

Iteration 3 Report

Project Title: Stock Price Dashboard

Team Members: Boris Gordeev, Inshaal Chaudhury

Project Leader (This Week): Boris Gordeev

Instructor / Mentor: Fatima Nafa

Course: EECE2140 – Fall 2025

Submission Date: 2025-11-04

Repository Link: <https://github.com/gordeevb/stock-price-dashboard>

Contents

1	Summary	2
2	Deliverables Checklist (This Iteration)	2
3	Technical stack	2
4	Project structure	2
5	Core logic	3
6	Leadership Rotation (This Week)	4
7	Tuesday Check-In: Status Report	4
8	Thursday Peer Review – Presentation & Feedback	4
9	Sunday Deadline – Core Features Completion	5

1 Summary

This project fetches historical stock market data, performs core financial analysis, and produces publication-ready charts (price with moving averages, volume, rolling volatility, and daily returns distribution). The codebase is split into modules:

- `fetch_data.py` handles downloading, caching, and parameter validation.
- `analyze.py` computes returns, cumulative return, moving averages, volatility, volume and money-flow metrics, and summary statistics.
- `plot.py` draws consistent charts and applies a unified visual style.
- `main.py` is the user-facing entry point that orchestrates data fetch, analysis, and plotting.
- `config.py` centralizes defaults and tunable settings.

2 Deliverables Checklist (This Iteration)

- Functional core system architecture (end-to-end path demonstrated)
- Project Leader assigned and duties executed
- Status report (Tue check-in) with documented action items
- Peer review presentation + feedback summary & improvement plan
- All core features completed and functional (Sun deadline)
- Feature checklist compiled + testing coverage ensured
- Known issues recorded and prioritized for next phase

3 Technical stack

- Python 3.8
- `yfinance` (Yahoo Finance API wrapper)
- `pandas` (data manipulation and analysis)
- `numpy` (numerical calculations)
- `matplotlib` (data visualization)

4 Project structure

- `main.py` : application entry point and orchestration
- `fetch_data.py` : data download, caching, retries, input validation
- `analyze.py` : financial computations and summary statistics
- `plot.py` : chart utilities and style setup
- `config.py` : settings and defaults

5 Core logic

Input

A user provides a stock ticker (for example AAPL) and optional parameters such as period and interval.

Acquisition (fetch_data.py)

1. a. Validate ticker and parameters.
2. b. Check local cache for fresh data; otherwise download with yfinance.
3. c. Normalize columns and index.
4. d. Raise a DataFetchError on network or validation failures.
5. e. Optionally apply retry logic with exponential backoff.

Analysis (analyze.py)

1. a. Compute simple returns and log returns where needed.
2. b. Compute cumulative return series.
3. c. Compute moving averages for the configured windows.
4. d. Compute rolling volatility (for example standard deviation of daily returns).
5. e. Compute Chaikin money-flow indicator.
6. f. Generate summary statistics (date span, min, max, drawdown proxies, etc.).
7. g. Raise an AnalysisError if required columns are missing or input is invalid.

Visualization (plot.py)

1. a. Apply a consistent matplotlib style via a style setup function.
2. b. Draw a price chart, overlaying moving averages.
3. c. Draw a volume bar chart aligned with price.
4. d. Draw a rolling volatility chart for the selected window.
5. e. Draw a daily returns histogram or density view.
6. f. Raise a PlotError when inputs are invalid.

Orchestration (main.py)

1. a. Parse interactive inputs or read defaults from config.py.
2. b. Call fetch data to obtain a DataFrame.
3. c. Call analyze functions to compute metrics and summary statistics.
4. d. Call plot functions to render the four charts.
5. e. Display the figures on screen and optionally save them to disk.

6 Leadership Rotation (This Week)

Project Leader: Boris

Scope Executed:

- Coordinated daily progress: Held brief daily check-ins to track module completion + schedule meetings, and clear blockers (Matplotlib Integration issues)
- Ensured task distribution: Assigned group (Inshaal) to handle Yahoo Finance Data retrieval and initial visualization setup; Boris was to finalize data processing logic and manage Pandas table formatting
- Maintained team communication: Used a shared GitHub Repository workspace for progress tracking and scheduled meetings to ensure iteration completion by deadline

Reflection:

- Consistent communication and meetings on scheduled days helped to maintain progress and meet the core development deadline
- Clear task division reduced overlap and confusion between modules
- Daily check-ins can be shortened and more focused on blockers and issues with data

7 Tuesday Check-In: Status Report

Progress Since Last Update

The code for config (central configuration file), fetch_data.py (fetching financial data) and analyze.py (mathematical analysis) were written. The files lack appropriate comments and some functions still do not have logger code implemented.

The project still lacks a complete file that creates plots using pandas, testing, and a UI interface. Also, main.py (files which acts as the main starting point and combines functionality of all other files together) still does not have the ability to display multiple charts.

Next Steps (Through Thursday)

- Complete main.py
- Complete plot.py (creation of pandas tables) file
- Add comments throughout fetch_data, analyze, plot and main
- Complete testing pytest files

8 Thursday Peer Review – Presentation & Feedback

Feedback Summary

All core components (fetch_data.py, analyze.py, plot.py, main.py, config.py) were completed and work without bugs.

Improvement Plan

- Start writing testing pytest files

Testing And Debugging Initial Results

- Tests: ran AAPL and MSFT on intervals 1d and 1wk. Verified price, volume, volatility, and returns charts rendered. Confirmed caching layer creates and reuses files. Verified error path by forcing an invalid ticker.
- Planned unit tests: pytest modules for parameter validation in fetch_data, analysis computations (returns, volatility, moving averages), and plotting input guards.
- Coverage status: tests in progress, to be measured with pytest-cov, coverage not yet recorded.

Challenges Faced & How We Addressed Them

1. Challenge: Matplotlib style conflicts caused chart overlays to render incorrectly on some intervals.
Solution: Standardized a single style initializer and validated figure lifecycles.
2. Challenge: yfinance rate limits and timeouts
Solution: Implemented on-disk caching, reduced repeated API calls and failures.

9 Sunday Deadline – Core Features Completion

Completed core features

Component	Description	Key Functions	Status
Data Retrieval Module (fetch_data.py)	Fetches stock data from Yahoo Finance via yfinance. Implements local caching, validation, and retries to reduce API calls.	fetch_stock_data; retry_on_failure; validate_ticker; validate_parameters; get_cache_filename	Complete
Analysis Module (analyze.py)	Calculates technical indicators and statistical metrics.	calculate_returns; calculate_cumulative_return; calculate_moving_average; calculate_volatility; calculate_volume_metrics	Complete
Visualization Module (plot.py)	Creates professional charts using matplotlib. Generates four separate full-size figures.	setup_plot_style; plot_price_chart; plot_volume_chart; plot_volatility; plot_returns_distribution	Complete
User Interface Module (main.py)	Provides direct analysis mode with automatic chart generation and summary output.	run_interactive_mode; main; get_user_input; display_multiple_charts	Complete
Configuration (config.py)	Settings for the entire application, including defaults and styling.	BASE_DIR; SRC_DIR; DATA_DIR; CACHE_DIR; LOG_DIR	Complete

Issues and future improvements:

- EMA indicator on the plot is inconsistent on some timeframe combinations
- pytest files yet to be completed
- Streamlit UI interface yet to be completed

Statement by the Individual Submitter

I, Boris Gordeev, confirm that the above table accurately reflects my personal contributions during Iteration 3.