Small Equipment Checkout System

Final Report

Website: http://sddec20-21.sd.ece.iastate.edu/team.html
Team Email: sddec20-21@iastate.edu
Team Number: sddec20-21
Client/Advisor: Leland Harker

Team Members/Role:
Thomas Smith - Scrum Master/Architect/Developer
Seth Jones - Integration Engineer
Samuel Sklar - Circuit Board Lead
Micheal Momot - Server/Database Engineer
Kailin Zheng - Hardware Design Lead
Shubham Chauhan - Interface Lead

Executive Summary

Development Standards & Practices Used

We participate in using an Agile focus in managing our deadlines and requirement coverage for the completion of the product. In using this method, we designated our client as the product owner, and one of our team members as our scrum master. Regular stand-up meetings are practiced to ensure proper communication is maintained. Effort points are used to gauge complexity of particular tasks that members need to complete as well as estimate the velocity of the team to better estimate deadline completion.

Peer reviewing will be practiced, to reinforce a high quality of code is maintained by the team. Quality control will be utilized by incorporating Git into our development process. Commit messages will be written by members using guidelines encouraged by the Git community.

Summary of Requirements

Please refer to section 2.1 for a summary of the requirements required for successful completion of the product.

Applicable Courses from Iowa State University Curriculum

Classes useful for software: CPRE 181, COMS 227, COMS 228, COMS 321, COMS 363, ENG 311, CPRE 388, COMS 309, COMS 319, SE 329, COMS 487

Classes useful for hardware: CPRE 281, CPRE 288, CPRE 381, EE 201, EE 230, CPRE 489

New Skills/Knowledge acquired that was not taught in courses

Skills learned include learning to interface with a third party API to access a remote database. We haven't dealt with that kind of challenge in our classes, though our classes have equipped us with the knowledge to work through this challenge. Linking a Raspberry Pi to a web-app and server backend is another skill set we have acquired.

Designing and integrating a circuit board with a One-Wire communication system is a very unique situation that none of us had ever been in. New skill sets have been acquired by teammates such as circuit design, soldering, and embedded testing.

Table of Contents

1 Introduction

1.1 Acknowledgement

We would like to thank Iowa State University for allowing us the opportunity to work on this product. We would also like to thank our client and advisor Lee Harker, along with the Electronic Technology Group(whom we will be referring to as ETG for the remainder of this document) for providing us with the resources and guidance needed to achieve the best possible product for our client. A moment should be taken to appreciate and pay respect to the documents from past teams who have worked on this product. Their past work has helped us realize what approaches work and don't work in terms of design.

1.2 Problem and Project Statement

Software, Computer, and Electrical Engineering students are often expected to use specialized equipment provided by the college to complete assignments, projects, or to simply enhance their learning experience. Currently students are expected to check out this equipment by way of interacting directly with the ETG's office and borrowing said equipment. While effective, this method can be cumbersome for the ETG staff and has certain limitations that can be hard to overcome.

Records of what students have checked out are currently kept manually, which can lead to mistakes and is generally a hassle if a large number of students check out a large number of things at once, which often happens during busy times of the academic year. In addition, staff must prepare the equipment for checkout as well as process a request. As a result students' time is also wasted as they must wait for ETG staff to complete the steps mentioned above.

The goal of our product is to alleviate these shortcomings by providing an automated system that enables students to check out equipment without having to interact with ETG staff, and allows ETG staff to manage checked out equipment without having to manually service students' requests. This will allow ETG staff to provide better services in other areas while providing students with a streamlined, smooth experience when it comes to equipment checkout.

1.3 Operational Environment

The finished product will be installed on the wall outside the ETG's office in Coover Hall. This will require the product to be hanging from a wall. The expected temperature the product will be expected to operate in is room temperature, as it will be held within the hall. Due to the public

space, each locker on the product is expected to prevent theft or other malicious actions against the items it stores. Part of this prevention will be a feature to detect if a door to a locker is left open by a user. If so, the product should notify the administration after a set amount of time that is yet to be determined.

The touch screen kiosk users will use to interact with the system will be attached to the product and will feature quick feedback to a user's interactions.

1.4 Intended Users and Uses

The product will accommodate two types of users. Normal users, which will consist of ISU students who have authorization to use the product. These users will be able to check out available items from the product by simply selecting the item they wish to check out and swiping their card or entering their ISU ID into the kiosk. This will prompt the product to open the appropriate locker and allow said user to take the item they requested.

Upon return of an item that was checked out, the student will simply repeat the process of swiping an ISU card or entering an ISU ID, followed by the user placing the item in the locker opened up by the system. Note that the locker selected by the system upon return of an item may not be the same locker that originally held that item.

The other type of user would be an Admin. Administrators will be able to access the system remotely by way of a web interface. This would allow them to see who has what items checked out, and if any subcomponents of the system need maintenance. Administrators would also have the ability to interact with the system though the touch screen interface if needed.

1.4.1 Engineering Constraints

Constraints:

- Product is to be mounted to wall, thus all wiring must be encapsulated in a way that supports this
- Main control board must fit within one of the storage lockers as required by client
- Product is to be left in a public space, and thus must have countermeasures to being tampered with.
- PCB boards need to fit into a provided mounting bracket on the locker doors.
- The customer has expressed a strong preference for a One-Wire system to connect the lockers together.

Non-functional Requirements:

- Allow code to be maintained by ETG post senior design
- Provide complete and useful documentation to support future maintenance of system
- Take steps to protect personal info of users of system
- Take steps to protect system from malicious attacks or accidental harm

1.5 Assumptions

Assumptions:

- Any SE, CPRE, or EE students and faculty are able to use the product.
- ISU card will be used to track who checks in/out items
- Product should be able to scale to have any amount of lockers
- Product should be available for use 24/7, with the exception of scheduled or emergency maintenance
- Normal Users will interact with product via a touch screen kiosk and card swipe
- Touch Kiosk is to be mounted/attached to product which will be hanging on wall
- Users will be able to manually enter their ISU ID in the event they do not have the ISU Card with them at the time of check out/in of an item
- Administrators will be able to access, modify, and update the system via a web interface

1.6 Expected End Product and Deliverables

- Four functional cabinets
- Source Code: Source code used in the product will be made available to ETG to allow maintenance and expansion of product.
- A website which can control the product remotely, and has the features present in the functional requirements.
- Documentation: Proper and thorough documentation will be provided to ETG on completion of the product to assist in understanding and maintenance of the product in the future.
- Operational Product:Four cabinets with magnetic locks and LED lights for each locker that are controlled via software, and a Printed Circuit Board for each locker that is connected to the server to enable software control of the locker.

