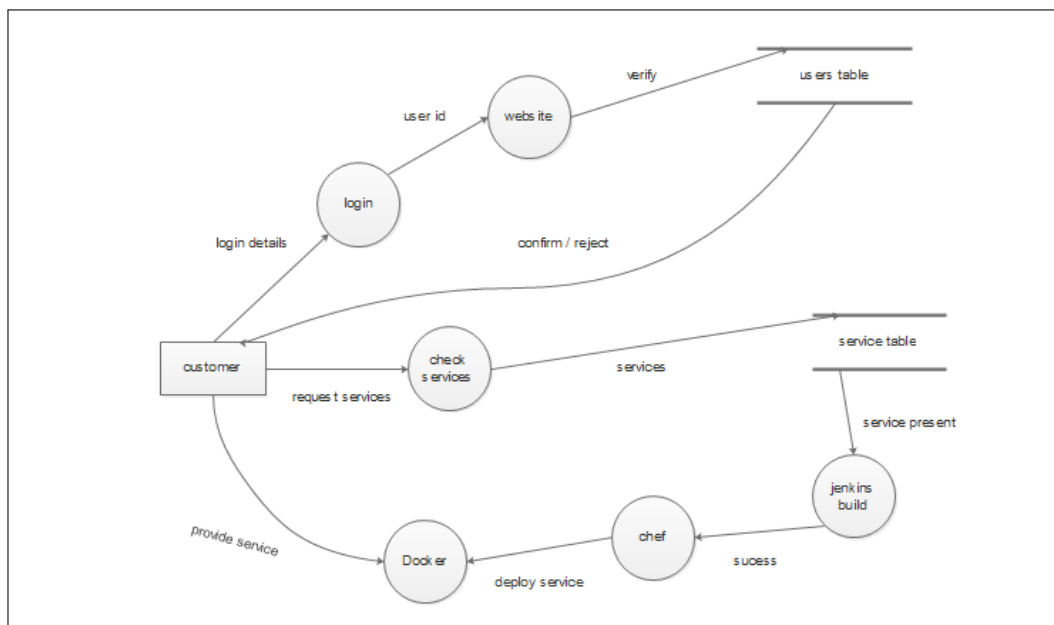


Figure B.4: Design Flow Diagram 1

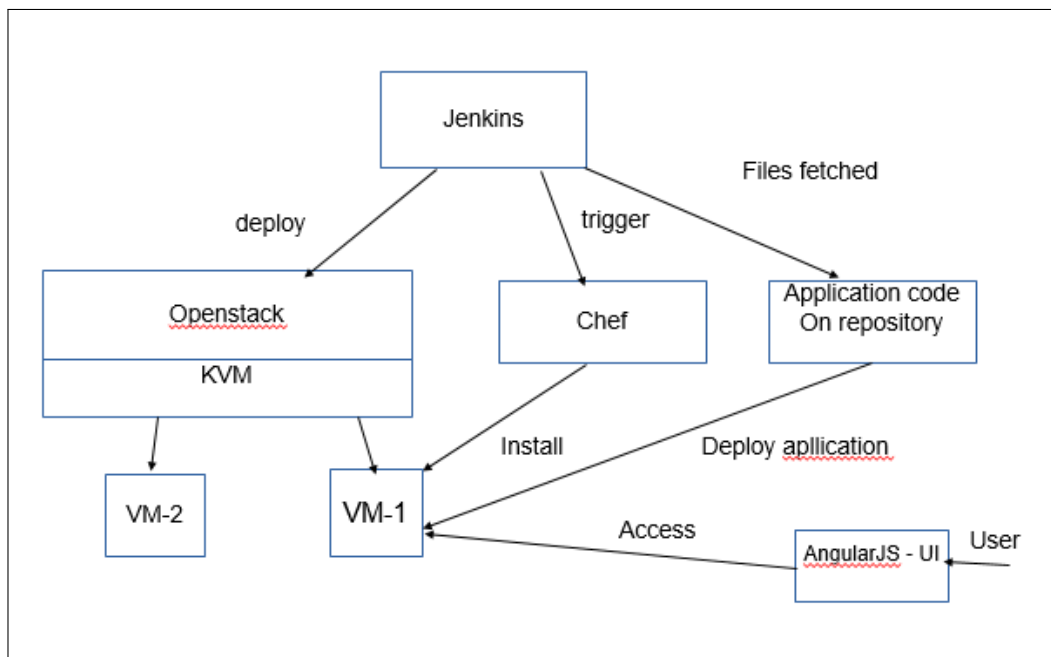


Figure B.5: Architecture Diagram

B.2 Testing Cases performed

Test Case ID	1
Test Case Description	Launching instnace of linux
Steps	Instance should get launched by triggering jenkins
Test Case Result	Instance get launched successfully
Action Result	Instance is up and running
Status	Pass.

