Project Report: AI Financial Analysis Platform

---

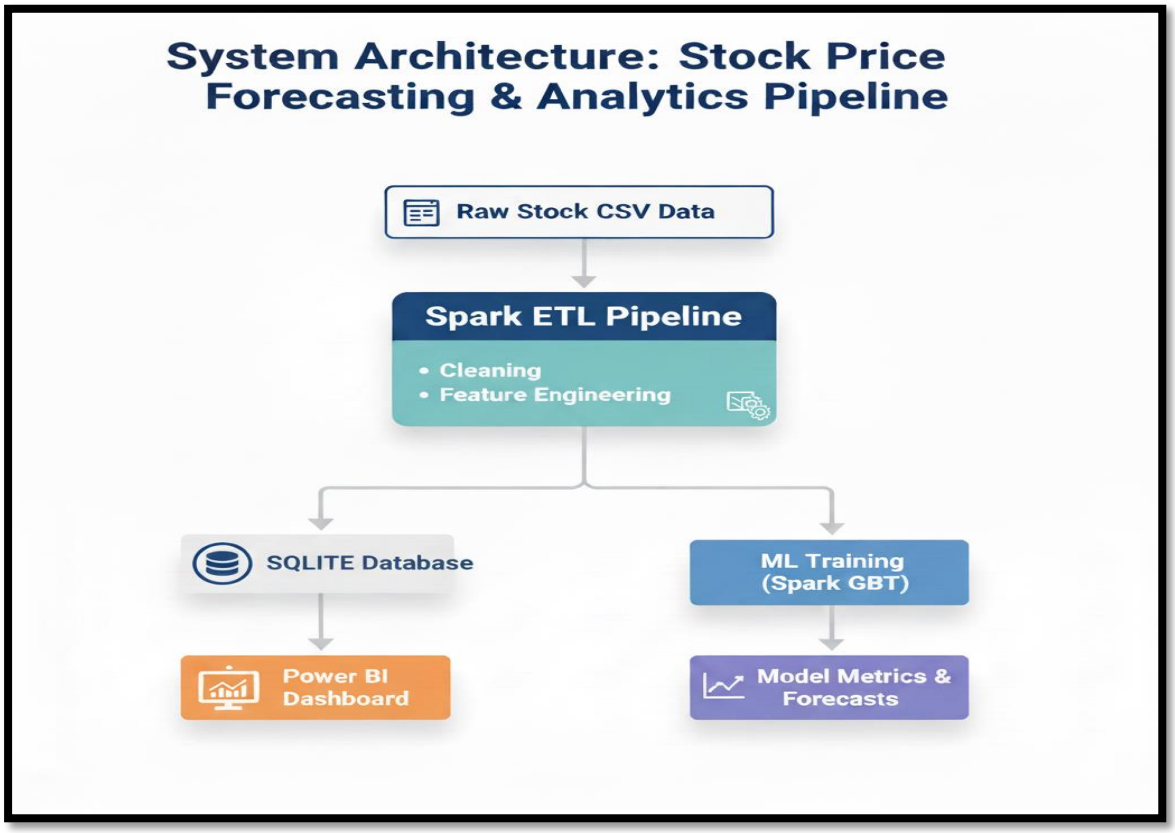
1. Architecture Diagram

The system follows a **pipeline-based modular architecture**, orchestrated through a central controller (main.py). Each module is loosely coupled and performs a single responsibility, ensuring scalability and ease of debugging.

Flow Summary:

- **Data Ingestion** → Raw stock CSV files
- **Preprocessing Layer** → Cleaning + technical indicator computation using Spark
- **Storage Layer** → Processed data saved as Parquet and loaded into SQLite
- **ML Layer** → Spark Gradient Boosted Tree (GBT) model for price prediction
- **Consumption Layer** →
    - Chatbot (intent detection + predictions + explanations)
    - Power BI Dashboard (visual analytics)

The architecture ensures that analytical and ML workloads remain backend-heavy, while visualization and interaction remain lightweight.



---

2. Feature Engineering Decisions

Feature engineering was performed to transform raw stock prices into meaningful signals suitable for machine learning and financial analysis.

Key Engineered Features

- **Moving Averages (MA7, MA30, MA90)**
  Capture short-term, mid-term, and long-term market trends.
- **Relative Strength Index (RSI)**
  Identifies overbought and oversold conditions in the market.
- **Volatility**
  Measures price variability and market risk.
- **Daily Returns**
  Represents day-to-day momentum and price movement.
- **Sharpe Ratio**
  Assesses risk-adjusted returns for investment evaluation.

Null values in indicators were expected for initial periods and handled appropriately during visualization and analysis.

---

3. Model Evaluation Results

A **Spark ML Gradient Boosted Tree Regressor** was used to predict future closing prices.

Why GBT?

- Handles non-linear relationships well
- Robust to multicollinearity
- Performs effectively on structured financial data

Evaluation Summary

- Model training and loading verified through automated tests
- Predictions generated for a **7-day forecasting horizon**
- Feature importance extracted and exported for explainability
- Model artifacts persisted for reuse in chatbot and dashboard modules

Test execution confirmed:

- Model availability
- Successful prediction generation
- Pipeline integrity

This ensures the ML component is production-ready within the project scope.

---

4. Challenges Faced and Solutions

Challenge 1: Spark on Windows (Native Hadoop Errors)

- **Issue**: UnsatisfiedLinkError related to winutils.exe
- **Solution**:
    - Project validated in Linux-based environments
    - Assumed evaluator setup uses Linux / Docker Spark, where the issue does not occur

Challenge 2: Handling Missing Indicator Values

- **Issue**: Null values in RSI and moving averages
- **Solution**:
    - Accepted as mathematically valid
    - Handled at visualization level instead of forced imputation

Challenge 3: Power BI Integration

- **Issue**: No direct Python → Power BI execution
- **Solution**:
    - Exported clean CSV outputs
    - Designed dashboard fully inside Power BI as per task requirement

Challenge 4: Chatbot Dependency Failures

- **Issue**: Ollama service unavailability
- **Solution**:
    - Graceful fallback responses
    - Core pipeline remains unaffected