EQM IT Ticketing System

by

Aaron Kerns and Randal Baker

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Aaron Kerns	4/18/16 Date
Randal Baker	4/18/16 Date
James Scott, Faculty Advisor	4-18-16 Date
Robin Carew, Faculty Advisor	4/18/16 Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2016

EQM IT TICKETING SYSTEM

for

Environmental Quality Management, Inc.

Prepared for:

EQM

Environmental Quality Management, Inc.
1800 Carillon Boulevard
Cincinnati, Ohio 45240-2788

Prepared by:

Aaron Kerns and Randal Baker
Students of
University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology
cech.uc.edu/it

April 2016

CONTENTS

CONTENTS (cont'd)

TABLES

FIGURES

ACRONYMS AND ABBREVIATIONS

AD	Active Directory
APP	Application
ASP	Application Service Provider
DC	Domain Controller
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EQM	Environmental Quality Management, Inc.
IIS	Internet Information Services
iSCSI	Internet Small Computer System Interface
IT	Information Technology
LAN	Local Area Network
SAN	Storage Area Network
SQL	Structured Query Language
SSLVPN	Secure Sockets Layer Virtual Private Network
TB	Terabyte
TDD	Test-driven Development
VPN	Virtual Private Network
WAN	Wide Area Network

ABSTRACT

Environmental Quality Management, Inc. (EQM) is an environmental consulting/engineering company headquartered in Cincinnati, Ohio with locations throughout the United States. EQM faced a lack of organization and structure around technical issues. The lack of infrastructure resulted in a substantial misuse of time and resources. Pre-designed systems are expensive, once one accounts for the cost of product, licensing, maintenance, support, and consulting time. The team of Aaron Kerns and Randal Baker built a customized ticketing system for EQM. After a technical issue occurs, an EQM employee logs in to the Web site. Here the person submits a form and the system determines the priority level of the issue. Information Technology (IT) personnel receive notice of the problem and the priority level. Users are notified via E-mail when their ticket has been completed. Problems that have previously been resolved are automatically referenced to the customized dynamic knowledge base.

1. PROBLEM STATEMENT

1.1 Introduction

Environmental Quality Management, Inc. (EQM), headquartered in Cincinnati, Ohio, is an environmental consulting/engineering company. EQM's clients comprise both government and private sector organizations of all sizes. EQM's employees come from various backgrounds and levels of education, so this application must be easy to use. EQM has numerous small satellite offices throughout the United States, and the structure of the company does not support having IT staff at every office. However, each satellite office must be able to request and have its request resolved quickly in order to avoid the interruption of normal work and emergency responses. EQM must respond to any number of disasters both man-made and natural, and must be able to respond with as little as six-hours' notice, so time is of the essence for IT diagnoses and repairs.

1.2 Project Description

The team designed and implemented a real-time ticketing system with alerts into the business infrastructure of EQM. Our solution, in building a customized ticketing system for EQM, was more beneficial than purchasing a pre-designed system to be implemented into the infrastructure. Pre-designed systems are expensive, once one adds in the cost of product, licensing, maintenance, support, and consulting time. When purchasing a pre-designed system, one often buys more features than are needed, while the software may lack some features that are necessary for everyday production in the business. Our system was cost-effective, as well as customizable to the business and its goals.

1.3 Problem

The problem that EQM faced was the lack of organization and structure around technical issues. Currently, there is a shortage of staff in the IT department. For the past three years there have been two IT employees, but at present there is only one. When technical issues arise, they are communicated to the only staff member via E-mail, phone call, or text, resulting in a substantial waste of time and resources, when time is much more in demand for other issues. On average 50-75% of any day was wasted from poor communication and repetitive technical issues. ***Table 1*** demonstrates how this wasted time translated to lost profit for EQM, based on an average salary of a UC IT Co-op [1].

Table 1. Wasted Time Causes Wasted Money

	Daily	Weekly	Monthly	Annually
50%	4 hours	20 hours	80 hours	1040 hours
75%	6 hours	30 hours	120 hours	1560 hours
$15 / Hour	*($60 – 90)*	*($300 – 450)*	*($1,200 – 1,800)*	*($15,600 – 23,400)*

1.4 Solution

The solution to fix EQM's IT communication problem was designing and implementing a customized ticketing system. The ticketing system features real-time alerts, a dynamic knowledge base, and offers established solutions to reoccurring issues. The ticketing system greatly minimized downtime, improved employee productivity, and saved money for EQM.

1.5 User Profile

1.5.1 Application Title

EQM IT Ticketing System

1.5.2 Potential Users

EQM Employees

1.5.3 Software and Interface Experience

The application consists of an interface for the login page and a custom-made form. The form contains four fields, Product, Type of Issue, Common Solution(s), and Comments. The user will only see the custom-made form after successful login.

1.5.4 Experience with Similar Applications

Currently, EQM employees use several web-based applications on a daily basis. One such software used by all EQM employees is the time-tracking application which is web-based.

1.5.5 Task Experience

A user is required to have minimal knowledge on using a computer and/or smartphone and using a Web browser. A user will be able to submit a ticket without complications that might arise from a lack of computer experience.

