SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

ENDPOINT SOFTWARE/APPLICATION MANAGEMENT PORTAL

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Computer Engineering)

BY

Tushar Dahibhate	Exam No: B120234243
Sushant Dharmadhikari	Exam No: B120234259
Krish Gambhir	Exam No: B120234272
Shruti Kothari	Exam No: B120234321

Under The Guidance of

Mrs. A.R Joshi

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

SINHGAD COLLEGE OF ENGINEERING, PUNE-41

Sinhgad Technical Education Society,

Department of Computer Engineering

Sinhgad College of Engineering, Pune - 41

CERTIFICATE

This is to certify that the Project Entitled

ENDPOINT SOFTWARE/APPLICATION MANAGEMENT PORTAL

Submitted by

Tushar Dahibhate	Exam No: B120234243
Sushant Dharmadhikari	Exam No: B120234259
Krish Gambhir	Exam No: B120234272
Shruti Kothari	Exam No: B120234321

is a bonafide work carried out by Students under the supervision of Mrs. A.R Joshi and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering) Project.

Date:

Mrs. A.R Joshi	Prof. P. R. Futane
(Internal Guide)	Head, Computer Engineering Department
Prof.	Dr. S. D. Lokhande
(External Examiner)	Principal

Abstract

The installation of any software requires prior knowledge of the software dependencies which only a person with some technical background will have. The installation process could be tedious and time consuming at times. Endpoint Software/Application management portal will facilitate to manage software and applications on any endpoint. It will be used to automate the process of software installation to reduce the manual efforts required during installation of a software such as- extraction, installation, post-configurations, etc. and will also be time saving.

The software will consist of a Web UI that would be used to take the installation requirements from the user such as - name of the software, version, end-point operating system. The extraction and installation of the software will be done by the chef tool. Installation of applications would be done by the docker tool. Rest API will act as an interface between the front end and the back end.

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***'ENDPOINT SOFTWARE/APPLICATION MANAGEMENT PORTAL'****.*

I would like to take this opportunity to thank my internal guide ***Mrs. A.R Joshi*** *for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.*

I am also grateful to ***Prof. P.R.Futane****, Head of Computer Engineering Department, Sinhgad College Of Engineering for his indispensable support, suggestions.*

In the end our special thanks to for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

Tushar Dahibhate
Sushant Dharmadhikari
Krish Gambhir
Shruti Kothari
(B.E. Computer Engg.)

INDEX

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Endpoint Software/Application management portal will work as a platform to manage software and applications on any endpoint. It will be used to automate the process of software installation. It will reduce the manual efforts required during installation of a software such as- extraction, installation, post-configurations, etc. and will also be time saving. The goals and objective of this project are:

Goal:

- To install softwares and applications on any endpoint.
- Install multiple softwares on a single endpoint in parallel.
- Install a single software on multiple endpoints in parallel.

Objective:

- To automate the process of installation.

1.2 PURPOSE

The conventional method of software installation comes with a lot of overheads . It is also difficult for a person with less technical knowledge to find correctly the software of his interest and perform the subsequent steps that follow.

Hence to overcome these overheads this project automates the process of software installation and is thus faster. The software will reduce the manual efforts required during the installation of any software.

1.3 DEFINITIONS AND ACRONYMS

Acronym	Full Form
SRS	Software Requirement Specification
API	Application Programming Interface
UML	Unified Modeling Language

Table 1.1: Acronyms

1.4 SCOPE

Endpoint Software/Application management portal will work as a platform to manage software and applications on any endpoint. It will be used to automate the process of software installation. It will reduce the manual efforts required during installation of a software such as- extraction, installation, post-configurations, etc. and will also be time saving. Using this portal we can deploy any application on any machine regardless of the operating system it would have.

Following feature is also included in the project:

- Parallel installation of multiple softwares on a single endpoint.

1.5 OUTLINE

This project report is divided into 4 chapters.
Chapter 1: It covers the introduction which comprises of scope and purpose of the project.
Chapter 2: It is Software Requirement Specification which covers system features, external interfaces required and non- functional requirements of the system.
Chapter 3: It is the Design Document chapter which consists of Architectural design and UML diagram. User interface and software interface design are also a part of it.
Chapter 4: It presents an overview of project tasks and the output of a project scheduling tool by analyzing various risks involved in the project.
Appendix A: It contains ACM keywords, Mathematical model and the feasibility study of system.
Appendix B: It deals with test plan, user acceptance testing and reliability testing.
Appendix C: Details about individual contribution of each team member are given in this chapter.
References and research paper published by the group is attached at the end of this project report.

CHAPTER 2

SRS DOCUMENT

2.1 INTRODUCTION

2.1.1 Purpose

To automate the tedious and the conventional process of software installation and to facilitate machine independent deployment of applications along with its dependencies. Endpoint Software/Application Management Portal will work as a platform to manage softwares and applications on any endpoint accessible.

2.1.2 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, testers and the end users. End users can be people from the IT industry or from other professions. Automatic installation being the key focus of the project, people from various professions without any explicit knowledge about computers will find ESAMP very useful.

2.1.3 Deliverables

The fully developed software will be delivered.

2.2 OVERALL DESCRIPTION

2.2.1 Product Perspective

Installing software can sometimes be difficult and time consuming as well. It is also not possible to keep tabs on the installation process and to keep pressing next occasionally until the whole process is complete. Also in some installations we often need to change the Environment variables in order to conform the environment to the desired settings for the software to smoothly install and run. People from non-IT professions often need to take the assistance of an IT expert in order to install particular software.
Automation would drastically reduce the manual efforts involved in installation process. We use many applications and while migrating to some other operating system we often face a lot of problem. Additionally, we also need to take into account the

dependencies involved to run a particular application. ESAMP will be able to run the applications enabling portability as well as considering the dependencies involved.

2.2.2 Product Functions

The main functionalities which the proposed system provides, described as an overall view, are the following:

- Accepting the user input.
  - Software Name/ Application Name
  - Operating System
  - Version
  - Credentials
- Query generation and searching for the chef recipe in the database.
- Installing Chef client.
- Running the chef recipe.
- Installing the desired software.
- Installation of docker.
- Pulling docker image from the hub.
- Starting docker container and running the application image.
- Creating a docker image.
- Exception Handling.

