MoonLight — Fixed Time İşlem AI (Proje Dosyası)

**Kısa Özet** MoonLight, Android (telefon/tablet) ve Windows 10/11 üzerinde çalışabilecek, arka planda işlem yapabilen, demo ve gerçek hesaplarda işlemleri otomatikleştirebilen modüler bir Fixed-Time (binary / turbo) işlem yapay zekâsı projesidir. Sistem; güvenlik, uyumluluk, geri test (backtest), paper-trading, risk yönetimi ve takip/izleme özelliklerine sahip olacaktır.

İçindekiler

1. Proje hedefleri ve kapsam
2. Hukuk, uyumluluk ve etik (ön koşullar)
3. Yüksek seviyeli mimari
4. Modüller ve her modülün açık adım adım geliştirme rehberi
5. Platforma göre dağıtım stratejisi (Android / Windows)
6. Güvenlik ve gizlilik uygulamaları
7. Test, validation ve dağıtım (CI / CD)
8. MVP (Minimum Viable Product) - Adım adım yol haritası ve zaman çizelgesi
9. Dosya / klasör yapısı önerisi
10. Kullanılacak kütüphaneler, araçlar ve öneriler
11. İzleme, loglama ve hata yönetimi
12. Risk yönetimi, para yönetimi ve backtesting
13. Etik kısıtlamalar ve yasaklar — neler yapılmaz
14. İleri aşama (opsiyonel): ML model eğitimi, dağıtık öğrenme, otomatik güncelleme
15. Geliştirme Kalitesi İçin Ek Öneriler
16. Sonraki adımlar — seni ilerletmek için ne yapacağım

1) Proje hedefleri ve kapsam

* Sabit zamanlı (Fixed Time) kontratlarda demo ve gerçek hesapta otomatik işlem açabilen bir sistem.
* Hem Android hem Windows üzerinde çalışacak; tercihen çekirdek motor (core engine) tek kod tabanlı olacak ve farklı istemciler (mobile/desktop) üzerinden kontrol edilebilecek.
* Kullanıcı kimlik bilgileri (e-posta, kullanıcı adı, şifre) kullanıcı tarafından girilecek; sistem bunları güvenli şekilde saklayıp API/oturum bazlı yetkilendirme ile işlem yapacaktır.
* Gerçek paraya bağlamadan önce demo (paper) hesapta kapsamlı test ve backtest yapılacak.

2) Hukuk, uyumluluk ve etik (ön koşullar)

* Önemli: Platformların hizmet şartlarını (Service Agreement) ve ülke yasalarını mutlaka incele. Binary/fixed-time ürünleri bazı ülkelerde yasak veya kısıtlıdır.
* Olymp Trade gibi bazı broker’ların resmi dokümanları “ticaret robotları kullanımına” izin verdiğini belirtebilir; ancak platformların ek kuralları, ek işlevsellik (örneğin üçüncü taraf yazılımların kullanımı) ve kötüye kullanım durumları için kısıtlamaları olabilir.
* **Kesin red**: Platform güvenlik önlemlerini atlatma, anti-bot sistemlerini kandırma, API anahtarlarını kötüye kullanma veya başka kullanıcıların hesaplarına yetkisiz erişim sağlama konularında yardımcı olamam. Bu tür davranışlar hem hukuka hem de platform kurallarına aykırıdır.

3) Yüksek seviyeli mimari

* **Core Engine (Python)**: Asenkron WebSocket/REST veri alma ve emir verme, strateji işleyici, risk yöneticisi, simülasyon / backtest modülleri.
* **API/Bridge**: Core ile istemciler (Android/Windows) arasında güvenli WebSocket/REST arayüzü (TLS). Kimlik doğrulama token’ları kullan.
* **Android Client**: Hafif istemci (Flutter veya native Kotlin) — arayüz, konfigürasyon, uzaktan başlat/durdur, bildirim.
* **Windows Client**: Opsiyonel masaüstü GUI (Electron/Flutter/Tauri) veya doğrudan core’u çalıştıracak CLI/servis.
* **Storage**: Kullanıcı verileri lokal olarak şifrelenmiş (OS keystore veya platforma özel) + opsiyonel merkezi sunucu (kullanıcı izin verirse) için şifreli veritabanı.
* **Logging & Monitoring**: Ayrı log kanalında; işlem geçmişi, strateji parametreleri, PnL, hata takibi.

4) Modüller ve adım adım geliştirme rehberi

(Önceki içerik aynı şekilde korunur — Market Connector, Authentication, Strategy Engine, Risk Manager, Executor, Persistence, UI, Background Service modülleri.)

5) Platforma göre dağıtım stratejisi

(Önceki içerik korunur.)

6) Güvenlik ve gizlilik

(Önceki içerik korunur.)

7) Test, validation ve dağıtım

(Önceki içerik korunur.)

8) MVP yol haritası

(Önceki içerik korunur.)

9) Önerilen dosya / klasör yapısı

(Önceki içerik korunur.)

10) Kullanılacak araçlar ve kütüphaneler

(Önceki içerik korunur.)

11) İzleme, loglama ve hata yönetimi

(Önceki içerik korunur.)

12) Risk yönetimi, para yönetimi ve backtesting

(Önceki içerik korunur.)

13) Etik kısıtlamalar ve yasaklar

(Önceki içerik korunur.)

14) İleri aşama (opsiyonel öneriler)

(Önceki içerik korunur.)

15) Geliştirme Kalitesi İçin Ek Öneriler

1. **Mimaride sağlam temeller**: Tek core engine, event-driven *asyncio* yapısı, worker modeli.
2. **Güvenlik**: Credential’lar OS KeyStore/DPAPI/Android Keystore ile korunacak; loglarda PII yok.
3. **Emir Yönetimi**: Idempotency key, state-machine (PREPARE→SEND→CONFIRM→SETTLED).
4. **Test**: Backtest + paper trading uzun süreli koşular; pytest ile birim/entegration test.
5. **Veri Kalitesi**: Ring buffer, latency ölçümü, yüksek gecikmede auto-suspend.
6. **Risk**: Circuit breaker, günlük max loss, consecutive loss limiti.
7. **Monitoring**: Prometheus/Grafana entegrasyonu, hata bildirimleri (push/email).
8. **Hot-reload Config**: JSON/YAML konfigürasyon, anında Worker’lara yansır.
9. **Model & Strateji**: Basit EMA/RSI → ML → PyTorch modelleri.
10. **Ölçek**: Gerekirse Docker/Kubernetes, Redis/RabbitMQ kuyruğu.
11. **UX**: Kill-switch, manual onay, demo→gerçek geçişte kademeli onay.
12. **Dokümantasyon & CI/CD**: Semver, release pipeline, örnek config dosyaları.

16) Sonraki adımlar

* PoC (connector + worker + EMA) kodu hazırlanacak.
* Config yapısı JSON/YAML olarak belirlenecek.
* Worker/scheduler sistemi kurulacak.
* Paper trading ile uzun test sonrası gerçek hesaba küçük lot ile geçiş yapılacak.

16) Geliştirme ve kaliteyi artırma — Öneriler (dâhil edildi)

Bu bölüm kullanıcı isteği üzerine projeye eklendi. Aşağıdaki maddeler proje kapsamında **standart** olarak uygulanacak ve ilgili modüllere entegre edilecek:

16.1 Mimaride sağlam temeller

* Core engine tek kod tabanında (Python) tutulacak; istemciler (Android/Windows) bu core’a bağlanacak.
* Asenkron, event-driven yapı (*asyncio*) kullanılacak.
* Her Worker tek sorumluluk prensibine göre tasarlanacak (her Worker bir (ürün,timeframe) işlesin).

16.2 Güvenlik ve credential yönetimi

* Şifreler OS KeyStore/Windows DPAPI ile saklanacak; disk üzerinde AES-GCM ile ek şifreleme uygulanacak.
* Token/refresh flow varsa token kullanılacak; kullanıcı şifresi doğrudan saklanmayacak.
* Loglarda PII/şifre kesinlikle yazılmayacak.

16.3 Güvenilir emir yönetimi

* Her emrin idempotency key (UUID) olacak; broker onayı gelmediğinde kontrollü retry yapılacak.
* Emir akışı state-machine ile uygulanacak: PREPARE → SEND → CONFIRM → SETTLED.
* At-most-once gönderim kuralı benimsenerek duplicate risk azaltılacak.

16.4 Test & validation

* Deterministik backtest altyapısı kurulacak; stratejiler geçmiş veride metriklerle değerlendirilecek (winrate, expectancy, max drawdown).
* Paper-trading uzun süre testi (öneri: minimum 2–4 hafta) zorunlu olarak yapılacak.
* Unit testler (pytest) ve entegrasyon testleri yazılacak; connector’lar mocklanarak test edilecek.

16.5 Veri kalitesi ve gecikme yönetimi

* Fiyat akışı lokal ring buffer’a kaydedilecek; worker’lar buradan okuyacak.
* Latency ölçümü eklenecek; yüksek gecikme tespitinde otomatik suspend yapılacak.

16.6 Risk yönetimi (zorunlu)

* Circuit-breaker, günlük max loss, ardışık kayıp limiti ve maksimum pozisyon kontrolleri zorunlu olacak.
* Gerçek hesaba otomatik geçiş için kademeli onay ve başarı kriterleri tanımlanacak.

16.7 Monitoring & Observability

* Prometheus/Grafana benzeri telemetri planı ve metrik endpoint’leri tanımlanacak.
* Hata/critical uyarılar için push/email/Sentry entegrasyonu planlanacak.

16.8 Config yönetimi ve hot-reload

* Konfigürasyon JSON/YAML formatında tutulacak; değişikliklerde worker’lar hot-reload ile anında yeni ayarı alacak.

16.9 Model ve strateji iyileştirme

* Başlangıçta deterministic indikatörler (EMA, RSI) kullanılacak; ileride feature engineering ve ML modelleri (scikit-learn → PyTorch) entegre edilecek.
* Canlı sonuçlardan otomatik feature/label kaydı yapılarak periyodik offline retrain süreçleri planlanacak.

16.10 Ölçek ve paralellik

* İlk etapta tek makine, ihtiyaç halinde Docker/Kubernetes ile yatay ölçekleme hedeflenecek.
* Message broker (Redis Streams/RabbitMQ) ile job queue mimarisi desteklenecek.

16.11 UX / Güvenlik Onayları

* Kullanıcıya panic-button ve manuel kill-switch sunulacak.
* Gerçek hesaba geçiş için zorunlu onay adımları ve başarı kriterleri uygulanacak.

16.12 Dokümantasyon & Sürüm Kontrolü

* Her API ve konfigürasyon için doküman ve örnek config dosyaları hazırlanacak.
* Semver ve CI/CD pipeline (unit test → build → release) önerilecek.

Bu eklemeler proje planının ilgili bölümlerine bağlantılı olarak entegre edildi: “MVP yol haritası”, “Risk Manager”, “Backtesting” ve “Deployment” bölümlerinde gerekli referans bağlantıları eklendi.

17) Strateji Kataloğu v1 — (Kullanıcı Listesi 1–50 dâhil)

Bu bölümde, kullanıcının paylaştığı **Kategori / İndikatör / Varsayılan Ayarlar / Notlar** içeren strateji seti **ID 1–50** olarak aynen kataloglanmıştır. Stratejiler MoonLight içinde tek tek etkinleştirilebilir ve ürün/zaman dilimine göre parametreleri override edilebilir.

17.1 Katalog veri modeli (config / import)

* **ID**: tamsayı (1..N)
* **category**: “Trend/MA”, “Osilatör”, “Momentum”, “2’li” vb.
* **name**: kısa ad (ör. “EMA Trend + RSI”)
* **params**: { key:value } (ör. {“ema\_fast”:9,“ema\_slow”:21,“rsi”:7})
* **rules**: sinyal mantığı (ör. “EMA9>EMA21 & RSI7>55; pullback: EMA9”)
* **sl\_tp**: SL/TP kuralı (ör. “SL: ATR14×1.2; TP: 2R”)
* **notes**: metin

**CSV başlıkları (öneri):**

id,category,name,params,rules,sl\_tp,notes

**JSON örneği:**

{ "id": 5, "category": "2’li", "name": "EMA Trend + RSI", "params": {"ema\_fast": 9, "ema\_slow": 21, "rsi": 7}, "rules": "EMA9>EMA21 & RSI7>55; EMA9’a pullback al", "sl\_tp": "SL: swing/ATR14×1; TP: 1.5R/EMA21 alt", "notes": "TF bağımsız, intraday uyumlu"}

17.2 UI eşlemesi

* **Ayarlar > Stratejiler** ekranında her satır ID ile listelenir (arama/filtre: kategori, isim, TF, aktif/pasif).
* Her strateji için: **Aktif**, **TF (1/5/15)**, **Başarı hedefi %**, **Permit min–max**, **Lot**, **SL/TP tipi** (R-multiple, ATR tabanlı), **Parametre override** alanları bulunur.
* Çoklu ürün yönetiminde, stratejiler ürün bazında kopyalanabilir veya kalıpla (template) uygulanabilir.

17.3 Uygulama (engine) eşlemesi

* Katalogtaki her satır **signal provider** olarak Worker’a plug-in edilir.
* Ensemble/voting katmanında ağırlıklar, gerçek zamanlı başarı oranına göre adaptif güncellenir.
* Backtest/paper sonuçlarıyla strateji başına **rolling win rate** tutulur; UI’da gösterilir.

17.4 İçe aktarma (import) şablonu

* *docs/strategies.csv* ve *docs/strategies.json* dosyaları sisteme yüklenebilir.
* İçe aktarma sırasında ID çakışması varsa mevcut kayıt üzerine yazma veya yeni ID verme seçenekleri sunulur.

17.5 Katalog — (kullanıcı tarafından daha sonra eklenecek)

Bu bölüm, kullanıcının sağlayacağı strateji satırları ile daha sonra doldurulacaktır. Şu an için boş bırakıldı.

17.6 Toplu giriş (batch) önerisi

* Mesaj/ekleme başına **~30 satır** önerilir (okunabilirlik ve sürat için). 50 satıra kadar işlenebilir.
* Daha büyük listeler için *.csv*/*.json* dosyası ile içe aktarma tercih edilmeli (*docs/strategies.csv*).

18) Master Teslimat Planı — Parça Bazlı (Tam Kapsam)

Notlar: - Çekirdek dil: **Python 3.10+** (core engine). UI: **Flutter** (Windows odak – Android sonraya ertelendi). Model: **scikit-learn → PyTorch (opsiyonel ileri)**. Veritabanı: **SQLite** + CSV/Parquet. İletişim: **REST/WebSocket (TLS)**. - TOS/yasalara uyum, 2FA bypass yok, anti-bot atlatma yok; yalnızca izinli/resmî yollar. - Multi-device senkronizasyon opsiyonel faz olarak kalır.

18.1 Sistem Mimarisi ve Teknolojiler

* **Core Engine (Python / asyncio)**: Worker Scheduler, Strategy/Signal Providers (plugin), Ensemble/Confidence, Risk Manager, Order Executor, Persistence, Telemetry.
* **Connector**: *aiohttp*/*websockets* ile broker API (varsa resmî). Fonksiyonlar: *login()*, *get\_candles()*, *get\_quote()*, *get\_current\_win\_rate()*, *place\_order()*, *confirm\_order()*, *heartbeat()*.
* **UI (Flutter - Windows)**: Dashboard, İşlem Menüsü, Gelişmiş Ayarlar, Grafik Paneli. Renk teması: Mor, Mavi, Yeşil, Kırmızı, Siyah, Beyaz.
* **Storage**: SQLite (orders, results, metrics, features, configs), CSV/Parquet (ML dataset). Dönen loglar için rotating file handler.
* **Security**: *keyring*/Windows DPAPI; *.env* yalnızca non-secret; TLS, PII-free logging.
* **Packaging**: **Windows Service** (pywin32/nssm) + masaüstü UI (Flutter).

18.2 Çoklu Hesap (Windows) — Tasarım

* Tek PC’de aynı anda **4 ayrı hesap** için eşzamanlı oturum.
* Her hesap **izole profil/kapsül**: ayrı credential vault girişi, ayrı cookie/token deposu, ayrı WebSocket/HTTP oturumu, ayrı SQLite şema önekleri (örn. *acc1\_*, *acc2\_*).
* Hesap bazında bağımsız **Worker havuzu**, **Risk Engine**, **Permit/Win-Threshold**; UI’da **Hesap Geçişi** ve **Toplu Görünüm**.
* Global ve hesap bazlı **parallel limit** UI’dan ayarlanabilir (kullanıcı belirler; varsayılan sabitlenmez).

18.3 Design Tokens (UI Tema)

* **Purple** *#6D28D9*, **Blue** *#2563EB*, **Green** *#16A34A*, **Red** *#DC2626*, **Black** *#111827*, **White** *#F9FAFB*.
* Tipografi: Inter/Roboto; Köşe: *rounded-2xl*; Gölge: *shadow-xl*; Spacing: *p-4 +*.

18.4 Modül Haritası (Kod Düzeyi)

moonlight/├─ core/│ ├─ api/│ ├─ connector/│ ├─ indicators.py│ ├─ strategies/│ ├─ ensemble.py│ ├─ risk.py│ ├─ worker.py│ ├─ storage.py│ ├─ telemetry.py│ ├─ scheduler.py│ └─ main.py├─ ui\_app/ (Flutter Windows)│ ├─ lib/│ │ ├─ screens/│ │ ├─ widgets/│ │ ├─ theme/│ │ └─ services/api\_client.dart├─ docs/├─ data/└─ tests/

18.5 Veri Şemaları (SQLite)

* **orders**(id, ts\_open, account\_id, product, timeframe, direction, amount, permit\_win\_min, permit\_win\_max, client\_req\_id)
* **results**(order\_id, ts\_close, status, pnl, duration\_s, latency\_ms)
* **features**(order\_id, tf, ema9, ema21, rsi14, macd\_hist, boll\_width, atr14, obv, mfi14, adx14, cmf, vwap\_dist, stoch\_k, stoch\_d, ichimoku\_state, supertrend\_state, …)
* **metrics**(scope, key, value, ts)
* **configs**(scope, json\_blob, ts)

18.6 Konfigürasyon Şeması (JSON) — Windows & Çoklu Hesap

{ "ensemble\_threshold": 0.70, "accounts": [ {"id": "acc1", "username": "...", "profile\_store": "profiles/acc1/"}, {"id": "acc2", "username": "...", "profile\_store": "profiles/acc2/"}, {"id": "acc3", "username": "...", "profile\_store": "profiles/acc3/"}, {"id": "acc4", "username": "...", "profile\_store": "profiles/acc4/"} ], "limits": { "max\_parallel\_global": **null**, "max\_parallel\_per\_account": **null** }, "products": [**...**]}

18.7 Ensemble, Risk, Öğrenme, Telemetry, UI, Dağıtım, Test & Backtest

* (Öncesiyle aynı; Windows odaklı düzenlemeler eklendi.)

18.8 Açık Noktalar

* **TOS uygunluğu**: Aynı kişi adına birden fazla hesap veya aynı PC’den birden fazla hesabın eşzamanlı kullanımının **platform kurallarına uygunluğu kullanıcı sorumluluğundadır**. Uygulama teknik olarak destekler; uyumluluk için kullanıcı doğrulaması gerekir.
* **2FA**: Şu an kapalı; **opsiyonel 2FA/OTP** giriş akışı hazır (gerektiğinde etkinleştirilebilir).

18.9 Parça Planı (Windows Odaklı – 30 Parça)

* (Önceki 30’luk plan; Windows odaklı notlarla korunur.)

Parça 1 — Proje Özeti & Kapsam (Windows 10/11 + 4 Hesap)

Bu parça **sadece Windows 10/11** odaklıdır. Android ve çoklu cihaz senkronizasyonu sonraki fazlardadır. Erişim, **Olymp Trade** platformuna **kullanıcının kendi hesap kimlikleri** ile yapılır. **Tek uygulama → aynı anda 4 ayrı hesap** oturumu desteklenir. Varsayılan paralellik üst sınırı **kullanıcı tarafından belirlenecek** (bu alanı kurulumda zorunlu ayar olarak bırakıyoruz; set edilmezse sistem yalnızca **Gözlem/Paper** modunda başlar).

1. Amaç ve Değer Önerisi

MoonLight, Windows’ta çalışan bir sabit zamanlı (Fixed Time) işlem yapay zekâsıdır. Teknik analiz + ensemble + (opsiyonel) ML katmanlarıyla **sinyal üretip** kullanıcı tarafından tanımlanan **risk kuralları** dahilinde **emir** verir. **Çoklu hesap** işletimi, **yüksek özelleştirilebilir UI**, **stabil servis**, **kayıt/öğrenme** ve **gözlemlenebilirlik** ana hedeflerdir.

2. Kapsam (Fonksiyonel)

2.1 Hesap/Oturum Yönetimi

* **4 Hesap Eşzamanlı**: Tek PC’de aynı anda 4 farklı kullanıcı hesabı için izole profil (token/cookie kasası, bağımsız HTTP/WS oturumu, ayrı DB önekleri).
* **Güvenli Giriş**: Kullanıcı adı/e‑posta + parola **Windows DPAPI/Keyring** ile saklanır. **2FA/OTP** opsiyoneldir (UI açılır penceresi, gerekirse etkin).
* **Oturum Sağlığı**: Heartbeat/yeniden bağlanma, token yenileme, hatada kontrollü Retry/Abort.

2.2 Veri Alma ve Zamanlama

* **Mumlar (OHLCV)** + mümkünse **anlık fiyat** ve **payout/win‑rate** bilgisi.
* **TF Destekleri**: 1, 5, 15 dakika. Mum kapanışlarına hizalı tetikleyiciler.
* **Gecikme Ölçümü**: Her çağrıda round‑trip latency log’lanır; eşik aşımlarında guardrail.

2.3 Sinyal Üretimi (Analiz)

* **İndikatörler**: EMA/SMA/WMA/HMA, MACD/PPO, RSI/Stoch/StochRSI, Bollinger/Keltner, ATR, OBV/MFI/VWAP/CMF, ADX, Ichimoku, Supertrend, Donchian, Keltner, Pivot vb.
* **Strateji Kataloğu (Plugin)**: Her strateji bir *signal provider* olarak çalışır; ürün/TF bazında **Aktif/Pasif** ve **parametre override** UI’dan.
* **Ensemble + Confidence**: Oy/ağırlık birleştirme → 0–1 **güven skoru**; **eşik altında emir yok**.

2.4 Emir ve Risk Yönetimi

* **Concurrency Kuralı**: Aynı **(hesap, ürün, TF)** için tek açık işlem. 1/5/15 TF’leri birbirinden bağımsız paralel çalışır.
* **Permit Penceresi**: Payout *permit\_min ≤ R ≤ permit\_max* aralığında değilse **giriş yok**.
* **Win‑Rate Eşiği**: Ürün/TF bazında **kullanıcı ayarlı** (örn. ≥ %70).
* **Lot/Tutar Politikası**: Fixed, Balance Fraction, (ops.) Kelly‑lite; *a\_min/a\_cap* tavanları.
* **Koruma Bariyerleri**: Günlük max kayıp, ardışık kayıp limiti, cool‑down, kill‑switch, circuit‑breaker.
* **Idempotent Emir**: *client\_request\_id* ile çift‑emir engeli; PREPARE→SEND→CONFIRM→SETTLED durum makinesi.

2.5 Öğrenme ve Kayıt

* **Feature Kaydı**: Her işlemde tüm indikatörler + bağlam özellikleri (payout, spread, latency) kaydedilir.
* **Sonuç Kaydı**: win/lose, PnL, süre, latency.
* **Win‑Rate İzleme**: **Demo/gerçek ayrı**; rolling pencereler (N=50/100).
* **Ağırlık Uyarlama**: Strateji/ensemble ağırlıkları son performansa göre adaptif.
* **(Opsiyonel) ML**: sklearn baseline (logistic RF), ileride PyTorch (LSTM/Transformer) inference.

2.6 UI (Windows Masaüstü)

* **Dashboard**: Çoklu hesap üstten seçim + bütünleşik görünüm; aktif işlemler, PnL, win‑rate, DD, latency.
* **İşlem Menüsü**: Ürün/TF seçimi, **win‑rate eşiği**, **permit min‑max**, **lot**, **guardrails**; başlat/durdur/kaldır.
* **Gelişmiş Ayarlar**: Ensemble ağırlıkları, öğrenme seçenekleri, cool‑down, saat filtreleri.
* **Grafik Paneli**: Mum + indikatör overlay; PnL/win‑rate trendleri.
* **Tema**: Mor/Mavi/Yeşil/Kırmızı/Siyah/Beyaz; koyu/açık mod; erişilebilirlik.

3. Kapsam (Fonksiyonel Olmayan)

* **Windows Servis**: Core engine arka planda servis olarak; UI ayrı süreçten REST/WS ile bağlanır.
* **Performans**: 100–300+ eşzamanlı (hesap×ürün×TF) izlemeyi hedefleyen *asyncio* işleyişi; back‑pressure ve rate‑limit kontrolleri.
* **Güvenlik**: DPAPI/keyring, TLS, PII‑free logging, profil izolasyonu (*profiles/accX/*).
* **Uyumluluk**: Platform TOS ve yerel mevzuata uyum; **anti‑bot atlatma yok**.
* **Gözlemlenebilirlik**: Yapılandırılmış JSON log, metrikler, alarm kanalları; crash‑free ≥ 24s hedefi.

4. Çalışma Topolojisi (Konu 3’ün Açıklaması)

* **Core Konumu**: Core **yerel Windows Servisi** olarak çalışır.
* **UI**: Masaüstü uygulaması (Flutter), *localhost* üzerinden core’a bağlanır (REST + WebSocket).
* **Neden Bu Tasarım?** Arka plan sürekliliği (UI kapalıyken de sürdürme), kaynak izolasyonu ve güvenlik.
* **Alternatif**: Gerekirse tek paket EXE içinde gömülü servis; fakat uzun süreli çalıştırmada ayrı servis tercih edilir.

5. Kullanıcı Kontrolü (Konu 4’ün Açıklaması)

* **“Kullanıcı nasıl isterse öyle çalışsın”** ilkesi: Tüm kritik parametreler UI’dan **kullanıcı tarafından ayarlanır**.
* **Demo↔Gerçek**: Dilediğin anda seçilebilir; istersen demo başarı eşiği şartı kaldırılabilir/aktif edilebilir.
* **Paralellik Sınırı**: **Kullanıcı belirler** (global ve hesap bazında). *Varsayılanı boş bırakıyoruz*; set edilmezse core **yalnız gözlem/paper** modunda kalır.

6. Varsayımlar ve Bağımlılıklar

* Broker erişimi için **resmî veya izinli API/kanallar** kullanılabilir olmalı; rate‑limit ve bakım pencereleri olabilir.
* Windows 10/11 üzerinde yönetici olmayan kullanıcı haklarıyla çalışabilmeli; port dinleme yalnızca loopback.
* Donanım: Çoklu iş için yeterli CPU/ram; disk alanı (SQLite + loglar).

7. Kabul Kriterleri (Parça 1)

* **Windows 10/11 odak** + **4 hesap** desteği dokümante.
* **Core = Windows Servisi**, UI = masaüstü istemci topolojisi net.
* **Kullanıcı kontrollü** win‑rate eşiği, permit penceresi, lot ve guardrails tanımlı.
* **Paralellik sınırları kullanıcıya bırakıldı**; set edilmezse **paper mod** davranışı belirli.
* Güvenlik/uyumluluk ve kayıt/öğrenme gereksinimleri net.

8. Başarı Ölçütleri

* **Uptime** ≥ %99 (iş saatleri); **emir turu** < 2s (bağlantıya bağlı).
* **Kaydı tutarlılık**: orders→results→features zincirinde *at‑least‑once*; boot’ta reconcile ile bütünlük.
* **UX**: Tüm ayarlar 1–2 tıkla; **panic/kill‑switch**; anlık metrik akışı (WS).

9. Kullanım Senaryoları (Örnek)

* **S1**: 4 hesapla EURUSD 1/5/15 dakikada, permit %89–%93, win‑threshold %72; 1 dakikada tek açık işlem kuralı.
* **S2**: BTCUSD’de yalnızca 5 dakikalık stratejiler aktif; RVOL yüksekse bekle; günlük max loss 5 birim.
* **S3**: Demo’da 2 hafta paper; 100 işlem penceresinde win‑rate ≥ hedef görülürse gerçek hesaba geçiş.

Parça 2 — Uyumluluk & Güvenlik İlkeleri (Windows 10/11 + 4 Hesap)

Amaç: MoonLight’ın **platform kuralları**, **yerel mevzuat** ve **güvenlik en iyi uygulamaları** ile uyumlu, çoklu hesapta dahi **güvenli ve denetlenebilir** çalışmasını sağlamak.

2.1 Hukuki & Platform Uyum İlkeleri

* **Kendi hesapların**: Yalnızca sana ait Olymp Trade hesaplarına erişim. Üçüncü taraf hesabı erişimi yok.
* **Çoklu hesap (4 oturum)**: Teknik olarak desteklenir; platform **Hizmet Şartları (TOS)** uygunluğu **kullanıcı sorumluluğundadır**.
* **İzinli erişim**: Sadece **resmî/izinli** API/kanallar; **anti‑bot atlatma, tersine mühendislik** yok.
* **Finansal uyarı**: Sonuçlar garanti değildir; yazılım **yatırım tavsiyesi vermez**.

2.2 Kimlik & Oturum Güvenliği

* **Parola kasası**: Windows **DPAPI/keyring**; config’te parola yok, yalnız *keyring\_service* + *username* tutulur.
* **Token yaşam döngüsü**: Kısa ömürlü erişim token’ları + refresh; hesap başına *profiles/accX/* altında **şifreli** saklama.
* **2FA/OTP (opsiyonel)**: Şu an kapalı; gerektiğinde UI modalıyla TOTP/SMS/e‑posta **tek seferlik** kod girişi. Kodlar loglanmaz, bellekte geçici.
* **Oturum izolasyonu**: Her hesap için ayrı HTTP/WS oturumu, çerez/jar ve header seti; **session locking** ile karışma önlenir.

2.3 Ağ Güvenliği

* **TLS zorunlu**: Sertifika doğrulaması açık; **downgrade** engeli.
* **Host allow‑list**: Yanlış yönlendirmeyi azaltmak için isteğe bağlı alan adı sabitleme.
* **Rate‑limit uyumu**: **Exponential backoff + jitter**; gereksiz istek yok; kalp atışı (heartbeat) aralığı sınırlandırılmış.
* **Loopback erişimi**: UI↔Core iletişimi *localhost* üzerinden; Windows Firewall allow‑list kuralı.

2.4 Loglama & Gizlilik

* **PII redaksiyonu**: Kullanıcı adı/e‑posta/token/OTP **maskeleme**; loglara düşmez.
* **Yapılandırılmış log**: JSON log; ayrı kanallar → **trade.log** (işlem olayları), **system.log** (servis ve hatalar).
* **Rotasyon**: Boyut/süre bazlı **rotating**; maksimum dosya adedi UI’dan ayarlanabilir.

2.5 Ayrıcalık ve Erişim İlkeleri

* **En az ayrıcalık**: Core yalnız gerekli dosya/portlara erişir (loopback). Yönetici hakları gerekmez.
* **Profil ayrımı**: *account\_id* alanı tüm tablolarda zorunlu; istenirse her hesap için ayrı DB dosyası.
* **Config değişimi**: UI’dan **onaylı** değişiklik; kritik alanlarda **atomik** devreye alma (hot‑reload ya da kontrollü restart).

2.6 Hata, Retry ve Güvenli Durma

* **Idempotent emir**: *client\_request\_id* ile çift emir önleme. Durum makinesi: *PREPARE → SEND → CONFIRM → (SETTLED | ABORT)*.
* **Retry politikası**: Sadece **ağ/5xx** sınıfı; finansal yan etkili noktalarda **tekrar yok** (*at‑most‑once*).
* **Fail‑safe**: Bağlantı kopması, latency sapması, veri tutarsızlığı tespitinde **otomatik durdurma** + kullanıcı uyarısı.

2.7 Guardrails (Bariyerler)

* **Kill‑switch / Panic**: UI’da tek tıkla **tüm emir süreçlerini durdur**.
* **Circuit‑breaker**:
  + Günlük kayıp ≥ limit → durdur.
  + Ardışık kayıp ≥ limit → durdur + cool‑down artır.
  + Permit penceresi dışı payout/win‑rate → **yeni işlem yok**.
  + Latency *> abort\_ms* → yeni işlem engeli.
* **Concurrency kuralı**: *(hesap, ürün, TF)* başına **tek açık işlem**; 1/5/15 bağımsız.
* **Paralellik sınırları**: **Kullanıcı belirler**. **Belirlenmemişse core “paper/gözlem” modunda** kalır.

2.8 Tedarik Zinciri Güvenliği & Güncelleme

* **İmzalı paket**: Kurulum ve güncellemeler kod‑imzalı.
* **Bağımlılıklar**: *requirements.txt* **sabit sürüm** + hash doğrulaması; SBOM (opsiyonel) ve güvenlik taraması.
* **Sürüm bütünlüğü**: Release SHA‑256 doğrulaması; isteğe bağlı **GPG imzalı** commitler.

2.9 Veri Koruma, Yedek & Arşiv

* **Disk koruması**: BitLocker önerilir; kritik kolonlar için uygulama‑katmanı şifreleme (ops.).
* **Yedekleme**: Günlük şifreli *.backup*; anahtar yönetimi kullanıcıda.
* **Arşiv**: 90+ gün **features** satırlarını Parquet arşivine taşı, DB’yi incelt.

2.10 Telemetry & Dışa Veri

* **Varsayılan KAPALI**: Dışa hiçbir telemetry gönderilmez. Kullanıcı açık izin verirse **anonimleştirilmiş** metrikler gönderilebilir.

2.11 Go‑Live Kontrol Listesi

* TOS/yerel mevzuat kontrol edildi (kullanıcı).
* 4 hesap profili izolasyonu doğrulandı.
* DPAPI/keyring kasası çalışıyor.
* TLS & sertifika doğrulama açık.
* PII redaksiyonu test edildi.
* Kill‑switch / circuit‑breaker senaryoları geçti.
* Idempotent emir ve retry/abort akışları test edildi.
* Güncelleme + bağımlılık güvenliği doğrulandı.

2.12 Kabul Kriterleri (Parça 2)

* Güvenlik/uyumluluk ilkeleri **uygulanabilir** ve **test edilebilir** durumda.
* Çoklu hesap izolasyonu, anahtar kasası, TLS, log gizliliği, guardrails ve idempotent emir akışı **tasarlandı**.
* Paralellik sınırı **kullanıcı kontrolünde**; set edilmediğinde core’un **paper modu** davranışı net.

Parça 3 — Yüksek Seviyeli Mimari (Windows Servis + 4 Hesap + UI)

Hedef: Bileşenleri, veri/denetim akışını, süreçleri, eşzamanlılığı ve arayüz sözleşmelerini netleştirmek. Bu katman, sonraki parçalardaki detayların çerçevesidir.

3.1 Mimari Katmanlar

1. **UI (Flutter / Windows masaüstü)**
   * Ekranlar: Dashboard, İşlem Menüsü, Gelişmiş Ayarlar, Strateji Kataloğu, Grafik Paneli, Loglar.
   * İletişim: *localhost* üstünden **REST + WebSocket** (loopback, opsiyonel TLS).
   * Rol: Ayarları düzenler, başlat/durdur komutları, anlık metrik/günlükleri gösterir.
2. **Core Engine (Python / asyncio, Windows Service)**
   * **Scheduler**: Hesap×Ürün×TF için worker oluşturma/yaşam döngüsü.
   * **Workers**: Sinyal → guardrails → emir akışı. (Her worker anahtarı = *(account, product, timeframe)*)
   * **Strategy Engine (plugin)**: Signal provider’lar (ID kataloğu)
   * **Ensemble & Confidence**: Ağırlıklandırma + eşiğe göre karar
   * **Risk Manager**: Lot/tutar, permit, guardrails, cool‑down
   * **Order Executor**: Idempotent emir, retry/abort, latency ölçümü
   * **Connector(s)**: Olymp Trade izinli uçlarıyla HTTP/WS
   * **Storage**: SQLite (orders, results, features, metrics, configs) + CSV/Parquet
   * **Telemetry**: Metrikler, structured log, alarm üretimi
3. **Profil/Account İzolasyonu**
   * *profiles/accX/* → cookie/token kasası, session ayarları, config override, DB sorgularında *account\_id* filtresi.

3.2 Veri & Denetim Akışı

UI --(REST: /config, /start, /stop, /accounts, /workers, /orders)--→ CoreUI ←────────────(WS: metrics, logs, trade\_updates, alerts)────────── CoreCore ↔ Connector (HTTP/WS): login, candles, quotes/payout, place/confirm orderCore → Storage: orders/results/features/metrics (WAL + indeks)

3.3 Süreç Yaşam Döngüsü

**A) Servis başlatma**  
1. Config yükle → JSON şema doğrula → keyring’den parolaları çek.  
2. Hesap profilleri aç → Connector *login()* (gerekirse OTP).  
3. Scheduler, aktif (hesap×ürün×TF) için **worker**’ları ayağa kaldırır.  
4. UI bağlanır; **/status** ve **/metrics** akışı başlar.

**B) Worker döngüsü (her *(acc, prod, tf)*)**  
1. **Veri Çek**: *get\_candles(n)* (+ varsa quote/payout).  
2. **Feature Hesapla**: İndikatörler/kanallar/volatilite/hacim.  
3. **Sinyal**: Aktif stratejilerden oy/skor al.  
4. **Ensemble**: Kombine et → *confidence∈[0,1]*.  
5. **Guardrails**: permit penceresi, win‑threshold, concurrency, cool‑down, limitler.  
6. **Emir** *(şart sağlanırsa)*: *place\_order()* → *confirm\_order()*; latency ölç.  
7. **Kayıt & Yayın**: orders/results/features yaz; metrik/log/olayları WS ile UI’a gönder.

**C) Hot‑reload**  
- *config.json* değişirse güvenli alanlar **anında** uygulanır; kapsamlı değişimde etkilenen worker’lar **kontrollü yeniden başlatılır**.

3.4 Eşzamanlılık, Kilitleme ve Performans

* **asyncio** tabanlı işleyiş; her hesap için bağımsız worker havuzu.
* **Concurrency kuralı**: Aynı *(hesap, ürün, TF)* için tek açık işlem (re‑entrancy lock).
* **Kuyruklar**: *asyncio.Queue* ile sinyal/emir/telemetry back‑pressure yönetimi.
* **Rate‑limit/Retry**: Connector seviyesinde throttling, **exponential backoff + jitter**.
* **Zamanlama**: TF=1 için ~250ms tik; TF=5/15 mum kapanışına hizalı.
* **Hedef kapasite**: 100–300+ aktif anahtar; CPU <%70 (donanıma bağlı); IO beklemelerinde bloklamasız.

3.5 UI ↔ Core API Taslağı (Loopback)

**REST (örnek)**  
- *GET /status* → servis, hesap, worker özetleri  
- *GET /accounts* → bağlı hesaplar/profiller  
- *POST /start* / *POST /stop* → global veya scoped (hesap/ürün/TF)  
- *GET /workers* → aktif *(acc, prod, tf)* listesi  
- *POST /orders/cancel* → (platform destekliyse) erken kapatma isteği  
- *GET /logs?level=INFO&tail=500* → son loglar  
- *PUT /config* → doğrulama + atomik devreye alma (sadece güvenli alanlar)

**WebSocket (kanallar)**  
- *metrics* (win‑rate, PnL, DD, latency, rvol, adx vb.)  
- *trade\_updates* (NEW/SETTLED/ABORT)  
- *alerts* (guardrail tetik, kill‑switch, bağlantı)  
- *logs* (system/trade, redakte edilmiş)

3.6 Modül ve Dizinyapısı (özet)

moonlight/├─ core/│ ├─ api/ (REST/WS loopback server)│ ├─ connector/ (olymp interface + mock)│ ├─ indicators/ (basic, advanced, states)│ ├─ strategies/ (plugin providers)│ ├─ ensemble.py│ ├─ risk.py│ ├─ worker.py│ ├─ scheduler.py│ ├─ storage.py│ ├─ telemetry.py│ └─ main.py (service bootstrap)├─ ui\_app/ (Flutter desktop)│ ├─ lib/screens, widgets, theme, services├─ data/ (trades.db, ml\_dataset.csv)├─ docs/ (config.schema.json, examples)└─ tests/

3.7 Durum Makineleri (özet)

* **Order FSM**: *IDLE → PREPARE → SEND → (RETRY?) → CONFIRM → (SETTLED | ABORT)*
* **Worker FSM**: *INIT → RUNNING → (RELOAD?) → STOPPING → STOPPED*
* **Connector FSM**: *DISCONNECTED → AUTHENTICATING → CONNECTED → (REAUTH/RECONNECT)*

3.8 Güvenlik/İzolasyon Bağlantıları

* UI yalnız **loopback** dinler; dış ağdan görünmez.
* Her hesap için **ayrı cookie/token** ve oturum; DB’de *account\_id* zorunludur.
* Keyring/DPAPI’de parolalar; loglarda PII maskelenir.

3.9 Performans Bütçeleri & Telemetry

* **Emir round‑trip** hedefi: <2s (ağa bağlı).
* **Worker döngüsü**: TF=1’de <1s işlem.
* **Metrikler**: win‑rate (demo/gerçek ayrı), expectancy, DD, latency, error rate; **WS ile anlık yayın**.

3.10 Kabul Kriterleri (Parça 3)

* Katmanlar, akışlar, API taslağı, FSM’ler ve kilitleme stratejisi **tanımlandı**.
* Performans hedefleri ve telemetry gereksinimleri **belirlendi**.
* Windows servis + loopback UI topolojisi **net**.

Parça 4 — Konfigürasyon Şeması (JSON/YAML) + Örnekler

Amaç: MoonLight’ın davranışını tek bir yapılandırma kaynağından kontrol etmek; **Windows + 4 hesap** senaryosunda güvenli, okunabilir ve hot‑reload uyumlu bir şema sağlamak.

4.1 Tasarım İlkeleri

* **Tek doğruluk kaynağı**: *config.json* (alternatif *config.yaml*).
* **Hiyerarşi**: Global → Hesap → Ürün → Timeframe → Strateji.
* **Hot‑reload**: Güvenli alanlar çalışma sırasında değiştirilebilir (örn. win‑threshold); kritik alanlar kontrollü yeniden başlatma ister (örn. DB yolu).
* **Sırlar dışarıda**: Parola/token **DPAPI/Keyring**’de; config içinde yalnız **profil/anahtar adı** tutulur.
* **Versiyonlama**: *config\_version* ile göç (migration) kolaylığı.

4.2 Alanların Anlamı (özet)

* *ensemble\_threshold* (0–1): Strateji birleşik güven eşiği.
* *limits*: Paralellik ve guardrail sınırları. (Set edilmezse **paper/gözlem** modu.)
* *accounts[]*: Çoklu hesap profilleri; her biri ayrı oturum/kasa/dizin.
* *products[]*: Ürünler; TF listesi ve strateji seçimi içerir.
* *timeframes[].win\_threshold* (0–1): O TF’de **başarı eşiği**.
* *timeframes[].permit\_min/permit\_max* (%): Payout/win‑rate penceresi.
* *risk*: Lot/SL/TP ve guardrail varsayılları.
* *engine*: Worker periyotları, kuyruk boyutları, latency gardı.
* *storage*: SQLite/CSV yol ve ayarları.
* *logging*: Log seviyesi, rotasyon, dosya yolları.
* *ui*: Tema/renkler/özellik bayrakları.

4.3 JSON Şeması (kısaltılmış)

{ "$schema": "http://json-schema.org/draft-07/schema#", "title": "MoonLight Config", "type": "object", "required": ["config\_version", "accounts", "products"], "properties": { "config\_version": {"type": "string", "pattern": "^1**\.\d**+**\.\d**+$"}, "ensemble\_threshold": {"type": "number", "minimum": 0, "maximum": 1, "default": 0.70}, "limits": { "type": "object", "properties": { "max\_parallel\_global": {"type": ["integer", "null"], "minimum": 1}, "max\_parallel\_per\_account": {"type": ["integer", "null"], "minimum": 1}, "max\_daily\_loss": {"type": ["number", "null"]}, "max\_consecutive\_losses": {"type": ["integer", "null"]} }, "additionalProperties": **false** }, "accounts": { "type": "array", "minItems": 1, "items": { "type": "object", "required": ["id", "username", "profile\_store"], "properties": { "id": {"type": "string"}, "username": {"type": "string"}, "profile\_store": {"type": "string"}, "keyring\_service": {"type": "string", "default": "moonlight-olymp"}, "session": { "type": "object", "properties": { "http\_user\_agent": {"type": "string"}, "ws\_heartbeat\_s": {"type": "integer", "default": 20} }, "additionalProperties": **false** } }, "additionalProperties": **false** } }, "products": { "type": "array", "items": { "type": "object", "required": ["product", "timeframes"], "properties": { "product": {"type": "string"}, "enabled": {"type": "boolean", "default": **true**}, "strategies": {"type": "array", "items": {"type": "integer"}}, "timeframes": { "type": "array", "items": { "type": "object", "required": ["tf", "enabled"], "properties": { "tf": {"type": "integer", "enum": [1,5,15]}, "enabled": {"type": "boolean"}, "win\_threshold": {"type": "number", "minimum": 0, "maximum": 1, "default": 0.70}, "permit\_min": {"type": "number", "minimum": 0, "maximum": 100}, "permit\_max": {"type": "number", "minimum": 0, "maximum": 100}, "risk": { "type": "object", "properties": { "lot": {"type": "number", "minimum": 0}, "tp\_R": {"type": ["number", "null"]}, "sl\_ATR\_mult": {"type": ["number", "null"]} }, "additionalProperties": **false** } }, "additionalProperties": **false** } } }, "additionalProperties": **false** } }, "risk": { "type": "object", "properties": { "default\_lot": {"type": "number", "default": 1}, "default\_tp\_R": {"type": ["number", "null"], "default": 1.5}, "default\_sl\_ATR\_mult": {"type": ["number", "null"], "default": 1.2} }, "additionalProperties": **false** }, "engine": { "type": "object", "properties": { "queue\_maxsize": {"type": "integer", "default": 1000}, "latency\_warn\_ms": {"type": "integer", "default": 800}, "latency\_abort\_ms": {"type": "integer", "default": 2500}, "tick\_interval\_ms": {"type": "integer", "default": 250} }, "additionalProperties": **false** }, "storage": { "type": "object", "properties": { "sqlite\_path": {"type": "string", "default": "data/trades.db"}, "dataset\_csv": {"type": "string", "default": "data/ml\_dataset.csv"} }, "additionalProperties": **false** }, "logging": { "type": "object", "properties": { "level": {"type": "string", "enum": ["DEBUG","INFO","WARN","ERROR"], "default": "INFO"}, "file": {"type": "string", "default": "logs/moonlight.log"}, "rotate\_mb": {"type": "integer", "default": 10}, "keep\_files": {"type": "integer", "default": 7} }, "additionalProperties": **false** }, "ui": { "type": "object", "properties": { "theme": {"type": "string", "enum": ["dark","light"], "default": "dark"}, "colors": {"type": "object"} }, "additionalProperties": **false** } }, "additionalProperties": **false**}

4.4 Örnek *config.json* (Windows + 4 hesap)

{ "config\_version": "1.0.0", "ensemble\_threshold": 0.72, "limits": { "max\_parallel\_global": **null**, "max\_parallel\_per\_account": **null**, "max\_daily\_loss": 5, "max\_consecutive\_losses": 5 }, "accounts": [ {"id": "acc1", "username": "user1@mail", "profile\_store": "profiles/acc1/", "keyring\_service": "moonlight-olymp"}, {"id": "acc2", "username": "user2@mail", "profile\_store": "profiles/acc2/"}, {"id": "acc3", "username": "user3@mail", "profile\_store": "profiles/acc3/"}, {"id": "acc4", "username": "user4@mail", "profile\_store": "profiles/acc4/"} ], "products": [ { "product": "EURUSD", "enabled": **true**, "strategies": [5,6,14,24,25,32], "timeframes": [ {"tf": 1, "enabled": **true**, "win\_threshold": 0.72, "permit\_min": 89, "permit\_max": 93, "risk": {"lot": 1, "tp\_R": 1.5, "sl\_ATR\_mult": 1.2} }, {"tf": 5, "enabled": **true**, "win\_threshold": 0.72, "permit\_min": 90, "permit\_max": 93, "risk": {"lot": 1, "tp\_R": 1.8, "sl\_ATR\_mult": 1.2} }, {"tf": 15, "enabled": **false**, "win\_threshold": 0.70, "permit\_min": 0, "permit\_max": 100, "risk": {"lot": 1, "tp\_R": **null**, "sl\_ATR\_mult": **null**} } ] }, { "product": "BTCUSD", "enabled": **true**, "strategies": [14,25,35], "timeframes": [ {"tf": 1, "enabled": **true**, "win\_threshold": 0.75, "permit\_min": 85, "permit\_max": 95, "risk": {"lot": 1, "tp\_R": 2.0, "sl\_ATR\_mult": 1.5} }, {"tf": 5, "enabled": **true**, "win\_threshold": 0.74, "permit\_min": 86, "permit\_max": 94, "risk": {"lot": 1, "tp\_R": 1.8, "sl\_ATR\_mult": 1.3} } ] } ], "risk": {"default\_lot": 1, "default\_tp\_R": 1.5, "default\_sl\_ATR\_mult": 1.2}, "engine": {"queue\_maxsize": 2000, "latency\_warn\_ms": 800, "latency\_abort\_ms": 2500, "tick\_interval\_ms": 250}, "storage": {"sqlite\_path": "data/trades.db", "dataset\_csv": "data/ml\_dataset.csv"}, "logging": {"level": "INFO", "file": "logs/moonlight.log", "rotate\_mb": 10, "keep\_files": 7}, "ui": {"theme": "dark", "colors": {"primary": "#6D28D9", "accent": "#2563EB"}}}

4.5 Aynı yapı YAML örneği

config\_version**:** "1.0.0"ensemble\_threshold**:** 0.72limits**:** max\_parallel\_global**:** null max\_parallel\_per\_account**:** null max\_daily\_loss**:** 5 max\_consecutive\_losses**:** 5accounts**:** **-** id**:** acc1 username**:** user1@mail profile\_store**:** profiles/acc1/ keyring\_service**:** moonlight-olymp **-** id**:** acc2 username**:** user2@mail profile\_store**:** profiles/acc2/products**:** **-** product**:** EURUSD enabled**:** true strategies**:** **[**5**,**6**,**14**,**24**,**25**,**32**]** timeframes**:** **-** tf**:** 1 enabled**:** true win\_threshold**:** 0.72 permit\_min**:** 89 permit\_max**:** 93 risk**:** **{** lot**:** 1**,** tp\_R**:** 1.5**,** sl\_ATR\_mult**:** 1.2 **}** **-** tf**:** 5 enabled**:** true win\_threshold**:** 0.72 permit\_min**:** 90 permit\_max**:** 93 risk**:** **{** lot**:** 1**,** tp\_R**:** 1.8**,** sl\_ATR\_mult**:** 1.2 **}** **-** product**:** BTCUSD enabled**:** true strategies**:** **[**14**,**25**,**35**]** timeframes**:** **-** tf**:** 1 enabled**:** true win\_threshold**:** 0.75 permit\_min**:** 85 permit\_max**:** 95 risk**:** **{** lot**:** 1**,** tp\_R**:** 2.0**,** sl\_ATR\_mult**:** 1.5 **}**

4.6 Hot‑Reload Kuralları

* **Yeniden başlatma gerektirmez**: *win\_threshold*, *permit\_\**, *strategies* (aktif/pasif), *risk.lot*, *logging.level*.
* **Kontrollü yeniden başlatma**: Ürün/TF ekleme‑çıkarma, *ensemble\_threshold*, *engine.tick\_interval\_ms*.
* **Servis yeniden başlatma**: *storage.sqlite\_path*, *accounts[]* (profil yolları), büyük topoloji değişimleri.

4.7 Geçerlilik Kuralları (validation)

* *permit\_min ≤ permit\_max* ve her ikisi de [0,100].
* *win\_threshold ≥ ensemble\_threshold* önerilir (aksi durumda uyarı).
* *(ürün,TF)* **tek açık işlem** kuralı sabittir; config ile artırılamaz.
* *max\_daily\_loss*/*max\_consecutive\_losses* **null** ise ilgili guardrail pasif.

4.8 Sırlar (Keyring) — Kullanım Notu

* Config’te **yalnız** *accounts[].username* ve *keyring\_service* saklanır.
* Parola kurulumda tek seferlik: *keyring.set\_password("moonlight-olymp", "user1@mail", "PAROLA")*.
* Core çalışırken *keyring.get\_password(...)* ile çeker; disk üzerinde parola **yoktur**.

4.9 Kabul Kriterleri (Parça 4)

* JSON/YAML şeması ve örnekler **eksiksiz**.
* Hot‑reload alanları ve yeniden başlatma kuralları **belirgin**.
* 4 hesap ve TF/ürün bazlı ayarlar **kapsandı**.
* Keyring/DPAPI modelinin uygulanma yolu **net**.

Parça 5 — Veri Modeli (SQLite) & Storage API

Amaç: İşlemler, sonuçlar, özellikler (features), metrikler ve öğrenme verisi için **sağlam, performanslı ve kurtarmaya elverişli** bir veri katmanı tasarlamak; Windows üzerinde tek dosyalı SQLite ile çalışıp gerektiğinde CSV/Parquet’e dışa aktarımı desteklemek.

5.1 Tasarım İlkeleri

* **Güvenilirlik:** At‑least‑once kayıt, idempotent upsert, çökme sonrası kurtarma (**reconcile**).
* **Performans:** WAL modu, uygun indeksler, toplu ekleme (batch), tek yazıcı prensibi.
* **Esneklik:** Sabit kolonlar + *extras\_json* ile şema esnekliği (SQLite JSON1).
* **İzole Profiller:** *account\_id* alanı tüm tablolarda zorunlu; 4 hesap aynı DB’de **şema ortak**, veriler *account\_id* ile ayrılır. (İstersen hesap başına ayrı DB de desteklenir.)

5.2 SQLite Ayarları (Önerilen)

* **WAL modu:** *PRAGMA journal\_mode=WAL;*
* **Senkronizasyon:** *PRAGMA synchronous=NORMAL;* (yükte) → kritik bölümde *FULL*.
* **Yabancı anahtar:** *PRAGMA foreign\_keys=ON;*
* **Tarih alanları:** Milisaniye epoch (*INTEGER*) + gerektikçe ISO string üretimi (VIEW ile).
* **Şifreleme:** Disk koruması OS seviyesinde (BitLocker). Gerekirse uygulama tarafı kolon şifreleme.

5.3 Şema (DDL)

*-- ORDERS: Emir verildiği anda yazılır***CREATE** **TABLE** **IF** **NOT** **EXISTS** orders ( **id** TEXT **PRIMARY** **KEY**, ts\_open\_ms INTEGER **NOT** **NULL**, account\_id TEXT **NOT** **NULL**, product TEXT **NOT** **NULL**, timeframe INTEGER **NOT** **NULL** **CHECK**(timeframe **IN** (1,5,15)), direction TEXT **NOT** **NULL** **CHECK**(direction **IN** ('call','put')), amount REAL **NOT** **NULL**, client\_req\_id TEXT **UNIQUE** **NOT** **NULL**, permit\_win\_min REAL, permit\_win\_max REAL, extras\_json TEXT); *-- RESULTS: Emir kapandığında yazılır (1:1)***CREATE** **TABLE** **IF** **NOT** **EXISTS** results ( order\_id TEXT **PRIMARY** **KEY** **REFERENCES** orders(**id**) **ON** **DELETE** **CASCADE**, ts\_close\_ms INTEGER **NOT** **NULL**, status TEXT **NOT** **NULL** **CHECK**(status **IN** ('win','lose','abort')), pnl REAL **NOT** **NULL**, duration\_ms INTEGER, latency\_ms INTEGER, extras\_json TEXT); *-- FEATURES: İşlem anındaki özellikler (sabit kolonlar + esnek JSON)***CREATE** **TABLE** **IF** **NOT** **EXISTS** features ( order\_id TEXT **PRIMARY** **KEY** **REFERENCES** orders(**id**) **ON** **DELETE** **CASCADE**, account\_id TEXT **NOT** **NULL**, timeframe INTEGER **NOT** **NULL**, ema9 REAL, ema21 REAL, rsi14 REAL, macd\_hist REAL, boll\_width REAL, atr14 REAL, obv REAL, mfi14 REAL, adx14 REAL, cmf REAL, vwap\_dist REAL, stoch\_k REAL, stoch\_d REAL, ichimoku\_state TEXT, supertrend\_state TEXT, extras\_json TEXT); *-- METRICS: Zaman içinde izlenen değerler (winrate, DD, latency, vb.)***CREATE** **TABLE** **IF** **NOT** **EXISTS** metrics ( **id** INTEGER **PRIMARY** **KEY** AUTOINCREMENT, **scope** TEXT **NOT** **NULL**, *-- global / acc:<id> / prod:<symbol> / tf:<n>* **key** TEXT **NOT** **NULL**, value REAL **NOT** **NULL**, ts\_ms INTEGER **NOT** **NULL**, tags TEXT, **UNIQUE**(**scope**, **key**, ts\_ms)); *-- STRATEGY PERFORMANCE: Ağırlık güncelleme için özetler***CREATE** **TABLE** **IF** **NOT** **EXISTS** strategy\_perf ( strategy\_id INTEGER **NOT** **NULL**, account\_id TEXT **NOT** **NULL**, product TEXT **NOT** **NULL**, timeframe INTEGER **NOT** **NULL**, window\_n INTEGER **NOT** **NULL**, wins INTEGER **NOT** **NULL** **DEFAULT** 0, losses INTEGER **NOT** **NULL** **DEFAULT** 0, updated\_ms INTEGER **NOT** **NULL**, **PRIMARY** **KEY**(strategy\_id, account\_id, product, timeframe, window\_n)); *-- İNDEKSLER***CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_orders\_acc\_ts **ON** orders(account\_id, ts\_open\_ms);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_orders\_prod\_tf **ON** orders(product, timeframe);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_results\_status\_ts **ON** results(status, ts\_close\_ms);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_features\_acc\_tf **ON** features(account\_id, timeframe);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_metrics\_scope\_key\_ts **ON** metrics(**scope**, **key**, ts\_ms);

Görünümler (Kolay raporlama)

**CREATE** **VIEW** **IF** **NOT** **EXISTS** v\_trades **AS****SELECT** o.**id**, o.account\_id, o.product, o.timeframe, o.direction, o.amount, o.ts\_open\_ms, r.ts\_close\_ms, r.status, r.pnl, r.duration\_ms, r.latency\_ms**FROM** orders o **LEFT** **JOIN** results r **ON** r.order\_id = o.**id**;

5.4 Storage API (Python, *aiosqlite*)

*# core/storage.py***import** json, aiosqlite**from** typing **import** Dict, Any, Optional, List**class** Storage: **def** \_\_init\_\_(self, db\_path: str): self.db\_path = db\_path **async** **def** init(self): **async** **with** aiosqlite.connect(self.db\_path) **as** db: **await** db.execute("PRAGMA journal\_mode=WAL;") **await** db.execute("PRAGMA foreign\_keys=ON;") *# DDL uygulanır (yukarıdaki SQL)* **await** db.commit() **async** **def** save\_order(self, order: Dict[str, Any]) -> None: sql = """ INSERT OR IGNORE INTO orders(id, ts\_open\_ms, account\_id, product, timeframe, direction, amount, client\_req\_id, permit\_win\_min, permit\_win\_max, extras\_json) VALUES(:id,:ts\_open\_ms,:account\_id,:product,:timeframe,:direction,:amount,:client\_req\_id,:permit\_win\_min,:permit\_win\_max,:extras\_json); """ **async** **with** aiosqlite.connect(self.db\_path) **as** db: payload = {\*\*order, 'extras\_json': json.dumps(order.get('extras', {}))} **await** db.execute(sql, payload) **await** db.commit() **async** **def** save\_features(self, order\_id: str, feats: Dict[str, Any], account\_id: str, timeframe: int) -> None: base\_cols = ['ema9','ema21','rsi14','macd\_hist','boll\_width','atr14','obv','mfi14','adx14','cmf','vwap\_dist','stoch\_k','stoch\_d','ichimoku\_state','supertrend\_state'] payload = { 'order\_id': order\_id, 'account\_id': account\_id, 'timeframe': timeframe, \*\*{k: feats.get(k) **for** k **in** base\_cols}, 'extras\_json': json.dumps(feats) } cols = ','.join(payload.keys()) params = ':' + ',:'.join(payload.keys()) sql = f"INSERT OR REPLACE INTO features({cols}) VALUES({params});" **async** **with** aiosqlite.connect(self.db\_path) **as** db: **await** db.execute(sql, payload) **await** db.commit() **async** **def** save\_result(self, result: Dict[str, Any]) -> None: sql = """ INSERT OR REPLACE INTO results(order\_id, ts\_close\_ms, status, pnl, duration\_ms, latency\_ms, extras\_json) VALUES(:order\_id,:ts\_close\_ms,:status,:pnl,:duration\_ms,:latency\_ms,:extras\_json); """ **async** **with** aiosqlite.connect(self.db\_path) **as** db: payload = {\*\*result, 'extras\_json': json.dumps(result.get('extras', {}))} **await** db.execute(sql, payload) **await** db.commit() **async** **def** record\_metric(self, scope: str, key: str, value: float, ts\_ms: int, tags: Optional[str] = None): sql = "INSERT OR IGNORE INTO metrics(scope, key, value, ts\_ms, tags) VALUES(?,?,?,?,?);" **async** **with** aiosqlite.connect(self.db\_path) **as** db: **await** db.execute(sql, (scope, key, value, ts\_ms, tags)) **await** db.commit() **async** **def** rolling\_winrate(self, account\_id: str, product: str, timeframe: int, last\_n: int = 100) -> Optional[float]: sql = """ SELECT AVG(CASE WHEN status='win' THEN 1.0 WHEN status='lose' THEN 0.0 ELSE NULL END) FROM ( SELECT r.status FROM orders o JOIN results r ON r.order\_id = o.id WHERE o.account\_id=? AND o.product=? AND o.timeframe=? ORDER BY r.ts\_close\_ms DESC LIMIT ? ) """ **async** **with** aiosqlite.connect(self.db\_path) **as** db: **async** **with** db.execute(sql, (account\_id, product, timeframe, last\_n)) **as** cur: row = **await** cur.fetchone() **return** row[0] **if** row **else** None **async** **def** consec\_losses(self, account\_id: str, product: str, timeframe: int, last\_n: int = 50) -> int: sql = """ SELECT status FROM ( SELECT r.status FROM orders o JOIN results r ON r.order\_id=o.id WHERE o.account\_id=? AND o.product=? AND o.timeframe=? ORDER BY r.ts\_close\_ms DESC LIMIT ? ); """ **async** **with** aiosqlite.connect(self.db\_path) **as** db: **async** **with** db.execute(sql, (account\_id, product, timeframe, last\_n)) **as** cur: rows = [r[0] **for** r **in** **await** cur.fetchall()] cnt = 0 **for** s **in** rows: **if** s == 'lose': cnt += 1 **else**: **break** **return** cnt

Atomik Akış (Order + Features + Result)

* **Akış:** *save\_order()* → *save\_features()* → (işlem bittiğinde) *save\_result()*
* **Kısmi yazımda kurtarma:** Boot’ta **reconcile** görevi *orders* içinde *results* olmayanları kontrol eder; connector’dan teyit alıp *results* tablosunu tamamlar.

5.5 CSV/Parquet Dataset (ML)

* **CSV satırı:** *id, ts, account, product, tf, direction, amount, ema9, ema21, rsi14, macd\_hist, boll\_width, atr14, obv, mfi14, status*
* **Politika:** Günlük batch append; büyük dosyalar için per‑day dosyalar; uzun vadede **Parquet** arşivi (sütun sıkıştırma)
* **UI dışa aktarım:** “Veriyi dışa aktar (CSV/Parquet)” butonu.

5.6 Bakım & Yedekleme

* **PRAGMA optimize;** periyodik *VACUUM* (uygulama idle iken)
* **.backup API:** Günlük şifreli yedek (isteğe bağlı)
* **Arşivleme:** 90 günden eski **features** satırlarını Parquet’e taşı → DB’yi incelt
* **Bütünlük testleri:** Başlangıçta *PRAGMA integrity\_check;* (opsiyonel)

5.7 Performans İpuçları

* Tek süreçte **tek yazıcı** kalıbı; Storage istekleri bir *asyncio.Queue* üzerinden **batch** işlenebilir.
* Toplu yazımlarda *BEGIN IMMEDIATE; ...; COMMIT;*
* Sadece gereken kolon setlerini doldur; geri kalanı *extras\_json*.

5.8 Kabul Kriterleri (Parça 5)

* DDL, indeksler ve görünüm **hazır**.
* Storage API, idempotent upsert ve rolling win‑rate / consec‑losses sorguları **tanımlandı**.
* Kurtarma (reconcile), WAL, yedekleme ve bakım politikası **belirlendi**.
* ML dataset dışa aktarım (CSV/Parquet) **tasarlandı**.

Parça 6 — Connector Tasarımı (async) & Mock (Windows 10/11, 4 Hesap)

Amaç: Olymp Trade’e **izinli/uyumlu** erişim için **asenkron bağlayıcı (connector)** tasarlamak; çoklu hesabı destekleyen, idempotent emir akışlı, geri‑dönüşe dayanıklı bir mimari oluşturmak. Bu bölüm **gerçek uç nokta bilgisi içermez**; yalnızca **sözleşmeler, akışlar ve iskelet kod** verir. Uygulama, yalnızca **resmî/izinli API** var ise kullanır; anti‑bot veya koruma atlatma yapılmaz.

6.1 Kapsam ve Hedefler

* **Per‑Account Connector**: Her hesap için bağımsız *Connector* örneği (oturum/cookie/token izole).
* **HTTP + WS**: Uygunsa WebSocket ile akış (quotes/olaylar); aksi durumda periyodik poll.
* **İşlevler**: *login*, *refresh\_token*, *get\_candles*, *get\_quote*, *get\_current\_win\_rate* (payout), *place\_order*, *confirm\_order*, *heartbeat*.
* **İdempotensi**: *client\_request\_id* ile **at‑most‑once** emir davranışı.
* **Rate‑limit**: Throttling + **exponential backoff + jitter**.
* **Hata sınıfları**: Auth/Throttle/Maintenance/Network/Format ayrımı.

6.2 API Sözleşmesi (Protocol)

*# core/connector/interface.py***from** typing **import** List, Dict, Any, Protocol, Optional**class** Connector(Protocol): account\_id: str **async** **def** login(self, username: str, password: str, otp: Optional[str] = None) -> None: ... **async** **def** refresh\_token(self) -> None: ... **async** **def** get\_candles(self, product: str, timeframe: int, n: int = 200, until\_ms: Optional[int] = None) -> List[Dict[str, Any]]: ... **async** **def** get\_current\_win\_rate(self, product: str) -> float: ... *# payout in [0,100]* **async** **def** get\_quote(self, product: str) -> Dict[str, Any]: ... *# {ts, bid, ask, mid}* **async** **def** place\_order(self, \*, product: str, amount: float, direction: str, timeframe: int, client\_req\_id: str) -> Dict[str, Any]: ... **async** **def** confirm\_order(self, order\_id: str) -> Dict[str, Any]: ... *# {status, pnl, ts\_close\_ms}* **async** **def** cancel\_order(self, order\_id: str) -> Dict[str, Any]: ... *# if platform supports early close* **async** **def** heartbeat(self) -> None: ... **async** **def** close(self) -> None: ...

6.3 Veri Sözleşmeleri (pydantic)

*# core/connector/models.py***from** pydantic **import** BaseModel, Field**from** typing **import** Optional**class** Candle(BaseModel): ts\_ms: int open: float high: float low: float close: float volume: float = 0.0**class** Quote(BaseModel): ts\_ms: int bid: float ask: float @property **def** mid(self): **return** (self.bid + self.ask) / 2**class** OrderAck(BaseModel): order\_id: str client\_req\_id: str ts\_open\_ms: int**class** OrderResult(BaseModel): order\_id: str status: str *# win | lose | abort | canceled* pnl: float ts\_close\_ms: int latency\_ms: int

6.4 Olymp Connector İskeleti (gerçek uçsuz)

*# core/connector/olymp.py***import** asyncio, time, aiohttp**from** typing **import** Any, Dict, List, Optional**from** .interface **import** Connector**from** .models **import** Candle, Quote, OrderAck, OrderResult**class** OlympConnector(Connector): **def** \_\_init\_\_(self, account\_id: str, base\_url: str, ws\_url: Optional[str] = None, timeout\_s: int = 10): self.account\_id = account\_id self.base\_url = base\_url.rstrip('/') self.ws\_url = ws\_url self.\_session: Optional[aiohttp.ClientSession] = None self.\_access\_token: Optional[str] = None self.\_refresh\_token: Optional[str] = None self.\_rate\_sem = asyncio.Semaphore(8) *# basit throttling* self.\_lock = asyncio.Lock() *# kritik bölgeler için* self.\_ws = None self.\_timeout = aiohttp.ClientTimeout(total=timeout\_s) **async** **def** \_http(self, method: str, path: str, \*\*kwargs) -> Dict[str, Any]: **async** **with** self.\_rate\_sem: **if** **not** self.\_session: self.\_session = aiohttp.ClientSession(timeout=self.\_timeout) headers = kwargs.pop('headers', {}) **if** self.\_access\_token: headers['Authorization'] = f'Bearer {self.\_access\_token}' url = f"{self.base\_url}{path}" **async** **with** self.\_session.request(method, url, headers=headers, \*\*kwargs) **as** r: **if** r.status **in** (401, 403): *# token yenileme dene (tek sefer)* **await** self.refresh\_token() headers['Authorization'] = f'Bearer {self.\_access\_token}' **if** self.\_access\_token **else** '' **async** **with** self.\_session.request(method, url, headers=headers, \*\*kwargs) **as** r2: r2.raise\_for\_status() **return** **await** r2.json() r.raise\_for\_status() **return** **await** r.json() **async** **def** login(self, username: str, password: str, otp: Optional[str] = None) -> None: *# PLACEHOLDER: Gerçek uçlar eklenecek (resmî/izinli ise)* payload = {"username": username, "password": password} **if** otp: payload["otp"] = otp *# resp = await self.\_http('POST', '/auth/login', json=payload)* *# self.\_access\_token = resp['access\_token']; self.\_refresh\_token = resp.get('refresh\_token')* self.\_access\_token = "DUMMY" **async** **def** refresh\_token(self) -> None: **if** **not** self.\_refresh\_token: *# fallback: yeniden login gerekebilir (uygun değilse abort)* **return** *# resp = await self.\_http('POST', '/auth/refresh', json={"refresh\_token": self.\_refresh\_token})* *# self.\_access\_token = resp['access\_token']* **async** **def** get\_candles(self, product: str, timeframe: int, n: int = 200, until\_ms: Optional[int] = None) -> List[Dict[str, Any]]: *# resp = await self.\_http('GET', f'/market/candles?symbol={product}&tf={timeframe}&n={n}&until={until\_ms or 0}')* *# return resp['candles']* **return** [] **async** **def** get\_current\_win\_rate(self, product: str) -> float: *# resp = await self.\_http('GET', f'/market/payout?symbol={product}')* *# return float(resp['payout'])* **return** 90.0 **async** **def** get\_quote(self, product: str) -> Dict[str, Any]: *# resp = await self.\_http('GET', f'/market/quote?symbol={product}')* *# return resp* **return** {"ts": int(time.time()\*1000), "bid": 1.0, "ask": 1.0} **async** **def** place\_order(self, \*, product: str, amount: float, direction: str, timeframe: int, client\_req\_id: str) -> Dict[str, Any]: *# resp = await self.\_http('POST', '/orders', json={"symbol": product, "amount": amount, "side": direction, "tf": timeframe, "client\_req\_id": client\_req\_id})* *# return resp* **return** {"order\_id": f"SIM-{int(time.time()\*1000)}", "client\_req\_id": client\_req\_id, "ts\_open\_ms": int(time.time()\*1000)} **async** **def** confirm\_order(self, order\_id: str) -> Dict[str, Any]: *# resp = await self.\_http('GET', f'/orders/{order\_id}')* *# return resp* **return** {"order\_id": order\_id, "status": "win", "pnl": 0.9, "ts\_close\_ms": int(time.time()\*1000)+60000, "latency\_ms": 150} **async** **def** cancel\_order(self, order\_id: str) -> Dict[str, Any]: *# resp = await self.\_http('POST', f'/orders/{order\_id}/cancel')* *# return resp* **return** {"order\_id": order\_id, "status": "canceled"} **async** **def** heartbeat(self) -> None: *# await self.\_http('GET', '/ping')* **return** **async** **def** close(self) -> None: **if** self.\_session: **await** self.\_session.close() self.\_session = None

6.5 Çoklu Hesap Bağlantı Yöneticisi

*# core/connector/manager.py***from** typing **import** Dict**from** .olymp **import** OlympConnector**class** ConnectorManager: **def** \_\_init\_\_(self): self.\_by\_account: Dict[str, OlympConnector] = {} **async** **def** ensure(self, account\_id: str, base\_url: str, \*\*kw) -> OlympConnector: c = self.\_by\_account.get(account\_id) **if** **not** c: c = OlympConnector(account\_id, base\_url, \*\*kw) self.\_by\_account[account\_id] = c **return** c **async** **def** login\_all(self, creds: Dict[str, Dict[str, str]]): *# creds: {acc\_id: {username, password, otp?}}* **for** acc, c **in** self.\_by\_account.items(): u = creds[acc]['username']; p = creds[acc]['password']; otp = creds[acc].get('otp') **await** c.login(u, p, otp) **async** **def** close\_all(self): **for** c **in** self.\_by\_account.values(): **await** c.close()

6.6 Rate‑Limit ve Retry Tasarımı

* **İstek kuyruğu**: Connector içi semaphore (*\_rate\_sem*) + *asyncio.Queue* ile sınırlama.
* **Backoff**: 429/5xx ve *TimeoutError*’da *base \* (2^n) + jitter*;
  + *n\_max = 5*; *base = 0.25s*; *jitter ∈ [0, 0.1s]*.
* **Bakım durumu**: Belirli hata kodlarında **read‑only** moda geç; emir fonksiyonları geçici devre dışı.

6.7 İdempotensi ve Emir Güvenliği

* Her *place\_order* çağrısında **benzersiz** *client\_request\_id*; Storage’da *orders.client\_req\_id UNIQUE*.
* *confirm\_order* sonucu alınmadan tekrar *place\_order* **yapılmaz** (Order FSM).
* Çift yanıt durumlarında *client\_req\_id* üzerinden **eşleşme ve yutma** (duplicate ignore).

6.8 Hata Sınıflandırması & Haritalama

* **AuthError** (401/403), **ThrottleError** (429), **MaintenanceError** (503/temporarily unavailable), **NetworkError** (timeout/connection reset), **FormatError** (parse/json), **BusinessRuleError** (permit dışı, concurrency, kural ihlali).
* Her hata **telemetry**’de sayılır; UI’ya **alert** olarak aktarılır.

6.9 Zamanlama ve Zaman Uyumlaması

* Tüm zamanlar **UTC ms**; UI’da *Europe/Istanbul* gösterimi.
* TF hizalama: *align\_to\_tf(ts, tf\_min)* yardımcı fonksiyonu; mum kapanışında tetikleme.
* Sistem saat kayması için opsiyonel **NTP drift** kontrolü (uyarı üretir).

6.10 Güvenlik Notları

* **TLS zorunlu**, sertifika doğrulama açık; host allow‑list opsiyonel.
* Token/çerezler *profiles/accX/* altında **şifreli**; loglarda **maskelenir**.
* **Scraping/otomasyon** yok; yalnızca izinli/ yayımlanmış uçlar kullanılacak.

6.11 MockConnector (Test & Paper)

*# core/connector/mock.py***import** math, random, time**from** typing **import** Dict, Any, List, Optional**class** MockConnector: **def** \_\_init\_\_(self, account\_id: str, seed: int = 42): self.account\_id = account\_id random.seed(seed) self.\_price = 1.0000 **async** **def** login(self, username: str, password: str, otp: Optional[str] = None): **return** **async** **def** get\_candles(self, product: str, timeframe: int, n: int = 200, until\_ms: Optional[int] = None) -> List[Dict[str, Any]]: *# Basit sinüzoidal + gürültü ile OHLCV üretimi* now = int(time.time()\*1000) out = [] p = self.\_price **for** i **in** range(n): t = now - i\*timeframe\*60\_000 base = math.sin(i/10.0)\*0.001 noise = (random.random()-0.5)\*0.0005 close = max(0.5, p + base + noise) high = max(close, p) + 0.0002 low = min(close, p) - 0.0002 out.append({"ts\_ms": t, "open": p, "high": high, "low": low, "close": close, "volume": random.randint(100,1000)}) p = close self.\_price = p **return** list(reversed(out)) **async** **def** get\_current\_win\_rate(self, product: str) -> float: **return** 90.0 + (random.random()-0.5)\*4 *# %88–%92 çevresi* **async** **def** get\_quote(self, product: str) -> Dict[str, Any]: self.\_price += (random.random()-0.5)\*0.0003 **return** {"ts": int(time.time()\*1000), "bid": self.\_price-0.0001, "ask": self.\_price+0.0001} **async** **def** place\_order(self, \*, product: str, amount: float, direction: str, timeframe: int, client\_req\_id: str) -> Dict[str, Any]: **return** {"order\_id": f"MOCK-{client\_req\_id}", "client\_req\_id": client\_req\_id, "ts\_open\_ms": int(time.time()\*1000)} **async** **def** confirm\_order(self, order\_id: str) -> Dict[str, Any]: *# Basit kural: son hareket yönüne göre win/lose simüle et* win = random.random() > 0.45 pnl = 0.9 **if** win **else** -1.0 **return** {"order\_id": order\_id, "status": "win" **if** win **else** "lose", "pnl": pnl, "ts\_close\_ms": int(time.time()\*1000)+60\_000, "latency\_ms": 120} **async** **def** cancel\_order(self, order\_id: str) -> Dict[str, Any]: **return** {"order\_id": order\_id, "status": "canceled"} **async** **def** heartbeat(self) -> None: **return**

6.12 Test Planı

* **Birim**: Mock ile *login/get\_candles/get\_quote/get\_current\_win\_rate/place/confirm* çağrıları; idempotent *client\_req\_id* testi.
* **Hata Enjeksiyonu**: 401/429/503/timeout simülasyonları; backoff ve mod geçişleri.
* **Çoklu Hesap**: 4 hesap paralel login + data çekme + order akışı; profil izolasyon doğrulaması.
* **Performans**: 100+ eşzamanlı *(acc,prod,tf)* örneğinde dakika başına istek sayısı ve ort. latency.

