

Silicon Tracking Solutions

Brett Hayes, Joseph Cronise, and Dylan Camus
CS463 / Spring 2016 / Group 03
Final Report

Abstract

This document contains the work of Brett Hayes, Joseph Cronise, and Dylan Camus pertaining to the Silicon Tracking project. This project is a web-based inventory system used by the Performance, Measurement, and Analysis team at Intel in Chandler, AZ. The inventory system we have designed tracks the information and status of CPU silicon used by the team and is made to help them to know where their high value inventory is at all times, as well as saving the team time and effort towards obtaining the items they need to get their work done. We have designed for them a system such that every time an engineer within their lab needs to use an item, they will need to check out the item through our system and check it back in when they are done. This ensures that highly sensitive materials don't go missing without a log of who has had ownership of the item, as well as letting others know who currently is in ownership of the item.

Silicon Tracking Solutions

I. INTRODUCTION

THE Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi viverra tortor at nunc tristique, et faucibus magna tempus. Vestibulum elementum dolor in porttitor facilisis. Vivamus commodo nibh nulla, eget suscipit felis tincidunt sed. Proin ut metus diam. Phasellus consectetur congue elit, quis vehicula nunc facilisis quis. Morbi dapibus risus vitae felis blandit pulvinar. Maecenas fringilla leo sed dictum pulvinar. Sed tempus, felis vel tincidunt varius, dui nulla laoreet orci, vel accumsan magna erat sed est. Aenean luctus quam a feugiat malesuada. Duis viverra lectus purus, id malesuada risus bibendum at. Aenean malesuada, magna id facilisis dapibus, leo nulla fermentum elit, nec imperdiet purus nisi non arcu. Suspendisse placerat, est nec feugiat tincidunt, ex risus dictum ante, id dignissim lectus lorem vitae odio. Fusce ac quam ut arcu porta ultrices. Quisque porta volutpat tortor, ut ultricies velit viverra eget.

Nulla facilisi. Donec cursus sodales est, quis eleifend urna congue sit amet. Nunc accumsan, mi sed dapibus convallis, nunc felis auctor libero, a maximus nulla mauris ut enim. Vivamus vulputate ipsum odio, vel tincidunt leo aliquam in. In varius urna quis purus iaculis aliquam. Donec egestas lectus vitae justo dapibus, et pharetra risus lobortis. Nullam pulvinar justo ut purus efficitur ornare. Proin rutrum dictum purus eget vehicula.

Proin ut ipsum ante. Nulla porttitor eros at metus pretium viverra. Vestibulum ut elit at dui accumsan convallis ac vel mauris. Cras venenatis ante ac arcu volutpat vehicula. Nullam id tortor vitae augue interdum ultricies. Cras pharetra elit sed viverra scelerisque. Aenean cursus nulla ut arcu posuere hendrerit. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Integer ipsum libero, tempor et ornare in, facilisis vel ex. Integer eget leo facilisis erat euismod rhoncus. Quisque sapien quam, dictum vitae arcu id, posuere aliquet lacus. Phasellus porttitor tortor metus, eu vehicula eros condimentum et. Praesent odio justo, sagittis id vestibulum sed, posuere sit amet neque.

Integer et ornare orci. Aliquam eget ligula et tortor finibus condimentum. Quisque at tempor risus. Vivamus fringilla lobortis eros, nec eleifend sapien lacinia eu. Mauris interdum pulvinar nisl vel aliquam. In eget nibh id risus aliquam efficitur. Vestibulum gravida ante quis tortor placerat dapibus. Aliquam dignissim vestibulum tempor. Integer nec porta sem, et egestas libero. Nam vel nisl quis nulla volutpat pretium. Aliquam luctus laoreet imperdiet.

II. THE ORIGINAL REQUIREMENTS DOCUMENT

This is the Requirements Document we wrote at the beginning of the project. It has been re-formatted to keep consistent styling across the document, but the content of the document has not been modified. We have also indexed the requirements in this document for easy reference. The original document in its original formatting is included on the usb storage drive.

A. Introduction

1) *About This Document:* This is the requirements document for the Silicon Tracker project. Included in the following sections are lists of user stories organized by categories. Each paragraph contains a feature to be added to the web application. They are all organized into their respective categories and are set up as individual tasks to be checked off as they are completed.