2.2.3 User Classes and Characteristics

There is only 1 type of user at the endpoint who may or may not have any knowledge about the software and the process of installation.

2.2.4 Operating Environment

- Client machine:
  The client or user machine on which the software gets installed
- Chef server:
  The server contains the repository of the software that have already been installed by some nodes and manage them in the database.
- Workstation:
  The Workstation helps in managing the nodes and deployment of the software on multiple nodes.

2.2.5 Design and Implementation Constraints

- Chef client for the specified operating system is not available.
- Recipes cannot be generated dynamically.
- Conventional Windows OS(7,8,8.1) cannot run inside a docker container. Only windows server can run inside a container.

2.3 EXTERNAL INTERFACE REQUIREMENTS

2.3.1 User Interface

The user interface includes the GUI designed for the user to get the requirements as input and install the required software. The user will only have to select his operating system and enter the software name and click install. The User Interface will be developed using HTML and CSS.

2.3.2 Communication Interface

The Communication Interfaces used are:

- Rest API:
  - The software details are taken from the UI.

  - These are then forwarded to chef and docker to create and run the respective recipe.

- Chef client-server interaction:
  - The Chef client needs to interact with the server to download the software and install it.
  - This process is automated by the chef itself and no external commands need to be carried out.
  - Only the client or the node should be in connection with the chef server.

- Docker interaction:
  - Docker pulls the image of the operating system with the application installed.
  - The required application name is passed in the form of environment variables from the REST API to Docker.

2.4 SYSTEM FEATURES

These are the functional requirements of the system.

2.4.1 UI features

Our system will be based on Client-server architecture. We will employ a powerful commodity server. The server will be responsible for accepting client requests and giving them results accordingly.

2.4.1.1 Selection of the OS

This feature will provide the option for the selection of the OS at the endpoint.

2.4.1.2 Enter the software details

Depending on the OS selected the user then will provide the s/w name and other credentials if required. He will then be asked to download the chef client for the endpoint OS.

2.4.2 Chef Client

The chef client will take the responsibility to install the software automatically by running the corresponding recipe.

2.4.3 Docker fuctionality

In case of an application the Docker is installed first on the endpoint and then the required application image using the base OS is fetched and run on the endpoint.

2.4.4 Chef Workstation

The chef-workstation will be responsible to manage the nodes in the network and manage the parallel installation of the software.

2.4.5 Display Information

Once the software is installed, the display message should be shown for successful installation. If there occurs any exception then the respective message should be displayed to handle the exception.

2.5 OTHER NONFUNCTIONAL REQUIREMENTS

2.5.1 Performance Requirements

Installation process could be speed up by using parallelization.

2.5.2 Safety Requirements

Installation through this tool should not cause any undesired settings or changes.

2.5.3 Security Requirements

There is no involvement of users personal data in our system. Therefore there will be no such security requirements. Only the administrative privileges for the chef client should be provided by the endpoint.

2.5.4 Software Quality Attributes

2.5.4.1 Correctness

The recipes created should run on any platform provided by the user. Therefore, Operating system specific recipes should be written correctly in the cookbook.

2.5.4.2 Maintainability

Our model will be robust. It can be used to install any software on any operating system. Therefore, to make our code more reusable, we will maintain it in small modules that encapsulate their functionality and hide it from the rest of the system. This will allow any changes within each module to be isolated from impacting the rest of the system.

2.5.4.3 Reliability

Extensive error handling should be utilized to ensure the end user experiences minimum system halts or system crashes.

2.5.4.4 Testability

Writing the application as a set of small modules will enable us to Unit test the individual components of the system more effectively.

2.5.4.5 Usability

All care should be taken to abstract unnecessary details away from the view of the enduser. The UI should be user-friendly and should provide easy navigation through webpages and also provide the required information.

2.5.5 Business Rules

The System does not have any limited usage rules. Anyone, be it an IT professional, people in banks or any organizations or even a person with no prior knowledge about the software can use the system for installing any software. No particular functions are restricted to any individual as well as there are no specific conditions to use the product.

CHAPTER 3

SOFTWARE DESIGN SPECIFICATION

3.1 INTRODUCTION

This section provides an overview of the entire design document. This document describes all architectural and interface and component-level design for the software.

3.2 ARCHITECTURAL AND COMPONENT LEVEL DESIGN

The figure below shows the architectural design of Endpoint Software/Application management portal.

