University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2021

Fay Jones Lab Request System

Aaron Faubion, Elliot Mason, Matthew Brooke, Tanner Paschal, and Zachary Thiher

Abstract

The Fay Jones School of Architecture has a fully-fledged build lab that includes a woodshop, laser routers, CNC routers, and 3D printers that architecture students can use in the basement of the Fay Jones building. During the COVID-19 pandemic the lab adapted, allowing students to submit job requests over email. This has led to problems like delays in job returns or miscommunications between shifts about what is required resulting in materials being wasted. Currently the lab utilizes R&R booking software where students can just sign up for time slots for the various machines in the lab. There are other functions, but since the software was initially intended to be used for library resource scheduling, many of the paid functions that are provided in the software are not being fully utilized.

This project aims to provide a solution that operates as a ticketing system, allowing students to register and submit job files to act as a replacement to the current emailing workflow. We utilized the MEAN (MongoDB, Express.js, Angular, Node.js) stack to develop a Single Page Application (SPA), pack it into a Docker image, and then deploy that image to Google Cloud Platform (GCP).

1.0 Problem

Currently, the build lab space has adapted due to COVID-19, not allowing students into the lab spaces to do laser cutting or 3D printing. The laser cutters allow for students to load Adobe Illustrator files and then cut, score, or etch their designs onto a variety of materials to get precision cuts for when they are building their models. Because of the changes, lab technicians now accept the illustrator files over email, along with what type of material they need the design cut on.

Normally the students would come in for a consultation, and the tech would identify any problems within the file and recommend changes before slicing and printing. The current queue system for 3D print jobs includes a Google forms page that propagates an excel sheet with the job name and basic details, where the student tech is responsible for ensuring that the correct information is recorded. This method fails to accommodate multiple files per job, so the

workflow ends up requiring the student tech to re-enter the same information on multiple forms for each individual file.

Often there are upwards of 10 files that need to be individually cut with the laser cutters, resulting in lots of front-end work just to get the records on the excel sheet. Then, even more work is needed to make sure that the files are saved in correct locations on Box (file storage) so the next shift workers can find the files. File storage is another potential problem because, if the previous technicians do not put all the files exactly where they need to be, there can be problems in the next shift trying to locate the files. This could lead to delays in the jobs being processed.

Using email as the primary file transfer medium also comes with its own problems. In large jobs with many files, multiple emails might be needed to transfer the entirety of the student's job requests. Another drawback of using email for file transmission is that only the initial emails have the job files. If a student tech needs to download files, they must search for the first email that contained them. The lab utilizes Box as backend storage for project files and requires manually creating folders and enforcing an organizational structure.

In the past, there have been surges with the demand for 3D printing. With the current email-based system, there are lots of mix-ups and other problems, such as things not getting printed on time. *Without* our straight-forward ticket system, jobs are lost in inboxes, and without a dedicated place for 3D jobs, everything is confusing – both for the student and the student technicians.

2.0 Objective

The main objective was to create a ticketing system that will aid in organizing the job submissions. This application contains a user management system and job tracking. This ticket tracking system allows students and lab technicians to communicate changes and share project files. The application is a seamless SPA in a web browser. In addition to a just a web client, the app is responsive and renders correctly on various mobile devices.

3.0 Background

3.1 Key Concepts

The actual SPA web interface is built with Angular. Angular describes itself as an "application design framework and development platform for creating efficient and sophisticated single-page apps" [1]. Angular is developed by Google, and there are a variety of themes that are available to enforce a unified user interface, reminiscent of stylings that Google uses on their own web applications. Using Angular has bolstered our ability to create the front-end web page and has simplified the development process. The backend hosting of our application uses Node.js as the main server process. In conjunction with Node.js, Express.js is a library that is used to handle all the incoming requests, either routing client requests to API endpoints or serving the SPA files. When the Node.js server starts, a Mongoose library is used to establish a database connection to a MongoDB instance hosted on Atlas. MongoDB describes itself as "a general purpose, document-based, distributed database" [2]. We used MongoDB for storing persistent data for the tickets and initially for the user database.

Before beginning development on this project, we first conducted market research for other similar projects that were already developed and at market. The first product that we

discovered in our research was MakerOS, which was designed as a commercial platform targeted at manufacturing shops [4]. This has some of the functionality that we are aiming to implement, however pricing starts at $77/month for 2 employee seats and $191/month for 5 employee seats. With an average staff of 15-20 each, this solution would greatly increase in price at an enterprise level license. The other product that we found was Fabman [5], which was a similar platform but included optional kiosks that would lock out different machines within the build lab until the user unlocked them with a Near Field Communication (NFC) RFID card. Both platforms have implemented sections of what we are planning to do, but much like the current R&R solution, many of the features that are being marketed would not be fully utilized by the build lab space, and the monthly expenses are too substantial to justify their use.

