Budget Spreadsheet Tracker

ECE508 - Python & Scripting Wksp

Final Project Proposal

Spring 2024 – Jean Paul Mugisha

April 21, 2024

Abram Fouts, Niko Nikolov

Overview

Personal financial management poses a significant challenge for many individuals.

We are proposing a Python-based application for the analysis and visualization of bank statement data. The tool will process CSV statements, generating pie charts for spending category breakdowns, income-versus-spending comparisons, and detailed averages across daily, monthly, and yearly intervals. The application aims to enhance user understanding of their financial habits, empowering them to make informed budgeting and saving decisions.

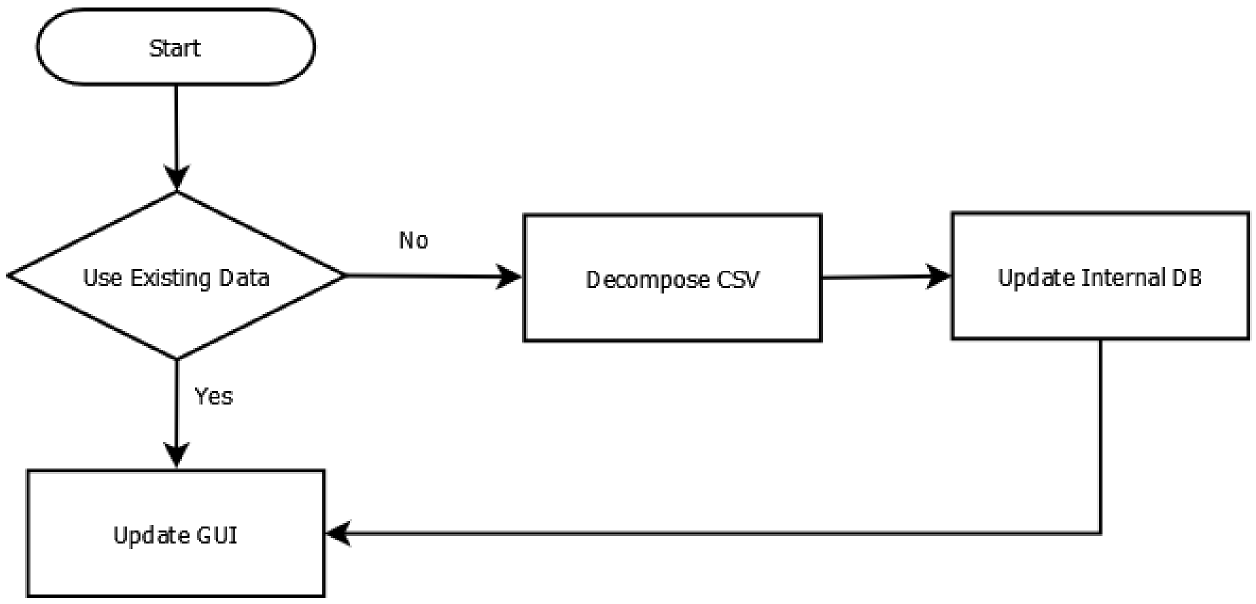


Figure 1. Backend High Level Flow

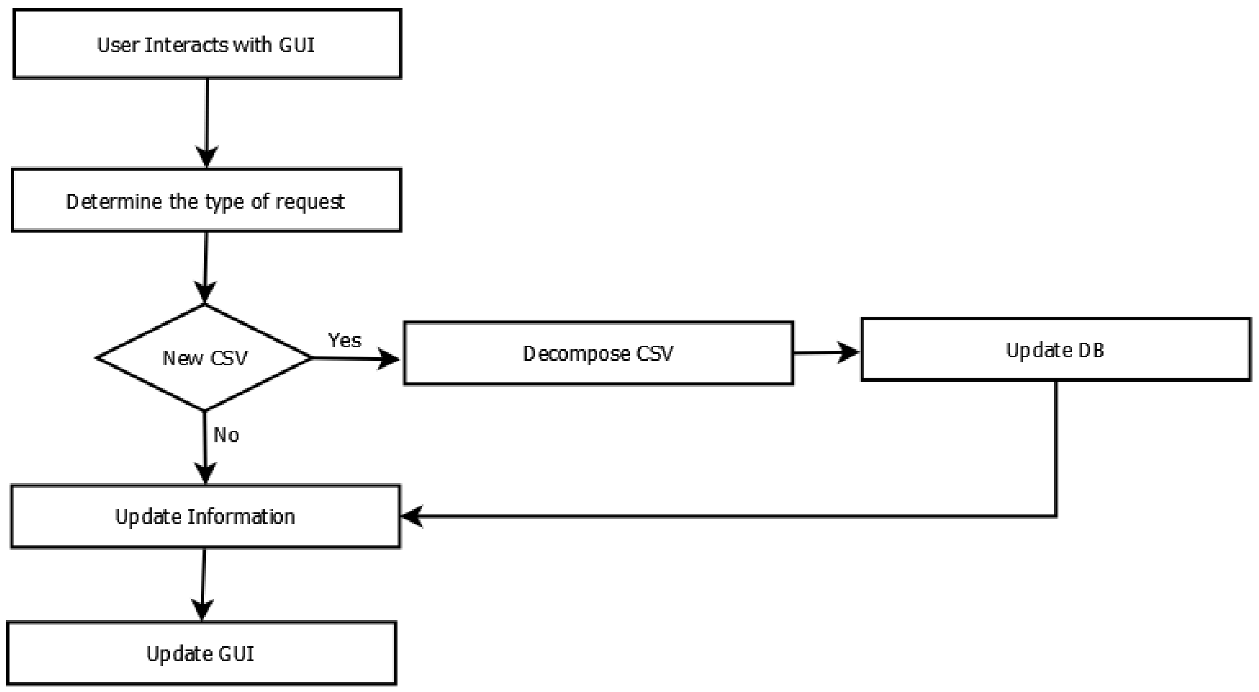


Figure 2. Frontend High Level Flow

Front-End, User Interface

User Interface Functionality

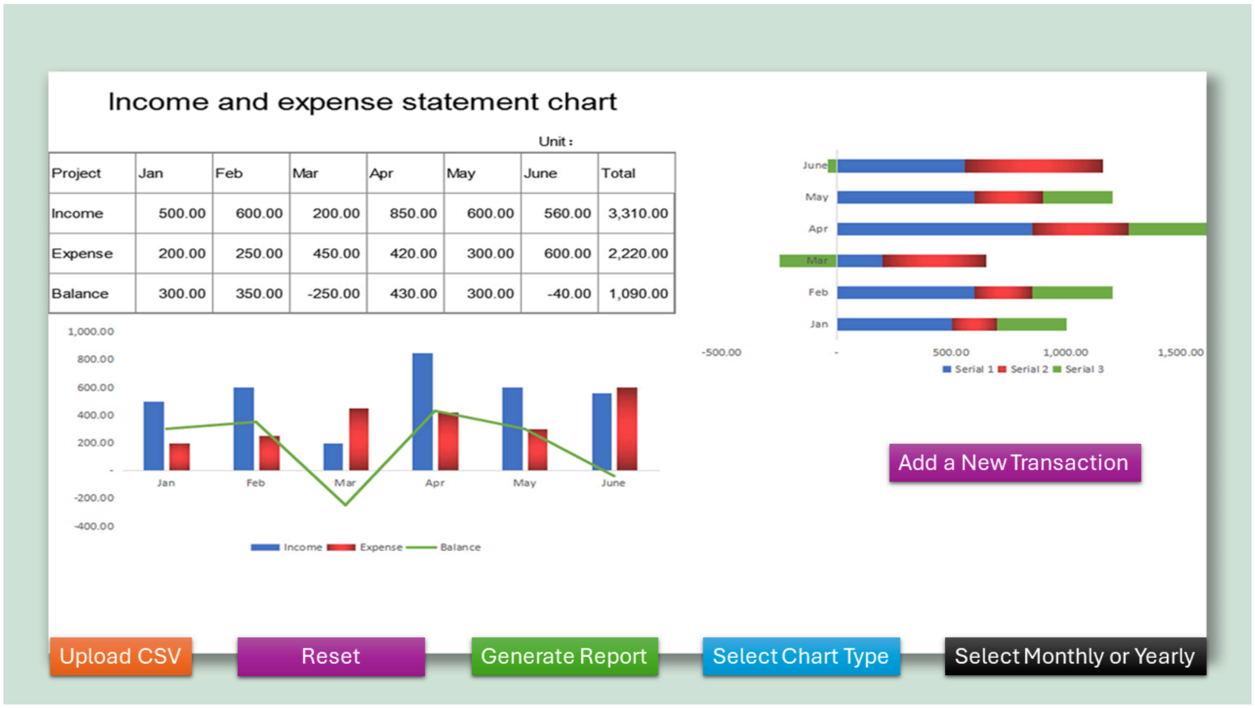
Our user interface is designed to be both intuitive and informative. It features a straightforward CSV file upload button and a dynamic default layout that highlights key financial trends like high-spending months and income spikes. A pie chart visualizes income versus expenses, and if the user provides an APR, it can even calculate debt payoff timelines.