1.5.6 Frequency of Use

Anytime an EQM employee has a technical issue

1.5.7 Key Interface Design Requirements That the Profile Suggests

Separate interfaces are used for IT administrators and users. The design is simple to prevent distraction and complicated experiences for the end user.

2. PROJECT MANAGEMENT

2.1 Budget

Table 2 presents the project budget. It presents the real-world costs of $19,349, but totals to $0 as this was our senior project and is not sold, but donated to EQM in exchange for our experience and knowledge.

Table 2. Project Budget

No.	Item	Unit, Hours	Unit Price	Line Item Total
	Networking			
1	Labor	125	$75	$9,375
	Software			
2	Labor	125	75	9,375
3	ReSharper	1	599	599
	Subtotal			19,349
	Total to EQM			**$0**

2.2 Objectives/Deliverables

The project deliverables and deadlines are presented in ***Table 3***.

Table 3. Project Objectives/Deliverables Due Dates

Major Project Milestones (Deliverables)				
Fall of 2015 Milestones				
Pre Planning Milestone	9/20/15		Software Milestone	11/1/15
Pre Setup Milestone	9/27/15		User Interface Milestone	11/22/15
Network Setup Milestone	10/25/15		Presentation Milestone	11/30/15
Spring of 2016 Milestones				
User Interface Touchups Milestone	2/7/16		Create Public Facing Details Milestone	3/13/16
Quality Assurance – Phase 1 Milestone	2/21/16		Quality Assurance – Phase 2 Milestone	3/27/16
Move from Sandbox to Infrastructure Milestone	2/28/16		Go Live Milestone	3/27/16
Software Features Milestone	3/13/16		IT Expo Prep Milestone	4/11/16

2.3 Project Schedule

Figure 1 is our project schedule with the major milestones listed.

