# Quantitative Finance Tech Stack

Quantitative Finance Management webapp and ios app the ml model that predict the assets like gold,stocksand crpyto that can intelligently predicts and tell what to buy and when to buy a stock or assest

Full Tech Stack – Quantitative Finance Management App

Machine Learning & Prediction

- Python – Core language for data processing and modeling
- Pandas & NumPy – Data wrangling and numerical computations
- Scikit-learn – Classical ML models (e.g., Random Forests, XGBoost, etc.)
- XGBoost / LightGBM – Boosted trees for asset return prediction
- Prophet / ARIMA / LSTM (Keras/TensorFlow) – Time series forecasting
- TA-Lib – Technical analysis indicators (RSI, MACD, Bollinger Bands)

Data Collection & ETL

- yfinance / Alpha Vantage / CoinGecko APIs – Financial and crypto price data
- BeautifulSoup / Requests – Web scraping economic indicators
- Cron + Python scripts – Scheduled ETL jobs

Model Evaluation & Experimentation

- Jupyter Notebooks / Google Colab – Exploratory Data Analysis (EDA), backtesting
- Matplotlib / Seaborn / Plotly – Data visualization and performance metrics

Backend (Web API & Core Logic)

- FastAPI – REST API for model inference and user interaction
- Uvicorn / Gunicorn – ASGI server for FastAPI
- SQLAlchemy – ORM for database access
- PostgreSQL / SQLite – Relational database for user data, asset history

Frontend

Web App (Dashboard):

- o Streamlit – Rapid prototyping and visualization dashboard
- o or React.js – For a production-ready, responsive frontend
- o Chart.js / Recharts – Asset chart visualization

iOS App:

- o React Native – Cross-platform mobile app development
- o Expo – Toolchain for fast app builds and testing
- o Axios – API communication from frontend to backend

Deployment & DevOps

- Render / Heroku / Railway – Fast deployment of FastAPI backend
- Firebase – For iOS app backend (alternative), real-time database and auth
- Docker – Containerizing the backend for portability
- GitHub Actions – CI/CD for testing and deployment
- AWS S3 / Cloudflare – Static file storage or CDN (if needed)

Security & Auth

- OAuth2 / JWT Tokens – User authentication
- HTTPS / CORS – Secure and controlled access
- Environment Variables (.env) – API keys and secrets management

Thank you! Here's a cleaned-up, professional, and **comprehensive tech stack breakdown** for your **Quantitative Finance Management Web & iOS App**, which includes a **machine learning engine to predict assets like gold, stocks, and crypto**, advising users on **what to buy and when**.

## ✅ Quantitative Finance Management App – Full Tech Stack

### 🎯 Goal

Build a cross-platform platform (web + iOS) that predicts financial asset movements using machine learning and technical indicators, then recommends actions like "Buy", "Hold", or "Sell".

## 🔍 1. Machine Learning & Asset Prediction Engine

| Layer | Tech | Purpose |
|---|---|---|
| Language | **Python** | Core language for all data science and backend logic |
| Libraries | **Pandas, NumPy** | Data manipulation & numerical computing |
| ML Models | **Scikit-learn** | Baseline ML models (Random Forest, SVM, etc.) |
| Advanced Models | **XGBoost, LightGBM** | Gradient boosting models for regression/classification |
| Time Series | **Facebook Prophet, ARIMA, LSTM (Keras/TensorFlow)** | Predict asset prices over time |
| Technical Indicators | **TA-Lib** | Calculate RSI, MACD, Bollinger Bands, SMA, EMA etc. |

## 🧪 2. Data Collection & ETL Pipeline

| Tool | Use |
|---|---|
| **yfinance, Alpha Vantage, CoinGecko APIs** | Real-time and historical market data (stocks, forex, crypto) |
| **Requests, BeautifulSoup** | Scrape additional economic or news data |
| **Python Scripts + Cron Jobs** | Scheduled daily/weekly ETL workflows |
| **SQLite / PostgreSQL** | Store cleaned data for model input |

## 📊 3. Model Evaluation & Experimentation

| Tool | Use |
|---|---|
| **Jupyter Notebooks / Google Colab** | Exploratory Data Analysis (EDA), backtesting, visualization |
| **Matplotlib, Seaborn, Plotly** | Asset performance, strategy comparison charts |

| Tool | Use |
|---|---|
| **Backtrader / Zipline (optional)** | Strategy backtesting simulation |

## 🖥️ 4. Backend (API & Core Logic)

| Component | Tech |
|---|---|
| API Framework | **FastAPI** – High-performance, async REST API |
| Server | **Uvicorn / Gunicorn** – ASGI-compliant servers |
| ORM | **SQLAlchemy** – Object-Relational Mapping |
| Database | **PostgreSQL / SQLite** – Asset prices, predictions, user data |

## 🌐 5. Frontend Interfaces

### 🔹 Web Dashboard (Prototyping / Internal Use)

| Tool | Purpose |
|---|---|
| **Streamlit** | Quick, interactive dashboards for data scientists |
| **Chart.js / Plotly / Recharts** | Charting historical prices, indicators, predictions |

### 🔸 Web App (Production)

| Tool | Purpose |
|---|---|
| **React.js** | Frontend framework for interactive UI |
| **Axios** | Connects React frontend to FastAPI backend |

### 📱 iOS App (Cross-Platform)

| Tool | Purpose |
|---|---|
| **React Native** | Build native-like mobile apps using JS/TS |
| **Expo** | Simplifies development and deployment of mobile apps |
| **Axios** | API communication layer |

## 🚀 6. Deployment & DevOps

| Service | Purpose |
|---|---|
| **Render / Railway / Heroku** | Deploy FastAPI backend |
| **Docker** | Containerize backend for reproducibility |
| **GitHub Actions** | CI/CD pipeline for testing & automatic deploys |
| **Firebase** | Real-time database & auth (optional for mobile) |
| **AWS S3 / Cloudflare** | Store and serve static assets / CDN acceleration |

## 🔐 7. Security & Authentication

| Tool | Use |
|---|---|
| **OAuth2 / JWT Tokens** | Secure user login & token-based auth |
| **HTTPS / CORS** | Secure API access |
| **.env Files** | Store API keys and secrets securely |

---

## 📦 Optional Additions

- **Redis** – Caching predictions / intermediate results
- **Celery + RabbitMQ** – Background tasks for training or data sync
- **Sentry** – Error tracking in frontend/backend

---

Would you like:

- A **GitHub-ready boilerplate template**?
- Sample code for any one module (e.g., FastAPI endpoint for predictions)?
- Help writing a **project case study or resume bullet** for this app?

Let me know how you want to proceed!