ShopTracker

An Inventory System and Access Tracker for the CECS Machine Shop.

CSM 2 – CSCI 370 - Advanced Software Engineering – Summer 2013
Team: Taylor Sallee, Chaeha Park, Nicola Hetrick, Shawn Toffel
Client: Dr. Kevin L. Moore, Dean of CECS at CSM

1.0 Introduction

Advanced Software Engineering is a required capstone course for all Computer Science undergraduates at the Colorado School of Mines (CSM). Students break up into small teams of two-six and complete a project within the first six weeks of summer, usually after completing their third academic year. Denver-area businesses and organizations often submit projects for field session groups to complete, and each year Mines departments/faculty members submit projects as well.

1.1 Client

Our project was completed for Dr. Kevin L. Moore, the dean of the College of Engineering and Computational Sciences at CSM. Dr. Moore is also the G.A. Dobelman Distinguished Chair of Electrical Engineering. Dr. Moore is interested in engineering education, and is consistently involved with CSM's signature Capstone Senior Design course. He received his B.S. in Electrical Engineering from Louisiana State University in 1982 and his M.S. in EE from the University of Southern California in 1983. In 1989 he earned his Ph.D. in EE with an emphasis in control theory from Texas A&M.

Before coming to CSM, Dr. Moore was a Senior Scientist at the Johns Hopkins Applied Physics Laboratory, a professor of electrical and computer engineering at Utah State University, and an engineering educator at Idaho State University. He has also worked at Hughes Aircraft, and is the sole proprietor of the System Analysis and Control Company. Out team was lucky enough to be assigned to Dr. Moore's project, and over the course of six short weeks we have had the amazing opportunity to get to know him while developing this software.

Although Dr. Moore is our client, the machine shop supervisor, John Jezek, will be the main user of our product. We worked closely with Dr. Moore while developing our product, but we also consulted Mr. Jezek about certain design decisions specific to his area of expertise. Our product was built to be used by Mr. Jezek and the campus members who utilize the machine shop, so we kept them in mind while programming, even spending much of our development time working right across the hall from the shop.

1.2 Product Vision

CECS maintains a machine shop in the basement of Brown building. The machine shop is home to many machines which require specific training to ensure safe operation. Our software will help create a safer and more organized environment in the machine shop by keeping track of machine safety certifications and allowing shop staff to have fast and accurate access to certification information and the inventory database.

In order to make the shop a safer place, our software must be accurate. Its two most important tasks will be tracking each user's machine certifications and keeping a running log of which users are in the shop at all times. The more the program is used, the more effective it will be. We therefore strive to make it as simple and user-friendly as possible for both basic users and administrators.

2.0 Requirements

Every project has requirements – both functional and non-functional. Most of our requirements were provided to us by Dr. Moore from the beginning of the project, but more revealed themselves during our development process.

2.1 Overview

ShopTracker must record who comes into the shop, which machines they use, and when they leave. It also must allow users to check out tools if they need to borrow something and leave the shop with it. The program must keep an inventory of all tools and machines, and make note of each user's certification status on each machine.

The majority of users will have a simple interface with only five options: sign in, sign out, use machines, check out tools, and return tools. The shop supervisor must be able to view the stored information in a variety of helpful ways, as well as edit the database to add/remove tools, machines, and users, and update the certification status for each user on each machine.

Most of our target users are people who will be coming into the machine shop to get work done. Therefore our program must be fast, simple, and easy to use, so that users won't find our system to be a hindrance when they are trying to. Since the allotted time is only six weeks, our software needs to support the client's required functionality, but must also be left open for extension by future.

The following list of requirements were either specifically asked for by our client, or inferred in the course of our client meetings and planning sessions.

2.2 Functional Requirements

- Every user can:
    - Use their CWID to sign in/out.
    - Check out and return tools.
    - View their machine certifications.
    - Use only the machines for which they are certified.
    - View their checked out tools.
    - View tools available for checkout.

- Administrators can:
    - View all active users.
    - See the status of all tools and machines.
    - Filter log by date, user, tool, machine, locked users.
    - Generate statistics for each report.

