C868 – Software Capstone Project Summary

Task 2 – Section C

Capstone Proposal Project Name: Tool Calibration Log

Student Name: ____

Table of Contents

Table of Contents

Task 2 Part C – C868 Software Development Capstone

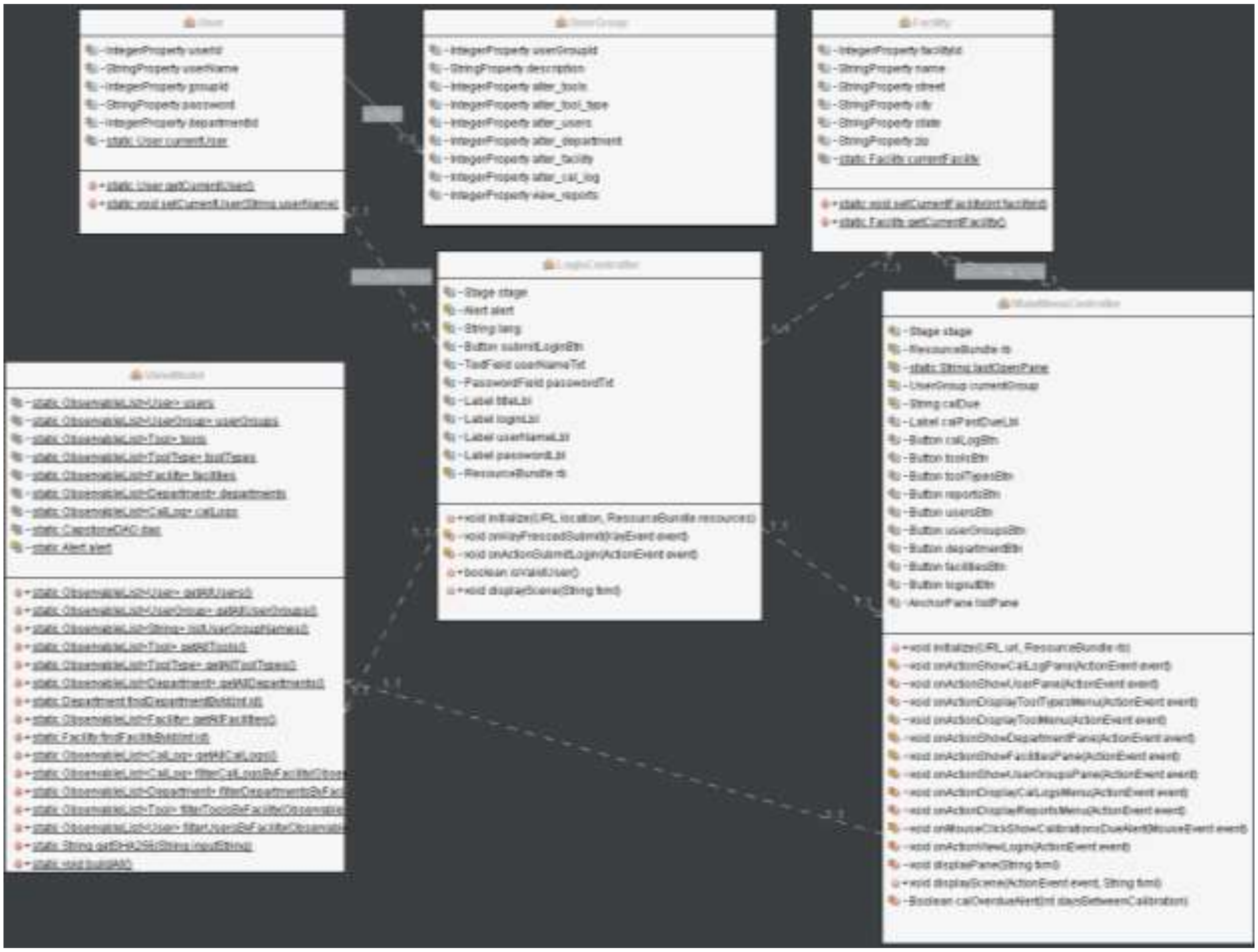
Application Design and Testing

Design Document

Class Design

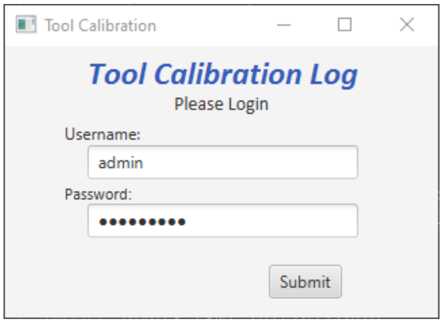
Below is a UML diagram of the classes responsible for creating the secure login, security groups, and filtering by site required by WWS. The LoginController class contains the method isValidUser which is triggered upon Username and Password submission. This method makes a call to the method getSHA256 within the ViewModel class where the password string is hashed using the SHA256 algorithm before being queried in the database.

Upon validation, setCurrentUser is called from the User class and setCurrentFacility is called from the Facility class. The MainMenuController makes features of the application accessible depending on the User Group assigned to that currentUser. As the desired feature is selected, the getCurrentFacility method is used in the filter[model]ByFacility method of the ViewModel class. The tables are then populated with only relevant objects for their facility.


