Gateway Parcel Service Routing Optimization Application

[Part A: Letter of Transmittal 2](#_Toc209783934)

[Part B: Executive Summary 3](#_Toc209783935)

[Project Summary 3](#_Toc209783936)

[Data Summary 3](#_Toc209783937)

[Implementation 4](#_Toc209783938)

[Timeline 4](#_Toc209783939)

[Evaluation Plan 5](#_Toc209783940)

[Costs 5](#_Toc209783941)

[Part C: Application 6](#_Toc209783942)

[Descriptive and Non-descriptive Methods 6](#_Toc209783943)

[Datasets and Data Preparation 6](#_Toc209783944)

[Exploration and Preparation Algorithms 6](#_Toc209783945)

[Interactive Queries and Decision Support 6](#_Toc209783946)

[Machine Learning and Model Implementation 6](#_Toc209783947)

[Accuracy Evaluation 7](#_Toc209783948)

[Security, Monitoring and Maintenance 7](#_Toc209783949)

[User-friendly Dashboard 7](#_Toc209783950)

[Part D: Post-implementation Report 8](#_Toc209783951)

[Solution Summary 8](#_Toc209783952)

[Data Summary 8](#_Toc209783953)

[Machine Learning 8](#_Toc209783954)

[Validation 9](#_Toc209783955)

[Visualizations 9](#_Toc209783956)

[User Guide 10](#_Toc209783957)

[Follow the steps below to install, run, and use the routing application. These instructions are written for a Windows 10 environment. 10](#_Toc209783958)

[Reference Page 14](#_Toc209783959)

# Part A: Letter of Transmittal

September 25th, 2025  
**To:** Senior Leadership, Gateway Parcel Service  
**From:** Emily Miller, Project Lead  
**Subject:** Proposal for Gateway Parcel Routing Application

Dear Senior Leadership,

I am submitting this proposal for the Gateway Parcel Routing Project to request your approval for implementation. As delivery volumes across St. Louis continue to increase, our current manual route-planning approach has become increasingly inefficient. Drivers must balance deadlines with vehicle capacity and route efficiency without technological support, creating risks of missed deadlines, excess mileage, and higher operating costs.

The proposed solution is a routing optimization application developed in Python that will automatically generate efficient delivery routes for our two trucks. By leveraging package and distance data, the system will calculate routes that minimize total mileage while meeting delivery deadlines and truck capacity limits. The application will also provide management with an interactive interface to view delivery statuses and track overall mileage.

This solution will benefit the organization by reducing fuel consumption, labor costs, and overall mileage while improving delivery reliability. Customers will experience more consistent on-time deliveries, and drivers will spend less time managing complex routes manually. Leadership will benefit from measurable cost savings and improved operational visibility. The project requires modest funding. Development will use free, open-source libraries such as pandas and matplotlib, and testing will be performed on existing hardware. Estimated development labor is 120 hours at $40/hour, totaling $4,800. Additional one-time costs include $500 for local testing equipment, with $600 annually budgeted for software maintenance and updates.

The new routing system will positively impact all stakeholders. Drivers will benefit from reduced planning burden and more predictable workloads. Customers will receive deliveries that are more timely and consistent. Management will gain cost savings and operational efficiency. IT staff will oversee a system that is straightforward to maintain and expand in the future. The project poses no ethical or legal concerns. The system uses only operational delivery data, such as package IDs, delivery addresses, and mileage between locations. No personal or sensitive customer data is involved, ensuring compliance with privacy and security standards.

I bring relevant expertise from my work with algorithms, data structures, and application development throughout my degree program, including prior experience developing routing simulations and data-driven applications. This background ensures that I can deliver a solution that meets Gateway Parcel Service’s operational needs.

I respectfully request your approval to proceed with the Gateway Parcel Routing Project. This system will modernize our delivery operations and provide measurable value to Gateway Parcel Service.

Sincerely,  
Emily Miller  
Project Lead

# Part B: Executive Summary

## Project Summary

Gateway Parcel Service will continue to face increasing pressure as delivery volumes in the St. Louis area grow. Current manual planning methods will no longer be sufficient to balance delivery deadlines, truck capacity, and route efficiency. If this approach continues, the company will risk late deliveries, excess mileage, and higher operating costs.