- System Administrators can:
    - Update each user's machine certification status.
    - Add / Remove Tools from the database.
    - Add / Remove Machines from the database.
    - Add / Remove Users from the database.
    - Edit privilege levels for each user (Make a user an administrator/system admin).
    - Lock Out Users (and unlock them)

2.3 Non-Functional Requirements

- The system will:
    - Use a magnetic strip card reader to let users scan their Blastercard to extract a CWID.
    - Be open to extension (new features should be easy to add).
    - Be installed on a computer in the machine shop.
    - Have an appealing user interface which is simple and easy to use.
    - Be cross-platform and flexible, in case it needs to be installed on a new computer.

3.0 System Architecture

Our application is made up of four main components:

- MongoDB for persistent storage.
- Java code to handle all of the program's logic.
- User Interface (three types, one for each level of user privilege).
- Mines Oracle Database for user verification.

3.1 MongoDB

We used MongoDB to persist all of the data needed for our system to operate. Our MongoDB instance is hosted locally on the computer in the machine shop, and stores information about the users, tools, and machines, as well as log information such as time in, time out, machines used, and tools checked out/returned.

3.2 Java Application

The heart and soul of our program is the Java application. The java code handles all of the logic for our program. It extracts data from the MongoDB instance and the Mines database, performs operations on it, and writes it back to the MongoDB instance for permanent storage.

3.3 User Interface

The UI is also written in Java, with Swing as the framework. Each user has a different view of the program, based on the permissions they have to perform certain functions. Instead of putting logic in to

every function to make sure it was being called by an authorized user, we simply made the UI show buttons that would provide access to the functions for which each user is authorized.

3.4 Mines Oracle Database

CSM maintains an Oracle database which contains information about all active students. We extract some of this information to verify that the users of our system are actually active Mines campus members. We only pull each user's name, CWID, department, and e-mail address from this database.

3.5 Overall System Architecture

Figure 3.5.0 shows how our system is constructed. The design is straightforward and simple.

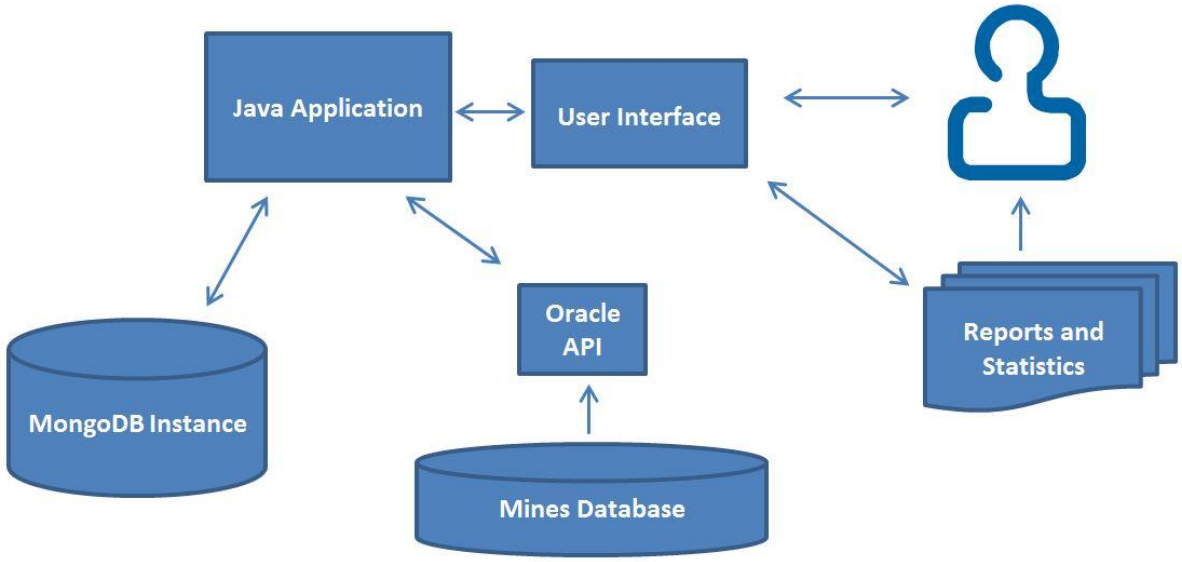

Figure 3.5.0 – System Architecture. The Java application connects every other piece of the system. The MongoDB instance is used to store information generated by the system, which comes in part from the Mines database. This database is talked to via our Oracle API, which is an API designed by us to pull specific information from the Mines Database. The User interface provides access for the user, and facilitates report and statistic generation.

