WEB ROUTING

Final Report

SLIPPERY ROCK UNIVERSITY

Joshua Gearhart jjg1018@sru.edu

Nicholas Bushée nab1017@sru.edu

Table of Contents

### 1. problem explanation 3 1.1 system overview 3 1.2 Roles - Agents 3

### 1.2.1 admin 3

### 1.2.2 shipper 3

### 1.2.3 carrier 3

### 1.2.4 master list 3 1.3 problem identification 4

### 2. Solutions description – task completion 4

### 2.1 completed contributions 4

### 2.2 incomplete tasks 6

### 2.2.1 modules not tested 7

### 3. system and hardware requirements 7 3.1 system requirements 7

### 3.2 hardware requirements 7 4. Setup and Installation 8

### 5. Login 8

### 5.1 registration 9

### 6. shipments and bidding 10

### 6.1 shipment creation 10

### 6.2 bid placement/acceptance 10

### 6.3 association assignments 11 7. Future Work 11 7.1 data sanitization 12

### 7.2 push notification system 12

### 7.3 technician role 12 7.4 rfid/gps tacking service 12

### 7.5 log interface – system commands 13

### 7.6 mobile application 13 7.7 expansion of profile pages 13

### 8. documentation 13 8.1 method documention 13

### 8.2 class documentation 13

### 8.3 outside documentation 13 8.4 javadocs documentation 14 9. code reusability 14 10. testing 14

### 10.1 human-computer testing 14

### 10.2 whitebox testing 14

### 10.3 unit testing 14 10.4 boundary analysis 14

### 10.5 unimplemented tests 15

### 11. post-mortem analysis 15

### 12. References 16

# 1. problem explanation

1.1 system overview

The ***Enterprise Routing System*** program was designed to provide CRM services to entities that participate in the freight shipping pipeline. It has an implemented bidding system that functions to connect **CARRIER** and **SHIPPER** roles for the purpose of creating shipments, bidding on them, and accepting the bids. Features to enable accounts to manage their user data (shipments, bids, contacts, vehicles, etc.) exist but can be further optimized.

The system is also intended to serve as a route generator for the shipments that are accepted; the routing features still require further development. The system does have a connection to Google Maps and generates routes from geocoordinate inputs. This could be heavily expanded upon and will be discussed in the Future Work section.

1.2 roles - Agents

Four roles currently exist in ***Enterprise Routing System***. They are **ADMIN**, **SHIPPER**, **CARRIER**,and **MASTER LIST**.

### 1.2.1 admin

Administrator accounts are responsible for handling user password resets, user account troubleshooting, and system database maintenance.

### 1.2.2 shipper

Shipper accounts are responsible for creating shipments for carriers to bid on. They can create shipments one-at-a-time or upload csv files containing multiple shipments. After bids are placed on the shipments, shipper accounts can accept the bids to complete the transaction.

### 1.2.3 carrier

Carrier accounts are responsible for bidding on shipments and managing the company’s vehicle fleet. They can view all available shipments, the bids on those shipments, and place bids themselves. The CRM management features for carriers allow for a ***Contacts*** list that aggregates the carrier’s book of business and also allow the carrier to view their own shipments and bids in aggregate.

### 1.2.4 master list

Master List accounts have access to a systemwide aggregate of shipments, bids, and carriers.

1.3 problem identification

***Enterprise Routing* System** has the basic functionalities of the intended bidding program. It requires a few notable features and updates to the UI to improve user experience and overall effectiveness. Software security is another focus for the application’s current functionality.

To progress the feature development in the program, we will focus on:

* Fixing functionality of updating user profiles to properly update user credentials to database and prohibit duplicate emails from persisting in the system
* Implementing a dynamic loading of the database upon program initialization. This will create more efficient deployment capabilities by ensuring the database schema is properly established upon boot.
* Improve software assurance through unit test suite implementation.

