RouteMaster: Technical Report

Team Members: Matthew Crosby, Danny Luther, Maxim Mamotlivi, Brian Myers, Seth Opatz

Problem Statement

Truck shipments at Northern Haserot utilize a Unix server and a Microsoft SQL server to determine cargo. The two servers currently have no direct connection with each other. The information is manually input into the Unix Server for the data to then be sent to a Pallet Builder application. This is done on a command line with no graphical interface. Our task is to help automate this process by designing a GUI with data visualisation.

Elicitation Plan

Questions asked:

Main issue to solve:

- **Inserting route assignments to delivery trucks to the company Unix server is done manually, line by line**
- **This problem wastes time, as it is required to be run for every truck**

Users of the GUI, roles, limitations, training:

- **Regular employees, roles consist of admin only, no limitations on this role, minimal training is required**

Do you have an existing database that would need to be linked with the new GUI?

- **We have an existing database that the GUI will need to pull data from, but it will send the data to the Unix server not the database**

If not, what categories/columns need to be in the database? Truck numbers, departure date, etc.?

- **New logging table only**
- **There will be no new table for truck numbers, departure date, etc.**

How large is the database expected to be/ will we need to implement some form of scheduled data removal?

- **Table creation required for logging user actions. Other data is pulled from RoadNet and the Unix server**

Do you want to be able to access this, on any network or only on your network?

- **Only on the company network**

Will we need access to a VPN or VM for your network to work on the GUI?

- **Will only run correctly on the company network**
- **We have access to mockup data to simulate real data retrieval with**

What data will the GUI retrieve from the database? What data will the GUI send to the destination database? What manipulations of the database will be required?

- **Taking route numbers from one app to another**
- **Example of a Unix command we will run: “jrun programName truck# route# route#”**

How is the GUI to be set up? Target aesthetics, formatting, etc.

- **Company colors, dark mode please. (Blue and Gold)**
    - **Dark Blue 14,62,155**
    - **Rich Blue 21,93,233**
    - **Light Blue 127,204,247**
    - **Hover Grey 199,199,199**

How should the corresponding documentation be formatted?

- **Miro diagrams and docx files**

How should errors be logged? What level of error logging is expected?

- **Info logging, (user did x operation at x time) provided file path, No error logging needed**

Will we need to implement an admin page so that certain users have more permissions?

- **No, but we will have windows verification logic to ensure the app is run on the company network by appropriate users**

Security is a concern, what level of security is desired?

- **We will handle security ourselves since the application can only be used on our network**

Usernames, passwords? SQL password verification? Other…

- **Already on a trusted device, Windows username for info logging purposes. Validate windows username, no password for this app**

APP name?

- **RouteMaster**

Are there any integrations with current systems that you would like.

- **Windows validation**
- **Low priority**

What python testing framework should we use?

- **Pytest, separate file, make it look good**
- **We will be inserting Unix commands into the Unix server**

Folder layout:

- **Main Repo**
- **Python folder**
  - **Py Files**
  - **Tests of py files**
- **Input Files**
- **Output Files**
- **Logs**
- **ReadMe**

Things to consider:
Dynamic folder paths, validate directory
snake_case

Assumptions and Risks

We will be assuming the following:

- The route numbers and information in RoadNet are accurate and line up with the route numbers that the Unix server is expecting
- RoadNet is up to date with its information

We will be risking the following:

- If RoadNet is out of date incorrect information is sent to the unix server
- A negative effect of this is that the unix server will not allow the command to be executed

Functional Requirements:

Data Retrieval: Modules which connect to the RoadNet database and retrieve data.

Run Processes: Functions which run processes on the remote Unix server.

Database Info Logging: Database tables which log user actions and commands.

Data Visualization & Interface: Graphs, charts and other structures that facilitate the assignment of routes to trucks and display statistics/information.

Error Recognition & Recovery: Informative error detection systems offer error correction actions to the user, including the restoration of error free states.

Solution Statement

The ultimate goal is to produce a GUI that anyone without Unix training can use so that the tasks can be migrated out of the IT department. The secondary objective is to visualize the data already present in the database to help the end user make an informed decision with their operations.

Deliverables

- GUI application
- GUI has a main tab, which contains a list of routes with a dropdown to select the trucks to assign to the routes.
- GUI has a help tab that has RouteMaster's user guide to help people new to the software learn how to use its different functions.
- GUI has a data tab, which holds a table with a change-log of what was done and who did it at what time.
- GUI needs a graph tab, which has a few graphs that we can view. These tables pull live data from the RoadNet DB for users to view and evaluate.
- Log DB that holds the logged info.
- We need to connect through SSH to the Unix Server to send commands.

