Software Project Report

for

VStack Project

Version 1.0 approved

Prepared by Group 5

San Jose State University

Dec 4th, 2016

Table of Contents

1. Introduction

1.1 Purpose

This document is a report for VStack project. The project was built based on requirement of course CMPE283-Virtualization for final project. VStack is an extension of OpenStack, which is built on top of OpenStack to provide additional features

1.2 Document Conventions

This document follows IEEE standard template, ***Copyright © 2005 by Karl E. Wiegers, IEEE.***

1.3 Intended Audience and Reading Suggestions

- Developer with basic knowledge of OpenStack interface (CLI and Rest API)
- Dr. Thomas Hildebrand and Teaching Assistant

The first part of the document will focus on the high level design of VStack. The later section describe the algorithm & technologies use in this project and finally describe the implementation process. Viewer are advised to read this document from beginning till the end in that order, as it is not dictionary type.

1.4 Product Scope

Built on top of OpenStack, VStack application provide a convenient way to set up infrastructure for web services. This process usually involve many complicated steps and time consuming, sometimes repetitive.

VStack provide several service templates to choose from and an integrated view to monitor all resources backing that service. This make the administrative job simple.

Powered by OpenStack, the service provided by VStack is highly scalable which can adapt well to business need for a scalable web application platform but private, on-premises.