4.0 Design

4.1 Requirements and Design Goals

First, we implemented a user account system. For this application, there are three user roles: Student, Staff, and Admin. Students have the role of Student, student lab technicians have Staff roles, and the full-time staff at the lab have the Admin role. For the user management system, users can create accounts and reset their passwords without staff intervention if a password is forgotten and/or lost. The different user roles within the application limit what users can do. Staff functions like closing job tickets, commenting, and updating invoices will be restricted to the Staff and Admin roles. Student user roles allow for job submission and commenting and closing tickets that that user owns.

Within the application there is a ticket module, where users can interact with the ticketing system. Users with Student roles can submit new job tickets, interact with the comments from staff, and manage the files that have been submitted on the ticket. Staff and Admin roles can add comments to the ticket, update the ticket status, and close out tickets once they have been completed. The main modules of the ticketing system are a module for the user interface that allows for job submission and a module that lists owned tickets and allows users to interact with those tickets.

4.2 Detailed Architecture

Fay Jones Lab Request System Angular App is made up of four main modules: account management, printer management, job ticket submission and management, and the Admin module. Each module is an Angular component, which allowed for easy segmentation of tasks during development.

When a user successfully authenticates to the webapp, a web token is generated that contains their UID which is used to determine which parts of the application that they have access to. For example, students only have access to the job submission and detail queue view for their own jobs, while the Admin and Staff roles have access to the above, as well as the Admin Management panel where they can modify the materials and printers that are available in

the lab. The users can request a password reset, and the system will generate a onetime use token and email it to their account for access to the password reset page.