User
-IntegerProperty userId
-StringProperty userName
-IntegerProperty groupId
-StringProperty password
-IntegerProperty departmentId
-static User currentUser
+static User getCurrentUser()
+static void setCurrentUser(String userName)
UserGroup
-IntegerProperty userGroupId
-StringProperty description
-IntegerProperty alter_tools
-IntegerProperty alter_tool_type
-IntegerProperty alter_users
-IntegerProperty alter_department
-IntegerProperty alter_facility
-IntegerProperty alter_cal_log
-IntegerProperty view_reports
Facility
-IntegerProperty facilityId
-StringProperty name
-StringProperty street
-StringProperty city
-StringProperty state
-StringProperty zip
-static Facility currentFacility
+static void setCurrentFacility(int facilityId)
+static Facility getCurrentFacility()
LoginController
-Stage stage
-Alert alert
-String lang
-Button submitLoginBtn
-TextField userNameTxt
-PasswordField passwordTxt
-Label titleLbl
-Label loginLbl
-Label userNameLbl
-Label passwordLbl
-ResourceBundle rb
+void initialize(URL location, ResourceBundle resources)
-void onKeyPressedSubmit(KeyEvent event)
-void onActionSubmitLogin(ActionEvent event)
+boolean isValidUser()
+void displayScene(String fxml)
MainMenuController
-Stage stage
-ResourceBundle rb
-static String lastOpenPane
-UserGroup currentGroup
-String calDue
-Label calPastDueLbl
-Button calLogBtn
-Button toolsBtn
-Button toolTypesBtn
-Button reportsBtn
-Button usersBtn
-Button userGroupsBtn
-Button departmentBtn
-Button facilitiesBtn
-Button logoutBtn
-AnchorPane listPane
+void initialize(URL url, ResourceBundle rb)
-void onActionShowCalLogPane(ActionEvent event)
-void onActionShowUserPane(ActionEvent event)
-void onActionDisplayToolTypesMenu(ActionEvent event)
-void onActionDisplayToolMenu(ActionEvent event)
-void onActionShowDepartmentPane(ActionEvent event)
-void onActionShowFacilitiesPane(ActionEvent event)
-void onActionShowUserGroupsPane(ActionEvent event)
-void onActionDisplayCalLogsMenu(ActionEvent event)
-void onActionDisplayReportsMenu(ActionEvent event)
-void onMouseClickShowCalibrationsDueAlert(MouseEvent event)
-void onActionViewLogin(ActionEvent event)
-void displayPane(String fxml)
+void displayScene(ActionEvent event, String fxml)
-Boolean calOverdueAlert(int daysBetweenCalibration)
-static ObservableList<User> users
-static ObservableList<UserGroup> userGroups
-static ObservableList<Tool> tools
-static ObservableList<ToolType> toolTypes
-static ObservableList<Facility> facilities
-static ObservableList<Department> departments
-static ObservableList<CalLog> calLogs
-static CapstoneDAO dao
-static Alert alert
+static ObservableList<User> getAllUsers()
+static ObservableList<UserGroup> getAllUserGroups()
+static ObservableList<String> listUserGroupNames()
+static ObservableList<Tool> getAllTools()
+static ObservableList<ToolType> getAllToolTypes()
+static ObservableList<Department> getAllDepartments()
+static Department findDepartmentById(int id)
+static ObservableList<Facility> getAllFacilities()
+static Facility findFacilityById(int id)
+static ObservableList<CalLog> getAllCalLogs()
+static ObservableList<CalLog> filterCalLogsByFacility(Obse
+static ObservableList<Department> filterDepartmentsByFaci
+static ObservableList<Tool> filterToolsByFacility(Observable
+static ObservableList<User> filterUsersByFacility(Observable
+static String getSHA256(String inputString)
+static void buildAll()
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1

UI Design

In the below images, Figure 1 demonstrates how the Login page looks to the user upon first opening the application. Figure 2 is the Main Menu screen that the user is greeted with when successfully logged in. This also shows the Calibration Log pane in view with data populated in the table.



