AUTONOMOUS DELIVERY AGENT

SUBMITTED BY :- Arjun Mishra

REGISTRATION NUMBER:- 25BCY10063

SUBMITTED TO :- VIT Bhopal

ACADEMIC YEAR :- 2025-2026

INTRODUCTION

This project is a simulation project where an autonomous agent (a virtual robot) navigates a 2D "grid city." The city is mapped out like a chessboard, filled with open roads and obstacles (buildings, traffic jams, or construction zones).

PROBLEM STATEMENT

Current delivery robots blindly follow the shortest path (distance). They often fail because they ignore real-world factors like steep hills, rough terrain, or crowded areas, leading to dead batteries, stuck robots, and late deliveries.

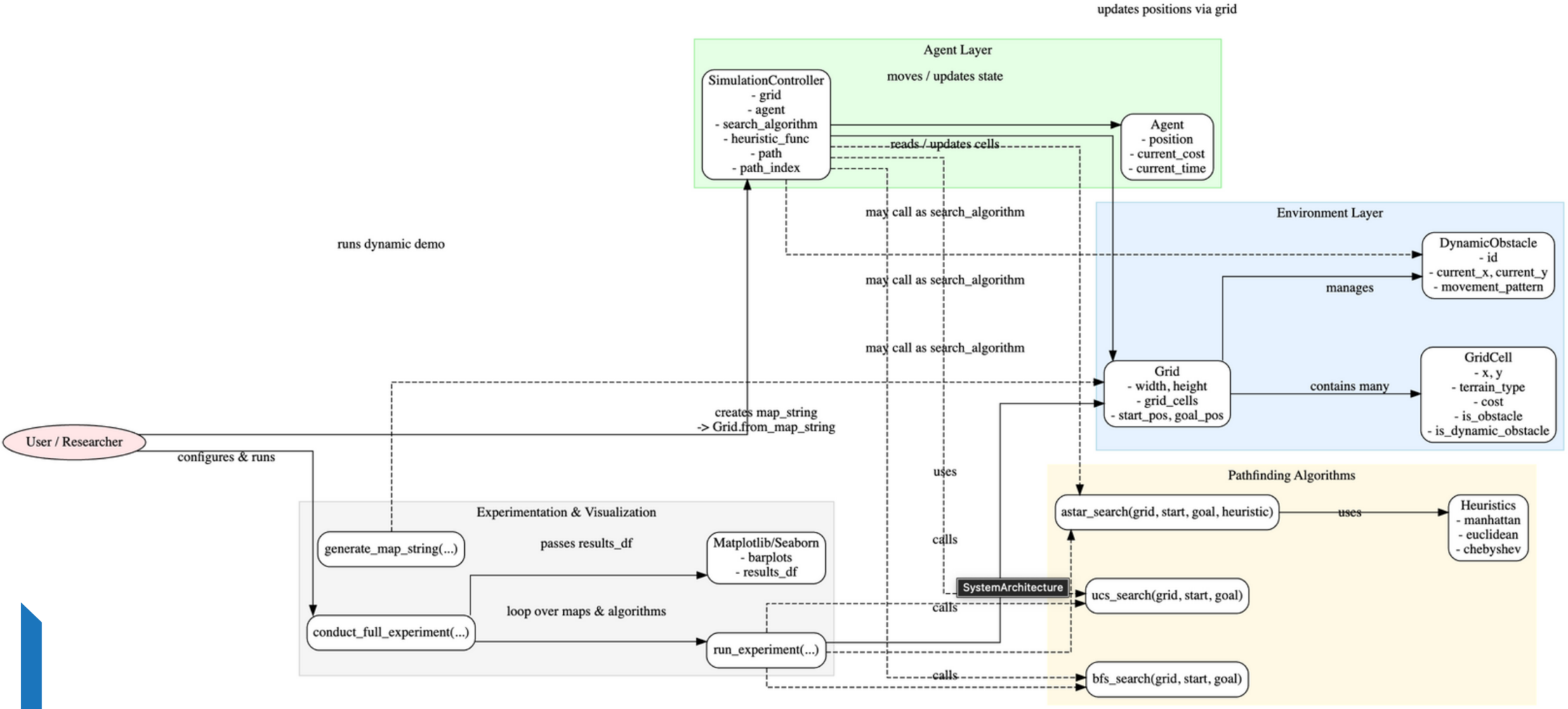
FUNCTIONAL REQUIREMENTS

- Manage Orders: Receive, assign, and track delivery orders.
- Route Optimization: Plan efficient delivery routes.
- Real-time Tracking: Monitor driver and package locations.
- Status Updates: Allow drivers to update delivery status and send notifications to customers.
- Proof of Delivery: Capture confirmation of delivery (e.g., signature, photo).