4.0 Technical Design

This chapter talks about how our system was actually developed, including how we fit everything together, and how each class/file contributes to the functionality of the product. As previously mentioned, the Java application is the heart of our system. We will therefore explain that part of our system in the most detail. We will, however, describe how we used the MongoDB instance, and describe the API we used to interface with the Mines Oracle database.

4.1 Java Application Design

Our Java program uses several important classes to store and manipulate the information about users, machines, tools, and log information. These classes are all shown in **Figure 4.1.0** below, and described in detail in the following sections.

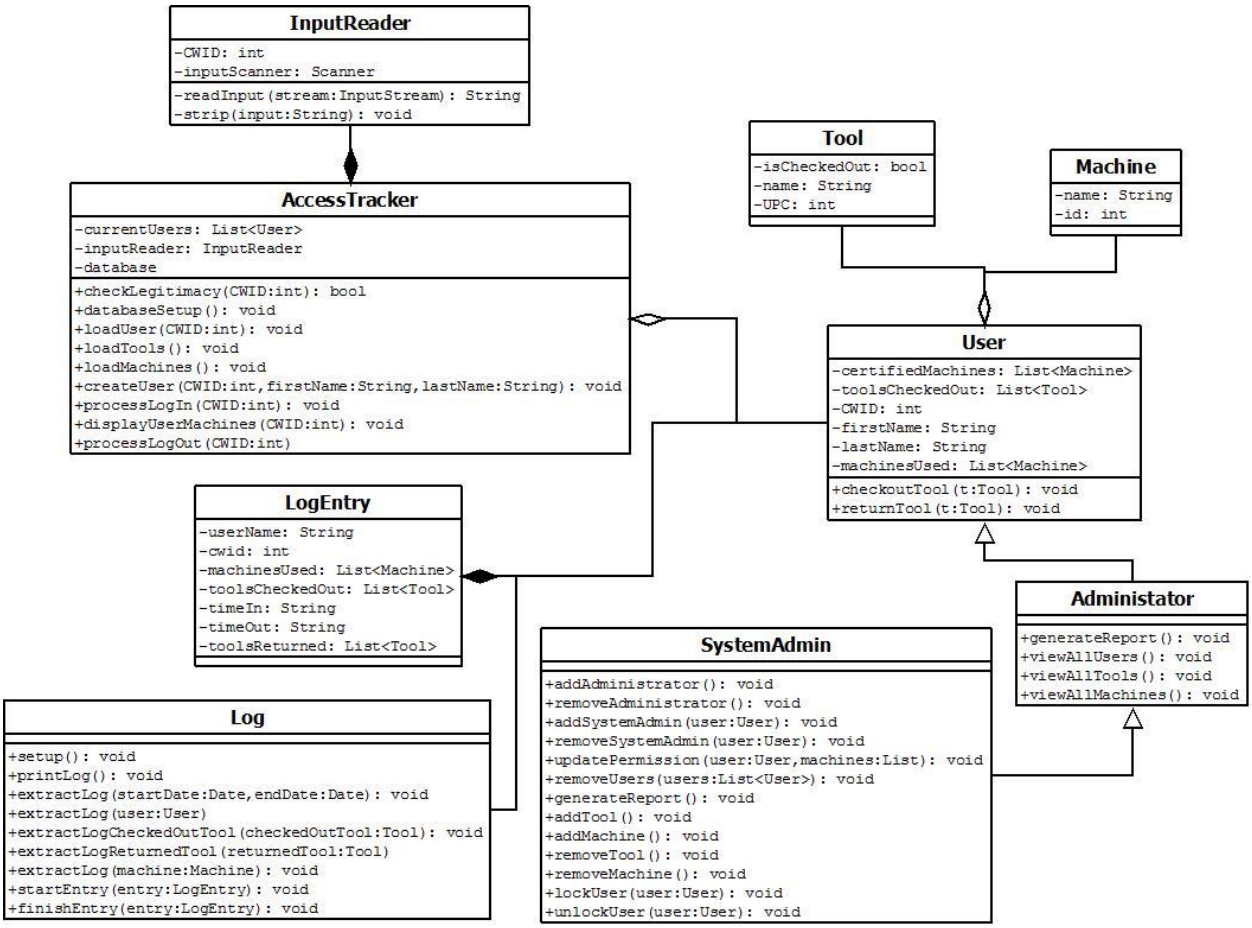

Figure 4.1.0 – Java Application Design. This UML diagram shows all of the main classes used by our Java application. Each class has a specific function. The most important ones are described below.

