**Project 2: Algorithms and Data Structure Enhancement - Narrative**

Diana Galvez Mendez

The artifact I selected for this enhancement is my Pirate Intelligent Agent: Artificial Intelligence. This project used Deep Q-Learning (DQN) to train a pirate agent to navigate a maze, avoid obstacles, and find treasure. The maze was represented as a 2D grid, with rewards and penalties shaping the agent’s learning process.

When I first created this, everything worked, but it was all in one Jupyter notebook, which made it hard to maintain or share. The structure wasn’t great, there wasn’t much visibility into how the agent was learning, and it felt more like a “class project” than something I’d proudly showcase.

I chose this project because it highlights my understanding of algorithms and how they can be applied to solve real-world problems. Reinforcement learning is a great way to show this, it’s about teaching a system to make decisions by learning from trial and error. This project gave me a way to demonstrate that in a visual, interactive way.

This artifact demonstrates:

* Implementing a **neural network** to approximate Q-values for each possible action
* Using a **memory buffer** for experience replay, improving learning efficiency
* Representing the maze environment as a **matrix** for state management
* Applying **epsilon-greedy logic** to balance exploration and exploitation
* Structuring the project into **modular files** for clarity and reuse

For this enhancement, I:

* **Refactored** the project into separate files: agent.py, environment.py, and train.py
* **Tuned parameters** such as learning rate, epsilon decay, and reward values
* **Added visualizations** to track rewards, win counts, and epsilon over time
* Improved **code organization** so it’s easier to follow, test, and reuse

This directly supports the program outcome of designing and evaluating solutions using algorithms while managing trade-offs. I not only applied reinforcement learning from scratch, but I also went back to improve performance, logic, and maintainability.

My goal was to improve structure, enhance performance, and make the learning process visible. The biggest challenge was troubleshooting why episode logs weren’t printing as expected, but I worked around it by focusing on the reward and epsilon plots to measure progress. I also had to handle TensorFlow/Keras warnings, but I didn’t let that slow me down.

The most rewarding part was seeing the agent’s learning curve, starting with random moves, then gradually figuring out the treasure’s path. That’s when it felt “real.”

This project now shows the difference between code that “just works” and code that’s built to last. It’s cleaner, easier to update, and something I can confidently present. It reflects how I solve problems with logic, pay attention to clean code, and am willing to revisit and improve my own work.