CyberPods

CyberPods

by

Dakota Blakey, Wade Griffith, Henry Griffiths

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

Dakota Blakey	4/26/2020 Date
Wade Griffith	4/26/2020 Date
Henry Griffiths	4/26/2020 Date
Ryan Moore	4/26/2020 Date

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

April 2020

CyberPods

Prepared by:

Dakota Blakey, Project Manager & Network Architect Assistant
Wade Griffith, Director, Hardware Engineering
Henry Griffiths, Documentation and Virtualization Specialist

Students of
University of Cincinnati
College of Education, Criminal Justice and Human Services
School of Information Technology

April 2020

TABLE OF CONTENTS

List of Illustrations

TABLES

FIGURES

LIST OF ACRONYMS

VMs Virtual Machines

ABSTRACT

According to The Hechinger Report "On the eastern edge of Silicon Valley, school districts have gone into debt by, authorizing a $16.2 million sale of general-obligation bonds intended almost exclusively for laptop purchases." (Marcus, 2019) CyberPods is a system designed to let high school classes delve into the world of Cyber Security, leveraging the cheap used hardware market. Designed using the existing Proxmox Virtualization platform, CyberPods provides a way of easy installation and operation of Virtual Machines, so the focus can be on learning about Cyber Security, rather than battling configuration files. By utilizing the power of the cloud, CyberPods automatically configures and installs itself on hardware. A web-based user interface provides easy management and deployment of pre-configured Virtual Machines that are ready for a classroom environment.

1. INTRODUCTION

1.1 Problem

Universities and schools around the country don't always have the funds or grants to spend on new technology. According to The Hechinger Report "on the eastern edge of Silicon Valley, just a 20-minute drive from Google's Mountain View headquarters, the district around has put these laptops into students' hands by going into debt, authorizing a $16.2 million sale of general-obligation bonds intended almost exclusively for the laptop purchases." (Marcus, 2019). Whether the education organization is poor, middle class, or wealthy, everyone should have the opportunity to work with technology in a world where if someone doesn't know technology, it would be hard to survive in. Most education organization have older hardware that needs to be used until the end of life and be compatible with newer technologies for a while. The Hechinger Report also states, "Using long-term debt to pay for technology that may be obsolete in a couple of years, he says, is "like taking out a mortgage to buy groceries." Yet that is precisely what school districts across the country, including some in California, Texas, Minnesota and Kansas, have done in a desperate push to get technology into classrooms." (Marcus, 2019) Software can cost in between the hundreds or thousands of dollars and most education organizations can't afford these prices.

The current marketplace doesn't have an easy to configure solution for pushing images and software to older technologies using open-source free software. There are many ways to spin up a virtual machine or even a few virtual machines, however there are not many ways to start up 20 or more Virtual Machines with an interface that is easy and simple to use for those who aren't extremely tech savvy. CyberPods gives these educational organizations these tools they are missing and for free as long as they can provide the hardware.

1.2 Solution

Our solution, CyberPods, it is a system that allows someone who has access or could buy fairly cheap hardware to quickly setup and deploy multiple virtual machines without them needing too much of a technical background. We combined all the different parts that goes into setting up a virtual machine and tied it together with a very simple and easy to use user interface. The interface works with older hardware, which most education organizations have in the real world. Some other software's that offer something similar to what is offered with CyberPods are flexVDI and FOSS-Cloud. FOSS-Cloud and flexVDI are both open source software's that allow users to create VDI environments, but some of the better features users must pay for.

CyberPods allows educational organizations to spend less time working on technology solutions for their classrooms, meeting rooms, etc. CyberPods provides a free cloud-based solution that allows schools to run virtual machines on older outdated hardware. The virtual machines have all the software, apps, and features needed for the students to be ready to learn as soon as the image has finished deploying to the end user's hardware. CyberPods is here to make technology deployment stress free.

1.3 Project Goals

With CyberPods our goal was to create a cheap cloud based virtual machine solution for schools that would let them, reduce spending on new hardware, reduce spending on software, reduce time on technology setup, and only require entry level technology knowledge to operate.

1.4 Overview

This report will cover the process our team underwent in detail on how we took the concept of CyberPods and turned it into a full on product solution for schools in need of a cheap virtual machine solution. This report includes the following sections: Project Concept, Methodology, Technical Discussion, Testing, Budgeting, Timeline, and Future recommendations.

2. DISCUSSION

2.1 Project Concept

CyberPods was a project recommended to us from our senior design advisor Ryan Moore. Ryan Moore has been wanting to work on this project in his spare time but hasn't found any yet. Our group came to Ryan looking for a senior design project and he suggested his idea which would eventually be named CyberPods.

As a team we ran through some other ideas as in doing something with magistrates on credit cards or developing an app of some kind. But after not coming up with a solid foundation for an idea, it was decided as a team to go with Ryan's idea of CyberPods. Our team is comprised of all networking majors and cloud-based products and virtual machines are right in our powerhouse for our team. We thought our team would be a perfect fit for coming up with an open source solution to deploying virtual machine images. We all have had experiences in real world dealing with virtual machines and we have decided we want to take our own spin on the way virtual machines will be deployed. After much consideration among the team we have decided to make CyberPods are senior design project.

2.2 Design Objective

Our objective was to develop a functional frontend that will provide students and teachers the ability to create and use virtual systems for learning about technology. CyberPods promotes the ease of use by providing a platform that is easy for teachers to set up on cheap hardware, as well as maintaining an up to date and functional library of virtual machine templates.

CyberPods is a system that includes the use of the following software; Express JS for the framework which is used to render webpages and host the website. For our server environment it uses NodeJS. Our database that stores our students and admins information is managed by Firestore. To manage our VMs we use Proxmox Virtualizor, all this is being hosted on a local in house server.

The features CyberPods includes:

- The ability to customize virtual machines based on needs, such as; RAM, CPU, HDD, and OS.
- Easily deployable on cheap hardware
- User friendly front end that's simple to use and does not require an extensive technological background
- Retrieves updated VMs as provided by the Ohio Cyber Range
- Freezes VMs when not in use, to avoid using excessive resources

One of the initial goals that we had to abandon was using the apache web server environment as it wasn't able to handle what we wanted to do for CyberPods, so we scratched it and switched over to the NodeJS and Express JS software to host and create the website for CyberPods.

Another initial goal we initially planned to use FAI to install Proxmox to get everything set up. We had to abandon this idea because getting Proxmox to be installed through FAI didn't appear possible, and we were consuming a lot of time testing due to having to go through the full install process each time. We still partially accomplished this goal, however, through the use of an automatic install script that installs our tools on top of Proxmox, with the only manual part being the installation of Proxmox itself.

