WEB ROUTING

Technical Manual

SLIPPERY ROCK UNIVERSITY

Thomas Haley – [tjh1019@sru.edu](mailto:tjh1019@sru.edu)

Beth Orgovan – [bro0700@sru.edu](mailto:bro0700@sru.edu)

Dakota Myers – [drm1022@sru.edu](mailto:drm1022@sru.edu)

Sinchana Kori – [ssk1002@sru.edu](mailto:ssk1002@sru.edu)

Table of Contents

[Overview 3](#_Toc132981630)

[1. System requirements 3](#_Toc132981631)

[3. HARDWARE REQUIREMENTS 3](#_Toc132981632)

[4. webrouting application CLASSES 4](#_Toc132981633)

[4.1 SPRING BOOT APPLICATION 4](#_Toc132981634)

[4.2 controller CLASSES 4](#_Toc132981635)

[4.2.1 controller class information 4](#_Toc132981636)

[4.3 Domain Entity Classes 6](#_Toc132981637)

[4.3.1 Domain Entity Class Information 6](#_Toc132981638)

[4.4 Repository Classes 7](#_Toc132981639)

[4.4.1 Repository Class Information 7](#_Toc132981640)

[4.5 security CLASSES 8](#_Toc132981641)

[4.5.1 Security Class Information 8](#_Toc132981642)

[4.6 Mailsending Classes 8](#_Toc132981643)

[4.6.1 Emailing Class Information 8](#_Toc132981644)

[4.7 ReCAPTCHA Classes 9](#_Toc132981645)

[4.7.1 ReCAPTCHA Class Information 9](#_Toc132981646)

[4.8 Service Class 9](#_Toc132981647)

[4.8.1 Service Class Information 9](#_Toc132981648)

[4.9 Configuration Classes 10](#_Toc132981649)

[4.9.1 Configuration Class Information 10](#_Toc132981650)

[5. Testing 10](#_Toc132981651)

[5.1 Junit Testing 10](#_Toc132981652)

[6. UML Diagrams 10](#_Toc132981653)

[6.1 Class Diagrams 10](#_Toc132981654)

[6.2 Sequence Diagrams 11](#_Toc132981655)

[6.3 Activity Diagrams 17](#_Toc132981656)

[6.4 State Diagrams 21](#_Toc132981657)

[6.5 use case Diagrams 21](#_Toc132981658)

[7. Figures 26](#_Toc132981659)

# Overview

This document addresses the technical aspects of the WebRouting application. Including system requirements, Java classes, a brief overview of testing, and UML diagrams.

# System requirements

**1.Operating System**

* 1. ***Required:*** Windows
     1. ***Version:*** 10
     2. ***Link:*** https://www.microsoft.com/en-us/software-download/windows10

**2.Java IDE**

* 1. ***Required:*** Eclipse IDE
     1. ***Version:*** 2022-06
     2. ***Link:*** https://www.eclipse.org/downloads/packages/release/2022-06/r/eclipse-ide-java-developers

**3.Database**

* 1. ***Required:*** MySQL Server & Workbench
     1. ***Version*** 8.0.30
     2. ***Link:*** https://downloads.mysql.com/archives/installer/

**4.Internet Browser**

* 1. Any updated Internet Browser
     1. ***Recommended:*** Updated Google Chrome
     2. ***Link:*** https://www.google.com/chrome/dr/download/?brand=CHBD&geo=US&gclid=Cj0KCQiAsoycBhC6ARIsAPPbeLsLUy2IR\_YPNp57SJUvP5Eobf3w9DZjd2dPmuhr4jSZOqSqYLZwUsUaAoCMEALw\_wcB&gclsrc=aw.ds

# *3.* HARDWARE REQUIREMENTS

**CPU: Multi Core 64bit x86 architecture CPU**

**RAM: 12 GB or higher**

**Display: 1024x768**

**Storage: 250 GBs**

**Note:** These metrics are for low-end optimal performance

# 4. webrouting application CLASSES

This section contains functional information on each class organized by what the class handles.

## 4.1 SPRING BOOT APPLICATION

This section refers to the “main” of the Spring Boot Application.

4.1.1 Spring boot application class information

\* Class can be located at: edu.sru.thangiah.webrouting

1. **WebroutingApplication:** Main class, used to start the application (**@SpringBootApplication** annotation), **@EnableAsync** annotation configures the **Spring Boot Application** to allow running. Run this as a **Java Application** to run the whole project (See Note for more details).

**Note:**Install Spring tools in eclipse to bypass netstat command.

## 4.2 controller CLASSES

See Sub-section 4.2.1 for information on each Controller class of the Webrouting application. Controller classes are web handlers and are annotated with **@Controller.** Controller classes documented for programmer readability and understanding.

**Note**: Attributes are used to display information to users.

**Ex**: Model objects, etc.

### 4.2.1 controller class information

\*Classes can be located at: edu.sru.thangiah.webrouting.controller.

