Group 4 Proposal: Stock Trends Analysis Dashboard

Background

In today's job market, individual investors are curious about stocks and want to analyze them, but turning raw price data into useful insights is challenging. Many users and students rely on spreadsheets or basic charts, which are ineffective or time-consuming when they want to compute indicators (moving averages, volatility, returns), compare time windows, or evaluate whether a trend-based strategy would have worked historically. This process becomes inefficient when analysis needs to be repeated for different stocks, because the steps are often manual and inconsistent.

Numerous finance platforms and charting tools exist (examples: trading/chart websites and brokerage dashboards), and they are useful, but they often fall short for learning and project work in two key ways: (1) they hide the data pipeline, so users cannot see or control how results are computed, and (2) they are not designed around a simple "bring your own dataset" workflow, where a student can upload a CSV, validate the structure, and generate an analysis dashboard that is repeatable and easy to present. In classroom settings, this gap matters because students need something that is transparent, reproducible, and aligned with what they learn in data analytics – cleaning data, computing features, and presenting results clearly.

Academic work in financial time-series analysis and visual analytics shows that interactive visual exploration helps users understand time-series behavior, compare regimes, and interpret model outputs more effectively than static graphs. However, there is still a gap between research-level analytics ideas and a simple, usable dashboard that takes a standard dataset input and produces actionable, beginner-friendly outputs.

Project Goals and Objectives

The goal of this project is to build an interactive Stock Trends Analysis Dashboard that takes a stock CSV as input and generates an easy-to-understand dashboard with analysis, strategy backtesting, and ML-based trend classification.

To successfully implement the project, we will complete the following objectives:

- Accept a stock CSV upload in a required format and validate it automatically.
- Clean/process the data and compute key indicators (returns, moving averages, volatility).
- Visualize stock performance with interactive charts (price, volume, indicators).
- Generate "trend labels" (invest and no-invest) using a simple labeling logic and show the results.
- Run a basic backtest comparing a label-based strategy vs buy-and-hold and show results clearly.

- Run one or two traditional ML models such as SVM and Random Forest and show performance metrics (accuracy, confusion matrix, etc.).
- Keep the UI simple, clean, and user-friendly.

Major Functionality

The dashboard's major functionality is to allow users to upload historical stock data in CSV format, have the data automatically processed and validated, and output an interactive analytical dashboard for financial exploration and plan review.

Primary user flow

The system enables the following workflow:

- **Data upload:** a user uploads a CSV file using the dashboard upload feature.
- **Data validation and processing:** the system verifies the presence of required columns, handles missing values, sorts by date, and confirms the valid date range to ensure data integrity.
- **Exploration page:** interactive charts are provided to examine stock behavior, such as closing prices, candlestick views (optional), trading volume, daily returns, and rolling volatility.
- **Indicators page:** the dashboard computes and displays commonly used indicators such as moving averages (20/50/200 days), drawdown metrics, and weekly or monthly returns.
- **Label generation page:** dashboard generates Green/Red (invest/no-invest) investment labels using a simple rule-based approach like the one already used in the repo.
- **Backtesting page:** a user can simulate a basic trading strategy (enter on Green, exit on Red) and compare it with buy-and-hold, showing final returns and equity curves.
- **Predictive modelling page:** run a selected ML classifier (SVM, Random Forest, AdaBoost etc.) to generate and display performance metrics such as accuracy, confusion matrix with predicted labels on the chart.
- **Export report (optional):** download a short summary report (PDF or HTML snapshot) for documentation and further analysis.

Data requirements

The system requires stock data in CSV format. Each record represents a single row per trading day and must include Date, Open, High, Low, Close, and Volume fields. Optional features like Adj Close and Ticker may be included to improve analysis and labeling.

Technologies

- **Backend + Dashboard Framework:** Django – used to build a robust backend system responsible for business logic, API management, and data handling.

- **Django REST Framework (DRF):** will be used to expose clean REST APIs.
- **Frontend:** React – for building a modern, component-based user interface.
- **Vite:** fast development server and build tool for improved performance and developer experience.
- **Data Processing:** Pandas, NumPy (cleaning, feature creation, indicators).
- **Visualization:** Plotly (interactive charts: hover, zoom, range sliders).
- **ML:** scikit-learn (SVM, Random Forest, AdaBoost, KMeans).
- **Data Source (optional):** yfinance only as a "helper" if the user doesn't have CSV.
- **Version Control:** Git/GitHub for teamwork and code review.

Project Roles and Responsibilities

Each team member takes clear ownership of their primary responsibilities while also cooperating across modules to provide assistance with testing, documentation, and interface enhancements as needed.

- **Project Manager:** Responsible for project planning, timeline coordination, organizing team meetings, task tracking, reporting progress, and preparing final presentations and documentation.
- **Dashboard/UI Developer:** Develops the user interface using React and Vite, including page layout, navigation, interactive components, data visualization integration, and overall user experience design.
- **Data/Backend Developer:** Implements backend services using Django and Django REST Framework, including API development, business logic, authentication, database management, and data processing pipelines.
- **ML/QA Engineer:** Develops and integrates machine learning modules, validates model performance, writes automated tests, verifies data accuracy, and tests system edge cases.

The following table illustrates the distribution of roles and responsibilities within the team:

	Ulzhalgas Seidaliyeva	**Kuanysh Amandos**	**Haoqian Zhang**	**Atharva Sharma**
Project Manager	Primary	-	-	-
Dashboard/UI Dev	Contributing	Primary	Primary	Contributing
Data/Backend Dev	Contributing	Primary	Primary	Contributing
ML/QA Engineer	Contributing	Contributing	Contributing	Primary