4.1.1 AccessTracker

AccessTracker is the class that holds everything together. It processes user log ins/log outs, takes care of loading tools and machines from the database, and performs initialization/system functions. It is basically the “tracker” in ShopTracker.

4.1.2 User

The User class is the parent class for both Administrator and SystemAdmin. This class contains the basic information and functionality common to all users. Each user has a CWID, a first and last name, and a list of machines for which they are certified, and a list of tools they currently have checked out. This

information is pulled from the database (and stored to it) each time a user needs to do something, such as check out or return a tool.

4.1.3 Administrator

This class has a couple of extra functions, including the ability to generate reports and view information.

4.1.3 SystemAdmin

This class contains all of the functions necessary to edit the database. Since the system administrators are the only users allowed to edit the database, the functionality to do so resides only in this class. Adding and removing users, tools, and machines, editing machine certifications and user permissions, and locking/unlocking users are all functions that only system administrators can use.

4.1.4 Tool and Machine

The Tool and Machine classes hold only their name and ID, and are manipulated by other classes in the program (namely the User class).

4.1.5 Input Reader

This class takes care of processing Blastercard scans, as well as keyboard input.

4.1.6 Log and LogEntry

The Log and LogEntry classes basically interface with the database to keep track of when people come and go, and what they do while they're there. The startEntry() function is called when the user first swipes their card (or enters their CWID), and finishEntry() is not called until they sign out for the day. Any tools they check out/return or machines they use during the current session are all added to the current log entry for that user.

4.2 MongoDB Instance

MongoDB is a document-oriented database management system which stores data formatted as JSON (JavaScript Object Notation) documents. We made use of MongoDB to persist all of the data used by our program, including information about users, tools, and machines. **Figure 4.2.0** shows the design of our database.

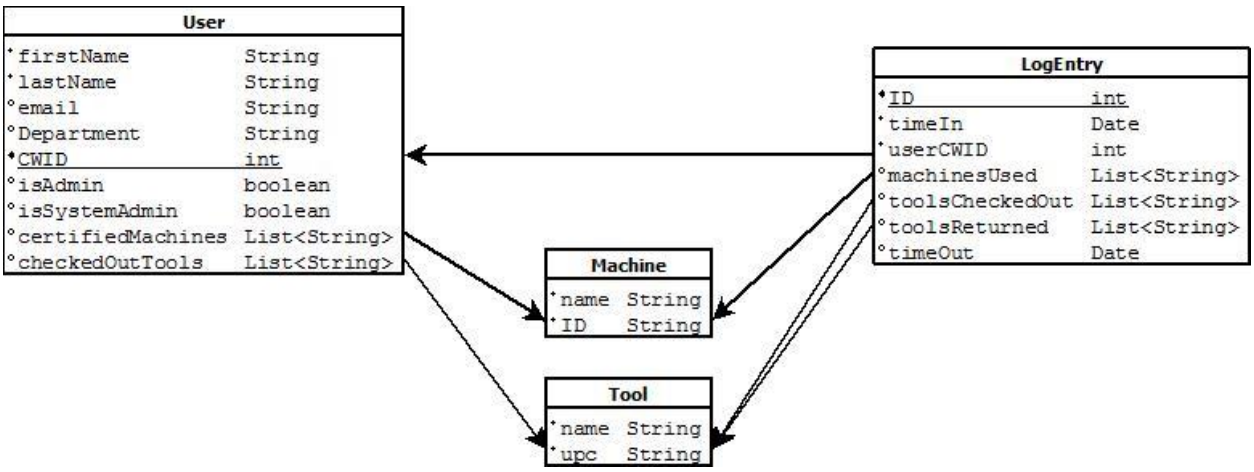

Figure 4.2.0 – The MongoDB Instance. Each table represents a document, and the arrows represent foreign keys to other documents. For example, the userCWID field in LogEntry refers to the CWID of a specific user. Users are extracted into RAM by CWID, machines and tools by ID and upc, respectively, and LogEntries by whatever parameter the SystemAdmin searches by.

Note that there is no Collection for Administrator or SystemAdmin. This is because all of the data associated with an Administrator/SystemAdmin is already stored in the User class, and the only difference between them is the methods available to each class. Instead of creating two entirely new collections, we just added boolean values to the User documents in the database. MongoDB does not make every document in a collection adhere to the same schema. Therefore if these two values (isAdmin and isSystemAdmin) don't exist in the user document being extracted, we know the user associated with that CWID is just a basic user.

