Southern New Hampshire University

CS 499 Module One Assignment – Andy Martinez

Complete this template by replacing the bracketed text with the relevant information.

I. **Self-Introduction:** Address all of the following questions to introduce yourself.

A. How long have you been in the Computer Science program?
I have been in the CS program since 2018, I took an extended break to focus my efforts on finding a career and establishing myself within the industry around civil engineering and computer science

B. What have you learned while in the program? List three of the most important concepts or skills you have learned.
Three of the most important concepts I have learned while in the program are Algorithms and Data Structures, Computer Organization, and Programming Paradigms.

Together, these areas teach how to choose and implement the right data structures and algorithms, understand how code maps onto real hardware, and select the most effective programming style for a given problem, laying the groundwork for writing efficient, reliable, and maintainable software.

C. Discuss the specific skills you aim to demonstrate through your enhancements to reach each of the course outcomes.
I will demonstrate mastery of algorithms and data structures by annotating every implementation with worst-, average-, and best-case complexity, providing both custom and library-based data structures, and supplying unit tests along with performance benchmarks that prove the impact of any optimizations. I will show my understanding of computer organization and systems by using CPU profilers or cache simulators to pinpoint hardware-level bottlenecks, refactoring code for better data locality or pipelining, and introducing a correctly synchronized multithreaded version to illustrate my grasp of memory hierarchies and OS resource management. Finally, I will showcase fluency in programming paradigms by offering imperative and declarative or functional solutions to the same problem, refactoring monolithic code into well-designed classes complete with diagrams, rewriting key modules in a functional style using pure functions and immutable data, and explaining in my documentation why each paradigm is the best fit.

D. How do the specific skills you will demonstrate align with your career plans related to your degree?
I see my civil-engineering career benefiting directly from the skills I'll hone in my computer-science coursework. When I annotate and benchmark algorithms and data structures, I'm building the analytical rigor I need to optimize structural-analysis routines, manage large datasets of survey or sensor data efficiently, and ensure my custom scripts run predictably on project-scale inputs. By delving into computer organization and systems, I'll learn how low-level performance bottlenecks emerge, knowledge I can apply when I develop or troubleshoot data-acquisition firmware for monitoring soil movement, bridge stresses, or environmental conditions, ensuring that real-time readings aren't lost to cache misses or improper threading.

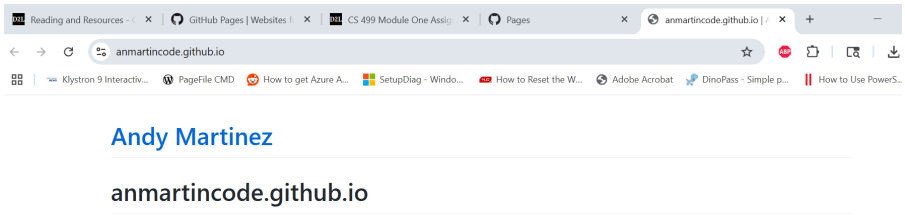
E. How does this contribute to the specialization you are targeting for your career?
By honing my algorithm and data-structure skills, I'll be able to design and optimize the core routines that drive automation pipelines, whether that's parsing massive sensor logs, scheduling recurring build or deployment jobs, or orchestrating robotic workflows on the construction site. My understanding of computer organization and systems ensures those routines run predictably and efficiently on the actual hardware, embedded controllers, GPU-accelerated 3D modeling workstations, or cloud-based inference servers, so I can push updates, run simulations, or retrain models without unexpected slowdowns. And because I'm fluent in multiple programming paradigms, I can pick the right style for each domain: using object-oriented frameworks to build modular plugins for 3D CAD tools, functional pipelines to transform and validate mesh or point-cloud data immutably, and declarative or domain-specific languages to define and evolve AI training configurations. Together, these capabilities will let me automate complex engineering workflows end-to-end, create high-performance 3D design tools, and integrate AI-driven analytics and decision-making into every stage of my civil-engineering projects, precisely the blend of automation, 3D design, and AI expertise I'm aiming to master.

II. ePortfolio Set Up:

A. Submit a **screen capture** of your ePortfolio GitHub Pages home page that clearly shows your URL.
   i. You already have a repository in GitHub where you uploaded projects in previous courses. Your ePortfolio will reside in GitHub but can link to work at other sites, such as Bitbucket.

