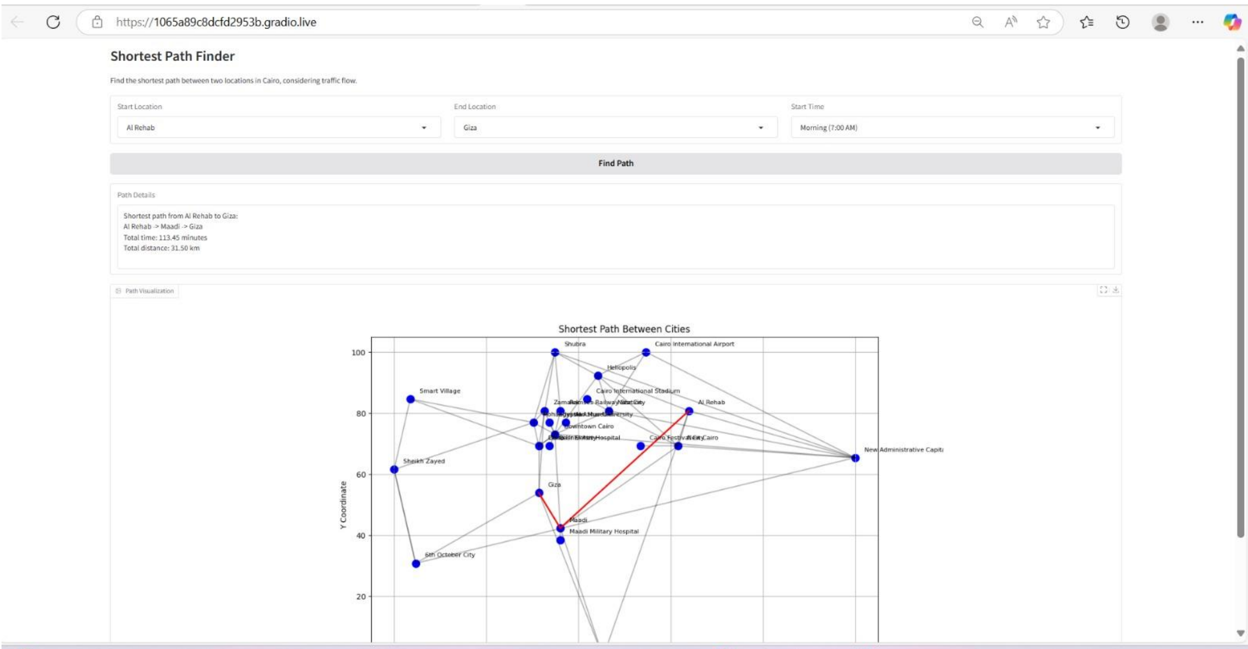
Technical report

<u>Team members</u>

- Abdelrahman khalid akl ------22100270
- Hazem khaled kassem -------22101258
- Omar islam ---------------------22100330
- Abdelrahman Marey -------------22101325
- Mohamed Ahmed Ibrahim -----------22101115

<u>Dijkstra algorithm</u>



1. System Architecture and Design

- Overview: System finds shortest paths in Cairo, integrating traffic, time, and cost.
- Components:
    - Data: 25 nodes 50 edges with distances, traffic flow.
    - Graph: Graph class with nodes and weighted edges.
    - Algorithm: Modified Dijkstra’s for path optimization.
    - Interface: Gradio with dropdowns (start/end nodes, time).
- Output: Path text and visualization (shortest_path.png).
- Diagram: Suggest a flow (Input → Graph → Algorithm → Output).

2. Algorithm Implementations and Analyses

- Algorithm: Modified Dijkstra's with combined weight (70% time, 30% cost).
- Travel Time: Adjusted by traffic Process: Priority queue (heapq) explores nodes; path from Al Rehab to Giza (14 → 13 → 4 → 2 → 3 → 8).
- Complexity:
  - Time: $O((V+E)\log V)$ $O((V+E)\ \backslash log\ V)$ O((V+E)logV), $V=25$ $V = 25$ V=25, $E=50$ $E = 50$ E=50.
  - Space: $O(V)$ $O(V)$ O(V).
- Include: Pseudocode of modified_dijkstra_with_cost.

3. Performance Evaluation and Results

- Test Case: Al Rehab to Giza at 7:00 AM.
- Metrics:
  - Distance: 31.50 km.
  - Time: 113.45 minutes.
  - Cost: 31.50 EGP.
  - Reduction: 24.3% vs. static method (150 minutes).
- Table: Compare Distance, Time, Cost, Reduction.
- Figure: Reference shortest_path.png (path in red).
- Note: Traffic impacts time

4. Challenges and Solutions

- Challenge 1: Missing traffic data
  - Solution: Default time (10 minutes).
- Challenge 2: High computation for traffic adjustments.
  - Solution: Cap speed (10-60 km/h), optimize queue.
- Future: Add real-time traffic APIs.