Because our database is JSON document-oriented, translating between Java objects and MongoDB documents was very simple, and saved us from spending a considerable amount of time writing SQL.

4.3 Mines Oracle Database/API

In order to validate that new users were valid Mines campus members, we needed to connect to a central Mines database that had information such as name and CWID. A secondary purpose for connecting to such a database was to pull information about that user so it wouldn't have to be manually entered. After talking to CCIT, we found out that, although they have the information we needed, we wouldn't be able to access it during development due to privacy issues. We did, however, construct an API that talked to a dummy database we set up which had the same fields as the Mines database (which is an Oracle relational database). Once CCIT gets us access to the real database, all we will need to do is switch the names to get the real information we need.

Figure 4.3.0 shows the schema of the dummy database table we set up, which we practiced pulling from, so that we knew our API would work once we got access to the real database.

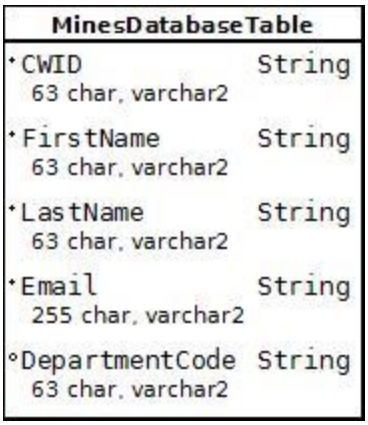

Figure 4.3.0 – The Mines Database. This table schema represents the dummy database we set up. The names of the table and fields are not the same as in the real Mines database, and our dummy table is much smaller than the table in the real database.

Since we only pull data from the table and never edit it, we only ever need one SQL query to select the data to populate the fields for the new user we are creating. If our query returns null, we know the CWID/Blastercard entered is not an active Mines campus member. **Figure 4.3.1** shows the code we used to extract information about new users from our dummy database. The names will obviously need to be changed once CCIT connects our program to the real database.

```
public ArrayList<String> select(String userCWID) throws SQLException {

    ArrayList<String> results = new ArrayList<String>();

    String sql = "SELECT " + cwidColumn + ", " + firstNameColumn + ", " + lastNameColumn + ", " +
            emailColumn + ", " + departmentCodeColumn + " FROM " + tableName + " WHERE " +
            cwidColumn + "=" + userCWID;
    Statement statement = connection.createStatement();
    ResultSet rows = statement.executeQuery(sql);

    if (rows.next()) {
        String cwid = rows.getString(cwidColumn);
        String firstName = rows.getString(firstNameColumn);
        String lastName = rows.getString(lastNameColumn);
        String email = rows.getString(emailColumn);
        String departmentCode = rows.getString(departmentCodeColumn);

        results.add(cwid);
        results.add(firstName);
        results.add(lastName);
        results.add(email);
        results.add(departmentCode);

    }

    return results;
}
```

Figure 4.3.1 – Extracting New Users. This Java/SQL code shows how we pull information about new users from the Mines database. If the query returns null, we prompt the user to go see the shop supervisor.

5.0 Design and Implementation Decisions

To build our system, we had to make many choices regarding which technologies to use and how to make them all fit together. The final system we created has many small parts which all function together to make the result a quality product. The following is a list of technical design and implementation decisions.

5.1 Using Java to develop the system

Our choice to use Java as our language platform was an easy one. Our application is fairly straightforward, without a lot of fancy graphics or math involved – one could say it is a basic business application. The main benefit of Java is that it is cross-platform. Almost any computer can run Java, and since we weren't sure which type of machine Dr. Moore would want to use for our application, we decided to make something that could easily run on many types of machines. Since our program needs to connect to two different databases, we also wanted a language that supported an API for these databases. Java is a universal programming language, and will make our product easier to extend if anyone decides to do so in the foreseeable future.

