Introduction

Hello, I'm Andy Martinez, and today I'll be walking you through my Computer Science ePortfolio code review. This presentation will cover three main artifacts that demonstrate my growth throughout the CS program: a Software Design & Engineering project, an Algorithms & Data Structures visualizer, and a Database management system. I'll show you the current functionality, identify areas for improvement, and outline my enhancement plans.

A. Existing Functionality Walkthrough

Software Design & Engineering: Animal Shelter Management System

Let me start by showing you my current Animal Shelter Management System. This is a Flask-based REST API that manages animal records in MongoDB.The system currently handles animal records with fields like name, age, animal type, breed, and outcome. It provides endpoints for creating new animals, retrieving animals with optional filtering, and updating existing records. The code includes proper validation using JSON schema and structured logging for debugging. The current implementation features basic CRUD operations with MongoDB integration, connection handling, JSON schema validation for data integrity, RESTful API endpoints for animal management, basic error handling and logging, and a health check endpoint for monitoring.

Algorithms & Data Structures: Interactive Algorithm Visualizer

Next, I have an interactive algorithm visualizer built with Flask that demonstrates various data structures and algorithms.The visualizer allows users to insert, delete, and search elements in AVL trees, traverse graphs with different algorithms, and perform heap operations. Each operation is logged with detailed steps to help users understand the algorithmic processes. The current system includes an AVL tree implementation with balancing operations, graph algorithms including BFS, DFS, and Dijkstra's shortest path, min/max heap data structure with full operations, step-by-step visualization of algorithm execution, a web-based interface for user interaction, and operation logging for educational purposes.

Databases: Contact Management System

Finally, I have a contact management system built with SQLite and Python.The system manages contacts with fields like name, email, phone, address, company, and job title. It includes performance monitoring, analytics reporting, and data export capabilities in multiple formats. The current implementation features a SQLite database with indexed queries for performance, a CLI interface for contact management operations, JSON file import/export capabilities, basic search and filtering functionality, contact analytics and reporting, database backup and restore functionality, and performance monitoring for slow query detection.

B. Code Analysis: Areas for Improvement

After reviewing my code, I've identified several areas where I can demonstrate advanced skills and professional development.

Software Design & Engineering Improvements

The current system is monolithic, so I can implement microservices architecture with service discovery and load balancing. Security is a major area for improvement - I need to add JWT authentication, role-based access control, and API rate limiting. The system lacks advanced analytics, so I plan to implement a real-time analytics dashboard with machine learning for outcome prediction. Performance can be enhanced through caching with Redis, connection pooling, and query optimization. Finally, I need to implement comprehensive unit and integration testing with a CI/CD pipeline.

Algorithms & Data Structures Improvements

I can add advanced data structures like Red-Black trees, B-trees, Skip lists, and Trie structures to demonstrate deeper algorithmic knowledge. Performance analytics is missing, so I'll implement real-time complexity analysis and memory usage tracking. Machine learning integration will allow for algorithm selection based on data characteristics. Distributed computing capabilities will include parallel algorithms and multi-threading support. The interactive features need enhancement with better step-by-step visualization and custom data input capabilities.

Database Improvements

The current schema needs proper normalization with separate tables for contacts, tags, and relationships. I can add flexible metadata storage using JSON for birthdays, social handles, and custom fields. A tags system with many-to-many relationships will improve contact organization. The system needs a modern web interface, so I'll create a Flask-based web application with REST API. Finally, I'll implement an automated database schema migration and versioning system.

C. Enhancement Plans and Skills Demonstrated

Software Design & Engineering Enhancements

I plan to transform the Animal Shelter system into a production-ready application that demonstrates advanced client/server development skills through RESTful API design with proper HTTP status codes and error handling, microservices architecture with service communication, authentication and authorization implementation, and API documentation and versioning.Security implementation will include JWT token-based authentication, role-based access control, input validation and SQL injection prevention, and rate limiting and request throttling. Advanced features will encompass a real-time analytics dashboard using Plotly and Dash, machine learning model for adoption outcome prediction, automated testing with pytest and coverage reporting, and Docker containerization and deployment automation.

Algorithms & Data Structures Enhancements

The enhanced visualizer will showcase advanced data structure implementation including Red-Black tree with O(log n) operations and self-balancing, B-tree implementation for database indexing scenarios, Skip list with probabilistic O(log n) average complexity, and Trie data structure for efficient string operations.Performance analysis will feature real-time Big O notation visualization, memory usage tracking and optimization, algorithm comparison tools with benchmarking, and complexity analysis for different input sizes. Interactive learning features will include custom data generation for algorithm testing, step-by-step execution with pause/resume functionality, performance metrics export and reporting, and educational content integration.