Employee Management System

Evaluation Manual

Paul Schlabach: [pms1006@sru.edu](mailto:pms1006@sru.edu)

Bradon Walters: [bdw1006@sru.edu](mailto:bdw1006@sru.edu)

Maya Smallwood: [mns1013@sru.edu](mailto:mns1013@sru.edu)

**Table of Contents**

* Problems with the system
  + Problem 1: Passwords and their restrictions
  + Problem 2: Unused Models
  + Problem 3: Encrypting Passwords From The data.sql File
  + Problem 4: Database Connections
  + Problem 5: Automatic Import of data.sql file
  + Problem 6: Redirecting users
* Improvement/Expansion
  + Messaging System
  + File Server
  + Updating employee information
  + Work Logging
  + System Logging

**Problems and How They Can Be Fixed**

**Problem 1: Passwords and their restrictions**

At this time the program does not make the user enter their new password twice in order to check that they match. This is an important security feature to ensure the user knows their password and did not mistype, therefore locking them out of the system if they happened to do so. At this time, we also have only placed a limit on the length of passwords which must be 6-10 characters long. Restrictions requiring the password to contain capital letters, special characters, numbers, etc have not been implemented. You can find the code regarding password creation inside “LoginController.Java” and “registration.html”. Currently, registration.html holds the value “minimum length 6” and “maximum length 10” values inside of it (as seen in Figure 1).

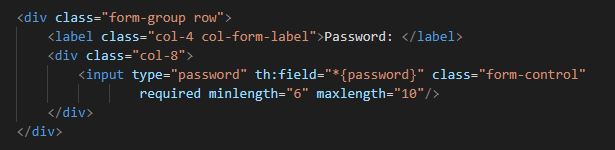


Figure 1. Limits password creation to 6-10 characters.

There may be a way to set password restrictions within the LoginController.java file by creating checks to ensure an allowed input has been entered. You can see this in Figure 2. Where we have checks in place to ensure the new password being entered is not null in the event of changing the user's password. Code similar to this may allow you to set further password restrictions.



Figure 2. Code for changing password. Checks are shown to ensure the new password is not null

**Problem 2: Unused Models**

Our project currently has models that were created to aid in improving our project that were never implemented. These models can be used as a start to build upon what we already have in the project. These unused models are Auditable.java, CommonObject.java, Contact.java, Country.java, Employee.java, EmployeeType.java, JobTitle.java, Location.java, Payroll.java, Person.java, and State.java. The basic framework for these was put in place in the early stages of our project, and was not gotten to in time for final submission but should be useful in aiding any future groups.

**Problem 3: Encrypting Passwords**

Within the data.sql file, you will see that the field for the user password is left blank. This is because we could not find a solution to having the password stored as an encrypted value within the MySQL database. It is because of this that the user will have to click on “Forgot Password” upon logging in to create their password for the first time. If a solution can be found that allows an administrator to include passwords into the data.sql file and have their encrypted values saved to the database, not plain text, then it would eliminate the need for the user to set a new password before first logging into the system.

**Problem 4: Database Connections**

Within the application.properties file found in resources is the information to log in to a MySQL workbench instance using your username and password. When we merge our code to GitHub the application properties file will not merge correctly as the rest of the project does. This causes developers to enter in their MySQL password each time they import the project from GitHub, which can cause confusion and errors if not changed before running the application. We could not find a fix for this and are unsure why the application.properties file will not save changes upon merging.

**Problem 5: Automatic Import of data.sql File**

Ideally, the data.sql file found in resources should be automatically imported upon running the project using spring boot tags that can be found within “editingdatasqlfile.txt”. For an unknown reason, the spring boot tags spring.sql.init.mode=always will continuously attempt to populate the database, however, MySQL will return an error seeing that there is a duplicate entry to a specific row. The tag spring.sql.init.mode=embedded will fill the database once, but upon rerunning the application it will not automatically update, also causing errors.

**Problem 6: Redirecting Users**

Currently in the program, on the Time Sheet and Add Users webpages when importing an Excel file, users are directed to a blank success screen rather than back to the previous webpage. Once on this screen users must click “Gazebo HQ” in the top right-hand corner, go back to the dashboard, and then click back into either Time Sheet or Add User to get back to where they were. Ideally, users should be sent directly back to the page they were on, to begin with. During the spring boot initiation process of importing the Excel file is when it sends the users to the success screen, and the addition of Thymeleaf tags inside of the HTML file redirecting users back to the correct page was unhelpful in trying to solve this issue.

**Future Work**

**Messaging System:**

Originally we wanted to implement a messaging system into our system, similar to Slack, where employees could communicate from within the system to one another, and have their own inboxes, etc. Unfortunately this goal was not achievable within our time frame due to the various issues we encountered. One way to implement this feature could be to set up an SMTP server and assign email addresses to each employee, then implement the server into the project in a way that displays all incoming and outgoing messages. Our reasoning for wanting to include this is that our goal was to create a system that could be used for companies who have recently moved to remote or hybrid working conditions. Having a system where you can keep track of payroll, employee databases, and also message other employees would help keep the number of programs required to function as a business minimal.

**File Server:**

Another system we were not able to implement was a file server. Ideally, this page would allow users to import files to the server, as well as search for files that have already been imported. Once a file from the server has been selected it would show up in a panel on the screen to be viewed before the user can choose to download it onto their machine. The system we had in mind was similar to that of D2L where a professor could upload a file to be viewed within the shell, and then download it if necessary. A system like this would allow companies to store important documents within their management systems such as employee contracts, employee guidelines, company notices, and any other files that may be appropriate. This system should have permissions that can be set so that certain employees can view certain documents and not all of them. This is important so that admins or managers can view all documents but employees can only view documents that pertain to them in their specific role.

**Updating employee information:**

This was a problem we ran into toward the end of our project. Currently in the top right-hand corner of each page is an avatar icon that can be hovered over. Upon hovering, you will see two options, settings, and log out. Currently, if you click on the settings option, it will send you to our account settings page where it will display the user’s information such as their name, email address, etc. Ideally, there should be an option within this page to adjust these values, for example, a user should be able to edit their saved email address, or address. We were not able to have this function because the system could not recognize which user to update within the database table. While displaying information to the page was able to select the currently logged-in user, sending information to update the database was producing errors due to the system not knowing which user to send the information to.

**Work Logging:**

Employees while working should receive updates from the system such as reminders about timesheet information, updates to their user information, and other important tasks completed within the system. There should also be periodic banners that appear after a specified time of inactivity prompting the user to sign out of the system for security purposes.

**System Logging**

There should be a system log accessible by admins that displays any and all information happening within the system. This should be a page that displays time stamps with exact changes made to the system in a list. For example:

* 04/24/2023 11:43:56 ADMIN pms1006 created new USER bdw1006
* 04/24/2023 15:22:13 ADMIN pms1006 added file Contract.txt to FILE SERVER

This should be done so that admins can properly track system usage and easily identify any mistakes made within the system during the workday or pinpoint misuse of the system to a specific user.