Open Virtual Desktop (OVD) Requirement Analysis Document

Submitted By:

Mark R Beussink

Andrew Cowden

Justin Sieling

Edward J. Byrne

Southern Illinois University, Carbondale

Table of Contents:

Version Control Information:

Version No.	Document Name	Date	Additional Info
1.0	RAD Document	01/20/2019	Initial Requirement Agreement for the OVD Project
2.0	RAD Document Revision 1	02/01/2019	Revised Requirement Agreement based off of Second Requirement Gathering Meeting and Presentation Review

I. Introduction

1. The Purpose of the System

Open Virtual Desktop (OVD) is a portal by which users may access virtual machines. The goal is to allow students and teachers remote access to a similar environment as that which they may find in campus computer lab settings and to allow the administrators of the system ease in managing the virtualized machines.

2. The Scope of the System

The system shall:

- Connect students to virtual desktops and remote applications.
- Manage students access to virtual machines.
- Inform administrators in regards to the current usage and performance of the virtual machines.
- Organize users into groups which determine which virtual machines they can access.
- Initiate automated workflows, i.e. scripted tasks, pertaining to managing virtual machines.

3. The Objectives of the Project

The objective of the project is the development of a web application to connect users via their browsers to virtualized desktops or single applications. This web application will also present administrators with information on the status of the virtual machines and the systems which support these machines as well as aid in operating the system by means of automated workflows. Administrators will also be able to manage groups of users which grant access to particular VMs.

4. Stakeholders of the Project

The project shall have the following as stakeholders:

- Michael Barkdoll
- The developers
- Southern Illinois University
- The end users

5. Definitions, etc.

- CS Dept. — Computer Science Department
- OS — operating system
- Remote Application — a instance of an application installed on a server and accessed over the internet
- SSH — secure shell, a terminal application for accessing remote machines via a terminal through a secure channel
- Virtualbox — a piece of virtualizing software
- Virtual Machine (VM) — a method or instance of abstracting a computer system purely as software

II. The Existing System

The existing system at SIU is allowing students to use computer labs which, depending on the lab and OS, may have some network storage allowing them to retrieve or remotely access their accounts. This remote access is done via SSH, granting access to the student's Linux account. From the CS department's labs, students may also access a Windows VM. Some courses also use the computer labs in the Applied Sciences building. These labs use Windows machines with an installation of Virtualbox and an image of Ubuntu Linux which does not save any changes and cannot be accessed remotely.

1. Organization of the Existing System

The existing system at SIU is allowing students to use computer labs which, depending on the lab and OS, may have some network storage allowing them to retrieve or remotely access their accounts. This remote access is done via SSH, granting access to the student's Linux account. From the CS department's labs, students may also access a Windows VM. Some courses also use the computer labs in the Applied Sciences building. These labs use Windows machines with an installation of Virtualbox and an

image of Ubuntu Linux which does not save any changes and cannot be accessed remotely.

i. Computer Science Department Computer Labs

In the CS Dept. labs:

- The labs are in two rooms next-door to each other.
- One lab holds class meetings; the other only holds classes in rare cases.
- Both labs are open at 8:00 AM (or whenever the office staff remembers to unlock them). The one commonly used for classes is closed at 4:30 PM, and the other lab is open based on a schedule of graduate students tasked with watching over it after normal school hours.
- Students may use either their official (campus wide) account details (known as their Dawg Tag account) or special CS Dept. logins.
- The machines run a version of the Ubuntu Linux distro which uses network storage to synchronize the files between different machines.
- The machines have a Windows 7 image which is accessible via a Virtual Box installation in Ubuntu.
- Students may access their accounts via SSH, which requires either a terminal or software like Putty.
- There is a smaller computer lab featuring several iMacs; however, that lab is outside of the aims of this project.
- Both of the labs are capable of holding class housing chalk boards and necessary presentation equipment. There is no computer for instructors present, however.
- There are currently 46 workstations in these labs, 25 in the class-holding lab and 21 in the other lab.

ii. Applied Science Computer Lab and Subdivisions

In the Applied Science lab:

- The main lab is open depending upon hours set by the IT department.

- The subdivided labs are open when there are classes or if the student works have yet to lock them.
- The Windows accounts are synchronized across the various computer labs operated by the IT department.
- Students use their Dawg Tag account to access the machines.
- The sub labs do have computers for the instructors to use along with necessary presentation equipment.

2. Identified Problems

With the existing system, students access the labs either in person or via remote access protocols, namely SSH. This has introduced the following problems:

- The Shell terminal is not capable to making use of all the software installed on the machines or in not the same way. Students must operate via command which require a degree of experience. Some applications do not have any support for the terminal; others have limited support. E.g. Matlab may run scripts written in it's language and show results from the terminal, but the GUI application is a full IDE (integrated development environment) featuring highlights for code, richer error messages, and visual results of Matlab scripts.
- It is mostly impossible to access the Windows VM remotely.
- The Applied Science labs have no remote access to the machines in any way.
- The Linux VMs on the Applied Science lab's computers will not save files or changes.

i. As-Is Scenario Descriptions

Scenario Name	Access Linux

Participating Actor Instances	Thom (Student)
Flow of Events	1. Thom is in a course with a lab. 2. To work on coursework, he goes to the lab. 3. He logs into the machine using his CS account credentials 4. The computer shows the student's Linux desktop

Scenario Name	**Access Windows VM**
Participating Actor Instances	Thom (Student)
Flow of Events	1. Thom is in a course with a lab, 2. To work on coursework, he goes to the lab. 3. He logs into the machine using his CS account credentials 4. The computer shows the student's Linux desktop 5. The student runs Virtualbox and selects the Windows 7 VM 6. Virtualbox runs the VM image

Scenario Name	**Access Linux during Lab**
Participating Actor Instances	Thom (Student), John (Teaching Assistant)
Flow of Events	1. Thom is in a course with a lab. 2. During the scheduled meeting of the class, the class meets in the Applied Science building. 3. John logs into the teaching computer with his Dawg Tag credentials. 4. Thom logs into his computer with his Dawg Tag credentials. 5. John and Thom both start the Linux VM on their respective machines.

III. The Proposed System

1. Overview

The OVD system will model and implement the following behaviors and constraints:

2. Functional Requirements

1. The system shall connect users to virtual desktops and remote applications.
2. OVD shall permit access to desktops and apps by group membership.
3. OVD shall organize users into groups.
4. OVD shall present users a choice of which image or app to connect.
5. OVD shall present users with information pertaining to the status of the system and other related resources (hardware, VMs, etc.).
6. OVD shall track sessions and inform users of those sessions.
7. OVD shall manage groups of users.
8. OVD shall assign VM images to groups.
9. OVD shall configure the number of hot spare VMs.
10. OVD shall configure the maximum and minimum number of VMs for a group.
11. OVD shall configure the image to user for a VM.
12. OVD shall configure the network settings for VMs.
13. OVD shall configure a group's access to RAM and CPU cores.
14. OVD shall specify the number of remote apps ready for connections.
15. OVD shall permit users to execute server related tasks.
16. OVD shall close connections after a specified amount of time.

17. OVD shall balance the load of VMs on its servers.

3. Nonfunctional Requirements

A nonfunctional requirement is a description of a system's attribute, quality, or other characteristic aside from its functionality. The reader may find the nonfunctional requirements of OVD bellow.

1. Users shall use a web browser to access OVD.
2. OVD will support Google Chrome and Mozilla Firefox.
3. OVD shall be scalable.
4. OVD shall be deployable on CloudStack.

4. Pseudo Requirements

A pseudo requirement is a constraint of the system specified by the client or environment of the system. The following specifications are OVD's pseudo requirements.

1. OVD shall consist of a single page AngularJS web application and a C# ASP.Net MVC application.
2. OVD shall communicate between the frontend and backend via a RESTful API.
3. OVD shall use SIU's LDAP Auth to authenticate users.
4. OVD shall interact with an Apache Guacamole Docker container.
5. OVD shall be developed to prevent future Guacamole changes from breaking OVD.

5. Actor Definitions

An actor is any external entity that interacts with the system. The following are the actors in the proposed system.

- **User** — A user is the standard user of OVD. In most cases, it is a student of the CS department. It will also include professors, instructors, and teaching assistants.
- **Admin** — This is a user whose task is to manage the underlying system and infrastructure.
- **Guacamole** — This is Apache Guacamole, a system which tunnels connections from virtual machines to a web browser via web sockets.
- **LDAP Auth** — SIU provides an authentication service via a Lightweight Directory Access Protocol for Dawg Tags.

6. System Models

i. Functional Model

Scenario Descriptions

A user scenario is describes the experience and flow of a user engaging with the system. Bellow are the anticipated scenarios users will encounter.

Scenario Name	**Login**
Participating Actor Instance	Shaun (student)
Flow of Events	1. Shaun uses his browser to access the OVD page. 2. His browser loads the OVD login page. 3. Shaun enters his Dawg Tag and password into the form and presses the login button. 4. Shaun logs in.

Scenario Name	**Login (Failure)**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael misses his morning cup of coffee. 2. Michael uses his browser to access the OVD page. 3. His browser loads the OVD login page. 4. Michael, in his uncaffeinated state, enters the wrong credentials. 5. The OVD page displays the error message “Invalid username or password.”

