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 dont 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

A. *Who Requested it?*

This project was requested by our client Scott, along with his team at Intel.

B. *Why Was It Requested?*

Scotts team needs to keep inventory of all of their CPU silicon in their lab. These silicon pieces have no real identifiable marks other than a serial number. The members of Scotts lab needed an easy way to retrieve information about the silicon. The other reason for this inventory system is to know who has which silicon. Each piece of silicon is considered very valuable to Intel, so they need a way to know who has taken silicon in order to do testing. They also want to know who has which silicon for their own teams sake. They like to know if a certain piece of silicon is being used by another team member, and to know as soon as their teammate is done with it.

C. *What Is Its Importance?*

It is important for the team members to have an easy-to-use system that helps them do their work more efficiently than they could without it. It is meant to save them time, and get to the important parts of their work. This system is also important in preventing the loss of silicon, which could be very costly.

D. *Who Was/Were Your Client(s)?*

Scott Oehrlein

E. *Who Are the Members of Your Team?*

Brett Hayes, Joseph Cronise, and Dylan Camus

F. *What Were Their Roles?*

None of the members took on a specific role for the project. There were many tasks to the project, so everyone had to take on multiple roles, with much overlap between roles. The description of the roles provided will be about which member was the primary caretaker for that part of the project.

Brett was responsible for the server-side implementation. He created the basic structure and maintained a lot of the server-side code. He was also responsible for a lot of the dynamic code running on the client. Brett also set-up a task list each term, to make sure the team was on track throughout the project.

Dylan was in charge of the database. He worked on the design structure, and created many of the stored procedures used in the current system. He was also responsible for the emailing system, which included sending immediate emails and scheduled emails.

Joseph came up with the original design of the website, and was responsible for the research involved in facial recognition.

G. *What Was the Role of the Client(s)? (I.e., Did They Supervise Only, or Did They Participate in Doing Development)*

None of the members took on a specific role for the project. There were many tasks to the project, so everyone had to take on multiple roles, with much overlap between roles. The description of the roles provided will be about which member was the primary caretaker for that part of the project.

Brett was responsible for the server-side implementation. He created the basic structure and maintained a lot of the server-side code. He was also responsible for a lot of the dynamic code running on the client. Brett also set-up a task list each term, to make sure the team was on track throughout the project.

Dylan was in charge of the database. He worked on the design structure, and created many of the stored procedures used in the current system. He was also responsible for the emailing system, which included sending immediate emails and scheduled emails.

Joseph came up with the original design of the website, and was responsible for the research involved in facial recognition.

II. The Original Requirements Document

This is the Requirements Document we wrote at the beginning of the project. It has been reformatted to keep consistent styling with this 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 doesnt 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 doesnt 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 items 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 wont try to scan their face on someone elses 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 doesnt 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 doesnt 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 doesnt 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.

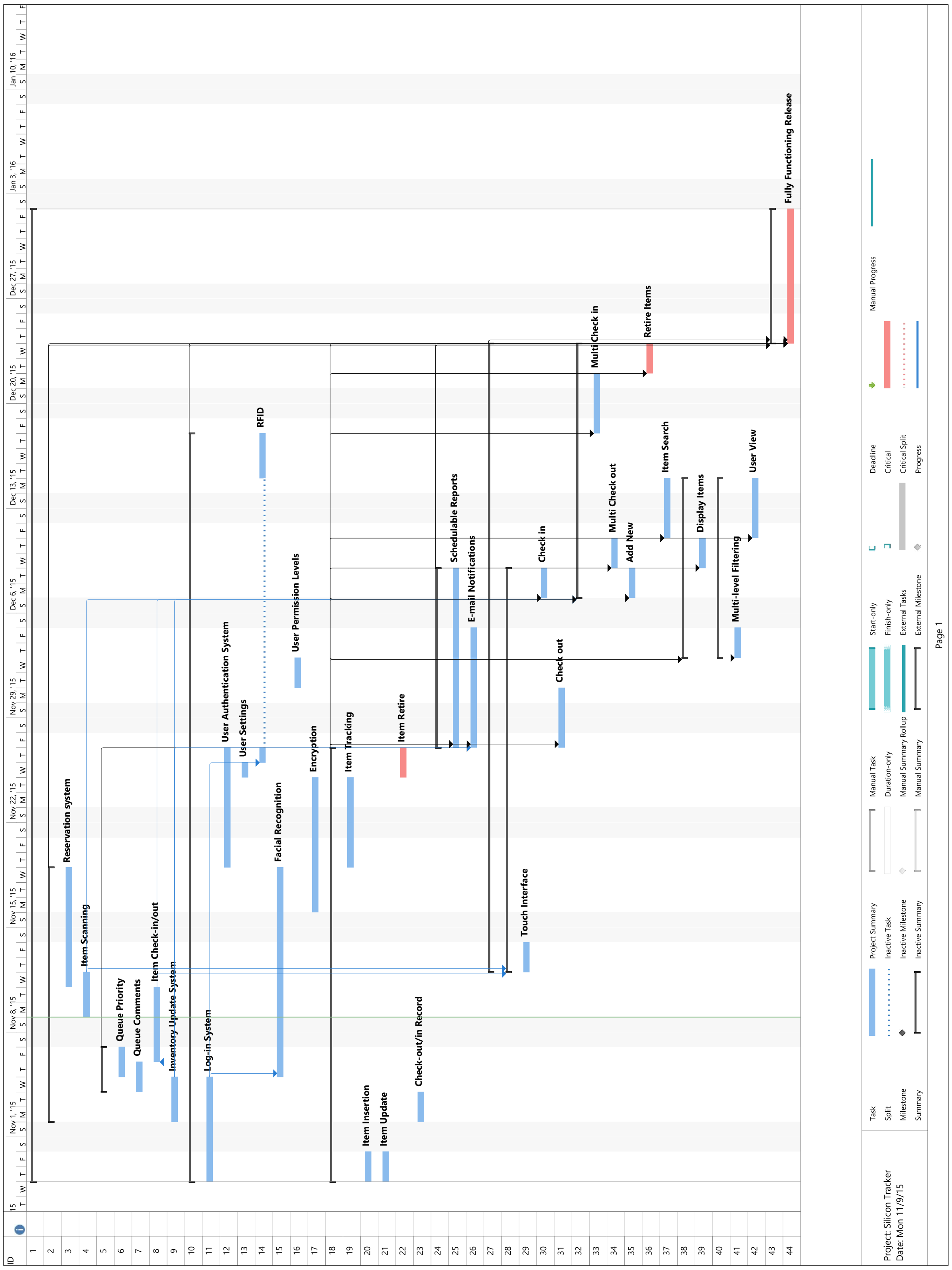
Req 60: Anybody can scroll through the different interfaces. They will have admin privileges so they can see all the features of the application.

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:



Reservation system
Item Scanning
Queue Priority
Queue Comments
Item Check-in/out
Inventory Update System
Log-in System
User Authentication System
User Settings
RFID
Facial Recognition
User Permission Levels
Encryption
Item Tracking
Item Insertion
Item Update
Item Retire
Check-out/in Record
Schedulable Reports
E-mail Notifications
Touch Interface
Check out
Check in
Multi Check in
Multi Check out
Add New
Retire Items
Item Search
Display Items
Multi-level Filtering
User View
Fully Functioning Release
Nov 1, '15
Nov 8, '15
Nov 15, '15
Nov 22, '15
Nov 29, '15
Dec 6, '15
Dec 13, '15
Dec 20, '15
Dec 27, '15
Jan 3, '16
Jan 10, '16
Project: Silicon Tracker
Date: Mon 11/9/15
Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Critical
Critical Split
Progress
Manual Progress
Page 1

III. 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. The Original Design Document

Below is the original Design Document written at the beginning of the project's lifespan, in its original formatting. It describes our course of action for designing the components, database design, and the workflow of the system. The original document is also included on our USB storage drive.

Design Document

Silicon Tracking Solutions

Client: Scott Oehrlein

Dylan Camus

Brett Hayes

Joseph Cronise

Revisions

Name	Date	Reason for change	Version
Brett, Dylan, Joseph	12/2/2015	Initial Doc	1.0

Table of Contents

Introduction

Purpose

This is the design document for the silicon tracking system. This document describes the concerns of the stakeholders and focuses on addressing those concerns by giving a detailed description of how the system will be built. These descriptions are essentially the blueprints of the system. They contain charts, graphs, function headers, and APIs to describe the system.

