University of Arkansas – CSCE Department

Capstone I – Preliminary Report – Fall/Spring 2020

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.

Initially this project aims to provide a minimum viable solution that will operate as a ticketing system, allowing students to register and submit job files to act as a replacement to the current emailing workflow. We are utilizing the MEAN (MongoDB, Express.js, Angular, Node.js) stack to develop a SPA (Single Page Application) and then deploying the Docker image to GCP (Google Cloud Platform). The initial scope of this project is to create a 3D print job management system, but there is a stretch goal to implement some sort of resource scheduling capabilities which would make this project a potential candidate to replace R&R, reducing licensing costs to the lab.

1.0 Problem

Currently the build lab space has had to adapt 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.

Regarding 3D printers, normally the student 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.

Regarding the remote laser cutter jobs, often there are upwards of 10 files that need to be individually cut 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 so that 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 job files. One of the drawbacks 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 the job files. 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 like things not getting printed in 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, to complete a minimum viable product, is to create a ticketing system that will aid in organizing the job submissions. This application will implement a user management system, job tracking, and integration with the 3D printers by using Raspberry Pi computers running OctoPi. This ticket tracking system will allow students and lab technicians to communicate changes and share project files, as well as calculating and generating invoices automatically. Students do have to pay for the time and materials that are used in their jobs, so the invoices act as both a receipt system as well as a tracking system throughout the job. Additionally, there are sometimes requests that come from external sources like other architecture firms in the area, and there is an additional invoice with different prices. The application itself utilizes the Angular framework to create a seamless SPA (Single Page Application) in a web browser. In addition to a just a web client, the app will be responsive and render correctly on various mobile devices.

3.0 Background

3.1 Key Concepts

The actual SPA (Single Page Application) web interface will be 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 will bolster our ability to create the front-end web page and simplify the development process. The backend hosting of our application will use Node.js as the main server process. Express.js is a library that will be used to handle all the incoming requests, either routing client requests to API endpoints, or serving the single-page application files. When the Node.js server starts, the Mongoose library will be 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 will use MongoDB for storing persistent data for the tickets and initially for the user database. Eventually we want to integrate our application to AD (Active Directory) to allow for native @uark account authentication. Currently, U of A Fab Lab licenses a service called R&R Booking. R&R is a web-based equipment checkout system that allows you to make any item available for reservation [3]. Although R&R is a powerful tool with lots of functionality, the university only uses it for basic scheduling. If we have time after completing the rest of our project, we could create a proprietary scheduling system and save money for licensing an outside product.

Before selecting 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, $191/month for 5 employee seats, but 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 an NFC (Near Field Communication RFID) card. Both platforms have implemented sections of what we are planning to do, but like the current R&R solution, the monthly expenses are a substantial justification, and many of the features that are being marketed would not be fully utilized by the build lab space.

