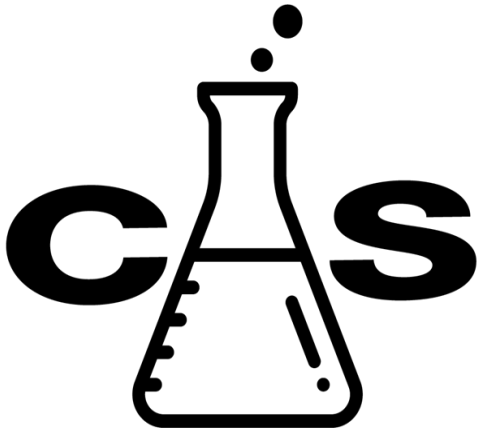
CSLabs Project Extension, Team 1 – 2021–2022

Software Requirements Specification Report



Project Team:

Jason Henry, Justin Hurst, Zaid Hussain

September 28th, 2021

Faculty Advisor:

Dr. Ronald B. Finkbine

Associate Professor of Computer Science

Key Personnel Information and Contribution Breakdown

Zaid Hussain – Project Leader, contributes to the analysis reports, the frontend and backend by implementing additional, essential functionalities

Justin Hurst – Full Stack Developer, contributes to the frontend and backend by implementing additional, essential functionalities

Jason Henry – Full Stack Developer, contributes to the frontend and backend by implementing additional, essential functionalities

1. Introduction

1.1 Purpose

CSLabs is a virtual lab learning environment created and operated by the Computer Science Group (CSG) at the Indiana University Southeast (IUS). It is used by IUS faculty and students to practice computer security concepts and other areas in computer science using virtual machines (VMs) that operate on high computing servers at the campus. CSLabs will serve the CSG's mission of providing technical education and hands on experience to its members and educating the public about security awareness.

The project is an extension of the CSLabs 2019-2020 Capstone projects. The purpose of this extension is to add new functionalities to the CSLabs project and expand the overall system capability.

1.2 Definitions

The following definitions were originally defined in the previous SRS report by Gallavin et al, organized by categories.

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, allowed to use public modules
Staff	An IU Staff member that can create and edit modules
Admin	A person who can do everything in the system
ReactJS	The primary frontend library
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	A reward given for the work users have done in the application
Module	Package of labs to provide comprehensive learning
Lab	A single or collection of VM's networked together
VM	A Virtual Machine
Proxmox	The hypervisor software used to control VMs

1.3 System Overview

Software

The system's frontend is written in Typescript with React. The front end will communicate with the .NET backend API. The backend will leverage the Proxmox API to create, turn on, turn off, and destroy VMs and networks.

Infrastructure

According to Clifton et al. (2019), CSLabs runs on donated hardware equipment that is clustered and spawns multiple virtual environments in the CSG datacenter. The datacenter is currently residing in IUS Natural Science building on a ten Giga-bit intranet and one Giga-bit internet connection. Internet connectivity is tunneled through IU's campus network and utilizing a cloud hosted router (CHR) service.

1.4 References

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

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

https://reactjs.org/

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

https://dotnet.microsoft.com/

2. Overall Description

CSLabs provides users access to virtual machines via modules or labs thru a web application using a modern web browser. The VMs operate on high computing servers in the CSG datacenter at the IUS campus. The system is designed to be interactive and easy to learn.