Scope

The Performance Measurement and Analysis team at Intel in Chandler, AZ desires a new inventory system. Their current system is inadequate for their needs. The Intel team wants a web-based system that can track the items they have as well as scrapped items.

Definitions

Active Directory – A domain controller that organized authorizations and authentications on a domain.

API – Application Program Interface. A series of tools that allow programmers to easily interact with a system.

JavaScript – A high-level language used for the web and other applications.

JSON – JavaScript Object Notation. A standard used for key-value pairs of data.

REST – Representational State Transfer. An interface for communicating over HTTP.

MySQl – A relational database management system.

ER Diagram – A chart that visually represents the relationship between database entities.

References

[1]N. Foundation, 'Node.js', Nodejs.org, 2015. [Online]. Available: http://nodejs.org. [Accessed: 02- Dec- 2015].

[2]A. Reinman, 'Nodemailer', Nodemailer, 2015. [Online]. Available: http://nodemailer.com/. [Accessed: 02- Dec- 2015].

[3]M. Patenaude, 'node-schedule/node-schedule', GitHub, 2015. [Online]. Available: https://github.com/node-schedule/node-schedule. [Accessed: 02- Dec- 2015].

[4] Docs.oracle.com, 'C API', 2015. [Online]. Available: https://docs.oracle.com/cd/A87860_01/doc/network.817/a86082/oidsdk.htm. [Accessed: 03- Dec- 2015].

[5] Software.intel.com, 'Intel® RealSense™ SDK 2015 R5 Documentation', 2015. [Online]. Available: https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?doc_devguide_introduction.html. [Accessed: 03- Dec- 2015].

Overview

The context of this document begins with a list of concerns from the stakeholders organized by category. The architecture is then described after the concerns. Listed is the high-level overview of the system, followed by a more detailed description of each component. Finally, the justification of our design is given.

Design Concerns

Authentication:

- How will users be authenticated?

Server:

- How will all data be secure?
- How will reporting be handled?

Scanning:

- How will items be identified by the system?

Interface:

- What will the design of the interface look like?
- How will users interact with the interface?

Database:

- What will be the design for the database?
- How will items be retrieved or inserted into the database?

General:

- Will the system be scalable?

High Level System Architecture

Concerns:

- Will the system be scalable?

We have separated this project into multiple components. There is the Interface, Server, Database, Authentication, and Scanner. Having every major section in its own separate component creates a separation of concerns which makes the system scalable. Below are the different major components of the system. Each component will be discussed in more detail in the following section.

- Interface – The front-end component where user interaction with the system occurs.
- Server – The back-end component where the logic of the system occurs.
- Database – Where all persistent data is stored.
- Authentication – Where the user is authorized by different means and credentials checked against the Active Directory server.

- Scanner – Where the items interact with the system.

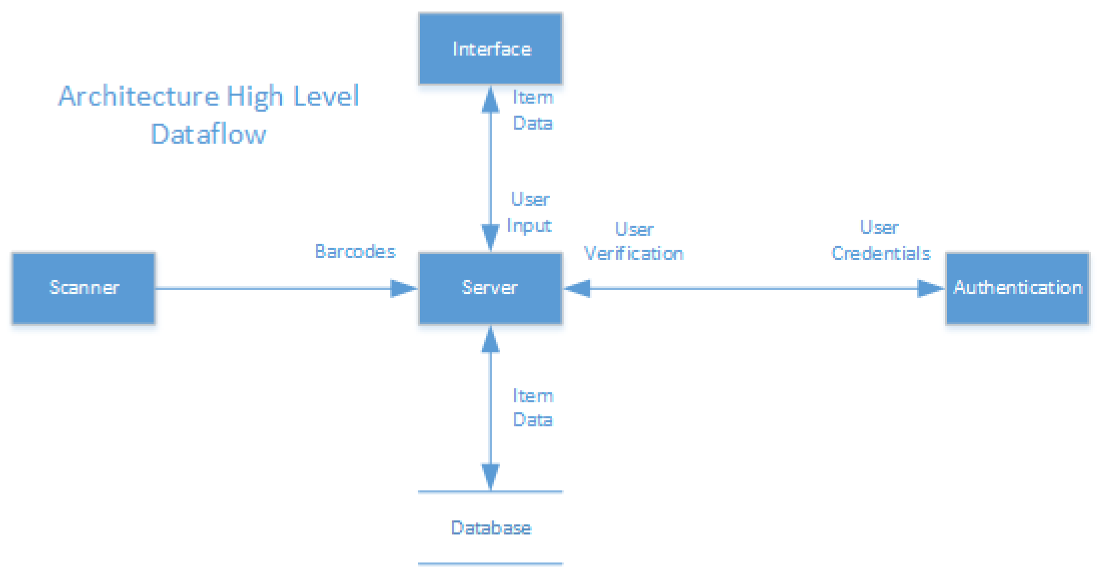


Figure 1: High-Level Dataflow of the System Architecture

Detailed Description of Components

Interface

Web Application

Design Entity Name: Web Interface
Type: Interface
Purpose:
The purpose of the web interface is to allow the user to interact with the database.

Concerns:

- What will the design of the interface look like?
- How will users interact with the interface?

The Web application will be HTML based and the user will be able to interact with it using any standard input method, but it will be designed with a focus on touch screen usage. Users will arrive at the initial login page, where they will select their preferred login method, they will then be redirected to that methods page and will be able to login t the system from there.

Once logged in users will default to the shopping cart interface, and from there they can either use that system, or click one of the tabs to move to a different page. Other available pages will be the user settings page, where users can register their faces for use with the facial recognition login method. The

Reporting page, where users can view, sort, and if they have the correct privileges, make changes to the database. And, the item input page, where users can input new items into the database.

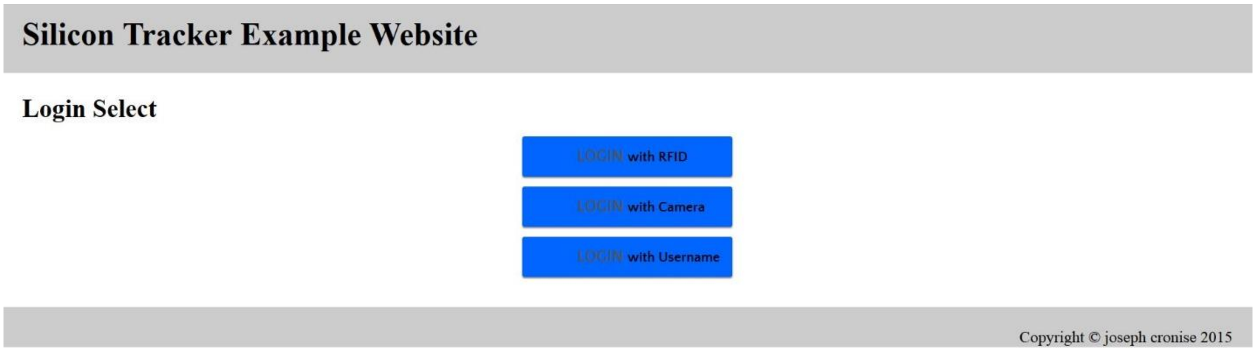


Figure 2: Users will be presented with the option to choose their login method from this page.