5.2 Using MongoDB for persistent storage

There is no real schema to a MongoDB instance, except for named collections of documents. In other words, the amount of structure the data must adhere to is completely up to the programmer and the application which uses MongoDB for support. MongoDB is useful in many ways, but one of its features that we found helpful in our project was its flexibility. For example, in our system, users can be certified on variable numbers of machine. A traditional relational database would use join tables and complex SQL queries to extract all of the machines for a given user. In MongoDB, each user is stored as a document, and each document can hold as little or as much information as needed. Extracting the user's machines is as simple as this:

```
BasicDBList machinePermissions = user.get( "certifiedMachines" );
```

We decided early on to use MongoDB to persist the data used by our program. We knew we only had 6 weeks to complete this project, and that we would be making frequent changes to the database design. With some other databases, we would not have been able to make these schema changes quickly enough to support our rapidly changing and limited-time project. Since MongoDB allows you to have a very flexible schema (basically no schema at all), changing the design of the system is easy. Another factor to our decision was that MongoDB is free, and setting up a practice account online to use while developing was fast and easy. One more benefit of using MongoDB is its JSON-format storage. Translating Java objects into JSON is quite simple, and doesn't involve complex SQL queries to retrieve pertinent data.

5.3 Using the Mines Oracle database to validate new users

Our system will be used almost exclusively by Mines students and faculty members. The system keeps track of users and adds new users to the database any time a student scans their Blastercard for the first

time. In order to know that a student is entering a valid CWID, we decided to use the existing Mines records to make sure the student exists before adding them to the database. This decision was made mostly for security purposes. The entire reason we developed this system was to make the Machine Shop safer, and this is one way to make sure that the activity in the shop is effectively monitored and logged.

5.4 Creating three permission levels for system users

In our application, a user can either be a basic user, an administrator, or a system administrator. Users can only log in, check out tools and use machines, and log out. Administrators (such as Dr. Moore) can do everything a basic user can, plus view tools, users, and machines, and generate reports and statistics about the activities in the shop. System Administrators have complete control over the system, including generating reports and statistics, adding/removing tools and machines, locking/unlocking users, adding/removing users, and updating machine certifications and user permissions. We decided to use these three levels because we knew that only the shop supervisor (John Jezek) would really need all the functionality to change the database, but that other school officials (such as Dr. Moore) would want to view reports, but might also want to actually make use of the system for tool checkout/machine use as well.

5.5 Using Swing for a GUI framework

We decided to use the Java Swing library to create our user interface because we wanted our product to be flexible if it ever needed to move to a new computer (or an old one, for that matter!). Since our application was written in Java, it was easy to add a GUI written in Java, because no external interfacing was necessary – we just imported the correct libraries and got right to work. Our GUI is more extensive than any other project any of us have done before, so we had to learn a few things. One main thing we learned was how to use GridBagLayout. We were familiar with GridLayout and BorderLayout, but using GridBagLayout turned out to be very helpful, because it gives almost absolute control over how things are positioned on the screen.

6.0 Results

Our program meets all of our client's functional and non-functional requirements. The MongoDB instance and java application combined perform all of the necessary functions to fully track the users and inventory of the machine shop. The user interface is simple and easy-to-use, and has been approved by both Dr. Moore and the machine shop supervisor, John Jezek.

6.1 Fulfillment of Client Requirements

Our system allows users to log in, select the machines they will be using for that session, check out/return tools, and then start working. They may then return to the computer any time throughout the session and select more machines, or check out/return more tools, and then return to work. When they are done working, they can sign out. These are the only functions available to basic users, as requested by the client. Administrators have all the functionality of a basic user, but can also generate statistics and reports about shop usage, view all tools and machines, and see all active users (those

currently logged in or those who have tools checked out). A System Administrator (the highest permission level) has all the functionality of an Administrator, plus the ability to add and remove tools, users, and machines from the database, lock/unlock users, edit each user's certification status on each machine, log in another user while still working, and update the security permissions of every user (User, Admin, or System Admin).

6.2 Technical Difficulties

The only problem we have had with getting our product fully functional is that CSM's CCIT has not yet given us server space for our MongoDB instance or access to the Mines database, so our project is waiting for those two essential pieces. For now we will install MongoDB locally on the computer that will be used for this system in the shop, and we will go without the Mines database access until CCIT finishes that work order. That part will be finished roughly six-eight weeks after our class is over, so the product we present now is not 100% complete, by no fault of ours.

6.3 Possible Future Features

There were a few 'if time allows'-type features that, given time constraints, we were not able to implement, but would be nice to have if someone decides to improve the system at a later time:

- We would have liked to be able to use a barcode scanner to keep track of tools when they are checked out/in.
- We wanted to allow users to reserve tools/machines (implement a waiting list of some kind).
- Multi-user access was bounced around as an idea (Allow data entry from one computer while the computer in the shop is being used by basic users, etc.), but we just didn't have time to put in all the necessary work to get this done.

