DATA COLLECTION AND GEOSPATIAL MAPPING

1. Use **OSMNx** to extract freight-relevant road networks from OpenStreetMap (ports, bonded terminals, warehouses, fuel depots, service points).
2. Apply **GeoPandas** to merge, clean, and reproject spatial layers.
3. Query **Google Maps API** for real-time and historical travel times.
4. Use **BeautifulSoup** to scrape supplementary logistics/network data from online sources.
5. Implement automated **fuel price scraping** to retrieve up-to-date cost data.

DATA PROCESSING AND ANALYSIS

1. Use **Python** as the main scripting environment for integration and workflow control.
2. Apply **Pandas** to structure datasets, handle joins, and manage missing values.
3. Use **NumPy** for efficient numerical operations.
4. Apply **Shapely** for geometric computations (distances, buffers, intersections).
5. Use **Fiona** to read and write spatial files (Shapefile, GeoJSON).

GRAPH MODELLING AND ROUTE OPTIMIZATION

1. Build the road network as a directed, weighted graph using **NetworkX**.
2. Integrate **OSMNx** data into the graph model.
3. Apply **SciPy** for optimization routines and advanced computations.

SIMULATION

1. Use **SimPy** to simulate discrete events (traffic delays, refueling, demand changes).
2. Develop **custom Python simulation scripts** to test route performance under various scenarios.

VISUALIZATION AND REPORTING

1. Create interactive dashboards in **Tableau** for efficiency scores, costs, and congestion advisories.

2. Generate visual analytics using **Matplotlib**, **Seaborn**, and **Plotly**.

3. Use **Kepler.gl** for geospatial visualization of routes and network topology.

SPECIAL FOCUS AREAS

- Ensure tight integration of **Tableau**, **SciPy**, **OSMNx**, **NetworkX**, and **Kepler.gl** for robust modeling, analysis, and presentation.