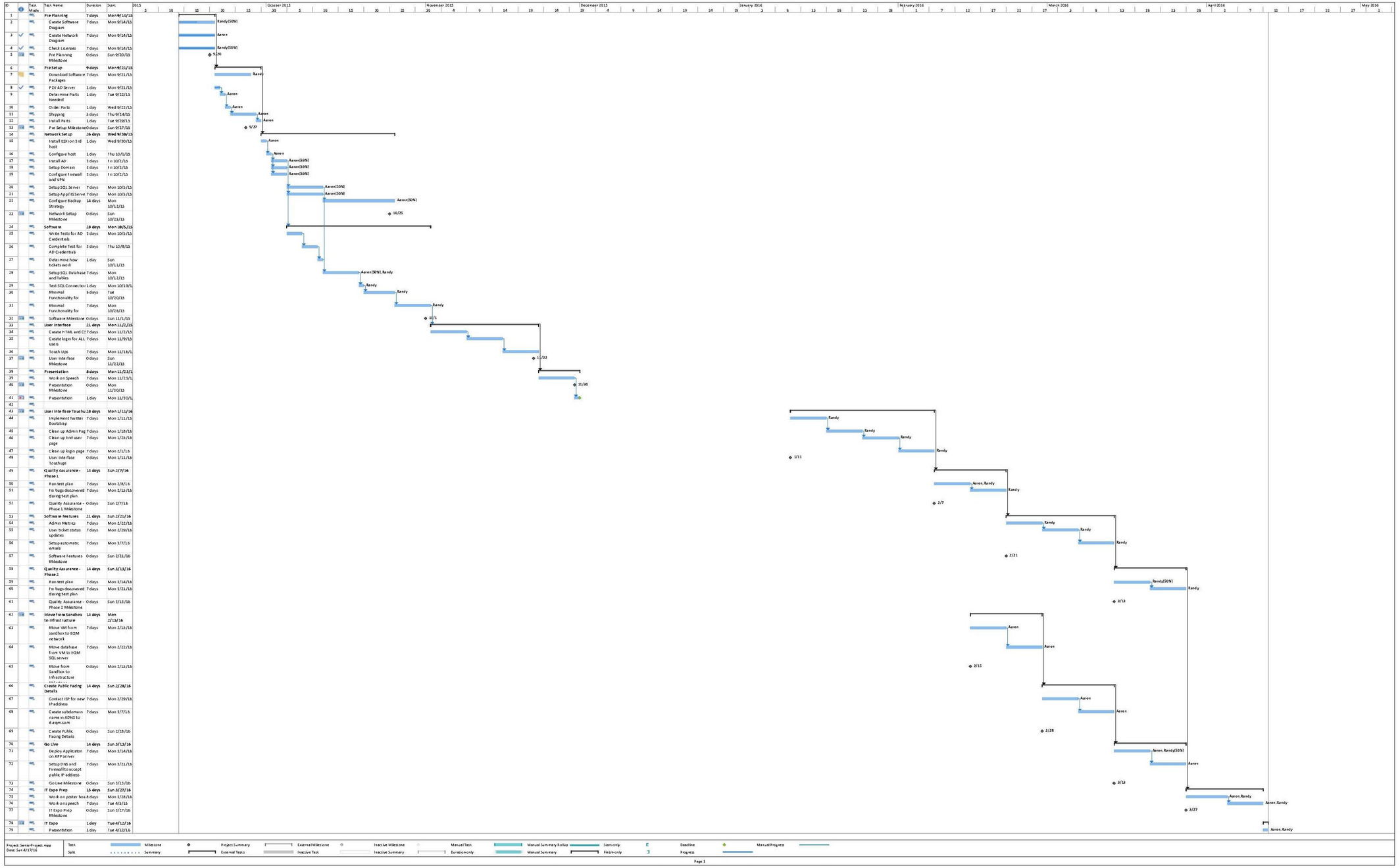

Figure 1. Project Schedule Gantt Chart

3. TECHNICAL ELEMENTS

3.1 Network

EQM's network runs in a VMware virtual environment. EQM uses VMware vSphere, with hosts that run VMware ESXi operating system. Servers are stored on a Storage Area Network (SAN) that communicates with the physical hosts via Internet Small Computer System Interface (iSCSI). EQM's servers run in a Microsoft Windows environment.

3.2 Sandbox

A new sandbox network was installed alongside the already existing network of EQM for testing purposes. The three servers were built in a Windows Server environment. The Domain Controller (DC) server includes Active Directory (AD), Domain Name System (DNS), and Dynamic Host Configuration Protocol (DHCP). The structured query language (SQL) server runs Microsoft SQL Server Management Studio, in which the database resides. The APP server runs the IT Ticketing System software, Application Service Provider (ASP), and Internet Information Services (IIS). The entire APP server and SQL database was moved into the EQM network.

Figure 2 shows the resulting structure of EQM's infrastructure after the installation of the Sandbox.

3.3 Firewall/VPN

EQM uses a physical firewall appliance that manages network traffic throughout the company. The main offices of EQM are connected via Virtual Private Network (VPN) and users located outside any main EQM location connect via Secure Sockets Layer Virtual Private Network (SSLVPN). The firewall will be reconfigured by adding a new network zone called Sandbox_Zone and denying all access to any other zone, especially Local Area Network (LAN), while allowing isolated access to Wide Area Network (WAN). A new local user group, named Sandbox, and new local users of the IT Ticketing System team will be added to the firewall and given access to only the Sandbox_Zone zone. The Sandbox was physically connected to a separate port on the firewall to ensure protection between the two networks.

3.4 Software

The design team developed the application using Microsoft Visual Studio Community 2015, which is free of charge for individual developers with a team of fewer than five [2]. The team used three Visual Studio extensions: Dapper, NUnit, and ReSharper. Other software to be noted includes Microsoft Windows Server and Microsoft SQL Server Management Studio.

3.5 Backup

EQM uses Unitrends backup and recovery software, which employs a redundant physical appliance with raw capacity of 19 terabytes (TB) and redundant cloud storage, also provided by Unitrends. The appliance was reconfigured to back up the servers along with the Sandbox infrastructure.

