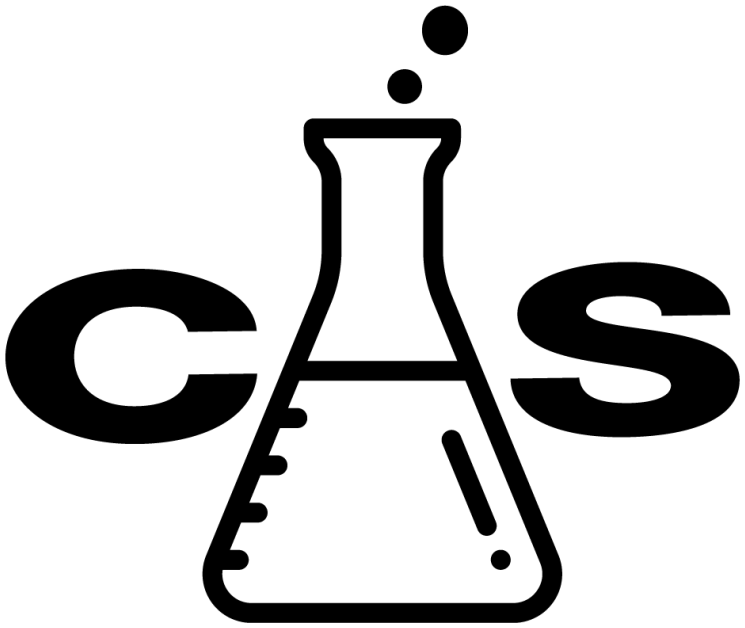


CS Labs – Web

Software Requirements Specification

Group Leader: Jason Gallavin

Members: Barel Musenga, Chase Benston

Oct 1, 2019

Contribution Breakdown and Key Personnel Information

Jason Gallavin – Project leader, contributes by creating reports, deciding architecture and larger project decisions.

Barel Musenga – Full Stack Developer, contributes by implementing the front end and back end, and parts of the lab automation.

Chase Benston – Full Stack Developer, contributes by implementing the front end and back end, and parts of the lab automation.

Introduction

Purpose

The purpose of this document is to detail the requirements of the application. The application shall serve students and other users as a learning source. The application will serve learning material in the form of interactive modules that allow users to work on real VM's in a controlled environment just through their web browser.

Definitions

User	Any person that is using the system whether it be a student or outside user
Standard User	A person with the least privileges in the system, they can use free modules
Staff	A IU Staff member that can create modules
Admin	A person who can do everything in the system
ReactJS	The frontend framework

ASP .Net Core	The backend framework
Kestrel	The ASP .NET Core web server for linux
NGINX	The forward facing linux web server that proxies connections to Kestrel
Badge	An award given in the application
Module	Package of labs to provide comprehensive learning
Lab	A single or collection on VM's networked together
VM	A Virtual Machine
Proxmox	The hypervisor software used to control VMs

System overview

The system will have a front end written in typescript. The front end will communicate with the ASP .NET Core backend API. The backend API will leverage the proxmox API to create, turn on, turn off, and destroy VMs and networks.