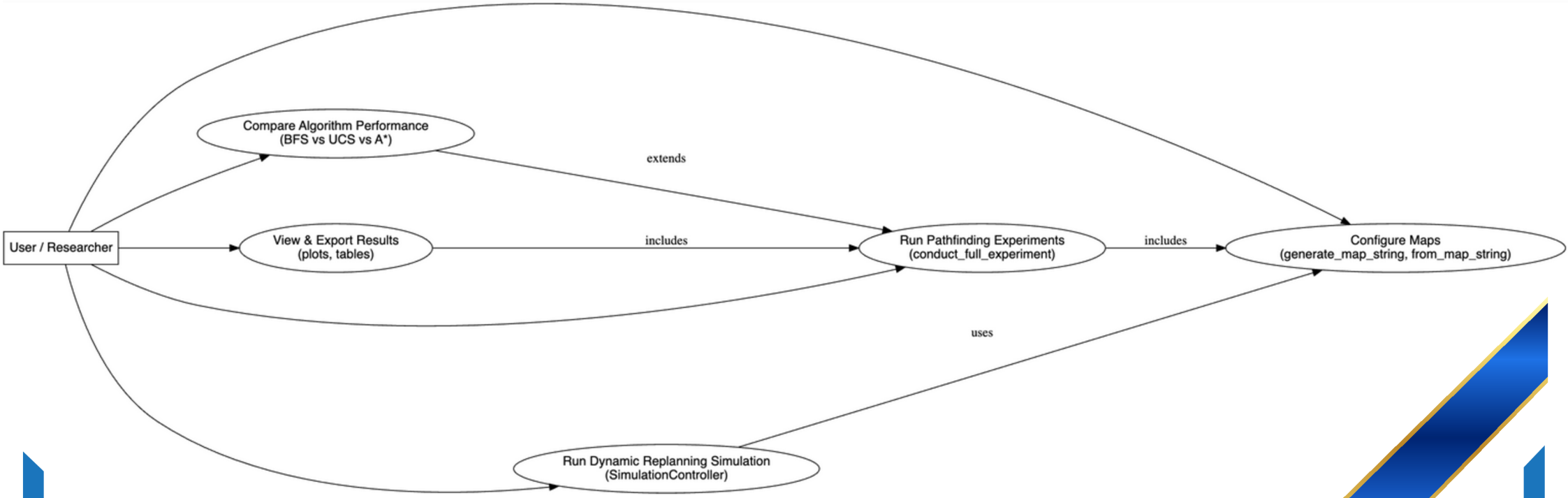
NON-FUNCTIONAL REQUIREMENTS

- Performance: Handle many orders quickly (scalability, response time).
- Reliability: Be available and recover from errors (uptime, fault tolerance).
- Usability: Be easy for drivers and dispatchers to use.
- Security: Protect data and prevent unauthorized access.
- Maintainability: Be easy to update and fix.
- Compatibility: Work with various devices and other systems.