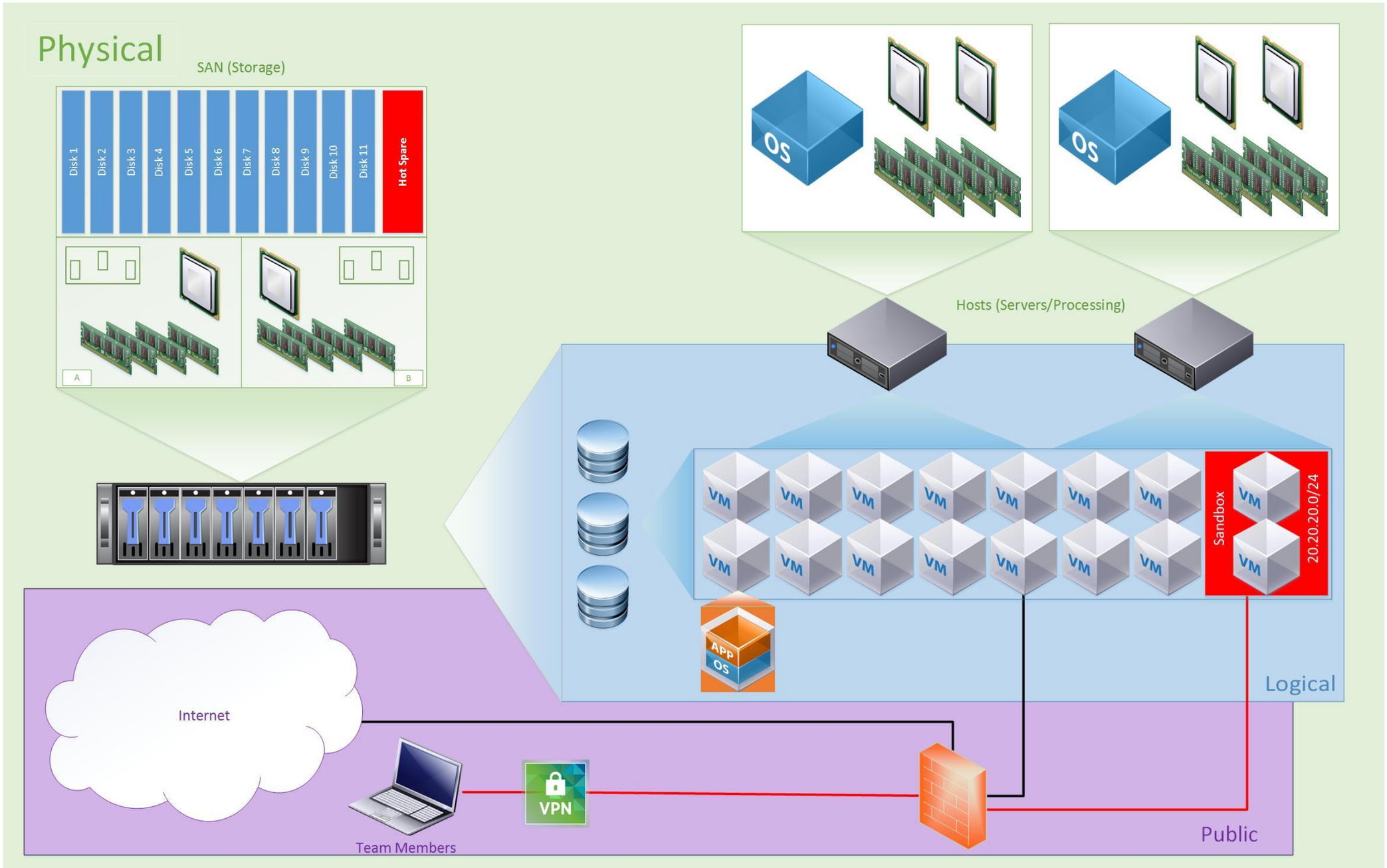

Figure 2. EQM's Network After Installation of the Sandbox

4. APPLICATION

4.1 Application Architecture

Figure 3 depicts the technology used in the application solution.

4.1.1 Library Project

The Library Project handles the logic as well as communication with the SQL database. This project includes the repositories and models. The repositories are code bases the developer calls back to for all features and functionality. The models are custom types that allow the developers to work with employees or tickets as opposed to integers or strings. The model allowed the development team to assign multiple properties to a single variable.

4.1.2 Test Project

The Test Project was designed so the development team could run a single method at a time using Test-driven Development (TDD). TDD required the developers to pass each test before moving on to another feature. The Test Project was only for development purposes and was not deployed to final production.

4.1.3 Web Project

The Web Project makes call backs to the Library Project for features and data. This project also includes the views that the user interacts with and the controllers which determine which view a user can see upon authentication. This project contains the images, style sheets, JavaScript, and anything else that would display on the webpages.

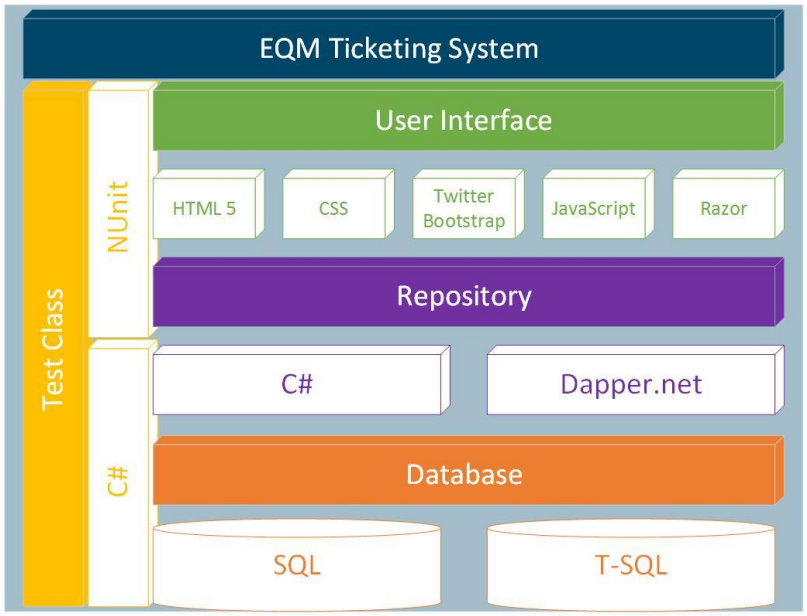

Figure 3. Application Architecture

4.2 Test-driven Development

TDD allowed the team to develop very loosely, and individually critique each method. First the developers wrote a test that required the minimal functionality of the method being worked on. Then the developer wrote code to pass the test. If the test failed, the developers would go back and write the minimalist code to pass the test. If the test passed, the developers would simply move on to the next feature. By repeating this process, the developers were able to build the application one method at a time. TDD allowed for easy changes, prevented scope creep, and showed future developers how the code was intended to work in a very clear and concise way. ***Figure 4*** is a graphical representation showing the logic of the TDD process.

