INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 9|| MAY 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Approved || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.05.13.09.003 PUBLISHED ON: MAY 04, 2026

PERSONAL FINANCE AND INVESTMENT PORTFOLIO TRACKER: A LOCALLY HOSTED WEB-BASED INVESTMENT MANAGEMENT SYSTEM

Chittiki Gangadhara [1], S. Usha Rani [2]
[1] Student, [2] Professor & Head
[1,2] Department of Computer Applications,
Viswam Engineering College, Madanapalle, Andhra Pradesh, India

Abstract

This paper presents a Personal Finance and Investment Portfolio Tracker, a locally hosted web application that simplifies and enhances the management of personal investments distributed across multiple accounts and brokerage platforms. Implemented in Python using the Flask framework, the system provides a centralized dashboard through which users can monitor stocks and other financial assets in a unified, real-time view. The application integrates the yfinance library to retrieve live market data — including current prices, previous closing values, 52-week highs and lows, and currency exchange rates — and stores all user, platform, account and asset records securely in a structured SQLite database following a four-level relational hierarchy. Interactive visualizations rendered with Google Charts present portfolio allocation, historical performance trends and asset-level comparisons, while a goal-tracking module allows users to set financial targets and visualise progress through dynamic indicators. Two financial simulation tools — a buy calculator and a sell calculator — let users evaluate hypothetical investment decisions before executing them by estimating profit or loss and changes in cost basis. Unlike cloud-based alternatives that depend on third-party data sharing and recurring subscriptions, the proposed system operates entirely on local hardware, ensuring data privacy and eliminating ongoing licensing cost. The application supports multiple users within a single instance, making it suitable for individual and household-level financial management. Overall, the project demonstrates the integration of modern web development, relational database management and financial analytics into an efficient, privacy-focused portfolio-tracking platform that supports real-time investment monitoring and informed decision making.

Keywords *Personal finance; portfolio tracking; Flask; SQLite; real-time market data; financial visualization*

Recommended Citation:
Gangadhara, C. , Rani, S. H. : " Personal Finance and Investment Portfolio Tracker: A Locally Hosted Web-Based Investment Management System", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (9), May 2026, pp. 1413-1418 https://doi.org/10.64672/IJIFR/26.05.13.09.003

1. Introduction

Personal finance and investment portfolio management have evolved substantially in the digital era. Investors who once relied on brokerage statements or hand-maintained spreadsheets now require unified real-time views that consolidate holdings across multiple platforms and account types. Manual approaches are time-consuming and error-prone; spreadsheet-based tools lack reliable real-time market integration; commercial desktop applications such as Quicken impose subscription costs and regional limitations; and cloud-based platforms such as Wealthica and Sharesight require recurring fees and the transmission of sensitive financial data to third parties, raising privacy concerns. Brokerage-provided dashboards are accurate but restricted to a single platform, leaving multi-broker investors without a consolidated view.

Against this background, the contribution of this paper is threefold. First, we propose the Personal Finance and Investment Portfolio Tracker, a locally hosted Flask-based web application [1] that consolidates investments across multiple accounts and platforms into a single dashboard while keeping all data on the user's hardware. Second, we describe a real-time data integration pipeline based on the yfinance library [3], which automatically retrieves live prices, 52-week ranges and FX rates without manual entry. Third, we document a complete system architecture and operational workflow — together with IEEE-style engineering diagrams and Mermaid source — that establish a transparent reference implementation suitable for academic replication and downstream extension.

2. Literature Survey

Modern web-based portfolio management systems sit at the intersection of relational database theory, lightweight web frameworks, financial analytics and interactive visualization. Codd's foundational relational model [2] underpins the structured organisation of users, accounts and assets used in this work and remain the basis for systems such as SQLite. On the application side, Python-based micro-frameworks have been widely adopted for finance-oriented dashboards: Grinberg [1] characterises Flask as a lightweight architecture that supports rapid development of database-driven web applications with minimal overhead, while its Jinja2 templating engine enables clean separation between Python logic and HTML presentation.

For market-data integration, the yfinance library [3] retrieves live and historical equity and FX series from Yahoo Finance without commercial licensing, automating valuation and removing manual update overhead. McKinney [4] further documents the broader Python data-manipulation ecosystem that underlies modern financial computing. On the presentation side, Bootstrap and responsive CSS layouts provide consistent cross-device rendering, and Google Charts contributes a mature, declarative API for portfolio allocation and performance visualisation. Foundational financial-theory texts such as Hull [5] and Graham and Dodd inform the simulation tools and risk metrics that support investment evaluation. Across the literature the consistent finding is that lightweight, locally hosted deployments based on Flask and SQLite remain attractive when data privacy, deployment simplicity and cost are decisive — precisely the design constraints addressed by the proposed system.

