Mizzou Student Union Desk Service

University of Missouri

CS3380: Database Applications and Information Systems

Justin Hofer, Holt Skinner, Jonathon Israel, Miranda Reese, Michael Kellmeyer

https://cs3380-jdhcp3.centralus.cloudapp.azure.com/project/index.php

May 13, 2016

# Introduction

Our project is focused on linking together an inventory management database with a front end service for the Mizzou Student Unions desk service. The software allows for checking in and checking out of items as well as adding/editing inventory. The backend will be a MySQL database that will contain entities as defined in the attached ERD. This will interface with a web based application. This front end will allow for a user to edit the condition of an item, add notes to an item, display all items in the system, and other features as defined later. All desk locations will interface with the same database to keep the data up to date.

# MIT License

# Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Queries

See PHP Files in “Code” Folder for more information.

**Login:** All Login-related queries are performed on the attemptlogin2.php script. The primary query utilized is

SELECT hashed\_password, username, permission\_id FROM employee WHERE username = ?;

This query returns all of the data necessary to validate the log in and determine the permissions for a particular user.

**Item Check In/Checkout:**

**View Inventory:** To populate the tables with information, a series of five queries are utilized, depending on the category chosen by the radio buttons.

SELECT \* FROM item WHERE category = 'bike'

SELECT \* FROM item WHERE category = 'PC'

SELECT \* FROM item WHERE category = 'Mac'

SELECT \* FROM item WHERE category = 'Phone Charger'

SELECT \* FROM item

In addition, the mysqli\_fetch\_field function is used to get the table headers describing the columns. The headers are currently the names of the actual tables in the database. To populate the analytics information at the top of the table, additional queries are used to separate the checked out items from the table and the mysqli\_num\_rows function is used to display the number of items.

SELECT \* FROM item WHERE category = 'bike' AND ischeckedout = 0

SELECT \* FROM item WHERE category = 'PC' AND ischeckedout = 0

SELECT \* FROM item WHERE category = 'Mac' AND ischeckedout = 0

SELECT \* FROM item WHERE category = 'Phone Charger' AND ischeckedout = 0

SELECT \* FROM item WHERE ischeckedout = 0

**Add Inventory:** To create the dynamic dropdown menus showing categories of items, the following queries are used. The results of these queries are place inside <option> tags that all allow the user to only select the values provided. This allows less opportunity for error.

SELECT name from item\_category

SELECT name from location

SELECT name from item\_condition

To perform the insert operation, the following query is used.

INSERT INTO item(id, name, category, location, notes, item\_condition) VALUES(?, ?, ?, ?, ?, ?)

Where the ? are the inputted values from the user.

**Add Location:**

To add a new location, location.php uses the query

INSERT INTO location(name) VALUES(?)

The other pages of the website that utilize location information are automatically updated with the new information. For example, the new location will appear in the dropdown menu on the add inventory page.

**Add New Student:**

The page is a standard form that is filled out by the admin to create new data that is used to identify the student in a table of many other students. The page uses an insertion SQL statement to insert information the user types into the text areas into specified areas in the database.

To insert new student data, student.php uses the query

INSERT INTO student(student\_id, username, email, name\_first, name\_last) VALUES(?, ?, ?, ?, ?)

If the information is all exactly the same, however, the data will not be sent to the database since there is already a data set for the given information, keeping the database from creating doubles

**View All Students**

This page uses the query

SELECT \* FROM student

To view all student information related to this application.

**Add New Employee**

This page works very similar to Add Student

It uses the following query to add a new user

INSERT INTO employee(username, name\_first, name\_last, employee\_id, hashed\_password, permission\_id) VALUES(?, ?, ?, ?, ?, ?)