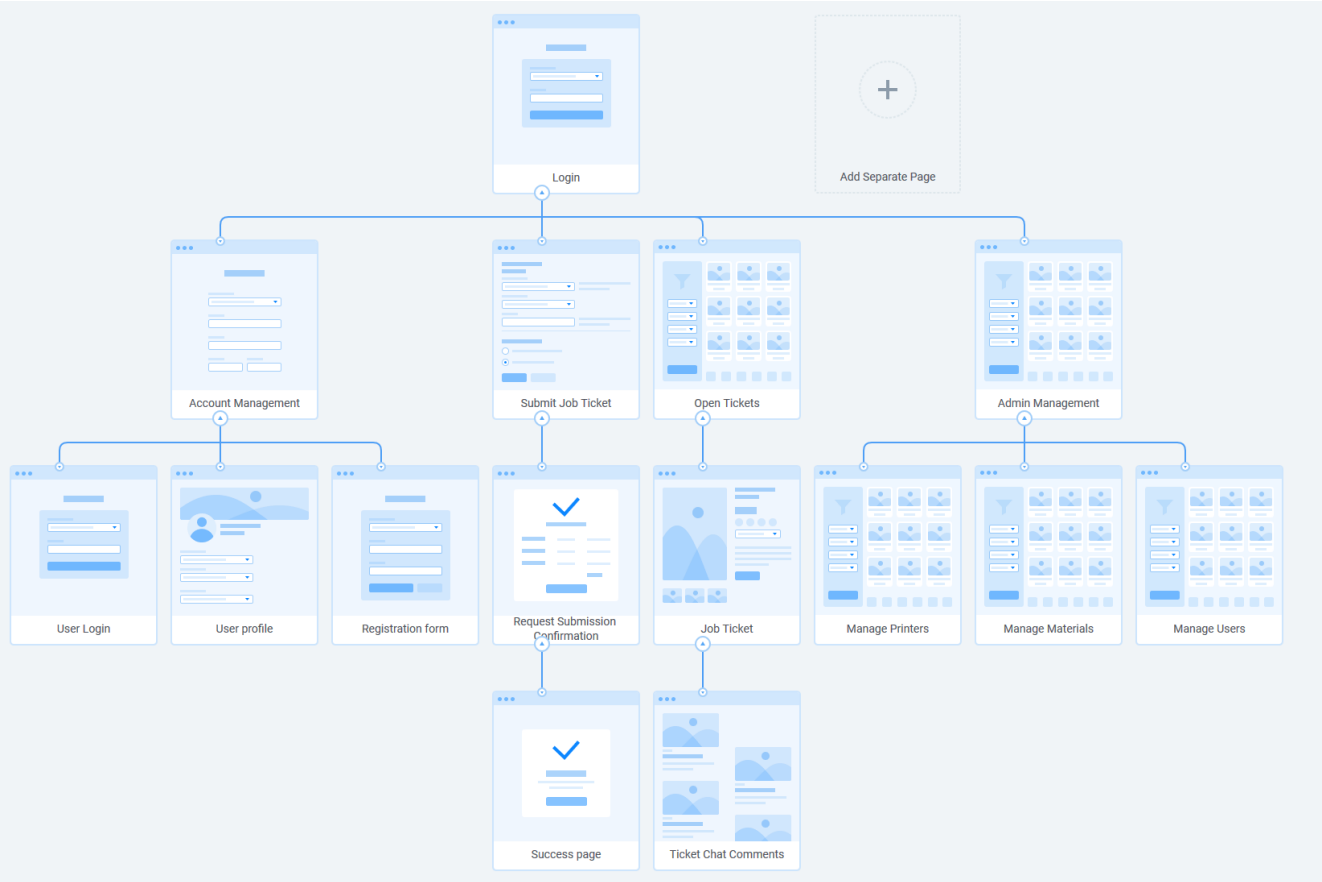


Figure 1: Application flow and module architecture

The job submission page contains three main fields, the file upload, the material selection, and a comment box for any additional information the student might want to supply the lab. Currently, the job submission workflow only supports a single STL file per request submit due to file storage limitations. Additionally, the material that the student can select for a job request is dynamic and pulls from the materials database. Once the job is submitted, it populates the print queue table that all users can access. The print queue also has a detailed view for each of the submitted jobs that can only be access by the job owner if the user has a student role. Admin and employee users can access all queue detail item views, as they need to be able to update the status on the jobs, add comments, and download the files required to start the 3D prints.