4.0 Design

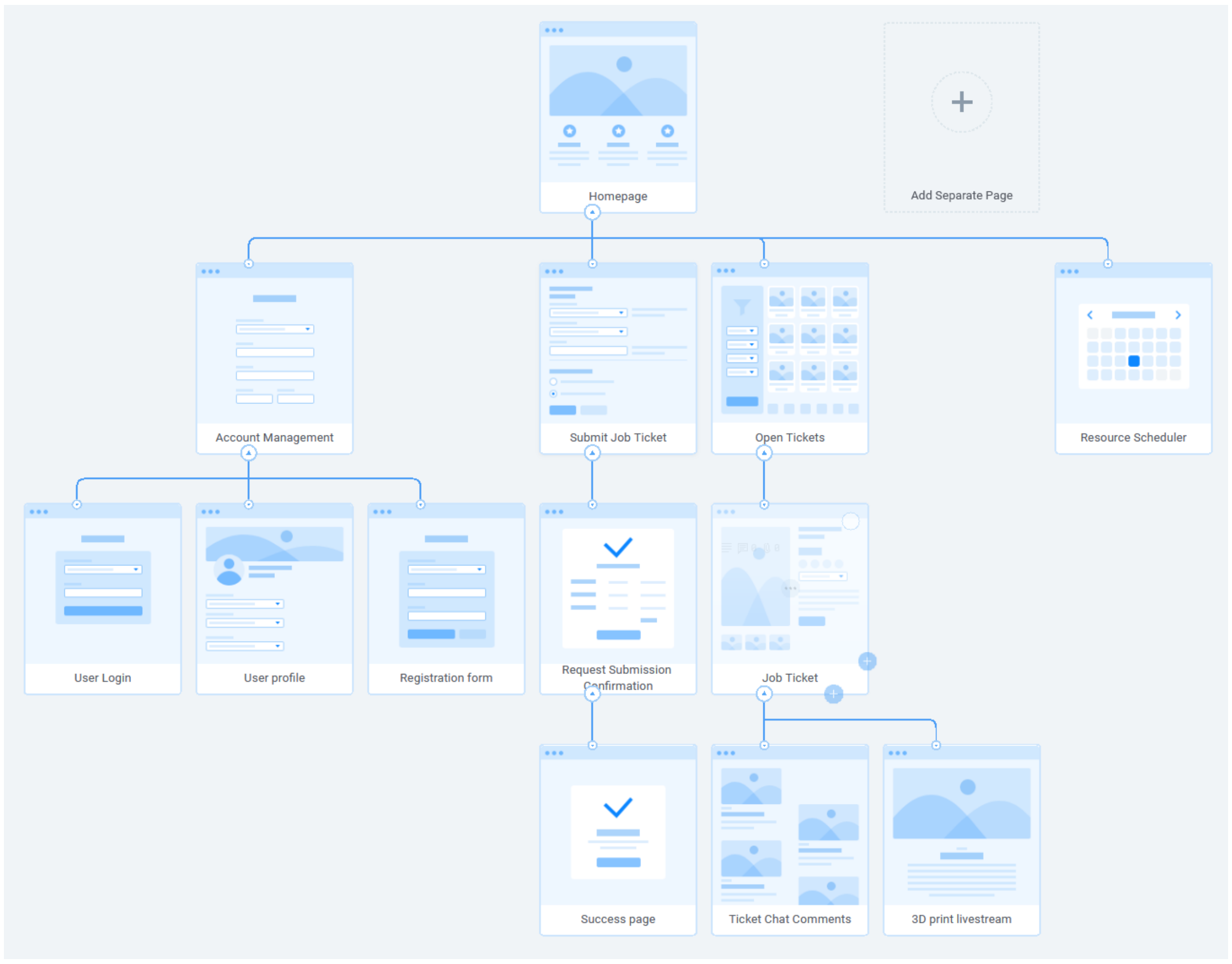
4.1 Requirements and Design Goals

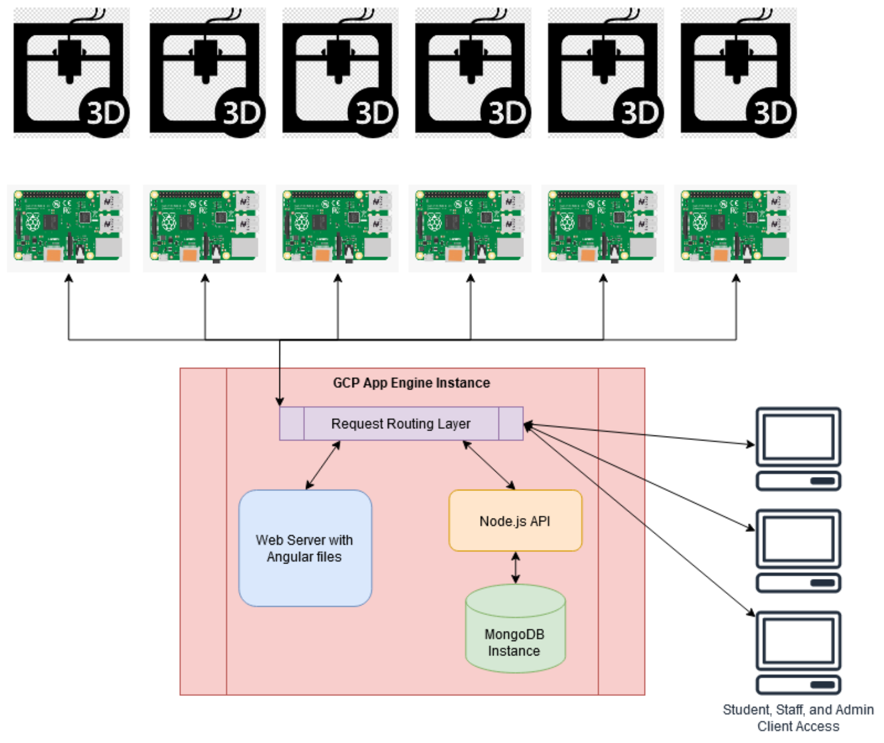
Initially, we are going to implement a user system. For this application, we are going to have three user roles; student, staff, and admin. Students will have the role of student, student lab technicians will have staff roles, and the full-time staff at the lab will have the admin role. For the user management system, users will be able to create accounts and be able to reset their password without staff intervention. After this application is fully integrated, we would like to work with UARK IT to allow for AD (Active Directory) authentication so that native uark accounts can be used to sign into the application. The different user roles within the application will limit what users can do, where staff functions like closing job tickets, commenting, and updating invoices will be restricted to the staff and admin roles. Student user roles should allow for job submission and commenting and closing tickets that that user owns.