3. Proposed Work and Methodology

3.1 Limitations of Existing Approaches

Existing approaches each fall short in at least one important dimension. Manual record-keeping is private but does not scale; spreadsheets are flexible but error-prone and weakly automated; commercial desktop tools (Quicken) carry subscription cost and regional rigidity; cloud-based platforms (Wealthica, Sharesight) require third-party data sharing and recurring fees; and brokerage dashboards are accurate but single-platform. None of them simultaneously delivers (i) automated multi-platform consolidation, (ii) real-time market data, (iii) interactive visualisation and (iv) full local data ownership. The proposed system targets exactly this combination.

3.2 Proposed System Overview

The proposed Portfolio Tracker is a Flask-based web application that runs locally on the user's hardware. The data model is a four-level relational hierarchy in SQLite: User → Platform → Account → Asset, supplemented by app_settings (financial goals) and portfolio_history (daily snapshots). When a user adds or edits an asset by ticker symbol, the Market Data Service fetches live values via yfinance — current price, previous close, 52-week high/low and FX rate when applicable — and the asset record is persisted with both static cost-basis fields and dynamically refreshed market values. CRUD controllers handle users, platforms, accounts and assets; Jinja2 templates render the dashboard; and Google Charts produces interactive visualisations of portfolio allocation, historical performance and asset comparisons. Two simulation tools — a sell calculator (estimating profit/loss at hypothetical sale prices) and a buy calculator (estimating new average cost and updated portfolio value) — let users evaluate trades without modifying stored data. The same instance supports multiple users, enabling household-level tracking with logical data separation.

Table 1 — Comparison of Existing Systems with the Proposed System

Feature	Existing Systems	Proposed System
Data entry	Mostly manual	Automated via yfinance API
Data privacy	Low (cloud-based)	High (local storage only)
Cost	Subscription-based	Free (locally hosted)
Multi-platform tracking	Limited	Supported
Real-time updates	Partial	Fully supported
Visualization	Basic	Advanced interactive charts
Customization	Limited	High flexibility

4. System Architecture and Diagrams

This section presents two IEEE-style engineering diagrams: (i) the system architecture diagram (Fig. 1), capturing the three-tier organisation of the application, and (ii) the workflow diagram (Fig. 2), capturing the temporal ordering of the asset-management and dashboard-refresh pipeline. Mermaid source is provided beneath each figure to support textual editing and machine-readable reproduction.

4.1 System Architecture Diagram

The system architecture, as shown in Fig. 1, follows a three-tier organisation. The Presentation Layer is a browser-based interface built with HTML, CSS, Bootstrap and Google Charts. The Application Layer is implemented in Flask and provides URL routing, Jinja2 template rendering, CRUD controllers for users, platforms, accounts and assets, the buy and sell simulation tools, and a market-data service that wraps yfinance. The Data Layer combines the local SQLite database — which holds users, platforms, accounts, assets, app_settings and portfolio_history — with the external Yahoo Finance API consumed for live prices and FX rates. Persisted data and external market data flow back into the application layer to drive the dashboard.

4.2 Workflow Diagram

The workflow, illustrated in Fig. 2, begins when an authenticated user opens the portfolio dashboard and adds or edits an asset by ticker symbol. Flask validates the input and fetches live market data via yfinance, including price, 52-week range and FX rate. The record is then persisted to SQLite and the system computes portfolio metrics — current value, profit/loss and percentage return — aggregated through the User → Platform → Account → Asset hierarchy. The dashboard is rendered with Google Charts, after which the user may optionally invoke the buy or sell simulator to evaluate hypothetical trades without altering stored data.

INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
IJIFR /202605/ V13/E9/003
VOLUME 13||ISSUE 9|| MAY 2026
https://www.ijifr.org/

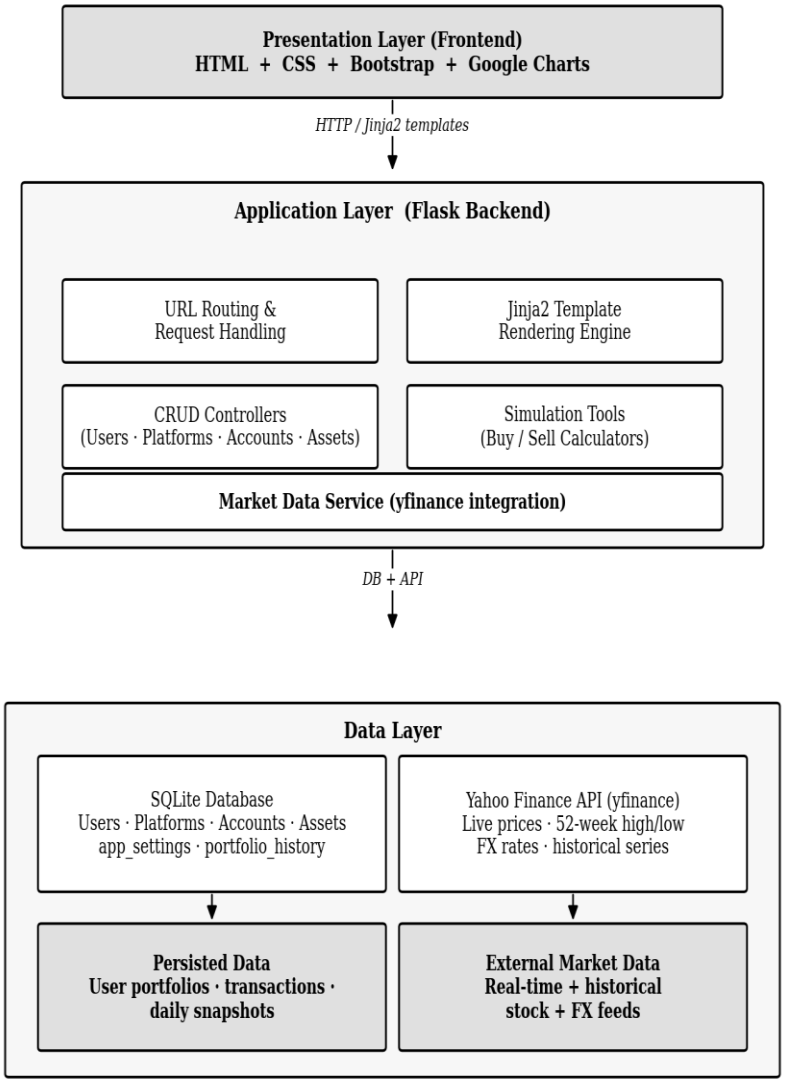

Figure 1 — System Architecture of the Personal Finance & Portfolio Tracker

Mermaid source for Fig. 1

```
```mermaid
flowchart TD
    UI["Presentation Layer<br/>HTML + CSS + Bootstrap + Google Charts"] -- HTTP/Jinja2 --> APP
    subgraph APP["Application Layer (Flask)"]
        R["URL Routing & Request Handling"]
        T["Jinja2 Template Rendering"]
        C["CRUD Controllers<br/>users · platforms · accounts · assets"]
        S["Simulation Tools<br/>Buy / Sell Calculators"]
        M["Market Data Service (yfinance)"]
    end
    APP --> DB[("SQLite Database<br/>users · platforms · accounts · assets ·<br/>app_settings · portfolio_history")]
    APP --> YF[("Yahoo Finance API<br/>live prices · 52-week · FX")]
```
```

INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
IJIFR /202605/ V13/E9/003
VOLUME 13||ISSUE 9|| MAY 2026
https://www.ijifr.org/

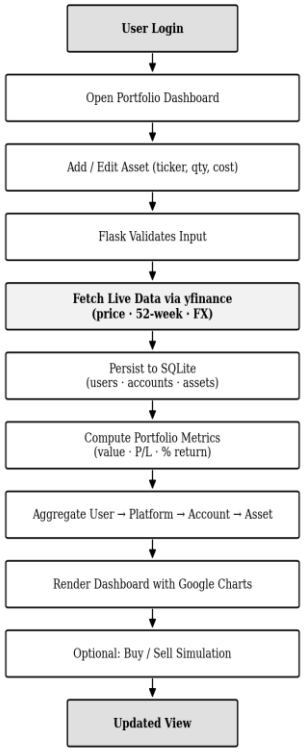

Figure 2 — Workflow Diagram of the Asset Management and Dashboard Refresh Pipeline