2 Project Design

2.1 Functional Requirements

Functional Requirements - Users:

- Select an available equipment item to checkout
- Return checked out equipment
- LED light to show contents of locker
- Ability to view checked out equipment
- Ability to view available equipment
- Report broken or missing items

- Report broken parts of system
- Ability to choose checkout duration
- Reminders for students in the form of an email

Function Requirements - Admins:

- Login/Logout functionality
- View available equipment
- View users(students) who have checked out equipment
- Modify privileges of users
- Receive status reports
- Ability to add new lockers to system
- Ability to assign items to lockers
- Add new users
- Set checkout limits in specific items

2.2 Selected Approach

Background

This following design was selected after a combination of two things. The first was the introduction of COVID and the global pandemic that occurred afterwards. The second was our failure to properly design a PCB board in an adequate amount of time, leading to us having to scale back once again. More details on these revisions can be found in **Appendix 2**.

High Level Overview

Our approach to the problem resulted in us utilizing a client/server approach. This was born out of a requirement for ETG staff to be able to access the locker system remotely for admin related maintenance, while still requiring standard users physical access to the kiosk provided with the locker system. This also allows for the possible future enhancement of standard users to be able to check available equipment remotely as well without a major restructure of the system.

Figure 2.a gives an overview of our system design. The locker system houses the Raspberry PI that is treated as a client that interacts with the server via a web browser, and the individual cabinets with a OWFS server that uses a 1-Wire® system. The server itself is hosted on a VM managed by ISU. The kiosk acts as the interface that students will use to login, control the PI, and interact with the locker system.

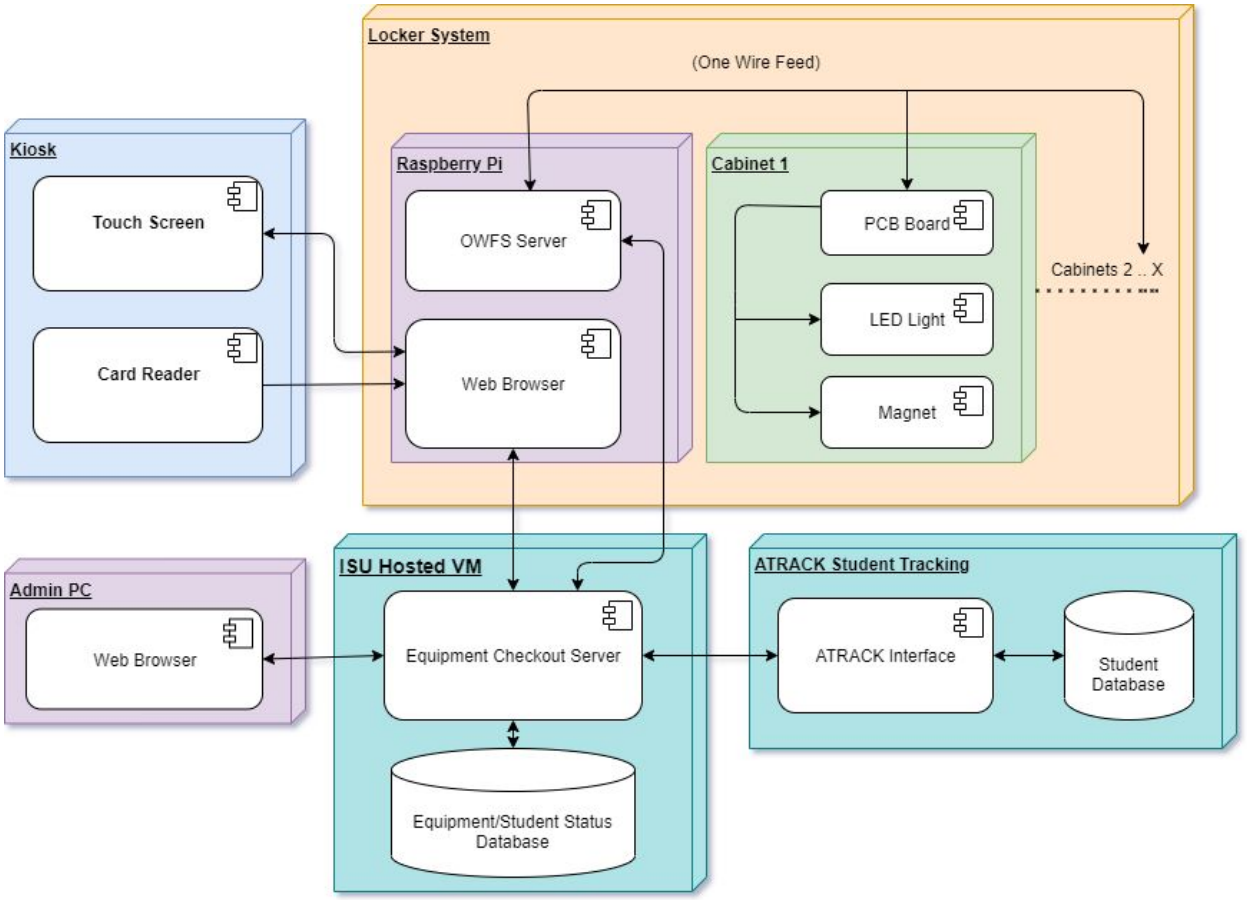


Figure 2.a

Hardware

Our client expressed a strong preference towards using a 1-Wire® system for controlling the locker system. We decided to use this approach in addition with a Raspberry Pi board that acts as the master device in the locker. This master device is responsible for issuing commands to the children cabinets, or slave devices via communication using 1-Wire®.

The customer also provided us with a touch screen kiosk and card reader. These devices are connected directly to the master device, and act as inputs to controlling it, similar to a keyboard and mouse.

Individual cabinets in the locker system will consist of a PCB board slave device hooked up to a 1-Wire® to allow for communication from the PI Board. The PCB board will control the magnet and LED light on the door. The LED light will be used to show the user what is present in the locker, and the magnets will be used to open/close the door to the cabinet.

