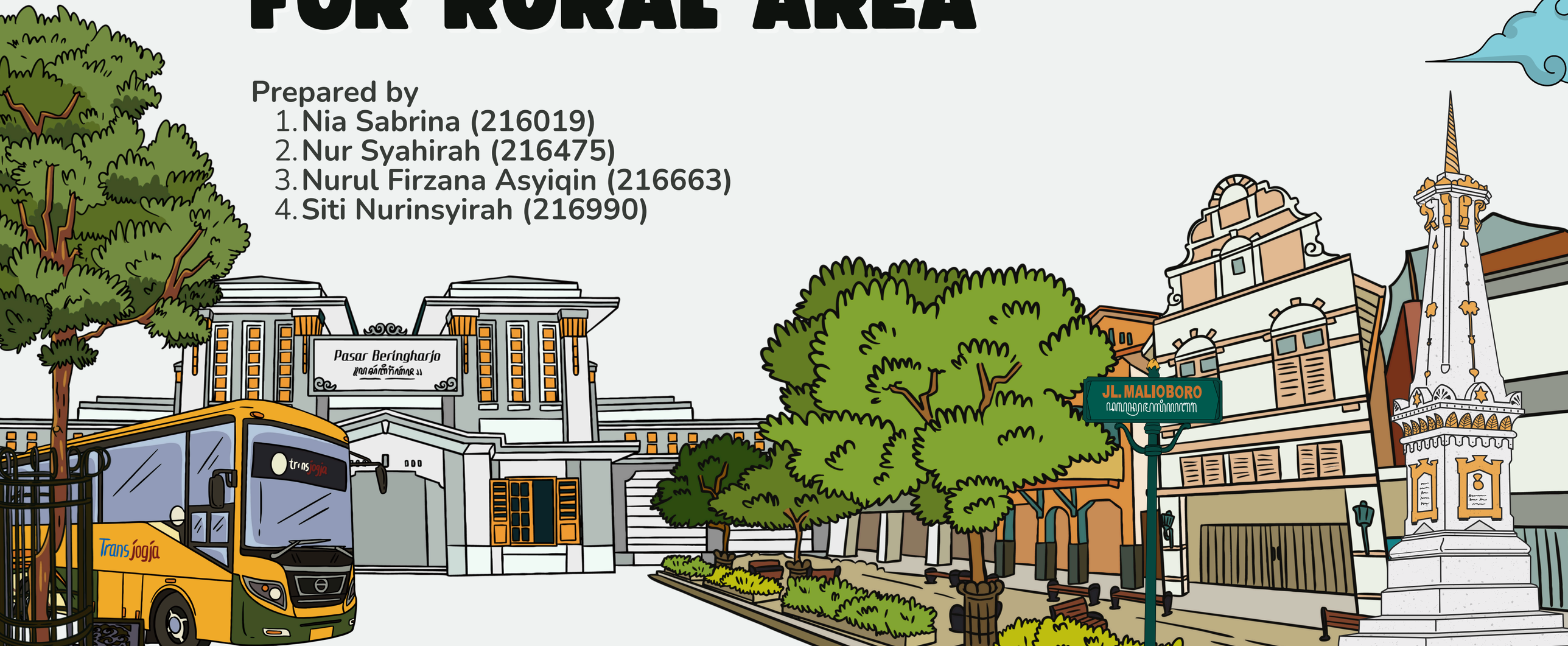
Group 5

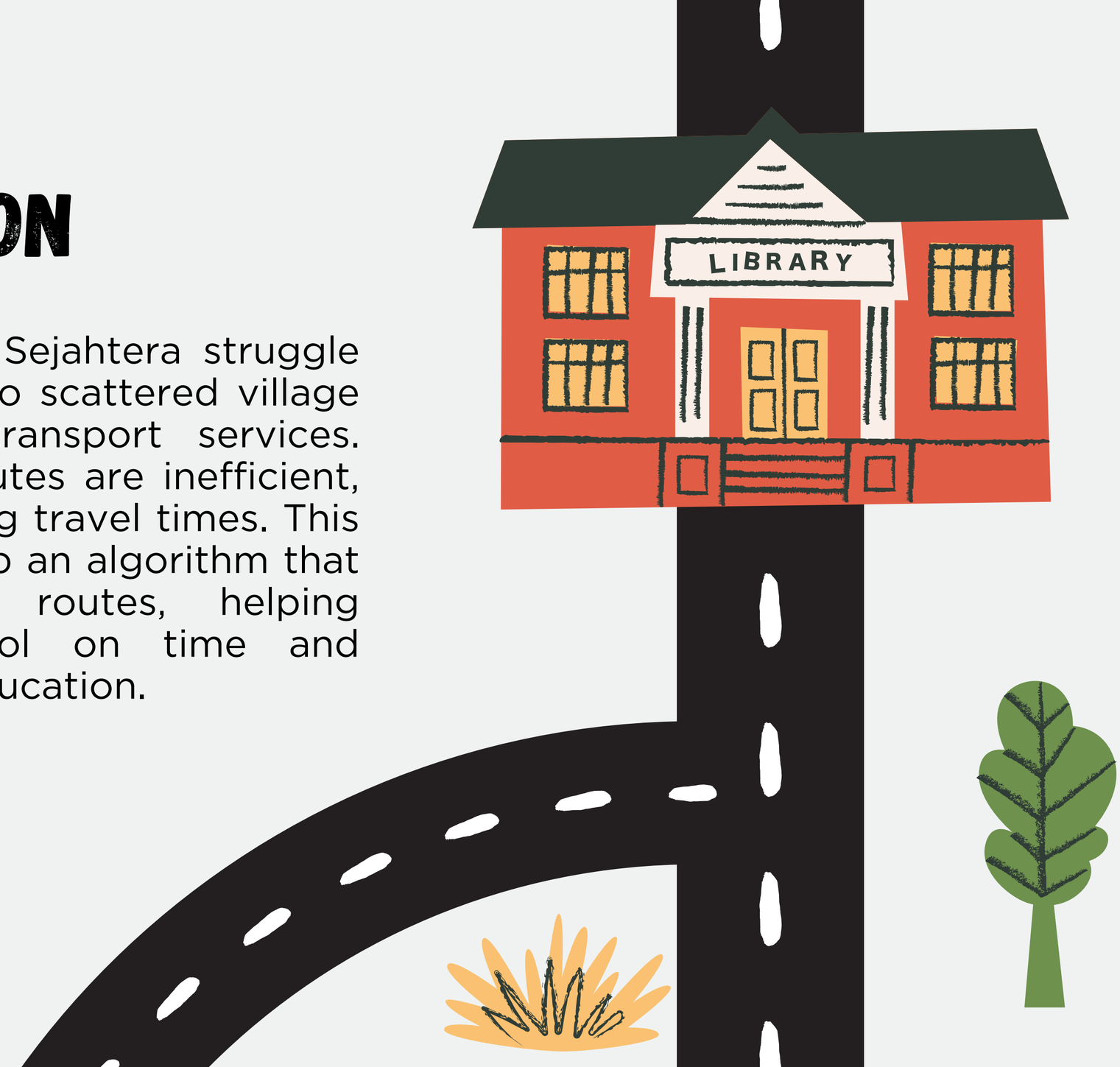
SCHOOL BUS ROUTING FOR RURAL AREA

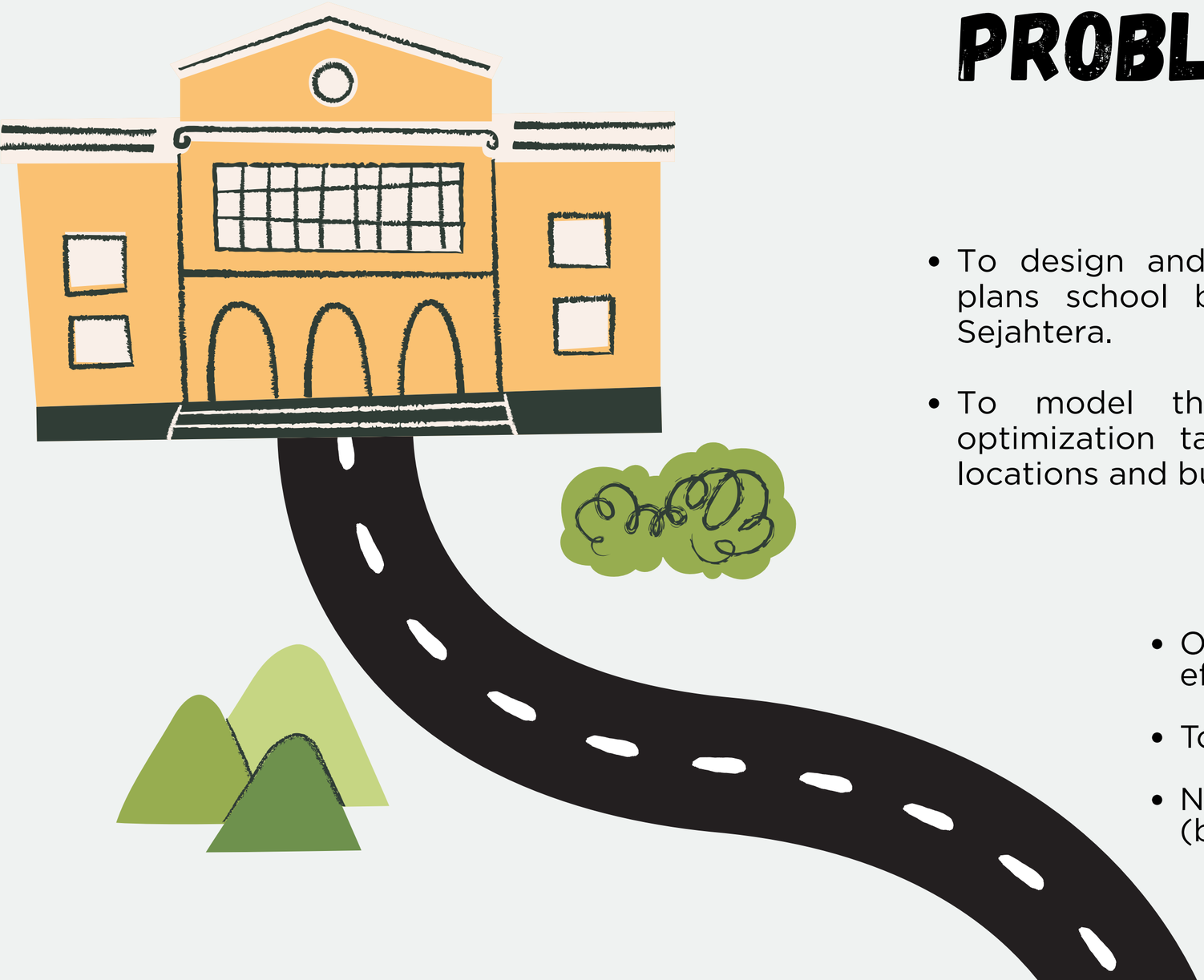
Prepared by
1. Nia Sabrina (216019)
2. Nur Syahirah (216475)
3. Nurul Firzana Asyiqin (216663)
4. Siti Nurinsyirah (216990)



INTRODUCTION

Students in Kampung Sejahtera struggle to get to school due to scattered village homes and limited transport services. Current school bus routes are inefficient, causing delays and long travel times. This project aims to develop an algorithm that creates better bus routes, helping students reach school on time and improving access to education.