2) *Definitions:* The kiosk mentioned in a few user stories is a nickname that the Intel PMA Labs has given for their check in/check out system. They have a computer with a QR scanner that sits on top of a large locker with all the items they store.

Although one of the main items being tracked will be silicon, we use the generic term item to specify that any part needed, not just silicon, can be inventoried by our system.

3) *Current System:* The current system Intel uses is inadequate for their needs. The items have to be checked in or out one at a time. This means the user has to enter their credentials for every item that goes through the system. They are wanting a more user-friendly web application where they can add items to their shopping cart and when they are ready, go to the kiosk and check in and out all their items in a single transaction.

B. The Client would like to...

1) Inventory:

Req 1: Queue an item that the user wants but is not checked in. If a person needs this item and it is currently checked out, they can be put on a queue for that item.

Req 2: Give a reason and a priority when queueing an item. This will create a priority queue for the users. If someone needs an item and it is important, they can give a reason why.

Req 3: When an item gets checked in and there were people waitlisting on the item, the people who have waitlisted will be notified with everyone's priority and reason for needing the item.

Req 4: Check out an item that is checked into the system. If nobody else has the item reserved, they can check out the item immediately.

Req 5: Check in an item that was checked out of the system. The user returns the item so other people can check out the item.

Req 6: Check in multiple items in a single transaction. The current system doesn't allow this feature. This would let the user check in multiple items without giving their credentials for every item.

Req 7: Check out multiple items in a single transaction. The current system doesn't allow this feature. This would let the user check out multiple items without giving their credentials for every item.

Req 8: Check in and check out multiple items in the same transaction. Along with the above two stories, the user should be able to both check in and out and only give their credentials once.

Req 9: Enter new items into the system. There should be a form that a user can fill out. They will scan the item to insert the item's ID.

Req 10: Retire old items in the system. They need to logically delete (scrap) items when an item is no longer being used. There should be a textbox to optionally give a reason for the item being retired.

Req 11: Be able to scan an item when checking in and have that automatically update the database. The user should just be able to log in, and then scan the item. The status of the item should be updated based on the item being scanned. This should happen when the user has the item currently checked out.

Req 12: Be able to scan an item when checking out have that automatically update the database. The user should just be able to log in, and then scan the item. The status of the item should be updated based on the item being scanned. This should happen when the user has the item.

Req 13: Be able to scan an item that is currently checked in to see who has the item reserved next. This will show the next person in line for the item.

Req 14: Be able to scan an item that is currently checked out to see who has the item checked out. For example, if there is an item that someone left out and nobody knows who has it checked out, the item can be scanned to see who it currently belongs to.

Workflow of the system:

Req 15: The initial screen is a welcome screen. This is where the user can scan their badge, use facial recognition, or enter a username/password to login. After proper authorization, the user is presented with the shopping cart interface.

Req 16: The user can now start scanning items.

Req 17: If the item is scanned and the item is checked in, it will now be checked out by the user. If the item was already checked out, information about who currently has the item checked out will be displayed on the screen.

Req 18: If the current user has an item checked out, they can scan the item and the item will be checked in.

Req 19: When the user is done, they will press a confirm button and will be shown a list of items they have checked in/out. The user can then proceed to commit their transaction to the database.

2) Security:

Req 20: Log into the application with a username and password.

Req 21: User settings and permissions will be maintained in the project database. Authenticate user credentials using the Active Directory server already set up at Intel. They already have a system in place and the users will have to sign in with their current credentials.

Req 22: Authenticate facial recognition using the project database. The data stored for facial recognition will have to be in the project database, since we cannot modify the Active Directory server.

Req 23: When setting up facial recognition for a user, authenticate user with username and password, or with RFID tag first. This is for security reasons, so other people won't try to scan their face on someone else's profile. When checking in or out item(s), the user should be able to authenticate with their current RFID tags using an RFID reader.

Req 24: When checking in or out item(s), the user should be able to authenticate with facial recognition.

Req 25: When checking in or out item(s), the user should be able to authenticate with username and password.

Req 26: Those with elevated permissions can retire old items from the system. This will be a logical delete, not an actual delete from the database.

Req 27: Those with elevated permissions can modify certain fields of the items.

Req 28: Have any communication between the web server and the client will be encrypted through SSL. The information about the items are all confidential, and they need to stay that way.