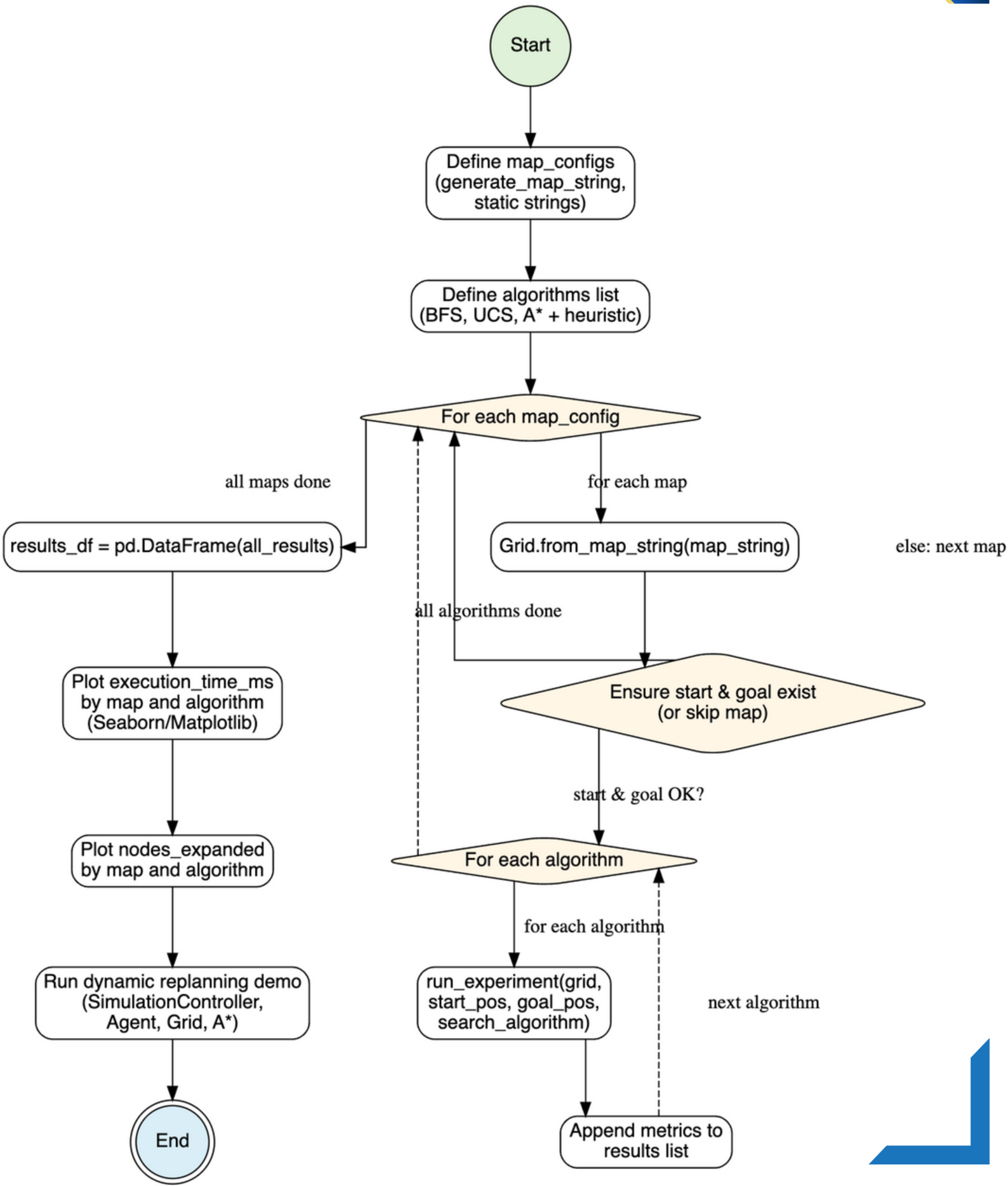
SYSTEM ARCHITECTURE



USE CASE DIAGRAM



WORKFLOW DIAGRAM



SEQUENCE DIAGRAM

DESIGN DECISIONS & RATIONALE

- Modularity: System components (Grid, Agent, Algorithms, Controller) are separated for clarity, reusability, and easier maintenance.
- Data Structures: deque for BFS, heapq for UCS/A*, and dictionaries for visited states ensure efficient algorithm performance.
- Reactive Replanning: Chosen for dynamic obstacles due to its simplicity, robustness to unpredictable movements, and efficiency, focusing on adapting to current environment changes rather than complex predictions.

IMPLEMENTATION DETAILS

- GridCell Flags: is_obstacle and is_dynamic_obstacle boolean flags efficiently track static and moving blockages.
- Dynamic Obstacle Movement: Obstacles follow predefined movement_patterns, with the Grid updating cell flags (is_dynamic_obstacle) accordingly each step.
- Node Objects: Store position, accumulated cost (g_cost), and a parent reference for path reconstruction.
- SimulationController's Replanning: At each step, it updates dynamic obstacles, checks if the agent's next step is blocked, and if so, triggers a new pathfinding search from the agent's current position.