6.13 Kabul Kriterleri (Parça 6)

* Connector **sözleşmeleri** ve **iskele kod** hazır.
* **Çoklu hesap** yönetimi, rate‑limit, backoff, idempotensi ve hata sınıflandırması tanımlı.
* MockConnector ile **paper/backtest** için çalışır demolar yapılabilir.
* Gerçek uç noktalar yalnız **resmî/izinli** olduğunda eklenecek; scraping/atlatma **yok**.

Parça 7 — İndikatör Kütüphanesi (Temel Set)

Amaç: Stratejilerin çoğunlukla ihtiyaç duyduğu **temel teknik göstergeleri** tek yerde, hızlı, tutarlı ve yeniden kullanılabilir biçimde sunmak. İleri set (ADX, Supertrend, Ichimoku, Keltner, Donchian, RVOL, vb.) **Parça 8**’dedir.

7.1 Veri Sözleşmesi ve Giriş/Çıkış

* **Girdi DataFrame şeması**: sütunlar *ts\_ms, open, high, low, close, volume*. Zaman **artan** sıralı ve tekil.
* **Uyum**: UTC ms; UI’da Europe/Istanbul gösterilir.
* **Çıkış**: *pd.Series* (veya tuple of Series) — **girdiyle aynı uzunluk** ve endeks; **ısınma (warm‑up)** döneminde *NaN*.
* **Performans**: **pandas/numpy vektörizasyonu**, *rolling().mean()/std()*; gereksiz *apply* yok.
* **Hafıza**: Kısa pencere hesaplarında kopyasız işlemler; uzun seriler için *float32* opsiyonu (parametre ile).

7.2 Kapsam (Temel Göstergeler)

* **Hareketli Ortalamalar**: *sma*, *ema*, *wma*, *hma*
* **Momentum/Osilatör**: *rsi*, *stochastic* (%K/%D)
* **Trend/Momentum**: *macd* (12/26/9) + histogram
* **Volatilite/Bantlar**: *bollinger\_bands* (20, 2σ) + *bollinger\_width*
* **Volatilite**: *true\_range*, *atr* (Wilder, 14)
* **Hacim Tabanlı**: *obv*, *mfi* (14)
* **VWAP (basit)**: *vwap* (gün/oturum reset destekli temel sürüm)

7.3 Modül ve API (Python)

core/indicators/ basic.py # bu parça \_\_init\_\_.py

*# core/indicators/basic.py***from** \_\_future\_\_ **import** annotations**import** pandas **as** pd**import** numpy **as** np**from** typing **import** Tuple, Optional *# === Hareketli Ortalamalar ===***def** sma(s: pd.Series, length: int = 20) -> pd.Series: **return** s.rolling(length, min\_periods=length).mean()**def** ema(s: pd.Series, length: int = 20) -> pd.Series: **return** s.ewm(span=length, adjust=False, min\_periods=length).mean()**def** wma(s: pd.Series, length: int = 20) -> pd.Series: w = np.arange(1, length + 1) **return** s.rolling(length).apply(**lambda** x: np.dot(x, w) / w.sum(), raw=True)**def** hma(s: pd.Series, length: int = 20) -> pd.Series: *# Hull MA: WMA( 2\*WMA(n/2) − WMA(n), sqrt(n) )* n1 = int(length) n2 = max(1, int(length/2)) n3 = max(1, int(np.sqrt(length))) wma\_n = wma(s, n1) wma\_n2 = wma(s, n2) hull = 2\*wma\_n2 - wma\_n **return** wma(hull, n3) *# === Momentum / Osilatör ===***def** rsi(s: pd.Series, length: int = 14) -> pd.Series: delta = s.diff() gain = delta.clip(lower=0) loss = -delta.clip(upper=0) avg\_gain = gain.ewm(alpha=1/length, adjust=False, min\_periods=length).mean() avg\_loss = loss.ewm(alpha=1/length, adjust=False, min\_periods=length).mean() rs = avg\_gain / avg\_loss.replace(0, np.nan) out = 100 - (100 / (1 + rs)) **return** out**def** stochastic(high: pd.Series, low: pd.Series, close: pd.Series, k: int = 14, k\_smooth: int = 3, d: int = 3) -> Tuple[pd.Series, pd.Series]: ll = low.rolling(k, min\_periods=k).min() hh = high.rolling(k, min\_periods=k).max() k\_raw = 100 \* (close - ll) / (hh - ll).replace(0, np.nan) k\_line = k\_raw.rolling(k\_smooth, min\_periods=k\_smooth).mean() d\_line = k\_line.rolling(d, min\_periods=d).mean() **return** k\_line, d\_line *# === MACD ===***def** macd(s: pd.Series, fast: int = 12, slow: int = 26, signal: int = 9) -> Tuple[pd.Series, pd.Series, pd.Series]: ema\_fast = ema(s, fast) ema\_slow = ema(s, slow) macd\_line = ema\_fast - ema\_slow signal\_line = macd\_line.ewm(span=signal, adjust=False, min\_periods=signal).mean() hist = macd\_line - signal\_line **return** macd\_line, signal\_line, hist *# === Bollinger ===***def** bollinger\_bands(s: pd.Series, length: int = 20, mult: float = 2.0) -> Tuple[pd.Series, pd.Series, pd.Series]: mid = sma(s, length) std = s.rolling(length, min\_periods=length).std(ddof=0) upper = mid + mult \* std lower = mid - mult \* std **return** upper, mid, lower**def** bollinger\_width(s: pd.Series, length: int = 20, mult: float = 2.0) -> pd.Series: u, m, l = bollinger\_bands(s, length, mult) **return** (u - l) / m *# === ATR / True Range ===***def** true\_range(high: pd.Series, low: pd.Series, close: pd.Series) -> pd.Series: prev\_close = close.shift(1) tr1 = high - low tr2 = (high - prev\_close).abs() tr3 = (low - prev\_close).abs() **return** pd.concat([tr1, tr2, tr3], axis=1).max(axis=1)**def** atr(high: pd.Series, low: pd.Series, close: pd.Series, length: int = 14) -> pd.Series: tr = true\_range(high, low, close) **return** tr.ewm(alpha=1/length, adjust=False, min\_periods=length).mean() *# === Hacim Tabanlı ===***def** obv(close: pd.Series, volume: pd.Series) -> pd.Series: direction = np.sign(close.diff()).fillna(0) **return** (direction \* volume).cumsum()**def** mfi(high: pd.Series, low: pd.Series, close: pd.Series, volume: pd.Series, length: int = 14) -> pd.Series: tp = (high + low + close) / 3.0 raw = tp \* volume pos = np.where(tp.diff() > 0, raw, 0.0) neg = np.where(tp.diff() < 0, raw, 0.0) pos\_mf = pd.Series(pos, index=tp.index).rolling(length, min\_periods=length).sum() neg\_mf = pd.Series(neg, index=tp.index).rolling(length, min\_periods=length).sum() mr = pos\_mf / neg\_mf.replace(0, np.nan) **return** 100 - (100 / (1 + mr)) *# === VWAP (basit) ===***def** vwap(high: pd.Series, low: pd.Series, close: pd.Series, volume: pd.Series, session: Optional[pd.Series] = None) -> pd.Series: tp = (high + low + close) / 3.0 **if** session **is** None: cum\_pv = (tp \* volume).cumsum() cum\_v = volume.cumsum().replace(0, np.nan) **return** cum\_pv / cum\_v *# session değişiminde sıfırlama* pv = tp \* volume groups = session.ne(session.shift()).cumsum() vwap\_series = pv.groupby(groups).cumsum() / volume.groupby(groups).cumsum().replace(0, np.nan) vwap\_series.index = tp.index **return** vwap\_series

7.4 Kenar Durumlar ve Uyum Testleri

* **Sıralama**: *ts\_ms* artan değilse **sıralayıp** uyarı ver.
* **Eksik veri**: *volume* boş ise *obv/mfi/vwap* hesapları *NaN*; UI’da veri eksik uyarısı.
* **Bölünme/hatalar**: 0’a bölmeyi *replace(0, np.nan)* ile koru; *inf* değerleri *NaN*’e çevir.
* **Warm‑up**: *min\_periods=length* → başlangıçta *NaN*; strategy bu barlarda **işlem açmaz**.
* **dtype**: Uzun serilerde *float32* seçeneği sun (hafıza için), fakat son hesap *float64* kalabilir.

7.5 Feature Adlandırma Standardı

* MA: *ema{L}*, *sma{L}*, *hma{L}*;
* MACD: *macd\_{f}\_{s}\_{sig}*, *macd\_hist\_{f}\_{s}\_{sig}*;
* RSI: *rsi{L}*;
* Stoch: *stoch\_k\_{L}*, *stoch\_d\_{L}*;
* BB: *bb\_upper\_{L}\_{mult}*, *bb\_mid\_{L}*, *bb\_lower\_{L}\_{mult}*, *bb\_width\_{L}\_{mult}*;
* ATR: *atr{L}*, TR: *tr*;
* Hacim: *obv*, *mfi{L}*;
* VWAP: *vwap* veya *vwap\_session*.

7.6 Performans İpuçları

* Tekrarlanan hesaplar için **önbellekleme** (aynı *Series.\_values* kimliği ve aynı parametre) — opsiyonel zayıf referanslı cache.
* Çoklu TF hesaplamalarında **tek pass**: Örn. 1 dakikalık seriden 5/15 dakikayı **resample** edip hesapla.
* **Numba** gerekmez; profil sonucu gerekiyorsa yalnız kritik fonksiyonlara eklenir (opsiyonel).

7.7 Test Planı

* **Doğruluk**:
  + RSI, MACD, BB, ATR için bilinen referans serilerle **tolerans** (*rtol=1e-6*).
  + Stoch %K/%D sınırları [0,100].
* **Performans**: 1M bar üzerinde toplam çalışma süresi ölçümü; hedef donanıma bağlı olarak **saniye‑altı/çok saniye** ölçeğinde.
* **Uyumluluk**: *NaN* yayılımı ve warm‑up; bölünme‑hatası testleri.

7.8 UI / Strateji Entegrasyonu

* Stratejiler basic set’i **doğrudan** kullanır; parametreleri **UI’dan override** edilebilir.
* Grafikte overlay: BB bantları, MA’lar; alt pencerelerde RSI, MACD, Stoch.
* Feature çıkarımı: Storage’a yazılan kolon adları **7.5 standardı** ile eşleşir.

7.9 Kabul Kriterleri (Parça 7)

* Temel göstergeler (MA’lar, RSI, Stoch, MACD, Bollinger, ATR/TR, OBV, MFI, VWAP) **uygulandı** (API imzaları ve iskelet kod hazır).
* Warm‑up/NaN, sıralama, bölünme hatası ve dtype politikaları **tanımlandı**.
* Feature adlandırma standardı & UI/strateji entegrasyonu **net**.

Parça 8 — İleri İndikatörler & Durum Çıkarımı (ADX, Supertrend, Ichimoku, Keltner, Donchian, RVOL, CMF, PPO, StochRSI, Pivot)

Amaç: Strateji ailelerinin (Supertrend+ADX, VWAP Reclaim+RVOL, Keltner Break, Bollinger Walk vb.) gerektirdiği **ileri göstergeleri** ve **durum/olay çıkarımı** yardımcılarını sağlamak. Temel set (Parça 7) üzerine kurulur.

8.1 Kapsam

* **Trend/Güç**: DMI/ADX (DI+, DI−, ADX), GMMA kısa/uzun demet (özet mesafe)
* **Momentum**: PPO (12/26/9), Stochastic RSI, CCI, Fisher Transform
* **Volatilite/Kanallar**: Keltner Channel (EMA20, ATR10, ×1.5..×1.8), Donchian (20)
* **Hacim/Akış**: CMF(20), RVOL(20), VWAP varyantları (günlük/seans/anchor)
* **Bulut**: Ichimoku (tenkan/kijun/senkouA/B/chikou) + **ichimoku\_state**
* **Trend Tersleme**: Supertrend(ATR10,×3.0) + **supertrend\_state**
* **Pivotlar**: Classical PP, R1/R2/R3, S1/S2/S3

8.2 Modüller & API

core/indicators/ advanced.py # ileri göstergeler states.py # durum/olay çıkarımı

8.2.1 *advanced.py* (imzalar)

**import** pandas **as** pd**from** typing **import** Tuple *# === Trend Gücü ===***def** dmi(high: pd.Series, low: pd.Series, close: pd.Series, length: int = 14) -> Tuple[pd.Series, pd.Series, pd.Series]: *"""return (di\_plus, di\_minus, adx)"""  
  
# === Momentum ===***def** ppo(close: pd.Series, fast: int = 12, slow: int = 26, signal: int = 9) -> Tuple[pd.Series, pd.Series, pd.Series]: *"""return (ppo\_line, signal, hist)"""***def** stoch\_rsi(close: pd.Series, rsi\_len: int = 14, stoch\_len: int = 14, k: int = 3, d: int = 3) -> Tuple[pd.Series, pd.Series]: *"""return (%K, %D) computed on RSI"""***def** cci(high: pd.Series, low: pd.Series, close: pd.Series, length: int = 20, c: float = 0.015) -> pd.Series: ...**def** fisher(close: pd.Series, length: int = 9) -> pd.Series: ... *# === Volatilite / Kanallar ===***def** keltner\_channel(close: pd.Series, high: pd.Series, low: pd.Series, ema\_len: int = 20, atr\_len: int = 10, mult: float = 1.5) -> Tuple[pd.Series, pd.Series, pd.Series]: *"""return (upper, mid, lower)"""***def** donchian(high: pd.Series, low: pd.Series, length: int = 20) -> Tuple[pd.Series, pd.Series, pd.Series]: *"""return (upper, mid, lower)"""  
  
# === Hacim/Akış ===***def** cmf(high: pd.Series, low: pd.Series, close: pd.Series, volume: pd.Series, length: int = 20) -> pd.Series: ...**def** rvol(volume: pd.Series, lookback: int = 20) -> pd.Series: ... *# === VWAP varyantları ===***from** .basic **import** vwap **as** vwap\_simple**def** session\_vwap(high: pd.Series, low: pd.Series, close: pd.Series, volume: pd.Series, session\_id: pd.Series) -> pd.Series: ... *# === Ichimoku ===***def** ichimoku(high: pd.Series, low: pd.Series, close: pd.Series, conv: int = 9, base: int = 26, span\_b: int = 52): *"""return (tenkan, kijun, senkou\_a, senkou\_b, chikou)"""  
  
# === Supertrend ===***def** supertrend(high: pd.Series, low: pd.Series, close: pd.Series, atr\_len: int = 10, mult: float = 3.0) -> Tuple[pd.Series, pd.Series]: *"""return (st\_line, direction) with direction ∈ {1,-1}"""  
  
# === Pivotlar ===***def** pivots\_classic(high: pd.Series, low: pd.Series, close: pd.Series): *"""return (pp, r1, r2, r3, s1, s2, s3)"""*

8.2.2 *states.py* (durum çıkarımı)

**import** pandas **as** pd**from** typing **import** Dict**def** ichimoku\_state(tenkan: pd.Series, kijun: pd.Series, span\_a: pd.Series, span\_b: pd.Series, close: pd.Series) -> pd.Series: *# basit sınıflama: bulut üstü/altı, tenkan>kijun* above\_cloud = (close > span\_a) & (close > span\_b) below\_cloud = (close < span\_a) & (close < span\_b) tk\_cross\_up = tenkan > kijun tk\_cross\_dn = tenkan < kijun state = pd.Series('neutral', index=close.index) state = state.mask(above\_cloud & tk\_cross\_up, 'bullish') state = state.mask(below\_cloud & tk\_cross\_dn, 'bearish') **return** state**def** supertrend\_state(direction: pd.Series) -> pd.Series: **return** direction.map(**lambda** d: 'bullish' **if** d == 1 **else** 'bearish')

8.3 Formüller (özet)

* **DMI/ADX**: +DM = max(H−H₋₁, 0); −DM = max(L₋₁−L, 0); TR = max(H−L, |H−C₋₁|, |L−C₋₁|). DI± = 100·RMA(DM±/TR, L); ADX = RMA(|DI+−DI−|/(DI++DI−), L)
* **PPO**: 100·(EMA\_f−EMA\_s)/EMA\_s; Signal = EMA(PPO, s); Hist = PPO−Signal
* **StochRSI**: RSI serisine stokastik uygulanır; %K/%D (3-SMA)
* **CCI**: (TP−SMA(TP,L)) / (c·MeanDev), TP=(H+L+C)/3
* **Fisher**: x∈[−1,1] ölçeğine normalize → F = 0.5·ln((1+x)/(1−x))
* **Keltner**: Mid=EMA(C,L); Upper=Mid+mult·ATR; Lower=Mid−mult·ATR
* **Donchian**: Upper=HHV(L); Lower=LLV(L); Mid=(U+L)/2
* **CMF**: Σ(((C−L)−(H−C))/(H−L)·V, L) / Σ(V, L)
* **RVOL**: V / mean(V, lookback)
* **Ichimoku**: Tenkan=(HH₉+LL₉)/2; Kijun=(HH₂₆+LL₂₆)/2; SpanA=(Tenkan+Kijun)/2 (26 ileri); SpanB=(HH₅₂+LL₅₂)/2 (26 ileri); Chikou=C (26 geri)
* **Supertrend**: ATR tabanlı bantlar ve yön flip mantığı
* **Pivot (Classic)**: PP=(H+L+C)/3; R1=2·PP−L; S1=2·PP−H; R2=PP+(H−L); S2=PP−(H−L); R3=H+2·(PP−L); S3=L−2·(H−PP)

8.4 Feature Adlandırma (ileri)

* *adx14*, *di\_plus14*, *di\_minus14*, *ppo\_hist\_12\_26\_9*, *stochrsi\_k\_14\_14\_3*, *cci20*, *fisher9*,
* *keltner\_upper\_20\_10\_1.5*, *keltner\_width\_20\_10\_1.5*, *donch\_upper\_20*, *donch\_mid\_20*,
* *cmf20*, *rvol20*, *ich\_state*, *st\_state*, *pp*, *r1*, *s1* …

8.5 Strateji Aileleriyle Kullanım

* **Supertrend + ADX**: *st\_state=='bullish'* **ve** *adx14>20* → long tercih; eşik yükseldikçe (22/24/26) filtre sıkılaşır.
* **VWAP Reclaim + RVOL**: Kapanış **VWAP üstüne reclaim** **ve** *rvol20>1.3* → onay; 1.5 güçlü onay.
* **Keltner Break**: Mum kapanışı *keltner\_upper* **üstünde**; *adx14* destekliyorsa giriş, SL *mid* veya *lower*.
* **Bollinger Walk**: (Parça 7) BB üst bantta ardışık kapanış + (burada) Donchian üst trend teyidi.
* **EMA Trend + RSI**: ADX ile trend gücü doğrulaması; StochRSI ile aşırı hız uyarısı.

8.6 Performans & Doğruluk

* **RMA/Wilder** kullanımı (ADX) TV sonuçlarına yakınlaştırır.
* Ichimoku’da ileri/geri kaydırmalar **indeks** ile açık biçimde uygulanır; durum çıkarımı bu kaydırmayı dikkate alır.
* RVOL saat etkisine duyarlı olabilir; opsiyonel **saat‑bazlı normalize** (ileri faz) eklenebilir.

8.7 Test Planı

* **Birim**: Sentetik seri karşılaştırmaları; NaN yayılımı, warm‑up; sınır değerler.
* **Regresyon**: Formül/parametre değişimi yok (semver ile korunur).
* **Performans**: 100k–1M bar üstünde toplu hesap süreleri profillenmiş.

8.8 UI Entegrasyonu

* Gelişmiş ayarlarda indikatör parametreleri override (örn. *adx\_len*, *keltner\_mult*).
* Grafik overlay: Supertrend çizgisi, Ichimoku bulutu, Keltner/Donchian bantları; alt pencerede PPO/StochRSI/CCI/Fisher.

8.9 Kabul Kriterleri (Parça 8)

* İleri indikatörler ve durum çıkarımı imzaları **hazır**.
* Strateji aileleriyle eşleşen kurallar **belirlendi**.
* Doğruluk, performans ve regresyon test politikaları **tanımlandı**.
* UI/feature adlandırma ve parametre override yönergeleri **net**.

Parça 9 — Ensemble & Confidence Katmanı

Amaç: Birden çok strateji/indikatör sağlayıcısından gelen sinyalleri birleştirip tek bir güven (confidence) değeri üretmek; bu değeri olasılık (p-hat) olarak kalibre etmek ve giriş eşiği / permit penceresi / risk kararlarına beslemek. Ayrıca strateji ağırlıklarını performansa göre uyarlamak ve rejim değişimlerinde sağlam kalmak.

9.1 Kavramlar

* Signal Provider: (context) → {vote∈{-1,0,1}, score∈R, meta}. Vote yön, score büyüklük/güven.
* Ensemble: Sağlayıcı çıktılarından tek skor S∈[-1,1] ve confidence c∈[0,1] üretir.
* p-hat (tahmini kazanma olasılığı): Kalibre edilmiş confidence; risk ve expectancy hesaplarına girer.
* Meta-öğrenme: Ağırlıklar ve/veya üst model, sonuçlardan (win/lose) online güncellenir.

9.2 Ensemble Stratejileri

1. Ağırlıklı Toplam (varsayılan)  
   S = tanh( sum\_i ( w\_i \* v\_i \* s\_i ) ) (v\_i ∈ {-1,0,1}, s\_i normalize skor).  
   Confidence: c = |S|.  
   Yön: sign(S). 0 eşiği yakınında işlem açmamak için ek eşik c\_min kullanılır.
2. Majority Vote + Güç  
   Basit oy çoğunluğu; eşitlikte sum |s\_i| ile kır. Trend-rejim filtreli varyant (ör. ADX/EMA rejimi aktifken).
3. Stacking (L1→L2)  
   L1: sağlayıcı skorları/oyları, rejim sinyalleri, payout vb.  
   L2: lojistik regresyon (veya hafif GBM) ile p-hat üretir. Eğitim offline (günlük) + online küçük adım (ops.).
4. Gating / Mixture-of-Experts  
   Rejim özelliklerine (volatilite, trend gücü, RVOL) göre uzman seçici; her kümeye farklı ağırlık vektörü w^(k).
5. Bandit (UCB/Thompson, ops.)  
   Kısa vadede hangi sağlayıcı/alt-strateji daha iyi? Çok kollu bandit ile keşif-kullanım dengesi.

Başlangıç sürümünde Ağırlıklı Toplam + Lojistik Kalibrasyon uygulanır. Stacking/gating ileriki fazda açılabilir.

9.3 Skor Normalizasyonu

Her sağlayıcı için skor s\_i standartlaştırılır:  
s’\_i = clip( (s\_i - mu\_i) / sigma\_i , -3 , 3 ) (kayan pencere N=500).  
UI’dan katkı sınırı: |w\_i \* s’\_i| ≤ s\_cap.

9.4 Kalibrasyon (confidence → p-hat)

* Platt Scaling (lojistik): p-hat = sigmoid( a\*S + b ).  
  Parametreler (a,b) günlük olarak geçmiş (S, outcome) çiftlerinden fit edilir (L2 reg.).
* Alternatif: İzotonik Regresyon (daha esnek, az veri ile aşırı uyum riski).
* Değerlendirme: Brier Loss, LogLoss, ECE (Expected Calibration Error).
* Politika: ECE kötüleşirse yeniden kalibrasyon tetikle.

9.5 Ağırlık Güncelleme (Online)

* Kayan pencere performansı: Her sağlayıcı için son N=200 işlemde win\_rate\_i, Sharpe\_i, latency\_penalty\_i ölç.
* Ağırlık kuralı (örnek): w\_i ← (1-α)*w\_i + α*softmax( k \* score\_i ),  
  burada score\_i = win\_rate\_i − lambda \* latency\_penalty\_i.
* Sınırlar: w\_i ≥ 0, toplam w\_i = 1, w\_i ≤ w\_max.
* Sıfırlama: Rejim değişimi tespitinde (bkz. 9.6) w ← uniform.

9.6 Rejim Tespiti & Drift

* Trend gücü (ADX, 20 üstü/altı), volatilite (ATR z-skoru), hacim (RVOL), payout rejimi (permit orta noktası) → rejim vektörü.
* Drift testleri: Page-Hinkley, ADWIN (opsiyonel, veri akışı).
* Tetikte: Ağırlıkları resetle, kalibrasyonu yenile, gereksiz sağlayıcıları devre dışı bırak (UI uyarısı).

9.7 Çoklu-TF / Çoklu-Ürün Birleşimi

* Anahtar düzeyi: Ensemble her (acc, product, timeframe) için ayrı çalışır.
* Çapraz teyit (ops.): TF=1 sinyali TF=5 ile uyumluysa bonus (gamma katsayısı) → S = S + gamma \* sign(S5) \* min(|S|, |S5|).

9.8 API (Python İskelet)

*# core/ensemble.py***from** dataclasses **import** dataclass, field**from** typing **import** Dict, List**import** math@dataclass**class** ProviderVote: pid: int vote: int *# -1,0,1* score: float *# ham skor*@dataclass**class** EnsembleState: mu: Dict[int, float] = field(default\_factory=dict) sigma: Dict[int, float] = field(default\_factory=dict) w: Dict[int, float] = field(default\_factory=dict) a: float = 1.0 *# Platt a* b: float = 0.0 *# Platt b***class** Ensemble: **def** \_\_init\_\_(self, state: EnsembleState, s\_cap: float = 2.0): self.st = state self.s\_cap = s\_cap **def** \_norm(self, pid: int, x: float) -> float: m = self.st.mu.get(pid, 0.0); s = max(1e-6, self.st.sigma.get(pid, 1.0)) z = max(-3.0, min(3.0, (x - m)/s)) **return** z **def** combine(self, votes: List[ProviderVote]) -> Dict: *# ağırlıklı toplam* acc = 0.0 **for** v **in** votes: wi = self.st.w.get(v.pid, 1.0/len(votes)) zi = self.\_norm(v.pid, v.score) contrib = wi \* v.vote \* zi *# katkı tavanı* **if** contrib > self.s\_cap: contrib = self.s\_cap **if** contrib < -self.s\_cap: contrib = -self.s\_cap acc += contrib S = math.tanh(acc) c = abs(S) *# Platt → p-hat* p\_hat = 1.0/(1.0 + math.exp(-(self.st.a\*S + self.st.b))) **return** {"S": S, "confidence": c, "p\_hat": p\_hat, "dir": (1 **if** S>0 **else** (-1 **if** S<0 **else** 0))} **def** update\_calibration(self, S\_list: List[float], y\_list: List[int]): *# küçük bir lojistik fit (basit SGD/türevli), burada iskelet* **pass** **def** update\_weights(self, provider\_scores: Dict[int, float], alpha: float = 0.1, w\_max: float = 0.4): *# softmax temelli güncelleme* **import** math keys = list(provider\_scores.keys()) vals = [provider\_scores[k] **for** k **in** keys] mx = max(vals); exps = [math.exp(v - mx) **for** v **in** vals] s = sum(exps) new = {k: min(w\_max, exps[i]/s) **for** i,k **in** enumerate(keys)} *# karışım güncellemesi* **for** k **in** keys: self.st.w[k] = (1-alpha)\*self.st.w.get(k, 1.0/len(keys)) + alpha\*new[k] *# normalize* tot = sum(self.st.w.values()) **for** k **in** self.st.w: self.st.w[k] /= max(tot, 1e-9)

9.9 Telemetry & Depolama

* metrics tablosuna: brier, logloss, ece, auc (ops.), calib\_a/b, w\_entropy, drift\_score yazılır.
* strategy\_perf tablosu: sağlayıcı bazlı kazanım/kayıp sayıları, son güncelleme.

9.10 UI Eşlemesi

* Ensemble Ayarları: ensemble\_mode (weighted/stacking), c\_min, s\_cap, alpha (weight\_lr), w\_max.
* Kalibrasyon: Durum (iyi/orta/kötü), son ECE, yeniden kalibre et butonu.
* Sağlayıcı Tablosu: w\_i, win%, latency, kapalı/açık anahtarı.
* Rejim Paneli: ADX, ATR z-skoru, RVOL, payout rejimi; drift uyarıları.

9.11 Test Planı

* Birim: combine() normalizasyon, sınır durumları (w=0, sigma→0), yön/|S| doğrulaması.
* Kalibrasyon: Sentetik veriyle Platt fit → Brier/ECE iyileşmesi.
* Ağırlık Uyarlama: Yapay performans profilleri ile w’lerin beklenen yöne kayması.
* Rejim/Drift: ADX/ATR/RVOL değişim senaryolarında reset ve uyarı.

9.12 Kabul Kriterleri (Parça 9)

* Weighted ensemble + Platt kalibrasyon uygulandı (tasarım + iskelet).
* Ağırlık güncelleme, rejim/drift tepkisi ve çoklu-TF politikaları tanımlandı.
* Telemetry alanları ve UI ayarları belirlendi.
* Birim ve kalibrasyon testleri için net ölçütler kondu.

Parça 10 — Risk Yönetimi & Lotlama

Amaç: Fixed Time (FTT) işlemlerinde tek gerçek risk kaldıraç unsurunun **tutar (stake)** olduğu gerçeğiyle; giriş/kaçınma kriterleri, günlük/oturum bazlı korumalar, lot boyutu ve soğuma (cool‑down) politikalarını tanımlamak. FTT doğası gereği işlem içi SL/TP yoktur; **risk, girişten önce** belirlenir.

10.1 Terimler ve Beklenti

* **Payout (R\_win)**: Kazanç oranı (ör. %90 → 0.90).
* **p̂ (tahmini başarı olasılığı)**: Ensemble/kalibrasyon çıktısı.
* **Beklenen değer** (tutar (a) için): ( E[a] = a,(p̂,R\_{win} − (1−p̂)) ).
* **Kâr eşiği**: ( p̂ > 1/(1+R\_{win}) ). Örn. R=0.90 → eşik ≈ **%52.63**.
* **UI politikası**: Kullanıcı **win\_threshold**’u bu teorik eşiğin **üstünde** (örn. %70+) ayarlar.

10.2 Giriş (Enter) Politikaları

1. **Permit Payout**: *permit\_min ≤ payout ≤ permit\_max* değilse **giriş yok**.
2. **Confidence Eşiği**: *confidence ≥ win\_threshold\_tf*.
3. **Concurrency**: *(hesap, ürün, TF)* başına **tek açık işlem**; 1/5/15 birbirinden bağımsız.
4. **Saat Filtresi (ops.)**: Düşük likidite saatleri hariç; UI’dan tanımlanır.
5. **Cool‑down**: Kayıptan sonra *t\_cooldown\_s* bekle; ardışık kayıpta süre **artan** (30s→60s→120s, tavan 5dk).
6. **Vol/Haber Filtresi (ops.)**: *ATR z‑skoru* veya *RVOL* aşımında bekle.
7. **Latency Guard**: Emir turu tahmini gecikme *> abort\_ms* ise giriş **engellenir**.

10.3 Lot (Tutar) Boyutlandırması

Modlar ve sınırlar:  
- **Fixed** *(varsayılan)*: Her işlemde aynı *a0*.  
- **Balance Fraction**: *a = clamp(a\_min, frac·balance, a\_cap)*; *frac*=1–3% tipik.  
- **Kelly‑lite (ops.)**: ( f^\* = (p̂(R+1)−1)/R ); gerçek tutar = *balance·max(0,f\*)·kelly\_scale* (0.1–0.3).  
- **ATR‑Normalize (deneysel)**: Volatiliteye göre göreli ayarlama; FTT’de sınırlı faydalı.  
- **Sert sınırlar**: *a\_min* ve **tavan** *a\_cap* her modda uygulanır.  
- **Martingale/anti‑martingale**: **Varsayılan kapalı**; UI’da ayrı “riskli” bayrak + sıkı tavanlar ile opsiyonel.

10.4 Guardrails (Bariyerler)

* **Günlük Kayıp**: *realized\_pnl\_day ≤ −max\_daily\_loss* → **hemen durdur**.
* **Ardışık Kayıp**: *consec\_losses ≥ Lmax* → durdur + cool‑down artır.
* **Günlük Kâr Hedefi (ops.)**: Hedefe ulaşıldığında **sadece gözlem** moduna geç.
* **Payout/Win‑rate Bozulması**: Permit dışına çıkıldığında **yeni işlem açma**.
* **Circuit‑breaker**: Hata fırtınasında (auth/429/timeout) otomatik pause + uyarı.

10.5 Çıkış (Exit) ve Erken Kapatma

* FTT’de işlem içi SL/TP **yoktur**. Platform **erken kapama** destekliyorsa opsiyonel: *min\_profit\_lock*, *max\_loss\_lock* politikaları.
* Aksi halde TP/SL kavramları **oturum düzeyi** hedeflerine (günlük TP/SL) taşınır.

10.6 Dinamik Eşik & Tutar Uyarlaması

* **Volatiliteye göre eşik**: *win\_threshold\_dyn = base + κ·z\_vol*; *κ* UI’dan 0–0.15.
* **Performansa göre lot**: Son *N* işlem pozitif → *a ← a·(1+δ)*; negatif → *a ← a·(1−δ)*; *δ* küçük (0.05–0.10), *a\_cap* ile sınırlı.
* **Payout’a göre lot**: Düşük payout’ta tutarı **azalt**; yüksek payout’ta **artırma** sınırlı.

10.7 Servis Akışı (psödo)

if enter\_allowed(ctx): amount = compute\_amount(ctx) ack = place\_order(ctx, amount) log\_order(ack) res = await confirm\_order(ack) log\_result(res) risk.on\_result(ctx, pnl=res.pnl, is\_win=(res.status=='win'))

10.8 Python API (İskelet)

*# core/risk.py***from** dataclasses **import** dataclass@dataclass**class** RiskLimits: max\_daily\_loss: float | None = 5.0 max\_consec\_losses: int | None = 5 a\_min: float = 1.0 a\_cap: float = 10.0@dataclass**class** AmountPolicy: mode: str = "fixed" *# fixed | fraction | kelly\_lite | atr\_norm* fixed\_a: float = 1.0 frac: float = 0.02 *# 2% balance* kelly\_scale: float = 0.2 *# 20% of Kelly***class** RiskEngine: **def** \_\_init\_\_(self, limits: RiskLimits, amt: AmountPolicy): self.limits = limits self.amt = amt self.\_loss\_streak = {} self.\_pnl\_day = {} **def** enter\_allowed(self, ctx) -> bool: **if** ctx.payout < ctx.permit\_min **or** ctx.payout > ctx.permit\_max: **return** False **if** ctx.confidence < ctx.win\_threshold: **return** False **if** ctx.concurrency\_blocked: **return** False **if** self.\_loss\_streak.get(ctx.account, 0) >= (self.limits.max\_consec\_losses **or** 9999): **return** False **if** self.\_pnl\_day.get(ctx.account, 0.0) <= -(self.limits.max\_daily\_loss **or** 9e9): **return** False **return** True **def** compute\_amount(self, ctx) -> float: **if** self.amt.mode == "fixed": a = self.amt.fixed\_a **elif** self.amt.mode == "fraction": a = ctx.balance \* self.amt.frac **elif** self.amt.mode == "kelly\_lite": R = ctx.payout; p = max(0.0, min(1.0, ctx.prob\_win)) f\_star = (p\*(R+1)-1)/max(R,1e-6) a = ctx.balance \* max(0.0, f\_star) \* self.amt.kelly\_scale **else**: a = self.amt.fixed\_a a = max(self.limits.a\_min, min(a, self.limits.a\_cap)) **return** float(a) **def** on\_result(self, ctx, pnl: float, is\_win: bool): self.\_pnl\_day[ctx.account] = self.\_pnl\_day.get(ctx.account, 0.0) + pnl self.\_loss\_streak[ctx.account] = 0 **if** is\_win **else** self.\_loss\_streak.get(ctx.account, 0) + 1

10.9 UI Eşlemesi

* **Ayarlar → Risk**: *max\_daily\_loss*, *max\_consec\_losses*, *cooldown (taban/tavan)*, *amount\_mode*, *fixed\_a*, *frac*, *kelly\_scale*, *a\_min*, *a\_cap*.
* **Ayarlar → Permit & Eşik**: Ürün/TF bazında *permit\_min/max*, *win\_threshold*.
* **Dashboard**: Günlük PnL, kayıp serisi, aktif cool‑down, **beklenen değer** (p̂, R\_win, E[a]).

10.10 Test Planı

* **Simülasyon**: MockConnector ile payout/p̂ varyasyonunda **pozitif beklenen değer** koşulunun seçildiği doğrulanır.
* **Guardrails**: Günlük/ardışık kayıp aşımında **durdur**; cool‑down artışı.
* **Lot sınırları**: *a\_min ≤ a ≤ a\_cap*; fraction/Kelly‑lite tavanlara uyar.
* **Concurrency**: *(hesap, ürün, TF)* başına tek açık işlem kuralı ihlalsiz.

10.11 Kabul Kriterleri (Parça 10)

* Permit, confidence ve concurrency tabanlı **giriş politikaları** tanımlandı.
* Lot modları (fixed, fraction, Kelly‑lite) ve sınırları **uygulandı** (varsayılan: fixed).
* Günlük/ardışık kayıp limitleri, cool‑down ve latency guard **dokümante** edildi.
* UI ayarları, test planı ve kabul ölçütleri **tamamlandı**.

Parça 11 — Order Executor & Idempotency (Durum Makinesi, Retry, Latency)

Amaç: Her *(hesap, ürün, TF)* için **tek açık işlem** kuralına uyan, **idempotent** ve **gözlemlenebilir** bir emir yürütme hattı kurmak. Ağ hataları, bakımlar ve belirsiz (unknown) durumlarda **güvenli** davranıp veri bütünlüğünü korumak.

11.1 Tasarım Hedefleri

* **At‑most‑once** finansal yan etki: Aynı işlem için **çift emir yok**.
* **Idempotensi**: *client\_request\_id* ile gateway/connector düzeyinde tekrarların **yutulması**.
* **Belirsizlikte güvenlik**: Timeout/bağlantı kesilmesinde önce **sorgula**, emin değilse **abort + reconcile**.
* **Gözlemlenebilirlik**: Segment bazlı latency ölçümü, durum makinesi olayları, structured log.

11.2 Durum Makinesi

IDLE └─prepare(ctx)──────────────────────────▶ PREPAREPREPARE ├─ preflight\_ok? ──no──▶ ABORT (reason=preflight) └─ yes └─ place() ─ack→ PLACED └─ timeout/5xx → PLACE\_RETRY (bounded)PLACED ├─ confirm() → SETTLED(win/lose/abort) ├─ early\_close() (ops.) → CANCELED └─ timeout → CONFIRM\_RETRY (bounded)SETTLED | CANCELED | ABORT → IDLE

**Preflight**: Son anda *payout* permit penceresi, *confidence ≥ win\_threshold*, concurrency lock, cool‑down, latency guard **yeniden** kontrol edilir.