B. Use the GitHub Pages link in the Resource section for directions on:
   i. How to create your GitHub website and publish code to GitHub Pages
   ii. Issues, such as adding links to other sites

C. Paste a screenshot of your GitHub Pages home page with your URL clearly showing in the space below.



Southern New Hampshire University

III. Enhancement Plan:

A. **Category One:** Software Engineering and Design

i. **Select an artifact** that is **aligned with the** software engineering and design **category** and explain its origin. Submit a file containing the code for the artifact you choose with your enhancement plan.

- **I will be choosing an artifact from the course (** CS 340: Advanced

Module Seven.zip

Programming Concepts) the file will be

Note: Your artifact may be work from the following courses:

- IT 145: Foundation in Application Development
- CS 250: Software Development Lifecycle
- CS 260: Data Structures and Algorithms
- IT 315: Object Oriented Analysis and Design
- CS 320: Software Testing, Automation, and Quality Assurance
- CS 330: Computational Graphics and Visualization
- CS 340: Advanced Programming Concepts
- CS 350: Emerging Systems Architectures and Technologies
- CS 360: Mobile Architecture and Programming
- IT 365: Operating Environments
- IT 380: Cybersecurity and Information Assurance
- CS 405: Secure Coding
- CS 410: Reverse Software engineering
- IT 340: Network and Telecommunication Management
- IT 380: Cybersecurity and Information Assurance

ii. **Describe** a practical, well-illustrated **plan** for enhancement in alignment with the category, including a pseudocode or flowchart that illustrates the planned enhancement.

In the current create / read / update / delete methods, invalid inputs (missing required fields, wrong types) will either silently fail or throw generic exceptions, and there's no record of what happened. By adding:

- **Schema validation (e.g. via Cerberus or JSON Schema)**
- **Structured logging (via Python's logging module)**

we ensure that:

- **Bad data is caught *before* it hits MongoDB**
- **All attempts—successful or not—are logged with enough context to debug later**

This aligns with the broader category of robustness and observability.

For this category of enhancement, consider improving a piece of software, transferring a project into a different language, reverse engineering a piece of software for a different operating system, or expanding a project's complexity. These are just recommendations. Consider being creative and proposing an alternative enhancement to your instructor.

Think about what additions to include to complete the enhancement criteria in this category. Since one example option is to port to a new language, that is the kind of scale that is expected. This does not mean you need to port to a new language but instead have an equivalent scale of enhancement. Underlying expectations of any enhancement include fixing errors, debugging, and cleaning up comments, but these are not enhancements themselves.

iii. Explain how the planned enhancement will **demonstrate** specific **skills** and align with course outcomes.

a. Identify and describe the specific skills you will demonstrate that align with the course outcome.

- **Cross-language translation: Interpreting Python idioms into Go's static-typed constructs**
- **Concurrency design: Employing goroutines and channels for parallel database operations**
- **Context management: Using context.Context to enforce timeouts and cancellations**
- **Structured logging: Integrating a production-grade logger (e.g. logrus)**
- **Robust error handling: Returning and checking error values at each call site**
- **Automated testing: Writing unit tests with Go's testing package and table-driven test patterns Module/version control: Managing dependencies with go.mod and semantic versioning**

b. Select one or more of the course outcomes below that your enhancement will align with.

- **CO1 – Programming Proficiency: Demonstrates ability to design and implement non-trivial software in multiple languages.**
- **CO2 – Software Design & Architecture: Applies best practices for structured logging, error handling, and concurrency.**
- **CO3 – Testing & Debugging: Develops and executes automated unit tests and handles compile-time & runtime errors.**
- **CO4 – Cross-Platform Portability: Shows understanding of software portability by adapting a Python module to Go.**

Course Outcomes:

1. Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
2. Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
3. Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.
4. Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
5. Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.

B. **Category Two:** Algorithms and Data Structures

i. **Select an artifact** that is **aligned with the** algorithms and data structures **category** and explain its origin. Submit a file containing the code for the artifact you choose with your enhancement plan. You may choose work from the courses listed under Category One.

For this artifact, I will create a self project that is under the algorithms and data structures category, as I do not have access to previous courses. The origin will be "Flask - AlgoViz"

Flask-AlgoViz will let users:

1. **Choose a data structure (e.g. red-black tree, AVL tree, graph, heap) or algorithm (e.g. Dijkstra, mergesort).**
2. **Step through its operations (insertion, deletion, search, relaxations, recursive calls, etc.).**
3. **See a live, animated visualization rendered purely in HTML/SVG and styled by CSS.**

All core logic lives in Python (Flask routes + algorithm implementations), while the front end is built with HTML templates and CSS for layout, theming, and simple transitions.

ii. **Describe** a practical, well-illustrated **plan** for enhancement in alignment with the category, including a pseudocode or flowchart that illustrates the planned enhancement.

