Open Virtual Desktop (OVD) System Design Document

Submitted By:

Mark R Beussink

Andrew Cowden

Justin Sieling

Edward J. Byrne

Southern Illinois University, Carbondale

Table of Contents

I. Introduction

It is now time for the project planning to shift from analyzing the specifications made by the customer involving various constraints and expected behaviors to focusing on the elements necessary to construct the proposed system. The previous requirements analysis sought to define the system using models as lenses to inspect the layers of nonfunctional requirements and constraints, behavioural scenarios, abstract entities, and finally sequences of events from the interactions of the constituent objects. The first step of system implementation is the modeling of the system's design. System design involves the knolling of system into data structures, software and hardware components, and subsystems; this also involves defining the design goals of the system. This document shall transform the initial models into the system design model.

1. Overview

The goal of this document is to lay forth the plans by which the developers may construct the system. First the document will refresh the reader on terminology and the nature of the proposed system. This document shall then define the goals of the system's design and criteria by which one may judge the accomplishment of the goals. Finally, this document shall prescribe the architecture of the proposed solution for the sake of development.

2. Definitions, etc.

Apache Guacamole — a open source project which allows OVD to connect a web browser to a VM

Application Programming Interface (API) — an exposed set of behaviors or data structures which allow for the communication of external software systems. In the context of this document an API is a *Web* API, a set of exposed behavioral protocols accessed via the Web.

Computer Science Department (CS Dept.) — the computer science department at SIU, whose students and faculty are the main users of the system

Docker — a software that allows an entire OS to be containerized and portable as an execution environment

Free and open source-software (FOSS) — software which has a license allowing the free use and distribution of it

Information Technology (IT) — the use of computer systems and networks in an organizational setting for the sake of managing and processing information and the maintenance of such systems

Open Virtual Desktop (OVD) — the subject of this document, a software system

RESTful API — a protocol allowing one application to interact with another as a service via a set of standard verbs

Single Page Application (SPA) — a style of web-based application which dynamically loads content by rewriting the loaded page instead of loading dynamic pages from a server

Southern Illinois University (SIU)

Virtual Machine (VM) — a method or instance of abstracting a computer system purely as software

3. References

For the reader to have a fuller understanding of this document and the proposal which follows, the authors of this document ask the reader to refer to the requirements analysis document which may be found on the project's website (cs.siu.edu/~ovd).

4. System Overview

OVD is a web application which connects students and teachers in the CS Dept. of SIU to VMs. These VMs augment the existing computer lab infrastructure. The ultimate purpose of the system is to allow students access to a similar computing environment as can be found in the various computer labs of which the department makes use. The department's IT administration shall also administer OVD and shall make use of SIU provided IT services.

II. Design Goals

The design goals of a system identify the priorities its designers have while planning the system. These goals reflect the customer's expectations in terms of the qualities of the system.

1. Criteria

Performance

Efficiency	The customer and developers have envisioned two scenarios the system will go between. There will be periods of heavy usage when a particular class of 20 to 30 students will use the system simultaneously because they are meeting in a lab, or there will be times when users are using the system sporadically. The system must be able to operate in both of these conditions with the same efficiency.
Responsiveness	There is a transition from the system's standard SPA to Guacamole's VM view. During that transition and while the VM is connected the user has no way of knowing if the system is working or has hung. The system must keep that period of time as short as possible.
Portability	One underlying requirement of the system is the use of a RESTful API and Docker. Both of these allow the system to be

portable.

Dependability

Robustness — Parts of the functionality take user input and use it for automated workflows. In this case erroneous input would trigger fatal errors. The system must be able to recover from such errors and deliver users alerts.

Reliability — Much of this system's functionality has affects not immediately visible to the user. For instance, the admin will change a configuration in anticipation of an increased demand. In such instances, the system must maintain the user's faith.

Security — The system must prevent the end user from any malicious activity.

Cost

Development of the system will consist entirely of FOSS. The only costs associated with the system will be the necessary hardware to operate the system and the paycheck of the administrators.

Maintenance

Extensibility — As the needs of the department change, needs for the system to extend with new functionality will arise. The developers must construct the system in a way that allows ease in future development.

