Capstone Reflection and Professional Portfolio

Rafael V. Canseco

SNHU

CS-499 Computer Science Capstone

Introduction

My journey through the Computer Science program at Southern New Hampshire University has allowed me to grow into a confident and well-rounded software developer. Throughout the program, I gained firsthand experience in software engineering, algorithms and data structures, database design, secure coding practices, mobile development, debugging, testing, and full-stack development. Completing the CS-499 Capstone has given me the opportunity to reflect on these skills and present a portfolio that demonstrates my professional abilities in a real-world context. The artifacts I selected showcase not only the technical competencies I developed, but also my capacity to solve problems, work independently, and continuously improve the quality of my work.

Strengths Developed Throughout the Program

Collaboration and Teamwork

Although many projects were individual, collaboration was still essential. I frequently engaged with peers and instructors, particularly when debugging complex code or validating design decisions. Working through discussions and code reviews helped me learn to communicate technical ideas clearly and receive feedback constructively. These experiences strengthened my ability to function as part of a team, respond to critique, and adapt my approach based on new insights—skills that translate directly into industry environments.

Communication With Stakeholders

Throughout the CS program, I learned to tailor communication depending on the audience. Whether writing milestone narratives, presenting design documentation, or creating diagrams for stakeholders, I practiced explaining technical concepts in clear, accessible language. In the capstone, the code review video required me to articulate the current state of the system, justify improvement

decisions, and describe enhancements. This mirrors industry expectations for communicating with managers, clients, and team members who may have varying levels of technical expertise.

Technical Competencies Demonstrated

Software Engineering

The Scene and View Management System from my CS-330 project demonstrates my software engineering growth. I improved the architecture by refactoring classes, strengthening memory management, and reorganizing responsibilities to improve clarity and maintainability. These enhancements show my ability to work with object-oriented design, modularity, abstraction, defensive programming, and clean architectural principles. I also enhanced file structure, error handling, and resource cleanup, which contributes to robust and sustainable code—a key skill for any software engineer.

Algorithms and Data Structures

My second enhancement focused on improving data handling inside SceneManager.cpp and SceneManager.h. I optimized lookup operations and improved control flow by using more efficient data structures. For example, I replaced repetitive condition checks and loosely organized structures with maps and better-structured containers, allowing for faster access times and reduced complexity. This demonstrates my understanding of algorithmic efficiency, Big-O considerations, and refactoring code to make it more scalable and performant.

Database Development

My third artifact enhancement was built around the FitTrack Android mobile application from CS-360. I expanded its Room database implementation by improving the schema design, adding user-

specific queries, implementing Kotlin Flow for real-time data updates, and strengthening the separation of concerns through MVVM. These improvements demonstrate my ability to design relational schemas, write DAO operations, maintain data integrity, and work with modern mobile development practices. The enhancement also reflects my understanding of persistent storage, indexing, querying, and data validation.

Security Practices

Security was integrated throughout the program and reflected in my enhancements. While improving FitTrack's database and logic, I applied secure coding principles such as preventing unintended data access, reducing the exposure of sensitive fields, and structuring the application to limit the risk of misuse. In the C++ project, I improved resource cleanup and defensive checks to avoid undefined behavior or memory misuse that directly relate to secure software development. These experiences contributed to developing a security-focused mindset that anticipates flaws and prioritizes safe design.

How My Artifacts Demonstrate the CS-499 Outcomes

Outcome 1: Employ Strategies for Building Collaborative Environments

Throughout the capstone, I practiced professional-level collaboration by preparing a detailed code review video, documenting enhancement plans, and communicating technical decisions clearly in written narratives. In the workplace, developers must justify design choices, break down complex architectures, and present improvements to peers and managers. My code review mirrors this process by explaining the original architecture of the CS-330 project, identifying weaknesses in SceneManager and ViewManager, and outlining a plan for refactoring.

- I incorporated instructor feedback into each milestone.

- My narratives serve as collaborative documents that explain the intent behind architectural and algorithmic decisions.
- The organization of the GitHub repository—with dedicated folders for original and enhanced artifacts—creates a transparent and collaborative development environment.

These steps simulate real-world teamwork, demonstrating my ability to contribute effectively to multi-developer environments.

Outcome 2: Design and Deliver Professional-Quality Communication

My portfolio demonstrates professional communication through:

- Formal narratives for each enhancement
- A structured README homepage for external audiences
- A code review video breaking down key technical concepts
- Organized GitHub folder structures following UI/UX expectations

For example, while refactoring the SceneManager logic, I created documentation describing how resource ownership, cleanup routines, scene loading, and update loops were redesigned for clarity. When enhancing the FitTrack database, I explained how new queries, Flow-based observers, and MVVM layering improved functionality. These explanations show my ability to communicate advanced concepts to both technical and non-technical stakeholders.

Outcome 3: Design and Evaluate Computing Solutions Using Algorithmic Principles

This outcome is demonstrated most directly in my Algorithms & Data Structures enhancement, where I significantly improved the efficiency and structure of the SceneManager system.

- Replacing repeated conditional branching with hash map lookups, reducing time complexity from O(n) to O(1).

- Improving scene registration and retrieval so that new scenes could be added dynamically without modifying multiple code locations—a more scalable algorithmic design.
- Refactoring class responsibilities to reduce unnecessary coupling and make algorithmic behavior more predictable.
- Improving iteration patterns and reducing redundant operations to optimize render-cycle performance.

The revised SceneManager.cpp and SceneManager.h demonstrate my ability to apply algorithmic reasoning, evaluate inefficiencies, and implement solutions that improve performance at scale.

Outcome 4: Use Techniques, Skills, and Tools in Computing Practices to Create Value

My Software Engineering enhancement illustrates mastery of modern development techniques and tools across multiple languages and frameworks.

- C++ / CS-330 Project
- Application of modular design by separating scene logic, rendering logic, and resource management.
- Incorporation of defensive programming and improved cleanup processes to prevent memory leaks.
- Use of GLFW/OpenGL architecture and C++ object-oriented patterns to create maintainable systems.
- Leveraging Git for version control and structured repository management.

Android / FitTrack App

Use of Room ORM, DAOs, Kotlin Coroutines, and MVVM architecture.

Implementation of reactive data flows using Kotlin Flow.

Structuring business logic away from UI components to create scalable, secure apps.

These enhancements show my ability to use innovative tools, frameworks, and techniques to build professional-quality software systems.

Outcome 5: Develop a Security Mindset to Identify and Mitigate Vulnerabilities

Security considerations were integrated into each artifact enhancement:

- C++ Security Improvements
- Improved pointer managing and object lifetime management to reduce the risk of memory misuse.
- Added input validation and safer resource cleanup to prevent undefined behavior.
- Strengthened class interfaces to prevent inappropriate access to internal data.

Android/Database Security Improvements

Scoped queries to ensure users retrieve only their own data (getWeightsForUser(username)), preventing unauthorized access.

Applied MVVM separation to avoid exposing internal logic to the UI.

Ensured Room schema integrity and type safety to prevent malformed or insecure data writes.

By anticipating potential failures and designing defensive strategies, I developed an initiative-taking security mindset aligned with industry expectations.

Conclusion

Completing the Computer Science program and the CS-499 Capstone has strengthened my confidence as a developer and prepared me to contribute effectively to real-world software engineering

environments. This portfolio showcases my technical growth, my commitment to writing clean and secure code, and my ability to analyze, enhance, and communicate complex systems. I am proud of the progress I have made, and I look forward to applying these skills as I continue advancing in the field of computer science.