- We need to connect to the RoadNet DB through local network connection.
- We need error handling to deal with a number of possible errors.
- We need data validation when users submit their changes.
- Light Mode/Dark Mode (preferably toggle-able)

Definitions

- NHB - Northern Haserot Brent, our customer
- RoadNet DB - RoadNet (or RN) Is the main Database that holds data that we pull for routes, graphs and visualizations. We write our changes to RoadNet by sending commands to the company's server. This is already created and we do not modify it.
- Log DB - Log Database has the table where we store all the logged information for our app. This DB is made by us, not provided by NHB.
- Unix Server - The Unix Server is the end of our data. The changes that we make in RoadNet are sent to it through an SSH connection and a command that is already used at NHB.
- RouteMaster - RouteMaster (or RM) is our application that allows its users to make truck assignments for shipping routes. There are also data visualization graphs that pull from RoadNet to help the users make more informed decisions.
- Routes - Routes are numbered delivery routes that are compiled and inserted into the RoadNet DB by the Unix Server. These only need one truck assigned per route.
- Trucks - Trucks have numbers that are assigned to each route to then make the deliveries.

System Architecture

Block Diagram

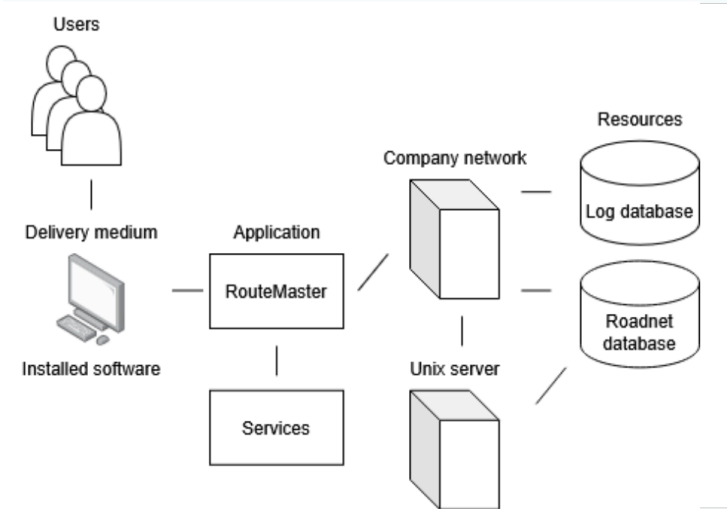

Figure 1.1: block diagram for RouteMaster systems architecture, a client-server architecture allowing employees of Northern Haserot to modify their RoadNet company database

- Users: Employees at Northern Haserot
- Delivery medium: GUI frontend (downloaded software)
- Application: RouteMaster
- Functionality: Route and Truck assignments, Data visualization, Error recognition and recovery
- Services: Data retrieval from: RoadNet database, log database. Run processes on Unix server, Version validation, Database info logging
- Resources: Log database, RoadNet database

Use-Case Diagram

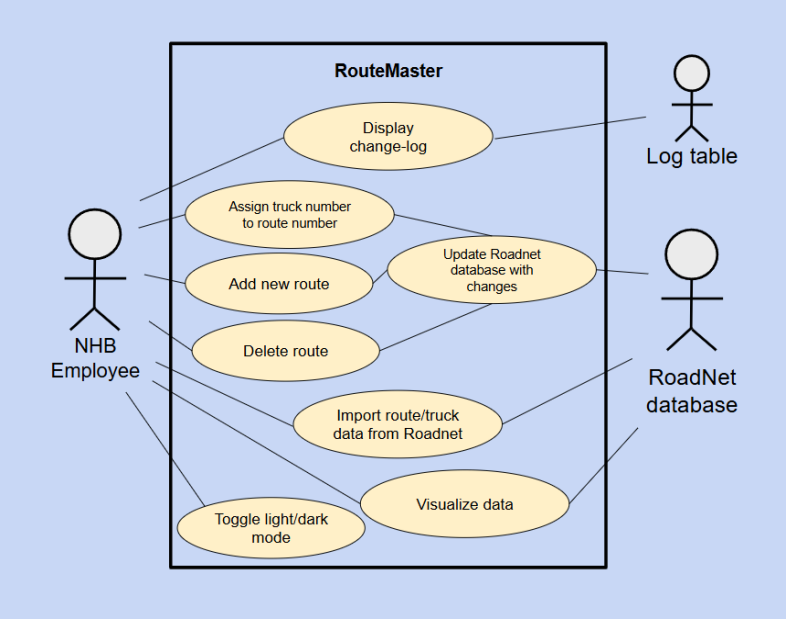

Figure 1.2: Use-case diagram displaying all actions the users of RouteMaster can take to modify and visualize the data in the RoadNet database