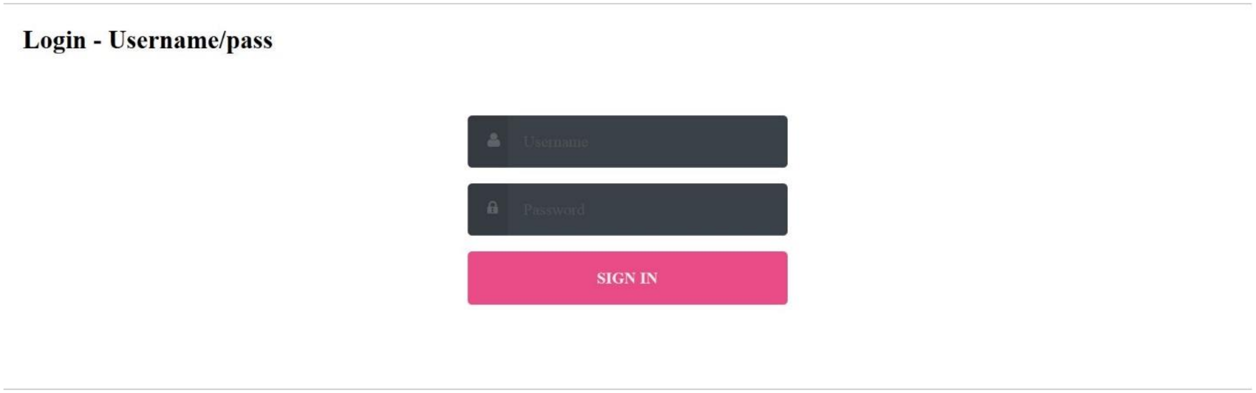
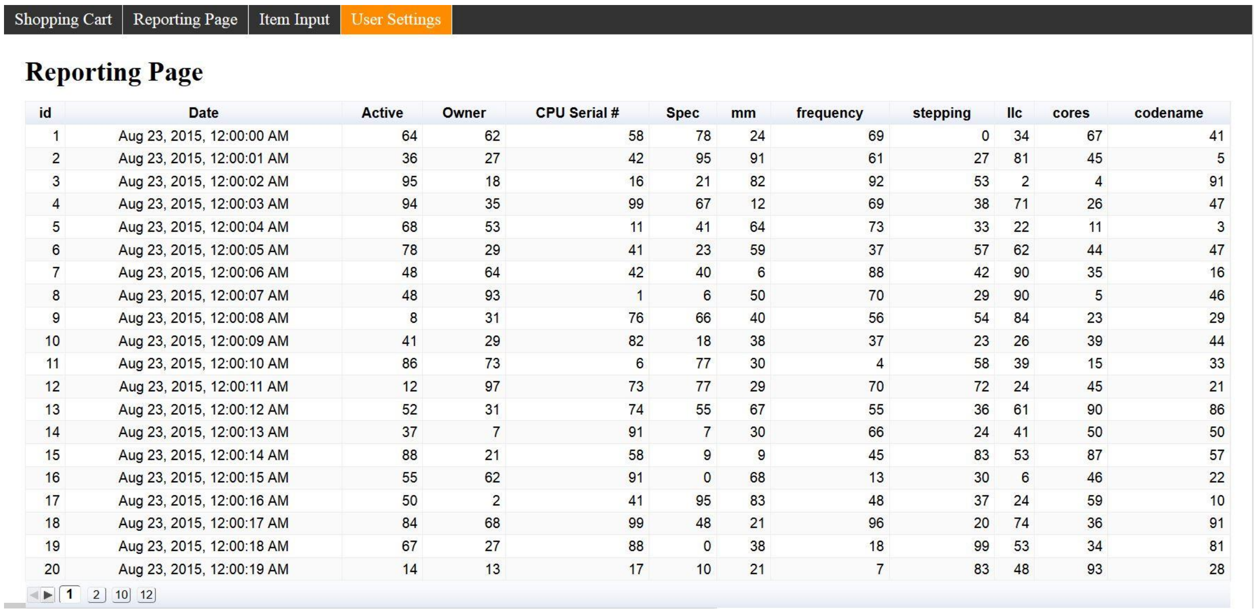


Figure 3: If users choose to login using their username/password they will be presented with this style of page.



Reporting Page

id	Date	Active	Owner	CPU Serial #	Spec	mm	frequency	stepping	llc	cores	codename
1	Aug 23, 2015, 12:00:00 AM	64	62	58	78	24	69	0	34	67	41
2	Aug 23, 2015, 12:00:01 AM	36	27	42	95	91	61	27	81	45	5
3	Aug 23, 2015, 12:00:02 AM	95	18	16	21	82	92	53	2	4	91
4	Aug 23, 2015, 12:00:03 AM	94	35	99	67	12	69	38	71	26	47
5	Aug 23, 2015, 12:00:04 AM	68	53	11	41	64	73	33	22	11	3
6	Aug 23, 2015, 12:00:05 AM	78	29	41	23	59	37	57	62	44	47
7	Aug 23, 2015, 12:00:06 AM	48	64	42	40	6	88	42	90	35	16
8	Aug 23, 2015, 12:00:07 AM	48	93	1	6	50	70	29	90	5	46
9	Aug 23, 2015, 12:00:08 AM	8	31	76	66	40	56	54	84	23	29
10	Aug 23, 2015, 12:00:09 AM	41	29	82	18	38	37	23	26	39	44
11	Aug 23, 2015, 12:00:10 AM	86	73	6	77	30	4	58	39	15	33
12	Aug 23, 2015, 12:00:11 AM	12	97	73	77	29	70	72	24	45	21
13	Aug 23, 2015, 12:00:12 AM	52	31	74	55	67	55	36	61	90	86
14	Aug 23, 2015, 12:00:13 AM	37	7	91	7	30	66	24	41	50	50
15	Aug 23, 2015, 12:00:14 AM	88	21	58	9	9	45	83	53	87	57
16	Aug 23, 2015, 12:00:15 AM	55	62	91	0	68	13	30	6	46	22
17	Aug 23, 2015, 12:00:16 AM	50	2	41	95	83	48	37	24	59	10
18	Aug 23, 2015, 12:00:17 AM	84	68	99	48	21	96	20	74	36	91
19	Aug 23, 2015, 12:00:18 AM	67	27	88	0	38	18	99	53	34	81
20	Aug 23, 2015, 12:00:19 AM	14	13	17	10	21	7	83	48	93	28

Figure 4: A sample Reporting page showing a possible output from the CPU table.

Server

Communication between Components

Concerns:

- How will items be retrieved or inserted into the database?
- How will users be authenticated?

The server is the central component to the system. All communication between components and outside systems will go through the server. In order to keep the system as loosely coupled as possible, there should be a minimal amount of entry points between components.

The high level architecture figure shows the data traveling between components. This section will go into greater detail on how these components will communicate through the server.

We have decided to use Node.js as our server system. Node.js is an event-driven system and is efficient at asynchronous calls. Node.js is also great at including packages created by other developers, such as a mailing agent, scheduler, etc.

Design Entity Name: Interface-Server Communication
Type: Events
Purpose:
The interface is how the user communicates with the system. This is the entry point for any user. This is also the entry point for human error input and possibly even malicious attacks on the system. The basic idea for connecting the interface with the server looks like the following:

- The user sends input data
- The server sends information from the database, and error messages if need be.

We will be implementing a RESTful API for the client and server to interact with each other. The pages will be as follows:

/ - The home page.

/shop - The shopping cart.

/scrap - The scrapping cart.

/reporting - The reporting page.

/settings - The user settings.

Many of the requests between the interface and server will be GET requests. This will be in the form of item information including data of the user who has checked out the item, or user settings.

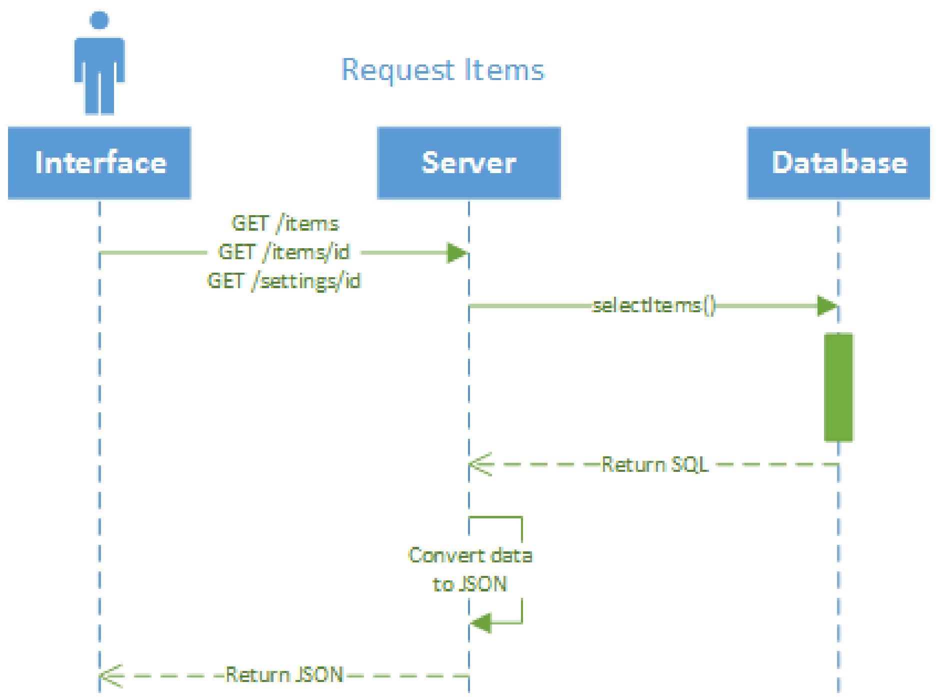


Figure 5: REST Sequence Diagram

Design Entity Name: Database-Server Communication
Type: Events
Purpose:
There will be two main types of requests from the server to the database. The first is the set of requests coming from the interface. The second are scheduled reports handled by the server.

