FINROBOT:An OPEN SOURCE AI AGENT PLATFORM

Team:

PS-G630

Team Members:

B. Anish, K. Manognya, T. Nikith, V. Saibharadhwaj, Y. Koushik

Mentor:

Vijaynag

Presentation by:

Sai Bharadhwaj

FinRobot: An Open-Source AI Agent Platform for Financial Analysis

FinRobot is an open-source AI agent platform designed to transform financial analysis and decision-making by leveraging multiple advanced Large Language Models (LLMs) through a modular, layered architecture.

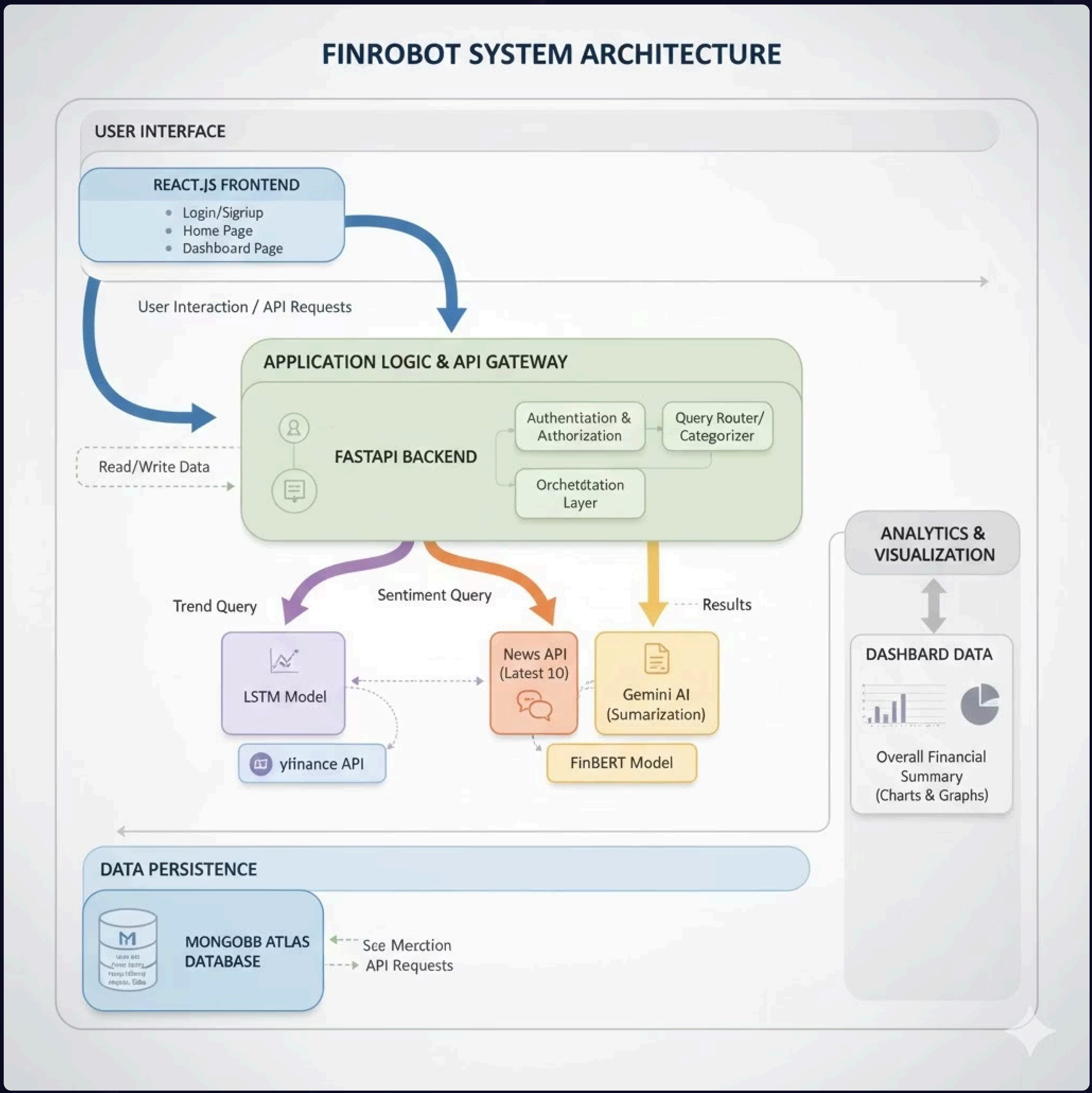
Purpose and Motivation

FinRobot bridges the gap between proprietary financial data and specialized AI knowledge, making sophisticated financial analytics accessible via open-source initiatives. It addresses the complexity and domain barriers faced by financial professionals integrating LLMs.