\*Deprecated Constructor methods are only used for testing, **@Autowired** Annotation handles this in Application

1. **AuctionController:**The Auction Controller class handles Thymeleaf controls for the pages dealing with auctioning. It Includes the methods related to auctioning shipments to carriers. Such as, push shipments to auction, remove shipments from auction, end the auction for a shipment, and enable/disable a user in the auction system.
2. **BackupController*:*** The Backup Controller class handles Thymeleaf controls for database backup, which automatically executes at the time of system start and every 20 minutes thereafter. It includes methods that allow the user to restore a backup file and convert it to an excel file.
3. **BidsController:**The Bids Controller class handles the Thymleaf controls for the pages dealing with bids on shipments. It includes methods related to the bidding aspect of the application. Such as adding, editing, resetting, deleting, accepting bids.
4. **CarriersController**: The Carriers Controller class handles all methods of the carrier pages. This can be viewed in html source at *carriers.html*, or the user interface *localhost:8080/carriers* when the application is running.
5. **ContactsController**: Handles the controls for pages dealing with Contacts. Viewed in html source at *contacts.html*orvia the user interface *localhost:8080/contacts* when the application is running.
6. **DriverController:** Web handler for pages dealing with Drivers. Viewed in html source at *drivers.html* ***or*** via the user interface *localhost:8080/drivers* when the application is running.
7. **EditAndAddController:**The Edit and Add Controller class handles the Thymleaf controls for the Carrier User pages dealing with the edit and add screens. These include methods to add/edit contacts, vehicle type, vehicle, location, and maintenance orders.
8. **ExcelController*:*** The Excel Controller class handles the Thymleaf controls for the pages dealing with importing from and exporting to an Excel file. It includes methods that allow Shipment users to upload shipments from an excel file. It allows Carrier users to upload or download contacts, vehicle type, vehicle, location, and maintenance orders, from or to excel files. As well as enable the Carrier user to upload their freight rates.
9. **ForgottenPasswordController:**Web handling class working with a user forgetting their password. Navigate mapping annotations for html and user interface.
10. **LocationController:** Web handler for pages dealing with Locations. Viewed in html source at *Locations.html* or via the user interface *localhost:8080/locations* when the application is running.
11. **LogController:** The Log Controller class handles the Thymleaf controls for logs.
12. **LoginController:** Web handler for pages dealing with Logging in and Registration. Viewed in html source at *registrationhome.html, registrationshipper.html, registrationcarrier.html and index.html or* via the user interface *localhost:8080/login or by following the create account registration when* the application is running.
13. **MaintenanceOrderController:** Web handler for the Thymeleaf controls for pages dealing with maintenance orders. Viewed in html source at *maintenanceorders.html, or* via the user interface *localhost:8080/maintenanceorders* when the application is running.
14. **NotificationsController*:***The Notifications Controller class provides the functionality needed to handle user notifications. Includes methods for read and unread notifications, as well as mark as read and mark as unread.
15. **RolesController:** Web controller for handling web aspects to the *roles.html* page and can be viewed at *localhost:8080/roles.*
16. **RoutesController:** Handles the Web controls for pages dealing with Routes, and functions dealing with routes Viewed in html source at routes.html or can be accessed via the user interface *localhost:8080/routes.*
17. **ShipmentsController:** Handles web controls for pages dealing with Shipments in all aspects. Html source can be located at *shipments.html* or via the user interface at *localhost:8080/shipments.*
18. **SimulationController*:***The Simulation Controller class handles the Thymleaf controls for pages that deal with the backend simulation of the Shipper user bidding process and directly assigning shipments to carriers.
19. **TechniciansController:** Works with web controls for pages working on technicians. Html source can be located at *technicians.html* or by accessing the user interface at *localhost:8080/technicians.*
20. **UserController:** Handles the web controls for pages dealing with Users. Notice Mapping Annotations above methods for html and user interface.
21. **VehiclesController:** Class for handling all web controls dealing with vehicles.
22. **VehicleTypesController:** Handles the web controls for pages dealing with Vehicle Types.
23. **VerifyEmailController:** Web controller class dealing with verifying a user’s email address and workflow of that process once registered. Follow mapping annotations for html and user interface.

## **4**.3 Domain Entity Classes

This section highlights the Domain Classes, each domain class is an entity located in the Webrouting schema. Domain classes are annotated with **@Entity** to apply this as well as explicitly named with **@Table** annotation to define table names.

### 4.3.1 Domain Entity Class Information

View section 4.3.1.1 for detailed information on Domain Class annotations.