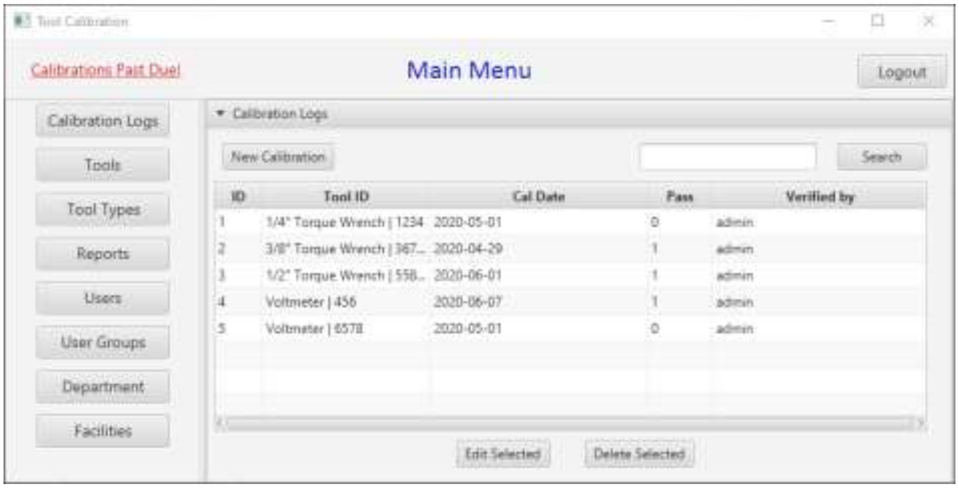
Tool Calibration Log

Please Login

Username: admin

Password: •••••••••

Figure 1: High-Fidelity Login Screen



ID	Tool ID	Cal Date	Pass	Verified by
1	1/4" Torque Wrench | 1234	2020-05-01	0	admin
2	3/8" Torque Wrench | 367...	2020-04-29	1	admin
3	1/2" Torque Wrench | 558...	2020-06-01	1	admin
4	Voltmeter | 456	2020-06-07	1	admin
5	Voltmeter | 6578	2020-05-01	0	admin

Figure 2: High-Fidelity Main Menu Screen with Calibration Log Table Active

Unit Test Plan

Introduction

Purpose

During the login process it is vital to determine that the correct username and password is being used for comparison. The password is especially critical because it needs to be hashed correctly prior to comparison.

Overview

In the LoginController class, the submit button triggers the isValidUser method which compares the username and password to all records from the user table in the database. The passwords are hashed using the SHA-256 algorithm, therefore, the isValidUser method calls ViewModel.getSHA256() to hash the password field prior to comparison.

Test Plan

Items

Two text fields within the LoginController class are used for username and password input. An ObservableList of User objects populated from the database is also used for validation by comparing the username and password properties for each User.

Features

During the login process ViewModel.getSHA256() is called against the passwordTxt field value to generate the SHA-256 hash. A for loop is called to iterate through each User object in the ObservableList generated by ViewModel.getAllUsers(). Each user in the list calls getUserName() for comparison to the plaintext username input and getPassword() is compared to the password input hash value.

Deliverables

The test produces console-based text output to show what information is produced from the text field compared to what information the ObservableList iterations produced.

Tasks

To run the test the System.out calls in the isValidUser() method of the LoginController class must be uncommented. The application can then be run in Netbeans by hitting F6 and inputting user login info. Clicking submit triggers the test.

Needs

Requirements for this test involves two primary applications:

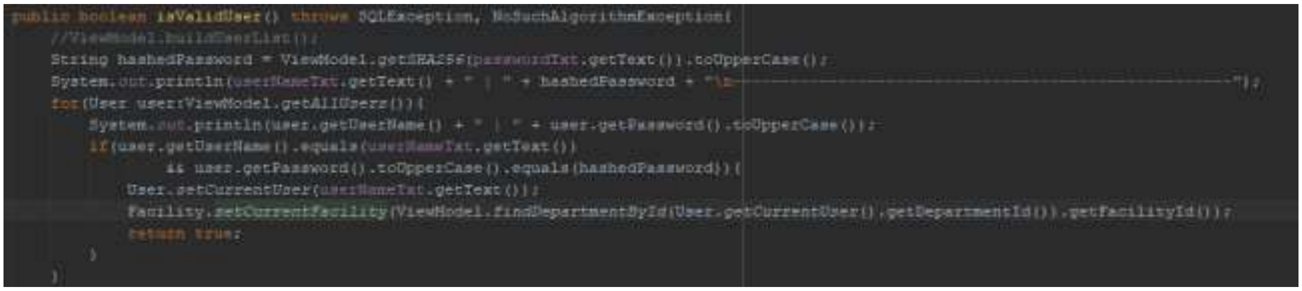
1. Netbeans 8.2 is needed to conduct testing.
2. MySQL Workbench for inserting test user.
3. SHA256 Hash Generator (https://passwordsgenerator.net/sha256-hash-generator/) for validating hash.

Pass/Fail Criteria

The result is considered a success when the output from the input fields matches the known username and password from the database.

Specifications

Below is a screenshot of the test code. This method is contained within the LoginController class.



```
public boolean isValidUser() throws SQLException, NoSuchAlgorithmException{
    //ViewModel.buildUserList();
    String hashedPassword = ViewModel.getSHA256(passwordTxt.getText()).toUpperCase();
    System.out.println(userNameTxt.getText() + " | " + hashedPassword + "\n-------------------------------------------------------------------");
    for(User user:ViewModel.getAllUsers()){
        System.out.println(user.getUserName() + " | " + user.getPassword().toUpperCase());
        if(user.getUserName().equals(userNameTxt.getText())
                && user.getPassword().toUpperCase().equals(hashedPassword)){
            User.setCurrentUser(userNameTxt.getText());
            Facility.setCurrentFacility(ViewModel.findDepartmentById(User.getCurrentUser().getDepartmentId()).getFacilityId());
            return true;
        }
    }
```

