**🧑‍💻 Project Overview – AlgoProject: AI Driven Smart Trading Platform, “Trade on AutoPilot”**

**Scope: Crypto (Multiple exchanges supported by CCXT) and Stocks (Fyers)**

**Security Standards: VAPT, OWASP Top 10, ISO 27001, SOC 2 Type II**

**Certainly! Below is the beginning of your Institutional-Grade AlgoProject Platform - User Requirement Specification (URS). Due to its length, I will provide it in multiple sections so you can easily copy or save it into your document editor.**

**📘 Institutional-Grade AlgoProject Platform - User Requirement Specification (URS)**

**1. PROJECT OVERVIEW**

AlgoProject is an institutional-grade trading automation platform that allows users to:

* Backtest multiple strategies on historical data.
* Analyze KPIs to determine best-performing asset-strategy pairs.
* Simulate trades using demo mode.
* Execute real-time trades with exchange integrations.
* Browse and subscribe to verified trading strategies from a strategy marketplace.
* Securely manage configurations, track P&L, and evaluate performance using over 29 KPIs.

**2. USER STORIES & FLOW**

**👤 As a user, I want to:**

1. Login using Google OAuth and manage my profile securely.
2. Select exchanges (e.g., Binance, Kraken) and available markets (spot, futures).
3. Select and save crypto pairs for testing and trading.
4. Choose strategies and timeframes to run backtests.
5. View backtest reports with detailed trade logs and KPIs.
6. Compare strategies per asset based on profitability.
7. Simulate trades in demo mode before going live.
8. Connect my exchange keys securely and execute trades in real-time.
9. Monitor trades, order book, PnL, and performance charts.
10. Explore strategy marketplace, view ratings, KPIs, and subscribe.
11. Track subscriptions, billing, and strategy performance in one dashboard.

**3. PLATFORM MODULES**

| **Module** | **Purpose** |
| --- | --- |
| Login/Signup | Google-based secure authentication, OAuth2, MFA |
| Dashboard | Entry point for performance overview and navigation |
| Strategy Manager | Upload, edit, delete or test strategies using version control |
| Backtest Engine | Select strategy + assets + timeframe → Run and store metrics |
| Trade Simulator | Run demo trades based on optimal strategy-asset-timeframe combo |
| Live Trading Panel | Exchange key management, real trades, PnL monitoring |
| KPI Analytics | Advanced reporting across all 29 KPIs + visual insights |
| Strategy Marketplace | Discover, subscribe, or sell strategies with performance metrics |
| Billing & Plans | Manage plan upgrades, usage limits, history |
| Security Center | Key encryption, 2FA, VAPT, suspicious login alert |
| FAQ & Support | 200+ common questions, help walkthroughs |

**4. KPI MATRIX FOR BACKTESTING**

**The platform will support the following KPIs:**

* Start Date, End Date, Duration
* Exposure Time [%]
* Equity Final [$], Equity Peak [$]
* Return [%], Buy & Hold Return [%], Return (Ann.) [%], CAGR [%]
* Volatility (Ann.) [%]
* Sharpe Ratio, Sortino Ratio, Calmar Ratio, Alpha [%], Beta
* Max. Drawdown [%], Avg. Drawdown [%], Max/Avg Drawdown Duration
* Total Trades, Win Rate [%], Best/Worst/Avg Trade [%]
* Max/Avg Trade Duration, Profit Factor, Expectancy [%]

**5. STRATEGY-ASSET COMPARATOR (BACKTEST LOGIC)**

**✅ Logic:**

**For each asset:**

* For each strategy:
  + Run backtest
  + Extract and store all KPIs
  + Save individual trade log (CSV)
  + Generate equity chart

**✅ Output Files:**

* output/trades/{symbol}\_{strategy}.csv
* output/strategy\_asset\_summary.csv
* output/best\_strategies\_by\_asset.csv
* charts/{symbol}\_{strategy}\_equity.png

**✅ Rating System (Stars):**

| **Profit Factor** | **Sharpe Ratio** | **Stars** |
| --- | --- | --- |
| ≥ 1.5 | ≥ 1.5 | ⭐⭐⭐⭐⭐ |
| ≥ 1.2 | ≥ 1.0 | ⭐⭐⭐⭐ |
| ≥ 1.0 | any | ⭐⭐⭐ |
| < 1.0 | any | ❌ |

🛠 **Recommendation:**

* Refactor: Use dynamic imports, configs, or class-based strategy handling instead of creating a file per asset/strategy.
* Organize structure as:
  + /core -> strategy engine, KPIs, trade execution
  + /ui -> frontend React/Vue or Flask views
  + /data -> data loaders, fetchers
  + /strategies -> dynamic strategy modules
  + /backtest -> runner, config, reporting
  + crypto -> use this for all crypto related files
  + crypto/output -> results, charts, logs
  + crypto/input – any input files like configuration, crypto pairs etc
  + crypto/logs – logs file
  + crypto/scripts – these are scripts that will be called in backend
  + crypto/tools – any scripts that will be used for test or utility
  + crypto/docs – all the documents \*.md files , FAQ, how do
  + stocks -> use this for all stocks related files
  + stocks/output -> results, charts, logs
  + stocks/input– any input files like configuration, crypto pairs etc
  + stocks/logs– logs file
  + stocks/scripts– these are scripts that will be called in backend
  + stocks/tools – any scripts that will be used for test or utility
  + stocks/docs– all the documents \*.md files , FAQ, how do

**🙋‍♀️ User Stories for AlgoProject Platform**

**1. User Onboarding & Asset Setup**

**As a user**, I want to:

* Log in via Google OAuth.
* Select the exchange I want to use (e.g., Binance, Bybit or any from the list of available exchanges).
* Select asset classes: Spot, Futures, or Options and any more available assets from that selected exchange.
* Choose specific trading pairs (e.g., BTC/USDT, ETH/USDT or Multiple from the list of available assets classes) to track.
* Save selected assets to my profile for analysis and trading.