2.3 Methodology

Early on as a team, it was decided to take an agile methodology approach to this project. This was decided base on the experience the team had using this methodology on previous projects along with the success on those projects using this approach. The goal with CyberPods was to design something that was practical and optimized to be used by almost anyone, anywhere, with little to no technical background. With these limitations and goals in mind. The team would meet routinely when adding new features or discussing front-end development, to make sure that the limitations and goal above were being meet with every new feature and change. This was done easily enough as using the Agile methodology it was easy to meet at the end of each cycle to discuss the changes that were made. The procedure that we used throughout the project was as followed; Meet with group to discuss next part of the project > Break up work between member, setting new goals for all > Work for a week on current goals > Hold a meeting during the middle of the cycle to give a status update on how each member's work is coming along and any issues someone might be having > Finish working on goals > End of cycle meeting to discuss how things went and to make sure things are working properly. This was done by every member throughout the projects life cycle.

2.4 User Profile

Table 1: User Profile, presents the user profile below that was used throughout the development of CyberPods. The user profile discuss potential users, related experiences with software, and similar software that is similar to what CyberPods has to offer. The user profile will also discuss frequency of use, task experience, and key project design requirements that the profile suggests.

PROJECT TITLE: CyberPods
POTENTIAL USERS: - Technological adept individuals - Non-Technological adept individuals
SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: - This project will be primarily targeted towards **technologically adept individuals**. These technologically adept individuals looking to teach non-technological adapt individuals. They will most likely work in the technology sector in some form as teachers or professors. The bulk of our features will attract this type of clientele. - **Non-technologically adept** users will be new to or not from the technology sector or have a technical capacity. They will still be able to use the product. They just may not have a need or understand at first some of the features. They will mostly be students.
EXPERIENCE WITH SIMILAR APPLICATIONS: - VMware - Hyper-V - Virtual Box - UC Sandbox
TASK EXPERIENCE: - Using a web browser to navigate a web application - Comfortable with using a web application to manage Virtual Machines - Basic understanding of how virtual machines are setup and deployed - Basic understanding of making images from operating systems. - Basic computer networking skills - Comfortable with computer hardware
FREQUENCY OF USE:

- Whenever a user needs to spin up virtual machines for a class. This occurrence can happen daily, weekly, monthly, or yearly.
KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS: - Quickly create Virtual Machines - Easily setup different specs such as CPU, HDD space, Memory, and OS - Easily use premade Virtual Machine templates - Easy to use user experience with seamless setup and deployment of Virtual Machines - Spend less time on setting up technology

Table 1: User Profile

2.5 Use Case Diagram

Figure 1: Use Case Diagram displays the use case for CyberPods. The diagram shows a use case diagram for the students (regular users) and teachers (admin users) and the tasks that associated with each user role.

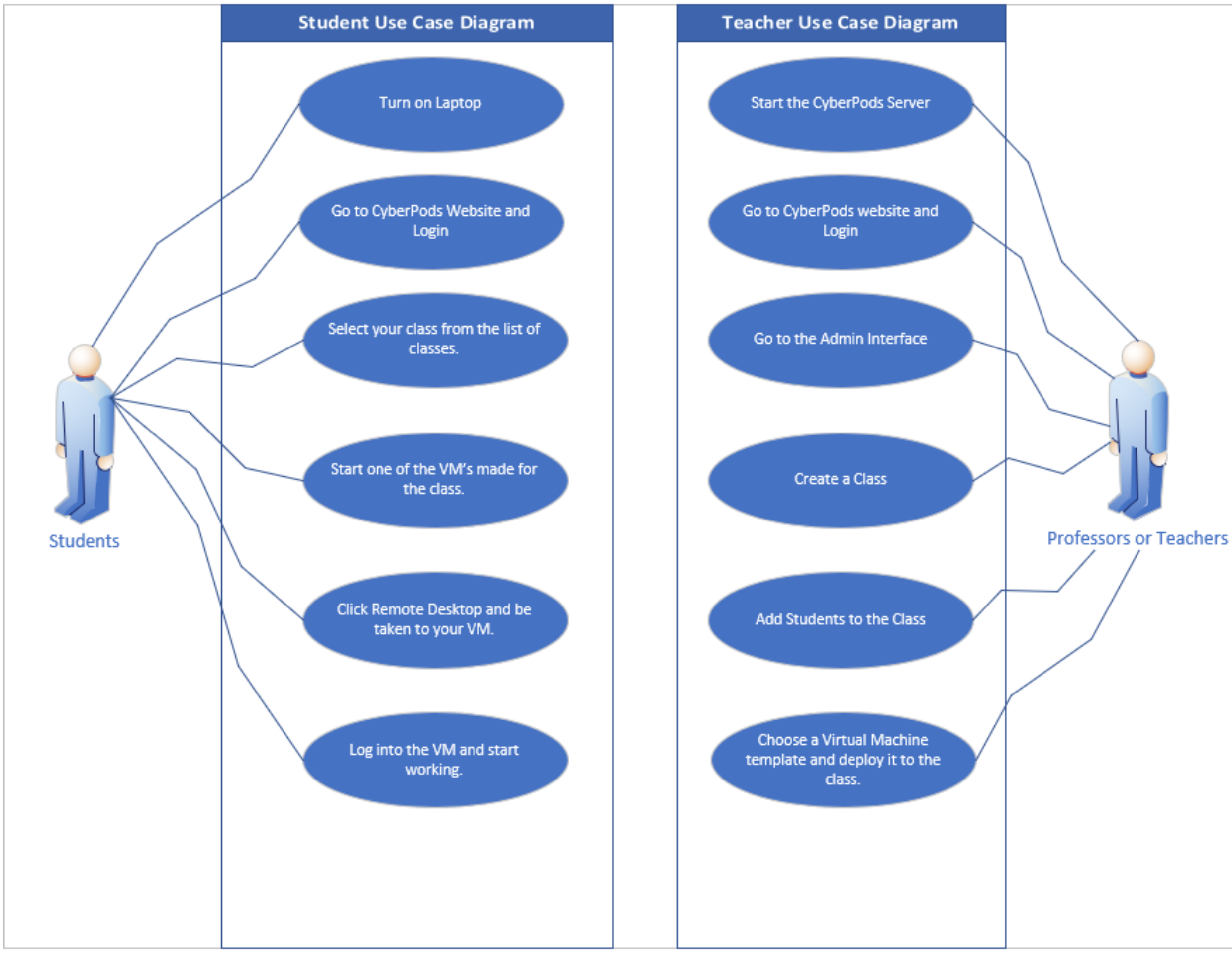


Figure 1: Use Case Diagram

3. TECHNICAL ELEMENTS

3.1 Network

The server that run Proxmox are setup in in house. The servers that we have used for testing are a Dell R710 the specs are Dual Xeon X5650 CPUS, 128 GB of ECCDDR3 RAM, 128 SSD boot drive and 6 TB of storage on HDD's. The templates created by the users are stored in the Firebase to be accessible from wherever there needed. We have created a VPN connection to the server in order to perform server maintenance and administration.