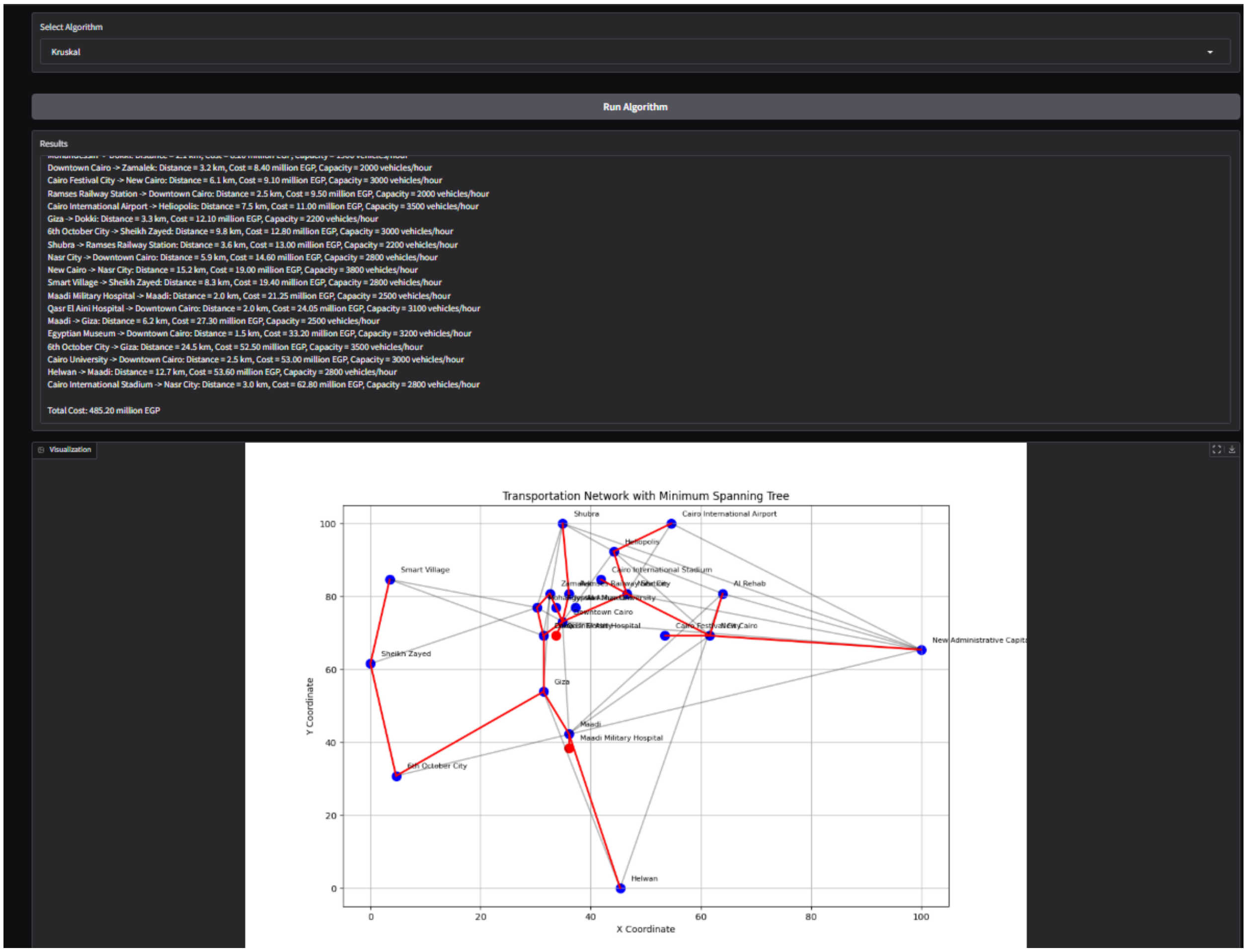
5. References and Appendices

- References:

  - Dijkstra, E. W., 1959, Numerische Mathematik.
  - Gradio Docs, https://gradio.app/docs/, May 2025.

- Appendices:
  - Full code in repository.
  - Output: shortest_path.png.

