Web Routing

Security Manual

Slippery Rock University

Fady Aziz [FAA1002@SRU.EDU](mailto:FAA1002@SRU.EDU)

IAN BLACK [IMB1007@SRU.EDU](mailto:IMB1007@SRU.EDU)

LOGAN KIRKWOOD [LLK1005@SRU.EDU](mailto:LLK1005@SRU.EDU)

Contents

[1. Overview 3](#_Toc101805450)

[2. Integration of Roles 3](#_Toc101805451)

[3. Limiting Access 3](#_Toc101805452)

[3.1 Limiting Access by Role / Permissions 3](#_Toc101805453)

[3.2 Limiting Access by HTML 4](#_Toc101805454)

[4. Password Hashing 5](#_Toc101805455)

[4.1 Password Hashing 5](#_Toc101805456)

[4.2 Hiding Hashed Password 6](#_Toc101805457)

[5. Password / Username Requirements 6](#_Toc101805458)

[6. Error Handling 6](#_Toc101805459)

[6.1 User Registration / Login Error Handling 6](#_Toc101805460)

[6.2 Invalid Data Type Error Handling 6](#_Toc101805461)

[6.3 Dependency Issue Handling 7](#_Toc101805462)

[7. Checking for Duplications 7](#_Toc101805463)

[8. Preventing SQL Injection 8](#_Toc101805464)

[9. Reading Files 8](#_Toc101805465)

[10. Use of Logger Class 8](#_Toc101805466)

[11. Figures 10](#_Toc101805467)

# 1. Overview

When completing this project, security was one thing that was kept in mind during the development. This document will be an overview of some of the security issues that came up during development and how they were handled.

# 2. Integration of Roles

Before roles were added to the project, all users were able to access all parts of the application. This means all users would be able to edit and add entities that any user can change or delete. In the case of this application, that would not be practical or acceptable as there are clear cases where more roles are necessary.

This issue was fixed by adding 4 different roles to our project that can access very specific portions of the application. This addition was absolutely critical to the application as now not all users will be able to have access to all parts of the application.

For example, before, a user that creates an account will be able to edit the details of other users in the system. Now, only users with the admin role can edit the different users of the system.

# 3. Limiting Access

As stated before, one of the biggest issues of the application was the lack of roles being integrated into the application. The different roles now have different permissions which will be addressed below in [Section 3.1](#_3.1_Limiting_Access)

## 3.1 Limiting Access by Role / Permissions

Different roles have access to different areas of the application. The look and fields on a page will look different as well depending on what the user’s role is. Below are the different roles that are in the application.

**Admin** – Identified as “ADMIN” – Admin is the only role that can change account information. If a user has trouble accessing their account, they should contact a user with the admin role. Admin has access to:

* Users

**Shipper** – Identified as “SHIPPER” – Shippers can create shipments for carriers to bid on. They can also accept bids that have been placed on shipments created by them. Shipper has access to:

* Shipments

**Carrier** – Identified as “CARRIER” – Carriers can view shipments created by shippers and place bids on those shipments. If a shipper accepts a bid placed by a carrier, that shipment is automatically assigned to the carrier that placed the bid and the carrier can then edit the details of that shipment. Carrier has access to:

* Contacts
* Locations
* Vehicle Types
* Vehicles
* Drivers
* Technicians
* Maintenance Orders
* Carriers
* Shipments
* Routes

**Master List –** Identified as “MASTERLIST” – The Master List can view *ALL* shipments, and carriers regardless of how many shipments have been created/assigned to them. They can also view the shipments assigned to certain vehicles and shipments that are set to go out on a certain date. Master List has access to:

* Carriers
* Shipments
* Routes

In addition to the pages listed below, each user that has access to the shipments page will show different options. The same goes for those with access to the carriers page. Carriers will only see the details of their own carrier while someone with the Master List role will see all of the carriers that have been added to the system. The difference with that is also that the carrier can edit the details of their own carrier while the master list cannot edit any of the details of any of the carriers.

## 3.2 Limiting Access by HTML

Before roles were implemented, all users would see all of the pages listed at the top of the page. When roles were first implemented, that problem was still present so regardless of the user’s role, they would see all of the available pages. This could pose a problem with security as the user would know exactly what pages are available in the application.

