**Cloud Lab**

**Non-Functional Requirements Specification**

# **Introduction**

This document will define the Functional Requirements, Non-Functional Requirements, System Interfaces, and Business Rules of the Cloud Lab System.

# **System-Wide Functional Requirements**

The Cloud Lab System will provide a drag-and-drop interface where you create and configure a lab, with the ability to make multiple copies of the lab to accommodate the number of students who will be using it. Using AWS cloud services and web access via a desktop or laptop.

# **System Qualities**

## **Usability**

Usability is one of the highest priority system qualities because simplicity and ease of use is one of the things that the commissioner is trying to achieve by developing this system, making it easier for teachers to set up a computer lab for students without having to have advanced technical skills. To achieve this we will implement look and feel standards

\*\*\*

## **Reliability**

The time between teachers and students is very valuable and extremely limited, any delays caused by unreliable software can have a huge impact on education. It is necessary that classrooms find this tool completely reliable otherwise it will be dropped in favor of the slower, more expensive, traditional methods of setting up computer labs, and valuable time will be wasted.

If software failure on our end causes a lesson to halt, the system will not meet its reliability criteria

## **Performance**

Significant time spent waiting for classes to start is unacceptable, if significant load times are unavoidable, there should be options in the software to enable scheduled preloading so that significant processing / setting up of lags can be done before the lesson starts.

The software need to at least be able to provide students with there cloud labs within a few minutes of a class starting and preferably a few seconds.

## **Supportability**

The system simply needs to be capable of supporting and scaling to the users of a single school, every school can be expected to host their own instance of the software service. There needs to be a quick and efficient way to report problems with the software, as well as dashboards showing overall server usage so that system administrators are able to judge whether the server hardware needs an upgrade.

# **System Interfaces**

*[Interface Requirements are part of the + in the FURPS+ classification of supporting requirements. Define the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, and so forth, so that the software can be developed and verified against the interface requirements.]*

Both the server and client side of the application will communicate through a REST api, over the HTTPS protocol. Users will receive the applications client through standard web protocols with a default port of 80. System administrators can either let users connect to the web interface via a local IP address…… or define a local domain like “labserver” in the routers dns table. The effect would be that whenever a user on the schools network types in exactly “labserver” into a web browser url, they will be directed to the web applications server that will provide them with the client.

## **User Interfaces**

*[Describe the user interfaces that are to be implemented by the software. The intention of this section is to state requirements relating to the interface. Interface design may overlap the requirements gathering process.]*

The User interface will have two main sections, one for the students to clone templates of cloud labs, and one for the teacher to create templates.

The teachers template creation tool will have drag and drop elements and a details panel on the right.

The students template clone tool will have a list of the templates they can clone

### *Look & Feel*

The look and feel of the application will be slim and cutdown, we don’t want to overload the users with excess information, the whole purpose of this project is to make a simple version of a more complex tool that does the same job.

### *Layout and Navigation Requirements*

*[Capture requirements on major screen areas and how they should be grouped together.]*

The first screen users will see is a login page, which will either lead to a student or lecturer dashboard, from the student dashboard they should be able to preview and then clone a template with a list of templates and buttons. Once done a list of the remote access details should appear in front of the student next to images displaying various lab components cloned.

The teachers dashboard will have previously used templates in a list, buttons to ship a template to the students, and buttons to create a new template

The new template page will be drag and drop.

All pages must have a link back the the main dashboard.

### *Consistency*

*[Consistency in the user interface enables users to predict what will happen. This section states requirements on the use of mechanisms to be employed in the user interface. This applies both within the system and with other systems and can be applied at different levels: navigation controls, screen areas sizes and shapes, placements for entering / presenting data, terminology.]*

We will use rounded buttons and flat screen design throughout, The navigation to dashboard buttons will always be in the same place. The list view for teachers and users should look as alike as possible.

The screen size will be designed for a standard 4:3 display.

The colour scheme once decided will be consistent on every panel.

### *User Personalization & Customization Requirements*

*[Requirements on content that should automatically displayed to users or available based on user attributes. Sometimes users allowed to customize the content displayed or to personalize displayed content.]*

A customisable Templates lists/categories for teachers and students will be essential to organising the volume of network templates that will be created.

## **Interfaces to External Systems or Devices**

*[Are there any external systems with which this system must interface? Are there any constraints on the nature of the interface between this system and any external system, such as the format of data passed between these systems, and any particular protocol used? Consider both provided and required interfaces.]*

### *Software Interfaces*

The software interfaces directly with the clients browser, it depends on a browser capable of parsing HTML, CSS and Javascript. The server side will interface with systemd for runtime management.

### *Hardware Interfaces*

The client side will depend on the users desktop, accessing its cpu, ram and, GPU.

### *Communications Interfaces*

This application will heavily interface with amazon web services…. The largest constraint is the dependence on the AWS API, if the api changes, the software needs an update in order to continue functioning, or if Amazon shutdown the API then we can no longer depend on AWS…. One last scenario is if amazon goes bankrupt or shut down as a company / the aws division…. This software will need to be re written for a different cloud platform. In short the software is heavily tied in with the AWS external service.

The data format for amazon's web api is standard HTTP REST POST/GET