Procedures

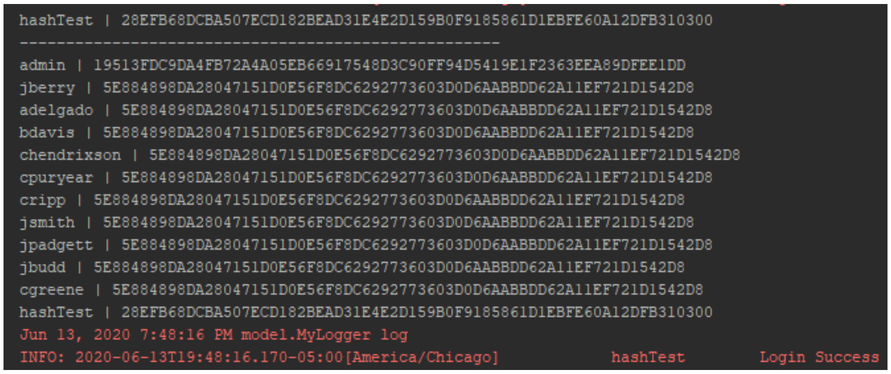
The procedures I used for performing this test are as follows:

1. Generated a SHA256 hash for the string “P@ssword”.
2. Using MySQL Workbench, I connected to the database and created a new entry for the user table (username: hashTest, password: <hash from step 1>.
3. I uncommented the System.out calls from isValidUser().

4. Started the application and entered hashTest and P@ssword in the input fields, then hit submit.
5. Verified that the output on the top line matched the information returned from the database.

Results

The results are output to the console to view what information was pulled from user input and what data was pulled from the database for comparison.



```
hashTest | 28EFB68DCBA507ECD182BEAD31E4E2D159B0F9185861D1EBFE60A12DFB310300
----------------------------------------------------
admin | 19513FDC9DA4FB72A4A05EB66917548D3C90FF94D5419E1F2363EEA89DFEE1DD
jberry | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
adelgado | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
bdavis | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
chendrixson | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
cpuryear | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
cripp | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
jsmith | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
jpadgett | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
jbudd | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
cgreene | 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
hashTest | 28EFB68DCBA507ECD182BEAD31E4E2D159B0F9185861D1EBFE60A12DFB310300
Jun 13, 2020 7:48:16 PM model.MyLogger log
INFO: 2020-06-13T19:48:16.170-05:00[America/Chicago]           hashTest        Login Success
```

C4. Source Code

The attached zip “capstone.zip” contains the source code for the application.

C5. Link to Live Version

The attached zip capstone_live.zip contains the live version. Once extracted run “Capstone.jar” and use the username “admin” and password “Password1” to gain access to the application. The live application is connected to a cloud-based MySQL instance, so no configuration is necessary for the tester.

Application Maintenance Guide

Installation and Debugging of the Application

The purpose of this section is to empower the user to be able to add features or alter the source code for the purposes of bug fixes. This tutorial can be used for Windows, Linux, or Mac.

*Note: The default Global Admin account is '**admin**' and the password is '**Password1**'. This is generated in step 4 below.*

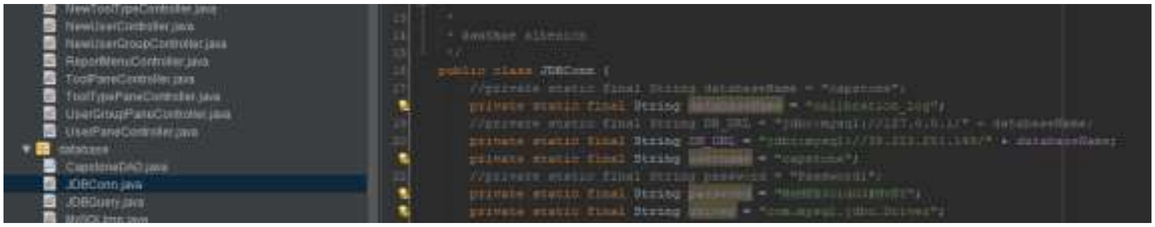
Prerequisites:

- Netbeans 8.2
- Java JDK 8
- Scene Builder (recommended)
- MySQL DBMS

Installation:

1. Open Netbeans and navigate to File > Import Project > from Zip.
2. Browse for the Zip file containing the source code (NickAlbers_Capstone.zip).
3. If the Folder: directory looks agreeable, click Import.
4. In the root directory of the Zip there are 2 SQL scripts that need to b e run. The first is ***DDL DB Build.sql*** which builds the database schema and the second is ***Initial DB population.sql*** which populates the database.
5. The final step is to tie it all together by altering the JDBConn.java class in the database Package. The recommendation is to uncomment the first DB_URL String line with 127.0.0.1 (if not localhost change to IP of MySQL host machine) and then

commenting the DB_URL below it.