3.2 Application

We used JavaScript, with NodeJS, Express, and Firebase. We chose those three because of the familiarity we had with these languages and tools. At the time, Henry was working on a project for his job that required the use of these tools, so we were able to quickly use his skills with the language to build our backend quickly. IN addition, JavaScript is a weakly typed language that provides extensive package support, so we wouldn't have to hook into other languages to perform any tasks, and could keep everything with JavaScript for the backend, and JavaScript, HTML and CSS for the frontend. We also used Proxmox, NodeJS, and Express mostly because our goal was to use free and open source software in order to make our solution as low cost as possible. Proxmox is an open source software that allows us to manage Virtual Machines for free. Node JS is an open source server environment and Express is also an open source software that allows us to host the website.

3.3 Security

In order to secure our web portal, we will be using OpenSSL to create a certificate signing request (CSR) with an SSL certificate. Another security element deployed into our solution is an HTTPS connection to our website. We wanted to make an HTTPS connection to our website in order to encrypt the traffic to our website. We used Let’s Encrypt in order to get certificates to secure our website. Some other ways we can help secure our web portal by disabling the server signature, banner, and trace http requests. Every action that is sent to the backend is checked for user permissions, and the passwords and permissions are handled through Firebase Auth. This allows Firebase to handle the passwords and permissions, and leave us to focus on the development of the backend, without having to build our own tools to handle password salting and hashing. The virtual machines are intended to be configured to run without access to any network other than an internal one local to the student’s vms. This prevents the students using the tools and techniques that they have acquired through the class, and their own experimentation, to do damage to their school’s network.

3.5 Technical Architecture Diagram

Figure 2: Technical Architecture Diagram, shows how all of the technology that is used in CyberPods is all connected and operates together.

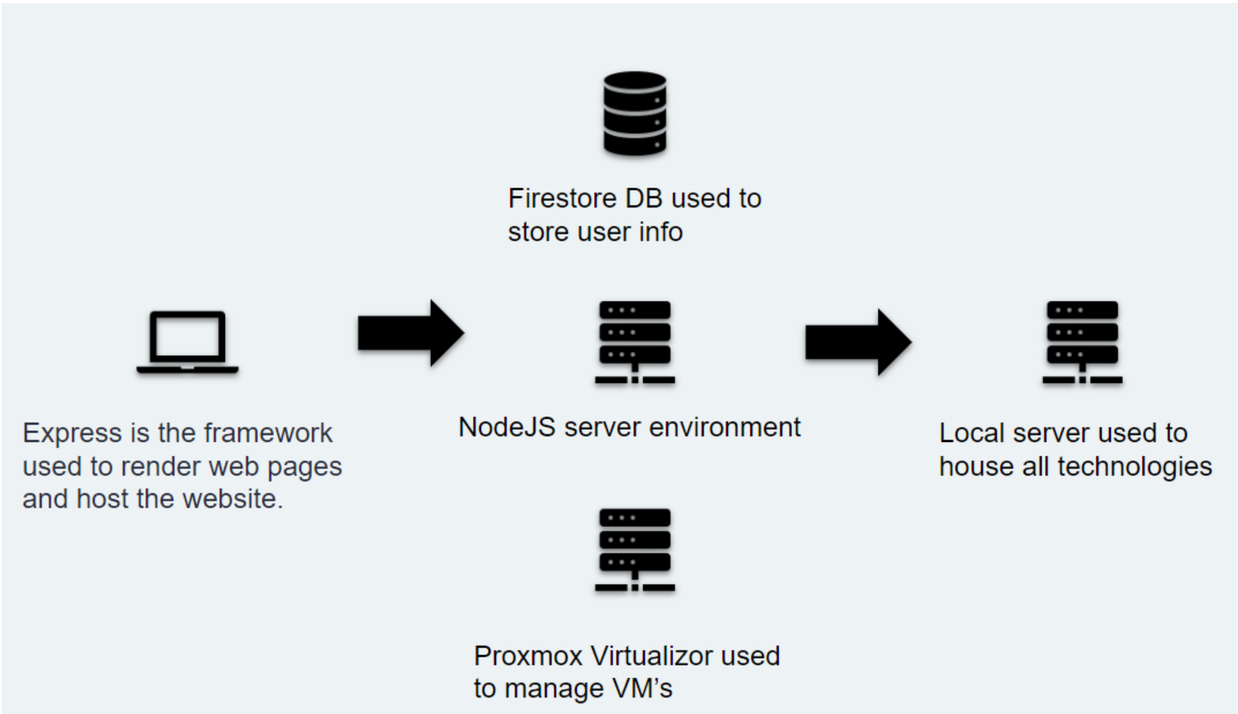


Figure 2: Technical Architecture Diagram

3.6 User Interface

Figure 3: Login Page, shows what users see when they first connect to the CyberPods website.

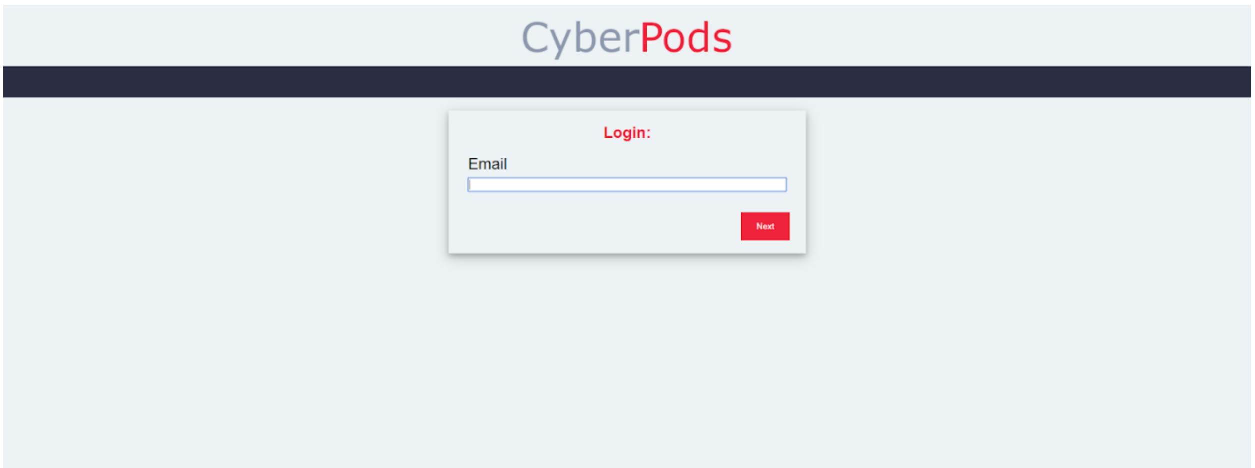


Figure 3: Login Page

Figure 4: Create VMs Page, is the page that admin users can setup and create new VMs.

