**THIS CONTAINS:**

1. A short project objective.
2. Concrete **backend** responsibilities (tasks, APIs, DB schema, infra notes).
3. Concrete **frontend** responsibilities (screens, components, realtime, UX flows).
4. Role-by-role task checklist for other team members.
5. Exactly **what Strategist** must keep doing and produce.
6. A short sprint/prioritisation roadmap so everyone knows what to build first.

**Objective (one line)**

Build a modular multi-symbol, multi-timeframe crypto research & semi-automated trading system that: ingests OHLCV, runs strategies/backtests, produces ranked trade suggestions (entry/SL/TP/RR/probability), exposes signals via APIs, and visualizes them in a realtime dashboard for users to approve.

**1) Backend — exact responsibilities & deliverables**

**Core services (each can be microservice or module)**

1. **Data Ingest & ETL**
   * Tasks:
     + CCXT-based fetcher (or scheduled job) per symbol/timeframe. Append only, per-timeframe retention.
     + CSV/Parquet storage + merged multi-TF generator.
     + Data validation, schema normalization (ts, open, high, low, close, volume).
   * Deliverables:
     + GET /internal/health (basic).
     + data/{symbol}/{timeframe}.csv (or parquet) and merged\_multi\_tf.csv.
   * Notes: expose a small internal CLI to force-update a symbol.
2. **Indicator & Feature Service**
   * Tasks:
     + Compute EMA, RSI, ATR, ADX, MACD, etc. on merged dataset.
     + Persist computed indicator columns (or compute on-the-fly with caching).
   * Deliverables:
     + Function/API to return a dataframe or array of indicator columns for an input timespan.
3. **Backtest & Strategy Engine**
   * Tasks:
     + Implement your Backtester class as a service/library.
     + Provide programmatic interface to run backtest on symbol/timeframe with given parameters.
     + Record trade history and metrics (win\_rate, TP/SL hits, avg\_RR, durations).
   * Deliverables:
     + POST /api/v1/backtest — body: {symbol, timeframe, strategy, params} → returns metrics + trade list.
4. **Signal Generator / Suggestions Service**
   * Tasks:
     + Run strategy on latest candle and historical stats to produce suggestions.
     + Implement filtering & ranking (win\_prob threshold, RR threshold, EV ranking).
   * Deliverables:
     + GET /api/v1/suggestions — returns list of suggestions JSON (see example below).
     + Optionally: GET /api/v1/suggestions/{symbol}.
5. **Order Execution (stub / testnet)**
   * Tasks:
     + Implement an execution module that can place test orders on Binance testnet.
     + Safety-first: only accept orders after explicit user approval and with audit logging.
   * Deliverables:
     + POST /api/v1/orders (sandbox only initially). Audit log required.
6. **Persistence & Audit**
   * Tasks:
     + DB for OHLCV meta, signals, trade\_history, users, API keys, audit logs.
   * Minimal schema (Postgres):
     + ohlcv(symbol, timeframe, ts, open, high, low, close, volume) — or keep CSV + index.
     + signals(id, symbol, timeframe, ts, side, entry, sl, tp, status, ctx\_json, expires\_at)
     + trades(id, signal\_id, entry\_time, exit\_time, entry\_price, exit\_price, sl, tp, result, pnl, meta)
     + users, api\_keys, audit\_logs
   * Deliverables:
     + DB migrations
7. **Workers / Scheduler**
   * Tasks:
     + Background workers: scheduled ETL, indicator recompute, strategy runner, email/Telegram notifier.
   * Tools: Celery/RQ/asyncio or cron + Docker.
8. **Monitoring & Observability**
   * Tasks:
     + Logging, error tracking (Sentry), metrics (Prometheus), basic dashboards.
   * Deliverables:
     + Error/health alerting.

**Example API response (Suggestion)**

{

"symbol":"BTC/USDT",

"timeframe":"5m",

"side":"LONG",

"entry":27250.0,

"SL":27100.0,

"TP":27450.0,

"expected\_duration\_min":35,

"win\_probability":55.0,

"avg\_rr":1.8,

"ctx":{"ema50":27220.5, "ema200":27100.0, "rsi":45.2}

}

**2) Frontend — exact responsibilities & deliverables**

**High-level pages & components**

1. **Dashboard (Landing)**
   * Shows ranked trade suggestions with key columns (symbol, timeframe, side, entry, SL, TP, win\_prob, RR, EV).
   * Filters (win\_prob min, RR min, timeframe selector).
   * Buttons: “Approve (paper)”, “Approve (execute - sandbox)”, “Ignore”.
2. **Symbol Detail / Chart View**
   * Candlestick chart with overlays (EMA50/200, ATR bands).
   * Trade markers: past trades (entry/exit), pending signal (SL/TP).
   * Multi-timeframe selector (view 5m/15m/1h stacked).
   * Quick backtest on visible window.
3. **Trade Log / History**
   * Table of all trades per symbol with ability to view trade context and export CSV.
4. **Strategy Editor & Tune**
   * UI to change EMA window, RSI thresholds, ATR multipliers and re-run backtest (calls POST /api/v1/backtest).
   * Save strategy presets.
5. **Alerts & Notifications**
   * Configure Telegram/Email push for new high-EV suggestions.
   * In-app web push / toast.
6. **User Settings / API Keys**
   * Store user-level preferences: risk per trade, max concurrent trades, testnet API keys.

