**Code Review Journal**

**Part 1: The Role and Importance of Code Review**

A code review is the systematic examination of source code by peers or developers to identify defects, improve quality, and ensure that the code follows organizational standards. Rather than just debugging, code reviews aim to improve readability, maintainability, and security while also serving as an opportunity for knowledge sharing across a team.

Code review is an important practice for computer science professionals because it reduces the likelihood of hidden defects reaching production, improves long-term maintainability, and helps ensure that the design aligns with business requirements. For example, in my own project, reviewing the AnimalShelter class and the Dash notebook allowed me to identify issues such as outdated database methods, hard-coded credentials, and missing error handling. Without review, these problems might not have been caught until much later, when they would be more costly to fix.

Some best practices I learned from resources include keeping reviews small and focused, setting clear objectives, and providing constructive feedback rather than criticism. It is also important to check for documentation and test coverage, not just logic correctness. A best practice is to review code early and often—ideally at each major milestone or after each feature is implemented—so that issues are caught before they cascade into other parts of the system. Conducting code reviews late in the development cycle risks rework and delays, whereas early reviews align with agile principles of iterative feedback and continuous improvement.

**Part 2: My Approach to Conducting the Code Review**

For recording my code review, I chose to use **Bandicam**, since it allows me to capture both my screen and audio narration. This makes it easy to display my CRUD.py file and my Jupyter Notebook dashboard.

My approach to preparing the code review began with writing an outline and later expanding it into a detailed script. I organized the script around the three categories that I will be reviewing:

1. **Software Design and Engineering** – Here I planned to walk through the overall design of the AnimalShelter class, highlighting strengths such as clear structure, but also pointing out weaknesses such as outdated method calls, lack of documentation, and poor error handling. My outline ensured I explained what the code currently does, why certain issues matter, and what enhancements I plan to make.
2. **Algorithms and Data Structures** – In this section, I focused on how filters and queries are built in the dashboard. My outline included noting inefficiencies like applying filters in Python after fetching data, reliance on raw strings for breed names, and the absence of pagination. I made sure to describe both the current logic and the improvements I plan, such as centralized query builders, using constants or enums, and leveraging MongoDB aggregation pipelines.
3. **Databases** – For the database review, my outline guided me to show how the AnimalShelter class connects to MongoDB and interacts with the animals collection. I discussed the strengths of direct CRUD operations but emphasized weaknesses such as lack of projections, missing index usage, and hard-coded credentials. My script included specific enhancement plans like adding projections, ensuring indices on frequent queries, and adding connection timeouts and error handling.

By writing out a script, I was able to ensure that my code review presentation would be thorough, organized, and aligned with the rubric expectations. It also helped me keep a professional and consistent tone while recording.