Modifiability	The proposed system must consist of separable subsystems with standard communication protocols. With this maintained, the future developers of the system may make changes to a single subsystem without affecting the whole.
Readability	Construction of the system must involve both common patterns in the existing languages and technologies and adequate documentation. This will allow future developers ease in understanding the system.
Forward Compatibility	This system relies on an external, actively developed project from a major FOSS provider. The system must be developed to prevent changes in that technology from breaking the system.

End User Criteria

Utility	The primary end goal of this system is convenience, both for students and teachers in a classroom or off-campus setting and to the administration operating the system. This system must be designed so that it provides utility to the users.
Usability	The user interface of this system must follow common design patterns to ensure that users understand its functionality.

III. Proposed Software Architecture

1. Overview

OVD is a new software solution for the lab environments found at SIU and does not depend on an existing software architecture. Included below are the overarching definitions that describe both the logical and physical software architectures.

Logical Software Architecture

The logical architecture of the OVD system is outlined using the model-view-controller (MVC) architecture. The MVC architecture was chosen as the architectural pattern as it best describes the flow of data throughout our web application. This logical architecture breakdown describes the OVD system independently of the rigid software or hardware requirements and includes only the conceptual layering and flow of information within the system. The OVD system is divided into the the following components which provide the core functionality of the system.

- The login view provides the components which authenticates users of the web app and connects them in a secure manner.

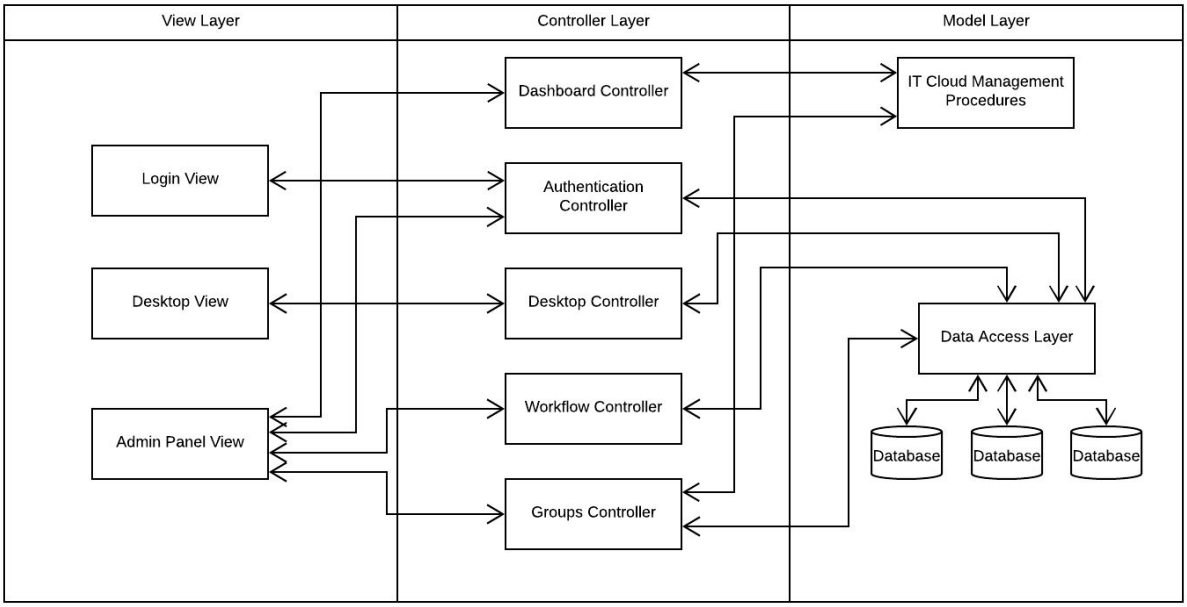
- The desktop view is the collection of components that allow standard users to view and select an option pertaining to the virtual desktop they would like to run.

- The admin panel view contains the primary components that allow users with administrative credentials to configure and view the OVD system from an IT Management standpoint.

- The collection of controllers below serve the purpose of transporting data from the view layer to each respective model layer. Once the data flow has been completed, the controllers will present the next graphical display to the view layers.

