**🧩**

**Step-by-Step Tech Stack and Tool Suggestions**

**✅**

**Step 1: Define MVP Scope**

* **Planning Tools**:
  + Notion / Trello / GitHub Projects (to track milestones)
  + draw.io / Whimsical (for diagrams)
  + Microsoft Word / Google Docs (for SRS)

**✅**

**Step 2: Build Core Backend**

* **Languages**: Python
* **Frameworks**:
  + [FastAPI](https://fastapi.tiangolo.com/) *(modern, async-friendly)*
  + [Flask](https://flask.palletsprojects.com/) *(simpler for smaller apps)*
* **Stock Price API Libraries**:
  + [yfinance](https://pypi.org/project/yfinance/) *(for Yahoo Finance data)*
  + [alpha\_vantage](https://github.com/RomelTorres/alpha_vantage)
  + [finnhub-python](https://github.com/Finnhub-Stock-API/finnhub-python)
  + [iexfinance](https://github.com/addisonlynch/iexfinance)
* **Other Tools**:
  + uvicorn (FastAPI server)
  + httpx or requests for async API calls

**✅**

**Step 3: Implement Alert System**

* **Scheduling Tools**:
  + [schedule](https://pypi.org/project/schedule/)
  + [apscheduler](https://pypi.org/project/APScheduler/)
  + asyncio (native async scheduling for FastAPI)
* **Notification Options**:
  + Console alerts (initial phase)
  + Desktop notifications via plyer
  + Slack or Discord webhook integration (for demo)

**✅**

**Step 4: Integrate Simple Gradio UI**

* **Frontend Library**:
  + [Gradio](https://gradio.app/) (perfect for testing and fast UI prototyping)
* **Optional Alternatives**:
  + Streamlit (another minimal Python UI library)

**✅**

**Step 5: Add AI-Powered News Summarization**

* **News Sources / APIs**:
  + [NewsAPI.org](https://newsapi.org/)
  + [Finnhub News](https://finnhub.io/docs/api#company-news)
  + RSS Feeds (e.g., Google News via feedparser)
  + Web scraping with BeautifulSoup (fallback only)
* **AI/NLP Libraries**:
  + [OpenAI API](https://platform.openai.com/) (e.g., GPT-3.5, GPT-4)
  + [LangChain](https://github.com/hwchase17/langchain) (for reasoning pipelines)
  + [HuggingFace Transformers](https://huggingface.co/transformers/) (summarization models)
* **Text Processing**:
  + spaCy or nltk for entity detection or sentiment analysis

**✅**

**Step 6: Paper Trading & Performance Logging**

* **Storage**:
  + SQLite or TinyDB (lightweight DB)
  + CSV logging via pandas
* **Trade Simulation Logic**:
  + Custom logic + moving average heuristics
  + Track buy/sell actions, timestamps, prices, profit/loss
* **Optional**:
  + Matplotlib/Plotly for performance charts

**✅**

**Step 7: Optional React Frontend**

* **Frontend Framework**:
  + React (with Vite or Create React App)
  + Tailwind CSS (for styling)
* **Charts & UI Components**:
  + Chart.js / Recharts
  + React Query / Axios (for fetching backend data)
* **Auth (if added)**:
  + Firebase Auth
  + Auth0
  + Clerk.dev

**✅**

**Step 8: Deployment & Monitoring**

* **Hosting**:
  + [Render](https://render.com/) / [Railway](https://railway.app/) *(great for FastAPI/Flask)*
  + Vercel or Netlify (for frontend)
* **Background Workers**:
  + Render cron jobs
  + GitHub Actions (scheduled workflows)
* **Monitoring Tools**:
  + UptimeRobot (simple uptime ping)
  + Sentry (for error monitoring)

**✅**

**Step 9: Future Enhancements**

* **Real Trading APIs**:
  + [Alpaca API](https://alpaca.markets/) *(paper trading supported)*
  + Robinhood API (unofficial, not recommended for production)
* **ML / RL Exploration**:
  + stable-baselines3 (for reinforcement learning simulation)
  + TensorFlow/Keras/PyTorch (for more advanced strategies)
* **Mobile Frameworks** (long-term):
  + React Native
  + Flutter