Milestone Four – Databases

Christian Bass

SNHU

Artifact:

This artifact is from “IT 145: Foundation in Application Development”. It is a basic stock system that accepts input from the user on how many apples and oranges they have, how many they should have total, and then does the calculation for how many they need to buy. This is then outputted to the user. This was created a few years ago when I initially took the course.

I feel this artifact is a perfect candidate for the addition of a database. It is already functionally doing what a database should do but far simpler. Having the ability to add more fruits, save the data externally so it persists across multiple executions of the software, and process data in greater quantities would make the software far more useful. I plan to showcase my ability to create a database and implement CRUD functionality to access and edit it within my code. I also will be refactoring from Java to C++ for easier maintainability and higher efficiency.

Course Outcomes:

My improved artifact meets four of the five course outcomes:

**Outcome one: Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science**

For this outcome I ensured that my code had complete in code commenting consistent with C++ professional standards. I also worked to make sure that my code was readable and understandable through the use of proper indentation, formatting, and object oriented design.

**Outcome three: 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**

To meet outcome three, I created pseudocode to plan out my implementation and solve any logic problems before I wrote the code itself. Throughout the design process I worked to plan for maintainability and eliminate any possible bugs or other issues.

**Outcome four: 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**

Outcome four was reached by developing my code using standard iterative testing techniques and following industry design standards. My code also uses a database system that is extremely prevalent in the industry, Sqlite3.

**Outcome five: 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**

For outcome five I ensured that my software did not have any logic issues by solving them in my pseudocode during the planning phase and worked to avoid security vulnerabilities by implementing one of the industry standard databases that is well maintained.

Reflection:

I struggled with my databases code primarily due to issues with my compiler and the installation of sqlite 3. When I was initially learning to implement the database in my code I found that there were many different options for the initial setup and I could not find an up to date guide on the best way to use it for C++. Due to this I spent a large amount of time trying various methods to implement the database before eventually settling on the amalgam method from the official documentation. This allowed me to include all needed headers directly in my project folder while doing my initial database creation through CMD.

I learned a lot about both how sqlite3 works and the process used to implement it within my code. Before now I had only worked with pre setup databases and never built my own from scratch. This was an excellent experience and I am now sure that I could create databases and code for a variety of applications in the future.

Pseudocode:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated