Self-Assessment: CS 499 Capstone Experience

Southern New Hampshire University

Matthew Berdecia

Introduction

As I approach the culmination of my Computer Science program at SNHU, I reflect on the skills I have developed, the challenges I have overcome, and the progress I have made in software engineering, algorithms and data structures, and database management. Throughout the CS 499 Capstone course, I have enhanced multiple artifacts, transforming them from academic exercises into professional-quality projects that showcase my technical capabilities. This self-assessment will provide an in-depth review of my growth, achievements, and areas where I can continue improving.

Software Design and Engineering

Artifact: To-Do List Application

Original Version:  
 The To-Do List Application began as a simple command-line program that allowed users to add, remove, and display tasks. It had no error handling, no user interface, and no persistent storage, limiting its real-world usability.

Enhancements Implemented:

1. Refactoring for Maintainability: Improved code readability by breaking down logic into modular methods and enhancing variable naming conventions.
2. Graphical User Interface (GUI) Development: Converted the console-based system into a JavaFX GUI application, making it more interactive and user-friendly.
3. Data Persistence with SQLite: Introduced SQLite integration to store tasks permanently, replacing the in-memory ArrayList storage.
4. File-Based Storage Alternative: Implemented text file storage as a lightweight persistence option.
5. Security Features: Introduced basic input validation and exception handling to prevent application crashes.

Skills Demonstrated:

* Software development lifecycle principles (CS 250)
* Object-oriented programming (OOP) best practices
* GUI development using JavaFX
* Database integration (SQLite)
* Exception handling and debugging

Reflections and Challenges:  
 One of the biggest challenges was integrating SQLite with the GUI while ensuring smooth performance. Additionally, learning JavaFX layout design required research and testing to create a responsive and user-friendly interface. The process taught me the importance of separating UI logic from business logic and using MVC (Model-View-Controller) principles for better maintainability.

Algorithms and Data Structures

Artifact: Sorting Algorithms Comparison

Original Version:  
 The original project implemented Bubble Sort, Quick Sort, and Merge Sort to compare their performance. However, it lacked advanced sorting methods and visual feedback for understanding algorithm efficiency.

Enhancements Implemented:

1. Added New Sorting Algorithms: Implemented Radix Sort and Heap Sort to provide a more comprehensive comparison.
2. Performance Optimization: Used efficient data structures to improve sorting speeds and reduce memory consumption.
3. Sorting Visualization: Developed a JavaFX-based visualization tool to animate sorting processes, making it easier to understand sorting mechanisms.
4. Unified Benchmarking System: Standardized execution time measurement for all algorithms, improving accuracy and consistency in comparisons.

Skills Demonstrated:

* Data structures and algorithms (CS 260)
* Performance benchmarking and efficiency analysis
* Graphical visualization of computational processes
* Big O notation analysis

Reflections and Challenges:  
 One of the main challenges was optimizing Radix Sort for larger datasets while ensuring it maintained linear time complexity. Additionally, creating an intuitive sorting visualization required deep knowledge of JavaFX animations. This experience strengthened my problem-solving skills and ability to analyze computational trade-offs in algorithm selection.

Database Management

Artifact: Library Management System

Original Version:  
 The Library Management System initially used MySQL for storing book, user, and transaction records. It operated through SQL queries in Java, requiring manual data management and having limited scalability.

Enhancements Implemented:

1. Migrated from MySQL to MongoDB: Transitioned from relational (SQL) to NoSQL, improving scalability and flexibility.
2. Developed a Web-Based Interface: Created a REST API using Java and a frontend web app to allow user interaction through a modern interface.
3. Implemented JWT Authentication: Introduced JSON Web Token (JWT)-based authentication to secure user sessions and access controls.
4. Optimized Query Performance: Used MongoDB indexing and aggregation pipelines to enhance database response times.

Skills Demonstrated:

* Database design and management (IT 340)
* SQL-to-NoSQL migration strategies
* Web-based database integration
* Authentication and security best practices

Reflections and Challenges:  
 Migrating relational data to MongoDB was challenging, especially converting SQL joins into efficient NoSQL queries. Additionally, implementing user authentication required a deeper understanding of token-based security models. These enhancements strengthened my ability to develop secure, scalable database solutions and apply real-world data management techniques.

Emerging and Disruptive Technologies

Quantum Computing and AI in Cybersecurity

Throughout the course, I explored emerging trends such as Quantum Computing and AI in Cybersecurity. These technologies are redefining:

* Data encryption and security protocols
* Threat detection and automated incident response
* Real-time computation efficiency

Career Relevance:  
 Since I am interested in full-stack development and database security, learning post-quantum cryptography and AI-powered security frameworks will be crucial for building secure, scalable software systems.

Course Outcomes Achieved and Remaining Areas of Improvement

Course Outcomes Achieved:

Software Design and Engineering:

* Developed a GUI-based application with database integration.
* Applied best practices in code refactoring and maintainability.

Algorithms and Data Structures:

* Implemented and optimized multiple sorting algorithms.
* Developed an interactive sorting visualization tool.

Database Management:

* Migrated a relational database (MySQL) to NoSQL (MongoDB).
* Improved query performance and security in a web-based system.

Security Mindset:

* Implemented JWT authentication and database security best practices.
* Explored AI-driven cybersecurity as an emerging field of interest.

Future Goals and Areas for Improvement

Despite my progress, there are still areas where I can improve:

1. Advanced Security Measures:

* Implement encryption techniques such as AES or RSA for data protection.
* Learn penetration testing and ethical hacking tools to strengthen security awareness.

2. Cloud Computing Integration:

* Deploy projects to AWS or Google Cloud for real-world scalability testing.
* Implement serverless functions (AWS Lambda, Firebase Functions) to improve performance.

3. Machine Learning for Data Processing:

* Integrate ML models for predictive analytics in database management.
* Explore AI-driven automation in cybersecurity threat detection.

Final Thoughts

This capstone experience has significantly enhanced my technical skills, problem-solving abilities, and software development expertise. By refining my ePortfolio artifacts, I have successfully demonstrated proficiency in software engineering, algorithms, and database management.

Moving forward, I aim to continue learning cloud computing, security best practices, and AI-driven solutions, ensuring I stay ahead in the rapidly evolving field of computer science.

This self-assessment serves as a testament to my growth and readiness to enter the professional field, confident in my ability to develop, secure, and optimize complex software systems.