Goal: Expand the catalog of visualized structures and ensure robust, reusable APIs.
Deliverables:

- **New Data Structures**
  - **AVL Tree**
  - **Min-/Max-Heap**
  - **Graph (adjacency-list basis)**

- **Unified .step() Interface**

Goal: Make the learning experience more intuitive, interactive, and visually rich.
Key Features:

1. **SVG-Based Canvas**
    - **Automatic layout (e.g. Reingold–Tilford for trees)**
    - **Smooth CSS transitions (transform, opacity)**
2. **Operation Highlights**
    - **Flash nodes/edges in different colors on each step**
    - **Sidebar "Operation Log" with clickable entries to replay steps**
3. **Responsive Design**
    - **Flex/Grid to adapt canvas and controls on mobile/tablet**
    - **Dark-mode via CSS custom properties**

Goal: Empower users to measure performance trade-offs between algorithms.
Components:

- **Benchmark Runner**
    - **Batch runs for random/ordered/worst-case inputs**
    - **Collects: run time, operation counts, tree height, memory footprint**
- **Results Dashboard**
    - **Matplotlib-generated plots (via on-server scripts) exposed as PNG/JSON**
    - **In-page charts (e.g. Chart.js) comparing:**
        - **Red-Black vs. AVL insertion times**
        - **Heapify vs. sequential insert**
        - **Dijkstra vs. Bellman-Ford on same graph**
- **Export Function**
    - **CSV download of raw metrics**
    - **"Shareable link" encoding parameters for reproducibility**

Goal: Transform Flask-AlgoViz into a social learning platform.

1. **Interactive Tutorials**
    - **Guided, step-by-step walkthroughs embedded in Jinja templates**
    - **Quiz questions after key operations (e.g. "What's the new black-height?")**
2. **User-Submitted Sequences**
    - **Allow learners to save & share their own operation scripts**
    - **"Load from URL" to replay others' examples**
3. **Discussion & Annotation**
    - **Inline comments on specific steps (e.g. why we rotate here)**
    - **Upvote/downvote examples, brief leaderboards**
4. **Deployment & CI/CD**
    - **Dockerize entire stack**
    - **GitHub Actions: run tests, lint, build docs, deploy to Heroku or Netlify + Flask backend**

For this category of enhancement, consider improving the efficiency of a project or expanding the complexity of the use of data structures and algorithms for your artifact. These are just recommendations. Consider being creative and proposing an alternative enhancement to your instructor. Note: You only need to choose one type of enhancement per category.

Think about what additions to include to complete the enhancement criteria in this category. Since one example option is to port to a new language, that is the kind of scale that is expected. Perhaps you might increase the efficiency and time complexity of an algorithm in an application and detail the logic of the increased time complexity. Remember, you do not need to port to a new language but instead have an equivalent scale of enhancement. Underlying expectations of any enhancement include fixing errors, debugging, and cleaning up comments, but these are not enhancements themselves.

iii. Explain how the planned enhancement will **demonstrate** specific **skills** and align with course outcomes.

a. Identify and describe the specific skills you will demonstrate to align with the course outcome.

- **Algorithm Design & Implementation**
- **What:** Extending and refactoring the .step() interface for multiple balanced trees (e.g. red-black, AVL) and graph algorithms (e.g. Dijkstra, Bellman–Ford).
- **Demonstrates:** Ability to translate theoretical invariants and pseudocode into correct, maintainable Python modules.

- **Complexity Analysis & Benchmarking**
- **What:** Building the benchmark runner that measures runtimes, operation counts, and memory footprints under various workloads.
- **Demonstrates:** Skill in designing experiments, collecting performance data, and interpreting big-O versus empirical results.

- **Full-Stack Web Development**
- **What:** Integrating Python/Flask back-end APIs with an HTML/SVG front-end, styled via CSS Grid/Flexbox and animated with CSS transitions.
- **Demonstrates:** Proficiency in RESTful API design, Jinja templating, responsive layouts, and separation of concerns between logic and presentation.

b. Select one or more of the course outcomes listed under Category One that your enhancement will align with.

- CS 260: Data Structures and Algorithms
- CS 250: Software Development Lifecycle

C. **Category Three: Databases**

i. **Select an artifact** that is **aligned with the** databases **category** and explain its origin. Submit a file containing the code for the artifact you choose with your enhancement plan. You may choose work from the courses listed under Category One.