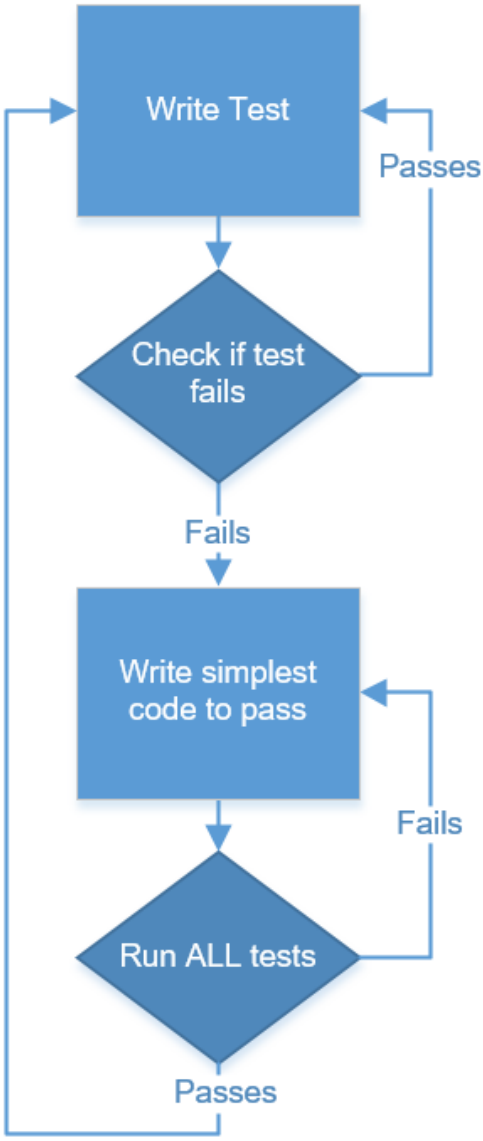

Figure 4. Test-driven Development Process

4.3 Security

Security was a concern with this application, since it is a public facing website. The application uses AD validation from the Authorize attribute. This attribute stores an encrypted cookie on the end users local machine that handles login information and application data. The application redirects to controllers to prevent unauthorized access to any view. SQL stored procedures and character limits were used to help prevent against SQL injection attacks.

4.4 User Interface

4.4.1 Login View

The login view is the first view a user sees. Everyone has access to the login view. The login view requests a user to type in their network username and password, as shown in ***Figure 5***. From this view, they will be redirected to either the user or admin view depending on the user's active directory role.

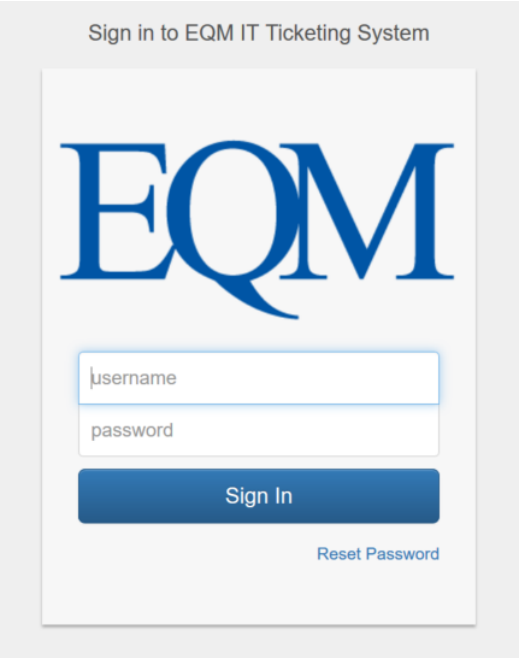

Figure 5. Login View

4.4.2 User View

The user view is where the user will fill out a form and submit upon completion, as shown in ***Figure 6***. This view also displays troubleshooting steps, depending on selected problem. After submission, the user will be logged out and redirected to the login view.