References

https://github.com/ius-csg/cslabs-backend

https://github.com/ius-csg/cslabs-webapp

https://github.com/ius-csg/cslabs-web

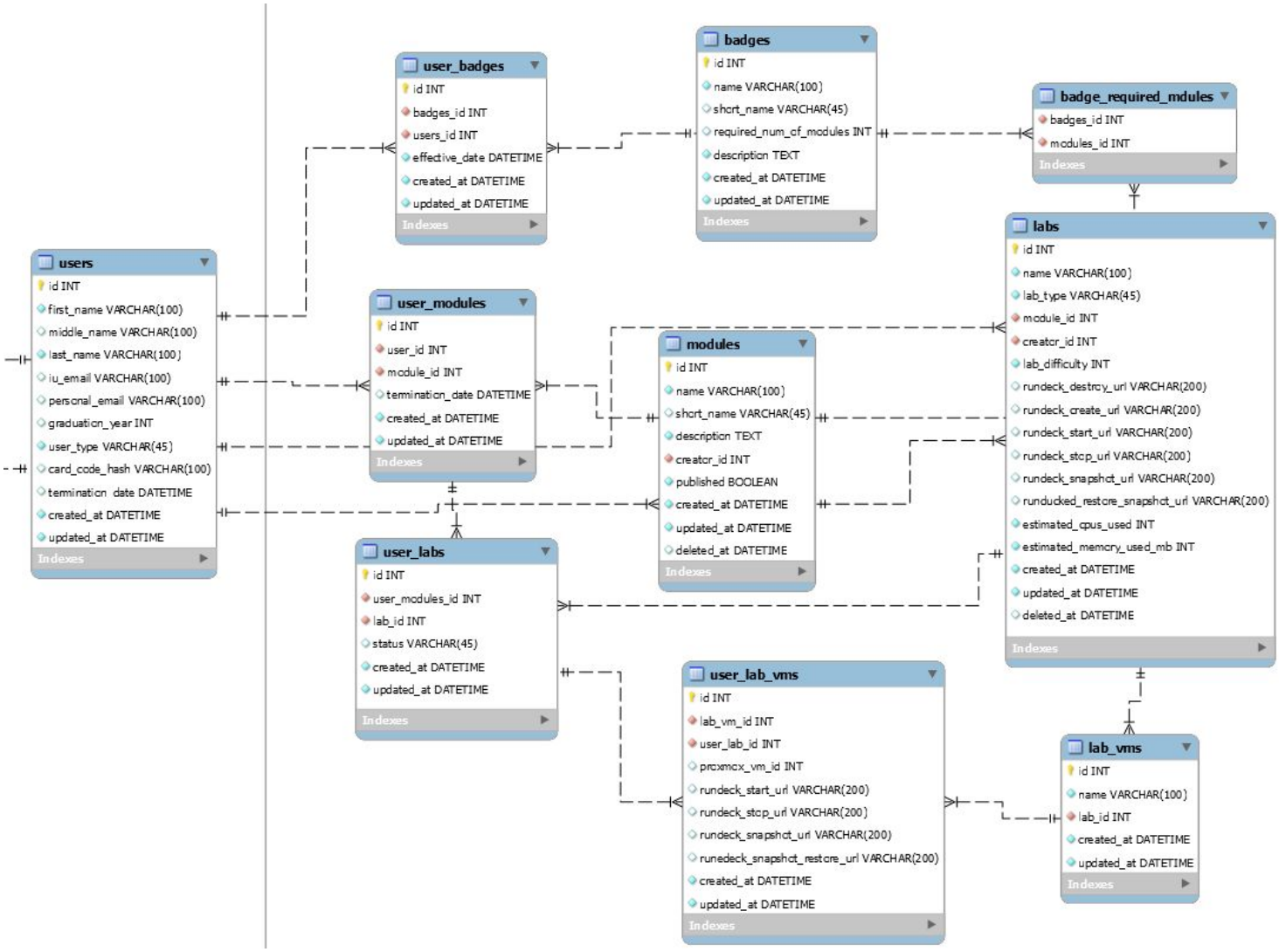
https://reactjs.org/

https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.2

https://www.proxmox.com/en/

Overall description

The system is designed to be an easy learning environment. With a click of a button, a lab is set up in a couple of minutes. The user can be sure everything will work correctly because it is in a reproducible environment. The ability to access the machines from the browser means people can access the material quicker. Users can open free modules and access the collection of labs inside them. Users can earn badges for the work that they have done.