Software

Login authentication will be done via the students using their ISU ID card with the card reader. The card info will be sent to the ATRACK database that is hosted by ISU for student verification. Afterwards, a final check will be done with the local database hosted on the VM that will check to make sure the student is in good standing. If that check passes, the student will have access to the locker for normal use.

Admins will log in via their remote workstation, providing their username and password to the login page. That info will be verified via the local database on the VM. On success, the Admin will have access to manage the locker system.

The website will have two separate versions. One for Admins, and the other for Students. The students page will focus on actively checking items out, seeing what items are available, and what items are checkout out, as well as any other function requirements detailed in section 2.1.

The Admin page will focus on adding new users, managing access, and any other functional requirements detailed in section 2.1. Figure 2.b details the flow of these use cases below.

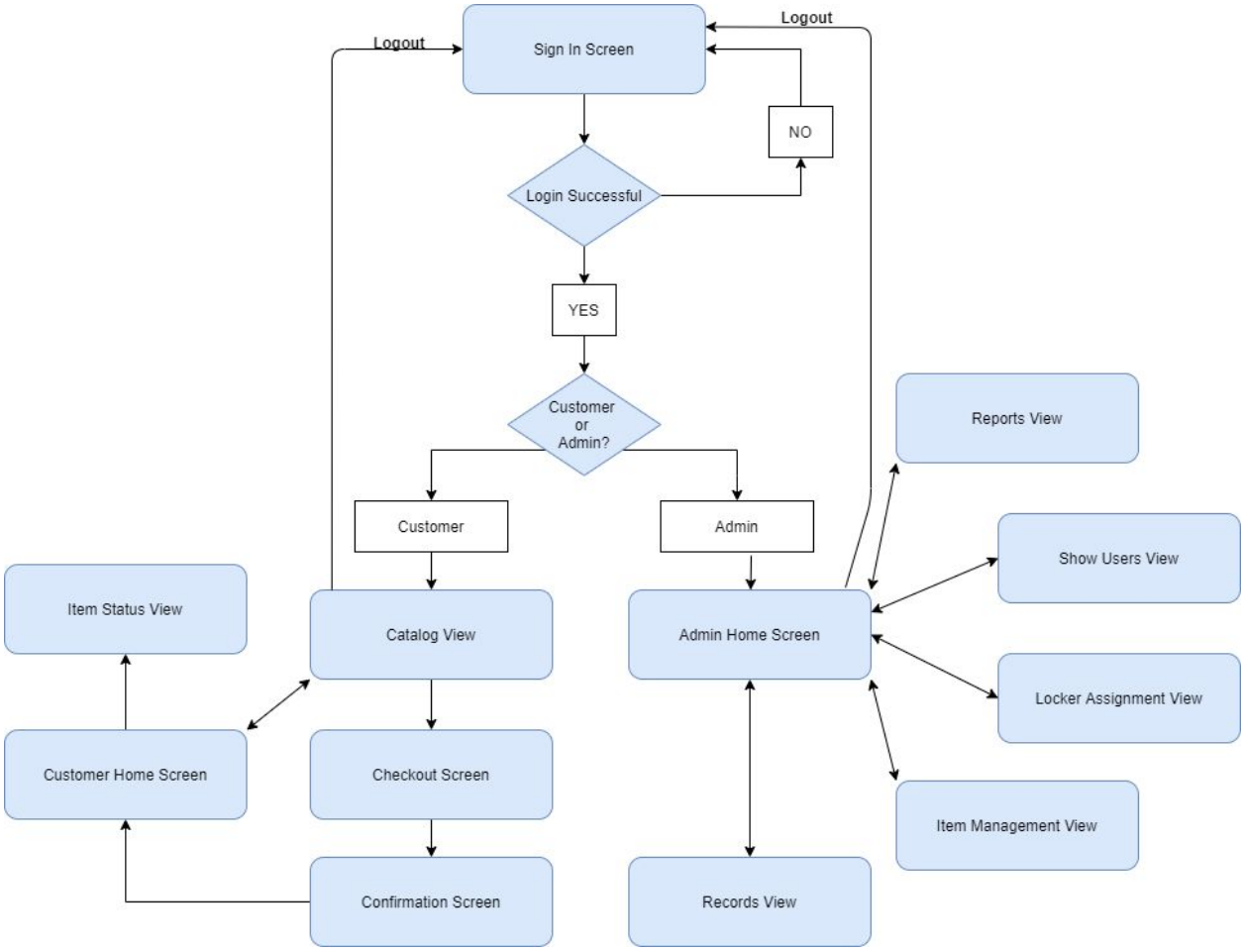


Figure 2.b

3 Implementation

3.1 Server

The server is hosted on a ISU VM, and is written using Node.JS. A challenge that we encountered was getting the Node.JS server to communicate with the 1-Wire® system present on the Pi. We found the solution in the form of an OWFS server. This server exposes the 1-Wire® system via TCP. Using NPM, a package management tool that comes with Node.JS, we were able to find a module that integrates with the OWFS server. This enabled our server to control the 1-Wire® remotely.

Express is used by the Node.JS server for routing. Express provided us with a lightweight web application framework that allowed us to make callbacks related to HTTP requests. These handler functions allow us to dynamically respond to user requests.

The codebase was written in Javascript. This choice was made because Javascript is agnostic to its environment, allowing it to be reused on both server and client side. This is helpful, as less code means less bugs, less maintenance, and less cost. The fact that Javascript is non-blocking language has its ups and downs. A particular down that we encountered was connecting our server to two separate servers. Those being the OWFS and MySQL servers. Race conditions have proven to be present, and required us to utilize the “promise” functionality of JavaScript.

A thin layer of Apache is run on top of our Node.JS server. This thin layer is responsible for re-routing unauthenticated users to the ATRACK service hosted by ISU. Once those users have been authenticated by ATRACK, Apache will send the user info to Node.JS via HTTPS headers.

3.2 Client

Clients access the server via any web browser of their choice. Currently, the only client a student would interact with is the Raspberry Pi, who is acting as the master device of the locker system. This means that the only way a student can currently access the locker is by interacting with the Kiosk connected to the locker system.

The intention is that this can be expanded out in the future, as all a new client needs is a web browser and a URLto connect to the server. An example of this would be expanding the server to allow for students to check the status of the locker system by use of their personal computers through a web browser.