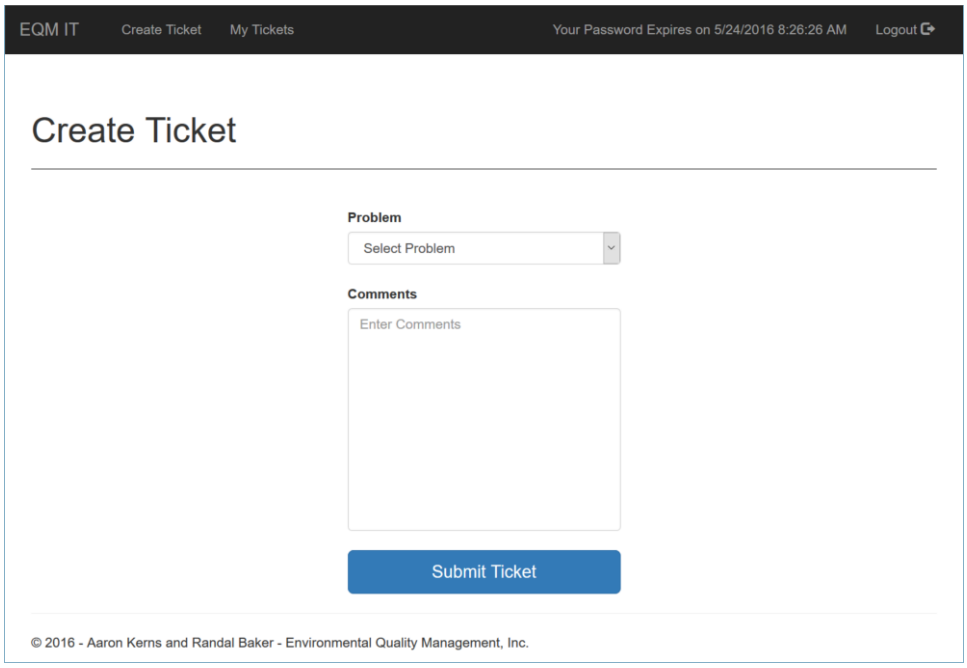

Figure 6. User View

4.4.3 Admin View

The admin view is only for users who are administrators of the application. This view initially displays the application landing page, which displays the administrator metrics, as shown in ***Figure 7***. Here, the admin can navigate to the tickets page, shown in ***Figure 8***, which shows all open and pending tickets. This page allows the administrator to modify tickets. The landing page also allows the administrator to view the history of tickets in the last three months.

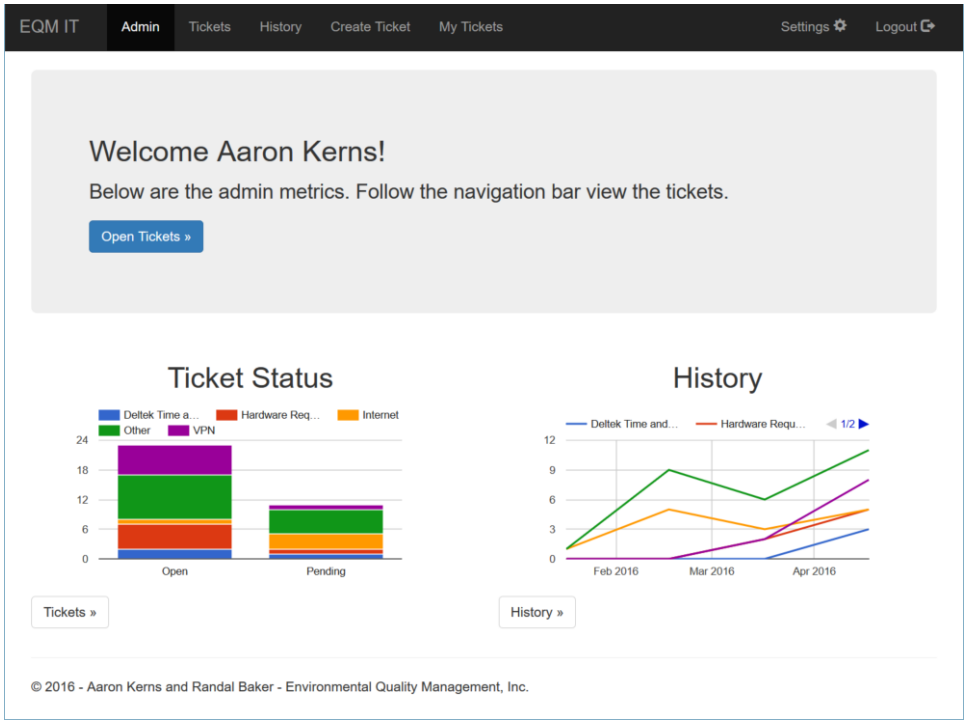

Figure 7. Admin View

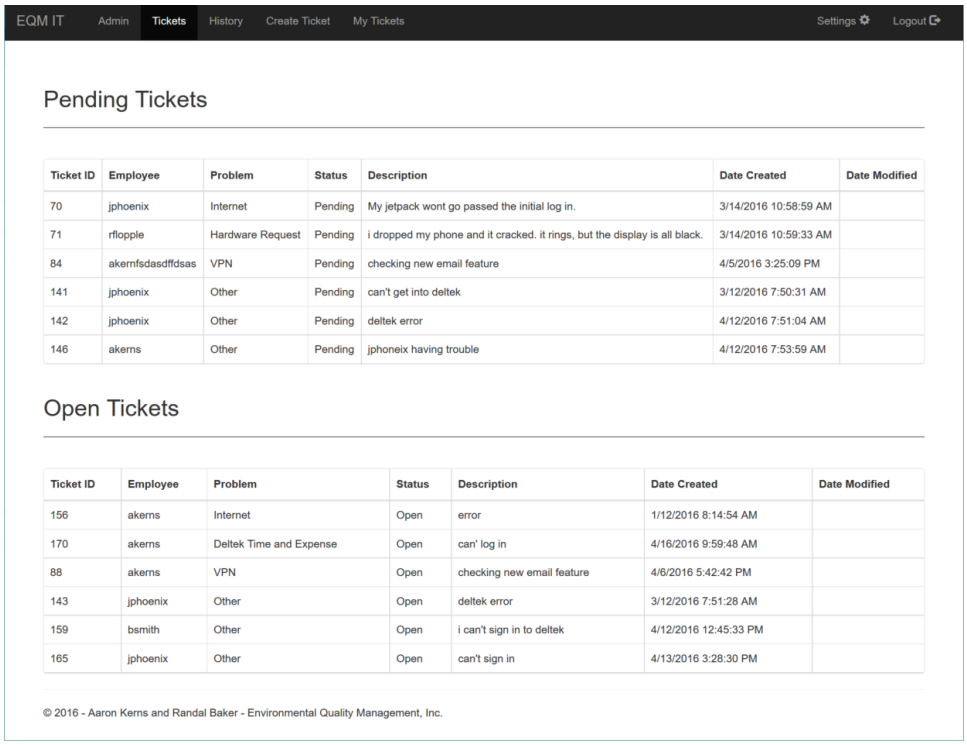

Figure 8. Ticket View

Figure 9 shows the flow of the application as the user successfully signs in with their AD credentials and submits a ticket.

