# GENESIS Platform – Replit Deployment & Trading Specification

This consolidated brief packages every requirement – functional, architectural, design, and execution – into a single hand‑off document for a developer working on Replit.

## 0. Primary Idea & Approach

Build an app that:

* • Pulls live prices from an \*\*OANDA demo\*\* account and renders real‑time charts.
* • Sends each chart image to \*\*ChatGPT Vision\*\* every 5 minutes (1 minute when a potential setup is detected).
* • Lets ChatGPT decide one of four outcomes: \*Anticipated‑Long\*, \*Anticipated‑Short\*, \*Buy‑Now\*, \*Sell‑Now\*.
* • If “Buy Now / Sell Now” → auto‑place trade on MT5 with SL, TP, trailing SL or scale‑in per Vision advice.
* • If only “Anticipated” → mark chart with an arrow overlay in MT5 while continuing 1‑minute scans.
* • Continually refines decisions via \*\*machine‑learning\*\* feedback (trade outcome ↔ Vision signals).

Symbols initially in scope: \*\*XAUUSD, GBPJPY, GBPUSD, EURUSD, AAPL, NAS100, BTCUSD\*\*.

Screenshot files are overwritten every 6 h unless retained for ML retraining.

## 1. What stays / what merges / what goes

|  |  |  |
| --- | --- | --- |
| Old area | Action | Notes / rationale |
| Dashboard | Keep (single “home” page) | Includes MT5 widgets and Vision signal badges. |
| Trading History | Keep untouched | Live MT5 trades mirrored here and cross‑checked with OANDA prices. |
| Settings | Consolidate into one page with tabs | Tabs: “Connections”, “Risk & ML”, “Notifications”, “Maintenance”. |
| MT5 Dashboard page | Remove | Functionality merged into Dashboard. |
| MT5 EA Setup page | Remove | Fields live in Settings → Connections; EA download link in small card. |
| Telegram Setup page | Remove | Fields in Settings → Notifications; keeps test button. |
| ML Insights page | Remove | Metrics appear in Dashboard “Insights” drawer; training controls in Settings → Risk & ML. |
| Logs / Admin pages | Backend only | Expose `/api/logs` JSON + flat text file. |

## 2. Skinny file/folder layout (max 10 Python modules)

\*\*Important: do NOT create extra .py files.\*\* Keep every service restricted to this structure:

genesis/  
├─ app/  
│ ├─ \_\_init\_\_.py # create\_app() – routers, DB, WS  
│ ├─ api.py # FastAPI routes  
│ ├─ models.py # SQLAlchemy models  
│ ├─ services/  
│ │ ├─ mt5\_service.py  
│ │ ├─ telegram\_service.py  
│ │ ├─ vision\_service.py  
│ │ ├─ risk\_service.py  
│ │ └─ oanda\_service.py  
│ ├─ workers.py # Celery / RQ tasks  
│ ├─ schemas.py # Pydantic DTOs  
│ └─ settings.py # Config loader  
├─ templates/ # dashboard.html, history.html, settings.html  
├─ static/ # JS/CSS/HTMX (Pepperstone‑style theme)  
├─ tests/ # pytest  
└─ README.md

## 3. Key routes & real‑time channels

|  |  |
| --- | --- |
| Method / Path | Purpose |
| GET / | Dashboard |
| GET /history | Trading History |
| GET /settings | Settings |
| GET /api/trades | Paginated trade list |
| POST /api/trade | Manual trade → MT5 |
| GET /api/signals/ws | WebSocket – signals + ticks |
| POST /api/settings/{section} | Save settings tab |
| GET /api/logs | Fetch/stream logs |

## 4. Database schema (PostgreSQL)

|  |  |
| --- | --- |
| Table | Key columns |
| users | id, email, hashed\_pw, role |
| settings | id, section, key, value (JSON) |
| signals | id, symbol, action, entry, sl, tp, confidence, status, created\_at |
| trades | id, signal\_id, ticket, symbol, side, lot, entry, exit, sl, tp, pnl, status, opened\_at, closed\_at |
| risk\_profiles | id, name, json\_rules |
| logs | id, ts, level, source, message, context (JSON) |

## 5. Service‑layer contracts

class MT5Service:  
 def open\_trade(signal: Signal) -> Trade: ...  
 def close\_trade(ticket: int) -> None: ...  
 def account\_info() -> dict: ...

## 6. Settings page tab mapping

|  |  |
| --- | --- |
| Tab | Fields |
| Connections | OANDA API key/ID, MT5 EA endpoint, OpenAI key, Telegram bot token, “Test” buttons |
| Risk & ML | Risk profile CRUD, ML model picker & “Retrain” |
| Notifications | Telegram / email / desktop alerts toggles |
| Maintenance | Backup/restore, cleanup, version check |

## 7. Data‑flow & failure handling

1. 1. OANDA price feed → chart image every 5 min (or 1 min during setup) → ChatGPT Vision.
2. 2. Vision outputs anticipated/entry signals stored in `signals` table.
3. 3. `risk\_service` validates vs. active profile.
4. 4. If entry signal: `mt5\_service.open\_trade()`; else arrow overlay on MT5.
5. 5. Missed MT5? – Fallback `telegram\_service.push\_backup\_signal()`.
6. 6. Trade outcomes feed ML retraining routine.

## 8. Auth strategy

• \*\*Now:\*\* single user, session cookie + CSRF.

• \*\*Future‑proof:\*\* JWT with `role`; DB already supports roles.

## 9. Replit deployment (Reserved VM)

# .replit  
run = "bash start.sh"  
  
# start.sh  
uvicorn app.api:app --host 0.0.0.0 --port $PORT &  
celery -A app.workers worker -l info --concurrency=2  
wait

\*\*Secrets to add\*\*: `OPENAI\_API\_KEY, OANDA\_API\_KEY, OANDA\_ACCOUNT\_ID, MT5\_URL, MT5\_TOKEN, TELEGRAM\_BOT\_TOKEN, DATABASE\_URL, REDIS\_URL`.

Style: replicate \*\*Pepperstone.com\*\* theme colours; layout according to provided screenshots for Dashboard, Trading History, ML page if necessary.

## 10. What the developer receives

• This brief.

• ZIP scaffold: structure (§2), `.env.example`, pytest suite, OpenAPI collection, placeholder Pepperstone‑style CSS.

\*\*Day‑1 tasks\*\*

1. Create Repl (Reserved VM) from repo.

2. Add secrets above.

3. Click \*\*Run\*\* – Uvicorn + Celery start; FastAPI serves site.

4. `alembic upgrade head` to migrate DB.

5. Verify stub Dashboard, MT5 connectivity, fallback Telegram.

6. Implement Vision → trade loop in `vision\_service.py` and `workers.py`.

7. Keep codebase to max 10 modules – no extra .py files.