CSLabs Project Extension, Team 1 – 2021–2022

Test Plan Document

For

Indiana University Southeast

January 25th, 2022

1 Introduction

This document will serve as a test plan to provide a list of tests that should be performed for each feature. Most of the testing aspects still hold from the previous project, and there have not been significant changes made to it. Each test will either be performed automatically or manually before deployment to ensure that new features are working properly. These tests will be run whenever additional features are added, or refactoring occurs to ensure working features stay working. There are automated processes that run when a pull request is open including building the project, checking for formatting errors, and known security issues. This ensures that the new feature meets the requirements before being able to merge the changes. To ensure the production system is performing as expected, status checks will be added to common fault points to ensure the system is fully operational.

2 Business Background

CSLabs is a web application that helps student learn in controlled virtual environments. This allows students to have a more hands on experience with software that is considered difficult to access due to either weak hardware or difficulty of setup. The Computer Science Group (CSG) at IUS would like student developers to work on CSLabs to implement enhanced, essential features and help deploy the application to production.

3 Test Objectives

The objective of these tests is to ensure that the application's accuracy does not affect negatively as features are added. Before a feature is ready to be merged, it is reviewed by the project manager and the lead developer to ensure the requirements have been completed. These tests will ensure that application's downtime is minimized, bugs are minimized, and the application state's risk of corruption is reduced. Backups are taken in case an event of corruption is found.

4 Scope

This section includes the scope of the tests, what will be covered and what will not be covered.

Includes

- Connection with Proxmox
- Feature Accuracy

- User interface Tests
- Database transactions
- Database rollbacks in communication failure scenarios

Excludes

- Hardware failure (handled by the infrastructure members)
- Web server and database server backups (handled by the infrastructure members)
- Bandwidth performance tests

5 Test Types Identified

There are several test types to be implemented. **Unit tests** will be performed on the backend. **Integration tests** will be performed on the backend and frontend to ensure components such as Proxmox and the backend communicate correctly, the frontend performing correct actions under certain scenarios, and the backend properly interactingwith the database. **End to end tests** will be performed on the application to ensure a lab can be created and connected to.

6 Problems Perceived

Performing full end to end testing of the application will be difficult since all the components need to be in place to work. Testing is generally performed in a mirrored staging environment but if a change is made by the infrastructure members that is not communicated, end to end testing may not guarantee production is working as intended.

7 Architecture

Frontend:

The frontend is written in React which is popularly used to design Single Page Applications (SPA) and its code is located in separate repository than the backend. It is not directly dependent on the backend, a different backend could be developed in its place, and it will continue to function. It is written in Typescript, SASS, and TSX utilizing the react library to render state to the screen using some class and functional components. The features for this project are implemented using functional components. The functional based approach makes testing components and normal function easier. Asynchronous operations are heavily used to perform

IO operations between the backend.

Backend:

The backend is written as a CRUD interface with some REST interaction. The backend is written in .NET with C# using an Object-Oriented approach. An MVC architecture is used, with Models that map to the MariaDB database, the Views which is primarily JSON (handled automatically) or email templates written in Blade, and the Controllers which handle the HTTP requests to power the application logic. Dependency injection pattern is used in order to facilitate unit tests and integration tests.

8 Environment

The frontend is running on the User's browser. Any user on the internet will be able to access the application as the front end, backend, and websocket connection to Proxmox are proxied through an AWS server. The backend runs inside the .NET runtime running on a CentOS server. The Proxmox servers are running on hardware located in the IUS campus in a secured room away from unauthorized personnel.

9 Assumptions

For this project to work continuously, we had to make large assumptions. We assume the servers we have will continue to work with minimal failures, and we wish to gather more servers for scalable use. Since we are using donated servers without many servers on hand, if enough of them run into hardware failure, it could result in a catastrophic event.

We also assume IUS does not accidentally break our routing to the internet through the IU network. This can happen which would be completely out of our control and would have to go through IUS IT administrators to fix it.

For the coming years, the infrastructure and the codebase need to be maintained, at a bare minimum dependency will need to be updated. CSG members can take this as an opportunity to add features and benefit from real world development experience on an application used in production.