Req 29: Admins will have the ability to enable/disable any of the three authentication methods (username/password, RFID tag, or facial recognition). If for whatever reason the admin does not want one of these methods available, they can disable that method of authentication.

3) Database:

Req 30: Have new items be inserted into the project database.

Req 31: Have old items continue to be stored in the database after they are retired (logically deleted). Certain users still need the information about items even if they are no longer active.

Req 32: Have a record of who checked in which items, and have that stored in the database. This creates an audit trail for all the items.

Req 33: Have a record of who checked out which items, and have that stored in the database. This creates an audit trail for all the items.

Req 34: Have a timestamp for every time an item is inserted, updated, or retired. This is also for auditing reasons.

4) Services:

Req 35: Have an e-mail sent out when an item is queued to any person that has the specific item checked out.

Req 36: Send out periodic schedulable reports via e-mail to each user based on the items they checked in and out, and have on queue.

Req 37: Have users be notified via e-mail when they check in an item. This should happen for every transaction, so the user doesn't get a separate e-mail for every item.

Req 38: Have users be notified via e-mail when they check out an item. This should happen for every transaction, so the user doesn't get a separate e-mail for every item.

Req 39: Have users be notified when an item is now considered retired and they have the item currently checked out. They will need the e-mail so they know to return it immediately.

C. The Interfaces...

1) Main Interface:

Req 40: Will be a touch interface without need of keyboard. The keyboard will still be available for certain functions, such as entering a new item. The idea is that the user doesn't have to use the keyboard for check-in and check-out transactions.

Req 41: Will be able to check in items from this interface. This should not require a keyboard. The user should simply login to the system and scan the item(s) to check them in.

Req 42: Will be able to check out items from this interface. This should not require a keyboard. If they have the item ready for check out, then they should simply login to the system and scan the item(s) to check them out.

Req 43: Will be able to scan an item and display the information about that item (i.e. in the case of silicon, Core Count, LLC Size, Frequency, Status (Checked in or Checked out), etc.)

2) Shopping Cart Interface:

Req 44: Will be able to check in items from this interface. If they are at the kiosk they should have a way to check in items using this interface.

Req 45: Will be able to check out items from this interface.

Req 46: Will be able to add new items to the system from this interface. When a new item comes in, they will need to enter information about the item, and then scan it so they have the serial number recorded.

Req 47: Will be able to retire items from circulation (with elevated privileges). The items will be scrapped, but still in the database for record keeping.

Req 48: Will be able to reuse information of items when the same item is being added to the inventory system. They will have multiples of the same item. That item should just be scanned and linked with the other items in the system.

Req 49: Will be able to search for items based on certain fields in the database. The user should be able to filter their search results based on the item attributes.

Req 50: Will have multi-level filtering on searches. For more information, see Appendix Multi-Level Filtering.

3) Scrapping Interface:

Req 51: If a user has the right elevated privileges, they can be taken to a scrapping interface. This interface will be similar to the shopping cart interface. The difference will lie in the fact that any item the user scans will be scrapped instead of checked in/out. They will confirm and commit their transaction, just like the shopping cart interface.

4) Reporting Interface:

Req 52: Will be able to view any item in circulation. This should be in the form of a user-friendly table with pagination.

Req 53: Will be able to view any item that is retired. This should also be in the form of a user-friendly table with pagination.

Req 54: Will be able to see what items a certain user has checked out. They should be able to click on a user, and it will give a list of items in a user-friendly table with pagination.

Req 55: Will be able to search for any user who checked out a certain item. This will display as a table with the users who have one or more of these items.

Req 56: Will have multi-level filtering on searches. For more information, see Appendix A Multi-Level Filtering.

D. The Expo Should Look Like...

1) Web Application:

Req 57: A Fully functioning version will be running during the event. This will be a system that the general audience can view and play with.

Req 58: Anybody can have their faces entered into the database. This will show how the facial recognition works.

Req 59: Anybody can scan sample RFID cards to check in or out items. We will have sample items for them to scan, and they can use fake RFID badges to simulate the experience.