\* Classes can be located at: edu.sru.thangiah.webrouting.domain

1. **Bids**: Defines Bids entity and contains methods associated with updating and retrieving information on bids.
2. **Carriers**: Defines Carriers entity and contains methods associated with updating and retrieving information on carriers.
3. **Contacts**: Defines Contacts entity and includes methods associated with updating and retrieving information on carriers.
4. **Driver**: Defines Driver entity and implements methods for updating and retrieving information on drivers.
5. **Filter**: Used for filtering data based on four attributes; start date, end date, level, and user.
6. **Locations**: Creates the locations entity and implements methods for retrieval and updating of locations
7. **Log**: Sets up logging for tracking activities in the system using attributes date, time, where, level, who, user, and msg. Also uses methods to get the date and return it as a String.
8. **MaintenanceOrders**: Defines Maintenance Orders entity and creates methods for updating and retrieving maintenance orders.
9. **Notification**: Constructs the template for the Notification Repository. Includes attributes id, user, message, time sent, is read. Also includes methods for getting and setting the attributes, as well as constructors for creating new Notifications.
10. **Role**: Sets up Role entity and methods allowing access to the Roles entity.
11. **Shipments**: Creates the Shipments Entity and methods allowing access to the Shipments Entity as well as updating and creating these entities.
12. **Technicians**: Generates Technicians entity and methods associated with assigning values and getting them.
13. **User**: Creates the User entity. Has **@Transient** Annotation for values not persisting into the database. Allows for user to be updated and user information retrieval.
14. **Vehicles**: Defines Vehicles entity and implements methods for updating and retrieving information on vehicles.
15. **VehicleTypes**: Creates the Vehicle Types entity and implements methods for updating and retrieving information on vehicle types.

#### 4.3.1.1 Annotation Information

Domainannotations are persisted through domain classes. **@Id** annotation signifies a primary key, **@GenericGenerator** and **@GeneratedValue** are coupled to automatically create user Id’s and incremented for each Id. Other annotations include **@Column** definition annotation to set values like name, nullability, variable type and length as well as default values. Other annotations included in domain classes are **@JoinColumn** annotations and **@OneToMany** as well as **@ManyToOne** annotations.

## 4.4 Repository Classes

Section 4.4 highlights the Repository Classes

### 4.4.1 Repository Class Information

\* Classes can be located at: edu.sru.thangiah.webrouting.repository

All Repository classes are inked to domain classes by domain data their perspective domain data type (See Section 4.3) repository classes extend CRUD Repositories (With exception to the User Repository which is extending the JpaRepository) allows for Spring Framework to interface Entity classes with Database and repository.

**Note:** UserRepository is the only class with defined methods for finding information regarding a User entity. **@Query** annotations are used to define a query upon method call.

## 4.5 security CLASSES

This section highlights classes focused on security, commonly implementing Spring Framework classes.

### 4.5.1 Security Class Information

\* Deprecated: Constructor methods are only used for testing, **@Autowired** Annotation handles this in Application

1. **WebSecurityConfig:** extends Spring’s WebSecurityConfigurerAdapter, used to explicitly define authority between pages based on roles and permissions. Also defines login page and logging out, continuing to define exception handling and access denied handling. Class located at: edu.sru.thangiah.webrouting.config
2. **UserValidator:** Implements Spring’s Validator, used in aspects of the program to validate and approve user input or deny using conditional statements which provide messages to be created to send to a user.

Class located at: edu.sru.thangiah.webrouting.services

1. **SecurityServiceImpl:** Class used for checking authentication within the context as well as permitting autologin, annotated with @Service. Class located at: edu.sru.thangiah.webrouting.services
2. **UserDetailsServiceImpl:** Class used for loading a user by their username and granting authorities with class annotated with @Service annotation, as well as a @Transactional annotation. Class located at: edu.sru.thangiah.webrouting.services
3. **MyAccessDeniedHandler:** Implements Springs AccessDeniedHandler to implement an access denied page to the user. Class located at: edu.sru.thangiah.webrouting.error

## 4.6 Mailsending Classes

This section highlights Emailing Classes and talks about the classes dealing with sending emails to users within the Webrouting Application.

### 4.6.1 Emailing Class Information

\* Deprecated: Constructor methods are only used for testing, **@Autowired** Annotation handles this in Application

\* Classes can be located at: edu.sru.thangiah.webrouting.mailsending

1. **Emailing:** Sets up the Interface for Emailing with methods for the verify Account, send Verification, find by Verification Code, and save Verification Code.
2. **EmailingImpl:** Implements the Emailing class and is annotated as a service, contains methods relevant to sending any form of emails to users.
3. **MailSending:** Class used for a helper to the **EmailingImpl** for parsing applications servlet path.

## 4.7 ReCAPTCHA Classes

Section 4.7 overviews classes relating to the implementation of the Google ReCAPTCHA v2.

### 4.7.1 ReCAPTCHA Class Information

\* Deprecated: Constructor methods are only used for testing, **@Autowired** Annotation handles this in Application

\* Classes can be located at: edu.sru.thangiah.webrouting.captcha

1. **ReCAPTCHA:** Class used to receive Captcha responses, and post the object to the API for Google ReCAPTCHA
2. **RecieveCaptcha:** Helper class used for googles response values are annotated with the @JsonProperty annotation due to how captcha responds.

## 4.8 Service Class

This section will talk about service classes within the **Webrouting Application**.

### 4.8.1 Service Class Information

\* Class can be located at: edu.sru.thangiah.webrouting.services

**ApiServiceImp**: A class which handles fetching information from the Google Maps API. One fetching by latitude and longitude and the other fetching the distance between origin and destination locations. It also formats String return to be a certain length.

**NotificationService**: Sets up the Interface for Notification Services with a method to add Notifications.

**NotificationServiceImpl**: Implementation class of NotificationService Interface that includes a method to provide the functionality to create a new notification and send an email if requested by the user.