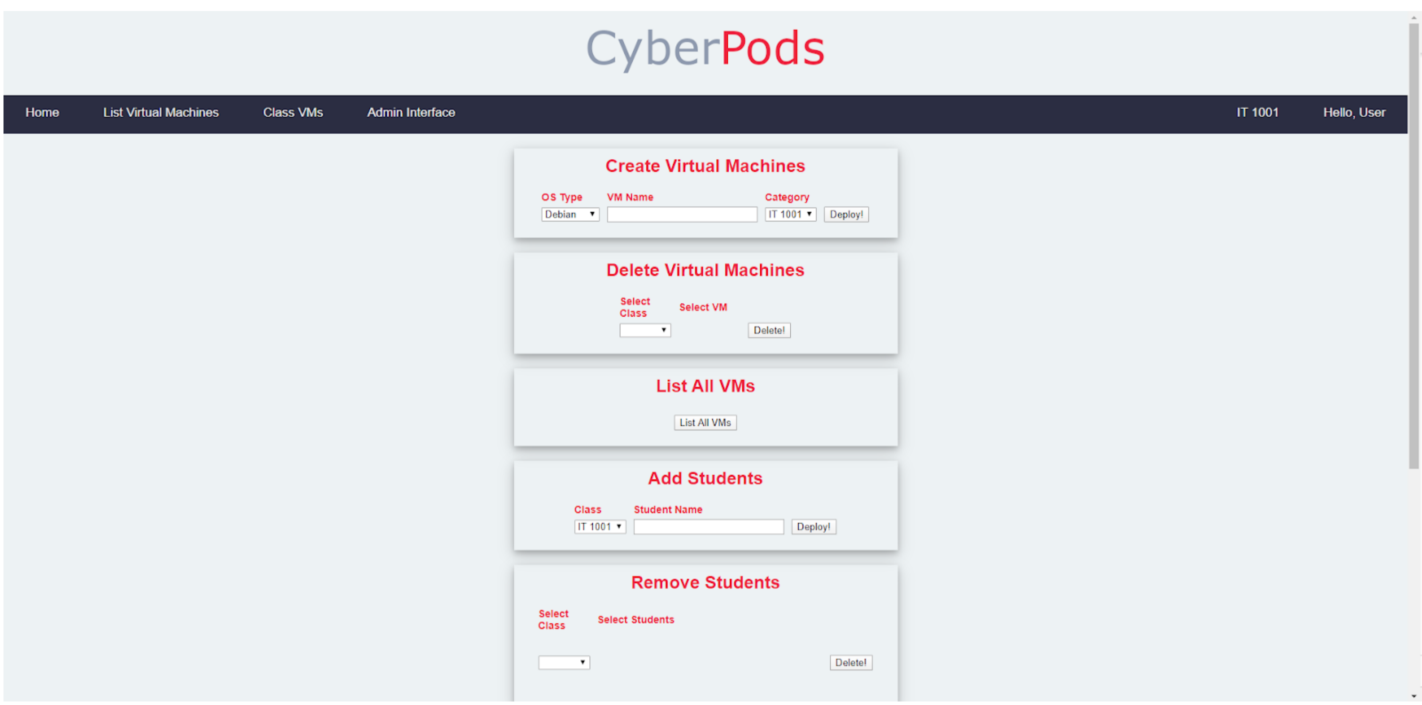


Figure 4: Create VMs Page

Figure 5: VMs Control Center, this is main control page for VMs, here users can start and stop the VMs along with connect to them via remote desktop.

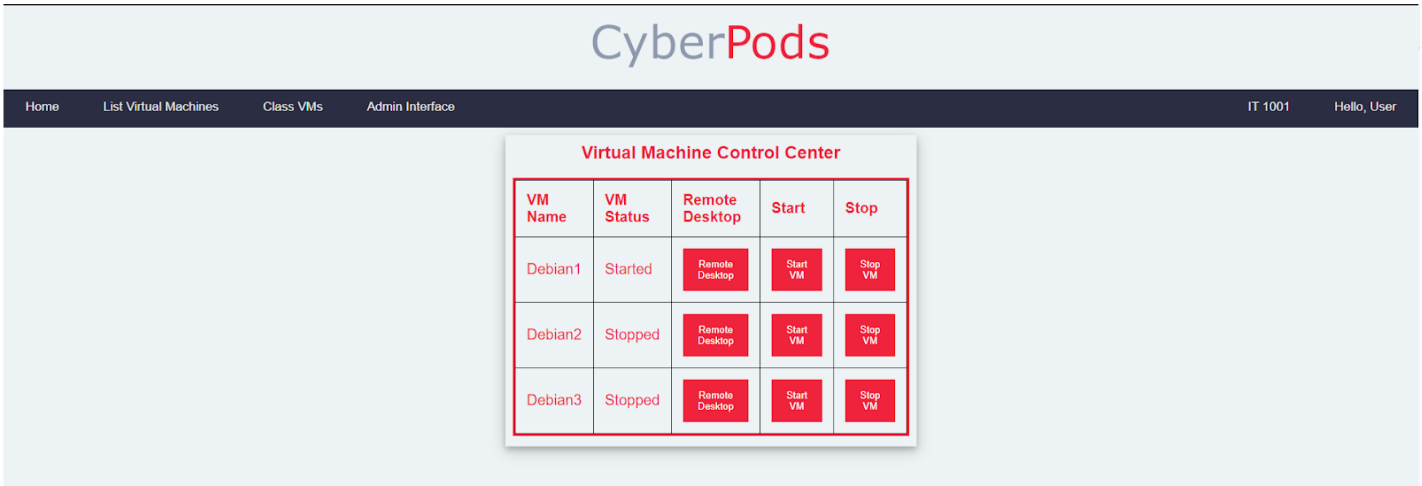


Figure 5: VMs Control Center

4. PROJECT MANAGEMENT

4.1 Testing

We decided to do in house or Alpha testing on our back-end and real user or beta testing on our front-end. Starting with backend testing we conducted a biweekly in house testing period where we would all meet on call and go through the different features we had add that week and run multiple different test on these features to check for bugs, errors, or unintended responses. When working on our project we used a more agile method approach to it so to a certain extent we were already doing in house testing quite frequently when we could get to the end of a cycle. There wasn't much information to record while doing in house testing on the back-end as if something didn't work properly we would work on it until it was working as intended. However during our beta test is when we started recording user feed.