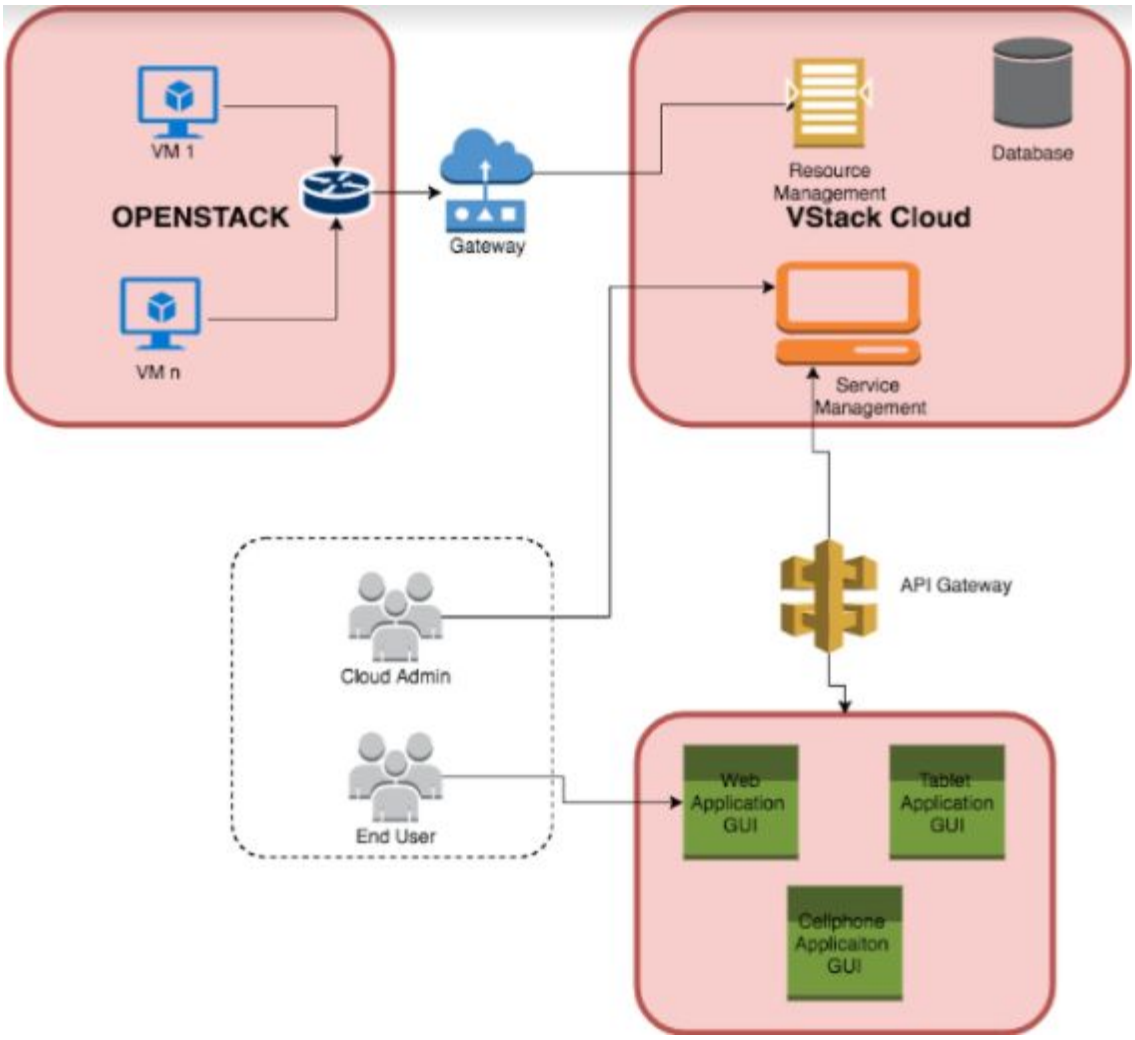
1.5 References

http://developer.openstack.org/

2. Overall Description

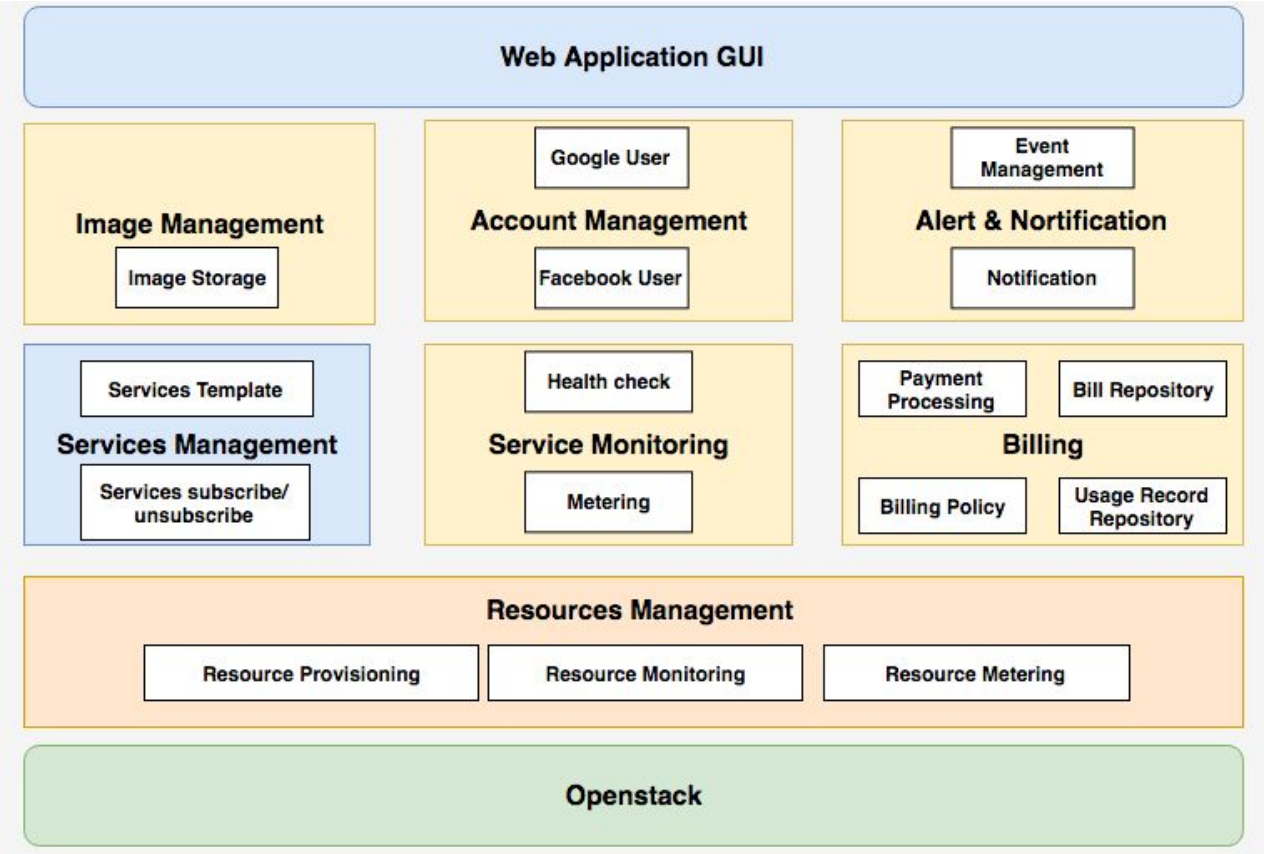
2.1 Product Perspective

OpenStack provide infrastructure as a Service, which provide on-demand, scalable infrastructure. However, it still require much more effort to build a platform for a particular services. In this project, VStack aim to build a web application platform on top of OpenStack to provide Web application platform as a service



2.2 Product Functions

- Service Management, start or stop the following service:
  - Simple web service: can host static web site
  - Dynamic web service: can host web site with database connection
- Service monitoring
- Account Management: single sign on with Google or Facebook as identity provider
- Alert and Notification: define event trigger and notify user by SMS, Email
- Billing: metering usage and calculate bill
- Image Management: manage predefined VM images