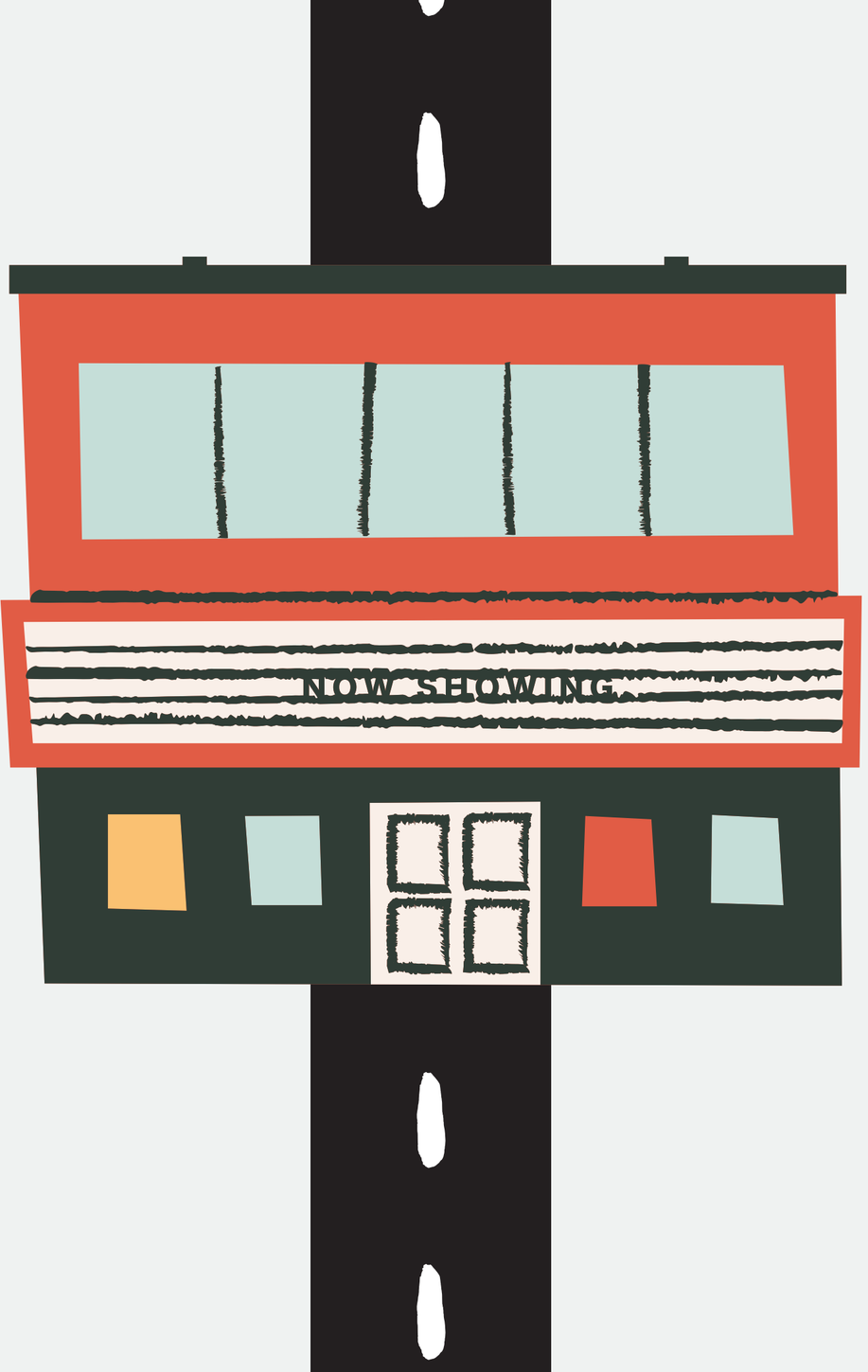
PROBLEM DEFINITION

GOALS :

- To design and implement an algorithm that efficiently plans school bus routes in rural areas like Kampung Sejahtera.
- To model the problem as a resource-constrained optimization task, considering factors such as student locations and bus capacity.

EXPECTED OUTPUT :

- Optimized bus routes cover the most students efficiently.
- Total travel distance per route.
- Number of students served within given constraints (bus capacity, time).



MODEL OF THE PROBLEM

Key Data Types

- Nodes: Villages & school as graph nodes.
- Edges: Distances (km) between nodes as weighted edges.
- Bus Object: (Key Properties)
  - capacity, the list of visited villages, and the current student load.
- Student Distribution:
  - Number of students per village.

Objective

- To minimize the total distance traveled by all buses

Constraints

- Every student must be picked up.
- No bus exceeds its capacity.
- Each village is visited only once.
- All buses start and end at the school.
- Use shortest paths between locations.

DESIGNING THE ALGORITHM

1. Solve problems step-by-step

- Algorithms help break down a complex problem (like bus routing) into smaller, manageable steps.

2. Find the most efficient solution

- Instead of guessing or trial-and-error, algorithms give faster, smarter results with less effort.

3. Handle constraints and rules

- Real-world limits (like bus capacity, time, or distance) can be included directly in the algorithm.

OVERVIEW (DIJKSTRA'S)

- Dijkstra's algorithm finds the shortest path from the school (node 0) to each village individually.