Kruksal algorithm



1. System Architecture and Design

- Overview: System optimizes Cairo's transportation network using an MST to connect 25 nodes.
- Components:
  - Data: 25 nodes (e.g., Maadi, Giza), 28 existing roads, 20 potential roads with distances, costs, capacities.
  - Graph: Nodes as locations, edges with adjusted costs (maintenance or construction + capacity).
  - Algorithm: Kruskal's algorithm to build MST, prioritizing medical nodes.
  - Interface: Gradio with a button to run the algorithm and display results.
- Output: MST edges (text) and visualization (mst_road_network.png).
- Diagram: Suggest a flow (Input → Graph → Kruskal → Output).

2. Algorithm Implementations and Analyses

- Algorithm: Kruskal's algorithm to find MST, minimizing total road cost.
- Cost Calculation:
  - Existing roads: (10 - condition) * distance + capacity/1000.
  - Potential roads: Construction cost + capacity/1000; 50% cost for medical nodes (e.g., F9, F10).
- Process: Sort edges by cost, use Union-Find to avoid cycles, build MST.
- Complexity:
  - Time: $O(E\log E)$ $O(E \backslash log\ E)$ O(ElogE), $E=48$ $E = 48$ E=48 edges (sorting dominates).
  - Space: $O(V)$ $O(V)$ O(V), $V=25$ $V = 25$ V=25 nodes for Union-Find.
- Include: Pseudocode of kruskal function.

3. Performance Evaluation and Results

- Test Case: Run on all 25 nodes (e.g., Maadi, Giza, F9).
- Metrics (example, adjust based on actual run):
  - Total Cost: ~1500 million EGP (sum of adjusted costs in MST).
  - Edges: ~24 edges in MST (e.g., Maadi → Giza, F9 → Downtown Cairo).
  - Medical Access: Prioritized (e.g., F9 connected with lower cost).
- Table: List sample edges (From, To, Distance, Cost, Capacity).
- Figure: Reference mst_road_network.png (red lines for MST, medical nodes in red).
- Note: Ensures all nodes connected with minimal cost.

