Based on the analysis of your uploaded files and the constraints of your university project (2 weeks, 3 people, Digital Twin goal), here is a recommended solution.

Executive Summary: The Solution Design

Recommended Approach: Data-Driven Metamodel-Based Optimization using Deep Neural Networks (DNN) and SPSA.

Why this approach? Running a 24-hour microscopic simulation (SUMO) thousands of times to calibrate parameters is computationally impossible within 2 weeks. You need a "shortcut" (surrogate model) that mimics SUMO but runs instantly.

- **Inspiration:** This design is directly grounded in Min et al. (2024), which explicitly proposes replacing the microscopic simulation with a Deep Neural Network (DNN) metamodel to enable real-time/fast estimation.
- **Feasibility:** It divides the work perfectly for 3 people (Data Generation, Metamodel Training, Optimization Loop) and uses standard tools (Python, PyTorch/TensorFlow, SUMO).

Detailed Methodology

You should break your project into three distinct phases.

Phase 1: Dataset Generation (The "Offline" Phase)

- **Goal:** Create a dataset to teach a Neural Network how your SUMO simulation behaves.
- **Action:**

1. Take your **24 initial OD matrices**.
2. Write a script to randomly perturb these matrices (e.g., variation on demand levels) to create synthetic scenarios (e.g., to).
3. Run the SUMO simulation for each scenario.
4. **Log Inputs & Outputs:**

- **Input ():** The perturbed OD matrix (flattened into a vector).
- **Output ():** The resulting detector measurements (Total Volume, Speed, Occupancy, etc.) aggregated at the same 5-minute intervals.
- **Paper Reference:** Min et al. (2024) emphasize using synthetic data generated by the simulation to train the metamodel, ensuring it captures the specific "physics" of your SUMO setup.

Phase 2: Metamodeling (The "Digital Twin" Proxy)

- **Goal:** Train a mathematical model that predicts Detector Data () from OD Demand () instantly.
- **Model Architecture:** A simple **Deep Neural Network (DNN)** / Multi-Layer Perceptron (MLP).
- **Input Layer:** Size = Number of OD pairs.
- **Hidden Layers:** 3 to 4 fully connected layers (e.g., sizes 256, 128, 64) with ReLU activation.
- **Output Layer:** Size = Number of Detectors Metrics (Volume, Speed, etc.).
- **Training:** Train this network on your dataset from Phase 1.
- **Paper Reference:** Min et al. (2024) demonstrate that a DNN can effectively map the correlation between the OD matrix and detector data, simplifying the computational process significantly compared to vehicle-level interactions.

Phase 3: Optimization (The Calibration)

- **Goal:** Find the OD matrix that minimizes the error between *Simulated* data and *Real-World* data.
- **Algorithm: SPSA (Simultaneous Perturbation Stochastic Approximation)**.
- **Process:**

1. Define a Loss Function (e.g., RMSE between Predicted Detector Data and Real-World Data).
2. Instead of running SUMO inside the optimization loop, pass the current OD estimate into your **DNN Metamodel**.
3. The DNN gives you the predicted detector values in milliseconds.
4. SPSA calculates the gradient and updates the OD matrix.
5. Repeat for 100-200 iterations until the error converges.

- **Validation:** Once SPSA gives you the "optimized" OD matrix, run it **once** in the actual SUMO simulation to verify the results.
- **Paper Reference:** Min et al. (2024) combine their data-driven metamodel with SPSA, reducing the time required for calibration while maintaining accuracy. Ho et al. (2023) also highlight SPSA as a standard, effective choice for high-dimensional OD calibration.

Feasibility Plan (2 Weeks / 3 People)

- **Student A (Simulation Specialist):**
- Day 1-3: Setup SUMO. Create the Python script (using TraCI or libsumo) to loop through the 24 hours, update ODs, and extract the 6 required metrics (Volume, Speed, Occupancy, etc.).
- Day 4-6: Run the "Sampling" generation to create the training data.
- **Student B (Data Scientist):**
- Day 1-4: Design the Neural Network architecture (Input/Output dimensions).
- Day 5-7: Train the model using the data generated by Student A. Validate that the NN predicts detector outputs accurately.

- **Student C (Optimizer):**
- Day 1-5: Implement the SPSA algorithm in Python. (It's a simple algorithm, <100 lines of code).
- Day 6-9: Connect SPSA to the trained Neural Network (from Student B) to run the calibration.
- **Group (Days 10-14):**
- Run the final "Optimized OD" in SUMO.
- Compare the output against the Real-World Data.
- Write the report citing Min et al. (2024) and Ho et al. (2023).

Academic "Bonus" (If you have time)

If you finish early, you can reference Ho et al. (2023) and try to implement **MSPSA**. This involves using the gradient of your Neural Network (which is differentiable!) to help guide the SPSA, potentially converging faster. This would add significant "academic weight" to your project, showing you explored advanced gradient-based guidance.