# Code Review Script

Good morning, everyone. My name is Joseph Klenk, and welcome to my CS 499 capstone code review.

(Show title slide with name, CS 499, and "Code Review Presentation")

Today I'll be walking you through three artifacts from my Computer Science program that I've selected for enhancement in my ePortfolio. These artifacts span the three critical areas of computer science: Software Design and Engineering, Algorithms and Data Structures, and Databases.

(Show slide with three artifact categories and brief descriptions)

The purpose of this code review is to examine the existing functionality of each artifact, identify areas for improvement, and outline my planned enhancements that will demonstrate mastery of the five program outcomes.

*(Show agenda slide: Introduction → Artifact 1 → Artifact 2 → Artifact 3 → ePortfolio Walkthrough → Conclusion)*

Let's begin by examining each artifact in detail.

**Artifact 1: Software Design and Engineering (4 minutes)**

**Existing Functionality (1.5 minutes)**

*(Show screenshot of Android Studio project structure)*

Our first artifact is an Android Weight Tracking Application from CS 360. This mobile application provides a comprehensive personal health management solution with several key features.

*(Show screenshot of login screen)*

The application begins with a user authentication system where users can log in with their credentials. The system uses SQLite database for local storage of user accounts and weight tracking data.

*(Show screenshot of main weight tracking interface)*

Once authenticated, users access the main weight tracking interface where they can log their current weight, view their progress over time, and set weight goals.

*(Show screenshot of weight entry form)*

The application supports full CRUD operations for weight management - users can create new weight entries, read their weight history, update existing entries, and delete records when necessary.

*(Show screenshot of SMS permission dialog)*

A notable feature is the SMS notification system that alerts users when they reach their target weight goal, requiring proper Android runtime permissions.

**Code Analysis (1.5 minutes)**

*(Show screenshot of MainActivity.java with mixed responsibilities highlighted)*

Now, let's examine the areas requiring improvement. The most significant issue is the lack of proper architecture. As you can see here, the MainActivity contains UI logic, database operations, and business logic all mixed together. This violates separation of concerns and makes the code difficult to maintain and test.

*(Show screenshot of database helper class with plaintext password storage)*

From a security perspective, we have a critical vulnerability - user passwords are stored in plaintext in the SQLite database. This is completely unacceptable for any health tracking application handling personal data and represents a major security risk.

*(Show screenshot of database query without error handling)*

The error handling is minimal throughout the application. Database operations lack try-catch blocks, which means any database errors could crash the application and potentially lose weight tracking data.

*(Show screenshot of loadAllWeightEntries method loading entire dataset)*

Performance is another concern. The application loads all weight entries at once without pagination, which would cause serious performance issues for users with extensive weight tracking history.

*(Show screenshot of code with poor commenting and inconsistent naming)*

Finally, code quality issues include lack of documentation, inconsistent naming conventions, and minimal comments explaining weight goal calculations and notification logic.

**Planned Enhancements (1 minute)**

*(Show slide with enhancement plan bullet points)*

To address these issues, I will implement several key enhancements:

First, I'll restructure the entire application using the Model-View-Controller architecture pattern to properly separate concerns and improve maintainability.

Second, I'll replace the plaintext password storage with BCrypt hashing to address the security vulnerability and protect user health data.

Third, I'll implement comprehensive error handling throughout the application, particularly for database operations and SMS communications.

Fourth, I'll add a pagination system for weight history display to handle extensive tracking data efficiently.

Finally, I'll improve code quality through JavaDoc comments, consistent naming conventions, and proper documentation of weight goal algorithms.

These enhancements will demonstrate my ability to design secure, maintainable health tracking applications while managing trade-offs between security, performance, and usability.

**Artifact 2: Algorithms and Data Structures (3 minutes)**

**Existing Functionality (1 minute)**

*(Show screenshot of Python project files for the Pirate Agent)*

Our second artifact is a Pirate Intelligent Agent from CS 370, implementing reinforcement learning for pathfinding in a treasure hunt game.

*(Show screenshot of game environment/maze)*

The agent navigates through a maze environment to reach treasure while avoiding obstacles, competing against a human player.

*(Show screenshot of Q-learning algorithm code)*

The core algorithm uses deep Q-learning with a neural network to predict optimal actions based on the current game state.

*(Show screenshot of epsilon-greedy exploration code)*

The agent employs epsilon-greedy exploration strategy, balancing between exploiting known good actions and exploring new possibilities.

*(Show screenshot of basic experience replay implementation)*

The current implementation uses basic experience replay, storing past experiences in a simple list and sampling randomly for training.

**Code Analysis (1 minute)**

*(Show screenshot highlighting random sampling in experience replay)*

The main algorithmic inefficiency lies in the experience replay mechanism. Random sampling treats all experiences equally, which is suboptimal because some experiences are more valuable for learning than others.

*(Show screenshot of simple list data structure)*

The use of a basic list for storing experiences lacks prioritization and doesn't take advantage of more efficient data structures for sampling operations.

*(Show screenshot of training metrics showing slow convergence)*

Performance metrics show that the agent requires many training episodes to reach optimal performance, indicating inefficient learning.

*(Show screenshot of code with magic numbers and poor documentation)*

Code clarity issues include magic numbers for hyperparameters and insufficient comments explaining the reinforcement learning concepts.