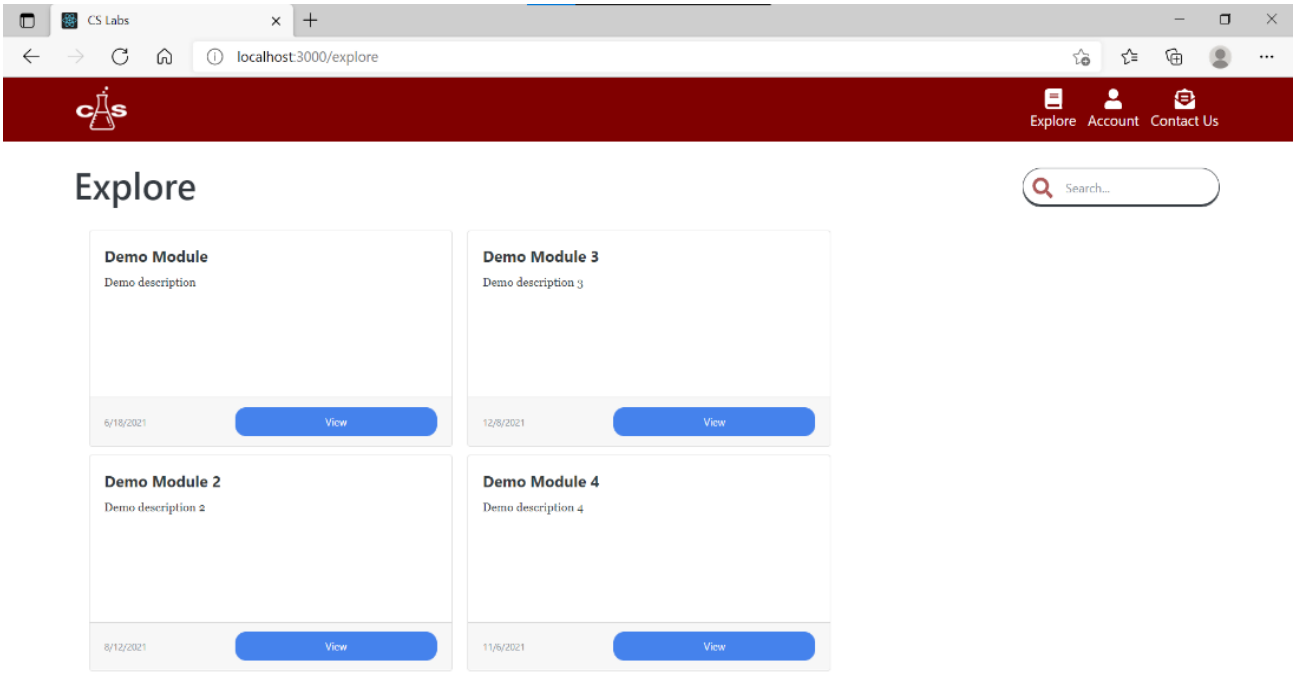
2.1 Product Perspective

a. System Interfaces

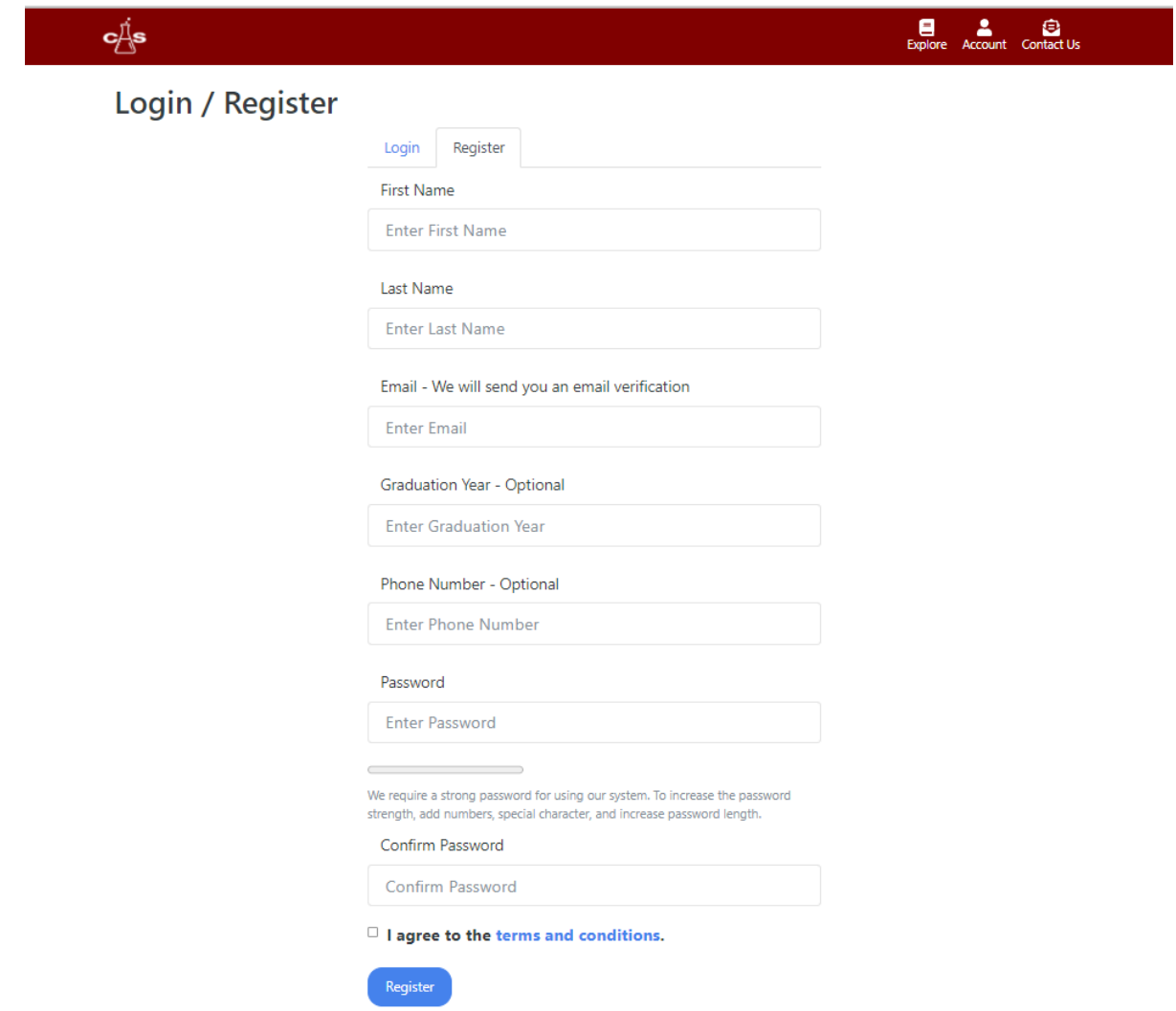
According to Gallavin et al. (2019), the backend HTTP API serves as the system interface for any staff member that wants to automate actions. It also serves as an interface for the user interface. The backend uses JSON to communicate data back and forth. The backend leverages TLS 1.2 encryption to provide a secure connection. NGINX will as the web server that proxies connection to Kestrel.