There will be SQL scripts stored on the server. These scripts will be dynamic so that depending on user input the correct information will be returned from the database. For example, a user requests all items that have an L1 cache greater than 32K. The script would include this in the WHERE clause to filter the results returned.

These scripts will be called by a function and that function is in charge of interacting with the database. The function will send the script to the database and after the database sends the results back to the server, the function will convert the data to JSON and return the results.

Function Name: ExecuteQuery
Parameters:

- Script - The script to be executed

Returns: JSON – The results from the database in JSON format

Design Entity Name: Authentication-Server Communication
Type: Events
Purpose:

Whenever a user needs to use the system, they will have to go through authentication in order to login to the system. We will be implementing an API that allows the server to interact with the Active Directory server.
[Add reference to Dylan's work here]

Scheduled Reporting

Concerns:

- How will reporting be handled?
- How will all data be secure?

The server will be in charge of sending out scheduled reports to users as well as unscheduled reports based on triggers. Examples of scheduled reports would be: reminders of items checked out. Examples of unscheduled reports would be: receipts after checking in or out items; mass e-mail sent by admin.

Design Entity Name: E-mail Agent
Type: Resource
Purpose:
The server will need an effective way of sending e-mails to the users. This will provide a quick and efficient way of compiling and sending e-mails to the correct users.

We decided to use Nodemailer, a mailing agent that works with the node.js server. With Nodemailer, we will be able to send out e-mails securely through SSL. We will also be able to easily create custom e-mails with the right information for the users.

The mailing agent can be set up with the following options for the mailer connection data:

- options.port is the port to connect to (defaults to 25 or 465)
- options.host is the hostname or IP address to connect to (defaults to 'localhost')
- options.secure defines if the connection should use SSL (if true) or not (if false)
- options.auth defines authentication data (see authentication section below)
- options.name optional hostname of the client, used for identifying to the server
- options.localAddress is the local interface to bind to for network connections
- options.connectionTimeout how many milliseconds to wait for the connection to establish
- options.greetingTimeout how many milliseconds to wait for the greeting after connection is established
- options.socketTimeout how many milliseconds of inactivity to allow

There are more options available and are displayed in the Nodemailer documentation.

Design Entity Name: Scheduler
Type: Resource
Purpose:
The scheduler is used for sending out the scheduled e-mails to their respective users.

Like the mailing agent, we are using a package developed by an outside programmer in order to implement the scheduling agent. For scheduling, we will be using a package called node-schedule. This system makes it simple to schedule tasks. The main method we will use is a date and time scheduler. The method takes in a JavaScript Date object, and executes a task once that date has been reached. A scheduled event can also be cancelled at any time.

Database

Database Design

Concerns:

- What will be the design for the database?

Design Entity Name: Inventory Database
Type: Resource
Purpose:
The Inventory Database is used to track items within the lab.

We will using a MySQL database to track different items within the lab. The database will also keep a log of all item check outs/ins and will be used as part of the item reservation system where a user can reserve an already checked out item. The database will work with the server in order to send out reports as well as allowing manual checks of the data.

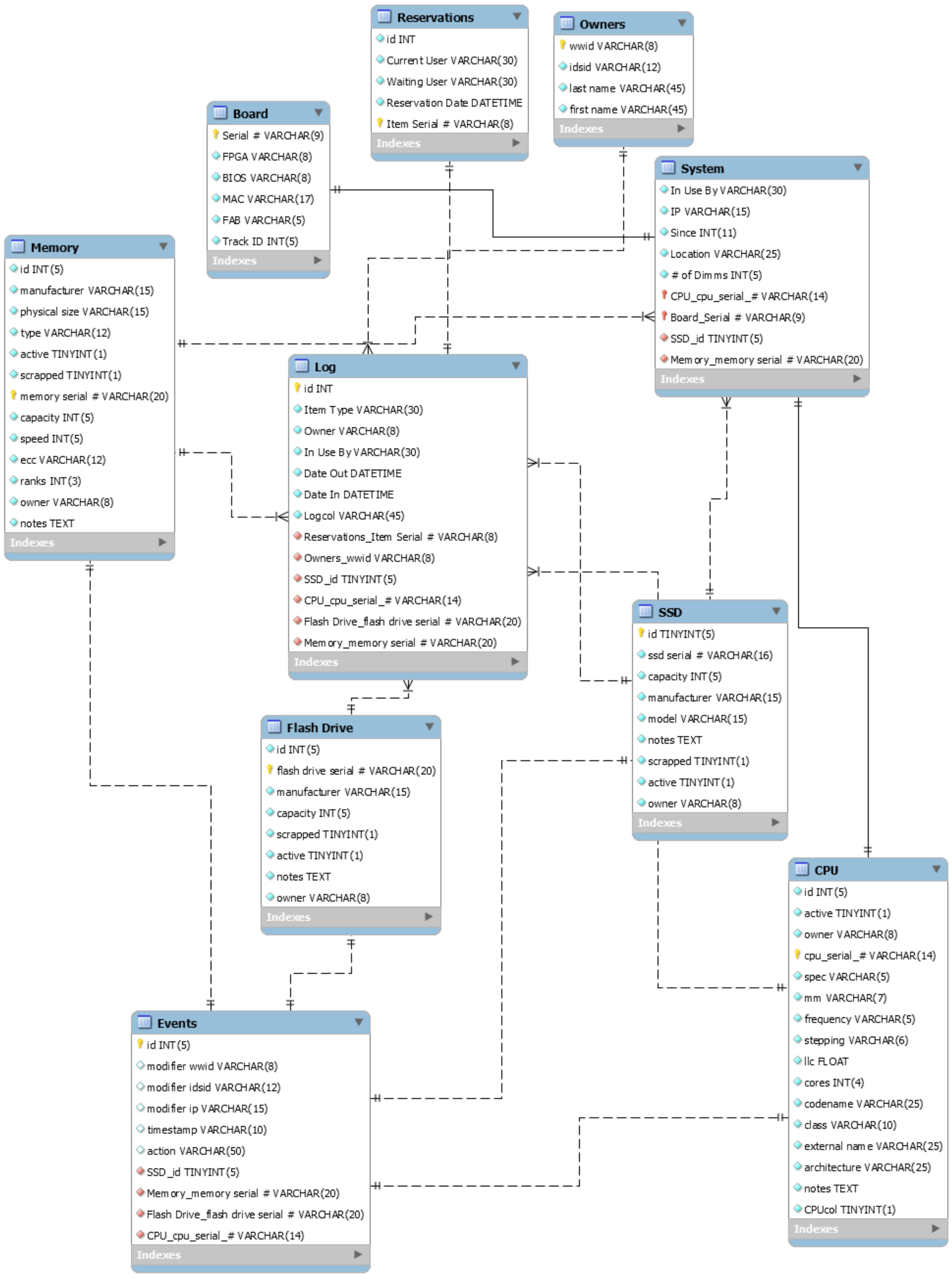


Figure 6: ER diagram detailing the specifics of the Inventory Database

Authentication

Active Directory Authentication

Concerns:

- How will users be authenticated?

One of the ways users will be authenticated is by entering in their security credentials or sliding their RFID and comparing with Intel's Active Directory. The Active Directory API allows an easy method of sending these credentials to the Active Directory securely with encryption.