**View All Employees**

This page uses the query

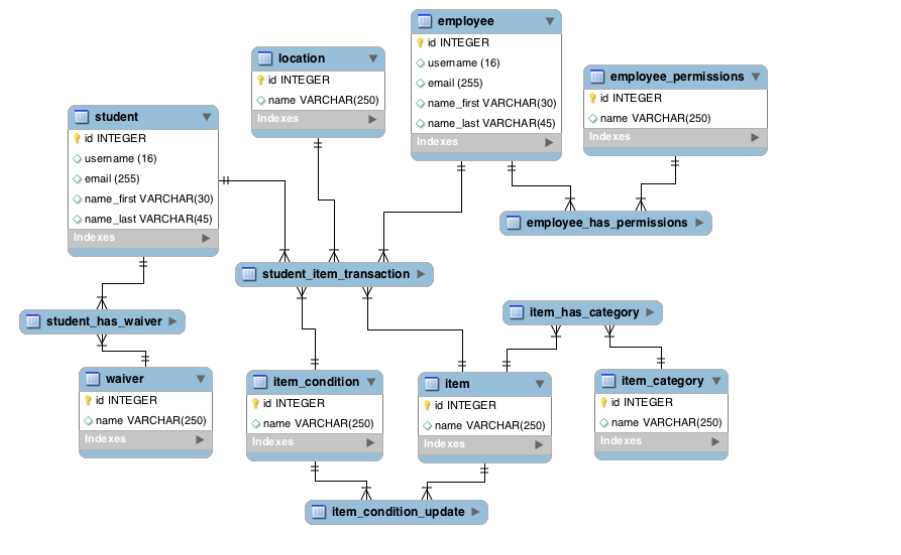
SELECT employee\_id, username, name\_first, name\_last, permission\_id FROM employee

To display employee information, without the hashed password.

We could not use the Select \* statement because that would include a hashed password for the employee. Since this would be an invasion of privacy, we instead have the select query take the ID, username, first name, last name, and permission type of each employee and print them out in a tabular format.

The page also gets the number of results and tells the admin the total number of employees with a mysqli\_num\_rows function. The table shows the names of the columns as they would appear in the sql database, getting across the point of each variable category. This table is set up so the information is both neatly shown and not showing any confidential information that could be used maliciously towards the employee, keeping the account password private.

# ERD



# SQL

For the SQL, most of our tables matched the ERD’s, plus a few elements we saw fit. The main entities were employee, student, item, along with a location, waiver, and condition entity. There were three main many-to-many tables created that required indexes for accessing important information that would most likely be used on the website. On all of the tables, constraints were added to the foreign keys for preventative measures in case an employee managed to find a way to delete matching information. First, there is an item\_has\_category table that utilized item\_id and item\_category\_id to match several different categories with the inventory items. This one was fairly straightforward, with two indexes created with each one based off of one of the foreign keys. The next table was item\_condition\_update, which will be utilized by employees upon return of an item. Here, the table utilizes item\_id, employee\_id, and condition\_id as foreign keys. This allows us to match a specific item (using the primary id key) to both an employee id that shows who made the condition update and a specific condition id. Condition id’s so far contain broken or good, but there is a separate notes section to make additional comments. Within the update table, there is also an updateTime field that keeps track of when the update was made, which could link to times the item was damaged and which student was using it at the time. Employees would also use this feature to analyze if the condition of the item has worn down due to extensive use instead of student damages. There are two indexes associated with this table, one with item\_id and one with employee\_id. Lastly, the most extensive table out of the three would be the student\_item\_transaction. There are several foreign keys for this one, sudent\_id to confirm which student is making the transaction, item\_id for which item is being checked out, condition\_id to see if the item is broken before use, location\_id to see where the item needs to be returned in case it was misplaced, and lastly, employee\_id to see which employee checked out the item to the student. For each of these foreign keys, an index has been created to view the transactions. Within this table, are additional check\_in and check\_out times with a due\_date to keep send a reminder to the student when their item is near due. Employees also have the ability to view what items have been checked out and their length of time. Other tables include student\_has\_waiver which checks if the student has signed either the physical paper waiver for bikes or the electronic waiver for laptops and chargers with a Boolean to check if it is filed and/or expired and an employee\_permissions table so that admins can enable or disable whether or not employees can make changes.

# Analytics

On the inventory page, the number of each type of item is shown at the top of the table listing the items. It also shows the number of items that are checked out and available, and it can be expanded to show more data, if needed. All of the “View” pages use similar analytics to describe the data.

# Normalization

All of the tables are in BCNF.

# Security

# Our project never directly allows users to query the database, instead relying on prepared statements and sanitized input. When registering new accounts, the program will ensure that both the password and username do not contain invalid characters, and that the username does not already exist. This prevents users from creating a username or password with invalid characters, then a sanitized version being used as the value. If this were to occur, users may not be able to access the account. All values entered by the user go through a mysqli bind parameter statement which keeps users from using integers in string data fields and vice versa. We also utilize drop down menus when new data is entered that should remain consistent, such as conditions and locations. Our project will also not output database errors to the user, instead giving bootstrap errors that inform the user of what went wrong. Pre-emptive queries also prevent the database from erroring out. HTTPS is enforced on all pages, and server will redirect to login if the user is not logged in.

