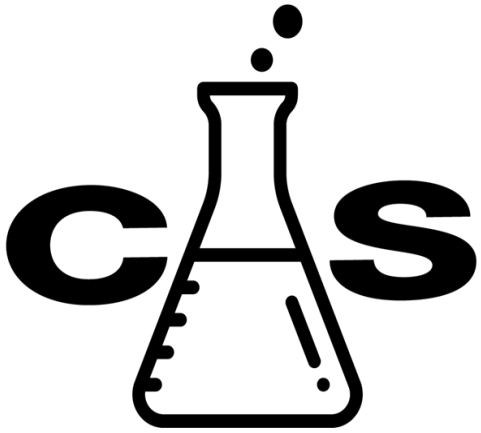
CSLabs Project Extension, Team 1 – 2021–2022

Software Requirements Specification Report



Project Team:

Jason Henry, Justin Hurst, Zaid Hussain

October 12th, 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

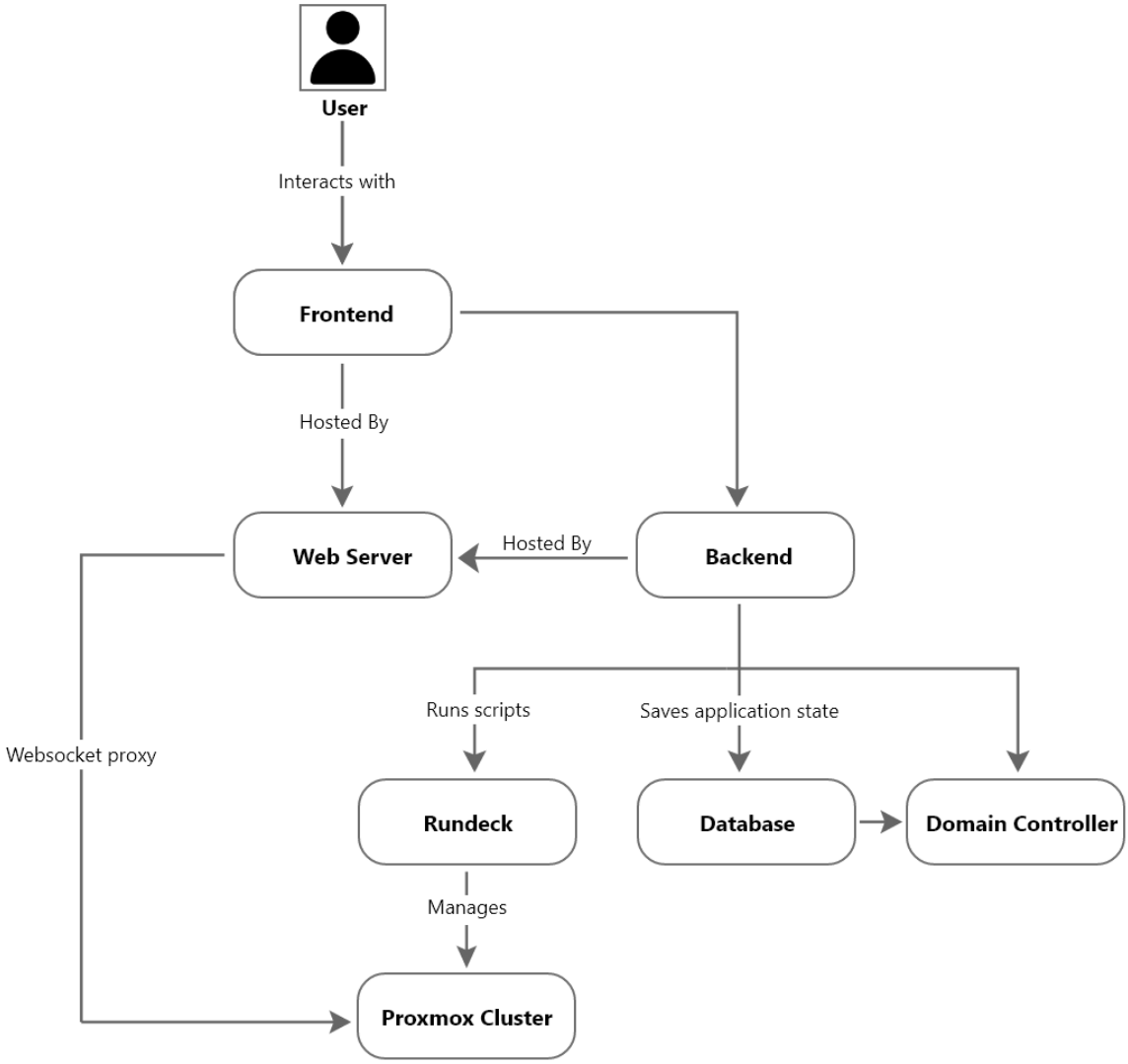
Jason Henry – Full Stack Developer, contributes to the frontend and backend by implementing additional, essential functionalities

