Revised MizzouCheckout System

University of Missouri

CS3380: Database Applications and Information Systems

Hunter Ginther, Jakob Daugherty, Zach Dolan, Kevin Free, Michael McLaughlin, and Alyssa Nielsen (Team Lead)

13th May 2016

# Introduction

Please provide a detailed description of the problem (as defined by your client).

For many of you, these are white label apps. For example, the Logboat project can be used by any brewery. When you share the information about the problem, focus on the general case as much as possible.

Include the MIT license disclaimer here as well. *This disclaimer should be added to each of the scripts used by your system as well.* You can find disclaimer text at the following link: <http://opensource.org/licenses/MIT>

Kevin Free

* I created the inventory page for displaying all of the items. I used radio buttons to filter the inventory. Back end, this utilizes php to run similar queries in a prepared statement format and display the results. You can sort the inventory by damaged, checked in, checked out, and by the different item categories.
* From this page, you are also able to edit the condition of any particular item. It will take you to the edit page, only accessible by an admin user through the use of sessions, where you can change the condition to any of the options from the item\_condition table. A drop down menu is populated with the values for every condition. Whatever the user selects is then put into the prepared statement and an update query is performed. We decided not to allow the ability to edit critical information about items and other tables because that shouldn’t be necessary once the items are in the system correctly.
* I also created all the insert functionality. This page has a drop down menu for what table you want to insert a new entry into. Once selected, the insert page includes different php files to allow for each different table to be added to. This operates very similar to Lab 6. The admin user fills out the fields and they are thrust into a prepared statement and run on the SQL server. Proper sessions and https requirements are met on all pages and it should be fairly unbreakable due to the error checking I included.
* These pages should meet all the requirements for adding, listing, and editing entries in the tables across the database.

# ERD



# Queries

All queries used by our system are detailed below. We have provided the SQL syntax and a detailed description of the expected return of all queries.

* SQL (1): SELECT i.id AS `Item ID`, i.name AS `Item Name`,(SELECT sit.student\_id FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Student`,(SELECT sit.employee\_id FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Employee`,available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location`,(SELECT sit.checkout\_window FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Time Due Back`FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND i.location\_id = 1 ORDER BY `Availability`, `Time Due Back` ASC, i.id;
  + Return: All attributes for items located at the Memorial Student Union desk ordered by availability
* SQL (1.1): SELECT i.id AS `Item ID`, i.name AS `Item Name`,(SELECT sit.student\_id FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Student`,(SELECT sit.employee\_id FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Employee`,available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location`,(SELECT sit.checkout\_window FROM student\_item\_transaction AS sit WHERE sit.item\_id = i.id AND sit.transaction\_datetime >= CURDATE() AND sit.transaction\_type = 'Out' AND sit.checkout\_window = (SELECT MAX(sit.checkout\_window) FROM student\_item\_transaction WHERE item\_id = sit.item\_id)) AS `Time Due Back`FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND i.location\_id = 0 ORDER BY `Availability`, `Time Due Back` ASC, i.id;
  + Return: All attributes for items located at the Student Center desk ordered by availability
* SQL (2): SELECT student.name\_first, student.name\_last, student.username, student.email, item.name FROM student inner join item on student.id = item.id where student.id = ?;
  + Return: The first name, last name, username, email, and any items checked out by that student where the student id matches that of the given search term
* SQL (2.1): SELECT student.name\_first, student.name\_last, student.username, student.email, item.name FROM student inner join item on student.id = item.id where student.username = ?;
  + Return: The first name, last name, username, email, and any times checked out by that student where the student username matches that of the given search term
* SQL (2.2): SELECT student.name\_first, student.name\_last, student.username, student.email, item.name FROM student inner join item on student.id = item.id where student.name\_last = ?;
  + Return: The first name, last name, username, email, and any items checked out by that student where the student’s last name matches that of the given search term
* SQL (3.0): INSERT INTO employee (id,username, user\_type, email, salt, hashed\_password, name\_first, name\_last) VALUES (?,?,?,?,?,?,?,?);
  + Return: Inserts a new employee into the employee table
* SQL (4.0): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all items in the inventory
* SQL (4.1): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND i.item\_condition\_id > 2 ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all items in the inventory that are damaged or in unworking condition
* SQL (4.2): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND i.available = 0 ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all items in the inventory that are currently checked out
* SQL (4.3): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND i.available = 1 ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all items in the inventory that are currently available for checkout
* SQL (4.4): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND LOWER(i.name) LIKE LOWER('bike%') ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all bikes in the inventory
* SQL (4.5): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND LOWER(i.name) LIKE LOWER('mac%') ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all Macs in the inventory
* SQL (4.6): SELECT i.id AS `Item ID`, i.name AS `Item Name`, available AS `Availability`, ic.name AS `Item Condition`, l.name AS `Location` FROM item AS i, item\_condition AS ic, location AS l WHERE i.item\_condition\_id = ic.id AND i.location\_id = l.id AND LOWER(i.name) LIKE LOWER('pc%') ORDER BY i.id;
  + Return: The id, name, availability, condition, and location of all PCs in the inventory
* SQL (5.0): INSERT INTO item\_category (id, name, waiver, item\_id) VALUES (?, ?, ?, ?);
  + Return: Creates a new row in the item\_category table
* SQL (6.0): INSERT INTO item\_condition (id, name) VALUES (?, ?);
  + Return: Creates a new row in the item\_condition table
* SQL (7.0): INSERT INTO item (id, name, available, item\_condition\_id, location\_id) VALUES (?, ?, 1, 1, ?);
  + Return: Creates a new row in the item table with a default availability and condition of available and good respectfully
* SQL (8.0): INSERT INTO location (id, name, terminal\_id) VALUES (?, ?, ?);
  + Return: Creates a new desk location in the location table
* SQL (9.0): INSERT INTO waiver (id, name) VALUES (?, ?);
  + Return: Creates a new waiver in the waiver table
* SQL (10.0): SELECT salt, hashed\_password, user\_type FROM employee WHERE username=?;
  + Return: The salt, hashed password, and user type from the employee table where the username matches the given search term
* SQL (11.0): SELECT id, name FROM item\_condition ORDER BY id;
  + Return: The id, and name of all rows in the item\_condition table ordered by id
* SQL (11.1): "UPDATE item SET item\_condition\_id = ' " . $\_POST[ ' Item\_Condition ' ] . " ' WHERE name = ' " . $\_POST[ 'Item\_Name' ] . " ' ";
  + Return: Update the item condition table to the new condition id, given as a post variable, where the item name matches the given post variable.