# Auxiliary Features

Program allows users to create a printable Code 128 barcode of entered alphanumeric string. Barcode exists as an image, so it can be copied and edited by an outside program if wished. This will allow client to add new items with ease, and to add/replace barcodes on their existing inventory with ease.

Program will also automatically switch to the barcode field from the id field after that field is changed. This feature was requested by the client to allow users to speed up the checkout process. Users will now be able to go to the checkout page and immediately swipe the student’s card then scan the barcode of the item without the hassle of clicking on the page multiple times. A button allows this feature to be turned off. This way if the id card peripheral is broken or removed, it will still be easy to checkout and check in items.

# User Manual

## Installation

The system is hosted on a Microsoft Azure Ubuntu Virtual Machine using PHP 5 and MySQL as well as the mysqlnd library. The database can be created by running the MySQL dump file finalproject.sql. First, log in to MySQL and crate a database called ‘group\_database’ then upload the finalproject.sql file into the home directory and run the following command.

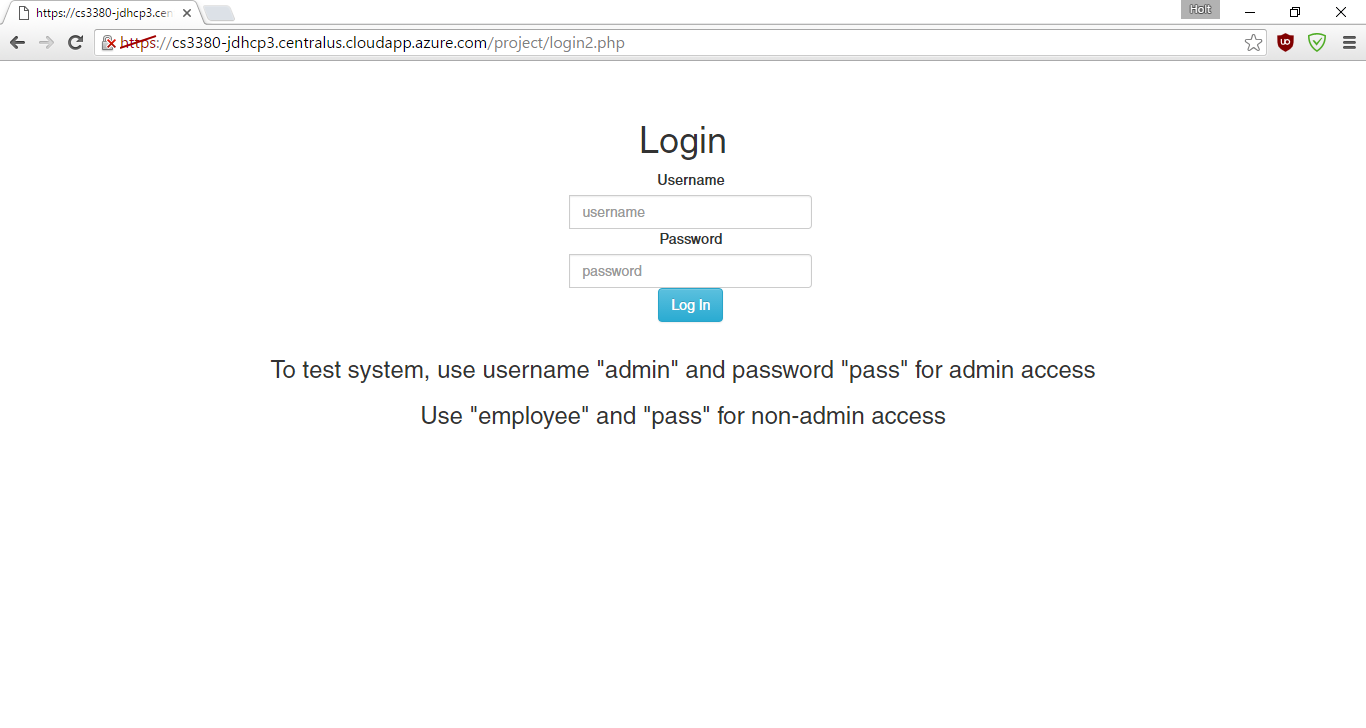
sudo mysql -u root -p group\_database < finalproject.sql

Copy all of the php and html files into the public\_html directory on the virtual machine.