Product perspective

System Interfaces

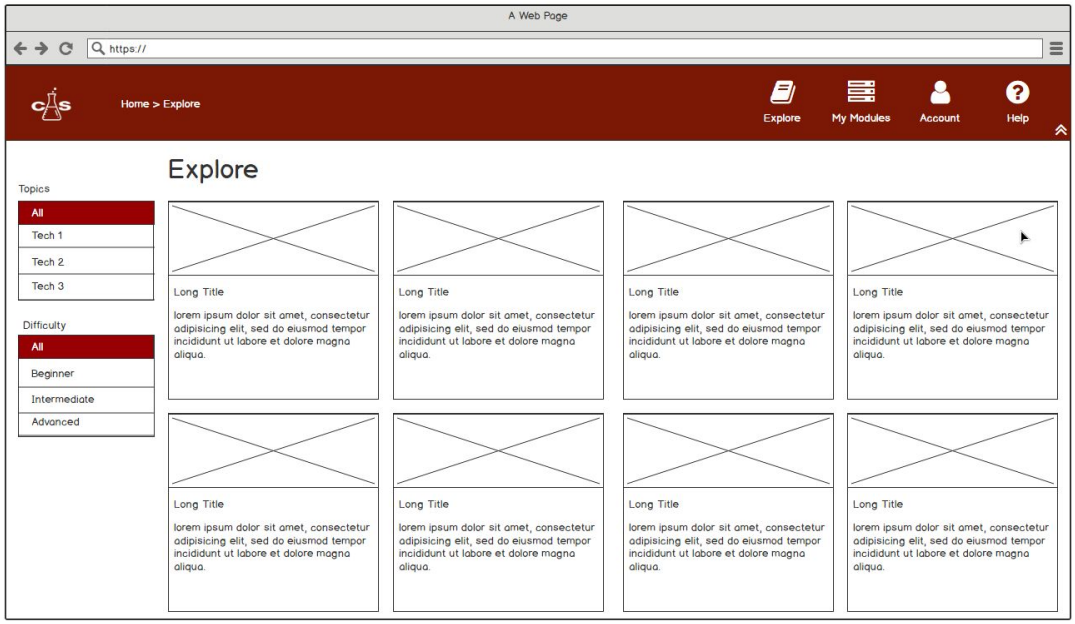
The backend HTTP API will serve as as system interface for any staff member that wants to automate actions. It also serves as an interface for the user interface. The backend uses JSON to communicate data back and forth.

The backend will leverage TLS 1.2 encryption to provide a secure connection. NGINX will server as the web server that proxies connection to Kestrel.

User Interfaces

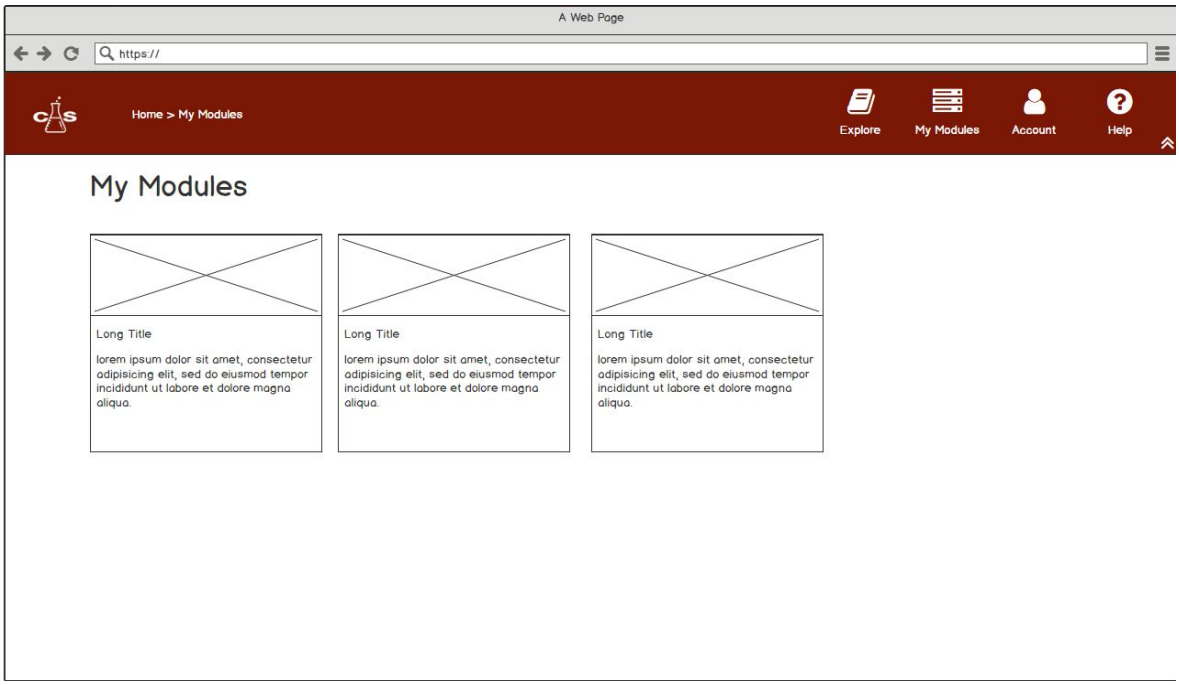
The React Typescript front end will drive the interactive experience allowing users to navigate seamlessly to other pages and VMs.