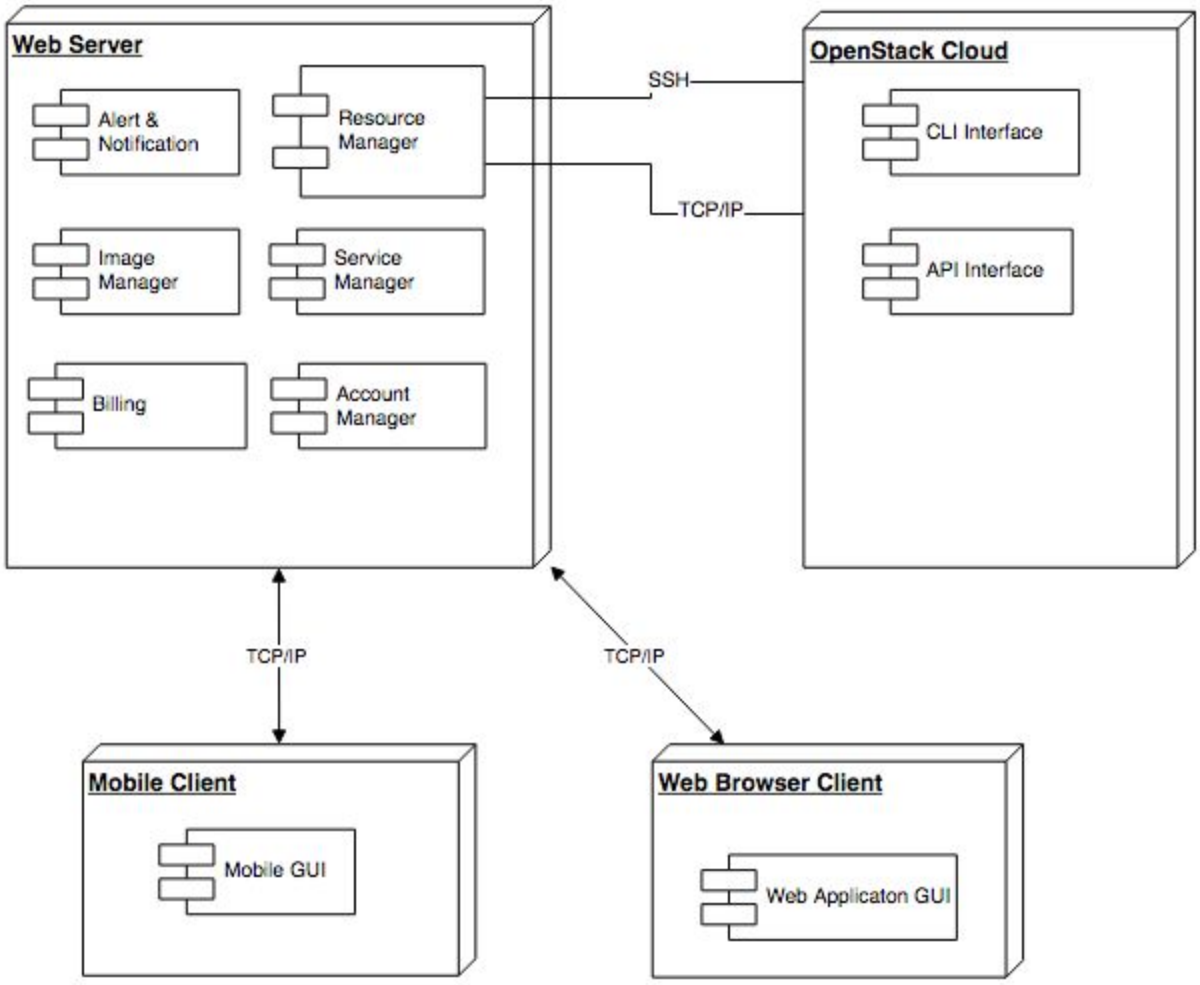
2.3 User Classes and Characteristics

- Web Application Developer: develop web application and have need for multiple environment for testing the product before rolling out to production

- Sysops: receive request for development team or other group in organization to build infrastructure for a service or set up service along with the infrastructure required by that service.

2.4 Operating Environment

VStack is a web application, and it need to be deployed to a web server (Apache Tomcat). VStack also require a running OpenStack in the LAN network, and access to CLI and API interface of OpenStack. The deployment diagram is as below.



2.5 Design and Implementation Constraints

- Technologies:
  - Project have to build on top of OpenStack and use both of its controlling interface: command line and API
  - OpenStack version: Mitaka, which is customized to be used in lab environment and so far can only be run with one compute node

- Hardware limitation: because of technology limitation on this lab version of OpenStack, it can only be run with one compute node. The compute node used for this project is a laptop with 16GB memory and Core i5 CPU. Therefore greatly limit the option to provision multiple virtual machines with different flavor.

