**Gianmarco Vendramin**

# Narrative - Enhancement Three: Databases.

The milestone four artifact continues with the path of enhancing a single project to take it to a higher professional level. The software name is "ABCU_Advising_Program," and it is the evolution of the final project of the course CS-300, "Data Structures and Algorithms Analysis and Design," developed in April 2024. The project was centered around the development of a tool to manage the university courses catalog and relative prerequisites. This artifact has been selected as the perfect fit for a step-by-step growing project that will be taken to an almost production level while traversing all the milestones. By keeping the final target in mind, each milestone enhancement category perfectly aligns with what it was missing from the initial project.

Bringing this artifact at a higher complexity level has showcased the ability to develop a professional-quality product by applying all the coding best practices learned like good commenting, attention to readability, lower complexity by modularizations, using object-oriented programming, and choosing the right tradeoff between code complexity and overall efficiency. For example, each file of the project has an initial description block to identify what it is willing to perform, each function is well described in its intentions, the number of activities performed by each function has been limited, and every critical dataset is used, instantiating objects from classes.

This milestone improvement was for sure the most significant endeavor till now. Bringing everything to a MongoDB database while implementing strict user security was harder than expected. First of all, all the data was moved to a MongoDB database (abcu_advising) collection named "courses" by creating the "mdb_api.py" module to develop all the needed functions to perform the same activities done before with the Binary Search Tree structure as CURD operations. This library, for example, opens the secure connection to the database and saves it into an abject, which is a type of library that comes from the "pymongo" module available in Phyton. Once this object is alive, it becomes the key to access the database by using the collections objects methods like "insert_one," "find_one," "find," "sort," and "delete_many," always poping the "_id" field that was not necessary for the application. So when the data was loaded from the JSON/CSV file, it was saved directly into the database "courses" collection. Once done, it was tested by updating the "main.py" source to use these functions instead of the BST ones. The second and most crucial step for data security was protecting access to the MongoDB database by reconfiguring the local instance to require authentification and creating a new user in the admin database. This step has to be reflected in the Phyton code by changing the database access by adding a username and password. This step was performed following the security best practice by avoiding hardcoding the username and password in the code but instead by using an environment variable and excluding the ".env," (which contains the sensitive username and password) from Git pushes by excluding it with the ".gitignore" configuration.

The second improvement step was adding the users to the application and managing the registration, authentication, and authorization. Here, a new user class has been created with some methods such as creating it, checking the password given, and, if a student, seeing if he can take a course with his prerequisites based on the currently completed courses. One natural improvement was to store also the user in the database by creating a specific collection. Application safety has been taken to another level by storing the password in the database as encrypted and hashing it with the "bycript" library. So, even if the database is corrupted, no one can reverse the hashed value to the original password.

The new users collection addition has permitted the addition of user authentication to the application from the beginning. It distinguishes two different roles (student and admin) where the student has limited access to the application functions (authorization), and the menu is printed according to this limitation. This last step contributed to creating a complete and secure application that follows safety best practices and is almost ready to move the database to the cloud without fear.

Lastly, the application was improved by adding useful functions that use the power of the data moved to the database, like adding and printing the completed course of a student, checking if the student can take a course based on the prerequisites, creating new users, and list current active users. This final addition aligned the milestone improvement to the enhancements listed in Module One.

This module's enhancing activity has made me more acquainted with databases, specifically NoSQL databases like MongoDB. It forced me to enable authentication to it with the outcome of learning how you perform maintenance in a MongoDB server. It was also the most challenging part as it was something I had never done before in a MongoDB installation under Windows OS, and most of the instructions you found were for Linux systems. The second challenge was understanding and correctly storing passwords inside the database, avoiding plain text, and storing the hashed version for later comparison during the initial login. This final enhancement has also trained me to use Python lists data structures, especially how to traverse them correctly, which were used extensively in the project.