- The data access layer and subsequence databases describe the need for data storage and access. These databases will be involved with connecting and communicating with the controller layer components in order to complete the request from the user.

- The IT Cloud Management Procedures layer provides an interface for administrative configurations and dashboards to be updated and executed.



Physical Software Architecture

Within the physical software architecture breakdown, this document shall use software and hardware specific components to describe the different portions of the system. The MVC architecture organizes software into three layers — the model, view and controller layers — and dictates how the three layers interact. The model layer is an abstraction of real world objects, the view layer presents users with information and accepts user input, and the controller layer communicates between the other two layers. The view layer of OVD will consist of AngularJS components. The controller layer will consist of some AngularJS

components along with a DotNet Core backend. These separate systems will communicate via a RESTful API. Finally, the remainder of the backend will comprise the model layer with a connection to a MySQL database providing data persistence.

Many of the important points that can be outlined within our physical software architecture stem from the specific requirements of storing and accessing data. The data access layer will have access to three distinct databases that are used for the authentication and virtual machine connection portions of the OVD system.

- Lightweight Access Directory Protocol (LDAP): This is an external actor within our system but is accessed through the data access layer for the authentication of users.
- Connection Information MySql Relational Database: This persistent database is used for managing general connection information used by the OVD system that is independent of the Guacamole Relational Database.

2. Subsystem Decomposition

In order to better understand the interactions and data flow occurring, the OVD project has been broken down into the following subsystems. Included alongside the subsystem descriptions are an overview of the off-the-shelf technologies.

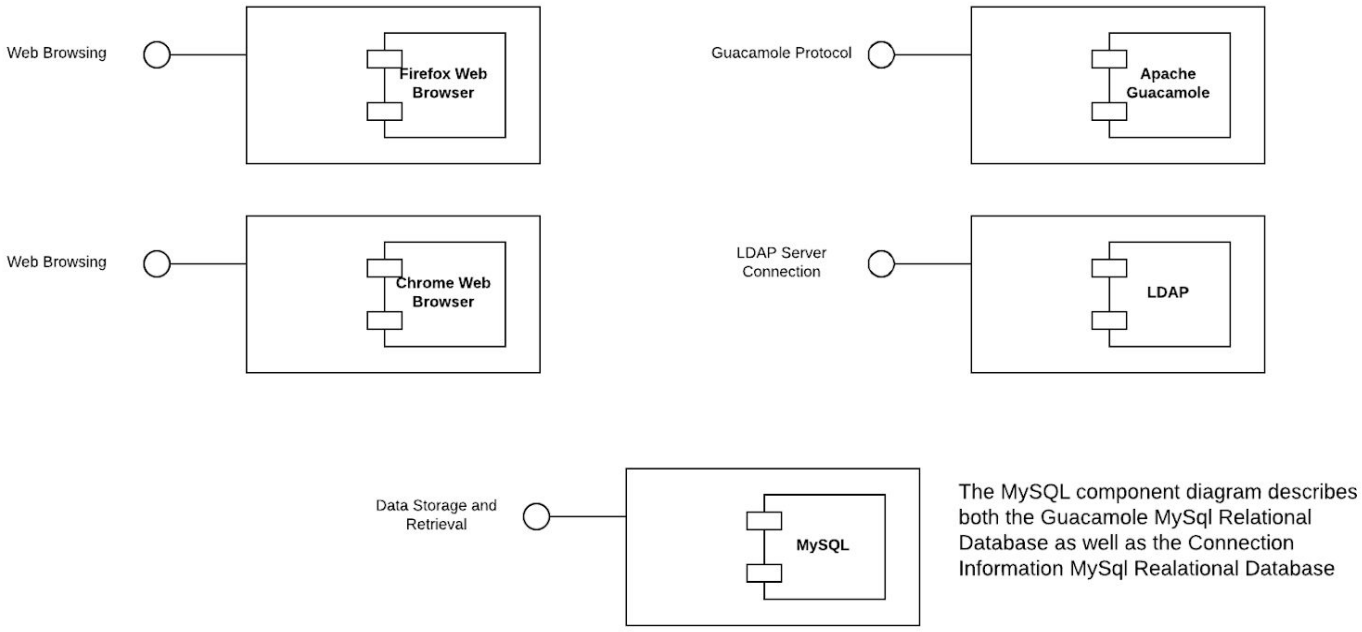
Firefox and Google Chrome Web Browsers: These are the modern browsers that will be directly supported by the web application. A browser is needed to view the OVD web application as the visual components of the application will be communicated to the user within the view layers of the architecture.

