*Florida International University*
*School of Computing and Information Sciences*

Software Engineering Focus

# Final Deliverable

## Virtual Machine Administration with Xen (VMAX) 1.0

**Team Members:**
D'Mita Levy
Dennis Obando
Qixiu Xin

**Product Owner(s)**:
Himanshu Upadhyay

**Mentor(s)**:
Walter Quintero

**Instructor**: Masoud Sadjadi

***Abstract***

*The Applied Research Center (ARC) at Florida International University has requested the development of an integrated management system to support research conducted on the Xen hypervisor platform. The current system is cumbersome and researchers have to rely heavily on multiple third-party applications to perform common virtual machine management and configuration tasks. The Virtual Machine Administration with Xen (VMAX) project was created to solve this problem; its primary goal is to develop a streamlined Xen virtual machine management system that is extensible and easy to use. To create the VMAX system, ARC has employed the assistance of the College of Engineering and Computer Science at Florida International University to design, deploy and develop key areas of the project.*

*This document provides a solution in the Virtual Machine Administration with Xen 1.0 system, a management interface and server application that allows users to configure, query and control the lifecycle of Xen hypervisor virtual machines and hosts. It provides hardware specifications, software specifications, software design documentation and describes validation tests performed on the system. In particular, the last few sections of the document contain appendices with Unified Modeling Language diagrams and a visual guide to system operation.*

## Table of Contents

# INTRODUCTION

Virtual Machine Administration with Xen (VMAX) 1.0 system is comprised of three key components: a web client, Windows application client and a remote server listener. The web client is provided as a way to circumvent the need for system administrator configuration of internal network device ports and Internet Protocol (IP) addresses; since web applications are typically unblocked in a private network environment. The Windows application client provides all the features and functionality of the web client with the addition of being able to poll Xen hypervisor hosts in the system and maintain state/status data about virtual machines. Although, some internal network configuration and access control may be necessary to run the Windows client application in more secure network environments. Lastly, the remote server listener resides on the Xen hypervisor host(s) and executes the requests of either connecting client through a targeted implementation of the Libvirt Virtualization Application Program Interface (API).

## Current System

The current system is a combination of third-party applications and manual commands entered into the console terminal of the host server. The Virtual Machine Manager (Virt-Manager) application is used to manage virtual machines on the Xen hypervisor and is powerful enough to perform all machine control functions. Yet, Virt-Manager has one key drawback: the most feature rich version can only be installed on a Linux based system thus requiring a virtual machine be installed on the administrative workstation if an administrator would like to use a Windows based system.

In addition, the Xen hypervisor host's terminal console will default to the root super-user and is always available. Therefore, any user that has physical access to the system can attempt to login or break into the system from the readily available command line prompt. The always available always accessible nature of the host machine is a security flaw that merits additional steps to remedy. Also, host queries such as performance statistics, disk usage and configuration require a secure shell (SSH) remote connection to the host machine or for the administrative user to login to the machine and execute commands manually.

## Purpose of New System

The purpose of VMAX 1.0 is to provide a simple client-server system that allows an administrator to control Xen hypervisor machines in a secure and robust way. VMAX has multiple clients that can each provided different ways of accessing a host and its virtual machines. The inclusion of a Go programming language server executable on the host means VMAX provides an extensible and native solution as well as avoiding some of the pitfalls that come with deploying software that requires installation.

The server side Go implementation of VMAX also provides a shield against malicious users that seek to exploit the Xen hypervisor's terminal console availability while providing an intuitive interface for configuring the server itself. As an added benefit to would be organizational users, VMAX uses the open-source Xen hypervisor as a platform, and the open-source Libvirt Virtualization API to issue commands to the host hypervisor. The use of open-source programming code means future versions of VMAX will adapt to changes easily as well as being cost effective.

# USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the Virtual machine administration with Xen Project Version 1.0 project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

### *Sprint 1*

**User Story#121-Access Virtual Machines**
As an administrator, I would like to be able to access the existing virtual machine so that I can retrieve the status of those machines.

**User Story#122-View Host Virtual Machines**
As an administrator, I would like to be able to visualize the virtual machines on a host so that I can manage those machines.

**User Story#141- View Settings of Configuration File**
As an administrator, I would like to have a web-based dashboard so that I can have online access to my virtual machine hosts.

**User Story#142- Building web based dashboard for VMs**
As an administrator, I would like to be able to see the current settings used by the host security agent so that I can confirm they are correct.

### *Sprint 2*

**User Story#143-Start a Virtual Machine**
As an administrator, I would like to be able to start a virtual machine so that the machine will be running and available for others to use.

**USER STORY#144- Stop a virtual machine**
As an administrator, I would like to be able to stop a virtual machine so that it is no longer running and available for others to use.

**User Story#145-Pause a Virtual Machine**
As an administrator, I would like to be able to pause a virtual machine so that its running state is saved and it can be resumed.

**User Story#146-Resume a Virtual Machine**
As an administrator, I would like to be able to resume a paused virtual machine so that it is running and once again available for use.

**User Story#155-Load and View VMAX Settings**
As an administrator, I would like to be able to load and view the settings of the VMAX application from a configuration file.

**User Story#150-Edit the settings of configuration file**
As an administrator, I would like to be able to edit the current settings used by the host security agent so that they are correct.

**User Story#164-Get the list of virtual machines for web dashboard**
As an administrator, I want to use the web application to get the list of virtual machines from the hypervisor so that user can do the connection through the interface.

**User Story#163-Update the VMAX database**
As an administrator, I would like to update the VMAX database with the retrieved list of virtual machines.

**User Story#153- Display the virtual machines on the web interface**
As an administrator, I wish to display the virtual machines on the web interface so that user can directly control the virtual machines through the interface.

## *Sprint 3*

**User Story#232-Create a New Virtual Machine**
As an administrator, I would like to be able to create a virtual machine so that others may use it.

**User Story#233-Force Shutdown Virtual Machine**
As an administrator, I want to be able to force a virtual machine to shut down so that I will be able to stop it even when it is not responding

**User Story#234-Create Virtual Machine Templates**
As an administrator, I would like to create virtual machine templates so that I can create new virtual machines.

**User Story #235-Display Virtual Machine Information**
As an administrator, I would like to have an interface for created virtual machines so that I can confirm the machine information is correct.

**User Story #236-Create a User Interface for the configuration file editing**
As an administrator, I would like to have a user interface to edit the configuration file.

**User Story #237-Use interface to edit Host IP Address**
As an administrator, I would like to edit the host IP address of the configuration file through the user interface.

**User Story #238-Use interface to edit Port Number**

As an administrator, I would like to edit the host port address through the user interface.

**User Story #275-List Virtual Machine Disk Configurations**
As an administrator, I want to store virtual machine disk configurations so that I can make new virtual machines.

**User Story #270-Modify the code for the .net format of the web**
As an administrator, I would like to modify the code for the .net format of the web, so that the user can open the website and modify it in Visual studio .net format.

**User Story #271-Add pause functionality; also will add resume functionality**
As an administrator, I would like to add pause functionality; Also, I will add resume functionality so that user can directly control the VMs for pausing and resume.

**User Story #272- Add stop functionality**
As an administrator, I would like to add stop functionality so that user can directly stop the functionality through web dashboard.

**User Story #273-Update database after each function call; Also, try to display the current virtual machine status**
As an administrator, I would like to update database after each function call; Also I will display the current VM status so that user can see the form for all VM information from html dashboard.

## *Sprint 4*

**User Story #312 - Create Virtual Machine from Existing Disk Image**
As an administrator, I would like to create a virtual machine from a pre-installed operating system image so that I will not have to go through the initial installation process.

**User Story #313 - Transfer Virtual Machine Images**

As an administrator, I would like to be able to deploy pre-install virtual disk images to the hypervisor so that I will be able to use those images in new virtual machines.

**User Story #314 - Clone Virtual Machine**

As an administrator, I would like to clone a virtual machine so that I will be able to have copies of an existing virtual machine.

**User Story #315 - Control Remote Virtual Machine**

As an administrator, I would like to manage a virtual machine remotely so that I will be able to work from that machine.

**User Story #317 - Display what is typed by the user into the interface**

As an administrator, I would like to see what is typed into the interface when changing values so that I can know the change on time.

**User Story #318 - Validate input values for both IP address and Port number**

As an administrator, I would like the user interface to validate that the correct format of the input values for both IP address and Port number.

**User Story #319 - Enter all the values to UI and update through one submission**

As an administrator, I would like to enter all the values needed to be edited at once and update file with one submission.

**User Story #320 - Clean up user interface**

As an administrator, I would like to clean up user interface so that the final web design looks nicer.

**User Story #321 - Implement “Create” Virtual Machine**

As an administrator, I would like to implement "create" Virtual Machines so that user can directly control the VM for" creating".

**User Story #322 - Display status of the VM machine**

As an administrator, I would like to display status of the virtual machines so that user can have a direct view for control VMs.

## *Sprint 5*

**User Story #359 - Delete Virtual Machine**

As an administrator, I would like to be able to delete a virtual machine so that I can remove it from a host permanently.

**User Story #360 - Display Performance Parameters**

As an administrator, I would like to be able to see performance metrics for each machine so that I can confirm they are operating correctly.

**User Story #186 - Display User Notifications**

As an administrator, I would like to log and be notified of any errors or messages that might occur while using the VMAX application so that I can track any potential problems.

**User Story #200 - Select VMAX Network Interface**

As an administrator, I want to select the VMAX applications local IP and port so that I can ensure it is connected to the right network interface.

**User Story #377-Add more information on the home page**

As an administrator, I would like to add the project general information to the home page so that user can see the whole picture and get the contact information easily.

**User Story #378-Design interface the existing VHD**

As an administrator, I would like to design an interface for listing the existing VHD so that user can directly check the list through interface.

**User Story $379-Create a VM from the existing VHD**

As an administrator, I would like to add the function to create a virtual machine from existing VHD so that the user can directly create the VM.

**User Story #369 - Clean up the Interface**

As an administrator, the User Interface needs to be cleaned up to be more user friendly.

**User Story #370 - Update all parameters with single selection**

As an administrator, I would like to update the parameters with a single selection.

**User Story #371 - View file should be a separate button/selection**

As an administrator, the view entire file should be a button/selection in itself.

**User Story #372 - Add a signal handler to catch the quit options**

As an administrator, I would like to have a signal handler for the quit option.

**User Story #377- Add more information on the home page**

As an administrator, I would like to add the project general information to the home- page so that user can see the whole picture and get the contact information easily.

**User Story #378- Design an interface for listing the existing VHD**

As an administrator, I would like to design an interface for listing the existing VHD so that user can directly check the list through interface.

**User Story #379- Create a virtual machine from existing VHD**

As an administrator, I would like to add the function to create a virtual machine from existing VHD so that the user can directly create the VM.

## *Sprint 6*

**User Story #417- Create Universally Unique Identifier's**

As an administrator, I would like to have Universally Unique identify for each host machine so that I can distinguish each host on the network.

**User Story #148- Display host remote configuration**

As an administrator, I would like to be able to see the host machines remote configuration so that I can ensure it is accurate.

**User Story #188- Add host to VMAX settings**

As an administrator, I would like to be able to add a host to the VMAX application so that I can interact with that host through the user interface.

**User Story #418- Display host disk usage**

As an administrator, I would like to be able to view the remote host disk usage so that I can monitor the free space left on the host machine.

**User Story #420- Design the interface for multiple inputs and update**

As an administrator, I would like to have multiple input boxes for the different parameters and update.

**User Story #421- Design the interface for user display**

As an administrator, I would like to refine the user interface for user display so that user can see better view for the interface.

**User Story #422- Delete Virtual Machine**

As an administrator, I would like to delete virtual machine so that user can control the virtual machine directly through the virtual machine list on the interface.

**User Story #424- Display hypervisors from the network using Tree View Control**

As an administrator, I would like to display hypervisors from the network using tree view control so that user can see the hypervisors clearly.

## Pending User Stories

**User Story #187- Handle host network errors**

As an administrator, I would like to be notified of any errors in the host ecosystem so that I can troubleshoot them if necessary.

**User Story #156- Edit VMAX Settings File**

As an administrator, I would like to be able to visually edit the VMAX application configuration files from its user interface so that I will be able to change the settings.

# PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint planning's are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The next section details a list of all hardware and software resources that were used in this project. Developers will also need a firm understanding of the following frameworks and languages: Go, C#, .NET, WPF, Visual Studio, ASP.NET, and Xen hypervisor Type 1 virtualization.

## Development Environment

To develop the system, the following minimum hardware and software is required and approximates the environment used to develop version 1.0 of the system:

Xen hypervisor host

- 6th or 7th Generation Intel processor (i5,i7 etc) with at least 8 cores
- 8 GB Primary Memory (RAM)
- 500 GB Secondary Memory (Hard Disk)
- 10 GB PCIe network adapter
- Motherboard with built in graphics
- 1 DVD +RW optical disk drive
- Keyboard
- Mouse
- Monitor

Required host software libraries and platform. All library files listed exist as a part of the standard GNU Linux distribution version repositories.

- Debian Linux distribution, CentOS 7+ recommended
- Windows operating system installation images
- Linux operating system installation images
- Go programming language version 1.6.3+
- Libfuse/SSHFS file system library
- OpenSSH secure shell library
- Virtual Machine Manager application
- Libvirt Virtualization API library 1.3.0+
- Xen project hypervisor version 4.6.0+
- Git revision control

VMAX web client

- Windows 7+ operating system platform
- 6th or 7th Generation Intel processor (i5,i7 etc)
- 4 GB Primary Memory (RAM)
- 100 GB Secondary Memory (Hard Disk)
- Visual Studio 2015+ Professional Edition
- Microsoft .NET framework version 4.0+
- Microsoft ISS server (remote or local)
- Git SCM revision control
- XenMaster VMAX library
- Putty terminal emulator

VMAX Windows client

- Windows 7+ operating system platform
- 6th or 7th Generation Intel processor (i5,i7 etc)
- 4 GB Primary Memory (RAM)
- Git SCM revision control
- 100 GB Secondary Memory (Hard Disk)
- Visual Studio 2015+ Professional Edition

- Microsoft .NET framework version 4.0+
- XenMaster VMAX library
- Renci.SshNet dynamic link library (DLL)
- VncSharpWPF dynamic link library (DLL)

Go server user interface

- Any operating system that can run Go
- 6th or 7th Generation Intel processor (i5,i7 etc)
- 4 GB Primary Memory (RAM)
- 100 GB Secondary Memory (Hard Disk)
- Gocui – Go Console User Inteface library by jroimartin

## Deployment Requirements

The system uses the Xen hypervisor as the centralized location for execution and development of code. Xen is a Type 1 "bare-metal" hyper and unlike Type 2 hypervisors such as VmWare's virtual box, it requires a stand-alone machine for operation. Because of this, developers will need to remote access the machine through the organization's network to run any code that will execute on the server. Login information on per user or group basis will be required once connected to the host and groups should plan accordingly.

In addition, the web client uses a SQL Server database that is proprietary to the Applied Research Center at Florida International University and special access will have be granted through their network administrator.

## Sprints Plan

These are the user stories that were released per sprint.

### *Sprint 1*

(08/27/2016 - 09/09/2016)

**User Story#121-Access Virtual Machines**

**Acceptance Criteria:**

- C# .NET Class library can connect to and communicate with a Xen host via TCP
- C# .NET Class library can retrieve host and virtual machine information

**Related Tasks:**

- Extend existing XML API
  Develop Management class to monitor Xen hosts
  Develop XenMaster C# XML API host manager class
  Modify Go Lang server to do synchronous message handling
  Setup Go Lang server and security agent file
  Create in-house source control
  Setup .NET WPF Project
  Testing
  Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 1 and Dynamic UML Diagrams, Figure 51.

**User Story#122-View Host Virtual Machines**

**Acceptance Criteria:**

- WPF application can connect to a Xen host server
- Host and Virtual machines are populated in the UI after connect to a Xen host server

- Host and Virtual machine information is visible in the UI

**Related Tasks:**

- Synchronize host and Node and VM details
  Retrieve host Node and VM details to display in interface
  Bind WPF elements for displaying VM/Host info to controller
  Add .NET controller event handling
  Develop .NET MVC controller bindings
  Design WPF Interface
  Testing
  Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 2 and Dynamic UML Diagrams, Figure 52.

**User Story#141- View Settings of Configuration File**

**Acceptance Criteria:**

- Golang program to read the configuration file and display settings of the server based on what the user chooses

**Related Tasks:**

- Test code with sample configuration file
- Write up code so user can select setting that they would like to see
- Write up code to read specific parts of file such as database server IP Address
- Write up code to read file
- Go over Golang to familiarize with the language

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 3 and Dynamic UML Diagrams, Figure 53.

**User Story#142- Building web based dashboard for VMs**

**Acceptance Criteria:**

- Interface uses CSS and html to display entry options for the user

- Include six tasks for create, delete, stop, snapshot, start, resume

**Related Tasks:**
- Design the mobile and pc view for dashboard
- Adding labels
- Create basic design for the web based dashboard
- Testing
- Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 4 and Dynamic UML Diagrams, Figure 54.

***Sprint 2***

(09/10/2016 - 09/23/2016)

**User Story#143-Start a Virtual Machine**

**Acceptance Criteria:**
- The Xen Connect library function sends an XML API based message to the host to start the virtual machine
- The virtual machine is started successful started (Running state)
- The WPF interface shows the virtual machine has started

**Related Tasks:**
- Add Context Click menu to virtual machine's in list to start virtual machine
- Document XML API
- Modify C# function to parse and verify XML response
- Documentation
- Unit Tests
- Bind WPF to code behind functions
- Design WPF interface elements to show that the virtual machine has started
- Add a function to Xen Master to send start XML request to Xen Host

- Define XML API object for Start to Xen Master

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 5 and Dynamic UML Diagrams, Figure 55.

**USER STORY#144- Stop a virtual machine**

**Acceptance Criteria:**

- The XenMaster class sends an XML message to the host to stop the running machine
- The virtual machine is stopped
- The WPF interface shows that the virtual machine has stopped

**Related Tasks:**

- Create Thread/Task to execute all context click functions asynchronously
- Documentation
- Testing
- Design WPF interface elements to show the virtual machine has started
- Bind WPF code behind functions
- Add a function to XenMaster to send XML stop request to the host
- Define XML API object for Stop in Xen Master

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 56 and Dynamic UML Diagrams, Figure 56.

**User Story#145-Pause a Virtual Machine**

**Acceptance Criteria:**

- The running virtual machine is paused
- The class library sends an XML message to the host to pause the machine
- The WPF interface shows that the virtual machine has been paused

**Related Tasks:**

- Modify vm listview to show vm is paused
- Add context click event handling for pause to list view
- Bind WPF to code behind
- Add pause vm function to Xen Master
- Define XML API pause vm object
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 7 and Dynamic UML Diagrams, Figure 57.

**User Story#146-Resume a Virtual Machine**

**Acceptance Criteria:**

- The Xen Master library sends an XML message to resume a paused machine
- The paused virtual machine returns to the running state
- The WPF interface shows the virtual machine is running

**Related Tasks:**

- Add interface elements to show vm resumed
- Bind WPF to code behind
- Add context click menu to resume vm
- Add XML API object builder for resume vm function
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 8 and Dynamic UML Diagrams, Figure 58.

**User Story#155-Load and View VMAX Settings**

**Acceptance Criteria:**

- The application settings are loaded from the configuration file
- The VMAX application uses the settings during its operation
- The user can view the current application settings
- The application settings are maintained in the file

**Related Tasks:**

- Design a C# class that will maintain and read the settings from configuration
- Bind interface settings display to code behind
- Design interface to display settings to the user
- Create application settings
- Research WPF App Configuration Manager
- Testing
- Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 10 and Dynamic UML Diagrams, Figure 60.

**User Story#150-Edit the settings of configuration file**

**Acceptance Criteria:**

- Edit the configuration file properties
- Edit the host IP address and port
- Edit the security agent IP address and port

**Related Tasks:**

- Using Golang function edit the security agent IP address and port
- Using Golang function edit the host IP address and port
- Edit the configuration file properties using Golang function

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 9 and Dynamic UML Diagrams, Figure 59.

**User Story#164-Get the list of virtual machines for web dashboard**
**Acceptance Criteria:**

- Adjusting web pages
- Connecting the web interface to the C# function library
- Changing the HTML format to ASP.NET
- Finishing the testing of the connection
- Finishing the documentation

**Related Tasks:**

- Adjusting web pages
- Documentation
- Connecting the web interface to the C# function library
- Testing the connection
- Changing the HTML format to ASP.NET

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 13 and Dynamic UML Diagrams, Figure 63.

**User Story#163-Update the VMAX database**
**Acceptance Criteria:**
Adjust the reference, configuration file
Connecting to the VMAX database of SQL server to retrieve list of virtual machine and related information
Testing and finish the documentation
**Related Tasks:**

- Connecting to the VMAX database of SQL server to retrieve list of virtual machine and related information
- Adjust the reference and configuration
- Documentation
- Testing to see if the database can be updated

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 12 and Dynamic UML Diagrams, Figure 62.

**User Story#153- Display the virtual machines on the web interface**
**Acceptance Criteria**:

- Creating and adding table in Visual Studio

- Connecting to the library, adjusting the columns of the table to show the current data for the virtual machine name, state, CPU, MAX Memory and so on
- Testing and finish the documentation

**Related Tasks:**

- Adding tables and adjust
- Design the tables
- Display the VM machine list table on interface
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 17 and Dynamic UML Diagrams, Figure 61.

### *Sprint 3*

(09/24/2016 - 10/07/2016)

**User Story#232-Create a New Virtual Machine**

**Acceptance Criteria:**

- The VMAX interface has all the necessary parameters to create the virtual machine
- The VMAX interface uses the XenMaster library to create a new virtual machine
- Created virtual machines are started in the "Running" state

**Related Tasks:**

- Add Go lang function to create virtual machine
- Add C# function to send virtual machine creation parameters
- Bind interface to C# codebehind
- Design WPF interface window to create new machine
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 14 and Dynamic UML Diagrams, Figure 64.

**User Story#233-Force Shutdown Virtual Machine**
**Acceptance Criteria:**

- The XenMaster class library uses XML messaging to shut down the virtual machine
- The virtual machine shuts down during any normal life cycle events
- The virtual machine shuts down during erroneous life cycle events
- The vmax interface has contextual menu options for shutting down the virtual machine

**Related Tasks:**

- Add menu item event handler to vm listview
- Bind codebehind to wpf interface
- Add Force Shutdown function to the XenConnect class
- Confirm connectivity using Xen Master test driver
- Create xml api message builder for destroy vm
- Add server destroy function and xml object handler
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 15 and Dynamic UML Diagrams, Figure 65.

**User Story#234-Create Virtual Machine Templates**

**Acceptance Criteria:**

- Template function uses Go Lang XML marshalling and unmarshalling of Go structs
- Generates Xen virtual machine XML template based on the Libvirt API specification
- Template can be used to to create a new virtual machine using the Libvirt API

**Related Tasks:**

- Develop vm template parameter parsing and generation function
- Documentation
- Testing
- Design virtual machine template
- Research virtual machine creation templates
- Research virtual machine storage pooling

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 16 and Dynamic UML Diagrams, Figure 66.

**User Story #235-Display Virtual Machine Information**

**Acceptance Criteria:**

- VMAX displays host virtual machine information in a new window

**Related Tasks:**

- Bind to contextual menu
- Add WPF Design for vm info window
- Add server function to view created vms
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 17 and Dynamic UML Diagrams, Figure 67.

**User Story #236-Create a User Interface for the configuration file editing**

**Acceptance Criteria:**

- User interface that a user can view the configuration file

- Using golang functions to edit the file using the interface

**Related Tasks:**

- Documentation
- Test
- Fit user interface for user story requirements
- Put together code for user interface
- Test different examples of code from Github
- Research different GUI repositories for Golang

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 18 and Dynamic UML Diagrams, Figure 68.

**User Story #237-Use interface to edit Host IP Address**
**Acceptance Criteria:**

- Using the user interface the user should be able to edit the host IP address of the configuration file

**Related Tasks:**

- Add previous code for editing IP address to user interface code

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 19 and Dynamic UML Diagrams, Figure 69.

**User Story #238-Use interface to edit Port Number**
**Acceptance Criteria:**

- Using the user interface the user should be able to edit the host port number of the configuration file

**Related Tasks:**

- Add previous code for editing port number to user interface
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 20 and Dynamic UML Diagrams, Figure 70.

**User Story #275-List Virtual Machine Disk Configurations**
**Acceptance Criteria:**

- VMAX application displays virtual machine disk image configurations and iso files
- XenMaster library retrieves virtual machine disk and ISO image files

**Related Tasks:**

- Design WPF Interface to display Disk configurations
- Bind display function to WPF codebehind
- Add C# function to XenMaster to retrieve iso images
- Create XML API function to return iso storage list
- Create function to return vm iso storage list
- Add XML API functions to return and create disk images
- Create function to return vm volume image list
- Wrap linux dd function to create vm .img files
- Research machine libvirt-go storage pools api
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 23 and Dynamic UML Diagrams, Figure 73.

**User Story #270-Modify the code for the .net format of the web**
**Acceptance Criteria:**

- Adjusting the interface for better view
- Modify and update the code for the .net format
- Finishing the testing of the web interface display
- Finishing the documentation

**Related Tasks:**

- Redesign the web interface
- Testing
- Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 21 and Dynamic UML Diagrams, Figure 71.

**User Story #271-Add pause functionality; also will add resume functionality**

**Acceptance Criteria:**

- Adjusting the interface for better view
- Modify and update the code for the .net format
- Finishing the testing of the web interface display
- Finishing the documentation

**Related Tasks:**

- Testing for both functions
- Do the documentation
- Add Resume Function
- Trying to connect with the back library
- Add Pause function

**Modeling:** No diagrams

**User Story #272- Add stop functionality**

**Acceptance Criteria:**

- After user click the stop button, the VMSTATE column change from 3 to 1

**Related Tasks:**

- Do the documentation
- Testing for the "stop" function
- Add function
- Connecting and adjusting the back library

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 22 and Dynamic UML Diagrams, Figure 72.

**User Story #273-Update database after each function call; Also, try to display the current virtual machine status**

**Acceptance Criteria:**

- Updating the database after each function calling
- Display the current VM information and status
- Finishing the testing to see if each function works ; also to see if all the database works
- Finishing the documentation

**Related Tasks:**

- Display current vm status
- Update database
- Testing
- Documentation

***Sprint 4***

(10/08/2016 - 10/21/2016)

**User Story #312 - Create Virtual Machine from Existing Disk Image**

**Acceptance Criteria:**

- VMAX 'New VM Wizard' has options to allow the creation of virtual machines from existing disk image
- Virtual machines are created and booted without requiring the completion of the operating system's installation process

**Related Tasks:**

- Add copy existing disk function to Go Lang codebase
- Modify New VM wizard with use existing disk options
- Modify VirtualMachineBuilder class to allow boot from disk instead of cdrom option
- Changed vm template options to boot from disk instead of cdrom
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 24 and Dynamic UML Diagrams, Figure 74.

**User Story #313 - Transfer Virtual Machine Images**

**Acceptance Criteria:**

- VMAX interface the allows the user to select local disk images and transfer remotely to the hypervisor
- XenMaster class library is used to transfer the disk image
- Virtual disk image transfer will not overwrite existing disk images

**Related Tasks:**

- Bind file transfer interface to C# codebehind
- Design WPF file transfer interface
- Extend XML API to do file transfers
- Research Go Lang FTP server libraries
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 25 and Dynamic UML Diagrams, Figure 75.

**User Story #314 - Clone Virtual Machine**

**Acceptance Criteria:**

- VMAX application has interface options to clone existing virtual machine
- XenMaster class library function clones virtual machine

**Related Tasks:**

- Add C# clone vm function
- Bind WPF Interface to C# code behind
- Add clone vm context menu to virtual machine list
- Add clone virtual machine to server
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 26 and Dynamic UML Diagrams, Figure 76.

User Story #315 - Control Remote Virtual Machine

**Acceptance Criteria:**

- Virtual machine desktop, mouse, keyboard can be controlled remotely
- Remote connection uses SSH tunneling
- Remote connection uses RFB (VNC) protocol

**Related Tasks:**

- Modify secure tunneling management to skip used ports
- Bind interface to codebehind
- Design interface to show remote desktop
- Integrate vnc library into XenMaster
- Implement Remote Viewing library
- Research vnc libraries
- Research vnc tunneling
- Documentation
- Testing

**Modeling:** No diagrams

**User Story #317 - Display what is typed by the user into the interface**

**Acceptance Criteria:**

- User is able to see what they type for editing configuration file

**Related Tasks:**

- Documentation
- Test
- Create Textbook for Host Port Number
- Create Textbox for Host IP Address

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 27 and Dynamic UML Diagrams, Figure 77.

**User Story #318 - Validate input values for both IP address and Port number**

**Acceptance Criteria:**

- Correct format validation for IP address
- Correct format validation for Port number

**Related Tasks:**

- Test
- Validate proper parameters for Host Port Number
- Validate proper parameters from Host IP address

**User Story #319 - Enter all the values to UI and update through one submission**

**Acceptance Criteria:**

- Update file through one submission of both IP address and Port number

**Related Tasks:**

- Documentation
- Test
- Implement previous functions to submit option
- Create submit option
- Read from the Textbox

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 29 and Dynamic UML Diagrams, Figure 79.

**User Story #320 - Clean up user interface**

**Acceptance Criteria:**

- Delete duplicate design buttons and function lines
- Interface looks more clear -forms of VM

**Related Tasks:**

- Documentation
- Testing
- Add connection from home page to the log in page
- Add home page information
- Check the code for the bootstrap part
- Delete the duplicate information

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 30 and Dynamic UML Diagrams, Figure 80.

**User Story #321 - Implement "Create" Virtual Machine**

**Acceptance Criteria:**

- Add " Create" functions
- The function should be tested successful
- **Related Tasks:**
  Implement "create "virtual machine
- Testing
- Documentation

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 31 and Dynamic UML Diagrams, Figure 81.

**User Story #322 - Display status of the VM machine**

**Acceptance Criteria:**

- Modify the column-VMSTATE
- Adding related functions for forms through bootstrap
- Create the direct view of control VMs

**Related Tasks:**

- Documentation
- Testing
- Add bootstrap code for the form on the interface

- Add status for the virtual machines

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 32 and Dynamic UML Diagrams, Figure 32.

### *Sprint 5*

(10/22/2016 - 11/04/2016)

**User Story #359 - Delete Virtual Machine**

**Acceptance Criteria:**

- The virtual machine XML configuration is removed from the host
- The user has the option to remove the virtual machine hard disk

**Related Tasks:**

- Bind WPF interface to codebehind
- Extend XML API function on server for delete vm
- Add C# delete function to XenMaster
- Add remove disk function to go lang server
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 35 and Dynamic UML Diagrams, Figure 49.

**User Story #360 - Display Performance Parameters**

**Acceptance Criteria:**

1. VMAX displays virtual machine performance and usage parameters

**Related Tasks:**

- Extend server api to include metrics functions
- Research libvirt api additional vm info nodes
- Add C# metric retrieval function to XenMaster
- Design interface to show metrics
- Add server function to retrieve vm tx and rx metrics
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 36 and Dynamic UML Diagrams, Figure 86.

**User Story #186 - Display User Notifications**

**Acceptance Criteria:**

- VMAX displays pop-up messages when an user error occurs
- VMAX displays pop-up messages when relevant user notifications occur
- VMAX logs errors/notifications and their levels in the application interface

**Related Tasks:**

- Bind logging to event notification and highlight security levels
- Add tooltips and generic application pop-up messages for background notification of errors
- Bind event handler to wpf code behind
- Add event handling and event delegate to Xen Connect
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 33 and Dynamic UML Diagrams, Figure 83.

**User Story #200 - Select VMAX Network Interface**

**Acceptance Criteria:**

- Bind logging to event notification and highlight security levels
- Add tooltips and generic application pop-up messages for background notification of errors
- Bind event handler to wpf code behind
- Add event handling and event delegate to Xen Connect

- Documentation
- Testing

**Related Tasks:**
- Design WPF elements to display network interfaces
- Modify VMAX code behind to accept network interface
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 34 and Dynamic UML Diagrams, Figure 84.

**User Story #377-Add more information on the home page**

**Acceptance Criteria:**
- Draw general project picture through draw.IO
- Adding information to the home page
- Testing successful

**Related Tasks:**
- Documentation
- Testing
- Add more information on the home page
- Clean up the interface

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 41 and Dynamic UML Diagrams, Figure 91.

**User Story #378-Design interface the existing VHD**

**Acceptance Criteria:**
- Design and add the interface for listing the existing VHD; After click" Create", there should be two buttons:ISO and VHD
- Successfully testing
- Finish documentation of the user story

**Related Tasks:**

- Documentation
- Do the testing for the interface
- Trying to adding the interface for the existing virual hard disk

**Modeling:** Refer to Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 42 and Dynamic UML Diagrams, Figure 83.

**User Story #379-Create a VM from the existing VHD**

**Acceptance Criteria:**

- After user click "Add virtual machine from VHD", the new virtual machine will be shown in the virtual machine list below.
- Finish testing
- Finish documentation

**Related Tasks:**

- Do the testing for create function
- Create a VM from existing VHD
-

**User Story #369 - Clean up the Interface**

**Acceptance Criteria:**

- The help options should be on the bottom
- The input boxes should be next to the options when selected

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 37 and Dynamic UML Diagrams, Figure 87.

**User Story #370 - Update all parameters with single selection**

**Acceptance Criteria:**

- One selection will bring up input boxes
- One selection will edit the configuration file

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 38 and Dynamic UML Diagrams, Figure 88.

**User Story #371 - View file should be a separate button/selection**

**Acceptance Criteria:**

- One button/selection for view file

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 39 and Dynamic UML Diagrams, Figure 89.

**User Story #372 - Add a signal handler to catch the quit options**

**Acceptance Criteria:**

- Yes/No option when quitting the interface

**Related Tasks:**

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 40 and Dynamic UML Diagrams, Figure 90.

**User Story #377- Add more information on the home page**

**Acceptance Criteria:**

- Draw general project picture through draw.IO
- Adding information to the home page
- Testing successful

**Related Tasks:**

- Documentation
- Testing
- Add more information on the home page
- Clean up the interface

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 41 and Dynamic UML Diagrams, Figure 91.

**User Story #378- Design an interface for listing the existing VHD**

**Acceptance Criteria:**

- Design and add the interface for listing the existing VHD; After click" Create", there should be two buttons: ISO and VHD
- Successfully testing

- Finish documentation of the user story

**Related Tasks:**

- Documentation
- Do the testing for the interface
- Trying to adding the interface for the existing virtual hard disk

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 42 and Dynamic UML Diagrams, Figure 92.

### *Sprint 6*

(11/05/2016 - 11/11/2016)

**User Story #417- Create Universally Unique Identifier's**
**Acceptance Criteria:**

- Host machine generates UUID and add's to configuration file if it is missing
- All host synchronization information requests return the UUID of the host

**Related Tasks:**

- Add UUID generation function to server
- Research machine specific UUID implementations
- Documentation
- Testing

**Modeling:** No diagrams

**User Story #148- Display host remote configuration**
**Acceptance Criteria:**

- Configuration file options: ip, port, etc are displayed in the interface
- Information is retrieved from the remote host

**Related Tasks:**

- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 43 and Dynamic UML Diagrams, Figure 93.

**User Story #188- Add host to VMAX settings**
**Acceptance Criteria:**
- Host is added to the VMAX settings file
- The new host shows up int he Xen Hosts list

**Related Tasks:**
- Bind WPF interface to code behind
- Add code behind to write to the existing configuration file
- Design WPF interface to accept host information
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 44 and Dynamic UML Diagrams, Figure 94.

**User Story #418- Display host disk usage**
**Acceptance Criteria:**
- VMAX mirrors the output of the df -h command on linux
- VMAX shows a graphical representation of the remote hosts disk usage
- VMAX shows remote host numerical disk usage

**Related Tasks:**
- Testing
- Documentation

**Modeling:** Refer to Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 46 and Dynamic UML Diagrams, Figure 83.

**User Story #420- Design the interface for multiple inputs and update**
**Acceptance Criteria:**
- Have two input boxes for IP Address and Port Number
- Have one submission

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 47 and Dynamic UML Diagrams, Figure 97.

**User Story #421- Design the interface for user display**

**Acceptance Criteria:**

- Make the user interface user friendly to use

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 48 and Dynamic UML Diagrams, Figure 98.

**User Story #422- Delete Virtual Machine**

**Acceptance Criteria:**

- Add" Delete" button for virtual machine list
- Add functions for virtual machine
- Do the testing
- Finish documentation

**Related Tasks:**

- Testing
- Add the function code for "delete"
- Add the button for "delete"

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 49 and Dynamic UML Diagrams, Figure 85.

**User Story #424- Display hypervisors from the network using Tree View Control**

**Acceptance Criteria:**

- The tree-view control should be added above the virtual machine list.
- Finish the testing for display hypervisors.
- Finish the documentation.

**Related Tasks:**

- show hypervisors from the network by tree view
- Documentation
- Testing

**Modeling:** Refer to UML diagrams in Appendix A: Static UML Diagrams, Figure 50 and Dynamic UML Diagrams, Figure 100.

# SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

VMAX 1.0 uses a simple client-server architecture. The system is a combination of a web client, windows application client and a server listener runtime on the host server. The web client is dependent upon a SQL Server database that is a proprietary installation of the Applied Research Center at Florida International University. The web client is built upon Microsoft's ASP.NET framework and uses a Microsoft Internet Information Services (IIS) server that is run locally. The web client uses the .NET Runtime version 4.6.1, is stand-alone once compiled and follows the client-server architectural model. The Go server listener performs all server tasks of the architecture and provides a subset of the capabilities of the host system.

## System and Subsystem Decomposition

XenMaster – The XenMaster C# library contains all the communication and connectivity functions for contacting the server listener. It communicates with the listener through a custom XML API and is the model layer for both the web and Windows application clients

Virsh Libvirt Service – The Virsh Libvirt service is internal to the host and VMAX server listener and is an extensive management library for the hypervisor.

Xen Hypervisor – Xen is a Type 1 "bare metal" hypervisor that does not run as a layer on top of an existing operating system. The hypervisor and host are synonymous in the system and all function calls from XenMaster terminate at the hypervisor

Libvir Listener – The Libvirt Listener is a custom class that handles and executes all XenMaster requests that come from the web client and Windows client. It wraps the functionality of the Libvirt virtualization API and makes calls directly to the Virsh Libvirt service.

Web Client - The web client is built on the ASP.NET framework and provides a user interface for connecting to and communicating with the Go server listener on the hypervisor host.

- Persistent Data – A proprietary implementation of the SQL Server database makes up the persistent data element necessary to run the web client.
- ASP.NET – Microsoft's standard framework for web hosting on the IIS system

Windows Client – The windows application client has all the functionality of the web client as well as the added benefit of maintain the state of each virtual machine through regular interval polling. It derives all of the same functions (start, stop, resume etc.) that are present in the Web client since it also uses the XenMaster library.

## Deployment Diagram

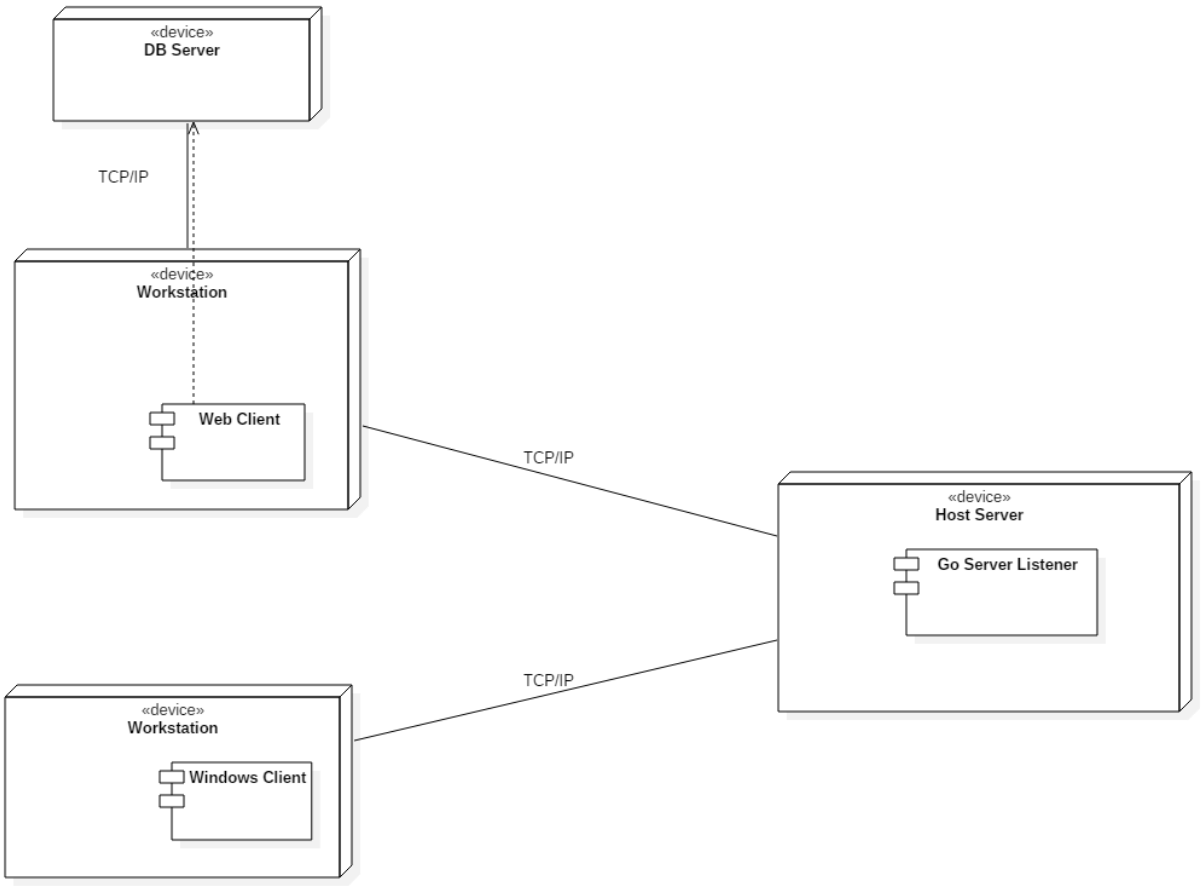



Figure A – General Deployment Diagram

## Design Patterns

Each of the following design patterns was used in the project to provide the shortest development time while still maintaining flexibility and use ability for future iterations:

- Observer Pattern – The Windows client application uses the Observer pattern to provide the easiest update between the View and View Model. The client executes regular interval polling of hosts and virtual machine and "Observable" objects from the .NET runtime were added to store View representations of physical devices in the system.

- Publish/Subcribe Pattern – The XenMaster class library that forms the basis for the XML communications API uses .NET runtime delegates/events to implement the Publish/Subscribe patterns. Subscribers can subscribe to various system events that are occurring during regular communication and choose the events that are relevant.

- Decorator Pattern – The Go server, as is the case with most Go programs, makes extensive use of the Decorator design pattern to add extensible functionality to the structs that make up the different objects used by the server.

## Package Diagram

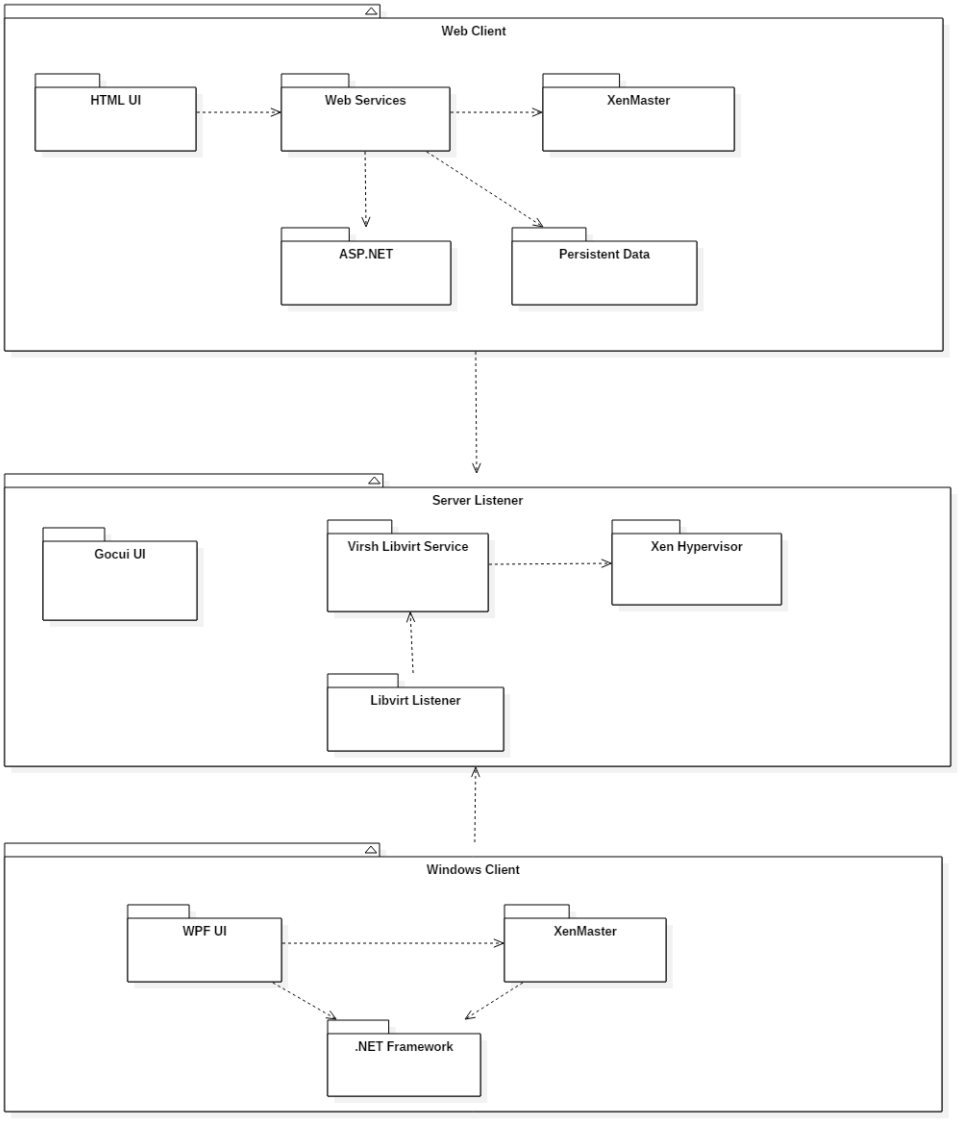



Figure B – Simplified Package Diagram

## SYSTEM VALIDATION

Unit test is essential to test all the layers within the application. All the test cases used to validate user stories developed are shown in this section.

**User Story #121 – Access Virtual Machines**

| **Test ID: U-121-1** | |
|---|---|
| **Purpose:** | Verify that XenConnect class can retrieve host and virtual machine details |
| **Preconditions:** | Xen host listener on the remote server is running |
| **Input:** | Host IP address |
| **Expected Output:** | Host details and virtual machine information |

**User Story #122 – View Host Virtual Machines**

| **Test ID: U-122-1** | |
|---|---|
| **Purpose:** | Verify the VMAX interface will display host information and virtual machine information |
| **Preconditions:** | VMAX application has started successfully and host list has been automatically updated |
| **Input:** | User selects a host from the host list |
| **Expected Output:** | VMAX ListView shows host virtual machine information |

**User Story #141 – View Settings of Configuration File**

| **Test ID: U-141-1** | |
|---|---|
| **Purpose:** | Verify that code can read file and display configuration settings |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file |
| **Input:** | User input of 1-4 |
| **Expected Output:** | Configuration information of the file being read |

| **Test ID: I-141-1** | |
|---|---|
| **Purpose:** | Verify that code can read file and display configuration settings in the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file contained in Xen Server |
| **Input:** | User input of 1-4 |
| **Expected Output:** | Configuration information of the file being read from the Xen Server directly |

**User Story #142 – Building web based dashboard for VMs**

| **Test ID: U-142-1** |
|---|

| | |
|---|---|
| **Purpose:** | Checking web interface design |
| **Preconditions:** | Finish the CSS/HTML code part and the basic web design framework is builded |
| **Input:** | User click connect button after typing in the user Host ID |
| **Expected Output:** | The VS information form list like CPU, mhz shows up |

| **Test ID: I-142-1** | |
|---|---|
| **Purpose:** | Checking web interface design |
| **Preconditions:** | Finish the css/html code part and the basic web design framework is builded, the web interface can be modified to both of Mobile reading format and the PC reading format |
| **Input:** | User click connect button after typing in the user Host ID |
| **Expected Output:** | The VS information form list like CPU, mhz, VMstate form shows up |

**User Story #143 – Start a Virtual Machine**

| **Test ID: U-143-1** | |
|---|---|
| **Purpose:** | Test the VMAX start virtual machine function |
| **Preconditions:** | • VMAX application has started successfully and host list has |

| | |
|---|---|
| | been automatically updated<br>• There is a vm that is shutdown |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Start” |
| **Expected Output:** | • The virtual machine icon turns to a blue color<br>• The PowerState of the virtual machine is “Running” |

**User Story #144 – Stop a Virtual Machine**

| **Test ID: U-144-1** | |
|---|---|
| **Purpose:** | Test the VMAX stop virtual machine function |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is a vm that is running |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Stop” |
| **Expected Output:** | • The virtual machine icon turns to a gray color<br>• The PowerState of the virtual machine is “Shutoff” |

**User Story #145 – Pause a Virtual Machine**

| **Test ID: U-145-1** | |
|---|---|
| **Purpose:** | Test the VMAX pause virtual machine function |

| | |
|---|---|
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is a vm that is shutdown |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Pause” |
| **Expected Output:** | • The PowerState of the virtual machine is “Paused” |

**User Story #146 – Resume a Virtual Machine**

| **Test ID: U-146-1** | |
|---|---|
| **Purpose:** | Test the VMAX resume virtual machine function |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is a vm that is paused |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Resume” |
| **Expected Output:** | • The PowerState of the virtual machine is “Running” |

**User Story #148– Display host remote configuration**

| **Test ID: I-148-1** |
|---|

| | |
|---|---|
| **Purpose:** | Test VMAX settings display |
| **Preconditions:** | • At least one host is available in the VMAX Xen Hosts list |
| **Input:** | 1. Select a host from the host list |
| **Expected Output:** | • The host settings will be displayed |

**User Story #150 – Edit the settings of configuration file**

| **Test ID: U-150-1** | |
|---|---|
| **Purpose:** | Verify that code can edit the file based on user input |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file |
| **Input:** | User input of 1-4 |
| **Expected Output:** | Configuration information of the file being edited to fit user expectation |

| **Test ID: I-150-1** | |
|---|---|
| **Purpose:** | Verify that code can edit the file based on user input on Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server |
| **Input:** | User input of 1-4 |

| | |
|---|---|
| **Expected Output:** | Configuration information of the file being edited to fit user expectation on Xen Server |

**User Story #153 – Display the virtual machines on the web interface**

| **Test ID: U-153-1** | |
|---|---|
| **Purpose:** | Display the virtual machines on the web interface |
| **Preconditions:** | Catch the current data flow from library |
| **Input:** | Users types in their host IDs |
| **Expected Output:** | Users can see all the information and see the VM list on the interface |

| **Test ID: I-153-1** | |
|---|---|
| **Purpose:** | Display the virtual machines on the web interface |
| **Preconditions:** | Catch the current data flow from library |
| **Input:** | Users types in their host ID for testing |
| **Expected Output:** | Users can see all the information and see the VM list on the interface includes the correct information from database after click connect |

**User Story #155 – Load and View VMAX Settings**

| **Test ID: U-155-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's ability to generate a settings file |
| **Preconditions:** | • The VMAX.cfg settings file does not exist<br>• The VMAX folder does not exist in the C:\ProgramData directory |
| **Input:** | The application is started |
| **Expected Output:** | • The application creates a "VMAX" folder in the C:\ProgramData directory<br>• The application generates a well-formed VMAX.cfg in the VMAX folder |

| **Test ID: U-155-2** | |
|---|---|
| **Purpose:** | Test the VMAX application's ability to display the values in the VMAX.cfg settings file |
| **Preconditions:** | • A well-formed VMAX.cfg settings file exists in the VMAX folder |
| **Input:** | The user clicks on the "Settings" menu item |
| **Expected Output:** | • A pop-up window shows the matching values from the VMAX.cfg settings file<br>• All settings values should be populated |

**User Story #163 – Update the VMAX Database**

| Test ID: U-163-1 | |
|---|---|
| **Purpose:** | Update the VMAX database with the retrieved list of virtual machines |
| **Preconditions:** | Connect to the library |
| **Input:** | Add reference and adjust configuration file |
| **Expected Output:** | The VMAX database can be successfully updated with the retrieved list of VMs |

| Test ID: I-163-1 | |
|---|---|
| **Purpose:** | Update the VMAX database with the retrieved list of virtual machines |
| **Preconditions:** | Connect to the VMAX library from Dmita |
| **Input:** | Add reference and adjust configuration file for the web interface |
| **Expected Output:** | The VMAX database can be successfully updated with the retrieved list of VMs |

**User Story #164 – Get the list of virtual machines for web dashboard**

| Test ID: U-164-1 | |
|---|---|
| **Purpose:** | Use the web application to get the list of virtual machines from the hypervisor |
| **Preconditions:** | Finish the web design and connecting to the hypervisor |

| | |
|---|---|
| **Input:** | Click connect |
| **Expected Output:** | The VS information shows up, also all the data will be show up on the interface. The data can successfully comes from the current operation of the VS machines. |

| **Test ID: I-164-1** | |
|---|---|
| **Purpose:** | Use the web application to get the list of virtual machines from the hypervisor |
| **Preconditions:** | Finish the web design and connecting to the hypervisor |
| **Input:** | User can double click “connect” |
| **Expected Output:** | The VS information should be successfully shows up, also all the data will be show up on the interface. The data can successfully comes from the current operation of the VS machines. |

**User Story #186 – Display User Notifications**

| **Test ID: I-186-1** | |
|---|---|
| **Purpose:** | Test the VMAX user notification logging |
| **Preconditions:** | • VMAX application has started successfully<br>• There is at least one host available in the network<br>• A host is configured in the VMAX configuration/settings file |
| **Input:** | 1. Select a host from the host list<br>2. Click the “Logs” tab |

| | |
|---|---|
| **Expected Output:** | • Logs showing the ISO disk retrieval and VHD retrieval status will be displayed:<br><br>`Notification ISO Retrieval Retrieved Host ISO storage List`<br>`Notification VHD Retrieval Retrieved Host VHD disk list` |

| **Test ID: I-186-2** | |
|---|---|
| **Purpose:** | Test the VMAX popup error/notification messages |
| **Preconditions:** | • VMAX application has started successfully<br>• There is no IP entry in the VMAX configuration/settings file |
| **Input:** | 1. Click the “Connection” dropdown box<br>2. Select the empty connection option (no ip address, blank) |
| **Expected Output:** | • A pop-up message will be displayed with the text :<br>`Please select a valid IP address interface to continue` |

| **Test ID: I-186-3** | |
|---|---|
| **Purpose:** | Test the VMAX popup notification messages |
| **Preconditions:** | • VMAX application has started successfully<br>• There is at least one host available in the network<br>• A host is configured in the VMAX configuration/settings file<br>• There is at least one host virtual machine in the “Shutdown” state |

| | |
|---|---|
| **Input:** | 1. Right-click the virtual machine to bring up the context menu<br>2. Click “Start” |
| **Expected Output:** | • A pop-up message will be displayed with the text : `The virtual machine with UUID [uuid] has been successfully started` |