Scenario Name	**Use a VM**
Participating Actor Instance	Shaun (Student)
Flow of Events	1. Shaun wishes to run a VM to complete homework. 2. Shaun logs in. 3. His browser shows the list of virtual desktops and remote apps available to him. 4. He clicks on the icon for the VM he needs. 5. His browser loads the Desktop page. 6. He uses the virtual desktop.

Scenario Name	**Use a Remote App**
Participating Actor Instance	Shaun (Student)
Flow of Events	1. Shaun needs to use a remote app on OVD to work on some homework. 2. Shaun logs in. 3. His browser shows the list of virtual desktops and apps available

	to him. 4. He clicks on the icon for the app he needs. 5. His browser loads the Remote App page. 6. He uses the remote app.

Scenario Name	**Create User Group**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael needs to create a user group for a class. 2. Michael logs onto OVD 3. His browser takes him to the Admin Dashboard page. 4. He clicks on a link to the Groups page. 5. His browser loads the Groups page. 6. He clicks on the Create Group button. 7. His browser loads that page. 8. He types the members Dawg Tags into a form. 9. He selects a VM image or remote app for the group. 10. He clicks on the Save Group button on the bottom of the page. 11. His browser sends a notification saying "Group Saved." 12. His browser redirects to the Admin Dashboard.

Scenario Name	**Create User Group (Unknown Failure)**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael needs to create a user group for a class. 2. Michael logs onto OVD.

	3. His browser takes him to the Admin Dashboard page. 4. He clicks on a link to the Groups page. 5. His browser loads the Groups page. 6. He clicks on the Create Group button. 7. His browser loads that page. 8. He types members' Dawg Tags into a form. 9. He selects a VM image or remote app for the group. 10. He clicks on the Save Group button on the bottom of the page. 11. His browser sends a notification for an unknown error. 12. His browser redirects to the Admin Dashboard.

Scenario Name	**Create User Group (Selection Failure)**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael needs to create a user group for a class. 2. Michael logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on a link to the Groups page. 5. His browser loads the Groups page. 6. He clicks on the Create Group button. 7. His browser loads that page. 8. He types members' Dawg Tags into a form. 9. In a rush, he forgets to select a VM image or remote app for the group. 10. He clicks on the Save Group button on the bottom of the page. 11. An alert appears on the page

	informing him that he needs to choose an image for the group.

Scenario Name	**Delete User Group**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. At the end of the semester, Michael needs to delete a user group. 2. Michael logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on a link to the Groups page. 5. His browser loads the Groups page. 6. From a list of existing groups, he clicks on the Delete Group button next to the particular group. 7. His browser sends an alert asking if he wishes to continue. 8. He confirms the action. 9. He receives a notification that the group was deleted successfully. 10. The group is removed from that pages list of existing groups.

Scenario Name	**Delete User Group (Failure)**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. At the end of the semester, Michael needs to delete a user group. 2. Michael logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on a link to the Groups page. 5. His browser loads the Groups

	page. 6. From a list of existing groups, he clicks on the Delete Group button next to the particular group. 7. His browser sends an alert asking if he wishes to continue. 8. He confirms the action. 9. He receives a notification that an error occurred.

Scenario Name	**Add User to Existing Group**
Participating Actor Instance	Taylor (Administrator)
Flow of Events	1. Taylor receives an email informing him that a student has enrolled late in a course and needs to be added to a group. 2. Taylor logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on the link to the Groups page. 5. His browser loads the Groups page. 6. From a list of existing groups, he clicks on the Add User button next to a particular group. 7. His browser navigates to the Add User page. 8. He types in the member's Dawg Tag. 9. He clicks the Save Group button. 10. His browser shows a notification that the save was successful. 11. His browser navigates back to the Groups page.

Scenario Name	**Add User to Existing Group (Failure)**
Participating Actor Instance	Taylor (Administrator)

Flow of Events	1. Taylor receives an email informing him that a student has enrolled late in a course and needs to be added to a group. 2. Taylor logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on the link to the Groups page. 5. His browser loads the Groups page. 6. From a list of existing groups, he clicks on the Add User button next to a particular group. 7. His browser navigates to the Add User page. 8. He types in the member's Dawg Tag. 9. He clicks the Save Group button. 10. His browser shows a notification that the save was not successful. 11. His browser navigates back to the Groups page.

Scenario Name	**Remove User from Existing Group**
Participating Actor Instance	Taylor (Administrator)
Flow of Events	1. Taylor receives an email informing him that a student has dropped a course and ought to be removed a group. 2. Taylor logs onto OVD. 3. His browser takes him to the Admin Dashboard page. 4. He clicks on the link to the Groups page. 5. His browser loads the Groups page. 6. From a list of existing groups, he clicks on the Remove User button next to a particular group. 7. His browser navigates to the

	Remove User page. 8. He selects the user from a list of users in the group. 9. He clicks the Save Group button. 10. His browser shows a notification that the save was successful. 11. His browser navigates back to the Groups page.

Scenario Name	**Remove User from Existing Group (Failure)**
Participating Actor Instance	Taylor (Administrator)
Flow of Events	1. Taylor receives an email informing him that a student has dropped a course and ought to be removed a group 2. Taylor logs onto OVD 3. His browser takes him to the Admin Dashboard page 4. He clicks on the link to the Groups page 5. His browser loads the Groups page 6. From a list of existing groups, he clicks on the Remove User button next to a particular group 7. His browser navigates to the Remove User page 8. He selects the user from a list of users in the group 9. He clicks the Save Group button 10. His browser shows a notification that the save was unsuccessful 11. His browser navigates back to the Groups page

Scenario Name	**View System Status**
Participating Actor Instance	Michael (Administrator)

Flow of Events	1. Michael is curious about how the OVD system is performing 2. He logs into OVD 3. His browser loads the Admin Dashboard 4. He looks at all the pretty data visualization on the page

Scenario Name	**Configure VM Settings**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael anticipates that a class may put a heavy load on the system and thinks he ought to make some changes to prepare 2. He logs onto OVD 3. His browser loads the Admin Dashboard 4. He clicks the Groups link 5. His browser loads the Groups page 6. From the list of existing groups, he selects the Configure button 7. His browser navigates to the Group Configuration page 8. He uses the form to change various settings for the VM 9. He clicks the Save Changes button 10. His browser shows a notification that the save was successful 11. His browser navigates back to the Groups page

Scenario Name	**Configure VM Settings (Failure)**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael anticipates that a class may put a heavy load on the system and thinks he ought to make some changes to prepare

	2. He logs onto OVD 3. His browser loads the Admin Dashboard 4. He clicks the Groups link 5. His browser loads the Groups page 6. From the list of existing groups, he selects the Configure button 7. His browser navigates to the Group Configuration page 8. He uses the form to change various settings for the VM 9. He clicks the Save Changes button 10. His browser shows a notification that the save was unsuccessful 11. His browser navigates back to the Groups page

Scenario Name	**Execute Workflows**
Participating Actor Instance	Michael (Administrator)
Flow of Events	1. Michael wants to run one of the automated workflows 2. Michael logs into OVD 3. His browser loads the Admin Dashboard 4. He clicks the Workflows link 5. His browser goes to the Workflows page 6. He chooses a workflow listed and inserts parameters into a form 7. His browser shows the results of the workflow

ii. Use Case Model

Use Case Number	**01**

Use Case Name	**Login**
Description	This will grant users access to OVD.
Actors	Users, Admin, LDAP Auth
Entry Conditions	1. User has Dawg Tag credentials 2. User is on OVD page
Flow of Events	1. User fills in Dawg Tag credentials 2. OVD checks that user is not admin 3. OVD sends credentials to LDAP 4. LDAP sends response for authorization 5. User logs into OVD
Alternative Flow	**Admin Login** 1. User fills in admin credentials 2. OVD checks that username is special 3. OVD checks password with stored admin password 4. User logs into OVD
Alternative Flow	**Invalid Login Credentials** 1. User enters incorrect credentials 2. OVD sends credentials to LDAP 3. LDAP sends response rejecting credentials 4. OVD notifies user of event 5. User repeats step 1
Exit Conditions	User is logged in User went to different website User closed browser

Use Case Number	**02**
Use Case Name	**Connect to desktop or single application**
Description	This will present a user with either a

	virtual desktop or remote application
Actors	User, Guacamole
Entry Conditions	1. User is logged in
Flow of Events	1. OVD loads lists of VMs and Apps user has access to 2. User clicks on icon for the VM to run 3. OVD sends connection request to Guacamole 4. Guacamole sends response to OVD thru Guacamole HTTP protocol 5. SPA navigates to desktop view
Alternative Flow	1. OVD loads lists of VMs and Apps user has access to 2. User clicks on icon for app to run 3. OVD sends connection request to Guacamole 4. Guacamole sends response to OVD thru Guacamole HTTP protocol 5. SPA navigates to app view
Exit Conditions	User accessed desktop User accessed single app User navigated to different component of OVD User went to different website User closed browser

Use Case Number	**03**
Use Case Name	**Create User Group**
Description	This will create a connection group
Actors	Admin, Guacamole
Entry Conditions	1. User is logged in 2. User has navigated to Groups page

Flow of Events	1. OVD presents user a page with a list of groups 2. User clicks Create Group button 3. User enters information into form 4. User clicks Save Group button 5. OVD stores group information 6. OVD notifies Guacamole of Connection Group 7. Group is created successfully 8. OVD notifies user of success 9. User returns to Groups page
Alternative Flow	**User Cancel Flow** 1. User clicks Cancel button 2. User is returned to Groups page
Alternative Flow	**Incomplete Form Flow** 1. User does not include valid or complete information 2. OVD deactivates Save Group button 3. User satisfies form requirements, continue to step 4
Alternative Flow	**Unknown Error Flow** 1. OVD encounters error when saving 2. OVD notifies user of error 3. OVD does not save group 4. OVD redirects user to Groups page
Exit Conditions	User closed browser User went to a different website User navigated to different component User Group was created successfully User Group was not created successfully