Key Technical Contributions

- Smart Scheduler for LLM selection.
- Financial Chain-of-Thought (CoT) Prompting for transparency.
- Retrieval Augmented Generation (RAG) for contextual data analysis.
- Multi-agent Workflow for task orchestration.

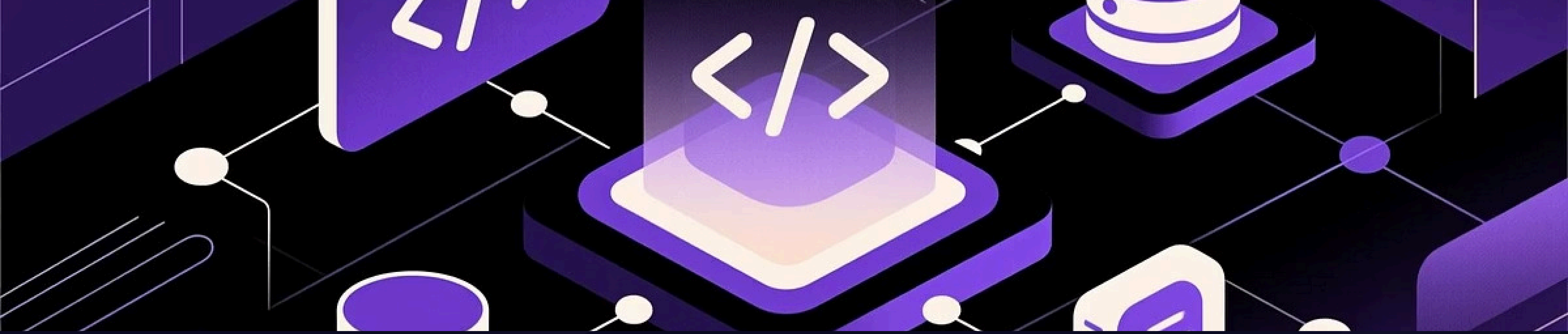
FinRobot System Architecture



The FinRobot platform follows a layered architecture with clear separation of concerns:

Key Components:

- **User Interface Layer:** React.js frontend with Login, Home, and Dashboard pages
- **Application Logic & API Gateway:** FastAPI backend handling authentication, authorization, query routing, and orchestration
- **AI Processing Layer:** Integration of multiple AI models including LSTM Model, News API (Hugging Face), FinBERT Model, and Gemini AI for summarization
- **Analytics & Visualization:** Dashboard data with overall financial summary, charts, and graphs
- **Data Persistence:** MongoDB Atlas database for storing user data and API requests
- **External APIs:** yfinance API for real-time market data



Technology Stack: Building a Robust Financial Platform

Frontend

Developed using **React JS** for a dynamic and responsive user interface.

Backend

Built with **FastAPI**, ensuring high performance and rapid development of APIs.

Key Libraries and Frameworks

A diverse set of tools powers data handling, LLM integration, deep learning, and security:

- **Web & Data:** FastAPI, Uvicorn, Pydantic, **yfinance**, Pandas, NumPy, Requests.
- **AI/ML:** scikit-learn, **TensorFlow**, **Keras**, **Transformers**, **Torch**.
- **LLM & NLP: LangChain**, langchain-google-genai, google-generativeai.
- **Document Processing:** PyMuPDF, pdf2image, pytesseract.
- **Security & Config:** python-dotenv, motor, passlib, bcrypt.

Development Environment and Deployment Strategy

Development Tools Utilised

- Initial AI model testing was performed using **Google Colab** for efficient GPU access.
- Primary development for the frontend and backend was conducted using **VS Code**, leveraging its integrated environment and extensive extensions.

Database Selection

We employed **MongoDB Atlas** as the non-relational database solution, offering flexibility and scalability for storing dynamic financial data and user information.

Deployment Platforms

Frontend

Deployed via **Vercel** for speed and easy continuous deployment.

Backend

Deployed on **Hugging Face Spaces**, offering a flexible environment suitable for AI/ML-heavy applications.

Mathematical Foundation: Core Formulas Driving FinRobot

FinRobot integrates sophisticated mathematical models from Language Models, Reinforcement Learning, and Quantitative Finance.