User interface Wireframes



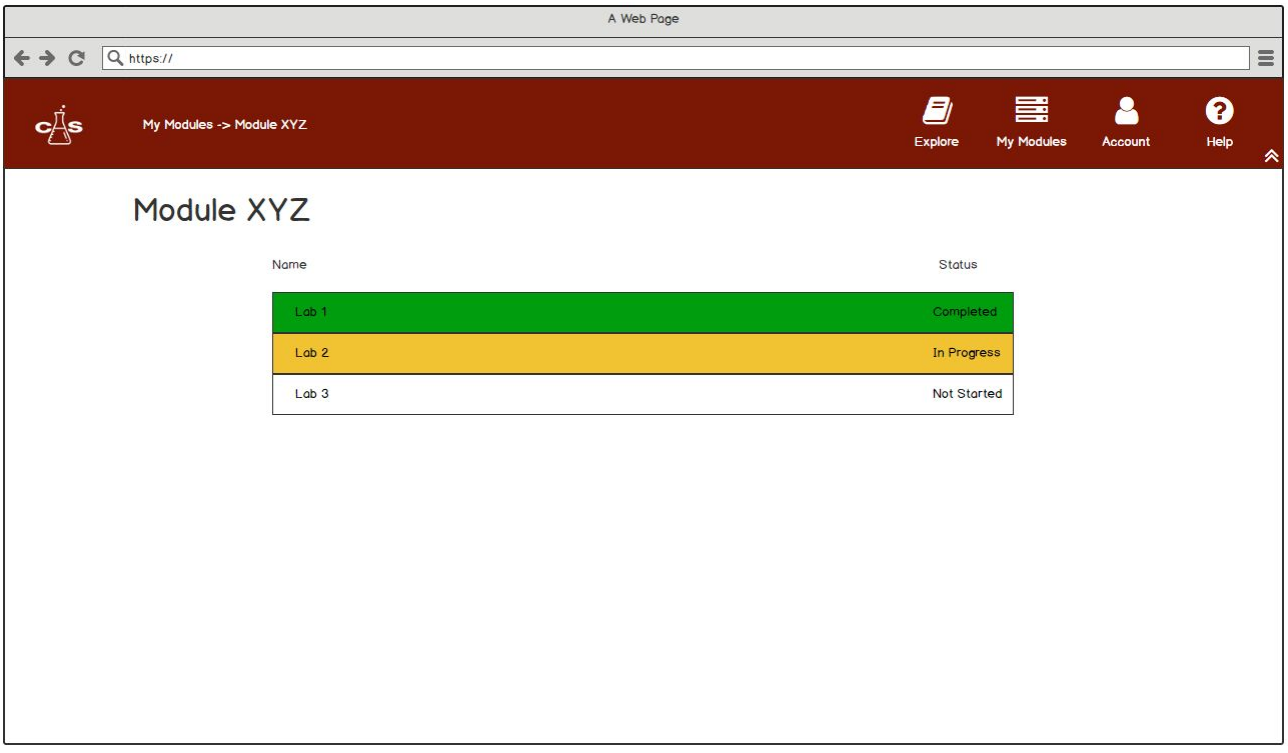
Here is where the user will select a module they are interested in.