NewToolTypeController.java
NewUserController.java
NewUserGroupController.java
ReportMenuController.java
ToolPaneController.java
ToolTypePaneController.java
UserGroupPaneController.java
UserPaneController.java
database
CapstoneDAO.java
JDBConn.java
JDBQuery.java
MySQLImp.java

```
 *
 * @author [illegible]
 */
public class JDBConn {
    //private static final String databaseName = "capstone";
    private static final String databaseName = "calibration_log";
    //private static final String DB_URL = "jdbc:mysql://127.0.0.1/" + databaseName;
    private static final String DB_URL = "jdbc:mysql://[illegible]/" + databaseName;
    private static final String username = "capstone";
    //private static final String password = "Password1";
    private static final String password = "[illegible]";
    private static final String driver = "com.mysql.jdbc.Driver";
```

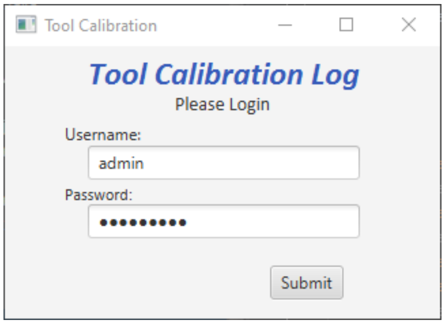
User Guide

Introduction

This guide is designed to give the user a firm understanding on how to login, administer accounts, tools, and facilities, as well as logging calibrations and pulling reports.

Login Steps

1. Click the Capstone.jar application in the Capstone Folder.
2. Enter your credentials in the Username and Password boxes and click Submit.



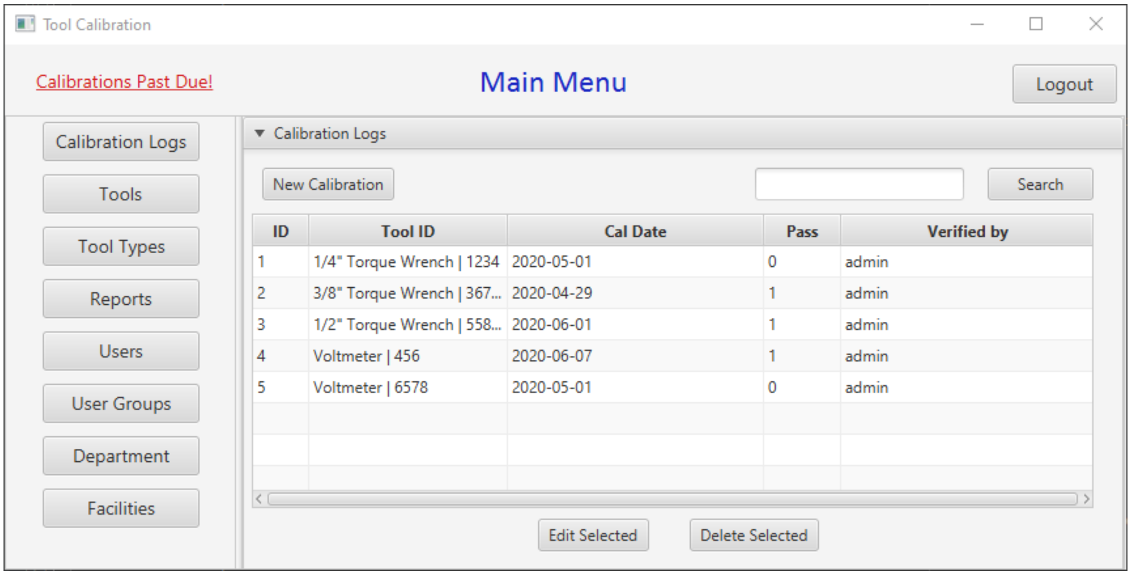
Username: admin

Password: ●●●●●●●●●

3. As of version 1.0 of the application, the users must request an account to be made and password created/reset by either the local administrator from their Facility or a Global Admin.

Navigating the Main Menu

The main menu is broken into two main components that are used to navigate the application.



ID	Tool ID	Cal Date	Pass	Verified by
1	1/4" Torque Wrench | 1234	2020-05-01	0	admin
2	3/8" Torque Wrench | 367...	2020-04-29	1	admin
3	1/2" Torque Wrench | 558...	2020-06-01	1	admin
4	Voltmeter | 456	2020-06-07	1	admin
5	Voltmeter | 6578	2020-05-01	0	admin

Navigation Bar

The navigation bar resides on the left and is used to traverse the applications many submenus. Clicking on each of the Buttons down the column will trigger the proper table with options on the right.

Table View

The table view is positioned on the right-hand side of the main menu and contains a quick view of the objects that exist in that category along with some information related to them. Each table view will have 3 options available:

1. **Create New**

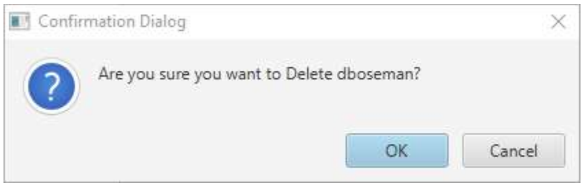
   This button is located at the top left of the table and will give bring up a new screen to input the necessary information to create a new user, tool, department, etc. Each screen will be slightly different and will be show in the following sections.

2. **Edit Selected**

This button is located at the bottom center of the table view. The first step to using this is to select one of the rows in the table first and then clicking this button. The screen that appears will be nearly identical to the Create New screen with the difference being that the screen is prepopulated with that selected object's information.

3. **Delete Selected**

   This button is also located at the bottom of the table view. When selecting on item from the table this button will allow the removal of that object from the list. It will bring up a verification dialog to ensure this action was not by mistake.