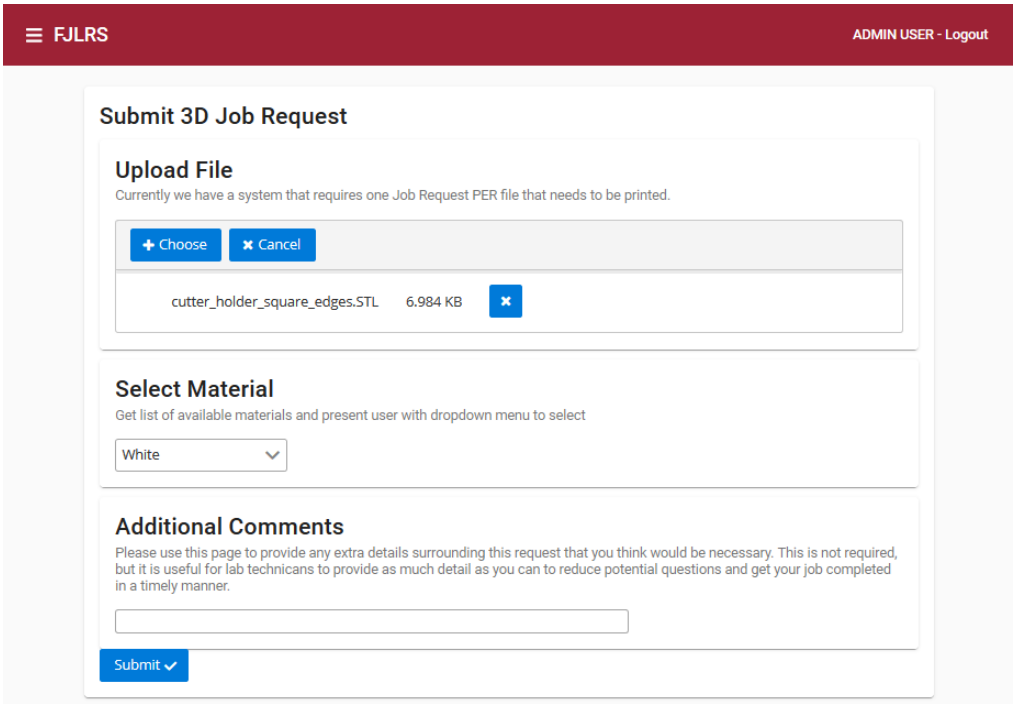


Figure 2: Job Request Submission Form

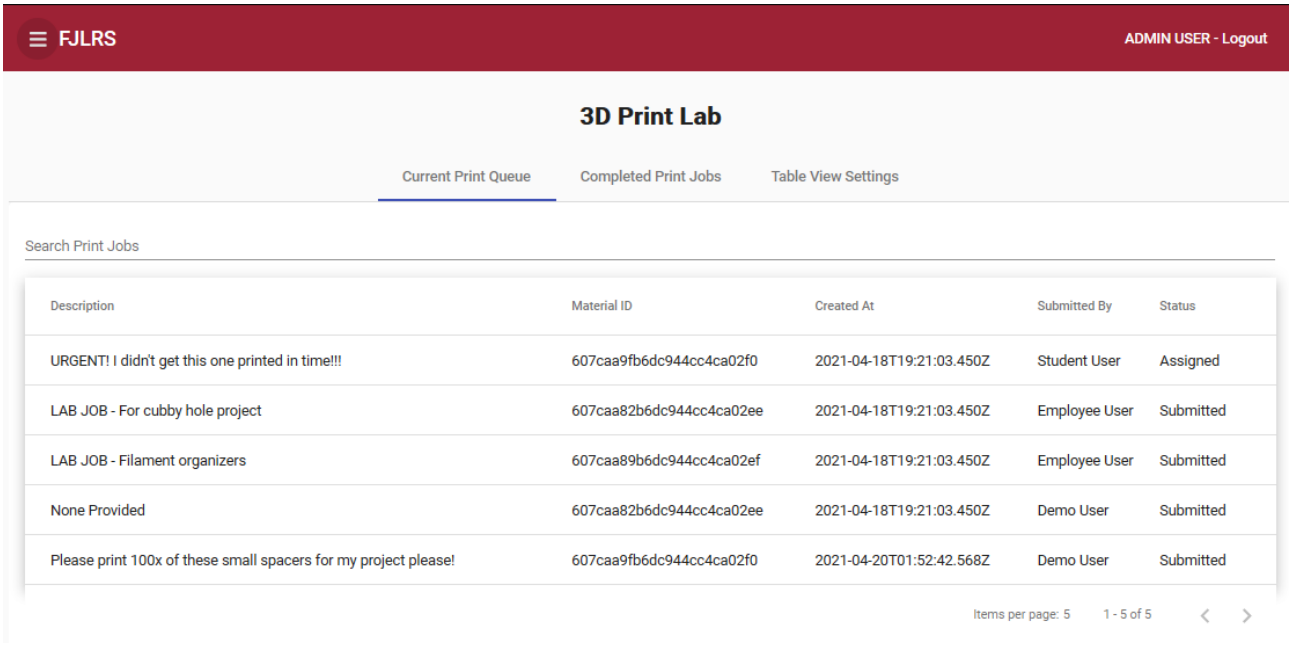


Figure 3: Print Queue Overview

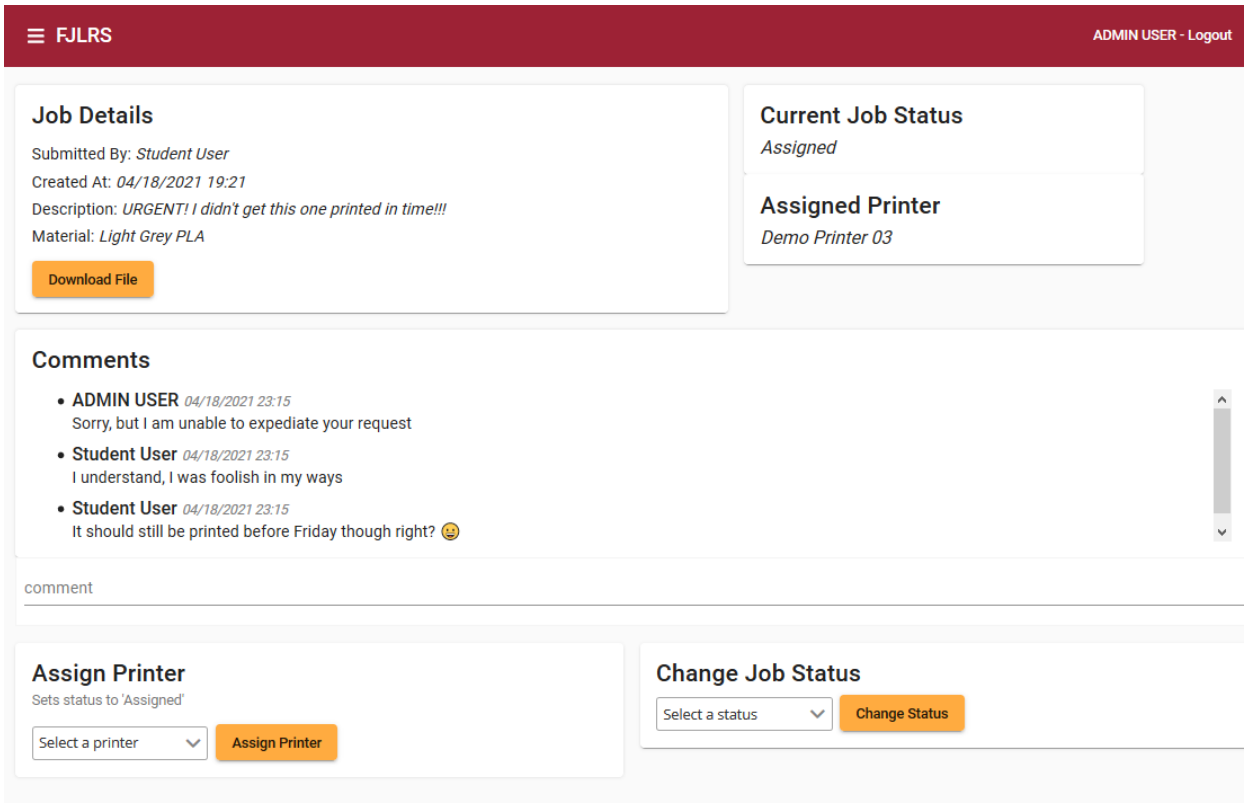


Figure 4: Detail Print Job View

The detail print job view panel is the main feature of the application. Here the staff can interact with the job, assigning it to a specific printer, update the job status, and communicate with the job owner. The 'Assign Printer' and 'Change Job Status' panels are only rendered for staff or admin users, as the students should