Gianmarco Vendramin

Narrative - Enhancement One: Software design and engineering.

The artifact chosen for this portfolio comes from the course CS-300 "Data Structures and Algorithms Analysis and Design," it was created in April 2024 and is a software to manage the university courses and prerequisites to be used as a helpful tool for a student advisor to register the student to the various courses. This artifact was chosen as it has been evaluated as a perfect starting point to apply numerous skills to bring it to a more realistic product that can be potentially employed in a production environment. The first improvement applied is to transition it to a more "modern" language like Python and use its characteristics to create a safer product. It has given me the occasion to show coding skills, especially in clean and well-written code that is easily understandable by applying the Phyton programming style guide. Another skill showcased is the ability to transition to a different data structure paradigm from standard C++ vectors to more modern lists and sets; moreover, risky pointers have been moved to safer objects.

Transitioning to a new language improved the artifact in various ways; for example, moving to a language with an automatic garbage collection mechanism intrinsically improves its safety, mainly when pointers are used like the ones on the nodes of the binary search tree. Another example of beneficial transition is the overall increased readability, which can be achieved by dividing the project into sub-sections of libraries like models, services, and utils. Dynamic typing is another improvement that comes free of charge from the Python language; however, if overused and not correctly understood, it could also generate problems and bugs. All of these advantages come with a cost, and it is the higher code execution time compared to C++. Still, if this drawback is correctly weighed, considering the final application requirements and the impact on usability, Python could be a good choice.

The first planned outcomes were to design, develop, and deliver a professional-quality product, and even a bit overkill by adding detailed comments to each part of the code so everyone

who reviews the code will gain an understanding of its logic immediately. Another outcome met is the design and evaluation of computing solutions that solve a problem or improvement. It was achieved by rewriting everything and optimizing the code by observing algorithmic principles, computer science practices, and standards. Finally, a security mindset was developed by paying greater attention to string sanitization while loading the course data from the CSV file. Another section where security improvement has been applied is where file opening and reading have taken place; here, every effort has been made to avoid any unwanted application "dirty" crash due to unexistent or corrupted files by using multiple try/catch blocks to avoid leaving unhandled exceptions.

While enhancing the artifact, I've learned how adding comments to uncommented parts of code can let you better reflect on the algorithm or process used to achieve a solution and consider different paths that could be more efficient and better aligned to security. Another personal improvement is becoming more used to modern data structures like lists and sets and understanding how helpful it can be to use the module's methods dedicated to these new datatypes, mainly to avoid design flaws leveraging the language automation already in these structures. During this journey, one of the biggest challenges faced was to adapt and being able to understand some new constructs for cycles that traverse lists of objects that are perfect for readability but sometimes hide the working principle by compressing the instructions in a single line of code, for example, when loading all the course numbers in a set with the row "`all_course_numbers = {crs.number for crs in courses}`", when every single line of the CSV file is analyzed with a for a statement like: "`for row in reader:`" or extracting the prerequisites with "`prerequisites=[field.strip() for field in row[2:] if field.strip()]`". It was not easy to correctly balance the code readability with the Python "dialect," where you can compress multiple operations on a single row that can become a bit cryptic for those unfamiliar with this language. However, efforts have been applied to avoid too complex single statement rows and to keep the focus on readability instead of compressing the code.