### **Prototype Description**

*Project Name:* AI Pathfinding Showcase: BFS vs A\*  
*Student Name:* Tony Djikigoue  
*Supervisor:* Aboozar Taherkhani

(500 words)

**1. Project Overview**

This prototype is an interactive visualization of two fundamental pathfinding algorithms—Breadth-First Search (BFS) and A\* within a simple 2D grid environment built in Unity.

The goal is to let users place a start and end tile, optionally add or remove obstacles, then watch each algorithm explore the grid step by step, highlighting the “open” frontier, the “closed” set, and finally the computed shortest path. It’s designed both as a teaching aid for someone learning how these algorithms differ in exploration pattern and as a technical showcase of implementing coroutines, custom shaders/colours, and UI workflows in Unity.

**2. Core Features**

* **Grid Generation**  
  Automatically spawns a configurable **width × height** grid of clickable tile prefabs, centred in world space.
* **Obstacle Toggling**Click on any non-start/end tile to toggle between “walkable” (white) and “blocked” (red).
* **Start/End Selection**  
  Four UI buttons allow you to “Select Start,” “Select End,” “Reset Grid,” and “Run BFS.” Selecting Start/End enters a mode where the next click designates that tile as the start (green) or end (magenta).
* **Step-by-Step BFS Visualisation**When you hit “Run BFS,” the prototype runs a coroutine that:  
  - Colours each dequeued tile orange (closed set) with a small delay.  
  - Colours each newly discovered neighbour yellow (open set).  
  - Upon reaching the end, backtracks through parent pointers to paint the final path green.  
  - Smoothly animates an agent sprite moving tile-by-tile along the discovered path.
* **Responsive UI**All control buttons are automatically disabled during the search/movement phase and re-enabled at the end.

**3. Technical Implementation**

**Engine & Language:** Unity 2021.3 (LTS) / C#

* **Grid & Tiles:** Each tile is a Tile MonoBehaviour with a SpriteRenderer. State flags *(isWalkable, isStart, isEnd)* drive its colour via *UpdateColor().* Clicks are handled in *OnMouseDown()* and forwarded to *GridManager.Instance.HandleTileClick(this).*
* **UI:** A Canvas in Screen Space–Overlay holds a dark, semi-transparent panel with a *VerticalLayoutGroup* and four Unity UI Buttons. Each button is wired (in Start()) via *Btn\_SelectStart.onClick.AddListener(SelectStartMode)* (and similarly for the others). Buttons are styled with sliced 9-patch sprites for rounded corners and a neutral grey. Text uses TextMeshPro for crisp rendering.
* **Agent Movement:** The agent prefab is instantiated on first run, positioned at the start tile, and then moved via Vector3.Lerp() inside a simple coroutine (MoveAgentAlong), giving a smooth slide between squares.

**4. Limitations & Future Development**

**Single Algorithm Implemented:** Currently only BFS is visualised; A\* (with a heuristic) will be added next, allowing direct performance comparison.

**Fixed Grid Size & Topology:** The grid is square and only supports 4-way connectivity. Future iterations could support user-driven resizing, non-uniform shapes, and 8-way (diagonal) movement.

**Performance at Scale:** At larger grid sizes (e.g. 100×100), the step delays and per-frame updates may become sluggish.

**UI/UX Enhancements:**

* Add “Step” and “Step Back” buttons for manual control.
* Display open/closed set counts, path length, and elapsed time.
* Allow dynamic obstacle drawing/erasing rather than per-tile clicking.