Within the application there will be a ticket module, where users will be able to interact with the ticketing system. Student roles should be allowed to 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 should be able to add comments to the ticket, update the ticket status, and close out tickets once they have been completed. Additionally, staff and admin roles should be able to interact with the OctoPrint interfaces running on the Raspberry Pi computers in order to start, stop, or pause print jobs on the 3D printers. Additionally, OctoPrint has the capability to present livestreams of the print jobs in action, so once the ticket is updated to a “in progress” state, the student will be able to view the livestream of their jobs being printed. The main goal within the ticketing module is to create a user interface to allow for job submission, and a module that will list owned tickets and allow users to interact with the ticket. After there is a mature ticketing system implemented, we have a stretch goal to implement a resource scheduler. For the resource scheduler module, this will allow students to schedule time slots for the laser cutters when the lab space resumes in person appointments. Currently the lab space is utilizing R&R which is a library software. However, this paid software has many features that simply do not have a place within the build lab, and result in the lab paying for a license that is not fully being utilized. Once we implement a resource schedule, we open the cost for the license to be re-invested in other areas within the lab.

4.2 Detailed Architecture

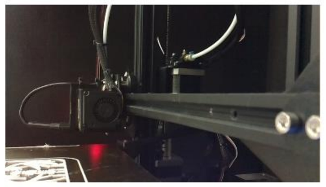
Fay Jones Lab Request System Angular App will initially have four main modules: account management, job ticket submission, ticket management page for staff, and a resource scheduler. Within each module, each of the pages is an Angular component, allowing for easy segmentation of tasks during development. Aside from the Angular frontend and API, we will be integrating with the 3D printers via the Raspberry Pi computers that are running OctoPi and connected to each printer.


Homepage
Add Separate Page
Account Management
Submit Job Ticket
Open Tickets
Resource Scheduler
User Login
User profile
Registration form
Request Submission Confirmation
Job Ticket
Success page
Ticket Chat Comments
3D print livestream