The code used to generate the front end contains HTML, SASS, React, and CSS. React is used for creating the dynamic behavior in the interface consumed by the user. Part of the reason we opted to use React was because it allows for our web pages to update without requiring a refresh. Seeing as non-Admin users will be interacting with the system via a touch screen kiosk, this was deemed favorable to the alternative of requiring a refresh of the page to show updated info. HTML, CSS, and SASS were used to enhance the look of the site in order to make it easier for the end user to interact with the site.

3.3 Database

For the database, we used MySQL. This database’s purpose is to manage the state of the equipment of the locker system, keep track of who has equipment checked out, and record the current standing of students. Sequalize is an Object Relational Mapping library that we used to allow our server to programmatically query the database schema. Our intention using this is to

allow the database to be easier to maintain by the ETG staff due to its use of the Model-View-Controller pattern.

The one case that we will not be using our own database is in authenticating student login. ISU currently uses ATRACK for managing login data for students. We have been granted access to this service and its database. Upon the server receiving a login request involving a username that indicates a card read, it will forward that info to the ATRACK service.

Once the ATRACK service replies with info validating that student, the server will then check its local database to make sure the user has no restrictions placed on themselves. Examples of restrictions include checking out the max number of items or being blacklisted by ETG staff. Figure 3.a below shows the layout of our database.

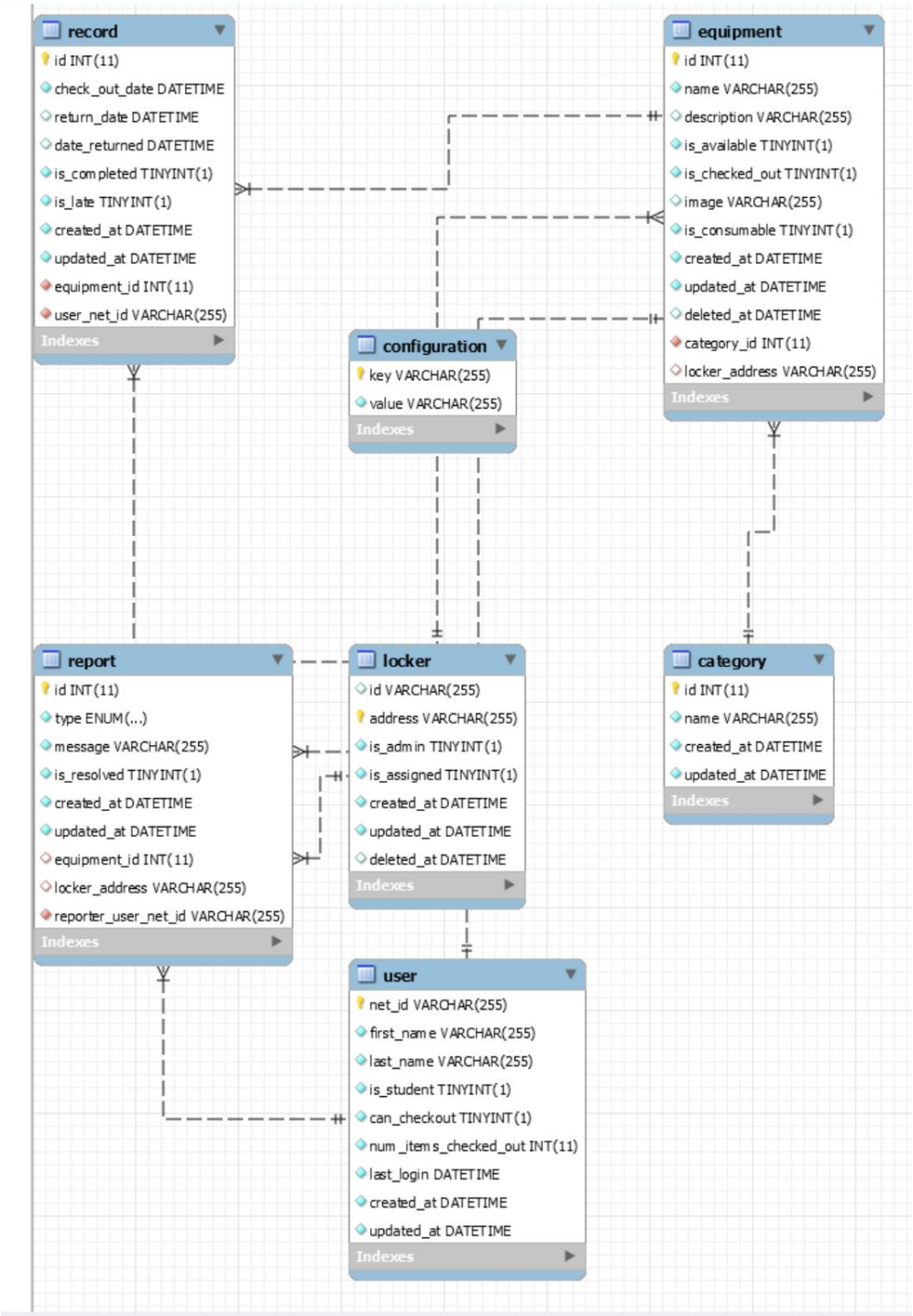


Figure 3.a

There are five main tables present in our database. These tables can be identified by the blue headers present on each section in Figure 3a. The entry in each table marked with a yellow diamond is the primary key.

The User table is used to represent the registered users in the locker system. Relevant fields are the “can_checkout” and “num_items_checked_out” fields, which are used to determine a user’s eligibility to use the system.

The Equipment table is used to represent the different equipment items present in the locker system at any given time. The Report and Record tables are used to keep track of which items are checked out and to who, as well as catalog issues reported about the system, such as faulty doors, missing equipment, or incorrect behaviour of the system.

The Locker table is used to represent the individual cabinets of the locker system. The decision was made to have the Equipment table contain the foriegn key that is used to match an item to a cabinet due to equipment not being “bound” to any given cabinet.

3.4 Hardware

A PCB Board was needed to control each cabinet in the locker system. This board needed to fit into a predetermined socket that is located on the inside of the cabinet door. This served as a constraint in the design process of this board.