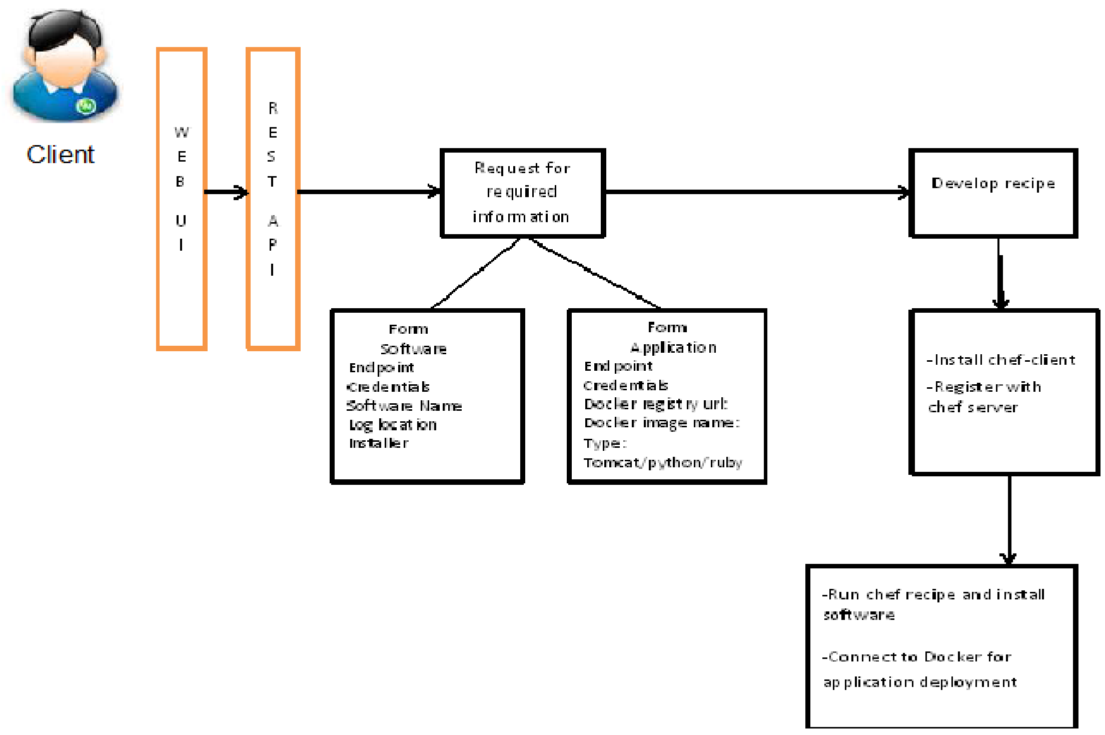

Client
WEB UI
REST API
Request for required information
Develop recipe
Form Software Endpoint Credentials Software Name Log location Installer
Form Application End point Credentials Docker registry url: Docker image name: Type: Tomcat/python/ruby
-Install chef-client -Register with chef server
-Run chef recipe and install software -Connect to Docker for application deployment

Figure 3.1: Architecture Diagram

The blocks which are shown in architecture diagram are as follows:

REST API:

- REST stands for REpresentational State Transfer. It is an interface between the front end and the back end.
- REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard

methods.

- In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources.
- Following well known HTTP methods are commonly used in REST based architecture.
  GET - Provides a read only access to a resource.
  PUT - Used to create a new resource.
  DELETE - Used to remove a resource.
  POST - Used to update a existing resource or create a new resource.
  OPTIONS - Used to get the supported operations on a resource.
- REST is the underlying architectural principle of the web. The amazing thing about the web is the fact that clients (browsers) and servers can interact in complex ways without the client knowing anything beforehand about the server and the resources it hosts.
- REST API simplifies the queries required to talk to the backend.

CHEF:

- Chef is a company configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration ”recipes”.
- Features: The user writes ”recipes” that describe how Chef manages server applications and utilities (such as Apache HTTP Server, MySQL, or Hadoop) and how they are to be configured. These recipes (which can be grouped together as a ”cookbook” for easier management) describe a series of resources that should be in a particular state: packages that should be installed, services that should be running, or files that should be written.
- Chef makes sure each resource is properly configured and corrects any resources that are not in the desired state.

- Chef can run in client/server mode, or in a standalone configuration named ”chef-solo”.
- Chef turns infrastructure into code. With Chef, you can automate how you build, deploy, and manage your infrastructure. Your infrastructure becomes as version able, testable, and repeatable as application code.
- Chef server stores your recipes as well as other configuration data. The Chef client is installed on each endpoint called node and is registered with the server.

DOCKER:

- Docker implements a high-level API to provide lightweight containers that run processes in isolation.
- Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.
- Using Docker,we can
  - Build Docker Images that hold the applications.
  - Create Docker Containers from that image to run the application.
  - Download Images from the Docker Hub(Docker pull)
  - Uploading images to Docker Hub(Docker push)

3.2.1 UML Diagrams

UML enables system developers to specify, visualize and document models in a manner that supports scalability, security and robust execution. Because UML, modeling raises the level of abstraction throughout the analysis and design process, it is easier to identify patterns of behavior and thus define opportunities for refactoring and reuse.

3.2.1.1 Use Case Diagram

A Use Case diagram is a type of behavioral diagram defined by the UML created from a use case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goal represented as use case and any dependencies between those use cases. It is a type of diagram that shows a set of use cases, actors and their relationships. It should have a distinct name. It commonly contains
a) Use Cases.
b) Actors (Primary and Secondary).
c) Dependency, Generalization and association relationships.

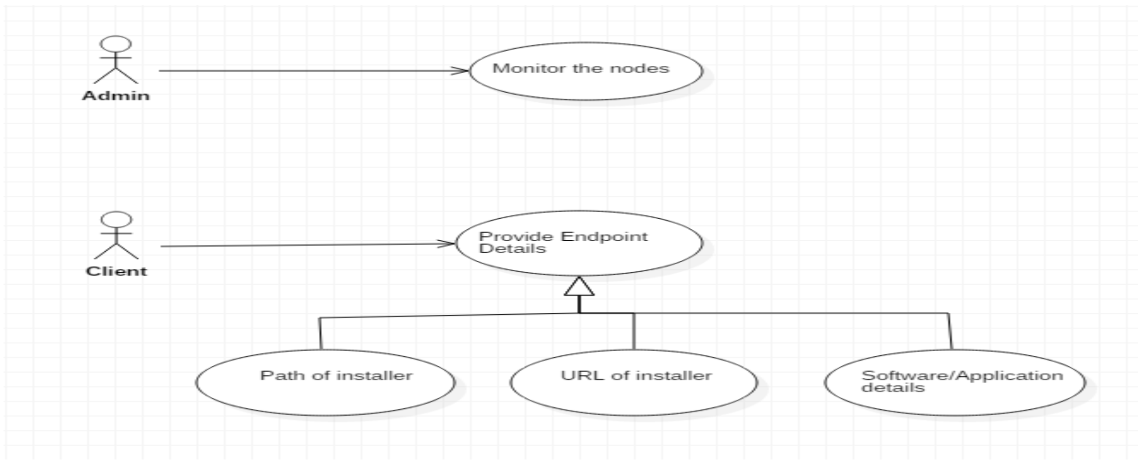


Figure 3.2: Use Case Diagram

3.2.1.2 Activity Diagram

An activity diagram refers to the flow of activity involved in triggering any event .The pre-requisite for the activity diagram are scope statement and Use Case diagram .An activity is ongoing non atomic execution within a state machine .Activity ultimately results in those events which are made up of executable automatic computation that results in change of state of system .Activity diagram commonly contains activities, states, action states, transition, object notes etc.

