FinAssist: A Personal Finance Assistant

Problem Statement & Goals

Managing personal finances can be overwhelming, especially when it comes to monthly spending patterns, keeping investments in perspective, and identifying practical ways to cut costs. While budgeting tools exist, they often focus only on tracking numbers without offering clear, actionable insights.

The goal of our project was to build **FinAssist**, a lightweight personal finance assistant that can:

1. Summarize monthly budgets over-/underspending.
2. Compare the performance of stocks or ETFs over various time horizons.
3. Provide actionable savings tips based on a curated finance knowledge base.

By combining these features, we aimed to create a tool that not only reports data but also helps users make smarter financial choices.

Methodology/Technical Approach

We structured FinAssist as a combination of data processing, a retrieval-augmented generation (RAG) module, and a lightweight rule-based router.

Dataset and Preprocessing

We used a transactions dataset from Kaggle, which contains household-level spending records across categories such as food, housing, utilities, and entertainment. We cleaned the data by normalizing transaction types, including categories not relevant to budgeting (e.g., paychecks, credit card payments), and aggregating spending by month. This produced a reliable monthly budget report that could be compared against a sample budget.

Knowledge Base + RAG Module

For financial advice, we created a small, curated knowledge base (KB) with topics such as the 50/30/20 rule, emergency funds, lowering bills, and meal-prep strategies. We embedded this KB with **MiniLM (all-MiniLM-L6-v2)**, using cosine similarity search to retrieve relevant sections. To improve precision, we applied **keyword-filtering** and **domain-specific boosts** (e.g., prioritizing “internet” or “dining” when the query includes those words). We also injected **safe dollar examples** to make the tips concrete and relatable.

Routing Logic

Queries are routed to one of three modules:

- **Budget:** For questions with keywords like “budget”, “over-budget”, or explicit dates (YYYY-MM), we generate a budget summary with top over- and under-budget categories.
- **Investments:** For queries with stock tickers or investment terms, we fetch historical OHLC data using “yfinance”, compute return percentages, and plot results.
- **Advice (RAG)**: For queries like “how can I lower my phone bill?” or “what is the 50/30/20 rule?” we return concise, educational answers from the KB.

User Interface

We wrapped the system in a **Gradio chat UI**, allowing natural question-answer interaction. The chat interface runs in Google Colab with a shareable link for live demos.

Challenges & How Addressed

One challenge was that early versions of the RAG module sometimes returned irrelevant bullets. For example, a question about lowering internet bills could include advice about dining out. We solved this by refining keyword filters and scoring heuristics to better match query intent.

Another issue was that the router sometimes mixed outputs (e.g., producing both a budget summary and advice for the same query). We fixed this by reordering the router so that advice queries are caught first, then budget queries, and finally investment queries.

We also encountered technical difficulties with GitHub integration. At one point, our notebook was marked “invalid” due to metadata conflicts. We resolved this by cleaning metadata outputs, using .gitignore to include cache files, and learning to handle rebase conflicts.

Results and Evaluation

FinAssist can now successfully:

- **Answer budgeting questions:** e.g., “Show me my top 3 over-budget categories in 2019-09” produces a summary where “Alcohol & Bars” and “Restaurants” are flagged as over-budget.
- **Answer advice questions:** e.g., “How can I lower my phone bill?” returns only relevant bullets such as “call your provider for promo pricing” and “bundle or auto-pay discounts.”
- **Answer investment questions:** e.g., “Compare TSLA vs SPY over 6mo” computes returns (+24.7% for TSLA, +10.9% for SPY in our test) and shows a bar chart.

While we did not formally benchmark accuracy, qualitative testing shows that answers are consistent with the knowledge base, and outputs are aligned with user expectations.

Future Improvements

There are several directions for improvement:

- Expanding the knowledge base with more budgeting rules and investment education.
- Using a larger or domain-specific embedding model for better semantic retrieval.
- Adding natural-language generation to make answers sound smoother and more conversational.
- Connecting to real-time financial APIs (e.g., Plaid for transactions) for personalized insights.
- Deploying the app as a HuggingFace Space for easier public access.