1. Software Architecture

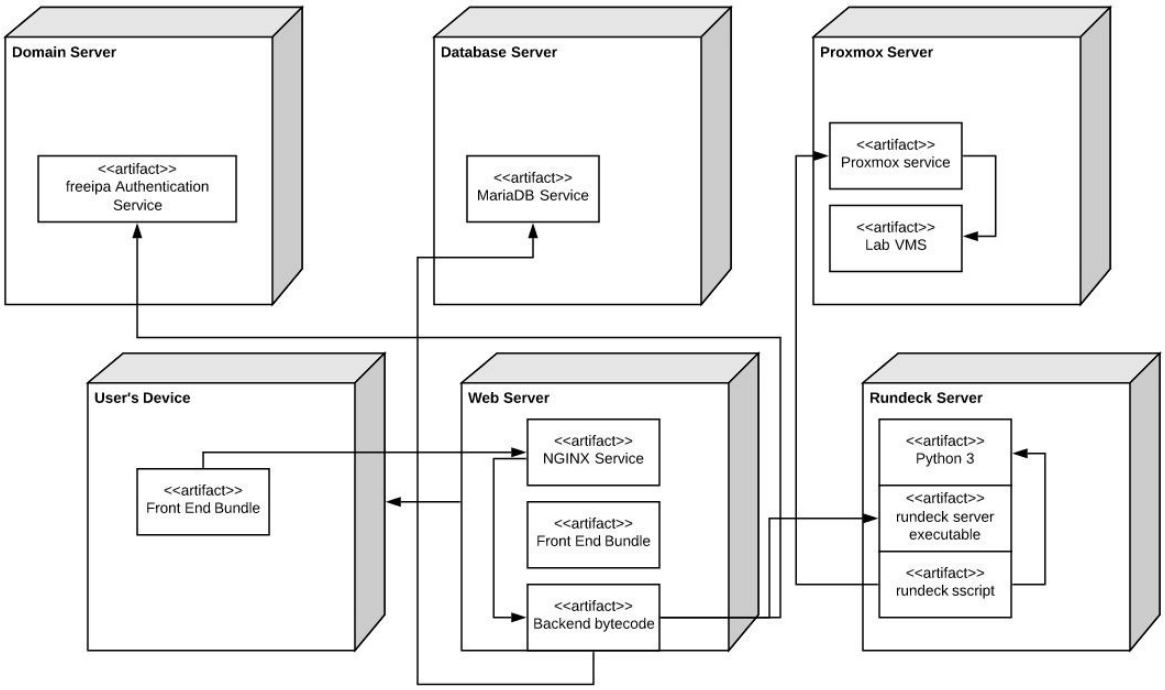
1.1 Overview

This software system is composed of the following parts: frontend, backend API, domain controller, rundeck, database, and the Proxmox cluster. The frontend is built with React and written in Typescript to provide maximum type safety with JavaScript during the development of this system. The frontend will make API calls to the backend for data access and authentication. The backend will be built with .NET Web API in C#, which also provides maximum type safety for validating requests. .NET allows us to host backend application on a linux server, which avoids any licensing costs. The backend uses the domain controller for authenticating users and rundeck to issue scripted commands to proxmox. Proxmox will be running, and housing VMs that are controlled by these rundeck scripts.[1]