**SecurityService**: Sets up the Interface for Security Services with an Authenticated method.

**UserService**: Sets up the Interface for User Services which include methods to find by Username, save, find by Otp, find by Email, assign Opt Code, reset Password, create Opt Code, and get Logged In User.

**UserServiceImpl:** Class used throughout the program to implement the UserService Interface methods.

**ValidationServiceImpl**: Class to handle all the validation for the database interactions, including the excel upload data and the data entered through the “add” pages.

## 4.9 Configuration Classes

Section 4.9 highlights classes used to configure various aspects of the application.

### 4.9.1 Configuration Class Information

\* Deprecated: Constructor methods are only used for testing, **@Autowired** Annotation handles this in Application

1. **PopulateDatabase:** Class used to populate database Implements **Springs Application Runner** to run when the application context starts.
2. **RestConfig:** Configuration **c**lass that sets up a bean of ‘RestTemplate” for HTTP requests.

# 5. Testing

This section covers testing within the Webrouting Application

## 5.1 Junit Testing

The Webrouting Application Junit Tests may be found in the projects ***src/test/java*** folder.

All Test classes can be found in their respective packages, test classes implement methods from their non-test versions. All Tests are written with **JUnit 5**. All Tests files can be run at once by right clicking src/test/java and selecting “**Run** **As JUnit Test.”**

There is further information on testing in the Webrouting Testing Manual.

# 6. UML Diagrams

This section will highlight UML diagrams conveying code pertinent to the Webrouting Application.

## 6.1 Class Diagrams

This section contains Class Diagrams of the Webrouting Application

Timeline

Description automatically generated

Figure 1 – Class Diagram

## 6.2 Sequence Diagrams

The section contains sequence diagrams relating to the Webrouting Application.

**Login**

Timeline

Description automatically generated

Figure 2 – Login System Sequence Diagram

**Upload Shipments from Excel/Push Shipments to Auction:**

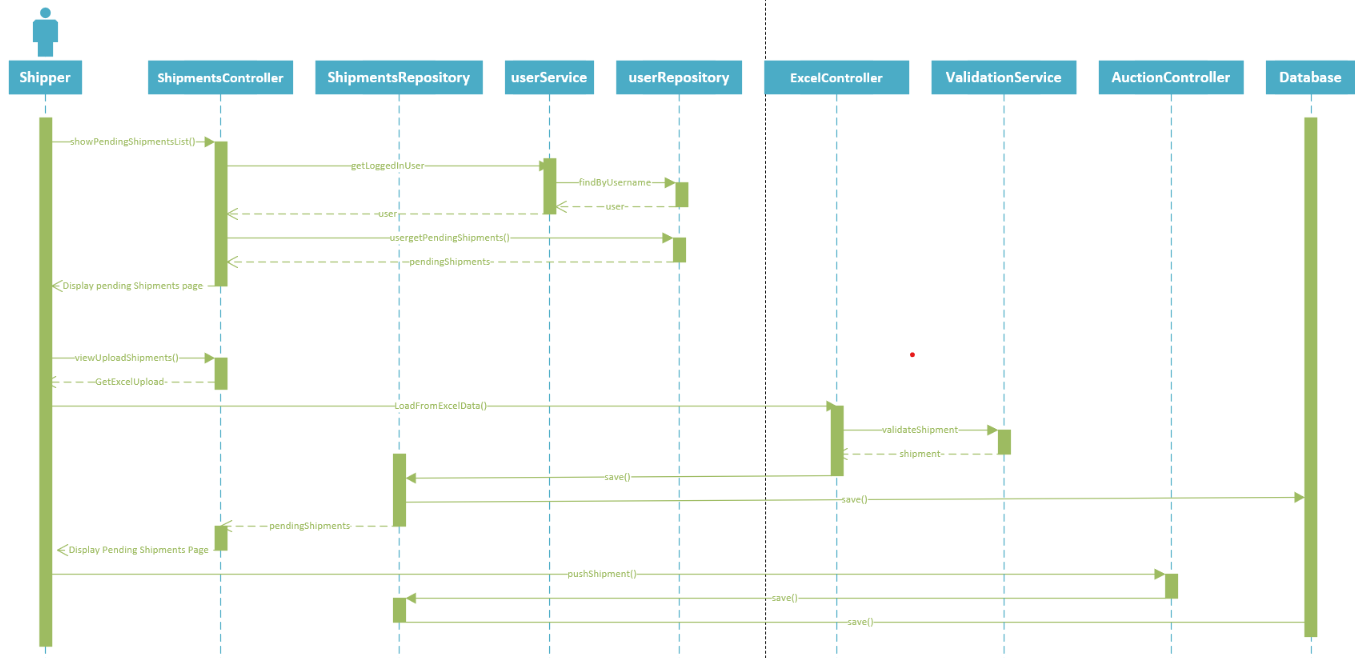


Figure 3 – Upload Shipments from Excel and Push Shipments to Auction Sequence Diagram