This project will create a routing optimization tool designed to automatically plan deliveries for two trucks. It will rely on package details and a mileage table to assign shipments and map out efficient routes, while ensuring that delivery deadlines and vehicle capacity limits are met. The finished application will include the routing program and built-in visualizations to illustrate delivery times, mileage comparisons, and per-truck totals. These deliverables will provide the client with a practical, cost-effective solution to improve efficiency, reduce costs, and strengthen customer satisfaction.

## Data Summary

Two structured CSV files will serve as the data foundation for this project: a package file and a distance matrix. The package file lists addresses, deadlines, tracking identifiers, and special delivery notes. The distance matrix provides mileage figures between each address in the delivery network.

Before development, both datasets will be checked for accuracy and formatting issues. The package file will be reviewed to confirm consistent structure, and the distance table will be inspected for gaps or irregular values. If problems are detected, the program will either reference mirrored distance values when available or mark the entries for manual correction.

These files will play a role at every stage of the project. During design, they will shape the program's structure of trucks and packages. During coding, they will provide test cases for routing logic and help confirm that optimization works as intended. Later, the files can be updated with new addresses or revised distances so the system stays accurate as business needs change. Since the information only covers operational details and does not include personal data, no ethical or legal concerns will arise from its use.

## Implementation

Development will follow an Agile style, with features added and tested in short cycles. The first stage will focus on building reliable routines for loading and processing the package and distance data. Next, a greedy algorithm will be applied to generate preliminary delivery routes. This approach will be enhanced with a 2-opt optimization technique to reduce mileage further. Once the routing functionality is complete, additional features such as visualizations and interactive queries will be added.

Testing will occur throughout each phase to confirm functionality, accuracy, and stability. Working in increments will make it possible to check progress early, fine-tune the algorithms as needed, and verify that the finished program aligns with the company's requirements.

## Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone / Deliverable** | **Resources** | **Dependencies** | **Start and End Date** | **Duration** |
| Data preparation & validation | Developer, Python 3.11 | None | Oct 01 - Oct 02 | 2 days |
| Baseline routing | Developer, Python 3.11 | Data prep | Oct 03 - Oct 05 | 3 days |
| Optimization | Developer, Python 3.11 | Baseline | Oct 06 - Oct 08 | 3 days |
| Visualization features | Developer, Python 3.11, Matplotlib | Optimization | Oct 09 - Oct 10 | 2 days |
| Verification & validation | Developer, Python 3.11 | Visualization | Oct 11 -Oct 12 | 2 days |
| Packaging & final review | Developer, Python 3.11 | All prior | Oct 13 -Oct 15 | 3 days |

## Evaluation Plan

**Verification (build correctness):** Unit tests will confirm that the data is loaded properly, that packages are not duplicated across trucks, and that all listed addresses are accessible within the routing system.

**Validation (model performance):** Performance will be validated against a manual-planning baseline using total miles traveled. Manual planning averages roughly 135 miles, while the optimized program is expected to average about 112 miles. This 17% reduction will confirm the solution’s measurable benefits.

## Costs

* **Labor:** 120 developer hours @ $40/hr = $4,800 (one-time).
* **Hardware/Testing:** Windows 10 test laptop = $500 (one-time).
* **Software:** Python 3.11, Matplotlib, VS Code = $0 (open-source).
* **Maintenance:** Annual data updates and support = $600/year.

**Total initial investment:** $5,300.

**Ongoing annual cost:** $600.

The operational savings achieved through reduced fuel consumption, lower labor hours, and improved on-time delivery performance will offset these costs.

# Part C: Application

## Descriptive and Non-descriptive Methods

The application includes both descriptive and prescriptive features. On the descriptive side, it produces three visuals: a histogram of delivery times, a bar chart comparing baseline and optimized mileage, and a diagram showing the miles driven by each truck. Together, these visuals give stakeholders a clear picture of delivery performance. Nondescriptively, the system applies a greedy algorithm followed by a 2-opt optimization process. This prescriptive method produces specific delivery routes that minimize mileage while respecting deadlines and capacity limits.

