Cross-Language Application Development

Group 1

https://github.com/baralsamrat/MSCS632_Project_Group

Samrat Baral

Prasanna Adhikari

Shashwat Baral

Sahithi Bontha

Spring 2025 Second Bi-term

Advanced Programming Languages -MSCS-632-B01

University of the Cumberlands

Dr. Dax Bradley

March 23, 2025

Introduction

This repository contains an Expense Tracker Application implemented in Python and C++, demonstrating how different programming languages handle data structures, memory management, and error handling. The application allows users to:

- Add expenses with fields: Date, Amount, Category, and Description.
- Filter expenses by date range or category.
- View total expenses by category and overall.
- Compare implementations between Python and C++.

Both files are analyzed for key metrics that reflect the implementation of the core features (e.g., functions for filtering, summarizing, etc.).

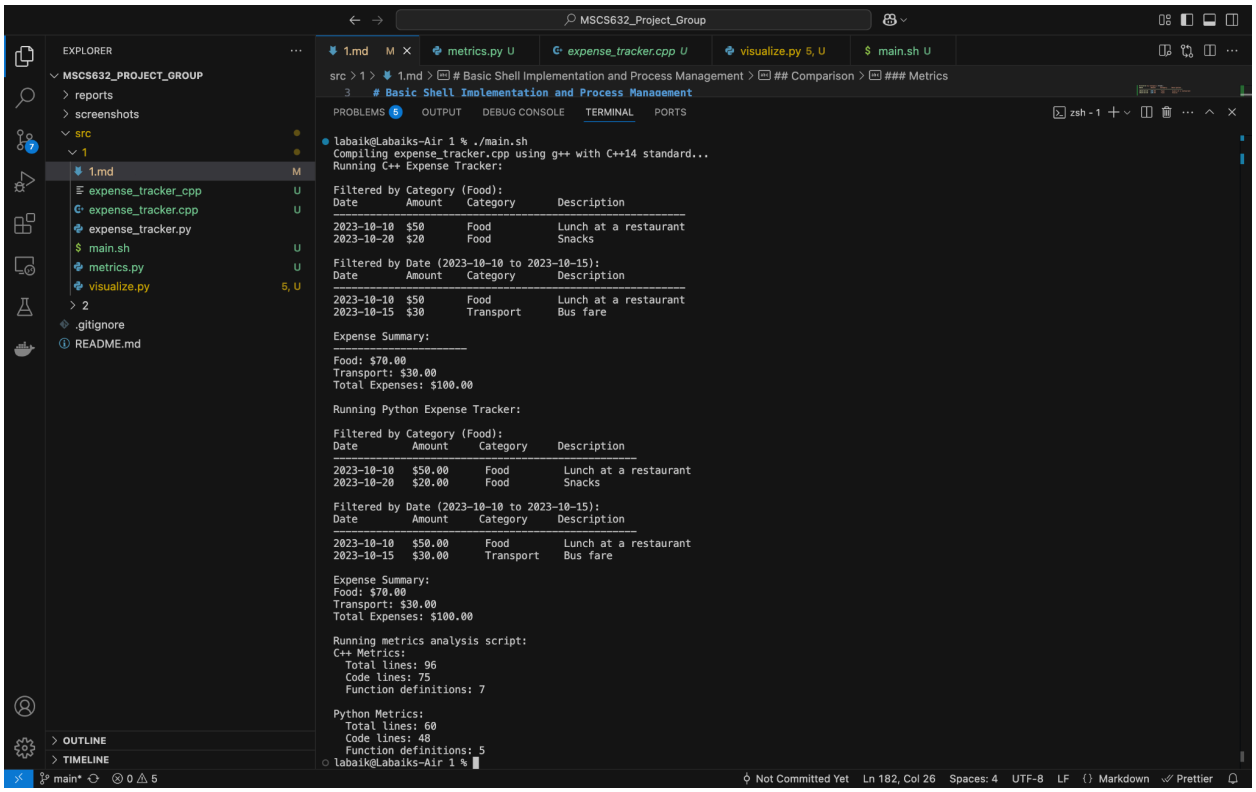
Language-Specific Features

The C++ implementation shows memory management and use of STL containers, while the Python implementation (not shown here in detail) would use dictionaries, dynamic typing, and libraries like datetime. The metrics (and perhaps further analysis using tools like radon for Python or similar tools for C++) can be extended to reveal more about complexity and structure.

Side-by-Side Comparison:

By generating and visualizing these metrics, you can clearly see how each language's implementation compares in terms of code size and structure, which can help illustrate that both meet the core requirements while highlighting language-specific coding styles.

Screenshots



Comparisons:

Requirement	C++ Implementation	Python Implementation
Data Storage	Uses a struct `Expense` (with fields for date, amount, category, and description) stored in a `std::vector` of smart pointers.	Uses dictionaries (or classes) and lists to store expense entries with keys/attributes such as date, amount, category, and description. (See README for details.)
Filtering Functionality	Provides functions like `filterExpensesByCategory` and `filterExpensesByDate` to filter expenses by category and date range.	Implements similar filtering logic, utilizing Python's dynamic typing and concise list comprehensions or loops.

Summary Function	The `showSummary` function uses a `std::map` to aggregate expenses by category and calculate the overall total.	Uses Python's dictionaries to accumulate totals per category and calculate overall expenses.
Language-Specific Features	Demonstrates explicit memory management via `std::unique_ptr` and uses STL containers (vector, map).	Showcases dynamic typing, leverages Python's `datetime` module for date handling, and simpler data structures.