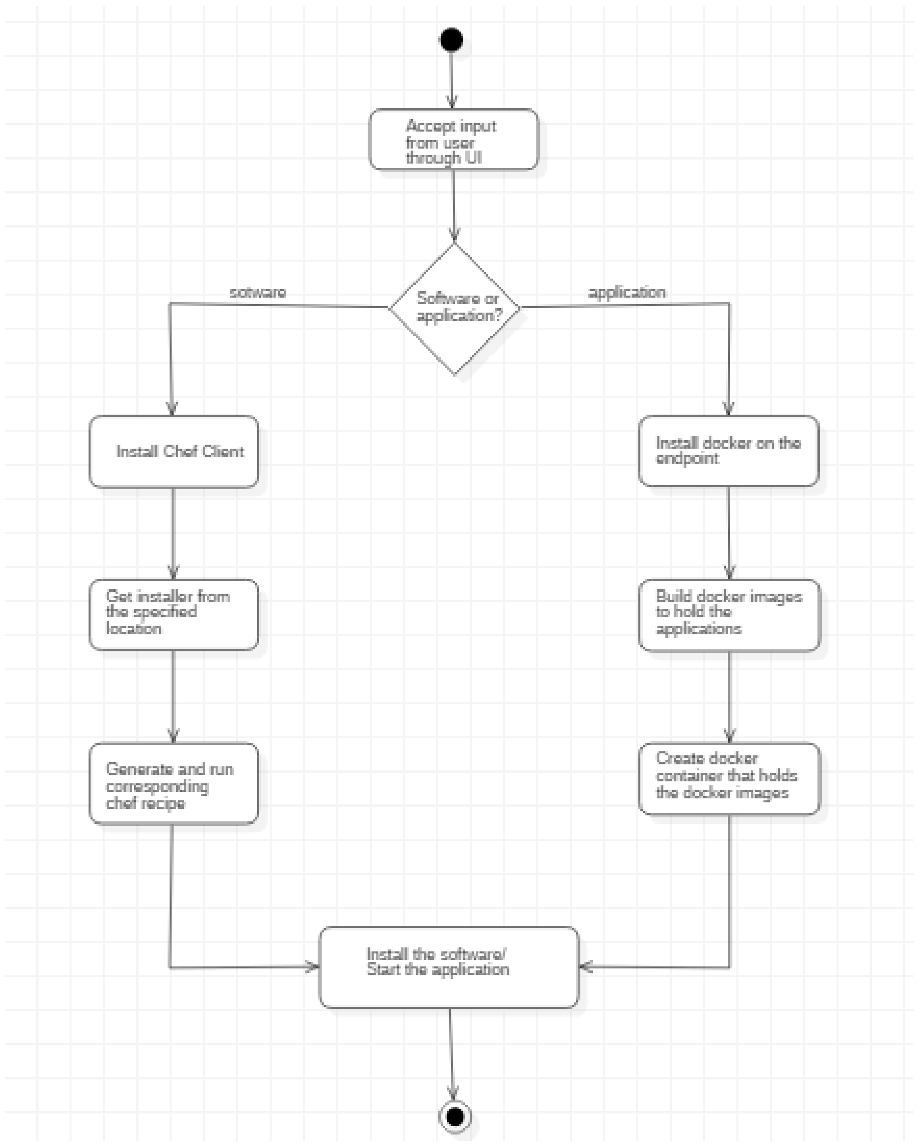

Accept input from user through UI
Software or application?
sotware
application
Install Chef Client
Get installer from the specified location
Generate and run corresponding chef recipe
Install docker on the endpoint
Build docker images to hold the applications
Create docker container that holds the docker images
Install the software/ Start the application

Figure 3.3: Activity Diagram

3.3 USER INTERFACE DESIGN

The User-interface will look like :