**Shipments**

Diagram

Description automatically generated

Figure 4 – Shipments Sequence Diagram

**Carrier Create Bid:**

Graphical user interface

Description automatically generated

Figure 5 – Carrier Create Bid Sequence Diagram

**Shipper Accept Bid:**

Graphical user interface, chart

Description automatically generated

Figure 6 – Accept Bid Sequence Diagram

**Carrier User Auction:**

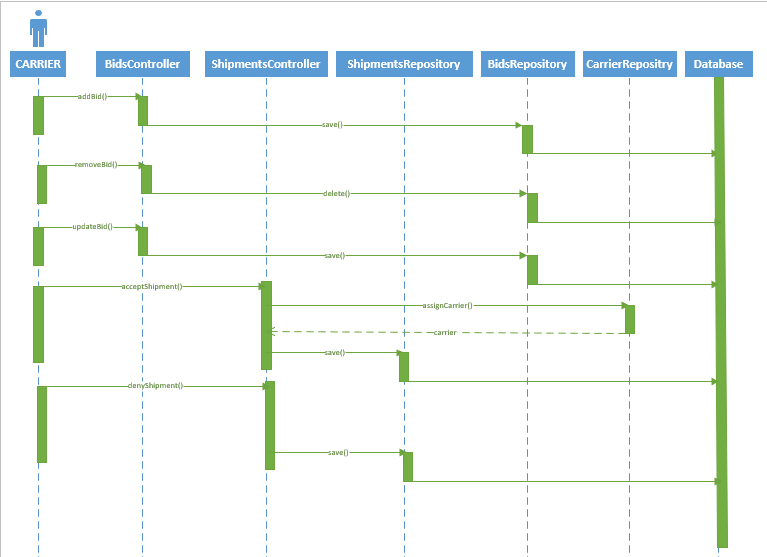


Figure 7 - Carrier user Auction Sequence Diagram

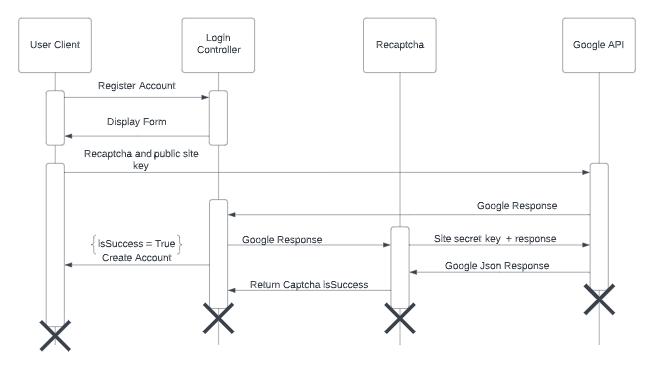
**Maintenance Order:**

Timeline

Description automatically generated

Figure 8– Maintenance Order Sequence Diagram

**Captcha:**

Figure 9 – Captcha State Sequence Diagram

## 6.3 Activity Diagrams

This section contains activity diagrams of functionality in the Webrouting Application.

**Email Verification:**

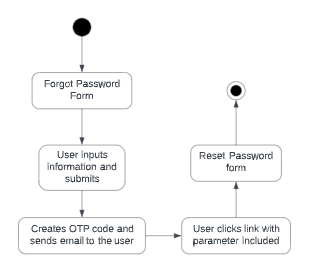


Figure 11 – Email verification diagram

**Carrier User**: Applies to Contacts, Drivers, Locations, Maintenance Orders, and Vehicle Types. (Fleet management)

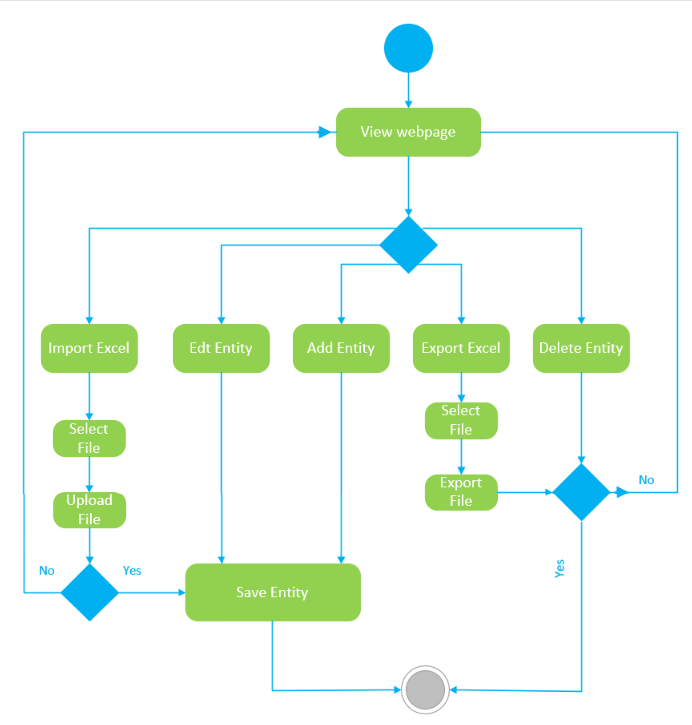


Figure 10 – Carrier Fields Activity Diagram

**Carrier Add Bids**

Graphical user interface, application

Description automatically generated

Figure 12 – Add Bid Activity Diagram

**Shipper Accept Bid:**

Graphical user interface, application, table, Excel

Description automatically generated

Figure 13 – Accept Bid Activity Diagram

**Update Maintenance Order:**

Diagram

Description automatically generated

Figure 14 – Update Maintenance Order Activity Diagram

## 6.4 State Diagrams

Section highlights and shows state diagrams relevant to the Webrouting Application