Use Case Number	**04**
Use Case Name	**Delete User Group**
Description	This will delete a connection Group

Actors	Admin, Guacamole
Entry Conditions	1. Admin is logged in 2. Admin has navigated Groups page
Flow of Events	1. OVD presents admin a page with a list of groups 2. User clicks Delete Group button next to the particular Group 3. OVD alerts user that this action is permanent 4. User confirms deletion 5. OVD notifies Guacamole of Connection Group deletion 6. Group is deleted successfully 7. OVD notifies user of success
Alternative Flow	1. User selects cancel in dialog 2. OVD does not delete group 3. OVD notifies user of cancelation
Exit Conditions	User closed browser User went to a different website User navigated to different component User Group was deleted successfully User Group was not deleted

Use Case Number	**05**
Use Case Name	**Add User to Group**
Description	This will add users to a Group
Actors	Admin
Entry Conditions	1. User is logged in 2. User is on Groups page
Flow of Events	1. User clicks Add User button next to the particular Group 2. OVD presents user with form for members 3. User adds Dawg Tags to list 4. User clicks Save Changes button

	5. OVD adds users to Group 6. OVD saves group changes 7. OVD navigates back to Groups page
Alternative Flow	**Cancel Add User** 1. User clicks Cancel button 2. OVD navigates to Groups page
Alternative Flow	**Save Error Occurs** 1. OVD encounters error when saving changes 2. OVD does not save changes 3. OVD notifies user of error event
Exit Conditions	User closed browser User went to a different website User navigated to different component Users were added to Group successfully Users were not added to Group successfully

Use Case Number	**06**
Use Case Name	**Remove User from Group**
Description	This will remove users from a Group
Actors	Admin
Entry Conditions	1. User is logged in 2. User is on Groups page
Flow of Events	1. User clicks Add User button next to the particular Group 2. OVD presents user with a list of users 3. User selects users from user list 4. OVD removes users from Group 5. User clicks Save Changes button 6. OVD saves group changes 7. OVD navigates back to Groups page

Alternative Flow	**Cancel Remove User** 1. User clicks Cancel button 2. OVD navigates to Groups page
Alternative Flow	**Save Error Occurs** 1. OVD encounters error when saving changes 2. OVD does not save changes 3. OVD notifies user of error event
Exit Conditions	User closed browser User went to a different website User navigated to different component Users were removed from Group successfully Users were not removed from Group successfully

Use Case Number	**07**
Use Case Name	**View System Stats**
Description	This will show system stats
Actors	Admin
Entry Conditions	1. This will show system stats 2. User is on Admin Dashboard page
Flow of Events	1. OVD sends request for system status to host server 2. Host Server provides stats 3. OVD presents stats to user
Alternative Flow	1. User does not look at screen
Exit Conditions	User closed browser User went to a different website User navigated to different component

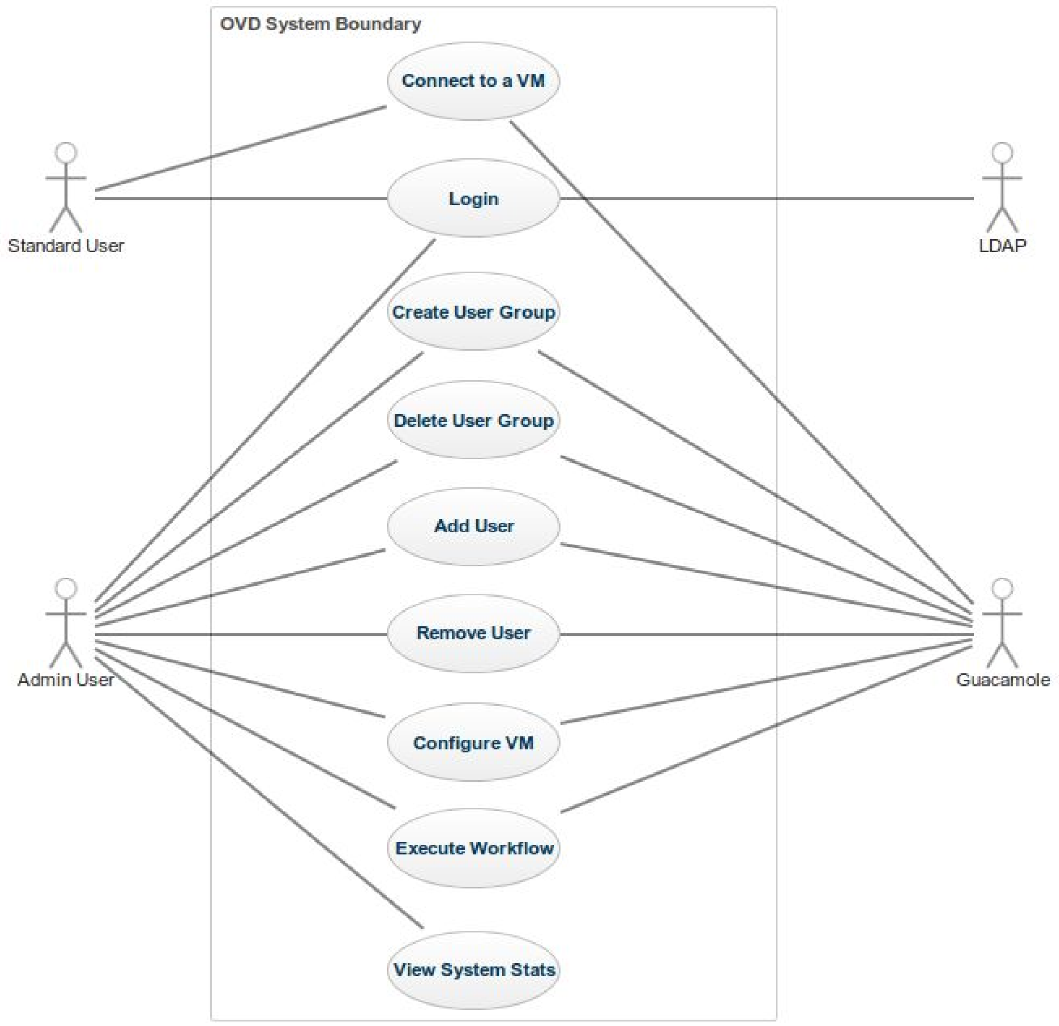
Use Case Number	**08**
Use Case Name	**Configure VM**

Description	This will configure the settings of a VM
Actors	Admin, Guacamole
Entry Conditions	1. User is logged in 2. User is on Groups page
Flow of Events	1. User clicks \`Configure\` button next to the particular Group 2. Configure form opens for corresponding Group 3. User uses form to change various settings for VM 4. User clicks \`Save Changes\` button 5. User returns to Groups page
Alternative Flow	1. User clicks Cancel button 2. User is returned to Groups page
Exit Conditions	User closed browser User went to a different website User navigated to different component VM settings for Group were updated successfully

Use Case Number	**09**
Use Case Name	**Execute Automated Workflow**
Description	This will execute administrative workflow
Actors	Admin
Entry Conditions	1. User is logged in 2. User is on Workflows page
Flow of Events	1. OVD presents user with list of workflows 2. User selects workflow to perform 3. OVD presents user with form for parameters for workflow 4. User provides parameters 5. User clicks Execute workflow button