Mermaid source for Fig. 2

```
```mermaid
flowchart TD
    L([User Login]) --> D[Open Portfolio Dashboard]
    D --> A[Add / Edit Asset - ticker, qty, cost]
    A --> V[Flask Validates Input]
    V --> F[Fetch Live Data via yfinance]
    F --> P[Persist to SQLite]
    P --> M[Compute Portfolio Metrics - value, P/L, % return]
    M --> AG[Aggregate User -> Platform -> Account -> Asset]
    AG --> R[Render Dashboard - Google Charts]
    R --> S[Optional: Buy / Sell Simulation]
    S --> H([Updated View])
```
```

5. Results and Discussion

The Personal Finance and Investment Portfolio Tracker was implemented and evaluated as an end-to-end Flask + SQLite + yfinance system. It successfully consolidates investments across multiple users, platforms, accounts and assets into a single unified dashboard built on the four-level hierarchical data model. Real-time stock prices, previous closing values, and 52-week high/low data are retrieved automatically via yfinance, eliminating manual entry and keeping portfolio valuations current. The dashboard reports total investment value, current market value, profit or loss, cash balance and category-wise allocation; interactive Google Charts surface asset performance comparisons and historical portfolio trends. The buy and sell simulation tools correctly estimate profit, loss and changes in average cost without modifying any persisted record, supporting better trade decisions. From a performance perspective, the application runs efficiently on standard hardware, with fast database queries, API calls and chart rendering. The use of SQLite ensures lightweight, reliable local storage at the cost of limited horizontal scalability.

Table 2 — Portfolio Performance Metrics Reported by the System

Metric	Description
Total Investment Value	Total amount invested by the user
Current Market Value	Real-time aggregated value of all assets
Profit / Loss	Difference between investment and current value
Return Percentage (%)	Profit or loss expressed in percentage terms
Best Performing Asset	Asset with the highest return
Worst Performing Asset	Asset with the lowest return

Three limitations were observed. First, the system runs locally and is reachable only from the host machine unless additional networking (VPN, port forwarding) is configured. Second, market data is sourced exclusively from Yahoo Finance via yfinance, so any temporary outage or rate-limit on that service propagates directly to the application; no fallback data source is currently configured. Third, the SQLite backend, while well suited to single-host personal use, becomes a bottleneck for concurrent multi-user or enterprise-scale deployments. These limitations are pragmatic trade-offs given the privacy-focused, low-cost design goals and motivate the future-work directions outlined below.

6. Conclusion and Future Enhancement

This paper presented a locally hosted web application for personal finance and investment portfolio tracking that combines Flask, SQLite and the yfinance library to deliver real-time, multi-platform portfolio consolidation while preserving full data ownership. The four-level User → Platform → Account → Asset data model supports detailed reporting; Google Charts provide interactive allocation, performance and trend visualisations; and the buy/sell simulation tools enable data-driven trade evaluation without affecting stored records. Future enhancements include secure multi-device deployment with authentication and authorization, direct brokerage-API integration to automate transaction import, advanced analytics (time-weighted return, benchmark comparison), modern visualisation libraries such as Plotly or Chart.js, multi-currency support, a companion mobile application with notifications, and structured data-export and tax-reporting features.

7. References

[1] M. Grinberg, “Flask Web Development: Developing Web Applications with Python,” 2nd ed., O’Reilly Media, Sebastopol, CA, USA, 2018.
[2] E. F. Codd, “A relational model of data for large shared data banks,” Communications of the ACM, vol. 13, no. 6, 1970, pp. 377–387.
[3] R. Aroussi, “yfinance: Yahoo! Finance market data downloader,” GitHub repository, 2019. [Online]. Available: https://github.com/ranaroussi/yfinance
[4] W. McKinney, “Data structures for statistical computing in Python,” in Proc. 9th Python in Science Conf. (SciPy), vol. 445, 2010, pp. 56–61.
[5] J. C. Hull, “Options, Futures, and Other Derivatives,” 10th ed., Pearson, 2018.
[6] R. Hettinger, F. Lundh, and G. van Rossum, “sqlite3 — DB-API 2.0 interface for SQLite databases,” Python Software Foundation, 2003. [Online]. Available: https://docs.python.org/3/library/sqlite3.html
[7] M. Otto and J. Thornton, “Bootstrap: The most popular HTML, CSS, and JS library in the world,” 2011. [Online]. Available: https://getbootstrap.com
[8] Google LLC, “Google Charts: Interactive charts for browsers and mobile devices,” 2023. [Online]. Available: https://developers.google.com/chart