To improve user experience, various items will be refactored in the GUI:

* Overall structure of page layout and content wrapping will be addressed to integrate layered elements with the z-index
* Data entry forms will be refactored into various, reusable form assets
* Dynamic button assets will be created to implement sitewide
* Navigation issues on various webpages will be resolved through appropriate buttons and links
* The navigation tabs stretching horizontally across the marquee of the page will be refactored into a content-push, dynamic (expandable/collapsible) side navigation bar. Navigation links will be restyled, respaced, and altered in other ways for QoL.
* Anticipating encountering various minor GUI bugs and will fix them as they are encountered.

To increase the program’s security, we plan to implement the following items:

* User verification via Google ReCaptcha
* Email validation for newly registered users
* Remove duplicate email bug when updating user details
* Add a Forgot Password feature for users to reset lost passwords

# 2. Solutions description – task completion

Completion Status: *Incomplete*

Progress Report in the subsections below.

2.1 solutions – completed contributions

The following is a comprehensive list of contributions made to ***Enterprise Routing System*** in its current iteration. Brief descriptions of the solutions implemented are included. The **Issue Numbers** coincide with the **Issue Numbers** for the group project on GitHub.

* **Issue #1 – Email Validation – Shipper**
* Sends an email to a user’s new registered email account with a verbose message containing a link with the user’s verification code implemented as a parameter. This allows the controller method to get the mapping and request and validate their verification code. ***See Figure 1***
* **Issue #2 – Google ReCaptcha**
* User verification is conducted during registration. If bot activity is suspected, heightened security, via a challenge question, is enabled. ***See Figure 2***
* **Issue #7 – Email Validation – Shipper**
* (See Issue #1 description and ***Figure 1***)
* **Issue #8.1 – Admin Update User**
* Utilized a transient data value to hold the user's old email and allow the value to persist if verified that it is unchanged. Does not allow for duplicate emails as it did previously. Admins also can no longer directly change the user's password; they can select a reset password option that emails the user a link to reset their password.
* **Issue #8.2 – Update User Details**
* Utilized a transient data value to hold the user's old email and allow the value to persist if verified that it is unchanged. If the email is changed, the database is checked for duplicates. User may also update their password, and they are required to validate it via ‘Confirm Password’ second entry before changes take place.
* **Issue #9 – QoL navigation of /createdshipments**
* Implemented Spring Security, permissions-based buttons for different role access.
* **Issue #10.0-10.13 – GUI Improvements**
* Altered marquee to display customer service number and remove previous nav bar
* Created reusable assets for forms, buttons, columns, *select* inputs,
* Placed *#content* div on z-index to layer GUI elements properly
* Implemented permissions-base buttons assets around site where needed
* Refactored all ***/add*** and ***/update*** templates with form assets
* Created a columnar home page for registration and account information
* All existing forms have been updated to new assets
* **Issue #13 – Forgot Password**
* Creates a one-time-password code (referred to as otpCode in the application) for the specified user. This is sent to the users email as a parameter, and ***ForgottenPasswordController*** validates the users link matches to the database, upon approval allows for the user to reset their password only if it is a valid input and matches a confirmation.
* **Issues #14/15/22 – Dynamic Generation of Database**
* ***Enterprise Routing System*** now starts by implementing application runner. App runner is an inherent method to Spring that allows functions to run during application context. Using the database url, user, and password, the java.sql.Connection library creates a connection to the database. This allows circumvention of NULL values in required SQL fields during initialization.
* Implementation of the dynamically loaded database required identifying fields that were necessary to be populated and coding the proper initial values into the backend java via java.sql.Connection
* Implemented verbose console messages indicating database is populating.
* **Issue #16 – Database Value Assignment**
* Capped strict character limit fields to proper data types in SQL database for memory optimization.
* **Issue #18 – Duplicate User Email Addresses**
* Updated the user validator to check for duplicate email address values
* **Issue #19 – SCAC Code Strict Character Limit**
* Set restrictions to a 2-4 character code to reflect real SCAC Code standard
* **Issue #20 – Navigation Bar Refactor**
* Made the sitewide navigation bar a dynamic (expandable/collapsible) sidebar that pushes content when expanded and recenters content when collapsed.
* Rewrote ***MenuBar.css*** during refactor.
* **Issue #21 – Post-Registration Pathing to Login Page**
* Email verification link now redirects to the login page with a message claiming successful email verification.
* **Issue #23 – Asynchronous Verification Email Sending**
* Sending the verification email kept registration pages busy, so the action was made asynchronous to free the program for further processing while email is being sent
* **Issue #24 – Ambiguous Order of Google ReCaptcha Display on Registration Pages**
* Changed ReCaptcha to load upon a proper ‘false’ submission of the registration form. The captcha then displays, is navigated, and completes the form submission.
* **Issue #25 – Additions and Refactoring of UserValidator Class**
* Complete refactoring and additions to the UserValidator Class, this was integral to many other additions in the program.
* Refactoring this class allowed for a more thorough UserValidator.