# Analytics

What sorts of analytics are conducted by your system? Reference the specific queries used to conduct those analytics. How are these analytics useful?

# Normalization

Every table in our database has the characteristics of a 3rd normal form database. Meeting the following requirements: no repeating groups, no non-prime attribute is functionally dependent on a proper subset of any candidate key, and no candidate key contains a transitive dependency. The expired\_waiver and item\_condition\_update tables are in 3rd normal form. This lower normal form allows does not require the use of a Super Key. For the expired\_waiver table, a super key could not be used because the same student may have to sign the same waiver again, if it is expired. So to ensure that we do not have large amounts of duplicate data, the candidate key {student\_id, waiver\_id} has been used. Additionally, the item\_condition\_update table contains time stamped data that may be a duplicate of previous data. For example, throughout its lifetime, a computer my go from good condition to un-working and back again. This made using a super key for the table difficult. With the time stamp element present we have decided to make the candidate key {item\_id, datetime} in order to allow for accurate logging of condition updates. All other tables in our database have been found to be in BCNF.

# Indexing

What indexes did you create for your database? Everyone’s system should have indexes other than the default indexes created by MySQL. Explain why you created those indexes and discuss the benefits.

On our database we have defined quite a few indexes to help with data retrieval, and integrity. We use table id indexes for most personal data and all BCNF tables. Now the indexing for these tables has been defined as the MySQL default because no special indexing was required. Additionally, we have created a unique index on the employee table using the username attribute. This allows for faster login and prevents duplicate username and password combinations. Finally, we have created a auto incrementing index on the student\_item\_transaction table. This auto increment allows for searching of recent transactions fast and easy.

# Security

What security measures are taken to ensure that the system isn’t vulnerable to the types of attacks that were discussed in class. (SQL injection, XSS attacks, enforcing HTTPS, etc..). Do you hash passwords? What hashing technique do you rely on? Do you encrypt any other sensitive data? This section should provide significant detail about the security measures taken.

# Other Topics

Do you use triggers? Share anything else that needs to be shared with potential users.

# User Manual

## Installation

Step by step, how do you deploy your system. What are the software dependencies (PHP, MySQL, etc..). Do you use any additional libraries (mysqlnd)? How do you install those dependencies? How do you setup the database (Create database, then import using SQL?)? How do you host the website? By reading through this section, I should be able to deploy your system on another server without any issues/questions.

## Website Usage

Create a section for each page in the site. How do you use each page. Include screenshots in each section so there is no confusion. Organize the sections (pages) so that it follows a logical order for using the site. (i.e. Login will likely be the first section). This will be especially important if some pages in your site have dependencies. For example, if one page requires you to select a category from a dropdown menu, the user would likely have been required to create that category using a different webpage first.