**User Story #188 – Add host to VMAX settings**

| **Test ID: I-188-1** | |
|---|---|
| **Purpose:** | Test VMAX settings display |
| **Preconditions:** | • The new host does not already exist as a part of the VMAX configuration |
| **Input:** | 1. Click the “Add Host” button<br>2. Enter the host information<br>Name: “Test Host”<br>IP Address: “172.16.10.115<br>Description : “Test<br>3. Click the “Add Host” button in the Add New Host Window |
| **Expected Output:** | • The new host will be added to the VMAX application<br>• The new host information will be written to the VMAX configuration file<br>• The new host will show up the Xen Hosts list |

**User Story #200– Select VMAX Network Interface**

| **Test ID: U-200-1** | |
|---|---|
| **Purpose:** | Test VMAX network interface selection and connectivity |
| **Preconditions:** | • At least one network interface is available and IP/Port socket are not already open<br>• VMAX settings does not already have an IP entry for the “System IP” field |
| **Input:** | 1. Select a network interface from the “Connection” drop down<br>2. Click the “Connect” button |
| **Expected Output:** | • The “Connect” button will turn blue and its content will change to “Disconnect”<br>• Available hosts will load in the “Xen Hosts” list |

| **Test ID: U-200-2** | |
|---|---|
| **Purpose:** | Test VMAX network interface selection and connectivity error handling |
| **Preconditions:** | • At least one network interface is available and IP/Port socket are not already open<br>• VMAX settings does not already have an IP entry for the “System IP” field |
| **Input:** | 1. Select an empty network interface from the “Connection” drop down<br>2. Click the “Connect” button |
| **Expected Output:** | • A popup message with the text “Please select a valid IP Address interface to continue” will appear. |

| | |
|---|---|

| **Test ID: I-200-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's network interface selection capability |
| **Preconditions:** | • At least one network interface is available and IP/Port socket are not already open<br>• VMAX settings does not already have an IP entry for the "System IP" field |
| **Input:** | 1. Select a network interface IP Address from the "Connection" dropdown<br>2. Click the "Connect" button<br>3. Select a host from the "Xen Hosts" list |
| **Expected Output:** | • The "Connect" button's content will change to "Disconnect"<br>• The "Xen Hosts" list will be populated with hosts<br>• The selected host will display its virtual machine list |

**User Story #232 – Create a New Virtual Machine**

| **Test ID: I-232-1** | |
|---|---|
| **Purpose:** | Test the VMAX create new virtual machine wizard |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is at least one host in the list |

| | |
|---|---|
| | • The host has pre-loaded OS ISO images |
| **Input:** | 1. Click “Create VM” from the toolbar<br>2. Select TCC-9-SENIOR-PROJECT<br>3. Required Params:<br>• Name = SeniorBuntu<br>• OS Image = Ubuntu14-04<br>• Memory = 2048<br>• Virtual Disk = 21<br>• VCPU = 2<br>4. Click “Create VM” from the New Virtual Machine Wizard |
| **Expected Output:** | • The pop up message “The creation of virtual machine ‘SeniorBuntu’ has been completed successfuly” is displayed<br>• The virtual machine “SeniorBuntu” shows up in the host list |

**User Story #233 – Force Shutdown Virtual Machine**

| **Test ID: I-233-1** | |
|---|---|
| **Purpose:** | Test the VMAX force shutdown function on “running” state virtual machine |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is a vm that is running |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Force Shutdown” |
| **Expected** | • The virtual machine icon turns to a gray color |

| | |
|---|---|
| **Output:** | • The PowerState of the virtual machine is “Shutoff” |

| **Test ID: I-233-2** | |
|---|---|
| **Purpose:** | Test the VMAX force shutdown function on an error state virtual machine |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is a vm that is running<br>• The running virtual machine will not respond to the regular “Shutdown” requests |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Force Shutdown” |
| **Expected Output:** | • The virtual machine icon turns to a gray color<br>• The PowerState of the virtual machine is “Shutoff” |

**User Story #234 – Creating Virtual Machine Templates**

| **Test ID: I-234-1** | |
|---|---|
| **Purpose:** | Test the Go Lang create virtual machine template function |
| **Preconditions:** | • Host has the libvirt service installed and running |
| **Input:** | 1. The user enters the following params into the vmxml NewVMXML.go file<br>• Params<br>•<br>o vm_name : = "BuntuD" |

| | |
|---|---|
| | ○ vmConfig : = NewVmConfiguration = vm_name, "", "", VM_TYPE_LINUX<br>○ SetMaxMemory = 500000 //5 GB<br>○ SetVcpus = 1<br>○ SetOsParameters = VIRT_XENFV, VIRT_TYPE<br>○ SetCrash = SHUTDOWN_STATE_DESTROY<br>○ SetReboot = SHUTDOWN_STATE_RESTART<br>○ SetPoweroff = SHUTDOWN_STATE_DESTROY<br>○ SetTimeZone = TIMEZONE_UTC<br>○ AddGraphicsAdapter = GRAPHICS_TYPE, -1, "", "", "127.0.0.1"<br>○ SetDeviceEmulator = "/usr/lib64/xen/bin/qemu-system-i386"<br>○ AddInputDevice = "mouse", "ps2"<br>○ AddInputDevice = "keyboard", "ps2"<br>○ AddDisk = DISK_TYPE_FILE, DEVICE_TYPE_DISK, "/home/guest_images/"+vm_name+".img", "drive", 0, 0, 0, 0, "hda", "ide", true, false<br>○ AddDisk = DISK_TYPE_FILE, DEVICE_TYPE_CDROM, "/var/lib/xen/images/Ubuntu14-04.iso", "drive", 0, 0, 0, 0, "hdc", "ide", false, true<br>○ AddNetworkInterface = INTERFACE_TYPE_BRIDGE, GenerateInterfaceMacAddress = , "xenbr0"<br>○ AddBootDevice = DEVICE_TYPE_HD<br>○ AddBootDevice = DEVICE_TYPE_CDROM<br>○ SetHypervisorFeatures = true, true, true |
| **Expected Output:** | • The console window displays a vm template<br><br><domain type="xen"><br><name>BuntuD</name><br><memory>500000</memory><br><vcpu>1</vcpu> |

| | |
|---|---|
| | <os><br><type arch="x86_64" machine="xenfv">hvm</type><br><boot dev="hd" /><br><boot dev="cdrom" /><br></os><br><features><br><acpi /><br><apic /><br><pae /><br></features><br><clock offset="utc" /><br><on_poweroff>destroy</on_poweroff><br><on_reboot>restart</on_reboot><br><on_crash>destroy</on_crash><br><devices><br><emulator>/usr/lib64/xen/bin/qemu-system-i386</emulator><br><disk type="file" device="disk"><br><source file="/home/guest_images/BuntuD.img" /><br><backingStore /><br><target dev="hda" bus="ide" /><br><address type="drive" controller="0" bus="0" target="0" unit="0" /><br></disk><br><disk type="file" device="cdrom"><br><source file="/var/lib/xen/images/Ubuntu14-04.iso" /><br><target dev="hdc" bus="ide" /><br><readonly /><br><address type="drive" controller="0" bus="0" target="0" unit="0" /><br></disk><br><interface type="bridge"><br><mac address="00:16:a1:fa:0b:6e" /><br><source bridge="xenbr0" /><br></interface><br><input type="mouse" bus="ps2" /><br><input type="keyboard" bus="ps2" /><br><graphics type="vnc" port="-1" listen="127.0.0.1" /><br></devices><br></domain> |

**User Story #235 – Display Virtual Machine Information**

| **Test ID: I-235-1** | |
|---|---|
| **Purpose:** | Test the VMAX show vm info interface function |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated<br>• There is at least one virtual machine in the host list |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Info” |
| **Expected Output:** | • The virtual machine information is displayed in a pop-up window |

**User Story #236 – Create User Interface for configuration file editing**

| **Test ID: U-236-1** | |
|---|---|
| **Purpose:** | Verify that code creates a user interface for the user |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file which is located on local machine and golang repositories being loaded to the local machine |
| **Input:** | None |
| **Expected Output:** | User interface for the user to select options |

| Test ID: I-236-1 | |
|---|---|
| **Purpose:** | Verify that code creates a user interface for the user on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | None |
| **Expected Output:** | User interface for the user to select options, reading the file located on the Xen Server |

**User Story #237 – Use Interface to Edit Host IP Address**

| Test ID: U-237-1 | |
|---|---|
| **Purpose:** | Verify that code can edit the Host IP Address based on user input |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of selecting the option of Editing the Host IP Address |
| **Expected Output:** | Host IP Address is edited to the configuration file on local machine |

| Test ID: I-237-1 | |
|---|---|
| **Purpose:** | Verify that code can edit the Host IP Address based on user input on the Xen Server |

| | |
|---|---|
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User chooses the option to edit Host IP Address |
| **Expected Output:** | Based on user input the Host IP Address is changed on the configuration file on the Xen Server through the User Interface |

**User Story #238 – Use Interface to Edit Port Number**

| **Test ID: U-238-1** | |
|---|---|
| **Purpose:** | Verify that code can edit the Host Port Number based on user input |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of selecting the option of Editing the Host Port Number |
| **Expected Output:** | Host Port Number is edited to the configuration file |

| **Test ID: I-238-1** | |
|---|---|
| **Purpose:** | Verify that code can edit the Host Port Number based on user input on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |

| | |
|---|---|
| **Input:** | User chooses the option to edit Host Port Number |
| **Expected Output:** | Based on user input the Host Port Number is changed on the configuration file on the Xen Server through the User Interface |

**User Story #270 – Modify the Code for the .Net Format of the Web**

| **Test ID: U-270-1** | |
|---|---|
| **Purpose:** | Redesign the web interface to display virtual machines |
| **Preconditions:** | Finished an basic interface includes tables and displays virtual machines |
| **Input:** | Click the connect and enter the host IDs |
| **Expected Output:** | The VS information shows up, also all the data will be show up on the interface. The data can successfully comes from the current operation of the VS machines. Also the better interface view should be shown. |

| **Test ID: I-270-1** | |
|---|---|
| **Purpose:** | Redesign the web interface to display virtual machines |
| **Preconditions:** | Finished an basic interface includes tables and displays virtual machines |
| **Input:** | Click the connect and enter the host IDs |
| **Expected** | The VS information shows up, also all the data will be show up on |

| | |
|---|---|
| **Output:** | the interface. The data will be load from the current operation of the VS machines. Also the better interface view should be shown. |

**User Story #271 – Add Pause Functionality; Also Will Add Resume Functionality**

| **Test ID: U-271-1** | |
|---|---|
| **Purpose:** | Add pause functionality and add resume functionality, test the function through the interface, so that the users can control the five VM machines |
| **Preconditions:** | Finished adding the buttons for the two functions. |
| **Input:** | After select a host from the virtual machine host list, click the virtual machine name, then click each button-“pause”, “resume” |
| **Expected Output:** | Users can successfully pause and resume the virtual machines. |

| **Test ID: I-271-1** | |
|---|---|
| **Purpose:** | Add pause functionality and add resume functionality, test the function through the interface, so that the users can control the five VM machines |
| **Preconditions:** | Finished adding the buttons for the two functions. |
| **Input:** | After select a host from the virtual machine host list, click the virtual machine name, then click each button-“pause”, “resume” |

| | |
|---|---|
| **Expected Output:** | Users can successfully pause and resume the virtual machines. |

**User Story #272 – Add Stop Functionality**

| **Test ID: U-271-1** | |
|---|---|
| **Purpose:** | Add STOP functionality and add resume functionality, test the function through the interface, so that the users can control the five VM machines |
| **Preconditions:** | Finished adding the buttons for stop. |
| **Input:** | After select a host from the virtual machine host list, click the virtual machine name, then click button”stop” |
| **Expected Output:** | Users can successfully stop the virtual machines. |

| **Test ID: U-271-1** | |
|---|---|
| **Purpose:** | Add STOP functionality and add resume functionality, test the function through the interface, so that the users can control the five VM machines |
| **Preconditions:** | Finished adding the buttons for stop. |
| **Input:** | After select a host from the virtual machine host list, click the virtual machine name, then click button”stop” |

| | |
|---|---|
| **Expected Output:** | Users can successfully stop the virtual machines. |

**User Story #273 – Update database after each function call; Also display the current vm status**

| **Test ID: U-273** | |
|---|---|
| **Purpose:** | All the functions should be work; also the information of the database should be successfully updated. |
| **Preconditions:** | The functions has been already added, the interface has been already adjusted. |
| **Input:** | Choosing one host ID;<br>Pick up one Virtual Machine name;<br>Click on each function;<br>Check the database situation. |
| **Expected Output:** | 1.All the functions should be work;<br>2.The information of the database should be successfully updated. |

**User Story #275 – List Virtual Machine Disk Configurations**

| **Test ID: I-275-1** | |
|---|---|
| **Purpose:** | Test the VMAX start virtual machine function |
| **Preconditions:** | • VMAX application has started successfully and host list has been automatically updated |

| | |
|---|---|
| | • There is a vm that is shutdown |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select “Start” |
| **Expected Output:** | • The virtual machine icon turns to a blue color<br>• The PowerState of the virtual machine is “Running” |

**User Story #312 – Create Virtual Machine From Existing Disk Image**

| **Test ID: U-312-1** | |
|---|---|
| **Purpose:** | Test VirtualMachineBuilder classes XML serialization of options to create existing virtual machine |
| **Preconditions:** | • Host object created successfully with IP and port of the TCC9 test server; used to initialize XenConnect |
| **Input:** | Vm Name: “Senior Buntu”<br>Memory: 2000000<br>Vcpus: 1<br>• Set Hard Drive Existing function parameters:<br>  • Backing Store : true<br>  • Disk Copy : true<br>  • Path : true<br>• Use LibvirtMessageWrapper.BuildMessage function to create message and marshall |
| **Expected Output:** | <LibvirtServerMessage><br><Command>CREATE VM</Command> |

| | |
|---|---|
| | <Requestor>Senior Project Xen Master Test</Requestor><br><RequestorId>172.16.10.76</RequestorId><br><ResponseCode>0</ResponseCode><br><VirtualMachineBuilder><br><ClockOffset>utc</ClockOffset><br><UseExistingDisk>true</UseExistingDisk><br><DomType>TRANSIENT</DomType><br><VmName>SeniorBuntu</VmName><br><Iso>/var/lib/xen/images/Ubuntu14-04.iso</Iso><br><Memory>200000</Memory><br><HDMemory>20000</HDMemory><br><Vcpu>1</Vcpu><br><HypervisorFeatures><br><Acpi>true</Acpi><br><Apic>true</Apic><br><Pae>true</Pae><br></HypervisorFeatures><br><Power><br><PowerOff>destroy</PowerOff><br><Reboot>restart</Reboot><br><Crash>destroy</Crash><br></Power><br><Devices><br><Emulator>/usr/lib64/xen/bin/qemu-system-i386</Emulator><br><Disks><br><Disk><br><BackingStore>false</BackingStore><br><ReadOnly>true</ReadOnly><br><SourceFile>/var/lib/xen/images/Ubuntu14-04.iso</SourceFile><br><AddressType>drive</AddressType><br><AddressController>0</AddressController><br><AddressBus>0</AddressBus><br><AddressTarget>0</AddressTarget><br><AddressUnit>0</AddressUnit><br><Type>file</Type><br><Device>cdrom</Device><br><TargetDevice>hdc</TargetDevice><br><TargetBus>ide</TargetBus><br><Role>secondary</Role><br></Disk><br><Disk><br><BackingStore>true</BackingStore><br><ReadOnly>false</ReadOnly> |

| | |
|---|---|
| | <SourceFile>/home/guest_images/BuntuMaster.img</SourceFile><br><AddressType>drive</AddressType><br><AddressController>0</AddressController><br><AddressBus>0</AddressBus><br><AddressTarget>0</AddressTarget><br><AddressUnit>0</AddressUnit><br><Type>file</Type><br><Device>disk</Device><br><TargetDevice>hda</TargetDevice><br><TargetBus>ide</TargetBus><br><Role>primary</Role><br></Disk><br></Disks><br></Devices><br></VirtualMachineBuilder><br><VMList /><br></LibvirtServerMessage> |

| **Test ID: I-312-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's create virtual machine from existing disk image capability |
| **Preconditions:** | • VMAX host list has been automatically updated<br>• A virtual disk image that is not in use by other virtual machine exists in the hosts \home\guest_images\ folder |
| **Input:** | 1. Click the "Create VM" button<br>2. Select the host from the dropdown menu<br>3. Check the "Use Existing Disk" box – the OS images dropdown will list the available virtual disks on the machine<br>4. Select the virtual disk that will be used to create the machine<br>5. Enter the memory, name, and VCPU count for the machine<br>6. Click the "Create VM" button in the New Virtual Machine |

| | |
|---|---|
| | Wizard window |
| **Expected Output:** | • A pop-up window showing the virtual machine creation status will be displayed<br>• The virtual machine will show up in the host virtual machine list |

**User Story #313 – Transfer Virtual Machine Images**

| **Test ID: U-313-1** | |
|---|---|
| **Purpose:** | Test XenConnect vm file transfer function |
| **Preconditions:** | • Host object created successfully with IP and port of the TCC9 test server; used to initialize XenConnect |
| **Input:** | File : the path to the file ex. “ C:\Users\DODTech\Desktop\Sample.txt”<br>Remote Directory: \home\guest_images\ |
| **Expected Output:** | • Return object XenError<br>  • Type: Success<br>  • Message: File transfer completed successfully |

| **Test ID: I-313-1** | |
|---|---|
| **Purpose:** | Test the VMAX application’s file transfer capability |
| **Preconditions:** | • VMAX host list has been automatically updated |

| | |
|---|---|
| | • There is at least one vm that is either “Running” or “Paused” |
| **Input:** | 1. Select the “File Manager” tab from the VMAX interface after selecting a host<br>2. Click the “Browse” button to open up a file picker window<br>3. Select a file to be transferred and click OK<br>4. Click the “Send” button to transfer the file |
| **Expected Output:** | • A pop-up window showing the file transfer status |

**User Story #314 – Clone Virtual Machine**

| **Test ID: U-314-1** | |
|---|---|
| **Purpose:** | Test XenConnect vm clone virtual machine capability |
| **Preconditions:** | • Host object created successfully with IP and port of the TCC9 test server<br>• The parent virtual machine is shutdown |
| **Input:** | VM UUID: “4cb12bd0-b1c6-a6bb-d1c4-46246585d351”<br>Clone Name: “CloneBuntu” |
| **Expected Output:** | • Return object XenError<br>  • Type: Success<br>  • Message: The virtual machine cloned successfully<br>• After some time the virtual machine will be cloned and running |

| **Test ID: I-314-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's clone virtual machine capability |
| **Preconditions:** | • VMAX host list has been automatically updated<br>• There is at least one virtual machine that is in the shutdown state |
| **Input:** | 1. Select a host from the host list (ex. TCC 9)<br>2. Right click on a shutdown virtual machine to bring up the context menu<br>3. Click the clone virtual machine option<br>4. A pop-up message will display showing the status of the clone request<br>5. After some time the virtual machine will be cloned and running |
| **Expected Output:** | • A pop-up window showing the clone request status |

**User Story #315 – Control Remote Virtual Machine**

| **Test ID: U-315-1** | |
|---|---|
| **Purpose:** | Test XenConnect vm VNC port retrieval function (getVmVNCPort) |
| **Preconditions:** | • Host object created successfully with IP and port of the TCC9 test server<br>• Virtual machine is online |
| **Input:** | VM UUID: "4cb12bd0-b1c6-a6bb-d1c4-46246585d351" |
| **Expected** | The VNC port number for the virtual machine will be returned |

| **Output:** | |
|---|---|

| **Test ID: U-315-2** | |
|---|---|
| **Purpose:** | Test port forwarding and SSH tunnel creation from VMAX to the host |
| **Preconditions:** | • Host allows SSH incoming and outgoing connections |
| **Input:** | Host IP: “172.16.10.115”<br>Host Username: “root”<br>Host Password: <omitted><br>• Execute VMViewer CreateSecureTunnel function |
| **Expected Output:** | There will be no messages or output if the tunnel is created correctly |

| **Test ID: U-315-3** | |
|---|---|
| **Purpose:** | Test VNC viewer remote frame buffer send/retrieve and capability to control virtual machine keyboard and mouse |
| **Preconditions:** | • Secure tunnel connection created to the machine and local machine port forwarded |
| **Input:** | Host Object with IP and Port information<br>Monitor: “0”<br>ViewOnly: false<br>Scaled : true<br>• Execute viewer ConnectSSH function |

| | |
|---|---|
| **Expected Output:** | The Virtual machine desktop will be displayed and the machine keyboard and mouse input will be controlled |

| **Test ID: I-315-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's remote viewing capability |
| **Preconditions:** | • VMAX host list has been automatically updated<br>• There is at least one vm that is either "Running" or "Paused" |
| **Input:** | 1. Select a host from the host list<br>2. Right click the virtual machine name<br>3. Select "Remote Control" |
| **Expected Output:** | • A pop-up window showing the virtual machine desktop is displayed |

**User Story #317 – Display what is typed by the user into the interface**

| **Test ID: U-317-1** | |
|---|---|
| **Purpose:** | Verify that code can display the user input |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of editing the configuration file |
| **Expected Output:** | User is able to see what is being typed into the interface |

| **Test ID: I-317-1** | |
|---|---|
| **Purpose:** | Verify that code can display what is being typed onto the user interface which will edit the configuration file on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User chooses the option to edit and inputs the value |
| **Expected Output:** | User is able to see what they are typing to edit the configuration file on the Xen Server |

**User Story #318 – Validate input values for both IP address and Port number**

| **Test ID: U-318-1** | |
|---|---|
| **Purpose:** | Verify that code can validate the user input |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of editing the configuration file |
| **Expected Output:** | User Interface verifies that input is in the correct format |

| **Test ID: I-318-1** |
|---|

| | |
|---|---|
| **Purpose:** | Verify that code can verify the format of what is being typed onto the user interface which will edit the configuration file on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User inputs the values of editing the configuration file |
| **Expected Output:** | User interface is able to verify the values are in the correct format for editing the configuration file |

**User Story #319– Enter all the values to UI and update through one submission**

| **Test ID: U-319-1** | |
|---|---|
| **Purpose:** | User can edit the file using one submission |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of editing the configuration file |
| **Expected Output:** | User Interface edits the configuration file all in one submission |

| **Test ID: I-319-1** | |
|---|---|
| **Purpose:** | Verify that the code can edit the configuration file in the Xen server using one submission |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration |

| | |
|---|---|
| | file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User inputs the values of editing the configuration file |
| **Expected Output:** | User interface is able to edit the configuration file in the Xen server when the user inputs both values with one submission |

**User Story #320– Clean up user interface**

| **Test ID: U-320-1** | |
|---|---|
| **Purpose:** | Clean up the user interface for Xen virtual machine |
| **Preconditions:** | “Pause”, “Stop” functions already added to interface; Also, part of the codes for the VM already updated. |
| **Input:** | Click “Connect”, then click “Home page” |
| **Expected Output:** | The log in page and the home page has connection; Also, user can see some information on the homepage. The duplicate part for log in page already delete. |

| **Test ID: I-320-1** | |
|---|---|
| **Purpose:** | Clean up the user interface for Xen virtual machine |
| **Preconditions:** | “Pause”, “Stop” functions already added to interface; Also, part of the codes for the VM already updated. |

| | |
|---|---|
| **Input:** | Click “Connect”, then click “Home page” |
| **Expected Output:** | User can see the connection between the log in page and the home page; The contact information is shown on the homepage; User will not see any duplicate part or label on the log in page |

**User Story #321– Implement “Create” Virtual Machine**

| **Test ID: U-321-1** | |
|---|---|
| **Purpose:** | Implement “Create” VM machine on the interface |
| **Preconditions:** | Finished the basic cleaned up interface |
| **Input:** | Click “Create VM” then click “add” to add the new information for the new virtual machine |
| **Expected Output:** | The new virtual machine should be created; it will be added under the list of all the virtual machines; All the information related about the new VM should be updated to the VM list |

| **Test ID: I-321-1** | |
|---|---|
| **Purpose:** | Implement “Create” VM machine on the interface |
| **Preconditions:** | Cleaned up the interface |
| **Input:** | Click “Create VM” then add the new information for the new virtual machine |

| | |
|---|---|
| **Expected Output:** | The new virtual machine should be created; it will be added under the list of all the virtual machines |

**User Story #322– Display status of virtual machines**

| **Test ID: U-322-1** | |
|---|---|
| **Purpose:** | Display status of virtual machines |
| **Preconditions:** | The basic library has been already added to the web page. |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button |
| **Expected Output:** | The system virtual machine status has been show up on the webpage.It includes all the information about the VM name, the Host ID, the host name, the VMUUID, the operating system, CPUTime, VMState, Memory, MaxMemory, VirtualCPUS, statusCode,comments and all the functions about the virtual machines include the created new virtual machine.<br>Also, it shows up the functions below:<br>"START"<br>"STOP"<br>"Force Shut Down"<br>"PAUSE" |

| **Test ID: U-322-1** |
|---|

| | |
|---|---|
| **Purpose:** | Display status of virtual machines |
| **Preconditions:** | The basic library has been already added to the web page. |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button |
| **Expected Output:** | The system virtual machine status has been show up on the webpage.It includes all the information about the VM name, the Host ID, the host name, the VMUUID, the operating system, CPUTime, VMState, Memory, MaxMemory, VirtualCPUS, statusCode,comments and all the functions about the virtual machines include the created new virtual machine.<br>Also, it shows up the functions below:<br>"START"<br>"STOP"<br>"Force Shut Down"<br>"PAUSE" |

**User Story #359– Delete Virtual Machine**

| **Test ID: U-359-1** | |
|---|---|
| **Purpose:** | Test XenMaster delete virtual machine capability |
| **Preconditions:** | • The virtual machine is in the shutdown state |
| **Input:** | VM UUID: (Example) "4cb12bd0-b1c6-a6bb-d1c4-46246585d351" |
| **Expected Output:** | • Return object XenError<br>  • Type: Success |

| | |
|---|---|
| | • Message: The virtual machine has been deleted successfully<br>• After some time the virtual machine will be deleted and running |

| **Test ID: I-359-1** | |
|---|---|
| **Purpose:** | Test the VMAX application's delete virtual machine capability |
| **Preconditions:** | • VMAX host list has been automatically updated<br>• There is at least one virtual machine that is in the shutdown state |
| **Input:** | 1. Select a host from the host list (ex. TCC 9)<br>2. Right click on a shutdown virtual machine to bring up the context menu<br>3. Click the delete virtual machine option<br>4. A pop-up confirmation message will be displayed<br>5. Click "Yes"<br>6. The virtual machine will be removed from the host machine list<br>7. A pop-up message will display the delete machine status |
| **Expected Output:** | • A pop-up window displaying the text "The virtual machine [name] has been successfully deleted" |

**User Story #360– Display Performance Parameters**

| **Test ID: U-360-1** | |
|---|---|
| **Purpose:** | Test cpu information retrieval function |

| | |
|---|---|
| **Preconditions:** | • At least one host is available in the network |
| **Input:** | 1. Execute the Test_ListVmStatistics function<br><br>**Host Ip**: 172.16.10.76<br>**Host Name**: TCC9<br>**Port**: 29171 |
| **Expected Output:** | • List of VcpuInfo objects |

| **Test ID: I-360-1** | |
|---|---|
| **Purpose:** | Test VMAX CPU performance parameters |
| **Preconditions:** | • At least one host is available |
| **Input:** | 1. Select a host from the hosts lists<br>2. Click the “Performance” tab |
| **Expected Output:** | • A bar graph showing the performance of virtual machines and the host cpu will be displayed |

**User Story #369– Clean up the interface**

| **Test ID: U-369-1** | |
|---|---|
| **Purpose:** | Clean up the interface to show help options on bottom and place |

| | |
|---|---|
| | dialog boxes next to options |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User selects options to see dialog boxes |
| **Expected Output:** | Cleaner interface with help options on bottom and dialog boxes next to options |

| **Test ID: I-369-1** | |
|---|---|
| **Purpose:** | Clean up interface that is displayed on Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User selects options to see dialog boxes |
| **Expected Output:** | Cleaner interface with help options on bottom and dialog boxes next to options |

**User Story #370– Update all parameters with single selection**

| **Test ID: U-370-1** | |
|---|---|
| **Purpose:** | Verify that code can update all parameters through one submission |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |

| | |
|---|---|
| **Input:** | User input of editing the configuration file |
| **Expected Output:** | User Interface verifies that input is in the correct format and then edits the file |

| **Test ID: I-370-1** | |
|---|---|
| **Purpose:** | Verify that code can update all parameters through one submission onto the user interface which will edit the configuration file on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User inputs the values of editing the configuration file |
| **Expected Output:** | User interface is able to verify the values are in the correct format for editing the configuration file and able to submit using one submission |

**User Story #371– View file should be a separate button/selection**

| **Test ID: U-371-1** | |
|---|---|
| **Purpose:** | The option to view the entire file should be a separate button/option |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User inputs Ctrl+Space to change the view and select View Configuration file |

| | |
|---|---|
| **Expected Output:** | User Interface displays configuration file |

| **Test ID: I-371-1** | |
|---|---|
| **Purpose:** | Verify that code can display the configuration file in different view on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User inputs Ctrl+Space to go to the different view and select View Configuration File |
| **Expected Output:** | User interface is able to display configuration file through the different vie |

**User Story #372– Add a signal handler to catch the quit option**

| **Test ID: U-372-1** | |
|---|---|
| **Purpose:** | Verify that code can throw signal handler for quit |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input of selecting Yes/No |
| **Expected Output:** | User Interface either closes if Yes is selected, User interface closes quit dialog box if No is selected |

| **Test ID: I-372-1** | |
|---|---|
| **Purpose:** | Verify that code can throw signal handler for quit on Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User inputs Yes or No on quit dialog box |
| **Expected Output:** | User interface quits if Yes is selected showing terminal of Xen Server, If No is selected, interface is still loaded |

**User Story #372– Add more information on the home page**

| **Test ID: U-377-1** | |
|---|---|
| **Purpose:** | Add more information on the home page |
| **Preconditions:** | Created an original home page for adding the introduction information |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button |
| **Expected Output:** | The system general picture is added to the about page; Also, the project contact information is added to the about page. User can easily check all the related information. |

| Test ID: I-377-1 | |
|---|---|
| **Purpose:** | Add more information on the home page |
| **Preconditions:** | Created an original home page for adding the introduction information |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button |
| **Expected Output:** | The new contact information and the general picture information should be successfully show up on the page. |

**User Story #378– Design an interface for listing the existing VHD**

| Test ID: U-378-1 | |
|---|---|
| **Purpose:** | Design an interface for listing the existing VHD |
| **Preconditions:** | The vm list shown on the interface; The log in interface has builded. |
| **Input:** | Click "Connect", then click "Create";<br>User add cpu information, memory to the VHD |
| **Expected Output:** | The information can be added to the VHD;<br>The VHD interface design can be shown on the VM list interface |

| **Test ID: I-378-1** | |
|---|---|
| **Purpose:** | Design an interface for listing the existing VHD |
| **Preconditions:** | The vm list shown on the interface; The log in interface has builded. |
| **Input:** | Click “Connect”, then click “Create”;<br>User add CPU information, memory to the VHD |
| **Expected Output:** | The information can be added to the VHD;<br>The VHD interface design can be shown on the VM list interface |

**User Story #417 – Create Universally Unique Identifier's**

| **Test ID: U-417-1** | |
|---|---|
| **Purpose:** | Test Golang server ability to generate uuid’s |
| **Preconditions:** | • The host server “LOCAL SERVER ID” field is set to “UNKNOWN” |
| **Input:** | 1. Run main executable ./main from console terminal |
| **Expected Output:** | • The console displays “Generating Introspector Id: 9B75904032”<br>• The server settings file LOCAL SERVER ID is set to the new identifier |

**User Story #418 – Display host disk usage**

| **Test ID: U-418-1** | |
|---|---|
| **Purpose:** | Test host disk usage retrieval function |
| **Preconditions:** | • Host is available and online |
| **Input:** | 1. Host IP = 172.16.10.77 Port = 29171<br>2. List<DiskStatistic> object reference |
| **Expected Output:** | • List of disk statistics for the remote host |

| **Test ID: I-418-1** | |
|---|---|
| **Purpose:** | Test VMAX disk usage statistics display |
| **Preconditions:** | • At least one host is available and online |
| **Input:** | 1. Select a host from the Xen Hosts list<br>2. Click the “Disk Usage” tab |
| **Expected Output:** | • Graphs showing the disk usage will be displayed |

**User Story #420– Design the interface for multiple inputs and update**

| **Test ID: U-420-1** | |
|---|---|
| **Purpose:** | Verify that the code can update all parameters with multiple input boxes through one submssion |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User input in multiple boxes |
| **Expected Output:** | With one submission file is edited |

| **Test ID: I-420-1** | |
|---|---|
| **Purpose:** | Verify that code can update all parameters through one submission onto the user interface which will edit the configuration file on the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User input in multiple boxes |
| **Expected Output:** | With one submission the file is edited on the Xen Server |

**User Story #421– Design the interface for user display**

| **Test ID: U-421-1** |
|---|

| | |
|---|---|
| **Purpose:** | Design the interface for user display so two input boxes are displayed to edit the file |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file and User interface being loaded onto the terminal |
| **Input:** | User selects options to see dialog boxes |
| **Expected Output:** | Cleaner interface with two dialog boxes for input |

| **Test ID: I-421-1** | |
|---|---|
| **Purpose:** | Design the interface for user display so two input boxes are displayed to edit the file in the Xen Server |
| **Preconditions:** | Golang code being pointed to the proper location of the configuration file on the Xen Server and golang repositories being loaded onto the Server. Program running on the console of the Xen Server. |
| **Input:** | User selects options to see dialog boxes |
| **Expected Output:** | Cleaner interface with two dialog boxes for input for editing file on Xen Server |

**User Story #422– Delete Virtual Machine**

| **Test ID: U-422-1** | |
|---|---|
| **Purpose:** | Delete virtual Machine from the VM list |

| | |
|---|---|
| **Preconditions:** | 1. The virtual machine list has already added to the interface.<br>2. The functions “ Create” “Stop” “Pause” “ Shut Down” “ Force Shut Down” has already been added on the virtual machine list.<br>3. The connection between home page and the log in window has already been built. |
| **Input:** | User type in their host ID to the log in page<br>User click on “ Connect” button<br>User pull out the current virtual machine list<br>User click” Delete” button |
| **Expected Output:** | The virtual machine which user picked up can be deleted from the list |

| **Test ID: I-422-1** | |
|---|---|
| **Purpose:** | Delete virtual Machine from the VM list for the database |
| **Preconditions:** | 1. The virtual machine list has already added to the interface.<br>2. The functions “ Create” “Stop” “Pause” “ Shut Down” “ Force Shut Down” has already been added on the virtual machine list.<br>3. The connection between home page and the log in window has already been built. |
| **Input:** | User type in their host ID to the log in page<br>User click on “ Connect” button<br>User pull out the current virtual machine list<br>User click” Delete” button |
| **Expected Output:** | The virtual machine which user choose can be deleted from the list |

**User Story #424– Display hypervisors from the network using tree view control**

| **Test ID: U-424-1** | |
|---|---|
| **Purpose:** | Display hypervisors from the network using tree view control |
| **Preconditions:** | 1. The virtual machine list is already built.<br>2. The connection between home page and the log in window has already been built. |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button<br>User check the tree view control list for the hypervisors |
| **Expected Output:** | The tree view should be shown up above the vm list. |

| **Test ID: I-424-1** | |
|---|---|
| **Purpose:** | Display hypervisors from the network using tree view control |
| **Preconditions:** | 1. The virtual machine list is already built.<br>2. The connection between home page and the log in window has already been built. |
| **Input:** | User type in their host ID to the log in page<br>User click on " Connect" button<br>User check the tree view control list for the hypervisors |

| | |
|---|---|
| **Expected Output:** | The tree view should be shown up above the virtual machine list. |

# GLOSSARY

**Hypervisor -** A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine (Wikipedia).

**Type 1 Hypervisor** - These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. For this reason, they are sometimes called bare metal hypervisors (Wikipedia).

**Type 2 Hypervisor** - These hypervisors run on a conventional operating system just as other computer programs do. A guest operating system runs as a process on the host. Type-2 hypervisors abstract guest operating systems from the host operating system (Wikipedia).