The HTML files were updated with a hasAuthority() statement being added to each tab at the top to hide those buttons from users who should not have access to those pages. Below are examples of the different tabs available (Figure 1), (Figure 2), (Figure 3), (Figure 4), (Figure 5)

A green screen with white text

Description automatically generated with low confidence

Figure – Tabs of a user logged in with the ADMIN role



Figure – Tabs of a user logged in with the CARRIER role

Graphical user interface, text, application

Description automatically generated

Figure – Tabs of a user logged in with the SHIPPER role

A screenshot of a computer

Description automatically generated with low confidence

Figure – Tabs of a user logged in with the MASTERLIST role

Graphical user interface, text, application

Description automatically generated

Figure – Tabs of a user that is not logged in

# 4. Password Hashing

Another large issue with security is the possibility of someone getting into the database and accessing the user’s table. At the start of this project, the user’s password was not hidden and was stored in plain text. Below details how this issue was fixed using hashing.

## 4.1 Password Hashing

As stated before, at the start of this project, the password was stored in plain text meaning if someone got access to the users table, they would have the passwords to all of the accounts in the system. To combat this, the BCryptPasswordEncoder class which is part of Spring Security was used to hash the password. Below is an example of a password that is plain text, and the same password after it is hashed.

|  |  |
| --- | --- |
| **Password in Plain Text:** | Password |
| **Password after being hashed:** | $2a$10$j4JWTe7EP6vPiptRf1WP1ujvRRNVPzvuQXkO7KH9Ot.YBY0znyKam |

Figure – Password in Plain Text vs. Hashed

## 4.2 Hiding Hashed Password

After implementing the hashed password into the application, another problem we ran into was that the hashed password was being displayed to the user that was logged in as an admin. This still means an unwanted person could get into and view that password. Even though it is hashed, it is still possible that the original password could be found.

To combat this, the password is simply no longer displayed to the user in the users page and is not displayed anywhere in the application.

# 5. Password / Username Requirements

When first implementing the login system, there were no username or password requirements. This means a username or more importantly a password could be made any length making it not very secure.

Now the username and password have requirements that are checked when signing up for an account or editing the details. More on that in [Section 6.1](#_6.1_User_Registration)

The requirements are as follows: the username MUST be **between 6 and 32 characters** **long**. The password MUST be **between 8 and 32 character long**.

# 6. Error Handling

Error handling is something that is critical when developing an application. It is something that came up many times during the development of this application. This poses a security risk as a lack of error handling can lead to acquisition of restricted data, adding unwanted data to a database, or just simply not displaying any sort of information regarding what went wrong. Below is the different types of error handling that had to be implemented into this application.

## 6.1 User Registration / Login Error Handling

If a user enters an incorrect username or password, an error message is displayed indicating that one of those fields is incorrect. It important to note that it does not show which one is incorrect. Doing so would give an intruder the knowledge if a username is correct which would entice them to keep trying different passwords to try to gain access.

A similar error message is displayed if a user tries to register for an account and the username or password do not meet the requirements outlined in [Section 5](#_5._Password_/)

## 6.2 Invalid Data Type Error Handling

Another type of error handling that is done is when a user inputs a data type that is not valid. For example, before a user could input a string into a box that is meant for integers. This issue has been resolved with an error saying that an invalid data type was entered, and the user needs to enter a different value into that box.

## 6.3 Dependency Issue Handling

Previously, an entity could be removed that had different entities depending on it. When doing so, it would cause errors on the pages that were dependent on that entity that was deleted.

To combat this, the entity is checked to see if there is anything that is dependent on that. If there are dependencies, an error message is displayed to the user saying there are dependencies and that element cannot be deleted (Figure 7)

Table, Excel

Description automatically generated

Figure – Error Handling when Deleting Entity with Dependencies

# 7. Checking for Duplications

Duplication checking was added for all entities within the program. For example, a someone trying to create an account may not be able to if the username has already been taken. Another example would be someone trying to create a shipment that is identical to another already created. These sorts of things are not possible in the program due to checks being put into place. Now, when a user tries to create a duplicate entity, or put in an entity that is not valid, they will be shown an error telling them that an entity with the same information has already been added to the database. (Figure 8)