STEP-BY-STEP (DIJKSTRA'S)

- It builds a graph of nodes (villages) and edges (roads with distances).
- Uses priority queue to always explore the shortest current known path.

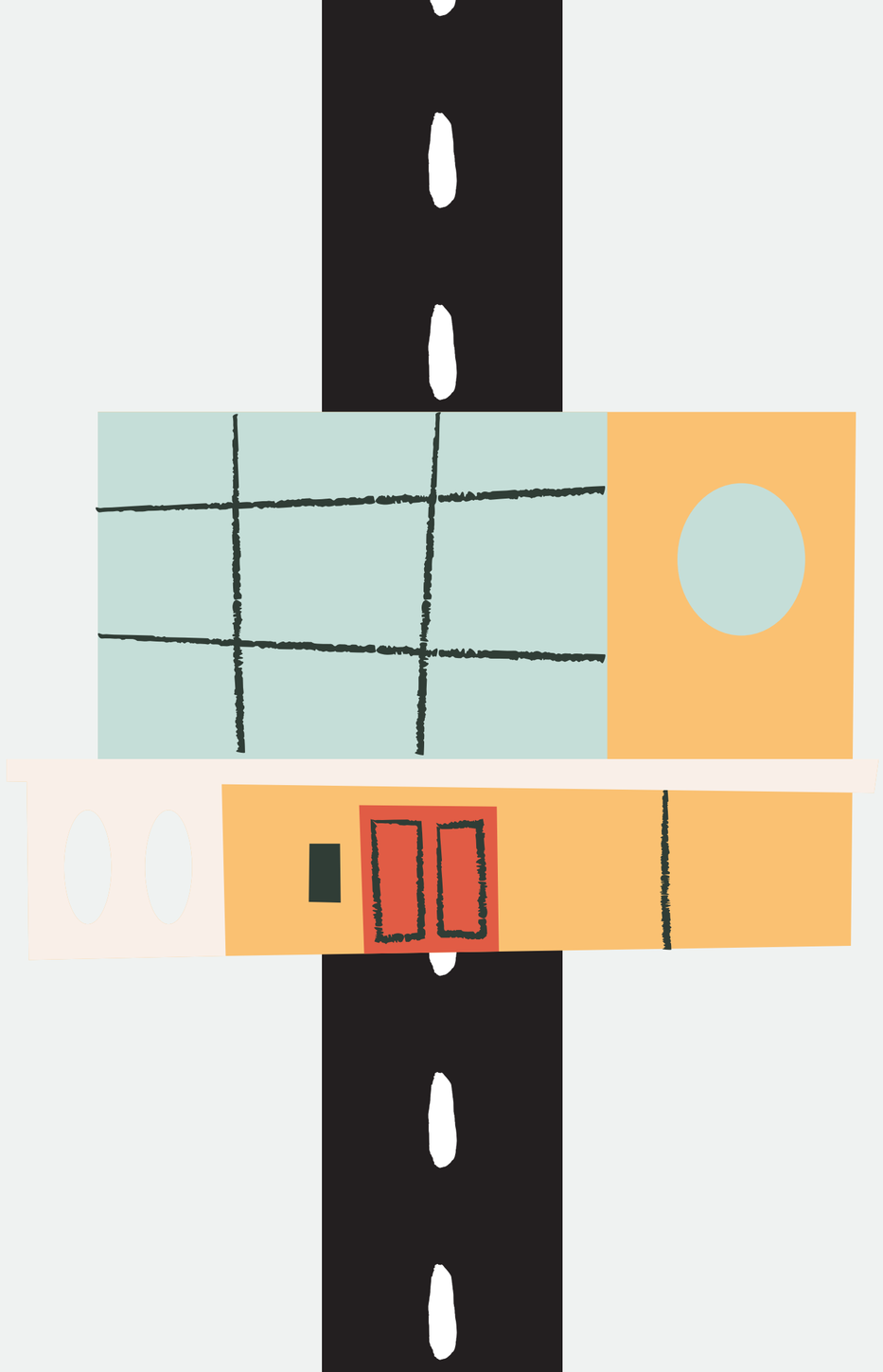
OVERVIEW (GREEDY'S)

Greedy routing builds a realistic bus route based on:

- Student capacity of the bus (9 students max).
- Shortest possible next village (nearest neighbor).
- Ability to return to school and take multiple trips.

STEP-BY-STEP (GREEDY'S)

- Start at the school (node 0).
- Look for the nearest unvisited village whose students can still fit in the bus.
- Add the village to the route.
- Repeat until the bus is full or no valid next village exists.
- Make sure all students from all villages are picked up.