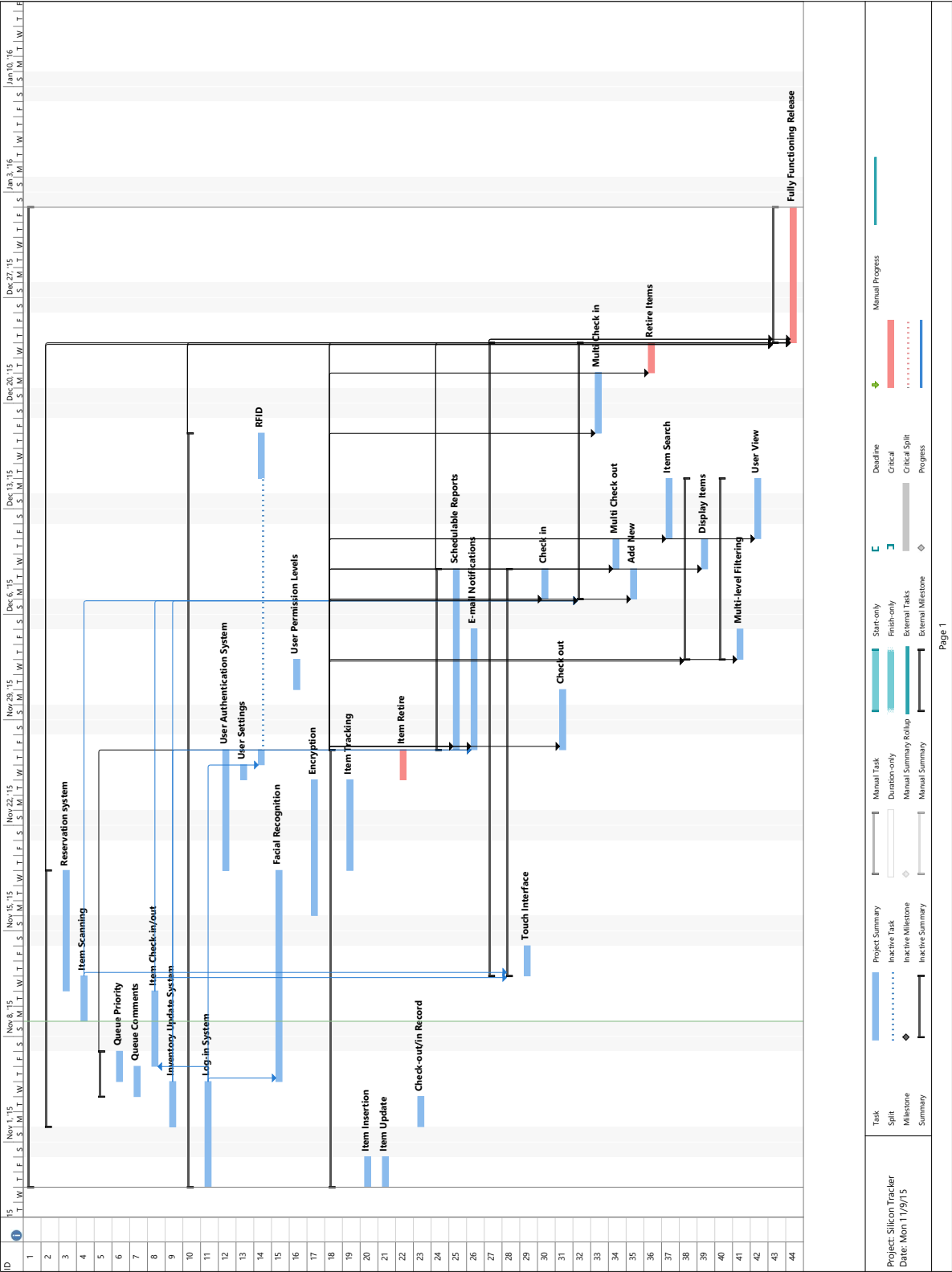
Req 60: Anybody can scroll through the different interfaces. They will have admin privileges so they can see all the features of the application.

Req 61: Anybody can check in or out items into the system. There will be a set of sample items people can scan so they can see how the system works.

E. Appendix

1) Multi-Level Filtering: This is a filtering process for items that are in the database system. It is used for easy search filtering based on multiple fields. For example, if the user is filtering items based on clock speed and they want to narrow their search further, then they can select another field to filter on. If they decide to filter again on cache size, then they will only see the possible filters for cache size based on the items with the specific clock speed they chose earlier.

