***DaBL queuing & Scheduling system (QSS – 5000X)***

**CSC650 - System Design Document**

Graduate Student Group

11 December 2019

American University

**Table of Contents**

[**Introduction**](#_heading=h.30j0zll)

[**Purpose**](#_heading=h.3dy6vkm)

[**System Overview**](#_heading=h.3dy6vkm)

[**Design Constraints**](#_heading=h.2hio093)

[**Roles and Responsibilities**](#_heading=h.wnyagw)

[**System Architecture**](#_heading=h.1vsw3ci)

[**Database Design**](#_heading=h.ihv636)

[**Software Detailed Design**](#_heading=h.4fsjm0b)

**LICENSE AND** [**System Security**](#_heading=h.3ep43zb)

# 

# 

# Introduction

This document has been created to outline the proposed system design for the Design and Build Lab (DaBL) queuing and scheduling system. By designing, testing, and deploying this system, students are able to register for workshops and activities based on the specialized devices and tools they have been certified on, while concurrently able to schedule appointments to independently use these tools. This system is intended to replace the current walk-in system of *first-come-first-serve* in order to implement order and predictable to both DaBL staff and students. A secondary system has been created to support data entry into DaBL user databases that will accordingly support the online registration system. This document and the technical specifications listed herein comply with all American University (AU) Information Technology (IT) policies.

# 

# Purpose

The purpose of this document is to provide a description on how DaBL queuing and scheduling system is designed and constructed. This document is created to ensure that DaBL design meets the requirements specified by the customer, and also provides a description of the system architecture, software and hardware, database design, and security.

# 

# System Overview

# 

The system designed is a scheduling application that demonstrates the ability for an authorized user to reserve a time on the DaBL website. This system is web-based and supports the current DaBL site with additions to the front end application. This *proof-of-concept* application (referred to hereafter simply as *application*) achieved the general concept of allowing users to reserve a specific tool at a specific date and time based on user certifications.

Although we assess that future requirements will require updates to the database and remote cloud services, the team has provided a workable foundation for implementing the application in future conditions. Furthermore, the design has established the backend system of query logic for interaction regarding student requirements and reservations, providing the foundation for future interactive design to develop upon. Supplementary to this logic is the proposed database design which enforces AU IT user information compliance practices. This is paramount to fulfilling the standard backend requirement as these key database and scripts can be easily integrated into the *Wordpress* environment.

Notably, application development revealed more understanding of how DaBL assets and components interact. This revealed critical systems and networking knowledge during the course of this exercise that will influence future design efforts.

# Design constraints

The team identified several constraints which impact and limit the design of the system in its current state. These constraints are beyond the scope of the team, but must be carefully factored into the system design. To date, the following constraints have been identified:

* **New technologies**: It is noteworthy that the team were unfamiliar with many programming languages and infrastructure affiliated with the DaBL system prior to the development of this system. Consequently, system design may not include more elegant methodologies employed by regular developers or users of these tools and languages.
* **Access & Security**: Certain limitations from AU IT policies require additional components to the system where *confidentiality* may become future problem sets. At this time, the user database proposed is compliant with AU IT policies, however this will naturally have to be monitored over time. Additional considerations include cloud services security and difficulties associated with implementing within this architecture due to security constraints. More specifically, MySQL database access issues has, at the time of this document, disallowed standard port 3306 opening.
* **Storage & Infrastructure**: Current database content will be required to be sanitized and migrated into the cloud instance database in order to deploy the web application and administrative application effectively.

# Roles and Responsibilities

The following table defines the system design roles and responsibilities. This matrix also serves as the list of points of contact for issues and concerns relating to the System Design.

|  |  |  |
| --- | --- | --- |
| **NAME** | **ROLE** | **EMAIL** |
| Myles Russell | Team member | mr7085a@student.american.edu |
| Abdul Tarawally | Team member | at4667a@student.american.edu |
| Isabella Sims | Team member | is3923a@student.american.edu |
| Jiachen Yao | Team member | jy3695a@student.american.edu |
| Li Jie Su | Team member | ls8613@student.american.edu |

# 

# 

# system architecture

**Hardware**

The design is based on existing hardware architecture already deployed in DABL. Because this supports several software applications, it is more worth to note for context the DaBL hardware components that the software supports scheduling for:

|  |  |
| --- | --- |
| **Series 1 Pro** | Printer |
| **3D Printer** | Printer |
| **Jaguar VLX** | Cutter/Plotter |
| **Bantam Tools** | Printer |
| **PLS475** | Laser Cutter |