The UI captures all necessary user input, seamlessly communicating with the backend to update data visualizations based on user requests. To streamline development, we'll use a mock data file initially, allowing us to focus on UI responsiveness independently of the backend.

UI Structure & User Interaction

The UI offers a flexible experience with dedicated buttons for resetting data (with a helpful warning, of course!), refreshing visualizations, and selecting chart types dynamically (think Excel-style chart switching).

Users can easily toggle between monthly, yearly, and weekly reports. Additionally, a transaction input button allows for manual adjustments if the CSV parsing needs a nudge. We're also considering adding the ability to export reports to PDF for easy sharing and record-keeping. The frontend code will be structured into UI elements and handlers, ensuring a clear separation between presentation and the backend communication logic.



Income and expense statement chart

Unit :

Project	Jan	Feb	Mar	Apr	May	June	Total
Income	500.00	600.00	200.00	850.00	600.00	560.00	3,310.00
Expense	200.00	250.00	450.00	420.00	300.00	600.00	2,220.00
Balance	300.00	350.00	-250.00	430.00	300.00	-40.00	1,090.00

Figure 3. UI Mockup Design

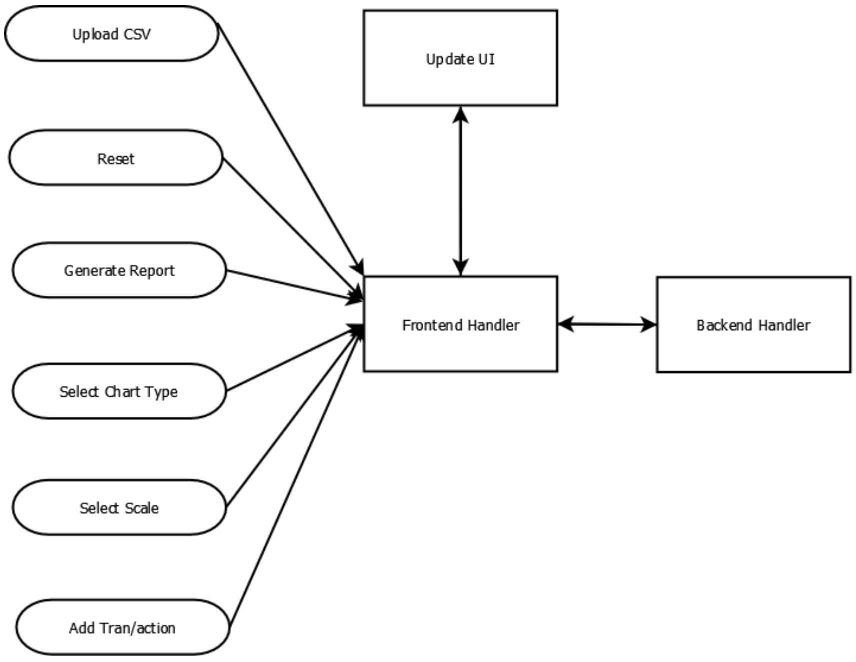


Figure 4. User interactions' flow

Backend

Backend Architecture and Data Flow

The backend architecture is designed for efficient processing and management of financial data. CSV parsing involves reading transactions into a global data structure, subsequently used to update a TinyDB database. The database meticulously categorizes information by month, date, year, and time, supporting flexible temporal queries. User-uploaded CSV files seamlessly integrate into the database, enabling cumulative analysis for insights spanning monthly, yearly, or custom periods. To optimize performance, the backend pre-calculates various spending summaries, minimizing runtime computations in the GUI. The focus is on ensuring data integrity through CSV validation and potential front-end/backend handshake mechanisms. Based on the format of the CSV the program will search for defining headers and transpose them into categories to collect the presented data in the CSV.