My Modules Page



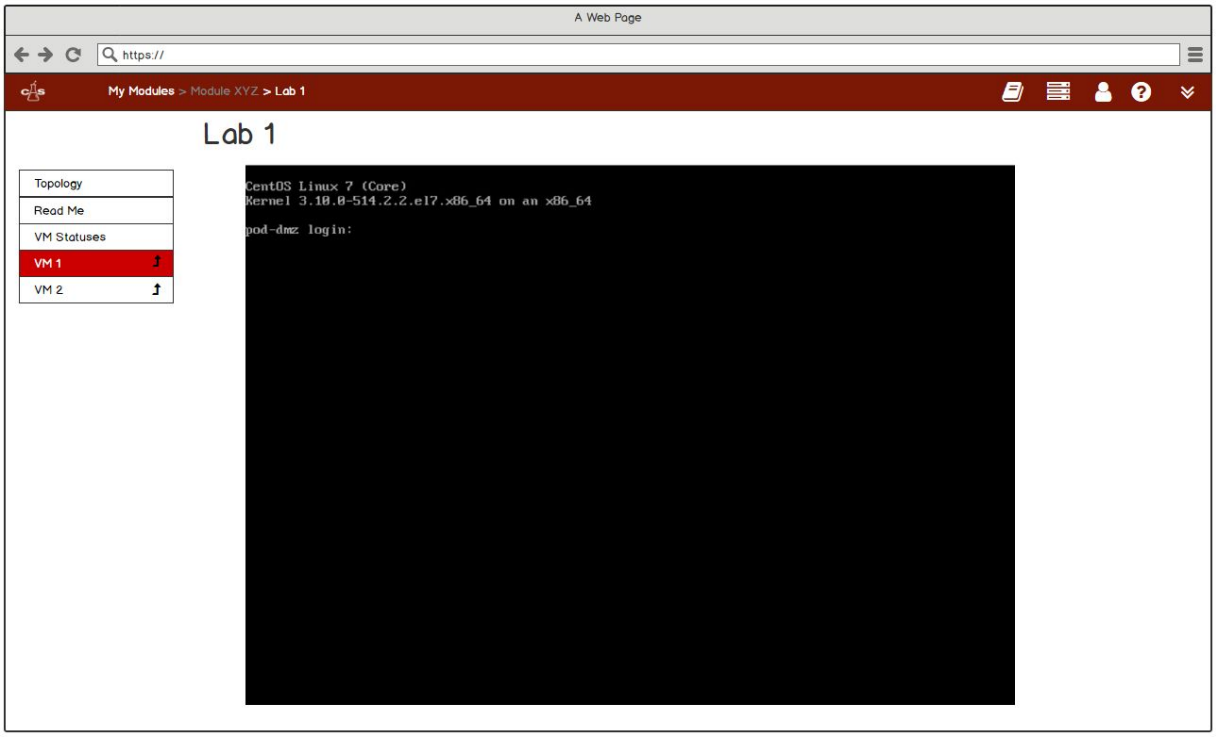
Here the user can browser modules they have started.