Design Entity Name: Active Directory API
Type: API
Purpose:
The Active Directory API specifies how the system architecture interacts with the Active Directory Server. The Active Directory server contains security credential information that needs to be referenced with the information gathered from the system. The responsibilities of the Active Directory API are to take a number of parameters and return back a value that indicates either success or failure of authentication. The data sent to active directory must be encrypted with SSL. The machine on which the system is run on must contain a properly formatted certification authority certificate that matches the certification authority of the active directory.

The subcomponents of the Active Directory API are the RFID_authenticate function, which takes an RFID data field as a parameter and returns either success or failure; and a credential_authenticate function, which takes a username and password and returns either success or failure. The Active Directory API interacts with the system server and the Active Directory server. The system server listens for user input from the interface. Once input is received, the system server encrypts the data and sends it to the active directory server using the Authentication API. The Authentication API then returns either a success or failure to the server.

The Active Directory API requires both a keyboard for inputting security credentials and an RFID scanner. It also requires access to the Active Directory Server. The authentication API performs its task by first receiving both username and password in the case of credential_authenticate, or the RFID data in the case of RFID_authenticate. First, ldap_initialize is called to initialize a connection to the Active Directory Server. Then, ldap_simple_bind_s is called with username and password or the RFID data.

Facial Recognition

Concerns:

- How will users be authenticated?

Facial recognition will be a method of authentication separate from the active directory. After logging in, a user may choose to set up facial recognition. This allows users to quickly log in without needing to type in their user credentials or swipe their RFID badge.

Design Entity Name: Facial Recognition API

Type: API
Purpose:
The Facial Recognition API specifies how the system architecture interacts with the Intel RealSense Camera. The Intel RealSense Camera possesses facial recognition capabilities that, once properly initialized, will allow users to quickly authenticate themselves without having to use the keyboard to type in their security credentials. The responsibilities of the Facial Recognition API is to register users for facial recognition in the database, and to perform facial recognition using the images stored in the database. The API must have access to the Intel RealSense Camera API. Additionally, the API requires access to the database for storing images of registered users. Finally, users must have logged in using either their security credentials or RFID before being able to register as a new user for facial recognition.

The subcomponents of the Facial Recognition API are the recognition_init function, which initializes the RealSense Camera for Facial Recognition and specifies where to store the image files created when registering new users; the recognition_register function, which registers a new user for facial recognition by taking a picture of the user's face and storing it in the database; and the recognition_authenticate function, which compares the users face with facial images stored in the database.

The Facial Recognition API interacts with the system server and the database. The API is called by the server when a user wishes to register as a new user or authenticate him or herself using facial recognition. The API then interacts with the database by either writing facial images or reading facial images from the database.

The only outside resources required by the Facial Recognition API is the Intel RealSense Camera itself and the database. The database must have generous storage capabilities to store multiple JPEG files per user. The Facial Recognition API performs its task by making use of the Intel RealSense API for facial recognition. When initializing facial recognition using recognition_init, the RecognitionConfiguration interface from the Intel RealSense API is used to enable face recognition, specify the database used for storing the facial images, and setting the registeration mode. For registering a new user, the recognition_register function makes use of the RegisterUser function from the Intel RealSense API, which takes a picture of the user and stores the image in the database. Finally, the recognition_authenticate function makes use of the RecognitionData interface of the Intel RealSense API to parse all images stored in the database and calls the QueryUserID function to compare the users face with the images stored in the database.

Scanner

Item Identifier

Concerns:

- How will items be identified by the system

A major concern of the system is how to identify items in an efficient manner. Users will potentially need to identify many items in a single transaction, and therefore a scanner has been chosen as the resource for this purpose as it allows the system to avoid having to go through the interface for each item in a transaction.

Design Entity Name: QR Scanner

Type: Resource
Purpose:
The QR Scanner is used to check items in and out of the database. The scanner reads a small QR code printed on the items. The scanner is useful in that it allows users to quickly identify an item, such as a piece of silicon, from within the database, list information about the item to the user, and automatically update its checkout status. The scanner works specifically with the server, which holds a list of all QR codes scanned by the user during a particular session. Once the user is finished scanning and ready for checkout, the server then writes the new checkout status to the database for each item scanned.

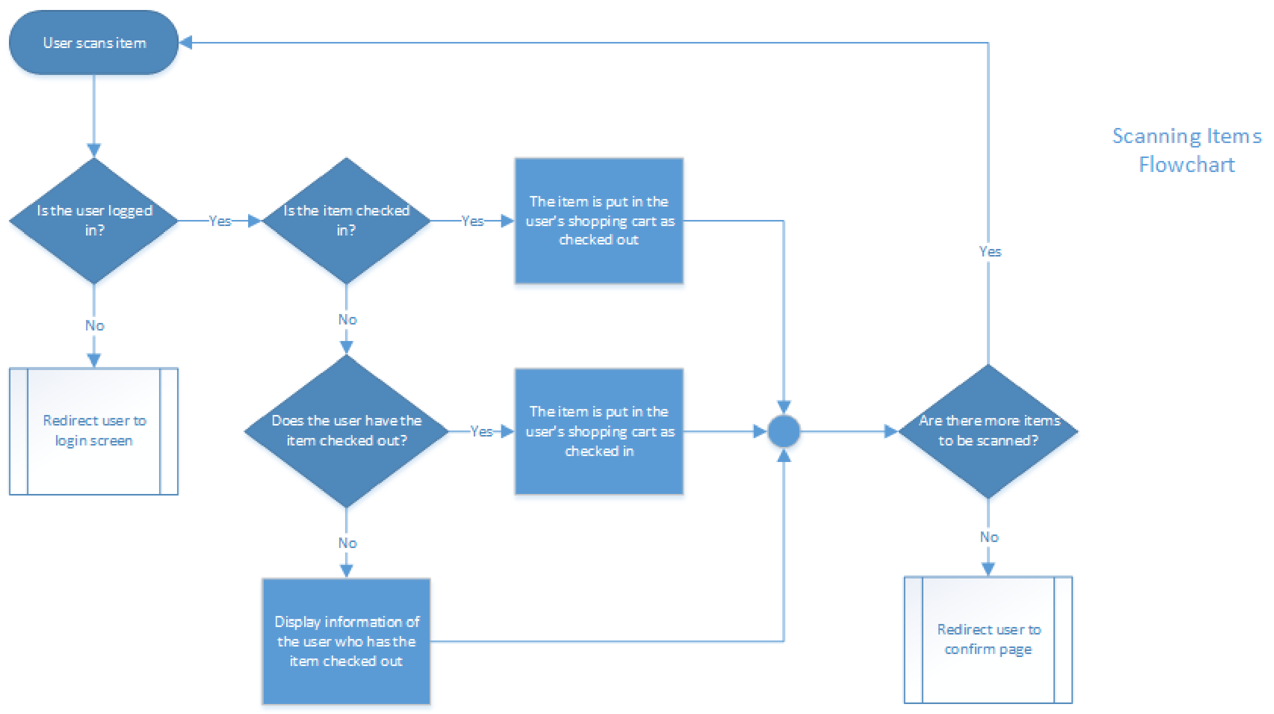

User scans item
Is the user logged in?
Yes
No
Redirect user to login screen
Is the item checked in?
Yes
No
The item is put in the user's shopping cart as checked out
Does the user have the item checked out?
Yes
No
The item is put in the user's shopping cart as checked in
Display information of the user who has the item checked out
Are there more items to be scanned?
Yes
No
Redirect user to confirm page
Scanning Items Flowchart

Figure 7: Flowchart detailing the decision process when scanning an item

Rationale

Our client had a clear vision of what he wanted in this new inventory system. He knew he wanted a web-based system that could scan items, work with their Active Directory server among other specific details. These details guided our decision-making when designing the system.

V. Changes Since the Original Design Document

We have changed the design of the high-level architecture. When we first started designing the project, we werent sure on how the scanner would work. We had thought that we would have to design a separate component for the scanner to handle events and sent barcode information to the server. After playing with the scanner, we learned of its behavior and was able to handle the barcode scan as an event inside the Interface component. The scanner simulates a keyboard and presses a key for each character of the barcode it just scanned. We were able to systematically determine when a barcode was scanned and when a human was typing on the keyboard, thus having no need to separate the scanner into its own component.

Along with the scanner being a part of the Interface component, we also moved the Authentication component into the Interface component. We basically created only a single component that the user can have direct access to. This created a much simpler design and made it easier to develop, since there were less components all needing to communicate with the server.