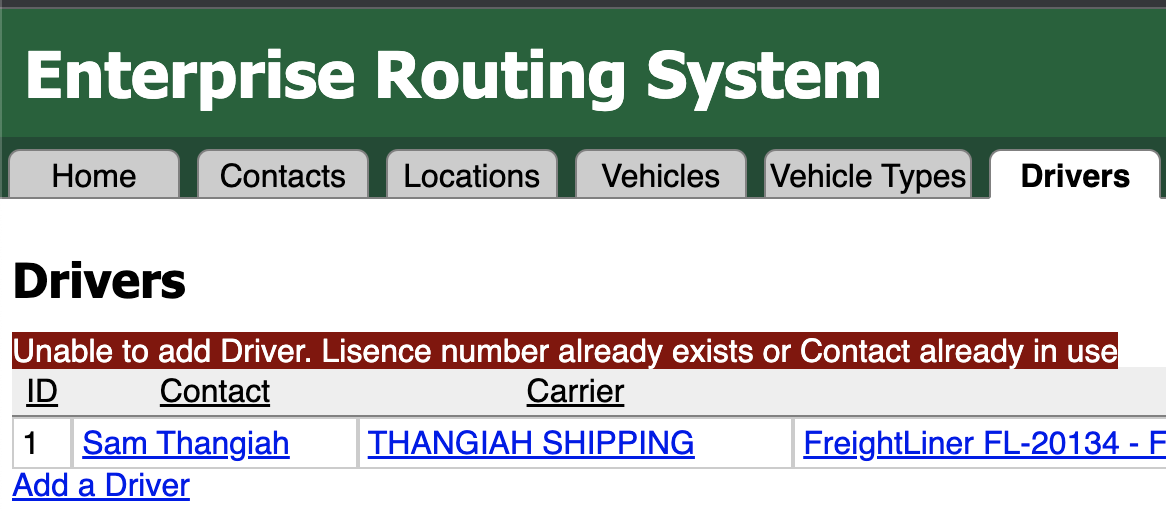


Figure – Duplication Error on Drivers Page

# 8. Preventing SQL Injection

SQL Injection is taken care of using the Hibernate library and the CRUD repository. Using a combination of both of those, the user is able to put SQL code into the field but it will be ignored as the parameters are placed in after the query is created. This means that regardless of what a user puts in to a particular field, they will not be able to access other areas of the program.

# 9. Reading Files

Within our program, there is only 1 case of a file being open and read from. This takes place inside of the ShipmentsController class. The whole sequence is placed inside of a try catch block where the file is iterated through and each entity is added to the database. If an error does occur when reading the file, the error is printed in the log where an admin can view and possibly fix the error.

# 10. Use of Logger Class

The logger class was implemented in both testing as well as the final project. In the final project, the logger class is implemented into the SecurityServiceImpl class as well as the MyAccessDeniedHandler class. In the SecurityServiceImpl class, the logger is implemented to signify when the auto login method is called so it logs when the user logs in. In the MyAccessDeniedHandler class, the logger is used to log when a user attempts to connect to a page that they do not have access to. The logger displays the username and the URL they tried to connect to. Implementing these logs has proven to be beneficial in terms of security as they log the most important things and identify that the program is working as it should.

When implementing new features into the program and testing them, the logger class was used many times when testing different algorithms. For example, when implementing the shipments class, all the shipments must be looped through to ensure they do not have a carrier associated with them. When implementing that method, it was important to use the logger to see what shipments, and how many shipments were displayed to the user. The logger class has been used many other times during testing and implementation as well to ensure that different parts of the program are working properly before deploying the application. The logger class proved to be a critical part of the development of this application.

# 11. Figures

[Figure 1 – Tabs of a user logged in with the ADMIN role 4](#_Toc101805408)

[Figure 2 – Tabs of a user logged in with the CARRIER role 4](#_Toc101805409)

[Figure 3 – Tabs of a user logged in with the SHIPPER role 5](#_Toc101805410)

[Figure 4 – Tabs of a user logged in with the MASTERLIST role 5](#_Toc101805411)

[Figure 5 – Tabs of a user that is not logged in 5](#_Toc101805412)

[Figure 6 – Password in Plain Text vs. Hashed 5](#_Toc101805413)

[Figure 7 – Error Handling when Deleting Entity with Dependencies 7](#_Toc101805414)

[Figure 8 – Duplication Error on Drivers Page 8](#_Toc101805415)