Senior Project Paper

Inventory Management System for Northern Michigan University's Noquemanon Trail Network Equipment Rental Program

Garrett Leach

December 2017

Contents

Introduction

I chose to create an Inventory Management System for Northern Michigan University's new Noquemanon Trail Network Equipment Rental Program for my senior project. The actual equipment rental system has not started as of writing this but the hope is that it starts within the year, as such the hope is for this codebase to be maintained even after I leave NMU. This project stemmed from working at NMU's Business Intelligence and Information Systems as a student programmer. I was tasked with making this for work and thought it would be a good thing to do for my Senior Project since I was going to be working on it anyway. There were many goals that I had for this project, chief among which was integrating a project with already existing systems.

What I Learned

There was a lot to be learned by doing a project like this. This was actually the first real website I've created and I have learned a lot in the process of making it. While not brand new to me, I still learned a lot about the languages used in this project. I was very inexperienced with PHP and JavaScript at the beginning of this project but compared to where I was then, I feel like I have grown as a developer and broadened my skillset.

One of the more relevant things that I learned was the necessity for patience and adaptation. This project relied on a few systems that were out of my control. As a student programmer, I was restricted from directly accessing some of NMU's systems, so I had to rely upon full-time staff to take time out of their work schedule to set up the systems for me. These slight delays, while not long, helped me to recognize that unlike working on a project that you have full control over, the reliance on other entities systems means that you will need the ability to be able to change what you are currently working on at any given time.

Prioritization is another skill that I have learned throughout the course of this program. Many times over the length of this project, I mixed up my priorities for the worse, starting a section that would end up taking a long time to complete even if it was not worth any points. One example of this was when I still had many items left on my points list to complete but instead I wanted to make the Bootstrap 'typeahead' feature on input fields to work. This small feature ended up taking longer than I thought it would and I should have realized earlier that

I should be working on something more important rather than continuing to try to complete the feature.

What I Would Do Differently

In a project such as this one, there are many things over time that you wish you had done differently, but cannot due to time restraints. At this point in time there is one thing that I wish I could change but it would require basically an entire re-write of the system, and that is the PHP library I chose for interacting with the MySQL database. The library, Joshcam, while decent has many pitfalls that did not show up until later along in the timeline of the project. The documentation should have signaled to me that I should use a different, more supported and popular, library. Joshcam did a good job for the more basic operations of interacting with the database, but when I required more complex operations later on in the project I had to rely, instead, on raw queries, which I would not have needed with other libraries. In terms of what I would have done differently is I should have taken more time to research the different libraries for interacting with databases and chosen the best one.

A very important aspect I would have, and should have, done differently is plan out the structure of this project. I feel that with a better design much of this project could have been done quicker and with a lot less pain. Because I waited longer than I should have to begin this project I did not take the time necessary to plan out what I should do beforehand, instead opting to just start writing code and fixing issues as they came up later.

If I could go back in time, I would tell myself not to fret over the little, not as important things. There were instances in the making of this system that I spent too long working on unimportant features that were not even worth points, but that I just wanted to complete. I would also save the HighCharts portion for last as working with raw queries for MySQL was required, and I am still unfamiliar with the more complex database commands.

Technologies Used

- PHP
- JavaScript
- MySQL
- Git
- SSO (Authenticating with NMU)
- JQuery
- Composer
- Joshcam (PHP database library)
- Bootstrap (Javascript/CSS Libary)
- HighCharts (Used for making charts)
- Typeahead (JavaScript/Bootstrap helper)
- NMU API (Customer Lookup)
- PHP-CRUD-API

I chose to use PHP for this project because it was required, but I am also glad that I used it. From what I have seen and heard over the years PHP has always been said to be a bad programming language, but even though it gets a bad rep it did what it was meant for really well: server side development, simple yet productive. The experience with PHP that I gained while creating this project will surely help me in the future, and I am looking forward to improving and refining my skill with the language.

You cannot have a website without JavaScript. JavaScript was yet another language that I am glad that I had time with to learn and improve. While it was not used as much as PHP throughout the codebase, the use of JavaScript was still prevalent. I will also be looking more into JQuery as time goes on to get more familiar with it as it is such a useful library.

One of the main parts of web development that was always off-putting to me, and why I have shied away from making websites in the past, was the styling aspect of it to make your website look nice. Luckily with Bootstrap

this became a second thought because of how easy it is to get a decent looking website with it. Bootstrap made it trivial to make a decent looking, responsive website.

MySQL, while I have worked with it before, is another deterring aspect of web development, but luckily made more manageable by a good abstraction library. Joshcam was the library I used to manage most of my database interactions and it's model implementation made it even easier to work with tables and relations. I still have much to learn when it comes to MySQL and I hope to one day be proficient with it.

PHP-CRUD-API was a library I found late into development that I would have like to have found earlier. It is a single file library that allows for easy CRUD operations. In my project I only used it sparsely and even then only used it for some reads.

Git was relatively easy to use since there is a built in feature of the PHPStorm IDE that I used to make this project.