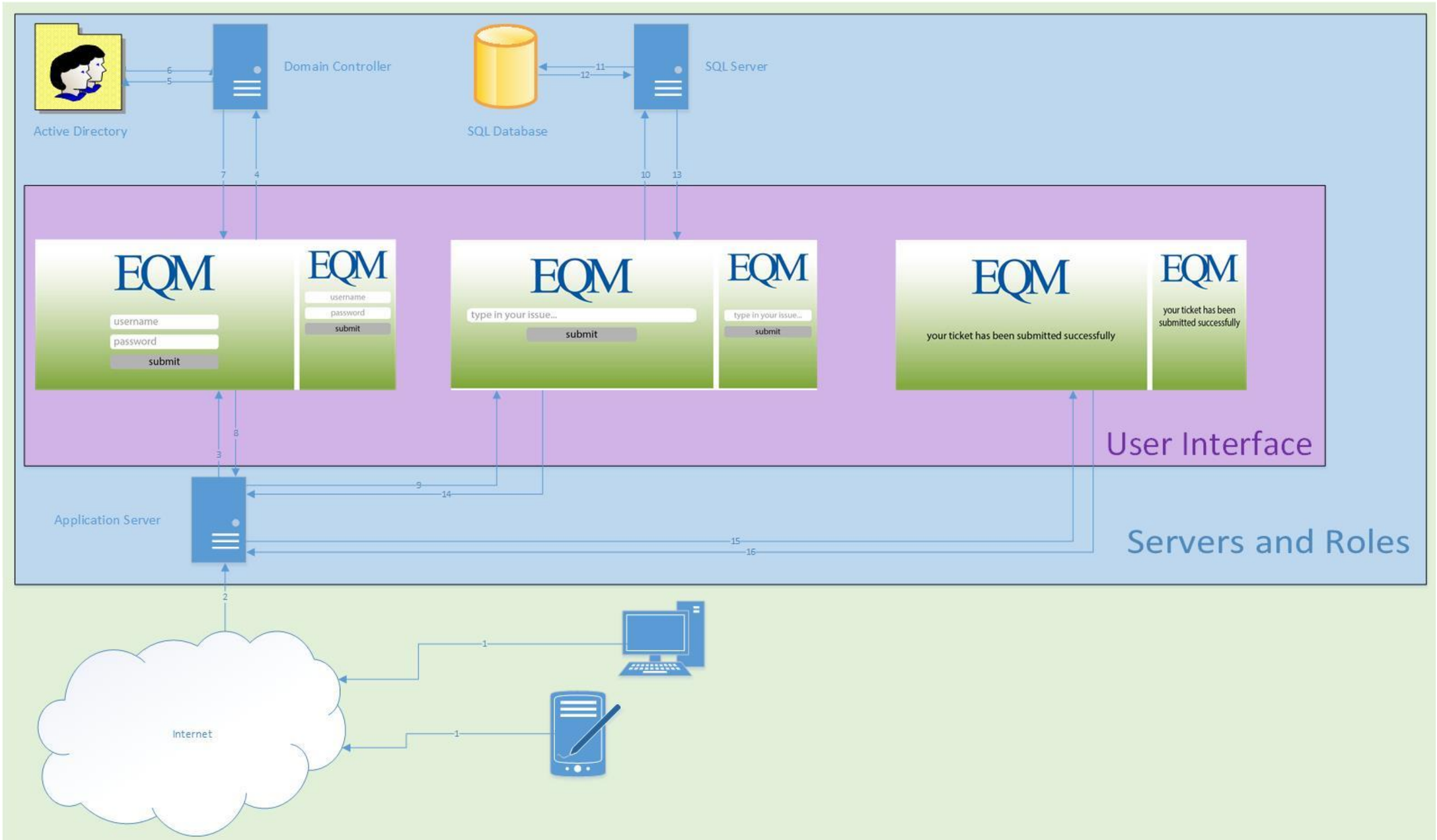

Figure 9. Application Flow

5. TEST PLAN

5.1 Overview

The testing chapter explains the steps and conditions that prevented bugs from entering production code. It includes the objective of testing the application and each test case in full detail.

5.2 Objective

The objective of testing is to remove known bugs from normal use of the application. Employees working for EQM were selected to participate in testing the application.

Figure 10 shows the form that was completed by each tester of the application. It includes each test case.