## Appendix A – UML Diagrams
### *Static UML Diagrams*

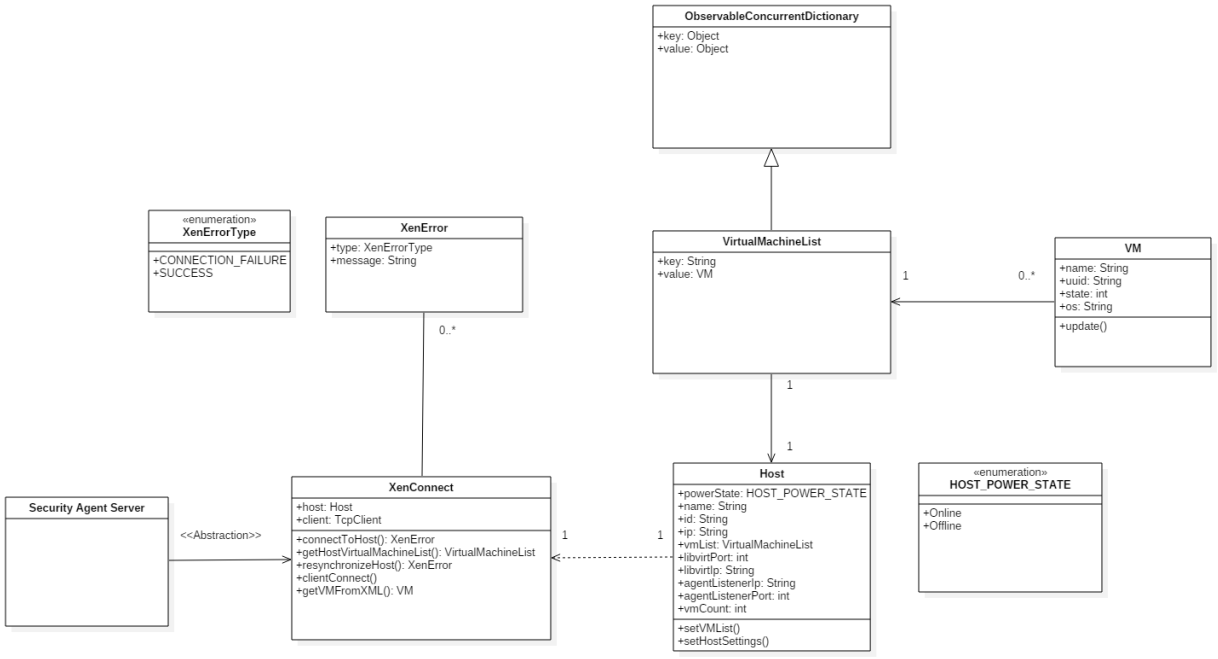



Figure 1 – Access Virtual Machines

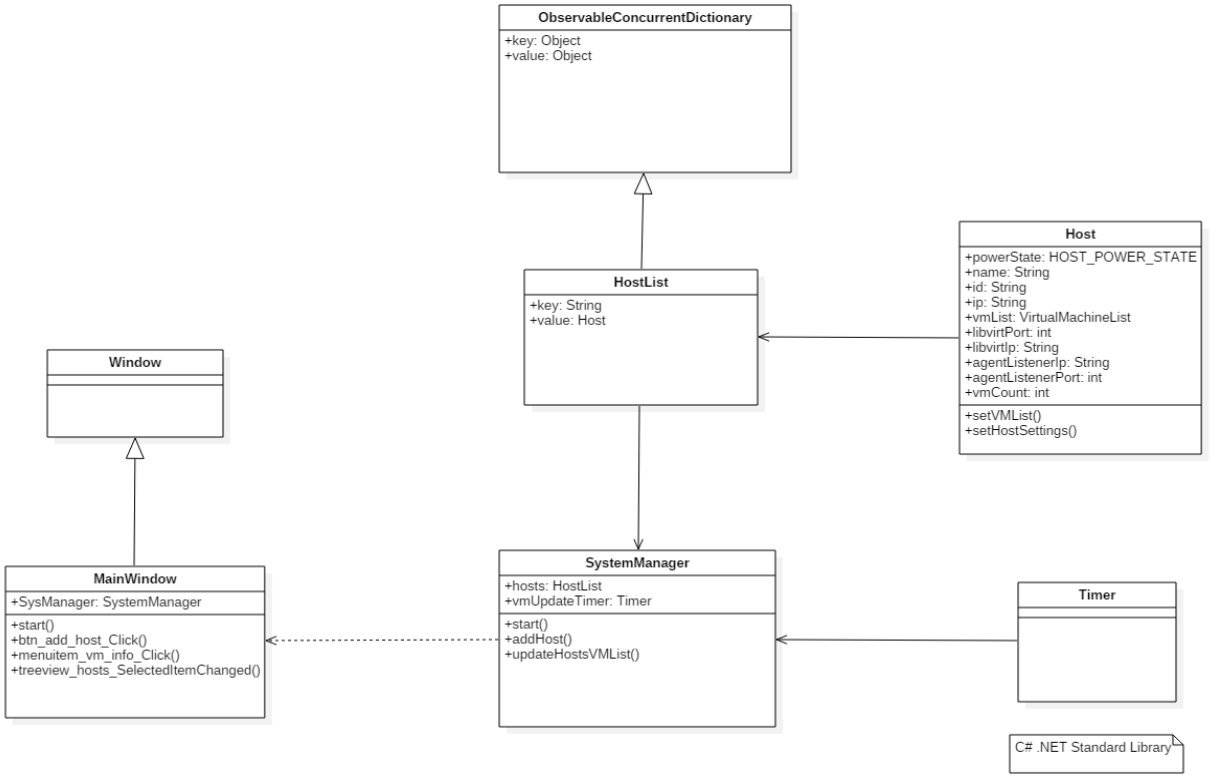



Figure 2 – View Host Virtual Machines

Model:: Get FileClass Diagram

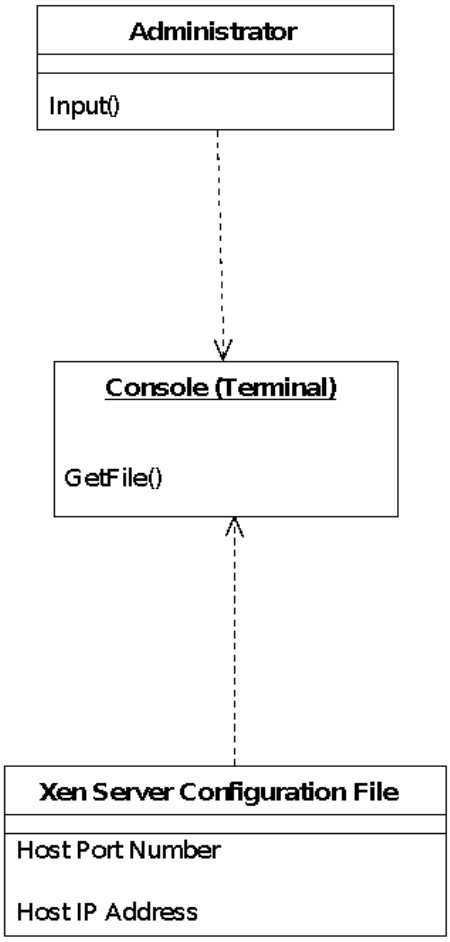


Figure 3 – View Settings of configuration File

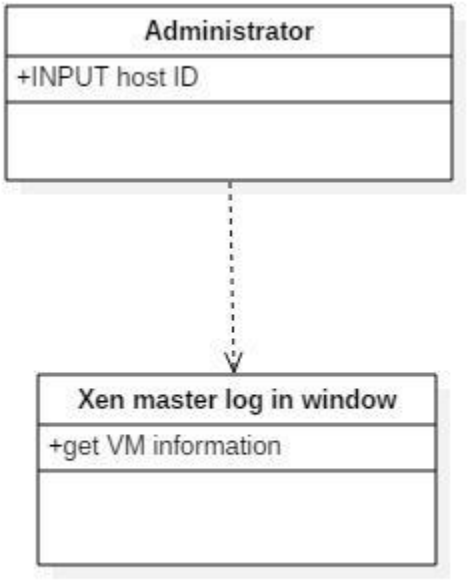



Figure 4 - Building web based dashboard for VMs

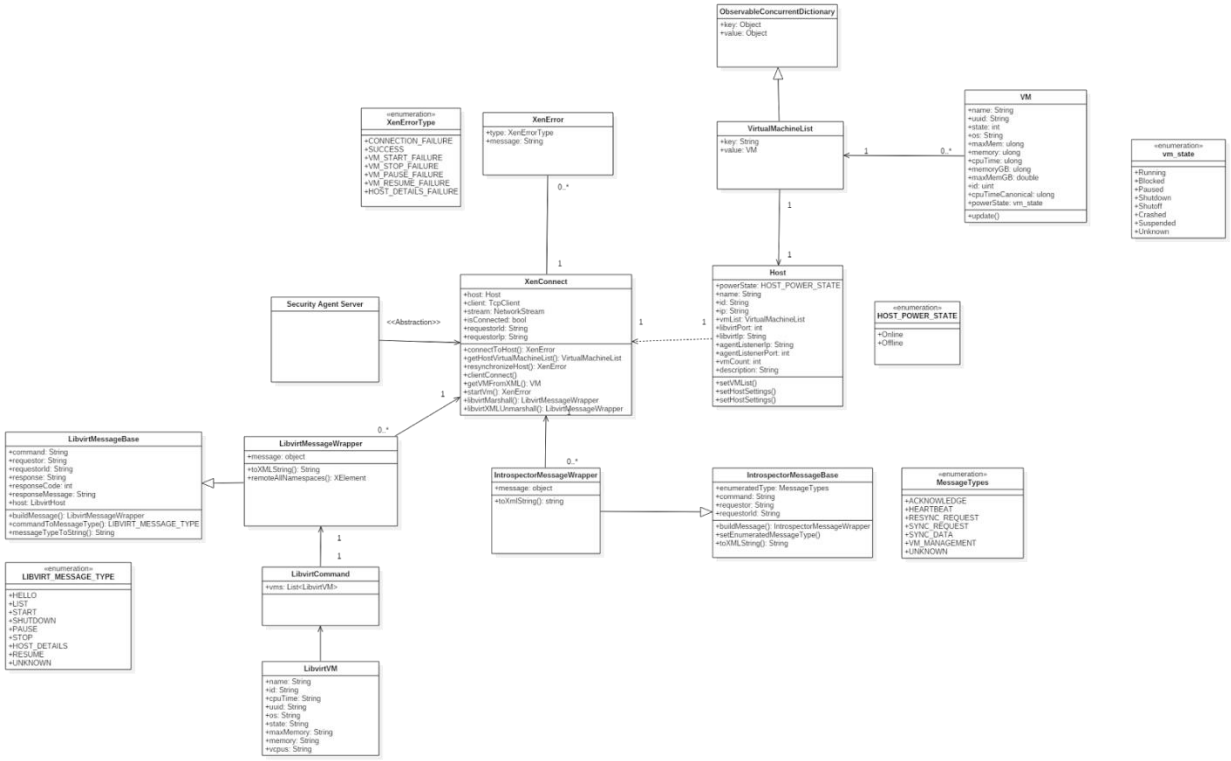



Figure 5 – Start a Virtual Machine

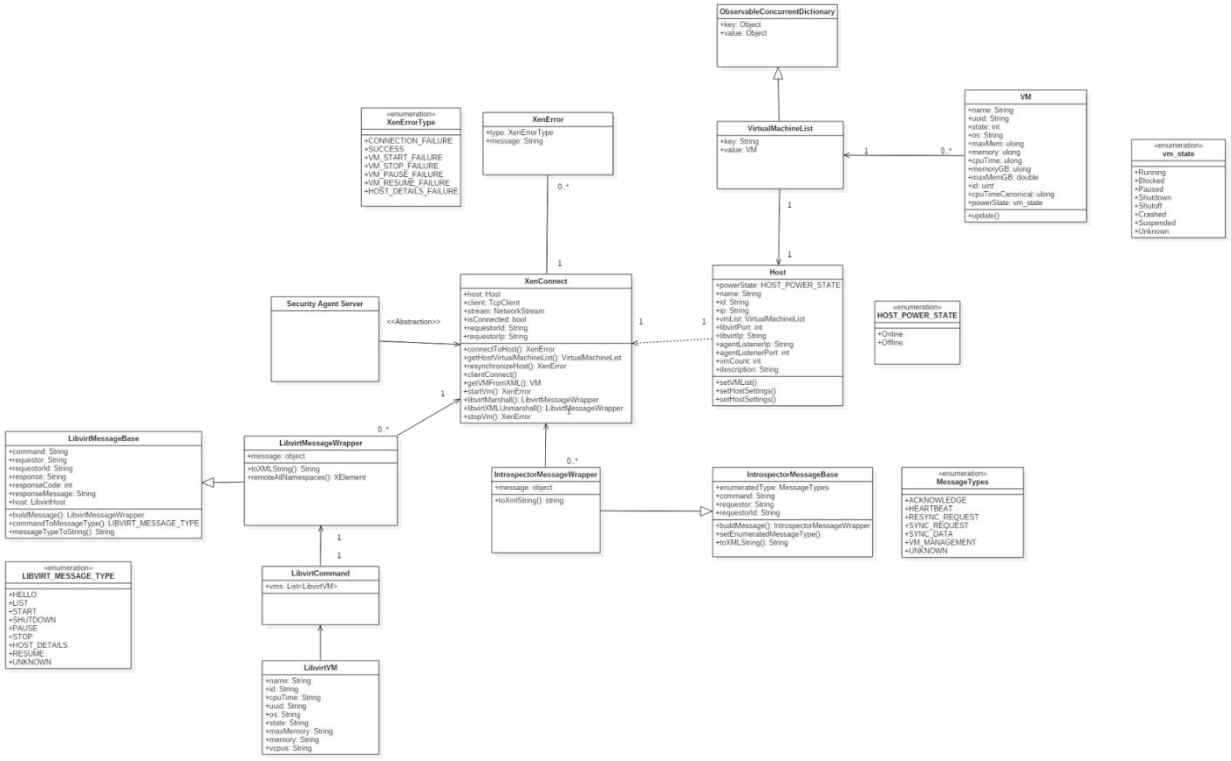



Figure 6 – Stop Virtual Machine

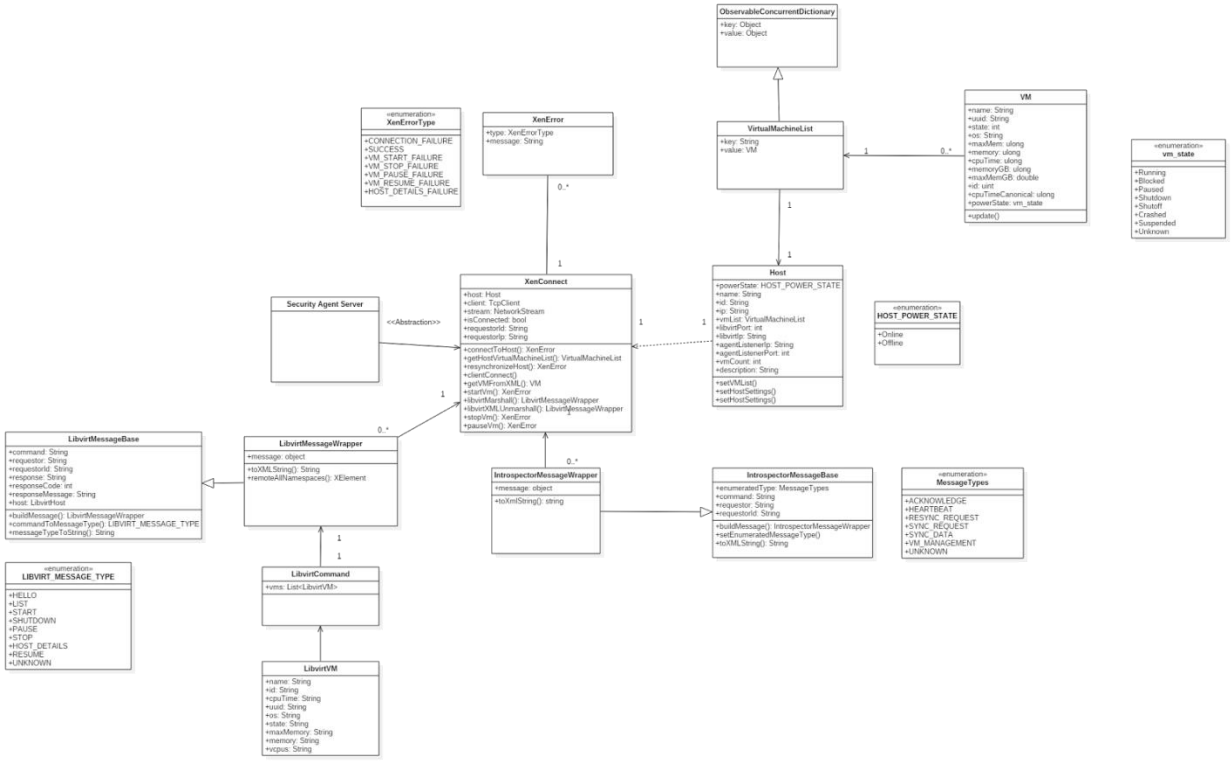



Figure 7 – Pause Virtual Machine

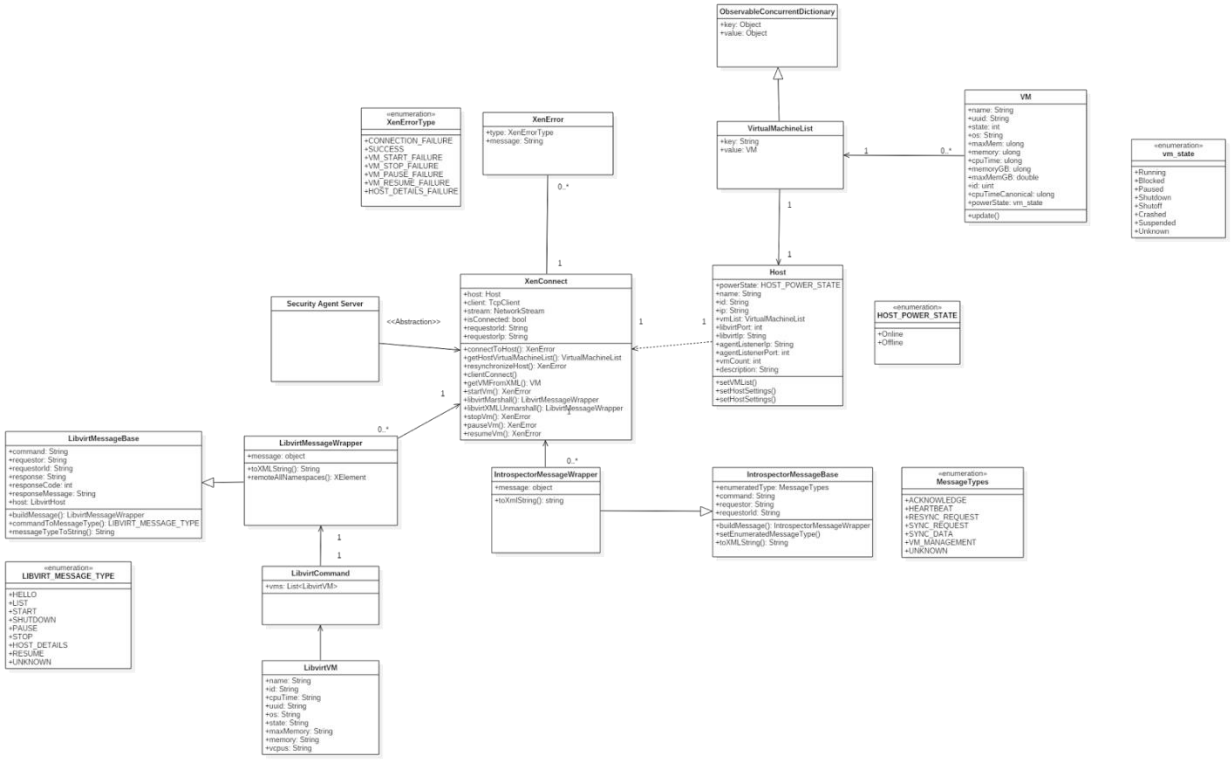



Figure 8 – Resume Virtual Machine

Model:: Edit FileClass Diagram

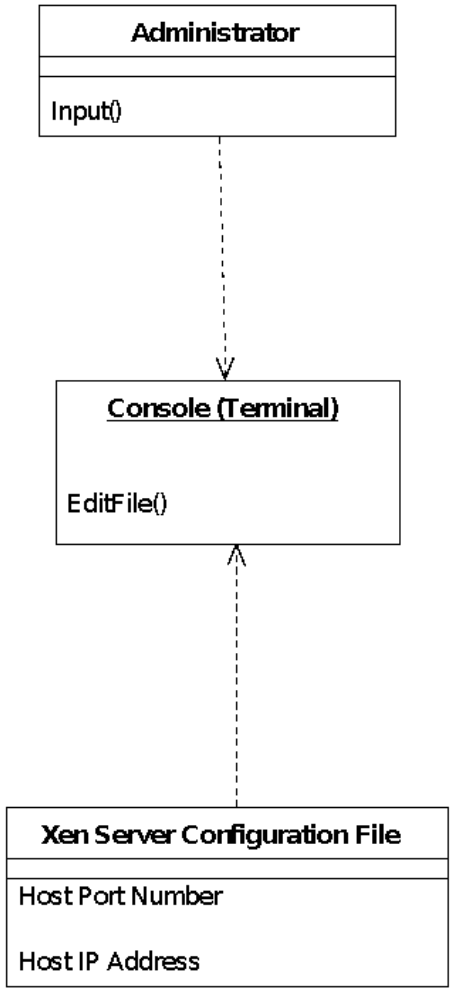


Figure 9 – Edit the settings of configuration file

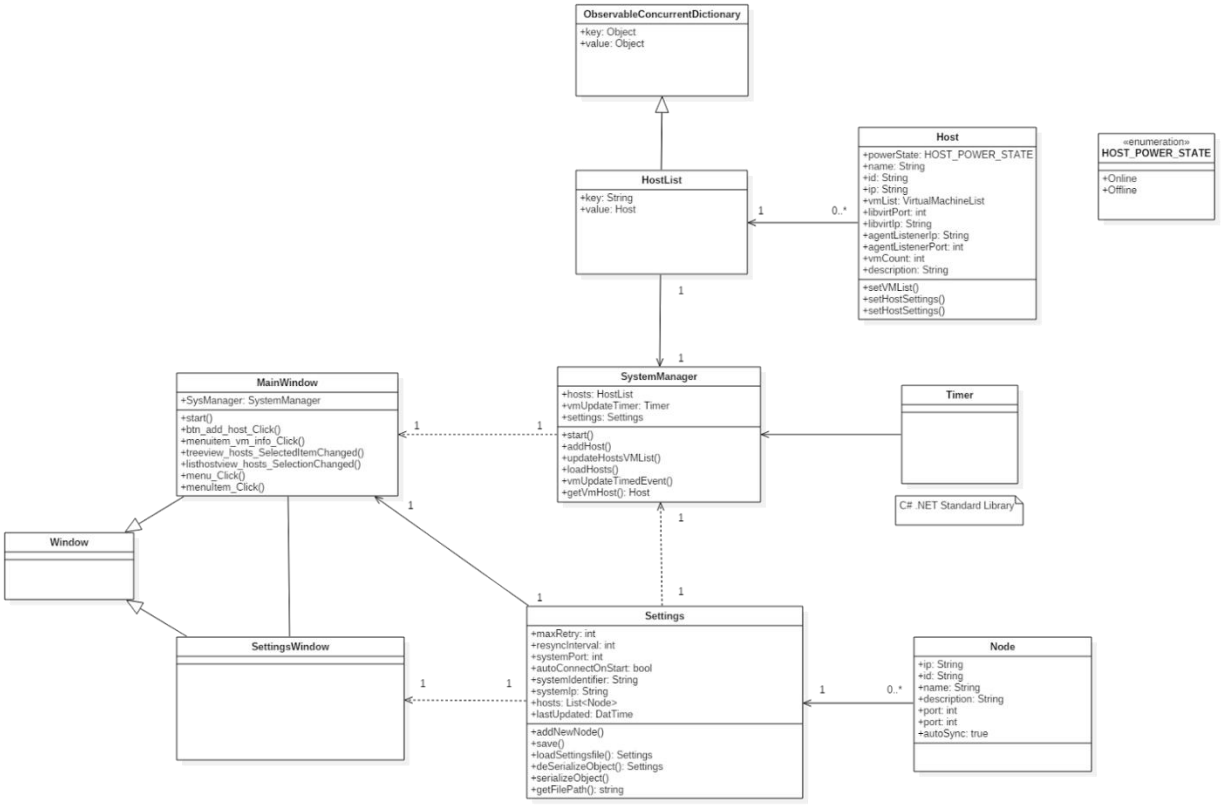



Figure 10 – Load and View VMAX Settings

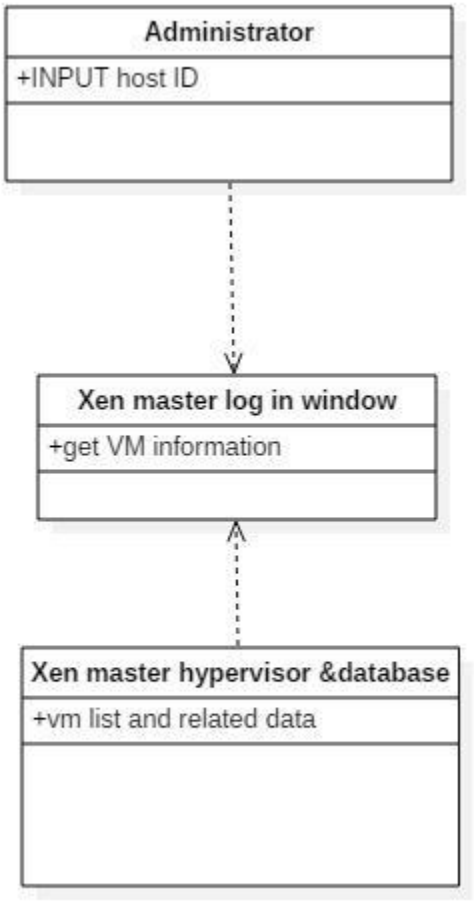



Figure 11 - Display the virtual machines on the web interface

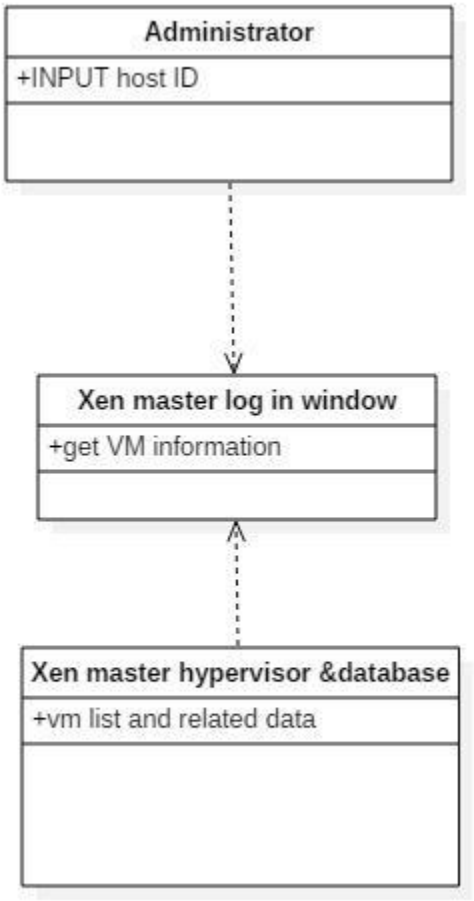



Figure 12 - Update the VMAX database

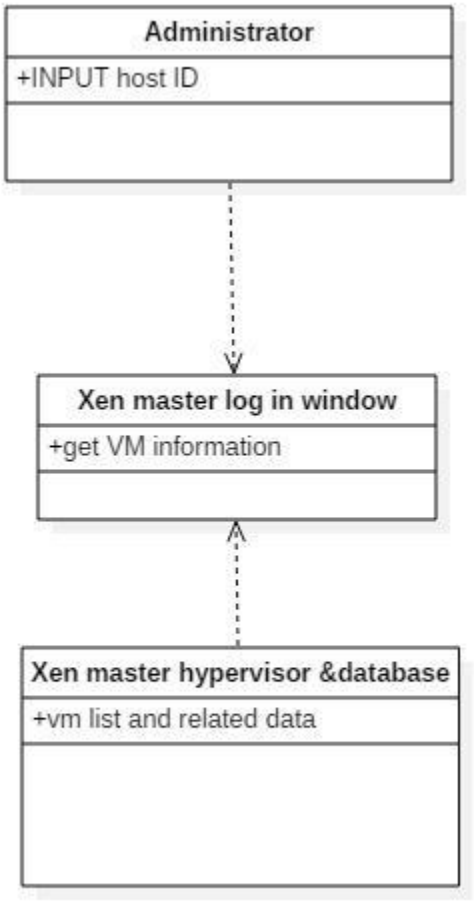



Figure 13 - Get the list of virtual machines for web dashboard

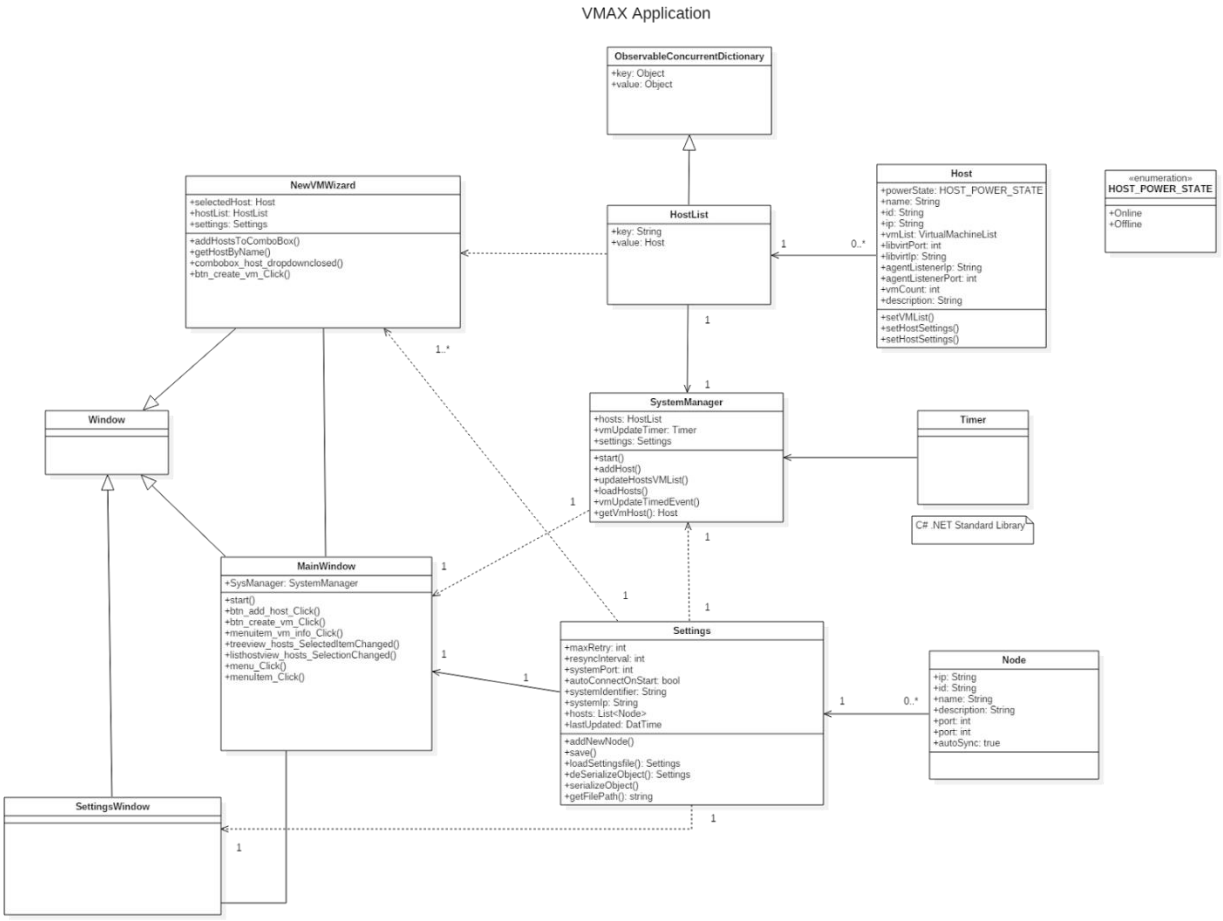



Figure 14 - Create Virtual Machine

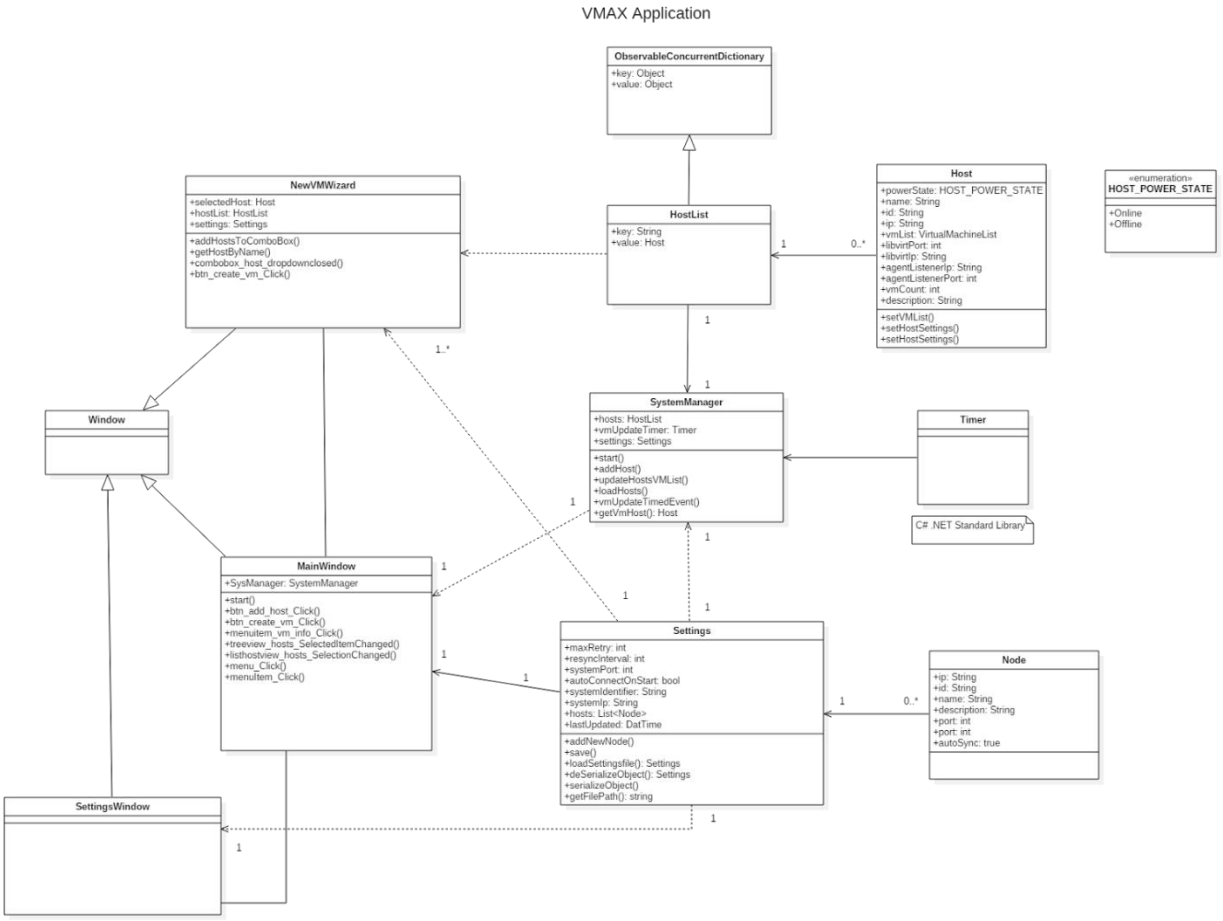



Figure 15 - Force Shutdown Virtual Machine

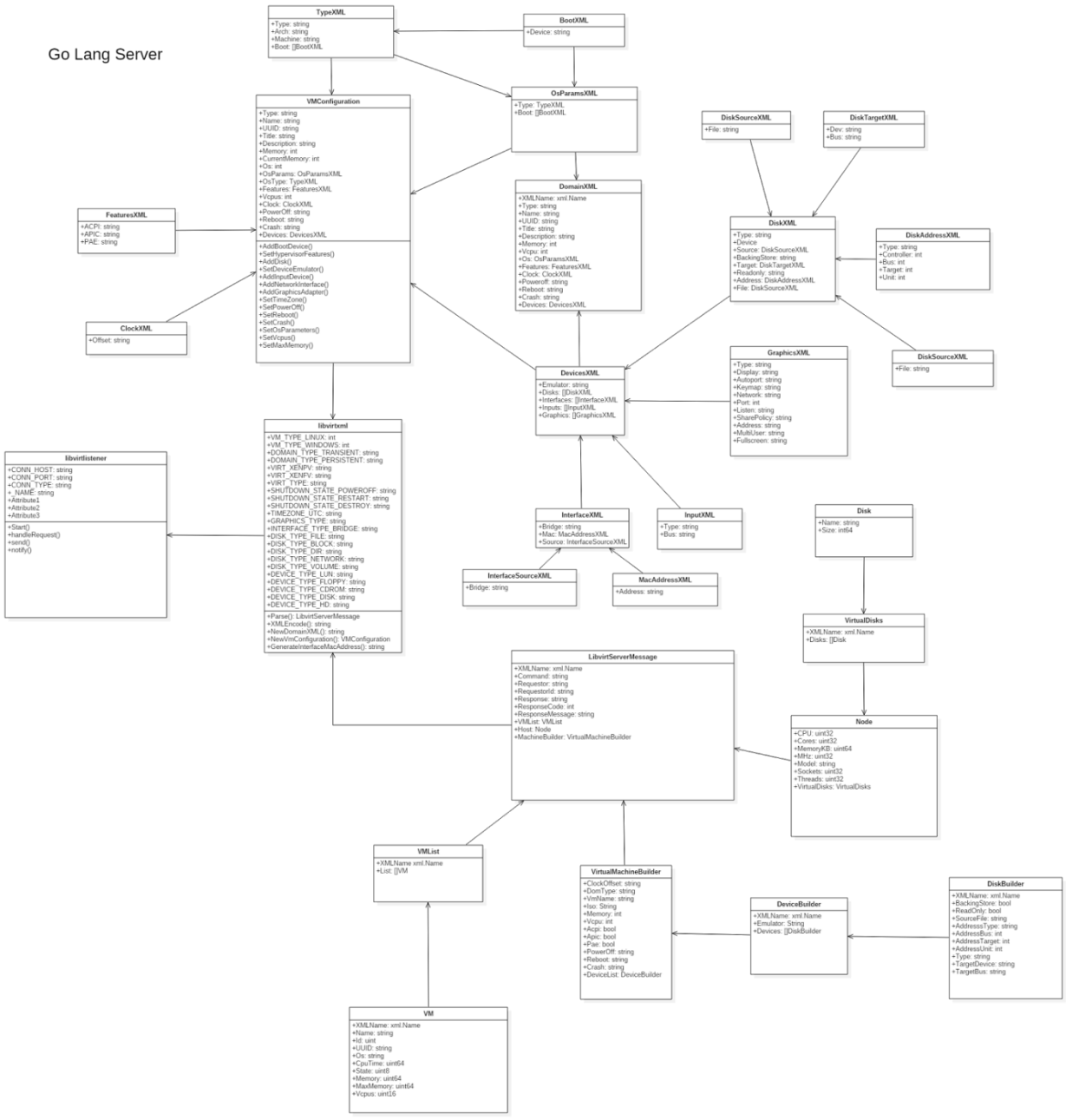

Go Lang Server
TypeXML
+Type: string
+Arch: string
+Machine: string
+Boot: []BootXML
BootXML
+Device: string
OsParamsXML
+Type: TypeXML
+Boot: []BootXML
VMConfiguration
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+CurrentMemory: int
+Os: int
+OsParams: OsParamsXML
+OsType: TypeXML
+Features: FeaturesXML
+Vcpus: int
+Clock: ClockXML
+PowerOff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
+AddBootDevice()
+SetHypervisorFeatures()
+AddDisk()
+SetDeviceEmulator()
+AddInputDevice()
+AddNetworkInterface()
+AddGraphicsAdapter()
+SetTimeZone()
+SetPowerOff()
+SetReboot()
+SetCrash()
+SetOsParameters()
+SetVcpus()
+SetMaxMemory()
FeaturesXML
+ACPI: string
+APIC: string
+PAE: string
ClockXML
+Offset: string
DomainXML
+XMLName: xml.Name
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+Vcpu: int
+Os: OsParamsXML
+Features: FeaturesXML
+Clock: ClockXML
+Poweroff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
DiskSourceXML
+File: string
DiskTargetXML
+Dev: string
+Bus: string
DiskXML
+Type: string
+Device
+Source: DiskSourceXML
+BackingStore: string
+Target: DiskTargetXML
+Readonly: string
+Address: DiskAddressXML
+File: DiskSourceXML
DiskAddressXML
+Type: string
+Controller: int
+Bus: int
+Target: int
+Unit: int
GraphicsXML
+Type: string
+Display: string
+Autoport: string
+Keymap: string
+Network: string
+Port: int
+Listen: string
+SharePolicy: string
+Address: string
+MultiUser: string
+Fullscreen: string
DiskSourceXML
+File: string
DevicesXML
+Emulator: string
+Disks: []DiskXML
+Interfaces: []InterfaceXML
+Inputs: []InputXML
+Graphics: []GraphicsXML
libvirtxml
+VM_TYPE_LINUX: int
+VM_TYPE_WINDOWS: int
+DOMAIN_TYPE_TRANSIENT: string
+DOMAIN_TYPE_PERSISTENT: string
+VIRT_XENPV: string
+VIRT_XENFV: string
+VIRT_TYPE: string
+SHUTDOWN_STATE_POWEROFF: string
+SHUTDOWN_STATE_RESTART: string
+SHUTDOWN_STATE_DESTROY: string
+TIMEZONE_UTC: string
+GRAPHICS_TYPE: string
+INTERFACE_TYPE_BRIDGE: string
+DISK_TYPE_FILE: string
+DISK_TYPE_BLOCK: string
+DISK_TYPE_DIR: string
+DISK_TYPE_NETWORK: string
+DISK_TYPE_VOLUME: string
+DEVICE_TYPE_LUN: string
+DEVICE_TYPE_FLOPPY: string
+DEVICE_TYPE_CDROM: string
+DEVICE_TYPE_DISK: string
+DEVICE_TYPE_HD: string
+Parse(): LibvirtServerMessage
+XMLEncode(): string
+NewDomainXML(): string
+NewVmConfiguration(): VMConfiguration
+GenerateInterfaceMacAddress(): string
libvirtlistener
+CONN_HOST: string
+CONN_PORT: string
+CONN_TYPE: string
+_NAME: string
+Attribute1
+Attribute2
+Attribute3
+Start()
+handleRequest()
+send()
+notify()
InterfaceXML
+Bridge: string
+Mac: MacAddressXML
+Source: InterfaceSourceXML
InputXML
+Type: string
+Bus: string
InterfaceSourceXML
+Bridge: string
MacAddressXML
+Address: string
Disk
+Name: string
+Size: int64
VirtualDisks
+XMLName: xml.Name
+Disks: []Disk
LibvirtServerMessage
+XMLName: xml.Name
+Command: string
+Requestor: string
+RequestorId: string
+Response: string
+ResponseCode: int
+ResponseMessage: string
+VMList: VMList
+Host: Node
+MachineBuilder: VirtualMachineBuilder
Node
+CPU: uint32
+Cores: uint32
+MemoryKB: uint64
+MHz: uint32
+Model: string
+Sockets: uint32
+Threads: uint32
+VirtualDisks: VirtualDisks
VMList
+XMLName xml.Name
+List: []VM
VirtualMachineBuilder
+ClockOffset: string
+DomType: string
+VmName: string
+Iso: String
+Memory: int
+Vcpu: int
+Acpi: bool
+Apic: bool
+Pae: bool
+PowerOff: string
+Reboot: string
+Crash: string
+DeviceList: DeviceBuilder
DeviceBuilder
+XMLName: xml.Name
+Emulator: String
+Devices: []DiskBuilder
DiskBuilder
+XMLName: xml.Name
+BackingStore: bool
+ReadOnly: bool
+SourceFile: string
+AddressType: string
+AddressBus: int
+AddressTarget: int
+AddressUnit: int
+Type: string
+TargetDevice: string
+TargetBus: string
VM
+XMLName: xml.Name
+Name: string
+Id: uint
+UUID: string
+Os: string
+CpuTime: uint64
+State: uint8
+Memory: uint64
+MaxMemory: uint64
+Vcpus: uint16

Figure 16 - Create Virtual Machine Templates

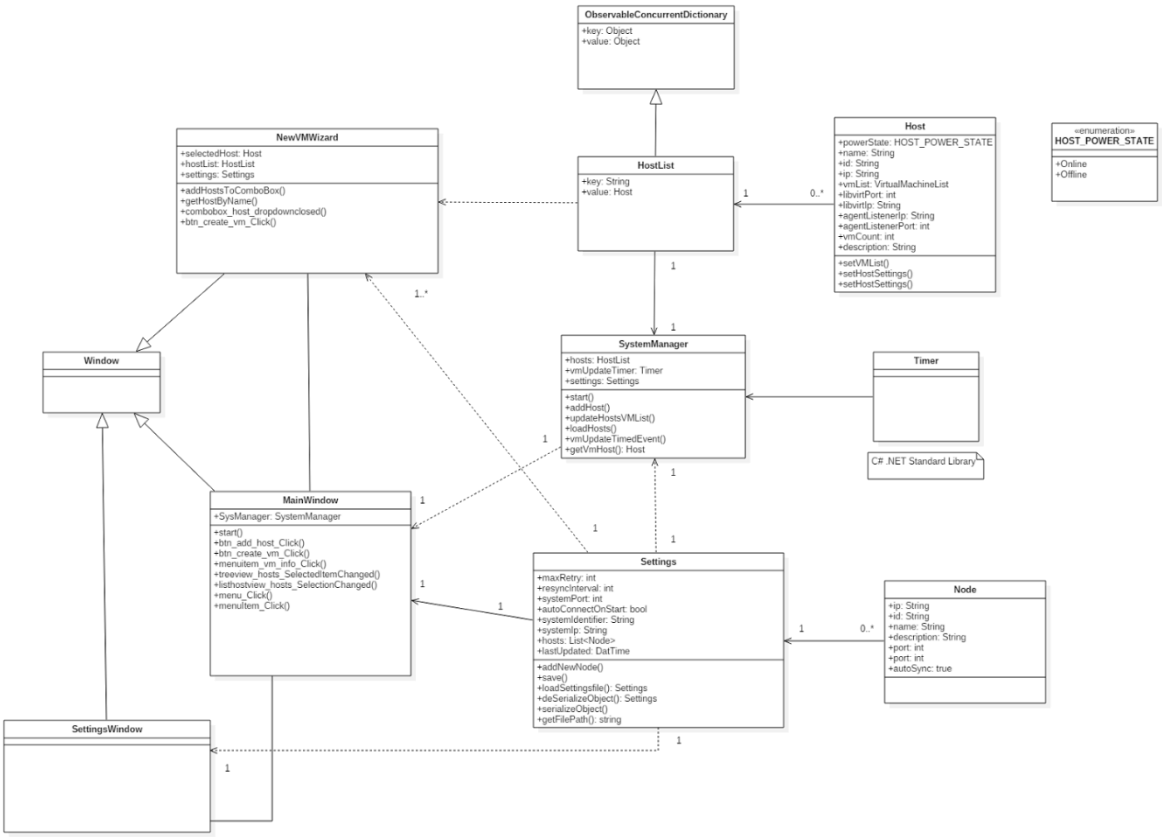



Figure 17 - Display Virtual Machine Information

Model::GoConsole Class Diagram

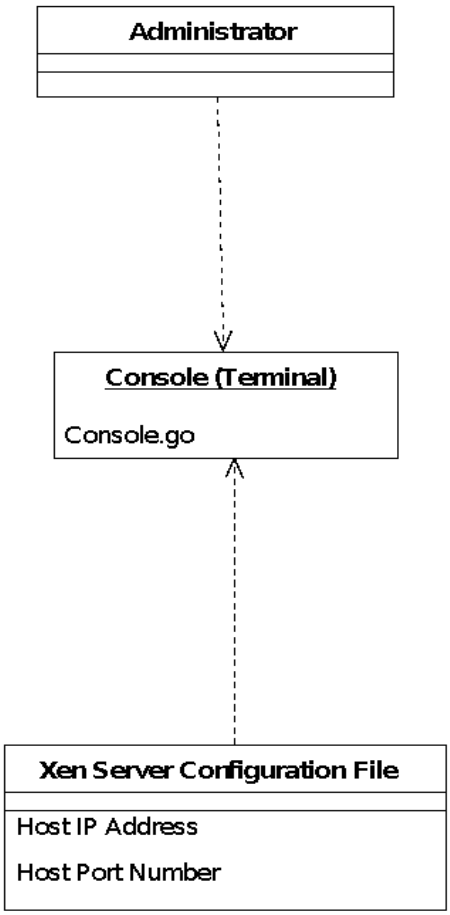


Figure 18 - Create a User Interface for the configuration file editing

Model:: Edit IP Address Class Diagram

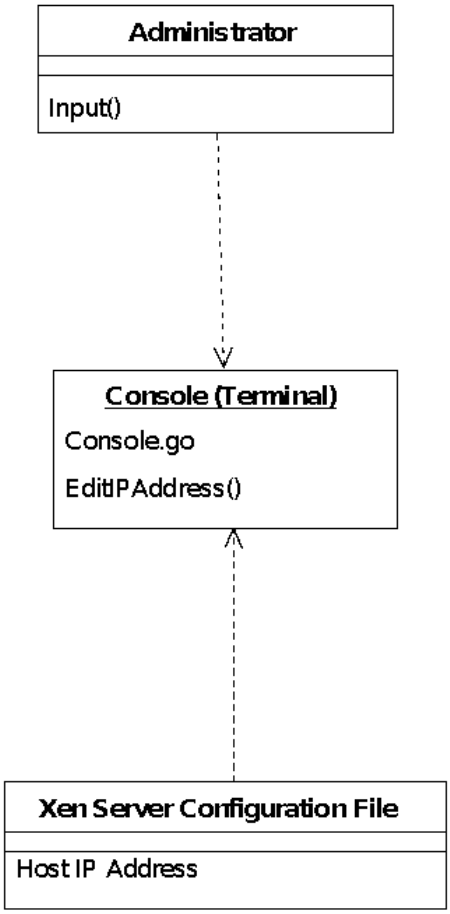


Figure 19 - Use interface to edit host ip address

Model:: Edit Port Number Class Diagram

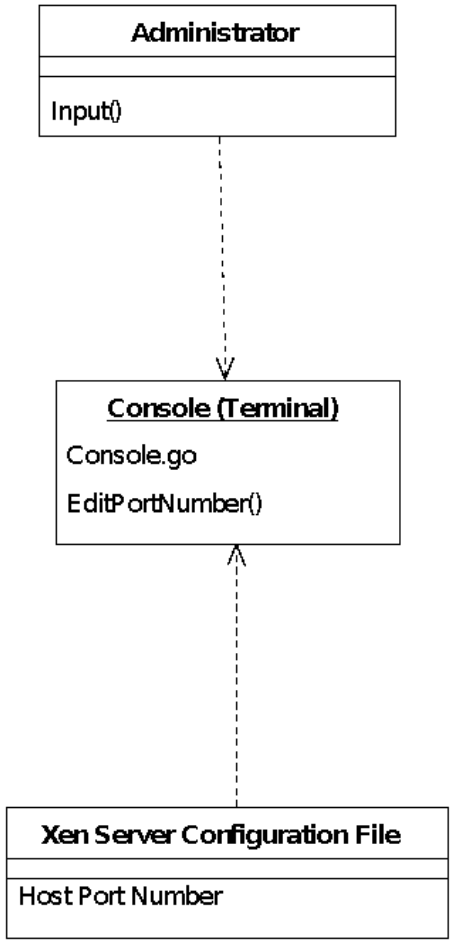


Figure 20 - Use interface to edit port number

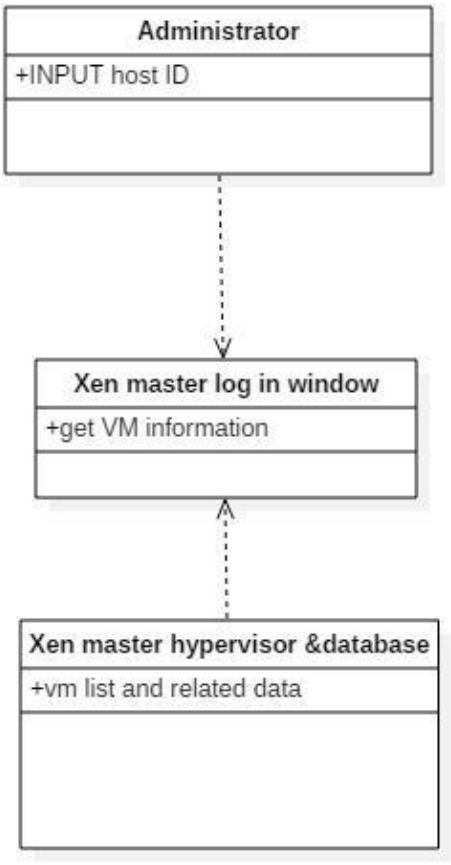



Figure 21 - Modify the code for the .net format of the web

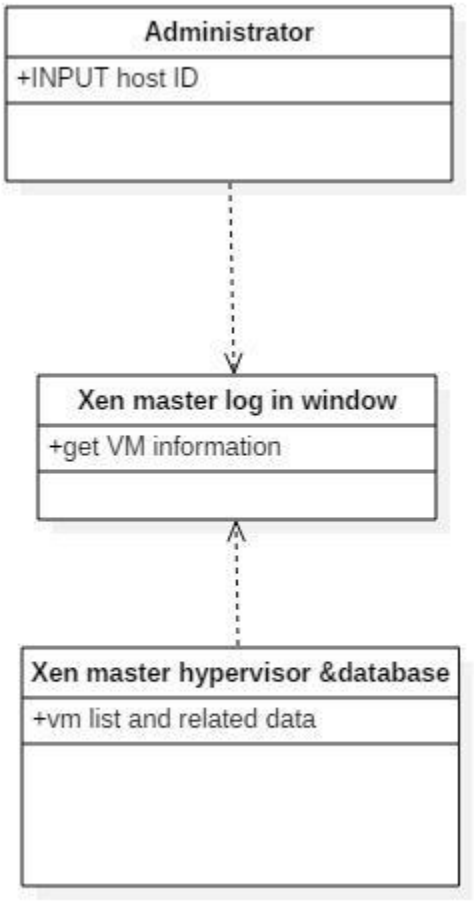



Figure 22 - Add stop functionality

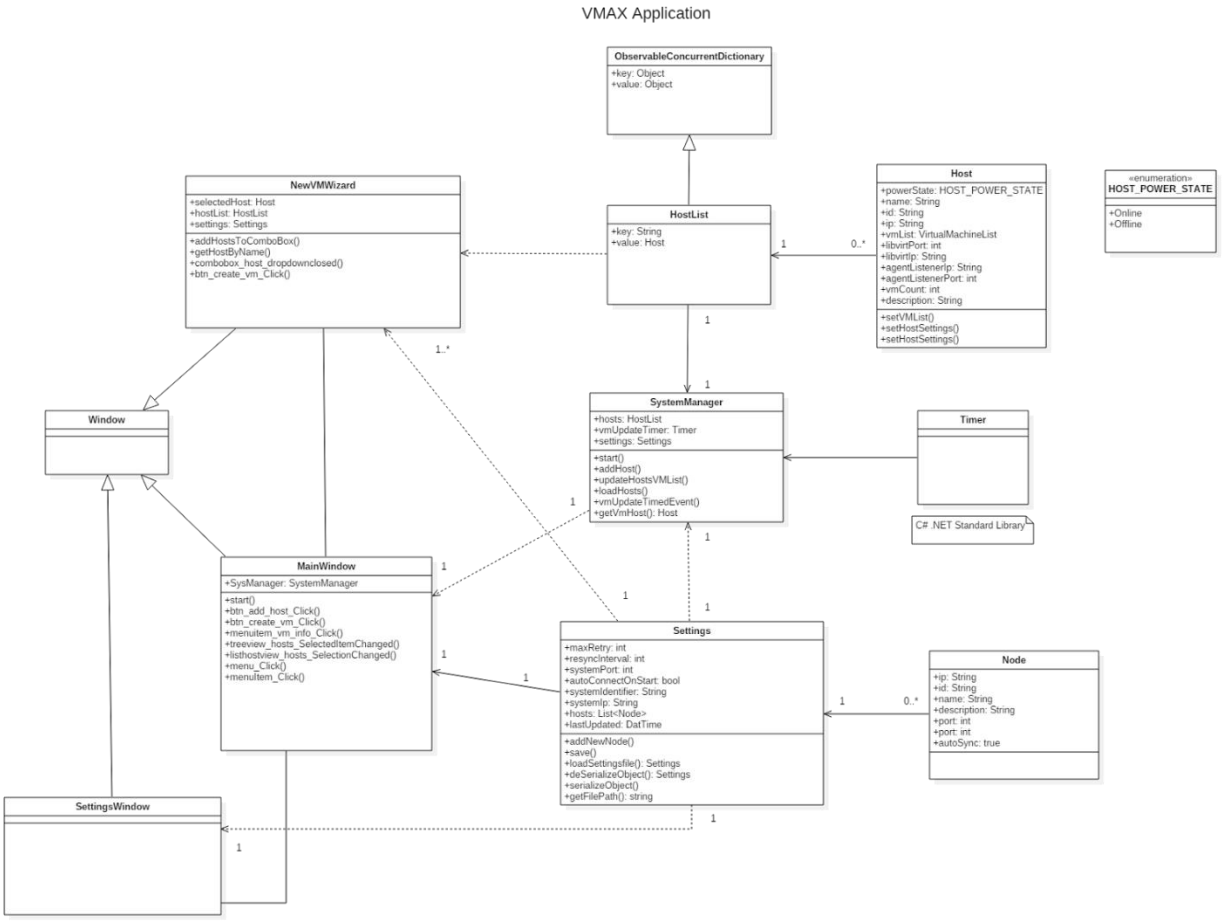



Figure 23 - List Virtual Machine Disk Configurations

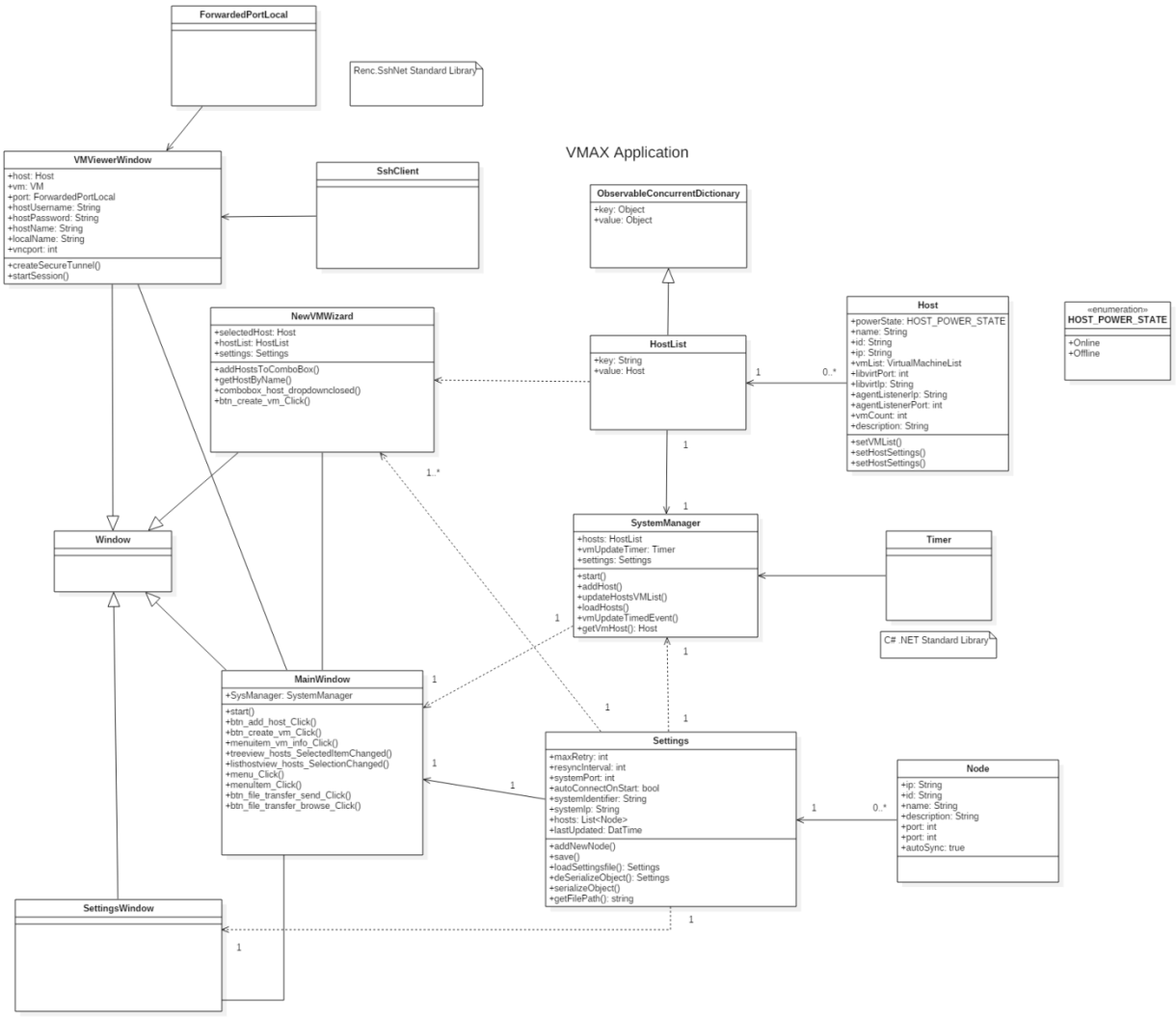



Figure 24 - Create Virtual Machine From Existing Disk Image

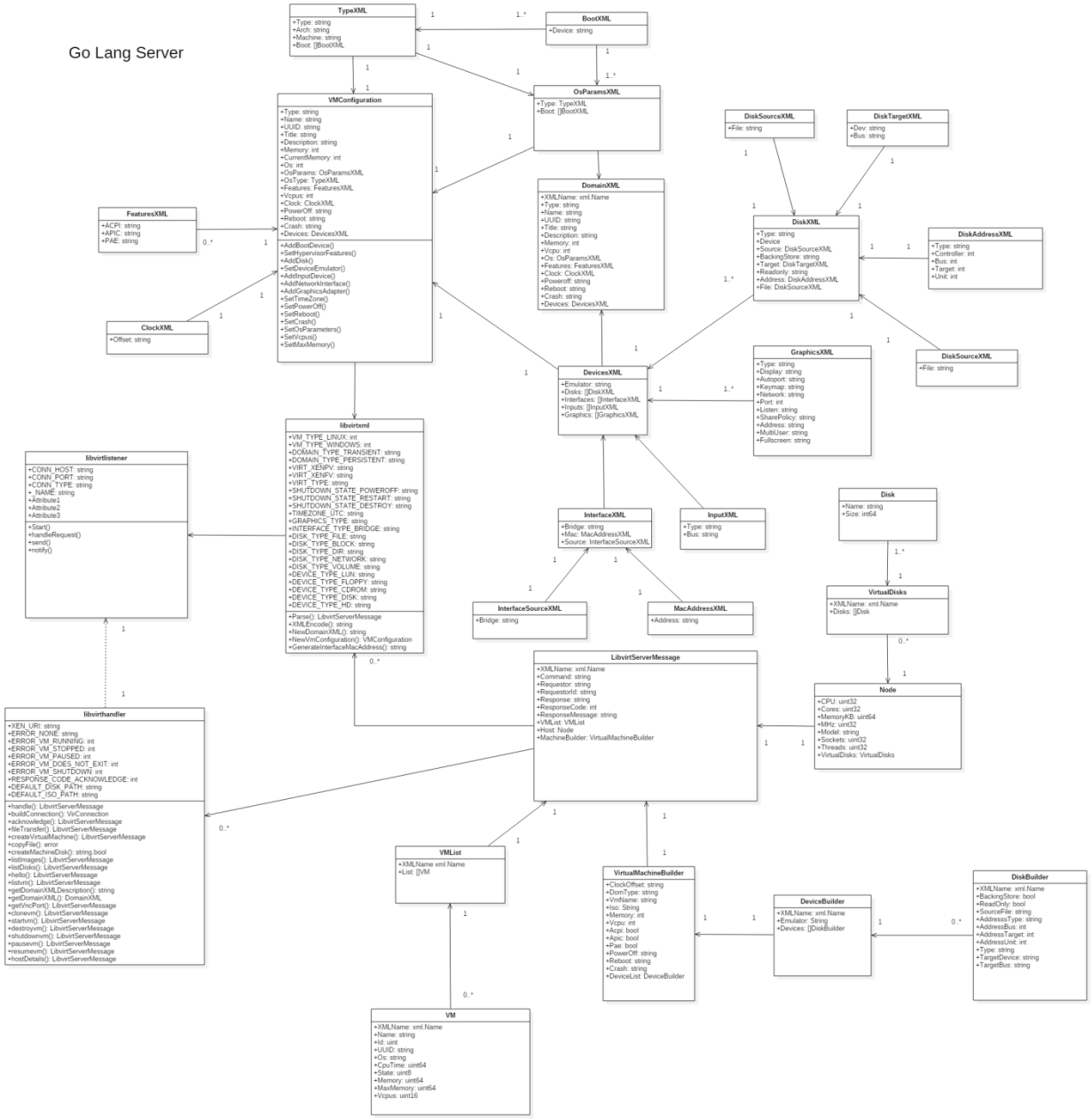

Go Lang Server
TypeXML
+Type: string
+Arch: string
+Machine: string
+Boot: []BootXML
BootXML
+Device: string
OsParamsXML
+Type: TypeXML
+Boot: []BootXML
VMConfiguration
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+CurrentMemory: int
+Os: int
+OsParams: OsParamsXML
+OsType: TypeXML
+Features: FeaturesXML
+Vcpus: int
+Clock: ClockXML
+PowerOff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
+AddBootDevice()
+SetHypervisorFeatures()
+AddDisk()
+SetDeviceEmulator()
+AddInputDevice()
+AddNetworkInterface()
+AddGraphicsAdapter()
+SetTimeZone()
+SetPowerOff()
+SetReboot()
+SetCrash()
+SetOsParameters()
+SetVcpus()
+SetMaxMemory()
FeaturesXML
+ACPI: string
+APIC: string
+PAE: string
ClockXML
+Offset: string
DomainXML
+XMLName: xml.Name
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+Vcpu: int
+Os: OsParamsXML
+Features: FeaturesXML
+Clock: ClockXML
+Poweroff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
DiskSourceXML
+File: string
DiskTargetXML
+Dev: string
+Bus: string
DiskXML
+Type: string
+Device
+Source: DiskSourceXML
+BackingStore: string
+Target: DiskTargetXML
+Readonly: string
+Address: DiskAddressXML
+File: DiskSourceXML
DiskAddressXML
+Type: string
+Controller: int
+Bus: int
+Target: int
+Unit: int
DevicesXML
+Emulator: string
+Disks: []DiskXML
+Interfaces: []InterfaceXML
+Inputs: []InputXML
+Graphics: []GraphicsXML
GraphicsXML
+Type: string
+Display: string
+Autoport: string
+Keymap: string
+Network: string
+Port: int
+Listen: string
+SharePolicy: string
+Address: string
+MultiUser: string
+Fullscreen: string
DiskSourceXML
+File: string
libvirtxml
+VM_TYPE_LINUX: int
+VM_TYPE_WINDOWS: int
+DOMAIN_TYPE_TRANSIENT: string
+DOMAIN_TYPE_PERSISTENT: string
+VIRT_XENPV: string
+VIRT_XENFV: string
+VIRT_TYPE: string
+SHUTDOWN_STATE_POWEROFF: string
+SHUTDOWN_STATE_RESTART: string
+SHUTDOWN_STATE_DESTROY: string
+TIMEZONE_UTC: string
+GRAPHICS_TYPE: string
+INTERFACE_TYPE_BRIDGE: string
+DISK_TYPE_FILE: string
+DISK_TYPE_BLOCK: string
+DISK_TYPE_DIR: string
+DISK_TYPE_NETWORK: string
+DISK_TYPE_VOLUME: string
+DEVICE_TYPE_LUN: string
+DEVICE_TYPE_FLOPPY: string
+DEVICE_TYPE_CDROM: string
+DEVICE_TYPE_DISK: string
+DEVICE_TYPE_HD: string
+Parse(): LibvirtServerMessage
+XMLEncode(): string
+NewDomainXML(): string
+NewVmConfiguration(): VMConfiguration
+GenerateInterfaceMacAddress(): string
libvirtlistener
+CONN_HOST: string
+CONN_PORT: string
+CONN_TYPE: string
+_NAME: string
+Attribute1
+Attribute2
+Attribute3
+Start()
+handleRequest()
+send()
+notify()
InterfaceXML
+Bridge: string
+Mac: MacAddressXML
+Source: InterfaceSourceXML
InputXML
+Type: string
+Bus: string
InterfaceSourceXML
+Bridge: string
MacAddressXML
+Address: string
Disk
+Name: string
+Size: int64
VirtualDisks
+XMLName: xml.Name
+Disks: []Disk
LibvirtServerMessage
+XMLName: xml.Name
+Command: string
+Requestor: string
+RequestorId: string
+Response: string
+ResponseCode: int
+ResponseMessage: string
+VMList: VMList
+Host: Node
+MachineBuilder: VirtualMachineBuilder
Node
+CPU: uint32
+Cores: uint32
+MemoryKB: uint64
+MHz: uint32
+Model: string
+Sockets: uint32
+Threads: uint32
+VirtualDisks: VirtualDisks
libvirthandler
+XEN_URI: string
+ERROR_NONE: string
+ERROR_VM_RUNNING: int
+ERROR_VM_STOPPED: int
+ERROR_VM_PAUSED: int
+ERROR_VM_DOES_NOT_EXIT: int
+ERROR_VM_SHUTDOWN: int
+RESPONSE_CODE_ACKNOWLEDGE: int
+DEFAULT_DISK_PATH: string
+DEFAULT_ISO_PATH: string
+handle(): LibvirtServerMessage
+buildConnection(): VirConnection
+acknowledge(): LibvirtServerMessage
+fileTransfer(): LibvirtServerMessage
+createVirtualMachine(): LibvirtServerMessage
+copyFile(): error
+createMachineDisk(): string,bool
+listImages(): LibvirtServerMessage
+listDisks(): LibvirtServerMessage
+hello(): LibvirtServerMessage
+listvm(): LibvirtServerMessage
+getDomainXMLDescription(): string
+getDomainXML(): DomainXML
+getVncPort(): LibvirtServerMessage
+clonevm(): LibvirtServerMessage
+startvm(): LibvirtServerMessage
+destroyvm(): LibvirtServerMessage
+shutdownvm(): LibvirtServerMessage
+pausevm(): LibvirtServerMessage
+resumevm(): LibvirtServerMessage
+hostDetails(): LibvirtServerMessage
VMList
+XMLName xml.Name
+List: []VM
VirtualMachineBuilder
+ClockOffset: string
+DomType: string
+VmName: string
+Iso: String
+Memory: int
+Vcpu: int
+Acpi: bool
+Apic: bool
+Pae: bool
+PowerOff: string
+Reboot: string
+Crash: string
+DeviceList: DeviceBuilder
DeviceBuilder
+XMLName: xml.Name
+Emulator: String
+Devices: []DiskBuilder
DiskBuilder
+XMLName: xml.Name
+BackingStore: bool
+ReadOnly: bool
+SourceFile: string
+AddressType: string
+AddressBus: int
+AddressTarget: int
+AddressUnit: int
+Type: string
+TargetDevice: string
+TargetBus: string
VM
+XMLName: xml.Name
+Name: string
+Id: uint
+UUID: string
+Os: string
+CpuTime: uint64
+State: uint8
+Memory: uint64
+MaxMemory: uint64
+Vcpus: uint16