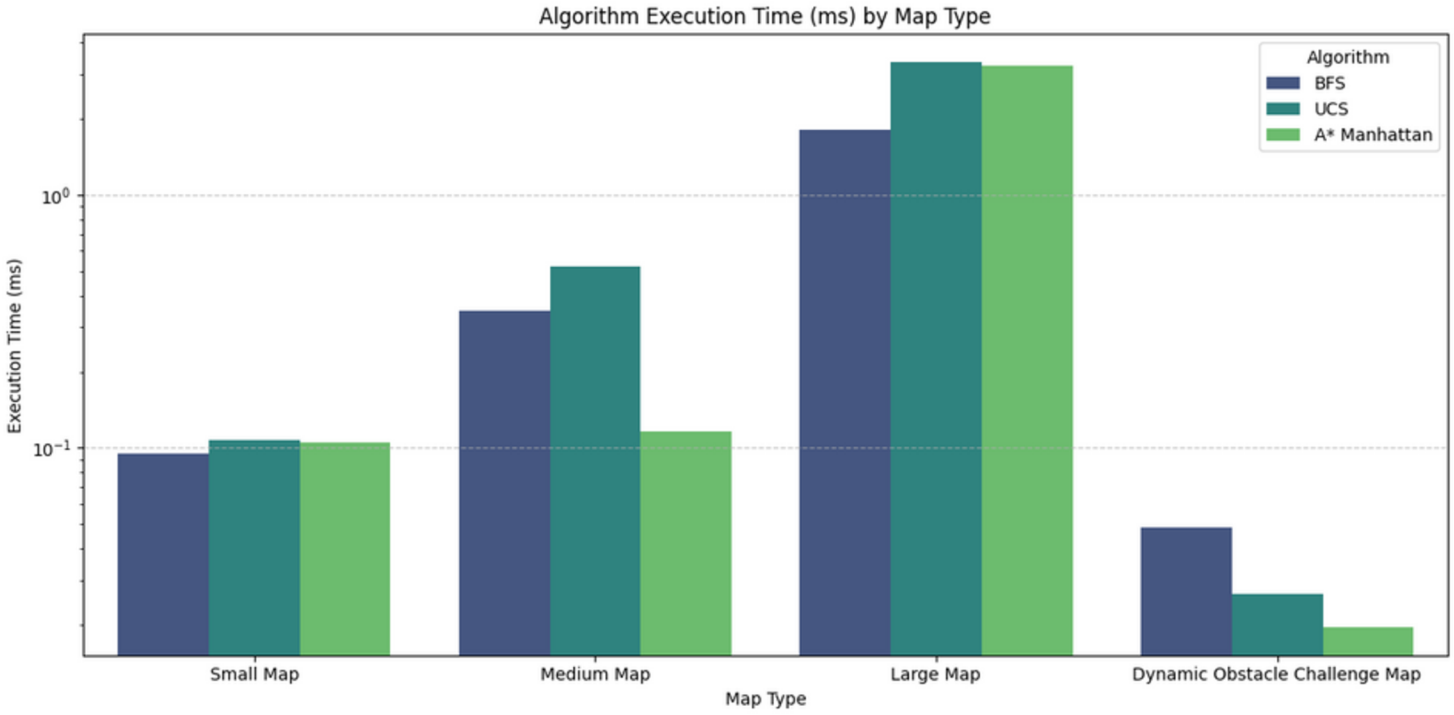
SCREENSHOTS AND RESULTS

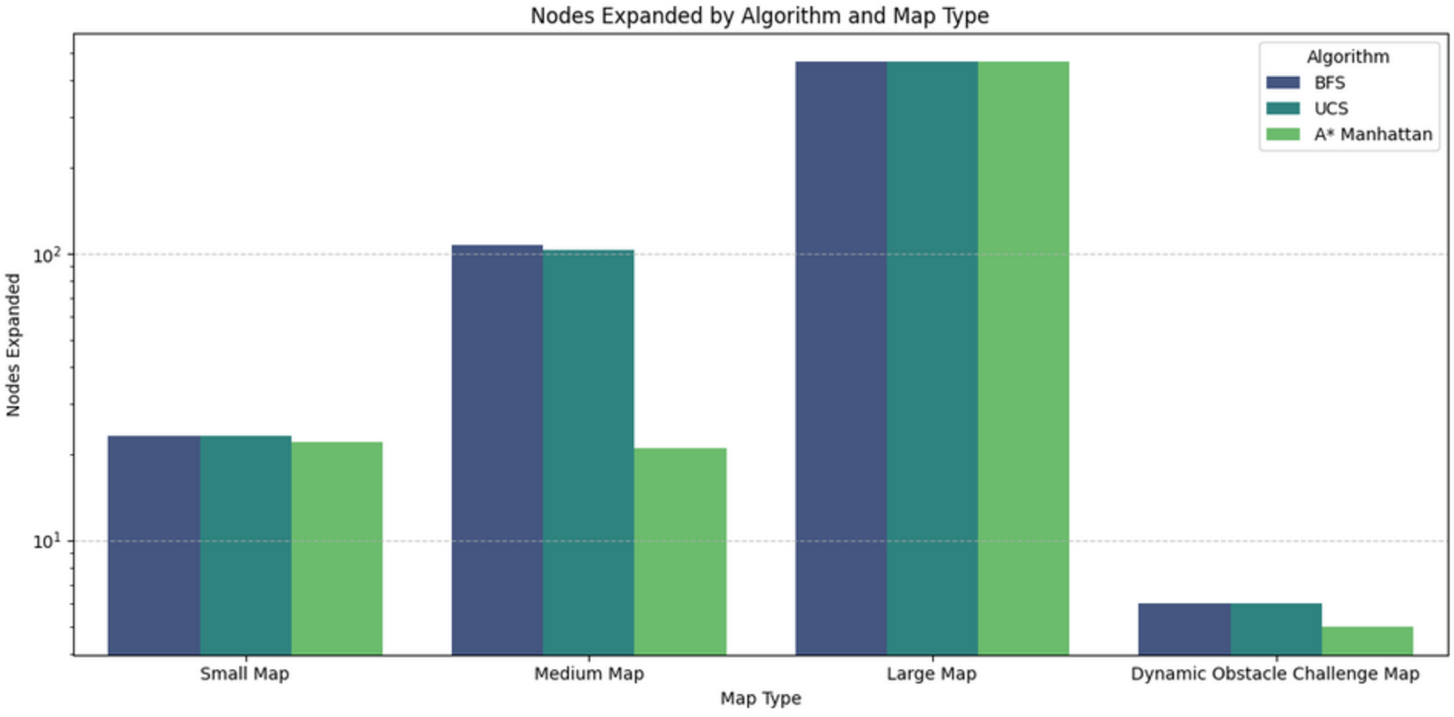

```
--- Setting Up Maps ---

Small Map:
S..#.
.....
.....
...#.
....G

Medium Map:
(Details hidden due to size, but loaded)

Large Map:
(Details hidden due to size, but loaded)

Dynamic Replanning Demo Map:

S.X
.D.
.G.


--- Conducting Full Experiment ---

Processing Map: Small Map
  Running BFS on Small Map...
  Running UCS on Small Map...
  Running A* Manhattan on Small Map...

Processing Map: Medium Map
  Running BFS on Medium Map...
  Running UCS on Medium Map...
  Running A* Manhattan on Medium Map...

Processing Map: Large Map
  Running BFS on Large Map...
  Running UCS on Large Map...
  Running A* Manhattan on Large Map...

Processing Map: Dynamic Obstacle Challenge Map
  Running BFS on Dynamic Obstacle Challenge Map...
  Running UCS on Dynamic Obstacle Challenge Map...
  Running A* Manhattan on Dynamic Obstacle Challenge Map...
```

```
--- Running Dynamic Replanning Simulation (Proof-of-Concept) ---
Initial path planned: [(0, 0), (0, 1), (0, 2), (1, 2)] with cost 3.00

>>> Simulation Step 1 <<<

--- Updating Dynamic Obstacles ---
  Obstacle 0 moved to (0, 1)
!!! Collision detected at next step (0, 1)! Replanning needed.
--- Replanning Path ---
New path planned: [(0, 0), (1, 0), (1, 1), (1, 2)] with cost 3.00
Agent moved to (1, 0). Total cost: 1.00, Time: 1
Agent Status: Position=(1, 0), Cost=1.00, Time=1

>>> Simulation Step 2 <<<

--- Updating Dynamic Obstacles ---
  Obstacle 0 moved to (0, 1)
Agent moved to (1, 1). Total cost: 2.00, Time: 2
Agent Status: Position=(1, 1), Cost=2.00, Time=2

>>> Simulation Step 3 <<<

--- Updating Dynamic Obstacles ---
  Obstacle 0 moved to (0, 1)
Agent moved to (1, 2). Total cost: 3.00, Time: 3
Agent reached the goal!
Agent Status: Position=(1, 2), Cost=3.00, Time=3

--- Dynamic Replanning Simulation Summary ---
Final Agent Position: (1, 2)
Total Cost: 3.00
Total Time Steps: 3
Goal Reached: True

All simulations and experiments completed.
```

```
--- Experiment Results ---
| algorithm     | start_pos   | goal_pos   | path_found   |   path_length |   path_cost |   execution_time_ms |   nodes_expanded | map_name                          |   map_width |   map_height |
|:--------------|:------------|:-----------|:-------------|--------------:|------------:|--------------------:|-----------------:|:----------------------------------|------------:|-------------:|
