FinTerm (Finance Terminal) User Guide

A finance calculator that lives in your terminal!

By **vsv** ♡

Introduction and Overview

FinTerm is a simple python-based application designed to simulate banking, provide financial analysis, and transaction management capabilities directly from your terminal. Retro-inspired, It is a spiritual predecessor of modern banking softwares.

The system operates through an interactive command-line interface that accepts structured commands, processes financial data, and returns results in JSON format for easy integration and parsing. FinTerm maintains user profiles, manages multiple accounts, tracks transactions, and provides analytical insights into financial behavior patterns.

Getting Started with FinTerm

Installation and Launch

To begin using FinTerm, execute the core.py file from your terminal. Upon launch, the system displays an intro and enters an interactive command prompt mode where you can input commands continuously until you choose to exit.

```
python3 core.py
```

The application initializes all necessary components including the financial calculator, user manager, transaction processor, and analytics engine. The system remains active and maintains state throughout your session, allowing for multi-step financial operations.

Understanding the Interface

The FinTerm interface operates through a command prompt that accepts structured commands following a specific syntax pattern. Each command consists of a primary action, an optional subcommand, and various parameters specified with flags. The system processes each command immediately and returns results in JSON format, providing both success confirmations and detailed error messages when necessary.

Complete Command Reference

Financial Calculations Module

The calculation module provides rudimentary financial computation capabilities for various scenarios.

Interest Calculation

The interest calculation feature supports both simple and compound interest computations. As an example: *rate input must be in decimal form

```
calculate interest --principal 10000 --rate 0.05 --time 3 --type compound
```

This command calculates compound interest on $10,000 at 5% annual rate over 3 years, returning both the interest earned and the final amount. For simple interest calculations, change the type parameter to "simple". The system returns results with two decimal place precision.

Loan Payment Calculation

The loan calculation feature determines monthly payments, total payment amounts, and total interest for fixed-rate loans. As an example: *rate input must be in decimal form

```
calculate loan --principal 250000 --rate 0.045 --years 30
```

This example calculates the monthly payment for a $250,000 mortgage at 4.5% annual interest over 30 years. The system provides the monthly EMI (Equated Monthly Installment), total amount to be paid over the loan term, and the total interest component.

Compound Annual Growth Rate (CAGR)

The CAGR calculation helps determine the annual growth rate of investments over time.

```
calculate cagr --initial 50000 --final 85000 --years 5
```

This command calculates the compound annual growth rate for an investment that grew from $50,000 to $85,000 over 5 years, returning the percentage growth rate (output as percentage) that, when applied annually, would result in the final value.

Investment Management Module

Portfolio Recommendation

The investment recommendation system generates portfolio allocations based on risk profiles and available capital.

```
investment recommend --risk_profile moderate --amount 100000
```

This command generates a portfolio allocation for $100,000 based on a moderate risk profile. The system supports three risk profiles: conservative (emphasizing bonds and stable assets), moderate (balanced approach between growth and stability), and aggressive (prioritizing stocks and alternative investments). The output provides specific amounts for each asset class.

User and Account Management

User Creation

Creating a user establishes a profile within the system and automatically generates a default checking account.

```
user create --user_id bitch_stewie --name "Bitch Stewie"
```

This command creates a new user profile with the identifier "bitch_stewie" and the full name "Bitch Stewie". The system automatically creates an associated checking account using the pattern "{user_id}_checking" for immediate transaction capability.

User Deletion

The deletion feature removes users and all associated data from the system.

```
user delete --user_id bitch_stewie
```

This command permanently removes the specified user, including all accounts and transaction history. The system returns confirmation including the number of accounts deleted and the total balance that was removed from the system.

User Listing

The list command provides an overview of all users currently in the system.

```
user list
```

This returns a list showing each user's ID, name, number of accounts, and total balance across all accounts. This feature is particularly useful for system administration and overview purposes.

Transaction Processing

Processing Transactions

The transaction processor handles both deposits and withdrawals, maintaining account balances and transaction history.

```
transaction process --user_id bitch_stewie --amount 1500
--category salary --description "Monthly paycheck"
```

This command processes a $1500 deposit to bitch_stewie's account, categorized as salary with a descriptive note. For withdrawals, use a negative amount. The system validates sufficient funds for withdrawals and maintains a complete transaction log with timestamps. For balance inquiries, just input 0 and put “Balance Inquiry”

Behavioral Analysis

Transaction Analysis

The behavioral analysis module examines transaction patterns and spending habits.

```
behavior analyze --user_id bitch_stewie
```

This command analyzes all transactions for the specified user, providing insights into spending patterns, category distributions, and average transaction values. While the current implementation provides basic analytics, the framework supports expansion for more sophisticated behavioral analysis.

Session Management

Saving Session State

The save feature preserves the current system state to disk for later retrieval. There can only be one savestate and loadstate.

```
state save
```

This command saves all user profiles, accounts, balances, and transaction histories to a hidden file in your home directory (.finterm_savestate.pkl). This enables persistence across sessions and provides backup capability.