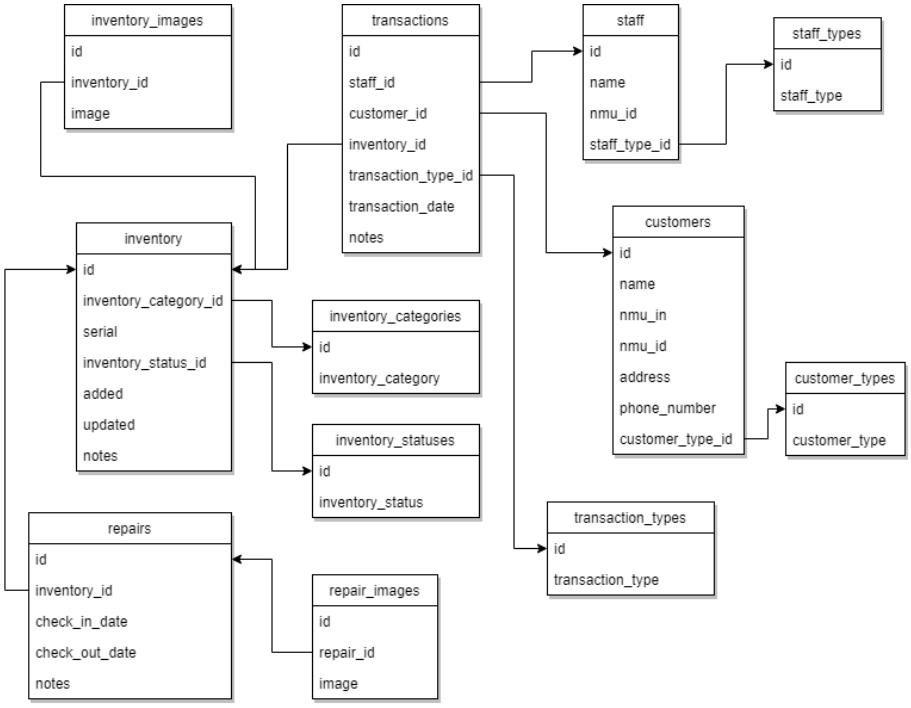
Source Organization

The source code is well organized. The main used files are at the top level, the admin files – used for table management – are located in an admin subdirectory, the stats pages are in a stats subdirectory, and some javascript files are in a js subdirectory.

Database

The database structure went though many iterations, and now looking back at it could be organized better in some parts. One part that is still in there is the phone numbers and address for the customers, which I found later wasn't really needed because the system for right now is only dealing with NMU students/staff. Instead of taking the fields out I left them in there in case they are needed in the future, in case this equipment rental system ever goes out to the general public instead of just NMU students/staff. I did not have a lot of experience with databases before this project and so I'm still not sure if this is the best solution for this system, but it is doing it's job well for now. I had a lot of difficulty with the relations portion of these tables. While they are sometimes nice, other times like in the HighCharts portion it made getting the actual in-

formation I needed really complicated, at least while using the Joshcam library, part of this is most likely my general inexperience with mysql and databases in general.



Difficulty

In terms of the actual difficulty of this project compared to what I predicted the difficulty to be is hard to say. There were some parts that were more difficult that I had anticipated and others that were a lot easier than I had thought. I thought that the SSO part of the project would be very hard as I did not have a lot of prior knowledge of what it exactly entailed going into the project, I was just told by my boss that I would be utilizing a SAML SSO for logging in workers through NMU's authorization system. It's a pretty interesting system, basically what happens is if the site visitor is not logged in it redirects to the SSO url, the visitor logs in using their NMU login and is redirected back to the login page. When redirected back the "sessid" is no longer empty and it performs some database actions to validate the user and sets the log in session variables.

At the beginning of every page there is a check for the "logged_in" SESSION, and if it is not set or False sends the visitor through the login procedure.

Code Sample 1: SSO Auth

```
if (!empty($_GET["sessid"])) {
    $NMUSSO = new mysqli('mydb.nmu.edu', $nmusso_username,
        $nmusso_password, 'Security');

    $session = $NMUSSO->real_escape_string($_GET["sessid"]);

    if ($result = $NMUSSO->query("SELECT udata FROM nmu_ssoview WHERE
        sessionid = '".$session."' LIMIT 1")) {
        while ($row = $result->fetch_assoc()) {
            $username = $row["udata"];
        }

        $result->close();
    }

    $NMUSSO->close();
    unset($NMUSSO);

    $models = $_SERVER['DOCUMENT_ROOT'] . '/galeach/models'; // Location
        of the database models
    dbObject::autoload($models);
    $staff_row = staff::where('nmu_id', $username)->getOne();

    if ($staff_row) {
        $_SESSION['staff_name'] = $staff_row->name;
        $_SESSION['staff_nmu_id'] = $staff_row->nmu_id;
        $_SESSION['staff_type'] = $staff_row->staff_type_id->staff_type;
        $_SESSION['logged_in'] = TRUE;


        header('Location: index.php');
        exit;
    }
}

header('Location: https://myuser.nmu.edu/NMUsso?sys=sso_noqrental');
```

```
exit;
```

A part that I thought would be a lot easier than it was was the HighCharts portion of the project. While the HighCharts javascript is actually straightforward and easy, actually getting all the relevant info from the database was taking me a lot of time, since I was trying to use the Joshcam PHP library for getting all the info and was running into complications. I spent way to much time getting the original diagram's data and ended up resorting to a raw query instead of using the ORM.