11.3 Idempotency Tasarımı

* *client\_request\_id = f"{account\_id}:{product}:{tf}:{entry\_slot}:{uuid8}"*
  + *entry\_slot*: TF hizalı zaman damgası (örn. 2025‑10‑06T12:30:00Z@tf=1).
  + Aynı işlemi tekrar denerken **aynı** *client\_request\_id* kullanılır.
* **Storage**: *orders.client\_req\_id UNIQUE* (Parça 5).
* **Retry sonrası eşleşme**: *confirm\_by\_client\_req\_id()* uç noktası varsa kullan; yoksa *reconcile()* periodyk görevi *client\_req\_id* ile tamamlar.

11.4 Retry Matrisi (Özet)

| Aşama | Hata | Politika |
| --- | --- | --- |
| place() | 401/403 | **Abort** (yeniden login + sonraki işlemler için). |
| place() | 429/5xx/timeout | **Retry** *n≤3*, *backoff=0.25·2^n + jitter*; **aynı client\_request\_id**. |
| confirm() | timeout/5xx | **Retry** *n≤6*, aralık 1–5s artan. |
| confirm() | 404 (bulunamadı) | Önce *client\_req\_id* ile sorgula; hala yoksa **Abort+reconcile**. |

Not: **Finansal at‑most‑once** için *place()* **yalnızca** belirsizlikte tekrar eder; zaten “fail” yanıtı geldiyse tekrar **etmez**.

11.5 Latency Ölçümü (Segmentler)

* **t\_prepare**: Preflight süresi.
* **t\_place\_http**: Connector çağrısı round‑trip.
* **t\_confirm\_wait**: Kapanışa kadar geçen süre (poll/WS).
* **latency\_ms (sonuç)**: *t\_place\_http*.
* Metrikler: *metrics(scope='acc:<id>', key='latency\_ms', ...)* ve hatalarda *error\_rate*.

11.6 Concurrency Lock & Cut‑off

* **Lock anahtarı**: *(account\_id, product, timeframe)* → *asyncio.Lock*.
* **Kural**: 1/5/15 kendi içinde **tek açık işlem**; farklı TF’lerde paralel mümkündür.
* **Giriş cut‑off** (opsiyonel): TF=1’de **son X sn** içinde yeni giriş yok (ayar: *entry\_cutoff\_s*, default 5s).

11.7 Preflight Kontrolleri (son saniye)

1. *payout ∈ [permit\_min, permit\_max]*
2. *confidence ≥ win\_threshold*
3. Concurrency lock **boş**
4. Cool‑down **kapalı**
5. *risk.enter\_allowed(ctx) == True*
6. *latency\_estimate ≤ abort\_ms*

11.8 Python İskeleti

*# core/executor.py***import** asyncio, time, uuid**from** dataclasses **import** dataclass**from** typing **import** Optional@dataclass**class** TradeCtx: account: str product: str timeframe: int *# 1,5,15* direction: str *# 'call'|'put'* payout: float *# % (örn. 90.0)* confidence: float prob\_win: float win\_threshold: float permit\_min: float permit\_max: float balance: float**class** OrderExecutor: **def** \_\_init\_\_(self, connector, storage, risk\_engine, locks, confirm\_timeout\_s: int = 120): self.cx = connector self.db = storage self.risk = risk\_engine self.locks = locks self.confirm\_timeout\_s = confirm\_timeout\_s **def** \_entry\_slot(self, timeframe: int) -> int: now = int(time.time()\*1000) tf\_ms = timeframe \* 60\_000 **return** now - (now % tf\_ms) **async** **def** execute(self, ctx: TradeCtx): *# Preflight* **if** **not** (ctx.permit\_min <= ctx.payout <= ctx.permit\_max): **return** {"status":"skipped","reason":"permit"} **if** ctx.confidence < ctx.win\_threshold: **return** {"status":"skipped","reason":"confidence"} key = (ctx.account, ctx.product, ctx.timeframe) lock = self.locks.setdefault(key, asyncio.Lock()) **if** lock.locked(): **return** {"status":"skipped","reason":"concurrency"} **async** **with** lock: **if** **not** self.risk.enter\_allowed(ctx): **return** {"status":"skipped","reason":"risk\_guard"} amount = self.risk.compute\_amount(ctx) entry\_slot = self.\_entry\_slot(ctx.timeframe) client\_req\_id = f"{ctx.account}:{ctx.product}:{ctx.timeframe}:{entry\_slot}:{uuid.uuid4().hex[:8]}" *# PLACE* t0 = time.time() **try**: ack = **await** self.cx.place\_order(product=ctx.product, amount=amount, direction=ctx.direction, timeframe=ctx.timeframe, client\_req\_id=client\_req\_id) **except** Exception **as** e: *# belirsizlikte tekrar: aynı client\_req\_id ile* **for** n **in** range(1,4): **await** asyncio.sleep(0.25\*(2\*\*n)) **try**: ack = **await** self.cx.place\_order(product=ctx.product, amount=amount, direction=ctx.direction, timeframe=ctx.timeframe, client\_req\_id=client\_req\_id) **break** **except** Exception: ack = None **if** ack **is** None: **return** {"status":"abort","reason":"place\_failed"} t1 = time.time() *# Persist ORDER* order\_id = ack.get("order\_id") **await** self.db.save\_order({ "id": order\_id, "ts\_open\_ms": int(t0\*1000), "account\_id": ctx.account, "product": ctx.product, "timeframe": ctx.timeframe, "direction": ctx.direction, "amount": amount, "client\_req\_id": client\_req\_id, "permit\_win\_min": ctx.permit\_min, "permit\_win\_max": ctx.permit\_max, }) *# Confirm (poll)* deadline = time.time() + self.confirm\_timeout\_s res = None **while** time.time() < deadline: **try**: res = **await** self.cx.confirm\_order(order\_id) **if** res.get("status") **in** ("win","lose","abort","canceled"): **break** **except** Exception: **await** asyncio.sleep(1.0) **await** asyncio.sleep(1.0) **if** res **is** None: **return** {"status":"abort","reason":"confirm\_timeout"} *# Persist RESULT* **await** self.db.save\_result({ "order\_id": order\_id, "ts\_close\_ms": res.get("ts\_close\_ms"), "status": res.get("status"), "pnl": float(res.get("pnl", 0.0)), "duration\_ms": int(res.get("ts\_close\_ms", 0) - int(t0\*1000)), "latency\_ms": int((t1 - t0)\*1000) }) *# Risk feedback* self.risk.on\_result(ctx, pnl=float(res.get("pnl", 0.0)), is\_win=(res.get("status") == "win")) **return** {"status": res.get("status"), "order\_id": order\_id}

11.9 Telemetry & Log Olayları

* *trade\_events*: *PREPARE*, *PLACE\_OK*, *PLACE\_RETRY*, *CONFIRM\_OK*, *CONFIRM\_TIMEOUT*, *SETTLED\_WIN/LOSE/ABORT*, *ABORT\_REASON*.
* Latency dağılımı (p50/p90/p99), retry sayı histogramı, concurrency skip sayısı.

11.10 Reconcile (Boot’ta Kurtarma)

* *orders* içinde *results* olmayan kayıtları topla → connector’dan *order\_id* / *client\_req\_id* ile **durumu sor**.
* Bulunursa *results* ekle; bulunamazsa *status='abort'* ile kapat ve kullanıcıya uyarı.

11.11 UI Eşlemesi

* **Ayarlar → Yürütme**: *confirm\_timeout\_s*, *entry\_cutoff\_s*, *retry\_max\_place*, *retry\_max\_confirm*.
* **Dashboard**: Anlık durum (FSM state), latency, retry sayısı; son işlem özet.
* **Loglar**: *trade.log* içinde redakte edilmiş olaylar; *system.log* hata/sistem olayları.

11.12 Test Planı

* **Birim**: FSM geçişleri, concurrency lock, cut‑off, idempotent *client\_req\_id*.
* **Entegrasyon**: MockConnector ile place/confirm retry senaryoları; belirsizlikte **tek** emir.
* **Dayanıklılık**: Boot’ta reconcile, ağ kesintisi, 429/5xx fırtınası, bakım modu.
* **Zamanlama**: TF hizalama ve *entry\_cutoff\_s* sınaması.

11.13 Kabul Kriterleri (Parça 11)

* Idempotent, at‑most‑once yürütme hattı **tanımlandı** ve iskelet kodu yazıldı.
* FSM, retry matrisi, latency ölçümü, concurrency lock ve reconcile **net**.
* UI ve telemetry eşlemeleri **belirlendi**.

Parça 12 — Worker & Scheduler (TF Hizalama, Zamanlayıcı, Back‑Pressure)

Amaç: Her *(hesap, ürün, TF)* için izole **worker** süreçleri ile veriyi zamanında çekmek, sinyal/ensemble/risk zincirini işletmek, emir akışını başlatmak; bunu yaparken **back‑pressure**, **rate‑limit** ve **eşzamanlılık** kurallarına uymak.

12.1 Tasarım İlkeleri

* **İzolasyon**: Worker anahtarı = *(account\_id, product, timeframe)*; birbirinden bağımsızdır.
* **TF hizalama**: 1/5/15 dk mum kapanışlarına **hizalı** tetikleme (TF=1’de kısa aralıklar).
* **Akış**: *fetch → features → providers → ensemble → risk/preflight → execute → record*.
* **Back‑pressure**: Kuyruklar ve zaman pencereleri ile gecikmiş tiklerin **atlanması** (drop‑old), kritik olmayan iş için **debounce**.
* **Hata toleransı**: Veri/bağlantı hatasında **skip + uyarı**; ardışık hatada **pause** ve otomatik yeniden deneme.

12.2 Zamanlama (Scheduling)

* **Tick üretimi**
  + TF=1: *tick\_interval\_ms=250* ile nazik poll; son kapanışın 0–1s sonrasında **tetik**.
  + TF=5/15: *align\_to\_tf(ts, tf)* ile hizala; kapanıştan sonra *grace\_ms* kadar bekleyip tetik.
* **Jitter**: Aynı anda çok worker çarpışmasını azaltmak için tetiklere ±*jitter\_ms* (örn. 50–150ms) ekle.
* **Cut‑off**: *entry\_cutoff\_s* aktifse mum kapanışına çok yakın **giriş** yapılmaz.

12.3 Back‑Pressure & Rate‑Limit

* **Queue**: *asyncio.Queue(maxsize=N)* ile veri/sinyal/telemetry için ayrık kuyruklar.
* **Drop policy**: Worker, yeni tik geldiğinde eski iş **tamamlanmadıysa** → eski atanır (*skip\_reason='overrun'*).
* **Account semaforu**: Hesap başına *asyncio.Semaphore(R)* ile connector çağrılarında **RPS** sınırı.
* **Global semafor**: Toplam istek sınırı için opsiyonel.

12.4 Worker Döngüsü (Aşamalar)

1. **Fetch**: *get\_candles(product, tf, n=lookback)*; ekran için son *M* (örn. 200) bar.
2. **Feature**: Parça 7–8 indikatör hesapları; NaN/warm‑up kontrolü.
3. **Providers**: Aktif stratejiler → *ProviderVote[]*.
4. **Ensemble**: *combine()* → *{S, confidence, p̂, dir}*.
5. **Risk/Preflight**: permit penceresi, eşik, concurrency, cool‑down.
6. **Execute**: *OrderExecutor.execute(ctx)* (gerekirse).
7. **Record/Publish**: Storage’a yaz ve WS ile UI’a olay/metrik akıt.

12.5 Hata ve Geri Kazanım

* **Tekil hata**: Aşamadan bağımsız *try/except*; *skip* ve **alert**.
* **Ardışık hata sayacı**: *k≥Kmax* ise worker **PAUSED**; *cooldown\_backoff = base·2^k* ile tekrar dene.
* **Reconcile tetik**: Boot’ta ve belirli aralıklarla orphan orders kontrolü.

12.6 Python İskelet — Scheduler

*# core/scheduler.py***import** asyncio, time**from** typing **import** Dict, Tuple**class** Scheduler: **def** \_\_init\_\_(self, worker\_factory, tick\_ms=250, jitter\_ms=100): self.worker\_factory = worker\_factory self.tick\_ms = tick\_ms self.jitter\_ms = jitter\_ms self.workers: Dict[Tuple[str,str,int], asyncio.Task] = {} **def** \_key(self, acc, prod, tf): **return** (acc, prod, tf) **async** **def** start\_worker(self, acc: str, prod: str, tf: int): key = self.\_key(acc, prod, tf) **if** key **in** self.workers **and** **not** self.workers[key].done(): **return** w = self.worker\_factory(acc, prod, tf) task = asyncio.create\_task(w.run(), name=f"worker:{acc}:{prod}:{tf}") self.workers[key] = task **async** **def** stop\_worker(self, acc: str, prod: str, tf: int): key = self.\_key(acc, prod, tf) t = self.workers.get(key) **if** t **and** **not** t.done(): t.cancel() **try**: **await** t **except** asyncio.CancelledError: **pass** **del** self.workers[key] **async** **def** stop\_all(self): **for** key **in** list(self.workers.keys()): **await** self.stop\_worker(\*key)

12.7 Python İskelet — Worker

*# core/worker.py***import** asyncio, time, math**from** dataclasses **import** dataclass**from** typing **import** Optional@dataclass**class** WorkerConfig: lookback: int = 300 grace\_ms: int = 500 tick\_ms: int = 250 jitter\_ms: int = 100 entry\_cutoff\_s: int = 5 overrun\_skip: bool = True**class** Worker: **def** \_\_init\_\_(self, account\_id, product, timeframe, connector, storage, indicators, providers, ensemble, risk, executor, cfg: WorkerConfig): self.acc = account\_id; self.prod = product; self.tf = timeframe self.cx = connector; self.db = storage self.ind = indicators; self.providers = providers self.ens = ensemble; self.risk = risk; self.exec = executor self.cfg = cfg self.\_running = True self.\_last\_close\_slot = None **def** \_tf\_slot(self, ts\_ms: int) -> int: tf\_ms = self.tf \* 60\_000 **return** ts\_ms - (ts\_ms % tf\_ms) **async** **def** run(self): **try**: **while** self.\_running: now = int(time.time()\*1000) slot = self.\_tf\_slot(now) **if** self.\_last\_close\_slot **is** None: self.\_last\_close\_slot = slot *# Kapanış tetiği (bir önceki slot tamamlandı mı?)* **if** slot > self.\_last\_close\_slot: **await** self.\_on\_close(self.\_last\_close\_slot) self.\_last\_close\_slot = slot *# Kısa tiklerde ara işlemler (TF=1 için daha sık refresh)* **await** asyncio.sleep(self.cfg.tick\_ms/1000.0) **except** asyncio.CancelledError: self.\_running = False **async** **def** \_on\_close(self, close\_slot\_ms: int): *# Cut‑off: Kapanıştan hemen önce girmeyi engelle (uygunsa)* **try**: *# 1) Fetch* candles = **await** self.cx.get\_candles(self.prod, self.tf, n=self.cfg.lookback) **if** **not** candles **or** len(candles) < 30: **return** *# pandas DF’e dönüştür (helper varsayılıyor)* df = to\_dataframe(candles) *# ts\_ms, open, high, low, close, volume* *# 2) Features* feats = compute\_features(df, self.ind) *# 3) Providers* votes = [] **for** p **in** self.providers: *# p: StrategyProvider* v = p.evaluate(df, feats) **if** v **is** **not** None: votes.append(v) **if** **not** votes: **return** *# 4) Ensemble* comb = self.ens.combine(votes) direction = 'call' **if** comb['dir']>0 **else** ('put' **if** comb['dir']<0 **else** None) **if** **not** direction: **return** *# 5) Risk/Preflight* payout = **await** self.cx.get\_current\_win\_rate(self.prod) ctx = build\_trade\_ctx(self.acc, self.prod, self.tf, direction, payout, comb) **if** **not** self.risk.enter\_allowed(ctx): **return** *# 6) Execute* result = **await** self.exec.execute(ctx) *# 7) Record/Publish (executor zaten yazar; burada ek metrik/alert üretilebilir)* **except** Exception **as** e: *# log + alert* **return**

12.8 UI Eşlemesi

* **Ayarlar → Motor**: *tick\_ms*, *grace\_ms*, *jitter\_ms*, *entry\_cutoff\_s*, *lookback*.
* **Durum**: Worker listesi (acc, prod, tf, state=RUNNING/PAUSED), son kapanış zamanı, ardışık hata sayacı.
* **Kontrol**: Worker başlat/durdur; ürün/TF bazında hızlı aktivasyon.

12.9 Test Planı

* **Fonksiyonel**: TF hizalaması doğru; kapanış tetiklerinin kaçmaması.
* **Overrun**: Uzun süren bar işleminde bir sonraki tetikte **skip**; sistem tıkanmıyor.
* **Rate‑limit**: Semaforla istek sayısı sınırlanıyor; 429 testlerinde backoff çalışıyor.
* **Çoklu hesap**: 4 hesapla paralel workers; izolasyon korunuyor.
* **Dayanıklılık**: Ardışık hata sonrası *PAUSED* → backoff → *RUNNING* döngüsü.

12.10 Kabul Kriterleri (Parça 12)

* Worker/scheduler iskeleti, TF hizalama ve back‑pressure politikaları **tanımlandı**.
* Rate‑limit semaforları ve drop‑old stratejisi **belirlendi**.
* Hata sayacı, pause/backoff ve UI eşleşmesi **dokümante** edildi.
* Çoklu hesap paralel çalışmada stabilite hedefleri **net**.

Parça 13 — Strateji Kataloğu (Plugin Sistemi) & Sinyal Sağlayıcıları

Amaç: İndikatör/özellik katmanının üzerine oturan **sinyal sağlayıcıları (strategy providers)** için **eklenti (plugin)** mimarisi kurmak; kullanıcıya ürün/TF bazında strateji seçimi ve parametre override imkânı vermek; her sağlayıcıdan **vote ∈ {−1,0,1}** ve **score ∈ ℝ** üretmek.

13.1 Mimarinin İlkeleri

* **Gevşek Bağlılık**: Stratejiler *core/strategies/providers/\*.py* altında **bağımsız modüller**dir.
* **Sözleşme**: Tüm stratejiler tek bir **Provider** arayüzünü uygular; **registry** üzerinden keşfedilir.
* **Konfigüre Edilebilirlik**: Varsayılan parametreler → **config/UI** ile ürün/TF bazında override.
* **İzlenebilirlik**: Her evaluate çağrısı *meta* sözlüğünde katkı gerekçesini taşır (örn. *{"ema20":..., "rsi14":...}*).
* **Isınma/Warm‑up**: Gerekli bar sayısı strateji tarafından bildirilecek; yetersizse *vote=0* döner.

13.2 Dizin Yapısı

core/strategies/ base.py # Provider arayüzü & yardımcılar registry.py # Keşif & kayıt providers/ ema\_rsi.py # (ID: 5/6/7/8/9/10 varyantları) vwap\_rvol.py # (ID: 15..20 varyantları) st\_adx.py # (ID: 25..30 varyantları) keltner\_break.py # (ID: 35..40 varyantları) bb\_walk.py # (ID: 45..50 varyantları) ema\_cross.py # (ID: 14, 9/21 EMA) triple\_ma.py # (ID: 34) # ileride: ichimoku\_trend, gmmaz, pivot\_reversal vb.

13.3 Provider Arayüzü

*# core/strategies/base.py***from** \_\_future\_\_ **import** annotations**from** dataclasses **import** dataclass, field**from** typing **import** Dict, Any, Optional**import** pandas **as** pd**from** core.ensemble **import** ProviderVote@dataclass**class** ProviderContext: product: str timeframe: int payout: float@dataclass**class** ProviderConfig: id: int name: str group: str params: Dict[str, Any] = field(default\_factory=dict)**class** StrategyProvider: cfg: ProviderConfig **def** warmup\_bars(self) -> int: *"""Dönmesi gereken minimum bar sayısı."""* **raise** NotImplementedError **def** evaluate(self, df: pd.DataFrame, feats: Dict[str, Any], ctx: ProviderContext) -> Optional[ProviderVote]: *"""  
 df: ts\_ms, open, high, low, close, volume  
 feats: Parça 7-8'den hesaplanmış Series'ler veya tekil değerler  
 return ProviderVote(pid=self.cfg.id, vote=-1|0|1, score=float)  
 """* **raise** NotImplementedError

13.4 Registry (Keşif/Kayıt)

*# core/strategies/registry.py***from** typing **import** Dict, Type, List**from** .base **import** StrategyProvider**import** importlib, pkgutil\_REG: Dict[int, Type[StrategyProvider]] = {}**def** register(pid: int): **def** deco(cls: Type[StrategyProvider]): \_REG[pid] = cls **return** cls **return** deco**def** load\_all(): **import** core.strategies.providers **as** pkg **for** m **in** pkgutil.iter\_modules(pkg.\_\_path\_\_, pkg.\_\_name\_\_ + "."): importlib.import\_module(m.name)**def** build(pid: int, \*\*kw) -> StrategyProvider: cls = \_REG[pid] **return** cls(\*\*kw)**def** all\_metadata() -> List[dict]: **return** [getattr(v, 'META', {"id": k}) | {"id": k} **for** k,v **in** \_REG.items()]

13.5 Örnek Sağlayıcılar (Mantık + Parametreler)

13.5.1 EMA Trend + RSI (ID 5/6/7/8/9/10 varyantları)

* **Mantık**: *EMA(20) > EMA(50)* **ve** *RSI14 > 55* → **long** oyu (*vote=+1*). Tersi *EMA20 < EMA50* & *RSI14 < 45* → **short** (*vote=-1*). Aksi 0.
* **Skor**: *score = w1·slope(EMA20) + w2·(RSI14−50)/50* (normalize); tavan *±s\_cap*.
* **Parametreler**: *ema\_fast=20*, *ema\_slow=50*, *rsi\_len=14*, *rsi\_up=55*, *rsi\_dn=45*, *w1=1.0*, *w2=0.7*.
* **Warm‑up**: *max(ema\_slow, rsi\_len)+5*.
* **Not**: Kullanıcının tablodaki SL/TP referansları FTT’de oturum‑risk politikasıyla karşılanır (Parça 10).

*# core/strategies/providers/ema\_rsi.py***from** .base **import** StrategyProvider, ProviderConfig, ProviderContext**from** core.ensemble **import** ProviderVote**from** core.indicators.basic **import** ema, rsi**import** pandas **as** pd**class** EMA\_RSI(StrategyProvider): META = {"name":"EMA Trend + RSI", "group":"trend+osc"} **def** \_\_init\_\_(self, cfg: ProviderConfig): self.cfg = cfg p = self.cfg.params self.ema\_fast = p.get('ema\_fast', 20) self.ema\_slow = p.get('ema\_slow', 50) self.rsi\_len = p.get('rsi\_len', 14) self.rsi\_up = p.get('rsi\_up', 55) self.rsi\_dn = p.get('rsi\_dn', 45) self.w1 = p.get('w1', 1.0); self.w2 = p.get('w2', 0.7) **def** warmup\_bars(self) -> int: **return** max(self.ema\_slow, self.rsi\_len) + 5 **def** evaluate(self, df: pd.DataFrame, feats, ctx: ProviderContext): close = df['close'] e20 = ema(close, self.ema\_fast) e50 = ema(close, self.ema\_slow) r = rsi(close, self.rsi\_len) **if** len(close) < self.warmup\_bars(): **return** None trend\_up = e20.iloc[-1] > e50.iloc[-1] trend\_dn = e20.iloc[-1] < e50.iloc[-1] vote = 0 **if** trend\_up **and** r.iloc[-1] > self.rsi\_up: vote = 1 **elif** trend\_dn **and** r.iloc[-1] < self.rsi\_dn: vote = -1 **if** vote == 0: **return** ProviderVote(pid=self.cfg.id, vote=0, score=0.0) slope = (e20.iloc[-1] - e20.iloc[-3]) / max(1e-6, e20.iloc[-3]) score = self.w1 \* slope + self.w2 \* ((r.iloc[-1] - 50.0)/50.0) **return** ProviderVote(pid=self.cfg.id, vote=vote, score=float(score))

13.5.2 VWAP Reclaim + RVOL (ID 15..20)

* **Mantık**: Fiyat **VWAP altından üstüne** *reclaim* **ve** *RVOL>thr* → **long**; tersi için short.
* **Skor**: *score = α·reclaim\_strength + β·(rvol−1)*; *reclaim\_strength = (close−vwap)/vwap* sonrası *>0*.
* **Parametreler**: *rvol\_thr ∈ {1.1, 1.15, 1.2, 1.3, 1.5}*, *α=1.0*, *β=0.5*.
* **Warm‑up**: *max(20, rvol\_lb)*.

13.5.3 Supertrend + ADX (ID 25..30)

* **Mantık**: *supertrend\_state == 'bullish'* **ve** *adx14 > thr* → **long**; bearish & adx>thr → **short**.
* **Parametreler**: *atr\_len=10*, *mult=3.0*, *adx\_thr ∈ {18,20,22,24,26}*.
* **Skor**: *score = (adx14−thr)/30 + γ·dist\_to\_st*, *dist\_to\_st = (close−st\_line)/close* işaretli.

13.5.4 Keltner Break (ID 35..40)

* **Mantık**: Mum kapanışı *KC\_upper* **üstünde** (veya short için *KC\_lower* altında). ADX ile opsiyonel doğrulama.
* **Parametreler**: *ema\_len=20*, *atr\_len=10*, *mult ∈ {1.5..1.8}*, *adx\_min=0|20*.
* **Skor**: *score = (close−KC\_upper)/KC\_mid + δ·(adx14/50)*.

13.5.5 Bollinger Walk (ID 45..50)

* **Mantık**: **Ardışık ≥2** kapanış **üst banda**; mid altına inmedikçe long yürüyüşü sürer (short için alt bant).
* **Parametreler**: *length=20*, *mult∈{2.0,2.2}*, *walk\_len=2*.
* **Skor**: *score = (close−bb\_upper)/bb\_mid + ζ·bandwidth*.

13.5.6 9/21 EMA Crossover (ID 14)

* **Mantık**: *EMA9* **üstüne keserse** long; altına keserse short. Whipsaw azaltmak için *confirm\_len=1..3* (bar sayısı) ve *adx\_min* filtresi.
* **Skor**: *score = slope(EMA9−EMA21)*.

13.5.7 Triple MA 5/20/50 (ID 34)

* **Mantık**: *5>20>50* long, *5<20<50* short; aksi 0.
* **Skor**: *score = min( d(5,20), d(20,50) )* (normalize farkların min’i), trend gücü ölçümü.

13.6 Parametre Varyantları ve ID Eşleme

* Kullanıcının verdiği tabloya göre **aynı mantığın farklı eşiğe sahip varyantları** ayrı ID olarak kayıtlıdır (örn. VWAP+RVOL 1.10,1.15,1.2…). UI’da tek strateji altında **ön tanım presetleri** listelenir; kullanıcı isterse **serbest** değer girer ve bu özel varyant **custom** ID ile tutulur.

13.7 Skor ve Vote Politikaları

* **Vote** yalnızca ana koşullar sağlandığında ±1; aksi 0.
* **Score** sürekli bir güven ölçüsü: eşik mesafeleri, eğimler, bant dışı mesafeler, hacim teyitleri gibi bileşenlerin **lineer kombinasyonu**, *z‑score* veya *tanh* ile yumuşatma; **s\_cap** ile katkı tavanı (Parça 9).
* **Normalize**: Ensemble katmanı her sağlayıcı için kayan pencere ort/sigma ile standardize eder.

13.8 UI / Konfig Entegrasyonu

* **Strateji Kataloğu** ekranı: ID, ad, grup, warm‑up, varsayılan paramlar, açıklama, notlar.
* Ürün/TF sayfasında **aktif/pasif** anahtarı, preset seçimi veya **özel param** girişi.
* Her sağlayıcı için **son N işlem win%**, ort. katkı, ort. latency ve **w\_i** ağırlığı gösterilir (Parça 9 telemetrisi).
* **Profil Kaydet/Yükle**: Ürün/TF parametre setleri profil olarak kaydedilip çağrılabilir.

13.9 Test Planı

* **Birim**: Her sağlayıcı için sınır koşulları (ör. reclaim tespiti, ardışık kapanış sayacı, MA kesişim tespiti).
* **Regresyon**: Param değişimlerinin beklenmeyen davranış üretmediği (ör. mult 1.5→1.6 değişimi).
* **Performans**: 100+ provider kombinasyonunda değerlendirme süresi TF=1 için < 50ms hedef.
* **Uyumluluk**: Warm‑up yetersizse vote=0; NaN yayılımı testi.

13.10 Kabul Kriterleri (Parça 13)

* Plugin mimarisi, base arayüz, registry ve örnek stratejiler **tanımlandı**.
* Kullanıcı tablosundaki strateji aileleri ID eşleşmeleriyle **kataloğa dahil** edildi.
* UI/konfig override, telemetry ve test planı **net**.
* Vote/score üretim politikaları ve warm‑up kuralları **belirlendi**.

Parça 14 — UI (Flutter) Bilgi Mimarisi & Ekran Akışları

Amaç: Windows 10/11 üzerinde çalışan Flutter masaüstü uygulamasının **bilgi mimarisi**, **ekran akışları**, **temel bileşenleri** ve **durum yönetimi**ni tanımlamak. Kullanıcı, 4 hesaba kadar aynı anda yönetim yapacak; tüm kritik ayarlar görsel olarak kontrol edilebilir olacak.

14.1 Teknoloji ve Mimarî

* **Framework**: Flutter (stable channel), hedef **Windows desktop**.
* **Durum Yönetimi**: Riverpod (StateNotifier/AsyncNotifier) — basit, test edilebilir.
* **İletişim**: *Core* servis ile **REST (http)** ve **WebSocket** (anlık metrik/olay).
* **Modülerlik**: Özellik bazlı paketleme (feature-first): *accounts*, *products*, *strategies*, *trading*, *settings*, *logs*, *charts*.
* **Theming**: Light/Dark; renk tokenları (mor/mavi/yeşil/kırmızı/siyah/beyaz) ve erişilebilir kontrast.
* **Yerelleştirme**: *tr-TR* varsayılan; *intl* ile çok dillilik.

14.2 Ana Gezinti (Navigation)

* **Sol Kenar Çubuk** (NavigationRail):
  1. **Dashboard**
  2. **Hesaplar**
  3. **Ürünler & TF**
  4. **Strateji Kataloğu**
  5. **İşlemler & Geçmiş**
  6. **Grafikler**
  7. **Ayarlar**
  8. **Loglar/Alarm**
* **Üst Bar**: Aktif hesap seçici (dropdown, çoklu seçim), **Kill-Switch**, **Play/Pause** (global), tema anahtarı, durum LED’leri (Core bağlantı, WS canlı, servis durumu).

14.3 Ekranlar ve İçerikler

14.3.1 Dashboard

* **KPI Kartları**: Toplam PnL (gün/hafta/ay), win-rate (demo/gerçek ayrı), açık işlem sayısı, ort. latency, ardışık kayıp, günlük risk kullanımı.
* **Canlı Akış**: Son işlemler tablosu (hesap, ürün, TF, yön, tutar, durum, payout, p̂, latency).
* **Durum Paneli**: Core servis trạng, scheduler/worker sayısı, hata sayacı, alarm bildirimleri.
* **Hızlı Aksiyonlar**: Tüm ürünlerde Başlat/Durdur; Profil yükle/kaydet.

14.3.2 Hesaplar

* **Hesap Listesi** (en fazla 4): *id, kullanıcı adı, profil yolu, durum (bağlı/bağsız), son login, bakiye (okunabiliyorsa)*.
* **Güvenli Giriş**: Keyring servis adı; parola girişi ayrı modal (parola UI’da saklanmaz).
* **Oturum Testi**: Login denemesi, heartbeat, izin/payout sorgusu.
* **Limitler**: Hesap bazlı *max\_parallel*, *max\_daily\_loss*, *max\_consecutive\_losses*.

14.3.3 Ürünler & TF

* **Ağaç Görünüm**: Ürün → TF(1/5/15).
* Her TF için kart: **Enabled**, *win\_threshold*, *permit\_min/max*, *lot/tutar modu*, **concurrency=1 (sabit)**, cool‑down.
* **Çoklu seç-ke uygulama**: Birden fazla TF’ye aynı ayarı uygula (ör. permit aralığını topluca değiştir).

14.3.4 Strateji Kataloğu

* **Kart Izgara**: (ID, ad, grup) + kısa açıklama.
* **Detay Modal**: Parametreler (ör. EMA 20/50, RSI 14, RVOL thr…), warm‑up, son N işlemde win%.
* **Presetler**: Kullanıcının sunduğu varyantlar (ör. RVOL 1.1/1.15/1.2/1.3/1.5) tek tıkla.
* **Aktifleştirme**: Ürün/TF bazında stratejiyi aç/kapat ve param override.

14.3.5 İşlemler & Geçmiş

* **Tablo**: Zaman, hesap, ürün, TF, yön, tutar, payout, p̂, sonuç (win/lose/abort), PnL, latency.
* **Filtreler**: Tarih aralığı, ürün, TF, hesap, sonuç.
* **Dışa Aktarım**: CSV/Parquet.
* **Detay**: Bir işlem tıklandığında order/result/features özet (gizlilik kurallarıyla).

14.3.6 Grafikler

* **Mum Grafiği** + overlay: EMA/SMA/BB/Keltner/Supertrend/Ichimoku seçilebilir.
* **Alt Paneller**: RSI, MACD, Stoch, ADX, CMF, RVOL.
* **Sinyal İşaretleri**: Sağlayıcı vote’larının işaretçileri (↑/↓), giriş noktaları, sonuç renklendirmesi.
* **Performans Grafikleri**: Rolling win‑rate, PnL eğrisi, latency histogramı.

14.3.7 Ayarlar

* **Genel**: Dil, tema, tablo yoğunluğu, tarih/saat formatı.
* **Motor (Engine)**: *tick\_ms*, *grace\_ms*, *jitter\_ms*, *entry\_cutoff\_s*, *lookback*.
* **Risk**: *max\_daily\_loss*, *max\_consec\_losses*, *amount\_mode*, *fixed\_a*, *frac*, *kelly\_scale*, *a\_min*, *a\_cap*.
* **Permit/Eşik**: Ürün/TF bazlı varsayılanlar.
* **Ensemble**: *ensemble\_mode*, *c\_min*, *s\_cap*, *alpha*, *w\_max*, kalibrasyon durumu ve yeniden kalibre et.
* **Bağlantı**: Core loopback adresi, WS tercihi; yalnızca yerel izin.

14.3.8 Loglar & Alarm

* **Canlı Log** (WS): *system.log* ve *trade.log* akışı; arama/filtreleme.
* **Alarm Merkezi**: Guardrail tetikleri (günlük kayıp, ardışık kayıp, latency abort, permit dışı, connector hata).
* **Bildirim**: Windows toast (ops.) + uygulama içi badge.

14.3.9 Onboarding

* 3 adım sihirbaz: (1) Hesap ekle & parola kasası, (2) Ürün/TF + eşikler, (3) Strateji presetleri & risk.
* Son adımda **Paper Mod** ile başlat önerisi; kill‑switch öğretimi.

14.4 Durum Mimarisi (Riverpod)

* **Providers**:
  + *coreClientProvider* (REST/WS istemcisi)
  + *accountsProvider* (liste + bağlantı durumu)
  + *productsProvider* (ağaç yapı)
  + *workersProvider* (aktif *(acc,prod,tf)*)
  + *metricsStreamProvider* (WS’den anlık metrik)
  + *logsStreamProvider* (log akışı)
  + *ordersNotifier* (arama/filtre/indir)
  + *settingsNotifier* (config görüntüle/güncelle)
* **Hata Durumları**: AsyncValue ile yükleniyor/başarısız; kullanıcıya aksiyon önerisi (yeniden dene, config aç).

14.5 Performans & UX

* **Sanal Liste**: Büyük tablolarda virtualization.
* **Debounce/Throttle**: Arama/filtre alanları.
* **Grafik Performansı**: İzlence penceresi (ör. 500–2000 bar) + lazily compute.
* **Klavye Kısayolları**: *Ctrl+K* (komut paleti), *Ctrl+P* (Play/Pause), *Ctrl+Shift+X* (kill‑switch).
* **Erişilebilirlik**: Kontrast ≥ 4.5:1, focus ring, screen reader etiketleri.

14.6 Tema & Renkler (Tokenlar)

* **Temel Renkler**:
  + Mor: *#6D28D9*
  + Mavi: *#2563EB*
  + Yeşil (pozitif): *#10B981*
  + Kırmızı (negatif/uyarı): *#EF4444*
  + Siyah: *#0B0F17* (dark bg)
  + Beyaz: *#FFFFFF* (light bg)
* **Durum Renkleri**:
  + **WIN**: Yeşil
  + **LOSE**: Kırmızı
  + **ABORT/PAUSE**: Kehribar *#F59E0B*
  + **WS/CORE LED**: Gri→Mavi→Yeşil aşamalı.

14.7 REST/WS Uçları — UI Kullanımı

* **GET** */status*, */accounts*, */workers*, */products*, */orders?filter=...*
* **PUT** */config* (hot‑reload alanları)
* **POST** */start*, */stop*, */orders/cancel*
* **WS kanalları**: *metrics*, *trade\_updates*, *alerts*, *logs* (ayrı topic’ler)

14.8 Proje Yapısı (Flutter)

ui\_app/├─ lib/│ ├─ main.dart│ ├─ app.dart (Router / Theme)│ ├─ core/ (http/ws client, models)│ ├─ features/│ │ ├─ dashboard/│ │ ├─ accounts/│ │ ├─ products/│ │ ├─ strategies/│ │ ├─ trading/│ │ ├─ charts/│ │ ├─ settings/│ │ └─ logs/│ ├─ widgets/ (cards, tables, forms, dialogs)│ └─ theme/ (tokens, light.dart, dark.dart)└─ pubspec.yaml

14.9 Bileşenler (Örnek)

* **AccountCard**: durum LED, login testi, limit ayarları.
* **TFCard**: permit, win‑threshold, lot modu; concurrency/guardrail rozetleri.
* **StrategyTile**: ad, açıklama, preset seçici, param formu.
* **MetricSparkline**: mini win‑rate/PnL eğrisi.
* **LogView**: filtrelenebilir canlı log.
* **OrderTable**: sanal liste + sütun seçici + CSV indir.

14.10 Hata/Alarm UX Politikası

* Kritik (kill‑switch tetik, günlük kayıp aşımı): **Modal + Toast + Alarm sayacı**.
* Orta (retry, backoff): Banner/Toast.
* Düşük (UI doğrulama): inline hata mesajı.
* **Tekrar eden alarmlar** konsolide edilir; **sessize al** seçeneği.

14.11 Güvenlik & Gizlilik UX

* Parola girişleri **maskeli** ve yalnız keyring’e gider.
* Loglarda PII **yok**; hassas alanlar *•••* maskeli gösterilir.
* Dışa aktarımda kullanıcıya **anonimleştirme** seçeneği sunulur.

14.12 Kabul Kriterleri (Parça 14)

* Tüm ana ekranlar ve gezinti akışı **tanımlandı**.
* Riverpod tabanlı durum modeli ve WS/REST entegrasyonu **netleşti**.
* Renk/tema tokenları, kontrast ve erişilebilirlik ilkeleri **belirlendi**.
* Tablo/grafik performansı, hata/ alarm UX politikaları ve bileşen listesi **hazır**.