Development Methodology and Design

The project prioritizes a modular backend design with dedicated modules for database management, computations, CSV parsing, and request handling. This promotes code organization and maintainability. A well-defined API facilitates communication between the backend and the user interface. The development process emphasizes efficiency, utilizing lists, classes, and dictionaries for optimal data representation. Self-documenting code, docstrings, and comprehensive comments ensure clarity. Code quality is rigorously maintained through the use of the Black formatter, static linters, and a strong emphasis on unit testing for each function.

Expected Libraries

1. Operations & Workings – OS, Math, other common libraries
2. CSV Read – Pandas
3. GUI – Tkinter
4. Graph / Plots – Matplotlib

5. Database – TinyDB

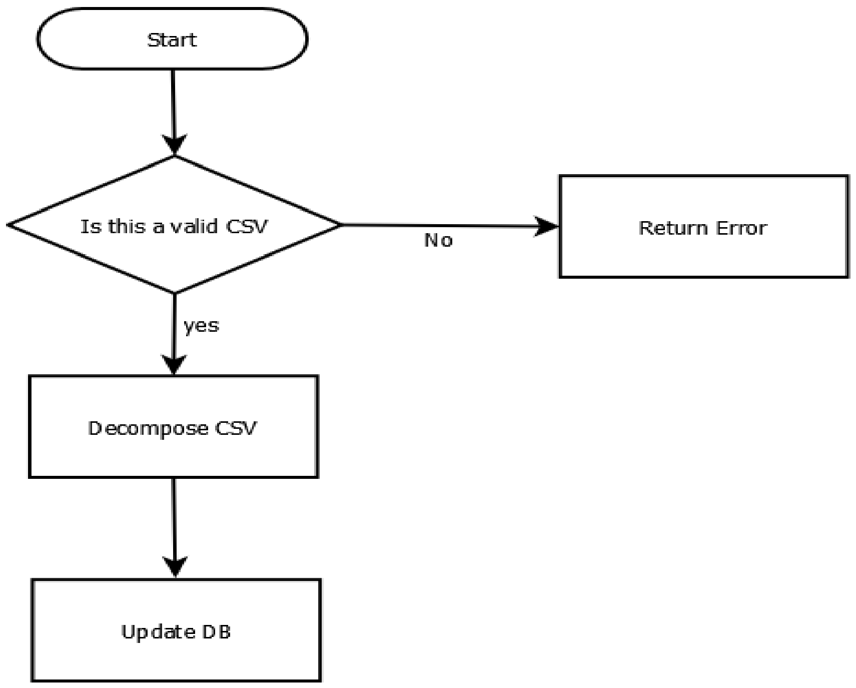
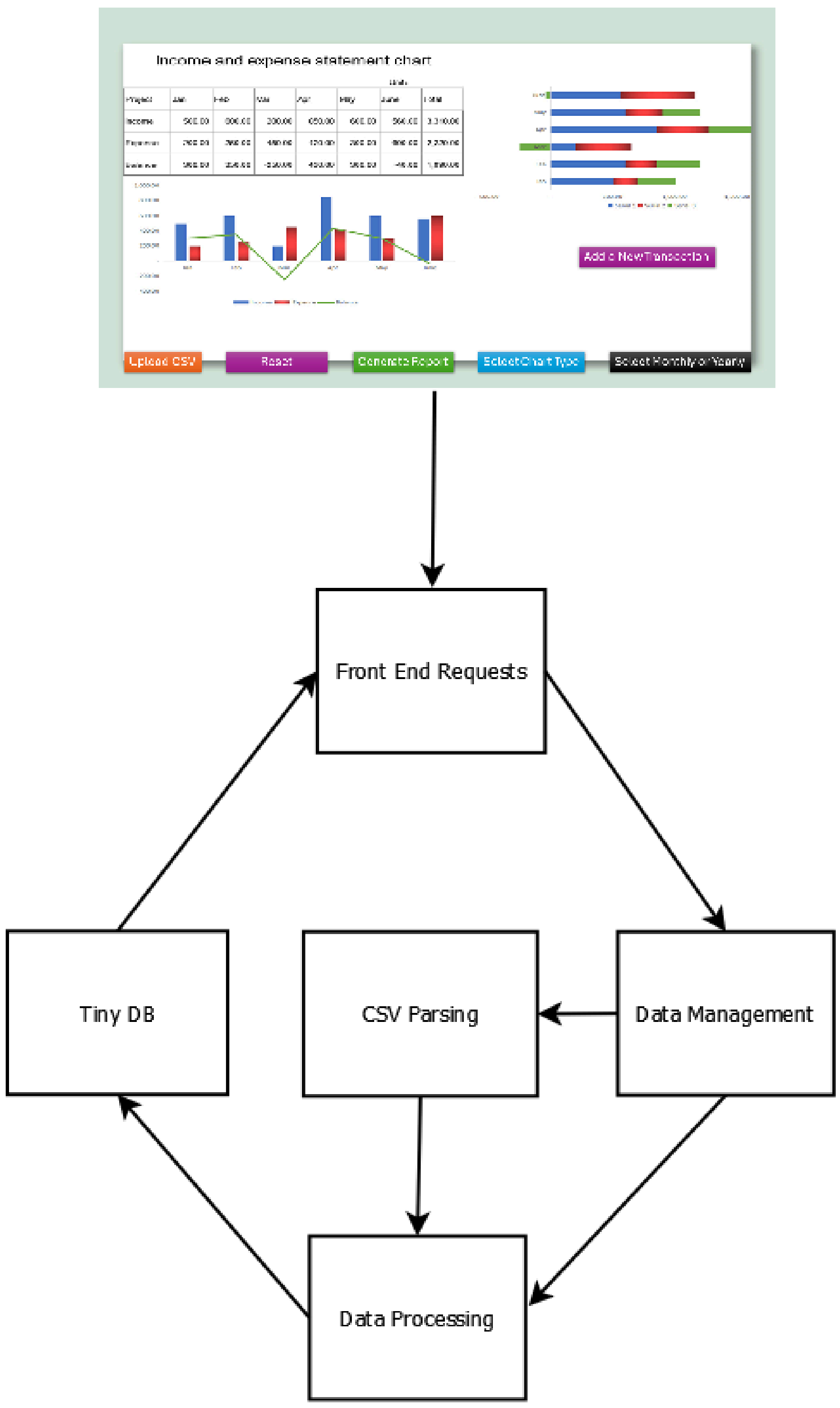