Figure 25 - Transfer Virtual Machine Images

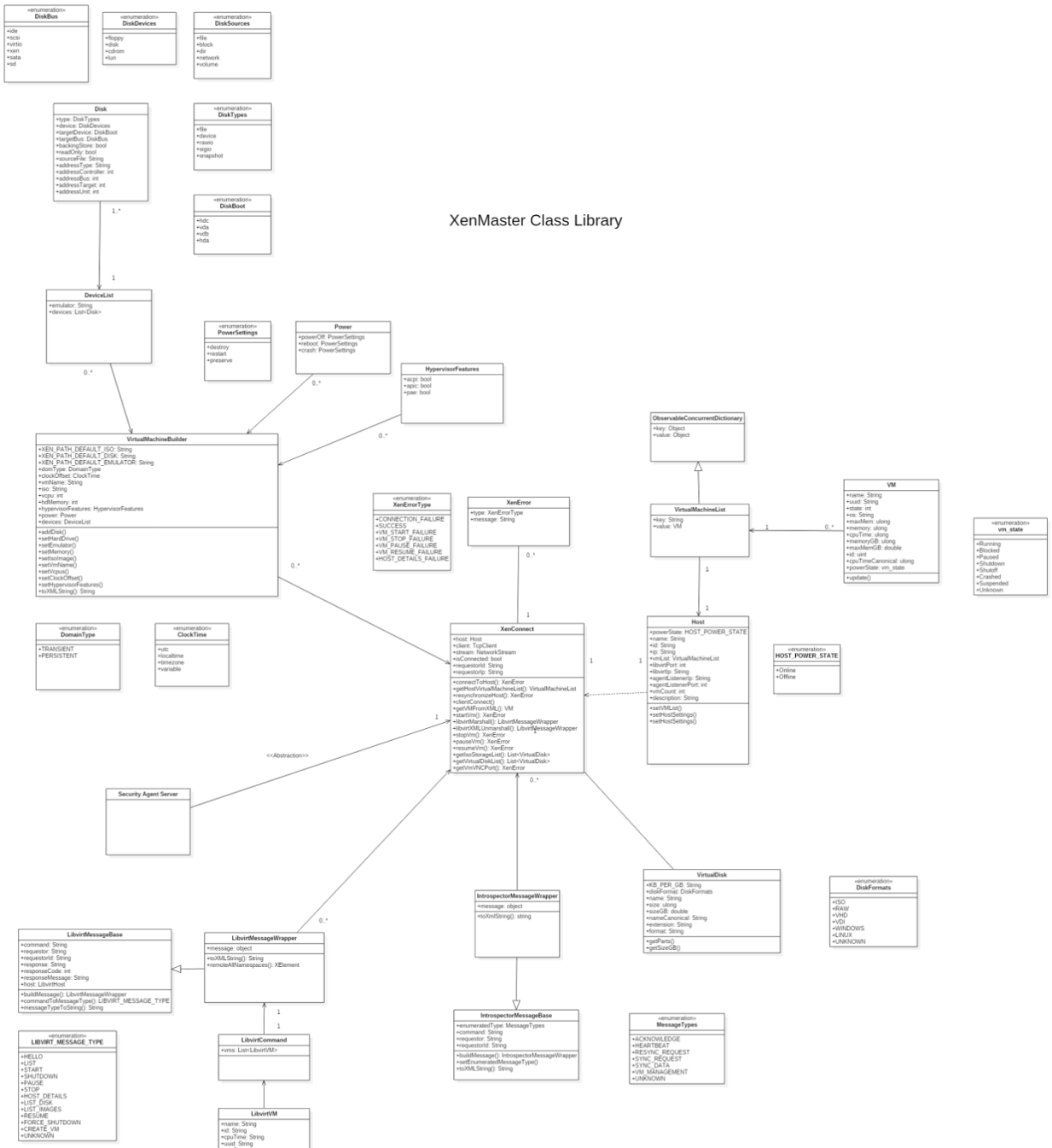

XenMaster Class Library
«enumeration»
DiskBus
+ide
+scsi
+virtio
+xen
+sata
+sd
«enumeration»
DiskDevices
+floppy
+disk
+cdrom
+lun
«enumeration»
DiskSources
+file
+block
+dir
+network
+volume
Disk
+type: DiskTypes
+device: DiskDevices
+targetDevice: DiskBoot
+targetBus: DiskBus
+backingStore: bool
+readOnly: bool
+sourceFile: String
+addressType: String
+addressController: int
+addressBus: int
+addressTarget: int
+addressUnit: int
«enumeration»
DiskTypes
+file
+device
+raxio
+sigio
+snapshot
«enumeration»
DiskBoot
+hdc
+vda
+vdb
+hda
DeviceList
+emulator: String
+devices: List<Disk>
«enumeration»
PowerSettings
+destroy
+restart
+preserve
Power
+powerOff: PowerSettings
+reboot: PowerSettings
+crash: PowerSettings
HypervisorFeatures
+acpi: bool
+apic: bool
+pae: bool
VirtualMachineBuilder
+XEN_PATH_DEFAULT_ISO: String
+XEN_PATH_DEFAULT_DISK: String
+XEN_PATH_DEFAULT_EMULATOR: String
+domType: DomainType
+clockOffset: ClockTime
+vmName: String
+iso: String
+cpu: int
+hdMemory: int
+hypervisorFeatures: HypervisorFeatures
+power: Power
+devices: DeviceList
+addDisk()
+setHardDrive()
+setEmulator()
+setMemory()
+setIsoImage()
+setVmName()
+setVcpus()
+setClockOffset()
+setHypervisorFeatures()
+toXMLString(): String
«enumeration»
XenErrorType
+CONNECTION_FAILURE
+SUCCESS
+VM_START_FAILURE
+VM_STOP_FAILURE
+VM_PAUSE_FAILURE
+VM_RESUME_FAILURE
+HOST_DETAILS_FAILURE
XenError
+type: XenErrorType
+message: String
ObservableConcurrentDictionary
+key: Object
+value: Object
VirtualMachineList
+key: String
+value: VM
VM
+name: String
+uuid: String
+state: int
+os: String
+maxMem: ulong
+memory: ulong
+cpuTime: ulong
+memoryGB: ulong
+maxMemGB: double
+id: uint
+cpuTimeCanonical: ulong
+powerState: vm_state
+update()
«enumeration»
vm_state
+Running
+Blocked
+Paused
+Shutdown
+Shutoff
+Crashed
+Suspended
+Unknown
«enumeration»
DomainType
+TRANSIENT
+PERSISTENT
«enumeration»
ClockTime
+utc
+localtime
+timezone
+variable
XenConnect
+host: Host
+client: TcpClient
+stream: NetworkStream
+isConnected: bool
+requestorId: String
+requestorIp: String
+connectToHost(): XenError
+getHostVirtualMachineList(): VirtualMachineList
+resynchronizeHost(): XenError
+clientConnect()
+getVMFromXML(): VM
+startVm(): XenError
+libvirtMarshall(): LibvirtMessageWrapper
+libvirtXMLUnmarshall(): LibvirtMessageWrapper
+stopVm(): XenError
+pauseVm(): XenError
+resumeVm(): XenError
+getIsoStorageList(): List<VirtualDisk>
+getVirtualDiskList(): List<VirtualDisk>
+getVmVNCPort(): XenError
Host
+powerState: HOST_POWER_STATE
+name: String
+id: String
+ip: String
+vmList: VirtualMachineList
+libvirtPort: int
+libvirtIp: String
+agentListenerIp: String
+agentListenerPort: int
+vmCount: int
+description: String
+setVMList()
+setHostSettings()
+setHostSettings()
«enumeration»
HOST_POWER_STATE
+Online
+Offline
<<Abstraction>>
Security Agent Server
VirtualDisk
+KB_PER_GB: String
+diskFormat: DiskFormats
+name: String
+size: ulong
+sizeGB: double
+nameCanonical: String
+extension: String
+format: String
+getParts()
+getSizeGB()
«enumeration»
DiskFormats
+ISO
+RAW
+VHD
+VDI
+WINDOWS
+LINUX
+UNKNOWN
IntrospectorMessageWrapper
+message: object
+toXmlString(): string
LibvirtMessageBase
+command: String
+requestor: String
+requestorId: String
+response: String
+responseCode: int
+responseMessage: String
+host: LibvirtHost
+buildMessage(): LibvirtMessageWrapper
+commandToMessageType(): LIBVIRT_MESSAGE_TYPE
+messageTypeToString(): String
LibvirtMessageWrapper
+message: object
+toXMLString(): String
+remoteAllNamespaces(): XElement
IntrospectorMessageBase
+enumeratedType: MessageTypes
+command: String
+requestor: String
+requestorId: String
+buildMessage(): IntrospectorMessageWrapper
+setEnumeratedMessageType()
+toXMLString(): String
«enumeration»
MessageTypes
+ACKNOWLEDGE
+HEARTBEAT
+RESYNC_REQUEST
+SYNC_REQUEST
+SYNC_DATA
+VM_MANAGEMENT
+UNKNOWN
«enumeration»
LIBVIRT_MESSAGE_TYPE
+HELLO
+LIST
+START
+SHUTDOWN
+PAUSE
+STOP
+HOST_DETAILS
+LIST_DISK
+LIST_IMAGES
+RESUME
+FORCE_SHUTDOWN
+CREATE_VM
+UNKNOWN
LibvirtCommand
+vms: List<LibvirtVM>
LibvirtVM
+name: String
+id: String
+cpuTime: String
+uuid: String
+os: String
+state: String
+maxMemory: String
+memory: String
+vcpus: String

Figure 26 - Clone Virtual Machine

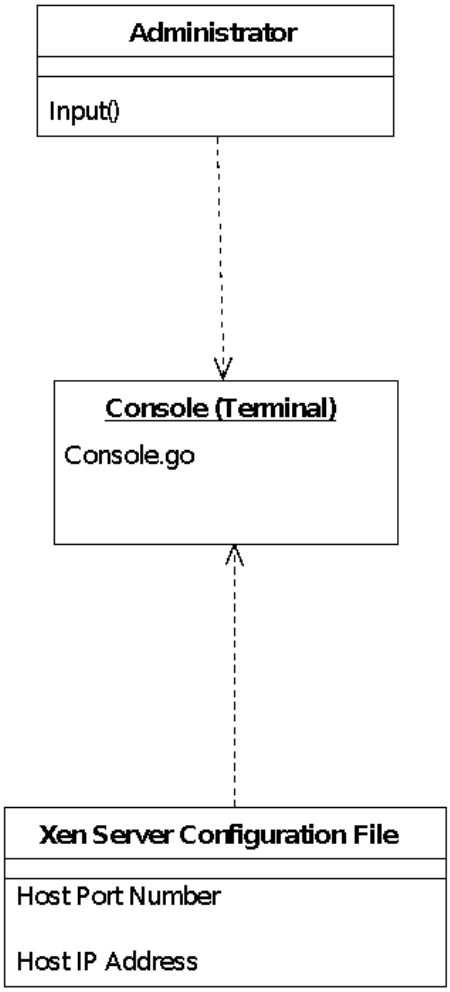

Administrator
Input()
Console (Terminal)
Console.go
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 27 - Display what is typed by the user into the interface

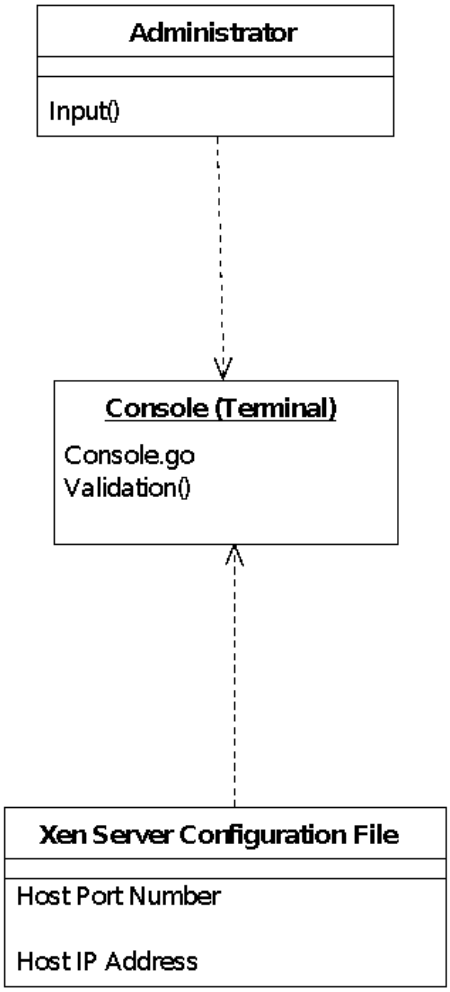

Administrator
Input()
Console (Terminal)
Console.go
Validation()
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 28 - Validate input values for both IP address and Port

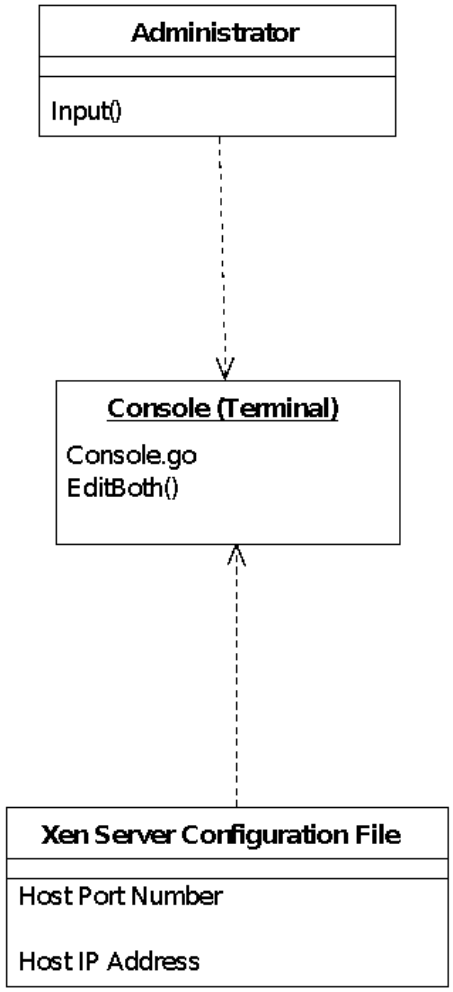

Administrator
Input()
Console (Terminal)
Console.go
EditBoth()
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 29 - Enter all the values to UI and update through one submission

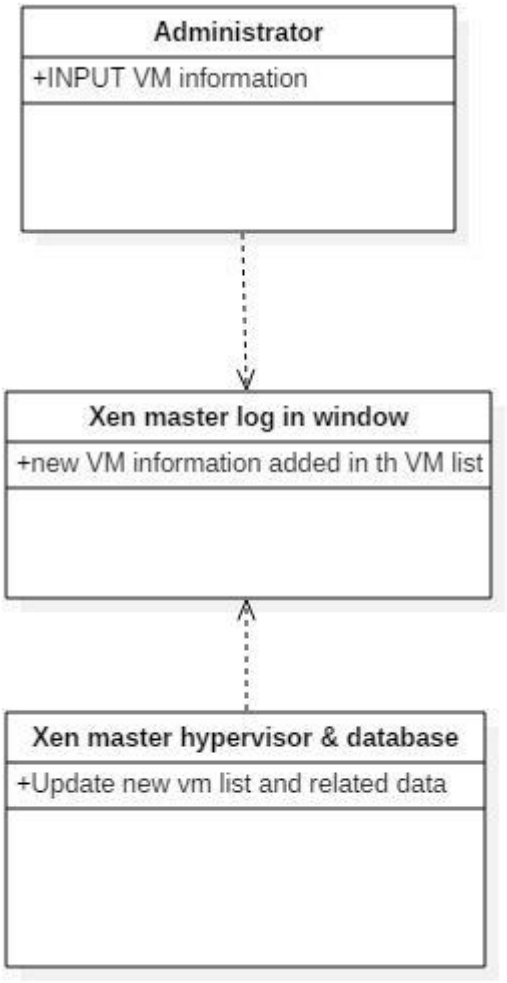



Figure 30 - Clean up user interface

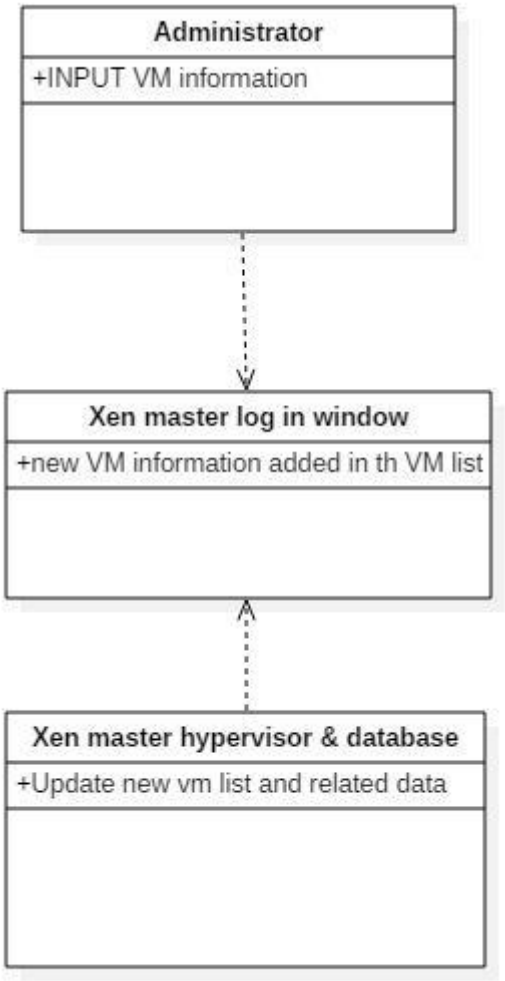



Figure 31 - Implement “Create” Virtual Machine

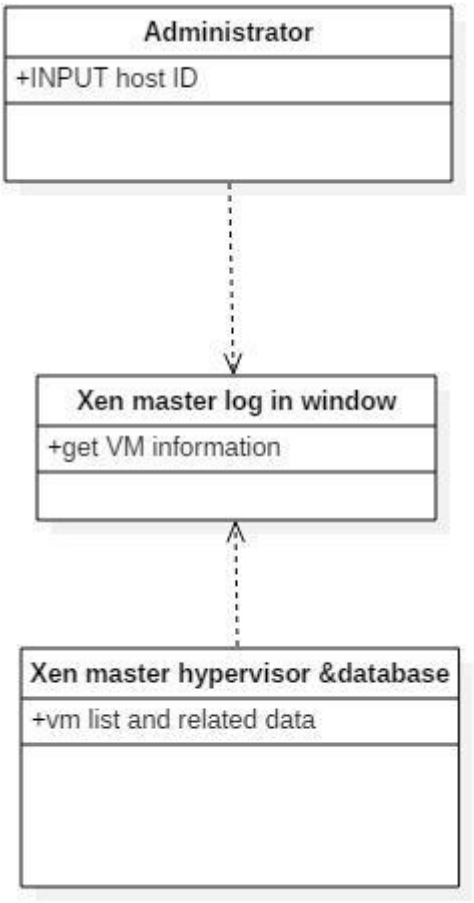



Figure 32 - Display status of virtual machines

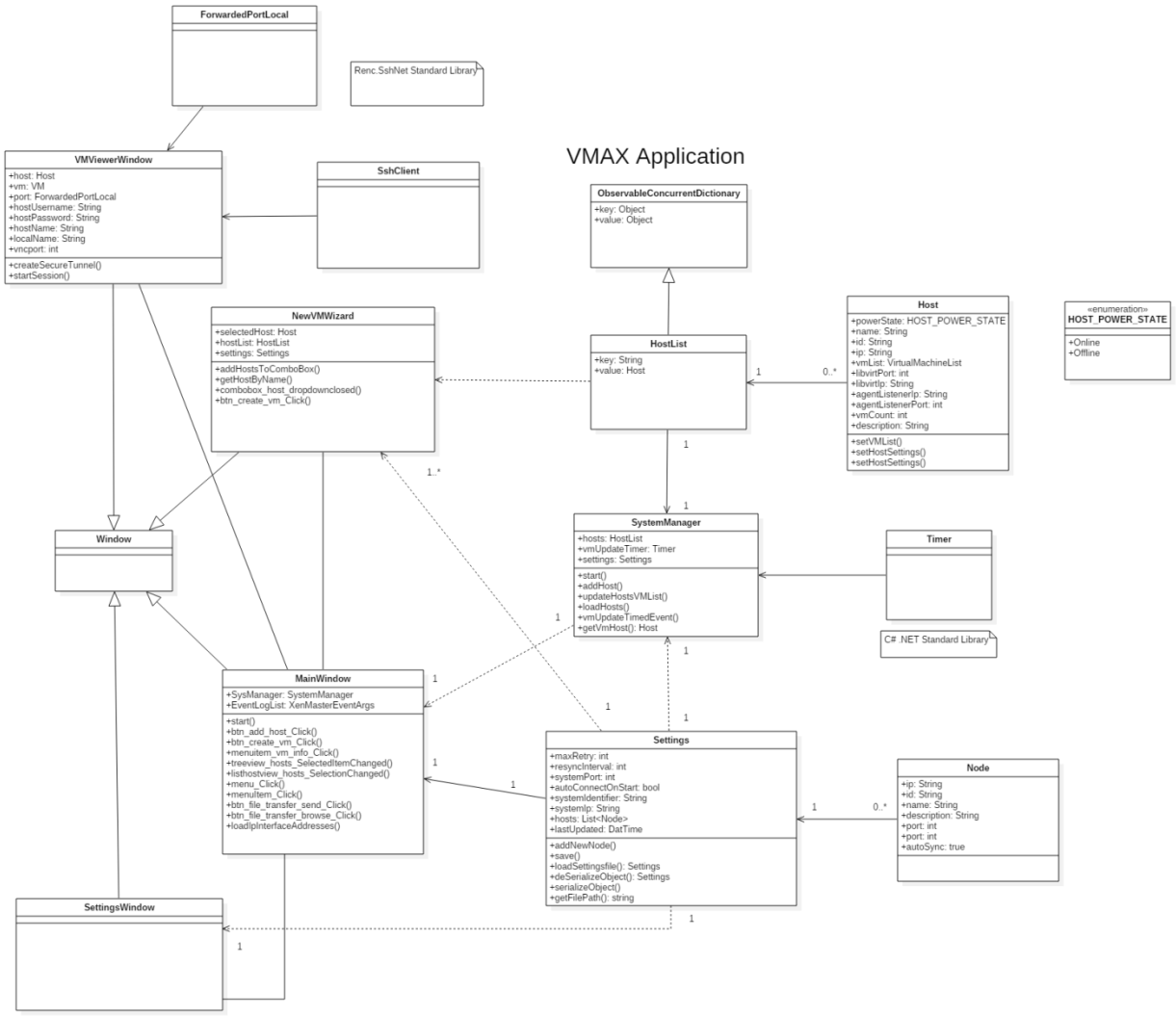



Figure 33 - Display User Notifications

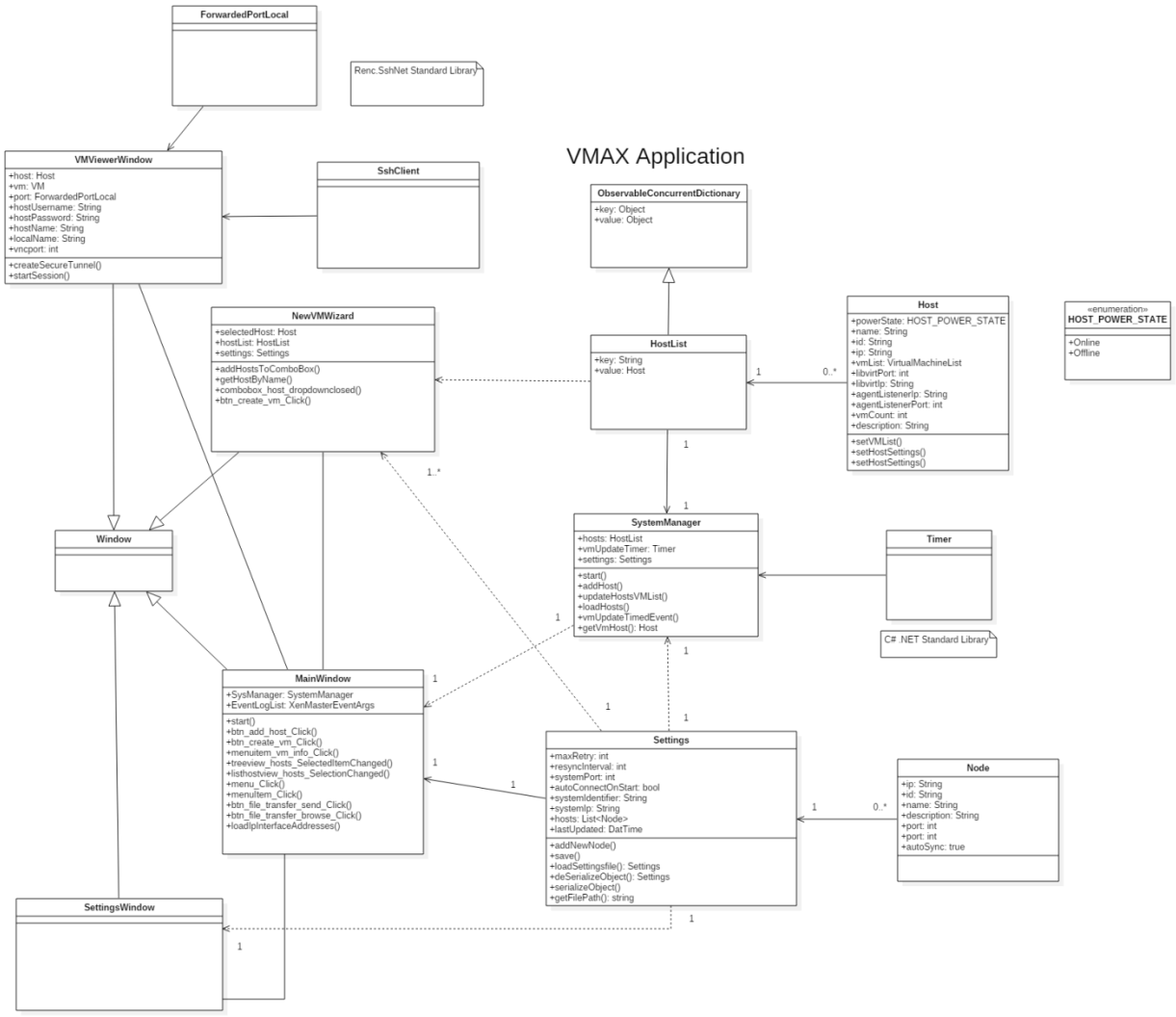



Figure 34 - Select VMAX Network Interface

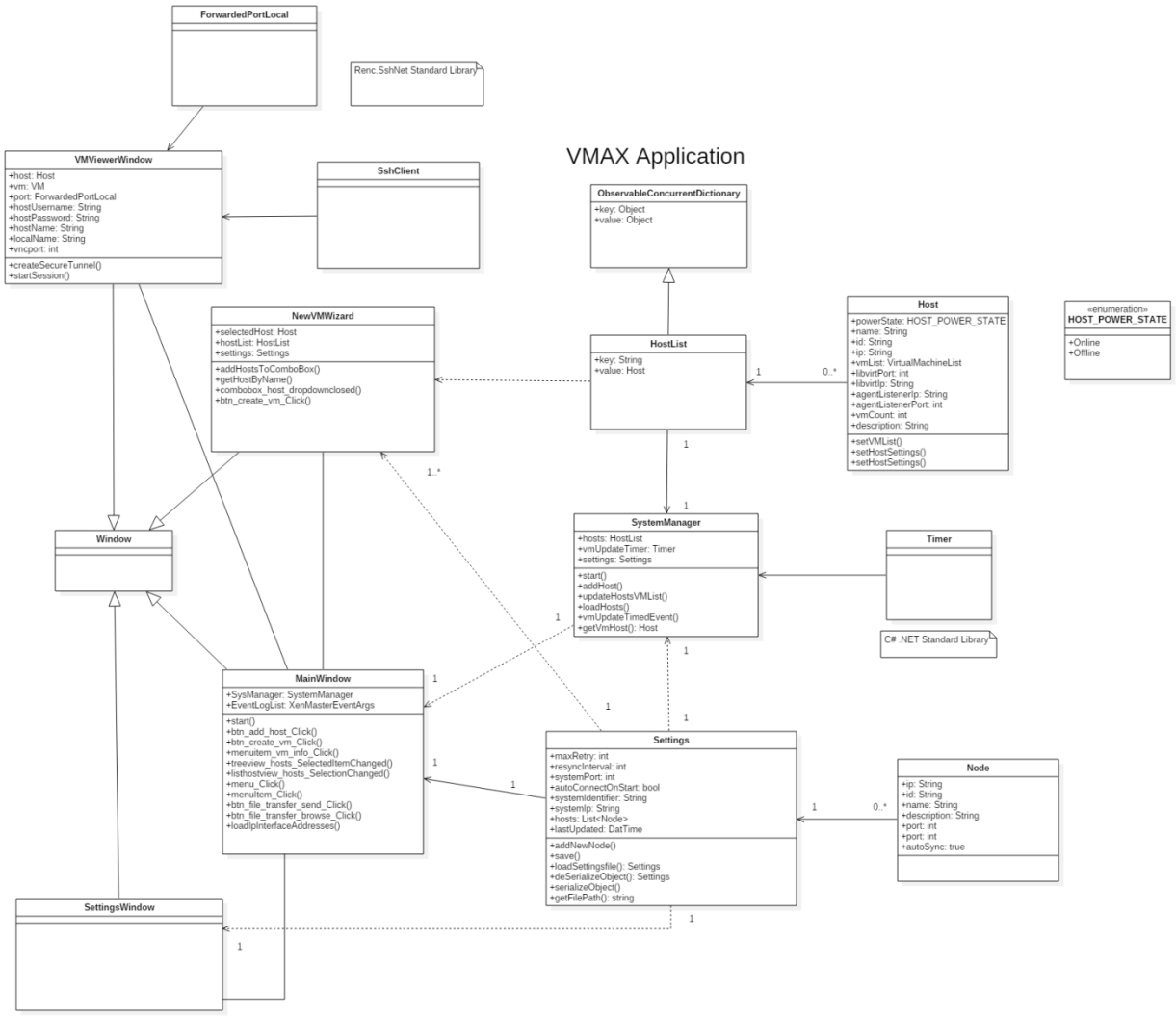



Figure 35 - Delete Virtual Machine

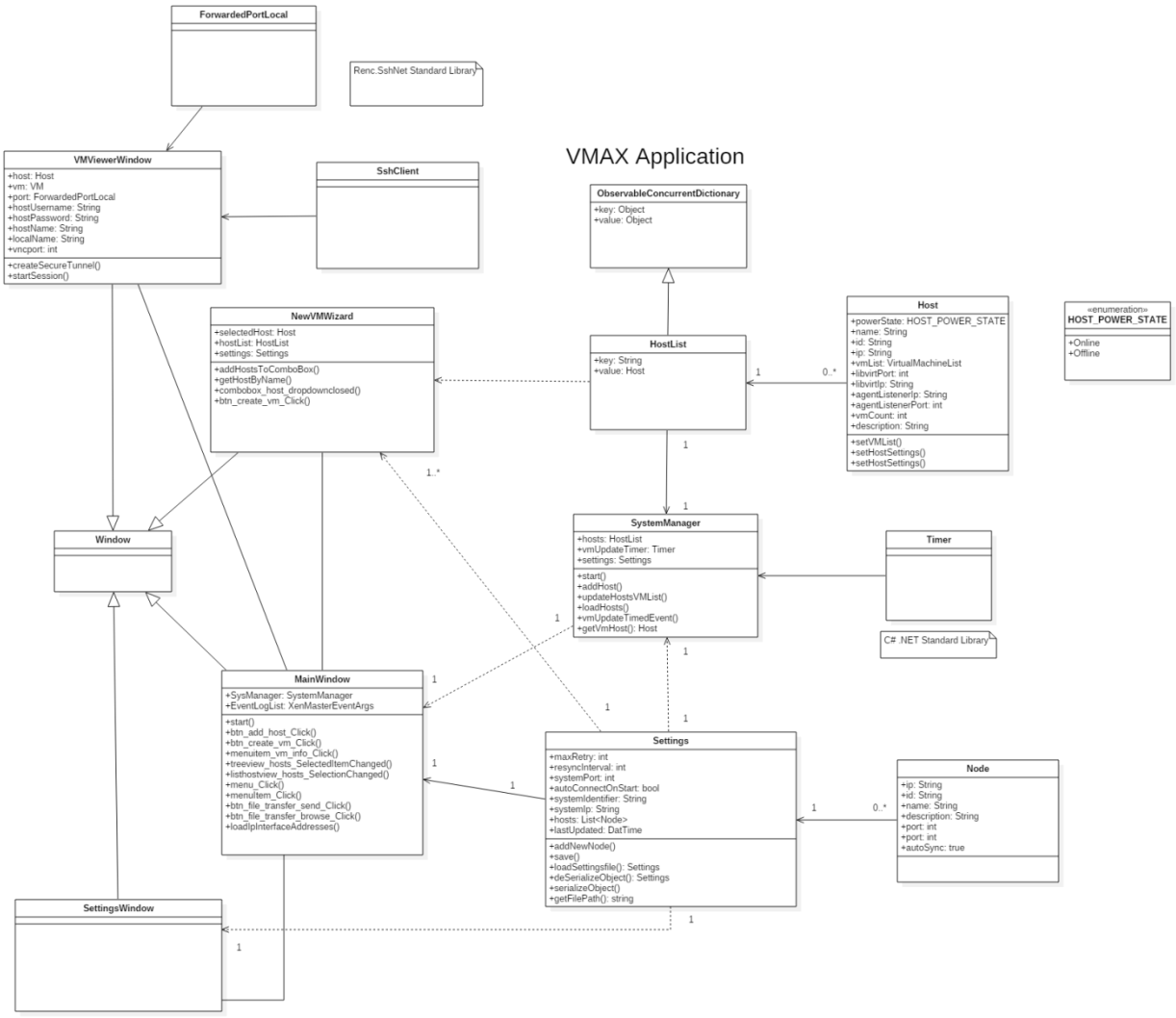



Figure 36 - Display Performance Parameters

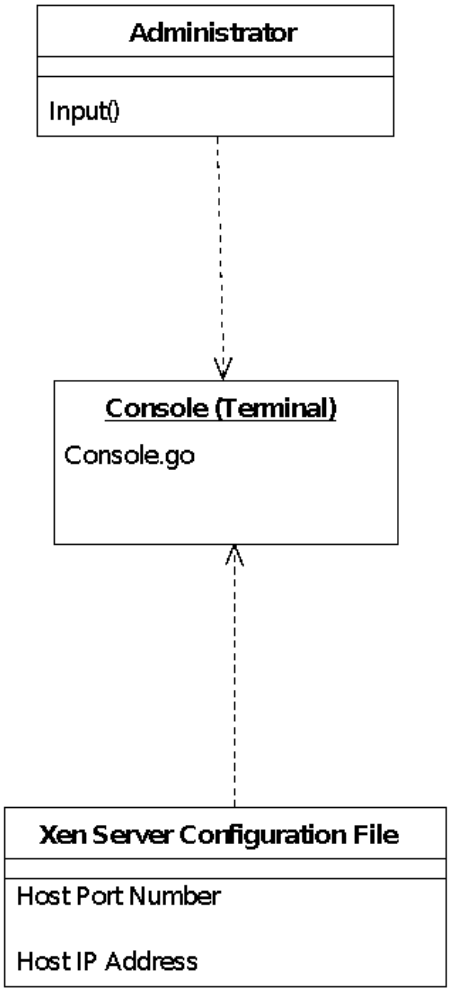

Administrator
Input()
Console (Terminal)
Console.go
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 37 - Clean up the interface

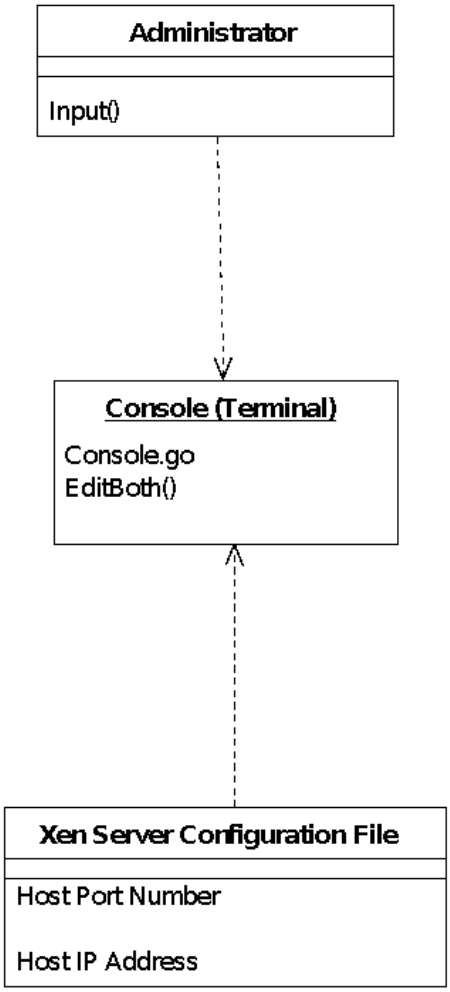

Administrator
Input()
Console (Terminal)
Console.go
EditBoth()
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 38 - Update all parameters with single selection

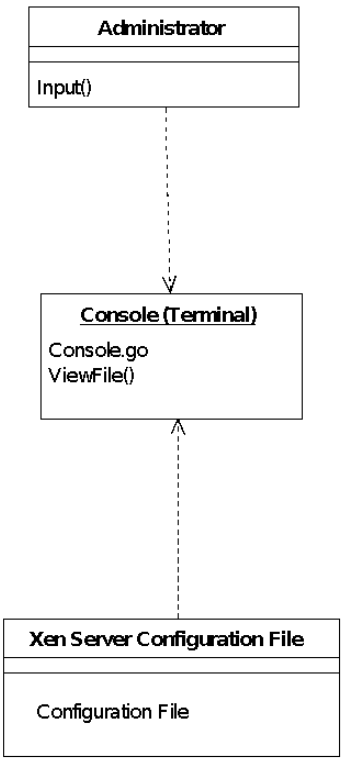



Figure 39 - View file should be a separate button/selection

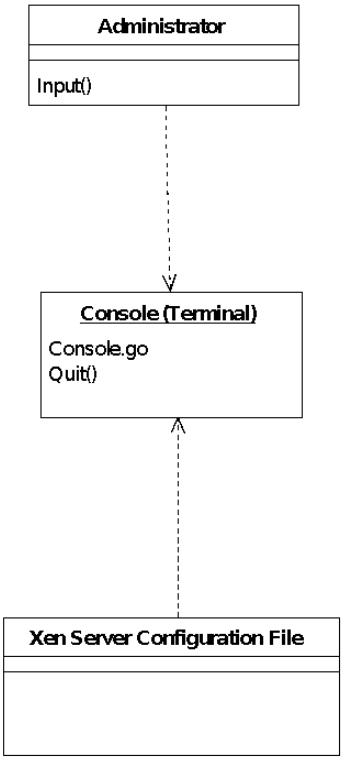



Figure 40 - Add a signal handler to catch the quit option

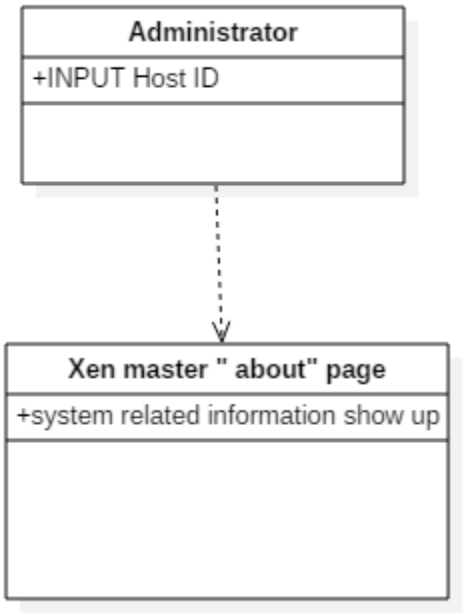



Figure 41 - Add more information on the home page

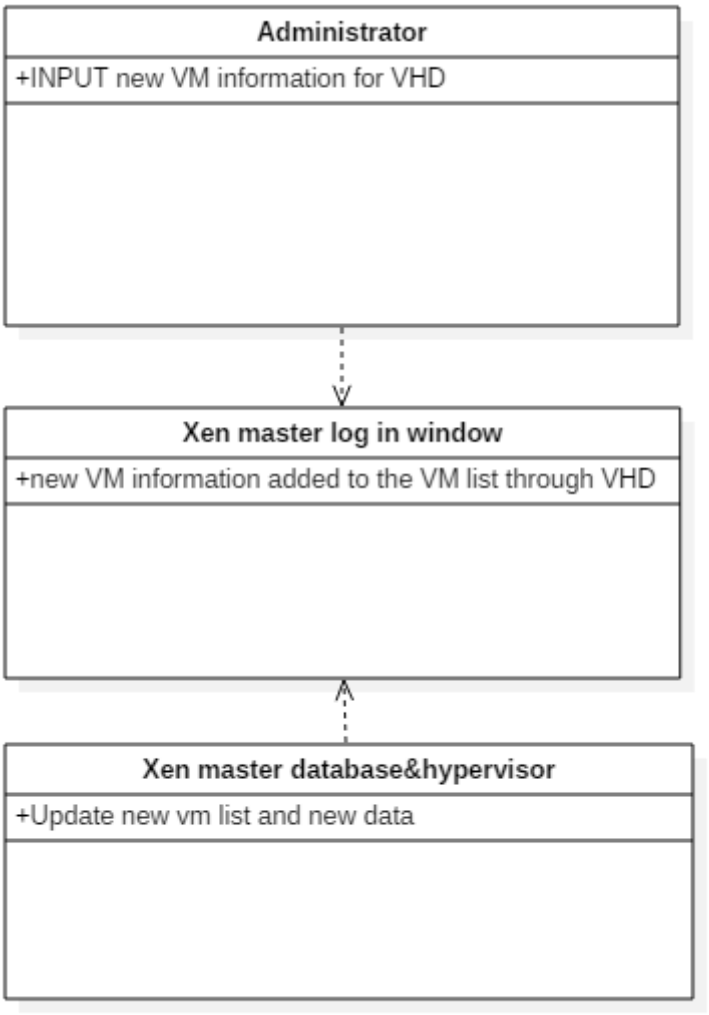



Figure 42 - Design an interface for listing the existing VHD

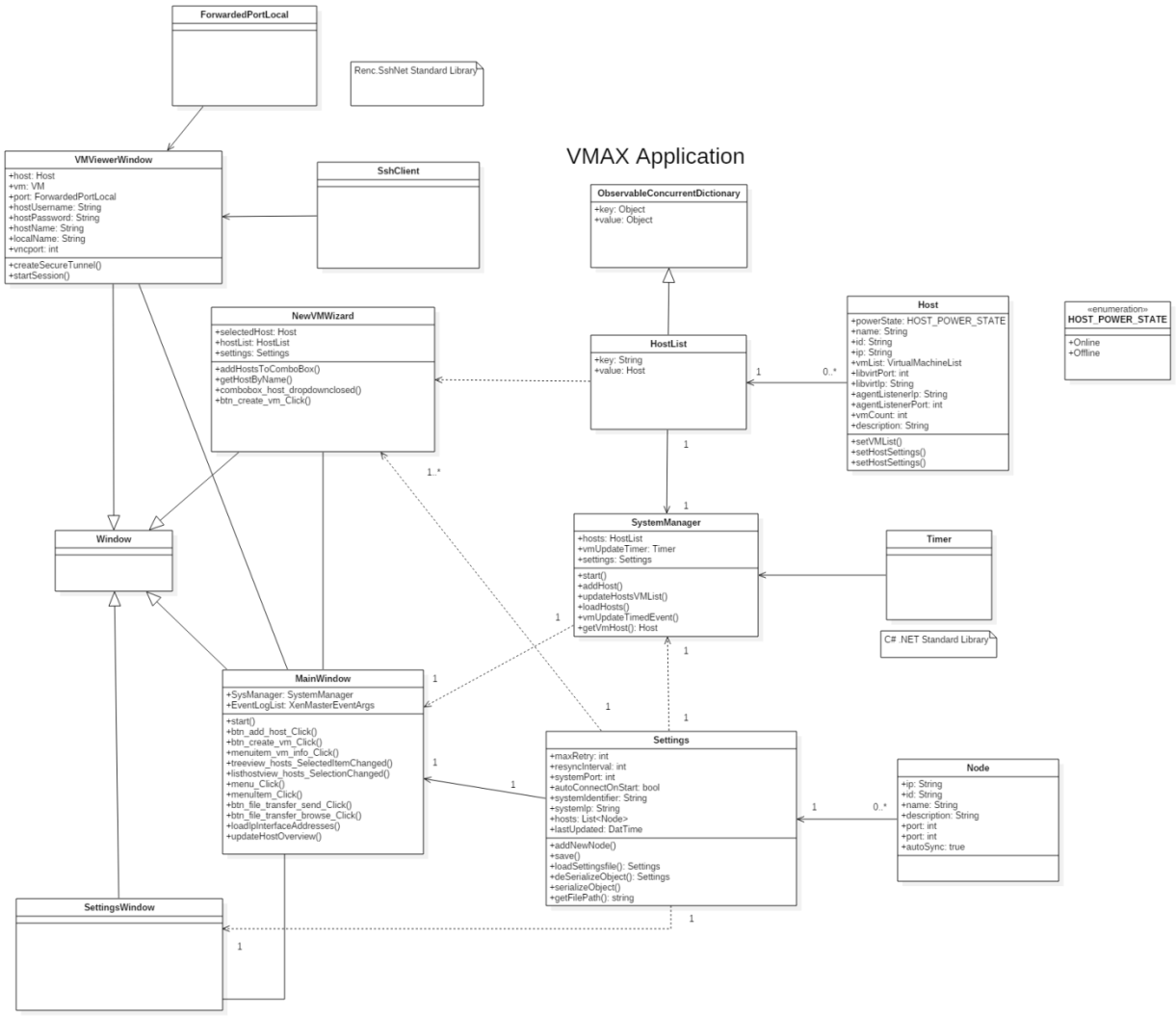



Figure 43 - Display host remote configuration

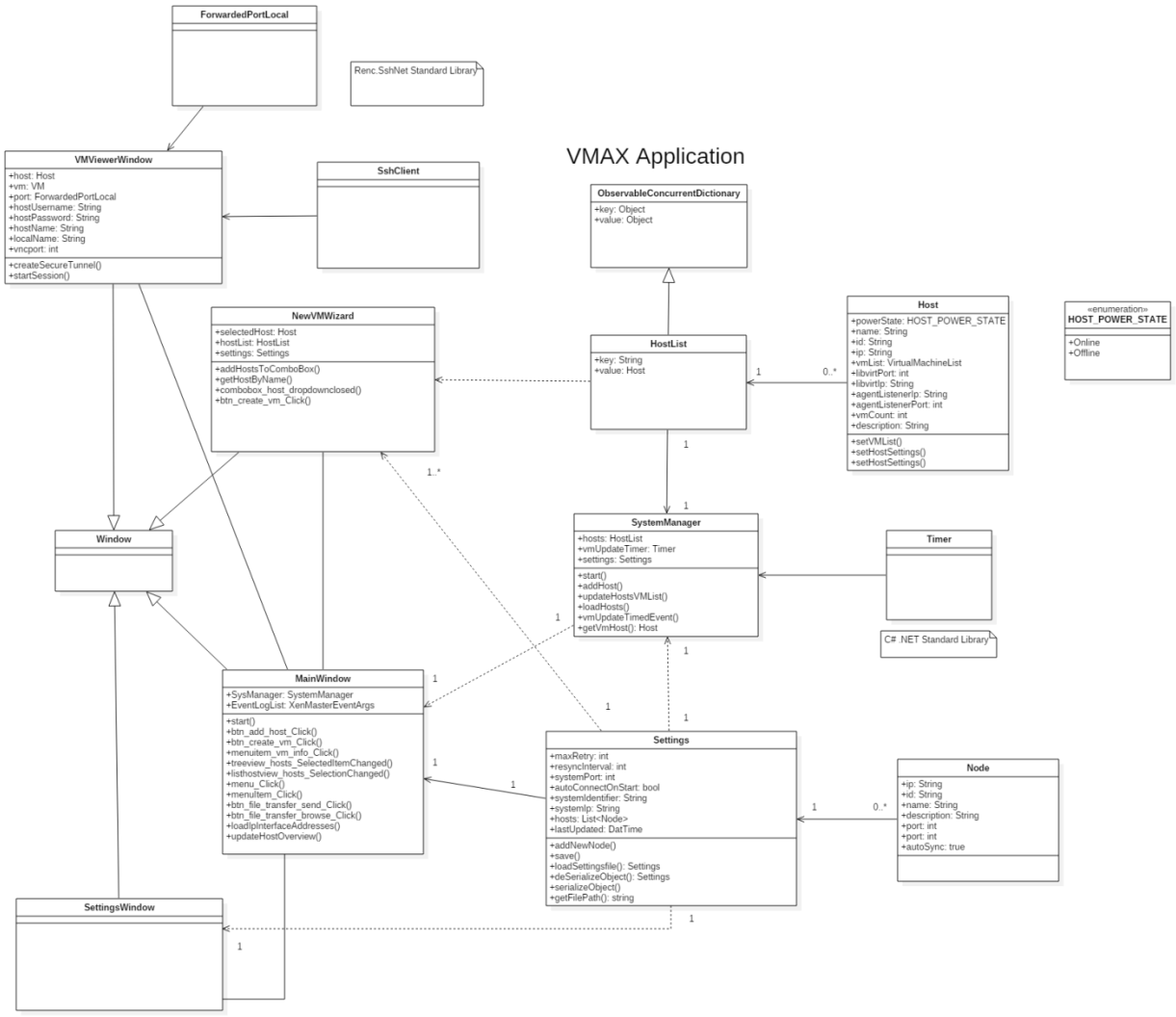



Figure 44 - Add host to VMAX settings

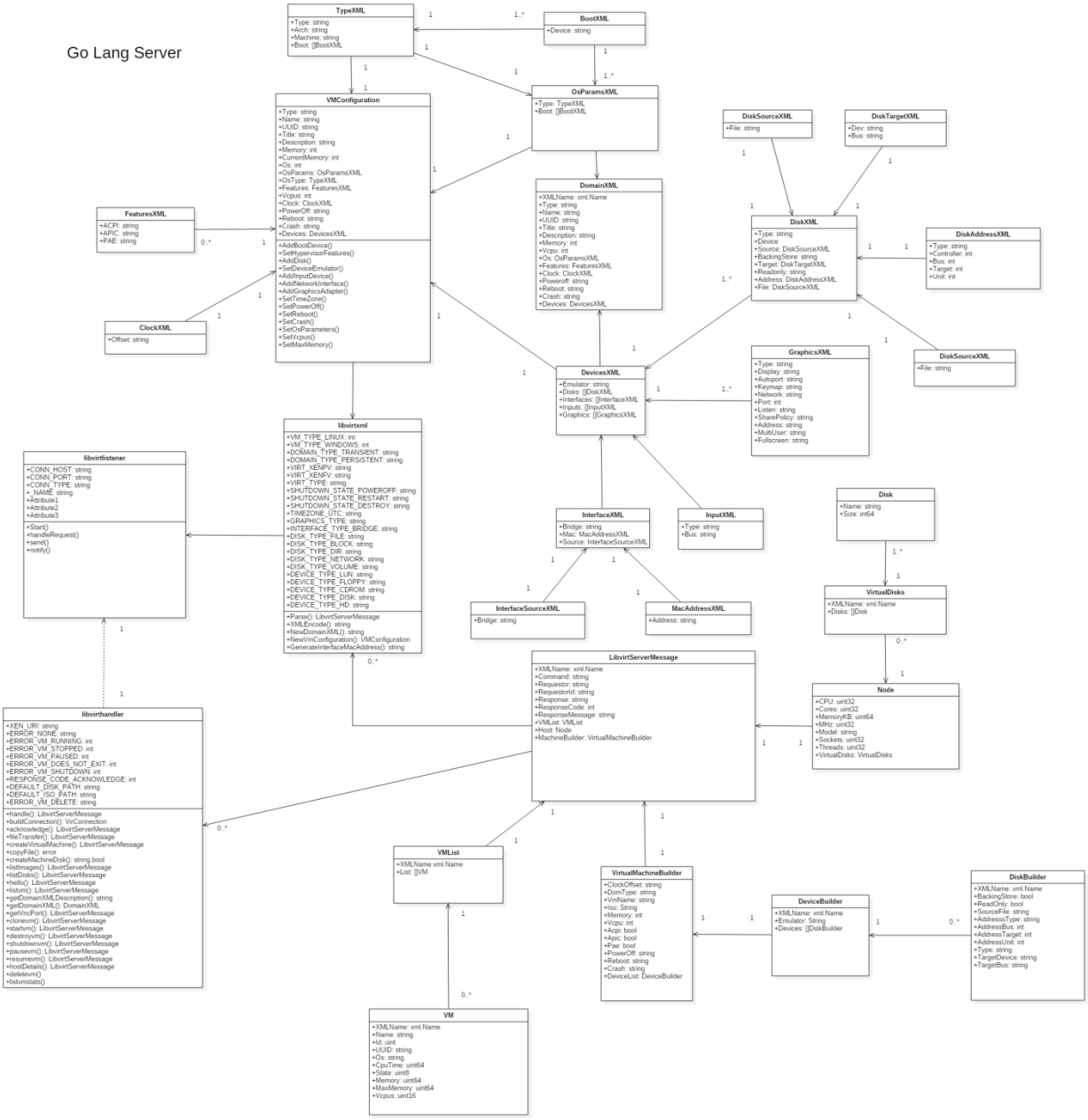

Go Lang Server
TypeXML
+Type: string
+Arch: string
+Machine: string
+Boot: []BootXML
BootXML
+Device: string
OsParamsXML
+Type: TypeXML
+Boot: []BootXML
VMConfiguration
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+CurrentMemory: int
+Os: int
+OsParams: OsParamsXML
+OsType: TypeXML
+Features: FeaturesXML
+Vcpus: int
+Clock: ClockXML
+PowerOff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
+AddBootDevice()
+SetHypervisorFeatures()
+AddDisk()
+SetDeviceEmulator()
+AddInputDevice()
+AddNetworkInterface()
+AddGraphicsAdapter()
+SetTimeZone()
+SetPowerOff()
+SetReboot()
+SetCrash()
+SetOsParameters()
+SetVcpus()
+SetMaxMemory()
FeaturesXML
+ACPI: string
+APIC: string
+PAE: string
ClockXML
+Offset: string
DomainXML
+XMLName: xml.Name
+Type: string
+Name: string
+UUID: string
+Title: string
+Description: string
+Memory: int
+Vcpu: int
+Os: OsParamsXML
+Features: FeaturesXML
+Clock: ClockXML
+Poweroff: string
+Reboot: string
+Crash: string
+Devices: DevicesXML
DiskSourceXML
+File: string
DiskTargetXML
+Dev: string
+Bus: string
DiskXML
+Type: string
+Device
+Source: DiskSourceXML
+BackingStore: string
+Target: DiskTargetXML
+Readonly: string
+Address: DiskAddressXML
+File: DiskSourceXML
DiskAddressXML
+Type: string
+Controller: int
+Bus: int
+Target: int
+Unit: int
DevicesXML
+Emulator: string
+Disks: []DiskXML
+Interfaces: []InterfaceXML
+Inputs: []InputXML
+Graphics: []GraphicsXML
GraphicsXML
+Type: string
+Display: string
+Autoport: string
+Keymap: string
+Network: string
+Port: int
+Listen: string
+SharePolicy: string
+Address: string
+MultiUser: string
+Fullscreen: string
DiskSourceXML
+File: string
libvirtxml
+VM_TYPE_LINUX: int
+VM_TYPE_WINDOWS: int
+DOMAIN_TYPE_TRANSIENT: string
+DOMAIN_TYPE_PERSISTENT: string
+VIRT_XENPV: string
+VIRT_XENFV: string
+VIRT_TYPE: string
+SHUTDOWN_STATE_POWEROFF: string
+SHUTDOWN_STATE_RESTART: string
+SHUTDOWN_STATE_DESTROY: string
+TIMEZONE_UTC: string
+GRAPHICS_TYPE: string
+INTERFACE_TYPE_BRIDGE: string
+DISK_TYPE_FILE: string
+DISK_TYPE_BLOCK: string
+DISK_TYPE_DIR: string
+DISK_TYPE_NETWORK: string
+DISK_TYPE_VOLUME: string
+DEVICE_TYPE_LUN: string
+DEVICE_TYPE_FLOPPY: string
+DEVICE_TYPE_CDROM: string
+DEVICE_TYPE_DISK: string
+DEVICE_TYPE_HD: string
+Parse(): LibvirtServerMessage
+XMLEncode(): string
+NewDomainXML(): string
+NewVmConfiguration(): VMConfiguration
+GenerateInterfaceMacAddress(): string
libvirtlistener
+CONN_HOST: string
+CONN_PORT: string
+CONN_TYPE: string
+ _NAME: string
+Attribute1
+Attribute2
+Attribute3
+Start()
+handleRequest()
+send()
+notify()
InterfaceXML
+Bridge: string
+Mac: MacAddressXML
+Source: InterfaceSourceXML
InputXML
+Type: string
+Bus: string
InterfaceSourceXML
+Bridge: string
MacAddressXML
+Address: string
Disk
+Name: string
+Size: int64
VirtualDisks
+XMLName: xml.Name
+Disks: []Disk
LibvirtServerMessage
+XMLName: xml.Name
+Command: string
+Requestor: string
+RequestorId: string
+Response: string
+ResponseCode: int
+ResponseMessage: string
+VMList: VMList
+Host: Node
+MachineBuilder: VirtualMachineBuilder
Node
+CPU: uint32
+Cores: uint32
+MemoryKB: uint64
+MHz: uint32
+Model: string
+Sockets: uint32
+Threads: uint32
+VirtualDisks: VirtualDisks
libvirthandler
+XEN_URI: string
+ERROR_NONE: string
+ERROR_VM_RUNNING: int
+ERROR_VM_STOPPED: int
+ERROR_VM_PAUSED: int
+ERROR_VM_DOES_NOT_EXIT: int
+ERROR_VM_SHUTDOWN: int
+RESPONSE_CODE_ACKNOWLEDGE: int
+DEFAULT_DISK_PATH: string
+DEFAULT_ISO_PATH: string
+ERROR_VM_DELETE: string
+handle(): LibvirtServerMessage
+buildConnection(): VirConnection
+acknowledge(): LibvirtServerMessage
+fileTransfer(): LibvirtServerMessage
+createVirtualMachine(): LibvirtServerMessage
+copyFile(): error
+createMachineDisk(): string,bool
+listImages(): LibvirtServerMessage
+listDisks(): LibvirtServerMessage
+hello(): LibvirtServerMessage
+listvm(): LibvirtServerMessage
+getDomainXMLDescription(): string
+getDomainXML(): DomainXML
+getVncPort(): LibvirtServerMessage
+clonevm(): LibvirtServerMessage
+startvm(): LibvirtServerMessage
+destroyvm(): LibvirtServerMessage
+shutdownvm(): LibvirtServerMessage
+pausevm(): LibvirtServerMessage
+resumevm(): LibvirtServerMessage
+hostDetails(): LibvirtServerMessage
+deletevm()
+listvmstats()
VMList
+XMLName xml.Name
+List: []VM
VirtualMachineBuilder
+ClockOffset: string
+DomType: string
+VmName: string
+Iso: String
+Memory: int
+Vcpu: int
+Acpi: bool
+Apic: bool
+Pae: bool
+PowerOff: string
+Reboot: string
+Crash: string
+DeviceList: DeviceBuilder
DeviceBuilder
+XMLName: xml.Name
+Emulator: String
+Devices: []DiskBuilder
DiskBuilder
+XMLName: xml.Name
+BackingStore: bool
+ReadOnly: bool
+SourceFile: string
+AddressType: string
+AddressBus: int
+AddressTarget: int
+AddressUnit: int
+Type: string
+TargetDevice: string
+TargetBus: string
VM
+XMLName: xml.Name
+Name: string
+Id: uint
+UUID: string
+Os: string
+CpuTime: uint64
+State: uint8
+Memory: uint64
+MaxMemory: uint64
+Vcpus: uint16