Module Page



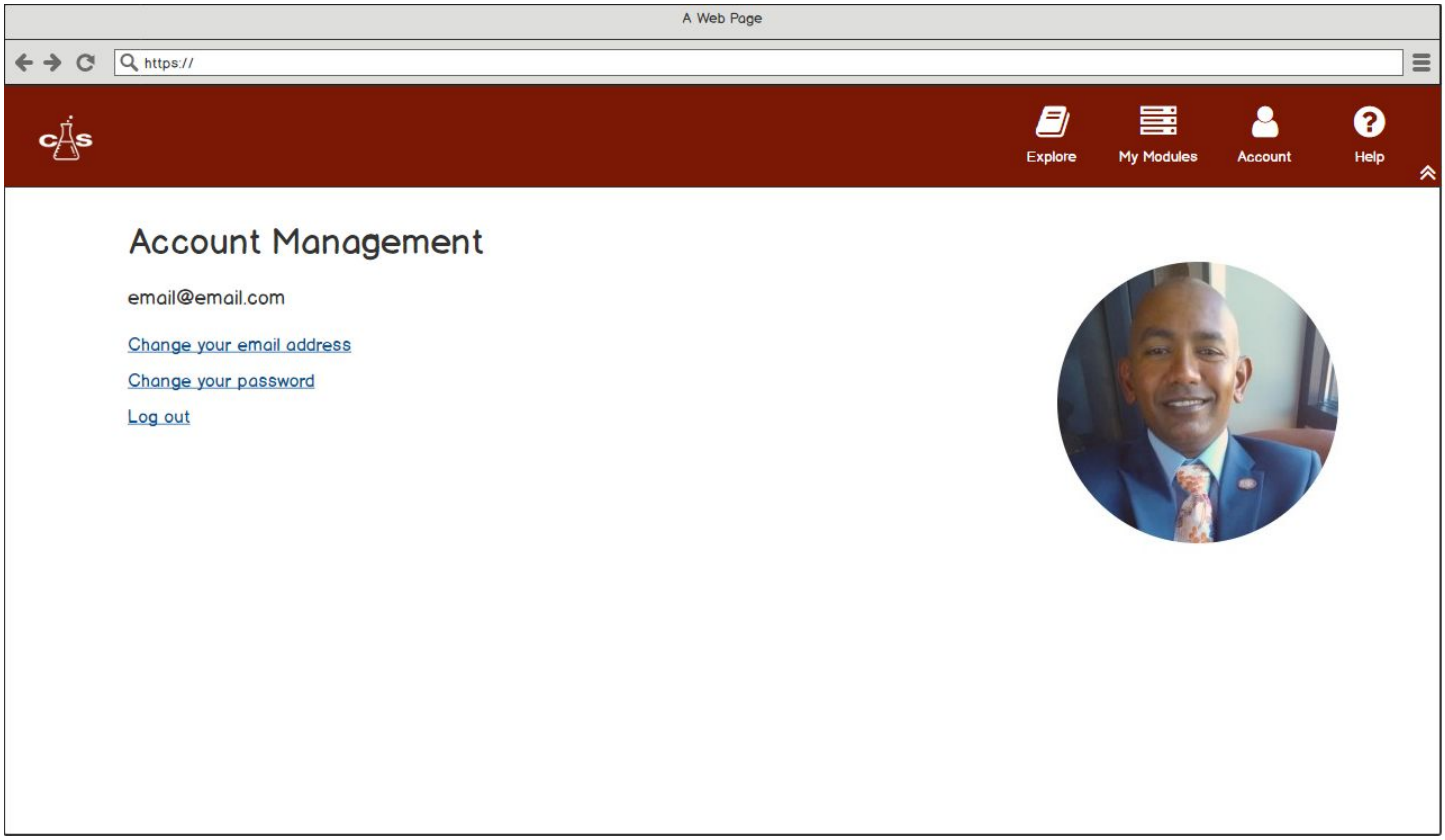
The user will select which lab they want to start. This page also shows your progression through the module.