Apache Guacamole — a clientless remote desktop gateway that will serve as the primary location for managing and communicating virtual desktops to the end users.

Lightweight Directory Access Protocol (LDAP) — an authentication service offered by SIU's IT department allowing users to log into any SIU service using their campus ID number (a.k.a. DawgTag)

MySQL: This serves as the relational database management system that will assist us in data storage and retrieval methods.

The following component diagrams outline the off-the-shelf component requirements within the OVD system and their included provided interfaces.



Authentication Subsystem

The authentication subsystem will provide the services involved with authenticating a user interacting with the OVD web application. In order to verify the credentials of a user, the authentication subsystem will interact with the off-the-shelf LDAP component. This subsystem's services can be defined as follows:

- Password protection
- Username and password authentication

Established Connection Management Subsystem

This subsystem provides the overarching services that manage the connection information of users currently interacting with the web application as defined below:

- User timeouts
- Monitoring allocated Apache Guacamole resources for a single user
- Disconnecting users

This subsystem depends upon the connection information database subsystem in order to assist in monitoring system utilization for each user.

Dashboard Subsystem

The dashboard subsystem is consists of services that involve presenting users with administrative privileges status information regarding the physical servers the OVD system will be deployed on. The server information will provide information such as RAM and CPU utilization as well as an overview of how many virtual desktops are currently being utilized.

Group Management Subsystem

The group management subsystem consists of the following services which assist in the tasks involved with connection group management for both standard users and users with administrative credentials.

- Creating a new connection group
- Deleting an existing connection group
- Configuring an existing connection group
- Adding users to a connection group
- Removing users from a connection group

Since connection groups are a pivotal area within the OVD web application, storing and accessing connection group information is an important functionality. The group management subsystem connections to both the Guacamole database subsystem as well as the connection information database subsystem.

Workflow Subsystem

The workflow subsystem provides services to administrative users of the OVD web application that allows them to run prewritten scripts for advanced administration of the system. Here admin users can see a list of written workflow scripts, run workflow scripts, and see their corresponding output. The workflow subsystem makes use of the connection information database to retrieve the possible scripts that can be executed. It will also connect to the Guacamole database subsystem in order to see and change Guacamole configuration information.

Desktop Subsystem

The desktop subsystem is the primary subsystem that provides services for establishing a connection to a virtual desktop image as outlined:

- Establish virtual machine connection

- Establish single application program connection
- Get list of virtual machines
- Get list of single application programs

The desktop subsystem requires connections to the Guacamole database subsystem as well as the connection information database subsystem to allow for validation when using the virtual desktops.