**2. Backtest Exploration**

**As a user**, I want to:

* Choose **one or more strategies**, **timeframes**, and **assets** for backtesting.
* Run backtests on **historical data** using the selected combinations.
* Get complete available bars for historical data based on the timeframe selected
* Mention how many bars are available next to the time selection in UI. A script is available which can fetch this information.
* View and compare:
  + 📊 Strategy KPIs (Profit Factor, Sharpe Ratio, Win Rate, etc. refer : 📚 Refer: [Backtesting.py KPIs] (https://github.com/kernc/backtesting.py)
  + 📈 Equity curves and drawdown charts
  + 🏅 Best strategy for each asset and time window
* Save the configuration (strategy + assets + timeframes) for backtest use.

**3. Demo Trade Simulation**

**As a user**, once I identify the best combination of asset + strategy + timeframe from the backtest I did:

* I want to **simulate trades** ("demo trading") for several days or weeks.
* View performance as if real trades were placed (based on live data).
* Track ongoing demo PnL, equity, trades, and risk metrics.
* Receive alerts if the strategy breaks key thresholds (e.g., drawdown).

**4. Live Trade Execution**

**As a user**, once I trust the setup:

* I want to input my **API keys securely** for live trading.
* Connect to my exchange account and test the connection.
* Launch live trading using selected strategy/asset settings.
* View:
  + 📘 Order book, positions, open/closed trades
  + 🔁 Real-time performance metrics and risk stats
  + 📉 29+ KPIs for live monitoring (same as backtest)

**🔁 GitHub Copilot Instructions: Strategy vs Asset Backtest Comparator**

# 🚀 GOAL:

# Run multiple strategies over multiple assets to identify the best performing pair

# Using advanced KPIs like Profit Factor and Sharpe Ratio.

# Clear visible Table and recommendation which is the strategy, assests and timeframe to use for user and a button next to deploy live.

# ✅ 1. Backtest Matrix:

# For each asset in asset\_list:

# For each strategy in strategy\_list:

# - Run backtest

# - Save KPIs and trades

# ✅ 2. KPIs to Compute:

# - Final Return [%], CAGR [%], Total Trades

# - Win Rate [%], Avg Trade [%], Expectancy [%]

# - Profit Factor, Sharpe Ratio, Sortino Ratio

# - Max/Avg Drawdown [%], Volatility

# - Max/Avg Trade Duration

# - Best/Worst Trade [%]

# ✅ 3. Save Results:

# - Save per-trade logs: output/trades/{symbol}\_{strategy}.csv

# - Save KPI summary: output/strategy\_asset\_summary.csv

# ✅ 4. Strategy Ranking per Asset:

# - Sort by Profit Factor DESC, then Sharpe Ratio DESC

# - Top scoring strategy gets:

# ⭐⭐⭐⭐⭐ if PF ≥ 1.5 and Sharpe ≥ 1.5

# ⭐⭐⭐⭐ if PF ≥ 1.2 and Sharpe ≥ 1.0

# ⭐⭐⭐ if PF ≥ 1.0

# ❌ if PF < 1.0

# ✅ 5. Display Top Results:

# - Use tabulate or rich to show color-coded table

# - Print top 5 and bottom 5 strategy-asset combos

# ✅ 6. Visualization:

# - Plot equity curve per run and save as PNG

# - Optionally generate radar charts for strategy comparison

# ✅ 7. Configurable Thresholds (YAML):

# pf\_threshold: 1.5

# sharpe\_threshold: 1.5

# min\_trades: 5

# ✅ 8. CLI & UI Support:

# - Enable CLI to run: python backtest.py --strategies=all --assets=top10 --days=90

# - Prepare API/Flask endpoints for frontend to trigger backtests and fetch results

**📦 Output Structure (recommended)**

/output/

├── charts/

│ └── BTCUSDT\_momentum\_equity.png

├── demo\_trades/

│ └── BTCUSDT\_momentum.csv

├── back\_test/

│ └── BTCUSDT\_momentum.csv

├── strategy\_asset\_summary.csv

└── best\_strategies\_by\_asset.csv

Clean the output folder when user runs the backtest strategy , demo test again, we will keep only latest runs or can keep last 5 runs records of best\_strategies\_by\_asset.csv, all other individual files can be deleted as the summary is captured and this will be only used to display of past runs.

**✅ Final Notes & Next Steps**

* **Refactor Codebase**: Consolidate the excessive number of Python files into strategy classes, modular loaders, and centralized runners.
* **UI Kickoff**: You’re ready to begin the frontend (React or Flask with templates).
* **Secure API Key Handling**: Use encrypted vault or environment variables.
* **Run Configs via YAML/JSON**: Avoid hardcoded params.
* **Cleanup**: delete all empty files and unwanted files that are not being used. If the files are used independently, move them to tools folder as it has function that are used by single file execution.