Figure 45 - Create Host Identifier

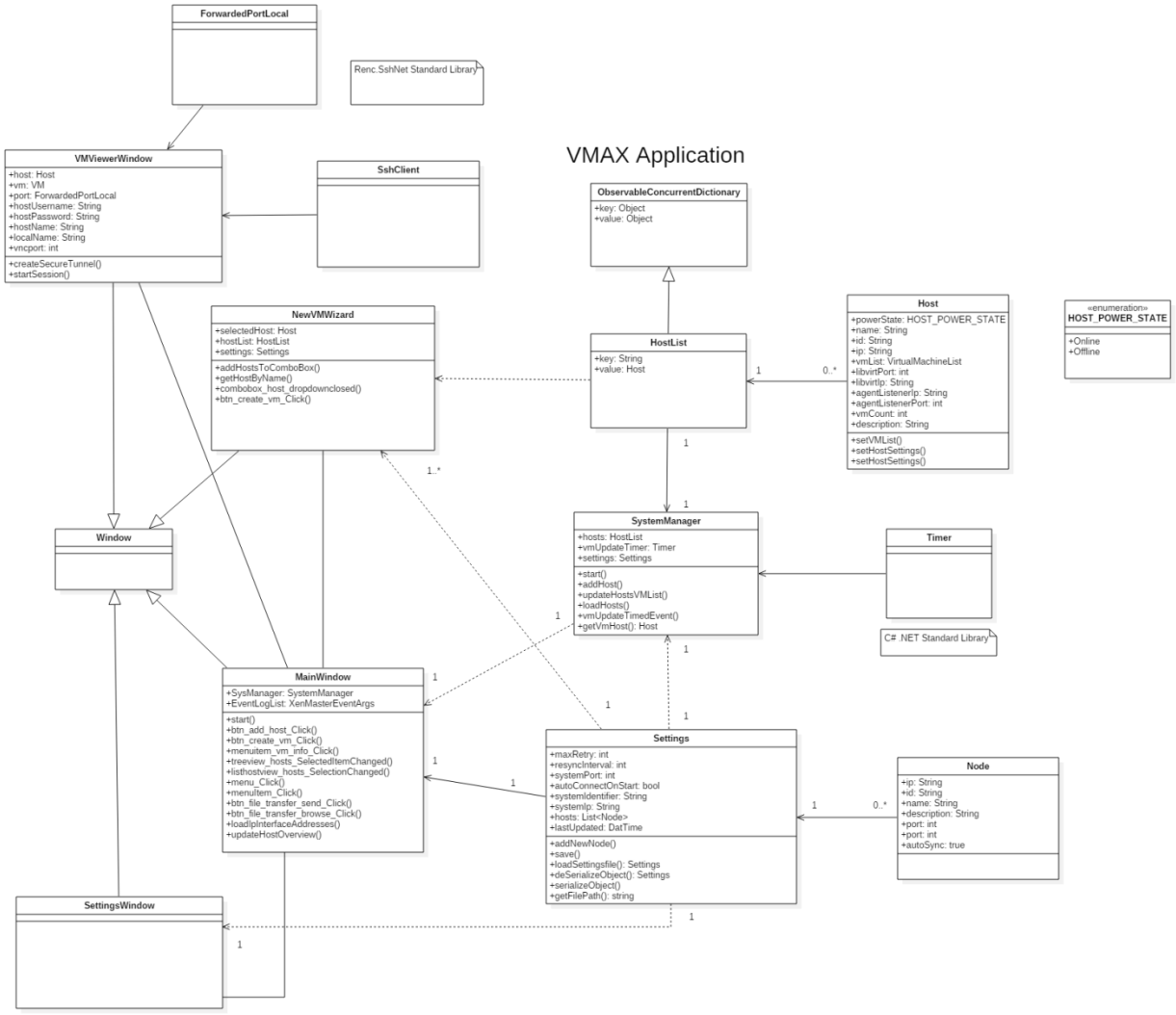



Figure 46 - Display host disk usage

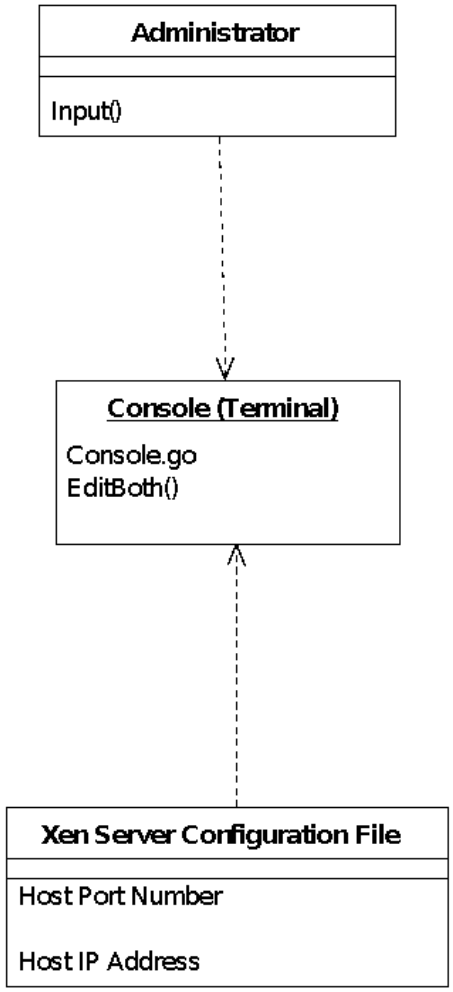

Administrator
Input()
Console (Terminal)
Console.go
EditBoth()
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 47 - Design the interface for multiple inputs and update

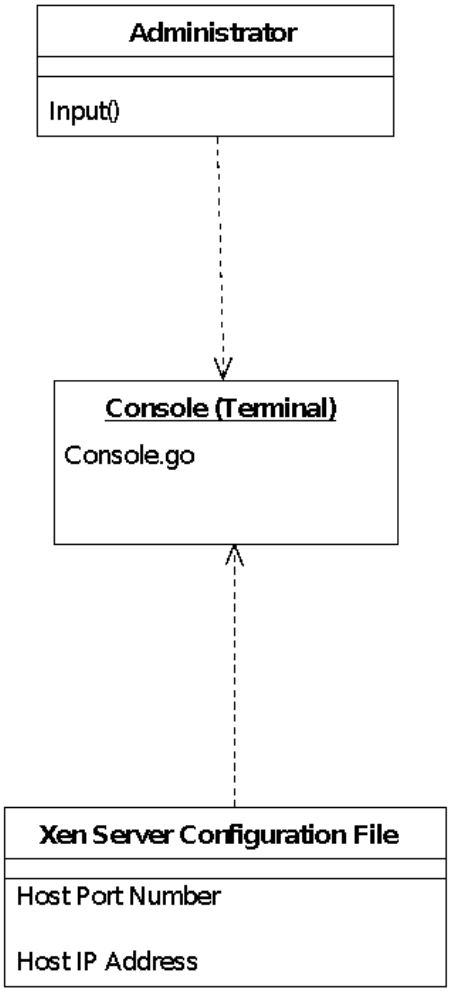

Administrator
Input()
Console (Terminal)
Console.go
Xen Server Configuration File
Host Port Number
Host IP Address