Context Diagram

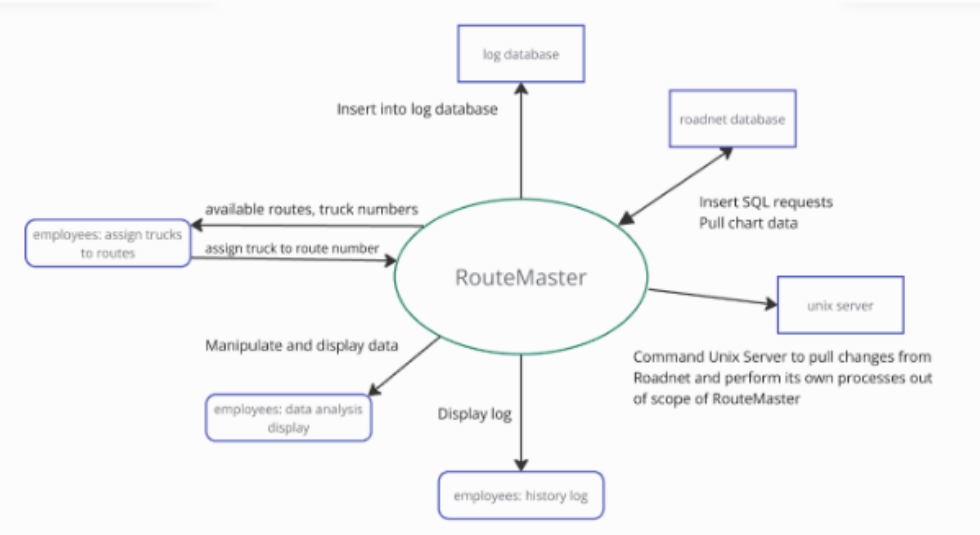

Figure 1.3: Context diagram showing the basic interactions RouteMaster has

Data Flow Diagram

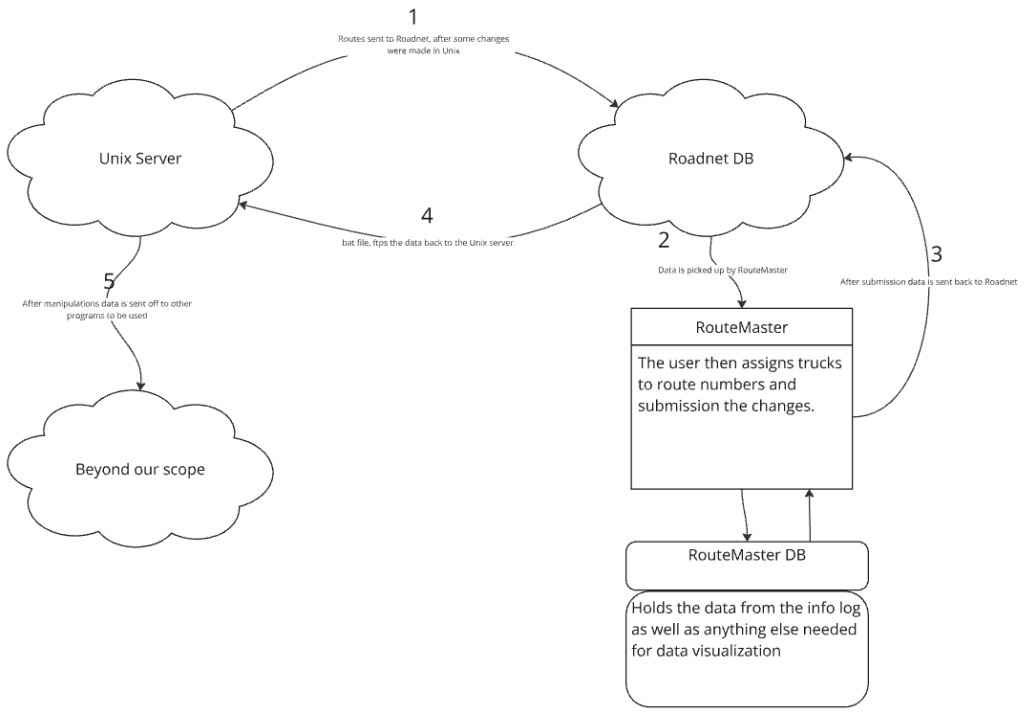

Figure 1.4: Data flow diagram displaying the general route data flows through RouteMaster`

User Interfaces

Aesthetic Guidelines

Company RGB colors as listed, to be implemented in dark mode; hierarchy at the discretion of developers:

- **Dark Blue (14,62,155)**
- **Rich Blue (21,93,233)**
- **Light Blue (127,204,247)**
- **Hover Grey (199,199,199)**

Screen Partitioning

Screens are partitioned by tabs according to their prospective significance in usage. The “Home” tab is opened by default and contains all database-interactive elements. The “Help”, “Data”, and “Map” tabs follow with respect to the aforementioned scheme.

Screens and Contents

Home Tab:

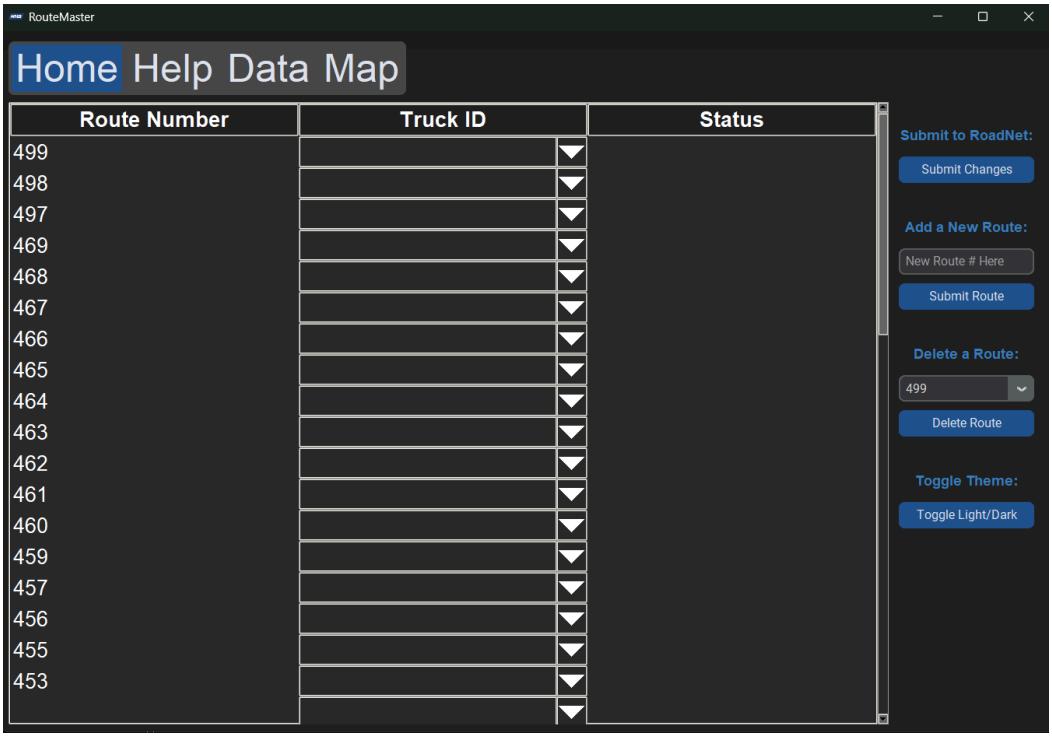

Figure 2.1: Home screen of RouteMaster containing all the important database modification functionalities

Help Tab:

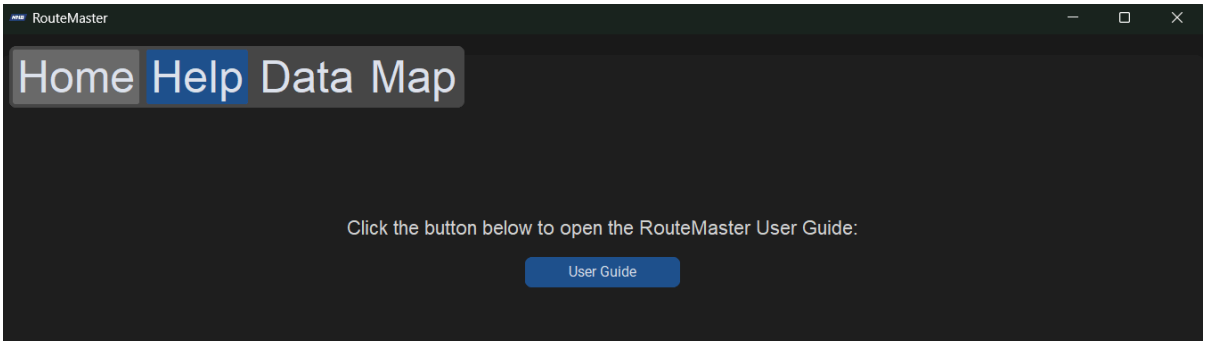

Figure 2.2: Help page containing a button that sends the user to RouteMaster's User Guide

Data Tab:

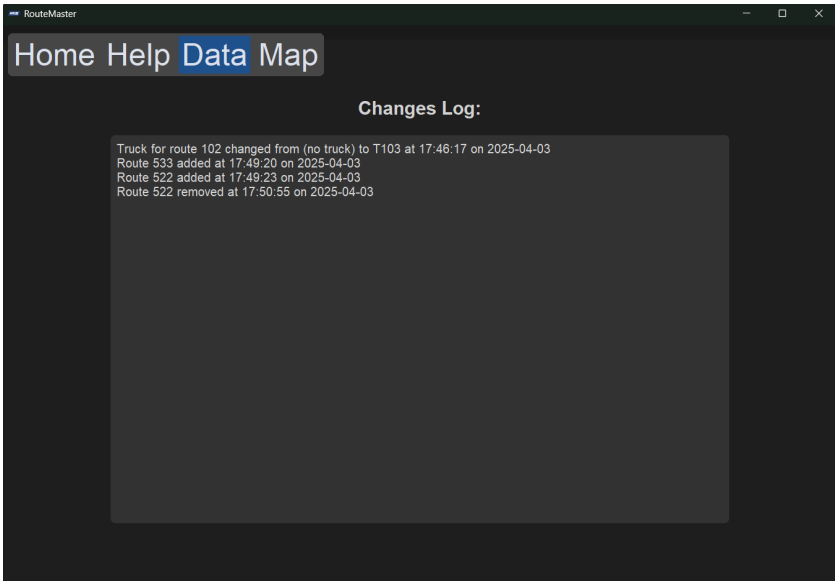

Figure 2.3: Change-log containing all submitted changes made on the “Home” tab, including when these changes were made and who made them

Map Tab:

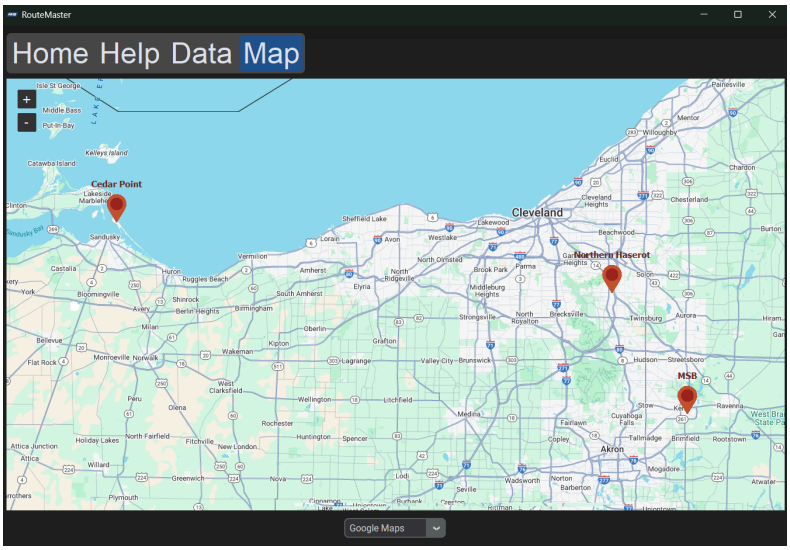

Figure 2.4: Map containing labeled points located at shipping route destinations

The Following Figures Show Report and Alert Messages:

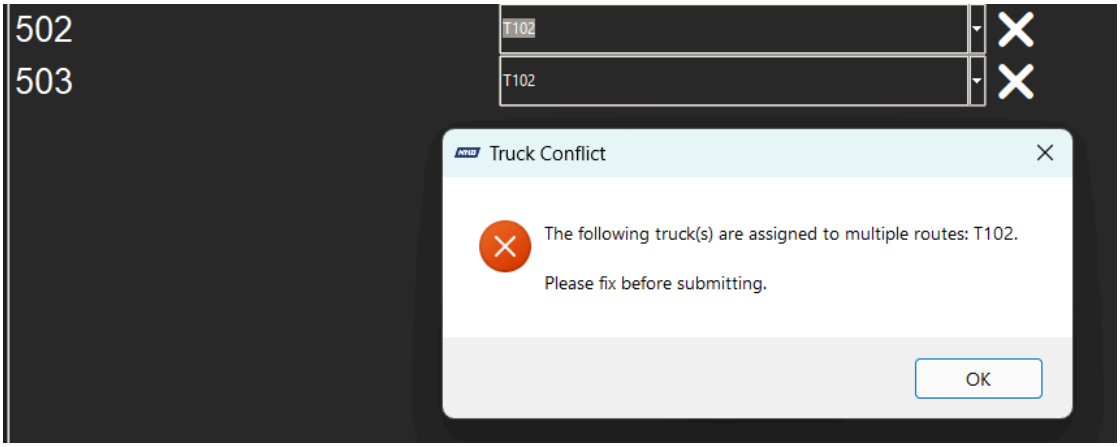

Figure 3.1: An error message detailing confliction will appear if trucks are double-booked onto multiple routes

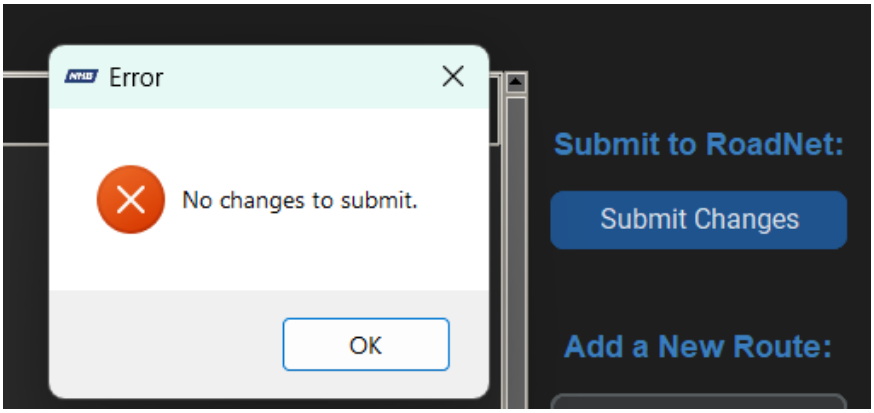

Figure 3.2: An error message informing the user that no changes are currently staged for submission