| BFS           | (0, 0)      | (4, 4)     | True         |             9 |           8 |            0.094595 |               23 | Small Map                         |           5 |            5 |
| UCS           | (0, 0)      | (4, 4)     | True         |             9 |           8 |            0.106895 |               23 | Small Map                         |           5 |            5 |
| A* Manhattan  | (0, 0)      | (4, 4)     | True         |             9 |           8 |            0.1051   |               22 | Small Map                         |           5 |            5 |
| BFS           | (6, 7)      | (5, 13)    | True         |            10 |           9 |            0.347142 |              107 | Medium Map                        |          15 |           15 |
| UCS           | (6, 7)      | (5, 13)    | True         |            10 |           9 |            0.523372 |              103 | Medium Map                        |          15 |           15 |
| A* Manhattan  | (6, 7)      | (5, 13)    | True         |            10 |           9 |            0.116129 |               21 | Medium Map                        |          15 |           15 |
| BFS           | (24, 15)    | (18, 0)    | False        |             0 |         inf |            1.80804  |              465 | Large Map                         |          25 |           25 |
| UCS           | (24, 15)    | (18, 0)    | False        |             0 |         inf |            3.33924  |              465 | Large Map                         |          25 |           25 |
| A* Manhattan  | (24, 15)    | (18, 0)    | False        |             0 |         inf |            3.23899  |              465 | Large Map                         |          25 |           25 |
| BFS           | (0, 0)      | (1, 2)     | True         |             4 |           3 |            0.048572 |                6 | Dynamic Obstacle Challenge Map    |           3 |            3 |
| UCS           | (0, 0)      | (1, 2)     | True         |             4 |           3 |            0.026568 |                6 | Dynamic Obstacle Challenge Map    |           3 |            3 |
| A* Manhattan  | (0, 0)      | (1, 2)     | True         |             4 |           3 |            0.019611 |                5 | Dynamic Obstacle Challenge Map    |           3 |            3 |
```