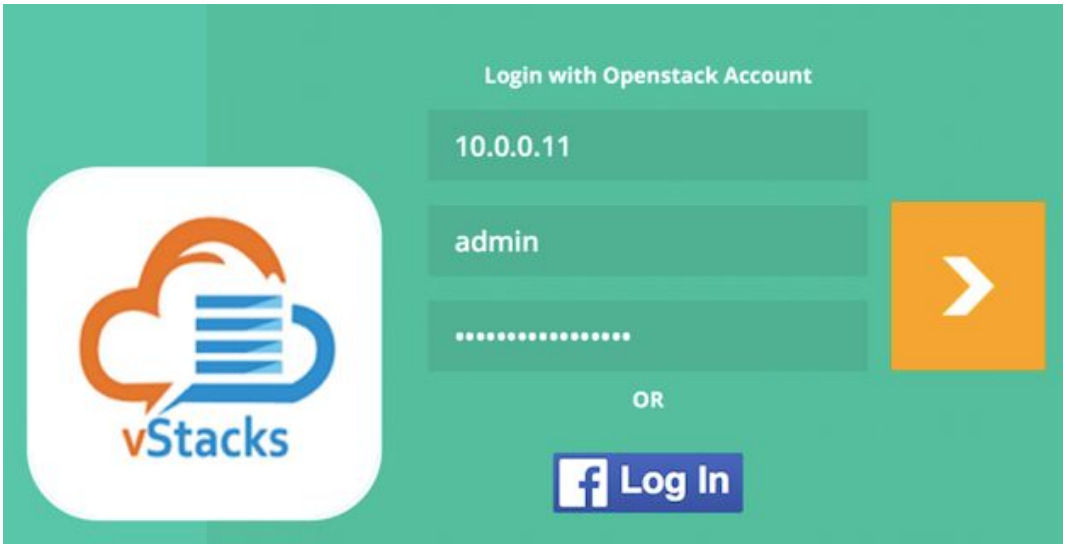
2.6 Assumptions and Dependencies

- OpenStack Mitaka version is running
- Apache Tomcat server 8 or higher is used to deployed VStack
- Access to OpenStack interface (Cli and API) is provided to VStack

3. External Interface Requirements

3.1 User Interfaces

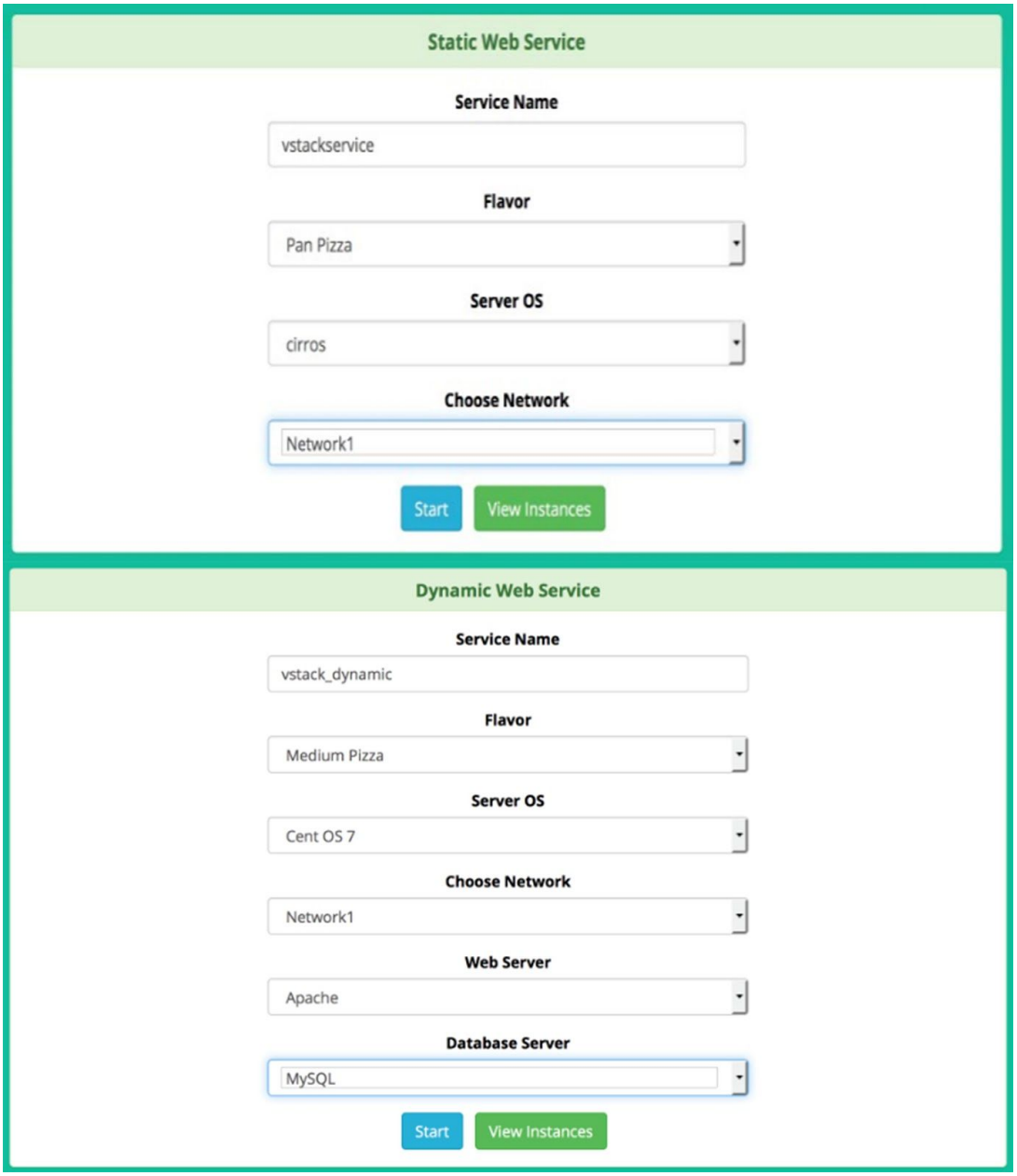
3.1.1 Login Screen:



The login screen support Single Sign On and social network account sign up. User can use their existing Facebook or Google account to log in. From the second login, VStack will determine if user is already logged in Facebook/Google and automatically authenticate user to VStack and bypass the login screen.

3.1.2 Service management screen:

This is the main service screen that allow user to provision web service based on template. There are currently 2 templates Static Web Service and Dynamic Web Service


Static Web Service
Service Name
vstackservice
Flavor
Pan Pizza
Server OS
cirros
Choose Network
Network1
Start
View Instances
Dynamic Web Service
Service Name
vstack_dynamic
Flavor
Medium Pizza
Server OS
Cent OS 7
Choose Network
Network1
Web Server
Apache
Database Server
MySQL
Start
View Instances

3.1.2 Service monitoring:

3.2 Hardware Interfaces

Physical machine that run as compute node should support Intel EPT (or AMD RVI) and VT-x (or AMD-V)

3.3 Software Interfaces

System Service - Java Spring library has been used to communicate front end to backend.
Command Line Interface Implementation - JSCH Java library has been used to SSh into the controller node of OpenStack and and establish a session. Export The environment variable for authentication through command line. Execute the OpenStack command and close the session.
REST API Implementation - HttpClient library has been used to do GET and POST calls to OpenStack. Jackson and Json Library is used to parse the Json response.

3.4 Communications Interfaces

- HTTP: to access VStack from Web GUI or Mobile GUI and to access OpenStack API interface from VStack (web server)

- SSH: to access OpenStack CLI interface from VStack

4. System Features

4.1 Single Sign On with Social Network Credential

4.1.1 Description

Almost every internet user have a social network account, and this feature allows users to set up their account in VStack using their existing Facebook or Google account. From the second login, user is not required to login anymore if they already logged in with Facebook or Google

4.1.2 Stimulus/Response Sequences

From login page, click on “Facebook” or “Google”, then authorize VStack to access Google/Facebook.

4.1.3 Functional Requirements

This feature utilize the following API/Cli command

a) Assign Role to the User (CLI):

$ openstack rola add --user USER_NAME --project PROJECT_NAME ROLE_NAME

4.2 Web Service Provisioning

4.2.1 Description

Allow user to start or stop creating a web service from a service template. Currently support Simple Web Service and Dynamic Web Service

4.2.2 Stimulus/Response Sequences

Base on the service template that user chose, VStack will initiate a list of predefined command to provision Virtual Machine in OpenStack and set up web server, database server...and wire those server together with appropriate network interface

4.2.3 Functional Requirements

This feature utilize the following API/Cli command:

a) Create Instances (REST API):

POST http://{{Host}}:8774/v2.1/servers

b) Create floating IP (CLI):

$ nova floating-ip-create PROVIDER_POOL

c) Associate Floating IP to Instance (CLI):

$ nova floating-ip-associate INSTANCE_NAME FLOATING_IP

d) Set up web server and database server:

We have written following script for automated Database & Server deployment:

- apache_install.sh
- mysql_install.sh
- nginx_install.sh

Above script will detect the Linux version & on the basis of version, it will deploy the database and server.