Virtual Machine Page

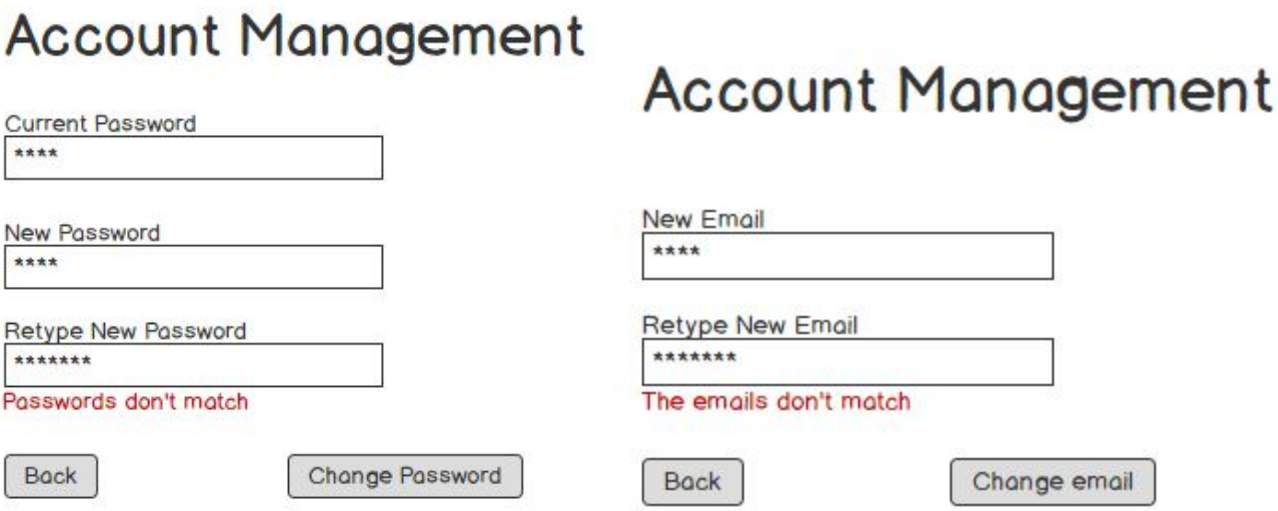


Here the user can interact with virtual machines and complete the lab.

Account Management Page



Here the user can change their email address, and password. When you press either link, the respective content shows in place. The user can upload a new photo to replace the current one.



Hardware interfaces

User Hardware Requirements

- Any OS that supports the latest of these browsers: IE 11, Edge, Firefox, Chrome, Safari.
- A browser which supports HTML & Javascript.

Backend Hardware Requirements

- Several large servers with around 32 - 64 GB of RAM each.
- Networked storage providing terabytes of data.
- VM with 2GB of ram and 1 dedicated vCPU to serve as the backend server.

Software interfaces

- The FreeIPA Kerberos server will be used for authentication providing a single password for all CSG applications.
- Proxmox will be used to manage VMs.
- A Logstash server will be provided by CSG to provide analytics on errors and performance of the application.
- Rundeck will server as an automation repository for managing VMs. This will help make communicating with proxmox a lot easier.
- The backend will utilize C# for a strong typed experience.
- The backend will utilize .Net Core to run C# on linux

- The Database (Maria DB) will be used to store the application's entities.
- Typescript and React will serve as the front end software interface to create beautiful UIs.

Communication Interfaces

This project supports the following web browser IE 11, Edge, Firefox, Chrome, Safari. Email will also be used to serve registration and forgot password messages.

Memory Constraints

The Proxmox servers will be under memory constraints due to the nature of VMs. We plan to be able to support 30 labs at once if each lab had only 1 VM. We may well succeed this number but that is the initial target.

Operations

Updates will be applied monthly to ensure the servers remain secure. Updates to the frontend and backend frameworks will also be applied to mitigate exploits and reduce our attack vector.

Site Adaptation Requirements

For a self hosted solution, we recommend a server with at least 64 GB of ram. Any Server that proxmox can run on should work. The kerberos authentication is optional which could be pointed at your own authentication server. The Logstash integrations is optional but recommended. Rundeck will be required as well to run the automations.

Product functions

User characteristics

The users of this system are generally beginner to upper level computer science students or self taught computer scientists. The lab modules will put the user straight into VM(s) to perform interactive work. This environment serves as an easy to set up solution compared to running VMs on the users machine which reduces the difficulty.