Table B.1: Test Case 1

Test Case ID	2
Test Case Description	Installing dependencies by chef
Steps	Chef server installs dependencies
Test Case Result	Dependencies should get installed
Action Result	Dependencies installed Successfully
Status	Pass

Table B.2: Test Case 2

Test Case ID	3
Test Case Description	User login verification
Steps	User should login with credentials
Test Case Result	User should login with correct details
Action Result	User logged in successfully
Status	Pass

Table B.3: Test Case 3

ANNEXURE C
PROJECT PLANNER

C.1 Plan of Project Execution

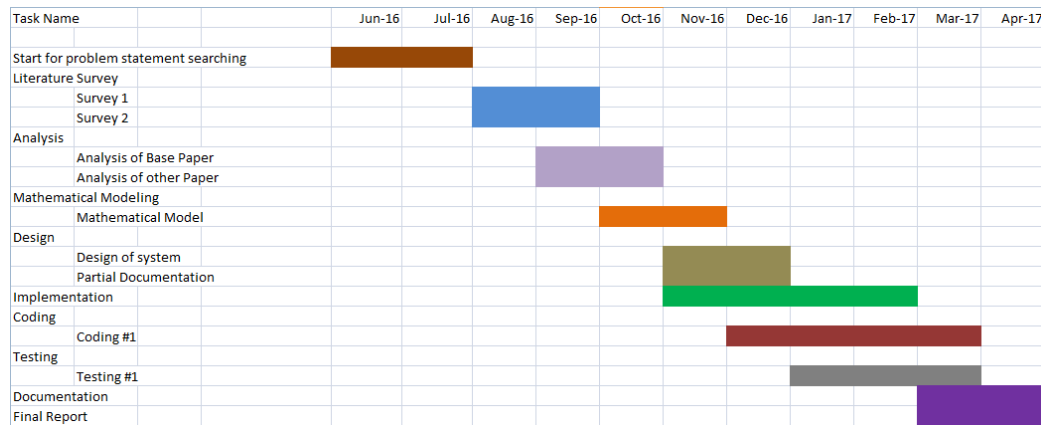


Figure C.1: Plan Of Execution

ANNEXURE D
REVIEWERS COMMENTS OF
PAPER SUBMITTED

1. Review Paper in Semester VII


- (a) Paper Title:Development of integrated private cloud
- (b) Name of the Conference/Journal where paper submitted :
International Journal of modern Computer Science and Applications(IJMCSA)
- (c) Paper accepted/rejected : Accepted
- (d) Review comments by reviewer :Published in
- (e) Corrective actions if any : not available

2. Implementation paper in semester VIII


- (a) Paper Title:Development of integrated private cloud using KVM
- (b) Name of the Conference/Journal where paper submitted :
International Journal of modern Computer Science and Applications(IJMCSA)
- (c) Paper accepted/rejected : Accepted
- (d) Review comments by reviewer :Published in
- (e) Corrective actions if any : not available


ANNEXURE E
PLAGIARISM REPORT

E.1 Plagiarism report


 **PaperRater** (<http://www.PaperRater.com>)

1. Openstack Openstack is open source community managed software for creating and managing private and public cloud. OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the Openstack API. Any organization or individual can use openstack for building their own cloud environment (IaaS). Openstack use modularized design in which core modules are Swift (Object storage service), Nova (Computing service), Neutron (Networking services) and Keystone (Database service). Storage service supports storage of object,replication of data, archieving data and massive data access service. Computation service provide managment tool for running instances, user and network. Mirror... (only first 800 chars shown)

 **Analysis complete.** Our feedback is listed below in printable form. Some of the items have been truncated or removed to provide better print compatibility.