Parça 15 — Core↔UI API (REST/WS) Sözleşmeleri & Şema (Windows Loopback)

Amaç: Core servisi ile Flutter masaüstü UI arasında **kararlı**, **versiyonlu** ve **test edilebilir** bir API tanımlamak. İletişim yalnızca **yerel loopback** üzerinden (varsayılan *127.0.0.1*) yapılır; dış ağ erişimi yoktur. İsteğe bağlı yerel **API anahtarı** ile ek doğrulama.

15.1 Tasarım İlkeleri

* **Yerel ve güvenli**: Varsayılan bağlayıcı *http://127.0.0.1:8750* ve *ws://127.0.0.1:8751* (opsiyonel TLS: *https/wss*).
* **Basit kimlik**: Loopback üzerinde **auth gereksiz**; istenirse *X-API-Key* başlığı ile tek anahtar.
* **Sürümleme**: *api\_version* (örn. *1.0*), *schema\_version* ve *config\_version* alanları.
* **İdempotent işlemler**: Emir işlemleri *client\_request\_id* ile tekrara dayanıklı.
* **Sözleşme önce**: Açık OpenAPI çıktısı (ops.) + UI tarafında tip güvenli istemci.
* **Zaman & Yerel ayar**: API tüm zamanları **UTC ms** döndürür; UI sistem saat diliminde (**Europe/Istanbul**) gösterir.

15.2 REST Uçları (taslak)

15.2.1 *GET /status*

**Yanıt**

{ "api\_version": "1.0", "service": {"state":"running","uptime\_s": 12345, "tz":"UTC"}, "core": {"workers": 42, "cpu\_pct": 18.2, "mem\_mb": 512}, "accounts": [{"id":"acc1","state":"connected"},{"id":"acc2","state":"disconnected"}], "ws": {"metrics":"connected","alerts":"connected"}}

15.2.2 *GET /accounts*

Liste + durum bilgisi döner.

[{"id":"acc1","username\_mask":"u\*\*\*@mail","profile\_store":"profiles/acc1/","state":"connected","last\_login\_ms":1728291000000,"balance":1200.50}]

15.2.3 *POST /accounts/login*

**Girdi**: *{ "id":"acc1", "otp":"123456" }* (OTP opsiyonel). Parola keyring’den çekilir. **Çıktı**: *{ "ok": true, "state": "connected" }*

15.2.4 *GET /workers*

Aktif *(account, product, timeframe)* listesi ve durumları.

[{"account":"acc1","product":"EURUSD","tf":1,"state":"RUNNING","last\_close\_ms":1728291600000}]

15.2.5 *POST /start* / *POST /stop*

**Girdi** (örnek, kapsam bazlı):

{"scope":"global"}{"scope":"account","account":"acc1"}{"scope":"worker","account":"acc1","product":"EURUSD","tf":1}

**Çıktı**: *{ "ok": true, "affected": [ ... ] }*

15.2.6 *GET /products*

UI ağacı için ürün & TF yapılandırması döner (aktif/pasif, eşi̇kler, permit aralığı, lot modu).

15.2.7 *PUT /config*

Hot‑reload alanları anında uygulanır; diğerleri kontrollü restart ister. **Girdi**: *partial* (yalnız değişen alanlar) veya *full* (tam dosya). **Çıktı**:

{"applied":["products[EURUSD].timeframes[1].win\_threshold"],"restarted":["workers:acc1:EURUSD:1"],"warnings":["win\_threshold < ensemble\_threshold"]}

15.2.8 *GET /orders* / *GET /orders/{id}*

Filtreler: *from\_ms,to\_ms,account,product,tf,status*.

15.2.9 *POST /orders/cancel*

Platform destekliyse erkenden kapama isteği. **Girdi**: *{ "order\_id":"..." }* → **not**: onay broker kabiliyetine bağlı.

15.2.10 *GET /logs* / *GET /metrics*

Son *tail* satırı veya *since\_ms*’ten itibaren özet metrikler.

15.2.11 *POST /reconcile*

Boot’taki otomatik sürece ek olarak manuel tetik.

15.2.12 Hata Zarfı (tüm uçlar)

{ "error": { "code": "CONCURRENCY\_BLOCK",  "message": "(acc,product,tf) başına tek açık işlem kuralı", "details": {"account":"acc1","product":"EURUSD","tf":1} }}

Standart kodlar: *VALIDATION*, *AUTH*, *THROTTLE*, *MAINTENANCE*, *NETWORK*, *BUSINESS\_RULE*.

15.3 WebSocket Kanalları

* **Endpoint**: *ws://127.0.0.1:8751/ws* (sorgu: *?topic=metrics|trade\_updates|alerts|logs*)
* **Kalp atışı**: *{ "type":"ping" }* ↔ *{ "type":"pong" }* her 20sn.
* **Yeniden bağlanma**: UI exponential backoff (250ms→4s), durum LED’i güncellenir.

15.3.1 *metrics* örnekleri

{"topic":"metrics","ts\_ms":1728291660000,"scope":"acc:acc1","key":"win\_rate\_1m","value":0.73}{"topic":"metrics","ts\_ms":1728291660000,"scope":"prod:EURUSD","key":"latency\_p50\_ms","value":140}{"topic":"metrics","ts\_ms":1728291660000,"scope":"global","key":"pnl\_day","value":2.4}

15.3.2 *trade\_updates*

{"topic":"trade\_updates","event":"NEW","order\_id":"SIM-1728291600123","account":"acc1","product":"EURUSD","tf":1,"amount":1.0,"payout":0.9,"p\_hat":0.74}{"topic":"trade\_updates","event":"SETTLED","order\_id":"SIM-1728291600123","status":"win","pnl":0.9,"latency\_ms":120,"ts\_close\_ms":1728291660000}

15.3.3 *alerts*

{"topic":"alerts","level":"WARN","code":"DAILY\_LOSS\_NEAR","message":"Günlük kayıp limitine yaklaşıldı","account":"acc1","pnl\_day":-4.2}{"topic":"alerts","level":"ERROR","code":"CIRCUIT\_BREAKER\_TRIP","message":"Ardışık kayıp limiti aşıldı"}

15.3.4 *logs*

{"topic":"logs","chan":"trade","level":"INFO","msg":"PLACE\_OK","order\_id":"SIM-...","masked":"u\*\*\*@mail"}

15.4 Güvenlik & Ağ Politikaları

* **Loopback zorunlu**: Varsayılan olarak yalnız *127.0.0.1* dinlenir. İstenirse UI’dan **yalnızca yerel** bayrağı kaldırılıp **IP allow‑list** ile sınırlanır.
* **TLS (ops.)**: Self‑signed sertifika desteklenir; UI’da güven düğümü. Windows Firewall’da **özel kural** önerilir.
* **CORS**: Kapalı; UI native istemci olduğu için gerekmez.
* **Rate‑limit**: REST 30 RPS, WS 1 kanal başına 10 msg/sn (UI yeterlidir).
* **PII maskesi**: Kullanıcı adı/e‑posta/OTP log ve ws’de maskeleme (*u\*\*\*@mail*).

15.5 Versiyonlama ve Uyumluluk

* *GET /status* → *{ "api\_version": "1.0", "schema\_version": "2025-10-01", "config\_version": "1.0.0" }*
* Kırıcı değişikliklerde ana sürüm artar; UI sürüm kontrolü uyarı verir ve uyum moduna geçer (salt okunur).

15.6 OpenAPI & İstemci (opsiyonel)

* Core servis başlatıldığında *GET /openapi.json* üretir.
* UI tarafında *openapi-generator* ile Dart istemci kodu üretilebilir (isteğe bağlı). Elle yazılmış hafif istemci de yeterli.

15.7 Örnek Sunucu İskeleti (FastAPI)

*# core/api/server.py***from** fastapi **import** FastAPI, WebSocket, WebSocketDisconnect, Header**from** fastapi.responses **import** JSONResponseapp = FastAPI(title="MoonLight Core API", version="1.0")API\_KEY = None *# opsiyonel*@app.get("/status")**async** **def** status(x\_api\_key: str | None = Header(default=None)): **if** API\_KEY **and** x\_api\_key != API\_KEY: **return** JSONResponse(status\_code=401, content={"error":{"code":"AUTH","message":"bad key"}}) **return** {"api\_version":"1.0","service":{"state":"running","uptime\_s":0,"tz":"UTC"}}@app.websocket("/ws")**async** **def** ws\_endpoint(ws: WebSocket): **await** ws.accept() **try**: **while** True: msg = **await** ws.receive\_text() **if** msg == '{"type":"ping"}': **await** ws.send\_text('{"type":"pong"}') **except** WebSocketDisconnect: **pass**

15.8 Test Planı (API)

* **Birlikte çalışabilirlik**: UI tüm uçları çağırır; 200/4xx/5xx durumları doğru ele alınır.
* **Yük**: *GET /status* 50 RPS, *metrics* WS 10 msg/sn; UI kare hızında gecikme yok.
* **Hata zarfı**: Tüm hata senaryoları standardize dönüyor (JSON).
* **Güvenlik**: Loopback dışında bağlanma **başarısız**; API anahtarı açıkken yanlış anahtar → 401.

15.9 Kabul Kriterleri (Parça 15)

* REST ve WS sözleşmeleri **tanımlandı** ve örnek payload’lar verildi.
* Hata zarfı, sürümleme, güvenlik ve oran sınırlama **net**.
* UI için gereken tüm endpoint ve kanallar mevcut; idempotency ve loopback ilkeleri korunuyor.

Parça 16 — Backtest & Paper Trading Motoru (Metrikler, Walk‑Forward)

Amaç: Strateji/ensemble/risk zincirini **tarihsel veride** ve **anlık simülasyonda** (paper) gerçekçi kurallarla test etmek; FTT doğasına uygun **vade** ve **payout** modellemesi yapmak; sonuçları ayrıntılı metriklerle raporlamak ve **aşırı uyum** riskini azaltmak.

16.1 Kapsam ve Varsayımlar

* **Enstrüman**: Fixed‑Time Trade (FTT) — giriş bar **kapanışında** yapılır; vade **TF adımı** kadardır (1/5/15 dk).
* **Payout (R)**: Zamanla **değişken** olabilir. Backtest’te iki mod:
  1. **Sabit**: Kullanıcı tanımlı (örn. %90).
  2. **Seri**: Zaman serisi (csv) ile her bara özel *payout%*.
* **Gecikme/Slip**: İsteğe bağlı *latency\_ms* ve *price\_offset* modeli; FTT’de fiyat ofseti sınırlı etkide, giriş/kapanış **bar kapanış fiyatı**yla varsayılır.
* **Concurrency**: Aynı *(ürün, TF)* için **tek açık** pozisyon.
* **Permit**: *permit\_min ≤ payout ≤ permit\_max* değilse **giriş yok**.

16.2 Veri Girişi

* **Candles**: *ts\_ms, open, high, low, close, volume* (UTC ms, artan sıralı).
* **Payout**: *ts\_ms, payout* (yüzde). Yoksa sabit değer kullanılır.
* **Split & Align**: Candles ile payout serisi **TF slotu** üzerinden birleştirilir.

16.3 Simülasyon Döngüsü (Olay Akışı)

for her kapanan bar (t): 1) Özellikleri hesapla (Parça 7–8) 2) Aktif sağlayıcıları değerlendir → ProviderVote[] 3) Ensemble.combine → {S, confidence, p̂, dir} 4) Risk/Permit/Concurrency önkontrolü 5) Giriş kararı: TRUE ise trade aç → vade = TF adımı 6) t+TF sonunda kapanış: kazan/kaybet belirle (FTT outcome) 7) PnL = +amount\*R (win) veya −amount (lose) 8) Kayıt/metrik güncelle → rapor

**FTT sonuç kuralı** (örn. CALL): *close(t+TF) > close(t)* → **win**, eşitse **push** (varsayılanı lose değil; platform kuralına göre ayarlanabilir).

16.4 Metrikler

* **Temel**: İşlem sayısı, win‑rate, ort. payout, expectancy *E = p̂·R − (1−p̂)*, kâr faktörü, ort. işlem kârı.
* **Risk**: Max DD, Calmar, Sharpe, Sortino (ops.), max ardışık kayıp, günlük PnL dağılımı.
* **Stabilite**: Ürün/TF kırılımında win‑rate, saat‑gün ısı haritası, rejimlere göre (ADX↑/↓, RVOL↑/↓) performans.
* **Kalibrasyon**: Brier, LogLoss, **reliability diagram** (p̂ dilimleri vs gerçek win).
* **Operasyonel**: Permit skip oranı, concurrency skip, ort. latency varsayımı etkisi.

16.5 Walk‑Forward & CV

* **Zaman temelli ayrım**: **Genişleyen** (expanding) veya **kaydırmalı** (rolling) pencereler.
* **WF protokolü**:
  + Train: [t0, t1) → ağırlık/kalibrasyon/param fit
  + Test: [t1, t2) → donmuş paramlarla skor
  + Kaydır: t0→t1, t1→t2 …
* **Nested CV (ops.)**: İç döngüde hiperparametre seçimi; dış döngü raporlar.
* **Model Gov**: Aşırı uyumu engellemek için **bilgi sızıntısı yok**, test setine bakarak param güncellenmez.

16.6 Hiperparametre Taraması

* **Grid/Random**: Strateji eşikleri (örn. *rsi\_up*, *adx\_thr*, *keltner\_mult*), ensemble *c\_min*, *s\_cap*, risk *amount\_mode* vb.
* **Bayes Opt. (ops.)**: Hedef fonksiyonu *Sharpe·λ + Expectancy·(1−λ)* ve **işlem sayısı** alt eşiği.
* **Ceza**: Çok karmaşık konfig için AIC/BIC benzeri **basit ceza** veya minimum işlem sayısı filtresi.

16.7 Paper Trading

* **MockConnector** (Parça 6) + gerçek zamanlı worker/scheduler ile **emir açmadan** simülasyon.
* **Gerçek payout akışı** varsa onu kullanır; yoksa sabit/parametrik.
* UI’da canlı akış + “Paper” rozeti; PnL sanal, kayıtlar *account\_id='paper'* ile ayrılır.

16.8 Raporlama (UI + Dosya)

* **UI**:
  + Özet kartları: win‑rate, PnL, PF, DD, Sharpe, işlem sayısı, ort. payout.
  + Grafikler: **Equity curve**, **Drawdown**, **Reliability diagram**, **Saat/Gün ısı haritası**, **TF/Ürün kırılımı**.
* **Dışa aktarım**: *backtest\_report.json*, *trades.csv*, *equity.csv*, *metrics.json*.

16.9 Python API (İskelet)

*# core/backtest.py***from** dataclasses **import** dataclass**import** pandas **as** pd**from** typing **import** Dict, List, Optional@dataclass**class** BacktestConfig: timeframe: int *# 1,5,15* payout\_mode: str = "fixed" *# fixed | series* payout\_fixed: float = 90.0 latency\_ms: int = 0 push\_is\_win: bool = True min\_trades: int = 200@dataclass**class** BacktestReport: trades: pd.DataFrame metrics: Dict[str, float]**class** Backtester: **def** \_\_init\_\_(self, indicators, providers, ensemble, risk): self.ind = indicators self.providers = providers self.ens = ensemble self.risk = risk **def** run(self, candles: pd.DataFrame, payout\_series: Optional[pd.Series], cfg: BacktestConfig) -> BacktestReport: df = candles.copy() *# 1) features* feats = compute\_features(df, self.ind) *# 2) iterate bars → sinyal & karar* recs = [] open\_until = None **for** i **in** range(1, len(df)): *# bar kapanışı* **if** open\_until **is** **not** None **and** i >= open\_until: *# sonucu hesapla* entry = recs[-1] entry\_close = df['close'].iloc[i] win = (entry\_close > entry['entry\_close']) **if** entry['dir']==1 **else** (entry\_close < entry['entry\_close']) **if** cfg.push\_is\_win **and** entry\_close == entry['entry\_close']: win = True R = entry['payout']/100.0 pnl = entry['amount']\*R **if** win **else** -entry['amount'] entry.update({"exit\_idx": i, "status": "win" **if** win **else** "lose", "pnl": pnl}) open\_until = None **continue** **if** open\_until **is** **not** None: **continue** *# yeni giriş değerlendirmesi* votes = [] **for** p **in** self.providers: v = p.evaluate(df.iloc[:i+1], feats) **if** v: votes.append(v) **if** **not** votes: **continue** comb = self.ens.combine(votes) **if** comb['dir']==0: **continue** payout = float(cfg.payout\_fixed **if** cfg.payout\_mode=="fixed" **else** payout\_series.iloc[i]) ctx = build\_backtest\_ctx(df, i, comb, payout) **if** **not** self.risk.enter\_allowed(ctx): **continue** amount = self.risk.compute\_amount(ctx) recs.append({ "entry\_idx": i, "entry\_ts": int(df['ts\_ms'].iloc[i]), "entry\_close": float(df['close'].iloc[i]), "dir": comb['dir'], "payout": payout, "amount": amount }) open\_until = i + bars\_of(cfg.timeframe) trades = pd.DataFrame(recs) metrics = compute\_metrics(trades) **return** BacktestReport(trades=trades, metrics=metrics)

16.10 Doğruluk & Kenar Durumlar

* **Warm‑up**: Yetersiz barlarda sinyal **üretilmez**.
* **Tie/push**: Platform kuralına göre ayarlanabilir; varsayılan *push\_is\_win=True*.
* **Payout eksikliği**: Seri varsa eksik değerler **taşınmaz**, bar **atlanır**.
* **Saat dilimi**: Giriş/rapor **UTC ms**, UI’da **Europe/Istanbul** gösterimi.
* **Repro**: Rastgelelik içeren kararlar *seed* ile sabitlenir.

16.11 Test Planı

* **Birim**: FTT outcome doğrulaması (CALL/PUT), push davranışı, payout sabit/seri uyumu, concurrency kuralı.
* **Entegrasyon**: Parça 7–13 ile uçtan uca backtest; belirli veri setinde beklenen metriklerle karşılaştırma.
* **Walk‑Forward**: Sentetik rejim değişimi setinde kalibrasyon/weight reset etkisi.
* **Performans**: 1M bar’lık veri -> süre ve bellek bütçesi; işlemsel sayaçlar.

16.12 UI Eşlemesi

* **Backtest sihirbazı**: Veri seç (csv), TF, payout modu, tarih aralığı, strateji presetleri, risk ayarları.
* **Rapor ekranı**: Özet metrik, equity/drawdown grafikleri, p̂ kalibrasyon grafiği, ısı haritaları, işlem listesi.
* **Karşılaştırma**: Farklı preset/param setleri yan yana (tablo + grafikli).
* **Dışa aktarım**: Tek tıkla *.json/.csv*.

16.13 Kabul Kriterleri (Parça 16)

* FTT backtest/paper motoru **tasarlandı** ve iskelet kodu verildi.
* Payout modeli (sabit/seri), concurrency ve permit kuralları **uygulandı**.
* Zengin metrik seti, walk‑forward ve hiperparam taraması yönergeleri **tanımlandı**.
* UI raporlama ve dışa aktarım **belirlendi**; test planı ve kenar durumları **kapsandı**.

Parça 17 — Telemetry & Logging (Metrikler, Log Yapısı, Uyarılar) + UI Panelleri

Amaç: MoonLight’ın **gözlemlenebilirliğini** (observability) artırmak için standart bir **metrik ve log** şeması tanımlamak; uyarı kuralları ve **UI panelleri** ile gerçek‑zamanlı takip ve hata ayıklamayı kolaylaştırmak.

17.1 Tasarım İlkeleri

* **Gizlilik‑öncelikli**: PII redaksiyonu varsayılan; hesap/kullanıcı alanları maskelenir (*u\*\*\*@mail*).
* **Yapılandırılmış JSON**: Her log tek satır JSON (JSONL). İnsan‑okunur mesaj **opsiyonel**; makine‑okunur alanlar **zorunlu**.
* **Düşük ek yük**: Metrik toplama **asenkron** ve **batch flush**; yüksek frekansta **downsample**.
* **Tutarlılık**: Tüm bileşenlerde aynı alan adları; *order\_id*, *client\_req\_id*, *trace\_id* ile korelasyon.

17.2 Metrik Taksonomisi

* **Sayaç (counter)**: Birikimli artar. Ör: *orders\_total*, *orders\_win\_total*, *errors\_total*.
* **Gösterge (gauge)**: Anlık değer. Ör: *pnl\_day*, *latency\_ms\_last*, *workers\_running*.
* **Histogram/Özet**: p50/p90/p99 ve ortalama. Ör: *latency\_ms*, *retry\_count*, *equity\_curve*.

17.2.1 Çekirdek Metrikler (örnek anahtarlar)

* **Emir/Yürütme**
  + *orders\_total{scope}*
  + *orders\_win\_total{scope}* / *orders\_lose\_total{scope}*
  + *orders\_skipped\_total{reason}* (permit/confidence/concurrency/risk\_guard/overrun)
  + *latency\_ms\_{p50,p90,p99}{scope}*
* **Performans**
  + *win\_rate\_{tf}* (1/5/15)
  + *payout\_avg{product}*
  + *expectancy*
  + *pnl\_day{account}* / *pnl\_week* / *pnl\_total*
  + *drawdown\_max* (rolling)
* **Risk/Guardrails**
  + *consec\_losses{scope}*
  + *daily\_loss{account}*
  + *circuit\_breaker\_trips\_total*
  + *cooldown\_active{account}* (gauge)
* **Bağlantı/Veri**
  + *connector\_errors\_total{class}* (AUTH/THROTTLE/NETWORK/MAINT/FORMAT)
  + *rps\_account{account}*
  + *ws\_heartbeat\_lag\_ms*
  + *rate\_limit\_hits\_total*
* **Worker/Scheduler**
  + *workers\_running* / *workers\_paused*
  + *tick\_overrun\_total*
  + *queue\_depth{chan}* (metrics/logs/trades)

*scope* değeri: *global*, *acc:<id>*, *prod:<symbol>*, *tf:<1|5|15>*.

17.3 Log Yapısı (JSONL)

Her satır örnek:

{ "ts\_ms": 1728291660123, "lvl": "INFO", "chan": "trade|system|security|connector|worker|scheduler|api", "event": "PLACE\_OK", "account": "acc1", "product": "EURUSD", "tf": 1, "order\_id": "SIM-1728291600123", "client\_req\_id": "acc1:EURUSD:1:1728291600000:a1b2c3d4", "trace\_id": "b7e1c...", "msg": "placed", "fields": {"amount":1.0, "payout":90.0, "latency\_ms":120}}

17.3.1 Kanallar

* *trade*: Emir akışı, yerleştirme/sonuç, idempotensi, payout, confidence.
* *system*: Başlatma, konfig değişimi, hot‑reload, shutdown.
* *security*: Giriş/çıkış, OTP istenmesi, kilit/kilit açma; PII **maskeli**.
* *connector*: HTTP/WS çağrıları, retry, backoff, rate‑limit vuruşları (toplu/sampled).
* *worker*/*scheduler*: TF tetikleri, overrun/drop, pause/backoff.
* *api*: UI çağrıları, doğrulama hataları, izin sorunları.

17.3.2 Maskeleme & Redaksiyon

* Kullanıcı adı, e‑posta, telefon, token/çerez/OTP **loglanmaz**; gerektiğinde **hash** (*sha256:...*) veya mask.
* Hata dump’larında istisna mesajı **kısaltılır**; stack trace **opsiyonel** ve yalnız *DEBUG* seviyesinde.

17.3.3 Rotasyon & Saklama

* *logs/* altında kanal bazlı dosyalar: *trade.log*, *system.log*, *security.log*, *connector.log*.
* **Boyut/süre bazlı** rotasyon (örn. 10MB veya 3 gün). Maks 7 dosya (UI’da ayarlanabilir).
* Arşivler **sıkıştırılır** (*.gz*) ve 30 gün sonra otomatik silinebilir (opsiyonel).

17.4 Toplama & Flush Politikası

* Metrikler **bellekte** toplanır ve **5 sn** aralıklarla toplu *metrics* tablosuna yazılır (Parça 5).
* Yüksek frekanslı histogramlar **downsample** edilir (örn. yalnız p50/p90/p99 tutulur).
* Tekrarlayan hatalar **örneklenir** (örn. ilk 1, dakikada en fazla 5).
* WS ile UI yayını **debounce** ve **rate‑limit** (10 msg/sn/kanal).

17.5 Uyarı Kuralları (Rules)

* **Risk**
  + *daily\_loss ≤ −max\_daily\_loss* → **ERROR** *CIRCUIT\_BREAKER\_TRIP* (otomatik durdur).
  + *consec\_losses ≥ Lmax−1* → **WARN** *LOSS\_STREAK\_NEAR*.
* **Performans**
  + *win\_rate\_{tf} < threshold\_warn* (örn. 0.55) → **WARN**.
  + *expectancy < 0* 5 bar üst üste → **WARN**.
* **Bağlantı**
  + *ws\_heartbeat\_lag\_ms > 10\_000* → **WARN**.
  + *rate\_limit\_hits\_total* artışı belirli eşiği aşarsa → **INFO/WARN**.
* **Latency**
  + *latency\_ms\_p90 > abort\_ms·0.7* → **WARN**; *> abort\_ms* → **ERROR**.
* **Payout/Permit**
  + Uzun süre permit penceresi dışında kalırsa → **INFO** (strateji pasifleştirme önerisi).

17.5.1 Bildirim Kanalları

* UI ***alerts*** WS kanalı (Parça 15).
* Opsiyonel **Windows Toast** bildirimi (kullanıcı izniyle).
* E‑posta/Discord/Telegram **YOK** (varsayılan; güvenlik/telemetry politikasına göre sonradan opsiyonel).

17.6 UI Panelleri

* **Dashboard**
  + Kartlar: *win\_rate\_1/5/15*, *pnl\_day*, *expectancy*, *drawdown*, *latency\_p90*, *orders\_total*, *skips\_by\_reason*.
  + Grafikler: “Equity curve”, “Drawdown”, “Latency histogram”, “Reliability diagram (p̂ vs gerçek)”.
* **Hesaplar**
  + Her hesap için: bakiye, günlük PnL, ardışık kayıp, cooldown, WS lag, connector hata sayıları.
  + “Panic/kill‑switch” butonu.
* **Ürün/TF İzleme**
  + Ürün+TF kırılımı: permit penceresi, ort. payout, işlem sıklığı, win‑rate, skip nedenleri, son 20 işlem listesi.
  + Strateji katkı tablosu (w\_i, ort. skor katkısı, latency).
* **Log Görüntüleyici**
  + Kanal/seviye/alan filtreleri; canlı akış/stop; arama; CSV/JSON dışa aktarım.
* **Uyarı Merkezi**
  + Aktif uyarılar listesi; önem seviyesi renkleri (kırmızı/amber/mavi/yeşil).
  + Kuralları düzenleme (eşikler) ve geçici erteleme (snooze).

17.7 Örnek SQL Sorguları

* **Win‑rate (son 200)**

**SELECT** AVG(**CASE** **WHEN** status='win' **THEN** 1.0 **WHEN** status='lose' **THEN** 0.0 **END**)**FROM** ( **SELECT** r.status **FROM** orders o **JOIN** results r **ON** r.order\_id=o.**id** **WHERE** o.product='EURUSD' **AND** o.timeframe=1 **ORDER** **BY** r.ts\_close\_ms **DESC** **LIMIT** 200);

* **Skip nedenleri (son 1 gün)**

**SELECT** reason, COUNT(\*) **FROM** trade\_skips**WHERE** ts\_ms > strftime('%s','now')\*1000 - 86\_400\_000**GROUP** **BY** reason **ORDER** **BY** COUNT(\*) **DESC**;

Not: *trade\_skips* tablo/kanalı opsiyoneldir; skip olayları metrics/loglardan da türetilebilir.

17.8 Sağlık (Health) & Kalp Atışı

* **Liveness**: Core servis thread/loop çalışıyor mu?
* **Readiness**: DB ve connector hazır mı?
* **Heartbeat**: Connector *heartbeat()*; WS lag ölçümü ve uyarı üretimi.

17.9 Test Planı

* **Metrik doğrulama**: Emir akışında sayaçların tutarlı artması, histogramların p50/p90/p99 üretmesi.
* **Log redaksiyonu**: “gizli alan” testleri; *DEBUG*’de bile PII yok.
* **Uyarı kuralları**: Eşik üstü/salt‑okunur mod testleri; circuit‑breaker’ın tetiklenmesi.
* **UI performansı**: 10 msg/sn WS yayınında çizim/kare hızının stabil kalması.
* **Rotasyon**: Dosya boyutu ve gün bazlı rotasyonun beklenen sayıda dosya tutması.

17.10 Kabul Kriterleri (Parça 17)

* Metrik taksonomisi, ana anahtarlar ve downsample/flush politikası **tanımlandı**.
* JSONL log şeması, kanallar, redaksiyon ve rotasyon **net**.
* Uyarı kuralları & bildirim kanalları, UI panelleri ve örnek SQL’ler **eklendi**.
* Sağlık/heartbeat ve test planı **kapsandı**.

Parça 18 — UI (Flutter) Tema, Bileşen Kütüphanesi & Ekran Tasarımları (Windows 10/11)

Amaç: Masaüstü (Flutter) arayüzünün görünüş, etkileşim ve bilgi mimarisini netleştirmek; çoklu hesap/ürün/TF yönetimi, telemetri/log ve ayarlar için **hızlı**, **erişilebilir** ve **kullanıcı dostu** bir UI tasarlamak. Renk paleti: **mor, mavi, yeşil, kırmızı, siyah, beyaz**.

18.1 Teknoloji Seçimleri

* **Flutter 3.x (Desktop)** + **Material 3** (M3) bileşenleri.
* **Durum yönetimi**: *flutter\_riverpod* (veya *riverpod*), immutable modeller için *freezed* + *json\_serializable*.
* **Navigasyon**: *go\_router* (deep‑link, URL benzeri yollar, state‑restore).
* **Grafikler**: *fl\_chart* (equity, drawdown, histogram), custom painter ile ince ayar.
* **WebSocket**: *web\_socket\_channel* (Parça 15 */ws*).
* **HTTP istemci**: *dio* (retry/interceptor, timeout), REST sözleşmeleri için generated client (ops.).
* **Yerel depolama**: Ayarlar/UI tercihleri için *shared\_preferences* (veya *hive*), büyük veri **yok** (asıl veri core’da).
* **Pencereler**: *window\_manager* veya *bitsdojo\_window* (başlık çubuğu/kenarlık, minimize/always‑on‑top).
* **Tablo/virtualization**: Büyük listeler için *PaginatedDataTable2* veya *syncfusion\_flutter\_datagrid*.
* **İkonlar**: *material\_symbols* (rounded) + özel durum ikonları.

18.2 Renk Teması & Tasarım Token’ları

* **Ana (brand) mor**: *#6D28D9*
* **Vurgu mavi**: *#2563EB*
* **Başarı yeşil**: *#10B981*
* **Uyarı kırmızı**: *#EF4444*
* **Nötr siyah**: *#0B0F14* (arka plan – dark)
* **Nötr beyaz**: *#FFFFFF* (arka plan – light)

**Durum renk haritası**  
- **Win**: yeşil  
- **Lose**: kırmızı  
- **Paused**: sarımsı amber *#F59E0B*  
- **Running**: mavi  
- **Risk near**: turuncu *#F97316*  
- **Circuit breaker**: koyu kırmızı + uyarı simgesi

**Tipografi**  
- Başlık: *TitleLarge* kalın, metin: *BodyMedium*; sayılar *tabular‑nums* (eşit aralıklı) – okunabilirlik için.

**Yoğunluk & Layout**  
- Desktop grid: Sol **NavigationRail** (dar/geniş mod), üst **AppBar** (hesap anahtarı), içerik **ResponsiveGrid** (1–3 sütun).

18.3 Proje Yapısı (UI)

ui/ main.dart app/ app.dart # MaterialApp, theme, router theme.dart # renk paleti, koyu/açık şema router.dart # go\_router rotaları core/ api\_client.dart # REST/WS istemci sarmalayıcı models.dart # DTO'lar (freezed) store.dart # Riverpod provider'ları (state) format.dart # para, tarih, yüzde yardımcıları charts.dart # standart graf bileşenleri features/ dashboard/ accounts/ products/ strategies/ backtest/ orders/ logs/ settings/ alerts/ widgets/ cards.dart tables.dart forms.dart badges.dart

18.4 Ekran Akışları & Bileşenler

18.4.1 **Giriş & Hesap Yöneticisi**

* **Ekran**: *AccountsPage*
  + Sol: hesap listesi (maks. 4), durum rozeti (Connected/Disconnected/OTP Required).
  + Sağ: seçili hesabın ayrıntısı; “Giriş yap” butonu, OTP alanı (gerekirse), **profil yolu** göstergesi (salt‑okunur).
  + **Toplu giriş**: “Tüm hesaplarla bağlan”.

18.4.2 **Dashboard**

* **Kartlar**: *Win‑rate 1/5/15*, *PnL (Gün)*, *Expectancy*, *Drawdown*, *Latency p90*, *Orders (Toplam)*, *Skips by Reason*.
* **Grafikler**: Equity curve (çizgi), Drawdown (alan), Reliability diagram (bar), Latency histogram.
* **Canlı besleme**: WS *metrics* + *trade\_updates*.
* **Hızlı filtre**: Hesap/Ürün/TF açılırları; tarih aralığı.

18.4.3 **Ürün & TF Yönetimi**

* **Ekran**: *ProductsPage*
  + Sol: Ürün ağacı → *EURUSD*, *BTCUSD*, …; her ürün altında *1/5/15* node’ları.
  + Sağ paneller:
    1. **Permit & Eşik**: *permit\_min/max* aralığı (çift sürgü), *win\_threshold* (sürgü), canlı *p̂*/payout gösterimi.
    2. **Risk**: *amount\_mode*, *fixed\_a*, *frac*, *kelly\_scale*, *a\_min/a\_cap*.
    3. **Stratejiler**: Aktif/pasif anahtarları, preset seçimi, param override (form).
    4. **Worker**: RUNNING/PAUSED, son kapanış zamanı, hata sayacı; **Başlat/Durdur**.

18.4.4 **Strateji Kataloğu**

* **Liste**: ID, ad, grup, warm‑up, kısa açıklama.
* **Detay**: Parametre formu (spinner/slider), **varsayılanı geri yükle**.
* **Performans paneli**: Son N işlemlerde win%, katkı, gecikme; ağırlık *w\_i* (Parça 9 telemetrisi).

18.4.5 **İşlemler (Orders)**

* **Tablo**: Zaman, hesap, ürün, TF, yön, tutar, payout, p̂, durum, PnL, latency.
* **Filtreler**: Hesap/Ürün/TF/Status/tarih; **CSV dışa aktar**.
* **Ayrıntı çekmecesi**: Seçili işlemin indikatör snapshot’ı, provider oyları & skorları (şeffaflık).

18.4.6 **Backtest / Paper**

* **Sihirbaz**: Veri dosyası, TF, payout modu, preset/param seti, risk seçenekleri.
* **Rapor**: Metrik kartları + grafikleri, karşılaştırma tablosu, **profil kaydet**.

18.4.7 **Log Görüntüleyici**

* **Canlı akış**: Kanal/seviye filtresi, arama, *pause/resume*, *tail*.
* **Satır tıklama**: JSON ayrıntı modal + **kopyala**.
* **Export**: Seçili aralık JSON/CSV.

18.4.8 **Uyarı Merkezi**

* **Liste**: Zaman, seviye, kod, mesaj, kapsam (acc/prod/tf).
* **Eylemler**: Eşik ayarına git, **snooze (5/15/60 dk)**, “hepsini okundu işaretle”.

18.4.9 **Ayarlar**

* **Genel**: Tema (koyu/açık/otomatik), renk presetleri (mor/mavi/yeşil/kırmızı), sayı biçimleri, dil (TR/EN), Europe/Istanbul yerel saat seçimi.
* **Motor**: *tick\_ms*, *grace\_ms*, *jitter\_ms*, *entry\_cutoff\_s*, lookback.
* **Risk**: global guardrail default’ları.
* **API**: Loopback portları, API anahtarı (ops.).
* **Veri**: Dışa aktarım, yedekleme, log rotasyon sınırları.

18.5 Kullanılabilirlik & Erişilebilirlik

* **Klavye kısayolları**:
  + *Ctrl+L* Loglar, *Ctrl+D* Dashboard, *Ctrl+P* Products, *Ctrl+B* Backtest.
  + *Ctrl+F* Filtre/Arama; *F5* yenile; *Space* pause/resume canlı akış.
* **Erişilebilirlik**: Yüksek kontrast teması, odak halkası (focus ring), ekran okuyucu etiketleri (Semantics).
* **Tablo performansı**: Sanallaştırma, sayfalama; 10k+ satır akıcı.

18.6 Durum (State) Modeli

* **GlobalAppState**: tema, dil, WS bağlantı durumu, bildirim rozet sayısı.
* **AccountsState**: hesap listesi, bağlantı durumu, bakiye, günlük PnL, cooldown.
* **ProductsState**: ürün/TF ağaç, aktif worker’lar, permit/eşik/risk paramları (UI kopyası).
* **MetricsState**: son metrikler (gauge/sayaç), grafik veri arabellekleri (downsample).
* **OrdersState**: sayfalı sorgu ve canlı akış kuyruğu.
* **LogsState**: aktif filtreler, akış durumu, tail işareti.

18.7 Örnek Tema Kodu (kısaltılmış)

*// ui/app/theme.dart***import** 'package:flutter/material.dart';const brandPurple = Color(0xFF6D28D9);const accentBlue = Color(0xFF2563EB);const successGreen= Color(0xFF10B981);const dangerRed = Color(0xFFEF4444);ThemeData buildDarkTheme() { final base = ThemeData.dark(useMaterial3: true); **return** base.copyWith( colorScheme: base.colorScheme.copyWith( primary: brandPurple, secondary: accentBlue, error: dangerRed, surface: const Color(0xFF0B0F14), onSurface: Colors.white, ), appBarTheme: const AppBarTheme(centerTitle: false, elevation: 0), inputDecorationTheme: const InputDecorationTheme(border: OutlineInputBorder()), );}

18.8 Performans & Stabilite

* **WS tüketimi**: Arabellekli kuyruk + *debounce(100ms)*; graf verileri **downsample** (örn. her 10. nokta).
* **İzole işler**: Büyük JSON parse ve rapor hesaplamaları **Isolate** üzerinde.
* **Hata dayanıklılığı**: WS yeniden bağlanma (exponential backoff), REST retry (limited). Hatalarda kullanıcıya **masum** mesaj + detay butonu.

18.9 Test Planı (UI)

* **Widget testleri**: Form doğrulama (permit min/max, threshold aralığı), tablo filtreleri, kısayollar.
* **Entegrasyon**: Sahte API (mock server) ile */status*, */workers*, */orders* akışları.
* **Erişilebilirlik**: Semantics ağaç doğrulaması; renk kontrast ölçümü (AA).
* **Performans**: 10 msg/sn WS yayını; 10k satırlı tablo kaydırma FPS ≥ 55.

18.10 Kabul Kriterleri (Parça 18)

* Tema/renkler (mor, mavi, yeşil, kırmızı, siyah, beyaz) uygulandı; koyu/açık mod hazır.
* Ekran akışları (Hesap, Dashboard, Ürün/TF, Strateji, Orders, Backtest, Log, Alerts, Ayarlar) tanımlandı.
* Bileşen kütüphanesi ve performans/erişilebilirlik kuralları belirlendi.
* Kod iskeleti ve durum modeli örnekleri eklendi.