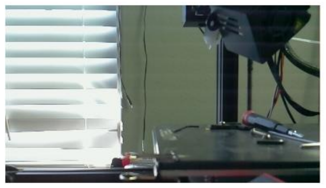


picam

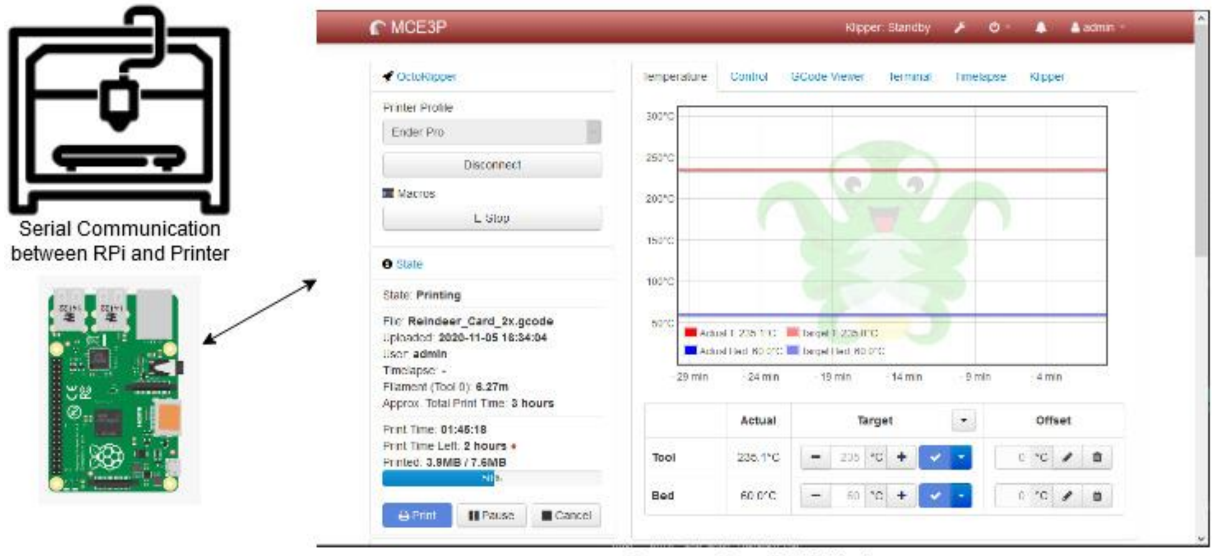
picam



picam



The OctoPi application interfaces with the printer control board and sends the code for the 3D print jobs over a serial connection. When building out this project, there are webcams available that will allow for remote viewing of the printers. OctoPi also has a REST API that can be used to query info about stats in real time, automatic uploading of job files, allow students to get notification and a livestream of their print jobs, and alert when jobs are finished.



State: **Printing**

File: **Reindeer_Card_2x.gcode**
Uploaded: **2020-11-05 18:34:04**
User: **admin**
Timelapse: **-**
Filament (Tool 0): **6.27m**
Approx. Total Print Time: **3 hours**

Print Time: **01:45:18**
Print Time Left: **2 hours**
Printed: **3.8MB / 7.6MB**

	Actual	Target	Offset
Tool	235.1°C	235 °C	0 °C
Bed	60.0°C	60 °C	0 °C

Serves OctoPi Web App

By utilizing webcams, we will provide staff with a viewing page that will give an all-in-one view to all the printers, instead of requiring multiple tabs open to view each separate machine’s video streams. Additionally, this will allow the job owner to watch their print or request as a time-lapse.

4.3 Risks

Risk	Risk Reduction
Hosting Solution	We will package software in a Docker container, a package that contains all the software, libraries, and configuration files needed to run the application. This will allow for the most flexibility in how and where the app can be hosted.
Internet goes out on campus	The application will be hosted on GCP (Google Cloud Platform) to allow for high availability. Printer updates would be interrupted because of the lack of internet on campus, but the application would 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 would provide 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 will have list of staff members that should be alerted either via email or text message when new jobs are submitted.
Job tickets price tracking could be incorrect.	Testing will be performed to ensure that total calculations are correct. Application will handle edge cases and do basic error checking before generating invoices.

