**Final Report**

Project: Finance Manager

Juan Higuera

COMPSCI‑2

04/22/2025

For this project I chose to build a personal Finance Manager application that lets users track income and expenses, set budgets by category, and save all data between runs. I wanted something directly relevant to my everyday life and interests, so I designed it in C++ to showcase the newer C++ features like file I/O, smart pointers, and unit testing. Over the semester we studied streams, classes, inheritance, templates, and algorithms. This project gave me a strong way to tie those features together into a portfolio‑worthy application.

At the core of the design is a FinanceManager class that holds two collections: a vector of Transaction objects and a vector of Budget objects. Each Transaction has its own type which are income or expense, amount, category, description, and timestamp, Budget tracks an allocated amount, running “spent” total, whether it resets monthly. I used unique\_ptr everywhere to enforce ownership and prevent leaks and factored file reading/writing into a reusable FileUtils template so that export logic lives in one place. All user interactions like adding, listing, deleting are separated in Main.cpp, which calls the manager’s public interface.

Persistence is handled by simple CSV files “transactions.csv” and “budgets.csv”. On startup we loadData(), rebuilding any expense totals into their budgets, on each add or delete we immediately saveData(). This ensures that a crash or power loss never leaves the user in an inconsistent state. I paid attention to error handling around user input (rejecting non‑numeric choices or out‑of‑range deletes) and to other cases like displaying “(none)” when no items exist.

To verify the program’s accuracy I wrote a suite of Catch2 unit tests that covers all three core classes. TransactionTests.cpp checks getters and CSV round‑trips, BudgetTests.cpp ensures budgets start empty, can be added, deleted, and correctly exported/imported, FinanceManagerTests.cpp validates that the manager starts empty, can add and remove both transactions and budgets, and properly saves and reloads data. This testing layer gave me confidence that future changes won’t introduce issues.

The biggest challenges came when adding the file paths between the executable and test runner, and in managing ownership when deleting items making sure that budgets’ spent totals stayed in sync after both load and delete operations. I solved these by standardizing on relative paths plus “Copy to Output Directory” for CSVs, and by keeping budget adjustments inside the FinanceManager methods themselves.

In future work I’d like to add date‑range filters, visual charts of spending trends, and maybe a plugin system for importing from online banking APIs. But right now this project demonstrates advanced C++ design, modularity, memory management, stream Input/output, and a robust test suite.