Parça 19 — Windows Kurulum & Servis: Paketleme (MSIX/ZIP), Auto‑Update, Güvenlik

Amaç: MoonLight’ın **Windows 10/11**’de kolay kurulumunu, arka planda **servis** olarak çalışmasını, güvenli güncellemelerini ve çoklu hesap kullanımına uygun **dosya/dizin** düzenini tanımlamak. UI (Flutter) ve Core (Python/FastAPI + async engine) birlikte dağıtılır.

19.1 Dağıtım Matrisi

| Bileşen | Biçim | Nerede Çalışır | Not |
| --- | --- | --- | --- |
| **Core** (moonlightd) | EXE (tek dosya) | **Windows Servis** | PyInstaller veya PyOxidizer; Event Log & Service API entegrasyonu. |
| **UI** (Flutter) | MSIX (önerilen) | Kullanıcı oturumunda | App Installer ile auto‑update. Portable ZIP opsiyonu. |
| **Ara yüz** | REST/WS loopback | 127.0.0.1 | TLS opsiyonel, API‑Key opsiyonel (Parça 15). |

19.2 Dizin Yapısı

* **Program Files**: *C:\Program Files\MoonLight\*
  + *moonlightd.exe* (core), *service.yaml* (servis ayarı), *openapi.json*
* **ProgramData** (ortak veri): *C:\ProgramData\MoonLight\*
  + *db\moonlight.sqlite*, *profiles\acc1\*, *logs\\*.log*, *backups\*
* **LocalAppData (UI)**: *%LocalAppData%\MoonLightUI\*
  + *config\_ui.json*, cache, kullanıcı tercihleri

Not: DB/loglar **ProgramData**’da; UI kaldırıldığında veriler silinmez. Yedekler şifreli (isteğe bağlı).

19.3 Core Paketleme (EXE)

**Seçenek A — PyInstaller**  
- Komut: *pyinstaller --onefile --noconsole moonlightd.spec*  
- Spec: *hiddenimports* (uvloop/aiosqlite), *datas* (DDL/şema), *--add-data* ile şablonlar.  
- Artılar: Kurulumu kolay, yaygın.  
- Eksiler: Boyut büyük, anti‑virus heurstikleri—**kod imzalama** şart.

**Seçenek B — PyOxidizer**  
- Tek statik EXE; başlatma süresi daha kısa.  
- Öğrenme eğrisi daha dik; fakat servis için **idealde**.

19.4 Windows Servis Kurulumu

* Servis adı: ***MoonLight Core***
* Başlangıç: *Automatic (Delayed Start)*
* Hesap: ***LocalService*** (önerilir; *LocalSystem* yerine düşük yetki)
* Kurtarma: 1., 2., 3. hatada **Restart the Service** (1dk, 5dk, 5dk gecikme)

**PowerShell örneği (yerel EXE ile)**

$exe = "C:\Program Files\MoonLight\moonlightd.exe"New-Service -Name "MoonLight Core" -BinaryPathName "`"$exe`" --service" -DisplayName "MoonLight Core" -StartupType Automaticsc.exe failure "MoonLight Core" reset= 86400 actions= restart/60000/restart/300000/restart/300000sc.exe sidtype "MoonLight Core" unrestricted

Alternatif: **NSSM** ile servis sarma: *nssm install "MoonLight Core" C:\...\moonlightd.exe --service*

**Servis Parametreleri**  
- *--service* : Event Log kayıtlarını açar, graceful shutdown dinler.  
- *--db C:\ProgramData\MoonLight\db\moonlight.sqlite*  
- *--http 127.0.0.1:8750 --ws 127.0.0.1:8751*  
- *--log-dir C:\ProgramData\MoonLight\logs*  
- *--profiles C:\ProgramData\MoonLight\profiles*

19.5 UI Paketleme (MSIX + Portable)

**MSIX (önerilen)**  
- *flutter build windows* → *msix* ile paketleme.  
- **App Installer** (.appinstaller) dosyasıyla güncelleme akışı (bkz. 19.6).  
- Gereksinim: **Kod imzalama** sertifikası (kurumsal veya self‑signed → yerel güven).

**Portable ZIP (opsiyon)**  
- Zip’e otomatik **sürüm klasörü**: *MoonLightUI-1.2.3-win64.zip*.  
- *MoonLightUI.exe* ilk açılışta *MoonLight Core* servisinin çalışma durumunu kontrol eder; çalışmıyorsa başlatma talimatı gösterir.

19.6 Auto‑Update Stratejisi

**UI (MSIX) — App Installer**  
- *MoonLightUI.appinstaller* örneği:

<**AppInstaller** Uri="https://updates.moonlight.app/MoonLightUI.appinstaller" Version="1.2.3.0" xmlns="http://schemas.microsoft.com/appx/appinstaller/2017/2"> <**MainPackage** Name="com.moonlight.ui" Version="1.2.3.0" Publisher="CN=MoonLight" Uri="https://updates.moonlight.app/MoonLightUI\_1.2.3.0\_x64.msix" /> <**UpdateSettings**> <**OnLaunch** HoursBetweenUpdateChecks="12" ShowPrompt="true" UpdateBlocksActivation="false"/> </**UpdateSettings**></**AppInstaller**>

* Avantaj: Windows App Installer otomatik kontrol eder; **kullanıcı onayı** ile günceller.

**Core (Servis) — Self‑Update (güvenli)**  
- **Update feed** (JSON): sürüm, SHA‑256/Ed25519 imza, indirme URL’leri.  
- Akış: Servis **duraksız** güncelleme için *moonlightd\_new.exe* indirir → imza doğrula → **maintenance window**’da servis kendini **swap** eder → başarısızsa otomatik **rollback**.  
- Güncelleme tetik: UI’dan “Güncellemeyi kontrol et” veya **zamanlanmış** (günde 1).  
- Not: Güncelleme dosyalarının **kod imzalı** olması **şart**.

19.7 Güvenlik & Yetkiler

* **Kod İmzalama**: UI MSIX ve core EXE **imzalanır** → SmartScreen uyarıları azalır.
* **Loopback Zorunlu**: REST/WS sadece *127.0.0.1* bağlar; **uzaktan erişim kapalı**.
* **Windows Güvenlik Duvarı**: Kural gerekmeyebilir (loopback); opsiyonel olarak yalnızca yerel profil için izin.
* **Kimlik Bilgisi Kasası**: Parolalar **saklanmaz**. Mümkünse **token**/**session** kullanılır; gerekirse **Windows Credential Manager** + **DPAPI** ile şifreleme (*keyring* modülü).
* **Dosya İzinleri**: *C:\ProgramData\MoonLight\* yalnız **Administrators** ve **LocalService** erişebilir.
* **Log Redaksiyonu**: PII maskesi (Parça 17); *security.log* ayrı ve kısıtlı ACL.
* **Anti‑bot/uyum**: Yalnızca **resmî/izinli API** kullanımı; koruma atlatma, scraping/otomasyon **yok**.

19.8 Çoklu Hesap (4 Hesap) Kurulum Notları

* Her hesap için *profiles\accX\* altında **izole cookie/token** saklanır.
* UI “Toplu Giriş” ile tüm profiller için login çağrısı yapar; servis **eşzamanlı** rate‑limit uygular.
* Log/DB’de tüm kayıtlar *account\_id* ile etiketlenir; filtrelenebilir.

19.9 İlk Çalıştırma Sihirbazı

1. Lisans ve gizlilik sözleşmesi.
2. Depo/Log dizin onayı (varsayılanlar).
3. API portları (loopback) ve opsiyonel API‑Key üretimi.
4. Profil klasörleri ve izin ayarı testleri.
5. Servis **başlat** ve durum testi (*/status*).

19.10 Kaldırma & Temizlik

* UI (MSIX) “Uygulamalar ve Özellikler” üzerinden kaldırılır; **ProgramData** *silinmez*.
* Core servis: *Stop-Service "MoonLight Core"; sc.exe delete "MoonLight Core"*
* Veri tamamını silmek için: *C:\ProgramData\MoonLight\* → **manuel** onaylı temizlik aracı.

19.11 Sürümleme & Yapı Pipeline’ı

* **SemVer**: *MAJOR.MINOR.PATCH* (core ve UI ayrı izlenir).
* CI’de:
  + **Core**: Test → PyInstaller/PyOxidizer build → imzalama → *moonlightd-1.4.0-win64.exe*.
  + **UI**: *flutter build windows* → *msix* → imzalama → *.msix* + *.appinstaller*.
  + Update feed güncellemesi (imza/sha256).
* Release notları ve **geri dönüş** yönergesi (rollback) yayınlanır.

19.12 Kabul Kriterleri (Parça 19)

* Core servis **Windows Service** olarak kurulabilir, otomatik başlar, çökmede kurtarma politikaları aktiftir.
* UI MSIX paketlenir ve **App Installer** ile auto‑update alır; Portable ZIP alternatifi çalışır.
* Güncelleme kanalı **imza doğrulamalı** ve rollback’li.
* Dizin/izinler güvenli; kimlik bilgileri Credential Manager/DPAPI ile korunur.
* Çoklu hesap profilleri izole; log/DB etiketleme doğru.
* Kaldırma/temizlik ve sürümleme/pipeline adımları tanımlandı.

Parça 20 — Test & QA Planı (E2E, Hata Fırtınası, Performans/Dayanıklılık) + Release Süreci

Amaç: MoonLight’ın uçtan uca **doğruluğunu**, **kararlılığını** ve **operasyonel güvenilirliğini** üretim öncesi kanıtlamak; ölçülebilir **başarı kriterleri** ile sürüm yayın/geri alma (rollback) süreçlerini tanımlamak.

20.1 Test Kapsamı

* **Birim (Unit)**: İndikatörler (Parça 7–8), ensemble (Parça 9), risk (Parça 10), executor FSM (Parça 11), worker/scheduler (Parça 12), stratejiler (Parça 13), API (Parça 15).
* **Bütünleşik (Integration)**: Core ↔ Connector (MockConnector + gerçek sandbox), Core ↔ DB, Core ↔ UI (REST/WS), Paper/Backtest (Parça 16).
* **E2E (Uçtan Uca)**: Çoklu hesap + çoklu ürün/TF akışı; giriş → yürütme → sonuç → telemetry → UI.
* **Dayanıklılık/Kaos**: Ağ kesintisi, 429/5xx fırtınası, disk dolu, DB kilidi, saat kayması, servis yeniden başlatma.
* **Performans**: RPS sınırları, latans dağılımı, bellek/CPU bütçeleri.
* **Güvenlik/Uyum**: PII redaksiyonu, loopback kısıtı, parola/anahtar yönetimi, log erişim yetkileri.

20.2 E2E Senaryolar (Örnekler)

| ID | Senaryo | Adımlar | Beklenen |
| --- | --- | --- | --- |
| E2E‑01 | **Tek hesap / EURUSD / TF=1** | Login → Worker başlat → Permit 93–89 → p̂ eşiği %70 → 10 işlem | Tek açık işlem kuralı, sonuçlar kaydedilir, skip nedenleri doğru. |
| E2E‑02 | **3 ürün / 3 TF paralel** | EURUSD(1), GBPUSD(5), XAUUSD(15) | Her TF kendi kuyruğunu işletir; çakışma yok; telemetry WS akıyor. |
| E2E‑03 | **Permit dışı payout** | Payout 85’e düşer | Giriş yapılmaz; *skipped: permit* metriği artar; uyarı gelir. |
| E2E‑04 | **Loss streak guardrail** | Ardışık 5 kayıp | Circuit‑breaker tetikler; yeni giriş yok; UI’da kırmızı durum. |
| E2E‑05 | **Paper→Live geçişi** | Paper açıkken başarı eşiğine zorla; ardından Live | Mod değişimi loglanır; hesap etiketleri ayırt edilir; risk sınırları sürer. |
| E2E‑06 | **Restart & Reconcile** | İşlem açıkken core restart | Boot’ta reconcile sonuçlandırır; çift kayıt yok; idempotensi korunur. |
| E2E‑07 | **Param hot‑reload** | *win\_threshold* ve *amount\_mode* güncelle | Sonraki girişlerde yeni değerler; geçmiş etkilenmez; API *applied/restarted* alanlarını döner. |
| E2E‑08 | **OTP akışı (ops.)** | 2FA isteyen hesaba giriş | UI OTP alanı, hatalı/timeout durumları; *security.log* maske. |

20.3 Birim Test Matrisi (özet)

* **İndikatörler**: RSI/MACD/ATR/BB/Stoch/DMI/Ichimoku/Supertrend—referans veri ile *rtol ≤ 1e‑6* doğrulama; warm‑up/NaN yayılımı.
* **Ensemble**: Skor normalizasyonu, ağırlık tavanı *s\_cap*, Platt kalibrasyon Brier/LogLoss düşüşü.
* **Risk**: Kelly‑lite hesapları, permit/threshold kontrolü, günlük kayıp/ardışık kayıp limitleri.
* **Executor FSM**: Tüm geçişler; retry limitleri; idempotent *client\_req\_id* çakışma testi.
* **Scheduler/Worker**: TF hizalama, overrun/drop politikası.
* **API**: Hata zarfı; *GET /status* 200/401; */orders* filtreleri; WS ping/pong.

20.4 Kaos & Hata Fırtınası Testleri

* **Ağ**: Paket kaybı %10–30, gecikme 200–1000ms; *place/confirm* timeouts; beklenen retry ve geri kazanım.
* **Rate‑limit**: 429 fırtınası; backoff uygulanır; *rate\_limit\_hits\_total* artar; başarıyla iyileşme.
* **Disk**: *logs/* yazılamaz → graceful degrade; *db* kilitli → kısa retry; uzun sürerse *PAUSED*.
* **Saat**: Sistem saati ±60s kayar → TF slot/faz kontrolü; toleranslı hizalama.
* **Servis Restart**: *Stop-Service* → *Start-Service*; yetim emirlerin reconcile edilmesi.

20.5 Performans Hedefleri (Referans Donanım)

* **Donanım**: 4C/8T CPU, 16 GB RAM, SSD.
* **Workers**: 4 hesap × (10 ürün × 3 TF) → **120 worker** (yük testi).
* **İndikatör hesap**: TF=1 bar kapanışında **< 30 ms** (ortalama); p90 **< 60 ms**.
* **Ensemble + risk + preflight**: **< 5 ms** (ortalama).
* **place() round‑trip**: p50 **< 200 ms**, p90 **< 400 ms** (connector’a bağlı).
* **Bellek**: Core **< 800 MB**, UI **< 400 MB**.
* **CPU**: Ortalama **< %40**, pik **< %80** (120 worker testinde).

20.6 Test Verisi & Determinizm

* **Sentetik veri setleri**: Trend, range, yüksek volatilite, düşük hacim rejimleri.
* **Gerçek veri**: Tarihsel mum + payout serileri (kamuya açık kaynaklardan veya kullanıcı sağlıyorsa).
* **Seed’ler**: Rastgelelik içeren her test için *SEED=42*; raporlarda seed kaydı.
* **Altın dosyalar**: *golden/* klasöründe referans metrik çıkışları; CI “sapma” toleransı yüzde bazlı.

20.7 Kalibrasyon & Güven Değerlendirmesi

* **Reliability diagram** ve **ECE** izlenir; ECE > 0.05 ise yeniden kalibrasyon.
* **Brier skor** hedefi: < 0.22 (başlangıç); daha iyiye gidildikçe eşik sıkılaştırılır.

20.8 Güvenlik Testleri

* **Loopback** dışında bağlanma girişimi → reddedilir.
* **API‑Key** açıksa yanlış anahtar → 401; log’larda PII yok.
* **Credential Manager** erişim hatası → giriş reddi; güvenli degrade.
* **Log sızıntısı**: *security.log*’da maske/anonimlik doğrulaması.

20.9 UI QA Kontrol Listesi

* **Form doğrulama**: permit min<max, win\_threshold %50–%95, amount sınırları.
* **Kısayollar**: Çalışıyor ve çakışmıyor.
* **Tema**: Koyu/açık; renk körlüğü dostu kontrast.
* **WS yeniden bağlanma**: Ağ kesintisinde durum geri kazanımı.
* **Büyük tablolar**: 10k satırda akıcı kaydırma.

20.10 Yayın (Release) Süreci

1. **Branching**: *main* (stabil), *develop* (aktif), *feature/\**.
2. **CI**: Birim + entegrasyon + UI widget testleri; codecov raporu.
3. **Pre‑release**: *rc* etiketi; **kanarya** yayın (kısıtlı kullanıcı/cihaz).
4. **Sürüm Onayı Kapıları** (hedef eşikler):
   * Test kapsama ≥ %80 (core), ≥ %60 (UI).
   * E2E senaryoların **%100**’ü geçer.
   * Performans hedefleri karşılanır.
   * Brier ≤ 0.22, ECE ≤ 0.05.
   * Güvenlik kontrolleri geçer.
5. **Dağıtım**: UI **MSIX** + *.appinstaller*, core **EXE** ve update feed.
6. **Sonrası**: İlk 48 saatte telemetri yakın izleme (payout/latency/skip/CR).
7. **Rollback**: Kritik hata → UI: App Installer ile önceki *MainPackage*’a dönüş; Core: self‑update rollback; *release‑notes* yönlendirmesi.

20.11 Dokümantasyon & Eğitim

* **Kullanıcı kılavuzu**: Kurulum, ilk ayar sihirbazı, paper vs live farkları, risk uyarıları.
* **Operasyon kılavuzu**: Log/telemetry yorumlama, uyarı kodları, backtest/paper pratikleri.
* **Geliştirici rehberi**: Plugin yazımı (strateji sağlayıcıları), API şeması, veri sözleşmeleri.

20.12 Kabul Kriterleri (Parça 20)

* E2E senaryolar, birim/entegrasyon/kaos/perf planları **eksiksiz**.
* Ölçülebilir hedefler (latency, bellek, Brier/ECE, win‑rate eşikleri) **tanımlandı**.
* Release süreci, kanarya ve rollback adımları **net**.
* QA kontrol listeleri ve dokümantasyon gereksinimleri **yazıldı**.

Parça 21 — Konfigürasyon Profilleri, Override Zinciri, Dışa Aktarım & Geri Alma

Amaç: 4 hesaplı Windows kurulumunda **esnek ama kontrol edilebilir** bir konfigürasyon/profil sistemi kurmak; hiyerarşik override ile (global → hesap → ürün → TF → strateji) davranışı belirlemek; hot‑reload, denetim izi (audit), versiyonlama ve geri alma (rollback) sağlamak.

21.1 Neden Profil Sistemi?

* **Çoklu hesap**: Her hesap farklı izin penceresi, lot ve strateji seti kullanabilir.
* **Senaryo geçişleri**: “Gündüz/Gece”, “Yüksek/Normal volatilite”, “Sadece 1 dak.” gibi tek tıkla profil değişimi.
* **Risk guardrails**: Demo/gerçek ve farklı risk iştahları için kalıp profiller.
* **Taşınabilirlik**: Profili dışa aktarıp yedekleyebilme ve başka kurulumda içe aktarma.

21.2 Kavramlar ve Türler

* **Global Varsayılan**: Tüm sistem için başlangıç değerleri (Parça 4 – *config.json*).
* **Hesap Profili**: *profiles/<acc>/profile.yaml* — yalnız ilgili hesaba uygulanır.
* **Ürün/TF Override**: Hesap profili içinde ürün ve TF bazlı alanlar.
* **Strateji Preset**: Tek bir strateji ailesi için parametre paketi; profil içinde referans verilir.
* **Çalışma Zamanı Override**: UI’dan anlık yapılan ve hot‑reload ile devreye alınan değişiklikler (oturum boyunca kalıcı). İstenirse profile yazılabilir.

21.3 Override Çözümleme (Resolution Order)

En düşükten en yükseğe doğru uygulanır; en son gelen kazanır:

GLOBAL → ACCOUNT → PRODUCT → TIMEFRAME → STRATEGY\_PRESET → RUNTIME\_OVERRIDES

**Kurallar**: - Sayısal alanlar: son değer **tamamen** geçersiz kılar (merge değil). - Nesne alanları: **derin birleştirme** (deep‑merge) — aynı anahtarda son gelen kazanır. - Diziler: *merge\_mode* ile belirlenir — *replace* (varsayılan) veya *append\_unique* (ID’lere göre benzersiz ekleme).

21.4 Dosya Yapısı (Windows)

moonlight/ ├─ config.json # global ├─ profiles/ │ ├─ acc1/ │ │ ├─ profile.yaml # hesap için varsayılanlar │ │ ├─ overrides.yaml # UI’dan yazılan runtime değişiklikler (ops.) │ │ ├─ presets/ # hesap‑özel preset kopyaları │ │ └─ history/ # audit diff kayıtları (JSON Patch) │ └─ acc2/ ... acc4/ └─ backups/ # export çıktıları (şifreli arşiv – ops.)

21.5 Örnek Hesap Profili (*profiles/acc1/profile.yaml*)

schema\_version**:** 1meta**:** name**:** "Gündüz-Agresif" created\_by**:** "local-user" created\_at**:** "2025-10-07T10:00:00Z" description**:** "1/5 dak, permit sıkı; EMA/RSI + VWAP/RVOL"merge**:** arrays**:** "replace" *# strateji listelerinde tamamen değiştir*limits**:** max\_parallel\_per\_account**:** 6 *# global null ise burası geçerli olur* max\_daily\_loss**:** 6 max\_consecutive\_losses**:** 4products**:** **-** product**:** EURUSD strategies**:** **[**5**,**14**,**15**,**25**]** timeframes**:** **-** tf**:** 1 enabled**:** true win\_threshold**:** 0.74 permit\_min**:** 89 permit\_max**:** 93 risk**:** **{** lot**:** 1.0 **}** **-** tf**:** 5 enabled**:** true win\_threshold**:** 0.72 permit\_min**:** 90 permit\_max**:** 93 risk**:** **{** lot**:** 1.0 **}** **-** product**:** BTCUSD strategies**:** **[**15**,**25**,**35**]** timeframes**:** **-** tf**:** 5 enabled**:** true win\_threshold**:** 0.75 permit\_min**:** 86 permit\_max**:** 94presets**:** vwap\_rvol**:** id**:** 15 params**:** **{** rvol\_thr**:** 1.3**,** alpha**:** 1.0**,** beta**:** 0.5 **}** st\_adx**:** id**:** 25 params**:** **{** adx\_thr**:** 22**,** atr\_len**:** 10**,** mult**:** 3.0 **}**

21.6 Pydantic Model İskeleti

*# core/config/profile\_models.py***from** pydantic **import** BaseModel, Field**from** typing **import** List, Optional, Dict, Any**class** RiskCfg(BaseModel): lot: Optional[float] = None tp\_R: Optional[float] = None sl\_ATR\_mult: Optional[float] = None**class** TFOverride(BaseModel): tf: int enabled: bool = True win\_threshold: Optional[float] = None permit\_min: Optional[float] = None permit\_max: Optional[float] = None risk: Optional[RiskCfg] = None**class** ProductOverride(BaseModel): product: str enabled: bool = True strategies: Optional[List[int]] = None timeframes: Optional[List[TFOverride]] = None**class** Preset(BaseModel): id: int params: Dict[str, Any] = Field(default\_factory=dict)**class** AccountProfile(BaseModel): schema\_version: int = 1 meta: Dict[str, Any] = Field(default\_factory=dict) merge: Dict[str, Any] = Field(default\_factory=**lambda**: {"arrays": "replace"}) limits: Dict[str, Any] = Field(default\_factory=dict) products: List[ProductOverride] = Field(default\_factory=list) presets: Dict[str, Preset] = Field(default\_factory=dict)

21.7 Birleştirme Algoritması (Deep‑Merge)

1. Global *config.json* yüklenir ve şema doğrulanır.
2. Hesap profili okunur → *merge.arrays* kuralı alınır.
3. **Nesneler** için anahtar bazlı derin birleştirme; *None* gelen değerler ilgili anahtarı **silmez**, sadece override etmez.
4. **Diziler**:
   * *replace*: Tüm dizi yeni diziyle değiştirilir (strateji listesi gibi).
   * *append\_unique*: Öğeler ID/*name* alanına göre benzersiz eklenir.
5. **Runtime overrides** en son uygulanır (UI’dan gelen parçalı *PUT /config*).
6. Uygulama; etkilenen worker’lar için **hot‑reload** veya **kontrollü restart**.

21.8 Hot‑Reload & Restart Matrisi

* **Anında (hot)**: *win\_threshold*, *permit\_min/max*, *risk.lot*, *strategies (aktif/pasif)*, *logging.level*.
* **Worker‑restart**: Ürün/TF ekleme‑çıkarma, preset değişimi, ensemble paramları.
* **Servis‑restart**: *accounts[]* topolojisi, *storage.sqlite\_path* vb. UI’ya *CONFIG\_APPLIED* olayı yayınlanır: *{applied:[...], restarted:[...], warnings:[...]}*.

21.9 Dışa Aktarım / İçe Aktarım

* **Export**: *profiles/<acc>/profile.yaml* + opsiyonel *overrides.yaml* tek bir *.zip* arşive; *manifest.json* içinde *schema\_version*, *hash*, *created\_at*.
* **İçeri Aktar**: *schema\_version* uyumu kontrol; uymazsa **migration** (bkz. 21.12).
* **Gizlilik**: Parola/token **ihracata dahil edilmez**; yalnız *keyring\_service* adı bulunur.

21.10 Audit Log & Diff/Rollback

* Her *PUT /config* ve profil yazımı için **JSON Patch (RFC6902)** saklanır.
* **Tablo (SQLite)**: *config\_audit(id, ts\_ms, scope, actor, patch\_json, comment)*
* **Rollback**: Seçilen *id*’ye kadar (veya belirli *id*) patch’ler **tersine** uygulanır; *dry‑run* diff UI’da gösterilir.

21.11 UI Haritalaması

* **Ayarlar → Profiller**: Hesap seç, aktif profil adı, açıklama, **Uygula** / **Geri Al** / **Kopyala** / **Dışa aktar**.
* **Preset Galerisi**: Strateji aileleri için hazır paketler (RVOL eşikleri, ADX sınırları vb.).
* **Diff & Geri Alma**: Sol/sağ JSON/YAML görünümü; değişen anahtarlar listesi; *Apply* butonu.
* **Scope Anahtarı**: Editörde *hedef kapsam* etiketi (global/account/product/tf).

21.12 Migrasyon Stratejisi

* *config\_version* semver ile tutulur; profil *schema\_version* tamsayıdır.
* **Yukarı dönüşüm** (upgrade) fonksiyonları: *migrate\_1\_to\_2(profile)* vb.
* Eksik alanlar **varsayılan** ile doldurulur; kaldırılan alanlar uyarı olarak raporlanır.

21.13 CLI (opsiyonel)

moonlight-cli profile listmoonlight-cli profile apply --account acc1 profiles/acc1/profile.yamlmoonlight-cli profile diff --account acc1 --against runningmoonlight-cli profile rollback --account acc1 --to 2025-10-07T10:15:00Z

21.14 Test Planı

* **Birim**: Deep‑merge kuralları (obj/dizi), *append\_unique*, override sırası, hot‑reload sinyali.
* **Entegrasyon**: UI → REST *PUT /config* → Core; etkilenmiş worker yeniden başlatma; audit satırı eklenmesi.
* **Geri Alma**: Rasgele N değişiklikten sonra tek tık rollback; çalışır config ile sonuçlanır.
* **İçe/ Dışa Aktarım**: Export‑import döngüsünde hash ve *schema\_version* doğrulaması.

21.15 Kabul Kriterleri (Parça 21)

* Profil dosyası şeması **tanımlandı** (YAML örnek + Pydantic model).
* Override zinciri, merge kuralları ve hot‑reload/restart matrisi **net**.
* Audit/diff/rollback ve export/import mekanizmaları **tasarlandı**.
* UI ekranları ve opsiyonel CLI akışları **belirlendi**.

Parça 22 — Veri Modeli & Depolama (SQLite DDL, İndeksler, Migrasyon)

Amaç: MoonLight’ın çekirdek verilerinin **tutarlı**, **hızlı** ve **dayanıklı** biçimde saklanması; UI sorguları için uygun indeksleme; metrik/log/audit izleri; güvenli yedekleme ve şema migrasyonları.

22.1 Kısıtlar & Hedefler

* **Motor**: SQLite (tek dosya, kolay kurulum). Yüksek IO’da **WAL** modu, *synchronous=NORMAL*.
* **Eşzamanlılık**: Core tek süreç, çok async görev → yazma çakışmaları sınırlı. UI yalnız **okur**.
* **Güvenilirlik**: *client\_req\_id UNIQUE* ile idempotensi. Yabancı anahtarlar **açık**.
* **Sorgu kalıpları**: Son işlemler, ürün/TF kırılımı, günlük PnL, skip nedenleri, latency dağılımı.

22.2 SQLite Ayarları (ilk açılışta)

PRAGMA journal\_mode=WAL; *-- daha iyi eşzamanlılık*PRAGMA synchronous=**NORMAL**; *-- WAL ile önerilir*PRAGMA foreign\_keys=**ON**; *-- referans bütünlüğü*PRAGMA temp\_store=MEMORY; *-- küçük temp’ler RAM’de*PRAGMA page\_size=4096; *-- modern diskler için uygun*PRAGMA cache\_size=-20000; *-- ~20MB sayfa önbelleği*

22.3 ER Şeması (özet)

* **accounts( id )** — hesap profili ve durum
* **products( symbol )** — sembol meta
* **orders( id )** — giriş kaydı (snapshot: payout, amount, permit)
* **results( order\_id )** — sonuç/pnl/latency
* **trade\_skips** — neden bazlı atlanan girişler
* **metrics\_raw / metrics\_rollup\_5m** — zaman serisi metrikler
* **provider\_perf** — sağlayıcı/strateji performans özetleri
* **ensemble\_calibration** — Platt/kalibrasyon paramları
* **config\_audit** — config değişim izleri (JSON Patch)
* **trade\_events** — kritik olay indeks tablosu (JSONL log’a paralel)
* **candles\_cache** — kısa ömürlü mum önbellek (ops.)
* **schema\_migrations** — migrasyon geçmişi

22.4 DDL — Çekirdek Tablolar

*-- accounts: kullanıcı kimliği maskeli tutulur***CREATE** **TABLE** **IF** **NOT** **EXISTS** accounts ( **id** TEXT **PRIMARY** **KEY**, username\_mask TEXT, profile\_path TEXT, state TEXT **CHECK**(state **IN** ('connected','disconnected','otp\_required')), last\_login\_ms INTEGER, balance REAL);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_accounts\_state **ON** accounts(state); *-- products: sembol meta***CREATE** **TABLE** **IF** **NOT** **EXISTS** products ( symbol TEXT **PRIMARY** **KEY**, kind TEXT **DEFAULT** 'forex' *-- forex|crypto|metal|stock|index*); *-- orders: FTT giriş kaydı***CREATE** **TABLE** **IF** **NOT** **EXISTS** orders ( **id** TEXT **PRIMARY** **KEY**, ts\_open\_ms INTEGER **NOT** **NULL**, account\_id TEXT **NOT** **NULL** **REFERENCES** accounts(**id**) **ON** **DELETE** **CASCADE**, product TEXT **NOT** **NULL** **REFERENCES** products(symbol) **ON** **DELETE** **RESTRICT**, timeframe INTEGER **NOT** **NULL**, *-- 1|5|15 (dakika)* direction INTEGER **NOT** **NULL**, *-- +1=call, -1=put* amount REAL **NOT** **NULL**, payout\_pct REAL **NOT** **NULL**, *-- girişteki payout snapshot* client\_req\_id TEXT **NOT** **NULL** **UNIQUE**, *-- idempotensi* permit\_win\_min REAL, permit\_win\_max REAL);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_orders\_acc\_time **ON** orders(account\_id, ts\_open\_ms **DESC**);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_orders\_prod\_tf **ON** orders(product, timeframe, ts\_open\_ms **DESC**); *-- results: FTT sonuç***CREATE** **TABLE** **IF** **NOT** **EXISTS** results ( order\_id TEXT **PRIMARY** **KEY** **REFERENCES** orders(**id**) **ON** **DELETE** **CASCADE**, ts\_close\_ms INTEGER **NOT** **NULL**, status TEXT **NOT** **NULL** **CHECK**(status **IN** ('win','lose','push','abort','canceled')), pnl REAL **NOT** **NULL**, duration\_ms INTEGER, latency\_ms INTEGER);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_results\_status **ON** results(status); *-- trade\_skips: neden bazlı atlamalar***CREATE** **TABLE** **IF** **NOT** **EXISTS** trade\_skips ( **id** INTEGER **PRIMARY** **KEY** AUTOINCREMENT, ts\_ms INTEGER **NOT** **NULL**, account\_id TEXT, product TEXT, timeframe INTEGER, reason TEXT **NOT** **NULL**, *-- permit|confidence|concurrency|risk\_guard|overrun|latency\_guard* details TEXT *-- küçük JSON*);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_skips\_ts **ON** trade\_skips(ts\_ms **DESC**);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_skips\_reason **ON** trade\_skips(reason); *-- metrics: ham ve 5 dakikalık rollup***CREATE** **TABLE** **IF** **NOT** **EXISTS** metrics\_raw ( ts\_ms INTEGER **NOT** **NULL**, **scope** TEXT **NOT** **NULL**, *-- global|acc:..|prod:..|tf:..* **key** TEXT **NOT** **NULL**, value REAL **NOT** **NULL**, **PRIMARY** **KEY**(ts\_ms, **scope**, **key**));**CREATE** **TABLE** **IF** **NOT** **EXISTS** metrics\_rollup\_5m ( bucket\_ms INTEGER **NOT** **NULL**, **scope** TEXT **NOT** **NULL**, **key** TEXT **NOT** **NULL**, cnt INTEGER **NOT** **NULL**, avg REAL, p50 REAL, p90 REAL, p99 REAL, min REAL, max REAL, **PRIMARY** **KEY**(bucket\_ms, **scope**, **key**));**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_metrics5\_scope **ON** metrics\_rollup\_5m(**scope**, **key**, bucket\_ms **DESC**); *-- sağlayıcı/strateji performansı (kayan pencere özet)***CREATE** **TABLE** **IF** **NOT** **EXISTS** provider\_perf ( pid INTEGER **NOT** **NULL**, **scope** TEXT **NOT** **NULL**, *-- ör. EURUSD:1 veya acc1:EURUSD:1* window INTEGER **NOT** **NULL**, *-- 50/200 gibi* win INTEGER **NOT** **NULL** **DEFAULT** 0, lose INTEGER **NOT** **NULL** **DEFAULT** 0, last\_updated\_ms INTEGER, **PRIMARY** **KEY**(pid, **scope**, window)); *-- ensemble kalibrasyonu (Platt)***CREATE** **TABLE** **IF** **NOT** **EXISTS** ensemble\_calibration ( **scope** TEXT **PRIMARY** **KEY**, *-- acc:prod:tf* a REAL **NOT** **NULL**, b REAL **NOT** **NULL**, ece REAL, brier REAL, updated\_ms INTEGER); *-- config audit (RFC6902 JSON Patch)***CREATE** **TABLE** **IF** **NOT** **EXISTS** config\_audit ( **id** INTEGER **PRIMARY** **KEY** AUTOINCREMENT, ts\_ms INTEGER **NOT** **NULL**, actor TEXT, **scope** TEXT, patch\_json TEXT **NOT** **NULL**, **comment** TEXT);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_audit\_ts **ON** config\_audit(ts\_ms **DESC**); *-- kritik olayların hızlı indekslenmesi (tam log JSONL dosyadadır)***CREATE** **TABLE** **IF** **NOT** **EXISTS** trade\_events ( **id** INTEGER **PRIMARY** **KEY** AUTOINCREMENT, ts\_ms INTEGER **NOT** **NULL**, **level** TEXT **CHECK**(**level** **IN** ('INFO','WARN','ERROR')), chan TEXT, *-- trade|system|security|connector|worker|api* event TEXT, *-- PLACE\_OK|CONFIRM\_TIMEOUT|...* **account** TEXT, product TEXT, timeframe INTEGER, order\_id TEXT, message TEXT);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_events\_ts **ON** trade\_events(ts\_ms **DESC**);**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_events\_event **ON** trade\_events(event); *-- mum önbelleği (opsiyonel)***CREATE** **TABLE** **IF** **NOT** **EXISTS** candles\_cache ( product TEXT **NOT** **NULL**, timeframe INTEGER **NOT** **NULL**, ts\_ms INTEGER **NOT** **NULL**, **open** REAL, high REAL, low REAL, **close** REAL, volume REAL, **PRIMARY** **KEY**(product, timeframe, ts\_ms));**CREATE** **INDEX** **IF** **NOT** **EXISTS** idx\_candles\_recent **ON** candles\_cache(product, timeframe, ts\_ms **DESC**); *-- şema migrasyonları***CREATE** **TABLE** **IF** **NOT** **EXISTS** schema\_migrations ( version TEXT **PRIMARY** **KEY**, applied\_ms INTEGER **NOT** **NULL**);

Görünümler (Views)

**CREATE** **VIEW** **IF** **NOT** **EXISTS** v\_trades **AS****SELECT** o.**id**, o.ts\_open\_ms, o.account\_id, o.product, o.timeframe, o.direction, o.amount, o.payout\_pct, r.ts\_close\_ms, r.status, r.pnl, r.latency\_ms**FROM** orders o **LEFT** **JOIN** results r **ON** r.order\_id = o.**id**;

22.5 Sorgu Örnekleri

* **Son 200 işlem (EURUSD, TF=1)**

**SELECT** \* **FROM** v\_trades **WHERE** product='EURUSD' **AND** timeframe=1**ORDER** **BY** ts\_open\_ms **DESC** **LIMIT** 200;

* **Win‑rate (son 7 gün)**

**SELECT** AVG(**CASE** **WHEN** status='win' **THEN** 1.0 **WHEN** status='lose' **THEN** 0 **ELSE** **NULL** **END**) **AS** win\_rate**FROM** results r **JOIN** orders o **ON** o.**id**=r.order\_id**WHERE** r.ts\_close\_ms > strftime('%s','now')\*1000 - 7\*86400000;

* **Skip nedenleri dağılımı (24 saat)**

**SELECT** reason, COUNT(\*) **FROM** trade\_skips**WHERE** ts\_ms > strftime('%s','now')\*1000 - 86400000**GROUP** **BY** reason **ORDER** **BY** 2 **DESC**;

* **Payout ve beklenen değer trendi (rollup)**

**SELECT** bucket\_ms, avg **FROM** metrics\_rollup\_5m**WHERE** **scope**='prod:EURUSD' **AND** **key**='payout\_avg' **ORDER** **BY** bucket\_ms **DESC** **LIMIT** 288;

22.6 Tutma Politikaları (Retention)

* **orders/results**: **Süresiz** (önerilen) veya kullanıcı tanımlı (örn. 365 gün).
* **trade\_events**: 30–90 gün.
* **metrics\_raw**: 7–14 gün; **rollup** kalıcı.
* **candles\_cache**: TF’e göre 1–7 gün.
* Otomatik görev: *retention\_job()* — zamanlanmış **DELETE** + *VACUUM* (haftalık) veya *VACUUM INTO* ile çevrim içi yedek.

