**Cloud Lab**

**Architecture Notebook**

# **Purpose**

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

# **Architectural goals and philosophy**

* Simplicity, its replacing a more complicated system that’s difficult for educators to use
* Speed, its being used in a classroom where lost time means lost education
* Robust, The software won’t be actively developer after deployment, so it needs to be solid
* Lightweight, Given the software is self hosted it needs to handle a whole school on a single server

# **Assumptions and dependencies**

* Amazon web services won’t change or disable their API
* Amazon web services won’t go out of business
* Web browsers remain compatible with current HTML and Javascript standards
* The Gnu ecosystem powering the server still allows the server to run as intended
* The schools connection to the outside internet is working
* Each student and teacher have access to a computer with a web browser

# **Architecturally significant requirements**

* A server to run the schools lab management software
* Each student and teacher have a computer with a web browser
* Constant internet access
* A local area network with all involved teachers, students and main server on the same vlan

# **Decisions, constraints, and justifications**

* Use of Django because ⅔ team members have only ever programmed in python
* Use of Bootstrap because we have a team member with bootstrap experience
* Use of a client server model because its convenient given we are already dealing with web requests

# **Architectural Mechanisms**

## **Api requests / template deployment**

These will be sent from client to server, and from server to AWS.

## **Client access**

A main locally hosted server will provide the client through serving users with a web page

# **Key abstractions**

* Server
* Client
* API

# **Layers or architectural framework**

[Describe the architectural pattern that you will use or how the architecture will be consistent and uniform. This could be a simple reference to an existing or well-known architectural pattern, such as the Layer framework, a reference to a high-level model of the framework, or a description of how the major system components should be put together.]

We will be using one set of rules to design the server that both acts as a middle man between the client and the api, it is also in charge of serving the client to the users

# **Architectural views**

## **Views that will be used**

* **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
* **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.