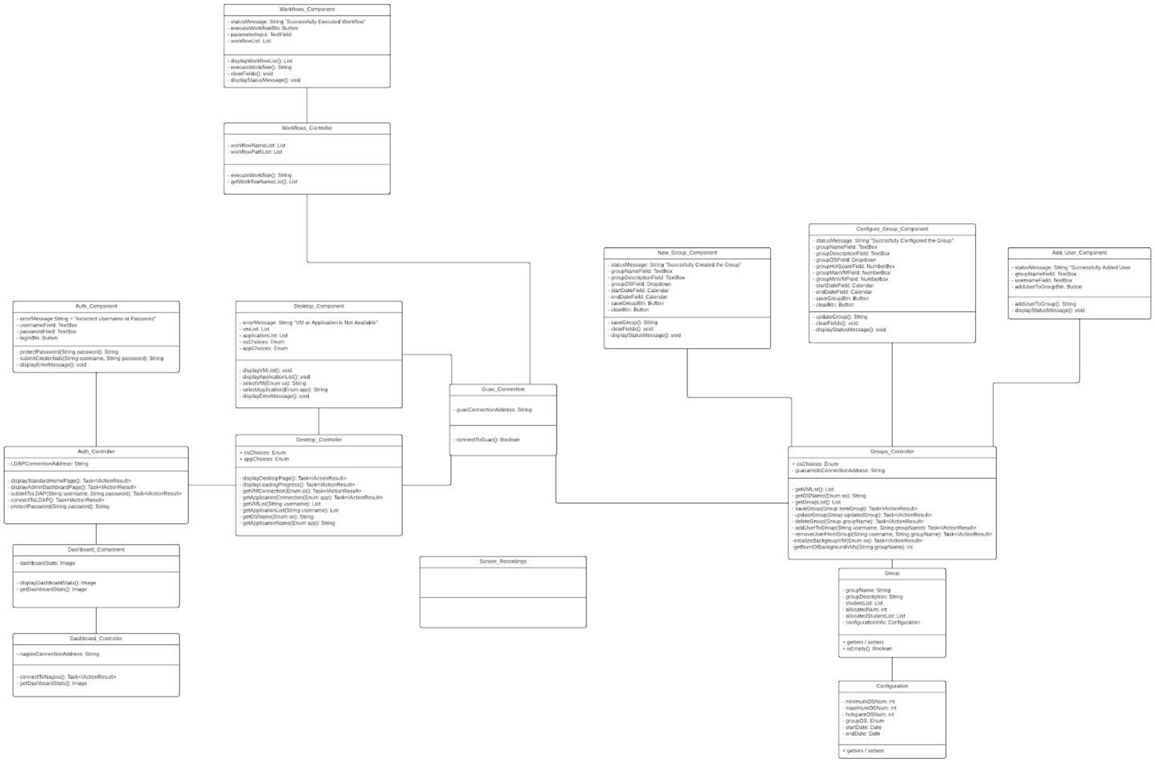
	6. OVD presents user dialog to confirm execution 7. User confirms 8. OVD navigates to Workflows page 9. OVD sends execution request to Host Server 10. Host Server executes workflow 11. Host Server reports execution status to OVD 12. OVD stores workflow report information 13. OVD informs user of successful execution 14. OVD navigates to Workflows page
Alternative Flow	**User Cancels Execution** 1. User cancels 2. OVD does not send request to Host Server 3. OVD navigates to Workflows page
Alternative Flow	**Workflow Error Occurs** 1. Host Server encountered error during execution 2. Host Server notifies OVD of execution error 3. OVD notifies user of execution error
Exit Conditions	User closed browser User went to a different website User navigated to different component Host Server executed workflow Host Server did not execute workflow

Use Case Diagram



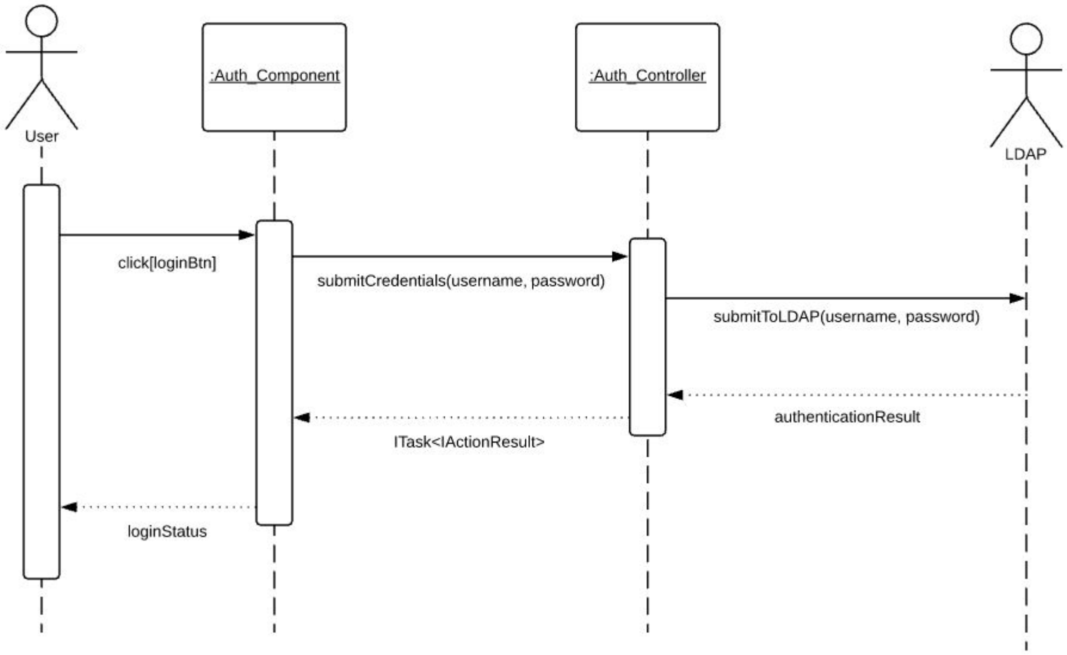
iii. Object Model

Class Diagram

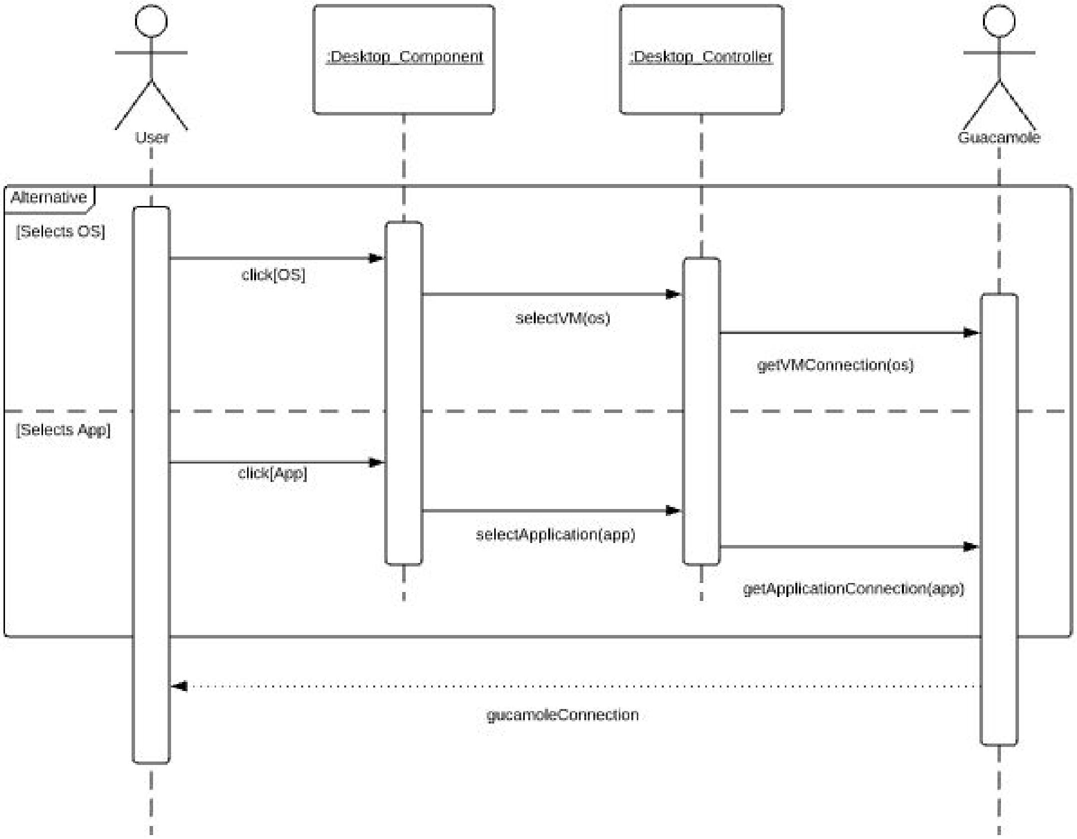

Workflows_Component
- statusMessage: String "Successfully Executed Workflow"
- executeWorkflowBtn: Button
- parameterInput: TextField
- workflowList: List
- displayWorkflowList(): List
- executeWorkflow(): String
- clearFields(): void
- displayStatusMessage(): void
Workflows_Controller
- workflowNameList: List
- workflowPathList: List
- executeWorkflow(): String
- getWorkflowNameList(): List
Auth_Component
- errorMessage:String = "Incorrect Username or Password"
- usernameField: TextBox
- passwordFiled: TextBox
- loginBtn: Button
- protectPassword(String password): String
- submitCredentials(String username, String password): String
- displayErrorMessage(): void
Auth_Controller
- LDAPConnectionAddress: String
- displayStandardHomePage(): Task<IActionResult>
- displayAdminDashboardPage(): Task<IActionResult>
- submitToLDAP(String username, String password): Task<IActionResult>
- connectToLDAP(): Task<IActionResult>
- protectPassword(String password): String
Dashboard_Component
- dashboardStats: Image
- displayDashboardStats(): Image
- getDashboardStats(): Image
Dashboard_Controller
- nagiosConnectionAddress: String
- connectToNagios(): Task<IActionResult>
- getDashboardStats(): Image
Desktop_Component
- errorMessage: String "VM or Application is Not Available"
- vmList: List
- applicationList: List
- osChoices: Enum
- appChoices: Enum
- displayVMList(): void
- displayApplicationList(): void
- selectVM(Enum os): String
- selectApplication(Enum app): String
- displayErrorMessage(): void
Desktop_Controller
+ osChoices: Enum
+ appChoices: Enum
- displayDesktopPage(): Task<IActionResult>
- displayLoadingProgress(): Task<IActionResult>
- getVMConnection(Enum os): Task<IActionResult>
- getApplicationConnection(Enum app): Task<IActionResult>
- getVMList(String username): List
- getApplicationList(String username): List
- getOSName(Enum os): String
- getApplicationName(Enum app): String
Guac_Connection
- guacConnectionAddress: String
- connectToGuac(): Boolean
Screen_Recordings
New_Group_Component
- statusMessage: String "Succssfully Created the Group"
- groupNameField: TextBox
- groupDescriptionField: TextBox
- groupOSField: Dropdown
- startDateField: Calendar
- endDateField: Calendar
- saveGroupBtn: Button
- clearBtn: Button
- saveGroup(): String
- clearFields(): void
- displayStatusMessage(): void
Configure_Group_Component
- statusMessage: String "Succssfully Configured the Group"
- groupNameField: TextBox
- groupDescriptionField: TextBox
- groupOSField: Dropdown
- groupHotSpareField: NumberBox
- groupMaxVMField: NumberBox
- groupMinVMField: Numberbox
- startDateField: Calendar
- endDateField: Calendar
- saveGroupBtn: Button
- clearBtn: Button
- updateGroup(): String
- clearFields(): void
- displayStatusMessage(): void
Add_User_Component
- statusMessage: String "Successfully Added User
- groupNameField: TextBox
- usernameField: TextBox
- addUserToGroupBtn: Button
- addUserToGroup(): String
- displayStatusMessage(): void
Groups_Controller
+ osChoices: Enum
- guacamoleConnectionAddress: String
- getVMList(): List
- getOSName(Enum os): String
- getGroupList(): List
- saveGroup(Group newGroup): Task<IActionResult>
- updateGroup(Group updatedGroup): Task<IActionResult>
- deleteGroup(Group groupName): Task<IActionResult>
- addUserToGroup(String username, String groupName): Task<IActionResult>
- removeUserFromGroup(String username, String groupName): Task<IActionResult>
-initializeBackgroupVM(Enum os): Task<IActionResult>
-getNumOfBackgroundVMs(String groupName): int
Group
- groupName: String
- groupDescription: String
- studentList: List
- allocatedNum: int
- allocatedStudentList: List
- configurationInfo: Configuration
+ getters / setters
+ isEmpty(): Boolean
Configuration
- minimumOSNum: int
- maximumOSNum: int
- hotspareOSNum: int
- groupOS: Enum
- startDate: Date
- endDate: Date
+ getters / setters