4.3 Service Monitoring

4.3.1 Description

Allow user to monitor services health and status

4.3.2 Stimulus/Response Sequences

When user access monitoring page, VStack will make API call to OpenStack at regular interval to update status of overall system resources.

4.3.3 Functional Requirements

This feature utilize the following API/Cli command:

e) Check status of VM instances (REST API):

GET http://{{Host}}:8774/v2.1/servers/{serverid}

f) Check status of web server and database server:

service httpd status
service mariadb status

4.4 Alert and Notification

4.4.1 Description
Allow users to get alerts and notifications regarding their respective instance details.

4.5 Metering and Billing

4.5.1 Description

Allow user to see their usage and an estimated bill

4.5.2 Stimulus/Response Sequences

VStak will initial a API call to OpenStack to collect user usage on following metric

- Uptime
- CPU
- Memory
- Hard Disk

Billing will then be calculated base on this formula:

Expense = Uptime(hours) * nCPU * m.512MB Memory * cost per Unit

*Cost per Unit = 10 cent

4.5.3 Functional Requirements

This feature utilize the following API/Cli command:

g) Get usage of VM instances:

GET http://{{Host}}:8774/v2.1/os-simple-tenant-usage/<tenant ID>

5. Other Nonfunctional Requirements

5.1 Performance Requirements

No performance requirement but extensive error handling is required as OpenStack can produce wide range of issues.

5.2 Safety Requirements

OpenStack snapshot should be taken daily and before running this application

5.3 Security Requirements

This application is run in lab environment and doesn't spend any serious effort on security issues for now

5.4 Software Quality Attributes

Good error handling and don't crash too often.

5.5 Business Rules

This application is run on lab environment for experiment purpose, therefore, not comply to any specific business rule

6. Implementation

6.1 Project Team Bibliography

- Tran Pham
  SJSU ID: 011345646
- Pooja Shah
  SJSU ID: 010828311
- Prajakta Morale
  SJSU ID: 010834369
- Monika Dudhmogre
  SJSU ID: 011502699
- Amit Kumar Rajoriya
  SJSU ID: 011498201
- Anvit Saxena
  SJSU ID: 010953436
- Sneha Kasetty Sudarshan
  SJSU ID: 010128950
- Mohamad Shafaat Ali Khan
  SJSU ID: 011480430

6.2 Task break-down

#	Task	Description	Asgn.	Status
1	Document	Project Report: Abstract, introduction, design, diagram, screenshots, content, UML diagram(if applicable), challenges	Tran	Completed
2	OpenStack API	Study Openstack Rest API Interface, code the feature that use Rest API Interface	Pooja, Amit, Tran	Completed
3	OpenStack Cli	Study Openstack Command Line Int. , code the	Sneha	Completed

		feature that uses command line Interface		
5	Design Compute Services	Design application services that receive request from Front-end and make call to OpenStack to do the work: #1 - OpenStack API services #2 - OpenStack Cli services #3 - "Orchestration" service that control other services to complete the big task (ex: build big web service require build multiple instance and network link between them)	Pooja, Tran, Sneha	Completed
6	Front-end Development	Create mock-up screen to describe project features Designing using JavaScript, CSS OR AngularJS (TBD). This includes designing Login, Dashboard, Health check, interface pages	Anvit, Monika, Prajakta	Completed
7	Study OpenStack compute node	Study if we can deploy openstack with more than 1 compute node	Mohamad	Completed
8	Back end Development 1	Prepare the image instance for instances (web, db server, web & db server)	Mohamad	Completed
9	Shell Scripting	Application Deployment using shell scripts	Amit	Completed
10	Back end Development 2	The service that implement this interface is the core service that connect UI with RestAPI, Cli and database (Spring Implementation)	Pooja	Completed
11	Testing	Test cases	Pooja	Completed
11	PowerPoint Presentation	Prepare the presentation slide, final summary present in class.	Tran	Completed