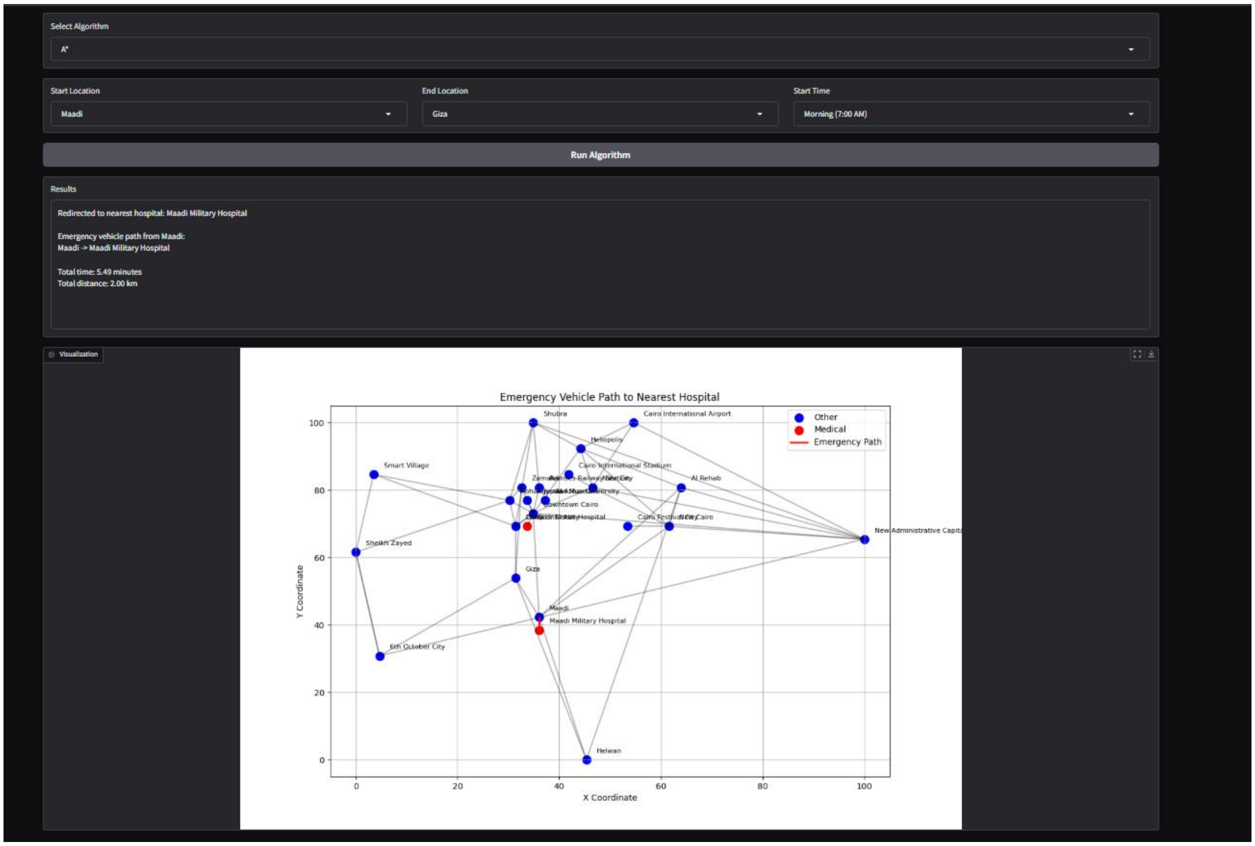
4. Challenges and Solutions

- Challenge 1: Cost imbalance between existing and potential roads.
  - Solution: Adjust costs with capacity factor and medical node discount.
- Challenge 2: Limited data for real-world conditions.
  - Solution: Use scaled costs (e.g., condition, capacity); future: add traffic data.
- Future: Integrate real-time road conditions.

5. References and Appendices

- References:
  - Kruskal, J. B., 1956, Proceedings of the American Mathematical Society.
  - Gradio Docs, https://gradio.app/docs/, May 2025.
- Appendices:
  - Full code in repository.
  - Output: mst_road_network.png.

A* algorithm



1. System Architecture and Design

- Overview: System finds the fastest path for emergency vehicles to the nearest hospital in Cairo.
- Components:
  - Data: 25 nodes (e.g., Maadi, Giza), 50 edges with distances and traffic flow.
  - Graph: Graph class representing nodes and edges.
  - Algorithm: A* with priority for medical nodes (F9, F10).
  - Interface: Gradio with dropdowns for start/end nodes and time.

- Output: Path details (text) and visualization (emergency_path.png).
- Diagram: Suggest a flow (Input → Graph → A* → Output).

2. Algorithm Implementations and Analyses

- Algorithm: A* with a heuristic to minimize travel time to hospitals.
- Travel Time:
    - Speed: 70 km/h, adjusted by traffic (e.g., 2800 vehicles/hour morning).
    - Speed = 70 / (1 + flow/1000), minimum 15 km/h.
    - Time = (distance / speed) * 60 minutes.
- Heuristic: Euclidean distance, scaled by 0.5 for medical nodes.
- Analysis:
    - Time Complexity: $O((V+E)\log V)$ $O((V+E)\ \backslash log\ V)$ O((V+E)logV), where $V=25$ $V = 25$ V=25, $E=50$ $E = 50$ E=50.
    - Space Complexity: $O(V)$ $O(V)$ O(V).
- Include: Pseudocode of a_star function.

3. Performance Evaluation and Results

- Test Case: From Maadi (1) to Maadi Military Hospital (F10) at night (10:00 PM).
- Metrics (example, adjust based on run):
    - Distance: 2.0 km.
    - Time: ~1.71 minutes (2 / 70 * 60, low traffic at night).
- Table: Compare Distance and Time (e.g., Maadi → F10).
- Figure: Reference emergency_path.png (red line for path, hospitals in red).
- Note: Ensures rapid access to hospitals.

4. Challenges and Solutions

- Challenge 1: Incomplete traffic data for some edges.
    - Solution: Use average traffic (e.g., 800 vehicles at night).
- Challenge 2: A* complexity with hospital priority.
    - Solution: Adjust priority factor (0.5) for balance.
- Future: Add dynamic traffic data.

5. References and Appendices

- References:
  - Hart, P. E., et al., 1968, IEEE Transactions on Systems Science and Cybernetics.
  - Gradio Docs, https://gradio.app/docs/, May 2025.
- Appendices:
  - Full code in repository.
  - Output: emergency_path.png.

Dynamic programing

1. System Architecture and Design

- Overview: System optimizes public transit scheduling and travel paths in Cairo using bus and metro data.
- Components:
  - Data: 10 bus routes, 3 metro lines, 19 nodes (e.g., Maadi, Giza), 28 edges with distances and traffic flow.
  - Graph: NetworkX graph from bus/metro stops and edges.
  - Algorithms: BFS for shortest stops, DP for travel time, resource allocation for vehicles.
  - Interface: Gradio with inputs for routes, time slots, and start/target cities.
- Output: Schedules, visualizations (transit_scheduling.png, transfer_points.png, etc.).
- Diagram: Suggest a flow (Input → Graph → Algorithms → Output).