Code Sample 2: Getting info for HighCharts

```
    $layout = [];
    $res = $db->rawQuery("SELECT inventory_category FROM
        inventory_categories;");
    $counter = 0;
    foreach($res as $re) {
        // This matches the JSON data layout for Highcharts
        $layout[$re['inventory_category']] =
            ["y" => 0,
            "color" => $colors[$counter],
            "drilldown" => [
                "name" => $re['inventory_category'],
                "categories" => [],
                "data" => [],
                "color" => $colors[$counter]
            ]];
        $counter += 1;
    }
    $result = $db->rawQuery("SELECT DISTINCT transactions.id,
        inventory.serial, inventory_categories.inventory_category FROM
        transactions
JOIN inventory ON transactions.inventory_id = inventory.id
JOIN inventory_categories ON inventory.inventory_category_id =
    inventory_categories.id
JOIN transaction_types ON transactions.transaction_type_id = (SELECT id
    FROM transaction_types WHERE transaction_types.transaction_type =
    'check out');");

    foreach($result as $r) {
```

```
        $layout[$r['inventory_category']]["y"] += 1;
        array_push($layout[$r['inventory_category']]["drilldown"]["categories"],
            $r['serial']) - 1;
        $index = array_search($r['serial'],
            $layout[$r['inventory_category']]["drilldown"]["categories"]);
        $layout[$r['inventory_category']]["drilldown"]["data"][$index]
            += 1;
    }

    // Gets rid of the key names from $layout to match HighCharts
        requirements
    $used_data = [];
    foreach ($layout as $l) {
        array_push($used_data, $l);
    }

    // Array of category names for HighCharts
    $category_names = [];
    foreach($layout as $k => $v) {
        array_push($category_names, $k);
    }

    $category_names = json_encode($category_names);
    $used_data = json_encode($used_data);
```

Another slightly more difficult, and annoying, than anticipated portion of the project was the error checking of all the transactions. There is a ton of error checking in this project as is standard when dealing with databases. Luckily I created a rather elegant system for notifying the workers that something went wrong with any of their actions. There still may yet be some bugs that I overlooked but most of the code is organized in a way that it should be easy to find a solution. Below is the method that was used many times throughout the course of this project. It sets data for a notification bar that will pop up to provide feedback on worker actions.

Code Sample 3: Notification Bar Helper

```
function set_notification_and_exit($bar_type, $bold, $extra, $redirect =
    NULL) {
    // Bar Types: TRUE = SUCCESS, FALSE = WARNING
```

```
    if ($bar_type == TRUE || $bar_type == 'SUCCESS') {
        $_SESSION['notification'] = [$bar_type, $bold];
    } else if ($bar_type == FALSE || $bar_type == 'WARNING' || $bar_type
        == 'DANGER') {
        $_SESSION['notification'] = [$bar_type, $bold, $extra];
    }
    if ($redirect == NULL) {
        $redirect = "https://" . $_SERVER["HTTP_HOST"] .
            $_SERVER["REQUEST_URI"]; // The current page
    }
    header("Location: {$redirect}");
    exit;
}
```

Grade

- (10/10) | User Management System (Admin/Authorized Users) allowing for addition/removal of authorized workers. Also allows for logging/seeing worker actions.
- (10/10) | Use NMU API to look-up and validate students
- (20/20) | Integrate with SAML Single Sign-on to log in through NMU
- (20/20) | Responsive Displays (Everything looks good)
  - (10/10) | Desktop/Laptop
  - (10/10) | Phone/Tablet
- (20/20) | MySQL
  - (10/10) | Uses MySQL
  - (10/10) | Tables are organized
- (60/90) | Inventory
  - (10/10) | Equipment can be added and removed from the inventory system
  - (20/20) | Transaction info is stored (who, what, when)

    - (10/10) | Checking in equipment
    - (10/10) | Checking out equipment
  - (10/10) | Ability to flag (make a note) an item on check-in (damages, etc.)
  - **(0/10) | Attach and store images of equipment (proof of damage)**
  - **(10/20) | Equipment checkout**
    - (10/10) | Check if person checking out equipment (the customer) to is allowed to be checking out equipment.
    - **(0/10) | Display any flags before checking out an item**
  - (10/10) | Report what equipment is/isn't available (status of each item)
  - **(0/10) | Email notification when an item is not returned on time**
- (30/40) | Stats Page
  - (10/10) | Display most popular items – (categories)
  - (10/10) | Display what items are in for repair
  - **(0/10) | Display most frequent users**
  - (10/10) | Display which item has been checked out the most
- (10/10) | Uses Git

Total: 180/220
A: 190
B: 170
C: 150
D: 130
F: < 130

As of writing this paper I have completed 180 out of 220 points. I expect to be over 190 points for a grade of A by the time of my presentation.