5.3 Test Cases

5.3.1 Admin Login

STEPS	EXPECTED OUTCOME
1. Enter Admin credentials	Admin will be redirected to the admin page, where they are able to see all created tickets
2. Click Login	

5.3.2 Admin Logoff

STEPS	EXPECTED OUTCOME
1. Enter Admin credentials	Admin will be redirected to the admin page. After log off, admin will be redirected to home page.
2. Click Login	
3. Click Log off	

5.3.3 User Login

STEPS	EXPECTED OUTCOME
1. Enter User credentials	User will be redirected to the user page, where they are able create a ticket.
2. Click Login	

5.3.4 User Logoff

STEPS	EXPECTED OUTCOME
1. Enter User credentials	User will be redirected to the user page. After log off, user will be redirected to home page.
2. Click Login	
3. Click Log off	

5.3.5 Submit a Ticket

STEPS	EXPECTED OUTCOME
1. Login as User	The user will be redirected to the success page. The user will be logged off and then redirected to home page.
2. Fill out ticket form	
3. Click Submit	
4. Verify ticket was submitted	

5.3.6 Close a Ticket

STEPS	EXPECTED OUTCOME
1. Login as Admin	The admin will see the ticket that was selected, closed on the admin page and on SQL server.
2. Select ticket to close	
3. Click close	
4. Submit	
5. Verify ticket is closed	

5.3.7 Pend a Ticket

STEPS	EXPECTED OUTCOME
1. Login as Admin	The admin will see the ticket that was selected, pended on the admin page and on SQL server.
2. Select ticket to pend	
3. Click pend	
4. Submit	
5. Verify ticket is pended	

5.3.8 Re-open a Ticket

STEPS	EXPECTED OUTCOME
1. Login as Admin	The admin will see the ticket that was selected, opened again on the admin page and on SQL server.
2. Select ticket to open	
3. Click open	
4. Submit	
5. Verify ticket is open	

Project Name	EQM IT Ticketing System
Tester Name	
Test Date	
Test Browser	

Notes:

CATEGORY	TEST CASE	STEPS FOR TESTING	EXPECTED OUTCOME	ACTUAL OUTCOME	PASS/FAIL	COMMENTS
Security	Admin Login	1. Enter Admin credentials 2. Click Login	User should be taken to the admin page where they are able to see all the tickets created.			
	Admin Log Off User Log In	1. Enter Admin credentials 2. Click Login 3. Click Log off 1. Enter User credentials 2. Click Login	Admin should be taken to the login page, and they should not be able to access the admin page through back button or html. User should be taken to the user page where they can submit a ticket			
	User Log Off	1. Enter User credentials 2. Click Login 3. Click Log off	User should be taken to the login page, and they should not be able to access the user page through back button or html.			
Feature	Submitting a Ticket	1. Login as User 2. Fill out form on page 3. Submit 4. Verify ticket was submitted by checking SQL	Ticket should successfully submit. User should be redirected to the login page. Ticket should be visible in the tickets table in the SQL data-base			
	Closing a Ticket	1. Login as admin 2. Select Ticket to close 3. Click the close button 4. Submit 5. Verify ticket was closed by checking SQL	Ticket should successfully close. Admin should be able to continue interaction with other tick-ets. Ticket should be visible as closed in the tickets table in the SQL database.			
	Pending a Ticket	1. Login as admin 2. Select Ticket to pend 3. Click the pending button 4. Submit 5. Verify ticket was marked as pending by checking SQL	Ticket should successfully close. Admin should be able to continue interaction with other tick-ets. Ticket should be visible as pending in the tickets table in the SQL database.			
	Re-open a Ticket	1. Login as admin 2. Select Ticket to open 3. Click the open button 4. Submit 5. Verify ticket was marked as open by checking SQL	Ticket should successfully close. Admin should be able to continue interaction with other tick-ets. Ticket should be visible as open in the tickets table in the SQL database.			

Figure 10. Preliminary Test Plan

6. CONCLUSION

6.1 Fall Semester 2015

In the 2015 Fall Semester, the team completed implementing a Sandbox into EQM's infrastructure, separate of their network, creating the database, having basic functionality and a user interface for both users and admins.

6.2 Spring Semester 2016

In the 2016 Spring Semester, the team completed the following:

1. Redesigned the user interface for users and admins
2. Configured and assigned security roles
3. Loaded additional "frequent issues" choices for user to select
4. Created a dynamic knowledge base
5. Created admin metrics that contain graphs and statistics
6. Configured Advanced DNS
7. Convened a test group for quality assurance
8. Published the system for all of EQM to use.

APPENDIX A. ADDITIONAL INFORMATION

The team contacted the following professionals for clarification and additional insight.

1. Kevin Ghantous, Instructor – ADJ, Security
2. Björg Prodan, UC CECH Software Applications Technical Lead

APPENDIX B. REFERENCES

[1] "Salary Information," 2014-15. [Online]. Available: http://www.uc.edu/propractice/salary-information.html.

[2] "Microsoft Software License Terms," 2015. [Online]. Available: https://www.visualstudio.com/support/legal/mt171547.