 **Plagiarism Detection**

Original Work
Originality: 100%

 **This paper does not exhibit any signs of plagiarism.** Well done!

For more information on our originality scoring process, click here (<http://www.PaperRater.com/page/plagiarism-detection>) .

Percentage of document containing quotes: 0.0%
No quotations were found in this document.

Figure E.1: Plagiarism report from Paperrater.com

ANNEXURE F
TERM-II PROJECT
LABORATORY ASSIGNMENTS

F.1 Assignment No. 01

F.1.1 Title

Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment, and other competitions/conferences participated like IIT, Central Universities, University Conferences or equivalent centers of excellence etc.

Name of Journal	Status
International Journal of modern Computer Science and Applications (IJMCSA)	Published

Table F.1: Review Paper

Name of Paper	Status
International Journal of modern Computer Science and Applications (IJMCSA)	Published

Table F.2: Implementation Paper

Sr. No	Corrective actions specied	Actual actions performed
1	Changes in References format	Made corrections in diagrams as per suggested by Guide.
2	In synopsis use of technical words according to ACM Keywords with the provided link	ACM technical keywords used.
3	Changing fonts in architecture diagram	Font changed to readable format

Table F.3: Review of Guide

F.2 Assignment No. 02

F.2.1 Title

Project workstation selection, installations along with setup and installation report preparations.

Software requirements are as follows

1. Openstack -Openstack is open source community managed software for creating and managing private and public cloud. OpenStack software controls large pools of compute, storage, and networking resources throughout a data center, managed through a dashboard or via the Openstack API.
2. Jenkins -Jenkins is an open source,cross-platform, continuous integration and delivery application that increases productivity.
3. Chef -Chef is a configuration management tool widely used in DevOps technology. Chef server enables to manage different nodes present in the network.

- Insatallation steps

1. Openstack -
 - 1) Install git

```
sudo apt-get install git
```

- 2) Clone the DevStack repository into your computer and cd into it. This is the code which will set up the cloud for you.

```
git clone http://github.com/openstack-dev/devstack  
cd devstack/
```

- 3) Check with ls command you will see a stack.sh file

- 4) Execute the stack.sh script

```
./stack.sh
```

This will take 10-15 minute, depending on Internet speed.

This is your host IP address: localhost
Horizon is now available at <http://localhost/dashboard>
Keystone is serving at <http://localhost/identity/>
The default users are: admin and demo
The password: 123
2016-11-13 20:00:56.473 — WARNING:
2016-11-13 20:00:56.473 — Using lib/neutron-legacy is deprecated,
and it will be removed in the future
2016-11-13 20:00:56.473 — stack.sh completed in 1880 seconds.
aniket@ubuntu: /devstack

2. Chef -

1. Download chefdk*.deb package from www.chef.io/chef
2. Then use command - `sudo dpkg i chefdk*.deb`

To install chef-server on Ubuntu 14.04 LTS

1. Download chef-server*.deb package from www.chef.io/chef
2. Then use command - `sudo dpkg -i chef-server.deb`
3. Run the command to start all of the services - `sudo chef-server-ctl reconfigure`

To verify installed components of chef use command - `chef verify`

3. Jenkins - Jenkins Installation Steps in Ubuntu :

1. Open the terminal and type following command
2. Firstly we add key apt using following command
`wget -q -O - http://pkg.jenkins-ci.org/debian-stable/jenkins-ci.org.key`
— `sudo apt-key add`
3. Secondly create sources list for jenkins
`sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ & /etc/apt/sources.list.d/jenkins.list'`
4. Update the apt cache using following command


```
sudo apt-get update
```

5. Install the jenkins

```
sudo apt-get install jenkins
```

6. Now restart the jenkins service

```
sudo service jenkins restart
```

7. To open the jenkins dashboard

Open browser and type `http://localhost:8080`

8. Enter the password from file `/var/lib/jenkins/secrets/initialAdminPassword` when asked

4. KVM -

KVM only works if your CPU has hardware virtualization support.