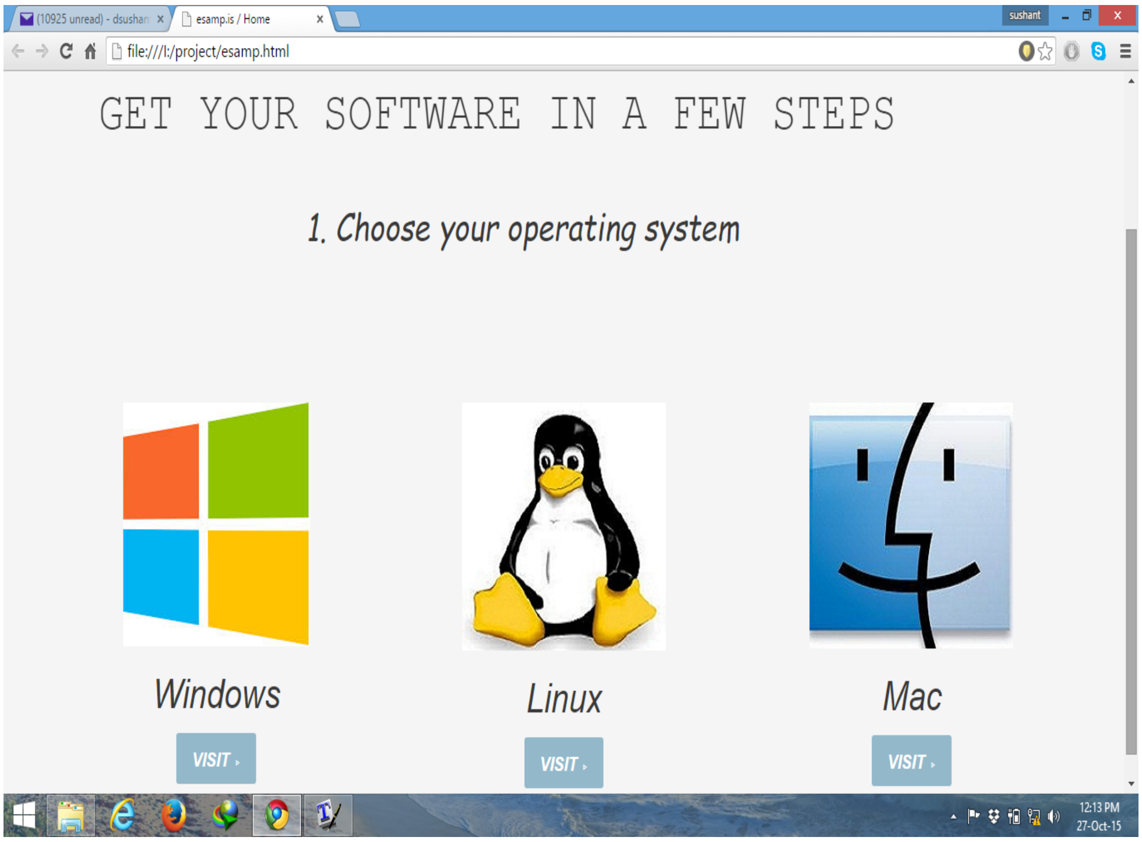


Figure 3.4: User Interface

CHAPTER 4

PROJECT PLAN AND RISK ANALYSIS

4.1 PROJECT TASK SET

We have adopted the linear sequential model for software development.

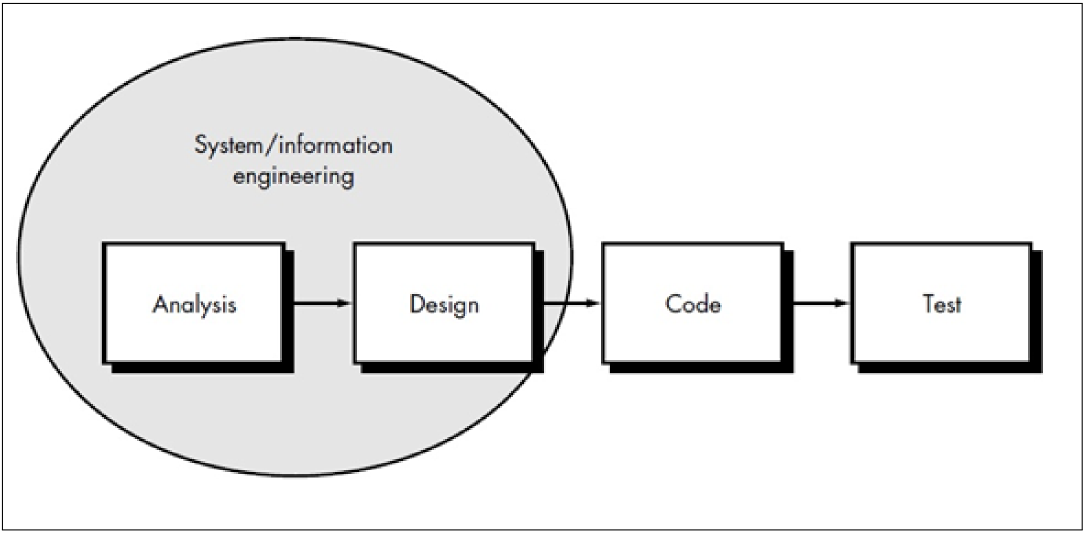


Figure 4.1: Waterfall Model

Sometimes called the classic life cycle or the waterfall model, the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support. The above figure illustrates the linear sequential model for software engineering.

System/information engineering and modeling: Because software is always part of a larger system (or business), work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software.

This system view is essential when software must interact with other elements such as hardware, people, and databases. System engineering and analysis encompass requirements gathering at the system level with a small amount of top level design and analysis. Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

Major Tasks in the Project stages are:

- **Software requirements analysis:** The requirements gathering process is intensified and focused specifically on software. To understand the nature of the program(s) to be built, the software engineer (”analyst”) must understand the information domain for the software, as well as required function, behavior, performance, and interface. Requirements for both the system and the software are documented and reviewed with the customer.
- **Design:** Software design is a multistep process that focuses on four distinct attributes of a program: data structure, software architecture, interface representations, and procedural (algorithmic) detail. The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.
- **Planning:** Planning activity guides the team in the development of the project. The software project plan defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.
- **Code generation:** The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.
- **Testing:** Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

4.2 FUNCTIONAL DECOMPOSITION

The functional break-down schema of the entire project and brief analysis of each stage is explained below:

4.2.0.1 Get requirements from user

In this phase we are going to collect as much data from the user regarding the OS type and the required software or application name and dependencies required.

4.2.0.2 Get the prerequisites for installation

In this phase, the software needs to get installed using a client. This client gets the process of software installation automated. We use Chef to get this automation done. The chef client runs the respective recipes for the software and Docker takes care for the application installation part.

4.3 SOFTWARE SIZING

Project Duration = 8 Months

Team Size = 4

Total Effort by 4 people in 8 months = 2 person months.

4.4 TIMELINE CHART

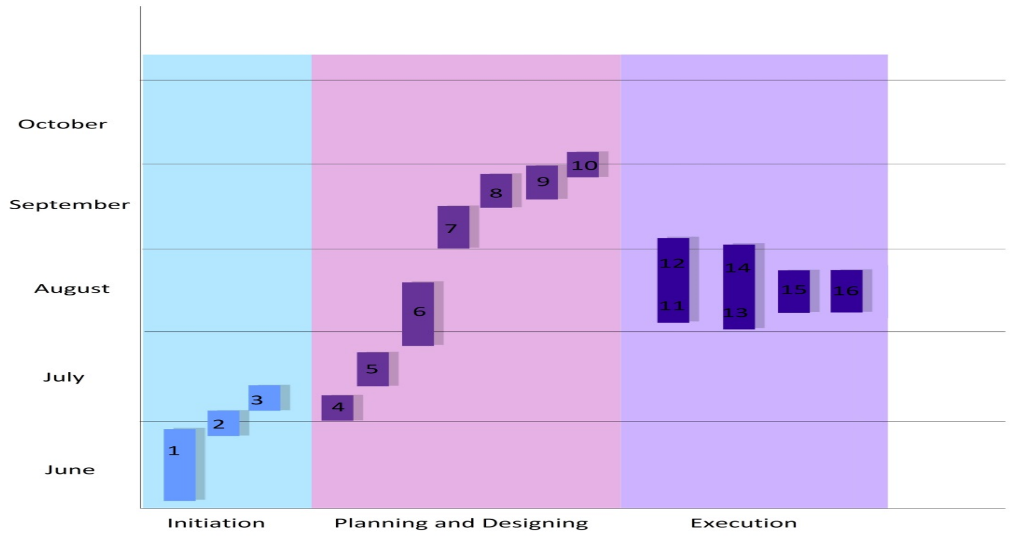


Figure 4.2: Timeline Chart

4.5 RISK ANALYSIS

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

Sr.No.	Risk	Category	Probability(%)	Impact	Risk Management
1	Team members leave or become sick	SS	10	3	Ensure the plan has contingency built into it to allow for less than expected resource availability
2	Technical solution has major flaws	TB	15	2	Invest in appropriate levels of testing. Consider a period of parallel running. Have a fallback contingency plan to revert to a previous system if necessary.