The users that participated in our testing where students, co-workers, and faculty at the Lindner College of Business. We also each had a few family and friends do user testing as well. It should also be noted that these users had a wide variety of technology backgrounds some with lots of experience others with not so much. It's very important as it allows us to design something that can work for everyone no matter their experience with IT, as that is one of our main goals of the project, since its being focused more towards students in High School who might not have a lot of experience with this side of technology we want to create something that they can understand and quickly learn to operate the site.

How we conducted the test: We had a computer setup that had our website already pulled up and on the home page. We asked them first a few questions about their initial reaction to the site and things they noticed right away. We then asked them to complete task without giving them any idea how to do so. Once they completed a task we asked them on a scale of 1 – 5 how they would rate the difficulty of completing said task, with 1 being hard to complete and 5 being easy to complete. We got a total of 25 people to complete our front-end beta testing.

Figure 6: Test Results, below are the average scores for each question.

Question Website Layout	AVG. (1-5 Scale)
How do you like the color scheme ?	3.64
How is the flow of the site?	4.04
How is the readability of text?	4.52
Grammar or Spelling Isuess?	4.92
Question Website Task	**AVG. (1-5 Scale)**
Navigate to Login Screen	5
Navigate to Home Page	5
Navigate to VM Page	5
Select How Many VMS to Create	4.2
Select VM Operating System	4.2
Start VM	3.64
Stop VM	4.64
Delete VM	4.76

Figure 6: Test Results

4.2 Budget

Table 2: Project Budget, presents the project budget. It presents the real-world cost of X, but totals to $0 as this was our senior project and is not contracted, but developed and donated to our community in exchange for our experience and knowledge we gain.

Human Resources	Units/Hrs.	Cost/Unit/Hr.	Subtotals
Project Manager	1200	$40.00	$48,000
Hardware Engineer	1200	$50.00	$60,000
Virtualization Specialist	1200	$40.00	$48,000
Equipment			
Workstations	3	$500	$1,500
Servers	1	$1,000	$1,000
Total project cost est.			$158,000

Table 2: Project Budget

4.3 Work Breakdown Structure

Table 3: Project Timeline, is a breakdown of all the different elements that went into creating CyberPods and the dates they were started along with a target end date to complete them.

Task Name	Duration (Days)	Start Date	End Date
1.0 Project Management and Deliverables	252	8/26/19	4/15/20
1.1 Team Building	7	8/26/19	9/2/19
1.2 Ideas and Brainstorming	7	8/26/19	9/2/19
1.3 Fall Semester Assignment 0: Team Members & Project Name	7	8/26/19	9/2/19
1.4 Fall Semester Assignment 1: Team Contract	21	9/2/19	9/23/19
1.5 Fall Semester Assignment 2: Project Abstract for Tech Expo	20	9/24/19	10/14/19
1.6 Fall Semester Assignment 3: Team Contract Resubmission	20	9/24/19	10/14/19
1.7 Fall Semester Elevator Speech	7	10/14/19	10/21/19
1.8 Fall Semester Assignment 4: User Profile	7	10/14/19	10/21/19
1.9 Fall Semester Assignment 5: Use Case Diagram	7	10/14/19	10/21/19
1.10 Fall Semester Assignment 6: Draft Report	7	10/28/19	11/4/19
1.11 Fall Semester Assignment 7: Final Fall Semester Report	14	11/18/19	12/2/19

1.12 Spring Semester Assignment 1: Testing plan	23	1/13/20	2/5/20
1.13 Spring Semester Assignment 2: Abstract	28	1/20/20	2/17/20
1.14 Spring Semester Assignment 3: Draft Tech Expo poster	28	2/3/20	3/2/20
1.15 Spring Semester Assignment 4: Final Poster	28	2/10/20	3/9/20
1.16 Spring Semester Assignment 5: Final Report	35	3/2/20	4/6/20
1.17 Spring Semester Assignment 6: Safe Assign for Final Report	35	3/2/20	4/6/20
1.18 Spring Semester Assignment 7: Final Library Copy	42	3/9/20	4/20/20
2.0 Research	28	9/2/19	9/30/19
2.1 Software Requirements	21	9/2/19	9/23/19
2.2 Hardware Requirements	21	9/2/19	9/23/19
2.3 Network Requirements	21	9/2/19	9/23/19
2.4 Security Requirements	21	9/9/19	9/30/19
2.5 Miscellaneous Research	28	9/2/19	9/30/19
3.0 System Design	35	9/23/19	10/28/19
3.1 Create System Diagrams	35	9/23/19	10/28/19
3.2 Create Network Diagrams	35	9/23/19	10/28/19

4.0 Environment Set-up	149	10/7/19	3/4/20
4.1 Automated Server install	18	10/7/19	10/25/19
4.2 Automated installation on Hardware	11	10/14/19	10/25/19
4.3 Automated install of software	17	10/30/19	11/16/19
4.4 Automated software Configuration	30	11/16/19	12/16/19
4.5 Automated file transfer	30	11/16/19	12/16/19
4.6 Automated Update Retrieval	30	11/16/19	12/16/19
5.0 Frameworks	132	9/24/19	2/3/20
5.1 Setup Front End Framework	132	9/24/19	2/3/20
5.1.1 Login Interface	24	1/4/20	1/28/20
5.1.2 Remote Console interface	24	1/4/20	1/28/20
5.2 Setup Back End Framework	132	9/24/19	2/3/20
5.2.1 Deploy VMs	2	11/4/19	11/6/19
5.2.2 VM status monitoring	2	11/4/19	11/6/19
5.2.2 Overprovision protection	15	11/15/19	11/30/19
5.2.3 Authentication	15	11/15/19	11/30/19
6.0 Development	182	10/7/19	4/6/20
6.1 User Interface	53	10/8/19	11/30/19
6.1.1 HTML Page Layout Concept	2	10/8/19	10/10/19
6.1.2 HTML Page Basic Layout	1	10/9/19	10/19/19
6.1.3 CSS Page	1	10/9/19	10/9/19
7.0 Testing	98	1/6/20	4/13/20
7.1 Set Up Testing Environment	98	1/6/20	4/13/20
7.2 Security Test	98	1/6/20	4/13/20
7.3 Final Testing	98	1/6/20	4/13/20

Table 3: Project Timeline

4.4 Gantt Chart

Figure 7: Gantt Chart, is a better visualization of the Project Timeline above, that helps shows our project goals and the dates we started and completed them.