FinGPT Language Modeling Objective

The model is trained using Supervised Fine-Tuning (SFT) to maximise the probability of generating correct financial responses by minimizing the Negative Log-Likelihood (NLL):

$$L_{\text{CausalLM}} = -\sum_{t=1}^{T} \log P(w_t \mid w_1, w_2, \ldots, w_{t-1}; \theta)$$

FinRL: Reinforcement Learning for Trading

Trading is modeled as an MDP. The objective is to find a policy π_θ^* that maximizes the expected discounted reward $J(\pi_\theta)$ (total returns while reducing risk):

$$\pi_\theta^* = \arg\max_\theta J(\pi_\theta) \quad \text{where} \quad J(\pi_\theta) = E\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, s_{t+1})\right]$$

Financial Machine Learning: Log Returns

Log-returns are used for better risk-adjusted comparisons between stocks, aiding in volatility analysis and trend prediction:

$$r_{T+f,i} = \log\left(\frac{S_{T+f,i}}{S_{T,i}}\right)$$

Financial Multimodal LLM Fusion

The model processes and fuses representations from text (x_t), graphical (x_g), and tabular data (x_h) to reason like an analyst:

$$F(x_t, x_g, x_h) = L(T(x_t),\ G(x_g),\ H(x_h))$$

Operationalising AI: Algorithms and Model Parameters

The Smart Scheduler

This component automatically selects the optimal LLM for a given financial task, boosting accuracy and efficiency based on a weighted performance score.

Weighted Score Calculation:

$$\text{Score} = \sum_{k=1}^{n} w_k \cdot \hat{m}_k$$

Where $\hat{m}_k$ is the normalised metric (e.g., accuracy, latency) and w_k is its weight.

Significance of Key Parameters

FinGPT (LLMs)	Token embeddings, θ	Controls language understanding and generation of financial text.
FinRL (RL Agent)	Policy parameters, γ (discount factor)	Determines trading behavior and risk appetite for maximizing returns.
Multimodal LLM	Text/Table/Chart embedding dimensions	Enables the model to reason comprehensively by fusing diverse data types.

Key Insights and Learnings from Development

The project provided deep learning experience in integrating various financial technologies and data sources.

Sentiment Analysis and Preprocessing

Mastered sentiment analysis using the FinBERT model and understood the necessity of normalizing stock data and news articles for models like LSTM and FinBERT.

Real-World Application

Gained experience in developing real-time web applications, managing deployments, and integrating diverse LLMs and deep learning models in practical scenarios.

Real-World Applications

Automated Stock Trend Prediction

Sentiment-Based Market Analysis

Financial Document Summarization

Personal Investment Assistance

Future Roadmap: Extensions and Potential Improvements

Advanced Trading Mechanics

Extend support for **intraday trading (day trading)** and integrate a PPO-based portfolio management system for automated decision-making.

Real-Time Data Integration

Connect with paid, low-latency stock-market APIs (e.g., Zerodha Kite, Angel One SmartAPI) for higher accuracy and real-time market data ingestion.

Enhanced LLM Capabilities

Fine-tune advanced LLMs (like Gemini or ChatGPT) on financial datasets for smarter trading suggestions, risk analysis, and automated strategy generation.

Goal:

These additions will make the system more practical for active traders and individuals keen on algorithmic trading, solidifying FinRobot's utility in the real world.

Performance and Applicability to the Indian Market

Resource Efficiency vs. Research Paper

> The original paper focuses on American/Chinese markets using heavy, resource-intensive models. Our project is designed specifically for **Indian market data**, making it highly applicable for local traders. We use efficient technologies (yfinance, FinBERT, LSTM) suitable for moderate hardware and real-time use.

Dynamic Training Approach

Instead of a single model for all stocks, the LSTM model trains dynamically on the user-selected stock. This customized approach ensures **optimal prediction accuracy** for individual stocks.

Performance Analysis

90%

Accuracy for the Sentiment Analysis module (highly reliable based on practical testing).

Variable

Trend Prediction Accuracy (varies by stock; improved by dynamic training).

For financial document analysis and summarization, the **Gemini AI model** is used, providing fast and accurate text extraction and summary generation.

Limitations of the Implementation

Real-time Data Constraints

Yfinance data fetching is slow with low frequency updates, which affects the speed of prediction and responsiveness.

Hardware Limitations

The LSTM model runs on a CPU instead of a GPU, making real-time predictions slower than optimal.

Deployment Constraints

The backend is deployed on Hugging Face CPU infrastructure due to the high cost associated with GPU deployments.

LLM Access Limitations

Originally, multiple LLMs (ChatGPT, Gemini) were planned, but cost restrictions limited the project to a single Gemini AI model.

Single LLM Dependency

Reliance on a single LLM reduces reliability and primarily limits its use to document analysis rather than broader capabilities.

Conclusion :Despite these limitations, the project successfully demonstrates financial prediction using cost-effective tools and innovative techniques.