10 Functionality

Constraints and Resolutions

Parameter	**Customer Constraints**	**Infosys Limitations**
Registration	User must be able to create an account	Application successfully pulls information from the register form and creates a User in the database
Proxmox API	User must be able to access and control VM(s)	The Application uses APIs to allow the user to control the power status of VM(s) and makeuse of their function for labs
Account Management	The user must be able to alter their password and email	The application correctly modifies data for the user entry in the database
Login	A user must be able to sign in	The application performs authorization checks when auser logs in
Read only data for users	The user must not be ableto make changes to anything in the modules outside of using the VM(s)	The application allows no control to the user to change the modules' content
Adding content	The lab creators must have a GUI to create new labs	The application will allow for labs to be created and stored in the database
Admin Panel	An admin must be able to perform upper-level operations such as managing users, clusters, schedule downtimes, and observe application statistics	The application performs the action successfully and updates the database as necessary

Risk Identified & Mitigation Planned

Untested code will not be on the production server or database. All code will be tested locally by the author. Problems could arise if the production server loses connection.

Test Strategy

APIs are tested using Postman, then unit tests are performed on each new piece of code responsible for a single functionality before being added to production.

Automation Plans

Integration tests will be used to test the functionality of the backend when requests are made to create new labs and start up VM(s).

Deliverables

A hosted web application is available for users to access and create accounts. The application is currently able to deliver most functionality and the finished deliverable willbe able to complete all operations required by both student users and lecturer users.

11 Security

Constraints and Resolutions

Parameter	**Customer Constraints**	**Infosys Limitations**
Authentication Credentials	The user enters an email address and password and presses login	The system must be able issue the user a JWT tokenupon successful authorization or show an error message on failure.

VM ID	The user presses on a VM to accessthe VNC view of a VM	The system must determine if the user has access to the requested VM and issue a ticket thatcan be used to authenticate with the VM
User Role and User Authentication Token	The user will only be able to have access to their own entities	The system must prevent entities that are not owned by the user from being modified

Risk Identified & Mitigation Planned

In order to authenticate the VNC connection to the VM, Proxmox credentials must be stored on the web server directly for proper proxying. If the web server is compromised, then the credentials are compromised. Mitigation involves using an account that contains only the VNC permission and swapping out the credentials when a compromise is found.

Test Strategy

Integration testing will be performed to ensure only entities the user has access to can be accessed.

Automation Plans

Integration tests will be performed on the backend with test credentials to ensure a JWT token is issued on success and a message on failure.

Deliverables

The deliverables will be a list of the results of the tests presented prior.

12 Performance

Constraints and Resolutions

Parameter	Customer Constraints	Infosys Limitations
Constraint 1	Teachers and students (alumni) will interact and create labsfor different learning purposes.	The application should respond in a timely manner. Reduce response time in milliseconds.
Constraint 2	Users should be able to manage multiple labs ina single session.	Authenticated users can open multiple labs and get them finished according to lab timing or the lab deconstructs when the timeis up.
Constraint 3	Users should be able to collaborate with others on their module labs.	Good relations are made to help users to request a collaboration or invite other users to their modules.
Constraint 4	The browser should be able to store user activity and data locally on every database request.	Cookies and sessions would be used to avoid expensive database requests.

Risk Identified & Mitigation Planned

When many users are connected at the same time as every lab uses a VM, the application response time can increase due to server limitations. So, adding more server capacities can alleviate the situation. Also, disconnecting inactive users can also free up more space for active users.

Test Strategy

Progressive unit and integration tests have been conducted that addresses any issue that can impact the performance of the application.

Automation Plans

Meta recommends using Jest for testing React applications. But many more test automations tools are available such as Mocha, Protractor. The project utilized Jest as the testing framework. Jest is based on Jasmine that adds few helpful features for testing React applications such as a fake DOM and auto-mocking of modules.

Deliverables

Deliver to the end users an application that responds in a friendly timely manner,
meaning, the load time should be in milliseconds.

13 Usability

Constraints and Resolutions

Parameter	Customer Constraints	Infosys Limitations
Constraint 1	The user should be ableto easily navigate the application	The application follows common website layouts to maintain consistency

Constraint 2	The user should be ableto perform the tasks required by the lab	The application allows the user to access modules and labs to use the VM(s) without issues
Constraint 3	The appearance of the site does not impede the user	The application is designed in away to ensure the user is not distracted from the primary feature

Risk Identified & Mitigation Planned

There could be errors with the virtual machine's submitted being able to complete the correlating tasks, this will be monitored by allowing users to submit screenshots with a message on the contact form.

Test Strategy

Tests have and will be completed per new feature. Any bug that appears to the usersafter can be submitted through the contact form.

Automation Plans

Tests using Jest or Jasmine will ensure that the frontend is functioning correctly which will ensure that the users are able to participate in labs without being faced by navigation bugs.