22.7 Yedekleme & Geri Yükleme

* **WAL** ile çevrim içi yedek: *sqlite3 .backup* eşleniği veya *VACUUM INTO 'backup.sqlite'*.
* Yedek dizini: *C:\ProgramData\MoonLight\backups\db-YYYYMMDD.sqlite*.
* UI’da “**Anlık Yedek Al**” ve “**Geri Yükle**” butonları; işlem öncesi servis **pause** edilir.

22.8 Migrasyon Stratejisi

* Her sürümde **idempotent** DDL (IF NOT EXISTS).
* Kırıcı değişiklikte *schema\_migrations*’a **version** eklenir; kod içi **migrator** gerekli ALTER/CREATE’ları çalıştırır.
* Rollback için önceki dosya **yedeği** gerekir (DDL geri sarımı SQLite’ta sınırlı).

22.9 Erişim Katmanı (Python, aiosqlite — iskelet)

*# core/storage/sqlite.py***import** aiosqlite**from** contextlib **import** asynccontextmanager@asynccontextmanager**async** **def** open\_db(path: str): db = **await** aiosqlite.connect(path) **await** db.execute('PRAGMA foreign\_keys=ON') **await** db.execute('PRAGMA journal\_mode=WAL') **try**: **yield** db **finally**: **await** db.close()**async** **def** save\_order(db, row: dict): sql = """ INSERT INTO orders(id, ts\_open\_ms, account\_id, product, timeframe, direction, amount, payout\_pct, client\_req\_id, permit\_win\_min, permit\_win\_max) VALUES(:id,:ts\_open\_ms,:account\_id,:product,:timeframe,:direction, :amount,:payout\_pct,:client\_req\_id,:permit\_win\_min,:permit\_win\_max) ON CONFLICT(id) DO NOTHING; """ **await** db.execute(sql, row); **await** db.commit()**async** **def** save\_result(db, row: dict): sql = """ INSERT INTO results(order\_id, ts\_close\_ms, status, pnl, duration\_ms, latency\_ms) VALUES(:order\_id,:ts\_close\_ms,:status,:pnl,:duration\_ms,:latency\_ms) ON CONFLICT(order\_id) DO UPDATE SET ts\_close\_ms=excluded.ts\_close\_ms, status=excluded.status, pnl=excluded.pnl, duration\_ms=excluded.duration\_ms, latency\_ms=excluded.latency\_ms; """ **await** db.execute(sql, row); **await** db.commit()

22.10 Performans Notları

* Zaman aralığı + sıralı sorgular için **bileşik indeksler** (örn. *product,timeframe,ts\_open\_ms DESC*).
* Büyük *INSERT* akışlarında **transaction batching** (100–500 satır).
* *metrics\_raw* için **PRIMARY KEY (ts, scope, key)** — tekilleştirme ve hızlı upsert.
* UI raporlarına özel **görünümler**: *v\_trades*, *v\_daily\_pnl* (ops.).

22.11 Güvenlik & PII

* Kullanıcı adı/e‑posta **maskeli**; gerçek kimlik bilgileri DB’de **yok**.
* *security.log* eşleniği DB’ye **yazılmaz**; yalnız kritik olaylar *trade\_events*’e.
* Dışa aktarımda (CSV) PII kolonlar **hariç** tutulur.

22.12 Test Planı

* **Bütünlük**: *FOREIGN KEY* ihlali testleri; *client\_req\_id UNIQUE* ile çifte kayıt önleme.
* **Performans**: 1M *v\_trades* satırında son 200 sorgusu < 50 ms hedef (SSD).
* **Migrasyon**: Eski sürümden yeni sürüme ALTER/CREATE akışı; *schema\_migrations* satırı eklenir.
* **Retention**: Otomatik görev sonrası dosya boyutu düşüşü ve indeks sağlığı.

22.13 Kabul Kriterleri (Parça 22)

* Çekirdek tablolar & indeksler **tanımlandı** ve DDL hazır.
* Metrik, audit, kalibrasyon ve olay tabloları **eklendi**; görünümler ile UI sorguları destekleniyor.
* Migrasyon/retention/yedekleme politikaları ve Python erişim iskeleti **tamamlandı**.

Parça 23 — Strateji Geliştirme SDK’sı (Plugin API, Test Harness, Sandbox)

Amaç: Üçüncü taraf veya ekip içi geliştiricilerin **güvenli**, **standart** ve **test edilebilir** biçimde strateji/sinyal sağlayıcı yazabilmesi için SDK ve araçları tanımlamak.

23.1 Tasarım İlkeleri

* **Tek sözleşme**: Parça 13’teki *StrategyProvider* arayüzü esastır; **semver** ile sürümlenir (*provider\_api=1.x*).
* **İzolasyon**: Strateji kodu **sandbox** alt‑süreçte çalışabilir; hafıza/CPU sınırları; **ağ erişimi yok**.
* **Deterministik**: *seed* geçirilebilir; veri yalnız *df/feats* ile sınırlıdır (I/O yasak).
* **Gözlemlenebilir**: Her değerlendirme çağrısı **meta** döndürür; debug için *provider\_logs* kanalı.
* **Dağıtılabilir**: Stratejiler **paket** (zip/wheel) veya **klasör** olarak yüklenebilir.

23.2 SDK Dizinleri

sdk/ cookiecutter\_provider/ # şablon proje harness/ # test/bench araçları sandbox/ # izole çalıştırıcı docs/ # rehberler ve örnekler

23.3 Provider Manifesti

Hem paket hem klasör yüklemelerinde *provider.yaml* zorunlu:

id**:** 105 *# benzersiz (int)*name**:** "EMA+RSI (Custom)"version**:** "1.0.0"api**:** 1 *# provider\_api major*author**:** "you"entry**:** "provider:CustomEMA\_RSI" *# <module>:<class>*warmup\_bars**:** 60params\_schema**:** *# basit doğrulama; UI formu üretimi için* ema\_fast**:** **{**type**:** int**,** min**:** 3**,** max**:** 200**,** default**:** 20**}** ema\_slow**:** **{**type**:** int**,** min**:** 5**,** max**:** 400**,** default**:** 50**}** rsi\_len**:** **{**type**:** int**,** min**:** 2**,** max**:** 50**,** default**:** 14**}** rsi\_up**:** **{**type**:** float**,** min**:** 50**,** max**:** 90**,** default**:** 55**}** rsi\_dn**:** **{**type**:** float**,** min**:** 10**,** max**:** 50**,** default**:** 45**}**notes**:** "Trend+momentum doğrulaması; range’de whipsaw olabilir."

23.4 Paketleme Modları

* **Folder plugin**: *providers/<slug>/provider.py* + *provider.yaml*. Geliştirme için ideal.
* **Zip/Wheel**: *mlsdk pack providers/<slug> → dist/<slug>-1.0.0.zip*. UI’dan **Yükle** ile eklenir.
* **İmza (ops.)**: Zip imza dosyası (*.sig*) ve geliştirici anahtarı; UI doğrular.

23.5 Yükleme & Kayıt

* *registry.load\_all()* klasörleri ve *dist/* paketlerini tarar, *entry*’den sınıfı yükler.
* Çakışan *id* algılanır → **hata**; yalnız **enable** edilenler aktif olur.
* API major uyuşmazlığı (örn. 2.x vs 1.x) → **uyumsuz** olarak işaretlenir.

23.6 Sandbox Çalıştırıcı (Windows)

* **Model**: Her provider *eval* çağrısı, **opsiyonel** olarak *subprocess* içinde yürütülür.
* **Kısıtlar**:
  + **Job Object** ile **CPU süresi** ve **RAM** sınırlaması (örn. 100ms/batch, 128MB).
  + **Ağ** ve **disk** erişimi **yasak** (import hook + policy).
  + *sys.modules* beyaz liste: numerik/pandas, SDK yardımcıları.
* **Hata modeli**: Sınır ihlali → *vote=0, score=0* + alert.

*# sdk/sandbox/run\_eval.py (özet)***class** SandboxedProvider: **def** \_\_init\_\_(self, provider): self.p = provider **def** evaluate(self, df, feats, ctx): **with** cpu\_time\_budget(0.1), memory\_limit\_mb(128), no\_network(): **return** self.p.evaluate(df, feats, ctx)

23.7 Test Harness

* **Unit fixtures**: Sentetik OHLCV jeneratörü (trend/range/vol), hazır **golden** veri seti.
* **Check‑list**: warm‑up, NaN, param aralığı, *vote∈{-1,0,1}*, *score* sınırlaması, süre/CPU.
* **Run modes**:
  1. **Quick**: 10k bar, tek TF; smoke test, < 2 sn.
  2. **Full**: 100k bar, 3 TF; latency ve stabilite metrikleri.
* **Rapor**: JSON + HTML — win%, katkı, p̂ istatistikleri (Parça 16 motorunu kullanır).

mlsdk test providers/my\_strat --tf 1 --data data/eurusd\_1m.csv --report out/

23.8 Yardımcı Kütüphaneler (SDK Helpers)

* *sdk.ta* — hızlı indikatörler (EMA, RSI, ST, ADX, BB, KC, VWAP, RVOL).
* *sdk.features* — pencere fonksiyonları: *slope*, *zscore*, *bandwidth*, *crossover*, *reclaim\_detector*.
* *sdk.metrics* — p̂ kalibrasyonu için yardımcılar (Brier, reliability).
* *sdk.tools* — *assert\_shape*, *safe\_clip*, *as\_float* vb.

23.9 Örnek Sağlayıcı — RSI Band Reversal

*# providers/rsi\_band/provider.py***from** core.strategies.base **import** StrategyProvider, ProviderConfig, ProviderContext**from** core.ensemble **import** ProviderVote**from** sdk.ta **import** rsi**class** RSIBand(StrategyProvider): META = {"name":"RSI Band Reversal","group":"reversal"} **def** \_\_init\_\_(self, cfg: ProviderConfig): self.cfg = cfg; P = cfg.params self.len = P.get('len', 14) self.up = P.get('up', 70) self.dn = P.get('dn', 30) **def** warmup\_bars(self): **return** self.len + 5 **def** evaluate(self, df, feats, ctx: ProviderContext): r = rsi(df['close'], self.len) **if** len(r) < self.warmup\_bars(): **return** None v = 0 **if** r.iloc[-2] > self.up **and** r.iloc[-1] < self.up: v = -1 *# aşağı kırılım → put* **if** r.iloc[-2] < self.dn **and** r.iloc[-1] > self.dn: v = +1 *# yukarı kırılım → call* s = float((r.iloc[-1] - 50.0)/50.0) **if** v != 0 **else** 0.0 **return** ProviderVote(pid=self.cfg.id, vote=v, score=s)

23.10 Örnek Sağlayıcı — VWAP Reclaim

*# providers/vwap\_reclaim/provider.py***from** sdk.ta **import** vwap, rolling\_mean**from** core.strategies.base **import** StrategyProvider, ProviderConfig, ProviderContext**from** core.ensemble **import** ProviderVote**class** VWAPReclaim(StrategyProvider): META = {"name":"VWAP Reclaim + RVOL","group":"trend+volume"} **def** \_\_init\_\_(self, cfg): self.cfg = cfg; P = cfg.params self.rvol\_lb = P.get('rvol\_lb', 20) self.rvol\_thr= P.get('rvol\_thr', 1.3) **def** warmup\_bars(self): **return** max(20, self.rvol\_lb) + 5 **def** evaluate(self, df, feats, ctx: ProviderContext): vw = vwap(df) rvol = feats['rvol\_lb20'] *# Parça 8’de hesaplanmış kabul* **if** len(vw) < self.warmup\_bars(): **return** None reclaim = df['close'].iloc[-2] < vw.iloc[-2] **and** df['close'].iloc[-1] > vw.iloc[-1] **if** reclaim **and** rvol.iloc[-1] > self.rvol\_thr: dist = (df['close'].iloc[-1]-vw.iloc[-1]) / vw.iloc[-1] score = 0.8\*dist + 0.2\*(rvol.iloc[-1]-1) **return** ProviderVote(pid=self.cfg.id, vote=+1, score=float(score)) **return** ProviderVote(pid=self.cfg.id, vote=0, score=0.0)

23.11 CLI Araçları

* *mlsdk new provider <slug>* → cookiecutter ile iskelet.
* *mlsdk validate providers/<slug>* → manifest/param/warm‑up doğrulama.
* *mlsdk pack providers/<slug>* → zip oluştur, opsiyonel imzala.
* *mlsdk test ...* → harness ile backtest & rapor (Parça 16).

23.12 UI Entegrasyonu

* **Strateji Kataloğu**: “Dışarıdan Yükle” (zip), manifest önizleme, **güvenlik uyarısı** (ağ erişimi yok).
* **Param Formu**: *params\_schema*’dan otomatik form; tip/alt‑üst sınır; “varsayılanı geri yükle”.
* **Performans Sekmesi**: Son N işlem katkıları, latency, hata oranı.

23.13 Güvenlik ve Uyum Notları

* **Ağ ve dosya erişimi yasak**; yalnız core’un verdiği veri kullanılabilir.
* **Zaman sınırlı** çalıştırma; aşımda vote=0 ve WARN.
* **Uyum**: Platform **kullanım şartlarını** ihlal eden, scraping/otomasyon içeren sağlayıcılar **yüklenemez**.
* **İmza** (ops.): Şirket içi anahtarla imzalı paketler yalnız güvenilir listeden yüklenir.

23.14 CI/QA

* **Statik analiz**: *ruff* ve *mypy* kuralları; *provider\_api* sürüm kontrolü.
* **Birim**: Smoke test (Quick), golden veri karşılaştırması.
* **Benchmark**: Latency p50/p90 raporu; bellek ayak izi.
* **Uyumluluk**: API major değişiminde *FAIL* ve göç rehberi linki.

23.15 Kabul Kriterleri (Parça 23)

* Manifest, yükleme ve kayıt mekanizması **çalışır**; çakışma/uyumsuzluk tespiti var.
* Sandbox (opsiyonel) kaynak sınırlı çalıştırma ve ihlal tespiti yapar.
* CLI araçları ile *new/validate/pack/test* akışları **tam**.
* En az iki örnek strateji ve temel test harness’i **hazır**; UI entegrasyonu tanımlı.

Parça 23 — Strateji Geliştirme SDK’sı (Plugin API, Test Harness, Sandbox)

Amaç: MoonLight’a **yeni stratejilerin** güvenli, tekrarlanabilir ve kolay biçimde eklenebilmesi için bir **SDK** sağlamak: eklenti (plugin) iskeleti, paketleme, test/benchmark araçları, güvenli **sandbox** çalıştırma ve sürümleme.

23.1 Tasarım İlkeleri

* **Sözleşme‑önce**: Stratejiler *StrategyProvider* arayüzünü uygular (Parça 13).
* **Deterministik**: Aynı girişe aynı çıktı; rasgelelik varsa *seed* ile sabitlenir.
* **Hız Bütçesi**: TF=1’de **< 1ms/provider** ort., p95 **< 3ms**.
* **İzolasyon**: Plugin kodu **sandbox process** içinde, IPC ile çağrılır.
* **Gözlemlenebilir**: Her çağrıda *meta* alanı (sebep, özellik özetleri).
* **Geriye Uyumluluk**: *strategy\_api\_version* alanı ile.

23.2 Dizin & Paketleme

core/ strategies/ base.py # Provider arayüzü registry.py # Keşif/kayıtsdk/ cli.py # moonlight‑sdk komutları templates/ provider/ \_\_init\_\_.py provider.py.j2 # iskelet test\_provider.py.j2 sandbox/ runner.py # alt‑süreç çalıştırıcı policy.py # import/policy kısıtları docs/ provider\_guidelines.md

23.3 Strateji Yaşam Döngüsü

1. **Oluştur**: *moonlight‑sdk new provider --name my\_momentum --id 101*
2. **Geliştir**: *evaluate()* mantığını yaz; param şemasını tanımla.
3. **Test et**: *moonlight‑sdk test providers/my\_momentum* (unit + golden + perf).
4. **Benchmark**: Örnek veri setinde p95 süre/latency ölç.
5. **Paketle**: *moonlight‑sdk pack* → *.mlprov* (zip).
6. **Yükle**: UI → Strateji Kataloğu → “İçe aktar (.mlprov)”.

23.4 Public API (özet)

*# sdk public yüzeyi***class** StrategyProvider: **def** warmup\_bars(self) -> int: ... **def** evaluate(self, df: pd.DataFrame, feats: Dict[str, Any], ctx: ProviderContext) -> Optional[ProviderVote]: ...@dataclass**class** ProviderContext: product: str timeframe: int payout: float *# anlık payout%* seed: int | None = None@dataclass**class** ProviderVote: pid: int vote: int *# -1 / 0 / +1* score: float *# [-inf, +inf]* meta: Dict[str, Any] = field(default\_factory=dict)

23.5 İskelet Şablonu (Jinja2)

*# templates/provider/provider.py.j2***from** core.strategies.base **import** StrategyProvider, ProviderConfig, ProviderContext**from** core.ensemble **import** ProviderVote**import** pandas **as** pd**class** {{ClassName}}(StrategyProvider): META = {"name": "{{DisplayName}}", "group": "{{Group}}", "api": 1} **def** \_\_init\_\_(self, cfg: ProviderConfig): self.cfg = cfg p = cfg.params self.len = int(p.get('len', 14)) self.up = float(p.get('up', 55)) self.dn = float(p.get('dn', 45)) **def** warmup\_bars(self) -> int: **return** max(30, self.len + 5) **def** evaluate(self, df: pd.DataFrame, feats, ctx: ProviderContext): **if** len(df) < self.warmup\_bars(): **return** None rsi = feats['rsi'](df['close'], self.len) *# feature helper (Parça 8)* val = float(rsi.iloc[-1]) vote = 1 **if** val > self.up **else** (-1 **if** val < self.dn **else** 0) score = (val - 50.0)/50.0 meta = {"rsi": val, "up": self.up, "dn": self.dn} **return** ProviderVote(pid=self.cfg.id, vote=vote, score=score, meta=meta)

23.6 Test Harness

* **Birim Testi**: Kenar durumları (NaN, warm‑up, flat seriler) ve **doğru vote/score**.
* **Golden Test**: Örnek veri üzerinde sabit çıktı dosyasıyla karşılaştırma (kırılma uyarısı).
* **Performans Testi**: 10k evaluate çağrısı; p95 süre ölçümü; hedefler aşılırsa **fail**.
* **Stres**: TF=1 sentetik veri; *append\_only* modunda bellek sızıntısı kontrolü.

*# templates/provider/test\_provider.py.j2***import** pandas **as** pd, numpy **as** np, time**from** core.strategies.registry **import** build**from** core.strategies.base **import** ProviderConfig, ProviderContext**def** make\_df(n=500): ts = np.arange(n)\*60\_000 close = np.cumsum(np.random.randn(n))\*0.1 + 100 **return** pd.DataFrame({"ts\_ms": ts, "open": close, "high": close, "low": close, "close": close, "volume": 1})**def** test\_basic\_vote(): df = make\_df() p = build({{ID}}, cfg=ProviderConfig(id={{ID}}, name="{{DisplayName}}", group="{{Group}}")) ctx = ProviderContext(product="EURUSD", timeframe=1, payout=90.0, seed=42) feats = get\_feature\_helpers() *# Parça 8* v = p.evaluate(df, feats, ctx) **assert** v **is** None **or** v.vote **in** (-1,0,1)**def** test\_perf(): df = make\_df(5000) p = build({{ID}}, cfg=ProviderConfig(id={{ID}}, name="{{DisplayName}}", group="{{Group}}")) ctx = ProviderContext(product="EURUSD", timeframe=1, payout=90.0) feats = get\_feature\_helpers() t0 = time.perf\_counter() **for** \_ **in** range(1000): p.evaluate(df, feats, ctx) p95 = (time.perf\_counter()-t0)/1000\*1000 **assert** p95 < 3.0 *# ms*

23.7 Sandbox Çalıştırma

Hedef: Pluginlerin ana süreçten **izole** ve sınırlı kaynakla çalışması; hatalarda güvenli kapanış.

* **Model**: Her provider **aynı süreçte** yüklenir; **evaluate** çağrıları **sandbox subprocess** üzerinden yapılır (paylaşımlı bellek maliyetini azaltmak için *opsiyon*).
* **Windows**: *multiprocessing* + *spawn*; zaman aşımı ve sağlık kontrolü.
* **Policy**:
  + İzinli import listesi: *numpy*, *pandas*, *math*, *statistics*.
  + **Yasak**: *os*, *subprocess*, *socket*, *ctypes*, *threading* (sandbox süreçte engellenir).
  + Maks bellek: örn. **256 MB**; CPU zamanı: **50 ms** / çağrı (watchdog).
* **IPC**: *multiprocessing.Queue* veya *pyzmq* (ops.).
* **Arıza yönetimi**: Zaman aşımı → vote=0 + *alerts* uyarısı; tekrar eden aşım → provider otomatik **disable** (oturumluk).

*# sdk/sandbox/runner.py (kısaltılmış)***import** multiprocessing **as** mp, time, signal**class** Sandbox: **def** \_\_init\_\_(self, provider): self.p = provider **def** call(self, df, feats, ctx, timeout\_ms=50): q = mp.Queue(1) proc = mp.Process(target=self.\_target, args=(q, df, feats, ctx)) proc.start(); proc.join(timeout\_ms/1000) **if** proc.is\_alive(): proc.terminate(); proc.join(1) **return** None, {"error":"timeout"} **try**: **return** q.get\_nowait(), None **except** Exception **as** e: **return** None, {"error": str(e)} **def** \_target(self, q, df, feats, ctx): *# burada kısıtlı import/policy uygulanır (policy.py)* v = self.p.evaluate(df, feats, ctx) q.put(v)

23.8 CLI Komutları (sdk/cli.py)

moonlight-sd k new provider --name ema\_rsi\_v2 --id 105 --group trend+oscmoonlight-sd k test providers/ema\_rsi\_v2moonlight-sd k bench providers/ema\_rsi\_v2 --dataset data/sample/EURUSD\_1m.csvmoonlight-sd k pack providers/ema\_rsi\_v2 # → build/ema\_rsi\_v2-105.mlprovmoonlight-sd k lint providers/ema\_rsi\_v2 # stil ve policy kontrol

23.9 Kalite Kuralları (Lint/Policy)

* **Nümerik istikrar**: Küçük paydaya bölme yok; *max(ε, denom)* ( ε=1e‑9).
* **NaN yönetimi**: NaN gördüğünde **vote=0**, *meta.nan=...*.
* **Warm‑up saygısı**: *warmup\_bars()* altı **None** döndür.
* **Kısa dizi**: Son bar yerine son **kapanmış** bar kullan (TF hizalama).
* **Yan etkisizlik**: Global durum tutma yok; sınıf içi durum **salt okunur** paramlar.

23.10 Belgeleme Şablonu

# {{DisplayName}} (ID: {{ID}})- Grup: {{Group}}- Amaç: Kısa açıklama- Parametreler: tablo (ad, tip, varsayılan, aralık)- Warm‑up: N bar- Sinyal Mantığı: formül ve koşullar- Skor: hesaplama ve sınırlar (s\_cap önerisi)- Kenar durumlar: permit/payout, flat bar, NaN- Performans: p95 < 3ms (test kanıtı)- Değişim Günlüğü: v1.0 → v1.1

23.11 Sürümleme & Uyumluluk

* *provider.json* meta: *{ id, name, group, version, api: 1, min\_core: "1.0.0" }*
* **SemVer**: Çekirdek mantık kırıcı değişirse MAJOR artar.
* UI Katalogu kısmi **uyumsuzluk** durumunda uyarı verir ve provider’ı pasif listeler.

23.12 Güvenlik & Uyum

* Yalnız **resmî/izinli API** ve veri kaynakları; **scraping, RPA veya güvenlik atlatma yok**.
* 2FA varsa kullanıcı **manuel OTP** girişi; OTP saklanmaz.
* Pluginler dosya sistemi/ağ erişimi **olmadan** çalışır (sandbox policy).
* Performans aşımlarında otomatik **throttle/disable** + uyarı.

23.13 Kabul Kriterleri (Parça 23)

* *moonlight‑sdk* CLI, şablonlar ve test harness **tanımlandı**.
* Sandbox politika ve zaman aşımı yönetimi açıklandı.
* Paketleme *.mlprov* ve meta sürümleme kuralı hazır.
* Kalite kuralları ve belgeleme şablonu sağlandı.

17.7 Örnek SQL Sorguları (devam)

* **Skip nedenleri (son 1 gün)**

*-- Not: trade\_skips bir VIEW olabilir; loglardan ETL ile doldurulan özet bir tablo olarak da tutulabilir.***SELECT** reason, COUNT(\*) **AS** cnt**FROM** trade\_skips**WHERE** ts\_ms > (strftime('%s','now')\*1000 - 86\_400\_000)**GROUP** **BY** reason**ORDER** **BY** cnt **DESC**;

* **Strateji bazında win‑rate (son 200)**

**SELECT** strategy\_id, AVG(**CASE** **WHEN** status='win' **THEN** 1.0 **WHEN** status='lose' **THEN** 0.0 **END**) **AS** win\_rate**FROM** ( **SELECT** p.strategy\_id, r.status **FROM** orders o **JOIN** results r **ON** r.order\_id=o.**id** **JOIN** features f **ON** f.order\_id=o.**id** **JOIN** strategy\_perf p **ON** p.account\_id=o.account\_id **AND** p.product=o.product **AND** p.timeframe=o.timeframe **WHERE** o.product='EURUSD' **AND** o.timeframe=1 **ORDER** **BY** r.ts\_close\_ms **DESC** **LIMIT** 200)**GROUP** **BY** strategy\_id**ORDER** **BY** win\_rate **DESC**;

* **Latency p90 (son 24 saat)**

**SELECT** CAST( (**SELECT** value **FROM** metrics **WHERE** **key**='latency\_ms' **AND** ts\_ms>strftime('%s','now')\*1000 - 86\_400\_000 **ORDER** **BY** value **LIMIT** 1 OFFSET (**SELECT** COUNT(\*)\*9/10 **FROM** metrics **WHERE** **key**='latency\_ms' **AND** ts\_ms>strftime('%s','now')\*1000 - 86\_400\_000) ) **AS** INTEGER) **AS** p90\_latency\_ms;

17.8 Kabul Kriterleri (Parça 17)

* Metrik taksonomisi, JSONL log formatı ve uyarı kuralları tanımlandı.
* WS yayın oran sınırları ve downsample/flush politikası yazıldı.
* UI panelleri (dashboard, ürün/TF, log/alert) için gerekli veri alanları listelendi.
* SQL/VIEW örnekleri ile raporlama akışı gösterildi.

Parça 18 — CI/CD & Sürümleme (Windows Odaklı)

18.1 Sürümleme ve Dallanma

* **SemVer**: *MAJOR.MINOR.PATCH* (örn. *1.0.0*).
* **Branch modeli**: *main* (kararlı), *dev* (güncel geliştirme), *release/x.y* (kandida). Hotfix: *hotfix/x.y.z*.
* **Tag**: Her kararlı sürüm *vX.Y.Z* ile etiketlenir; release notları otomatik üretilir.

18.2 Pipeline Aşamaları (GitHub Actions ör.)

1. **Lint/Format**: *ruff*, *black* (check), Flutter *dart format --set-exit-if-changed*.
2. **Type Check**: *mypy* (core), *dart analyze* (UI).
3. **Test**: *pytest -q* (unit+integration, MockConnector); Flutter *flutter test*.
4. **Build**: Core → PyInstaller (*onefolder*), UI → *flutter build windows*.
5. **Package**: Installer (Inno Setup/MSIX/ WiX).
6. **Sign**: Kod imzalama sertifikasıyla EXE/MSI imzalanır.
7. **Publish**: Artifakt yükleme; *beta*/*stable* kanallarına dağıtım.
8. **Smoke**: Otomatik yerel kurulum + *GET /status* sağlığı.

18.3 Gizli Bilgiler ve Anahtarlar

* Action secret’larında sertifika PFX, imza parolası, opsiyonel *API\_KEY*.
* *requirements.txt* **versiyon pinli**; *pip hash-checking*/*pip-tools* önerilir.

18.4 Veritabanı Göçleri (SQLite)

* **DDL göç dosyaları**: *migrations/0001\_init.sql*, *0002\_add\_metrics.sql*…
* **Runner**: Core boot’ta *schema\_version* kontrolü ve sıralı uygulama.
* **Yedek**: Göç öncesi *.backup* alınır.

18.5 Release Kanalları

* **Beta**: Telemetry **yalnız yerel** (dosyaya), kullanıcı onaylı A/B bayrakları.
* **Stable**: İmza zorunlu, migration testleri geçmiş.

18.6 Çıkış Kriterleri

* Test kapsamı ≥ %70 (core). Kritik modüller (risk/executor/connector/worker) için ≥ %85 satır.
* Lint sıfır hata; *mypy --strict* temiz.
* Smoke + kurulum/çalıştırma otomasyonu **yeşil**.

18.7 Kabul Kriterleri (Parça 18)

* CI/CD aşamaları dokümante, script’ler repo içinde.
* İmzalı paket üretimi ve artifakt yayın akışı çalışır.
* Migration mekanizması ve yedek politikası net.

Parça 19 — Paketleme & Kurulum (Installer + Servis + Updater)

19.1 Paketleme Seçenekleri

* **Core (Python)**: PyInstaller *onefolder*; servis runner ayrı EXE.
* **UI (Flutter)**: *flutter build windows* çıktısı.
* **Tek kurulum**: Inno Setup veya WiX ile **UI + Core** birlikte kurulur.

19.2 Windows Servis Kurulumu

* Seçenek A: *pywin32* ile native servis.
* Seçenek B: *nssm* (Non‑Sucking Service Manager) ile EXE’yi servis olarak sar.
* Servis **LocalService** yetkisiyle, yalnız loopback portlarını dinler.

19.3 İlk Çalıştırma Sihirbazı

* Profil dizinleri (*profiles/accX/*), *data/* ve *logs/* oluşturulur.
* DPAPI/keyring testi; parola ekleme adımı.
* Firewall kuralı (loopback dışında engel yok; istenirse dış erişim kapalı kalır).

19.4 Kod İmzalama & Güvenlik

* EXE/MSI **kod‑imzalı**; SmartScreen uyarılarını azaltır.
* Kurulum dosyası **SHA‑256** özeti yayınlanır.

19.5 Güncelleme & Geri Alma

* **Updater**: Versiyon kontrolü (ops. çevrimdışı); indir → doğrula → **kesintisiz** servis restart.
* **Rollback**: Son sürüm yedeği saklanır; başarısız güncellemede otomatik geri dönüş.

19.6 Kaldırma/Onarım

* Uninstall: Servis durdurulur, dosyalar kaldırılır, isteğe bağlı *data/* ve *logs/* korunur.
* Repair: Eksik dosyaları yeniden yazar; config ve veri **dokunulmaz**.

19.7 Kabul Kriterleri (Parça 19)

* Kurulum tek adımda UI+Core’u kurar, servis başlatır.
* Kod imzalama ve SHA‑256 doğrulaması mevcut.
* Updater ile sürüm yükseltme ve rollback çalışır.

Parça 20 — QA Test Matrisi & Otomasyon

20.1 Kapsam

* **Fonksiyonel**: Çoklu hesap, permit/eşik, risk guard, concurrency, idempotent emir.
* **Performans**: TF=1’de 50+ aktif worker; latency p90 bütçesi.
* **Güvenlik**: PII redaksiyonu, DPAPI, TLS, loopback, rate‑limit.
* **Uyumluluk**: TOS ihlali yok, anti‑bot atlatma yok.
* **Kullanılabilirlik**: Kill‑switch, onboarding, ayarların bulunabilirliği.

20.2 Örnek Testler (özet tablo)

* Çoklu hesap izolasyonu → token/cookie karışmıyor.
* Concurrency kuralı → aynı *(acc,prod,tf)* çift emir oluşmuyor.
* Daily loss / consec loss trip → otomatik durdur.
* Idempotent retry → tek finansal etki.
* Paper→Gerçek geçişte onay akışı.
* Loglarda PII maskesi.

20.3 Otomasyon

* *pytest* entegrasyon senaryoları (MockConnector).
* E2E: Core başlat + UI headless → *GET /status* / WS smoke.
* Flutter *integration\_test* ile kritik akışlar (onboarding, ayar, başlat/durdur).

20.4 Kabul Kriterleri (Parça 20)

* QA matrisi tanımlı, otomasyon senaryoları koşuyor.
* Performans ve güvenlik testleri raporlanır; p90 latency hedefi içinde.

Parça 21 — 4 Haftalık MVP Sprint Planı (Windows + 4 Hesap)

21.1 Sprint‑0 (Hazırlık, 2–3 gün)

* Repo, temel iskelet (core/api, connector/mock, indicators/basic), CI smoke.
* Config şeması ve örnek *config.json*.

1. Sprint (Hafta 1)

* **Core**: Storage (DDL+API), Indicators (Parça 7), MockConnector, Executor (iskele).
* **UI**: Proje iskeleti, Dashboard kabası, */status* entegrasyonu.
* **Çıktı**: Tek hesap, TF=1, paper akışı uçtan uca.

2. Sprint (Hafta 2)

* **Core**: Worker/Scheduler, Ensemble (weighted), Risk Engine (fixed lot), WS metrics.
* **UI**: Hesaplar, Ürün/TF ayar ekranları, canlı metrik akışı.
* **Çıktı**: 2 hesap paralel, permit/eşik/guardrails çalışıyor.

3. Sprint (Hafta 3)

* **Core**: Gelişmiş indikatörler (Parça 8), 3–4 provider, backtest iskeleti.
* **UI**: Strateji kataloğu, grafik paneli (mum+RSI/MACD), log görüntüleyici.
* **Çıktı**: Backtest raporu (özet metrikler), 4 hesap paralellik testi.

4. Sprint (Hafta 4)

* **Core**: Idempotency tamamlama, reconcile, telemetry/alerts, QA otomasyon.
* **UI**: Onboarding sihirbazı, risk sayfası, kill‑switch; paketleme + imza + installer.
* **Çıktı**: Windows kurulum paketi (MVP). Paper modda 1–2 günlük deneme koşusu.

21.2 Başarı Ölçütleri

* Paper modda **stabil** 24s+ uptime; crash‑free.
* Latency p90 < 2.5s, concurrency ihlali yok.
* UI’dan tüm kritik ayarlar yönetilebilir.

21.3 Riskler & Azaltma

* **API belirsizliği** → Mock/abstraction ile ilerle, gerçek uçlar yalnız izinliyse.
* **Payout veri erişimi** → Yoksa sabit/parametrik; UI’da uyarı.
* **Performans** → Profil + kuyruk/back‑pressure ayarı, TF=1 iş yükü dağıtımı.

21.4 Done Definition (MVP)

* Kurulum → Çalıştır → Paper akışında uçtan uca işlem; raporlama ve loglar eksiksiz.
* 4 hesap paralel; izin/guardrail/kill‑switch doğrulanmış.
* Dokümantasyon (kullanım + config) tamamlanmış.

Parça 24 — Operasyonel Runbook (Günlük İşletim, Olay Yönetimi, Bakım)

Amaç: MoonLight’ın Windows’ta **günlük kullanım**, **olay (incident) yönetimi**, **bakım** ve **geri kazanım** adımlarını tek bir pratik el kitabında toplamak. Bu runbook, 4 hesap/çoklu ürün‑TF senaryosunda operatöre adım adım yol gösterir.

24.1 Günlük Kontroller (Sabah Rutini ~3 dk)

1. **Servis durumu**: UI → Sağ üst **Core bağlandı** rozeti (yeşil). Gerekirse *Services.msc* → *MoonLight Core* → Running.
2. **Saat & NTP**: UI → Ayarlar → Sistem → “Saat kayması” gösterimi **≤ 250 ms** olmalı (Parça 12/17).
3. **Hesaplar**: *Hesaplar* sayfası → 4 hesap **Connected**; gerekiyorsa giriş/OTP.
4. **Uyarılar**: *Uyarı Merkezi* → aktif kırmızı yok; varsa **incele → düzelt → kapat**.
5. **Payout pencereleri**: *Ürün/TF* → permit aralığına uygun (ör. 89–93). Uzun süredir dışındaysa çalışma **PAUSED**.
6. **Guardrails**: Günlük kayıp sayacı sıfır, *cooldown\_active=false*.
7. **Disk & Yedek**: *C:\ProgramData\MoonLight\* altında boş alan **> 2 GB**; son yedek tarihine bak.

24.2 Başlat / Durdur / Bakım Penceresi

* **Başlat**: UI → *Workers* panelinden ürün/TF bazında **Start**.
* **Durdur (yumuşak)**: **Pause** (yeni giriş yok; açık işlemler kapanır).
* **Anında durdur (Kill‑Switch)**: Üst çubuk kırmızı düğme → tüm hesaplarda **acil stop**.
* **Bakım modu**: UI → Ayarlar → *Bakım Penceresi* → “Başlat”: core **PAUSED**; güncelleme/yedek güvenli.

24.3 Olay (Incident) Tipleri & Playbook’lar

Aşağıdaki tabloda **belirti → tanı → çözüm** akışı verilir. Kod/anahtarlar Parça 17 metrik/log yapısına uygundur.

24.3.1 AUTH/OTP Gerekiyor

* **Belirti**: Hesap durumu *otp\_required*, *Uyarı Merkezi*: *AUTH\_OTP\_REQUIRED* (WARN).
* **Tanı**: *security.log* satırı; UI’da OTP alanı açılır.
* **Çözüm**: OTP gir → **Trust this device** tik (varsa). Başarısızsa hesabı **Disconnect→Reconnect**.

24.3.2 429 (Rate‑Limit) / 5xx Fırtınası

* **Belirti**: Uyarı *RATE\_LIMIT\_HIT*, *connector\_errors\_total{THROTTLE}* artıyor.
* **Tanı**: *connector.log* p90/p99 latency artışı; *ws\_heartbeat\_lag\_ms* yükselir.
* **Çözüm**: Core otomatik **exponential backoff** uygular. Uzunsa **Workers→Pause**; dakika sonra **Resume**. Gündelikte **rps\_account** eşiğini düşür (Ayarlar→Motor).

24.3.3 Yüksek Gecikme

* **Belirti**: *latency\_ms\_p90 > abort\_ms·0.7*, kırmızı uyarı *LATENCY\_HIGH*.
* **Tanı**: CPU/RAM yüksek, disk yoğun I/O, ağ gecikmesi.
* **Çözüm**: Aktif worker sayısını azalt; TF=1’i geçici **Pause** et; *tick\_ms* ve *debounce* arttır; gereksiz UI grafiklerini kapat (performans modu).

24.3.4 Loss Streak / Günlük Limit Trip

* **Belirti**: *consec\_losses* uyarısı, *CIRCUIT\_BREAKER\_TRIP* (ERROR).
* **Tanı**: *Orders* tablosu son 20 işlem; *Products* panelinde p̂ ve payout logları.
* **Çözüm**: Sistem **auto‑stop**; profili gözden geçir (win\_threshold ↑, permit aralığı sıkılaştır). En az 30 dk **cooldown**; yeniden başlatmadan önce backtest/paper ile doğrula.

24.3.5 Disk Dolu / DB Kilidi

* **Belirti**: Uyarı *DISK\_FULL* veya DB hata mesajı; yazma gecikmeleri.
* **Tanı**: *logs/* büyümüş; *metrics\_raw* retention çalışmamış.
* **Çözüm**: *Bakım modu* → “Yedek & Temizle” sihirbazı → retention job’u çalıştır; gerekirse eski logları arşivle/sil. Çok nadir kilitte servis **Restart**.