As previously mentioned, the board will be controlled via the 1-Wire® system. The OWFS server is responsible for handling incoming requests and transitioning those requests into actions for the 1-Wire® system to take. Communication between the Node.JS server and the OWFS server will be accomplished via TCP traffic that is parsed and managed by the owfs package provided by npm. From there, the master device, which is attached to the Raspberry Pi, sends commands to the slave device PCB boards. There are three data lines between the Pi and each locker. Two are to send data to the lockers, to turn the LED on/off and to open the locker, and one is to receive information about whether the door is open, which is done with a hall effect sensor.

The boards are chained together with 3-pin connectors between each other. Since each locker has a unique 1-Wire identifier, they can all be individually controlled with one data line. In addition to the data line, there is a 12V rail and a ground. Together, these three lines (data, 12V, ground) make up the bus.

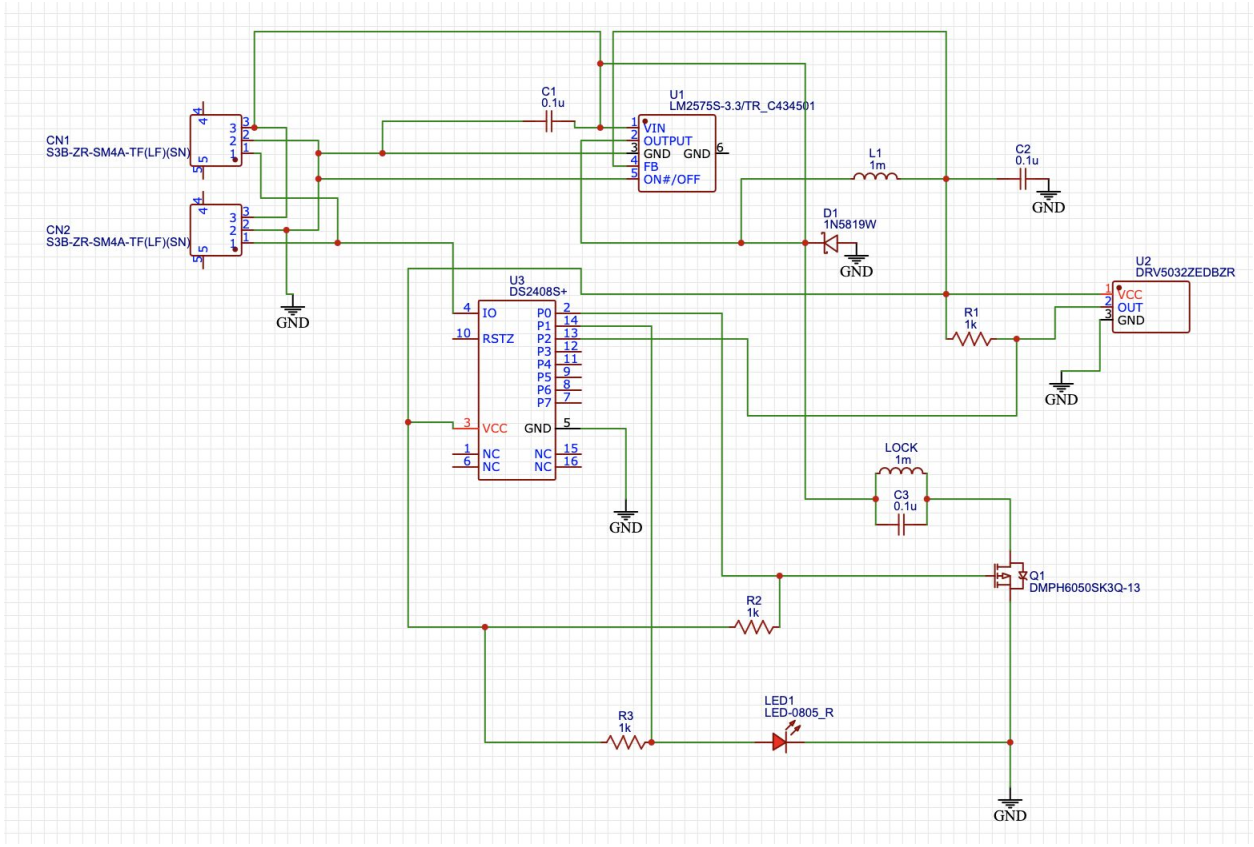


Figure 3.b

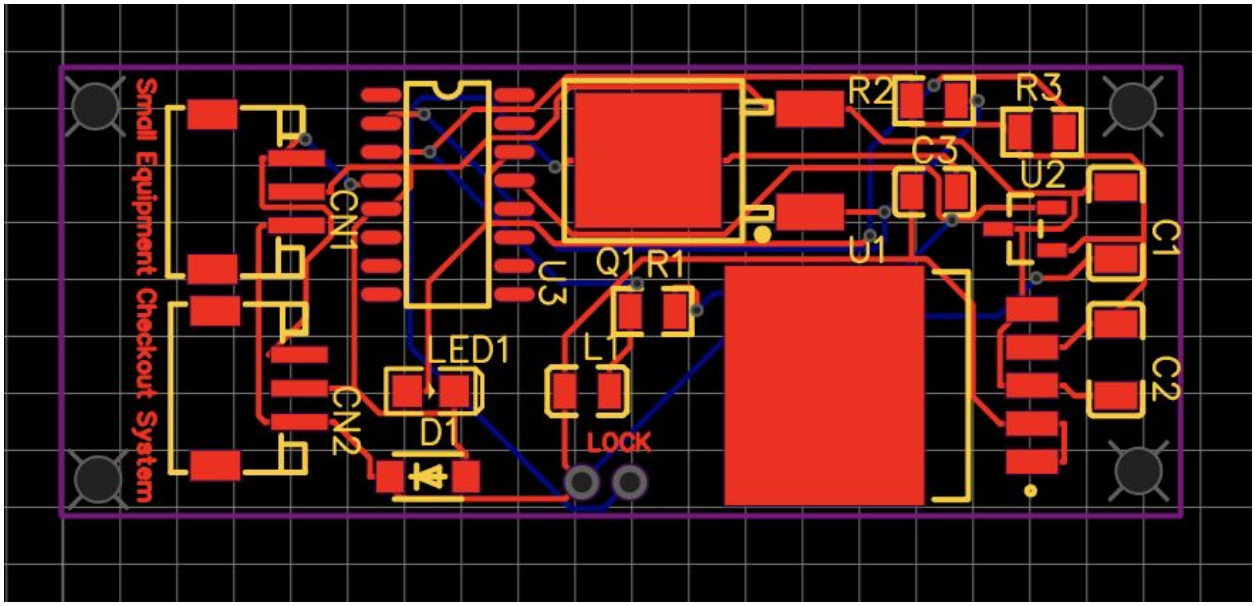


Figure 3.c

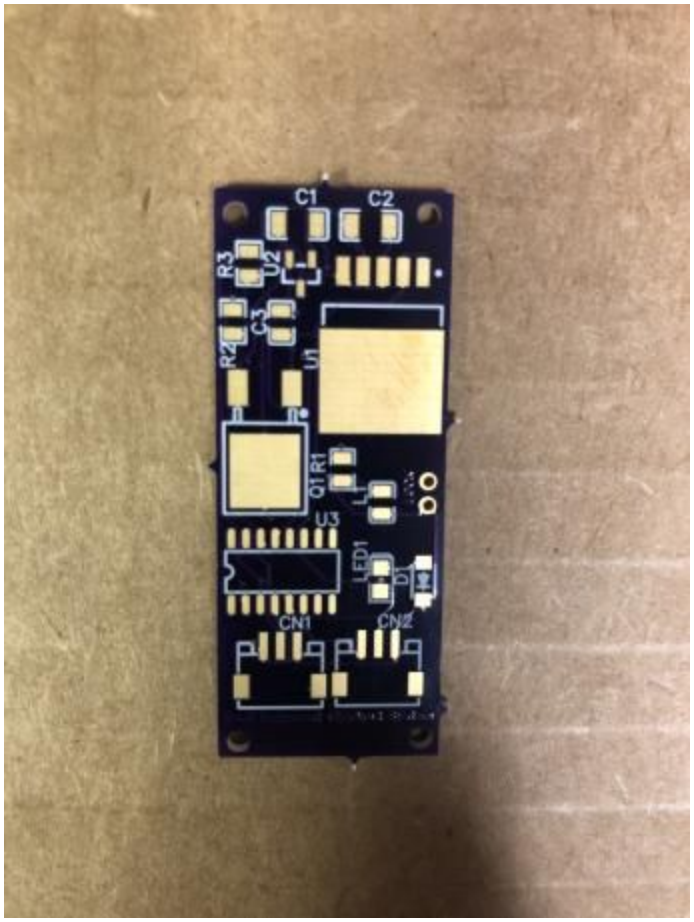


Figure 3.d

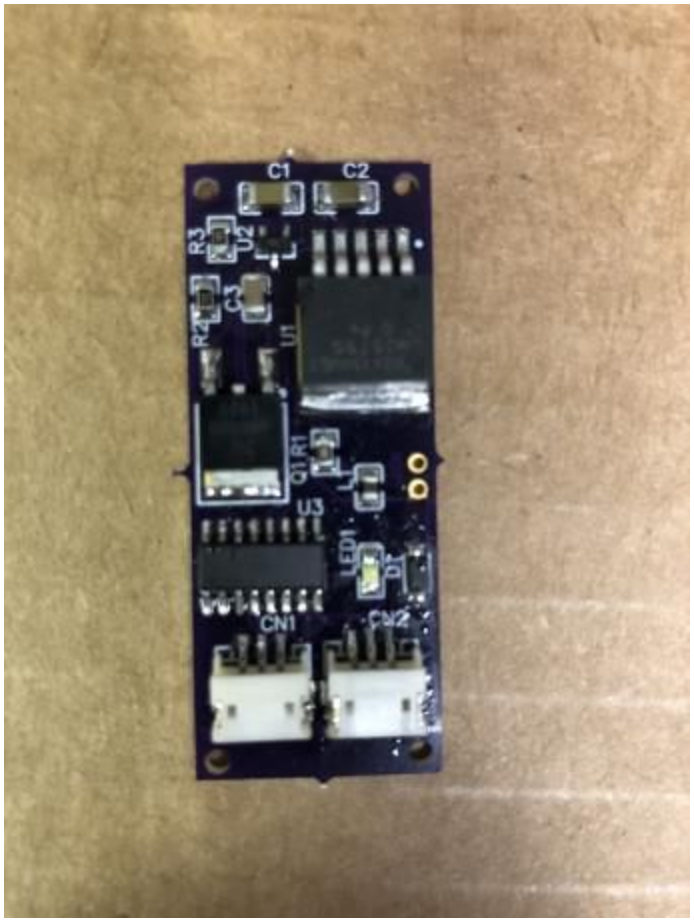


Figure 3.e

3.5 List of Tools Used

UI Frameworks/Libraries

- HTML
- CSS
- JavaScript
- Sass
- Bootstrap
- React
- Redux

Building:

- Babel
- Webpack

Helper Libraries:

- Sequalize
- ESlint
- EsDoc
- Apache HTTP SErver
- ATRACK

Runtime Environment:

- Node.JS

Skeleton Frameworks:

- Express
- OWFS

Miscellaneous:

- Testing -> Jest
- Package Manager -> NPM
- Database -> MySQL
- Version Control -> GitLab

4 Testing Processes and Testing Results

4.1 Software

Approach Taken