not be able to update the job status if they have no part in actually running the jobs.

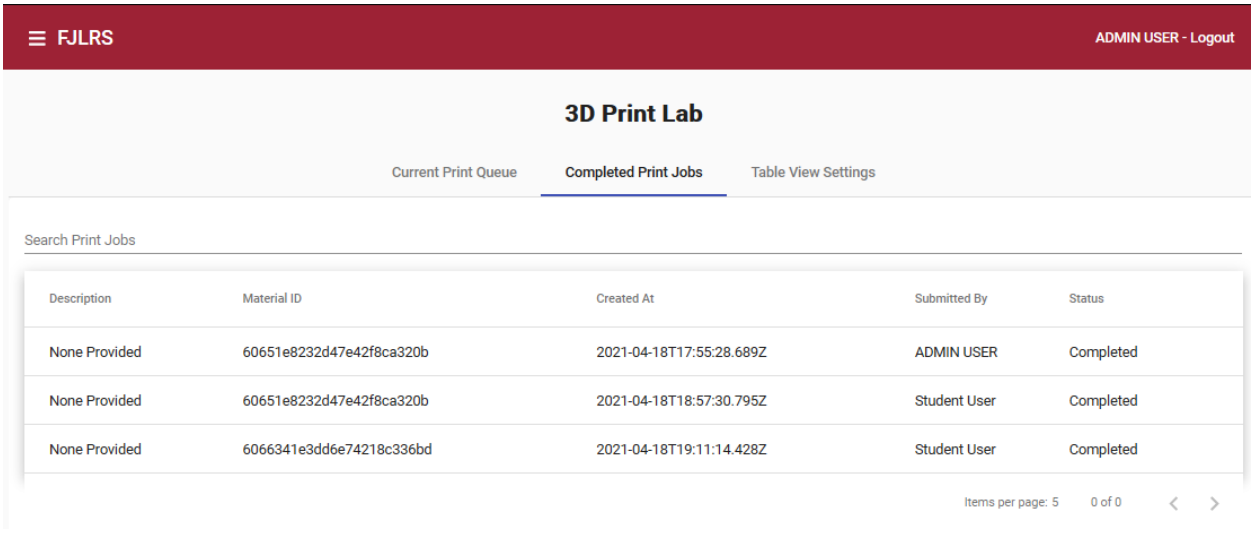


Figure 5: Completed Job View

Once the print jobs are completed and the student has picked up their printed items, the status is updated to 'Completed' and is no longer shown in the current print queue but is still

available for historical lookup via the ‘Completed Print Jobs’ table view. Additionally, if a specific user is looked up on the user management admin panel, all their queue jobs are available for lookup as well.