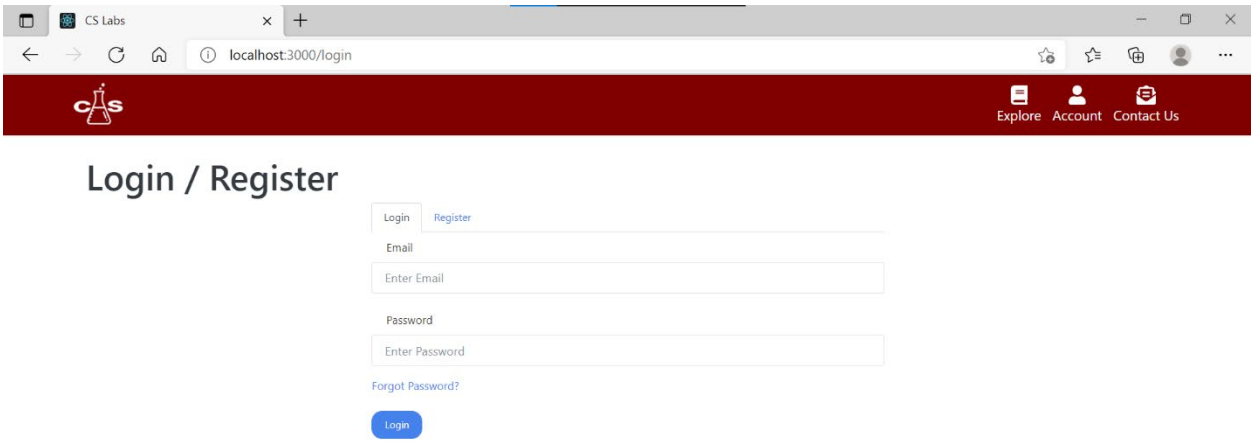
b. User Interfaces

The user will interact with the web application written in Typescript using React which drives the interactive experience allowing users to navigate seamlessly to other pages and VMs. Below are a few screenshots to provide a glimpse of the current application.

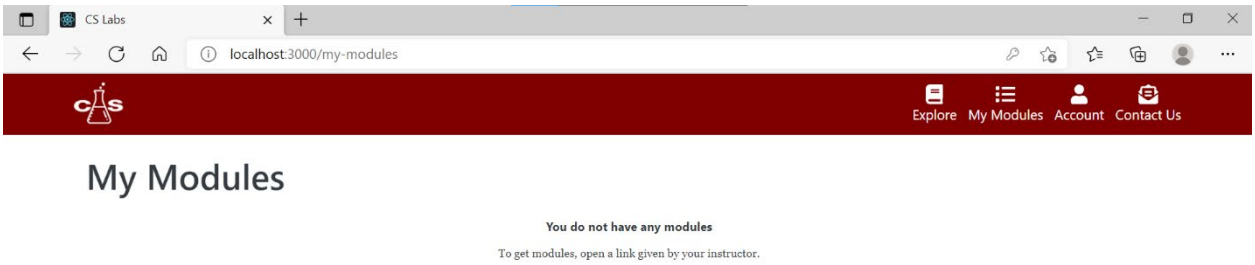


This is what a user will see to explore the public modules





The above is what a user will see for login/registration



This is what a registered user will see, they will have access to their own modules

c. Hardware Interfaces

User Hardware Requirements

- Any OS with a browser that supports the latest version of the browsers: Firefox, Chrome, Edge, Safari, IE 11.

Infrastructure Hardware Requirements

- Several high computing servers with ~ 32 – 64 GB of RAM each.
- Networked storage providing space for terabytes of data.
- VM with 2GB of RAM and 1 dedicated vCPU to serve as the backend server.
- 10 Giga-bit network switches and routers.
- Backup infrastructure including storage in case of hardware failures.

d. Software Interfaces

The following passages are derived from the 2019 - 2020 SRS report per Gallavin et al. (2019)

- The FreeIPA Kerberos server is used for authentication providing a single password for all CSG applications.
- Proxmox is used to manage VMs.
- A Logstash server o provide analytics on errors and performance of the application.
- Rundeck will serve as an automation repository for managing VMs. This will help manage communications with Proxmox much easier.
- The backend will utilize C# for a strong typed experience.
- The backend will utilize .NET to run C# on Linux
- The DBMS MariaDB will be used to store the application's entities.
- Typescript with React will serve as the front end software interface to create beautiful UI.

e. Communication Interfaces

CSLabs content is delivered using any supported modern browser using HTTPS. Email is used for account creation and other services. Other types of communication may be delivered during CSG meetings and class lectures.

f. Memory Constraints

The Proxmox VE servers cluster can face memory constraints due to large memory requirements of running multiple instances of VMs given the infrastructure. CSG expects to support running up to 30 VM instances simultaneously.

g. Operations

According to Gallavin et al. (2019), updates are expected to be applied monthly to ensure the security of the servers. Updates to the frontend and backend frameworks will also be applied to mitigate exploits and reduce our attack vector.

h. Site Adaptation Requirements

According to Gallavin et al. (2019), a server with at least 64 GB of RAM is recommended. Any server that Proxmox can run on should work. The Kerberos authentication is optional. The Logstash integration is optional but recommended. Rundeck will be required to run the automations.