iv. Dynamic Models

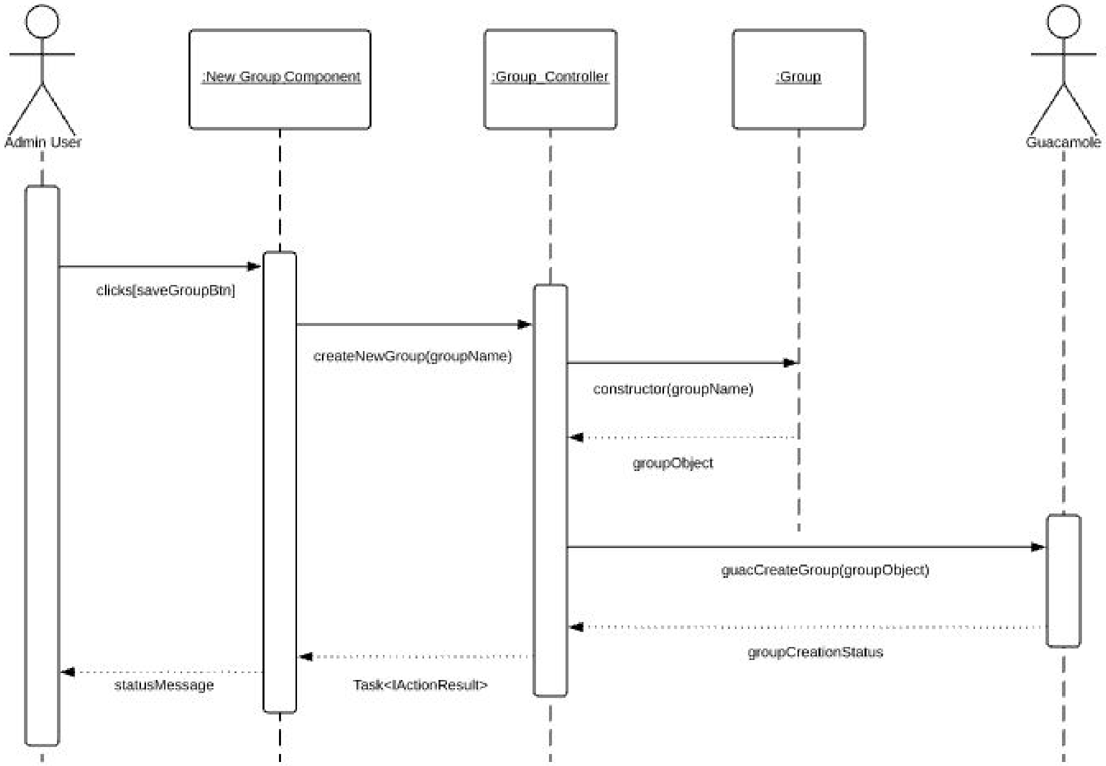
Sequence Diagrams



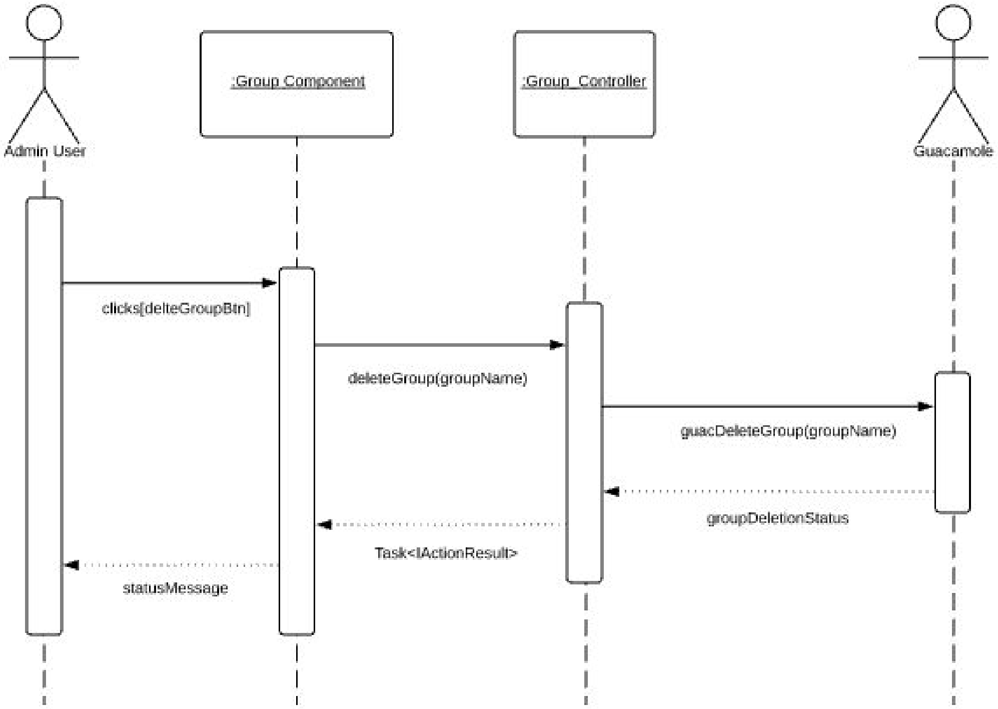
Sequence Diagram 01



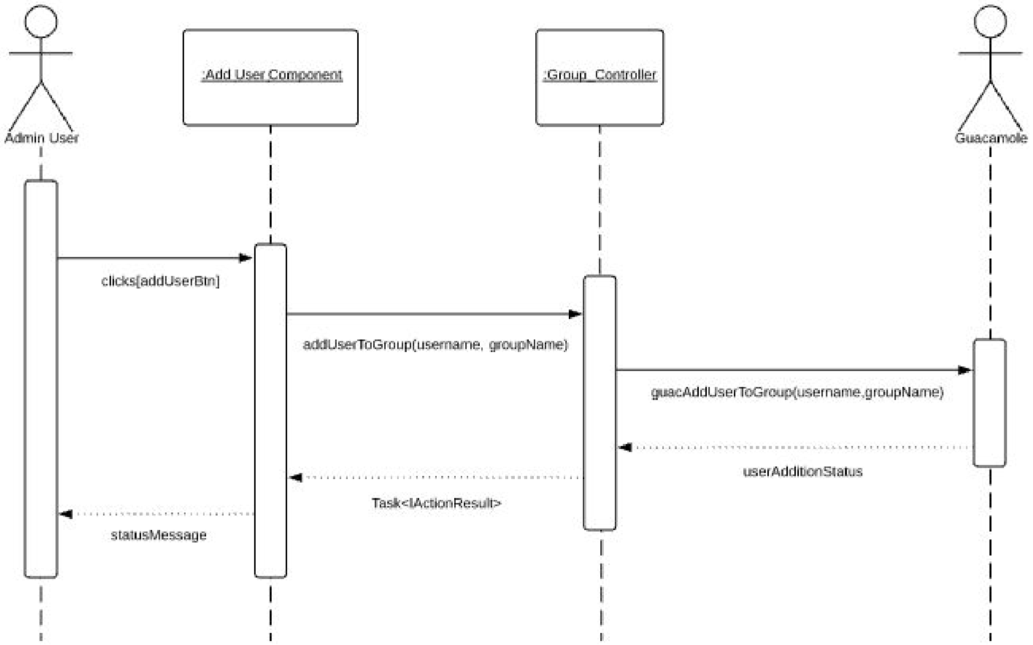
Sequence Diagram 02



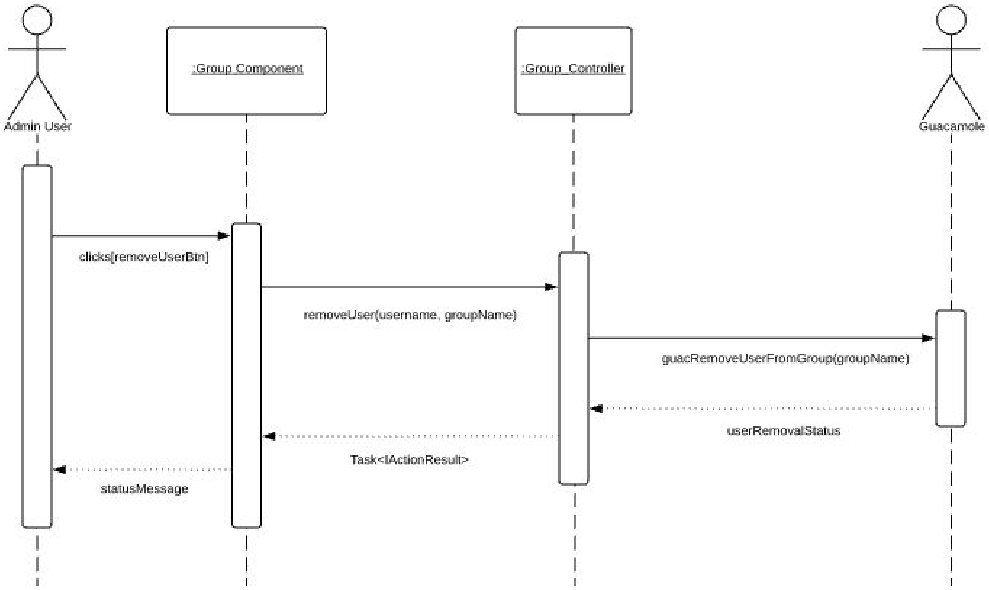
Sequence Diagram 03



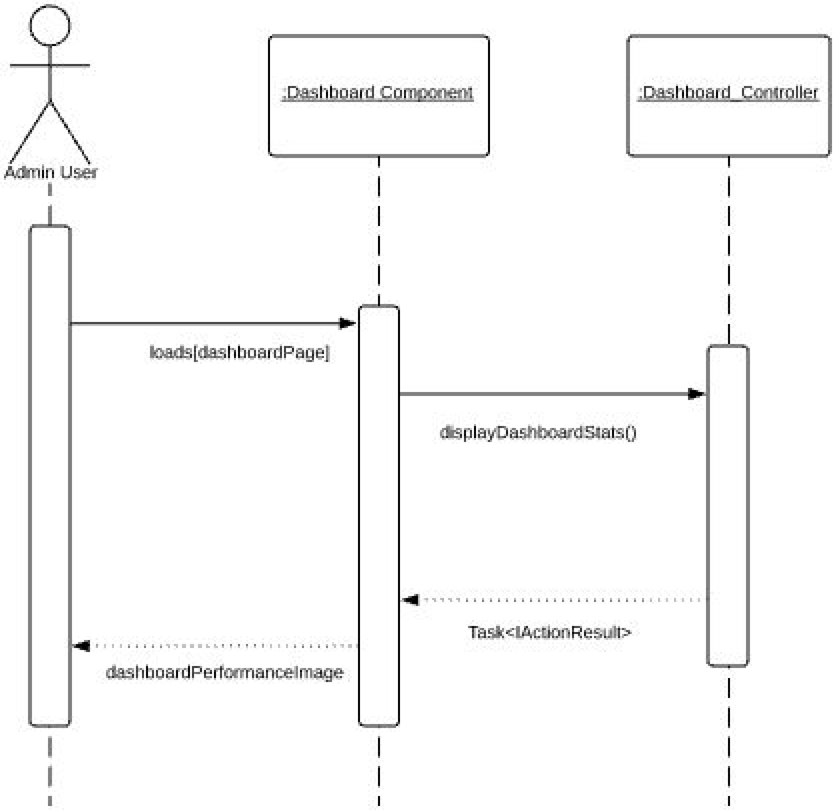
Sequence Diagram 04