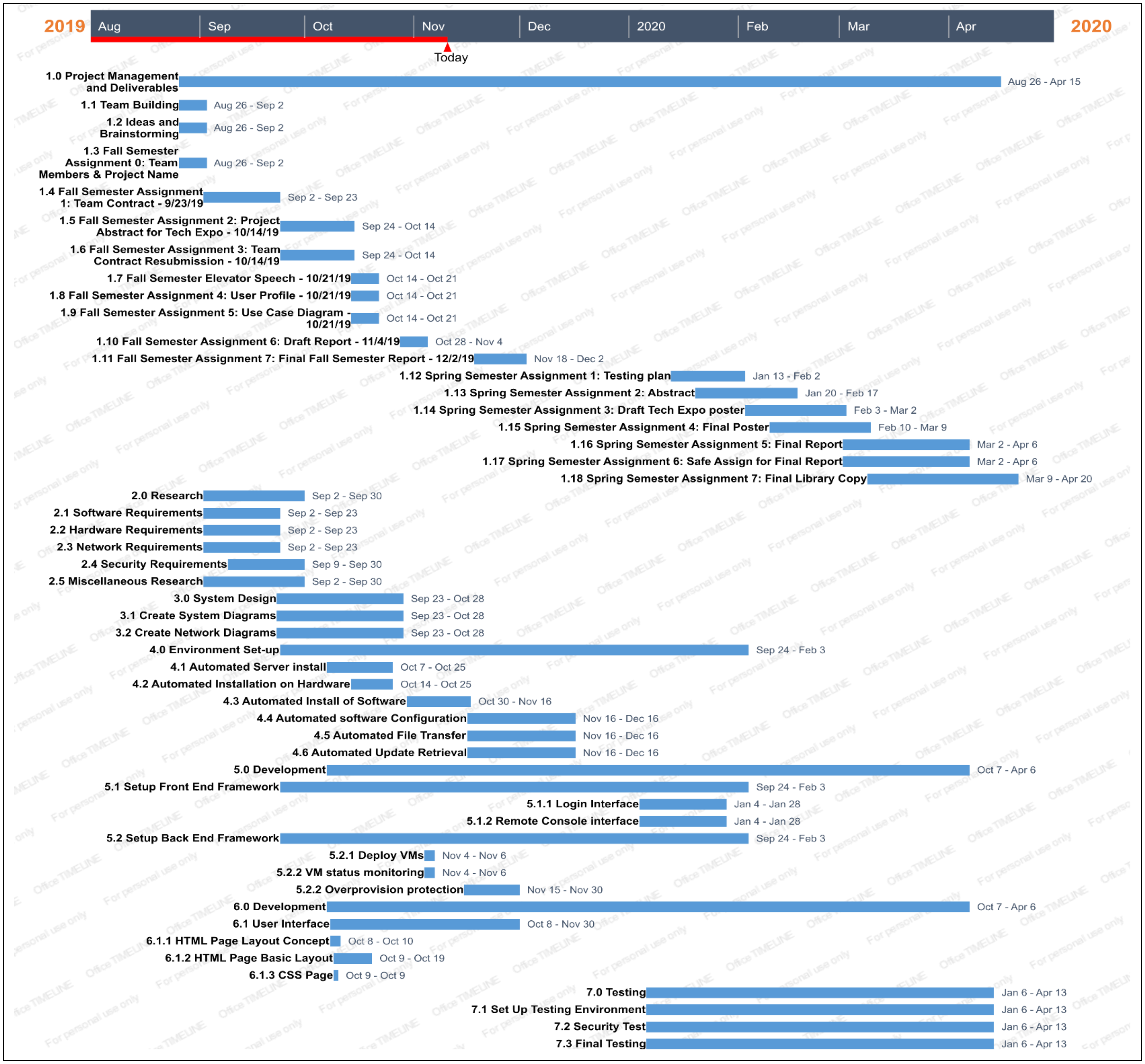


Figure 7: Gantt Chart

4.5 Problems Encountered

Some problems we encountered were we had the wrong server environment at first with Apache Web Server and we solved this by researching other server environments and deciding on NodeJS and Express as a replacement. Another problem we encountered was not being able to meet in person to discuss and work on the project, so our team decided that the best way to stay in constant contact was through Teams chat and WebEx meetings to do screen sharing. We were still able to make this senior design process happen.

Another issue was communication between Proxmox and our backend presented the biggest problem. Because Proxmox was not designed for this, we have a custom script that runs repeatedly to update our back end’s database with updated information about Proxmox. For example, shutting down a VM through the VM itself (and not the stop button in our frontend) would lead to a sync discrepancy, where our backend would think the VM is still running, when it actually isn't. This is solved by the start and stop buttons in our frontend only telling Proxmox what to do, and the status in the database being solely updated by data from Proxmox.

4.6 Future Recommendations

A future recommendation that we would have is to do a lot more research on technologies that someone wanting to use on a senior design project and make sure they are able to achieve the goal that they are setting out to do. If the technologies aren’t able to complete the goal of the project then they will be stuck finding new technologies and maybe having to restart the project completely. Another thing we would do differently would have a more solid foundational knowledge about the technologies that we would be using in the senior design solution. We would also build the system to support multiple hosts in a cluster. While we were able to test on a machine that had decent specifications, a full class of 30 students will require a lot more computing and storage power than a single server can handle.

If we had to do it over again we would switch away from Proxmox, and use pure KVM. Proxmox is based on the KVM technology for virtual machines, and adds a nice web frontend. CyberPods completely replaces that web frontend, so it would probably have been easier to install KVM on top of a Linux machine and skip Proxmox entirely. However, this would've provided its own challenges, as we are using the NoVNC feature that Proxmox provides to remotely access VMs, which is why we chose Proxmox in the first place.