2) Gantt Chart:



III. PROJECT CHANGES SINCE THE ORIGINAL CLIENT REQUIREMENTS DOCUMENT

#	Requirement	Reason For Changes	Comments
2	Give a reason and a priority when queueing an item. This will create a priority queue for the users. If someone needs an item and it is important, they can give a reason why.	Our client stated that since they are a small team, they only needed an email stating that the item has been checked in. All the people can meet up and talk about who gets the item next.	
10	Retire old items in the system. They need to logically delete (scrap) items when an item is no longer being used. There should be a textbox to optionally give a reason for the item being retired.	There is no textbox to give a reason for the item being scrapped. Each item has a notes field, and we decided that the best option was to place the reason for scrapping in the items notes field.	
13	Be able to scan an item that is currently checked in to see who has the item reserved next. This will show the next person in line for the item.	We decided not to be concerned about this, because everyone who reserved an item will discuss in person who needs the item next.	
17	If the item is scanned and the item is checked in, it will now be checked out by the user. If the item was already checked out, information about who currently has the item checked out will be displayed on the screen.	Our client asked if we could make it so one person can check in anothers item.	
19	When the user is done, they will press a confirm button and will be shown a list of items they have checked in/out. The user can then proceed to commit their transaction to the database.	We made it so they only need to press one button to commit their transaction. A message appears now to just show what items they checked in/out, after they have committed.	
29	Admins will have the ability to enable/disable any of the three authentication methods (username/password, RFID tag, or facial recognition). If for whatever reason the admin does not want one of these methods available, they can disable that method of authentication.	Our clients team implemented RFID security, and the facial recognition was implemented very late in the development phase of the project, so we didnt get a chance to implement this setting.	
N/A	The Interfaces	In our original Requirements Document, we had four interfaces. After starting work on this project, we simplified the system down to two interfaces. It made the system easier to discuss and navigate. We boiled down the interfaces into the Web interface and the Kiosk interface. The Shopping Cart interface, Scrapping interface, and Reporting interface have mostly been combined into the Web interface. The Main Interface has mostly been reworked into the Kiosk interface.	All of the requirements within the Interfaces section have been fulfilled, but not necessarily in all of the interfaces specified.

44 & 45	Will be able to check in items from this interface. If they are at the kiosk they should have a way to check in items using this interface. Will be able to check out items from this interface.	Our client let us know that they only need to check in and check out items from one interface. Our original Requirements Document had checking in and out of items from two interfaces. We removed the Checking in and out requirements to allow for these features to happen in only one place.	
59	Anybody can scan sample RFID cards to check in or out items. We will have sample items for them to scan, and they can use fake RFID badges to simulate the experience.	Our clients team implemented the RFID readers, so we did not show how they worked at expo.	
N/A	New Requirement: Sanitizing and Validation User Input	We did not place in our original document to clean up any user input and make sure that their input is valid. It was an important feature, and our client did ask that we created a system where data entered would be consistent.	
N/A	New Requirement: Editing Dropdown Menu Items	We setup a page for the admins to change the dropdown menu items. These dropdowns are found when inserting or editing an item. Our client liked the idea of not having to make edits directly in the database, so we implemented this feature.	
N/A	New Requirement: Quick Check in/out.	We added an extra feature for when a user scans an item to get the item's information, they can click a Check In/Out button to save the item for later. It makes it easier for the user to check in or out an item as soon as they scan it, and makes the workflow a little smoother.	Our client's team received this feature very well and were thankful to have it.

IV. DESCRIBE ANY PROBLEMS THAT HAVE IMPEDED YOUR PROGRESS, WITH ANY SOLUTIONS YOU HAVE

We have talked about a few of our blockers in the previous sections. For instance there is the issue of facial recognition. The other main issue is with authentication. These two main issues come from the security concerns and policies at Intel, and hardware issues that we are currently dealing with.

There is not really a way for us to get access to how Intel authenticates its users. So the best that we can do with authentication is come up with our own version of authentication and try to make it as close to how we believe their system works. As mentioned earlier, we have created such a system that handles authentication. It's a server that accepts web requests, and returns a WWID upon successful authentication. The only real information that we know about their authentication system is that our application will need to interact with their authentication server through a web request. Our service does exactly this. We have been as clear as possible in our code, because the employees at Intel will have to modify this portion of our code in order to make it work on their system. We thoroughly commented our code, and I have written to our client a detailed explanation of how we designed our authentication.

A smaller problem that we have been faced with is that we only have sample data for CPUs. All the other items behave almost identically. They will have a barcode to be scanned, and will check in and check out through the same methods as

the CPU. This is not a big issue, we just can't test any other items besides CPU, unless we take a CPU and put it in the system as another item.