**UI/UX specifics**

* Use React + Tailwind (you chose React earlier). Use Zustand/Redux toolkit for state, RTK Query or SWR for API caching.
* Charting: Recharts, TradingView lightweight, or lightweight candlestick (Lightweight Charts).
* Realtime: WebSocket or Server-Sent Events for live suggestions & market ticks.
* Components: TradeCard, ChartWithIndicators, SuggestionTable, FiltersPanel, StrategyForm.
* Mobile responsive, accessible and minimal.

**Deliverables**

* /dashboard with suggestion table + filters
* /symbol/:symbol chart with trade overlays
* /strategy editor to tweak and run backtests
* Notification integration (Telegram) for approvals

**3) Who does what — specific role checklist**

**Backend dev(s) (2–3)**

* Implement ETL & data store (1 dev).
* Implement indicators & backtester as a library/service (1 dev).
* Build suggestions API + filtering + ranking (1 dev).
* Tasks:
  + Build and test POST /api/v1/backtest.
  + Build GET /api/v1/suggestions.
  + DB schemas & migrations.
  + Unit tests for strategy/backtester functions.
  + Worker/scheduler for daily ETL.
  + Logging and health endpoints.

**Frontend dev(s) (2)**

* Implement React app pages listed above.
* Tasks:
  + Build Suggestion Table & filters (priority).
  + Chart view with overlays & trade markers.
  + Strategy editor & run-backtest UI.
  + Integrate with WebSocket for live updates.
  + Save user preferences & handling approvals.

**Data Engineer / Quant**

* Provide cleaned datasets and verification scripts.
* Write / tune strategies and parameters.
* Define labeling rules for successful trades (TP vs SL).
* Produce experiment definitions for A/B parameter tests.

**QA / Tester (1)**

* Test ETL, endpoints, UI flows, edge cases (missing data).
* Regression tests for backtester.

**DevOps (1)**

* Dockerize services, set up local env, CI pipeline, and scheduled runner (cron/Celery).
* Prepare sandbox/testnet keys and env variables (secrets management).
* Monitoring setup.

**Product / PM (optional)**

* Prioritize feature list, gather acceptance criteria, run UAT with you.

**4) What Strategist should do next — exact deliverables**

You’re the strategy owner — your focus should be:

1. **Strategy specs & params**
   * Document exact entry/exit rules, ATR multipliers, timeframe priorities.
   * Provide a list of parameter ranges for optimisation (e.g., EMA50 in [30,80], ATR multiplier [1.0–2.5]).
2. **Labeling / Historical ground truth**
   * Approve definition for “win” (hit TP before SL) and compute durations — ensure backtester uses this definition.
3. **Acceptance criteria**
   * Define thresholds for “suggestible” trades (min win\_prob, min RR, min sample size).
   * Define portfolio risk rules (max open trades, max exposure).
4. **Curate symbol list & priority**
   * List of symbols to support (top 20 pairs). Tag which are high-priority.
5. **Validation & UAT**
   * Regularly review suggestion output, pick a sample of signals and validate.
   * Provide feedback to developers when signals look wrong (e.g., SL/TP direction issues).
6. **Experimentation**
   * Create experiment runs: parameter grid or Bayesian optimisation tasks to tune the strategy.
7. **Training / Gamification content (future)**
   * For the training-game idea: design exercises, scoring rules, reward thresholds.

**5) Short sprint roadmap (what to build first — 3 sprints, ~2 weeks each)**

**Sprint 0 — Foundation (Week 1)**

* Backend: ETL + store + merged TF generation + compute indicators.
* Frontend: Basic dashboard skeleton + suggestion table placeholder.
* Deliverable: GET /api/v1/suggestions (basic) + UI table showing raw suggestions.

**Sprint 1 — Backtester & Metrics (Week 2)**

* Backend: complete Backtester service, store trade history, add POST /api/v1/backtest.
* Frontend: show per-suggestion metrics, download trade CSV.
* Deliverable: per-symbol backtest & trade history; UI plots of equity curve.

**Sprint 2 — Filtering, Ranking & UX polish (Week 3)**

* Backend: implement filter/rank engine, EV calculation, testnet order stubs.
* Frontend: interactive filters, symbol details, trade approval UI, WebSocket for live signals.
* Deliverable: production-like suggestion/approval flow and saved trades.

**6) Quick engineering “how” (tech choices)**

* Backend: **Python (FastAPI)** — easy to host, async, great for data + Celery.
* DB: **Postgres** for metadata + signals + trades.
* Cache/Realtime: **Redis** for ticks & pub/sub.
* Worker: **Celery + Redis** for scheduled ETL/backtests.
* Frontend: **React + Tailwind**. Charts: **Lightweight Charts** or **Recharts**.
* Container: **Docker** for local reproducibility.

**7) Example checklist you can hand to each person**

**Backend Dev Checklist**

* Implement CCXT fetcher and append logic.
* Build merged\_multi\_tf.csv generation.
* Implement indicator module (EMA/RSI/ATR/ADX).
* Implement Backtester library and unit tests.
* Add POST /api/v1/backtest and GET /api/v1/suggestions.
* Add DB models + migrations for signals/trades.
* Add logging & health endpoints.

**Frontend Dev Checklist**

* Implement suggestions table with sorting/filters.
* Build symbol detail view with candlestick + indicators.
* Add trade approval modal (review SL/TP and approve).
* Wire WebSocket to receive live suggestions.
* Save user preferences.

**Strategist**

* Deliver final strategy spec + parameter ranges.
* Define acceptance criteria for “suggestible” trades.
* Approve initial symbol list and timeframes.
* Run UAT and provide weekly feedback.