## Datasets and Data Preparation

The application relies on two structured CSV files: one with package details such as IDs, addresses, deadlines, and notes, and another with the distances between delivery points. Before the data is used, both files are loaded into Python structures and checked to ensure the formatting is consistent. Missing or irregular distance entries are corrected using symmetric values or flagged for review. These preparation steps ensure the datasets are reliable, maintainable, and ready for computation as the business evolves.

## Exploration and Preparation Algorithms

During processing, the system first uses a greedy approach to assign packages to trucks under deadline and capacity constraints. After the initial routes are created, a 2-opt algorithm refines them by testing different stop orders and keeping any changes that reduce total mileage. By repeating this process, the system gradually settles on efficient routes. This workflow shows how raw delivery data is cleaned, organized, and optimized into a practical routing plan.

## Interactive Queries and Decision Support

An interactive console menu lets users try the model directly. From the menu, they can look up a package’s status at a given time, check all package statuses, review total mileage, or see what each truck is carrying. These options turn the program into a decision support tool, giving dispatchers and managers quick answers they can act on during operations.

## Machine Learning and Model Implementation

The optimization process uses machine learning principles by running trial swaps and keeping improvements until no shorter routes remain. While less complex than large predictive models, the combined greedy and 2-opt methods act as a prescriptive approach that steadily improves efficiency. This balance keeps the system straightforward to maintain while still delivering clear performance gains.

## Accuracy Evaluation

The main way to measure success is total mileage compared to manual planning. Manual routing takes about 135 miles on average, while the optimized model reduces that to around 112 miles. This 17% cut shows the program delivers real efficiency gains. The system also verifies that no truck exceeds its 16-package limit and that every package is delivered before its deadline, further confirming accuracy.

## Security, Monitoring and Maintenance

The system relies only on operational details like addresses, deadlines, and mileage, so no personal or sensitive data is ever involved. Built-in checks catch formatting problems and stop corrupted files from being used. Console logs and reports confirm that deliveries are correctly assigned and routes remain efficient. Keeping the system current is simple: new CSV files can replace the old ones whenever addresses or distances change, without altering the code.

## User-friendly Dashboard

The application includes a straightforward dashboard in the form of a console menu. Users pick from numbered options, so no programming experience is required. Three visualizations are automatically created and saved alongside the menu as PNG files, giving evaluators both numbers and visuals to confirm how the system performs. The dashboard and visuals demonstrate that the application is user-friendly, interpretable, and aligned with industry standards for data products.

# Part D: Post-implementation Report

## Solution Summary

Gateway Parcel Service struggled with delivery efficiency because drivers were still planning routes by hand. They were left to juggle deadlines, vehicle limits, and travel distance without software support, which often meant longer routes, higher costs, and the chance of late deliveries. The application I developed took over that process by generating optimized delivery routes for the company’s two trucks. It worked by combining package and distance data to build schedules that cut down on total mileage while still meeting all deadlines and truck capacity limits. As a result, operations became smoother, costs decreased, and deliveries were more reliable.

## Data Summary

The program was built around two simple but essential CSV files: the first listed package IDs, addresses, deadlines, and any special notes. The second contained the distances between delivery locations. Before putting them to use, I validated both files. The raw versions were checked for errors like missing values or inconsistent formatting, and then cleaned versions were prepared so the system could process them without issues.

One crucial detail is that only operational information was used. No sensitive customer data ever entered the workflow, which meant there were no ethical or legal concerns to address.

## Machine Learning

**What:** The system aimed to create delivery routes that used as few miles as possible while meeting time and capacity rules.

**How:** To do this, it first applied a greedy algorithm to build an initial set of routes quickly. Once a workable plan was in place, the program used a 2-opt routine to shuffle the order of stops and test whether a shorter route was possible. Over repeated runs, the model steadily improved the routes until no further gains could be made.

**Why:** This method was chosen because it balanced efficiency with clarity. More advanced optimization techniques, like simulated annealing or genetic algorithms, were unnecessary for a two-truck system. The greedy + 2-opt pipeline delivered strong results while staying straightforward enough to maintain and adapt in the future.

## Validation

This approach fits into the reinforcement learning category because the program kept testing changes and adopting any that improved performance.