Type	Definition
PS	Product Size
BI	Business Impact
CC	Customer Characteristics
PD	Process Definition
DE	Development Environment
TB	Technology to be Built
SS	Staff Size and Experience

Table 4.1: Risk Categories

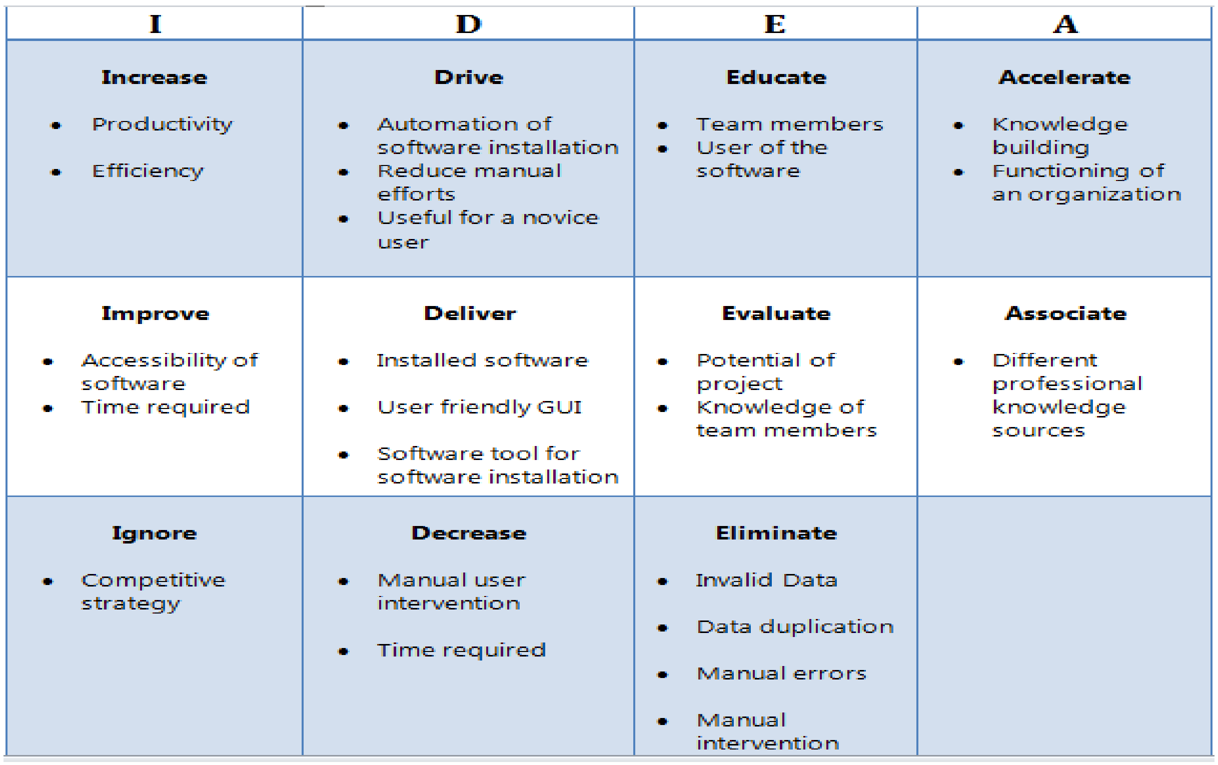
Impact	Value
Catastrophic	1
Critical	2
Marginal	3
Negligible	4

Table 4.2: Risk Impact definitions

CHAPTER 5

APPENDIX A

5.1 IDEA MATRIX



I	D	E	A
Increase • Productivity • Efficiency	**Drive** • Automation of software installation • Reduce manual efforts • Useful for a novice user	**Educate** • Team members • User of the software	**Accelerate** • Knowledge building • Functioning of an organization
Improve • Accessibility of software • Time required	**Deliver** • Installed software • User friendly GUI • Software tool for software installation	**Evaluate** • Potential of project • Knowledge of team members	**Associate** • Different professional knowledge sources
Ignore • Competitive strategy	**Decrease** • Manual user intervention • Time required	**Eliminate** • Invalid Data • Data duplication • Manual errors • Manual intervention	

Figure 5.1: Idea Matrix

5.2 MATHEMATICAL MODEL

System Description:{ I , O , F , Fs , Ff }

- Input I = {R1,R2,R3,...,Rn }

  R1 = Request made by Client1.

  R2 = Request made by Client2.

  R3 = Request made by Client3.

  . . Rn = Request made by Client4.

  For each Client R1 = {R11,R12,R13,...,R1n }

  R11,R12,R13,...,R1n can be the multiple requests made by the Client 1 in Parallel.

  U = {U1,U2,U3,U4 }

Where U1,U2,U3,U4 are tuples

U1 = Operating System name

U2 = Software name

U3 = User Credentials

U4 = Prerequisites

- Intermediate Output J = {J1,J2,J3,...,Jn }

  J1 = Request R1 is prcessed by job J1.

  J2 = Request R2 is prcessed by job J2.

  J3 = Request R3 is prcessed by job J3.

  . . Jn = Request Rn is prcessed by job Jn.

- Output O = Set of processed requests

  O = {PR1,PR2,PR3,...,PRn}

- Functional Requirements, F ={ F1, F2 }

  Where,

  F1 = Install a software.