1.2 Subsystem Decomposition

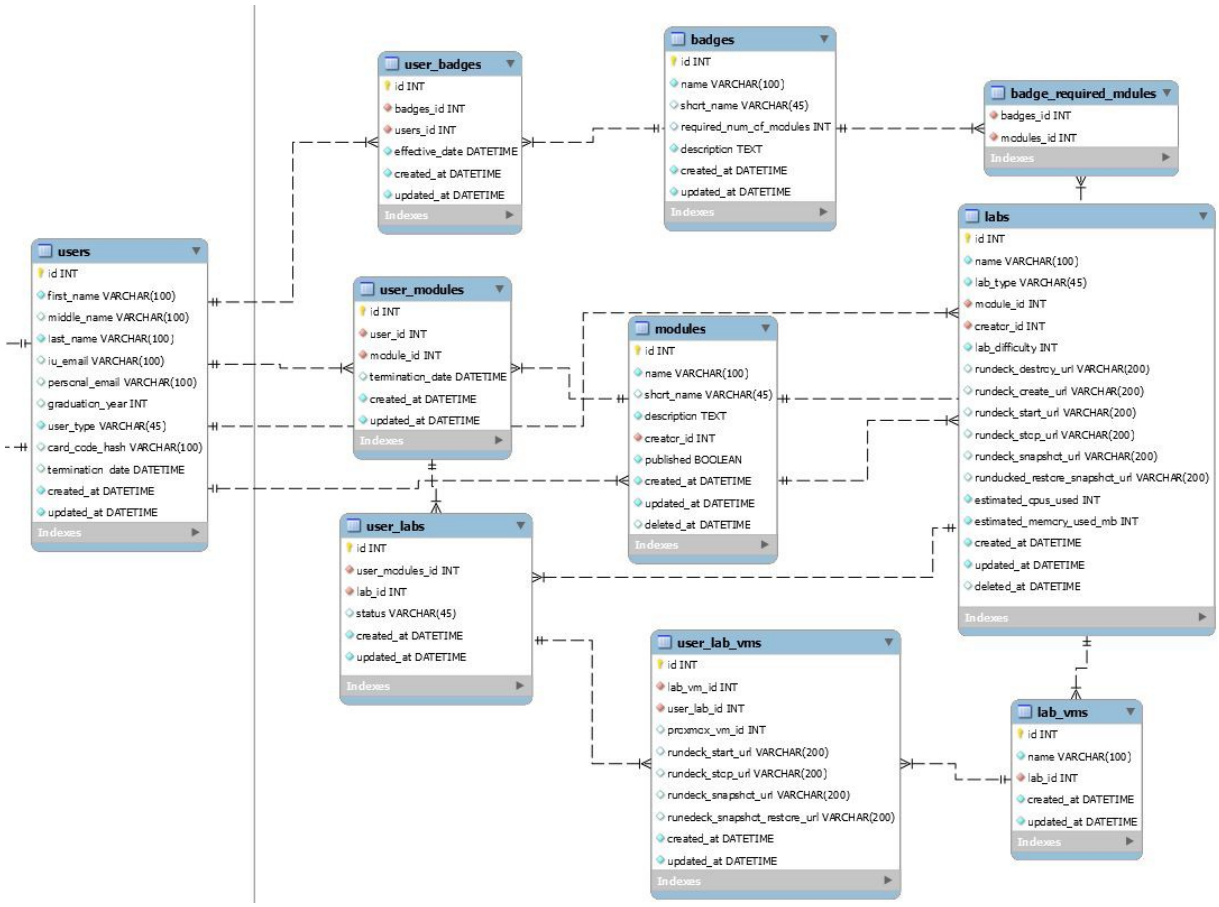


1.3 Hardware/software mapping



1.4 Persistent Data Management

MariaDB will be used as the database management software for this project. This database provides a SQL interface to store and retrieve application state. The database will reside on a separate server where the web server can access it with low latency. The application's entities are shown below in the database diagram. Most of the entities follow standard CRUD operations. The user_modules,user_labs, and user_vm's show instances of each entity which are created when a user starts the module. [1]



1.5 Access Control and Security

Each user is authenticated using a username and password. The default user group is user.

Only an admin has the ability to promote users to higher permission groups.

Admin

- Can do everything staff and User can do.
- Can create public modules
- Can promote users to staff or admin

Staff

- Can create private modules
- Can edit their private modules
- Can set a start and end date for the private modules they own.
- Can share their module using a share link

User

- Can start a module
- Can change their email address
- Can change their password
- Can start up a vm in their own lab
- Can shut down a vm in their own lab
- Can create a snapshot of a vm in their own lab
- Can restore a snapshot of a vm in their own lab

1.6 Global Software Control

The control flow of the backend is asynchronous. It uses a thread pool to run tasks in the background and start on the next line of program after the asynchronous call. This allows for efficient handling of API requests without blocking new connections.