2.2 incomplete tasks

***Enterprise Routing System*** has ample room for improvement on existing features as well as room for additional features. The testing suite for this system is also partially complete. The identified issues that remain incomplete this semester are as follows:

* **Issue #3 – Lockout for Failed Password attempts**
* **Issue #4 – Implement Contact Admin Feature via Email for Support**
* **Issue #5 – Creating the Technician Role and Job Quote Class**
* **Issue #6 – Validation of SCAC Code**
* **Issue #11 – Add Contact csv in Carrier Role**
* **Issue #17 – Add Proper Error Message when a User Attempts to Log In Prior to Validating Email Address**

### 2.2.1 Modules not tested

The testing suit for this system is mostly complete. No tests were inherited in this project except for one which was the Contacts Controller Class. Though this test does not actually appropriately test the Contacts Controller Class. For more information on Modules not tested see section 10.4.

# 3. System & Hardware Requirements

3.1 System Requirements

* **Operating System**
* **Required:** Window
* **Version:** 10
  + - **Link:** https://www.microsoft.com/en-us/software-download/windows10
* **Java IDE**
* **Required:** Eclipse IDE
* **Version:** 2022-06
  + - **Link:** https://www.eclipse.org/downloads/packages/release/2022-06/r/eclipse-ide-java-developers
* **Database**
* **Required:** MySQL Server & Workbench
* **Version:** 8.0.30
  + - **Link:** https://downloads.mysql.com/archives/installer/
* **Internet Browser**
* **Required:** Any updated Internet Browser
* **Recommended:** Upgraded Google Chrome
  + - **Link:** https://www.google.com/chrome/dr/download/?brand=CHBD&geo=US&gclid=Cj0KCQiAsoycBhC6ARIsAPPbeLsLUy2IR\_YPNp57SJUvP5Eobf3w9DZjd2dPmuhr4jSZOqSqYLZwUsUaAoCMEALw\_wcB&gclsrc=aw.ds

3.3 hardware requirements

* **CPU:** Multi-Core 64bit x86 architecture CPU
* **RAM:** 12 GB or higher
* **Display:** 1024x768
* **Storage:** 250 GB

**\*NOTE: Metrics of given Hardware Requirements are for low-end optimal performance.**

# 4. setup and installation

To set up the Webrouting application, it must be downloaded onto your machine/system and setup in an Eclipse workspace. To do this, follow this Github link: <https://github.com/samthangiah/Fall-2022-Group-5-Web-Routing> and grab the URI from the ***Code*** tab.

Import the project with Git in Eclipse, entering in both the URI and the Personal Access Token on your GitHub. Branching can be done in the Eclipse IDE or from the GitHub website. This process is detailed further in the Install/Configuration Manual

# 5. login

Upon program launch, the user is greeted by the Enterprise Routing System home page. The user can login by selecting the ***Log In***button and will be redirected to the login page (**Figure 1**). A table containing default credentials for each role is provided below:

|  |  |  |
| --- | --- | --- |
| **Default Credentials/Role** | **Username** | **Password** |
| **ADMIN** | AdminTry | AdminTry |
| **SHIPPER** | Shipper | Password |
| **CARRIER** | Carrier | Password |
| **MASTER LIST** | Master | Password |

Graphical user interface, website

Description automatically generated

**Figure 1 – Login Page of Enterprise Routing System**

After logging in, the user has all the permissions of the role registered to the account they are using.