PROGRAM TESTING

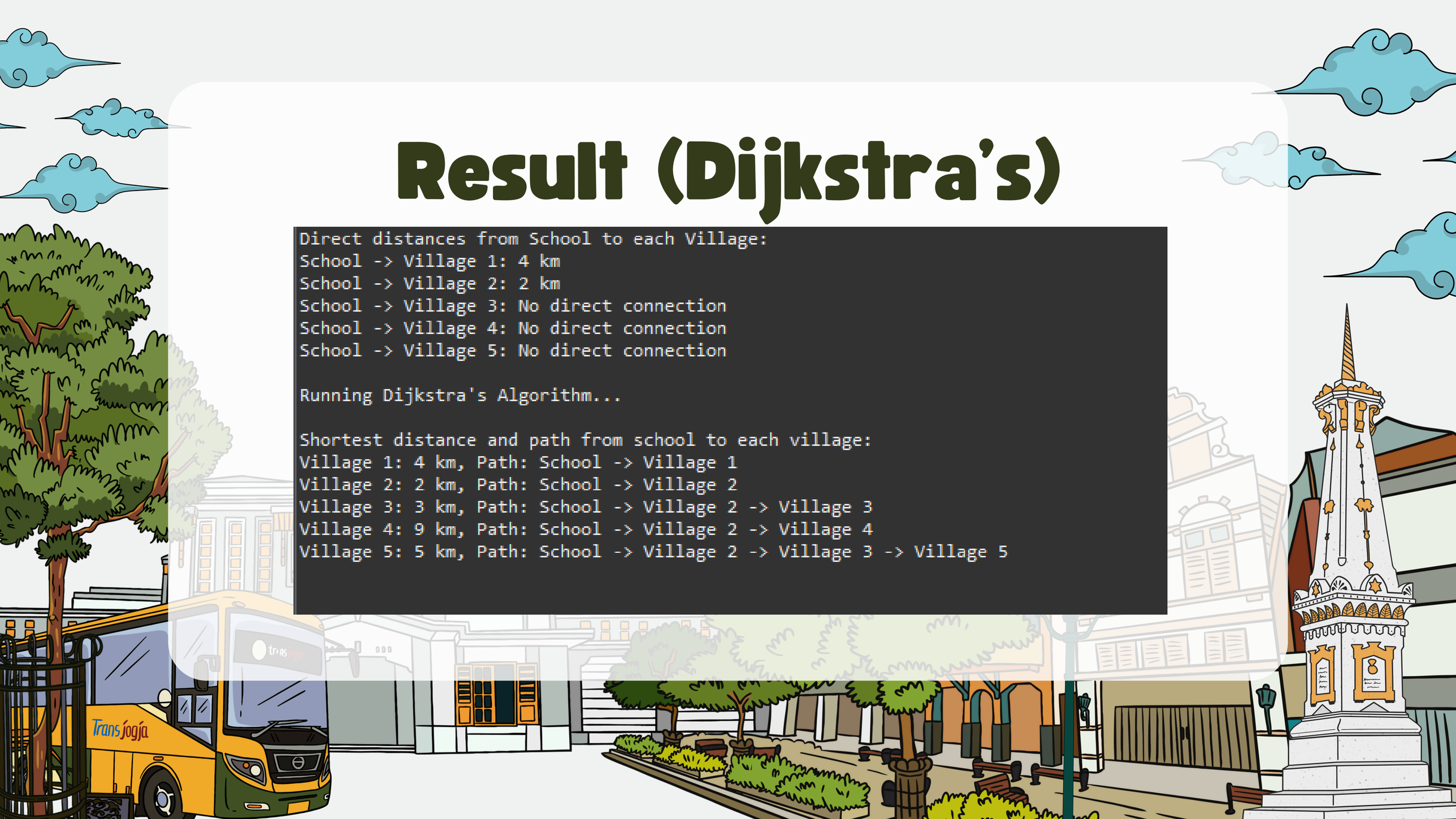
- Number of Nodes: 6 (1 school + 5 villages)
- Total Students: 20

Village	Students
Village 1	5
Village 2	4
Village 3	6
Village 4	2
Village 5	3

- Bus Capacity: 9 students

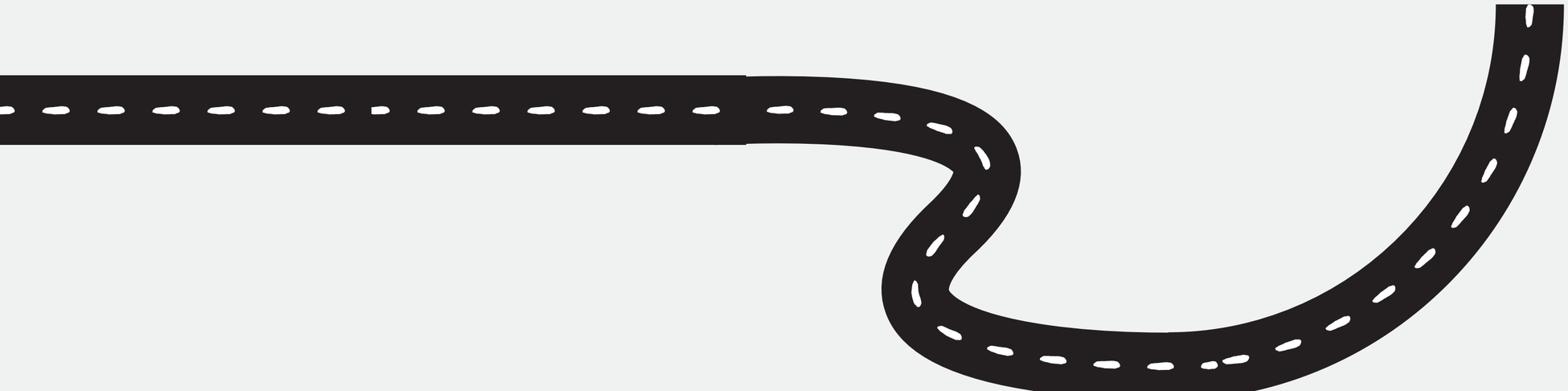
Result (Dijkstra's)

```
Direct distances from School to each Village:
School -> Village 1: 4 km
School -> Village 2: 2 km
School -> Village 3: No direct connection
School -> Village 4: No direct connection
School -> Village 5: No direct connection

Running Dijkstra's Algorithm...

Shortest distance and path from school to each village:
Village 1: 4 km, Path: School -> Village 1
Village 2: 2 km, Path: School -> Village 2
Village 3: 3 km, Path: School -> Village 2 -> Village 3
Village 4: 9 km, Path: School -> Village 2 -> Village 4
Village 5: 5 km, Path: School -> Village 2 -> Village 3 -> Village 5
```



ANALYSIS OF RESULT

- Dijkstra does not consider student counts or bus capacity.
- It only gives the shortest path to each village, not a complete route to pick up all students.
- It’s useful as a benchmark for distance efficiency, but not for full bus route planning.