The front end uses asynchronous and event-driven control flows. Asynchronous code flows as described above are used for network requests. However, JavaScript is single threaded, there

is no threadpool. Asynchronous methods are crucial to keeping the UI smooth. Button clicks and input changes are tracked using event based code flow. When the login button is pressed, it fires an event that initiates the form checking process.

1.7 Boundary Conditions

The backend system can be initiated using the dotnet CLI runtime. This is built into a systemd service that can be started and stopped. Running the command dotnet EF database update will update the database's schema to match the version deployed. This will run automatically when a new version of the system is deployed. Shutting down the backend can be triggered with the standard systemd stop command. The backend application will utilize logstash to log all unhandled errors into a database for logs. From there we can analyze the errors from a graphical UI enabling enhanced debugging. The front end will handle errors the same way. Any expected error like unauthorized or bad request are handled gracefully in the frontend to tell the user what went wrong.

As with any software, it should be frequently updated, or security threats might go unfixed. The frontend and backend dependencies should be updated at least every 6 months. When replacing servers, we will only need to export the MariaDB database using mysql workbench and import it into the new database server. The backend will store profile pic uploads on disk at /var/www/cslabs-backend/assets/img/profile/, A new version of the application must be built to the /var/www/cslabs-backend directory. The frontend is stored at /var/www/cslabs-webapp/. The NGINX configuration should also be copied over at /etc/nginx/sites-enabled/default

2. Basic Component Types

2.1 Use Cases

Teachers need a way for their students to easily have access to a working environment. A student may not have a machine that can run a high scale VM. The student can use our lab environment to work on their assignment regardless of the computer they are using.

Public users will come to this site to learn. There will be a few free public modules that users can learn from. This will gain the product popularity to pave the way for future paid modules.

2.2 Functions

a. **Log In**

   The login request is a stateless function that returns user's authentication token when valid credentials are provided, else access is denied.

b. **Navigating to public pages**

   Navigation to any public page only takes the url as input and returns the page without any contextual knowledge. Given the public page url, user will always get the correct page.

c. **Frontend components**

   Many of the frontend components are pure functions that accept arguments and return UI elements that provide the same output given the same input. Other components have state, mostly pages that are not functions.

2.3 Triggers

For the teachers, we would say it would be a time trigger. Students might take too long to set up the software locally which puts time pressure on the assignment. With this application, students can quickly dive into the concepts without muddling with configuration.

2.4 Data Stores

The MariaDB service is used to store the data. At rest, the data will sit there until it is deleted or modified by the backend. This will serve as a central store for the backend. The proxmox server will also hold data about the VM's statuses. This will sit on the proxmox server's storage.

2.5 Data Flows

There is a data flow between the webserver and the database server. The web server will need to retrieve entities from the database. Once retrieved from the database, the data flows over http in the form of a response to the frontend where the UI is then updated. Data also flows from the backend to the rundeck server and then to the proxmox server when managing VMs.

2.6 Data Elements

Badge	A reward given to a user for completing certain requirements
BadgeRequiredModule	Defines a required module for a badge

Lab	A specific lab in a learning module that has one or more VMs
LabVm	A VM template for a lab
Module	A learning module that houses a collection of labs
User	A user of CSLabs system
UserBadge	A user's instance of a badge when they have earned it.
UserLab	A user's instance of a lab
UserLabVm	A user's instance of a VM that can be run
UserModule	A user's instance of a learning module

2.7 Processors

There are several processors in this system. The web browser on the user's computer processes the frontend code. The backend server is processing the requests using .NET. The users are going to be processing step by step guides and performing actions on the VMs.

2.8 Data Storage

The database storage will be on a VM with daily snapshots. The disk is stored on the infrastructure's proxmox instance which uses RAID to increase redundancy. High volume hard drives are used as the physical storage device.

2.9 Data Connections

There will be a data connection between the user's device and the web server using HTTP as the transport protocol. The backend then has a data connection with the database during database calls using TCP with MariaDB protocol. Communication between the backend and the rundeck server will use HTTP. The rundeck server also uses HTTP to call the proxmox API. The application will utilize mailgun to send emails via SMTP.

2.10 Actors/External Entities

The system must interface with namecheap for access to the domain name. The system will have to interface with teachers to create modules and labs for their students. The software will also interface with students or public users to present labs.

References

1. 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
2. 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