5.1 registration

On the homepage, click ***Create Account*** or select the ***Registration*** tab on the sidebar to navigate to the ***Registration Home*** page (**Figure 2**). Here, new accounts of either the **SHIPPER** or **CARRIER** roles can be created. Each role’s account features are detailed to aid the user in making the right selection.

Graphical user interface, application

Description automatically generated

**Figure 2 – Registration Home Page**

After registering and verifying the email account used, the user can then access ***Enterprise Routing System*** with the permissions of the role associated with the user.

# 6. shipments and bidding

The core component of ***Enterprise Routing System*** is the shipment and bidding system implemented for **SHIPPER** and **CARRIER** accounts. The two roles are entities that have either the goods that need to be shipped or the vehicle fleet and logistical capability to do the shipping.

6.1 shipment creation

The creation of shipments requires access to a **SHIPPER** account and the pertinent information for a shipment. This includes locational information for the shipper and consignee, the type, class, and weight of the commodity being shipped, and the scheduled date of the shipment. An example form is given below:

Graphical user interface, text

Description automatically generated

**Figure 3 – Shipment Creation Form**

6.2 bid placement/acceptance

After shipments are created, they are bid on by **CARRIER** accounts. The carrier submits their bid price, and the bid is then generated and accessible via that shipment. Both the **SHIPPER** and **CARRIER** accounts can see available shipments and the bids placed on them. An example of the bids on a shipment is given below:

Table, Excel

Description automatically generated

**Figure 4 – Display of Bids on a Particular Shipment**

A **SHIPPER** account that has a shipment with a bid on it may accept that bid if the price is right. After accepting the bid, the shipment moves from ***Available Shipments*** to ***Accepted Shipments*** for both the **SHIPPER** and **CARRIER** accounts involved with the shipment. For uninvolved parties, the shipment record is simply no longer in ***Available Shipments***.

6.3 association assignments

The processes of shipment creation, bid placement, and bid acceptance create associations between multiple objects in ***Enterprise Routing System***.

1. **Shipment Creation** – Associates a shipment with a shipper, and attaches the shipper’s information to the shipment
2. **Bid Placement** – Associates both a bid and the carrier that placed it with the shipment it is placed on. The bid containing the bid price and carrier name is generated and viewable by **SHIPPER**, **CARRIER**, and **MASTER LIST** accounts.
3. **Bid Acceptance** – Completes the association between the **SHIPPER** and **CARRIER** roles by adding the carrier information to the shipment and moving the shipment from ***Available Shipments*** to ***Accepted Shipments***

# 7. future work

***Enterprise Routing System*** has plenty of room for improvement with the overall scope of its mission. Primary future objectives revolve around furthering the system’s capacity to communicate to users at various stages in the shipping and bidding processes. There are ample reasons to also dive into data sanitization to ensure accurate, uniform database entries. A push towards commercialization of this app would include creating a mobile variant, establishing in-app system commands and console interface for administrators, further expansion of the CRM offerings and profiles each role can build, and implementing other must-have features such as shipment tracking through either RFID or GPS.

7.1 Data sanitization

The backbone of ***Enterprise Routing System*** is form-based data transfer between accounts. While a number of these form inputs are properly set up, there are numerous forms that contain fields not properly sanitized for valid database entries.

Examples Include:

1. SCAC code – Should not allow for special characters (Ex. !@#$%)
2. Zip Code – Should be a 6-digit code or a 10-digit code and limited to those restrictions
3. VIN Number – Should have a strict 17-character requirement
4. Plate Number – Should prohibit special characters and allow only the range of valid characters used on American license plates.