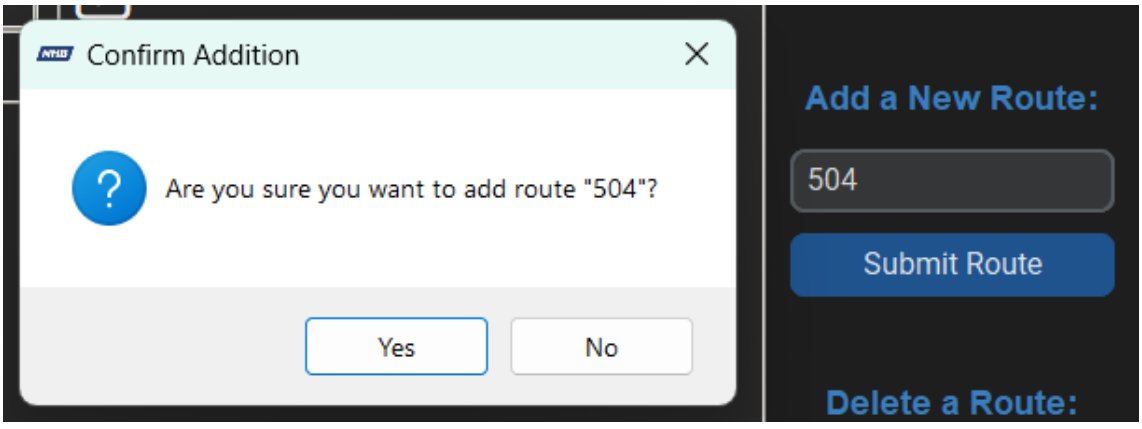

Figure 3.3: A confirmation message will appear when a route addition is attempted

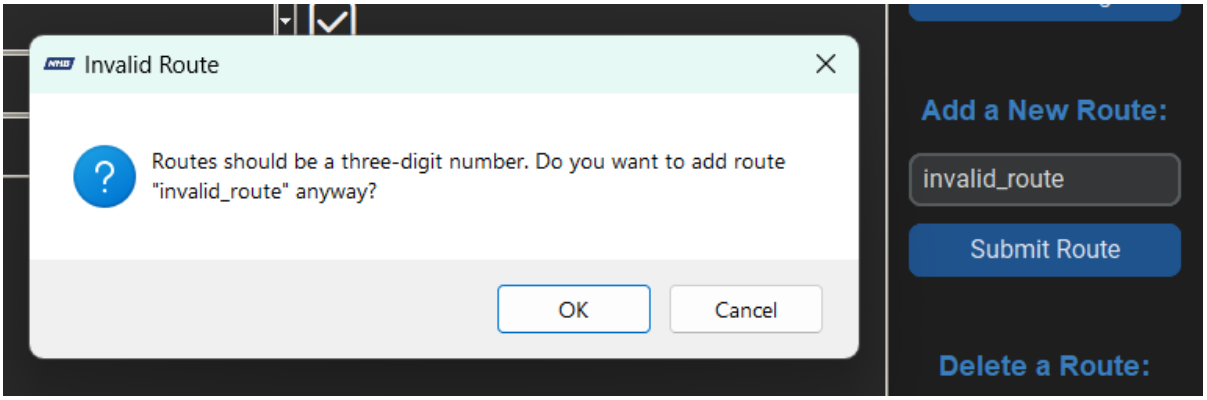

Figure 3.4: A warning message telling the user that the route number they have input for a new route does not adhere to the general route number format (a 3-digit number)

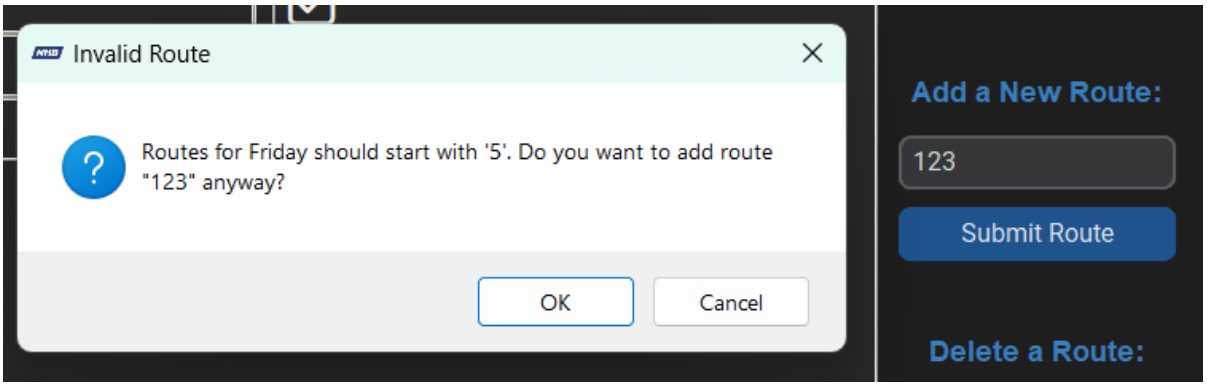

Figure 3.5: A warning message telling the user that the first number of their new route to add does not correctly correspond to the shipping date (1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday)