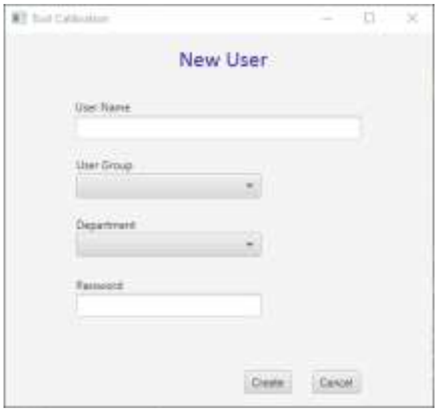
Are you sure you want to Delete dboseman?

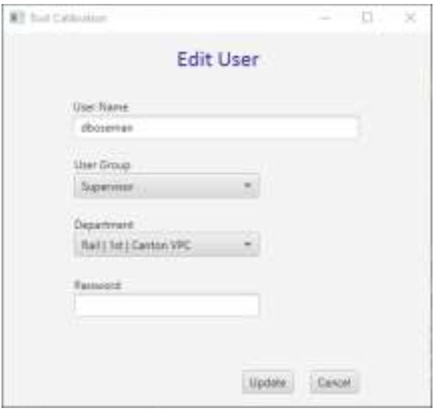
Managing Users

This option is only available to users in the Admin and Global Admin groups.

1. **New User**

   The Create New User button brings up a screen that allows the admin to enter a user's Username, User Group, Department, and Password. When the password is entered it is securely hashed with SHA-256 to ensure password security is maintained. Clicking Create will add the new user to the roster on the main menu.





User Name: dboseman

User Group: Supervisor

Department: Rail | 1st | Canton VPC

Password:

2. **Edit User**

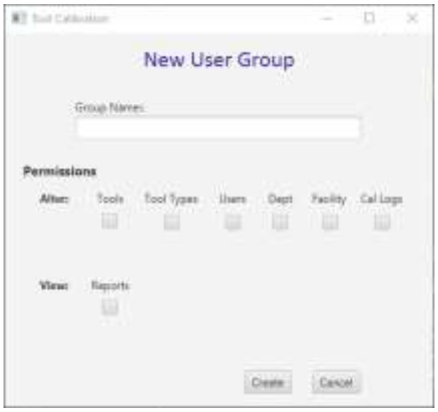
Clicking a user from the list and clicking the Edit Selected will bring up a similar page to the New User. This time the fields will be prepopulated with the selected user's information. The Password field will be blank and will only change the user's password if something is typed in that field.

Managing User Groups

This option is only available to users in the Admin and Global Admin groups.

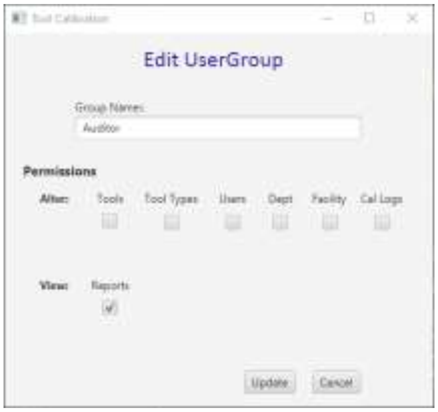
Add/Edit User Group

Within this menu the admin has a list of check boxes that align with sections of the application that can be granted or restricted for the group. In the example below, the Auditor group can only view reports and has no access to edit any data.



Alter: Tools □ Tool Types □ Users □ Dept □ Facility □ Cal Logs □

View: Reports □



Group Names: Auditor

Permissions

Alter: Tools □ Tool Types □ Users □ Dept □ Facility □ Cal Logs □

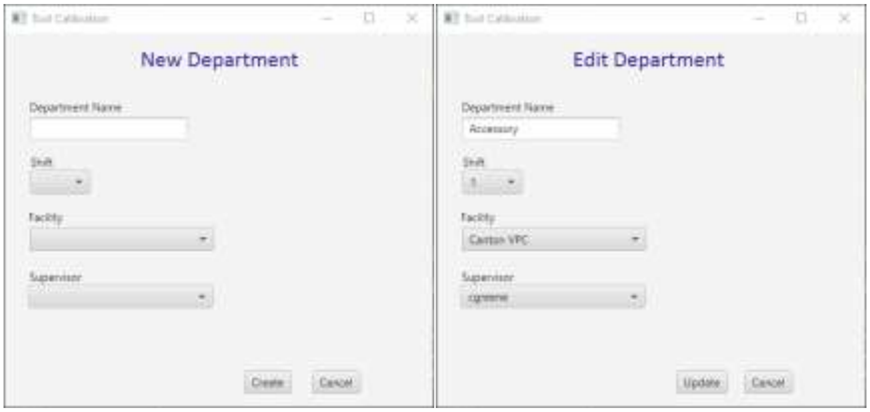
View: Reports ☑

Managing Departments

This option is only available to users in the Admin and Global Admin groups.

Add/Edit Department

Within the department screen the admin can assign a name, working shift (1 st, 2nd, 3rd), Facility, and the supervisor of that Department.