5. CONCLUSION

5.1 Lessons Learned

“Working on this project we believe we learned a lot of valuable skills that will transition to help me greatly in the workforce. Not just learning how to use new tools and technology but also working with a team to complete a goal. More so gaining experience as a project manager we think will be really valuable for creating future opportunities to lead teams. We also think or unique situation this year of having the challenge of working as a team remotely and overcoming that challenge was very beneficial as well. We also learned that not everything goes quite as planned and things that you might think will only take a little bit of time to complete can actually require a lot of time, and vice versa. Overall we’re happy with the performance of the team and each team member’s performance, but most importantly we’re proud of how our project turned out, it’s very special feeling to take an idea and make it into a reality.” – Dakota Blakey

“We believe that we learned a lot this semester. The biggest thing we learned this semester was how to work remotely in a group to accomplish a goal, due to the coronavirus making meetings possible due to us being quarantined in different parts of Ohio. This skill will help me out in the real world a lot, since more and more people are being able to work from home, even before the coronavirus. Other skills that we developed over this project was the ability to create and manage virtual machines using

Proxmox virtualizer and also reshape my skills of HTML/CSS when coding the website for CyberPods. All these skills we've learned will help me in the future." – Wade Griffith

"Debugging takes a lot longer than expected. Going into this each of us had some or little programming experience, being in the system administration track. This allowed us each to learn more about software development, while including our system administration experience to build CyberPods. We should have started working on the project a lot earlier than we did, as well as focused more on the programming side first, rather than the system administration side. This would have allowed us a lot more time to debug and troubleshoot nearer to the deadline, rather than having to spend long nights working." – Henry Griffith

5.2 Fall Conclusion

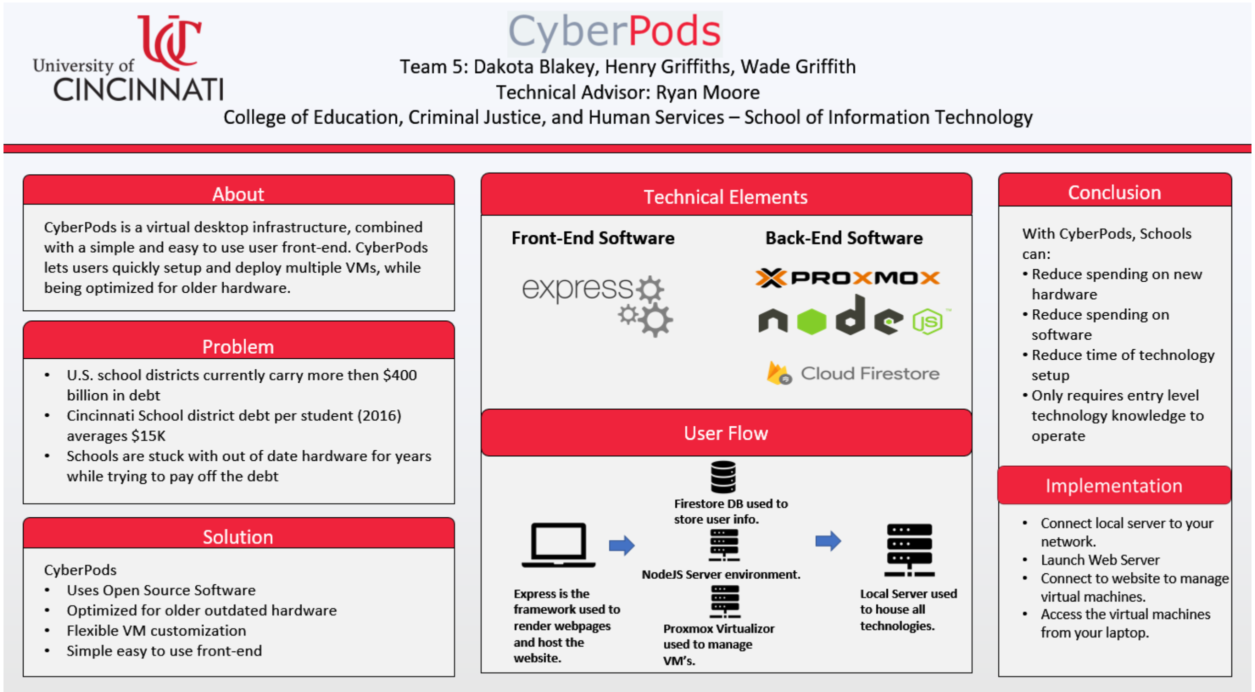
The entire process of making this virtual desktop infrastructure has been structured around keeping it as low cost as possible with open source software. The goal is to have a viable solution for making hardware life cycle longer without having to replace it with newer hardware. We are currently working on a cloud solution to store the created templates in to distribute to end users. We are looking forward to perfecting this process and make CyberPods a viable option for anyone wanting to use it.

5.3 Spring Conclusion

Since the fall a lot of work has been done on CyberPods, starting with user testing our old website and taking that feedback to recreate our new and current website. From there we began on working on security features to ensure that users information would secured when being stored in our database. We then began working on finishing up our backend first by switching from Apache Webserver as it was giving us too many problems to NodeJS and Express. From there we made a lot of progress quickly on our backend. Once the backend was completed we went through and did some final in house user testing, along with cleaning up the styling for the final version of the website. What we've learned from EXPO is that there is a large need for individuals who have experience with creating and developing new ideas to both new and existing problems. This class does a good job preparing us for just that, and gives us both the knowledge and experience we need to make us more competitive amongst others in the workforce.

