John Shumway

Narratives

**Part 1: Software Design**

The Software Design is composed of two main artifacts. The CRUD framework and unit test cases. The CRUD framework was previously created in Python for a different database project around May 2021. The CRUD framework is a system of functions that perform the basic queries to the database. Next is the unit test cases, which was done in an SNHU course around October 2020 . These test cases were something I enjoyed writing, and found very useful to make sure new changes didn’t break any core features. Unit test cases run basic functions using fixed variables and compare the output to a static value. If the values are different, then the user is warned and prompted to review the code.

I chose the CRUD framework, as I found that it both showcased how well python interacts with databases, and to show how easy it is to use the framework. Being able to take what existed in another project and improve upon it is another challenge I wanted to take on. With the Unit Tests, I found that the tests were very helpful as a concept to make sure that both your code meets a basic level of completeness and that any new changes didn’t break anything. When making changes in CRUDs, you want to be sure that the operations can still perform successfully.

When I started designing the overall framework and workspace for the project, I realized I would need a good system to store the Python files. The idea came to me when building the test cases, that I would structure the project in a similar way. When building the test cases, you generally want the scripts for testing in it’s own folder. This way it’s easier to locate and modify without touching the main code. To mimic this, I built each data topic (such as students’ GPA, Teacher ratings, ect) as it’s own ‘module’ and all the modules are put into its own category. This makes the code easy to read and find the data modules, and allows the project to easily be scaled up with new data categories.

Using a previous Python project from SNHU, I was able to build the CRUD with ease. The project used MongoDB instead of mySQL, so most of the work was researching and building the functions to use the new database repository instead. During this work, I was able to find good practices in importing and using 3-rd party tools for Python.

**Part 2:** **Algorithm and Data Structures**

For the Algorithm and Data Structures I choose to use the matplotlib for creating graphs. The matplotlib tool uses lists and dictionaries to gather the data and build the graphs. The tool contains multiple functions to customize the graph as needed to make it more presentable. Matplotlib was originally built in 2003 by John D. Hunter, and is still actively developed today. While I had experience with most data structures in the past, most of the code and examples were based on an SNHU project focused on them back in September of 2019. Originally written in C++, I researched the same concepts in Python to confirm the structure types existed, and held the same standards. Fortunately, Python still keeps the same ideas and the concepts from the previous course can be applied to the project.

Most of the Algorithm and Data structures for this project rely on organizing the data before sending it to matplotlib, and storing the data into the most appropriate structure. While the list structure is basic enough to use, I found it to be the best fit. Firstly it’s easy to set up and modify lists, secondly lists are easy to convert into dictionaries as needed for graphing, and finally unlike hash tables or binary trees, lists use the least amount of memory. While Hash tables and Binary trees have far more features, those features revolve around searching or sorting the data. Since the information from CRUD should contain the data we want in the first place, searching/sorting is not necessary. This artifact also shows how well I can adapt to using unique tools and format data into something usable.

The process of using matplotlib and incorporating the appropriate data structure was filled with trial and error. Since this was the first time I used the 3rd party tool, incorporating the tool with the database was where most of the work took place. Fortunately, the documentation on matplotlib was very helpful, and easy to use. Otherwise, most of the data structures used in this project are similar across languages so it was easy to write the code. These structures is just one example of why Python is a popular code with many programmers

**Part 3:** **Database**

The database artifact is the creation of a relational database structure using MySQL. MySQL is a well established and lightweight database service perfect for small projects. MySQL was first released in May 1995 and later bought by Sun Microsystems in 2010 and is still receiving updates today. Compared to the popular Microsoft SQL performance between the two are pretty similar in performance, though MySQL doesn’t come with many of the advanced features of MSSQL. That being said, MySQL is open-source and you can use the full tools as soon as you get the program which is perfect for this project. Additionally, the project’s main features are about reading and storing data into the database (from CRUD) so those advanced features are not a necessity.

This database artifact is here to show how I am able to take a subset of data then create and store it into a fresh database. From there, being able to interact and use the database efficiently with best practices. I didn’t just jump into the database creation, I sat down and planned out a chart to show the table relationships and the data types to store. In addition to my experience with Microsoft SQL and SQLite, The artifact shows that I have a good understanding of how to use and work with the foundations for the popular databases.

Working with MySQL was an enjoyable experience. Using the workbench to build the databases was much easier than using a shell window. The workbench tool was familiar enough to Microsoft’s SQL server to recognize and use the main features without trouble. Yet since the workbench was not as feature robust as MSSQL, there were some other tasks I had to work around. One of them was related to connection use. There was an issue where the connection from my data project was still connected but MySQL didn’t note it was, and trying to work on the database while the connection was active caused some problems. It taught me to build a system to make sure the connection was closed after the Python program was finished to prevent this kind of issue from happening again.