For this artifact, I will choose to create a project since I did not have access to previous courses. I will build a simple CLI app that reads a JSON file of "contacts," loads them into an SQLite database, and lets you look up, add, and delete contacts.

ii. **Describe** a practical, well-illustrated **plan** for enhancement in alignment with the category, including a pseudocode or flowchart that illustrates the planned enhancement.

- **Normalize & Extend the Schema**
- **Add an extra JSON column to store arbitrary metadata (birthdays, social handles, etc.)**
- **Introduce a tags table and a many-to-many join table to manage contact labels**
- **Leverage SQLite's JSON1 Extension**
- **Ingest JSON records that include both core fields and nested metadata**
- **Create indexes on key JSON paths (e.g., birthdays) and query those paths directly**
- **Build "Smart" CLI Commands**
- **Enable lookups by tag via JOINs between contacts and tags**
- **Support updating individual JSON fields (like adding a birthday) in place**
- **Add a Tiny Web UI (Optional)**
- **Backend with Flask (or similar) exposing REST endpoints for list, create, update**
- **Front-end that displays a paginated contact table and supports "advanced" text filters**
- **Automate & Harden**
- **Introduce a migration tool (e.g. Alembic or simple version table) for schema changes**
- **Schedule regular backups of the database file**
- **Implement validation at ingest time (email format, unique constraints on tags/emails)**
- **Why These Enhancements Matter**
- **Normalized schema prevents duplication and scales as you grow features**
- **JSON flexibility lets you add new metadata without constantly altering tables**
- **CLI + Web UI gives you practice across scripting, database design, and lightweight app development**

For this category of enhancement, consider adding more advanced concepts of MySQL, incorporating data mining, creating a MongoDB interface with HTML/JavaScript, or building a full stack with a different programming language for your artifact. These are just recommendations; consider being creative and proposing an alternative enhancement to your instructor. Note: You only need to choose one type of enhancement per category.

Think about what additions to include to complete the enhancement criteria in this category. Since one example option is to port to a new language, that is the kind of scale that is expected. Perhaps you might increase the efficiency and time complexity of an algorithm in an application and detail the logic of the increased time complexity. Remember, you do not need to port to a new language but instead have an equivalent scale of enhancement. Underlying expectations of any enhancement include fixing errors, debugging, and cleaning up comments, but these are not enhancements

themselves.

iii. Explain how the planned enhancement will **demonstrate** specific **skills** and align with course outcomes.

a. Identify and describe the specific skills you will demonstrate that align with the course outcome.

- **Python JSON Handling**
  Parse, validate, and serialize JSON data using Python's standard json module, demonstrating the ability to ingest semi-structured data and prepare it for storage.
- **Relational Schema Design & Normalization**
  Design a normalized SQLite schema including core entity tables, many-to-many join tables (for tags), and an extra JSON column—showing mastery of data modeling principles.
- **SQL CRUD & JSON Path Queries**
  Write SQL statements to create tables, insert/update/delete records, perform joins for tag-aware lookups, and query nested JSON fields via SQLite's JSON1 extension.

b. Select one or more of the course outcomes listed under Category One that your enhancement will align with.

- CS 405: Secure Coding
- CS 410: Reverse Software engineering

IV. ePortfolio Overall Skill Set

A. Accurately describe the **skill set** to be illustrated by the **ePortfolio overall**.

i. Skills and outcomes planned to be illustrated in the code review

- **Critically evaluate code for adherence to SOLID and DRY principles**
- **Identify and suggest refactorings to improve modularity and maintainability**
- **Verify correct implementation of data structures and algorithmic logic**
- **Assess database access layers for efficient query construction and proper use of transactions**
- **Review test suites for coverage, clarity, and meaningful assertions**

ii. Skills and outcomes planned to be illustrated in the narratives

- **Reflect on the rationale behind architectural and design-pattern decisions**
- **Describe the process of selecting and tuning algorithms/data structures for performance and scalability**

- **Explain challenges encountered in modeling data (both relational and JSON-driven) and how they were overcome**
- **Illustrate how iterative testing and code review cycles improved code quality**
- **Tie technical choices back to project requirements and end-user needs**

iii. Skills and outcomes planned to be illustrated in the professional self-assessment

- **Self-evaluate growth in applying software engineering best practices (testing, CI/CD, version control)**
- **Measure improvements in algorithmic problem-solving speed and accuracy**
- **Assess proficiency gained in designing normalized schemas and writing performant SQL/JSON queries**
- **Identify areas for continued learning (e.g., advanced database features, new design patterns, emerging algorithms)**
- **Set concrete goals for leveraging these skills in future projects or professional roles**