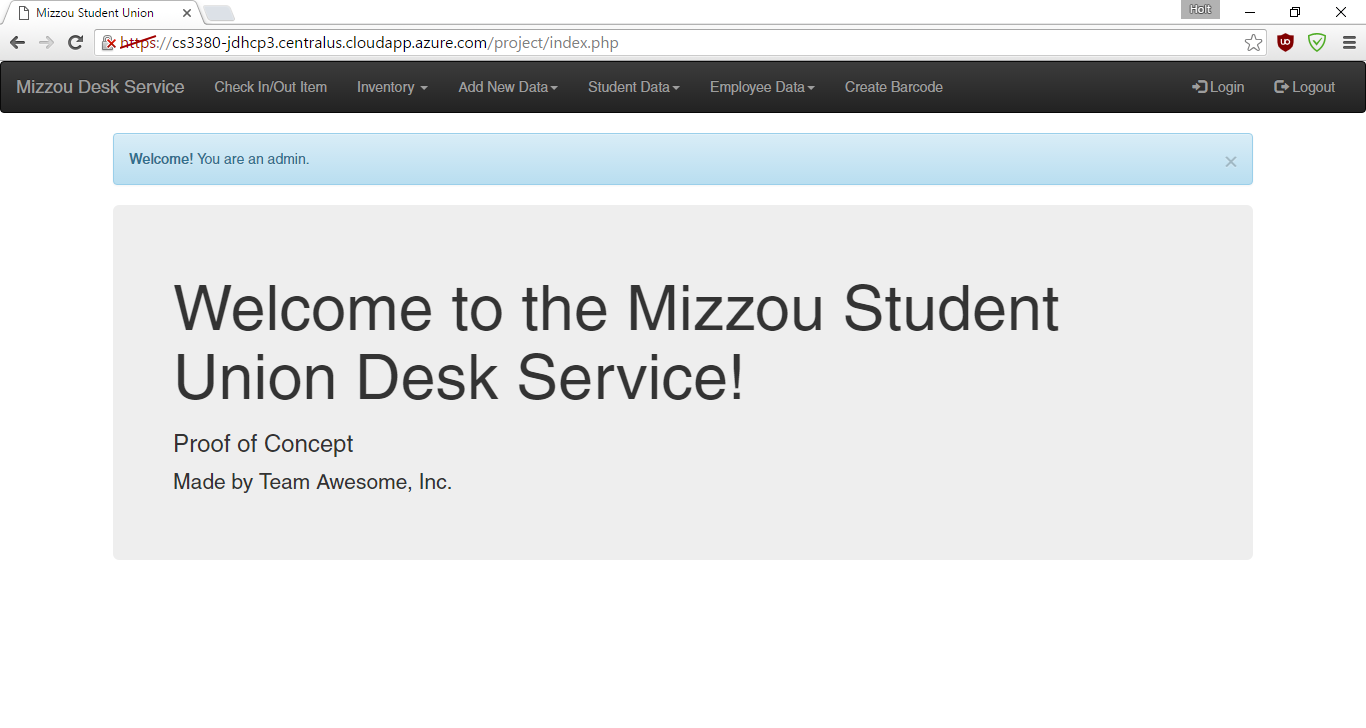
The php files that query the database will also have to be modified based upon the username and password created for mysql. The lines containing the function mysqliconnect will need to be altered. See the PHP documentation for more information.