Deliverables

The application is available for use on the web after a user registers.

Compatibility Constraints and Resolutions

Parameter	**Customer Constraints**	**Infosys Limitations**
Constraint 1	User wishes to log in from any OS	The application is run on the web and allows any user to complete labs

Constraint 2	Users attempt to work together on a lab	The application allows for multi-user VM(s) if the lab requires collaboration
Constraint 3	User needs to access multiple VM(s) for the lab	The application allows for multi-VM sessions

Risk Identified & Mitigation Planned

Allowing for students to work together on lab VM(s) could allow for students to harm other students' progress. This has been handled by allowing users only to join in multi-user sessions if the lab requires multiple users.

Test Strategy

These features are tested as they are added. Multiple checks will also be run on different multi-user VM(s) to ensure that they function correctly together.

Automation Plans

Integration tests will be used to test the functionality of the multi-VM and multi-user VM(s) to ensure the server only links multiple users to VM(s) that specify multiple users.

Deliverables

The deliverable will be labs that allow multiple users to access VM(s) across multiple operating systems with the allowance of multi-user labs and multi-VM labs.

14 Test Team Organization

Before submitting a feature's pull request the author of the feature is required to run tests and ensure it functions in the expected way. The team lead and project manager reviews the code before the pull request is accepted. If a feature does not pass the automated build tests when the pull request is submitted the request is rejected and the feature must be reworked.

15 Schedule

Tests will be added over the next coming months to add regression testing to the current features. When new features are added, tests will be added with the features to ensure the new features stay working. The project is scheduled out in sprints and the tests will be created in the sprint that the feature is made or the sprint after.

16 Defects Classification Mechanism

Type of Defects	Functionality	Performance	Security	Usability	Compatibility
Critical	Cross-site request forgery (also known as XSRF or CSRF)	Every form which accepts input from the user will include cross-site request forgery to prevent any attack from malicious person from the front-end.	These security measureswill also prevent SQL Injection and any kind of HTML attack.	Users will not be allowed toenter html or CSS content through the form. This measure prevents users fromsubmitting scripts through the forms.	Any user will be allowed to perform only safe and trusted activity when interacting with forms.

Major	Cross-Site Scripting (XSS)	Any form will not allow html and this is done from the backend. This is to prevent users from running client-side script from the form input and place them inthe web page.	The development team will ensure to not include untrusted data in html input. If this is the case, html encoder can be used for more safety. Always validate user input.	The user input will always be validated before being processed.	Users will still be able to perform safe transactions with the application.
Minor	N/A	N/A	N/A	N/A	N/A
Cosmetics	N/A	N/A	N/A	N/A	N/A

Defects Logging and Status Changing Mechanism

Defects will be logged through a text file and mailgun messaging system, so we are notified at the time the user reports a major or critical issue. The Admin team will be able to see and solve each through the admin dashboard.

Turn Around Time for Defect Fixes

In future, the CS department will need to have or hire a maintenance team inside the CSG group to address any issue in a timely manner.

17 Configuration Management

Configuration on the frontend is managed through a .env file. The only configuration itcurrently needs is just the URL of the backend. If the backend needed to be swappedwith a different URL, a modification of the URL in the .env and a rebuild will suffice.

The Configuration on the backend is managed through an appsettings.json file that contains the database connection string, logging settings, debug settings, Proxmox connection settings, email credentials, CORS (Cross Origin Request Security) settings, the frontend url, and some encryption keys. A backup of the configuration for productionis kept and a standard development copy with mocked values is provided to developers. We use mailtrap to send emails in a fake mailbox to prevent sending email to real customers in dev. We also have a test Proxmox server configured with the dev settings so developers can test their features on a real Proxmox server.

18 Release Criteria

The features that must be operational in the system are the following:

- Essential features for admins. An admin must be able to see application statistics, manage clusters, and manage users, and schedule downtimes
- Users must be able to search and filter through modules
- Modules can hold tags that represent related topics for the lab or module
- A creator must be able to edit modules and a topology can be added through GUI
- The ability for a user to start a lab and interact with a VM
- The ability for the VM to connect to the internet so assignments can be submitted through GitHub
- Multiple users can access the same lab for either resource constraints or teamwork based environments
- Multiple VMs can be used in a lab and communicate to each other
- Dr. Doyle and Prof. Sexton can use these features for their future courses
- User must be able to register and login to the system to access labs they have access to
- A user must be able to start a lab using a secret code given by the professor