SCREENSHOTS AND RESULTS





TEST APPROACH

- Quantitative Evaluation: Pathfinding algorithms (BFS, UCS, A[Asterisk]) are tested on various map sizes (small, medium, large) and complexities. Metrics like path cost, length, execution time, and nodes expanded are collected and analyzed.
- Qualitative Demo: A dynamic replanning scenario is used to validate the agent's adaptive behavior, showing its ability to detect and re-plan around moving obstacles.

CHALLENGES FACED

- Dynamic Updates
- Algorithm Correctness
- Environment Set-Up
- Importing Maps and Linking them to main.py

LEARNING AND KEY TAKEAWAYS

- End to end ML Work Flow.
- Getting familiar with pathfinding algorithms.
- Ability to handle multiple complexities.

FUTURE ENHANCEMENTS

- Advanced Obstacle Behaviors:- More complex, reactive, or predictive movements.
- Multi-Agent Systems:- Pathfinding and collision avoidance for multiple agents.
- Diverse Map Generation:- Create mazes, varied terrain costs, and more complex environments.
- Sophisticated Replanning:- Techniques like partial replanning or time-dependent pathfinding.
- Enhanced Visualization:- Graphical interface to observe simulation in real-time

REFRENCES

- Pandas Documentation
- Google Collab
- Git-hub