Success was measured by comparing the total miles driven against a manual baseline. Manual planning had historically required about 135 miles per day. With the optimized program, that number dropped to roughly 112 miles — a reduction of about 17 percent. This clear improvement showed that the application effectively solved the problem it addressed.

In addition to mileage, I checked that every package was delivered before its deadline and that no truck was assigned more than 16 packages. These checks confirmed the accuracy and correctness of the model.

## Visualizations

Three visualizations were generated to illustrate the performance of the routing system. Each image was created during the program's execution and saved as a PNG file in the working directory.

* **Delivery Times Histogram (delivery\_times\_hist.png):** This chart shows the distribution of package delivery times measured in minutes after 8:00 AM. It highlights when most packages were completed, making it easier to see delivery peaks throughout the day.
* **Mileage Comparison (mileage\_comparison.png):** This bar chart compares the total distance traveled using the initial baseline routes versus the optimized routes. The visualization demonstrates the efficiency gained by applying the optimization algorithm.
* **Miles per Truck (miles\_per\_truck.png):** This chart displays the distance traveled by each truck. It helps illustrate how the workload was divided between the two vehicles and verifies that both trucks were used efficiently.

Together, these images provide evidence of the model's improvements and give management a clear, visual representation of system performance.

## User Guide

## Follow the steps below to install, run, and use the routing application. These instructions are written for a Windows 10 environment.

**Step 1 – Install Python**

* Confirm that Python 3.10 or later is installed.
* Open Command Prompt and type:

Python --version

If Python is not installed, download it from <https://www.python.org/downloads> .

**Step 2 - Install Required Packages**

Open Command Prompt, navigate to the project folder, and install the required library:

pip install matplotlib

Other imports (csv, datetime, collections, math) are built into Python and require no installation.

**Step 3 - Verify Input Files**

Ensure the following files are in the same folder as main.py:

* WGUPS\_Distance\_Table.csv
* WGUPS\_package\_File.csv

**Step 4 - Run the Application**

In Command Prompt, from the project folder, type:

python main.py

The program will launch and display the Main Menu.

**Step 5 - Main Menu Options**

The application provides six options:

1. View the status of all packages

* Displays every package, showing delivery status (At hub, En route, Delivered), delivery deadline, assigned truck, and delivery time.

A screen shot of a computer screen

AI-generated content may be incorrect.

1. Query the status of a single package at a specific time.
   * Enter a time (HH:MM AM/PM) and a package ID to see the status at that moment.

A black screen with white text

AI-generated content may be incorrect.

1. Query the status of all packages at a specific time
   * Enter a time (HH:MM AM/PM) to display all package statuses at that time.

A screen shot of a computer code

AI-generated content may be incorrect.

1. View total mileage traveled by all trucks
   * Displays the combined mileage for both trucks.

A black screen with white text

AI-generated content may be incorrect.

1. View truck loads at a specific time
   * Enter a time (HH:MM AM/PM) to see which packages were on each truck and which had already been delivered.

A screen shot of a computer

AI-generated content may be incorrect.

1. Exit
   * Closes the program.

**Step 6 - Visualizations (Auto-Generated)**

When the program runs, three image files are created:

* Delivery\_times\_hist.png – Histogram showing distribution of delivery times.
* Mileage\_comparison.png – Bar chart comparing baseline vs. optimized total miles.
* Miles\_per\_truck.png – Bar chart showing mileage for each truck.

These files will be saved in the project folder.

**Step 7 - Example Walkthrough**

1. Launch the app (python main.py)
2. From the Main Menu, enter 3 and type 8:45 Am.
3. The system shows the status of all 40 packages at 8:45 AM. Some are delivered and some are en route.
4. Check the folder: the three visualization images (delivery\_times\_hist.png, mileage\_comparison.png, miles\_per\_truck.png) have been created.

**Step 8 - Troubleshooting**

If errors occurred:

* Confirm Python 3.10 or later was installed correctly.
* Verify that matplotlib was installed with pip.
* Ensure the required CSV files (WGUPS\_Distance\_Table.csv, WGUPS\_Package\_File.csv) were in the same folder as main.py.

# Reference Page

No external sources were referenced. All data and requirements were provided by Western Governors University course materials.