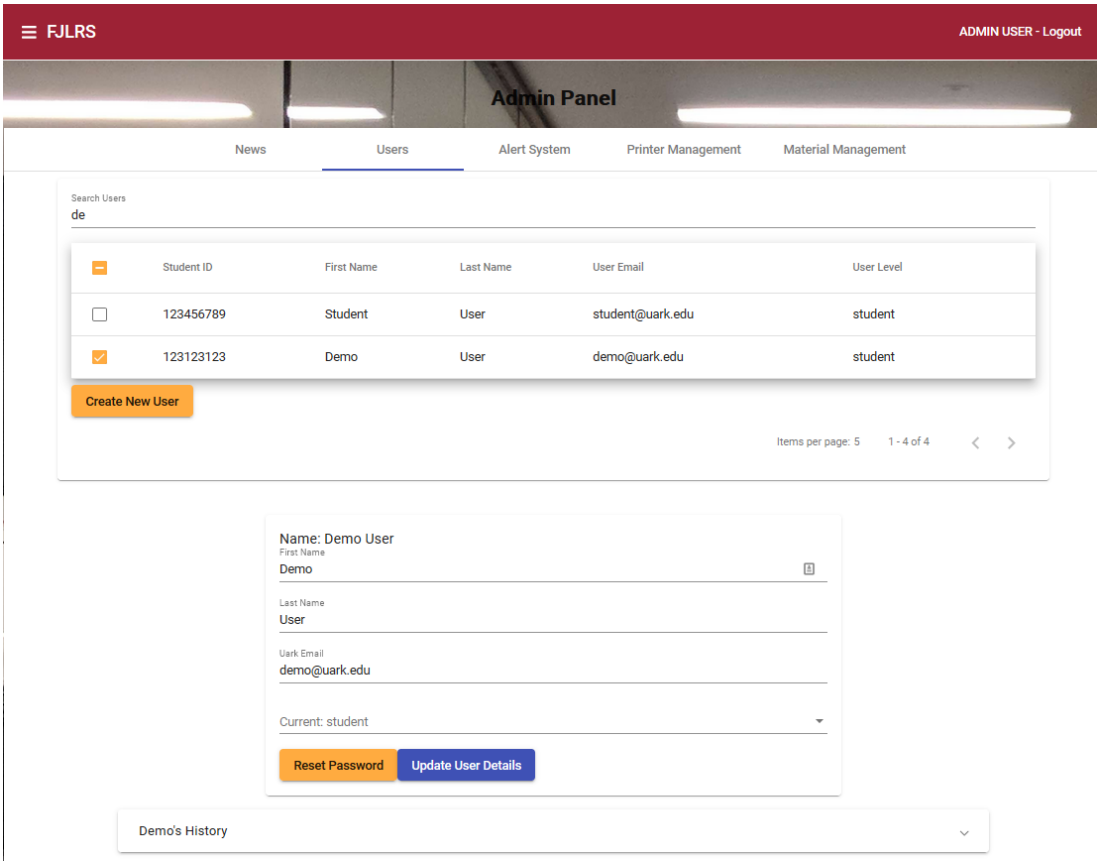


Figure 6: Admin Panel User Detail View

4.3 Risks

Risk	Risk Reduction
Hosting Solution	We packaged software in a Docker container, a package that contains all the software, libraries, and configuration files needed to run the application. This allows for the most flexibility in how and where the app can be hosted.
Internet goes out on campus	The application is hosted on GCP to allow for high availability. Printer updates are interrupted because of the lack of internet on campus, but the application will continue to work without octopi integration.
Power goes out on campus	All 3D printers and Raspberry Pi computers are connected to UPS (Uninterrupted Power Supply) that provides a grace period to pause all print jobs to allow for resuming after

	power is restored.
User account passwords	The application has capabilities to allow students to initiate a password reset workflow.
Staff not monitoring application for new jobs	The application has a list of staff members that can be alerted either via email or text message when new jobs are submitted.
File storage if application crashes	Files should be stored on an external file store, such as a NAS that the application has read and write permissions to. In the event that the application crashes, the file store should be available to the staff so that files can still be retrieved.

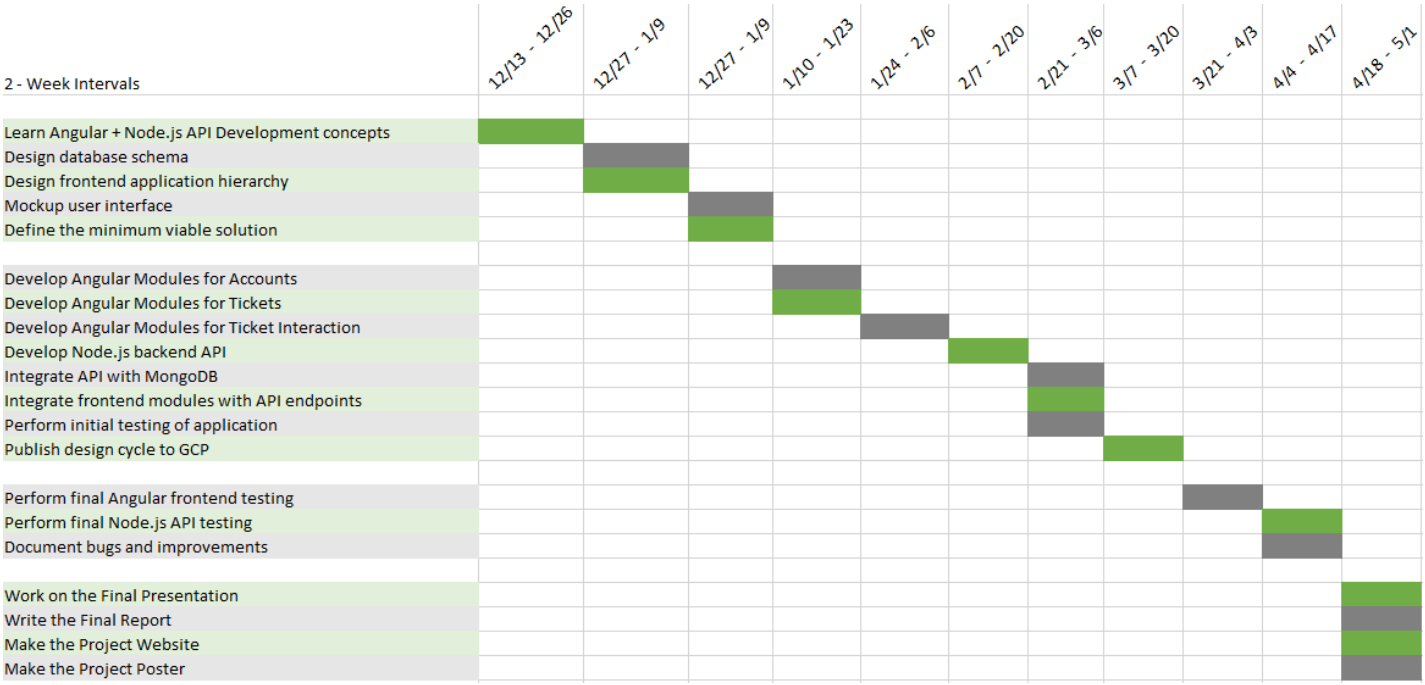
4.4 Tasks

1. Project Analysis
    1.1. Define current system functionality and where this project will make biggest impact.
    1.2. Document current workflows
    1.3. Define project goals
    1.4. Capstone I Documents
        1.4.1. Quad chart – 9/14
        1.4.2. Preliminary proposal – 11/13
        1.4.3. Final proposal presentation - 11/30
        1.4.4. Final proposal report – 12/10
2. Design
    2.1. Design database schema
    2.2. Design frontend application using Angular framework
    2.3. Mockup user interface pages
    2.4. Create design specifications for minimum viable solution application
3. Development
    3.1. Develop Angular Modules for Accounts
    3.2. Develop Angular Modules for Tickets
    3.3. Develop Angular Modules for Ticket Interaction
    3.4. Develop Node.js backend API
    3.5. Integrate API with MongoDB
    3.6. Integrate frontend modules with API endpoints
    3.7. Perform initial testing of application

3.8. Publish design cycle version to GCP

4. Testing
   4.1. Perform Angular frontend testing
   4.2. Perform Node.js API testing
   4.3. Document bugs and improvements for next development release
5. Finalizing
   5.1. Final Presentation
   5.2. Write the Final Report
   5.3. Complete the Project Website
   5.4. Create the Project Poster

4.5 Schedule



4.6 Deliverables

- README document describing how to setup development environment on the GitHub repository located at https://github.com/ejmason101/FJLRS/
- The Node.js server code: This can be run on a server and will serve the Angular frontend and interface with the MongoDB database.