**Planned Enhancements (1 minute)**

*(Show slide with algorithmic enhancement plan)*

To optimize this algorithm, I will implement priority-based experience replay using a heap data structure from Python's heapq module.

I'll calculate Temporal Difference errors for each experience and prioritize those with higher learning value, allowing the agent to learn from the most informative experiences first.

The enhanced system will use probability distributions for intelligent sampling rather than random selection.

I'll also refactor the code for better modularity, separating environment interaction, neural network definition, and training logic.

These improvements will demonstrate my mastery of advanced algorithms and data structures, showing measurable performance gains in learning efficiency.

**Artifact 3: Databases (3 minutes)**

**Existing Functionality (1 minute)**

*(Show screenshot of the Dash web application interface)*

Our third artifact is the Global Rain Animal Shelter Dashboard from CS 340, a Python/Dash web application that interfaces with MongoDB.

*(Show screenshot of the animal data table)*

The dashboard displays animal shelter data in an interactive table format, helping rescue organizations identify dogs suitable for specialized training programs.

*(Show screenshot of filter dropdown menus)*

Users can filter animals by rescue type and breed using dropdown menus, with results displayed in real-time.

*(Show screenshot of geolocation chart)*

The application includes geolocation visualization showing animal locations on an interactive map.

*(Show screenshot of basic find() query in code)*

The backend uses basic MongoDB find() queries to retrieve and filter data based on user selections.

**Code Analysis (1 minute)**

*(Show screenshot of simple find() queries highlighted)*

The primary limitation is the use of basic MongoDB queries without leveraging the database's advanced aggregation capabilities. Most data processing occurs in the application layer rather than at the database level.

*(Show screenshot of static dashboard that requires manual refresh)*

The dashboard displays static data and requires manual refresh to see database changes, lacking real-time updates.

*(Show screenshot of query retrieving all fields unnecessarily)*

Query efficiency is poor - the application retrieves entire documents when only specific fields are needed, wasting bandwidth and processing time.

*(Show screenshot of minimal error handling in database operations)*

Error handling for database operations is minimal, with no graceful handling of connection failures or query errors.

*(Show screenshot of scattered database code throughout application)*

Database access code is scattered throughout the application rather than centralized in a dedicated module.

**Planned Enhancements (1 minute)**

*(Show slide with database enhancement plan)*

I will implement MongoDB aggregation pipelines to perform complex data analysis directly at the database level, including age demographics and breed statistics.

I'll optimize queries using field projection and indexing strategies to improve performance with large datasets.

Real-time updates will be implemented through WebSocket connections, automatically refreshing the dashboard when data changes.

Comprehensive error handling will provide user-friendly messages for database connectivity issues and query failures.

Finally, I'll create a centralized database module to organize all MongoDB operations and improve code maintainability.

These enhancements will demonstrate my ability to design efficient database solutions that deliver significant value to end users.

**Course Outcomes Alignment (1 minute)**

*(Show slide with all five course outcomes listed)*

Before we look at the ePortfolio, let me quickly address how these enhancements align with our five program outcomes:

*(Highlight each outcome as you speak)*

**Collaborative environments:** My code review process and documentation improvements enable better team collaboration and knowledge sharing.

**Professional communications:** This presentation and my technical documentation demonstrate clear communication of complex technical concepts to diverse audiences.

**Computing solutions:** Each artifact shows algorithmic problem-solving with careful consideration of design trade-offs.

**Innovative techniques:** All three enhancements implement industry-standard best practices and cutting-edge methodologies.

**Security mindset:** Security improvements in the Android app and database validation demonstrate proactive vulnerability mitigation.

**ePortfolio Walkthrough (2 minutes)**

*(Show ePortfolio website homepage)*

Now let me walk you through my completed ePortfolio website, which showcases all these enhancements in a professional format.

*(Scroll through the Professional Self-Assessment section)*

The portfolio begins with my professional self-assessment, providing context for my growth throughout the program and how these artifacts demonstrate my capabilities.

*(Navigate to Code Review section)*

Here's the code review section where this very presentation will be embedded, along with links to my complete code review script.

*(Navigate to Software Engineering artifact section)*

Each artifact has its own dedicated section with detailed enhancement descriptions, skills demonstrated, and course outcome alignments. Notice how I've included links to both original and enhanced code repositories.

*(Navigate to Algorithms section)*

The algorithms section highlights the 28.8% performance improvement achieved through priority-based experience replay, demonstrating measurable results from my enhancements.

*(Navigate to Database section)*

The database section showcases the advanced MongoDB capabilities and real-time features that provide enhanced value to end users.

*(Scroll through technical skills and contact sections)*

The portfolio concludes with my technical skills gained throughout the program and professional contact information.

**Conclusion (1 minute)**

*(Show conclusion slide with key takeaways)*

In summary, these three artifacts demonstrate my comprehensive growth as a computer science professional across all major domains of the field.

The Android application shows my ability to design secure, maintainable mobile applications using industry-standard architectural patterns.

The intelligent agent demonstrates my mastery of advanced algorithms and data structures with measurable performance optimizations.

The database dashboard showcases my full-stack development capabilities and database optimization skills.

*(Show final slide with contact information)*

Together, these enhancements provide concrete evidence of my readiness to contribute to professional software development teams and tackle complex technical challenges.

Thank you for your time, and I'm happy to answer any questions you may have about my code review or enhancement plans.

*(End with thank you slide)*