Milestone Two Narrative

CS-499

Southern New Hampshire University

Steve Edmund

**Artifact Description:**

An essential aspect of Software Engineering is designing code that is not only easily understandable and scalable but also ensures functionalities are implemented such that users don't encounter problems during the application's use. This artifact encompasses code restructuring to eliminate redundant blocks and designing functions that execute the same functionality, thereby enhancing code structure and the reusability of code segments. Additionally, anticipating user behavior is crucial for developing a robust application. As such, error and exception handling have been incorporated. Unit tests have been crafted to ensure functions operate as expected. A significant emphasis of this artifact has also been on code maintainability, achieved by adhering to comprehensive commenting practices, facilitating future developers in understanding and extending the code.

**Inclusion of Artifact in E-Portfolio:**

My decision to incorporate this artifact into my E-Portfolio is rooted in my commitment to enhancing coding proficiency and crafting modular, clean code. As Software Engineers, we often collaborate in teams. Ensuring our code is scalable, maintainable, and structured to let other developers engage with it without significant overhead is paramount. Should I revisit this code in the future, well-structured and adequately commented code will streamline my workflow, minimizing time spent deciphering previous work.

Proper commenting conventions and eschewing repetitive code segments are essential to crafting pristine, scalable, and maintainable code. By refining the commentary, I have ensured each code block is elucidated, streamlining future modifications. Designing specialized functions and eradicating redundant code segments fosters future scalability and reduces complexity. Embracing software engineering best practices, every code segment underwent testing before further development. Unit tests for each function were formulated to guarantee seamless development, complemented by error and exception handling and maintaining logs to identify and rectify potential issues.

**Meeting Course Objectives:**

As highlighted in the code review, the enhancements designated for the second milestone have been accomplished. Several discrepancies were noted in the initial application: subpar commenting practices, inadequate error and exception handling, and recurring code sections. These issues were duly addressed, and the artifact was refined to meet these standards, informed by the curriculum of the following courses:

1. IT 145: Foundation in Application Development

2. CS 250: Software Development Lifecycle

3. CS 320: Software Testing, Automation, and Quality Assurance

These courses' techniques were instrumental in fostering clean, maintainable code with proficient error-handling strategies.

**Enhancing the Artifact:**

Augmenting the artifact was a voyage through the art of writing clean, maintainable code, and embracing a modular approach. The exercise of pruning repetitive code and crafting reusable functions was an application in the spectrum of coding methodologies, from rudimentary to advanced, acquired throughout my coursework. Furthermore, imbibing the ethos of robust code structuring, emphasizing scalability and maintainability, accentuated the nuances of professional product development. Integrating comprehensive error and exception handling illuminated the significance of diagnosing and addressing code anomalies, demanding a deep dive into user-GUI interaction studies and rigorous testing. The enhancement phase also underscored the utility of application logs in troubleshooting and crash diagnostics.

The development of unit tests for every function underpinned the quintessential role of software testing, ensuring a frictionless developmental trajectory. As codebases grow, troubleshooting can become a labyrinthine endeavor. Thus, a robust testing environment is pivotal to preempting unforeseen glitches and rogue code. Further, adhering to commenting conventions proved invaluable, enabling a seamless navigation of the evolving codebase without redundant recall of past tasks.

This developmental journey underscored the paramountcy of features such as error and exception handling, logging, and diligent code commentary in the software development lifecycle. The hands-on experience crafting maintainable, scalable code accentuated the imperativeness of mastering these techniques in software development.