## Website Usage (For the Client)

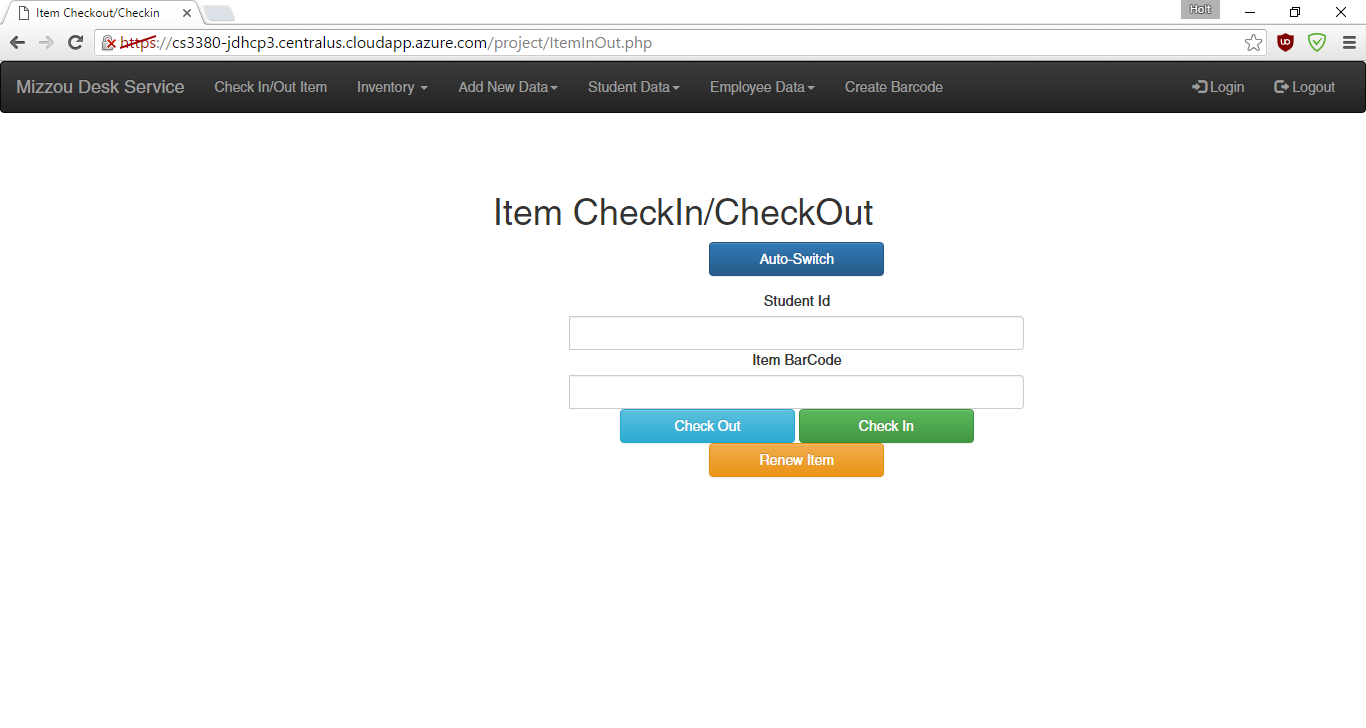
**Login:** The first page that is accessed is the login screen. To test the system, follow the on screen instructions to login as a guest user. Enter a username and password and click log in. If an incorrect username or password is entered or there is an internal error, a colored banner will appear prompting the user to re-enter information.