**Software**

The design is based on the individual design of various components in which users will enter and query data. The software architecture is designed to incorporate all data entry, input, and modifications into integrated databases which support user registration, as well as tracks available tools for scheduling. The components which comprise the software architecture include:

|  |  |  |
| --- | --- | --- |
| **DaBL Event Calendar** | php, html | AWS EC2 |
| **Administrative Application** | Electron framework, node.js, html | DaBL Administrative Workstation |
| **User Database** | MySQL | AWS EC2, DaBL Administrative Workstation |

# System Architecture Diagram

# 

The system diagram above highlights two perspectives: the user and the developer. Per the visualization, the system shows two databases, one locally placed at DaBL and one on the AWSinstance. Users schedule a reservation through the front end website, which is linked to the user’s student account. When physically present at DaBL, they use their AU Identification cards (AUID) to authenticate themselves to the lab and conduct tasks on certified devices. Developers access the front end through the *Wordpress* development interface in order to perform development tasks. Additionally, they can interact with the operating system hosting the *Wordpress* site via access into the AWS instance.

# Database Design

Our reservation system database is a MySQL database hosted on AWS server. The database stores user validation information for query by the DaBL web application. The database contains one table called *user*. The *user* table contains columns of the following columns:

|  |
| --- |
| **email address** |
| **waiver** |
| **Series 1 Pro** |
| **3D Printer** |
| **Jaguar VLX** |
| **Bantam Tools** |
| **PLS475** |

Based on AU’s policy, *email address* is the only information that is allowed to be stored outside of AU networks. Therefore, the *email address* serves as the unique id for each student using the system. Students will be asked to sign a DaBL waiver which educates students on how to protect themselves when using machines in DaBL. Both the *email address* and the *waiver* are used to validate users when they want to reserve a machine from the web application.

Other columns of the table contain information of the student accesses to each machine in DaBL. Before allowing students schedule a certain machine, DaBL requires students to attend a training session for the desired machine in certify students on the device. A user’s certification is stored in the database, allowing for the web application to check if the student request for specific machines are valid prior to scheduling.

# Software Design

***PHP***

***Interface with MySQL database***

The team imported and applied the MySQL library to interact with the MySQL database. The team built several functions in the *IDatabase* class to support interaction with the database. *Php* functions perform checks against the AWS database returning a boolean value to accept or deny the user from reserving equipment in DaBL.

***Event reservation and conflict resolution***

The team utilized the *Google Calendar API* to allow users to reserve events on their AU-associated Google Calendar. Furthermore, the *php* functions implemented also deals with event conflict if the user’s reservation has issues with scheduled events. If there is a conflict, it will return a *false* value to the main process.

***Main process***

The main process integrates the database interface and event reservation. Firstly, obtains string data passed by the *WordPress* frontend and formalizes the data. Next, it gets evaluated to check if the user is in the database, as well as if all reservation requirements have completed. If all requirements have been completed, it reserves the requested equipment informs the user the reservation has successfully processed. If it is rejected, it will prompt the user to complete all requirements before performing the reservation.