Figure 48 - Design the interface for user display

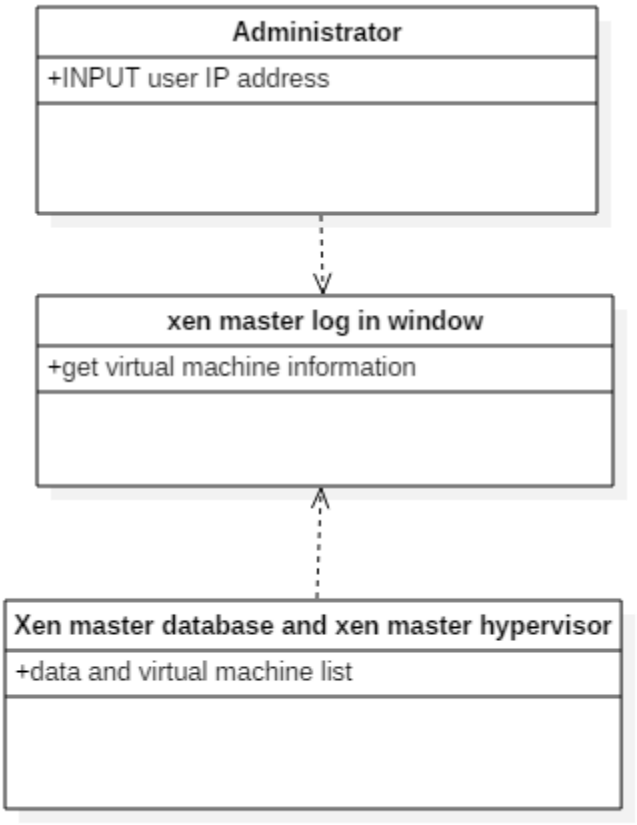



Figure 49 – Delete Virtual Machine

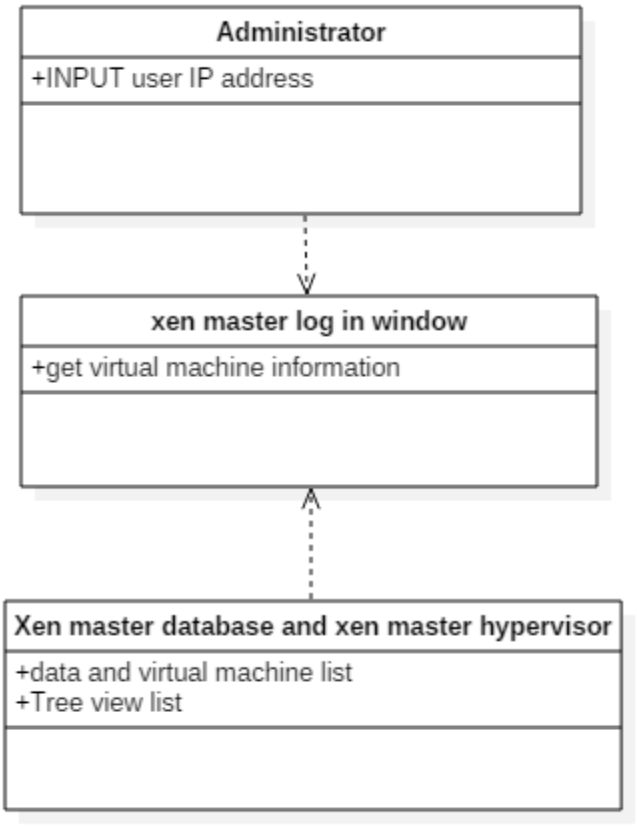



Figure 50 - Display hypervisors from the network using tree view control

***Dynamic UML Diagrams***

Figure 51 – Access Virtual Machines

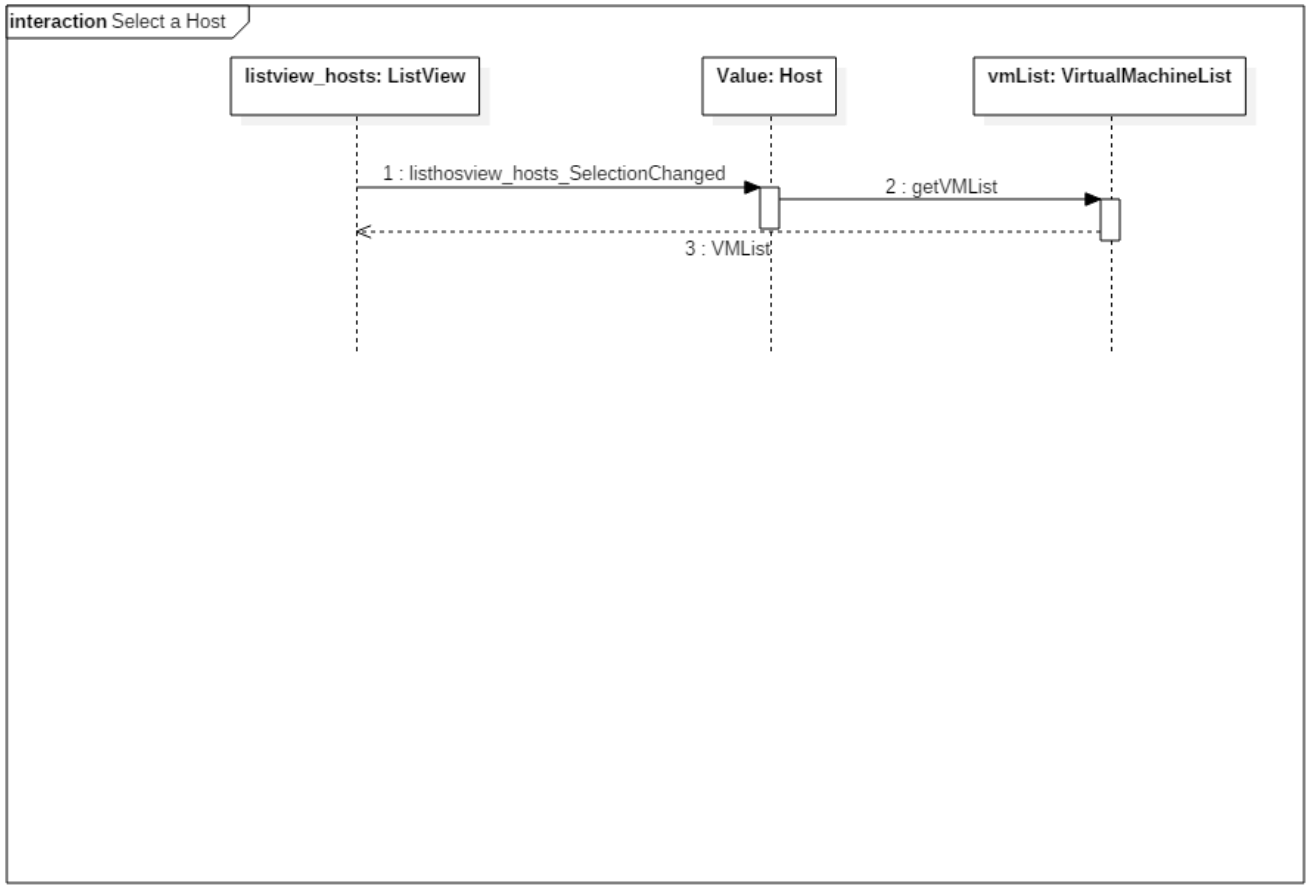



Figure 52 – View Host Virtual Machines

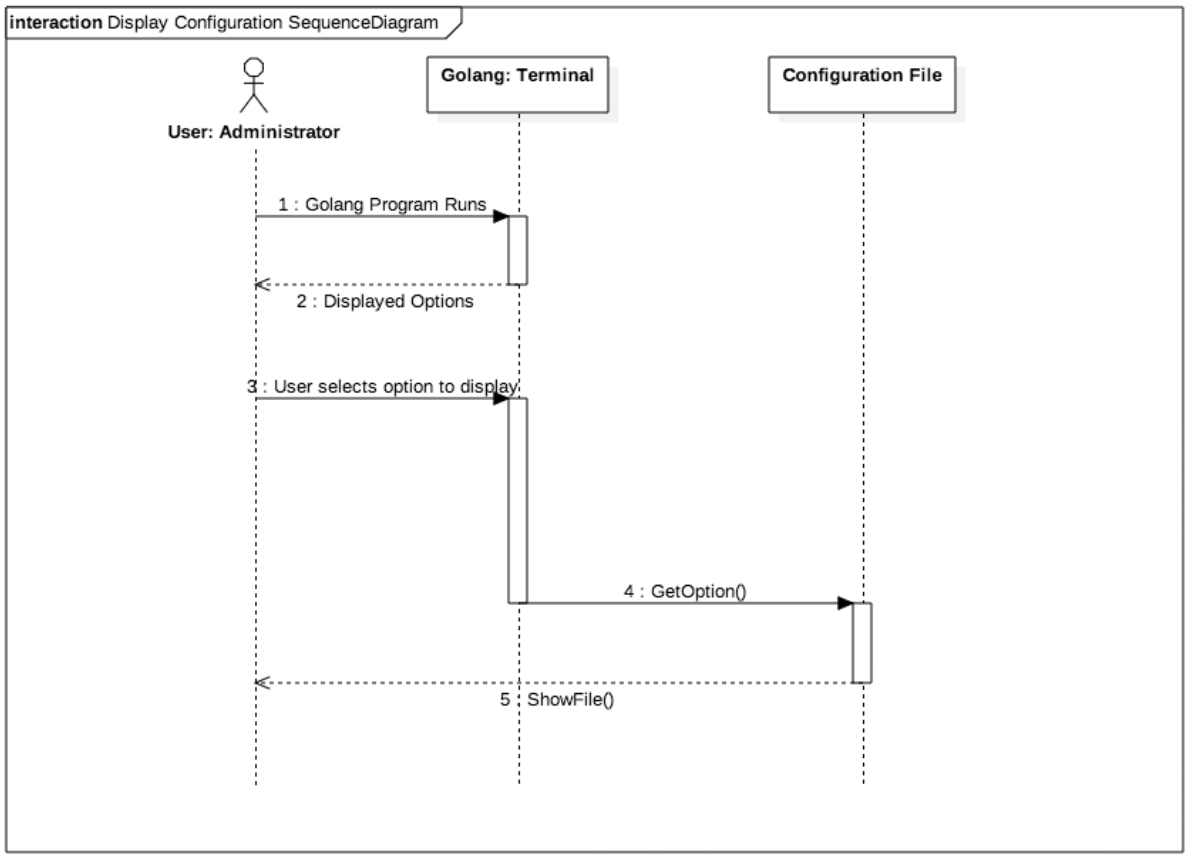



Figure 53 – View Settings of configuration File

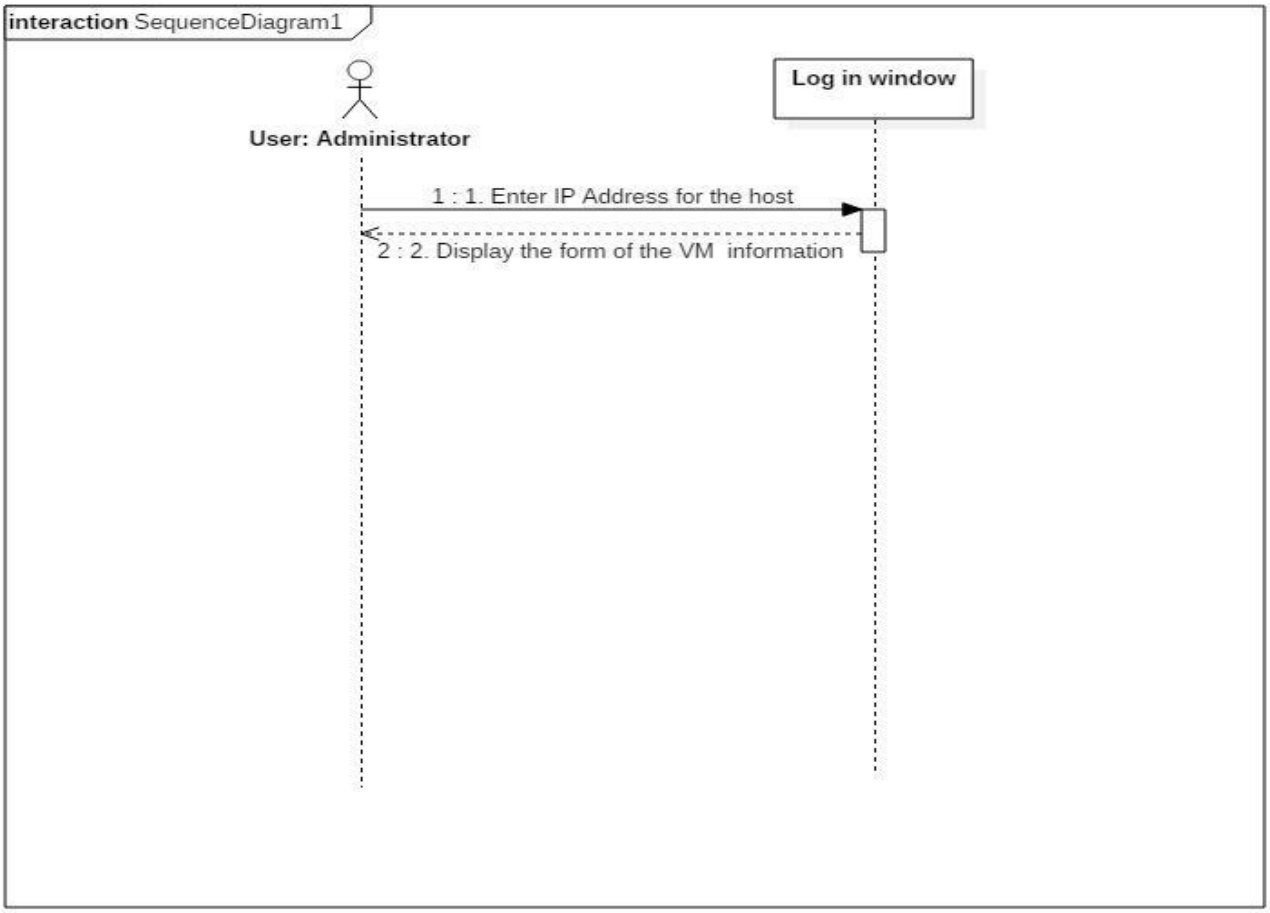



Figure 54 - Building web based dashboard for VMs

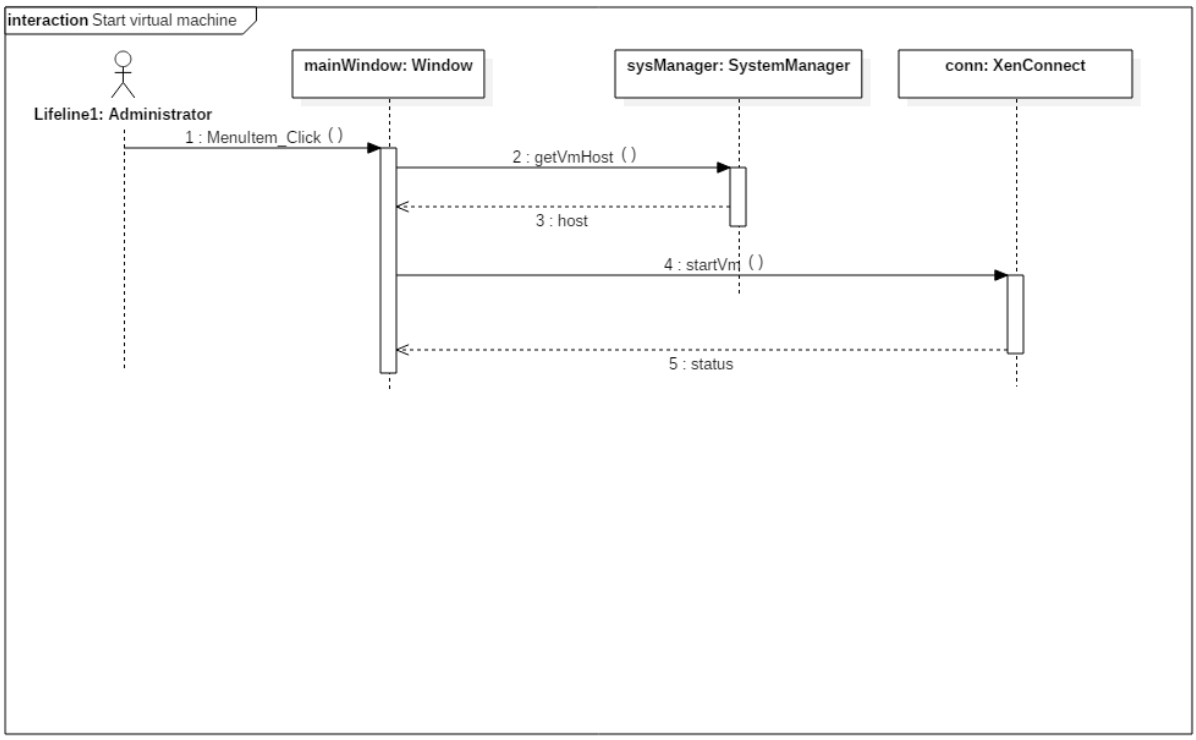



Figure 55 – Start a Virtual Machine

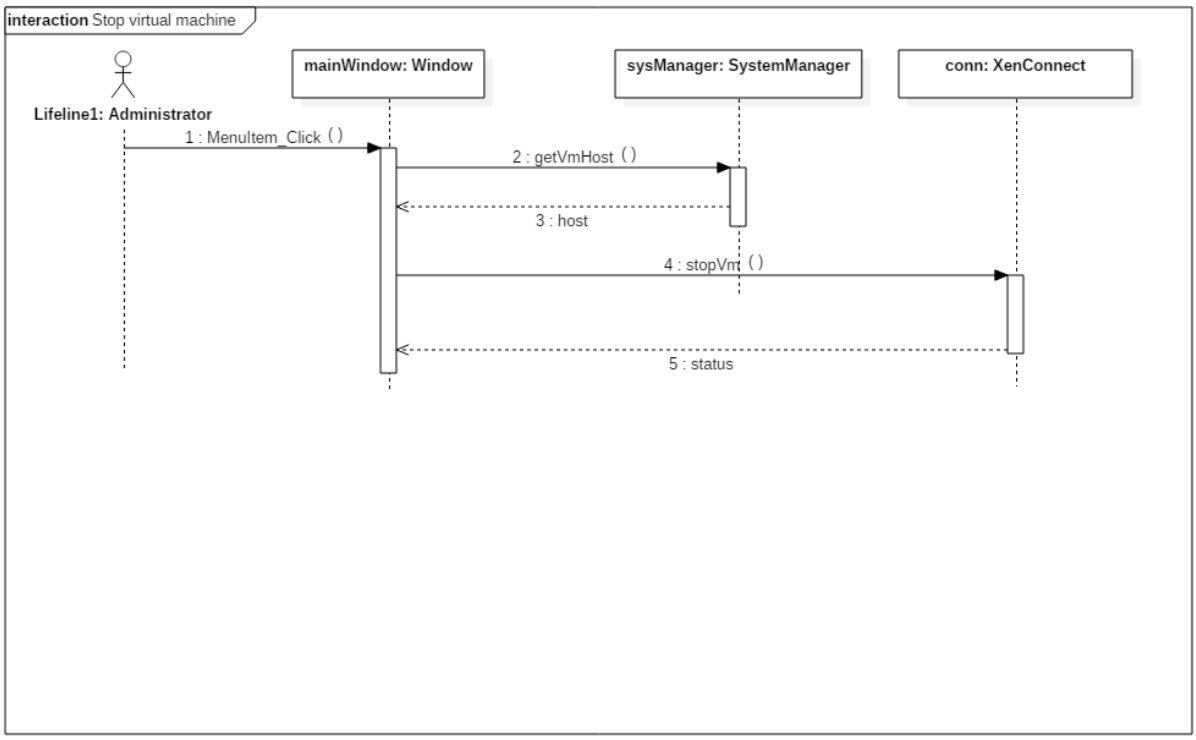



Figure 56 – Stop Virtual Machine

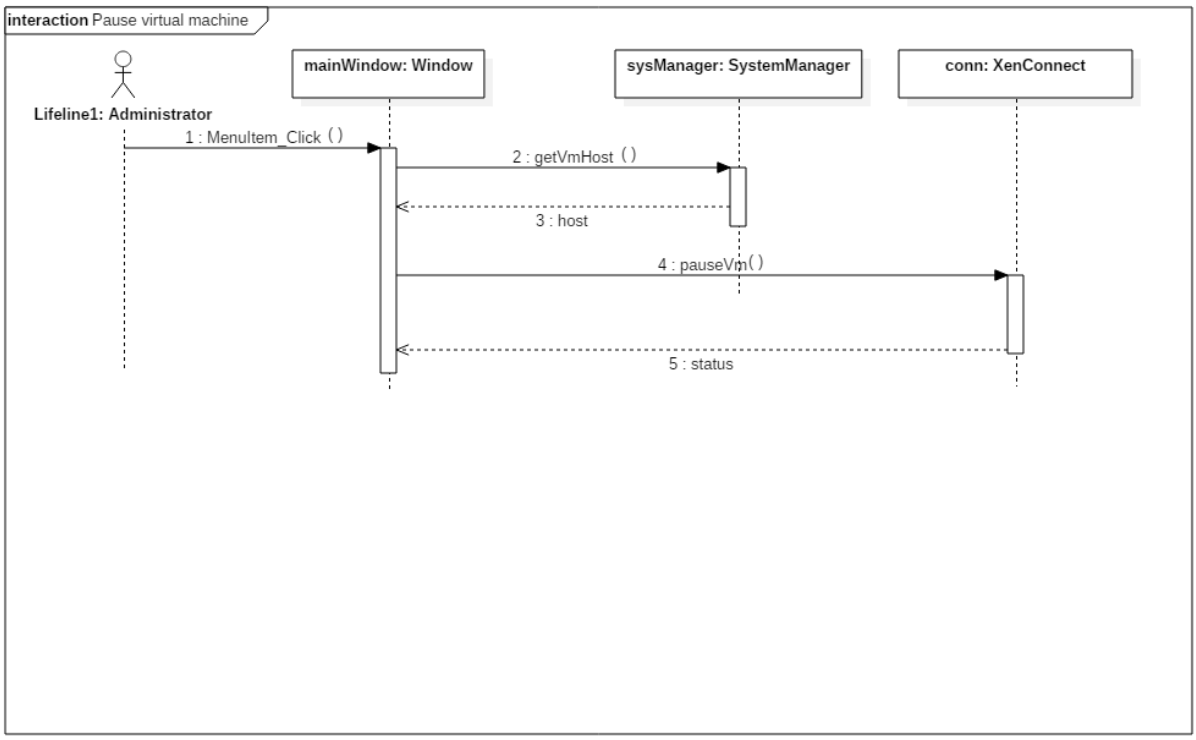



Figure 57 – Pause Virtual Machine

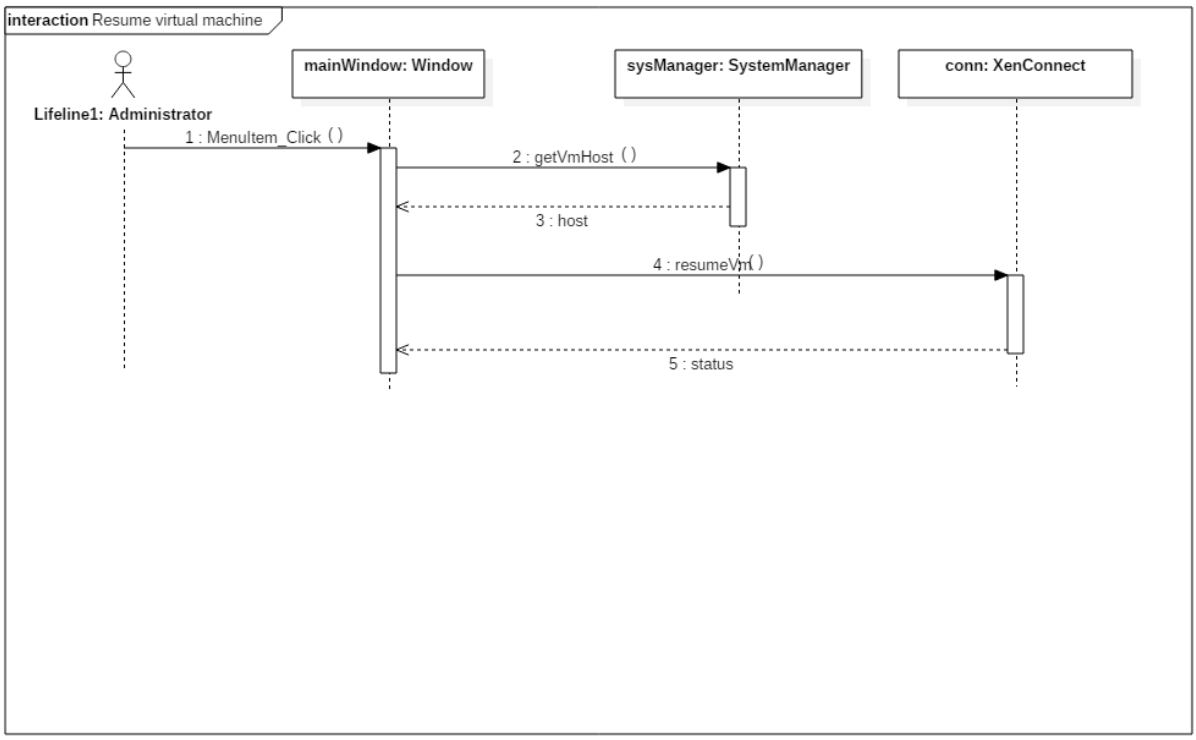



Figure 58 – Resume Virtual Machine

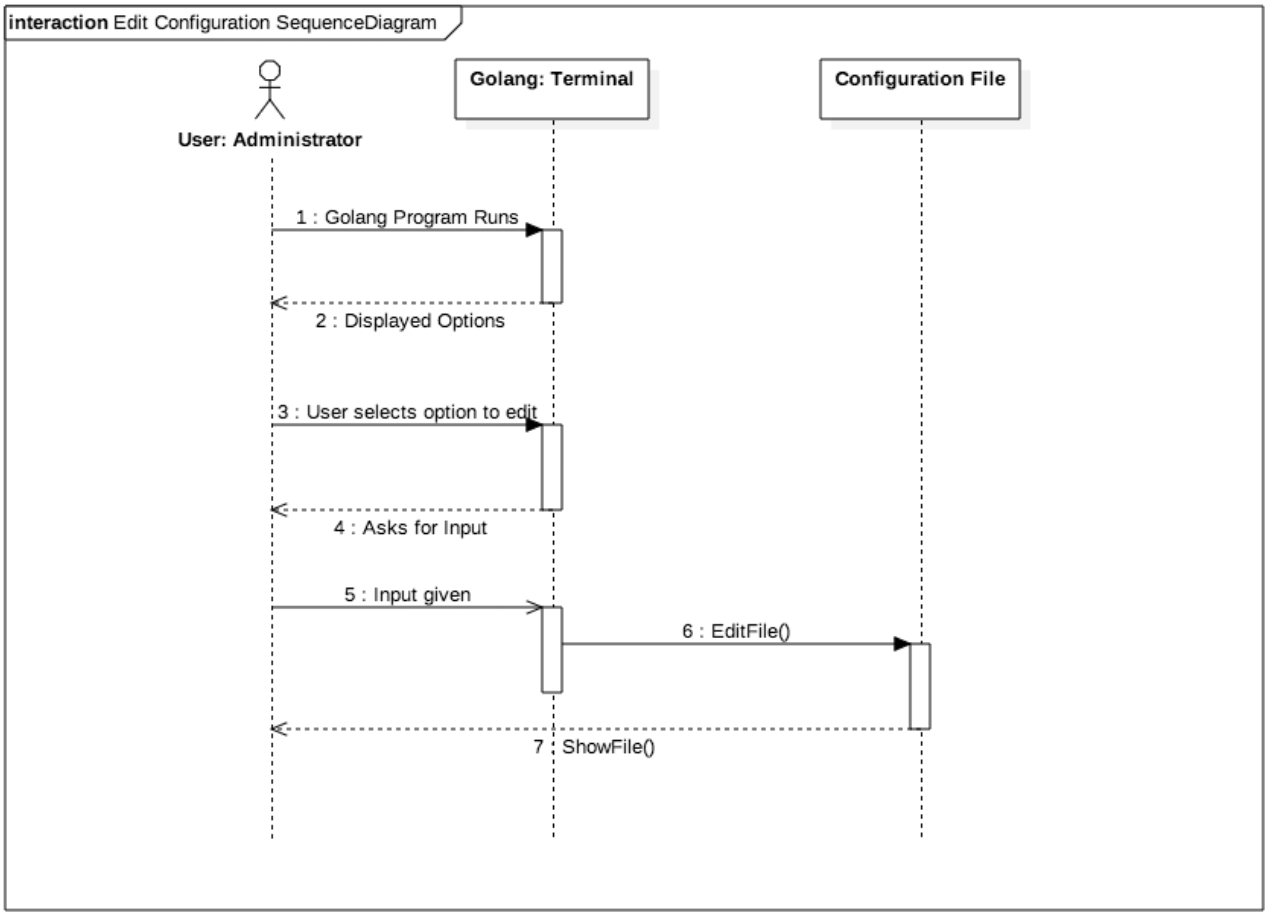



Figure 59 – Edit the settings of configuration file

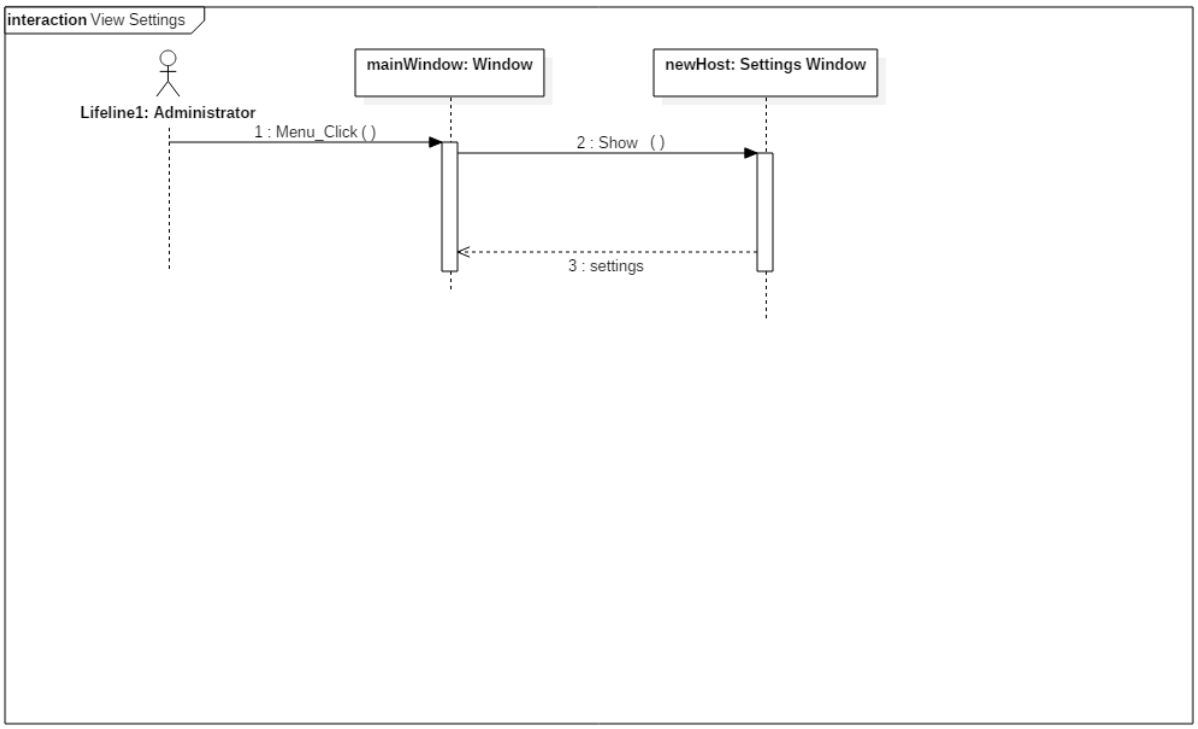



Figure 60 – Load and View VMAX Settings

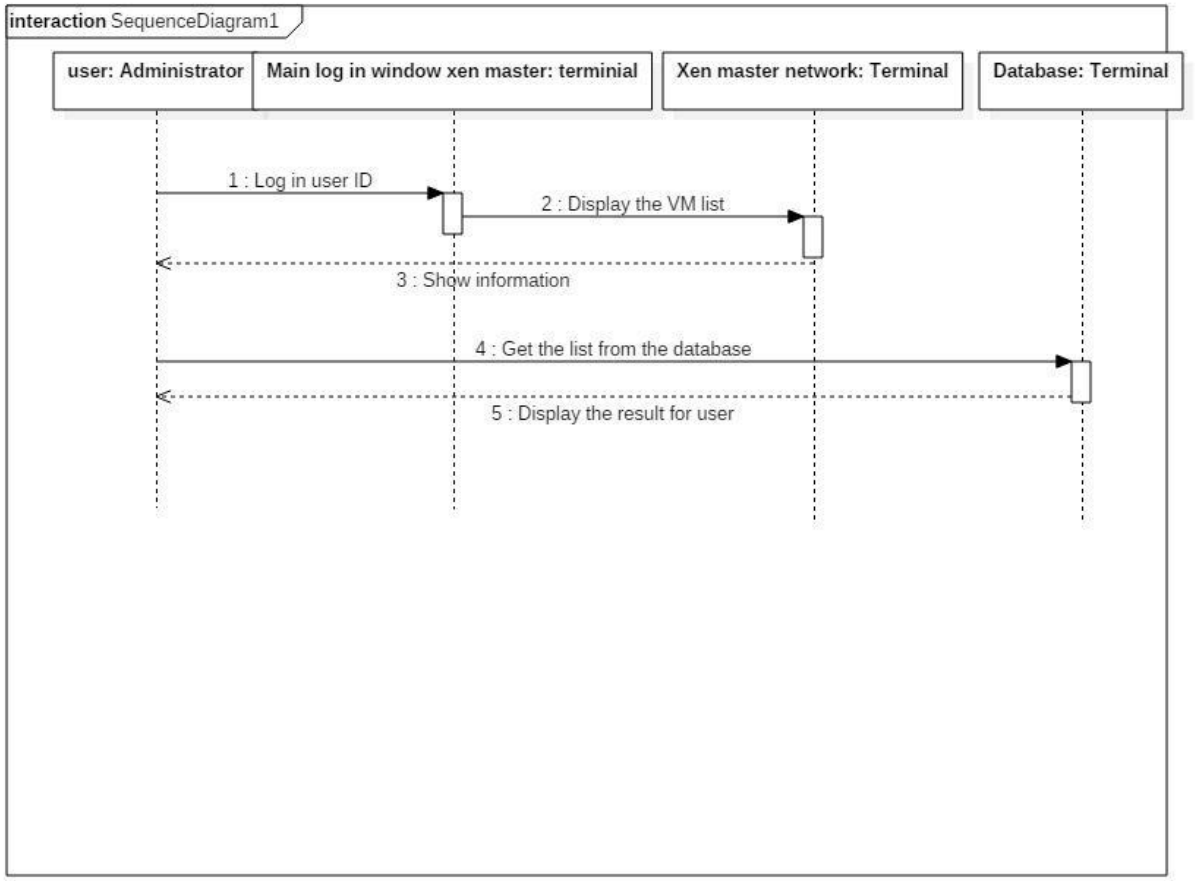



Figure 61 - Display the virtual machines on the web interface

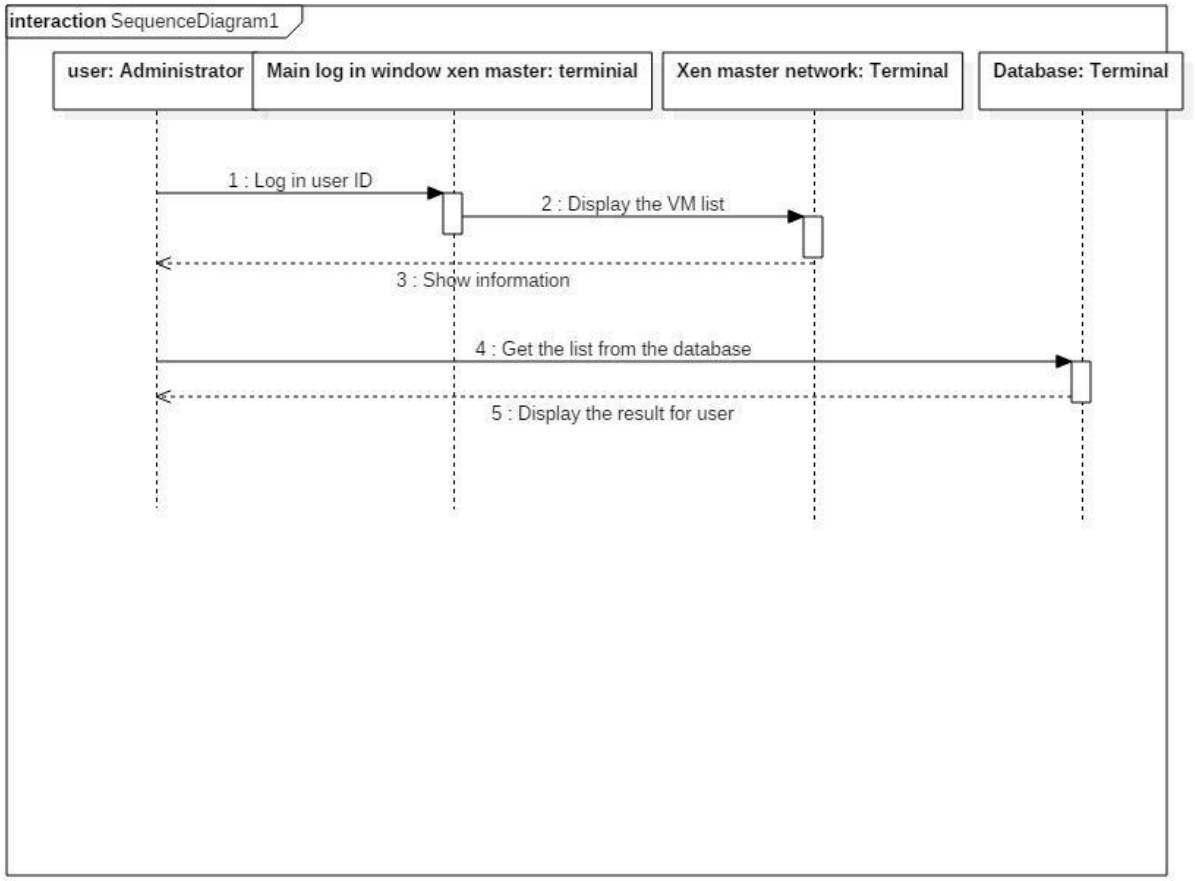



Figure 62 - Update the VMAX database

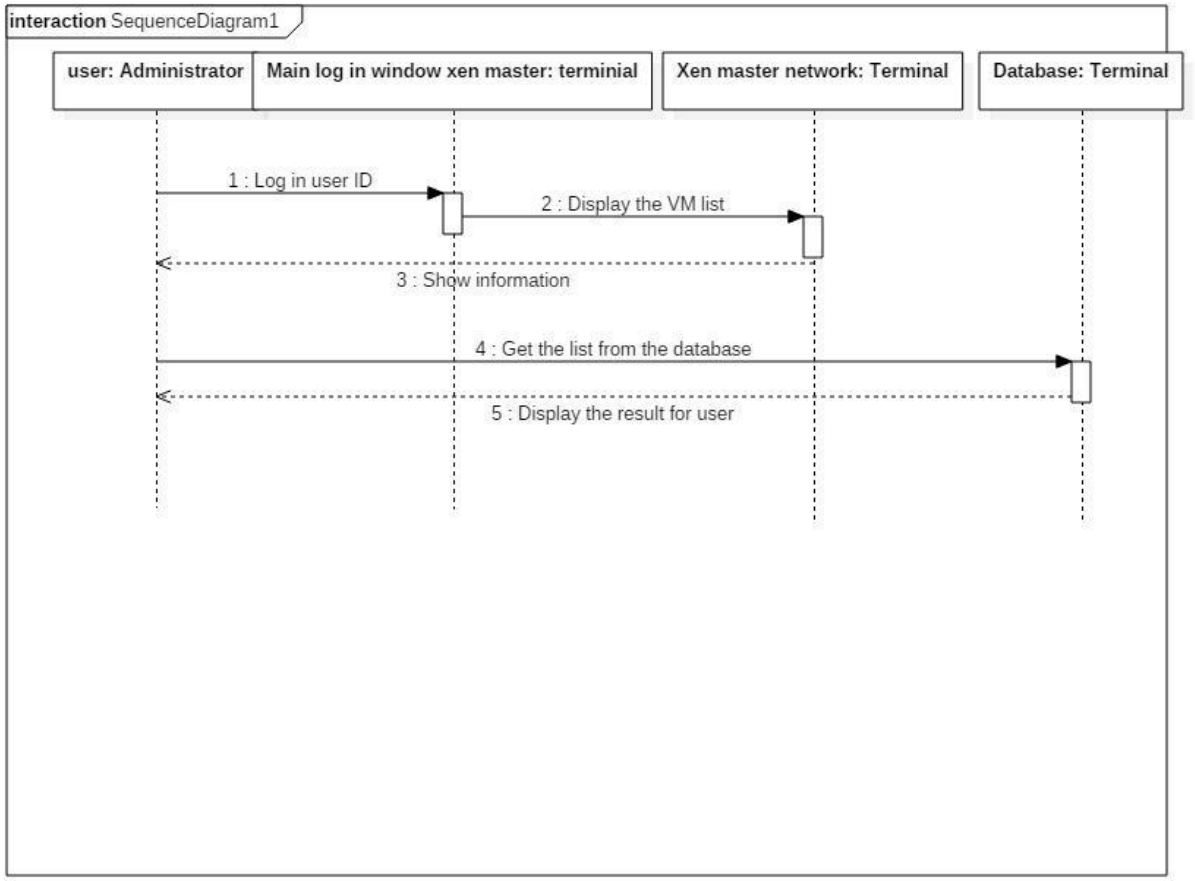



Figure 63 - Get the list of virtual machines for web dashboard

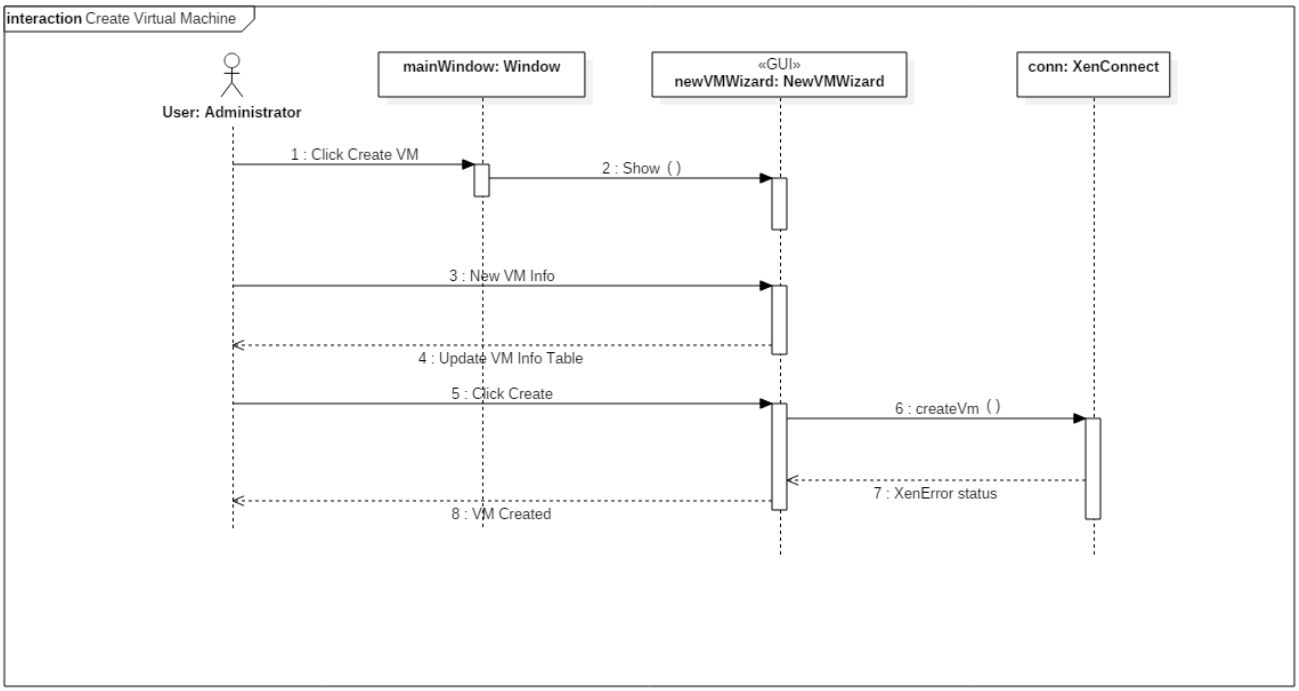



Figure 64 - Create Virtual Machine

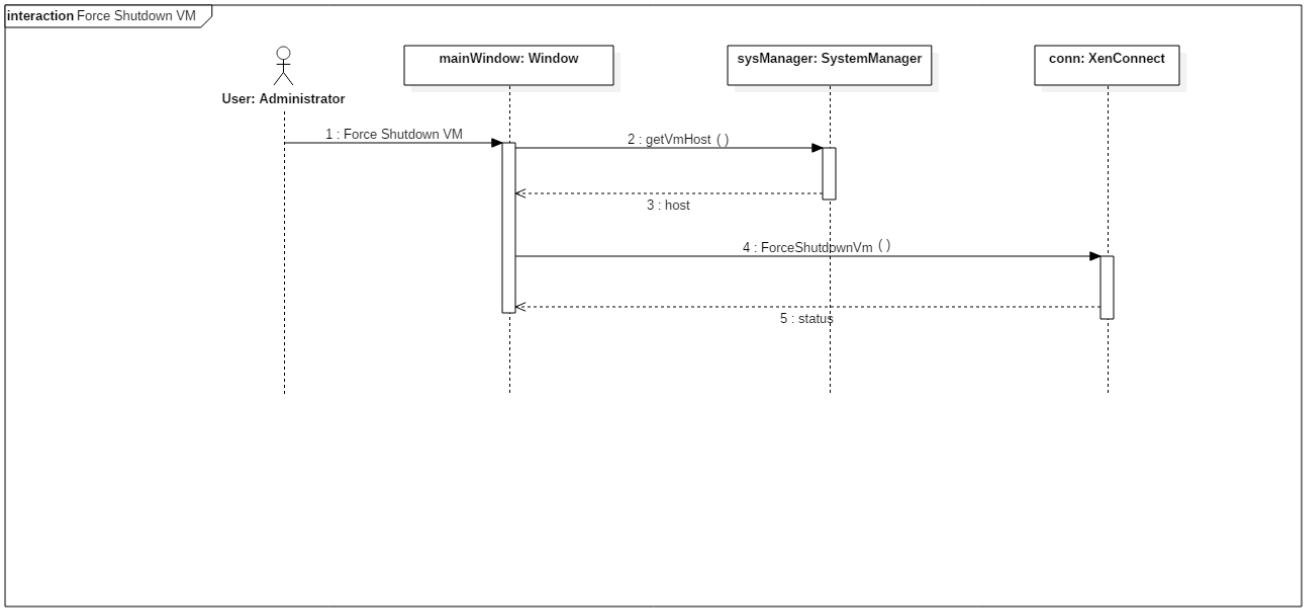



Figure 65 - Force Shutdown Virtual Machine

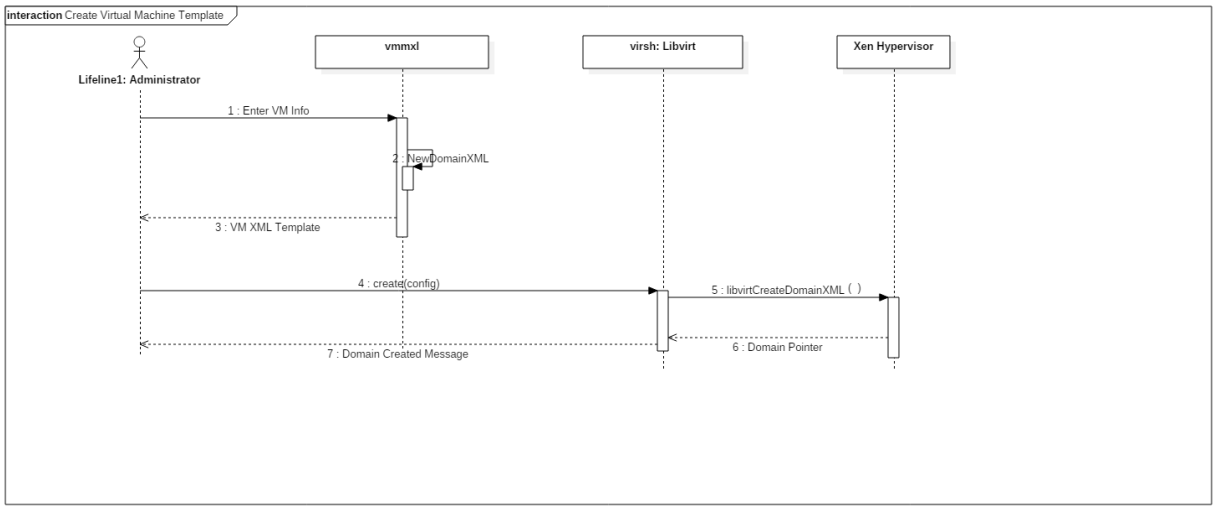


Figure 66 - Create Virtual Machine Templates

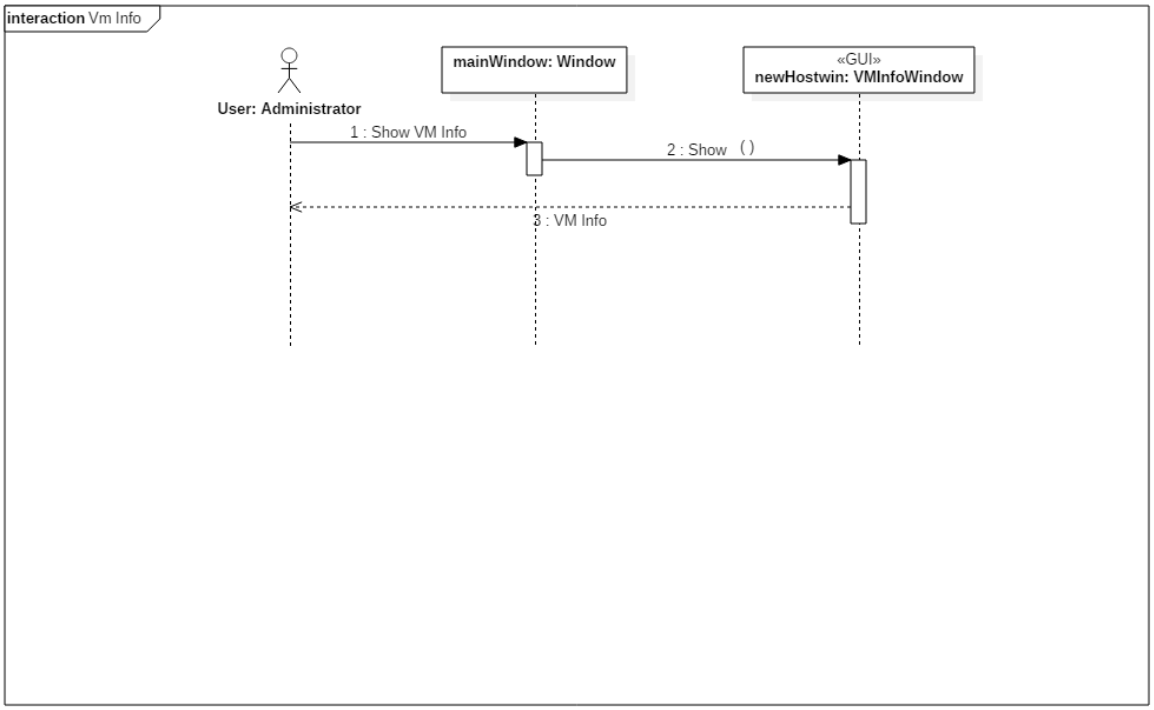



Figure 67 - Display Virtual Machine Information

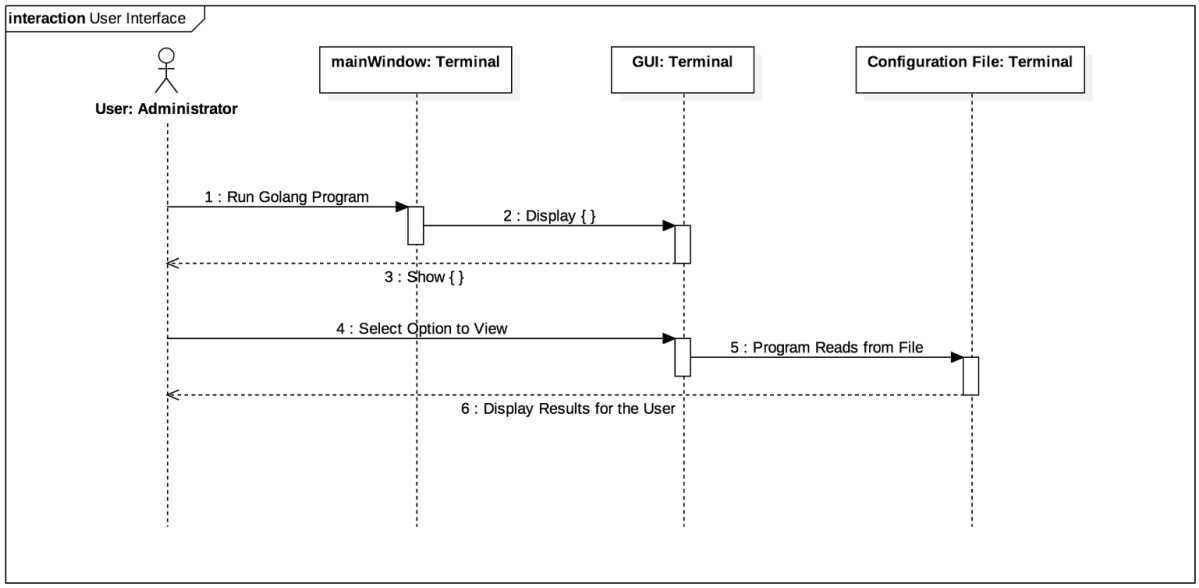



Figure 68 - Create a User Interface for the configuration file editing

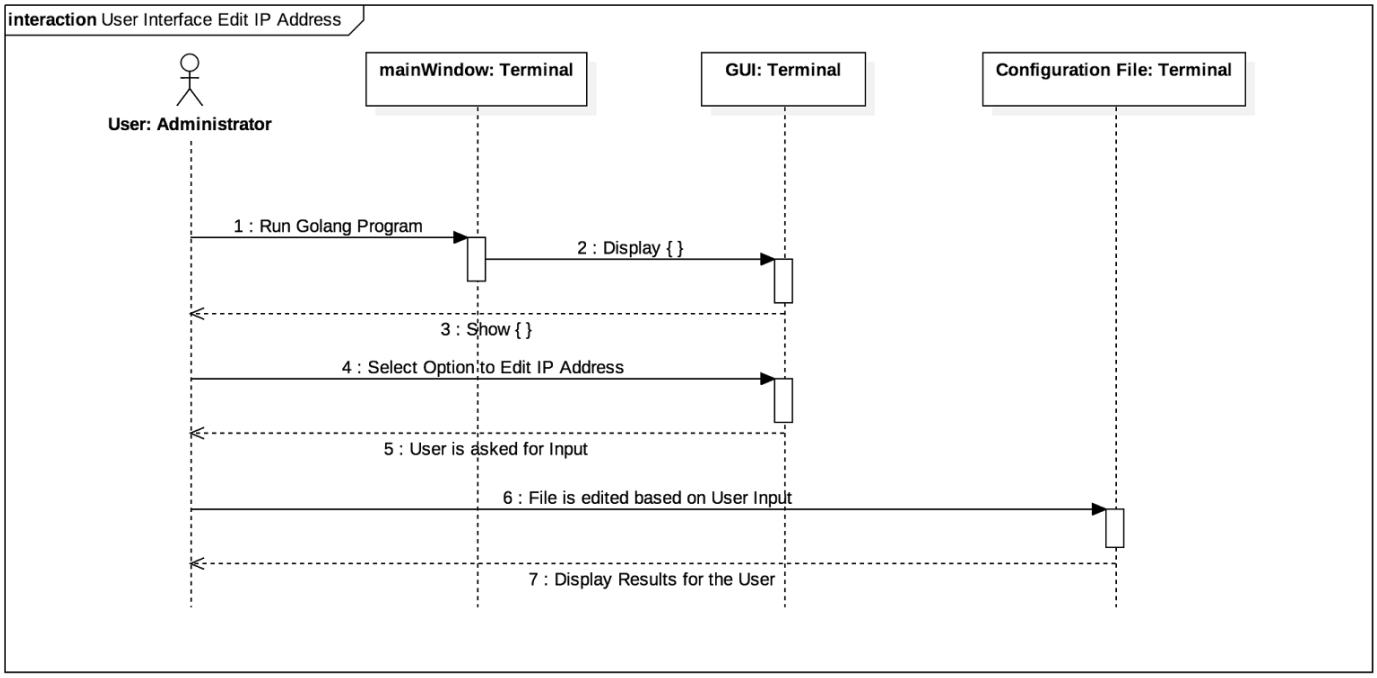



Figure 69 - Use interface to edit host ip address

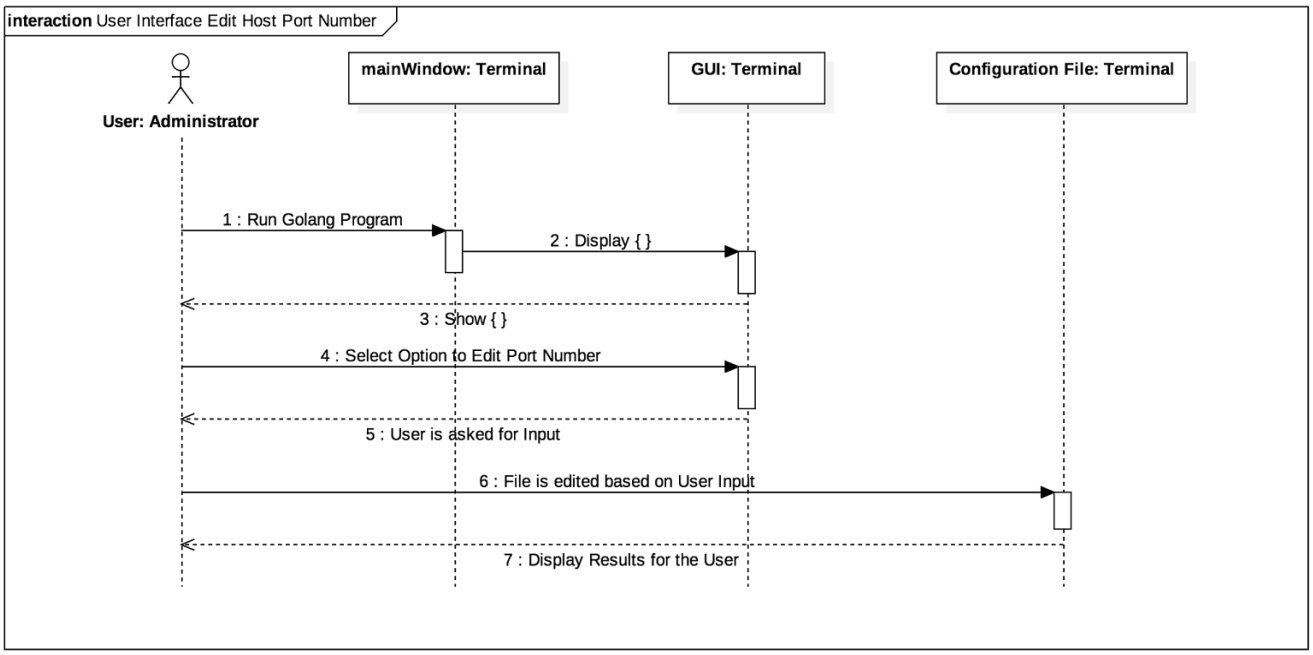



Figure 70 - Use interface to edit port number

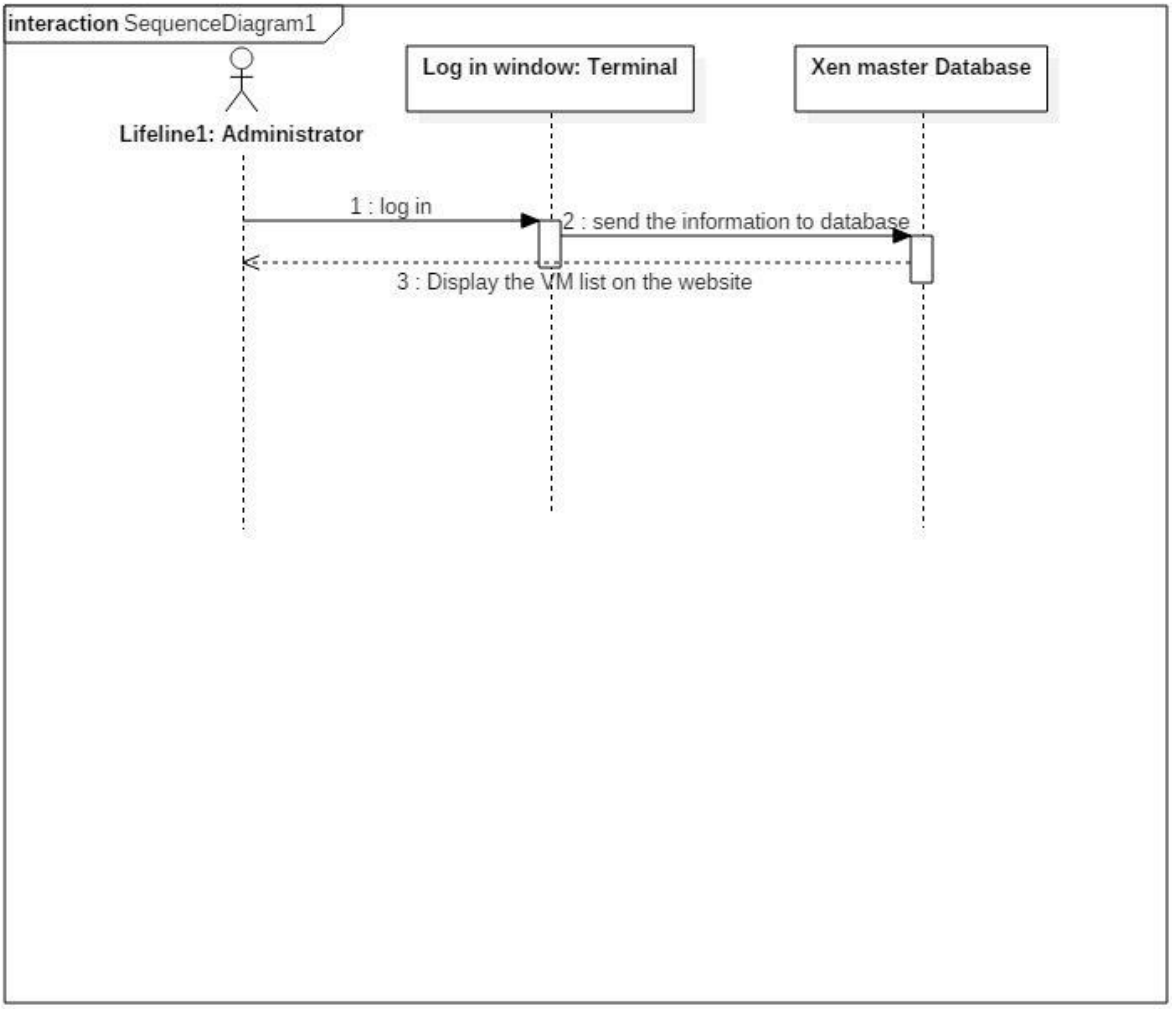



Figure 71 - Modify the code for the .net format of the web

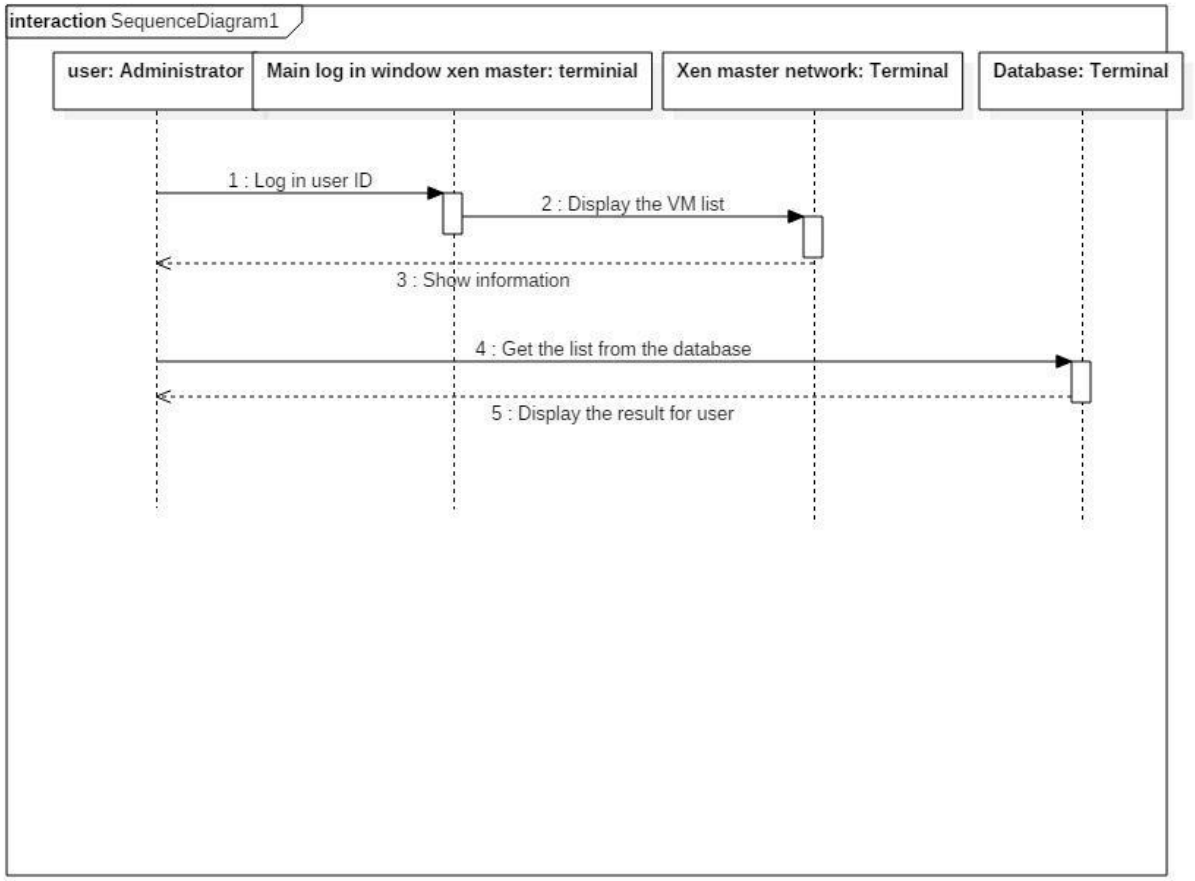



Figure 72 - Add stop functionality

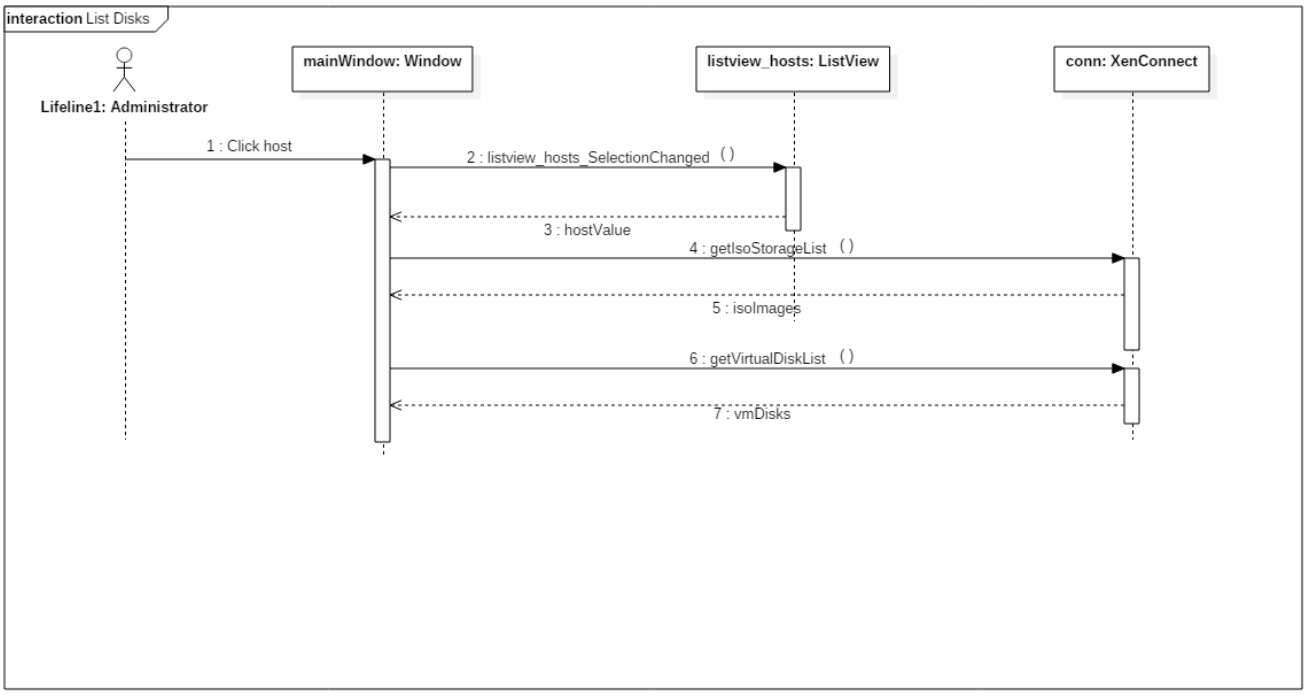



Figure 73 - List Virtual Machine Disk Configurations

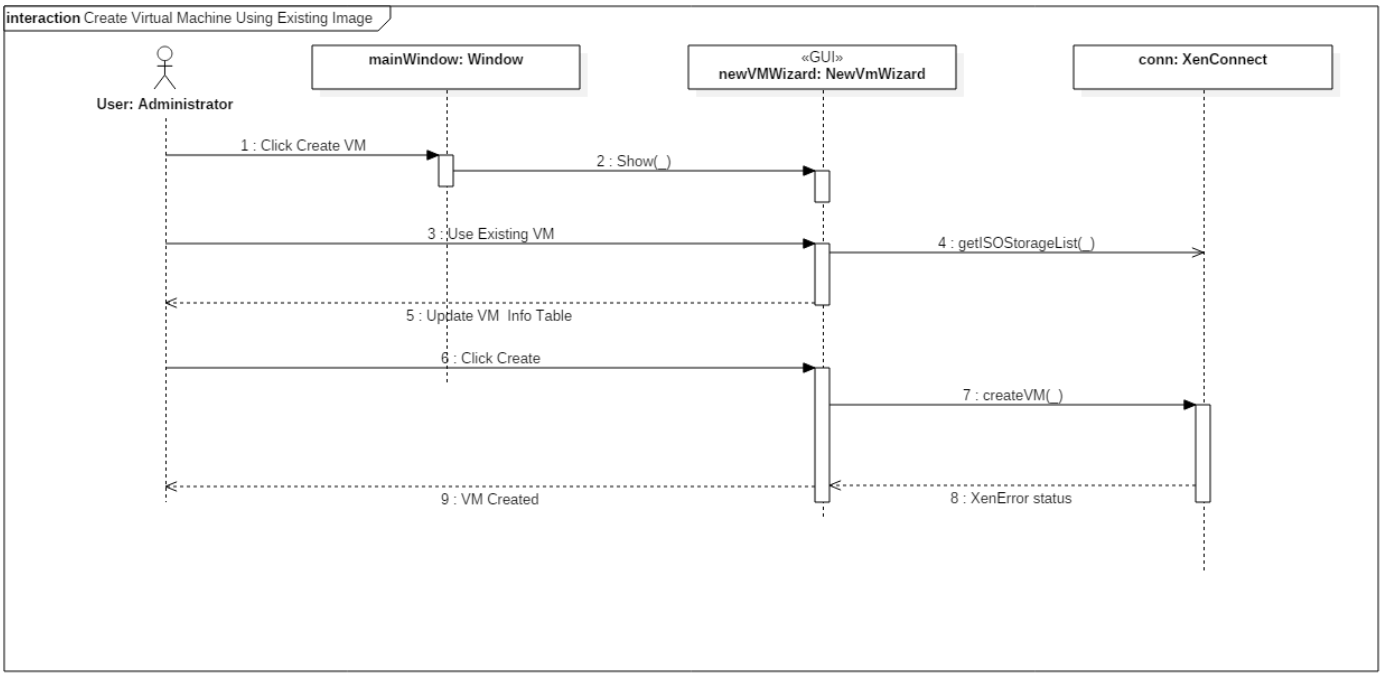



Figure 74 - Create Virtual Machine From Existing Disk Image

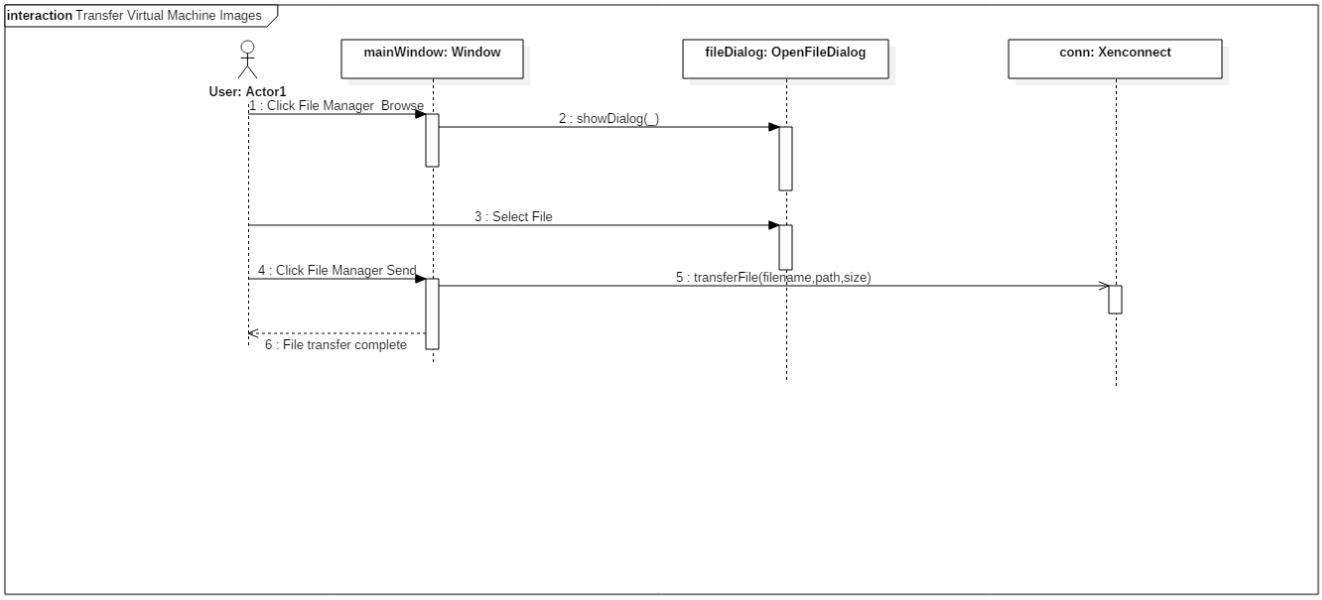



Figure 75 - Transfer Virtual Machine Images

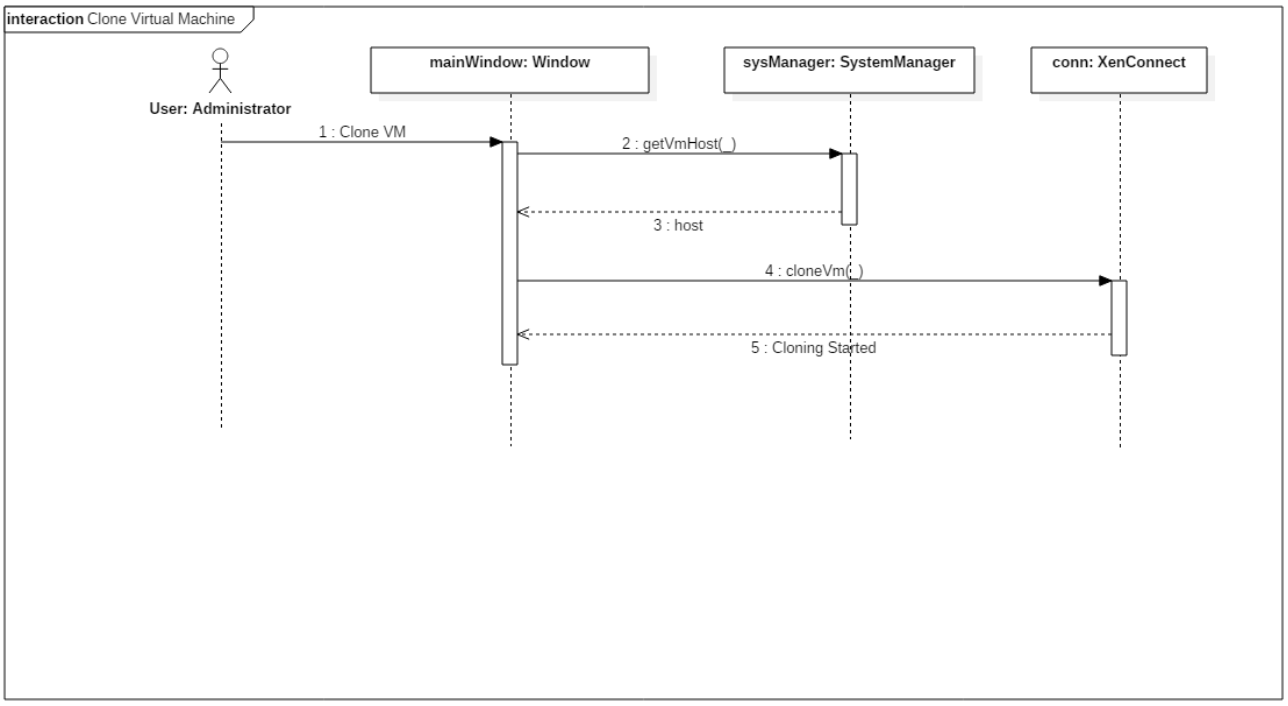



Figure 76 - Clone Virtual Machine

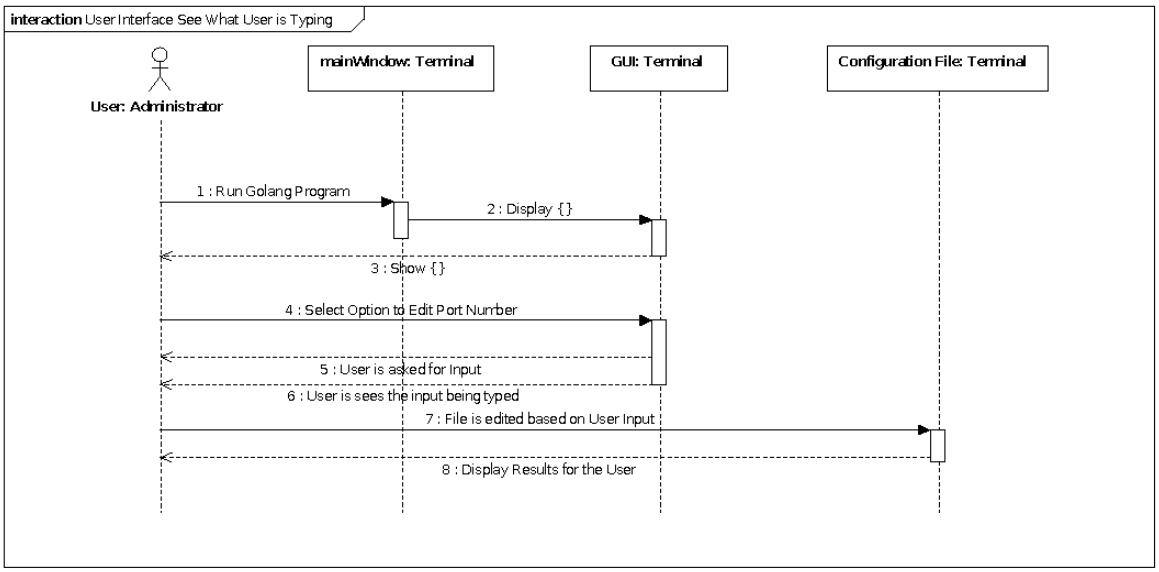



Figure 77 - Display what is typed by the user into the interface

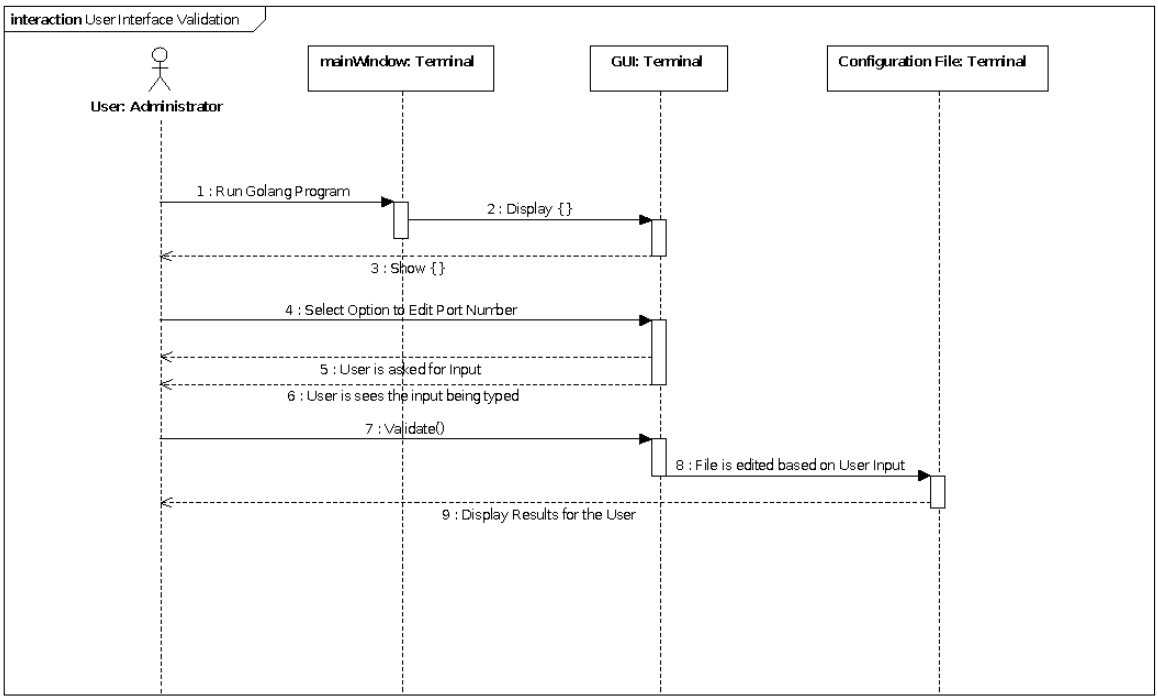