Result (Greedy's)

```
Total number of students to be picked up: 20

Bus 1 route: School -> Village 2 -> Village 4 -> Village 5 -> School
  Students picked up: 9, Distance: 29 km

Bus 2 route: School -> Village 1 -> School
  Students picked up: 5, Distance: 8 km

Bus 3 route: School -> Village 3 -> School
  Students picked up: 6, Distance: 20 km

All students picked up in 3 trips.
Total distance traveled: 57 km.
```

ANALYSIS OF RESULT

Strengths:

- Models actual bus operations with capacity, multi-trip, and pickup priority.
- Ensures all students are picked up even if it takes multiple trips.

Limitation:

- Picks only the locally nearest village, which may result in longer total distance.

CORRECTNESS OF THE ALGORITHM

Why the Greedy Algorithm is Correct:

1. Greedy Decision Making
Always chooses the nearest unvisited village within bus capacity, reducing unnecessary travel without evaluating every possibility.

2. Constraint-Adherence
Before adding a village to the route, the algorithm checks if the current load + students fits the bus capacity, ensuring no overload.

3. Single Visit Guarantee
A visited[] array ensures each village is picked up once only — avoids duplicate pickups and improves efficiency.

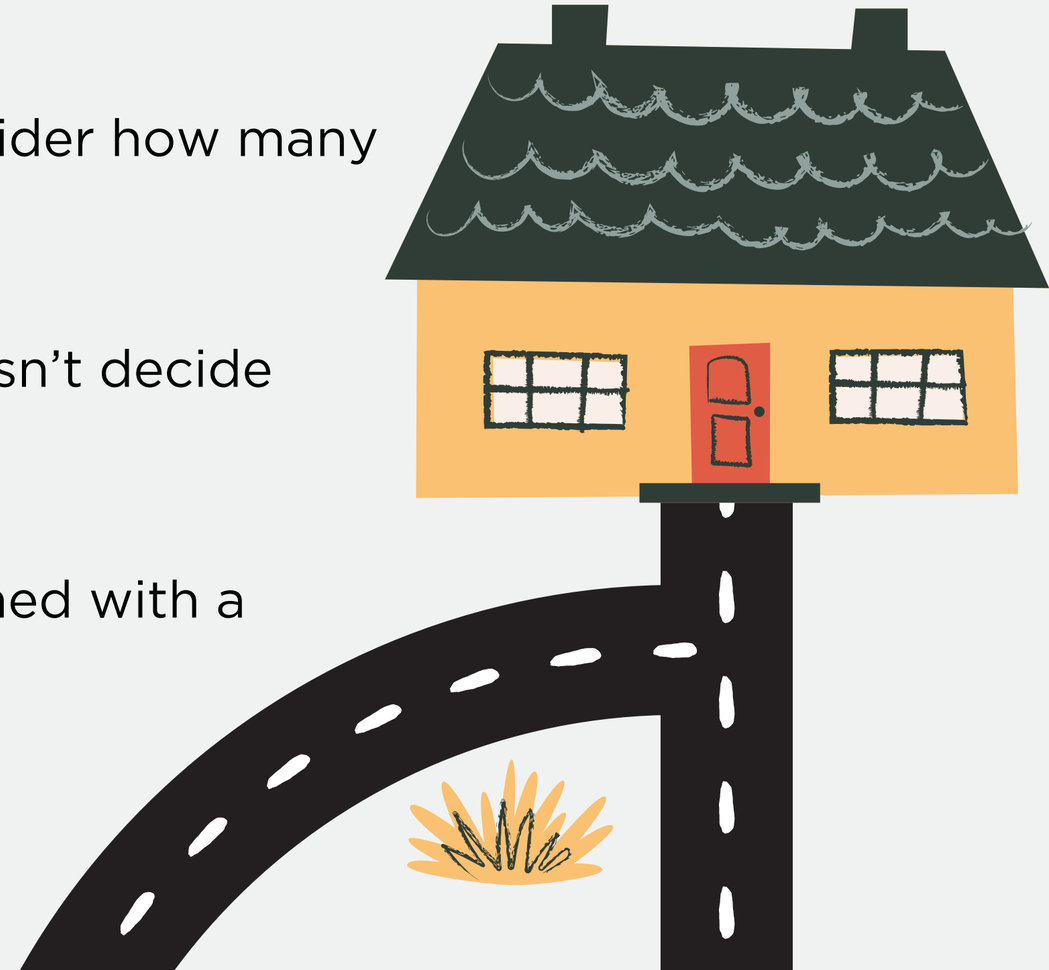
CORRECTNESS OF THE ALGORITHM

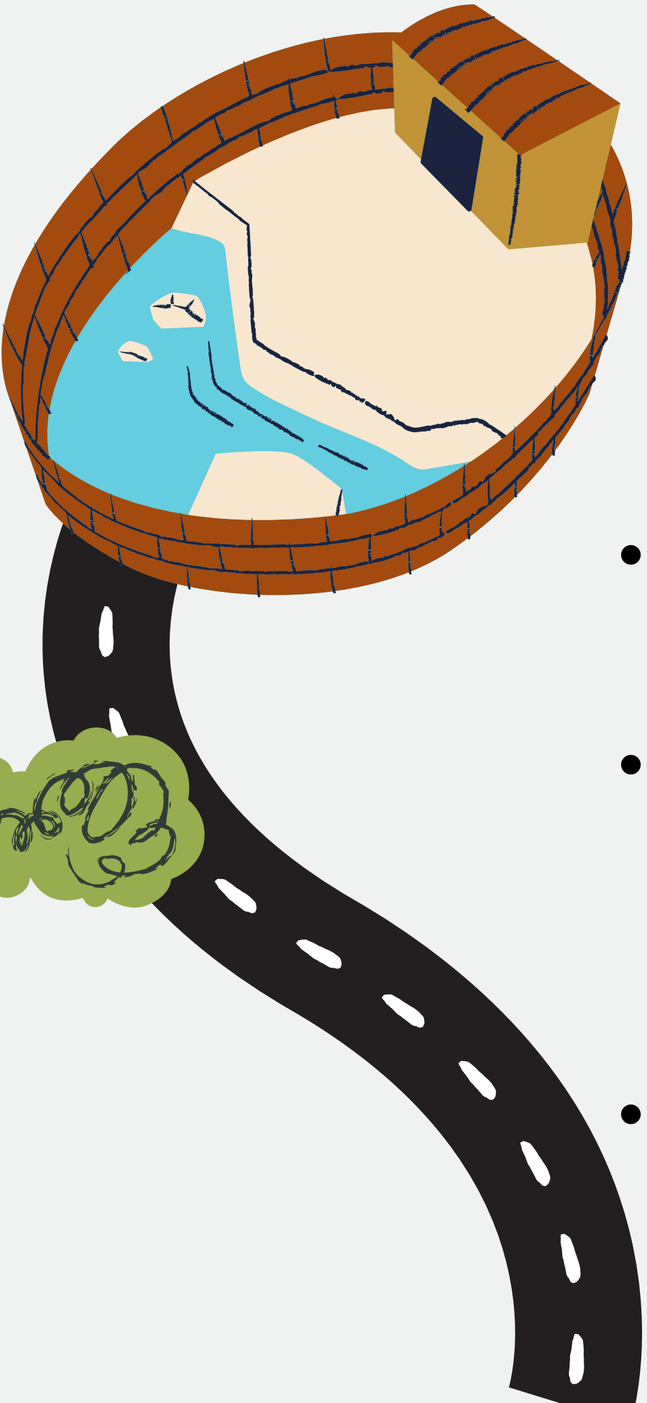
Limitations of Dijkstra's Algorithm compare to Greedy Algorithm:

1. Ignores Bus Capacity
Dijkstra only finds the shortest path — it doesn't consider how many students are in each village or if the bus is full.

2. No Route Grouping
It gives shortest paths to individual locations but doesn't decide which group of villages to visit in one trip.

3. Needs Extra Logic
Dijkstra can't plan full routes alone. It must be combined with a strategy like Greedy to make routing decisions.





PERFORMANCE ANALYSIS

GREEDY ALGORITHM

Time Complexity

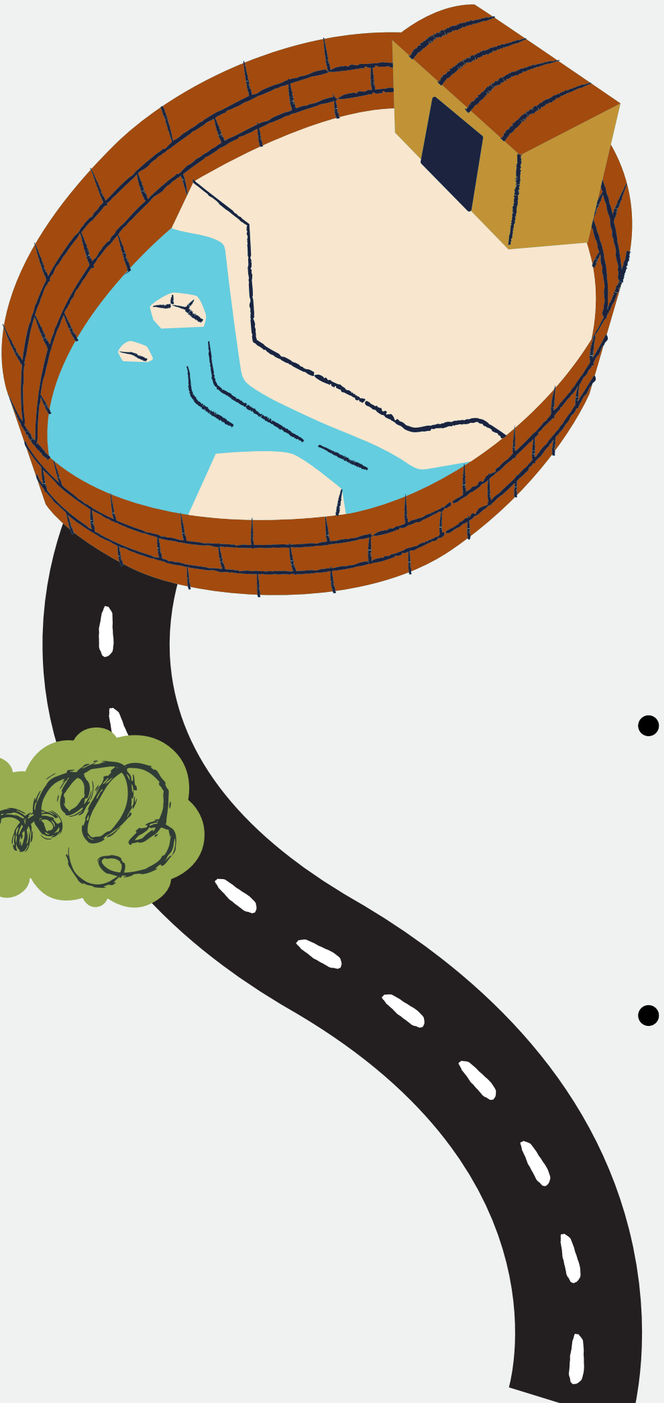
- Worst: $O(n^2)$ →The bus can only pick up one village at a time due to capacity limits,
- Average: O(n log n) → there's a more balanced distribution of students, uses efficient selection methods like priority queues.
- Best: O(n) → All villages picked up in one trip.