Constraints, assumptions and dependencies

Current constraints are memory requirements of the proxmox servers. Virtual machines take a decent amount of RAM and the application has to support 30 users at once. In proxmox you can set how much RAM a VM uses which is allocated fully to the machine. CPU is easier to provide because not all VMs will be running at their max capacity.

We assume there is enough memory for this project but if there is not we can collect recycled servers. There is also the assumption that our initial users on the system will be 30. If additional users are added we may have to adjust our hardware.

This project depends on the fact that multiple servers don't fail because they are older. We have multiple servers to provide redundancy. Here are a list of other dependencies in the system

- .Net Core - Backend framework
- C# - Backend language
- ReactJS - Front end framework
- Typescript - Front end language
- React Bootstrap - Theming framework for ReactJS
- Proxmox - Hypervisor to run Vms
- RunDeck - Required to run automation on Proxmox

Specific requirements

External interface requirements

- Kerberos requires the backend to communicate using it's protocol
- Proxmox requires you to use their API that is useable in .Net Core
- Rundeck provides an HTTP api which can be easily implemented in .Net Core

Functional requirements

General User

- A user must be able to register using their IU or personal email address and specify a password.
- A user must be able to login using their IU or personal email address and a password
- A user must be able to reset their password using their email address
- A user must be able to open a free module
- A user can open a private module if given access by a staff member or admin.
- A user must be able to turn on a lab if they have access to
- A user must be able to use multiple vms
- A user must be able to undock a VM into a new window

Staff

- A staff member must be able to create their own modules by uploading VM images and network schematics
- A staff member must be able to upload VM images
- A staff member must be able to specify network requirements

Admin

- An admin can change the role of any user
- An admin can do anything staff and a standard user can do

System

- The system should setup new vm's from images and turn them on
- The system should shut down a lab when a user is finished with it
- The system should create new networks from network schematics
- The system should email a user when they register
- The system should email the user when they request a forgot password email

Performance requirements

Labs must be able to be created in just a few minutes. If users have to wait too long, then it will ruin the whole experience. SSDs will be used on priority labs to speed up the process. The web frontend will only require as much resources as the web browser you are using. The application is very light and can be run on older machines with 2 GB of ram using a lightweight browser.

Design constraints

Standards Compliance

The application Abides by the HTTP and HTTPS standards allowing browsers to utilize the backend. In the future the application may take payments which will subject the application to The Payment Card Industry Data Security Standard. We will gain compliance by keeping the server up to date, and use a third party payment system.

Logical database requirement

The backend relies on MariaDB which is an alternative to MYSQL. MariaDB provides normalization of our entities allowing constraints to be applied to the relationships. The amount of storage space will be fairly minimal because we are only storing information about modules, labs, users, badges, and VMs.

Software System attributes

Reliability

The application contains automated tests alone with manual tests performed by the team. End to end tests help validate that things work in the real world when everything is communicating. These Unit, integration, and End to end testing will be performed before production releases to minimize bugs. When a bug is reported we will be on it in 24 hours.

Availability

During the first 3 months of use we guarantee 95% uptime, after that it will go up to 99.9%. In the early phases we will be providing rapid updates as features arrive. We have multiple nodes in our proxmox server to provide a HA environment.

Security

The servers will be running the latest fedora distro with the latest .NET Core runtime. The CSG group will be handling firewall rules on an external firewall to enhance the security of the application. NGINX will be used to proxy requests to the .Net Core Kestrel web server providing a tcp layer protection for the internal service.

Maintainability

The backend and front end systems are built in .Net Core and ReactJS in Typescript respectively. .Net Core is the future of Microsoft's efforts to expand the use of C# to other operating systems. The overall design of the architecture is simplified with .Net Core. ReactJS provides an easy to use and componentized front end to increase code reuse and reduce bugs.

Portability

The frontend is very portable as it can run in all the major browsers. The hosting of the VMs require significant investment of memory and disk heavy servers. Users can use either their school address or personal address allowing outside users to use this system as well.