We had originally thought we would need to create a system that authenticates users based off an Active Directory server. After creating this document, we found out that the system would need to just send a web request to authenticate. We werent told any more information, since that would be a security breach. We ended up creating a mock authentication system and we let our clients team handle the security from there.

In the Interface component, we had split the design work into two subcomponents. These subcomponents are the the Web interface and the Kiosk interface. The Web interface is for monitoring and viewing large sets of information, adding, editing, or removing items, etc. It is designed to be used for any work that would be easier to use a computer with keyboard and mouse. The Kiosk interface is designed in a way to be touch friendly, and to handle quick transactions, such as the checking in and out of the silicon chips.

The database schema has changed since the original design. We originally had a table for each type of item. Today we still have those item type tables, but we have a more generic table that specifies what the item type is, and references data in the more specific tables. The generic table, Items, holds the barcode, the notes for the item, if the item is checked in, and if the item has been scrapped, or logically deleted.

After a large amount of research and help from mentors, we had discovered that the SDK for the Intel RealSense Camera was not compatible with facial recognition on a web browser. This meant that we couldnt use the facial recognition software that was designed for this camera. We had to instead take two images using the RealSense camera. The camera takes an RGB image and a depth image, and uses Open Source Biometric Recognition (OpenBR), a runtime library for image processing, to do facial recognition on the RGB photo and an image comparison for the depth image.

VI. The Original Technology Document

This is the Technology Document we wrote at the beginning of the project. It has been reformatted to match the styling of this document, but the content of the document has not been modified. The original document in its original formatting is included on the usb storage drive.

A. *What We Are Trying to Accomplish*

The PMA team at Intel is in need of a new inventory tracking system for their silicon and other items they need to store. They would like a web-based inventory system where engineers can check in and check out items. All the items tracked will need to be stored on a database. This database will need to track items and user settings. The application will also need to communicate with the Active Directory server at Intel for user verification. Every inventory item has a QR code that needs to be scanned when it is checked in or out. When an item is scanned, the inventory system will update the database accordingly.

B. *Cameras*

1) Standard CCTV camera: The first possible option for the facial recognition camera would be to use a standard CCTV camera, this has the benefit of offering a wide range of products at varying price points, as well as allowing usage of a large number of pre-existing facial recognition softwares. However, many of those softwares are costly, and are more limited than an algorithm based around the capabilities of the next two options. A CCTV camera on its own has limited functionality.

a) Cost: Varies

2) Intel RealSense Camera(F200): The RealSense camera is Intels 3d camera designed for laptop and tablet integration. This Camera was designed to allow for gesture control, facial analysis, sound processing, and augmented reality. In terms of facial recognition, it has a range of 25-75cm. The camera has three display modes, Depth, IR, and RGB. The standalone developer version requires USB 3.0, but the final product would make use of an integrated camera. It supports windows 8.1 and newer and requires a 4th generation (or later) Intel Core processor. This camera has depth precision fine enough to detect gender, emotion, and even the users pulse. The main drawback is the products relative newness, meaning its developer community is much smaller as compared to the other options.

a) Cost: standalone dev kit: $99.00

3) Kinect Sensor: The Kinect Sensor is another possible option for the facial recognition camera. This sensor, originally designed for Microsofts Xbox 360 and significantly improved upon with the release of the Xbox One provides much the same functionality as the RealSense camera. It is a depth sensing camera capable of multiple modes, Color and IR, with a stated range of 5 meters. It was designed with full motion body tracking in mind, though it is fully capable of performing facial recognition. At 50cm the quality would be .75 mm per pixel. Historically the Kinect Sensor has had issues recognizing individuals with dark skin, though Microsoft has blamed those results on lighting levels. The sensor is also designed with the Xbox as its primary usage, there was a dedicated windows version, but that was discontinued in favor of a simple adapter for the Xbox version. The Camera is USB powered and has a robust developer community.

a) Cost: $99.99 + $49.99 adapter

4) Selection Criteria: The chosen camera must be able to output an image with enough detail to successfully recognize an individual. The output must be clear enough to differentiate between an image of a person and the actual person, and the Camera needs to be able to integrate with current systems already in place. Due to project criteria, and the specific request of the sponsor, the camera selected was the Intel RealSense Camera.

C. RFID Reader

1) 1126 Desktop UHF RFID Reader: This product is USB powered and offers both read and write capability for RFIC cards. It is specifically designed for PoS, document tracking, and access control applications and has an extensive SDK. It has the benefit of providing both audio and visual confirmation of a successful read and its built in antenna has a range of up to 1.5 meters, tag/environment dependent. It is designed as a desktop device and as such has no built in mounting bracket or the ability to add one. It works with Windows, Mac, and Linux and is a sturdy, drop resistant design.

2) ID CPR02.10-AD/-B RFID Card Reader: This is a wall mounted reader which requires an external power supply. It is unable to write cards and has two possible communications links. Serial (RS232 or RS485) or Data/Clock (Wiegand). It has a max read distance of 7cm and includes both visual and audio confirmation of a successful or unsuccessful read. It includes both the reader, and the wall- mounted housing. It functions with Windows, Mac, and Linux.

3) pcProx82 Series: This is a USB powered device which works with nearly all proximity, contactless, and magnetic stripe card technologies. It is designed to add additional functionality to already existing employee ID badges by integrating them into applications beyond simple door access. The main benefits of this technology are its versatility, ease of use, and the fact that it has no license restrictions. It is compatable with recent windows operating systems and is capable of sending data as serial ASCII or non-keystroke formats with an average maximum read range of 2.5-7.6cm. It includes a Tri-state LED and beeper for read confirmation and comes with a 1 year material/workmanship defect warranty.

4) Selection Criteria: The reader must be able to successfully read all relevant data from a standard Intel ID badge as well as integrate properly with a tablet PC. Due to project criteria, and the specific request of the sponsor, the RFID reader selected was the pcProx82 Series RFID reader.

D. Database Management System

1) MySQL: This is an open source relational Database Management System (DBM). It is one of the most widely known client-server DBM. One of the main benefits of MySQL is that it is already integrated into existing open source softwares such as WordPress and Drupal due to it being a part of the Linux Apache MySQL PHP (LAMP) package. Additionally, MySQL is supported on nearly every operating system and supports many programming languages. MySQL is known to be easy to setup and use and is the industry standard DBM. However, MySQL is known to have stability issues and suffers from poor performance scaling.

2) Microsoft SQL Server Express: This product is the free version of Microsofts SQL Server. The express version is open to distribute and use, similar to open source software. It implements many of the same features as MySQL but with a better query optimizer, the T-SQL procedural language, and a window-based management tool. However, due to being the express version, the DBM has some limitations. For example, there is a maximum size of 10 GB per database. Additionally, there are artificial hardware usage limits such as only one physical CPU and 1 GB of RAM can be used. Finally, SQL Server Express is limited to running only on Windows and supports only .Net, Java, PHP, Python, Ruby, and Visual Basic.

3) Oracle Database Express: Oracle is an object-relational database management system, meaning that it supports objects, classes, and inheritance directly in the query language. It is an express version and, like SQL Server Express, it has limitations set on data, allowing only 4 GB. Oracle Database is generally targeted at large enterprise applications for huge corporations. It is supported more operating systems and supports more languages than either MySQL or SQL Server.

4) Selection Criteria: Due to the nature of the project, licensing is very important. Therefore, MySQL is the clear answer due to it being open source. Additionally, since MySQL is inherently free, it does not have any artificial limitations imposed on it like the other options have. Microsoft SQL server and Oracle Database both lend themselves to large enterprise applications, whereas MySQL is generally used for smaller web applications. Finally, the team has much more experience with MySQL than the other two languages.

E. Web Application

1) Node.js / Angular.js: Node.js is a javascript runtime engine that runs on Googles V8 engine. We could use this as our web server. Node.js would handle any server-side functionality, and Angular.js would handle the client-side interface. Some of the advantages are that the system would be very scalable, testable, and reliable. Both Node and Angular are built for event-driven, non-blocking systems. This is particularly useful since we will have a number of I/O devices working with our application. It would also be able to run securely with SSL. There are a few downsides to these technologies as well. There is a learning curve that would be particularly larger since we would be learning two technologies at the same time.