Jest was used for unit testing both the back and front end applications. The unit tests validated functionality by providing mock input and checking for the expected result from the tested module. Beyond basic Jest tests, manual testing was done to validate the remaining aspects of the software. This was undesirable, but was chosen due to time constraints caused by COVID along other outside aspects that will be elaborated on in **Appendix III**.

Frontend UI elements were tested via a visual inspection. The integration of the backend and frontend was also performed manually, as we lacked a proper test framework to allow for automation of that process.

Database testing was performed using Sequalize. There is a terminal interface that can be used to manually inject data and queries into the database. This proved useful and allowed us to test our schema independently of the rest of the system.

Desired Approach

If more time was allotted, we would have liked to see a continuous integration and testing approach taken to validate functionality. This would have better reinforced the standards laid down the Agile process we followed, as well as provided easier maintenance by ETG in the future. An automated system would have been preferred, such as the Bamboo tool provided by the Atlassian suite. These tests would have been run on regular intervals, and on each merge into the master branch.

Results

The major functional requirements for the software have been tested and verified through the processes defined above. Issues are currently present in the calls to the MySQL and OWFS server hosted on the Raspberry PI. This is due to the asynchronous nature of Javascript. There is currently a chance the Node.JS server responds before each asynchronous call is complete by Javascript code. At the time of writing, this issue has not been resolved.