Whenever we need to use the barcode scanner, we have to go to the capstone lab on campus. This was a little frustrating, especially when going to the capstone lab was not an option. To remedy this, we came up with a program that simulates our barcode scanner. It's a small python script that behaves in the same way as our scanner when you give it a serial number. This has allowed us to develop features for the barcode scanner without even using the scanner!

V. INCLUDE ANY PARTICULARLY INTERESTING PIECES OF CODE

This is an example of some of our back-end code. We are using express.js for our back-end. Express is a package of node.js, a lightweight, asynchronous, open-source runtime environment in JavaScript. This example is showing when a user makes a POST request to the server, when the user adds a new CPU item. The server responds anytime there is a POST request to the URI `"/add/cpu"`. The server first sanitizes the input with our scrub library, and then checks user input with the validation library. If the validator returns with errors, the server sends the error messages back to the web interface. Only when the validator approves of the input does the server make a database call. With valid input, the server makes a call to the database for a stored procedure, which will insert the CPU information into the database.

```

1  app.post('/add/cpu', function(req, res) {
2    req.body = scrub.CPU(req.body);
3    var errors = validate.CPU(req.body);
4
5    if (errors) {
6      res.status(400).send(errors);
7    } else {
8      pool.getConnection(function(err, conn) {
9        conn.query("CALL check_serial_cpu('"+req.body.serial_num+"');",
10         function(error, results, fields){
11           if(error) {
12             throw error;
13           }
14
15           if(results[0].length == 0) {
16             conn.query("CALL put_cpu('"+req.body.serial_num+"','"+
17               req.body.spec+"','"+req.body.mm+"','"+
18               req.body.frequency+"','"+req.body.stepping+"','"+
19               req.body.llc+"','"+req.body.cores+"','"+
20               req.body.codename+"','"+req.body.cpu_class+"','"+
21               req.body.external_name+"','"+req.body.architecture+"','"+
22               req.body.notes+"');",
23             function(error, results, fields){
24               if(error) {
25                 throw error;
26               }
27             });
28           }
29           conn.release();
30           res.status(200).send(req.body);
31         });
32       });
33     }
34   });

```

Listing 1: Server-side Request Handler

The following code below shows how we implemented the barcode scanner. This code runs on the kiosk interface in the login page. It is continuously listening for keyboard input. The way the barcode scanner works is it acts like a keyboard. When it scans an item, it "types" in the letters of the serial number one at a time. We came up with an idea that if the barcode types faster than what is humanly possible, we should be able to decipher between a barcode scan and a human typing. Once some keyboard input comes in, our code starts a timer. The code will push all the keys pressed into an array. If there are ten key presses within 250 milliseconds, we will know that it is a barcode scan instead of a human typing! Once an item is scanned, it requests from the server information about the item just scanned. Once information is returned to the client, it displays the item information on the screen.

```

1  $(window).keypress(function(e) {
2    chars.push(String.fromCharCode(e.which));
3    if (pressed == false) {
4      setTimeout(function() {
5        // If there are ten characters inserted before the timeout
6        if (chars.length >= 10) {
7          var barcode = chars.join("");
8          // There was an item scanned. Enter response code here.
9          var itemInfo = "";
10         $.get('/serial/'+barcode, function(data) {
11           if (data.item_type === 'cpu') {$('#item-info').html(CPUInfo(data));}
12           else if (data.item_type === 'ssd') {$('#item-info').html(SSDInfo(data));}
13           else if (data.item_type === 'memory') {$('#item-info').html(MemoryInfo(data));}
14           else if (data.item_type === 'flash_drive') {$('#item-info').html(FlashDriveInfo(data));}
15           else if (data.item_type === 'board') {$('#item-info').html(BoardInfo(data));}

```

```
16         else {$('#item-info').html(ErrorInfo());}  
17     });  
18 }  
19     chars = [];  
20     pressed = false;  
21     },250); // <-- this is the timeout in milliseconds  
22 }  
23     pressed = true;  
24 });
```

Listing 2: Barcode Scanner Listener

VI. USER STUDIES

We continually talk with our client about the project and how it is working out for them. Our highest priority is to make sure that they are happy with the work we produce.