This system could be extended in many ways, and so we left it open to allow someone else to add to it in the future. Nice features like a barcode scanner will be easy to add to the system if someone decides they would like to have them.

6.4 Lessons Learned

Over these short six weeks, we have used the Agile development process to create real working software for a real client. We did daily scrums, worked in pairs, and used version control software to keep everyone up-to-date. We worked 40 hour weeks and had "managers" keeping track of our progress (our professors). We struggled with learning new technologies and new frameworks, and we learned how to work as a team. The entire project was such a learning experience that it's tough to list just a few things, but the following is a list of the most important lessons we learned from this project:

- Use the right tool for the job. We tried for a while to make do with GridLayout and BorderLayout, and they just weren't working out. We were familiar with them, so we tried to hang on as long as possible. When we tried GridBagLayout, we were frustrated and didn't understand it, but after working with it for a while, we found it to be a way better tool for our purposes than either of the first two layouts we had tried.

- Relying on other departments/outside groups can be difficult. We contacted CCIT with our requests within the second week of our project, but we still have nothing from them. One thing we really learned is that bureaucratic process kills speed. In the real world, things sometimes take way longer than you might like.

- The client pays your salary (figuratively for us), but they often aren't the same person as the end user, and they don't always know what the end user really needs. Dr. Moore was the client for our project, but he was not the one who would be using the system. John Jezek and the patrons of the machine shop would be the ones really using the system. We had to take into account what the client wanted, but also take into account what the end users needed. For this reason, we met Dr. Moore's requirements, but we also talked to Mr. Jezek throughout the process to see what he wanted out of the system.

7.0 Appendix

7.1 Product Installation

Using the provided Software CD,

1. Insert CD, Open CD location
2. Right-click on the ShopTracker folder and select copy. Navigate to the desired location and paste.
3. Once the ShopTracker folder is in your desired location, the application can be run by double clicking on ShopTracker.exe.
4. To create a Desktop shortcut, right-click on ShopTracker.exe and select Send To → Desktop to create.

If the executable does not run:

- Ensure the following items are within the ShopTracker folder:
    1) ShopTracker.exe
    2) mongodb
    3) jre6
    4) ShopTracker.jar

If the database appears empty and no previous data is shown once ShopTracker is running:

- Ensure the following items are within the "mongodb" folder (in ShopTracker folder):
    1) bin
    2) data (there should be a "db" folder with this location as well)
    3) log

If needed, see README.txt on the program disk in the ShopTracker folder. The document provides the same detailed Product Installation Instructions.

7.2 User Function Details

- **Select Machines** (select machines that the user will use during a given sign in period)
    - From the home screen: In the section titled "My Machines", select the machines that are going to be used during the sign in session. Once chosen, click the button "Select Machines" in the right hand button panel. A message will display providing a list of machines selected. A machine cannot be unselected once the "Select Machines" button has been clicked.
    - If at any point you do not see the "My Machines" section but wish to select machines either click "Select Machines" or the home button in the top left hand corner. Once back to the user home screen, follow the instructions above.

- **Checkout Tools** (check out a tool, meaning the user is leaving the machine shop with the tool)

    - Click the "Check Out Tools" button in the right hand side button panel. Either search by name or search by ID, and click "Go" respectively. From the results, select the tool(s) you wish to check out and click "Check Out". Follow Machine Shop procedures for finding the tools you selected.

- **Return Tools** (return a tool that was previously checked out, it must be physically returned to the machine shop)
    - From the home screen: In the section titled "Checked Out Tools", select the tools that are going to be returned during the sign in session. Once chosen, click the button "Return Tools" in the right hand button panel. The tool(s) will be removed from the checked out tools list and cannot be "un-returned" once the "Return Tools" button has been clicked.
    - If at any point you do not see the "Checked Out Tools" section but wish to return tools either click "Return Tools" or the home button in the top left hand corner. Once back to the user home screen, follow the instructions above.

- **Machine Certifications** (manage machine certifications for a given user)
    - Click the "Machine Certifications" button in the right hand side button panel. Either search by name or search by CWID and click "Go". Select the certifications the user has gained, unselect ones they have lost. Click "Save" once all changes have been made.