Login works as intended for both Admin and Student users. Communication to and from the ATRACK system has been shown to work to the point of completing the relevant functional requirements.

UI elements have been shown to work as intended, and are complete enough to fill out the relevant functional requirements, with one exception. There was a desire by the customer to have the UI show if a door was left open. We were unable to test this functionality well enough to call the feature complete.

4.2 Hardware

Approach Taken

Due to our teams inexperience with designing and testing hardware, we were unable to find a testing method that wasn't manual in nature that we could implement in the time frame we were given. Thus, each iteration of hardware designs we made were manually tested with the locker. This was accomplished by sending signals through the 1-Wire® to a cabinet and observing the result. Behaviour that was observed was whether or not the door opened or if the LED light turned on/off.

Desired Approach

We were not aware of the ability to test each part with an ammeter or voltmeter until we had already ordered our final iteration of the PCB board. This would have allowed us to make sure

the voltage worked correctly with the magnetic lock present on the door before we went to the trouble of fully integrating the system together.

Results

Our progress on each iteration of board design was long and slow due to the inexperience mentioned above. It took our team much longer than anticipated to reach our final board design. Our final board design did respect the constraints laid out by the customer by fitting into the provided socket located on the inside of the door.

4.3 Integration

Approach Taken

A manual testing approach was required due to the nature of integrating hardware with software. This proved to be one of the most difficult tasks, due to limited access to the TLA, as well as our primary hardware integrator being forced remote due to circumstances generated by the COVID pandemic. Testing was done by physically hooking up the Raspberry PI to the 1-Wire® system and attempting to open a lock via the kiosk attached to the Raspberry PI.

Results

The full stack approach we took to integrating led to bugs often popping up. These bugs were hard to track down, due to all the separate components present in the final system. This proved costly, and ate up alot of our time. This area was tagged as high risk, and that risk ended up becoming a reality.

5 Related Products

Why not use something already made?

A good question. While there are two main other versions of automated locker systems, neither allow for the desired functionality requested by the customer. In addition, these lockers would need to utilize third party software not maintained by ISU or ETG, thus removing a layer of independence from ETG when it comes to maintaining their equipment system.

AB&R Automated Locker System

A good locker system that has been optimized for use in warehouses or use in construction sites. Allows for in cabinet charging as well. The problem with this one though is that it does not allow for remote management via Web-App like the customer wants. It also doesn't provide the ability to authenticate user login with the pre-existing ATRACK system the ISU uses.

AUTOCRIB AutoLocker® FX

This locker system does feature a cloud based management system for tracking equipment status, making it possible for remote management by ETG. However, it does not have the ability to have the dynamic reporting the customer wants, and also does not allow for authentication using ATRACK.

6 Appendixes

I Operation Manual

I.a Operational Manual For Small Equipment Checkout System

Steps for the hardware product setup:

- Attach to the Raspberry Pi(Figure I.a) the master board(Figure I.b). The header extender will be added to the header, and instead of being off to the left, the board will sit directly above the Pi.

Figure I.a

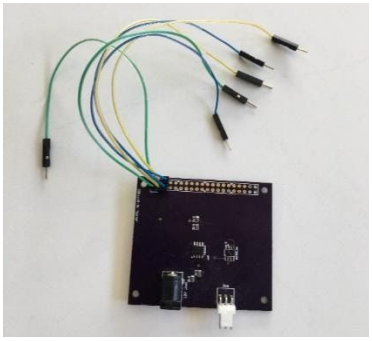


Figure I.b

- Plug in the Pi to an external power source for electricity. It will detect the master board when it boots up and begin the OWFS services automatically. Link the locks to the slave boards(Figure I.c) that fit.

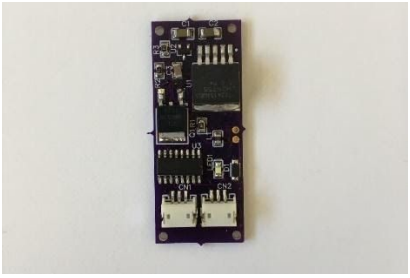


Figure I.c

- Link the bus to the Slaves and Master Board. To allow for daisy-chaining of the slave panels, the slaves have two links.

- On the master board, attach the 12-volt power source(Figure I.d) to the jack. This will allow the central bus to link to the 12-volt line.

Figure I.d

- Attach the card reader(Figure I.e) to the Raspberry Pi to enable card reading functionality.

Figure I.e

<u>Steps for the software usage of the product:</u>

- Students or Admin should swipe their Iowa State University card into the card reader located next to the locker system while making sure that the login entry field on the website is being clicked on. After the card is swiped it will be authenticated by the ATRACK to authenticate. After the authentication, the user should be logged in.

If an admin is logged in, then the below steps can be performed:

- Home Page -> Records -> Shows the Record number, Equipment, Returned, Is Late, date checked out, Due date, Net Id, First name, Last name.
- Home Page -> User -> Can see the user records like netId, first name, Last name, is Student, Can Checkout, Number of items Checkout.
- Home Page -> Equipment -> Register a new equipment -> Type necessary details as prompted -> Click Submit.

- Home Page -> Equipment -> Manage categories -> Enter the name of new category -> Click Submit.
- Home Page -> Lockers -> Register a new Locker -> Enter the name of the category and select the locker from the list -> Click Submit.
- Home Page -> Reports -> Shows the reports of any kind of damage.
- Home Page -> Click Logout -> Takes to the home screen.

If a student is logged in, then the below steps can be performed:

- Home Page -> Catalog -> Items Available to checkout -> Is Item in specified locker -> Yes -> Click Submit -> Successfully checked out.
- Home Page -> Return/View Activity -> Shows users past records of checked out equipment.
- Home Page -> Return/View Activity -> Select a checked out item -> Click return item -> Select yes/no if item is damaged -> Click Submit.
- Home Page -> Help/FAQ -> shows the equipment that can be checked out, duration it can be checked out for, if the item is broken -> click on Report an Issue.
- Home Page -> Help/FAQ -> Report an issue -> Type of Issue -> Specify locker item was returned -> Description of the issue -> Click submit.

II Alternative/Initial Versions of Design

II.a Initial Design

Our initial design featured a few extra features as well as more cabinets being delivered to the customer. One of those features was a pressure plate sensor present in each cabinet. This would have allowed the system the ability to detect if the equipment was present in a particular cabinet.