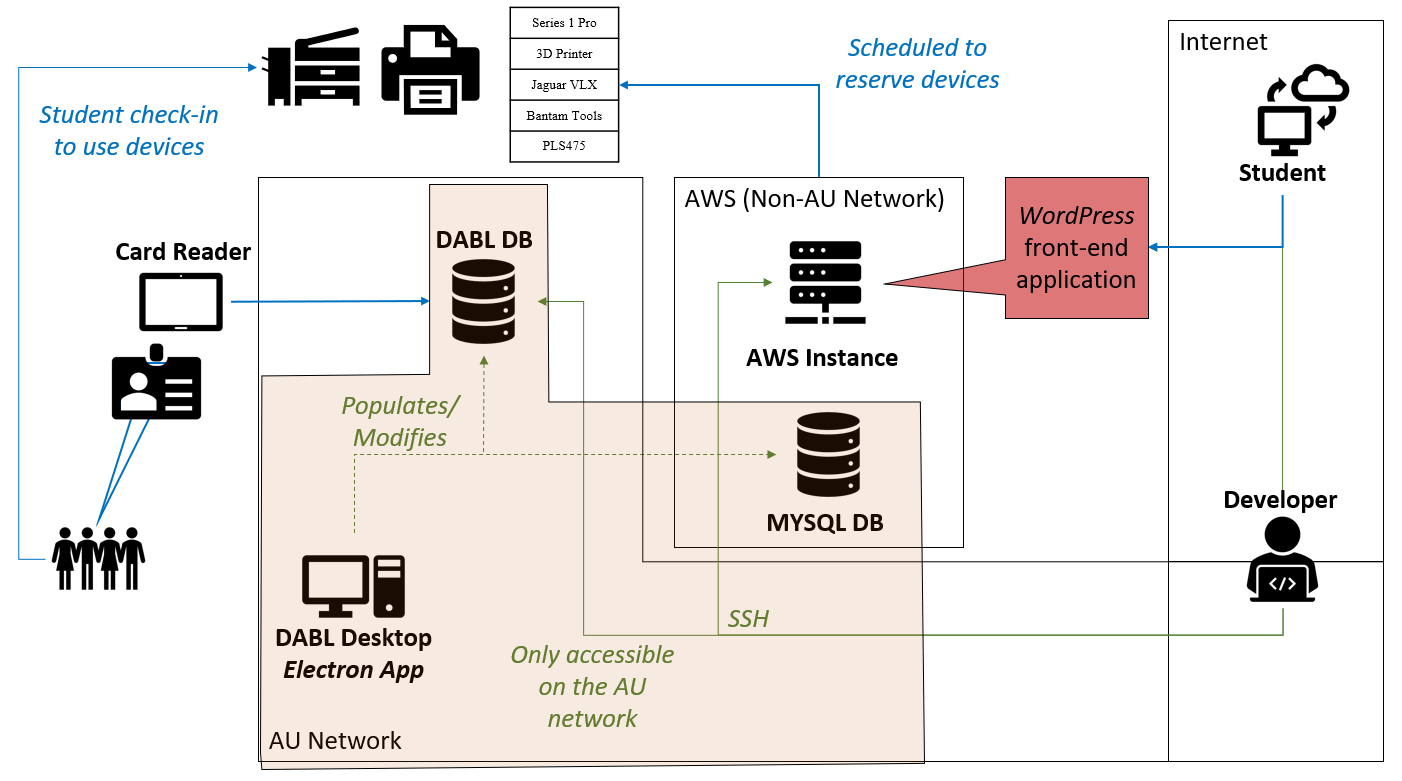
***Administrative Application***

The DaBL Administrative Application was built to abstract the concept of dual input and interaction for the user when dealing with two separate databases. It is intended for easily and quickly populating the DaBL database. Moreover, it is intended to populate and modify the local DaBL database as well as the sanitized AWS DaBL database which supports the DaBL Event Calendar on the DaBL website. The intended user behind this system are DaBL administrators that wish to enroll students into the lab.

The application is built using the *Electron* framework is run as an executable from the DaBL Administrative Workstation. *Electron* is based on web-based technologies, so it is a framework geared towards allowing web developers apply their skillset to create desktop applications. The primary programming languages employed are *HTML* and *javascript* for the front end, and *node.js* for the backend. There are three areas of the application:

* *Front-end*: User interaction to input information. This feature passes requirements to the *renderer process*.
* *Renderer Process*: Handles the requirements requested between the front-end to the back-end. Communicates the request to the *main process* where system level tasks can be performed.
* *Main Process*: Instantiates application and facilitates system level tasks to and from the front end through the *renderer process*.

The diagram highlights the portion where the the application performs with respect to the wider system.

Please see the DaBL Administrative Application Developer Guide for more information.

# System Security

The design incorporates several security and integrity controls to ensure that the system and data are continually protected. This is done to ensure data integrity is achieved through only authorized user functions and tasks. The first design consideration is user authentication and authorization. The system authorizes the user based on two factors. Firstly, users that have registered with DaBL have been placed into the DaBL user database (both locally and in a sanitized manner compliant with AU policy on AWS). Secondly, users have an AU gmail account that allows for additional protections. The AWS user database provides the Event Calendar application the ability to check what certifications the user has in order to authorize them to schedule certain devices. The additional benefit refers to Google gmail authentication that affiliates the user with AU and automates the linking of the calendar to their gmail. All users will be allowed to register based on their certification level.

From an external network perspective, the application resides in a *Wordpress* application, and therefore we are relying on *Wordpress* application security. Moreover, the website does not store sensitive information since the user database is a compliant, sanitized database. Ultimately, this means that there are no passwords or personally identifying information in the database.

The administrative application interacts with the DaBL user database which contains sensitive information. This database is stored locally on one computer and requires physical access to a password-protected computer in order to use. Despite being secure to physical access, the software contains possible SQL injection vulnerabilities since it does perform input checks prior to querying the DaBL user database. Therefore, the application is relying on physical security and remaining local and non-connected as its primary security posture.

Though the source code of the webpage (index.html) can be inspected and viewed like any other webpage, we prepared for this and were certain not to include any sensitive or personal information could be acquired upon inspection. We did not embed our PHP code into the HTML document in order to protect user information and the PHP code (which interacts with the database and sensitive information) is not viewable upon page inspection.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# DaBL Application License

The project is licensed under the *MIT License*. The team has selected this license becauese it is permissive and allows for unlimited use by DaBL. This encourages the use of the provided code, modification, publication with few restrictions.

**The license is as follows:**

Copyright 2019 American University Department of Computer Science

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS"WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.