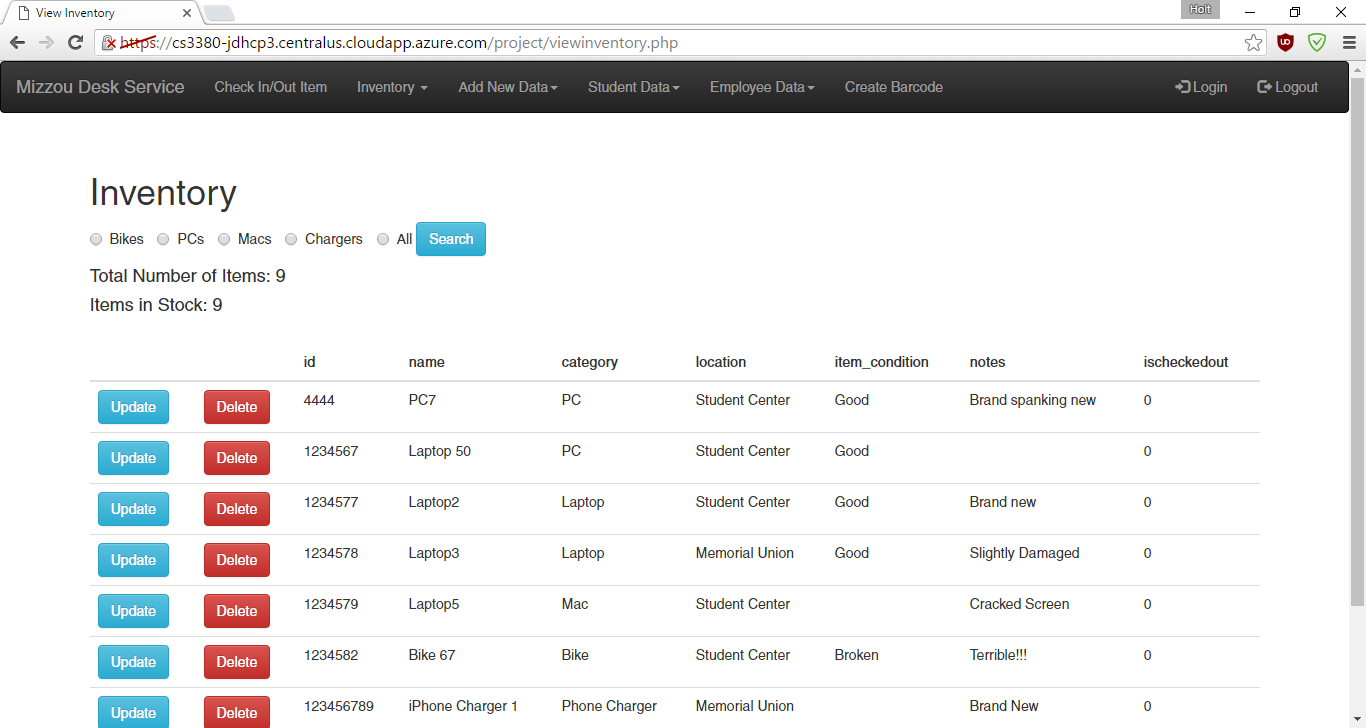
**Home:** This page will appear different if the current user is an administrator or a regular employee. A blue alert will appear if the user is an admin and more options will be available on the navigation bar. Non-admin users will only be able to check in or checkout items and view/edit the inventory. The pages will be described in order on the bar. The link “Mizzou Desk Service” will return to this screen.



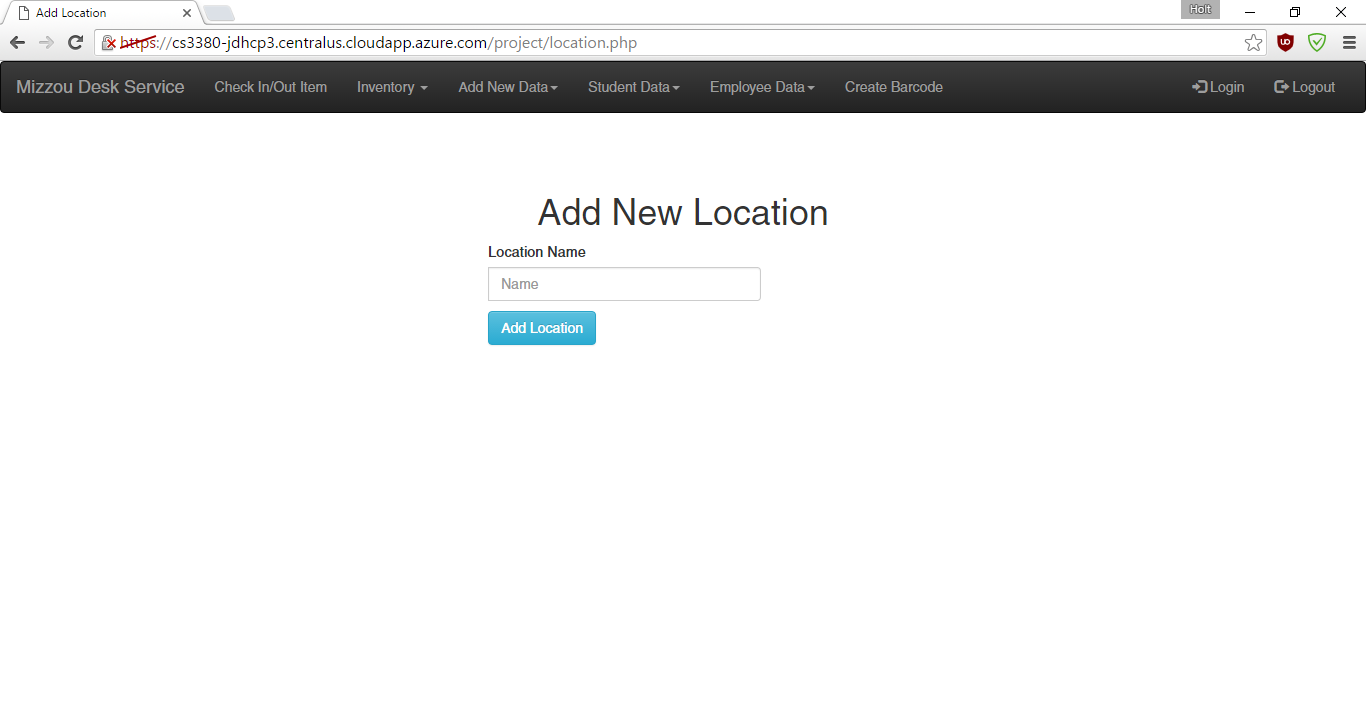
**CheckIn/CheckOut:** This page allows an employee to check in or check out an item. Simply click on the box labeled “Item BarCode” and scan the barcode. Then, if you are checking out an item, click the “student id” text field and swipe the student id. You do not have to enter a student ID if you are renewing or checking in an item. Click Check in, Check out, or Renew depending on the transaction. The system will automatically detect if the correct waiver has been filed and will show an alert if it has not been filed. A message will display if it completed successfully.