2) Visual Studio Community: This is a free version of Visual Studio that we could use to create the web application. VS Community is a full IDE with everything we would need to create a fully interactive web environment. Building an ASP.NET application would be a good option because it is such a powerful framework. The back-end would be written in C#, which is designed in a way to easily interact with databases, lists, string manipulation, and much more. It comes with Linq, which allows developers to create database queries, and sort through lists seamlessly. This is also a very large framework, and the learning curve is probably greater than any other option listed. There are also limitations to the licensing with the free version, so that also has to be kept in consideration.

3) Vanilla HTML/CSS/Javascript with PHP: A third option would be to go without a web framework, and use a more basic approach to developing the software. We would just write base web interfaces with HTML, CSS, and Javascript, and then use the server-side tool PHP, which is a more minimal scripting language. This would benefit us by giving us the freedom to come up with whatever design structure we want to use. We could essentially write the code in anyway that we feel would be best for the system without being imposed by framework rules and guidelines. On the other side of that argument is that we could have too much freedom and create a complex and chaotic system that is nearly impossible to understand, since we didnt impose any coding standards. It would have a smaller learning curve at the beginning, but as the system grows it might not be the best option.

4) Selection Criteria: As stated in other sections, licensing is very important. All three of these systems are free, but Visual Studio Community will have some limitations that would deter us from using it. We also have to make sure that we can read information from I/O devices, such as the RFID reader and the camera. Node.js has the ability to work with I/O systems easily. It has the ability to run C and C++ libraries within the system, which is useful for the RFID readers and cameras. Angular.js easily integrates with Node.js and can create dynamic web pages.

VII. Changes Since the Original Technology Document

Because of a large learning curve for software that wasn't necessary for development, we decided not to use Angular.js, a javascript web framework, for the interface. Instead, we used only HTML, CSS, and JavaScript/JQuery for our interface, without a framework.

We had discovered that the SDK for the Intel RealSense Camera could not do facial recognition if the camera is being used through a web browser. Because of this, we switched to using Open Source Biometric Recognition (OpenBR), a runtime library for image processing.

VIII. Weekly Blog Posts

Throughout the course of the project, we wrote a weekly update on a blog post whenever there was work done on the project. Each blog post is in chronological order and styled in a way that makes it easy to differentiate between them.

A. Welcome to my blog! by Cronise, Joseph Eric

This is where I'll be sharing my thoughts on topics that matter to me. Who knows... I might even share pictures, videos and links to other interesting stuff.

If I catch your interest, let me hear from you.

B. Weekly Update 10/23/2015 by Cronise, Joseph Eric

This week we held our weekly meeting with Scott Oehrlein on monday. We went over his changes to the design document as well as clarified a few matters and asked any questions we needed.

On tuesday we were given two Realsense cameras which were distributed to Dylan and Brett, Joseph has his own, albiet an older model. Each member took some time to briefly familiarize themselves with the camera.

Over the entire week we, individually, have been looking into different web framework options for the project as well as refamiliarizing ourselves in database operations.

C. Weekly Update 10/30/2015 by Camus, Dylan Michael

Again this week we held our weekly meeting with Scott Oehrlein on monday. We didn't have too much to talk about but we discussed the requirements document and planned to meet with Scott in person on Wednesday November 4th.

We have continued looking at web frameworks and database options and are starting to get a much better handle on what the project will look like.

D. Weekly Update 11/6/2015 by Camus, Dylan Michael

This week our weekly conference call with Scott was canceled because our client Scott Oehrlein was in Corvallis and we were able to meet with him in person. We went over the old web app that the Intel team currently uses and where he wanted to see improvements. We also received three RFID scanners from Scott to work with.

In addition, we had our first meeting with the TA this week. We went over our requirements document and where it needed improvement.

E. Weekly Update 11/13/2015 by Camus, Dylan Michael

We had a little trouble this week due to daylight savings time. We ended up missing our weekly meeting with Scott because Arizona does not recognize daylight savings time. Additionally, because of veterans day school was canceled, and therefore we did not meet with our TA either.

What we did, however, was complete our revisions on the requirements document and finished the tech review. We are looking to begin working on our design document soon.

F. Weekly Update 11/27/2015 by Hayes, Brett

This week has consisted of a number of different tasks.

We talked with Scott about a week ago about adding/deleting a few user stories from the requirements document. We have bounced ideas back and forth with our client and his needs for the system. We have a new revision of the document and hopefully we this version has everyone happy.

We still haven't had success with the RFID cards and readers. The cards from our client are blank, and we are finding it difficult to find anyone with an RFID writer so we can put fake data on the cards. We tried the RFID readers with our student IDs but have not had any success with them.

We prepared our elevator speech and are ready for our turn next week to give the speech. Brett is giving the intro, Dylan is giving the problem and Joseph will deliver the solution.

We plan on working on the design document this weekend and will work hard on it in the beginning of next week.

G. Weekly Update 12/5/2015 by Hayes, Brett

We had a lists of concerns for this project. We placed our list of concerns into the design document and wrote about how we plan to address those concerns.

Brett mapped out the high-level design and how data is going to flow throughout the system in a brief but informative way. He also went into detail about how the server will be constructed and some design features we will be implementing.

Joseph came up with some prototype designs for the interface. He was able to create a model of the system and what it will look like. Some screenshots of the mock-up are in the design document. Joseph also created a mapping diagram of the database.

Dylan wrote an interface for the Active Directory API and how we will be working with the RFID reader, facial recognition, and username/password authentication. He also wrote how the barcode scanner will be used and implemented into our system.

We spent most of the week on the design document. Aside from the document, we received the barcode scanner from Scott and were able to test out the scanner on some fake CPU silicon chips that Scott also provided.

H. Weekly Update 1/16 by Hayes, Brett

The previous week was geared towards setting up our coding environments. This week we have been building the basic blocks of our product.

Dylan continues to work on the database scripts. He is writing what will be the CREATE TABLE scripts.

Joseph has been developing a skeleton of all the web pages for the project. The welcome screen has been placed on our webpage. It contains a simple username and password form.

Brett has set up the database that the group will use to develop the project. It is connected with the hosted website that he put up online. He is also looking into methods of how to detect when a barcode scanner and an RFID reader will be hooked up to the kiosk.

We had our weekly meeting with Scott on the 15th. We decided that next week we will be going over the design of the interface considering we get a few more pages online. Scott plans on visiting OSU in February, so we also discussed a good meeting time for when he is in town.

I. Weekly Update 1/29 by Hayes, Brett

Joseph has been working hard on the interface, and making it look aesthetically pleasing. He has been the primary person to work on the front-end coding. Joseph has also been working on figuring out what is needed for the RFID tags.

Dylan has continued to work on the database scripts. After writing the scripts to create the tables for the database, he has been working on the stored procedures that will be called when creating an item, looking up an item, etc.

Brett has been working on the server-side code. He has been attempting to make the website secure through SSL security. Brett has also started looking into what it will take for an automated e-mailing system. He has begun the process of integrating OAuth authentication.

After our weekly meeting with Scott, we have decided to rewrite the styling of the website. He was also able to send us some sample data after the meeting so that we can begin testing some of the scripts we have written.

J. Weekly Blog Post 2/5 by Hayes, Brett

Joseph has continued to work on the interface design. He has made a lot of progress and is now waiting for our client Scott to review.

Dylan and Brett have been working together on the server-side code. They have made the server connect to the database and have the ability to pull data for display on the website. Brett has also made a table with multi-column filtering (one of the requirements).

The next steps are building a check in/out system, writing the midterm report, and recording a video for the midterm. Our hope is to get a somewhat polished version of our website before Scott arrives to Oregon on 2/17.

K. Weekly Update 2/12 by Camus, Dylan Michael

Our client Scott reviewed the updated interface design that Joseph created and was overall satisfied with the design. However, he has some concerns over the usability on smaller resolutions.

Dylan and Brett continue to update the server code. They are working on getting the SSL working and adding functionality for adding items to the database through the application. Additionally, work is being done on the kiosk app to allow for checking items in and out.

We finished our midterm report and video. Moving forward, we are looking to have the core functionality of adding items and checking items in and out ready for when scott comes on the 17th.

L. Weekly Blog Post 2/19 by Hayes, Brett

Scott came to visit from Arizona to see the progress we had made on the site. We actually came up with a lot of progress in the past week.