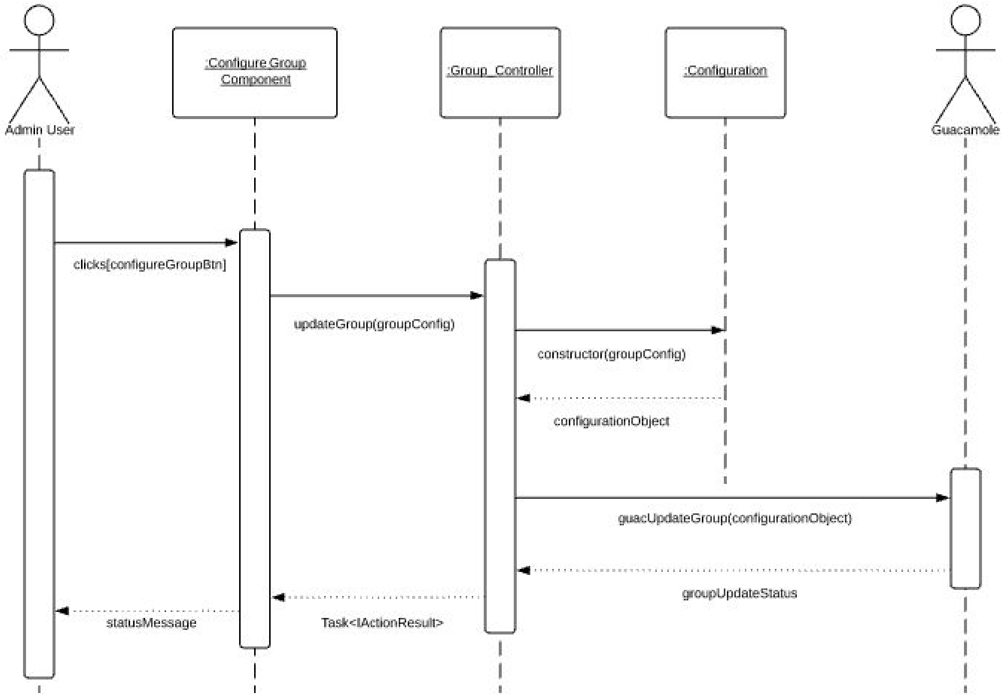
Sequence Diagram 05



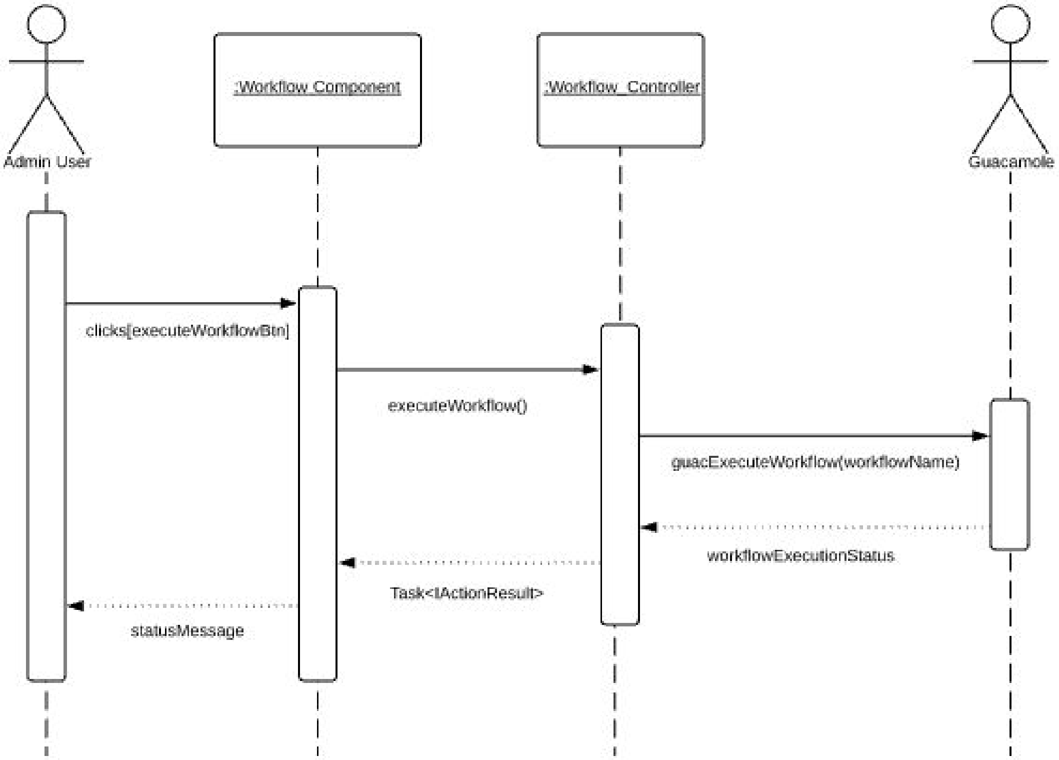
Sequence Diagram 06



Sequence Diagram 07

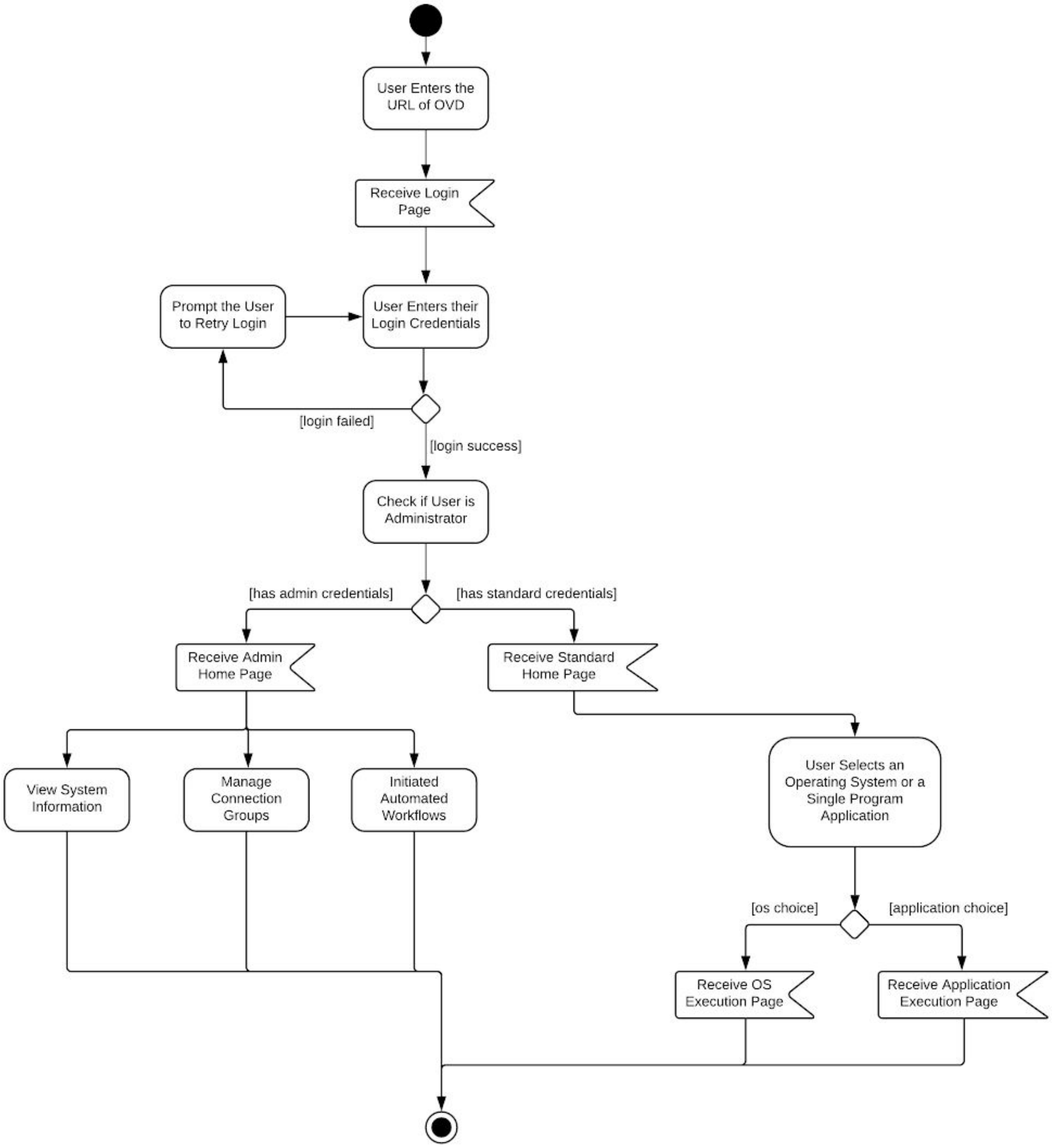


Sequence Diagram 08

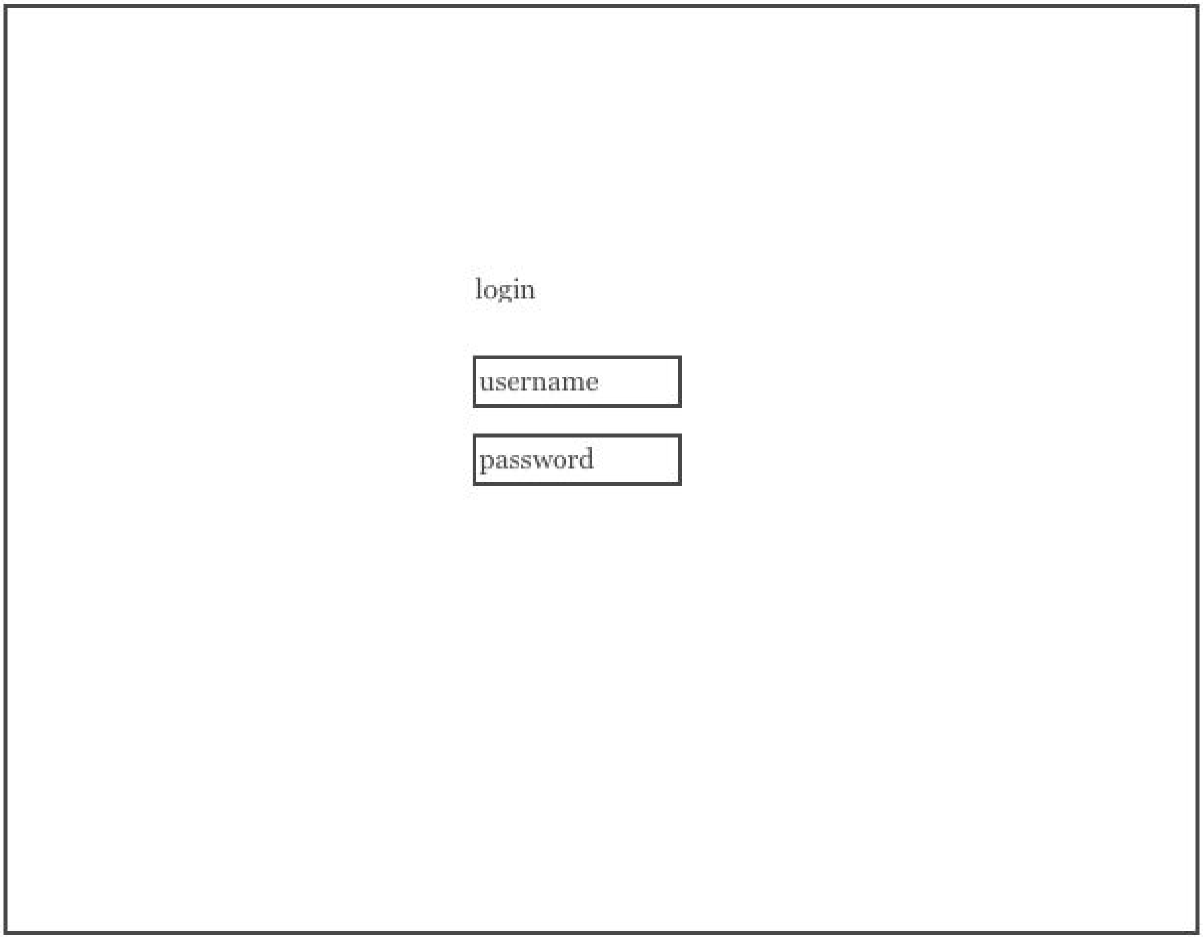


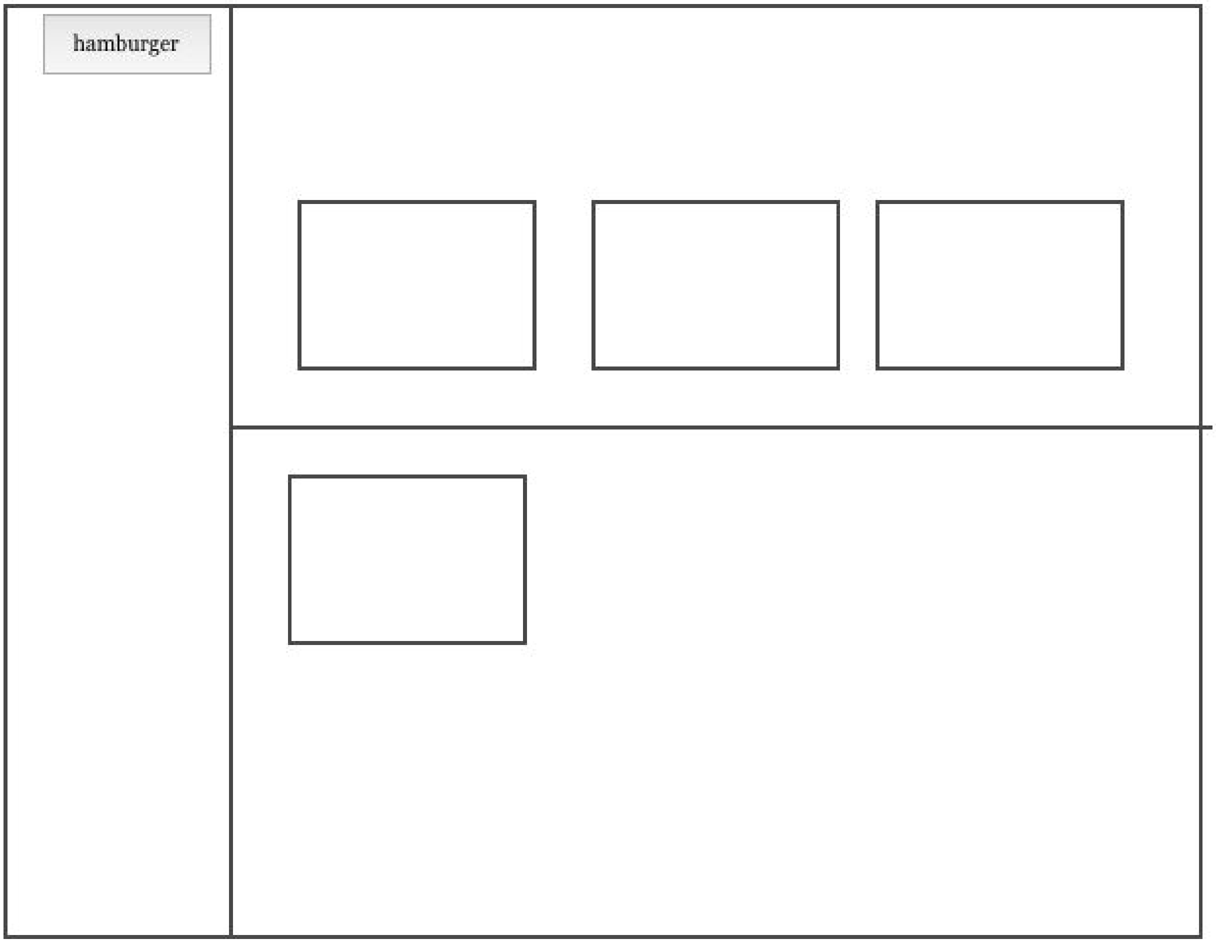
Sequence Diagram 09

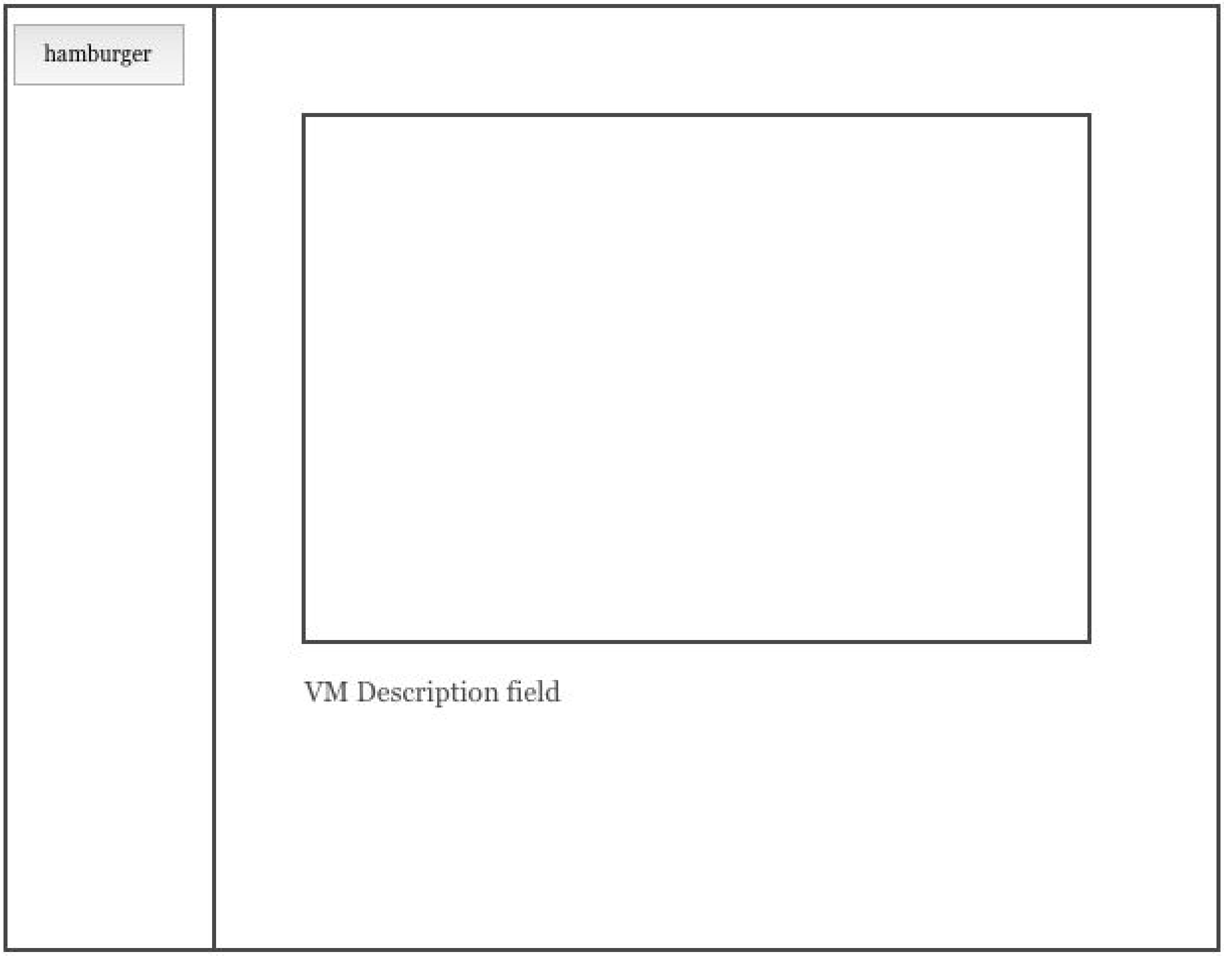