To check

```
egrep -c '(svm|vmx)' /proc/cpuinfo
```

0-not support

1 or more- support

To install graphical Virt-Manager

```
sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```

Only the root user and users in the libvirtd group have permission to use KVM virtual machines

```
sudo adduser name libvirtd
```

To check list of virtual machines

```
virsh -c qemu:///system list
```

F.3 Assignment No. 03

F.3.1 Title

Programming of the project functions, interfaces and GUI.

Login Code:
Index.html

```

<!DOCTYPE html>
<html>
<head>
<base href="/" />
<title>Angular 2 User Registration and Login Example</title>
<meta name="viewport"
content="width=device-width, initial-scale=1">

<!-- bootstrap css -->
<link href="//netdna.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
rel="stylesheet" />

<!-- application css -->
<link href="app.css" rel="stylesheet" />

<!-- polyfill(s) for older browsers -->
<script src="node_modules/core-js/client/shim.min.js"></script>
<script src="node_modules/zone.js/dist/zone.js"></script>
<script src="node_modules/systemjs/dist/system.src.js"></script>

<script src="systemjs.config.js"></script>
<script>
System.import('app').catch(function
(err) console.error(err));
</script>
</head>
<body>
<app>
<Loading...</app>
</body>
</html>

register.component.html
<div class="col-md-6 col-md-offset-3">
<h2>Register</h2>
<form name="form"
(ngSubmit)="f.form.valid & register()" f="ngForm" novalidate>
<div class="form-group" [ngClass]="{'has-error': f.submitted & !username.valid}">
<label for="firstName">First Name</label>

```

```



```

home.component.html

```

<div class="col-md-6 col-md-offset-3">
<h1><Hi currentUser.firstName!</h1>
<p><You're logged in with Angular 2!!</p>

```

```

<h3>All registered users:</h3>
<ul>
<li *ngFor="let user of users">
  user.username (user.firstName user.lastName)
  - <a (click)="deleteUser(user.id)">Delete</a>
</li>
</ul>
<p><a [routerLink]="['/login']">Logout</a></p>
</div>

```

```

main.ts
import platformBrowserDynamic from '@angular/platform-browser-dynamic';

```

```

import AppModule from './app.module';

```

```

platformBrowserDynamic().bootstrapModule(AppModule);

```

```

appcomponent.ts
import Component from '@angular/core';

```

```

@Component(
  moduleId: module.id,
  selector: 'app',
  templateUrl: 'app.component.html'
)
export class AppComponent

```

```

approuting.ts
import Routes, RouterModule from '@angular/router';

```

```

import HomeComponent from './home/index';
import LoginComponent from './login/index';
import RegisterComponent from './register/index';
import AuthGuard from './guards/index';

```

```

const appRoutes: Routes = [
  path: '', component: HomeComponent, canActivate: [AuthGuard] ,
  path: 'login', component: LoginComponent ,

```

```
path: 'register', component: RegisterComponent ,
```

```
// otherwise redirect to home  
path: '**', redirectTo: "  
];
```

```
appmodule.ts
```

```
export const routing = RouterModule.forRoot(appRoutes);  
import NgModule from '@angular/core';  
import BrowserModule from '@angular/platform-browser';  
import FormsModule from '@angular/forms';  
import HttpClientModule from '@angular/http';
```

```
// used to create fake backend  
import fakeBackendProvider from './helpers/index';  
import MockBackend, MockConnection from '@angular/http/testing';  
import BaseRequestOptions from '@angular/http';
```

```
import AppComponent from './app.component';  
import routing from './app.routing';
```

```
import AlertComponent from './directives/index';  
import AuthGuard from './guards/index';  
import AlertService, AuthenticationService, UserService from './services/index';  
import HomeComponent from './home/index';  
import LoginComponent from './login/index';  
import RegisterComponent from './register/index';
```

```
@NgModule(  
  imports: [  
    BrowserModule,  
    FormsModule,  
    HttpClientModule,  
    routing  
  ],  
  declarations: [  
    AppComponent,  
    AlertComponent,
```

```

    HomeComponent,
    LoginComponent,
    RegisterComponent
  ],
  providers: [
    AuthGuard,
    AlertService,
    AuthenticationService,
    UserService,

    // providers used to create fake backend
    fakeBackendProvider,
    MockBackend,
    BaseRequestOptions
  ],
  bootstrap: [AppComponent]
)

export class AppModule

logincomponent.ts
import Component, OnInit from '@angular/core';
import Router, ActivatedRoute from '@angular/router';

import AlertService, AuthenticationService from '../services/index';