Connection Information Database Subsystem

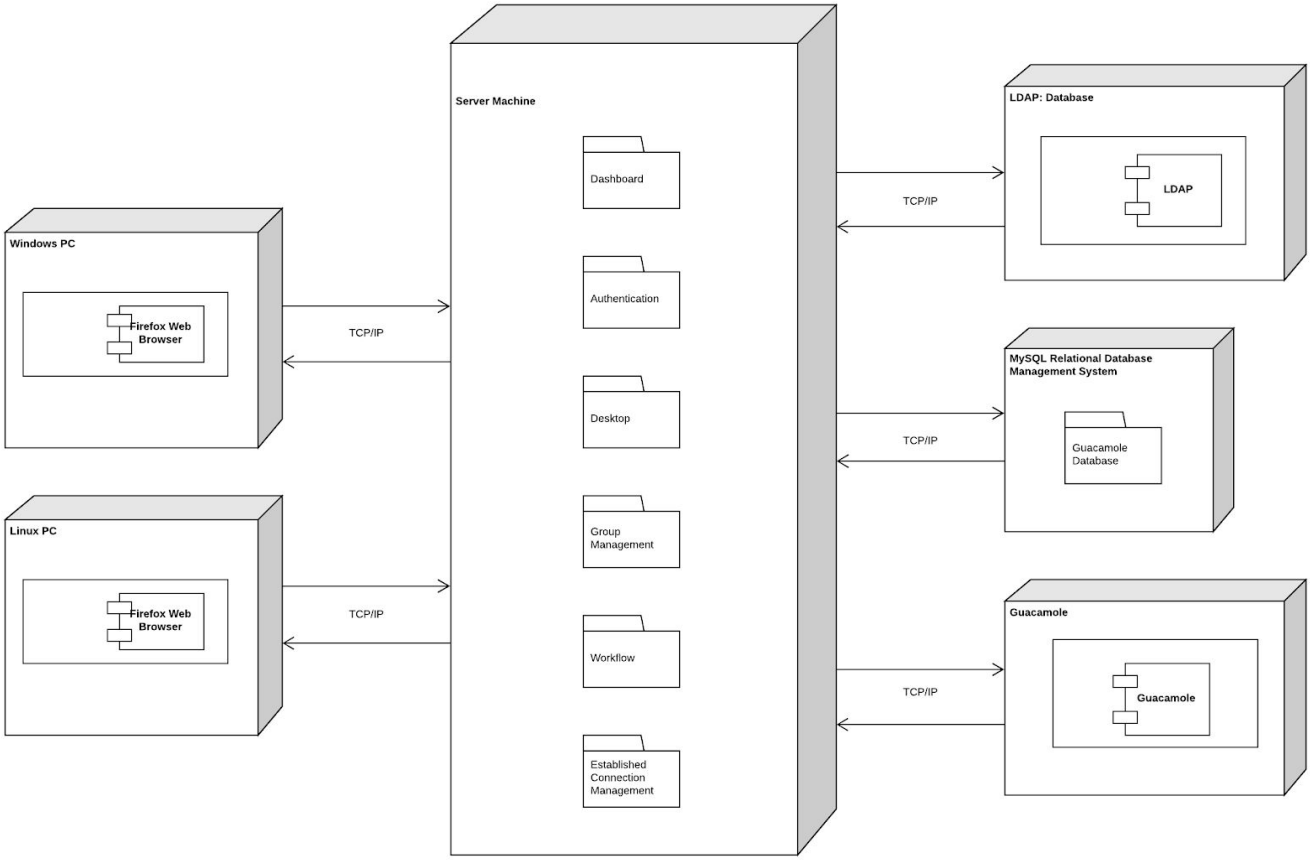
The connection information database subsystem contains the services required to manage configuration information for groups, manage user connections, and provide administrative system information.

3. Hardware/Software Mapping

The corresponding hardware/software mapping diagram associates each of the identified subsystems with a physical location within the infrastructure. The OVD system will consist of client machine and server machine based hardware. Since we are using independent client and server machines, the TCP/IP protocols are used to transfer information between each machine. The Guacamole protocol works over TCP/IP and will assist in the display of virtual machines when requested to the client machines.

The client side of the OVD project contains the off-the-shelf browser components which are required to interact with the web application. The client side machine is where the Angular components of the web application will be accessible. The application services of the system will be contained within the server machine and provide communication to the client side machines as well as the databases involved. These databases will be implemented using

off-the-shelf components such as the MySQL Relational Database Management System and the Lightweight Access Directory Protocol (LDAP).