**Email Verification:**

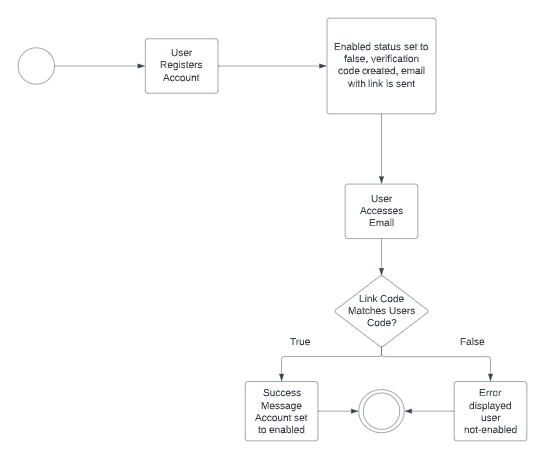


Figure 15 – Email Verification State Chart Diagram

## 6.5 use case Diagrams

Section 6.5 Will highlight and show the Use Case Diagrams relevant to the Webrouting Application

**Admin User:**

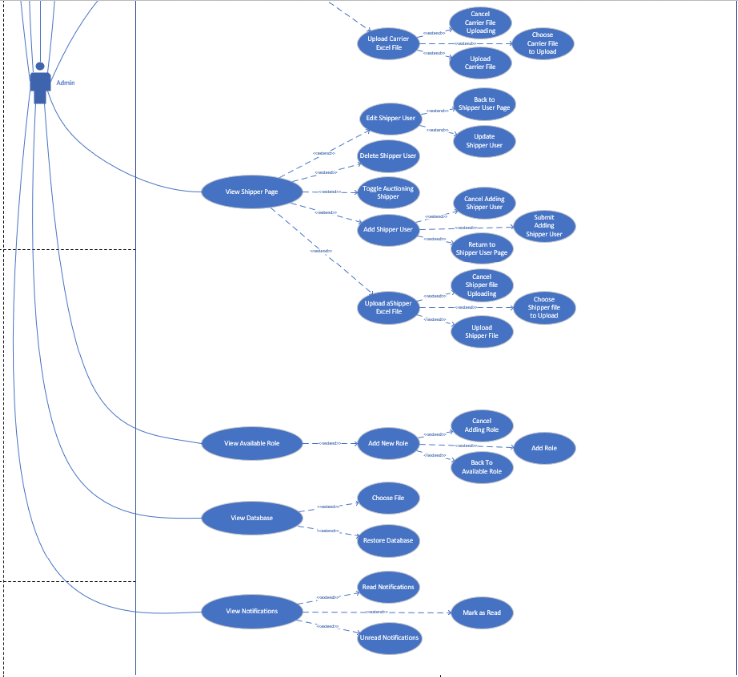
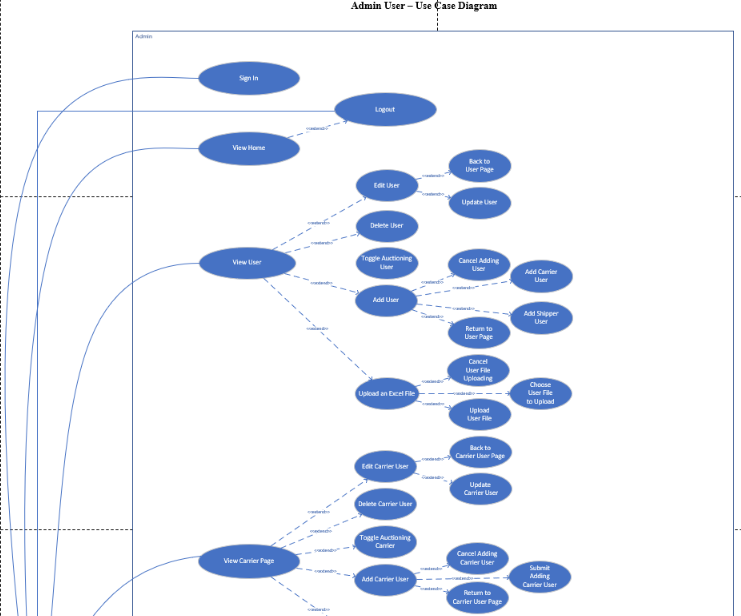


