CS 499 - Module 3 Enhancements

Conor Steward

conor.steward@snhu.edu

9/26/24

This Investment Calculator is a C++ program designed to help users evaluate various investment strategies over time. It supports three main investment strategies. The first is regular investment calculation which computes monthly and/or yearly compounded interest for a given initial investment, monthly deposit, and annual interest rate over an input number of years. Next, the dividend reinvestment plan (DRIP) simulates reinvesting dividends into additional shares, showing the compounding effect over time based on the dividend yield and stock price growth rate. Finally, the High-Risk vs. Low-Risk Investment Strategy Comparison compares the performance of a high-risk portfolio against a low-risk portfolio using different interest rates for each. This shows the different outcomes of each risk profile over the investment period.

The main functionality in the InvestmentCalculator class (InvestmentCalculator.h and InvestmentCalculator.cpp) is to manage the core calculations for the investment strategies using various functionalities. The High-Risk vs Low-Risk Strategy utilizes constructor overloading, where a default constructor is used to handle regular investment while an overloaded constructor is used for high-risk vs. low-risk comparison, accepting additional parameters for high-risk and low-risk rates.

The InvestmentReport class (InvestmentReport.h and InvestmentReport.cpp) is responsible for displaying the investment results in a user-friendly way. It includes methods to generate reports for static reports which display a year-end balance and earned interest for both regular investments and those with monthly deposits. The dividend reinvestment report displays how dividends are reinvested and how they impact the investment value over time. The risk comparison report compares the outcomes of high-risk vs. low-risk strategies, giving users insight into the different risk profiles.

Finally, the main program logic (main.cpp) collects user inputs for initial investment, monthly deposit, annual interest rate, high-risk rate and low-risk rate, dividend yield, stock price growth rate, and number of years to evaluate the investment. The main class then takes these user inputs to create an instance of the InvestmentCalculator and calls its methods to calculate the various investment outcomes. The results are displayed through the InvestmentReport class, which generates regular investment details, dividend reinvestment plan details, and high-risk vs low-risk strategy comparison.

This detailed, multifaceted report enables users to compare various investing strategies with ease, looking at how much they should invest and with what strategy they should approach investing to reach their financial goals. This program was originally created in a Data Analytics Professional Certification through Google Coursera.

This artifact originally showcased my ability to write in C++, implement input usage, and generate a user-friendly report for the user. It accomplishes these skills still, while also showcasing overloading, complex algorithms, and a more complex user friendly report. I selected this item because further implementation of investment strategies is well regarded in the eyes of businesses and enhancing it would allow me to dive deeper into complex algorithms and complex C++ coding. The improvements take this program from being a simple investment calculator with only 2 simple strategies into one that informs users of a variety of choices for investing including DRIP and HRHRvLRLR in addition to a standard investment strategy.

- Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.

    - My more thorough commenting in this artifact upholds this course outcome by creating a program that can be utilized easily by others as well as enhanced due to the commenting and simple nature of the program.

- Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
    - The README explains in detail about the new and old strategies and what each input parameter means, as well as explaining to the user how to go about executing the program.

- 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.
    - This course outcome is the most upheld portion within this artifact enhancement. The complex nature of investment calculating and overloading methods are both utilized to great effect in this program.

- 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.

    - As mentioned above, some higher level C++ tools like overloading are utilized in this program. While complex and modern libraries are not utilized, the very custom nature of this program needed custom code for its completion.

- 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.
    - This course outcome is not upheld within this program, as input validation is not utilized. That being said, the program does not have access to vulnerable pieces of a computer while running and messing with it in an intentionally nefarious way will simply crash the program in their own local instance, so deep security is not vital.

My coverage plan did not need updating, as I stuck to the planned enhancements.

Reimplementing the overloaded constructor is not a skill I have had to utilize in some months and the utilization of this technique took me some time to figure out. In addition, I had to carefully research how the algorithms should be constructed to properly calculate the outcome based on user inputs. Additionally, enhancing this program to support multiple investment strategies taught me the importance of modular design. By separating the investment calculations into different functions, I could extend the program's functionality without introducing needless complexity. Another challenge was enhancing the artifact without losing focus on its primary goal of providing accurate investment situations. The program's new features were built to complement the core functionality and I learned how to structure enhancements so that they align with the program's original goals while offering new insights.