- The Angular project and Final Build: We will include the Angular project so future development can be done once the final project is submitted.
- The final report that details what our project accomplished, and how it works.

5.0 Key Personnel

Aaron Faubion - Faubion is a senior Computer Science major studying at the University of Arkansas set to graduate in Spring 2021. Aaron is set to graduate with a Bachelor of Science in Computer Science and will be accepting a Software Engineer position upon graduation with IBM. Aaron has completed Programming Foundations I and II, Digital Design, Database Management Systems, Algorithms, Software Engineering, Programming Paradigms, Computer Organization, Cryptography, and various math courses and statistics courses. Faubion was responsible for working on the Node.js backend API as well as user interface page design and implementation. Faubion has built the main business website for a company, as well as attended various workshops and hackathons throughout his college career.

Elliot Mason – Mason is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Database Management Systems, Software Engineering, and Programming Paradigms. He was responsible for application hosting architecture, CI/CD pipelines, and both the API and the Angular frontend. He has gained valuable experience during his three years on the Linux Enterprise Administration team at Tyson Foods.

Matthew Brooke – Brooke is a senior Computer Science major in the Computer Science and Engineering Department at the University of Arkansas. Throughout his time at the University of Arkansas, he has completed several courses that will aid him in working on this project, including Software Engineering, Programming Paradigms, and Database Management Systems. He has gained work experience from his continued internship as a Software Engineer for the Tesseract Center for Immersive Environments and Game Design, where he has been tasked with implementing key project features, debugging core issues, working closely with design teams to implement UI features, and leading technical development on some projects. He supplied his skills for this project by working for front-end UI/UX design and assisting with back-end API development.

Tanner Paschal – Paschal is a senior Computer Science major in the CSCE department at the University of Arkansas. He has taken many courses that are relevant to this project, including Database Management Systems, Software Engineering, and Programming Paradigms. He has experience in web development from an internship with Arcbest Technologies. Paschal was responsible for full stack development on both the API and front-end of the project.

Zachary Thiher – Thiher is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Software Engineering and Programming Paradigms. Thiher assisted in all areas of the project's development, but with a specialization on the user interface and user experience.

Randall Dickinson – Dickinson is the head of the Digital Lab of the Fay Jones School of Architecture, which contains the CNC routers, 3D printers, and the laser routers. He has enjoyed

his time giving students the ability to learn and develop their digital design and craftsmanship skills.

6.0 Facilities and Equipment

- For development, we used our own computers and equipment.
- All meeting were conducted remotely.
- All aspects of the project use cloud services instead of self-owned hardware and servers. Some examples include GCP App Engine to host API and serve Angular application, and a MongoDB instance hosted on Atlas.

7.0 Future Works

A few features and goals didn't make it into the project within the development timeline, so they had to be cut. However, these features could be developed and implemented into the application sometime in the future. Here are some possible future development opportunities that didn't fit within the scope of the project.

The application could include a live video feed of the 3d printers and laser routers that users could view at any time. Cameras are already present on each of the machines so staff at the lab can monitor the jobs, but there's no unified system that allows easy access to all the video feeds. A system can be created that incorporates these video feeds directly on the application. Students would be notified that their submitted job has begun printing and they would be able to go to the application and view the live feed of their job. Staff could also have access to every feed from all the machines in one location, making spotting potential malfunctions and monitoring the machines quick and easy.

When a student wants to use the application, they need to make a new account if it is their first time accessing it. These accounts are exclusive to the application. Work can be done with the University of Arkansas' IT department to allow for Active Directory (AD) authentication. This would allow students to use their native University accounts like they would for other University systems.

The application currently doesn't have any replacement functionality for resource scheduling, so the lab still must use R&R Booking to solve this issue. Creating a new proprietary resource scheduling system for the lab would not only allow for more granular control of requirements and features needed for lab operations, but also lift the large financial burden that comes from licensing R&R Booking.

Finally, automatic invoicing could be added that would calculate the cost of the print based on the file and material selected. The student would be emailed their generated invoice. This would further reduce the amount of work for the lab employees.

8.0 References

[1] *Introduction to the Angular Docs*, https://angular.io/docs

[2] *What Is MongoDB?*, https://www.mongodb.com/what-is-mongodb

[3] *Search, Reserve and Collect*, https://www.randrbooking.com/

[4] *MakerOS*, https://makeros.com

[5] *Fabman.io*, https://fabman.io/