Loading Previous Session

The load feature restores a previously saved system state.

```
state load
```

This command retrieves and restores the most recently saved state, including all users, accounts, and transactions. This feature enables continuity across multiple work sessions and provides recovery capability after system restarts.

Practical Use Cases

Personal Financial Planning

Consider an individual planning for retirement who needs to evaluate various financial decisions. They begin by creating their user profile and then calculate how much their current savings will grow with compound interest over 20 years. They use the CAGR calculator to assess their investment performance over the past five years, helping them understand whether their current strategy meets their goals. The loan calculator helps them decide whether to refinance their mortgage by comparing monthly payments at different interest rates. Finally, they use the portfolio recommendation feature to rebalance their investments based on their risk tolerance as they approach retirement.

Small Business Financial Management

A small business owner uses FinTerm to manage multiple aspects of their business finances. They create separate user profiles for different business divisions, tracking income and expenses through the transaction processing system. The behavioral analysis helps identify spending patterns and areas for cost reduction. They use the interest calculator to evaluate different financing options for equipment purchases, comparing the total cost of loans versus leasing arrangements. The state save and load features ensure that all financial data persists between sessions, creating a reliable financial record.

Investment Advisory Application

Financial advisors utilize FinTerm to provide quick calculations for clients during consultations. They create temporary profiles for different client scenarios, using the portfolio recommendation system to demonstrate various allocation strategies based on risk profiles. The CAGR calculator helps explain historical performance, while the compound interest calculator shows the power of long-term investing. The loan calculator assists clients in understanding the true cost of debt and how different payment strategies affect total interest paid.

Understanding the Output Format

FinTerm returns all results in JSON format, providing structured data that can be easily parsed and integrated with other systems. Success responses include a status field set to "success" along with relevant data fields specific to each operation. For calculations, the output includes clearly labeled results with appropriate precision. Transaction processing returns the new balance and a transaction ID for reference. User management operations provide confirmation details including counts and affected resources.

Error responses maintain the same JSON structure with a status field set to "error" and a descriptive message field explaining the issue. This consistent format enables robust error handling and clear communication of issues to users. The system validates inputs and provides specific error messages for common issues such as insufficient funds, missing users, or invalid parameters.

Further Tips

Maintaining organized user profiles enhances the system's utility for tracking multiple financial scenarios or managing different accounts. Create distinct user IDs that clearly identify their purpose, such as "personal_checking" or "business_savings", making it easier to manage multiple profiles.

Regular use of the state save feature ensures data persistence and provides backup capability. Consider saving state after significant transactions or at the end of each session to prevent data loss. The load feature can quickly restore your entire financial environment, making it easy to resume work exactly where you left off.

When processing transactions, use descriptive categories and descriptions to enhance the value of behavioral analysis. Consistent categorization enables more meaningful insights into spending patterns and helps identify trends over time. The system maintains complete transaction history, so detailed descriptions provide valuable context for future reference.

For complex financial decisions, combine multiple calculator features to gain comprehensive insights. For example, use the loan calculator to understand payment obligations, then use the compound interest calculator to evaluate whether investing available funds might generate returns exceeding the loan interest rate.

The portfolio recommendation system provides starting points for investment allocation, but these should be adjusted based on individual circumstances and goals. Use the different risk profiles to understand how allocation strategies change with risk tolerance, helping inform investment decisions.

Quick Reference Card

Essential Commands

System Control

- Launch: `python3 core.py`
- Exit: `exit` or `quit` or `q`
- Help: `help` or `h` or `?`

Calculations

- Interest: `calculate interest --principal [amount] --rate [rate] --time [years] --type [simple|compound]`
- Loan: `calculate loan --principal [amount] --rate [rate] --years [years]`
- CAGR: `calculate cagr --initial [amount] --final [amount] --years [years]`

User Management

- Create: `user create --user_id [id] --name "[full name]"`
- Delete: `user delete --user_id [id]`
- List: `user list`

Transactions

- Process: `transaction process --user_id [id] --amount [amount] --category [category] --description "[description]"`

Investment

- Recommend: `investment recommend --risk_profile [conservative|moderate|aggressive] --amount [amount]`

Session

- Save: `state save`
- Load: `state load`

Analysis

- Behavioral: `behavior analyze --user_id [id]`

Synthesis

All monetary amounts are entered as decimal numbers without currency symbols. Interest rates are entered as decimals (for example, 0.05 for 5%). Time periods for interest calculations are specified in years and can include decimal values for partial years. Transaction amounts can be positive for deposits, negative for withdrawals, or zero for balance inquiries. User IDs should be alphanumeric with underscores permitted, while names and descriptions should be enclosed in quotes if they contain spaces.

The system processes commands sequentially and maintains state throughout the session, allowing for intermediate financial workflows and analysis. Each command returns immediate feedback in JSON format, ensuring clear communication of results and any errors that may occur during processing.