APPENDIX A. ADDITIONAL INFORMATION

Final Poster



Backend Code

Updating the database with the latest information from Proxmox

```
vmstatus = () => {
    console.log('vmstat', vmstat);
    console.log(new Date().toString())
    //Get the vm list from proxmox
    exec("qmlist", (err, stdout, stderr) => {
        //create qm list - an object with these values
        let list = stdout.trim().split(/\s+/);
        let qmlist = {};
        for (i = 6; i < list.length; i+=6) {
            qmlist['vm' + list[i]] = {
                vmid: list[i],
                status: list[i+2] == 'running' ? true : false,
            }
        }
        console.log('qmlist', qmlist)
        //loop through the list of vms from the database
        for (vm in vmstat) {
            console.log('vm', vmstat[vm]);
            console.log('qm', qmlist[vm]);
            //check if this vm exists in proxmox
            if (typeof(qmlist[vm]) == "undefined") {
                if (vmstat[vm]['available']) {
                    //this vm doesn't exist in either place, no problem
                    console.log('VM', vm, 'doesn\'t exist on Promxox or in the database.');
                } else {
                    //this vm exists in the database, but isn't in proxmox
                    console.log('VM', vm, 'got deleted in Proxmox - update it!');
                    vmstat[vm]['available'] = true;
                    db.collection('vms').doc(vm).update({
                        available: true,
                        name: null,
                        started: false,
                        template: null,
                        uid: null,
                    });
                }
            } else {
                if (vmstat[vm]['available']) {
                    //somehow this vm got created in proxmox, but it isn't in the database
                    console.log('VM got created in proxmox, update the database')
                    vmstat[vm]['available'] = false;
                    db.collection('vms').doc(vm).update({
                        available: false,
                    });
                }
                console.log(vm, 'qm status', qmlist[vm]['status'], 'node status', vmstat[vm]['started']);
                if (qmlist[vm]['status'] == vmstat[vm]['started']) {
                    //vm exists in both, and status matches
                    console.log('Status matches');
                } else {
                    //vm got started or shutdown - update it
                    console.log('The vm got started or shutdown - update it!');
                    vmstat[vm]['started'] = qmlist[vm]['status'];
                    db.collection('vms').doc(vm).update({
                        started: qmlist[vm]['status'],
                    });
                }
            }

        }
    });
}
```

Getting which classes a student is enrolled in

```
const getclassmembership = async uid => {
    let classes = [];
    let studentsnapshot = await new Promise(resolve => {
        db.collectionGroup('students').where('userid', '==', uid).get().then(snapshot => {
            if (snapshot.empty) {
                resolve(0)
            } else {
                resolve(snapshot)
            }
        }).catch(error => {
            console.log(error);
            resolve(0);
        });
    });
    if (studentsnapshot == 0) {
        return [];
    } else {
        for (let i = 0; i < studentsnapshot.size; i++) {
            console.log('Class', studentsnapshot.docs[i].ref.parent.parent.id)
            classes[i] = []
            classes[i].id = studentsnapshot.docs[i].ref.parent.parent.id;
            classes[i].name = await new Promise(resolve => {
                db.collection('classes').doc(classes[i].id).get().then(classsnapshot => {
                    resolve(classsnapshot.data().name);
                }).catch(error => {
                    console.log(error);
                    resolve('');
                });
            });
        }
        console.log('Classes', classes);
        classes.sort((a, b) => {
            if(a.name < b.name) { return -1; }
            if(a.name > b.name) { return 1; }
            return 0;
        });
        return classes;
    }
}
```

Finding what vms a student has access to

```
if (vmsnapshot == 0) {
    return [];
} else {
    users = [];
    classes = [];
    for (let i = 0; i < vmsnapshot.size; i++) {
        vms[i] = {};
        vms[i].id = vmsnapshot.docs[i].id.substring(2);
        vms[i].name = vmsnapshot.docs[i].data().name;
        if (vmsnapshot.docs[i].data().started) {
            vms[i].started = 'Started'
        } else {
            vms[i].started = 'Stopped'
        }
        vms[i].access = vmsnapshot.docs[i].data().vmid;
        vms[i].accesscode = vmsnapshot.docs[i].data().name + vmsnapshot.docs[i].data().vmid;
        if (detailed == true) {
            vms[i].uid = vmsnapshot.docs[i].data().uid;
            vms[i].classid = vmsnapshot.docs[i].data().class;
            if(users.indexOf(vmsnapshot.docs[i].data().uid) == -1) {
                users.push(vmsnapshot.docs[i].data().uid);
            }
            if(classes.indexOf(vmsnapshot.docs[i].data().class) == -1) {
                classes.push(vmsnapshot.docs[i].data().class);
            }
        }
    }
    if (detailed == true) {
        users = await new Promise(resolve => {
            getuserinfo(users).then(users => {
                resolve(users);
            }).catch(error => {
                console.log('Error: ', error);
                resolve([]);
            });
        });
        classes = await new Promise(resolve => {
            getclassinfo(classes).then(classes => {
                resolve(classes);
            }).catch(error => {
                console.log('Error: ', error);
                resolve([]);
            });
        });
        for (let i = 0; i < vms.length; i++) {
            vms[i].username = users[vms[i].uid].name;
            vms[i].email = users[vms[i].uid].email;
            vms[i].class = classes[vms[i].classid];
        }
    }
    console.log('Users:', users);
    console.log('Classes:', classes);
    return vms;
}
```

Creating vms

```
let availablevms = await new Promise(resolve => {
    db.collection('vms').where('available', '==', true).get().then(vmsnapshot => {
        if (vmsnapshot.empty) {
            resolve(false);
        } else {
            let availablevms = [];
            for (i = 0; i < vmsnapshot.size; i++) {
                availablevms[i] = vmsnapshot.docs[i].id;
                console.log('VM ', vmsnapshot.docs[i].id, 'is available.');
            }
            resolve(availablevms);
        }
    }).catch(error => {
        console.log('Error: ', error);
        resolve(false);
    });
});
if (availablevms == false) {
    return false;
} else {
    let students = await new Promise(resolve => {
        db.collection('classes').doc(req.body.class).collection('students').get().then(studentsnapshot => {
            if (studentsnapshot.empty) {
                resolve(false);
            } else {
                resolve(studentsnapshot);
            }
        }).catch(error => {
            console.log('Error: ', error);
            resolve(false);
        });
    });
    if (students == false) {
        return false;
    } else {
        let usedvms = -1;
        let successcount = 0;
        for (i = 0; i < students.length; i++) {
            exec('qm create ', availablevms[usedvms]);
            let addedvm = await new Promise(resolve => {
                db.collection('vms').doc(availablevms[usedvms]).update({
                    available: false,
                    name: req.body.name,
                    template: req.body.template,
                    uid: studentsnapshot.docs[i].id,
                    class: req.body.class,
                }).then(result => {
                    resolve(true);
                }).catch(error => {
                    console.log('Error: ', error);
                    resolve(false);
                });
            });
            if (addedvm) {
                successcount++;
            }
        }
        if (successcount == students.length) {
            return true;
        } else {
            return false;
        }
```

Hardware Specifications

We recommend at least 2GB of RAM per student, and 1 CPU core for every 2-3 students. While the more RAM and CPU cores the better, we believe that this provides an optimal price to performance per cost per student.

Currently tested on R310, R510, and R710

APPENDIX B. REFERENCES

Marcus, Jon, et al. "Cash-Strapped Schools Are Selling Bonds to Buy Education Technology." The Hechinger Report, Columbia University, 23 Apr. 2019, hechingerreport.org/school-districts-are-going-into-debt-to-keep-up-with-technology/.
(Marcus, 2019) (Marcus, 2019)