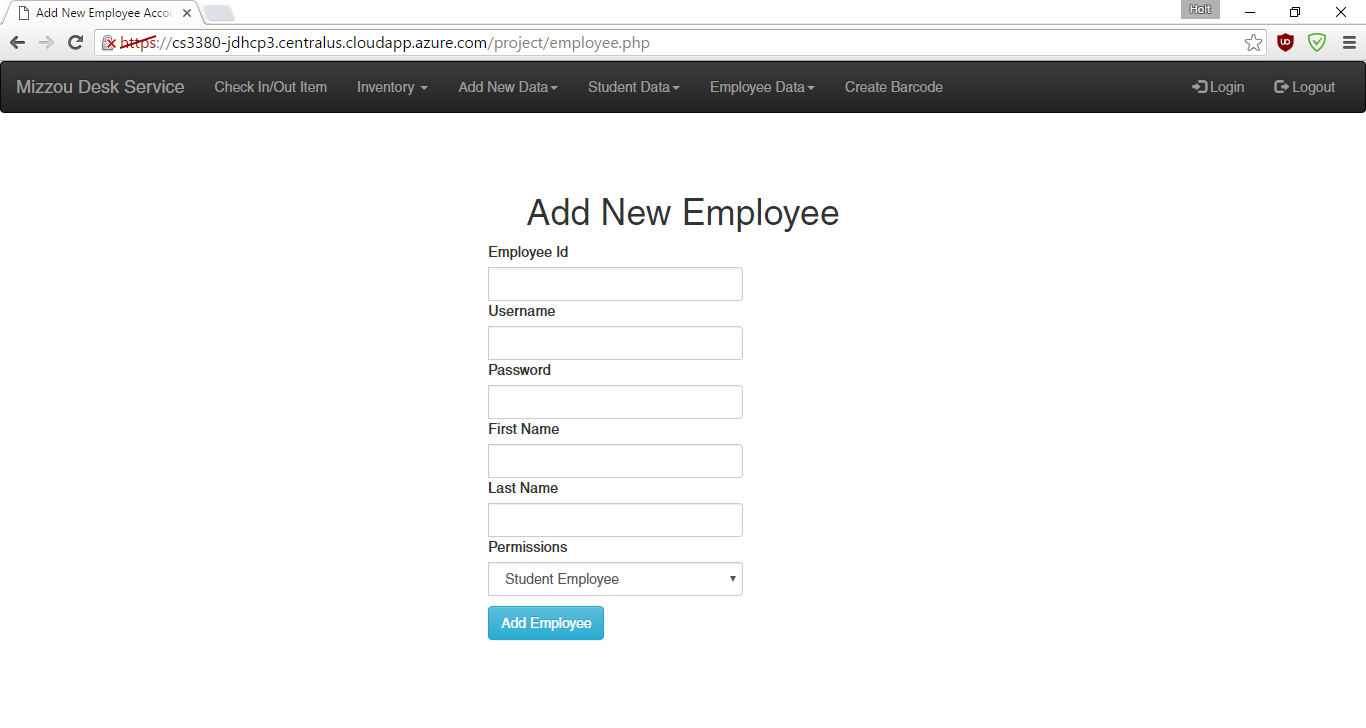
**Inventory:** This link is a dropdown menu on the navigation bar, it provides a link to view inventory or add a new item. On the view inventory page, select the category of item to view and click search to view all information about the items. To change information about an item or delete an item from inventory, select the “Update” or “Delete” button respectively next to the chosen item. If an item is deleted, a notification will be shown with the item id and if the deletion completed successfully. If an item is updated, a new screen will be shown with prompts to change the item information. On the add new item page, input the correct information into the form and press “Add Item.” A conformation will be shown if the addition completed correctly.