Space Complexity

- Use a distance matrix that takes $O(n^2)$ space,
- O(n) for visited array and other tracking variables

Scalability

- Efficient for 10–100 villages
- Less efficient for very large networks unless optimized



PERFORMANCE ANALYSIS

DIJKSTRA'S ALGORITHM

Time Complexity

- Its time complexity is O((V + E) log V) using a binary heap for the priority queue.
- This means it’s highly efficient for large graphs, even with thousands of nodes.

Space Complexity

- it uses O(V + E) — that includes the graph itself, stored as an adjacency list
- arrays for tracking the shortest distances and paths.

Scalability

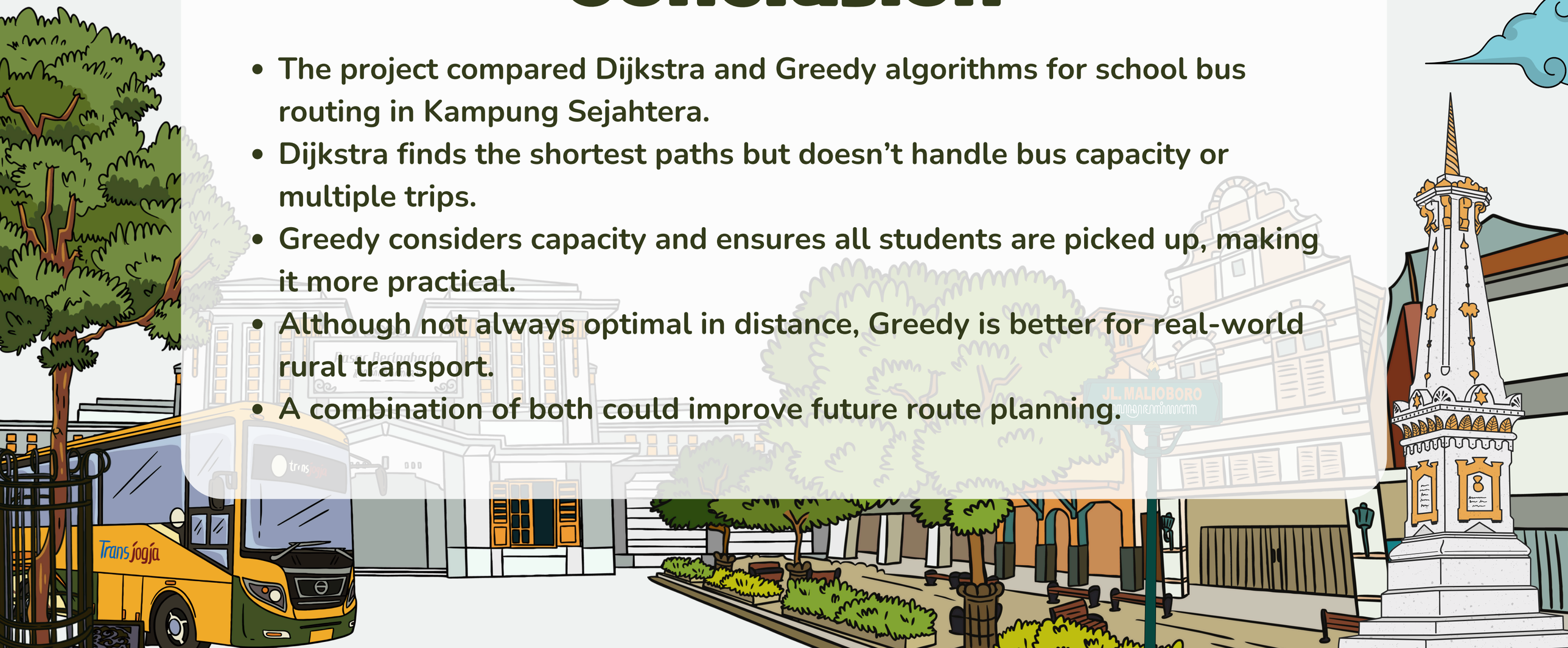
- Works well for large graphs
- Needs pairing with Greedy for complete route planning

ALGORITHM COMPARISON

Feature	Dijkstra Algorithm	Greedy Algorithm
Goal	Finds the shortest distance from one node to all others	Picks the next nearest unvisited village
Optimally	Always gives optimal shortest path in weighted graphs	Not guaranteed to give the best overall route
Speed	Fast: O((V + E)log V) with a priority queque	Very fast: O(n) per step, but less informed
Simplicity	Simple logic and implementation	Requires more structures
Scalability	Good for small to medium graphs	Good, but more complex for multiple routes
Best Use Case	Fast, approximate routing	Accurate pathfinding in complex maps

Conclusion

- The project compared Dijkstra and Greedy algorithms for school bus routing in Kampung Sejahtera.
- Dijkstra finds the shortest paths but doesn’t handle bus capacity or multiple trips.
- Greedy considers capacity and ensures all students are picked up, making it more practical.
- Although not always optimal in distance, Greedy is better for real-world rural transport.
- A combination of both could improve future route planning.



Thank You