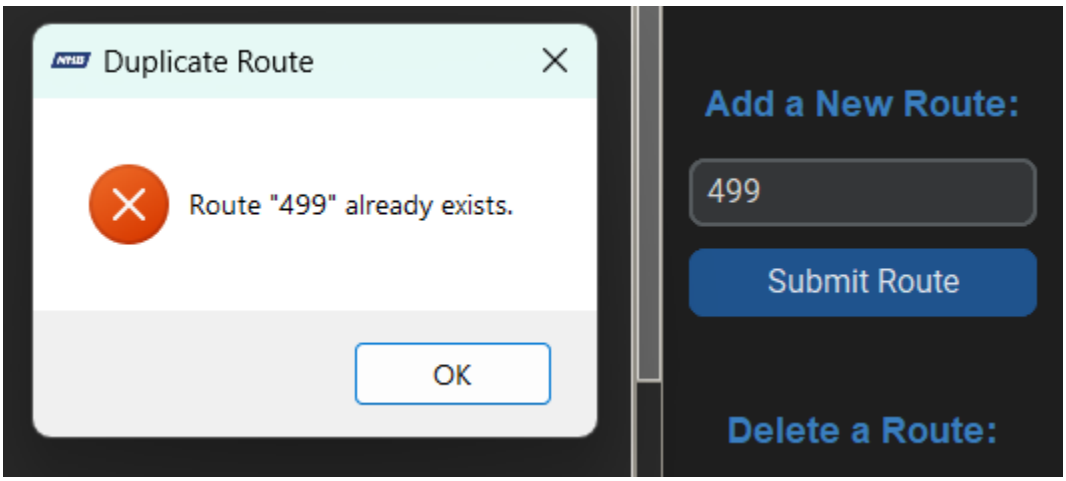

Figure 3.6: An error message telling the user that their submitted route number for a new route already exists

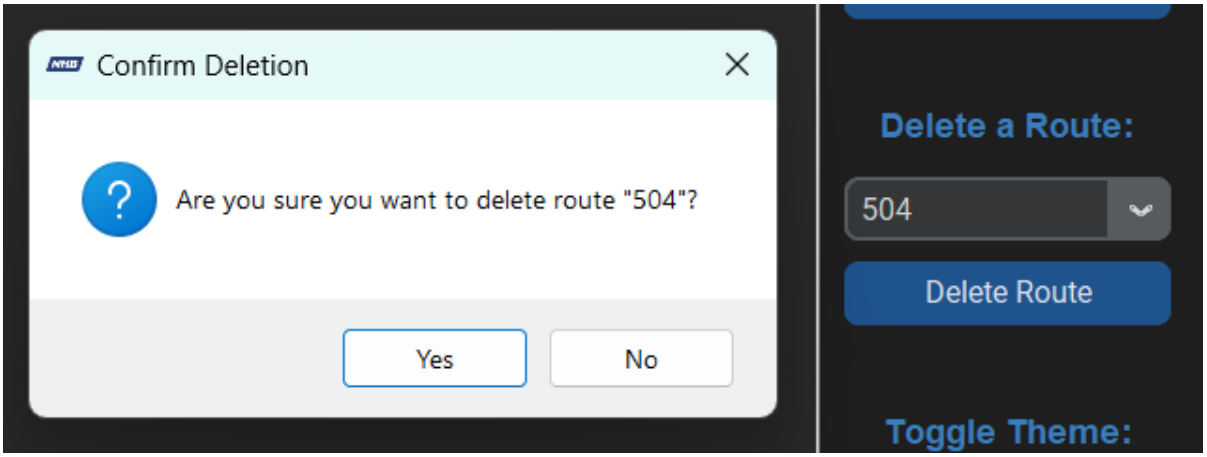

Figure 3.7: A confirmation message will appear when a route deletion is attempted

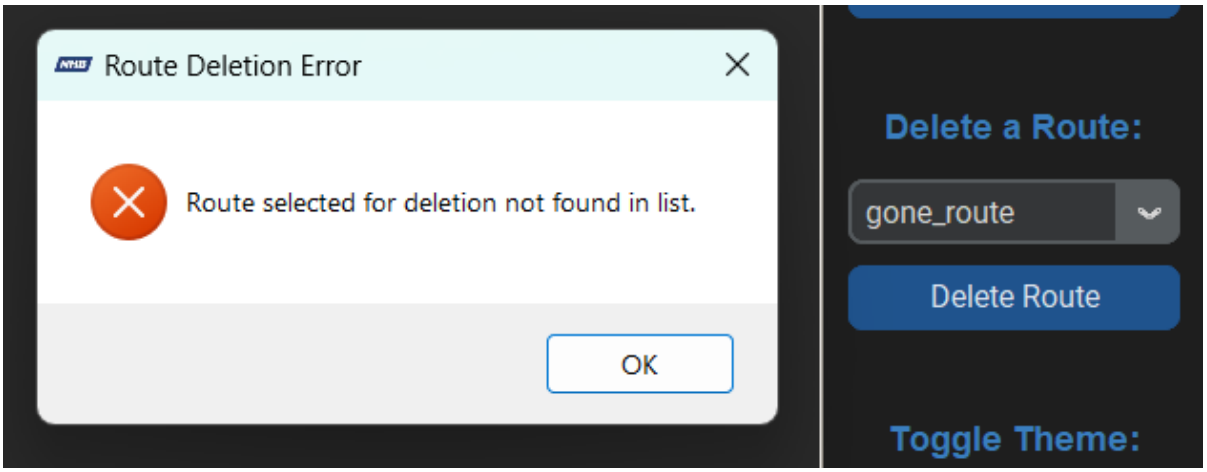

Figure 3.8: An error message is displayed when the user inputs a route for deletion that does not exist

**For documentation regarding user instruction, please see the User Guide*

Quality Control

Route Management: Tests are put in place to verify that addition, deletion, and editing of routes work as intended. There is error handling when invalid operations are performed. i.e. two routes are assigned to a truck, when a truck can only perform one route for the day.