  F2 = Install an application.

  F1 = {F11 , F12}

  Where,

  F11 = Install chef client on the endpoint.

  F12 = Get the installer from the specified location and generate a chef recipe.

  F2 = {F21 , F22 , F23} Where,

  F21 = Install docker on the endpoint.

  F22 = Build docker images to hold the applications.

  F23 = Create docker container that holds the docker images.

- Success Conditions:
  1. Software successfully installed.
  2. Application successfully started.
- Failure Conditions:
  1. Connection not established.
  2. Incompatible dependencies.
  3. Chef client not available for the Operating System.
  4. Docker installation not done properly.

5.3 ALGORITHM AND FEASIBILITY ANALYSIS

The algorithm steps are:

```
1. Find OS type
2. Get chef client
3. If (Software is to be installed)
    (a) Create recipe
    (b) Run the corresponding recipe for the software using chef client
4. Else
    (a) Install Docker at endpoint
    (b) Get the base image of the application and the OS in the Docker container
    (c) Run the Container
```

Thus, ESAMP can be categorized under NP complexity, for its complexity for yes or no depends upon the type of package the user is installing either a software package or an application. Thus, if(Software is to be installed) can return 1 or 0

Installation can be completed in polynomial time. Hence it is a P class problem.

CHAPTER 6

APPENDIX B

6.1 INTRODUCTION

This is the Test Plan for the project titled ESAMP. This plan will address only those items and elements that are related to the modules developed for this purpose. This document has the objective to determine the list of Test Cases that need to be executed when assuring the systems quality. The goal of this document is to establish the list of tasks that, if performed, will identify all of the requirements that have not been met in the software.

6.2 TEST ITEMS : FUNCTIONS

Following are the things that we intend to test within scope of this test plan.

- Gathering the required information about the software from the user .
- Getting the required software package and creating the recipe.
- Installing the chef client on the endpoint to run the recipe.
- Installing Docker on the endpoint and getting the required image of the application and the OS.

6.3 FEATURES TO BE TESTED

Following are the features which are required to be tested from Users viewpoint. The level of risk for each feature are done using simple rating scale such as (H, M, L): High, Medium and Low-

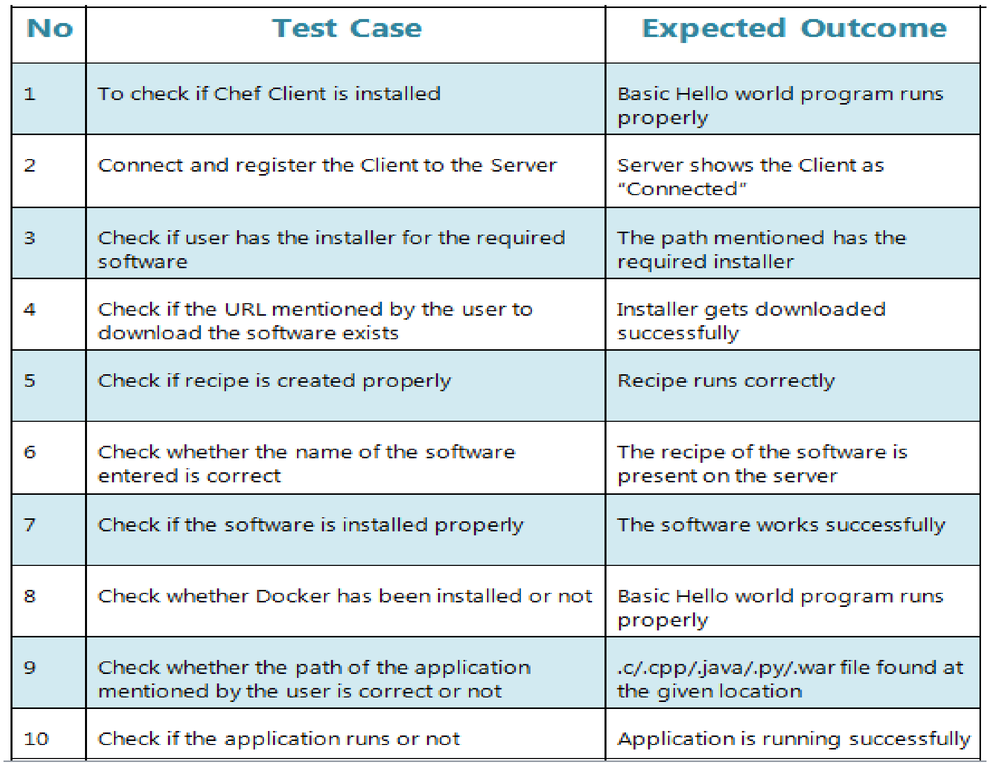
Sr.No.	Features to be tested	Level of Testing Required
1	Accepting input from the user	M
2	Check for availability of the chef client for the OS	H
3	Creating the corresponding recipe	H
4	Installation of Docker on client side	M
5	Create base image with required dependencies and the base OS	H

6.4 APPROACH : STRATEGY

The testing for the project will consist of Unit, System/Integration (combined) and Acceptance test levels. Most testing will be done by the developers themselves.
UNIT Testing will be done by the developers and will be approved by the Project guide (internal and external). All unit test information will also be provided to the project guides.
SYSTEM/INTEGRATION Testing will be performed by the development team with assistance from the Sponsored Organizations person as required... Programs will enter into System/Integration test after all critical defects have been corrected after unit testing.