Figure 16 - Admin User Use Case Diagram

**Carrier User:**

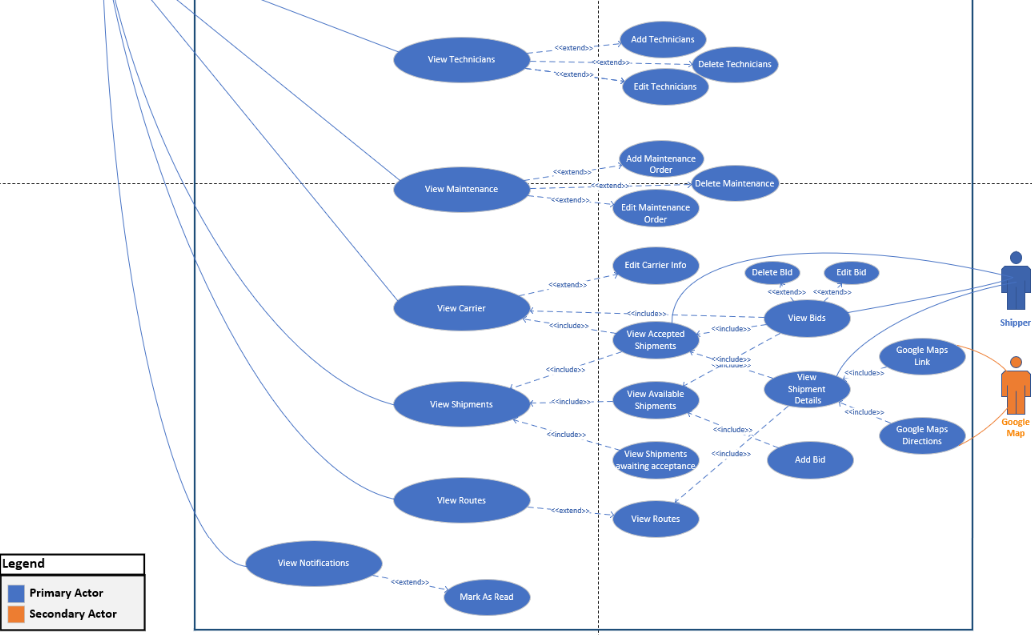
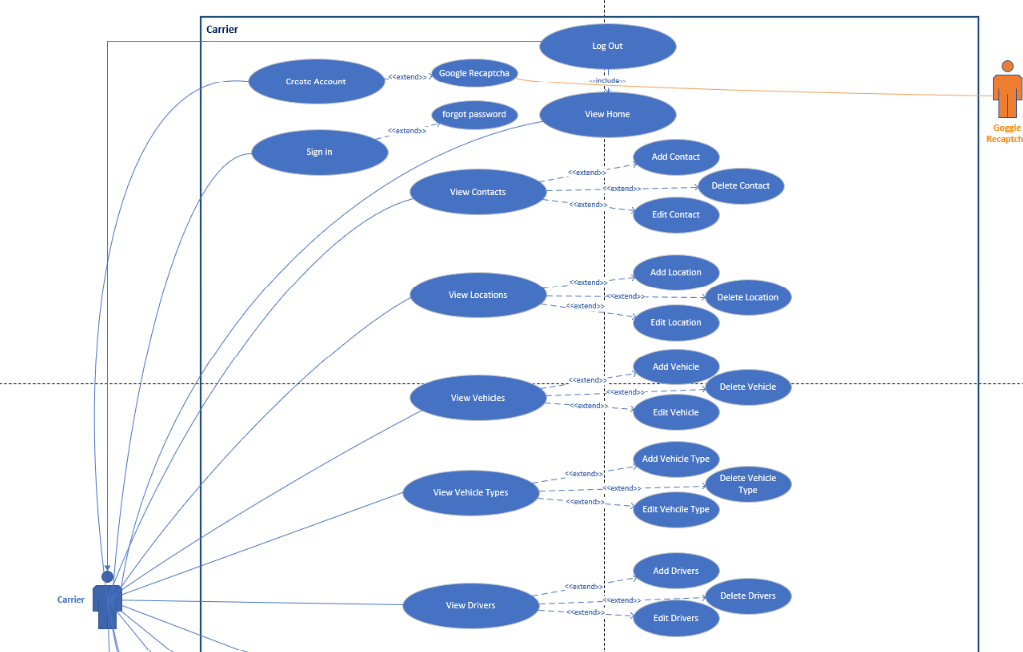