@Component(
  moduleId: module.id,
  templateUrl: 'login.component.html'
)

export class LoginComponent implements OnInit
model: any = ;
loading = false;
returnUrl: string;

constructor(
private route: ActivatedRoute

```

```
, private router: Router,
private authenticationService: AuthenticationService,
private alertService: AlertService)
```

```
ngOnInit() // reset login status
this.authenticationService.logout();
```

```
// get return url from route parameters or default to '/'
this.returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';
```

```
login()
this.loading = true;
this.authenticationService.login(this.model.username, this.model.password)
.subscribe(
data =>
this.router.navigate([this.returnUrl]);
,
error =>
this.alertService.error(error);
this.loading = false;
);
```

homecomponent.ts

```
import Component, OnInit from '@angular/core';
```

```
import User from '../models/index';
import UserService from '../services/index';
@Component({moduleId: module.id, templateUrl: 'home.component.html'})
export class HomeComponent implements OnInit { users: User[] = []; constructor(private userService: UserService) {} }
```

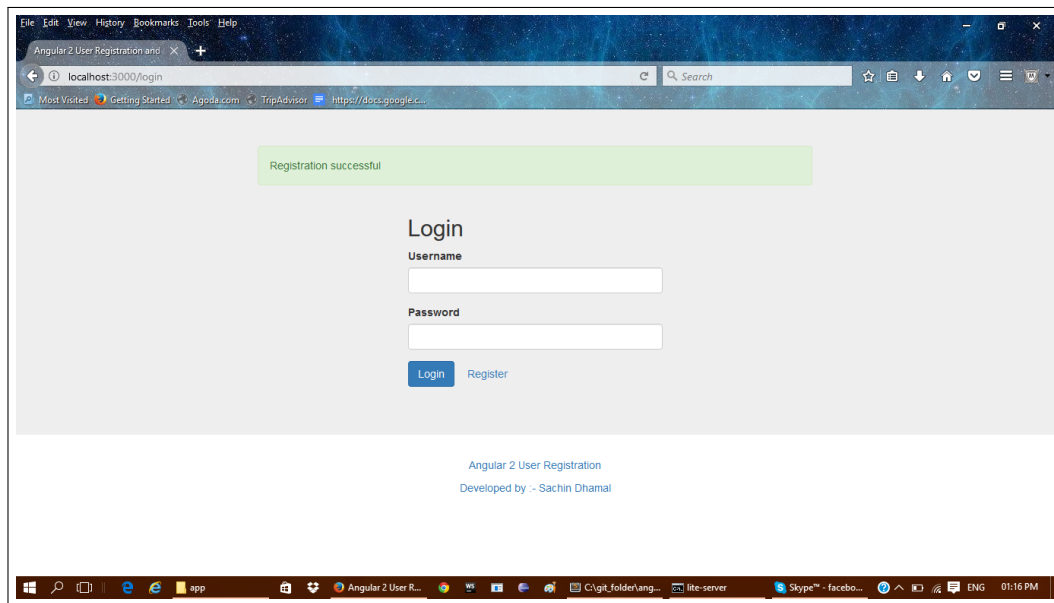



Figure F.1: Login Window1

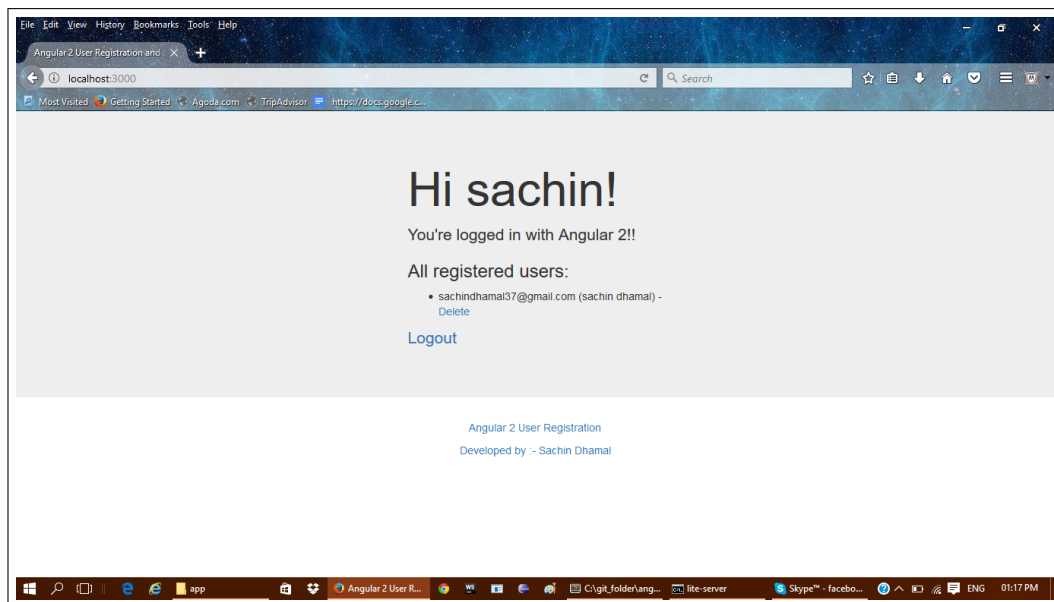


Figure F.2: Login window2

F.4 Assignment No. 04

F.4.1 Title

Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.

Table F.4: Functional Testing

Sr. no	Test Condition	Expected Result	Actual Result	Status
1	Deployment of application	The client should get the message about successful login	The success message displayed	Pass
2	Provides Admin login	The administrator of the application should be able to control all the operations.	The admin panel displayed successfully	Pass

Table F.5: Usability Testing

Sr. no	Test Condition	Status
1	Buttons ,Textboxes must be readable and consistent in their representation	Pass
2	No spelling and grammatical errors should be present	Pass
3	In case of errors informative prompts should be generated.	Pass
4	In case of errors informative prompts should be generated.	Pass

Table F.6: Compatibility Testing

Sr. no	Test Condition	Status