We now have the kiosk inserted into our live site. If a user wants to view the kiosk, they will have to type in the navbar after the website ´/kiosk¿

A user is now able to scan an item when in the kiosk interface. If the user is on the login screen, they can scan an item and the information about the item is returned for viewing. The same code will be used for implementing the check in/out sytstem.

In our meeting with Scott and his team (through Skype) we were able to show the current progress of our system. Scott and his team had some great feedback, and it was really good to hear what they had to say. We have a lot of details about their wants and needs in the system.

M. Weekly Blog Post 2/26 by Hayes, Brett

Brett worked on server administration mostly. He moved the repository to a public repo on github. He also spent time writing the readme on the github page explaining how to deploy the app onto a local machine, or with little modification, to deploy to a server. The new repo can be found here:

https://github.com/bretterism/silicontracker

Along with writing the directions and moving the repository, Brett was able to move the latest production environment to a home server. The reason for this was that our hosting space on OpenShift was handling a lot of the SSL security automatically, so we want to make sure that our app works specifically with SSL and implementing whatever we need for that.

We also didn't have our weekly meeting with Scott. He had to cancel, and we all had previously decided to set another time to get together anyways. We are currently figuring out what time works best for everone.

N. *Weekly Blog Post 3/4 by Camus, Dylan Michael*

This week we worked on getting the last of the features that need to be in before the beta implemented. Brett set up the website on a home server and set up ssl security. He also set up the github repository for our migrated code and is working on updating the checkin/out system with user profiles.

Dylan updated the database to have a more efficient way to store which items are curretnly checked out. He also setup an email scheduler that sends emails over an interval. He is currently working on creating a SQL stored proceedure to only send emails to users who have had an item checked out for some length of time.

Joseph has been working on the facial recognition authentication. He currently has it so that the web app will select the users web cam and display the feed on the login page. He has looked into implementing the facial recognition algorithm and has it almost finished pending more testing.

Our meeting times with Scott have been rescheduled to 9:30 AM on Mondays.

O. *Weekly Blog Post 3/11 by Hayes, Brett*

This week we discussed a few of the features with Scott, and he responded with good feedback. We added a feature to add new attributes for the dropdown lists when adding a new item into the system. A dynamic table is loaded to create edit or delete these dropdown attributes whenever there is a change to be made.

We also added a user settings page, which lets the users change some basic features if they desire. This is where the dropdown attributes are found.

Basically we have added a way for the user to login, and interact with the system as a user. We implemented sessions for user. Basically, the user's information is kept on the server and is passed to the client on a per-need basis. That way secure information, such as the user's WWID, can never reach the client, but their name and user settings can be reached.

This next week we plan to go through our requirements document and make sure we have all the requirements fulfulled. We will be working hard on writing the document and will get together in the next few days in order to create our video for the final presentation.

P. *Weekly Blog Post 4/8 by Hayes, Brett*

Starting back into the term, we have worked very hard at refactoring a lot of our code. At the end of the winter term, we had pushed a lot of code, so our code got pretty messy. We refactored it over spring break and started the term off with a clean code-base.

Part of the refactoring involved stripping the web template we were using for our web pages. The template did not look well on all screen sizes, so we stripped the template and have a responsive web site.

We have derived a task list that covers all the code-related tasks for this term. Our task list can be found here:

https://docs.google.com/spreadsheets/d/1ue5x-tGK24JxzuKh7rXZGm6fWu7G3c6rAJPGCW_5Hcc/edit?usp=sharing
Brett has been working on a few major projects since the beginning of the term. He has created the ability for users to edit rows on a table, as well as write notes for each row. Brett has also worked on writing libraries for form scrubbing and validation. When someone enters a form, the data gets cleaned up and ensured that all the values are in acceptable format.

Dylan has been working on the scrapping interface, so the users can logically delete items in the database.

Joseph has been hard at work on the facial recognition software and should hopefully have a working system implemented within the next few weeks.

Q. *Weekly Blog Post 4/15 by Hayes, Brett*

This week we have continued to work on our tasks and have been making our software more robust with less bugs. Brett has just about finished with validation and sanitization of the input forms. He has also been working on making sure all the tables on the web interface display nicely.

We have learned from our client that it may not be worth including the Intel logo in our work. This includes the poster and our project. So we are working on changing our poster and project to exclude the Intel logo. We are still allowed to use the word Intel in our work as long as it is used in the correct way.

R. Weekly Blog Post 5/9 by Hayes, Brett

We have forgotten the past few weeks to write a blog post, so this will be a larger post regarding what has been accomplished in the past few weeks.

April 16-23: We had a sanitizer validator for some of the tables, but not all of them. In this week we brought all of the tables currently in the system up to the same standard as the CPU table. The main reason why the other tables weren't at the same standard was because we were making constant changes to the structure of the tables, so once we came up with a solid standard for one table, we then made that standard for all tables. After getting all of the four tables in place, we added a fifth table for motherboards. These changes broght the tables into a finished state.

April 24-30: We changed the way the system handled database connections. Instead of a single connection, we made a pool of connections so that people can make database requests concurrently. This speeds up the system by removing the bottleneck of the system. We had also began further development on the displaying of item information on the kiosk. If the user scanned an item on the kiosk login page, they would get an alert with the item's information. It was only considered a temporary solution, and it only displayed CPU information. So we made a nicer solution for displaying information, and a user can scan any type of item and get the information. Finally, when developing for the barcode scanning portion, we wrote a program that simulates a barcode scan. It's a command-line program that, when given a serial number, will behave in the same way as our barcode scanner and type in each character of the serial number at a very fast rate. This has helped significantly with development, and allows us to test our code that requires a barcode scanner without actually using the barcode scanner.

May 1-7: We added a few new pages to the website. They both pertain to scrapping, or logically deleting, of items. The first page is the scrapped items page. It looks pretty much identical to the main page with tables. The only difference between the two pages is that the main page shows active items, and the new page shows scrapped items. The other page that was added is the "mass scrap" page. This allows a user to scan all the items they wish to scrap, and then once they are done scanning, can submit and set all those items to scrapped.

S. Weekly Blog Post 5/15 by Hayes, Brett

We have been putting in a lot of effort this past week in preparation for expo. All of us have been working very hard at making sure everything is finished on our project.

Brett has run through a number of bugs and features. He has fixed an issue with loging in from one interface and that session carrying over to the other interface. There is also a "quick" check in/out option on the kiosk login page. Admins can also edit the dropdown menus in the add item pages.

Dylan has been working on adding reservations to the system, so if an item is already checked out, it can then be reserved by another user.

Joseph is about done with the facial recognition, and should be up very soon. He has been having trouble with different SDKs and is finally about to have a working solution.

We are almost prepared for expo. We just have to make sure that our system can run between our machines, instead of relying on the internet in Kelley, since the internet won't be very reliable.

T. Weekly Blog Post 5/27 by Hayes, Brett

Due to expo being last Friday, we forgot to write our blog post last week. This blog post will contain the work accomplished within the past two weeks.

The week leading up to expo was filled with writing a lot of code and trying to make sure that everything was working before expo. We had finished a lot of tasks and bugs including the reservation of items, and facial recognition. The night before expo we had to stay on campus through most of the night in order to finish the project and make sure it was ready. One of the reasons we were up so late was to make sure that our system would work without hiccups at expo. That included making sure that our two laptops were sharing a database, and that our website could work if we turned the internet off of our laptops. The other reason we were up so late is because we were working on getting facial recognition to work, and we had to work on it until the last minute. The Intel Realsense Camera was really difficult to work with, and it took a lot of hours to make sure that it worked correctly.

The day of expo went very well. None of us had gotten much sleep, but we were all excited to show off our project and we all felt proud of our work. We ended up being between two projects that were both interesting to the general public. To our left was a virtual reality implementation, and to our right was an RC car controlled remotely. Despite our project not being quite as flashy, we still were able to show it off to a lot of people, including some people representing some of the local industries. Our client, Scott was in town for the expo and he seemed happy with the work that we had put into this project over the months.

We haven't worked much after expo. There was a lecture on Wednesday and Kevin talked about the report and presentation. We have two weeks to finish both of these, but we will try to submit them sooner. This next week will be dedicated towards working on these, and after that, we will be finished with our capstone project.

IX. Conclusion