7.2 Push notification system

The various roles in ***Enterprise Routing System*** would benefit from a role-specific, system-wide notification system.

**SHIPPER** accounts would benefit from push notifications when bids are placed on shipments as well as when shipments are picked up or dropped off.

**CARRIER** accounts would benefit in a variety of ways. They could be alerted to being outbid on a shipment or if their bid is accepted. They could also be alerted of assigned-vehicle arrivals and departures, as well as job quotes submitted by technicians providing maintenance for the vehicles.

**ADMIN** accounts could receive notifications from other accounts that require administrative assistance. They could also send push notifications to accounts that request password resets when the reset is conducted.

**ALL** accounts could each benefit from notifications when other user accounts are added into their book of business (or into the system for **ADMIN** accounts).

7.3 Technician Role

A new role should be created for the technician actor in this system. The individuals with **TECHNICIANS** accounts would benefit from the push notification system through real-time updates, and they would have the capability to fill out a ***Job Quote*** in response to the ***Maintenance Orders*** Carriers submit when vehicles require maintenance. The push notifications can alert the technician of a new ***Maintenance Order*** and also alert the carrier of the ***Job Quote*** submitted by the technician as well as its completion.

7.4 RFID/GPS Tracking Service

Most logistics servicing companies that employ programs like ***Enterprise Routing System*** offer tracking services for shipments, either by placing RFID chips on the package/truck or through GPS tracking. A simulation for this feature could be conducted using the coordinate entries of Start/Finish locations for shipments. Having the option to ping a package via package # lookup would be the logical integration of this service with a notification system.

7.5 Log interface – System Commands

Implementing a graphical interface for an in-browser console for **ADMIN** accounts to utilize to read logs and issue system commands to make for more effective troubleshooting with user accounts or the system itself.

7.6 Mobile Application

“There’s an app for that” absolutely applies to logistics and freight scheduling services. Truck drivers are always mobile and would greatly benefit from being able to interface with their company/contractor to gather information on the shipment manifest and accept jobs. Wireframing with Adobe XD/Figma to accommodate various form factors would allow for the mobile application to be properly styled from the existing web application.

7.6 Expansion of profile pages

Certain, profile-specific information is available to the roles in ***Enterprise Routing System***. The program would be made much clearer with a metric-comprehensive profile page for the users, developed with the roles in mind.

# 8. DOCUMENTATION

This section will overview the documentation accompanying the Webrouting Application

8.1 method documention

All methods in the Enterprise Routing System have been completely documented for anyone to pick up the system and understand methods.

8.2 class documentation

All classes incorporated into the Enterprise Routing System are documented on the class level as well as the functional level. This ensures that any programmer can pick up any part of the system with an insight into what is happening.

8.3 outside documentation

The Webrouting Application comes with lots of documentation packaged in with it. It also has outside documentation useful for anyone wishing to learn about the project. In this documentation you will see a ***User Manual*** is used for any persons who might want to use the program. Also included is a ***Technical Manual***, this manual is pertinent to the programmer. The technical manual highlights all classes for a programmer to understand what is happening.

8.4 javadocs documentation

Implemented into the project as well is ***Javadoc’s***, or Java Documentation. This documentation can be accessed by the index.html in the Javadoc folder located in the Documents folder of the application.

# 9. code reusability

The strategy employed with the front-end refactor was to first create reusable HTML blocks for form and button assets, style them with 2 main variant CSS styles, and then branch from those assets to cater to the different forms throughout the site. A ledger was kept to tally each of the 70+ templates that needed to receive sitewide updates.

Within our contributions code reusability was important. We utilized many of the methods defined by previous groups. The ***UserServiceImpl.class*** had many useful methods that were utilized throughout the program.

# 10. testing

Section 10 highlights the testing done with the Webrouting Application.