1	The Application compatible with any OS.	Pass
2	The Windows Application must be compatible with all the screen sizes.	Pass

Table F.7: Security Testing

Sr. no	Test Condition	Status
1	The user should get responsive and simple GUI for handling things.	Pass
2	User authentication properly work.	Pass

Table F.8: Security Testing

Sr. no	Test Condition	Status
1	The users should not be able to see the internal source code.	Pass
2	Only administrators can modify the database contents.	Pass

ANNEXURE G
INFORMATION OF PROJECT
GROUP MEMBERS



1. Name : Chatuphale Devesh Paresh
2. Date of Birth : 07 October 1995
3. Gender : Male
4. Permanent Address : 321/35, Gokhale Nagar,Pune-411 016
5. E-Mail : dcdevesh3@gmail.com
6. Mobile/Contact No. : 8308640433
7. Placement Details : Placed
8. Paper Published : Published at International Journal of modern Computer Science and Applications (ISSN-2321-2632) May issue-3 2017 volume-5



1. Name : Aniket Anil Dalal
2. Date of Birth : 20 may 1996
3. Gender : Male
4. Permanent Address : kudale lane,NR.nav maharashtra school,pimpri,pune-411 017
5. E-Mail : aniketdalal370@gmail.com
6. Mobile/Contact No. : 9970058913
7. Placement Details : Not-Placed
8. Paper Published : Published at International Journal of modern Computer Science and Applications (ISSN-2321-2632) May issue-3 2017 volume-5



1. Name : Sachin Ashok Dhamal
2. Date of Birth : 27 July 1994
3. Gender : Male
4. Permanent Address : 202 Shrinivas Vihar Narhe Pune-41
5. E-Mail : sachindhamal37@gmail.com
6. Mobile/Contact No. : 8605123724
7. Placement Details : Not-Placed
8. Paper Published : Published at International Journal of modern Computer Science and Applications (ISSN-2321-2632) May issue-3 2017 volume-5



1. Name : Balram Pappu Chavan
2. Date of Birth : 10 May 1996
3. Gender : Male
4. Permanent Address : sr no 135, yashodeep society, warje malwadi
pune-58
5. E-Mail : balramchavan67@gmail.com
6. Mobile/Contact No. : 9890338778
7. Placement Details : Not placed
8. Paper Published : Published at International Journal of modern
Computer Science and Applications (ISSN-2321-2632) May issue-
3 2017 volume-5