Since our code is on our repository on-line, they have access to our project in its current state. They have two instances of an inventory system right now. One instance is their old inventory system. The other instance is our inventory system in its current state. Our client has asked his team to use both inventory systems when checking in and out items. This gives us a lot of feedback on how our system is working in the intended environment, and we are taking all of their feedback into consideration.

A lot of the feedback we get pertains to bug fixes. Some of the bugs are specific to their setup, and we would have never known about if they weren't actively using it. For instance, with their system they have their server and database running on separate machines. If the server has not made any requests in a specified amount of time, the database will close the connection with the server. They would have to restart the server in order to get the connection back with the database. On our development environment, we have the server and database running on the same machine, so there is never a connection issue between the two.

We are constantly receiving positive feedback from our client and his team. This has allowed us to make the best possible system that we can, and serve their inventory needs.

VII. CONCLUSION

Overall we are very close to having a final release on our project and will be ready for expo. We have been talking to our client quite frequently and he always has positive feedback on our progress. He seems to be happy with the progress we have made.

We have implemented all the features in the project, and are now mainly working on bug fixes for the next few weeks. We intend to work very hard up until expo to make sure that we can hand over a product to our client that we can be proud of. This has been a tough year, but we have all learned so much from this project and feel very lucky to have such a great client.

VIII. INCLUDE IMAGES OF YOUR PROJECT













CPU SSD Memory Flash Drives Boards											
Show 50 entries											
Serial Number	Spec	MM	Freq	Step	LLC	Cores	Codename	CPU Class	External Name	Architecture	
Serial Number	Spec	MM	Freq	Step	LLC	Cores	Codename	CPU Class	External Name	Architecture	
35127210J0084	SR1XV	935852	2.2	M1	30	12	Haswell Server	EP	Xeon E5-2658 v3	Haswell	
2V127163B0110	SR195	929620	1.8	C0	6	4	Crystal Well	Mobile	Core i7-4860EQ	Haswell	
35127210J0092	SR0KQ	919841	2	C2	20	8	Jaketown	EP	Xeon E5-2650	Sandy Bridge	
2V127163B0675	SR2F1	944341	2.6	D1	4	2	Skylake	ULT	Core i7-6600U	Skylake	
35125272R0021	SR2E8	944076	2.7	G1	6	4	Broadwell	Mobile	Core i7-5850EQ	Broadwell	
2V127163B0804	SR1W5	934898	1.58	C0	1	2	Bay Trail	Atom	Celeron N2807	Silvermont	
35127210J0029	SR2E9	944077	1.8	G1	6	4	Broadwell	Mobile	Xeon E3-1258L v4	Broadwell	
35127210J0062	SR268	939656	1.8	F0	3	2	Broadwell	ULT	Core i5-5350U	Broadwell	
2V127163B0872	SR268	939656	1.8	F0	3	2	Broadwell	ULT	Core i5-5350U	Broadwell	
2V127163B0686	SR2F7	944075	2	G1	6	4	Broadwell	Mobile	Xeon E3-1278L v4	Broadwell	

Fig. 1: The CPU table.

CPU SSD Memory Flash Drives Boards											
Show 50 entries											
Serial Number	Spec	MM	Freq	Step	LLC	Cores	Codename	CPU Class	External Name	Architecture	
Serial Number	Spec	MM	Freq	Step	LLC	Cores	Codename	CPU Class	External Name	Architecture	
2V127163B0110	SR195	929620	1.8	C0	6	4	Crystal Well	Mobile	Core i7-4860EQ	Haswell	
35125272R0021	SR2E8	944076	2.7	G1	6	4	Broadwell	Mobile	Core i7-5850EQ	Broadwell	

Showing 1 to 2 of 2 entries (filtered from 13 total entries)

Previous 1 Next

Fig. 2: Filtering table based on number of cores and external name.

Silicon Tracker Home Add Item Login

Serial Number *
Frequency *
LLC (MB) *
Codename *
External Name
Notes

Spec *
MM *
Stepping *
Cores *
Class *
Architecture *

* Indicates required field

Fig. 3: The screen for adding a new CPU.

Silicon Tracker

Check Out

Serial Number
2V127163B0110
35127210J0084
35127210J0092

Showing 1 to 3 of 3 entries

Check In

Serial Number
No data available in table

Showing 0 to 0 of 0 entries

Submit

Fig. 4: The kiosk cart screen, with some items to check out.