Figure 17 – Carrier User Use Case Diagram

**Auctioneer User:**

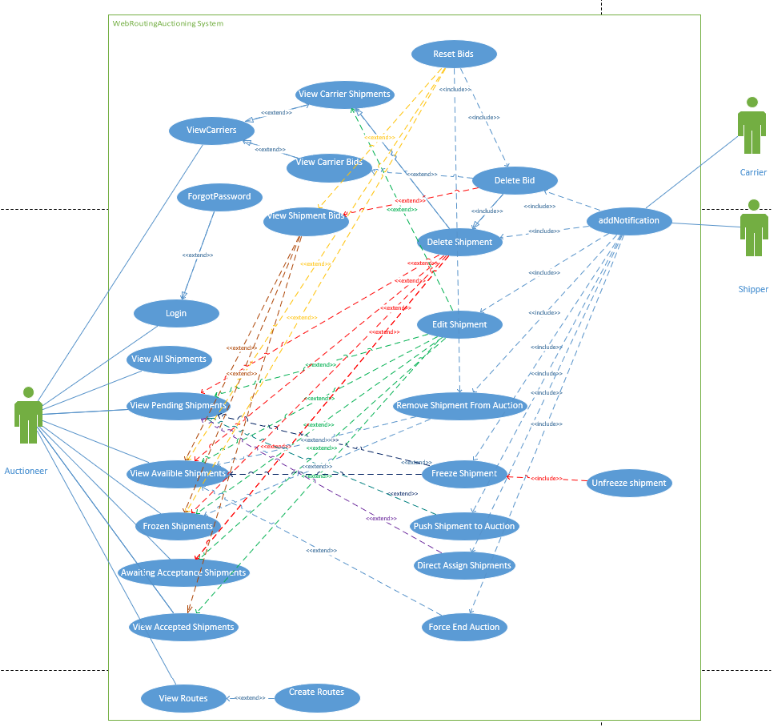
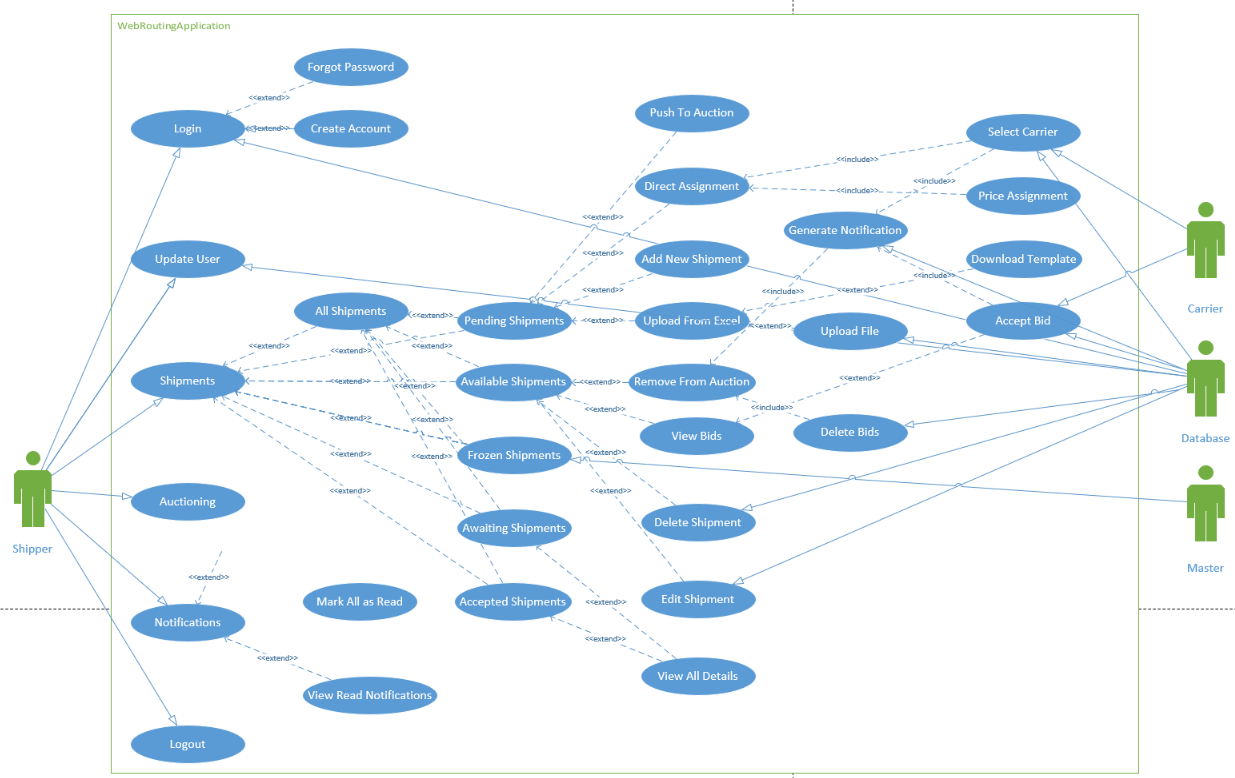


Figure 18 - Auction User Use Case Diagram

**Shipper User:**

Figure 19 – Shipper User Sequence Diagram

# 7. Figures

Figure 1 – Class Diagram 12

Figure 2 – Login System Sequence Diagram 13

Figure 3 – Upload Shipments from Excel and Push Shipments to Auction Sequence Diagram 13

Figure 4 – Shipments Sequence Diagram 14

Figure 5 – Carrier Create Bid Sequence Diagram 15

Figure 6 – Accept Bid Sequence Diagram 15

Figure 7 - Carrier user Auction Sequence Diagram 16

Figure 8– Maintenance Order Sequence Diagram 17

Figure 9 – Captcha State Sequence Diagram 18

Figure 10 – Carrier Fields Activity Diagram 18

Figure 11 – Email verification diagram 19

Figure 12 – Add Bid Activity Diagram 19

Figure 13 – Accept Bid Activity Diagram 20

Figure 14 – Update Maintenance Order Activity Diagram 21

Figure 15 – Email Verification State Chart Diagram 22

Figure 16 - Admin User Use Case Diagram 23

Figure 17 – Carrier User Use Case Diagram 24

Figure 18 - Auction User Use Case Diagram 25

Figure 19 – Shipper User Sequence Diagram 26