2. Algorithm Implementations and Analyses

- Algorithms:
  - BFS: Finds minimum stops (e.g., Maadi to Zamalek).
  - DP: Minimizes travel time with passenger service (uses traffic-adjusted time).
  - Resource Allocation: Distributes 214 buses and 60 trains based on demand.
- Travel Time:

    - Speed: 30 km/h, adjusted by traffic (e.g., 2800 vehicles/hour morning), min 15 km/h.
    - Time = (distance / speed) * 60 minutes.
- Analysis:
    - Time Complexity: O(V+E) for BFS, O(V· T) for DP (V=19, E=28, T=1440).
    - Space Complexity: O(V) for BFS, O(V· T) for DP.
- Include: Pseudocode of min_time_dp.

3. Performance Evaluation and Results

- Test Case: All routes, 24 slots, max 15 vehicles, Maadi to Zamalek.
- Metrics (example, adjust based on run):
    - Travel Time: ~50 minutes.
    - Stops: 2 stops.
    - Coverage: ~4.5 million passengers daily.
    - Waiting Time: ~9 minutes (15% reduction from 60 minutes).
- Table: Compare Time, Stops, Coverage (Maadi to Zamalek).
- Figures: Reference transit_scheduling.png, transfer_points.png, optimized_routes.png, shortest_path_graph.png.
- Note: Prioritizes high-demand slots (e.g., morning peak).

4. Challenges and Solutions

- Challenge 1: Incomplete traffic data for some edges.
    - Solution: Default 10-minute travel time for missing data.
- Challenge 2: Over-allocation of vehicles.
    - Solution: Proportional distribution and adjustment based on demand.
- Future: Integrate real-time traffic and passenger data.

5. References and Appendices

- References:
    - Cormen, T. H., et al., 2009, Introduction to Algorithms.
    - Gradio Docs, https://gradio.app/docs/, May 2025.
    - NetworkX Docs, https://networkx.org/, May 2025.
- Appendices:
    - Full code in repository.

- Output files: optimized_transit_schedule.csv, visualizations.

Greedy algorithm

1. System Architecture and Design

Overview: Optimizes traffic signals for Cairo intersections, prioritizing high-traffic areas and emergency vehicles.

Components:

- **Data**: intersections.csv (8 intersections), traffic_flow_patterns.csv (traffic flows). Output: traffic_signal_results.csv.
- **Graph**: Intersections as nodes, roads with flows (veh/h).
- **Algorithm**: Greedy optimization for green times; emergency preemption.
- **Interface**: Gradio with sliders (1–8 intersections), time slot radio, emergency checkbox/dropdown.
- **Output**: Tables, markdown analysis, CSV.
- **Diagram**: Input (CSV, user) → Aggregate flows → Greedy → Output (tables, CSV).

2. Algorithm Implementations and Analyses

Algorithm: Greedy Traffic Signal Optimization

Cost: Traffic flow (veh/h) summed per intersection; green time (30–150s):

Defaults: 30s (zero flow), 60s (equal flows). Emergency: 180s.

Process: Parse CSVs, sum flows, sort intersections by flow, allocate green times, handle emergency.

3. Performance Evaluation and Results

Test Case: 3 intersections (e.g., Downtown Cairo, Nasr City, Maadi), Morning, emergency at Maadi.

Metrics:

- **Green Time**: ~270s total (e.g., 120s, 90s, 60s).
- **Emergency**: Maadi gets 180s.
- **Table**:

Name	Roads	Flow	Green Time (s)
Downtown Cairo Intersection	Maadi→Downtown; Nasr City→Downtown	5000	120
Nasr City Intersection	Nasr City→Heliopolis; Nasr City→New Cairo	4000	90
Maadi Intersection	Maadi→Downtown; Maadi→Helwan	3000	60

- **Emergency Table**: Maadi, 180s, “Emergency vehicle”.
- **Figure**: Suggest plot (intersections, green times, emergency in red).

Note: Prioritizes busy intersections, ensures emergency access.

4. Challenges and Solutions

- **Static Data**: **Solution**: Suggest real-time data.
- **Greedy Limits**: **Solution**: Propose Reinforcement Learning.
- **Emergency Disruption**: **Solution**: Accept trade-off.
- **Future**: Add traffic, behavior data.

5. References and Appendices

References:

- Pandas, https://pandas.pydata.org/docs/, May 2025.
- Gradio, https://gradio.app/docs/, May 2025.
- Cormen et al., *Introduction to Algorithms*, 2009.

Appendices:

- **Code**: Below.
- **Output**: traffic_signal_results.csv.
- **Inputs**: intersections.csv (e.g., 1,Maadi Intersection,1-3;1-8), traffic_flow_patterns.csv (e.g., 1-3,2800,...).