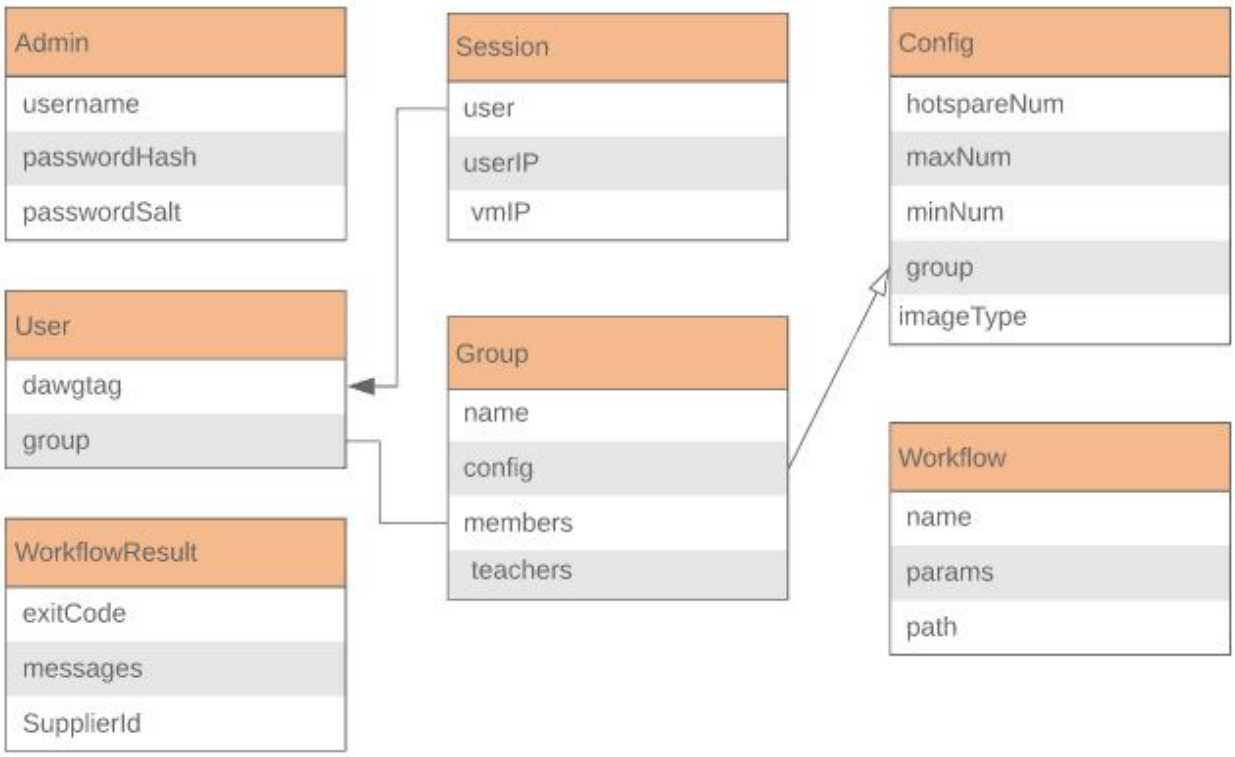


4. Persistent Data Management

In order for the OVD project to function properly, there is a need for persistent data management. To this end, the popular open source relational database management system MySQL is used. The database will contain the user, connection group, workflow, along with IP information for both users and VMs, and active session and system log information and system logs.

Mapping

MySQL uses tables, attributes, and relationships to store and reference data. In order to store the OVD project's data persistently, each data segment is mapped onto an object and stored in a corresponding table, with their attributes being used as columns in said table. Relationships between tables have to be created, and rules set in place to keep data consistent and congruent between tables. The types of relationships that are used among the tables are one-to-one and one-to-many. The following diagram maps the tables and relationships that are currently proposed.



5. Access Control and Security

The access control matrix below serves to outline which actors may utilize the critical resources with the OVD system. The primary actors outlined within this project are standard users, administrative users, Guacamole, and LDAP.

Actor	Access Privileges

Standard Users	- Access to an ssh session with their account - Access to any VMs or Remote Apps available through any group they may belong to
Administrative Users	- Read/Write access to Connection Information Database
Guacamole	- Handles VM access - Controls Internal relational database
LDAP	- Authenticates user login attempts

6. Global Software Control

Since OVD should be allowed to support many students using VM's simultaneously, the entire system will be thread based. OVD will deploy thread locks to avoid any race conditions or other resource hogging issues. Transactions will also be used on databases to ensure that users aren't reading and writing at the same time to create a conflict in database consistency.

7. Boundary Conditions

System initialization begins when the Guacamole, MYSQL, and TomCat components are set up on CloudStack and opened to user requests. Users may then request connection by typing the designated URL into their brower.

System shutdown occurs when either an administrator shuts down the components, the hosting CloudStack is closed to connections, or the hosting Cloudstack is shut down completely.