September 23, 2025

C964: Computer Science Capstone Template

Task 2 parts A, B, C and D

[Part A: Letter of Transmittal 2](#_Toc209550063)

[Part B: Project Proposal Plan 4](#_Toc209550064)

[Project Summary 4](#_Toc209550065)

[Data Summary 4](#_Toc209550066)

[Implementation 5](#_Toc209550067)

[Timeline 5](#_Toc209550068)

[Evaluation Plan 5](#_Toc209550069)

[Costs 6](#_Toc209550070)

[Part C: Application 7](#_Toc209550071)

[Part D: Post-implementation Report 8](#_Toc209550072)

[Solution Summary 8](#_Toc209550073)

[Data Summary 8](#_Toc209550074)

[Machine Learning 8](#_Toc209550075)

[Validation 9](#_Toc209550076)

[Visualizations 9](#_Toc209550077)

[User Guide 10](#_Toc209550078)

[Follow the steps below to install, run, and use the routing application. These instructions are written for a Windows 10 environment. 10](#_Toc209550079)

[Reference Page 14](#_Toc209550080)

# Part A: Letter of Transmittal

**April 25, 2025**

**Mr. Jordan Reynolds, Operations Director**

Gateway Parcel Service – St. Louis Division

**Subject:** Approval Request – *Gateway Parcel Routing Project*

Dear Mr. Reynolds,

Our St. Louis delivery team continues to face challenges in keeping mileage low while ensuring every package arrives by its promised time. At present, our drivers put routes together by hand, which makes it challenging to juggle delivery deadlines and efficient mileage. As shipments continue to increase, that method could lead to missed deadlines, unnecessary fuel use, and higher staffing costs.

To address this, I recommend building a straightforward tool that will plan the most efficient routes for our two delivery trucks. The system will review package information and travel distances, then recommend an order that completes every stop on time while using the fewest miles possible. Staff will be able to see clear schedules and mileage totals, helping us make fast, well-informed decisions each morning.

The project will require only a modest investment. Development will take about six weeks, using existing office computers and free, well-supported software. No sensitive customer data will be used in this stage; we will work with sample package and distance records to demonstrate results safely.

With my background in data-driven problem solving and experience building similar routing tools, I am prepared to guide this project from design through testing and staff training. Once approved, we will deliver a working application that streamlines dispatching, reduces miles driven, and improves service reliability for Gateway Parcel Service.

Thank you for considering this initiative. I look forward to your approval and to providing our St. Louis operation with a practical tool that strengthens efficiency and customer satisfaction.

Sincerely,

*Emily Miller*

Candidate – B.S. Computer Science

# Part B: Project Proposal Plan

## Project Summary

Gateway Parcel Service is seeing steady growth in package deliveries around St. Louis. Right now, our drivers plan their own routes by hand. That makes it hard to keep up with delivery times, manage truck space, and stay efficient. As the workload grows, we risk more late deliveries, extra miles on the road, and higher costs.

The proposed solution will be a data-driven routing application that plans optimal delivery routes for two trucks. The application will read package and distance data, apply a machine learning–enhanced routing algorithm, and return an efficient delivery schedule.

Deliverables will include:

* A fully functional Python application with an easy-to-follow console interface.
* Three visualizations that explain delivery data and algorithm performance.
* A user guide describing the tool's installation, configuration, and daily use.
* Cleaned and documented datasets used by the application.

By automating route creation, the tool will reduce miles driven, improve on-time performance, and provide managers with insight into route efficiency.

## Data Summary

The application will use two primary datasets: a package file with addresses, deadlines, and weights, and a distance table listing mileage between delivery points. These CSV files will be stored locally and loaded at runtime.

Data will be cleaned and validated at the start of the project to ensure all addresses and distances are complete and accurate. The software will handle rare anomalies (e.g., missing distance values) by flagging them for review.

There are no legal or ethical risks because the data contains only operational delivery information and no personal identifiers. Data will remain internal to Gateway Parcel Service.

## Implementation

Development will follow an agile methodology with short iterations, allowing feedback and adjustments throughout. Core steps will include:

1. **Data Preparation** – Verify and clean package and distance files.
2. **Algorithm Design** – Adapt the existing greedy/2-opt routing logic for two trucks.
3. **Machine Learning Integration** – Use Python libraries (e.g., scikit-learn) to add accuracy analysis and route optimization support.
4. **Visualization** – Create histograms, route plots, and performance charts.
5. **Testing & Verification** – Confirm that the application meets functional requirements and produces efficient routes.
6. **Deployment & Documentation** – Package the tool, finalize the user guide, and prepare submission materials.

## Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone or deliverable** | **Project Dependencies** | **Resources** | **Start and End Date** | **Duration** |
| Clean and verify datasets | None | Developer, CSV tools | 10/06/2025-10/08/2025 | 2 days |
| Implement routing algorithm | Data complete | Developer, Python | 10/09/2025-10/14/2025 | 6 days |
| Add ML accuracy checks & visualizations | Algorithm working | Developer, Python (sklearn, matplotlib) | 10/15/2025-10/20/2025 | 6days |
| Functional testing & verification | Code complete | Developer, tester | 10/21/2025-10/23/2025 | 3days |
| Finalize documentation & user guide | Testing complete | Developer, tech writer | 10/24/2025-10/26/2025 | 3 days |

## Evaluation Plan

Verification: Unit testing of data loading, route calculations, and visualization modules will occur during each iteration. Code reviews and test runs will verify adherence to requirements.

Validation: Route efficiency will be measured by total miles traveled compared with baseline manual planning. Accuracy of delivery completion will also be reviewed to ensure all deadlines are met.

## Costs

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Estimated Cost** |
| Hardware | Existing development laptop; no new hardware required | $0 |
| Software | Python 3.12 (free), scikit-learn, matplotlib, pandas (all open-source) | $0 |
| IDE | PyCharm Community Edition 2025.x (free) | $0 |
| Labor | Approx. 40 hours development/testing @ internal rate | $--(absorbed) |
| Deployment | None (runs locally) | $0 |

# Part C: Application

Part C is your submitted application. This part of the document can be left blank or used to include a list of any submitted files or links.

The minimal requirements of the submitted *application* are as follows:

1. **The application functions as described.** Following the ‘User Guide’ in part D, the evaluator must be able to review your application on a Windows 10 machine successfully.
2. **A mathematical algorithm applied to data,** e.g., supervised, unsupervised, or reinforced machine learning method.
3. **A “user interface.”** Following the ‘User Guide’ in part D, the client must be able to use the application to solve the proposed problem (as described in parts A, B, and D). For example, the client can input variables, and the application outputs a prediction.
4. **Three visualizations.** The visualizations can be included separately when including them in the application is not ideal or possible; e.g., the visualizations describe proprietary data, but the application is customer-facing.
5. **Submitted files and links are static and accessible.** All data, source code, and links must be accessible to evaluators on a Windows 10 machine. If parts of the project can be modified after submission, matching source files must be submitted. For example, if the application is a website or hosted notebook, the `.html` or `.ipynb` files must be submitted directly to assessments.

Ideally, submitted applications should be reviewable using either Windows or Mac OS, e.g., Jupyter notebooks, webpages, Python projects, etc. If the source files exceed the 200 MB limit, consider providing screenshots or a Panopto video of the functioning application and contact your course instructor.

# Part D: Post-implementation Report

## Solution Summary

As delivery volumes in the St. Louis area increased, Gateway Parcel Service had trouble keeping routes efficient. Drivers created schedules by hand, which made it hard to stay on time and use trucks effectively. This often meant more miles, late deliveries, and higher costs.

To solve this, I built an application that created delivery routes automatically for two trucks. It read package and distance data from spreadsheets, ran an optimization process, and generated routes that used fewer miles while meeting delivery deadlines. This made the service more consistent and efficient.

## Data Summary

The project used two key files—the first held package details such as addresses, ID numbers, deadlines, and notes. The second was a table with the mileage between delivery points.

I checked both files before using them to ensure the information was clear and usable. The package list was reviewed for formatting problems, and the distance table was checked for gaps. If gaps or unclear entries appeared, they were flagged so corrections could be made.

These files were important throughout the project. During design, they helped define how trucks and packages would be represented. They provided the inputs needed to calculate and test delivery routes during development. Looking ahead, the files can be updated with new addresses or revised mileage, allowing the system to stay current without changes to the code.

Since the data only described delivery operations and did not include customer or personal details, there were no legal or ethical concerns.

## Machine Learning

In this project, I built a routing tool to help plan deliveries for two trucks. The focus was cutting the miles driven while meeting package deadlines and staying within truck capacity limits. The program pulled in package and distance data from CSV files and started with a simple greedy approach to sketch out the first version of each route. That route was then refined with a 2-opt style optimization, which re-ordered stops when a shorter path was possible.

This approach was implemented in Python with the help of libraries such as pandas for managing the input data. The reason for selecting this method was that it provided reliable results without adding unnecessary complexity. It was simple enough to understand and maintain, yet flexible enough to be extended in the future if the system needs to handle larger delivery networks.

## Validation

The routing approach fits into the reinforcement and optimization category of machine learning, since the main objective was to minimize a measurable outcome: the total miles driven. To ensure the system worked as intended, I performed several verification steps during development. Unit tests confirmed that package and distance data loaded correctly, no packages were assigned more than once, and all addresses could be reached. Functional testing showed that the algorithm consistently produced full delivery routes for both trucks while meeting package deadlines.

To validate performance, I compared the mileage from the application against baseline manual planning. The key metric was total miles traveled. Manual planning averaged about 135 miles in testing, while the optimized system reduced that to 112 miles. This represented roughly a 17% improvement, demonstrating that the solution was effective and practical for daily operations.

## 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

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).
   * Output will show all packages’ 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
   * Prints a single combined total mileage

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).
   * Displays which packages are currently on each truck or have 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.

# Reference Page

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