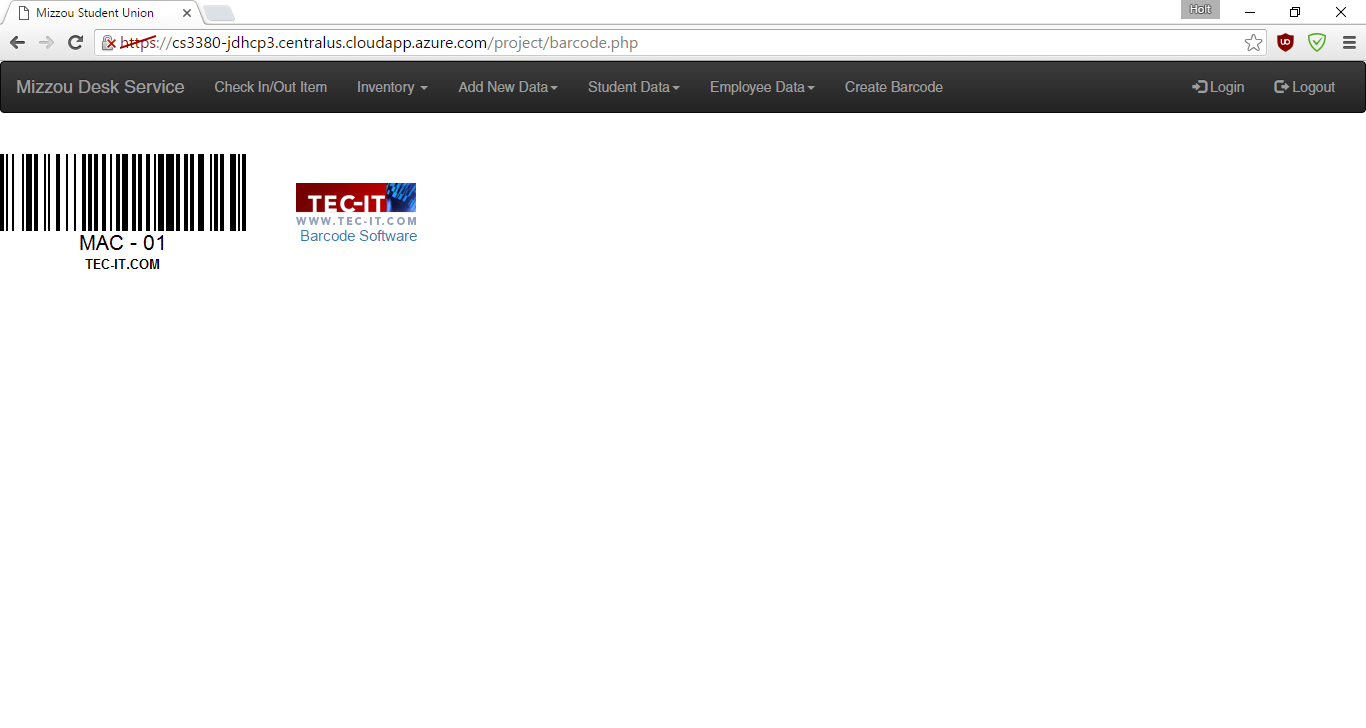
**Location:** This page allows the administrator to add a new desk location. Input the location name and press “Add Location.” It will automatically be added to all menus where location information is needed.



**Student Data/Employee Data:** See Add and View Inventory for information, these pages are very similar. On Employee Data, the Administrator users are able to make a new employee a regular employee or an administrator. Administrators have extra access to information stored on this website, be selective about administrative users.



**Barcode:** This page allows the user to create a barcode for any new item created. Simply type the barcode characters desired (e.g. MAC- 01), and a barcode image will be generated. This can be printed out and placed on the new item.



**Logout:** This button will log out the user and return to the login screen. The user will also be automatically logged out when the browser is closed.