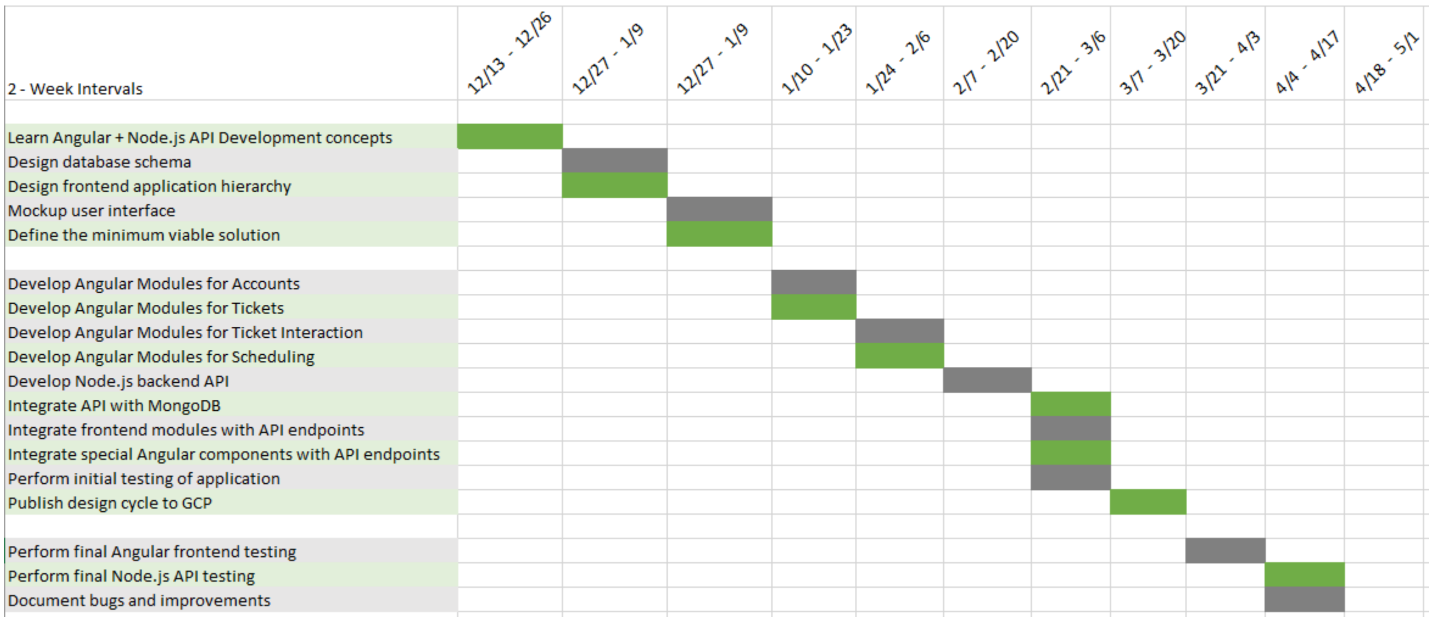
4.4 Tasks

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 Angular Modules for Scheduling
   - 3.5. Develop Node.js backend API
   - 3.6. Integrate API with MongoDB
   - 3.7. Integrate frontend modules with API endpoints
   - 3.8. Integrate Angular components with special functions with API endpoints
   - 3.9. Perform initial testing of application
   - 3.10. 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

4.5 Schedule



2 - Week Intervals	12/13 - 12/26	12/27 - 1/9	12/27 - 1/9	1/10 - 1/23	1/24 - 2/6	2/7 - 2/20	2/21 - 3/6	3/7 - 3/20	3/21 - 4/3	4/4 - 4/17	4/18 - 5/1
Learn Angular + Node.js API Development concepts	🟩										
Design database schema		⬛									
Design frontend application hierarchy		🟩									
Mockup user interface			⬛								
Define the minimum viable solution			🟩								
Develop Angular Modules for Accounts				⬛							
Develop Angular Modules for Tickets				🟩							
Develop Angular Modules for Ticket Interaction					⬛						
Develop Angular Modules for Scheduling					🟩						
Develop Node.js backend API						⬛					
Integrate API with MongoDB							🟩				
Integrate frontend modules with API endpoints							⬛				
Integrate special Angular components with API endpoints							🟩				
Perform initial testing of application							⬛				
Publish design cycle to GCP								🟩			
Perform final Angular frontend testing									⬛		
Perform final Node.js API testing										🟩	
Document bugs and improvements										⬛	

4.6 Deliverables

- API Documentation using swagger, a tool that provides auto generated documentation based on markup tags within the API codebase.
- README document describing how to setup development environment.
- Design Document: Will provide a detailed architecture of the final project, and the database schema to make it easier for future developers to work on the code.
- The MongoDB database: Will be managed by the university, and contains all tickets submitted – both historical and current.
- 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 is 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, Database Management Systems, and Algorithms. 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 will be supplying his skills for this project by taking responsibility for front-end UI/UX design and 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. Although his experience is with Vue, he's ready to apply his skills to a new framework. Paschal will be 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 will assist in all areas need, but with a specializing on 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 will use our own computers and equipment, and will not meet in-person.
- We will use Raspberry Pi's to integrate with the 3D printing lab's printers.
- All aspects of our project use cloud services instead of self-owned hardware and servers, some examples include: docker to package application, GCP App Engine to host API and serve Angular application, and a MongoDB instance hosted on Atlas.

7.0 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/