**Data Acquisition and Storage**

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 1.1 | Acquire OHLCV data (price/volume) | ✅ Done | Already stored in MongoDB stock\_\_prices. |
| 1.2 | Calculate & store technical indicators (SMA, EMA) | ✅ Done | Already integrated into your DB rows. |
| 1.3 | Acquire **Insider Sentiment (MSPR)** | ✅ Done | MSPR is retrieved and parsed from API. |
| 1.4 | Acquire **Social Sentiment** (-1 to 1) | 🟡 In Progress | Finish gathering sentiment from sources like Reddit, Twitter, or APIs like Finnhub. |
| 1.5 | **Store** MSPR + social sentiment into DB | 🟡 Upcoming | Write a script that iterates each doc and attaches the monthly mspr and daily/weekly social sentiment. |

**Feature Engineering (Signals for model input)**

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 2.1 | Compute **daily returns** for each stock | 🔜 Not started | Use pct\_change() on Close prices. |
| 2.2 | Compute **rolling volatility** (std deviation over 10 or 20 days) | 🔜 Not started | Add as another column per stock/date. |
| 2.3 | **Align sentiment** with stock date (merge MSPR monthly + social sentiment daily) | 🔜 Not started | Use merge\_asof() or similar to align sentiment to each row. |
| 2.4 | Create final features\_df with all these columns | 🔜 Not started | This becomes your X for training. |

**Model training and prediction**

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 3.1 | Choose ML model: LinearRegression or RandomForestRegressor | 🔜 Not started | Start with LinearRegression to predict next-day return. |
| 3.2 | Train model on feature columns → label (future return) | 🔜 Not started | Use supervised learning (e.g., predict t+1 return). |
| 3.3 | Save model or pipeline for inference | 🔜 Optional | Can use joblib or just reload trained model each time. |
| 3.4 | Run **inference** on most recent features | 🔜 Not started | Predict returns for current market data. |

**Portfolio Optimization**

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 4.1 | Calculate **covariance matrix** from returns | 🔜 Not started | Use returns\_df.cov() or pypfopt utility. |
| 4.2 | Send expected returns + cov matrix to optimizer | 🔜 Not started | pypfopt or Riskfolio-lib API. |
| 4.3 | Receive optimized weights | 🔜 Not started | Optimizer returns weights as JSON/dict. |
| 4.4 | Compare to current portfolio & generate signals | 🔜 Not started | Logic to detect changes and decide buy/sell/hold. |

**Back Testing and Evaluation**

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 5.1 | Backtest strategy on past data | 🔜 Not started | Use Backtrader, bt, or your own simulator. |
| 5.2 | Evaluate metrics: Sharpe ratio, max drawdown | 🔜 Not started | Plot portfolio value over time. |

User/Account System

| **Step** | **Task** | **Status** | **Action** |
| --- | --- | --- | --- |
| 6.1 | Implement login/signup system | 🔜 Later | Google login with Firebase or Supabase. |
| 6.2 | Store user portfolios, signals, and results | 🔜 Later | Track user-specific runs in MongoDB. |

**Frontend Dashboard Integration**

| **Step** | **Task** | **Description** | **Tools** |
| --- | --- | --- | --- |
| 7.1 | Build login page & auth | Google login or email/password | Firebase, Supabase |
| 7.2 | Create dashboard layout | Tabs: Portfolio, Signals, Strategy | React + Tailwind / Bootstrap |
| 7.3 | Show chart for portfolio performance | Time series of returns | Recharts, Plotly |
| 7.4 | Display optimized weights | Pie chart / bar graph | Chart.js or similar |
| 7.5 | Display buy/sell signals | Daily signals in a table | Filterable by date/asset |
| 7.6 | Strategy switcher | User selects swing or long-term | Save this in user profile |
| 7.7 | Connect MongoDB via API | Backend returns user-specific data | Flask / FastAPI / AWS Lambda (API Gateway) |
| **Long Term v Swing**  Phase | Task | Placement | Notes |
| 📌 1.0 | Add user input or config to choose strategy mode ("swing" or "long-term") | Before or within Phase 1 | Determines data window (e.g., 30 days for swing, 365+ for long-term). |
| 📌 2.0 | Feature engineering tailored to strategy | During Phase 2 | For swing: use short-term indicators (SMA10, RSI).For long-term: use fundamentals + SMA50/200. |
| 📌 3.0 | Train separate models (optional) | Phase 3 | Optionally, train different regressors for each strategy type. |
| 📌 4.0 | Optimizer can work on any time horizon | Phase 4 | Strategy affects input returns and risk assumptions. |
| 📌 5.0 | Run separate backtests for each strategy | Phase 5 | Swing tests daily rebalancing. Long-term tests monthly/quarterly. |
| 📌 6.0 | Show strategy toggle in UI | Phase 6 (UI) | User picks strategy in the interface. |