Figure 5. Backend High Level design



Units

Project	Jan	Feb	Mar	Apr	May	June	Total
Income	500.00	600.00	200.00	650.00	600.00	564.00	3,310.00
Expenses	[illegible]	[illegible]	450.00	170.00	268.00	[illegible]	2,270.00
Balance	[illegible]	[illegible]	-250.00	480.00	[illegible]	-46.00	[illegible]

Figure 6. Code Design and Code Relationship

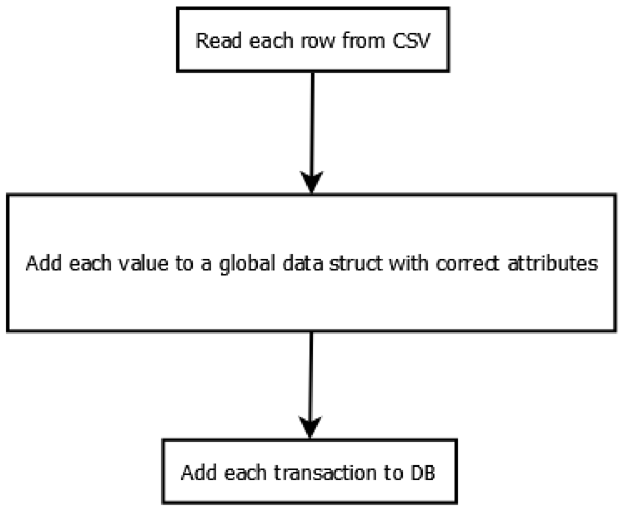

Read each row from CSV
Add each value to a global data struct with correct attributes
Add each transaction to DB

Figure 7. CSV file parsing High Level