New Department

Department Name

Shift

Facility

Supervisor

Edit Department

Department Name: Accessory

Shift: 1

Facility: Canton VPC

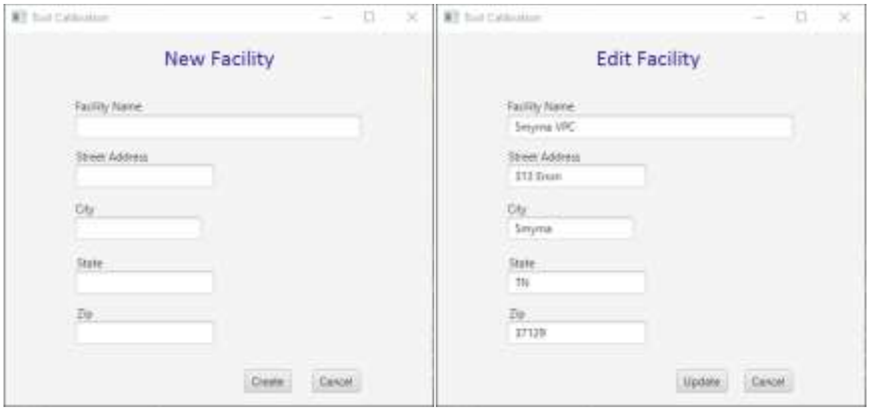
Supervisor: cgreene

Managing Facilities

This option is only available to Global Admins.

Add/Edit Facilities

This menu is very straight-forward and consists of basic address information for the given Facility.



New Facility

Facility Name

Street Address

City

State

Zip

Edit Facility

Facility Name: Smyrna VPC

Street Address: 513 Enon

City: Smyrna

State: TN

Zip: 37129

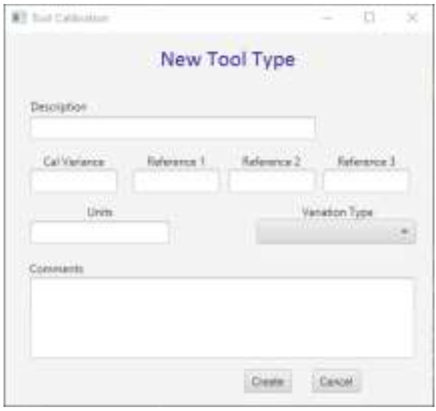
Managing Tool Types

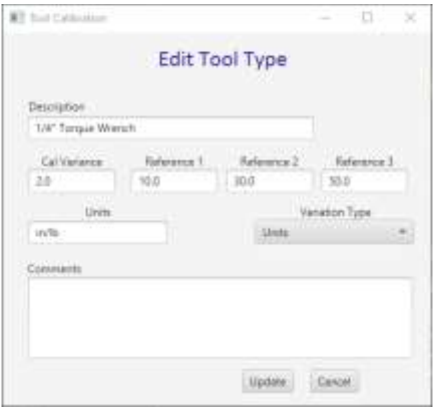
The tool types view is less restrictive than the previous sections and can be altered by the Supervisor group in addition to the Admin groups.

Add/Edit Tool Types

The tool type screen allows the user to define the testing parameters required on that tool. The most important fields to note here are the following:

- **Variation Type** – Defines whether the measured values of calibration are based on Units of measure (within 1 psi on a tire pressure gauge) or percentage (within 2% of the actual value).
- **Cal Variance** – This field defines the actual tolerance allowed to pass calibration. If percent is used in Variation Type field then this value is represented as a decimal (1 is 100%, 0.05 is 5%).
- **Reference 1,2,3** – These are the 3 known values that the tool is measured against.
- **Units** – Lists the unit of measure that the values will be recorded in.





Description: 1/4" Torque Wrench

Cal Variance: 2.0

Reference 1: 10.0

Reference 2: 30.0

Reference 3: 50.0

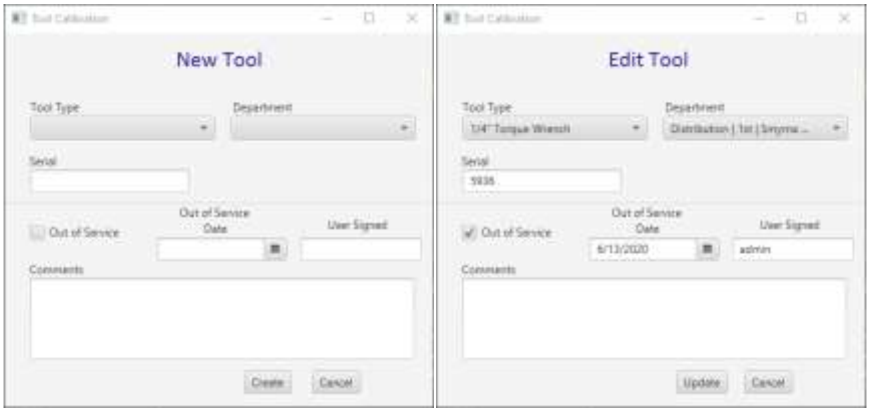
Units: in/lb

Variation Type: Units

Managing Tools

Add/Edit Tool

The tool menu allows the user to select a Tool Type that's been defined, an owning department, and the tool's serial number as the identifier. This menu also allows the user to take a tool out of service when it is no longer able to maintain calibration. The date will automatically choose the current date if a date is not chosen. The User Signature is autofilled with the currently logged in user.



New Tool

Tool Type:

Department:

Serial:

□ Out of Service

Out of Service Date:

User Signed:

Comments:

Edit Tool

Tool Type: 1/4" Torque Wrench

Department: Distribution | 1st | Smyrna ...

Serial: 5936

☑ Out of Service

Out of Service Date: 6/13/2020

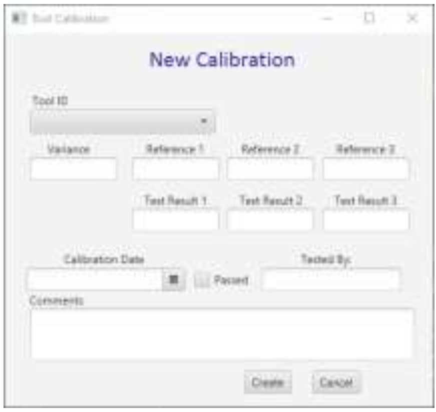
User Signed: admin

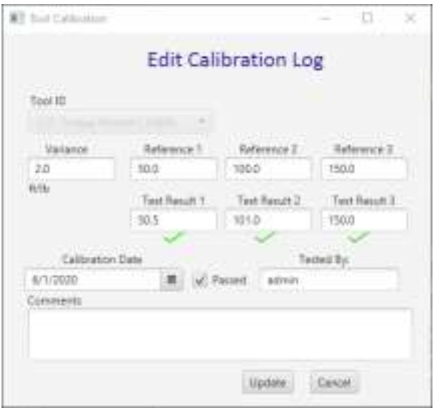
Comments:

Entering Calibration Logs

Add/Edit Calibration Logs

The Calibration screen gives the user a menu to select the proper tool from. Once selected the Variance and 3 Reference fields populate from the Tool Type of the selected. When the calibrator enters the tested values into the Results boxes a comparison is made. If that test passed it receives a green checkmark, assisting the calibrator improve testing efficiency and reducing chances for error. Once 3 checks are achieved the user can enter the date, flag passed, and submit.





Variance: 2.0 ft/lb

Reference 1: 50.0

Reference 2: 100.0

Reference 3: 150.0

Test Result 1: 50.5 ✓

Test Result 2: 101.0 ✓

Test Result 3: 150.0 ✓

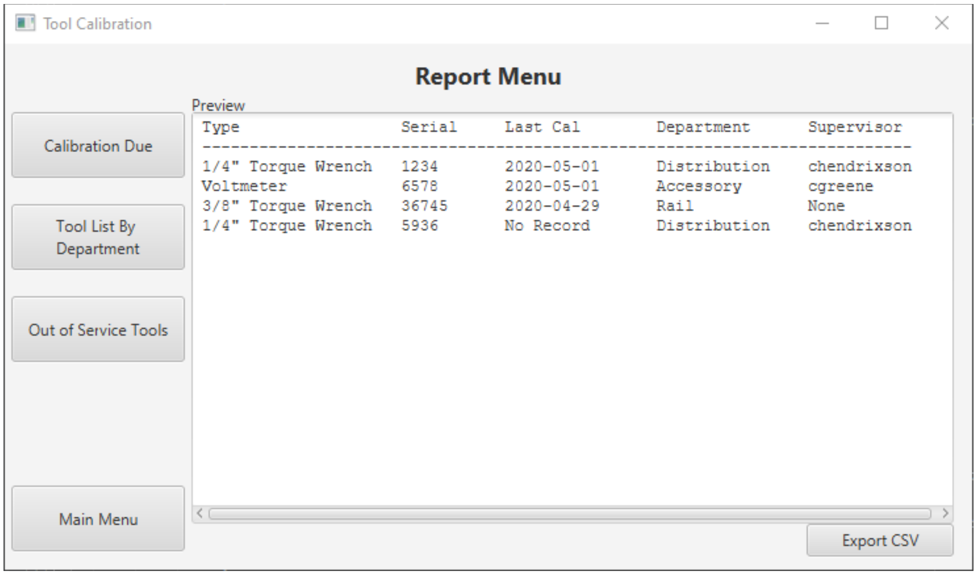
Calibration Date: 6/1/2020

☑ Passed

Tested By: admin

Reports

To access the Reports menu, select Reports from the Navigation bar. This will bring up a list of options on the left and an open text field on the right. Selecting one of the options on the left will bring up a preview of the report in the text field.



```
Type                Serial     Last Cal        Department      Supervisor
---------------------------------------------------------------------------
1/4" Torque Wrench  1234       2020-05-01      Distribution    chendrixson
Voltmeter           6578       2020-05-01      Accessory       cgreene
3/8" Torque Wrench  36745      2020-04-29      Rail            None
1/4" Torque Wrench  5936       No Record       Distribution    chendrixson
```

This information is great if you just need to get a quick look to gain information from the report; however, its usefulness is really limited to just that. If further analysis or report distribution is needed a better solution is to use the Export CSV button at the bottom right. This file can then be opened in a spreadsheet application or e-mailed.