2.2 Product Functions

This system will have the functionalities listed in the next section, and for other functions, please refer to the relevant sections for specific information.

2.3 User characteristics

The system's users are generally freshman to senior computer science students at IUS. The users are assumed to be equipped with basic computer knowledge, and knowledge about networking or computer security would be beneficial.

2.4 Constraints, assumptions, and dependencies

According to Gallavin et al. (2019), Proxmox servers' memory requirement is the largest system constraint. Running multiple VM instances can take a large amount of RAM, and the system is planned to support up to 30 VMs simultaneously. To optimize the overall system performance,

VM memory allocation and CPU threads can be configured depending on the available hardware resources.

The following is a list of other dependencies in the system:

- .NET Core - Backend development framework
- C# - Backend programming language
- ReactJS - Front end development library
- Typescript - Front end programming language
- React Bootstrap - Styling library for ReactJS
- Proxmox - Hypervisor to run VMs
- Rundeck - Required to run automation on Proxmox

3. Specific requirements

3.1 External interface requirements

- Kerberos requires the backend to communicate using its 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

3.2 Functional requirements

Our team expects to at least complete the implementation of the following functionalities, the functionalities will most likely extend depending on the circumstances:

- Implement important new features to the existing CS Labs web application.
- Create an API and a graphical interface for creating the labs in the modules section.

- Add available modules with details for unregistered users to explore.
- Create user friendly interface to easily navigate, configure, and manage modules for the registered users.
- Add options to the module's lab page where the user is able to access and interact with the Virtual Machines (VMs).
- Add options to VMs such as being able to manipulate VM's state, capture snapshots, etc.
- Create a topological diagram of the lab infrastructure, where the user is able to manipulate the diagram for manual configurations.
- Implement basic to advanced searchability features where the user is able to search and filter the modules.
- Implement features to allow registered users to make advanced configurations the modules and labs.
- Add functionality for the creators to be able to manage the modules, and perform tasks such as adding, removing, and disabling modules.
- Implement the modules and environments needed, a backbone of the internet environment.

3.3 Performance requirements

Labs must be created within few minutes. Delay in this process will result in unsatisfactory user 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. The application is very light and can be run even on older machines with 2 GB of RAM using a lightweight web browser.

3.4 Design constraints

a. 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.

3.5 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.

3.6 Software System attributes

a. Reliability

The application contains automated tests alone with manual tests performed by the team. End-to-end tests help validate the system is functional in a production environment 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 start working on it within 24 hrs. depending on how critical it is.

b. Availability

Assuming the infrastructure does not result in any failures, during the first 3 months of use we guarantee 90% uptime, after that it will increase to a much higher rate. 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.

c. Security

The servers will be running the latest fedora distribution 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.

d. 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. The .NET framework is used by a large number of projects, and there are continuous contributions made to it to ensure its compatibility and performance. ReactJS provides an easy to use and componentized front end to increase code reuse and reduce bugs. It also provides faster rendering by only updating the DOM with the changes and providing stable code overall.

e. Portability

The frontend is very portable as it can run in all the major browsers. The hosting of the VMs requires 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.

References

Gallavin, Jason et al. " CS Labs – Web Software Requirements Specification." 1 Oct. 2019, https://github.com/ius-csg/CSLabs-Capstone-Documentation/tree/master/cslabs-web-2019-2020/DesignDocs. 27 Sept. 2021

Clifton, Zac et al. " CS labs Infrastructure Hardware Requirements Specification for CS labs Operations and Application." 14 Oct. 2019, https://github.com/ius-csg/CSLabs-Capstone-Documentation/tree/master/cslabs-Infra-2019-2020/REPORTS. 27 Sept. 2021