Figure 78 - Validate input values for both IP address and Port

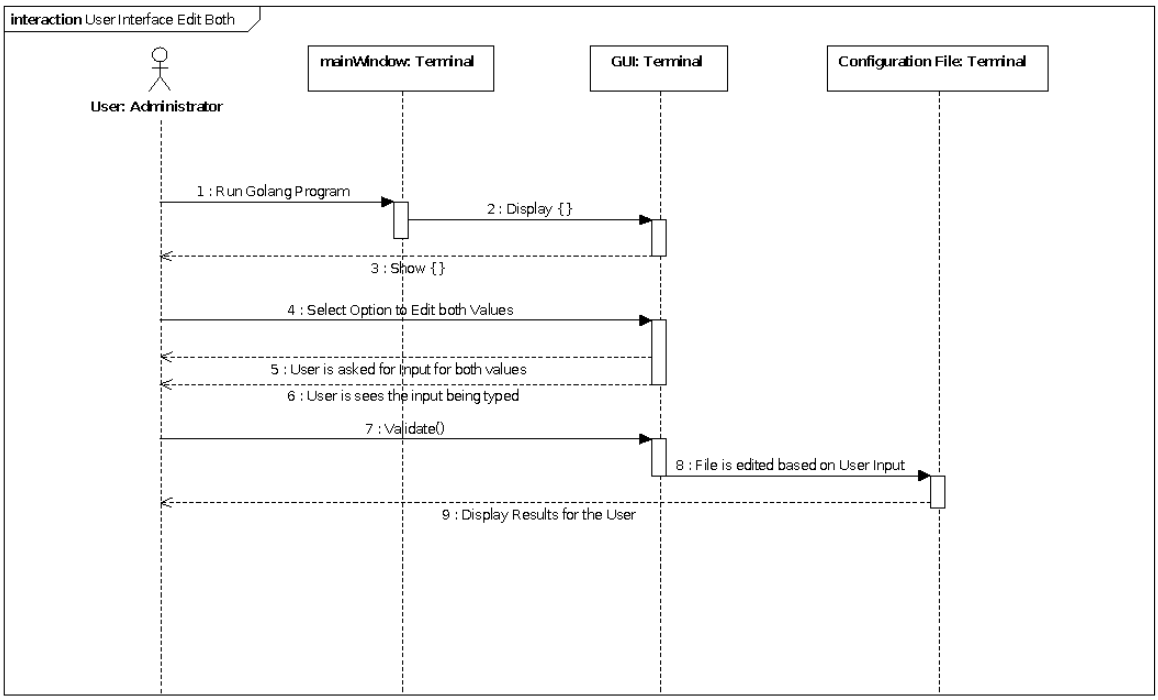



Figure 79 - Enter all the values to UI and update through one submission

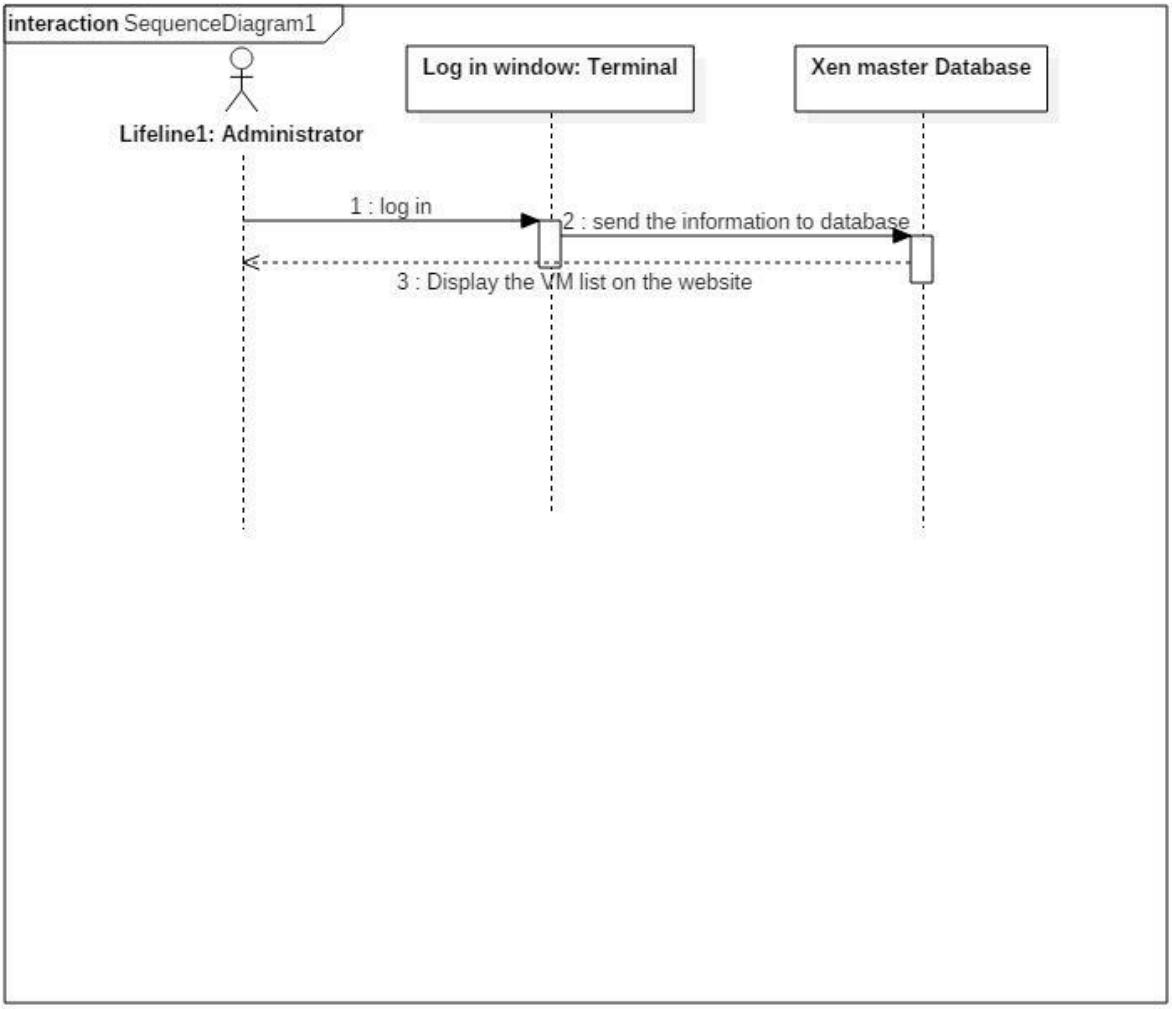



Figure 80 - Clean up user interface

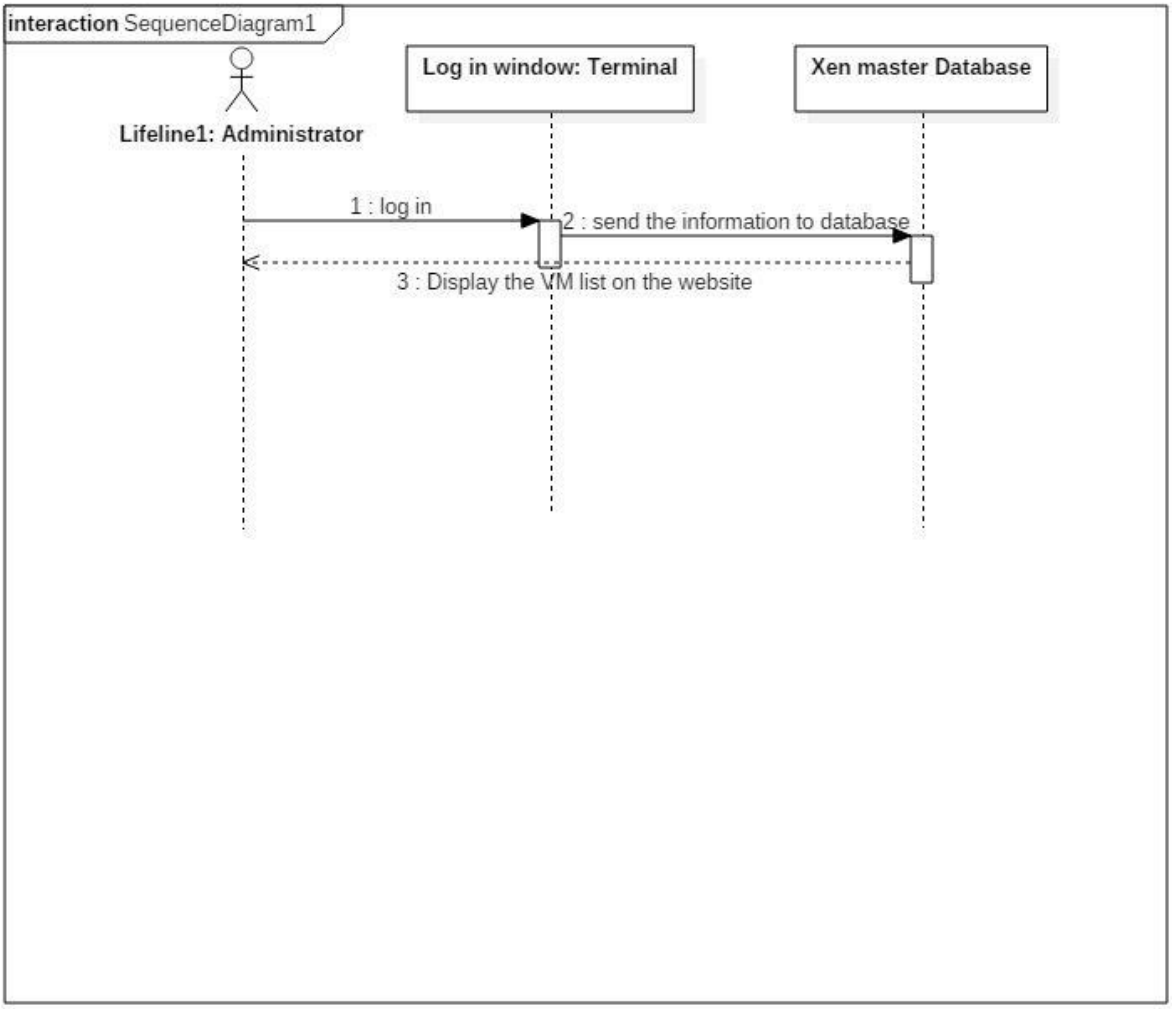



Figure 81 - Implement “Create” Virtual Machine

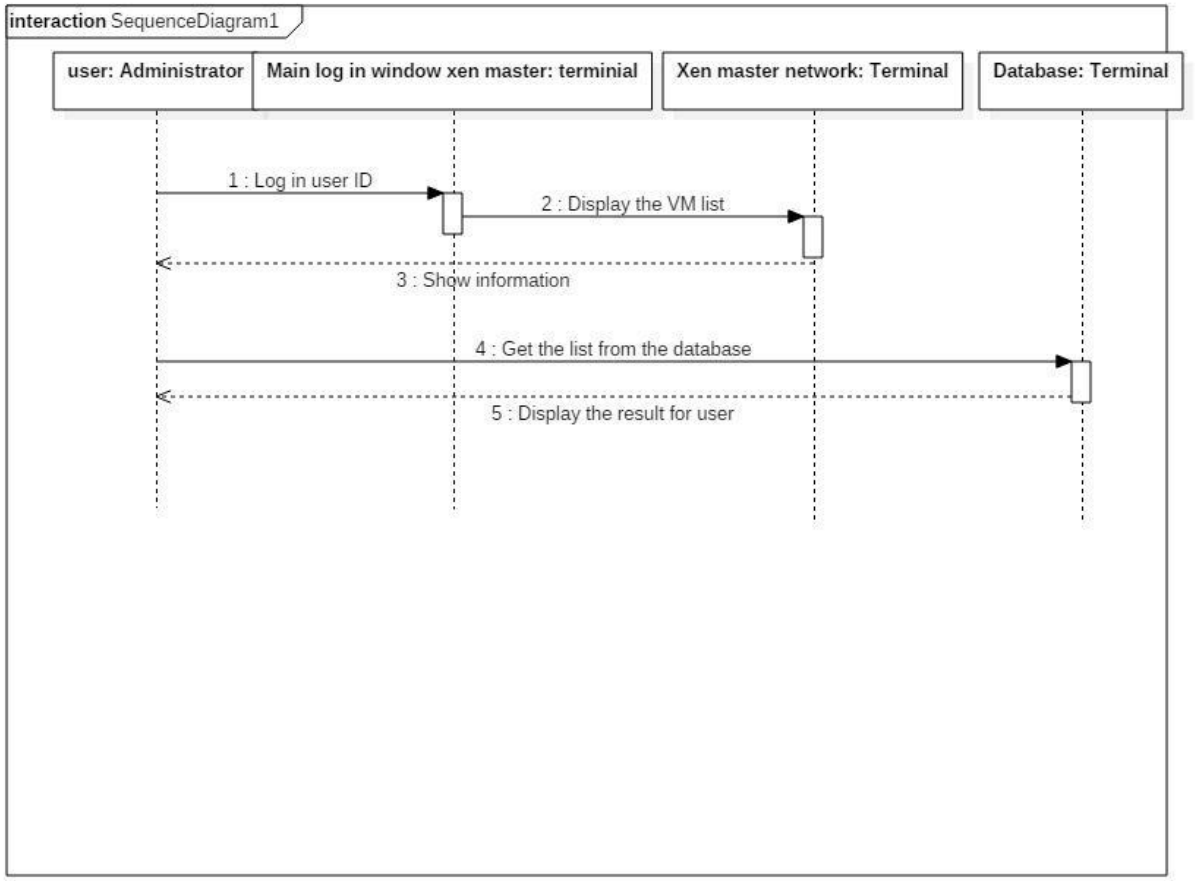



Figure 82 - Display status of virtual machines

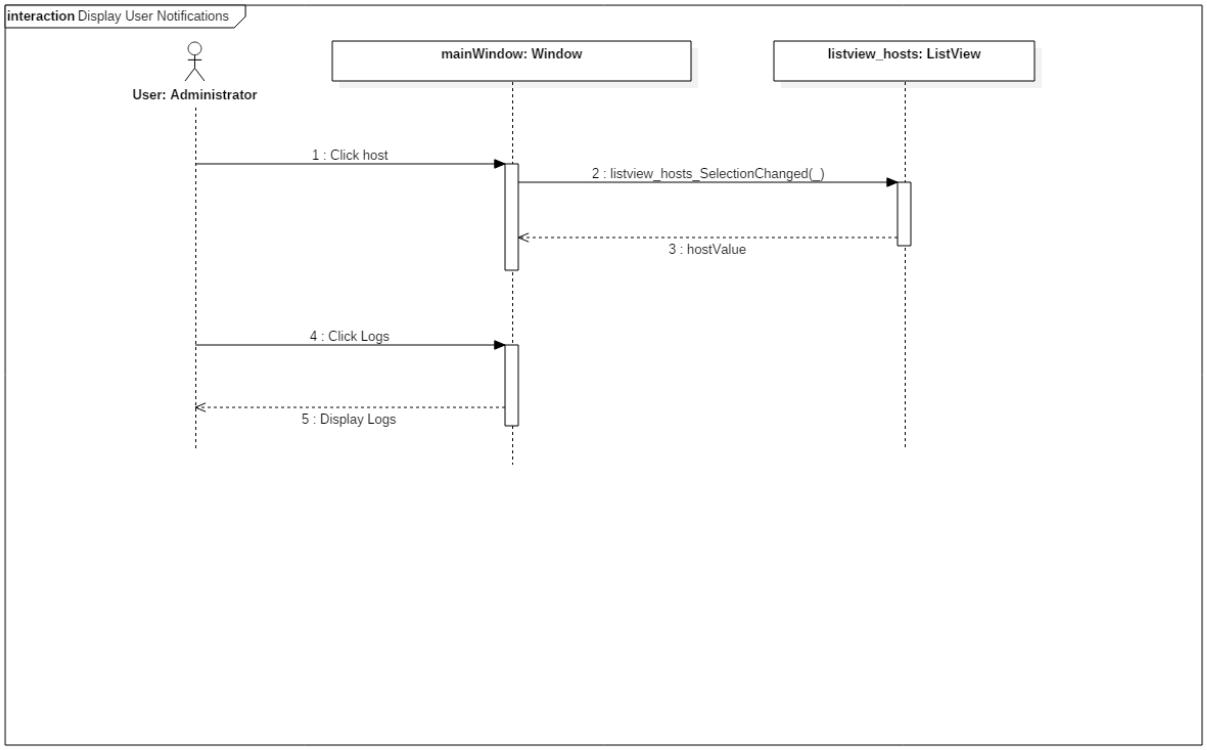



Figure 83 - Display User Notifications

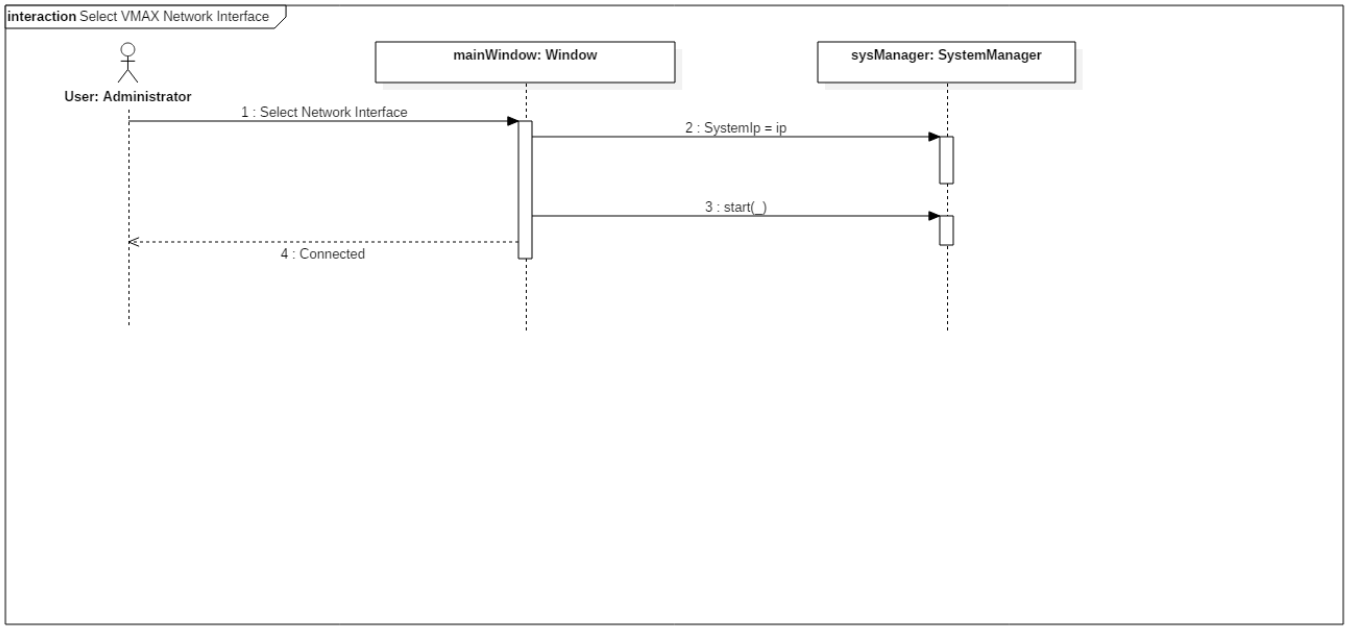



Figure 84 - Select VMAX Network Interface

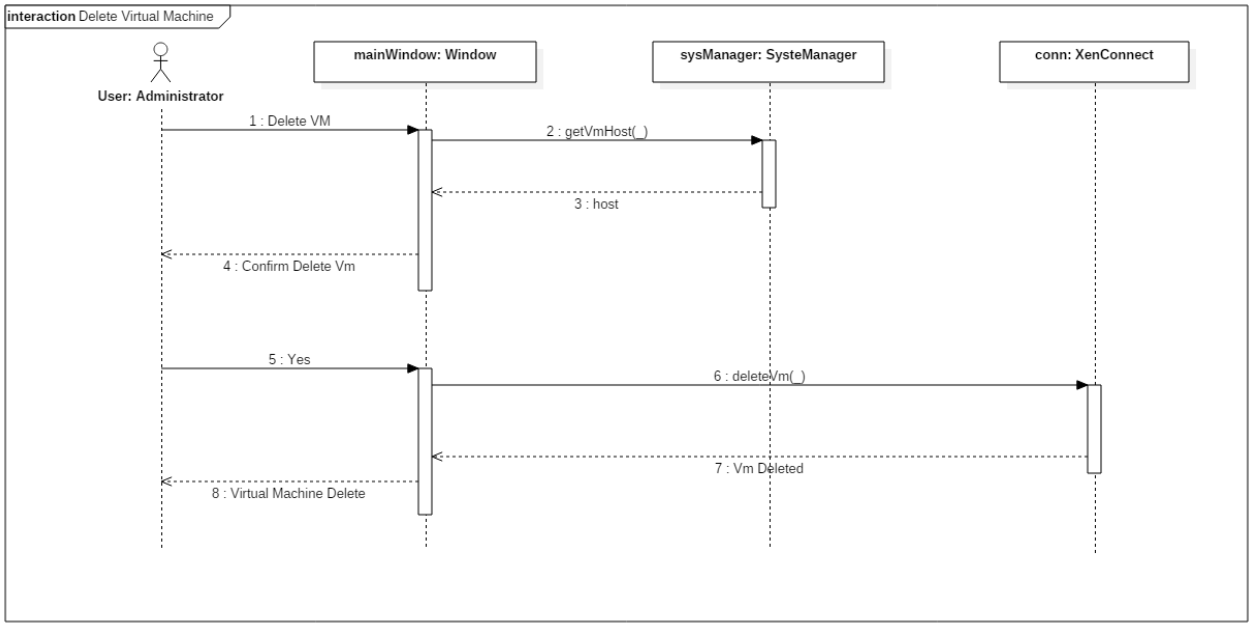



Figure 85 - Delete Virtual Machine

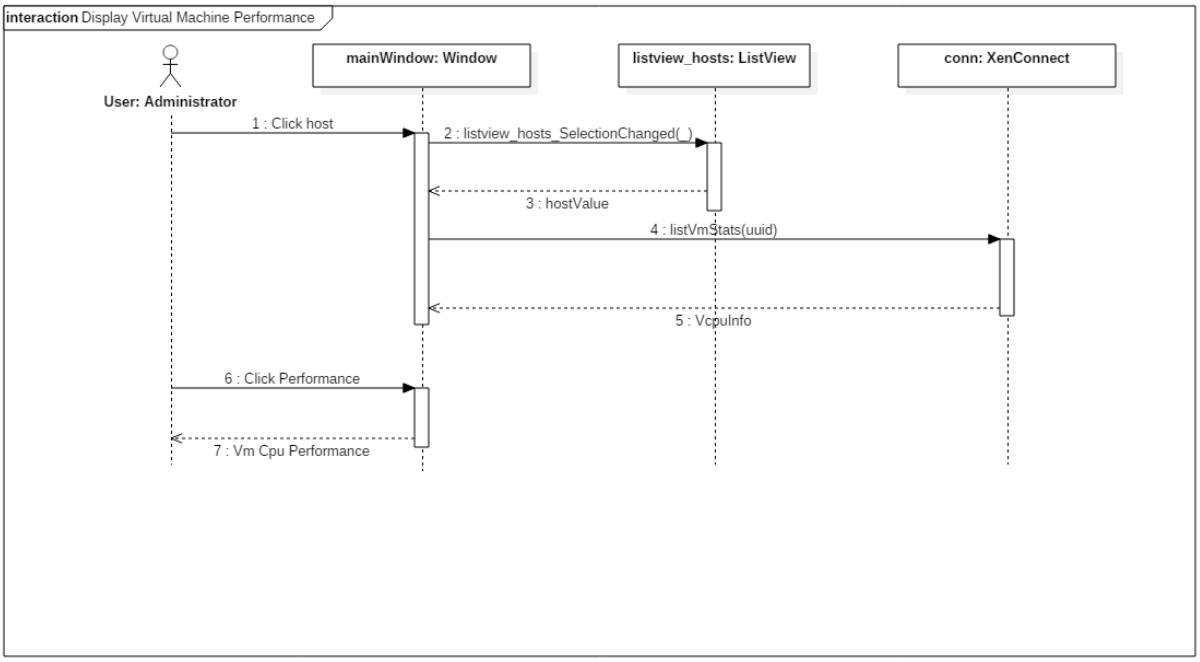



Figure 86 - Display Performance Parameters

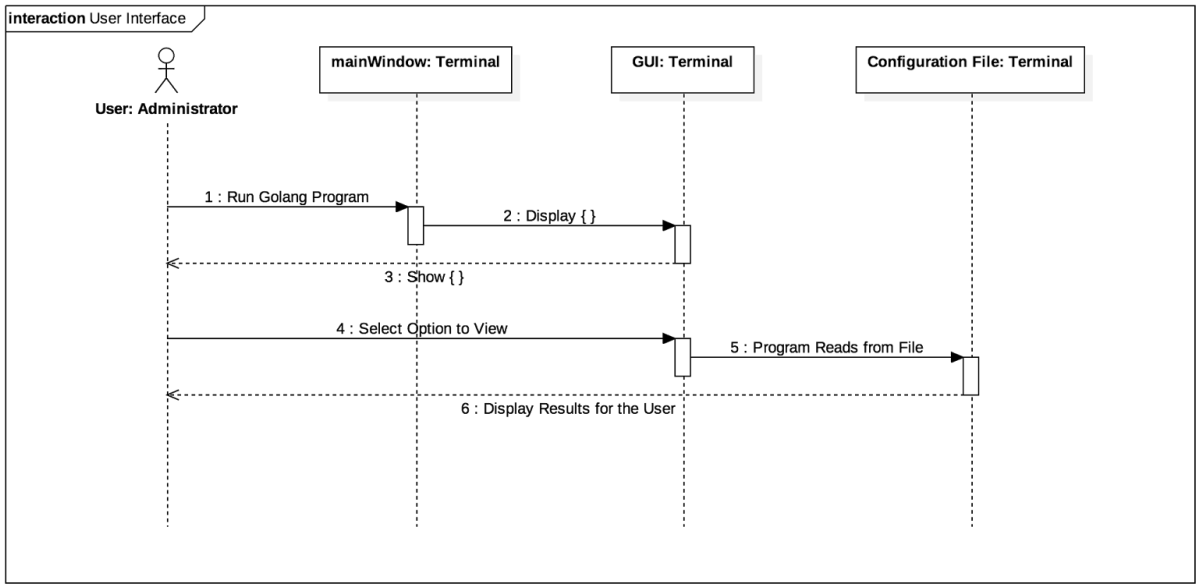



Figure 87 - Clean up the interface

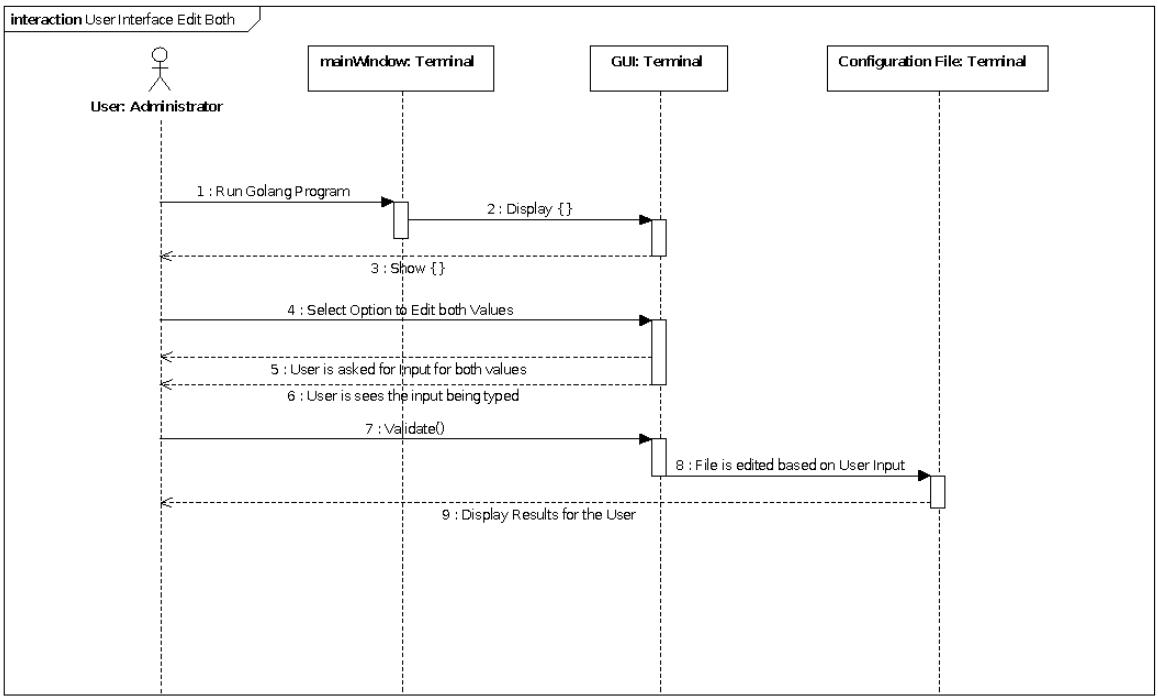



Figure 88 - Update all parameters with single selection

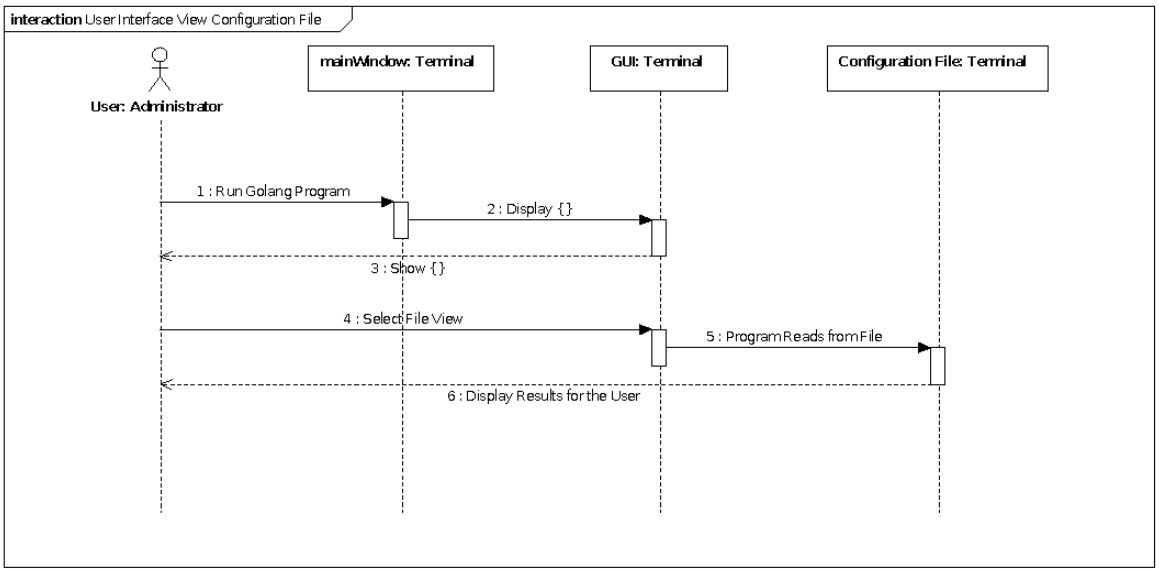



Figure 89 - View file should be a separate button/selection

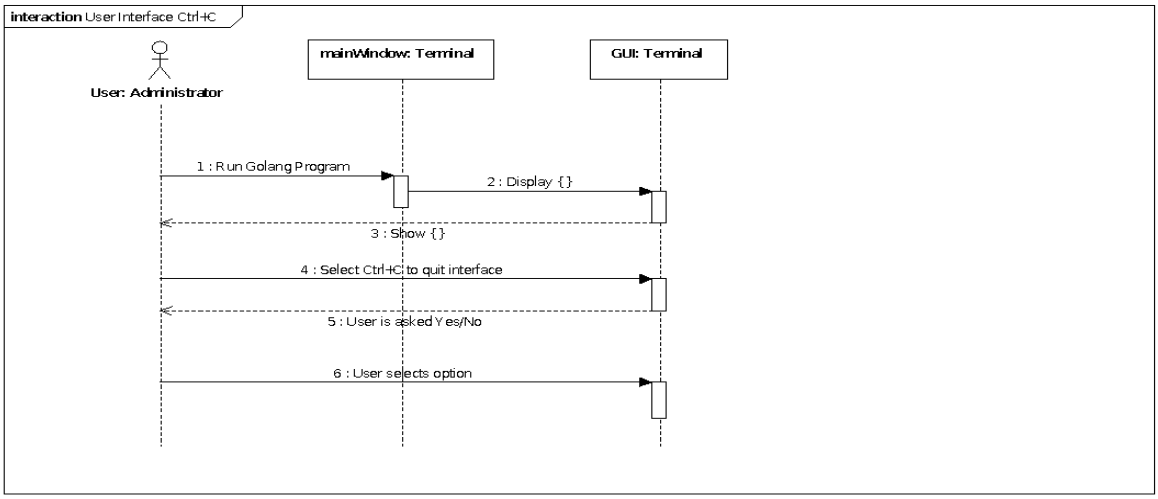



Figure 90 - Add a signal handler to catch the quit option

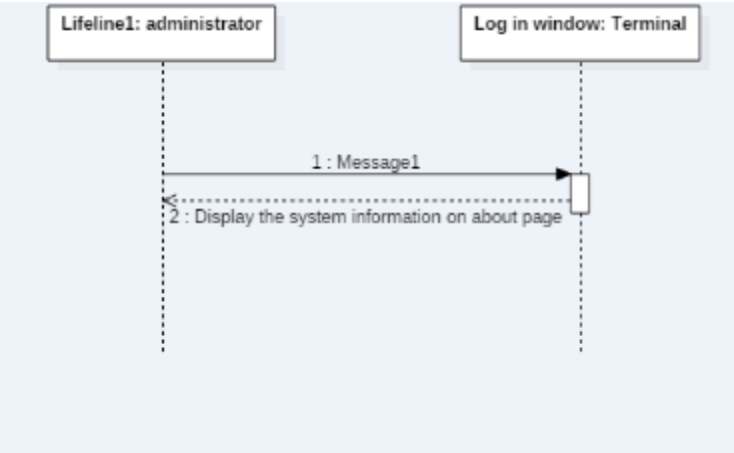



Figure 91 - Add more information on the home page

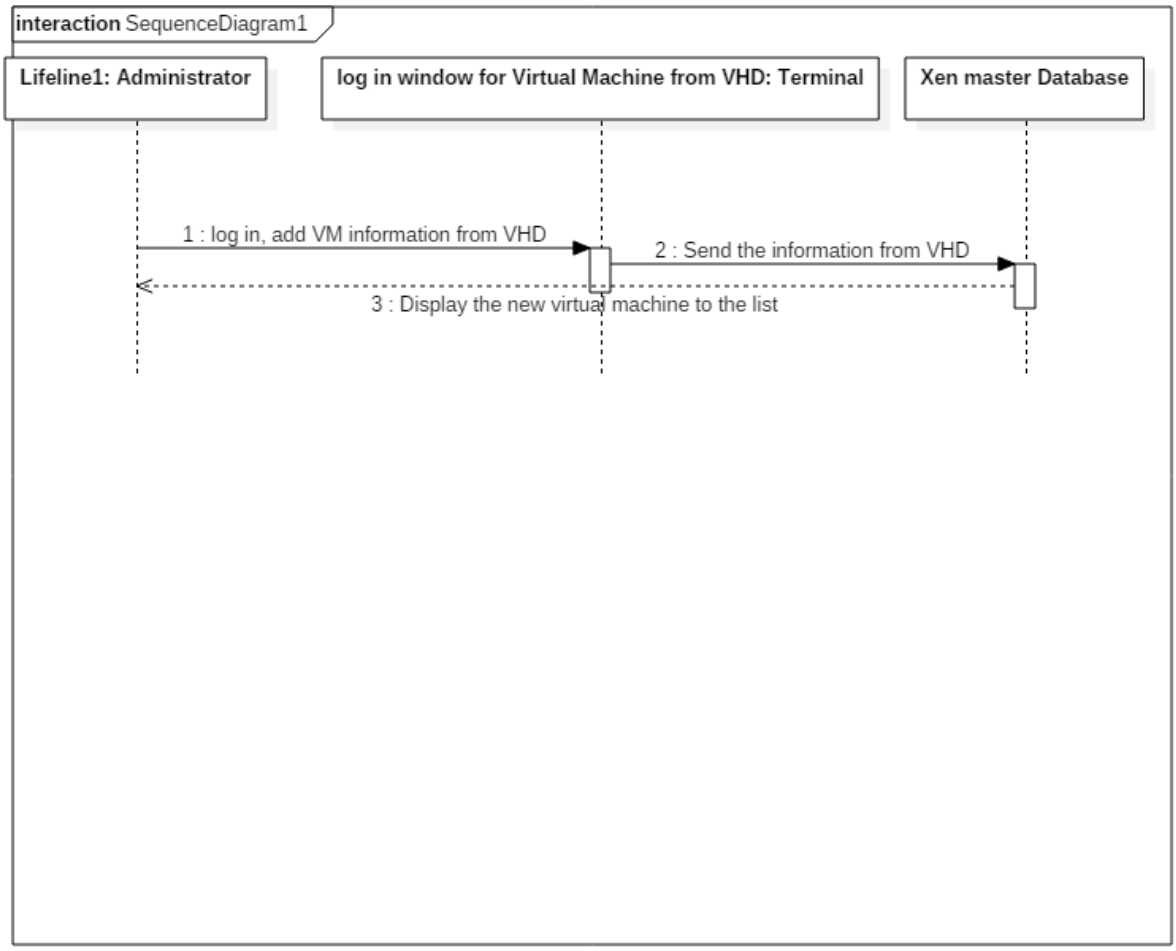



Figure 92 - Design an interface for listing the existing VHD

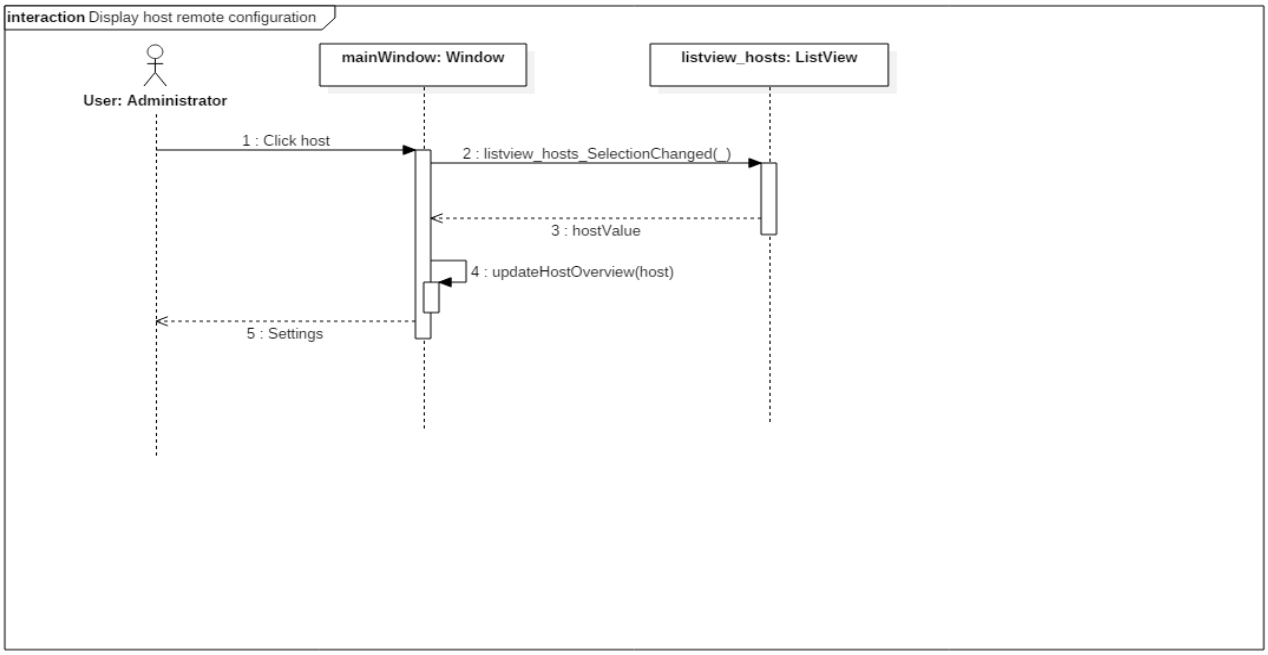



Figure 93 - Display host remote configuration

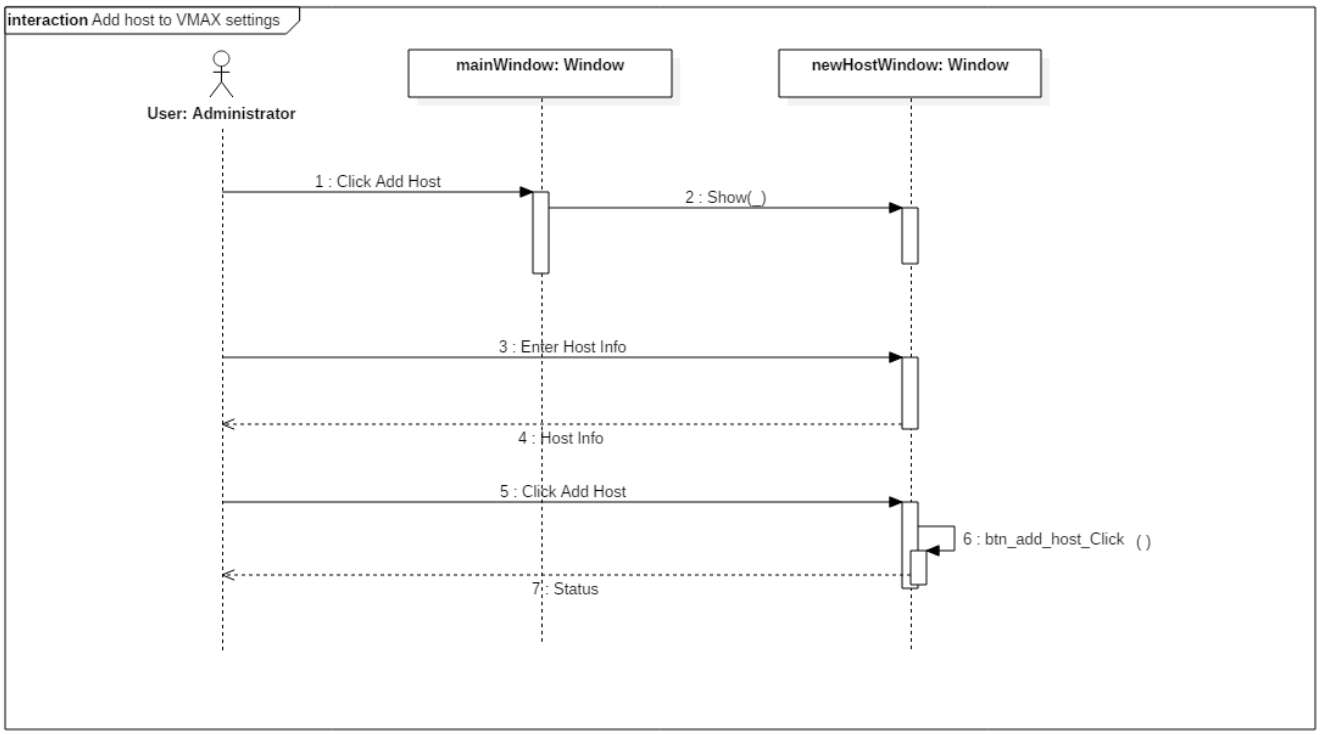



Figure 94 - Add host to VMAX settings

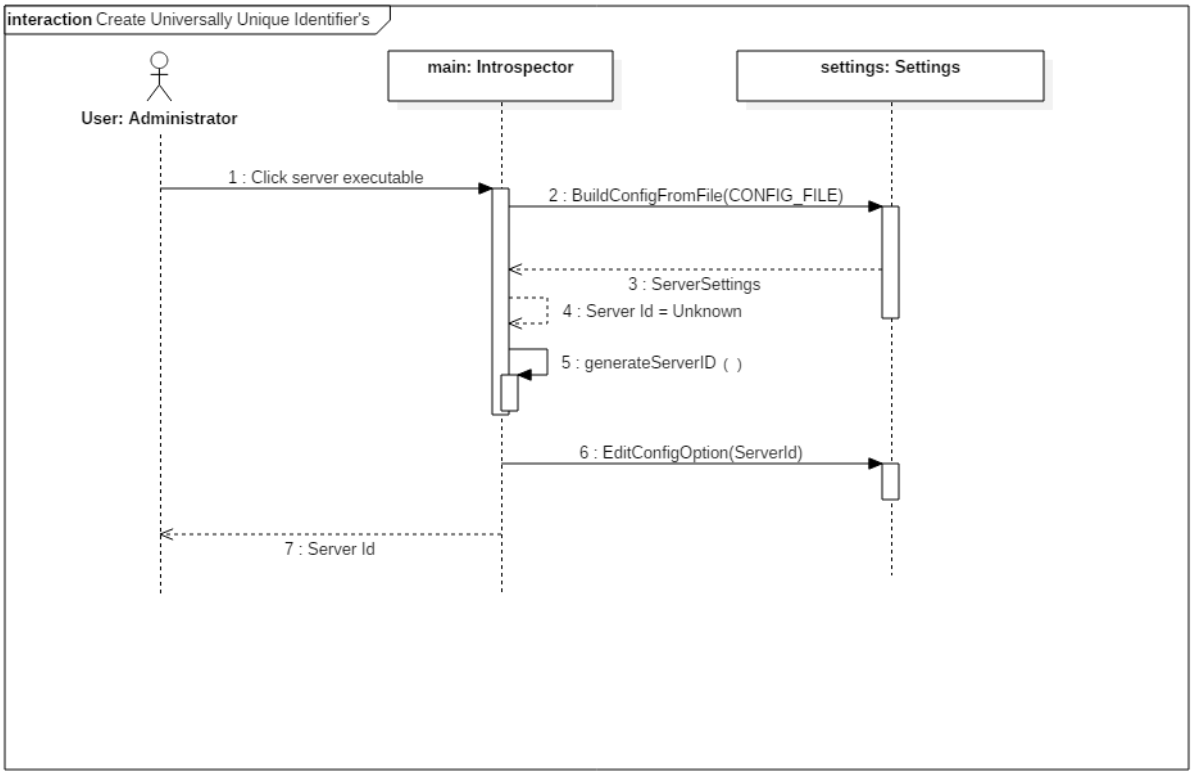



Figure 95 - Create Host Identifier

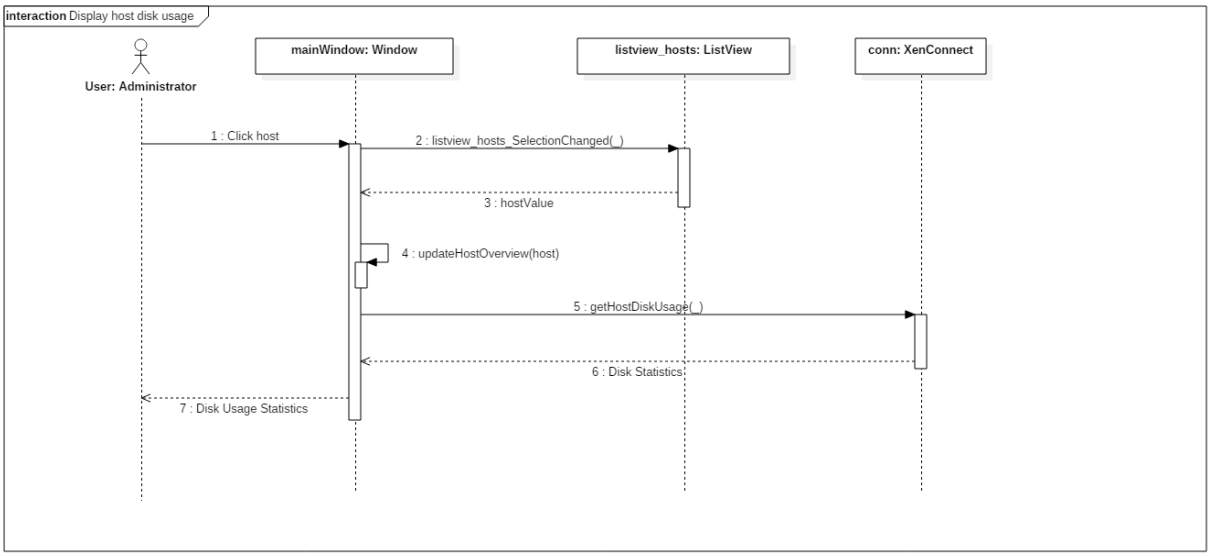



Figure 96 - Display host disk usage

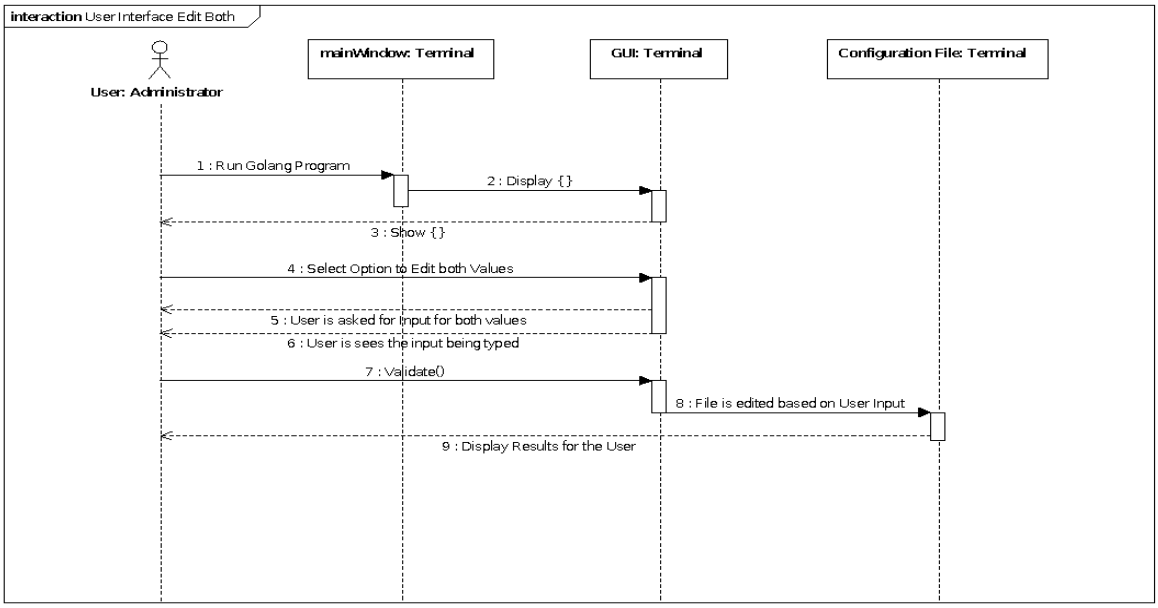



Figure 97 - Design the interface for multiple inputs and update

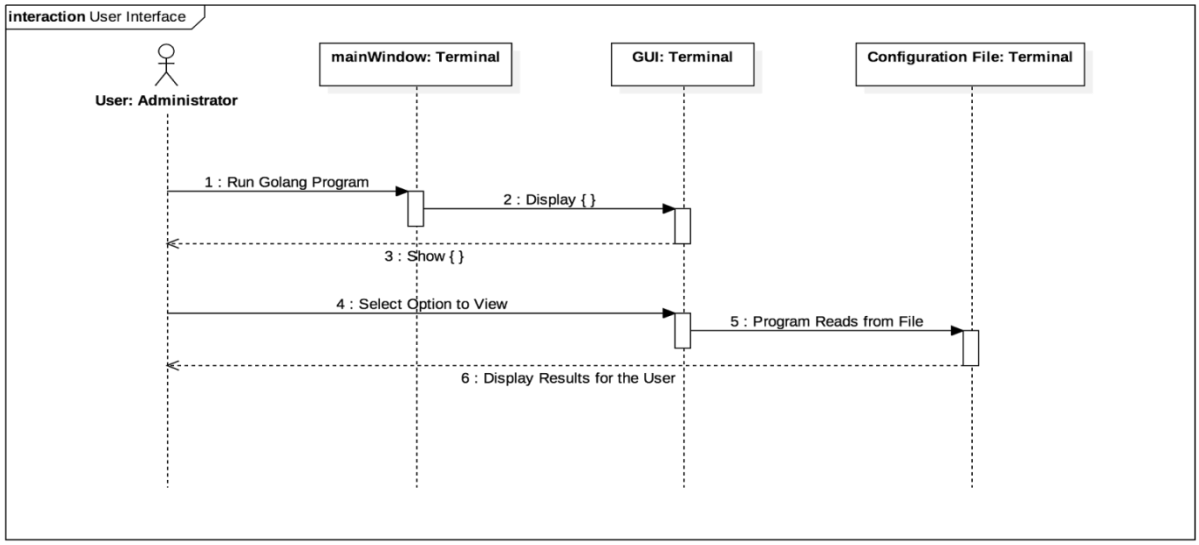



Figure 98 - Design the interface for user display

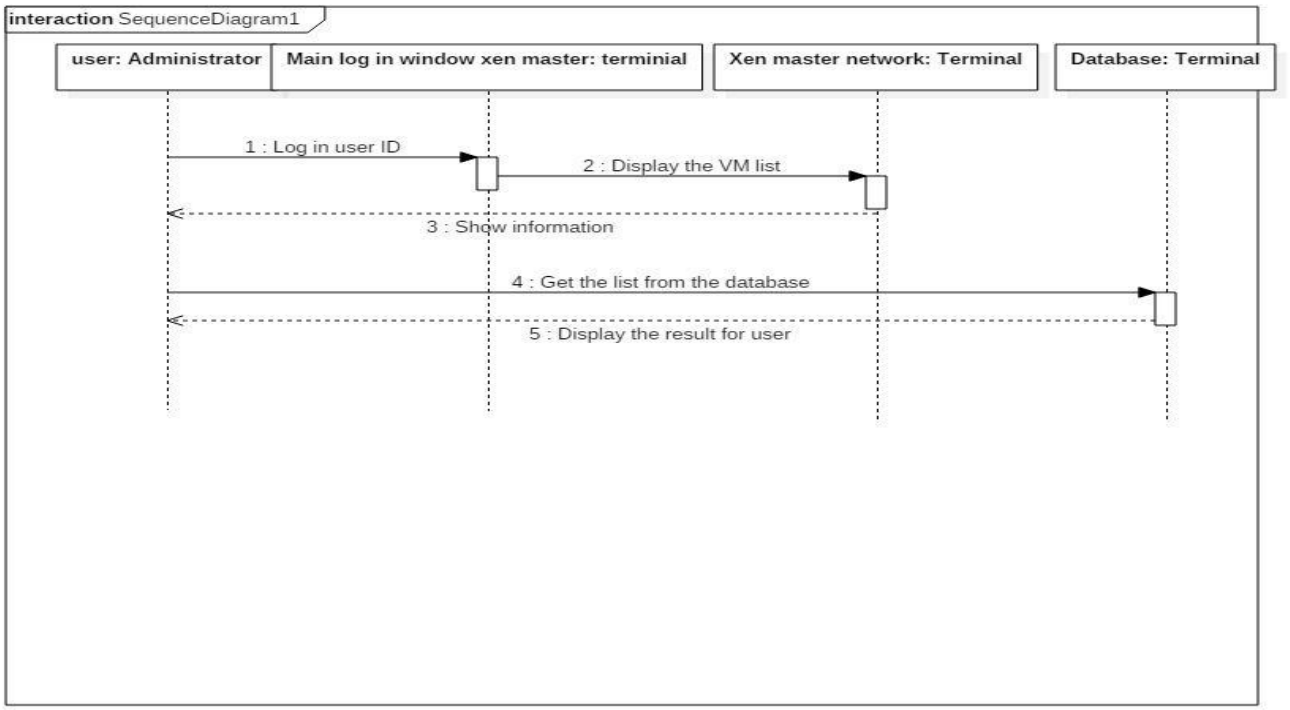



Figure 99 – Delete Virtual Machine

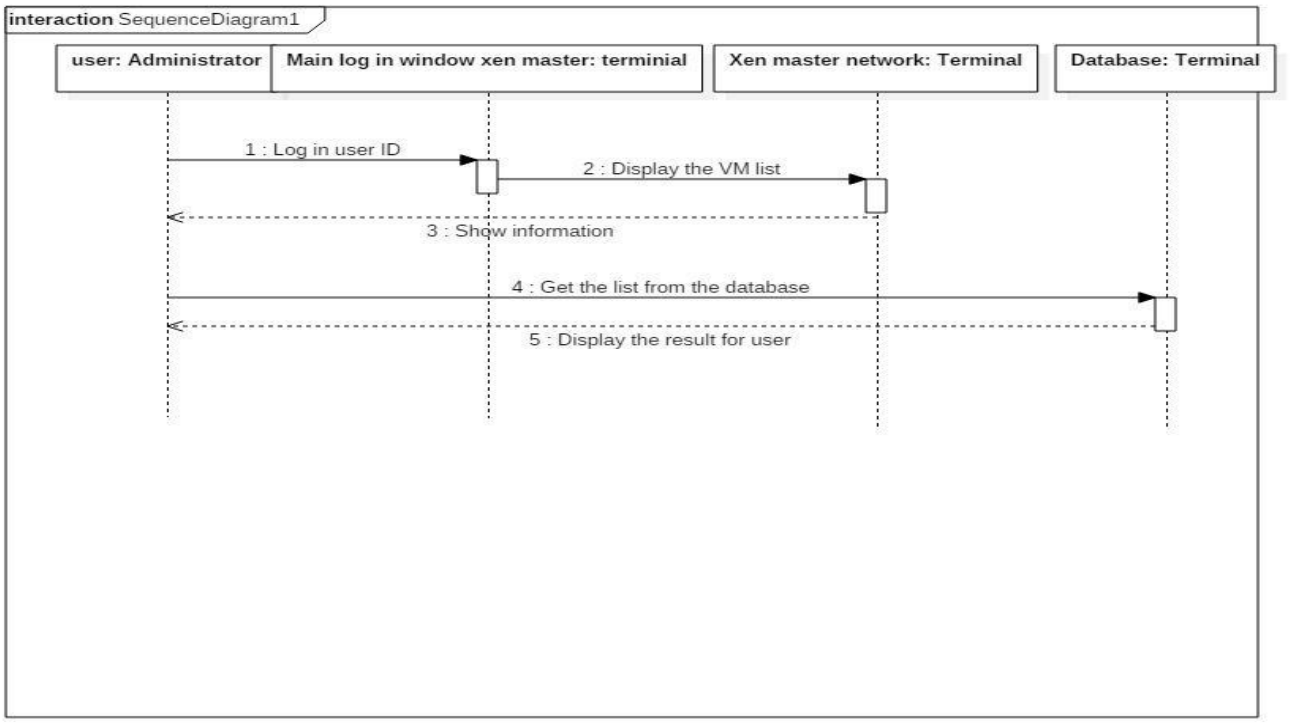



Figure 100 - Display hypervisors from the network using tree view control

## Appendix B – User Interface Design

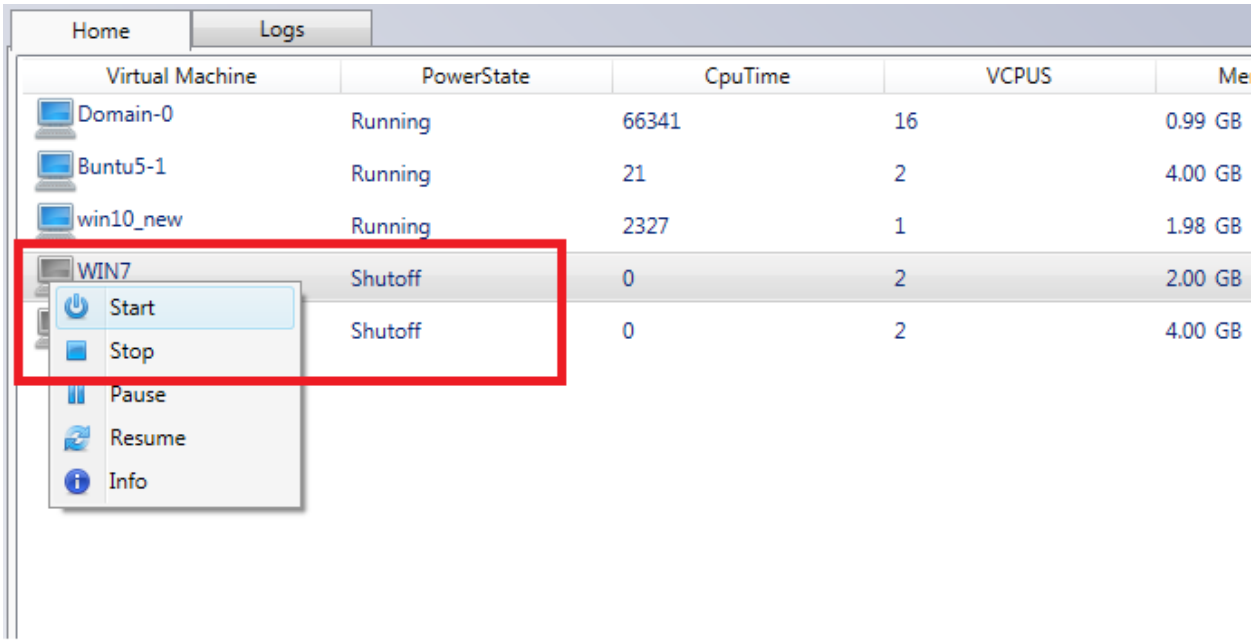



Figure 101 – VMAX Application – Start, Stop, Pause, Resume VM

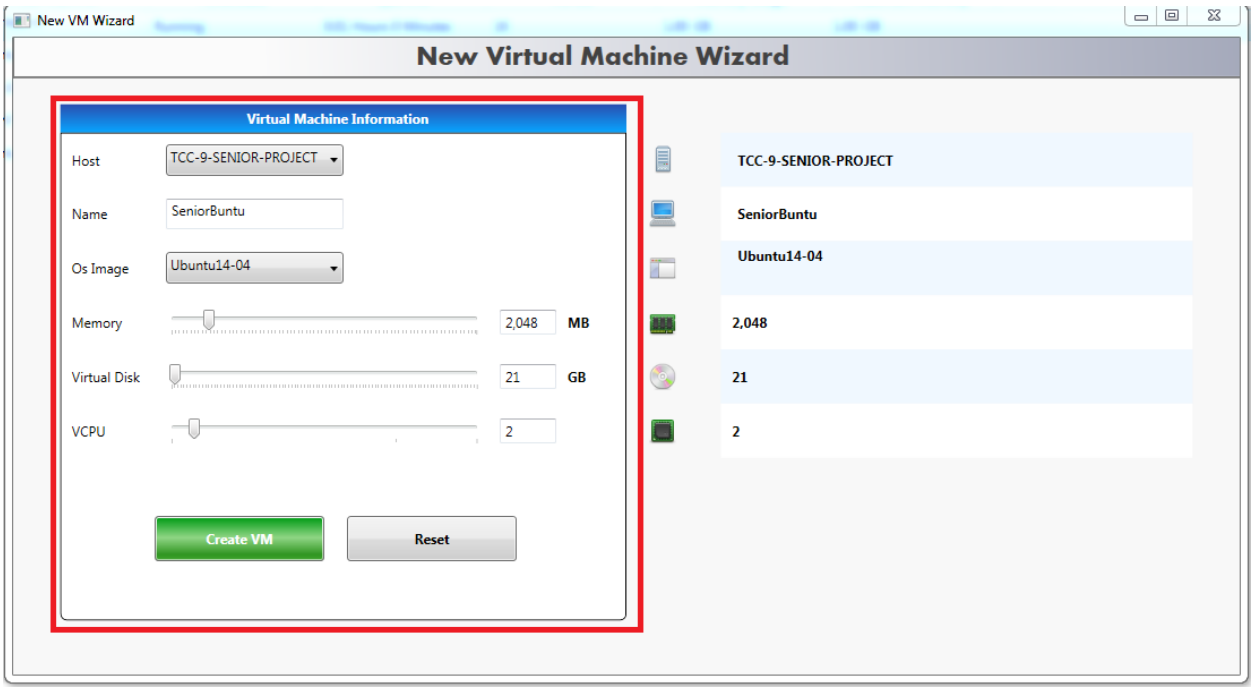



Figure 102 – VMAX Application – Create VM

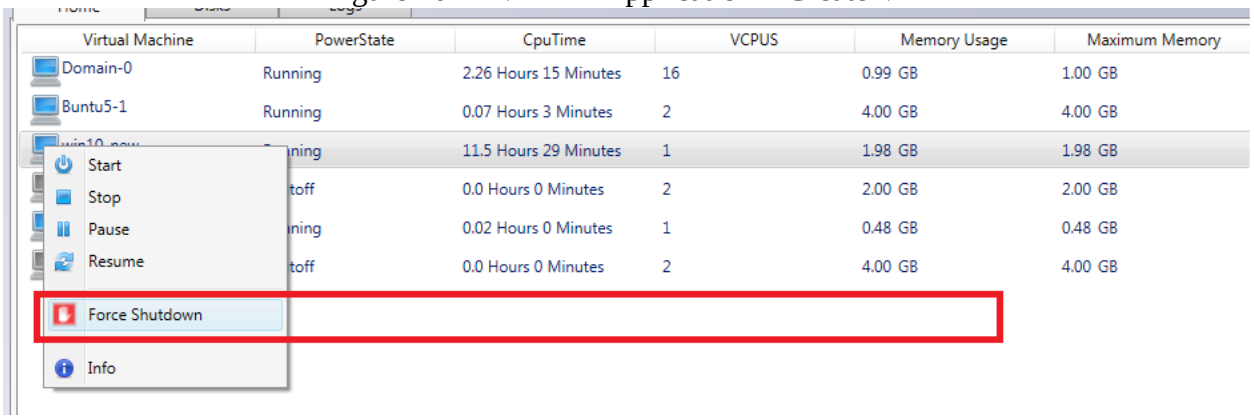



Figure 103 – VMAX Application – Force Shutdown VM

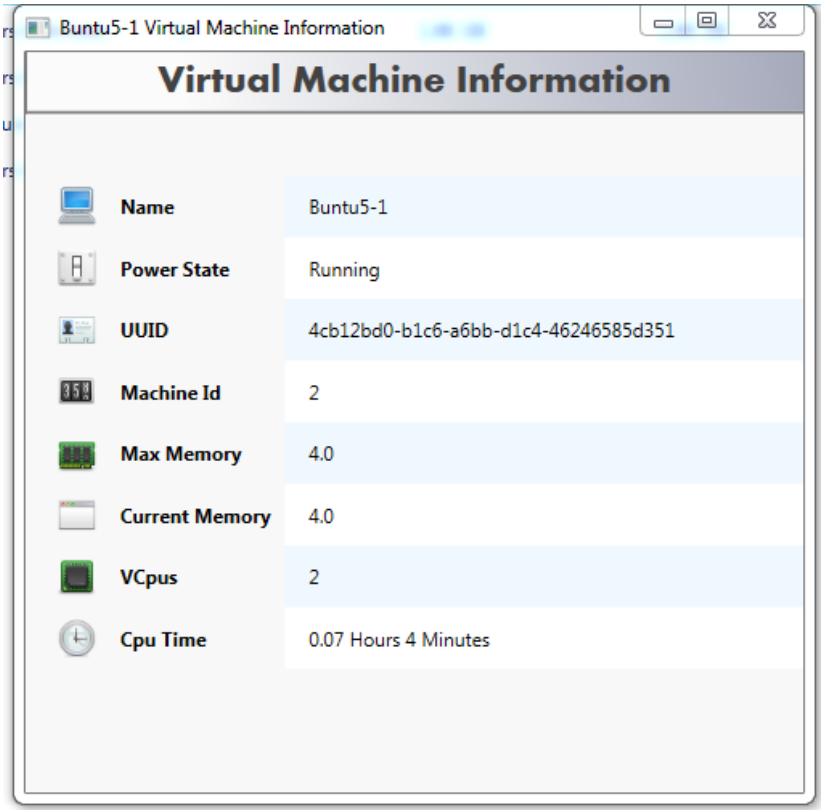



Figure 104 – VMAX Application – Info

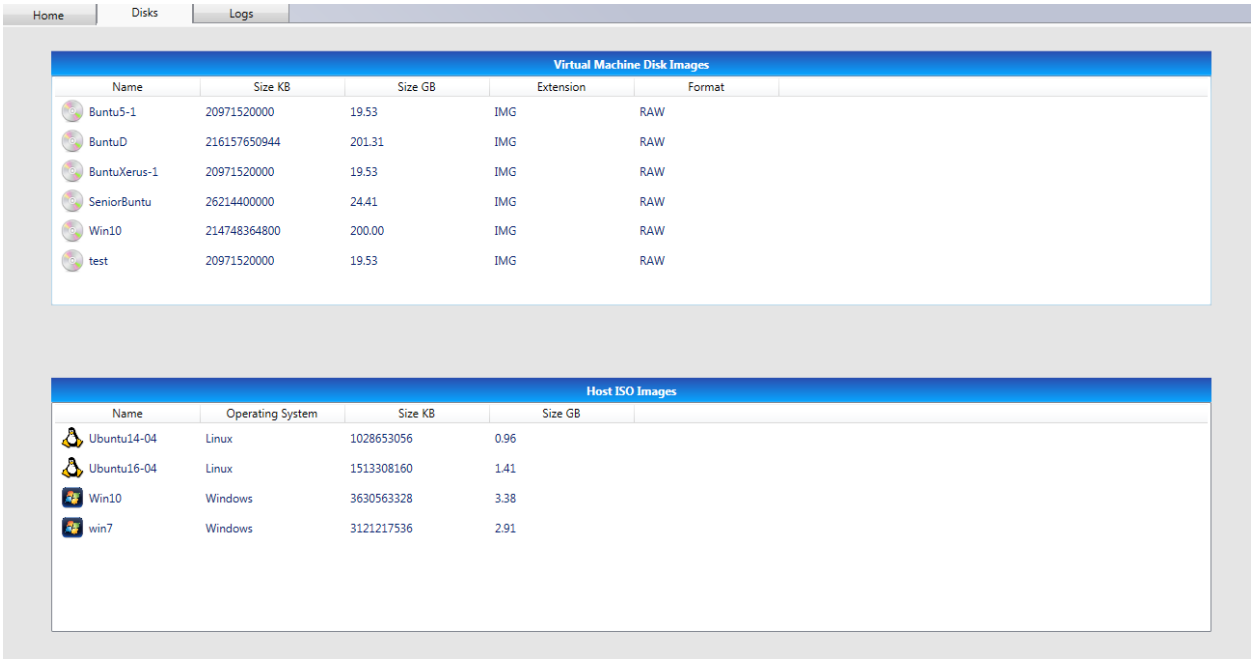



Figure 105 – VMAX Application – List VM Disks

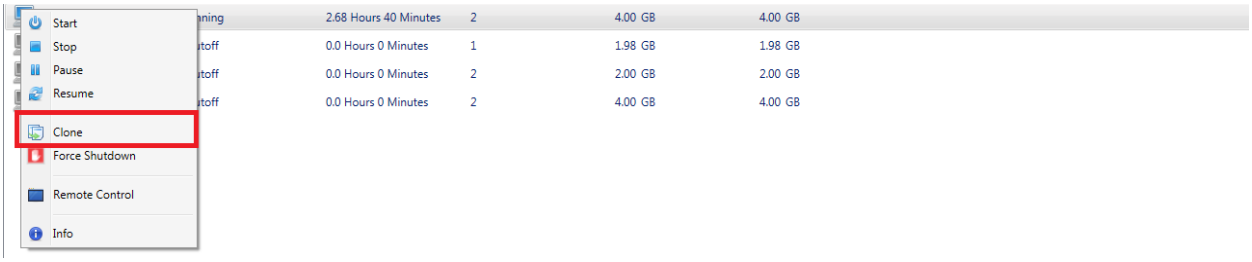
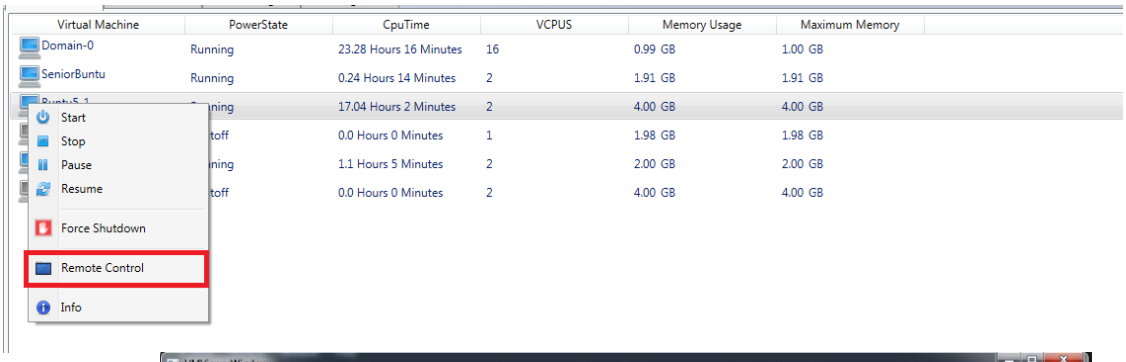



