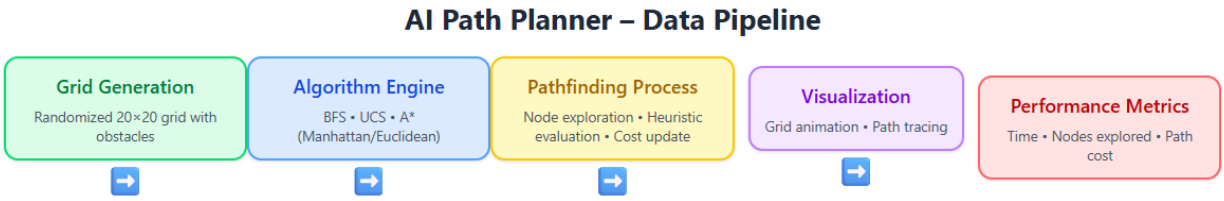
Project's Write Up

System Explanation

The AI Path Planner demonstrates grid-based navigation using classical AI search algorithms.
Input: 20×20 grid with random obstacles.
Output: Shortest path, path cost, execution time, and nodes explored.
Techniques: BFS, UCS, A*(Manhattan), A*(Euclidean), Goal-Conditioned Q-Learning (Reinforcement Learning)

Pipeline:
Grid Generation → Algorithm Execution → Visualization → Performance Metrics



Limitations: RL training currently limited to small grids and discrete actions
A* operates on static maps only (no dynamic obstacle re-planning)

Mitigation: Code modularized for future enhancements (dynamic path re-planning, weighted terrains).

Feature Table

Description	Platform	Complete (1–5)	Code	Notes
Grid generation	Local	5	Python	Fully functional with seed option
BFS	Local	5	Python	Guaranteed optimal path
UCS	Local	5	Python	Handles uniform edge cost
A* (Manhattan)	Local	5	Python	Most efficient and fastest

A* (Euclidean)	Local	4.5	Python	Slightly slower due to float operations
Visualization (4-panel)	Local	5	Matplotlib	Animated exploration comparison
Performance metrics & logging	Local	5	JSON / Matplotlib	Generates charts automatically
Goal-Conditioned Q-Learning	Local	5	Python	Goal-Conditioned Q-Learning
RL vs A* Comparison	Local	5	Python / Matplotlib	Displays both policies side-by-side

External Tools & Libraries

- **NumPy** : Grid representation and matrix operations
- **Matplotlib** : Visualization and charting
- **JSON** : Result logging
- **OS** :File system interaction

No external datasets used; all code is original and self-contained.

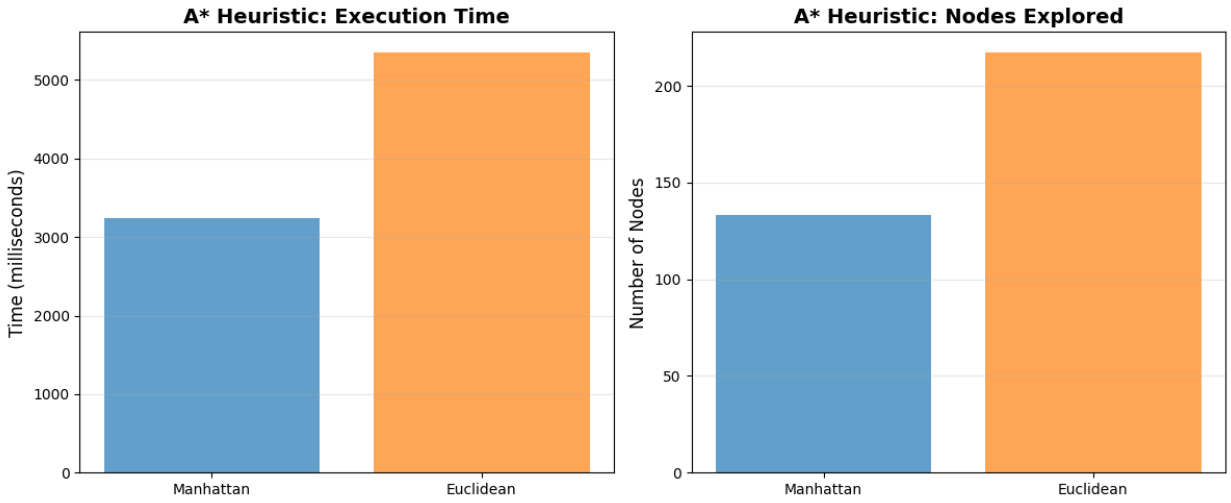
Results & Analysis

- All algorithms found the optimal path.
- **A*(Manhattan)** achieved the best efficiency with 54.9% fewer explored nodes than BFS and fastest runtime (3.38 s).
- BFS and UCS showed similar node counts but higher execution times.
- A*(Euclidean) was accurate but slightly slower due to floating-point distance computations.
  These results confirm that heuristic-guided search significantly reduces search space while preserving optimality.

Reinforcement Learning

- Introduced a **Goal-Conditioned Q-Learning agent** trained on user-selected goals via interactive grid clicks.
- Agent trained over **2400 episodes** (4 segments × 600 episodes each) and achieved **~100 % segment success** by convergence.

- During evaluation, both **RL** and **A*** completed the multi-goal route:



Interpretation:

The RL agent learns to generalize goal-directed behavior through experience, trading off minimal path optimality for **significant gains in efficiency**.
This experiment highlights how **heuristic search (A*)** and **reinforcement learning** can complement each other: A* ensures optimality, while RL enables faster, adaptive navigation.

Conclusion

The **AI Path Planner** successfully integrates both **classical search algorithms** and **reinforcement learning** to demonstrate how different AI paradigms approach problem solving.
While *A (Manhattan)** remains the most optimal in static environments, the **Goal-Conditioned Q-Learning agent** showcases adaptive, efficient learning suited for dynamic or partially known worlds.

Future Work:

- Incorporate **dynamic obstacle detection and real-time re-planning**.
- Extend the environment to **weighted terrains** (variable movement costs).
- Develop **reinforcement learning with continuous state spaces** and **policy gradient methods** for smoother navigation.
- Deploy a **hybrid planner** that switches between A* and RL based on uncertainty or environment changes.