- **Edit Users** (add, remove, lock, or unlock a user)
    - Click the "Edit Users" button in the right hand side button panel.
        - If you want to add a user, click "Add User":
            - Fill in the necessary fields, select any certifications the user already has and then click "Save".
        - If you want to remove a user, click "Remove User":
            - Search for the user by name or CWID, select the user(s) from the results and then click "Remove Users".
        - If you want to lock a user, click "Lock User":
            - Search for the user by name or CWID, select the user(s) from the results and then click "Lock Users".
        - If you want to unlock a user, click "Unlock User":

- Search for the user by name or CWID, select the user(s) from the results and then click "Unlock Users".

- **Edit Tools** (add or remove a tool)
  - Click the "Edit Tools" button in the right hand side button panel.
    - If you want to add a tool, click "Add Tool":
      - Fill in the necessary fields, then click "Save".
    - If you want to remove a tool, click "Remove Tool":
      - Search for the tool by name or ID, select the tool(s) from the results and then click "Remove Tools".

- **Edit Machines** (add or remove a machine)
  - Click the "Edit Machine" button in the right hand side button panel.
    - If you want to add a machine, click "Add Machine":
      - Fill in the necessary fields, then click "Save".
    - If you want to remove a machine, click "Remove Machine":
      - Search for the machine by name or ID, select the machine(s) from the results and then click "Remove Machines".

- **User Privileges** (manages user permission levels ,basic vs. admin, in the system)
  - Click the "User Privileges" button in the right hand side button panel. Either search for a specific user or search all to find the user you want to change. Check the box accordingly. No checked box means basic user.

- **Generate Reports** (query the log by date, user, tool, or machine)
  - Click the "Generate Report" button in the right hand side button panel. Select one of the parameters (date, user, tool, machine). Fill in the necessary parameters, such as user CWID, tool name, machine name, and/or date. If you do not select a date range, the default is the start and end of current day. Then click "Generate". You may now view the statistics of the your result set (the first 4 tabs) and the log (the final tab). You may now export the report by clicking "Save to Excel File". This will be saved to [softwareLocation]/ReportExports/

- **View Active Users** (view all users currently signed in and those who currently have tools checked out)
  - Click the "View Active Users" button in the right hand side button panel. The table will show both current users in the machine shop and those who are not currently signed in but do have tools checked out. The columns can be resized by dragging (like one would do in excel) if all data is not visible.

- **View Tools / Machines** (view the current status of a tool or machine)
  - Click the "View Tools / Machines" button in the right hand side button panel. The top section shows machines in use (and how many users are currently in the shop to use it) and machines not in use. The bottom section shows tools checked out and tools not checked out. Use the scroll bars to see all the tools and machines.

- **Sign In Another User** (provides same functionality as the basic user home screen)

    - Click "Sign In Another User" any time you wish you stay on the machine but someone needs to sign in. You remain as the current user but can perform all the functions the a basic user needs for them. This could be useful for a student worker.
- **Start Working** (keeps a user signed in, assuming he/she is still in the machine shop)
    - Click this button if you are done using the sign in machine but are not signing out. This means you are NOT leaving the machine shop.

- **Finish** (user has finished data entry / management and he/she is going to remain in the machine shop)
    - Click this button if you are done using the sign in machine but are not signing out. This means you are NOT leaving the machine shop.

  Note: This is the same function as Start Working but is specific to when a system administrator finishes entering data.

- **Sign Out** (signs a user out, meaning the user has finished using the machine shop for the time being and he/she is leaving)
    - Click this button if you are completely done using the sign in machine as well as the machine shop. This means you ARE leaving the machine shop for the time being.

If needed, see SysAdminHelp.txt, adminHelp.txt, and basicUserHelp.txt on the program disk or on the machine in the ShopTracker folder. These documents can be referenced once inside ShopTracker by clicking the “Help” button at the bottom of any user screen.

7.3 Moving the Application to a new machine

Copy the entire ShopTracker folder through any means to the new computer.

Follow the same process as stated in the Production Installation Instructions except use the ShopTracker folder that was copied off the old machine.

Test the move by trying to run the ShopTracker executable.

If the executable does not run:

- Ensure the following items are within the ShopTracker folder:
    1) ShopTracker.exe
    2) mongodb
    3) jre6
    4) ShopTracker.jar

If the database appears empty and no previous data is shown once ShopTracker is running:

- Ensure the following items are within the mongodb folder (in ShopTracker folder):
    1) bin
    2) data (there should be a “db” folder with this location aswell)
    3) log

If needed, see README.txt on the program disk in the ShopTracker folder. The document provides detailed instructions on how to move the program to a new machine.