10.1 HUMAN-COMPUTER TESTING

Human-Computer Testing was conducted throughout the continued development of the Webrouting Application. This was done by the contributors to the project at length. These tests were to ensure things were working like they should from a user perspective as well as to make changes and improvements to user experience.

10.2 WHITEBOX TESTING

Whitebox Testing was performed by contributors to this project. Whitebox testing is performed by testers who have an inside knowledge of the system. Whitebox testing was performed specifically with a focus on elevation of privilege and attempts to gain access to places where access should be denied. This was done by understanding the file structure of the System as well as the html. Whitebox testing was implemented to ensure that users could not access pages of the application with an account Due to Whitebox Testing the application errors were found and changes were made to the WebSecurityConfig Class to ensure that users could not access pages they were not supposed to access.

10.3 UNIT TESTING

The Webrouting Application was missing a complete testing framework before this update. While updating this project testing was focused on heavily. Many tests were written thoroughly to ensure that the program’s methods were working efficiently. Junit 5 was implemented into the program. Through this project we also mitigated the run all tests method, it is deprecated in Junit 5. However, all tests can be run at once by right clicking the ***src/test/java*** folder in the project and selecting ***Run As*** and then choosing ***Junit***.

***Tests written:***

* ReceiveCaptchaTest
* RestConfigTest
* All edu.sru.thangiah.webrouting.domain classes (Exception: VehicleTypesTest)
* EmailingImplTest
* UserRepositoryTest
* UserServiceImplTest
* UserValidatorTest

10.4 UNIMPLEMENTED TESTS

Due to some issues, with getting tests to run and applying values to those, not all tests were finished. However, in aspects of testing the application has seen great improvement. Before this update the application had one test class which was the contacts controller class. It wasn’t a real test class despite that, tests were implemented as time allowed. However, this was not complete. Below are the tests not written:

* RecaptchaTest
* All edu.sru.thangiah.webrouting.controller classes
* VehicleTypesTest
* MailSendingTest
* PopulateDatabaseTest
* MyAccessDeniedHandlerTest
* SecurityServiceImplTest
* UserDetailsServiceImpl

# 11. POST-MORTEM ANALYSIS

This project revealed its patterns and backbone structure throughout the semester’s duration. As such, it presented novel strategies to implement for the purpose of increasing efficiency of workflow. Proper understanding of the Spring Boot Application framework and Spring Security framework established the recognizable patterns in the front-end refactor. The knowledge of the frameworks and identification of repeatability within them, combined with the strategy of making reusable, styled assets, allowed for an efficient workflow during the GUI overhaul. In hindsight, leveraging CSS frameworks such as Bootstrap, Tailwind, or JQuery would offer even more intuitive styling of the UI. In this project I think moving onto different problems when an initial problem solving fails in the current problem you are working on would also be a good idea. Many times, you get frustrated and clouded judgement, I feel though as less time can be wasted refactoring just by re-writing functions that are bad and misunderstood. In this project more hands could have also greatly increased productivity.

# REREFERNCES

“How To Create a Side Navigation Menu,” W3schools.com, 2019. https://www.w3schools.com/howto/howto\_js\_sidenav.asp

‌ N. H. Minh, “How to implement forgot password feature for Java web application,” How to implement forgot password feature for Java Web Application, 04-Jul-2019. [Online]. Available: https://www.codejava.net/coding/how-to-implement-forgot-password-feature-for-java-web-application. [Accessed: 30-Nov-2022].

N. H. Minh, “How to integrate Google Recaptcha V2 in Spring Boot,” YouTube, 11-Sep-2020. [Online]. Available: https://www.youtube.com/watch?v=HUlLj6Do1Cs&amp;ab\_channel=CodeJava. [Accessed: 30-Nov-2022].

N. H. Minh, “Spring Boot Email Verification for User Registration Tutorial,” Spring boot email verification for User Registration Tutorial, 23-Dec-2020. [Online]. Available: https://www.codejava.net/frameworks/spring- boot/email-verification-example. [Accessed: 30-Nov-2022].