# Code Review Script

## Introduction

Hello, my name is Sydney Porter, and this is my code review for CS 499. In this video, I will be reviewing three artifacts that demonstrate my knowledge and skills in three key areas: Software Engineering and Design, Algorithms and Data Structures, and Databases. I will highlight the current functionality of the code, identify areas for improvement, and explain how I plan to enhance these artifacts to align with the course outcomes and prepare them for my ePortfolio.

## Category One: Software Engineering and Design

For this category, I have selected the `Task` and `TaskService` classes from a project I completed in CS 320. This project demonstrates my ability to design and implement software solutions using object-oriented principles.

### Existing Code Functionality

- The `Task` class represents a task with three attributes: `taskId`, `name`, and `description`. These attributes are private, with getter methods to ensure encapsulated access.  
- The `TaskService` class manages tasks using a `HashMap`. It provides methods to add, delete, and update tasks by interacting with the `Task` objects stored in the `HashMap`.

### Code Review Findings

- \*\*Strengths\*\*:  
 - The code adheres to object-oriented principles, with clear separation of concerns between data representation (`Task`) and business logic (`TaskService`).  
 - The use of a `HashMap` ensures efficient retrieval and modification of tasks.  
- \*\*Weaknesses\*\*:  
 - The code lacks error handling, such as for cases when a task ID is not found.  
 - There is no input validation for `taskId`, `name`, or `description`.  
 - Minimal comments and no Javadoc make the code harder to understand and maintain.

### Proposed Enhancements

To improve this artifact:  
1. Implement error handling, such as throwing exceptions for invalid operations like updating a non-existent task.  
2. Add input validation to ensure attributes are non-null and non-empty.  
3. Document the code with detailed Javadoc and inline comments.  
4. To address scalability, replace the `HashMap` with a thread-safe `ConcurrentHashMap` to ensure the system supports concurrent access.

### Skills and Outcomes

These enhancements will showcase my ability to design modular, scalable, and secure software solutions. They align with course outcomes focused on secure coding, modular development, and professional-quality documentation.

## Category Two: Algorithms and Data Structures

For this category, I have selected the Deep Q-Network (DQN) implementation from my CS 370 project. This project demonstrates my ability to solve real-world problems using machine learning algorithms.

### Existing Code Functionality

- This artifact uses a neural network to train a pirate agent to navigate an 8x8 maze and find a treasure.  
- Key features include:  
 - A reward system to incentivize exploration and avoid obstacles.  
 - Q-learning with epsilon-greedy exploration to balance learning efficiency and exploration.  
 - Replay memory to store and retrieve state-action pairs for training.

### Code Review Findings

- \*\*Strengths\*\*:  
 - The implementation of epsilon-greedy exploration ensures efficient agent learning.  
 - Replay memory optimizes learning by reusing past experiences.  
 - The code structure is modular, with separate classes for the maze environment and experience replay.  
- \*\*Weaknesses\*\*:  
 - The reward function is basic and may not fully guide the agent toward optimal paths.  
 - Parameters like epsilon decay and learning rate are not optimized.  
 - There is no input validation or error handling for invalid agent actions or state transitions.

### Proposed Enhancements

To enhance this artifact:  
1. Refine the reward function to include penalties for revisiting cells and bonuses for efficient paths.  
2. Experiment with different parameter values to optimize agent performance.  
3. Validate inputs to ensure actions are valid and add exception handling for edge cases.  
4. Expand the neural network to include additional layers for better scalability and performance.

### Skills and Outcomes

These enhancements will demonstrate my ability to apply algorithms and data structures to solve complex problems effectively. They align with course outcomes related to secure coding, innovation, and algorithmic problem-solving.

## Category Three: Databases

For this category, I have selected the `AnimalShelter` class from my CS 340 project. This artifact demonstrates my ability to design and manage databases.

### Existing Code Functionality

- The `AnimalShelter` class interacts with a MongoDB database to manage animal records.  
- It provides CRUD operations:  
 - `create(data)`: Inserts new records.  
 - `read(query)`: Retrieves records based on a filter.  
 - `update(query, data)`: Updates records matching a filter.  
 - `delete(query)`: Deletes records based on a filter.

### Code Review Findings

- \*\*Strengths\*\*:  
 - The code is modular, with all CRUD operations encapsulated in a single class.  
 - The use of MongoDB allows for flexible and scalable data storage.  
- \*\*Weaknesses\*\*:  
 - Hardcoded database credentials pose a security risk.  
 - CRUD methods lack input validation, increasing the risk of injection attacks.  
 - There is no error handling for database connection failures or invalid queries.  
 - The `read` method does not support pagination, which can lead to performance issues with large datasets.

### Proposed Enhancements

To improve this artifact:  
1. Store database credentials in environment variables to enhance security.  
2. Validate all inputs to prevent injection attacks and ensure data integrity.  
3. Add error handling for connection issues and query failures.  
4. Introduce pagination in the `read` method and indexing in the database to optimize query performance.

### Skills and Outcomes

These enhancements will demonstrate my ability to manage databases securely and efficiently. They align with course outcomes related to secure coding, data management, and scalable system design.

## Conclusion

In this review, I analyzed three artifacts:  
1. The `Task` and `TaskService` classes for Software Engineering and Design.  
2. The Deep Q-Network implementation for Algorithms and Data Structures.  
3. The `AnimalShelter` class for Databases.  
  
I identified areas for improvement and proposed enhancements to make the code more secure, efficient, and aligned with professional standards. These updates will demonstrate my skills in secure coding, modular design, algorithmic problem-solving, and database management, preparing these artifacts for inclusion in my ePortfolio. Thank you for watching my code review!