An advantage to this would be that the system would catch potential mistakes made by students, or if a student pretended to return something, but didn't. It could have even been scaled up to detect how much of something is left in the locker based off of weight, like how many pieces of wire are left in a spool.

Sadly, this feature was cut upon the introduction of COVID. It was never a functional requirement presented by the customer, but rather an enhancement suggested by the team, making its exclusion not critical to the delivery of the product.

Another difference was that the server was going to be hosted on a PC managed by ETG. This choice was scrapped once we considered the extra steps that would need to be taken by ETG in

the event they try to expand the system. Once a VM was suggested, it was quickly agreed that would be the preferred approach to hosting the server. A VM also allowed for easier security, and would easily be accessible to anyone on campus in the event that ETG choses to extend the product's functionality.

Limited remote access to the system was originally planned for students. This would allow for students to check the inventory of the locker system from their dorms, possibly saving them a disappointing visit to Coover.

This feature was also cut because of COVID. The intention is that our system will be scalable enough to allow ETG to include this feature in the future after delivery.

There are 36 cabinets present in the locker rack provided to us. That means that 35 cabinets are open for regular use (one must be reserved for the Raspberry PI). The original requirement laid out by the customer was to have all 35 of these cabinets be complete at the time of product delivery. This requirement was scaled back to 4 cabinets after the delays and restrictions imposed by COVID. This came with the caveat that our system be scalable so that ETG can easily include the remaining 31 cabinets after delivery.

Figure A.1 below shows the initial concept sketch of our system.

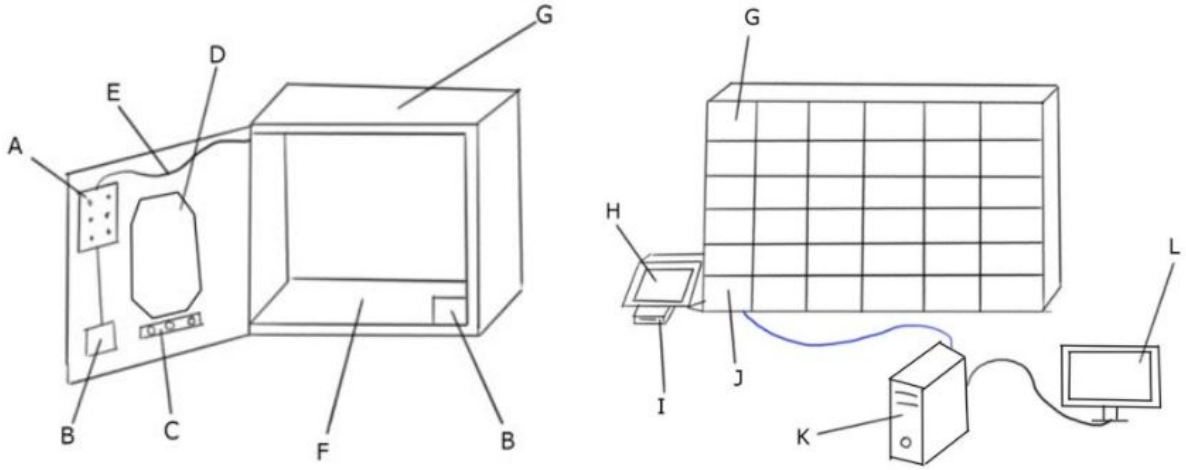
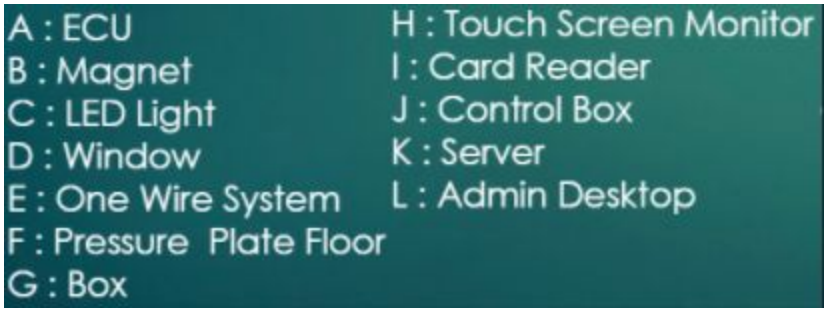


Figure A.1

Key



III Other Considerations

III.a Things Learned

Prior to this, none of us had experience with a 1-Wire® system, or the OWFS server that accompanies it. Learning this system has proven to be a challenge, but a rewarding one at that.

For three members on the team, this was their first time interacting with the React framework and Apache service. This proved beneficial as some of these members plan to pursue a career in the Web Development field.
Hosting a server on a VM and accessing the server from a separate device was a new experience for the team as well. Requesting certificates through ETG and configuring a server to use HTTPS was something we have never done before. Through accomplishing this, we gained valuable experience in networking that was not present prior to us working on this product.

III.b Unexpected External Factors

2020 has been a year full of unexpected circumstances. Sadly for our team all of those manifested in negative ways. Throughout the entire two semesters, our team was only granted 3 months of us all being able to meet in person. From March forward, we were forced to work remote on a product with hardware development involved.

Compounding this, when we finally regained access to work on the product, that access was limited to weekdays only and for allotted amounts of time. We also had our individual who was leading the hardware effort be forced to go remote due to circumstances caused by COVID. In addition, we had another individual who was leading integration test positive for COVID near the end of the semester, leading to an extended period of time with little integration happening.

Lastly, the second semester was shortened in order to accommodate the holiday season in the setting of the pandemic. This gave us less time than originally allotted.

III.c Hardware Learning Curve

This product included a large amount of electrical design in the form of the PCB board, along with the integration of the 1-Wire® system. Of the six team members present on this project, none of them were an Electrical Engineering student. This meant that we had to have a Computer Engineering student step up to fill the gap. This was deemed a high risk solution, and ended playing a large part in our team's inability to deliver the full vision of the product to the customer.

7 References

- https://facebook.github.io/jest/
- http://owfs.org/
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://jsx.github.io/
- https://httpd.apache.org/
- https://www.mysql.com/
- http://docs.sequelizejs.com/
- https://expressjs.com/
- http://sass-lang.com/
- https://www.npmjs.com/
- https://nodejs.org/en/
- https://reactjs.org/