Figure 106 – VMAX Application – Clone VM


Virtual Machine
PowerState
CpuTime
VCPUS
Memory Usage
Maximum Memory
Domain-0
Running
23.28 Hours 16 Minutes
16
0.99 GB
1.00 GB
SeniorBuntu
Running
0.24 Hours 14 Minutes
2
1.91 GB
1.91 GB
17.04 Hours 2 Minutes
2
4.00 GB
4.00 GB
0.0 Hours 0 Minutes
1
1.98 GB
1.98 GB
1.1 Hours 5 Minutes
2
2.00 GB
2.00 GB
0.0 Hours 0 Minutes
2
4.00 GB
4.00 GB
Start
Stop
Pause
Resume
Force Shutdown
Remote Control
Info

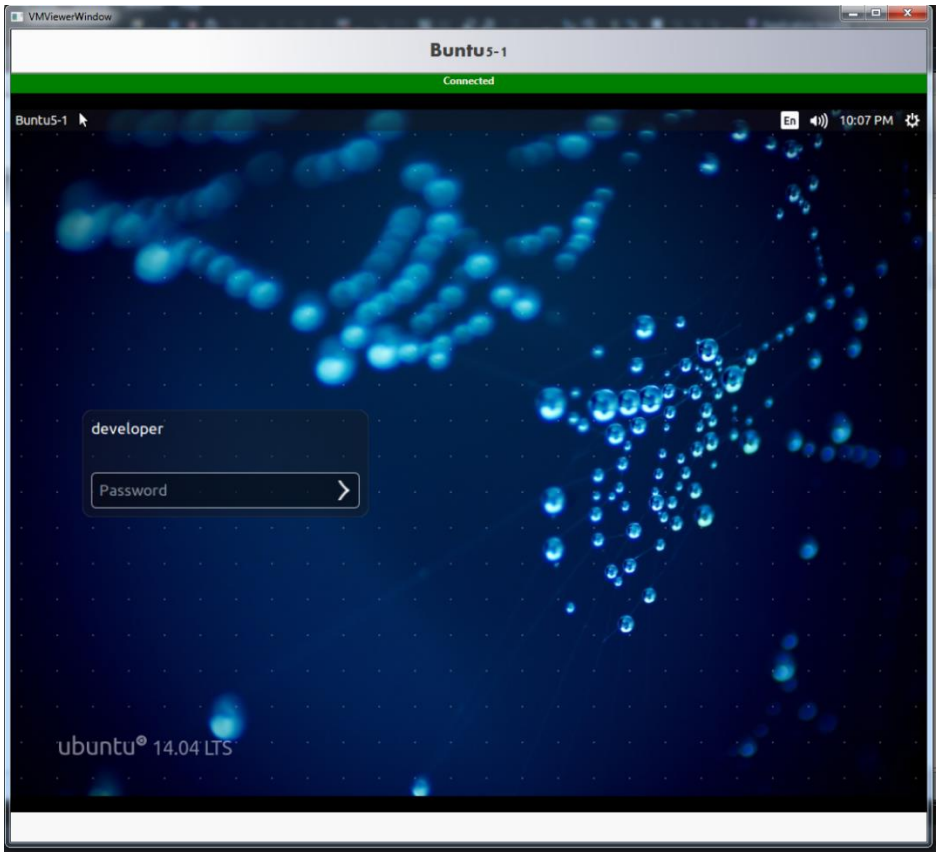

VMViewerWindow
Buntu5-1
Connected
Buntu5-1
En
10:07 PM
developer
Password
ubuntu® 14.04 LTS

Figure 107 – VMAX Application – Remote Connect VM


Figure 108 – VMAX Application – Show VM Logs

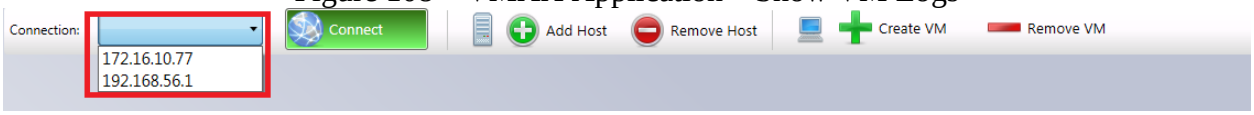



Figure 109 – VMAX Application – Select VMAX Network Interface

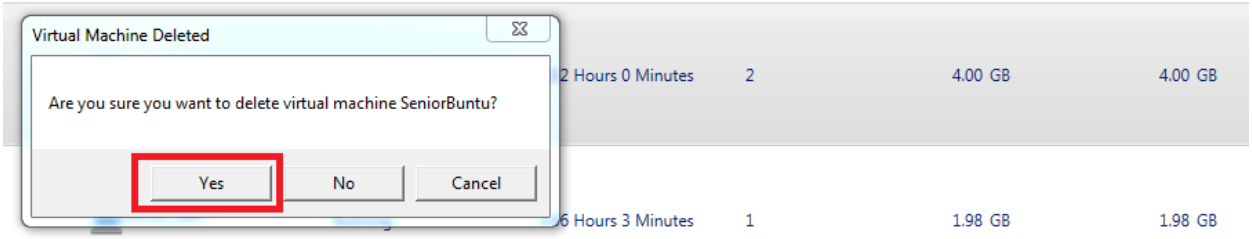



Figure 110 – VMAX Application – Delete VM

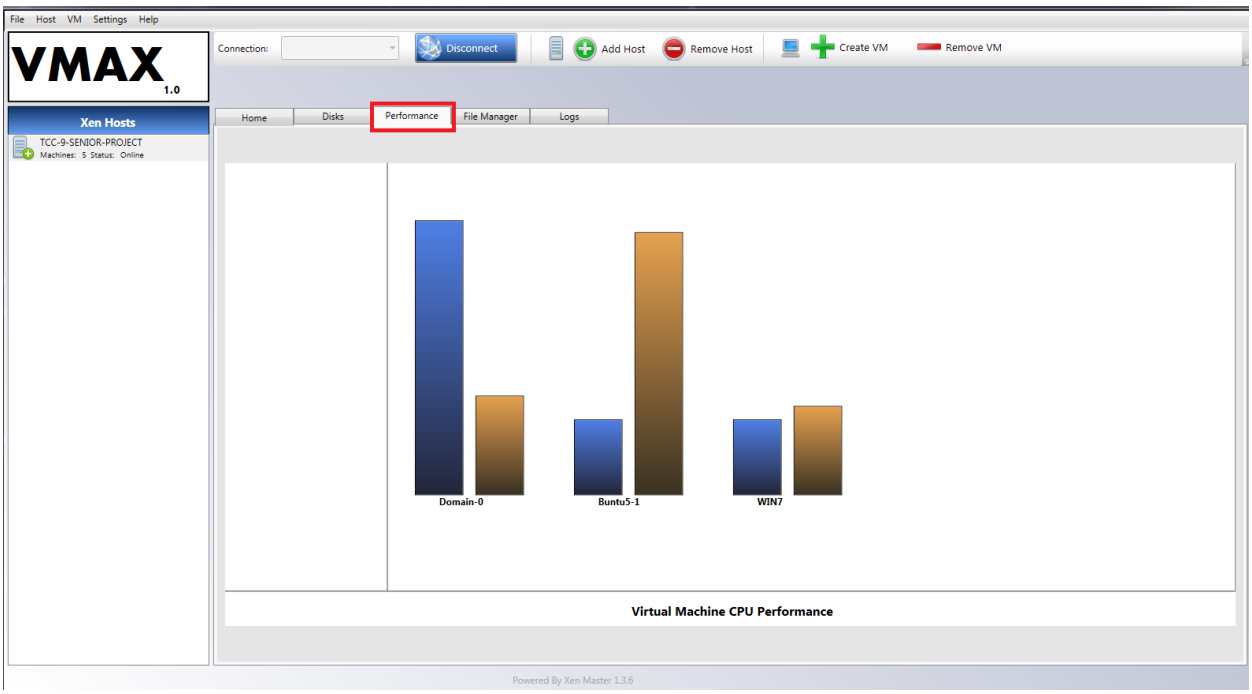



Figure 111 – VMAX Application – Display Performance Parameters

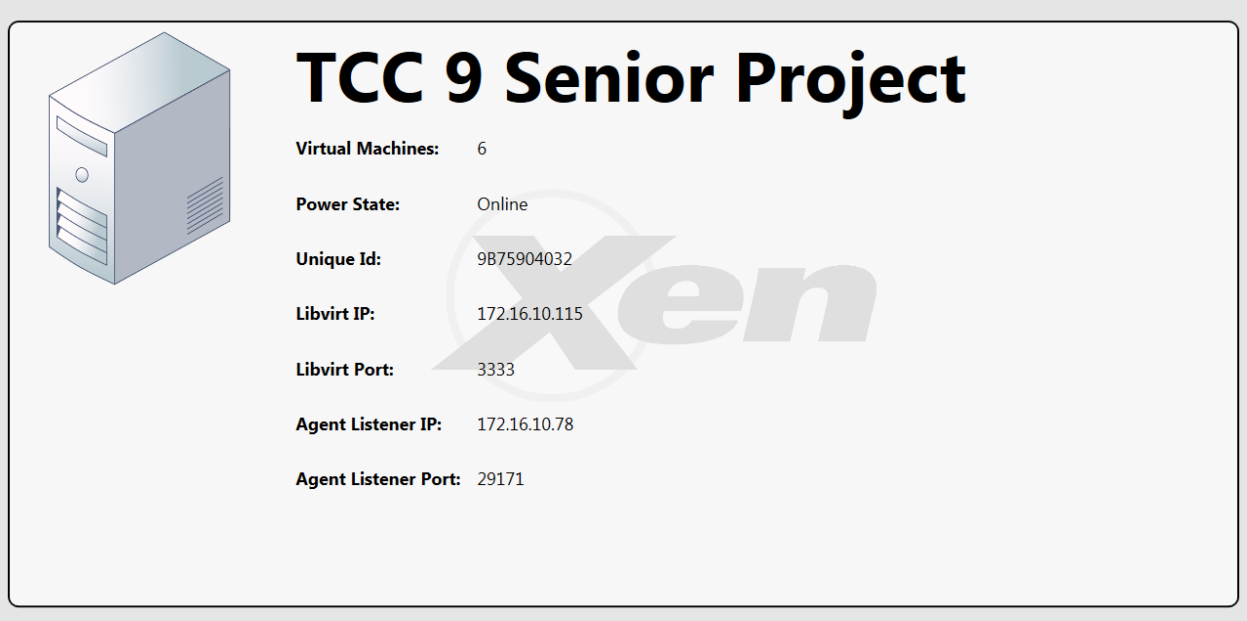



Figure 112 – VMAX Application – Display Host Settings

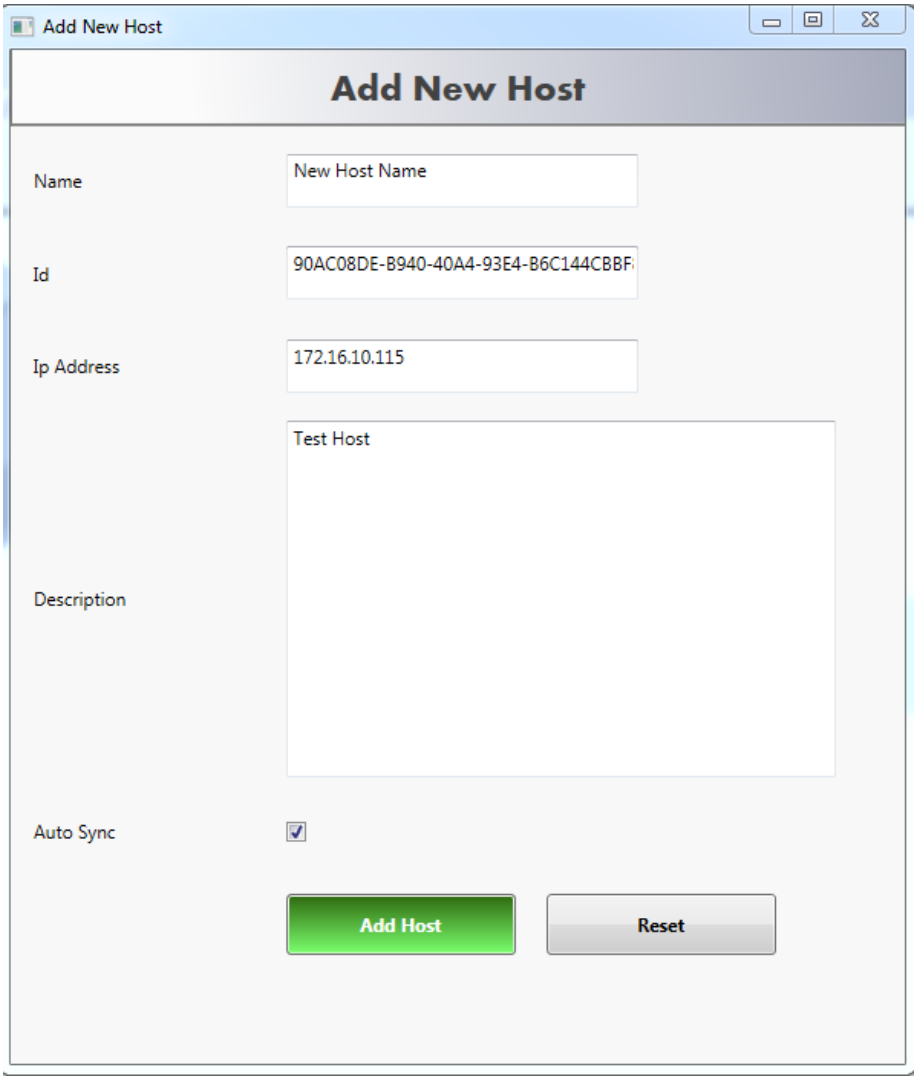



Figure 113 – VMAX Application – Add New Host

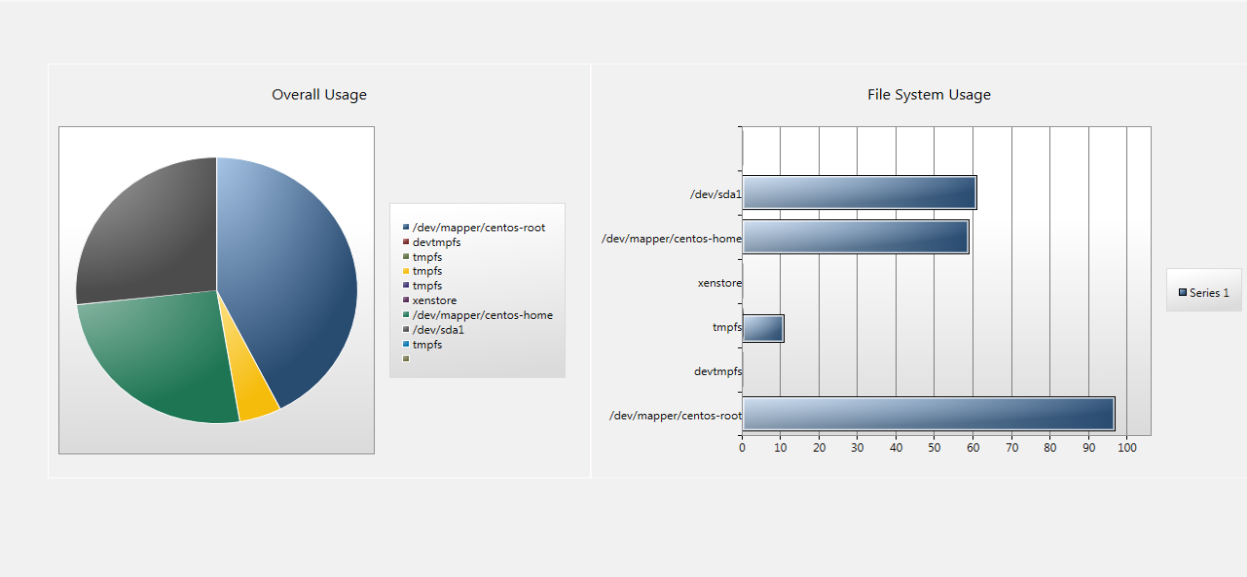



Figure 114 – VMAX Application – Host Disk Usage

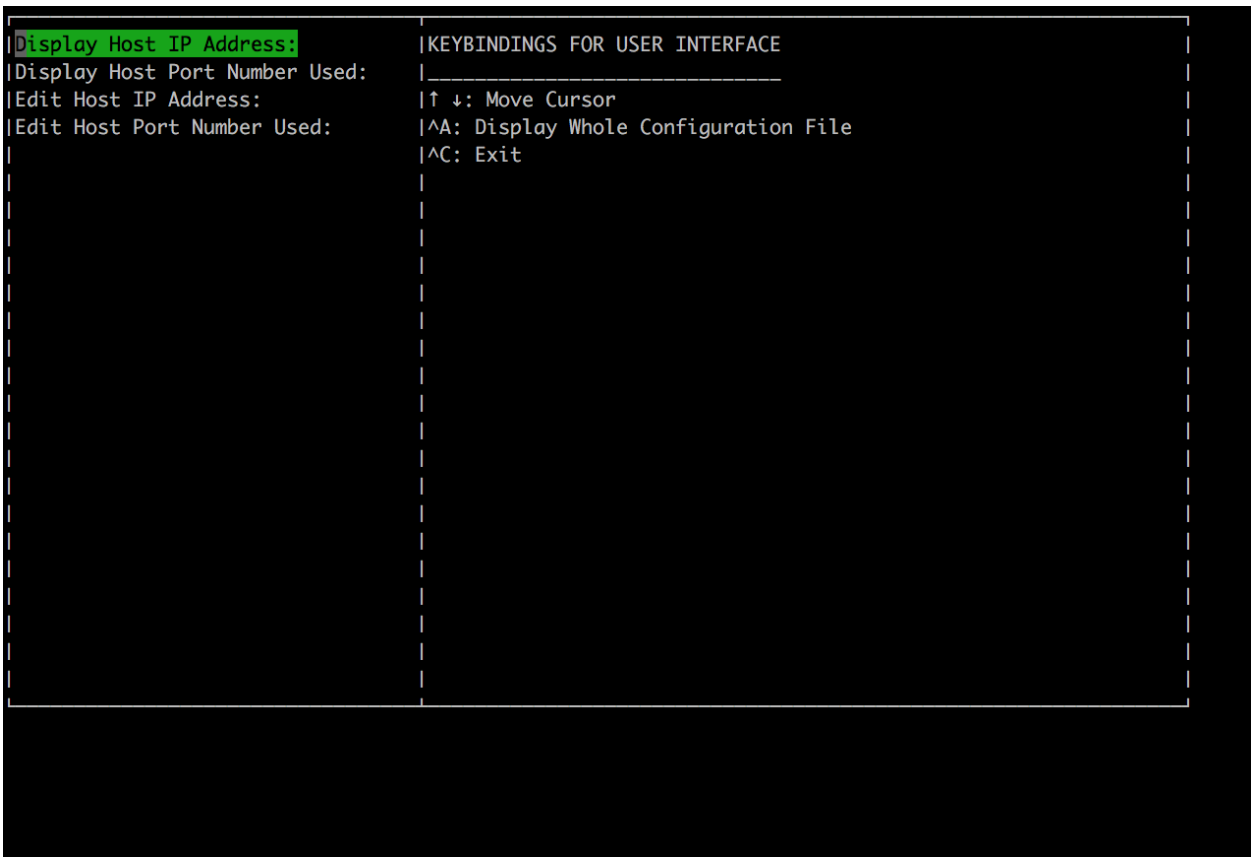



Figure 115 – GoLang UI – Create UI

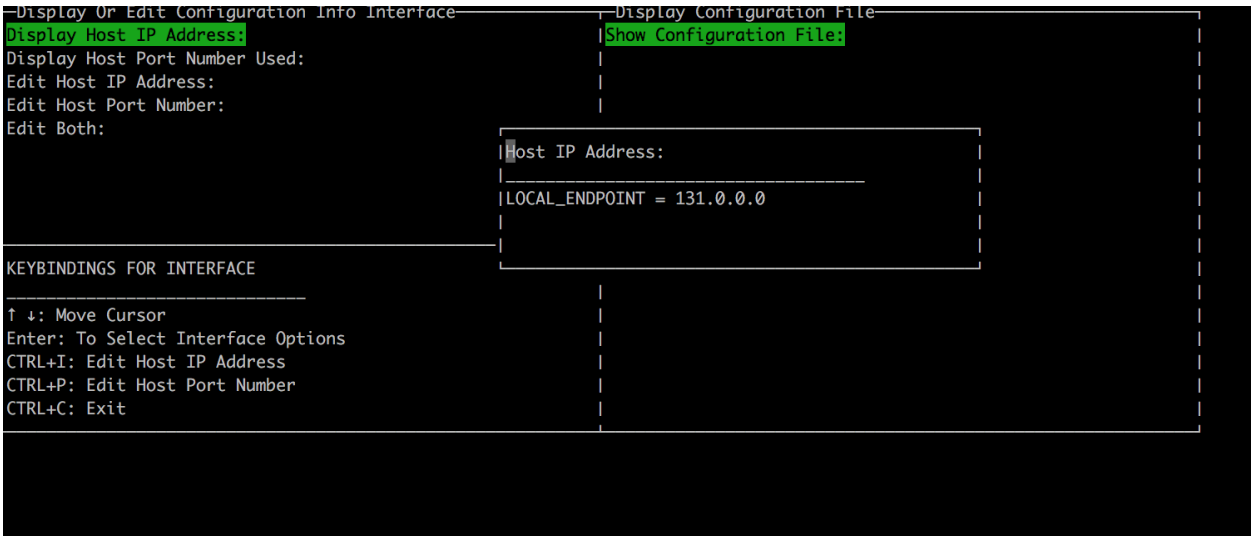



Figure 116 – GoLang UI – Display Configuration Settings

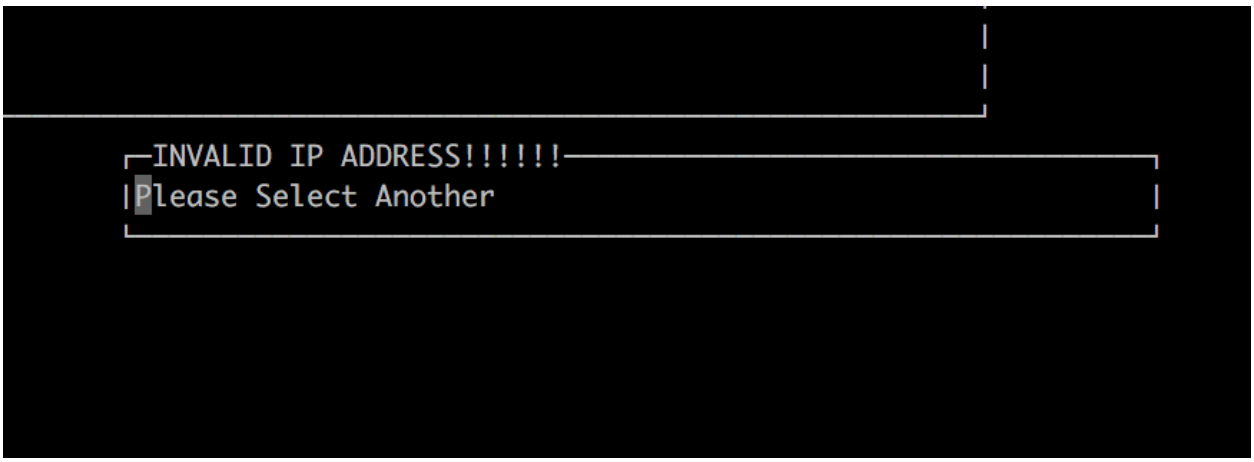
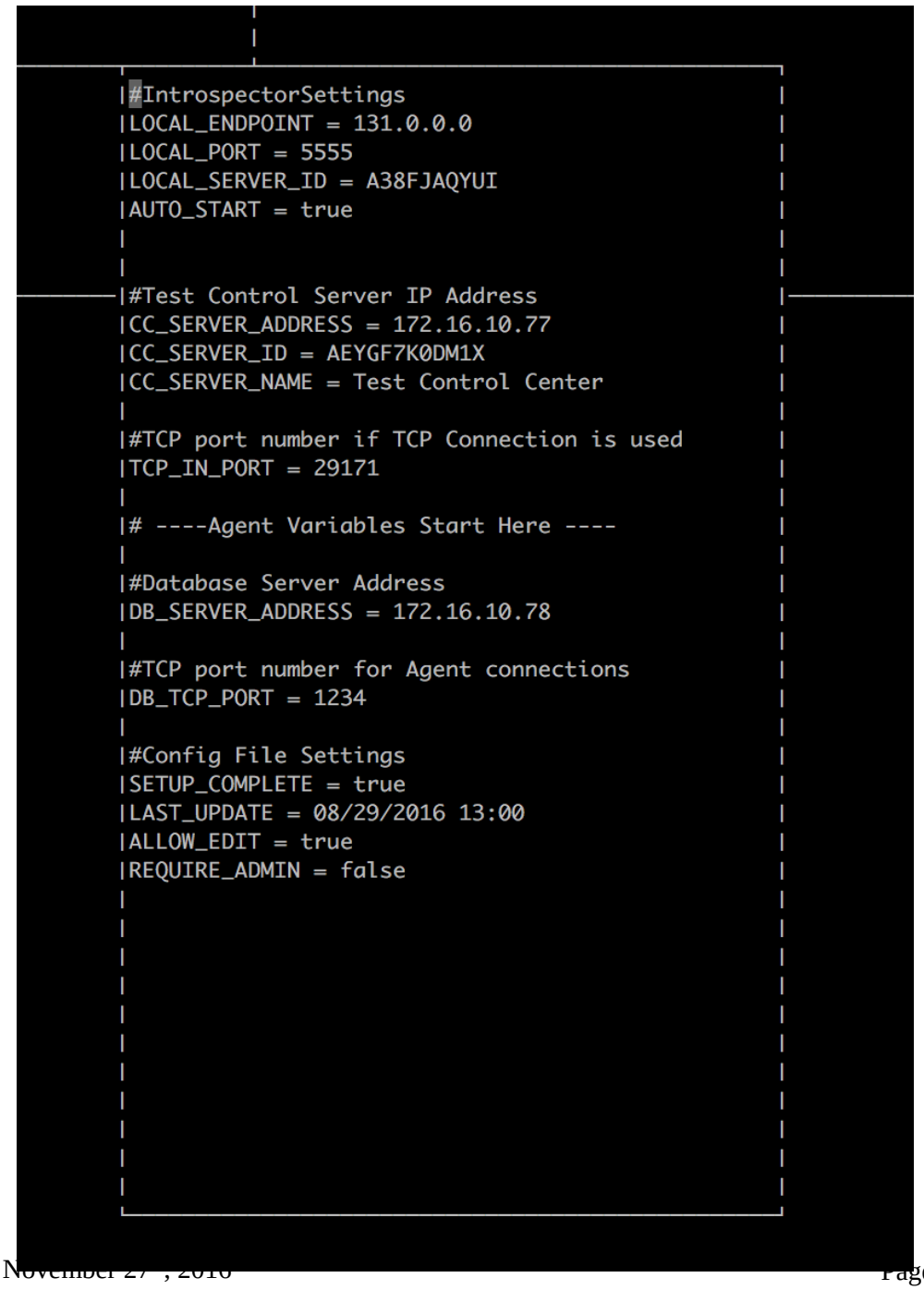



Figure 117 – GoLang UI – Validate Input Values



```
#IntrospectorSettings
LOCAL_ENDPOINT = 131.0.0.0
LOCAL_PORT = 5555
LOCAL_SERVER_ID = A38FJAQYUI
AUTO_START = true


#Test Control Server IP Address
CC_SERVER_ADDRESS = 172.16.10.77
CC_SERVER_ID = AEYGF7K0DM1X
CC_SERVER_NAME = Test Control Center

#TCP port number if TCP Connection is used
TCP_IN_PORT = 29171

# ----Agent Variables Start Here ----

#Database Server Address
DB_SERVER_ADDRESS = 172.16.10.78

#TCP port number for Agent connections
DB_TCP_PORT = 1234

#Config File Settings
SETUP_COMPLETE = true
LAST_UPDATE = 08/29/2016 13:00
ALLOW_EDIT = true
REQUIRE_ADMIN = false
```

Figure 118 – GoLang UI – View Config File


Figure 119 – GoLang UI – Signal Handler for Quit

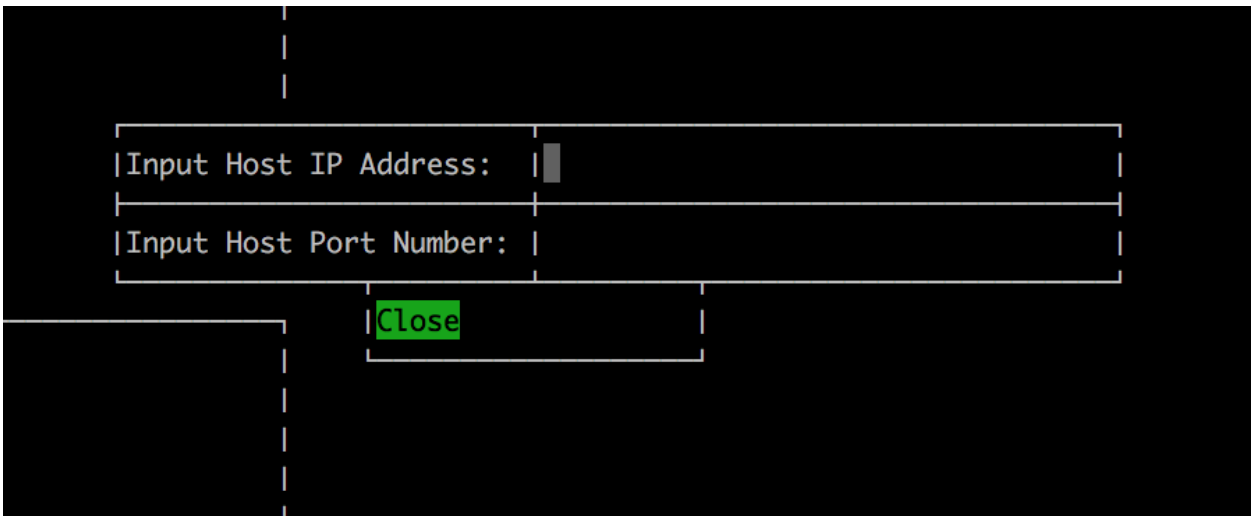



Figure 120 – GoLang UI – Update with one selection

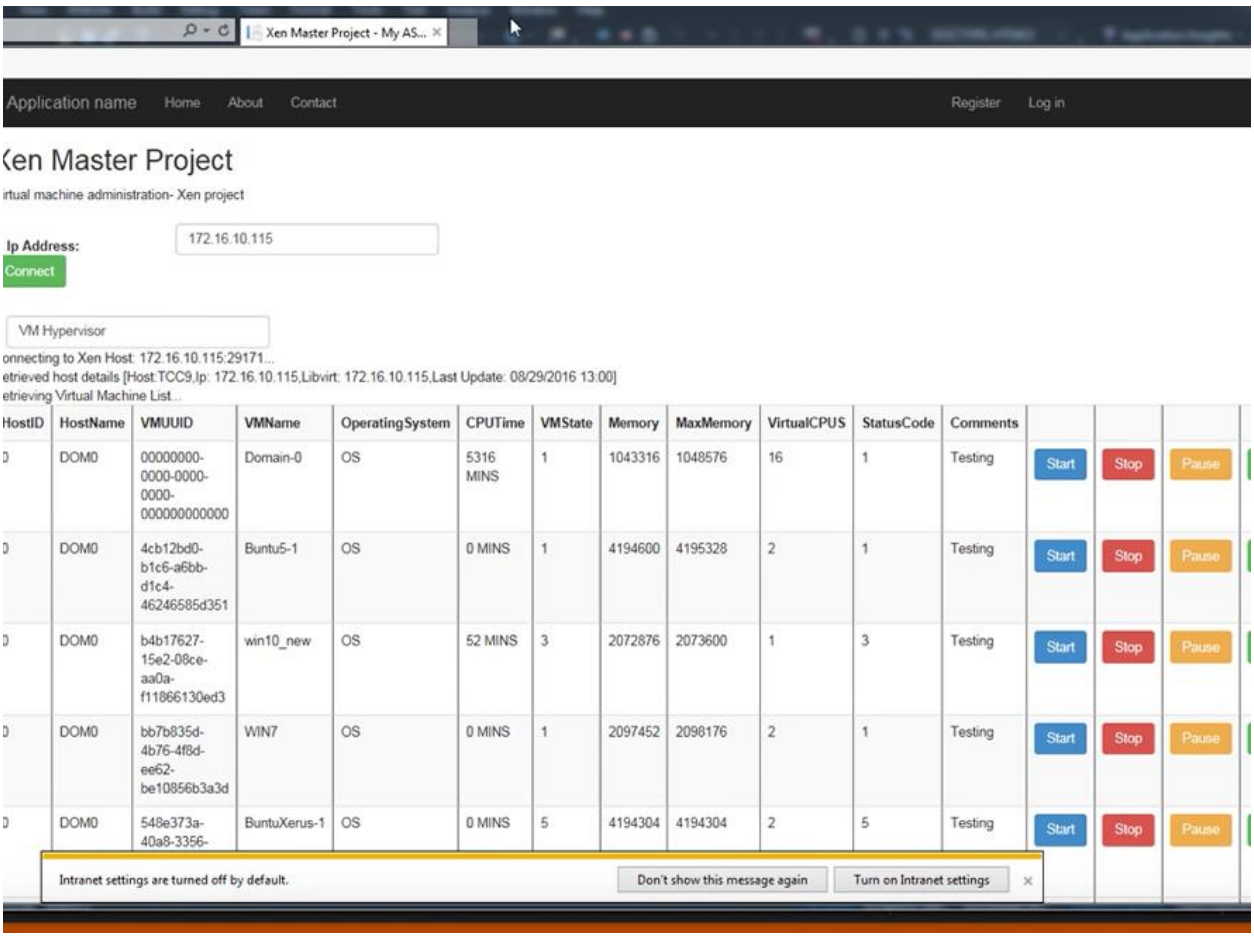



Figure 121 – Web Interface – Pause, Start, Stop Functionality

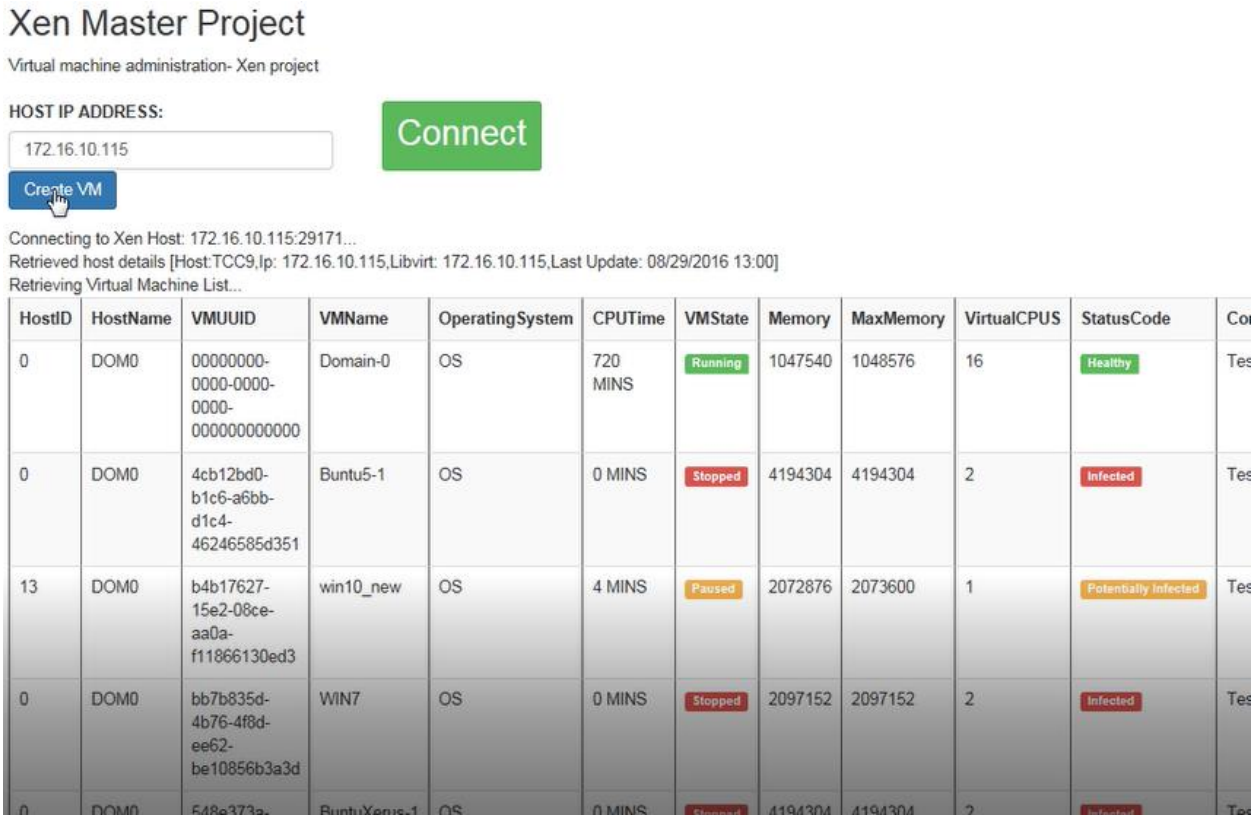



Figure 122 – Web Interface – Create VM

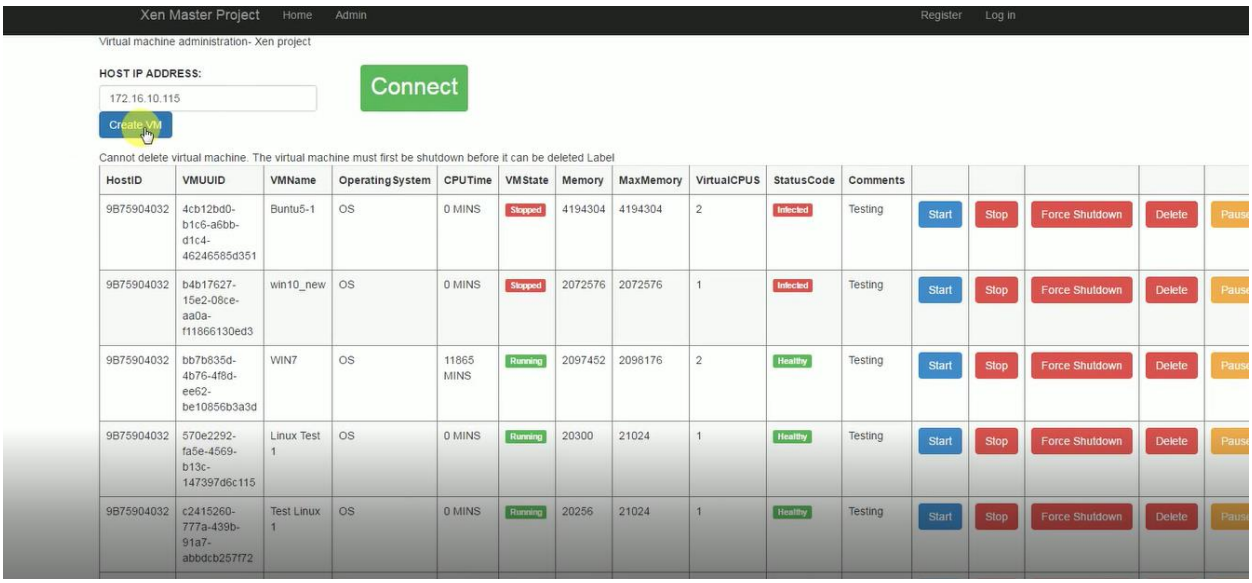



Figure 123 – Web Interface – Add VHD Information


Figure 124 – Web Interface – Delete VM

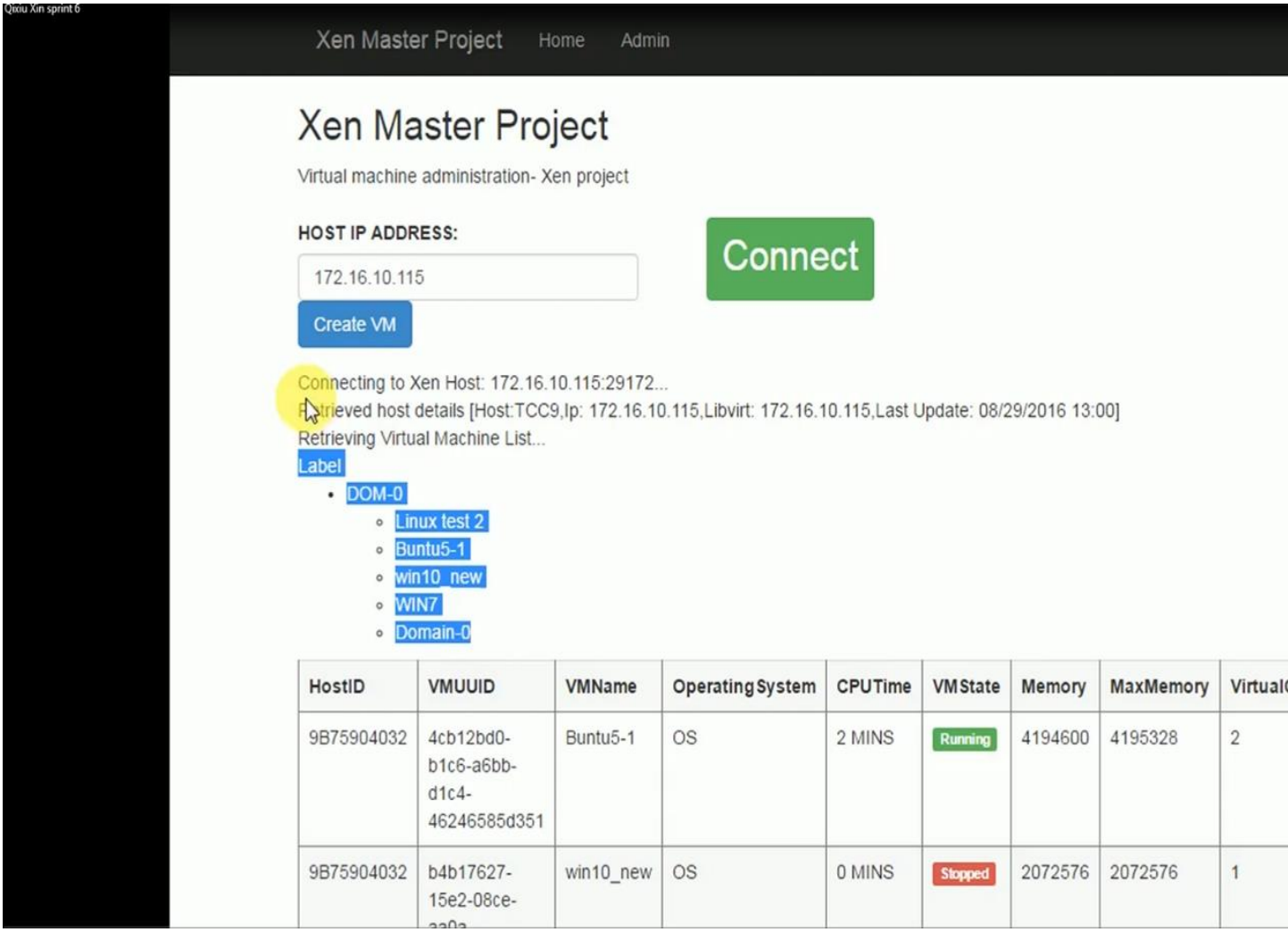



Figure 125 – Web Interface – Display Hypervisors

## Appendix C – Sprint Review Reports

**Sprint 1**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 14:00
End time: 14:30
Date: 9/9/2016

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- **User Story#121**: Access Virtual Machines
- **User Story#122**: View Host Virtual Machines
- **User Story#141**: View Settings of Configuration File
- **User Story#142**: View Settings of host security agent

**Sprint 2**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 14:10
End time: 14:46
Date:9/23/2016

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All:

- **User Story#143**: Start a Virtual Machine
- **User Story#144**: Stop a Virtual Machine
- **User Story#145**: Pause a Virtual Machine
- **User Story#146**: Resume a Virtual Machine
- **User Story#155**: Load and View VMAX Settings
- **User Story#150**: Edit the Settings of Configuration File
- **User Story#164**: Get the list of virtual machines for web dashboard
- **User Story#163**: Update the VMAX Database
- **User Story#153**: Display the virtual machines on the web interface

**Sprint 3**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 15:07
End time: 16:00
Date:10/10/2016

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All:

- **User Story#232:** Create a new virtual machine
- **User Story#233**: Force shutdown virtual machine
- **User Story#234:** Create virtual machine templates
- **User Story #235:** Display virtual machine information
- **User Story #236:** Create a UI for the configuration editing
- **User Story #237:** Use interface to edit host IP address
- **User Story #238:** Use interface to edit port number
- **User Story #275:** List virtual machine disk configurations
- **User Story #270:** Modify the code for .net format of the web
- **User Story #271:** Add pause functionality; add resume functionality
- **User Story #272:** Add stop functionality
- **User Story #273:**Update database after each function call; display current VM status

**Sprint 4**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 15:00
End time: 15:37

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- **User Story #312** - Create Virtual Machine From Existing Disk Image
- **User Story #313** - Transfer Virtual Machine Images
- **User Story #314** - Clone Virtual Machine
- **User Story #315** - Control Remote Virtual Machine
- **User Story #317** - Display what is typed by the user into the interface

- **User Story #318** - Validate input values for both IP address and Port number
- **User Story #319** - Enter all the values to UI and update through one submission
- **User Story #320** - Clean up user interface
- **User Story #321** - Implement " Create" Virtual Machine
- **User Story #322** - Display status of the VM machine

**Sprint 5**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 15:15
End time: 15:45

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- **User Story #359** - Delete Virtual Machine
- **User Story #360** - Display Performance Parameters
- **User Story #186** - Display User Notifications
- **User Story #200** - Select VMAX Network Interface
- **User Story #377**-Add more information on the home page
- **User Story #378**-Interface the existing VHD
- **User Story $379**-Create a VM from the existing VHD
- **User Story #369** - Clean up the Interface
- **User Story #370** - Update all parameters with single selection
- **User Story #371** - View file should be a separate button/selection
- **User Story #372** - Add a signal handler to catch the quit options

**Sprint 6**

Attendees: Dennis Obando, Qixiu Xin, D'Mita Levy, Dr. Himanshu Upadhyay
Start time: 15:45
End time: 16:15

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

- **User Story #420** – Design the interface for multiple inputs and update
- **User Story #421** – Design the interface for user display
- **User Story #422** – Delete Virtual Machine
- **User Story #423** – Display CPU Parameters
- **User Story #424** – Display CPU hypervisors from the network using Tree View Control
- **User Story #417**-Create Universally Unique Identifiers
- **User Story $148**-Display host remote configuration
- **User Story #188** – Add host to VMAX settings
- **User Story #418** – Display host disk usage

## Appendix D - Sprint Retrospective Reports

**Sprint 1**

Date: 09/09/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 14:30
End time: 15:05

What went wrong?

- Did we do a good job estimating our team's velocity?
  - Yes, we completed all product backlog items
  -
- Did we do a good job estimating the points (time required) for each user story?
  - Yes
- Did each team member work as scheduled?
  - Yes

What went right?

- All user stories and product backlog tasks were completed
- The velocity of the team for the current sprint cycle was adequate for the completion of the current user stories

- The learning requirements for the sprint were achievable given the user stories

How to address the issues in the next sprint?

- How to improve the process?
  - Spend more time sprint planning and breaking user stories down into smaller stories
  - Add more tasks per user stories instead of making each task so large that it could have been a user stories
- How to improve the product?
  - Add additional update, look, feel, and overall user responsiveness to the interface
  - Mirror some of the packages and libraries on the test system

**Sprint 2**

Date: 09/23/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 14:46
End time: 15:05

What went wrong?

- Did we do a good job estimating our team's velocity?
  - Yes, but we could fine tune the tasks time to ensure that some of them don't go over estimated times
- Did we do a good job estimating the points (time required) for each user story?
  - Yes.
- Did each team member work as scheduled?
  - No

What went right?

- We were able to complete all of our user stories and the product owner is pleased
- We did a great job following the agile/scrum process and updating Mingle
- The team maintained a good working relationship and was in close contact

How to address the issues in the next sprint?

- How to improve the process?

    - Separate the planning for each task in Mingle as they relate to each user story
    - Change our schedules as necessary to match the week of the sprint
- How to improve the product?
    - Improve the web interface coding part, create VM and enabling force vm shutdown
    - Better/more robust user interface design so that users will enjoy the feel of the application
    -

**Sprint 3**

Date: 10/07/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 20:20
End time: 20:40

What went wrong?

- Did we do a good job estimating our team's velocity?
    - Yes, our velocity estimates were much closer to our expectations but still slightly off. Some user stories took longer than expected.
- Did we do a good job estimating the points (time required) for each user story?
    - No, some user stories ended taking more time to implement than previously estimated
- Did each team member work as scheduled?
    - No

What went right?

- All users stories were completed and the project manager was happy with our progress
- We implemented some robust and rich features that really improve the system and applications

How to address the issues in the next sprint?

- How to improve the process?
    - Work and adjust schedules as necessary to ensure that we have no last minute changes

- How to improve the product?
    - Add more user interface elements to provide feedback to users that might be confused about what's going on in the background of the application
    - Remove some of the technical jargon and numerics that show up in the interface so that it is useable by anyone

**Sprint 4**

Date: 10/21/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 15:50
End time: 16:20

What went wrong?

- Did we do a good job estimating our team's velocity?
    - Yes.
- Did we do a good job estimating the points (time required) for each user story?
    - Yes.
- Did each team member work as scheduled?
    - No.

What went right?

- We finished all the user stories on time, before the sprint review with the project manager. Also, we had deep discussion through meeting time for the group sprint review and planning.
- Everyone did a great job with the interface portions of the application - nothing failed during demo
- We did a great job during scrum meetings and updating everything in mingle

How to address the issues in the next sprint?

- How to improve the process?
  Increase the testing time and improve testing procedure

- How to improve the product?
    - Add more information with the diagram on home page for user to search and track

- Add more error handling for the user interface and messages that provide better feedback to the user

**Sprint 5**

Date: 11/04/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 16:00
End time: 16:20

What went wrong?

- Did we do a good job estimating our team's velocity?
  - Yes.
- Did we do a good job estimating the points (time required) for each user story?
  - Yes.
- Did each team member work as scheduled?
  - No.

What went right?

- Our documentation has become very fine-tuned and if mistakes are made they are very minor mistakes

How to address the issues in the next sprint?

- How to improve the process?
  Review documentation as a collective group to ensure that we are producing the same format documents and making less mistakes

- How to improve the product?
  - The products could be a bit more user friendly, maybe some minor GUI changes to reflect that

**Sprint 6**

Date: 11/18/2016
Attendees: Qixiu Xin, D'Mita Levy, Dennis Obando
Start time: 16:00
End time: 16:15

What went wrong?

- Did we do a good job estimating our team's velocity?
  - Yes.
- Did we do a good job estimating the points (time required) for each user story?
  - Yes.
- Did each team member work as scheduled?
  - No.

What went right?

- Our documentation has become very fine-tuned and if mistakes are made they are very minor mistakes
- All user stories were complete and Project Manager was very pleased with all the work done throughout the semester

How to address the issues in the next sprint?

- How to improve the process?
  Review documentation as a collective group to ensure that we are producing the same format documents and making less mistakes

- How to improve the product?
  - The products could be a bit more user friendly, maybe some minor GUI changes to reflect that

# REFERENCES

Reference Manual for libvirt. (n.d.). Retrieved September 1, 2016, from http://libvirt.org/html/index.html

Design Patterns in Golang: Decorator : Blog.ralch.com. (2016). Retrieved September 1, 2016, from http://blog.ralch.com/tutorial/design-patterns/golang-decorator/

R. (2016). Rgbkrk/libvirt-go. Retrieved September 1, 2016, from https://github.com/rgbkrk/libvirt-go