Activity Diagram

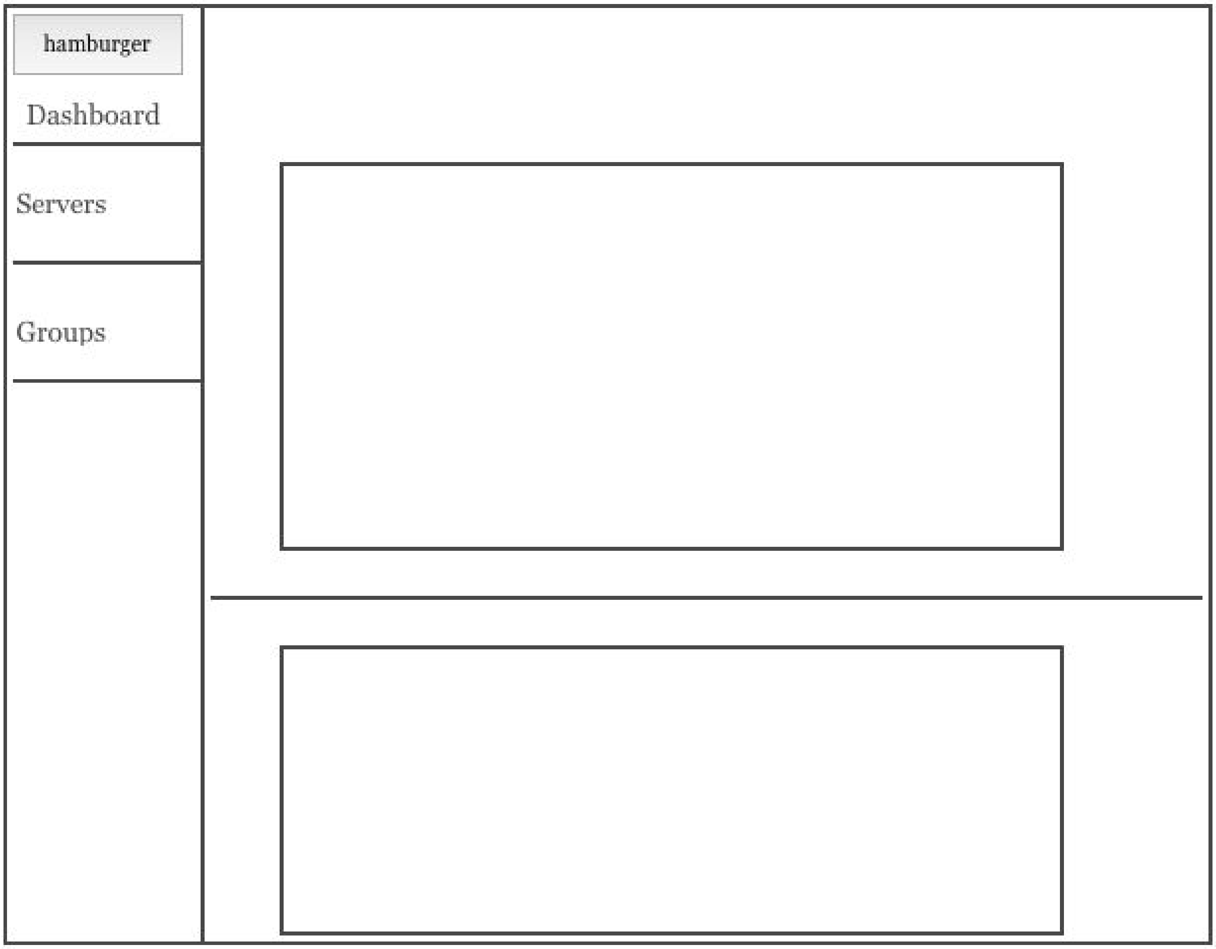

User Enters the URL of OVD
Receive Login Page
Prompt the User to Retry Login
User Enters their Login Credentials
[login failed]
[login success]
Check if User is Administrator
[has admin credentials]
[has standard credentials]
Receive Admin Home Page
Receive Standard Home Page
View System Information
Manage Connection Groups
Initiated Automated Workflows
User Selects an Operating System or a Single Program Application
[os choice]
[application choice]
Receive OS Execution Page
Receive Application Execution Page

v. User Interface




hamburger


hamburger
VM Description field


hamburger
Dashboard
Servers
Groups