6.5 TEST CASES

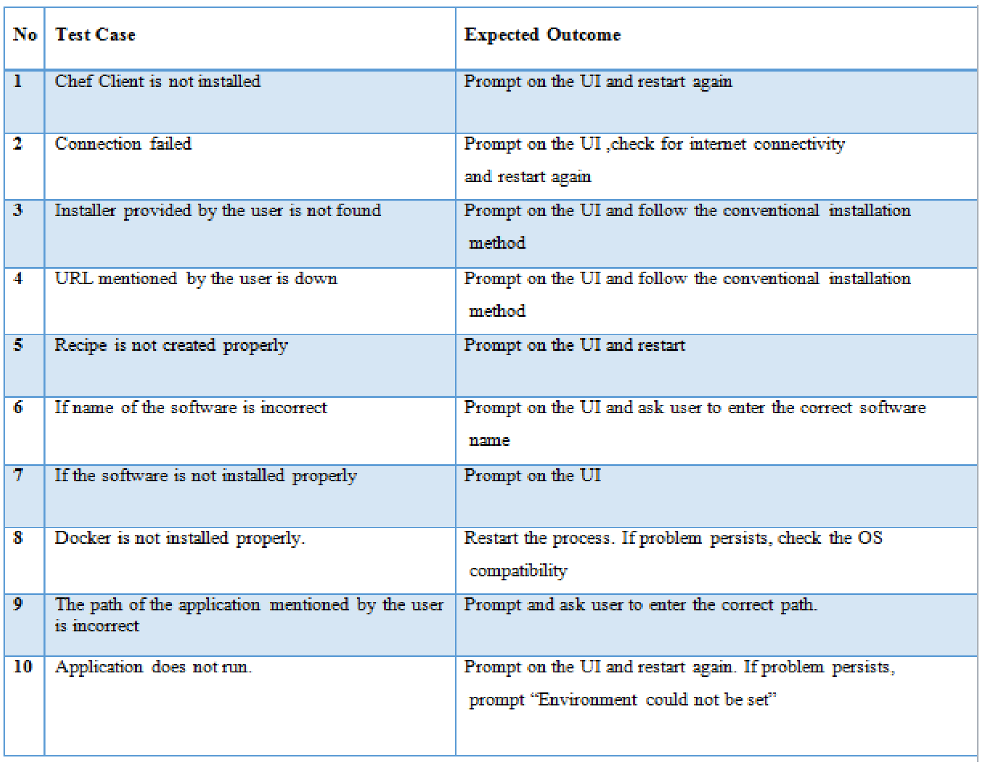
6.5.1 Positive Test Cases



No	Test Case	Expected Outcome
1	To check if Chef Client is installed	Basic Hello world program runs properly
2	Connect and register the Client to the Server	Server shows the Client as "Connected"
3	Check if user has the installer for the required software	The path mentioned has the required installer
4	Check if the URL mentioned by the user to download the software exists	Installer gets downloaded successfully
5	Check if recipe is created properly	Recipe runs correctly
6	Check whether the name of the software entered is correct	The recipe of the software is present on the server
7	Check if the software is installed properly	The software works successfully
8	Check whether Docker has been installed or not	Basic Hello world program runs properly
9	Check whether the path of the application mentioned by the user is correct or not	.c/.cpp/.java/.py/.war file found at the given location
10	Check if the application runs or not	Application is running successfully

Figure 6.1: Positive Test Cases

6.5.2 Negative Test Cases



No	Test Case	Expected Outcome
1	Chef Client is not installed	Prompt on the UI and restart again
2	Connection failed	Prompt on the UI ,check for internet connectivity and restart again
3	Installer provided by the user is not found	Prompt on the UI and follow the conventional installation method
4	URL mentioned by the user is down	Prompt on the UI and follow the conventional installation method
5	Recipe is not created properly	Prompt on the UI and restart
6	If name of the software is incorrect	Prompt on the UI and ask user to enter the correct software name
7	If the software is not installed properly	Prompt on the UI
8	Docker is not installed properly.	Restart the process. If problem persists, check the OS compatibility
9	The path of the application mentioned by the user is incorrect	Prompt and ask user to enter the correct path.
10	Application does not run.	Prompt on the UI and restart again. If problem persists, prompt "Environment could not be set"

Figure 6.2: Negative Test Cases

CHAPTER 7

APPENDIX C

7.1 INDIVIDUAL CONTRIBUTIONS

- List of Activity Performed By Tushar Dahibhate

S.No	Activity Performed	Start Date	End Date
1	Literature Survey	14-07-2015	30-07-2015
2	Study of Docker	30-07-2015	17-08-2015
3	Study of UML Diagrams	5-09-2015	12-09-2015
4	Study and implementation of Data Design	14-09-2015	20-09-2015
5	Study of Latex Environment.	25-09-2015	02-10-2015
6	Preparation of Project Report in Latex	15-10-2015	21-10-2015

- List of Activity Performed By Sushant Dharmadhikari

S.No	Activity Performed	Start Date	End Date
1	Literature Survey	14-07-2015	30-07-2015
2	Study of Chef	30-07-2015	17-08-2015
3	Study and implementation chef	5-09-2015	12-09-2015
4	Study of Feasibility Analysis	14-09-2015	20-09-2015
5	Study of Latex Environment	25-09-2015	02-10-2015
6	Preparation of Project Report in Latex	15-10-2015	21-10-2015

- List of Activity Performed By Krish Gambhir

S.No	Activity Performed	Start Date	End Date
1	Literature Survey	14-07-2015	30-07-2015
2	Study of Rest API	30-07-2015	17-08-2015
3	Study of Software Requirement Specification	5-09-2015	12-09-2015
4	Study of Mathematical Model	14-09-2015	20-09-2015
5	Study of Idea Matrix.	25-09-2015	02-10-2015
6	Preparation of Chapterwise Project Report	15-10-2015	21-10-2015

- List of Activity Performed By Shruti Kothari

S.No	Activity Performed	Start Date	End Date
1	Literature Survey	14-07-2015	30-07-2015
2	Study of development of Web UI using HTML/CSS	30-07-2015	17-08-2015
3	Study of Software Requirement Specification	5-09-2015	12-09-2015
4	Study of Goals and Objectives	14-09-2015	20-09-2015
5	Study of Idea Matrix.	25-09-2015	02-10-2015
6	Preparation of Chapterwise Project Report	15-10-2015	21-10-2015

CHAPTER 8

REFERENCES

[1] www.wikipedia.org

[2] www.youtube.com

[3] www.Learn.chef.io

[4] Steven c. MarkeyREST in the cloud www.ibm.com

[5] docs.docker.com

[6] www.digitalocean.com/community/tags/docker?type=tutorials

[7] blog.flux7.com/blogs/docker/docker-tutorial-series-part-1-an-introduction