Unix Server: Tests that the commands run correctly and successfully updates RoadNet.

User Interface: The UI is designed to be intuitive for usability and navigation. The drop down menus feature a drop down arrow and allow users to scroll within them. The table with which work is performed is also scrollable. The navigation consists of large buttons with large text at the top of the application. The design follows what a typical user would expect.

Interactive Feedback: Text box alerts inform the user of invalid operations. When a truck receives an assignment and the process completes, there is a checkmark put next to it. When the process is invalid there is an ‘x’ symbol put next to it. In process operations use a circle symbol. Hovering over an intractable button changes its color, and the current page within the navigation bar is highlighted to indicate that it is the current page.

Documentation: There is a tab to allow a user to access a user guide. The manual indicates how to install and run the software. It also describes how to use the software.

OA&M

We have created a user manual for RouteMaster, viewable at the following link: https://docs.google.com/document/d/1LH2F8eSt-ZdQ2vnOupn6iZ33jodsoaygvrYyrL9S_Mg/edit?tab=t.0#heading=h.xwubp2257nw5

Ethics of RouteMaster

All protocols and responsibilities are based on the ACM code of ethics, which can be found here: https://www.acm.org/code-of-ethics

Professional responsibilities of RouteMaster:

- 2.1: Make our solution fulfill all requirements set forth by the customer
- 2.1: Proper truck number assignment, avoiding logistical issues
- 2.4: Provide help/support to the user as well as manual/documentation
- 2.2: Validate input data so improper data is not kept
- 2.8: Ensure that sensitive data is not displayed to unauthorized users
- 1.7 & 2.9: Implement NIST security
- 2.1: Product is sustainable with company work flow and will be used in the future

Protocols in Case of Principle Violations:

- Protocol - 1: Meet with the stakeholder to confirm the software acts as intended
- Protocol - 2: Ensure with tests that all assignments occur correctly
- Protocol - 3: Create a “Help” tab to provide the user with support
- Protocol - 4: Data shown to the user is limited and controlled

Ethical issues of RouteMaster:

- 1.6: Possibility for authors of code to include malicious elements such as a backdoor or a worm
- 1.4: We must ensure that our program is accessible by all protected classes (examples: color blind, visually impaired, etc)
- 1.6 & 1.7: Truck location/driver information should be kept secure
- 1.7: Handle data responsibly to avoid leaks or misuse
- 1.2: RouteMaster will not unintentionally create inefficient route assignments
- 1.5: Decisions made and actions taken are logged to distinguish authorship

Protocols in Case of Ethical Violations:

- Protocol - 1: All code written is for the purpose of furthering the functionality of the application, nothing else
- Protocol - 2: Application will not use color coded features, GUI features big readable text and large interactables
- Protocol - 3: Access to information pertaining to anything confidential is strictly accessed on company network
- Protocol - 4: All actions performed on application by client must be logged and accounted for with timestamps, etc.

Legal issues of RouteMaster:

- 1.6: Breaking data protection rules such as the ones laid out in laws like GDPR when storing data
- 2.3: Automation leading to violations of labor agreements

Protocols in Case of Legal Violations

- Protocol - 1: Research the data that we are handling and ensure that any sensitive data is handled properly.
- Protocol - 2: Research union bylaws to ensure that our automation does not violate NHB's agreements with the union terms.
- Protocol - 3: Ensure not to handle data or send commands that go beyond the intended scope of our project.

Error Recovery

Errors are detected and handled in two ways:

The first way is to validate user inputs so that users can not send an incorrect command to the Unix Server:

- If the user tries to double book a truck, they will get an error message and be unable to submit changes until they resolve the issue
- When adding a route, the user cannot give this new route the same route number as an already existing route, and attempting to do so will produce an error message
- When adding a route, the user gets a warning message if they attempt to give this new route a route number that is not 3 numerical digits with the first number corresponding to the shipping date
- If the user attempts to delete a route that does not exist, they will be unable to do so, and RouteMaster will produce the user an error message
- If the user attempts to submit without route changes an error message will also be produced.

The second way is to ensure that the command runs. This can be validated after the command is run. After a successful command is run, there is a new file in the Unix system that is tied to this route. We make sure that this file exists, then we can imply that the command was run correctly.

Summary/Conclusion

The RouteMaster program is being made to reduce the workload on the NHB IT team, who currently enters truck assignments directly into the Unix terminal. It is a GUI that anyone at Northern Haserot can utilize, not just the IT department. We have used Python with the tKinter library to code this GUI. The program pulls information from the RoadNet database, allows the user to make changes such as assigning trucks to routes, adding new routes, etc, and pushes these changes to the company's Unix server. This Unix server then modifies RoadNet with the new assignments and/or routes. The Unix server can only be accessed on the company network, meaning the program's security is the pre-existing security on Northern Haserot's network.

Acknowledgements

Stakeholder: Tyler Redinger, Northern Haserot

Professor: Augustine Samba, Kent State University