24.3.6 Saat Kayması (Clock Skew)

* **Belirti**: TF hizasızlığı, tetik kaçırma, *heartbeat\_lag\_ms* sapması.
* **Tanı**: UI saat farkı paneli > 250 ms.
* **Çözüm**: Windows “Saat otomatik ayarla” + NTP eşitliği; düzelmeden **TF=1** geçici durdur.

24.3.7 Çökme / Yeniden Başlatma

* **Belirti**: Servis durdu/başladı; Windows Event Log kaydı.
* **Tanı**: *system.log* son satırlar; *trade\_events* → *RECONCILE\_START*.
* **Çözüm**: Core otomatik **reconcile** eder (Parça 11/12). Çifte emir yok; post‑mortem için log paketini arşivle.

24.4 Reconcile Akışı (Özet)

1. Core açılışta *orders* içinde sonucu olmayan kayıtları listeler.
2. Connector’dan durum sorgusu; eşleşenler *results*’a yazılır.
3. Süre aşımı/ bilinmeyenler *status='abort'* olarak kapanır; *alerts*’a bilgi düşer.
4. UI *Orders* sayfasında “reconcile edildi” rozeti görülür.

24.5 Yedekleme & Geri Yükleme (Adım Adım)

* **Anlık Yedek**: UI → *Veri* → **Yedek Al** → *VACUUM INTO backup.sqlite* → *backups/* klasörü.
* **Geri Yükle**: *Bakım modu* → **Durdur** → yedeği seç → doğrulama → **Yükle** → **Başlat**.
* **Otomatik**: Günde 1 kez, saat 03:00 (config ile değiştirilebilir).

24.6 Performans Ayarları (Hızlı Rehber)

* **Workers**: *(hesap × ürün × TF)* eşzamanlılığı düşür.
* **Tick & Jitter**: *tick\_ms ↑*, *jitter\_ms* ekle (çekişmeyi azaltır).
* **WS Yayın**: UI canlı log/metrics örneklemesini kıs.
* **Donanım**: SSD şart; 4C/8T ve 16 GB RAM öneri (yük testi hedefleri Parça 20.5).

24.7 Değişiklik Yönetimi (Change Management)

* **Önce Paper** → küçük pencerede **canlı**.
* *PUT /config* sonrası UI **Diff**’i ve *CONFIG\_APPLIED* olayını kontrol et.
* Profil sürümlemesi: her önemli değişiklik → *profiles/.../history/* patch.

24.8 Sık Görülen Semptomlar → Kontrol Listeleri

* **“10 dk’dır işlem açmıyor”**: Permit dışı mı? p̂ eşiği çok mu yüksek? Concurrency kilidi mi açık? *trade\_skips* nedenlerine bak.
* **“Hep permit yüzünden skip”**: Ürün payout düşmüş olabilir → farklı ürün/TF’ye geç veya permit aralığını ayarla.
* **“Win‑rate düştü”**: Kalibrasyon bayat → Ensemble kalibrasyonu yenile (Parça 9/16). Strateji/parametreleri gözden geçir.

24.9 Güncelleme / Rollback Prosedürü

1. *Bakım modu* → **Yedek al**.
2. UI → **Güncellemeyi kontrol et** → sürümü indir/imza doğrula.
3. **Uygula**; smoke test (Dashboard, Orders, Logs).
4. Sorun varsa **Rollback**: Önceki sürüm paketi ve DB yedeği ile geri dön.

24.10 Hata Kodları ve Log Anahtarları (Kısa Tablo)

| Kod | Anlam | Log Kanalı |
| --- | --- | --- |
| *AUTH\_OTP\_REQUIRED* | OTP girişi gerekiyor | *security* |
| *RATE\_LIMIT\_HIT* | 429 limiti | *connector* |
| *LATENCY\_HIGH* | p90 latency yüksek | *trade/system* |
| *CIRCUIT\_BREAKER\_TRIP* | Guardrail tetikledi | *risk/system* |
| *RECONCILE\_START/END* | Yeniden eşleme döngüsü | *system* |
| *PLACE\_TIMEOUT* | Emir onayı gecikti | *trade/connector* |

24.11 Kabul Kriterleri (Parça 24)

* Günlük kontrol listesi, olay playbook’ları ve bakım/yedek prosedürü **tam**.
* Reconcile, güncelleme/rollback ve performans ayar rehberi **uygulanabilir**.
* Hata kodları tablo halinde; UI öğeleriyle eşleşiyor.

Parça 25 — Güvenlik & Uyumluluk Kontrol Listesi (Secrets, Yetkiler, Log PII, Kill‑Switch, TOS)

Amaç: MoonLight’ın **Windows 10/11** üzerinde güvenli çalışmasını, kullanıcı verilerinin korunmasını ve **platform kuralları/TOS** ile uyumlu bir kullanımını garanti etmek. Bu bölüm; mimari ilkeler, uygulanacak kontroller, test planı ve düzenli denetim listelerini içerir.

25.1 Kapsam & Hedefler

* **Kimlik bilgisi yönetimi**: Parola/tokenlerin güvenli saklanması, hiçbir zaman düz metin yok.
* **Yetki sınırları**: Servis ve dosya izinleri, en az ayrıcalık (PoLP).
* **Veri & log koruması**: PII maskesi, şifreleme, saklama süresi (retention).
* **Ağ & API güvenliği**: Loopback, opsiyonel TLS, rate‑limit, isteğe özgü kimlik.
* **Operasyonel güvenlik**: Kill‑switch, guardrails, değişiklik/audit izleri.
* **Uyumluluk**: Yalnız **resmî/izinli API**; TOS ihlali veya bot algılama atlatma **yok**.

25.2 Kimlik Bilgisi & Sırlar (Secrets)

* **Saklama yeri**: Windows **Credential Manager** + **DPAPI**; dosyada saklama **yok**.
  + UI yalnız **başvuru adı** (ref) tutar: *ML\_acc1\_session*, *ML\_accX\_refresh*.
* **Parola/OTP**: OTP **anlık** alınır, depolanmaz. Parola mümkünse hiç istenmez; **token/sesyon** modeline öncelik.
* **Döndürme**: Session/refresh token’lar **90 günde** bir otomatik yenileme hatırlatıcısı (Ayarlar → Güvenlik).
* **İzleme**: Secret erişim denemeleri *security.log*’a olay olarak yazılır (maskeli).
* **Girdi Maskesi**: UI form alanları *type=password*; panoya kopyalama kapalı (ops.).

25.3 Yetkiler, Dosyalar & Servis Sertleştirme

* **Servis hesabı**: *LocalService* (veya özel kısıtlı hesap). *LocalSystem* **kullanma**.
* **ACL**: *C:\ProgramData\MoonLight\* yalnız **Administrators** + servis hesabı.
* **Çalışma yüzeyi**: REST/WS **yalnız** *127.0.0.1*. Uzak bağlanma **kapalı**.
* **Kod imzalama**: Core EXE ve UI MSIX **imzalı** (Parça 19).
* **Exploit azaltma**: ASLR/DEP varsayılan; Defender hariç bırakma **eklenmez** (gerekmedikçe).
* **Başlangıç**: *Automatic (Delayed Start)* + crash recovery (1/5/5 dk).

25.4 Veri Koruma & PII

* **Şifreleme (ops.)**: DB için **DPAPI‑Protected** anahtar ile sütun seviyesinde şifreleme (örn. *username\_mask* zaten maskeli; gerçek kimlik **yok**).
* **Log PII**: Kullanıcı adı/e‑posta **maskeli** (*ab\*\*\*@d\*\*.com*). Parolalar/loglarda **asla** yok.
* **Retention**: *trade\_events*: 30–90g, *metrics\_raw*: 7–14g, *orders/results*: uzun süreli (Parça 22).
* **Dışa Aktarım**: CSV/ZIP dışa aktarımda PII kolonları **hariç** tutulur; dosyalar kullanıcı onayıyla şifrelenebilir.

25.5 Ağ & API Güvenliği

* **Loopback**: *http/ws* bağlayıcıları *127.0.0.1*’e sabit.
* **TLS (ops.)**: Yerel olmasına rağmen, **mTLS** seçeneği sunulabilir (gelişmiş).
* **Auth**: Yerel API isteklerine **API‑Key** opsiyonu (Parça 15).
* **Rate‑limit**: *POST /orders* ve *PUT /config* için kaynak başı dakika sınırları; 429 yanıtı + bekle.
* **İstek kimliği**: *X‑Request‑ID* zorunlu; idempotent *client\_req\_id* (Parça 11).
* **Input doğrulama**: Pydantic şemaları; uzunluk/sayı aralıkları; *reason* alanları whitelist.

25.6 Gerçek Hesap Güvenliği & Guardrails

* **Kill‑Switch**: Tek tık tüm hesaplarda **PAUSE**; tekrar açma onaylı.
* **Limitler**: Günlük zarar limiti, ardışık zarar limiti, tek işlem tavanı (UI görünür).
* **Demo→Gerçek geçiş**: Varsayılan **kilitli**; kullanıcı **bilinçli onay** (checkbox + uyarı) olmadan açılmaz.
* **Permit & eşik**: Payout penceresi dışı **asla** giriş yok.
* **Öğrenme**: Canlı sonuçlar *yalnız model kalibrasyonu* içindir; özerk agresif öğrenme **yok** (offline onaylı).
* **İşlem sıklığı**: Ürün/TF başına **concurrency=1**; ardışık hızlı girişler için minimum aralık (cooldown).

25.7 Uyumluluk & TOS

* **Yalnız izinli kullanım**: **Resmî/izinli API** üzerinden erişim; sayfa otomasyonu, scraping, güvenlik önlemi atlatma, kimlik gizleme vb. **yasaktır** ve desteklenmez.
* **Hesap sahipliği**: Yalnız **kullanıcının kendine ait** hesaplar; üçüncü şahıs adına yetkisiz kullanım yok.
* **2FA/OTP**: Kullanıcı manuel girer; saklanmaz.
* **Şeffaflık**: UI’da “Bu yazılım üçüncü taraf platform kurallarına tabidir” bildirimi.
* **Bölge hukuku**: Finansal regülasyonlar ülkeye göre değişir; kullanıcı kendi yükümlülüklerini kabul eder.

25.8 Tedarik Zinciri (Supply Chain)

* **Bağımlılık sabitleme**: *requirements.txt* + **hash** kilidi; *pip-tools*/*uv* önerilir.
* **Güvenlik taraması**: *pip-audit*/*safety*, *bandit*, lisans denetimi; **SBOM** (CycloneDX).
* **İmza & checksum**: Installer ve çekirdek için SHA‑256; imzalı yayın.
* **Üçüncü taraf plugin**: *.mlprov* paketleri yalnız imzalı/whitelist kaynaktan yüklenir.

25.9 Tehdit Modellemesi (STRIDE özeti)

* **Spoofing**: API‑Key (ops.), loopback zorunlu, Windows kullanıcı ayrımı.
* **Tampering**: Kod imzası, audit log, config diff/rollback; DB için bütünlük kontrolleri.
* **Repudiation**: *config\_audit*, *trade\_events*, *orders/results* ile iz sürülebilirlik.
* **Information Disclosure**: PII maskesi, yetkili ACL, opsiyonel şifreleme.
* **Denial of Service**: Rate‑limit, backoff, circuit‑breaker, watchdog.
* **Elevation of Privilege**: Servis hesabı kısıtlı; dosya izinleri; uzak API yok.

25.10 Güvenlik Test Planı

* **SAST**: *ruff*/*bandit*/*mypy --strict* → her PR.
* **DAST (lokal)**: REST/WS uçlarına fuzz (şema tabanlı); hatalı inputlar 4xx.
* **Secret sızması**: *gitleaks* veya eşdeğeri; repo ve loglarda gizli yok.
* **Penetrasyon (lokal)**: Yetkisiz port/erişim testi → başarısız olmalı.
* **E2E**: OTP akışında PII maskeleme; kill‑switch gecikmesi < 500 ms.
* **TOS guard**: TOS’a aykırı fonksiyonlar (ör. RPA) kod tabanında **bulunmamalı**.

25.11 Olay (Incident) & İhlal Yönetimi

* **Sınıflandırma**: Seviyeler — INFO/WARN/ERROR/CRIT.
* **Acil durum**: Kill‑switch, servis *Pause*, ağ erişimi (ops. kes).
* **Toplama**: *logs/*, *trade\_events*, *config\_audit* paketlenir; PII maskeli.
* **Kök neden**: Post‑mortem şablonu (zaman çizelgesi, etkiler, düzeltici eylemler).
* **Bildirim**: Üçüncü taraf platforma/hizmete ihlal varsa kullanıcı sorumluluğunda.

25.12 Denetim Listeleri (Checklists)

**Günlük**  
- [ ] Servis Running, saat kayması ≤ 250 ms  
- [ ] Uyarı kırmızı yok; permit aralığı uygun  
- [ ] Günlük zarar/seri limitleri sıfırlandı

**Sürüm Öncesi**  
- [ ] Kod imzalı, checksum yayınlandı  
- [ ] SBOM ve güvenlik taramaları yeşil  
- [ ] Migrasyon + yedek doğrulandı

**Üç Aylık**  
- [ ] Token döndürme  
- [ ] Retention politikası gözden geçirildi  
- [ ] Bağımlılık yükseltmeleri ve lisans kontrolleri

25.13 Kabul Kriterleri (Parça 25)

* Secrets yönetimi **Credential Manager + DPAPI** ile uygulanmış; parolalar/loglarda yok.
* Servis yetkileri kısıtlı; dosya izinleri sıkı.
* PII maskesi, retention ve dışa aktarımlar güvenli.
* Loopback + opsiyonel API‑Key/TLS uygulanabilir.
* Kill‑switch/guardrails görünür ve testlerle doğrulanmış.
* **TOS uyumu**: Resmî/izinli API dışında akışlar **yok**; RPA/bot atlatma **yok**.
* Güvenlik test planı ve periyodik denetim listeleri hazır.

Parça 26 — Canlıya Alma (Go/No‑Go) & Rollback Planı

Amaç: MoonLight’ın **paper → live** geçişini kontrollü, ölçülebilir ve geri dönüşü garanti eden bir planla yönetmek. Bu bölüm; **Go/No‑Go kapıları**, canlıya alma **adım adım prosedürü**, **erken yaşam izlemesi (ELM)** ve **rollback** karar/uygulama adımlarını içerir.

26.1 Kapsam

* Hedef ortam: **Windows 10/11**, 4 hesap, çoklu ürün/TF (1/5/15).
* Sürüm: *core x.y.z*, *ui a.b.c* (stable etiketi).
* Mod: Önce **paper**, ardından **kademeli live**.

26.2 Go/No‑Go Kapıları (Zorunlu Eşikler)

**Teknik**  
- CI **yeşil**, birim/entegrasyon kapsamı ≥ %80 (çekirdek), UI ≥ %60.  
- E2E senaryoların %100’ü geçti (Parça 20.2).  
- Performans: TF=1’de provider p95 < **3 ms**, place round‑trip p90 < **400 ms**.  
- Kararlılık: 24s **soak** testinde crash‑free, bellek sızıntısı yok.  
- Güvenlik: Loopback, PII maske, secrets/DPAPI testi **PASS** (Parça 25).

**Model/Strateji**  
- Backtest: Son 3 ay, ürün/TF bazında p̂ kalibrasyonu **ECE ≤ 0.05**, **Brier ≤ 0.22**.  
- Forward (paper): En az **5 iş günü**; guardrail ihlali **yok**; p̂>eşik girişlerde **win‑rate ≥ hedef** (kullanıcı hedefi).

**Operasyonel**  
- Runbook (Parça 24) okunmuş; kill‑switch test edilmiş (< 500 ms tepki).  
- Yedek/rollback denemesi **başarılı** (DB *VACUUM INTO* + sürüm geri dönüş).  
- TOS/Uyumluluk kontrol listesi imzalı (Parça 25).

Her madde **PASS** değilse → **No‑Go** ve düzeltme/TODO listesi çıkar.

26.3 Preflight (Canlı Öncesi)

1. **Konfig dondur**: *config.lock* oluştur; değişiklikler için onay gerektir.
2. **Yedek**: *db-<ts>.sqlite* + *profiles.zip*.
3. **Log seviyesi**: *INFO*; **DEBUG** yalnızca hedefli (noise azalt).
4. **Alert eşikleri**: *LOSS\_STREAK\_NEAR=3*, *DAILY\_LOSS\_NEAR=80%* (kullanıcıya uyarı).
5. **Auto‑update kapalı**: Canlı pencerede sürüm değişme.
6. **Saat/NTP**: Sapma ≤ 250 ms.
7. **Kapasite**: Aktif *(acc×prod×tf)* hedefinin %70’i ile başla.

26.4 Kademeli Canlıya Alma (Canary)

* **Aşama 0** (0–2 saat): 1 hesap, 1 ürün, **TF=5**. Permit 89–93, *win\_threshold=0.74*.
* **Aşama 1** (2–6 saat): Aynı hesap, 2. ürün; **TF=1** hâlâ kapalı.
* **Aşama 2** (Gün 1 sonu): 2–3 hesap, TF=5/15; guardrail ihlali **yoksa** TF=1 küçük hacimle aç.
* **Aşama 3** (Gün 2–3): Hedef topolojiye yükselt (4 hesap). CPU/RAM/latency p90 sınırları içinde kalmalı.

**Yükseltme kriteri**: Son 2 saat **kırmızı uyarı yok** + p̂ kalibrasyon sapması ≤ 0.02 + skip dağılımında *permit* ana neden değil.  
**Duraklatma kriteri**: Ardı ardına **5 kayıp** veya günlük zarar **%60**’ı geçti → **pause & inceleme**.

26.5 Canlıya Alma — Adım Adım

1. *Bakım modu*: **Pause**; konfig son kontrol.
2. Modu **live** yap; yalnız **seçilen hesap/ürün/TF**’leri enable et.
3. **Workers Start** (Aşama 0 kapsamı).
4. İlk 3–5 işlemi **yakın izle**: place/confirm latency, payout snapshot, idempotency.
5. **Telemetry**: p50/p90 latency, skip nedenleri, win/loss akışı; UI grafiklerini basit tut.
6. Aşama kriteri PASS ise bir sonraki aşamaya geç; değilse **duraklat & analiz**.

26.6 Erken Yaşam İzlemesi (ELM)

* **Süre**: İlk 48 saat.
* **Ana paneller**: latency (place/confirm), skip dağılımı, p̂ güvenilirliği (reliability diagram/ECE), guardrail sayacı, payout trendi.
* **Alarm kuralları**:
  + *CIRCUIT\_BREAKER\_TRIP* → anında **Kill‑Switch** öner.
  + *LATENCY\_HIGH* (p90>400ms, 10 dk) → TF=1 **pause**.
  + *PERMIT\_DOMINANT* (skip> %70 permit) → ürün/TF değişikliği öner.

26.7 Rollback Tetikleyicileri (Karar Matrisi)

| Olay | Şiddet | Eşik/Belirti | Aksiyon |
| --- | --- | --- | --- |
| Crash/freeze | Kritik | Servis durur, reconcile başarısız | **Hemen rollback** + log toplama |
| Yanlış emir | Kritik | Yan ürün/TF/tutar | **Kill‑Switch** + **full rollback** |
| Guardrail trip | Yüksek | Günlük/seri limit | **Pause**, param gözden geçir; tekrarda rollback |
| Latency fırtınası | Orta | p90>600ms, 30 dk | TF=1 pause → düzelmezse rollback |
| 429/5xx fırtınası | Orta | 15 dk sürekli | Backoff → sürmezse rollback |
| Kalibrasyon bozulması | Orta | ECE>0.08 | Paper’a dönüş + yeniden kalibrasyon |

26.8 Rollback Prosedürleri

**A — Çalışma Zamanı Rollback (Konfig düzeyinde)**  
1) **Pause All**.  
2) *profiles/history*’den son **stabil** patch’e geri dön.  
3) **Hot‑reload**; *CONFIG\_APPLIED* olayını doğrula.  
4) Aşama 0 topolojisine geri dön; ELM sür.

**B — Sürüm Rollback (Core/UI)**  
1) *Bakım modu* → **Yedek al**.  
2) Bir önceki **stable** Installer/EXE ile geri yükle (imza/sha256 kontrol).  
3) Core servis restart → **/status** kontrol.  
4) E2E smoke: login, permit/threshold, tek işlem kuralı, log/metrics akışı.  
5) Canary (Aşama 0) ile yeniden başlat.

**C — Veri Rollback (Son çare)**  
1) Core **Pause**.  
2) *VACUUM INTO backup.sqlite* ile alınmış en son yedeği **geri yükle**.  
3) *schema\_version* uyumunu doğrula; gerekirse migrator çalıştır.  
4) Sistem **readiness** yeşil ise Aşama 0’dan devam.

26.9 İletişim & Audit

* **Değişiklik kaydı**: *config\_audit* ve *trade\_events*’te *Go/No‑Go* kararları.
* **Olay raporu**: Rollback sonrası **post‑mortem** (özet, kök neden, aksiyonlar).
* **Versiyon etiketi**: UI “Hakkında” ekranında görünür.

26.10 “Go” Sonrası Ölçme & Sertleştirme

* İlk hafta:
  + **Parametre sertleştirme**: permit aralığı ve *win\_threshold* ince ayar.
  + **Kalibrasyon yenileme** (Platt a/b).
  + **Performans**: TF=1 iş yükü ve örnekleme oranlarını optimize et.
* Haftalık rapor: Win‑rate, ECE/Brier, latency, skip dağılımı, guardrail tetikleri.

26.11 Kabul Kriterleri (Parça 26)

* Go/No‑Go eşikleri **net** ve ölçülebilir.
* Canary canlıya alma planı ve **ELM** metrikleri tanımlı.
* Rollback **karar matrisi** ve **üç seviye** prosedür (konfig/sürüm/veri) hazır.
* İletişim/audit ve post‑mortem akışı yazıldı.

Parça 27 — Backtest & Forward Test Rapor Şablonları (Metrikler, Grafikler, Çıktılar)

Amaç: Strateji/ensemble değişikliklerinin **kanıta dayalı** olarak değerlendirilmesi için **standartlaştırılmış** rapor şablonları sağlamak; geriye dönük (backtest) ve ileriye dönük (forward/paper & live) performansı **aynı dilde** karşılaştırmak.

27.1 Hedefler

* **Tek format**: Backtest ve forward sonuçları aynı veri şemasına dökülür.
* **Karşılaştırılabilirlik**: Ürün/TF/hesap ve tarih aralığı bazlı **elma‑elma** kıyas.
* **Güvenilirlik ölçümü**: Kalibrasyon (reliability/ECE), p̂→win ilişkisinin korunumu.
* **Risk perspektifi**: Drawdown, kâr beklentisi (expectancy), Kelly‑lite güvenli bölge.

27.2 Çıkış Dizini & Dosyalar

C:\ProgramData\MoonLight\reports\ backtest\ <run\_id>\ run.json # meta ve param özet trades.parquet # her işlem satırı metrics.json # toplu metrikler curves.parquet # equity/Drawdown/WinProb vs time figs\\*.png # grafikler (matplotlib) report.html # Jinja2 ile oluşturulmuş rapor forward\ <run\_id>\ (paper/live) ... (aynı şema)

27.3 Veri Şemaları

**27.3.1 *run.json***

{ "id": "bt\_2025-10-07\_103000", "mode": "backtest", "env": "paper", "period": {"from": "2025-07-01", "to": "2025-09-30"}, "universe": [{"product":"EURUSD","tf":1},{"product":"XAUUSD","tf":5}], "profiles": ["scalper\_1m","swing\_5m"], "params": {"ensemble": {"c\_min": 0.72}}, "commit": "a1b2c3d", "generated\_at": 1696665000000}

**27.3.2 *trades.parquet*** (kolonlar) - *ts\_open\_ms*, *ts\_close\_ms*, *product*, *tf*, *acc* (forward’da), *dir* (+1/−1), *amount*, *payout\_pct*, *p\_hat* (ensemble olasılığı), *status* (*win/lose/push/abort*), *pnl*, *latency\_ms*, *provider\_ids* (liste), *features\_json* (ops.).

**27.3.3 *metrics.json*** (örnek)

{ "global": {"trades": 1240, "win\_rate": 0.714, "pf": 1.42, "exp": 0.06, "max\_dd": -0.12, "sharpe": 1.1, "sortino": 1.6}, "by\_product": {"EURUSD": {"trades": 800, "win\_rate": 0.72, "pf": 1.45}}, "by\_tf": {"1": {"trades": 900, "win\_rate": 0.71}, "5": {"trades": 340, "win\_rate": 0.72}}, "calibration": {"ece": 0.038, "brier": 0.205}, "latency": {"p50": 180, "p90": 340}}

27.4 Metrik Tanımları (özet)

* **Win‑rate**: *wins/(wins+losses)*; *push/abort* hariç.
* **Profit Factor (PF)**: *gross\_profit / gross\_loss*.
* **Expectancy (Exp)**: *E[pnl/amount]* (işlem başına ortalama R).
* **Max Drawdown (MaxDD)**: normalized equity serisinden en büyük tepe‑dip.
* **Sharpe/Sortino**: Saatlik veya günlük getiri serisine; risk‑free≈0 varsayımı.
* **Kelly‑lite öneri**: *kelly = win\*p - (1-win) / R*; **lite** = *kelly \* 0.2* (tavan: 2%).
* **Calibration**: Reliability diagram + **ECE** (Expected Calibration Error) + **Brier score**.
* **Lift**: En yüksek p̂ dilimlerinin taban win‑rate’e göre iyileşmesi.
* **Latency p50/p90**: place→confirm gecikmesi (forward’da).

27.5 Segmentasyonlar

* **Ürün/TF**: EURUSD:1, XAUUSD:5, BTCUSD:15…
* **Zaman**: Saat dilimi (market session), haftanın günü.
* **Payout bandı**: 93–91–89–… (permit etkisi).
* **Sinyal gücü**: p̂ dilimleri (0.50–0.55, …, 0.90–0.95).
* **Strateji grubu**: trend, mean‑revert, momentum…

27.6 Grafikler (Matplotlib)

* **Equity & Drawdown** (zamana karşı).
* **Win‑rate Rolling** (örn. 200 işlem penceresi).
* **Reliability Diagram** (p̂→gerçek win).
* **Distribution**: PnL histogramı, işlem süresi, payout.
* **Segment Bar/Heatmap**: ürün×TF, saat×win‑rate ısı haritası.
* **Latency CDF** (forward).

27.7 HTML Rapor Şablonu (Jinja2 iskeleti)

<!doctype html><**html**><**head**> <**meta** charset="utf-8"/> <**title**>MoonLight Rapor — {{ run.id }}</**title**> <**style**> body{**font-family**:Inter,Segoe UI,Arial;**margin**:24px} .kpi{**display**:flex;**gap**:16px;**flex-wrap**:wrap} .card{**padding**:12px 16px;**border**:1px solid #eee;**border-radius**:12px} h2{**margin-top**:28px} table{**border-collapse**:collapse} td,th{**border-bottom**:1px solid #eee;**padding**:6px 10px;**text-align**:right} th{**text-align**:left} .ok{**color**:#0a7} .bad{**color**:#c33} </**style**></**head**><**body**> <**h1**>MoonLight Rapor — {{ run.mode|upper }} — {{ run.period.from }} → {{ run.period.to }}</**h1**> <**div** class="kpi"> <**div** class="card">Win‑rate: <**b** class="{{ 'ok' if m.global.win\_rate>=0.7 else 'bad' }}">{{ (100\*m.global.win\_rate)|round(1) }}%</**b**></**div**> <**div** class="card">PF: <**b**>{{ m.global.pf|round(2) }}</**b**></**div**> <**div** class="card">Exp: <**b**>{{ m.global.exp|round(3) }} R</**b**></**div**> <**div** class="card">MaxDD: <**b**>{{ (100\*m.global.max\_dd)|round(1) }}%</**b**></**div**> <**div** class="card">ECE: <**b**>{{ m.calibration.ece|round(3) }}</**b**></**div**> <**div** class="card">Brier: <**b**>{{ m.calibration.brier|round(3) }}</**b**></**div**> </**div**> <**h2**>Grafikler</**h2**> <**img** src="figs/equity.png" width="800"/> <**img** src="figs/drawdown.png" width="800"/> <**img** src="figs/reliability.png" width="800"/> <**img** src="figs/winrate\_rolling.png" width="800"/> <**img** src="figs/heatmap\_product\_tf.png" width="800"/> <**h2**>Segment Metrikleri</**h2**> <**table**> <**tr**><**th**>Segment</**th**><**th**>Trades</**th**><**th**>Win%</**th**><**th**>PF</**th**><**th**>Exp (R)</**th**></**tr**> {% for seg in segments %} <**tr**><**td**>{{ seg.name }}</**td**><**td**>{{ seg.trades }}</**td**><**td**>{{ (100\*seg.win\_rate)|round(1) }}</**td**><**td**>{{ seg.pf|round(2) }}</**td**><**td**>{{ seg.exp|round(3) }}</**td**></**tr**> {% endfor %} </**table**> <**h2**>Kalibrasyon</**h2**> <**p**>Reliability ve ECE metrikleri; p̂→win uyumunun takibi. <**i**>Hedef: ECE ≤ 0.05, Brier ≤ 0.22</**i**>.</**p**> <**h2**>Ekler</**h2**> <**ul**> <**li**><**a** href="trades.parquet">İşlem satırları (Parquet)</**a**></**li**> <**li**><**a** href="metrics.json">Toplu metrikler</**a**></**li**> </**ul**></**body**></**html**>

27.8 Rapor Üretim Akışı

1. **Aggregate**: *trades.parquet* → segmentasyon & metrik hesapları (pandas).
2. **Figürler**: Matplotlib ile PNG (UI ile paylaşım kolay).
3. **Jinja2**: *run.json* + *metrics.json* + figürler → *report.html*.
4. **PDF** (ops.): Headless Chromium ile HTML→PDF dönüştürme.
5. **UI Entegrasyonu**: Raporlar panelinde listele, aç; “Dışa aktar” butonu.

27.9 Hesaplamalar (Python iskeletleri)

*# metrics.py (özet)***import** numpy **as** np, pandas **as** pd**def** win\_rate(df): d = df[df.status.isin(['win','lose'])] **return** (d.status.eq('win').mean())**def** profit\_factor(df): g = df.pnl.clip(lower=0).sum(); l = (-df.pnl.clip(upper=0)).sum() **return** np.inf **if** l==0 **else** g/max(1e-9,l)**def** expectancy\_R(df): d = df[df.status.isin(['win','lose'])] **return** ((d.pnl/d.amount).mean()) *# calibration***def** reliability\_bins(df, bins=np.arange(0.5, 1.01, 0.05)): d = df[df.status.isin(['win','lose'])] d = d.assign(bin=pd.cut(d.p\_hat, bins, include\_lowest=True)) grp = d.groupby('bin') **return** grp.status.eq('win').mean(), grp.p\_hat.mean()**def** ece(df, bins=np.arange(0.5, 1.01, 0.05)): obs, ph = reliability\_bins(df, bins) w = df[df.status.isin(['win','lose'])].groupby(pd.cut(df.p\_hat, bins)).size() **return** ((w \* (obs.values - ph.values).abs()).sum() / w.sum())

27.10 Forward İzleme (Drift & Sağlık)

* **Kayar pencere**: 100/200 işlem penceresinde win‑rate ve expectancy.
* **CUSUM/Page‑Hinkley**: Kalite düşüşünü erken yakalamak için sinyal (UI uyarısı).
* **p̂ sapması**: Eğitim/backtest p̂ dağılımı ↔ forward p̂ dağılımı **KL‑divergence** ile takip.
* **Kontrol sınırları**: Win‑rate için ±2σ bandı; çıkışta “yeniden kalibrasyon öner”.

27.11 Onay Kapıları (Rapor Bazlı)

* **Backtest → Paper**: PF ≥ 1.2, MaxDD ≥ −20%, ECE ≤ 0.06.
* **Paper → Live**: 5 iş günü; PF ≥ 1.1, MaxDD ≥ −25%, ECE ≤ 0.05, kill‑switch testi PASS.
* **Live Devam**: Haftalık rapor ECE ≤ 0.06; guardrail ihlali **yok**.

27.12 Test Planı

* Sentetik veri ile metrik doğrulama (bilinen win‑rate senaryoları).
* Reliability/ECE fonksiyonları için altın testleri.
* Segment metriklerinin toplamı global ile tutarlı.
* HTML üretimi ve figür yolları mevcut; açılan rapor 404 vermez.
* UI’dan rapor indirme/açma akışı.

27.13 Kabul Kriterleri (Parça 27)

* Backtest/forward için **tek şema** ve klasör yapısı tanımlandı.
* KPI’lar, segmentler ve grafik şablonları **hazır**.
* Jinja2 tabanlı HTML rapor ve opsiyonel PDF akışı açıklandı.
* Drift/kalibrasyon takibi ve onay kapıları **belirlendi**.

Parça 28 — Model Kalibrasyonu & Güven Puanı (Platt / Isotonic / ECE)

Amaç: Ensemble’ın ürettiği **ham skorları** gerçekçi **kazanma olasılığına** (p̂) çevirmek; **payout** ve **risk** koşullarına göre **dinamik eşik** belirlemek; **güvenilirlik** (reliability) ve **kalibrasyon hatasını** (ECE/Brier) düşürmek.

28.1 Kavramlar

* **Ham skor (s)**: Ensemble combine sonrası [−∞, +∞] aralığında bir değer (Parça 9).
* **Kalibre olasılık (p̂)**: *p̂ = P(win | s, scope)* — karar doğrudan p̂ üzerinden verilir.
* **Scope**: Kalibrasyonun geçerli olduğu bağlam: *acc:prod:tf* (hesap/ürün/TF) veya *prod:tf* (global).
* **Payout etkisi**: Sabit zamanlı işlemlerde beklenen değer: *EV = p̂·payout − (1−p̂)*; kırılma (breakeven) *p\* = 1/(1+payout)* (payout fraksiyon).
  + Örnek: payout=%90 → *p\* ≈ 1/1.9 ≈ 0.5263*.

28.2 Neden Kalibrasyon?

* **Karar eşiği** doğrudan p̂ ile verilebilirse (örn. *p̂ ≥ win\_threshold*), ürün/TF değişse bile mantık sabit kalır.
* **Güvenilirlik**: Aynı p̂ bandındaki örneklerin gerçek kazanma oranı p̂’ye yakın olur (Reliability).
* **Risk uyumu**: *p̂* → EV; payout değiştikçe **dinamik eşik** önerilebilir.

28.3 Yöntemler

1. **Platt (Lojistik)**: *p̂ = 1 / (1 + exp(a·s + b))*
   * Artıları: Hızlı, tekdüze (monotonik), pürüzsüz.
   * Eksileri: Doğrusal karar sınırı varsayar.
2. **İzotonik Regresyon**: Sıralı fit; tekdüzelik garantili, daha esnek.
   * Artıları: Skor→p̂ ilişkisinde esneklik.
   * Eksileri: Veri azlığında aşırı uyum; bakım için *min\_samples\_per\_bin* gerekir.
3. **Hibrit**: Veri azsa Platt, bolsa Isotonic; “detector” karar verir (*n>=Nmin*).

28.4 Veri Hazırlığı

* **Etiket**: *y ∈ {0,1}*; *push/abort* dışarıda.
* **Zaman bloklu CV**: *Forward chain* CV (sızıntı yok).
* **Denge**: Ağırlıklı kayıp kullan (sınıf dengesizliğinde).
* **Payout bantları**: İsteğe bağlı, *scope+payout\_bin* ile alt‑kalibrasyon.

28.5 Uygulama Akışı

1. **Topla**: Son *M* günün *v\_trades* kayıtları (örn. 30 gün veya ≥2k trade).
2. **Grupla**: Scope → (acc:prod:tf) öncelikli, veri azsa (prod:tf).
3. **Seç**: Veri ≥ Nmin (örn. 1000) → **Isotonic**, aksi **Platt**.
4. **Fit**: Parametreleri çıkar (*a,b* veya isotonic nokta listesi).
5. **Değerlendir**: ECE/Brier/LogLoss hesapla; hedef **ECE ≤ 0.05**.
6. **Kaydet**: *ensemble\_calibration* tablosuna yaz (Parça 22).
7. **Canlı kullanım**: Ensemble skoru *s* geldiğinde *p̂ = calibrate(scope, s)*.
8. **İzle**: Haftalık/aylık ECE trendi; gerekirse yeniden fit.

28.6 Dinamik Eşik (Payout’a Göre)

* **Breakeven**: *p\* = 1/(1+payout)* (payout oran).
* **Hedef EV marjı**: *m ≥ 0* (örn. 0.02).
* **Önerilen eşik**: *thr = max(win\_threshold\_cfg, p\* + m)*.
* UI’da “**Eşik Öner**” butonu: Güncel payout’a göre önerir; kullanıcı isterse uygular.

28.7 Python İskeletleri

*# core/ensemble/calibration.py***import** numpy **as** np**from** dataclasses **import** dataclass@dataclass**class** Platt: a: float b: float **def** \_\_call\_\_(self, s: np.ndarray) -> np.ndarray: z = np.clip(self.a\*s + self.b, -50, 50) **return** 1.0/(1.0 + np.exp(z))**class** Isotonic1D: **def** \_\_init\_\_(self, x, y): *# x: skor, y: gözlenen kazanma oranı (binlenmiş)* *# gerçek uygulamada sklearn.isotonic\_regression veya PAV* order = np.argsort(x) self.x = np.asarray(x)[order] self.y = np.asarray(y)[order] **def** \_\_call\_\_(self, s): s = np.asarray(s) idx = np.searchsorted(self.x, s, side='right') idx = np.clip(idx, 1, len(self.x)-1) x0, x1 = self.x[idx-1], self.x[idx] y0, y1 = self.y[idx-1], self.y[idx] t = np.where(x1==x0, 0.5, (s-x0)/(x1-x0)) **return** y0\*(1-t) + y1\*t@dataclass**class** Calibrator: kind: str *# 'platt' | 'isotonic'* scope: str *# 'prod:EURUSD:tf1' gibi* f: object *# çağrılabilir* ece: float brier: float **def** predict(self, s): **return** np.clip(self.f(s), 0.0, 1.0)

*# core/ensemble/threshold.py***def** breakeven\_threshold(payout\_frac: float) -> float: *# payout\_frac = 0.90 ( %90 ) gibi* **return** 1.0/(1.0 + payout\_frac)**def** suggest\_threshold(cfg\_thr: float, payout\_frac: float, margin: float=0.02) -> float: **return** max(cfg\_thr, breakeven\_threshold(payout\_frac) + margin)

28.8 DB Entegrasyonu

* **Tablo**: *ensemble\_calibration(scope TEXT PRIMARY KEY, a REAL, b REAL, ece REAL, brier REAL, updated\_ms INTEGER)* (Parça 22).
* **Isotonic** için: *a/b* yerine *blob*/JSON‑encoded düğüm listesi *nodes\_json* alanı (opsiyonel şema ekle).
* **API**: *GET /calibration/:scope* (UI grafikleri), *POST /calibration/retrain* (planlı görev).

28.9 UI Bileşenleri

* **Reliability diyagramı**: 10–20 bin ile; gözlenen vs beklenen.
* **ECE göstergesi**: Renk kodlu rozet; hedef ≤ 0.05.
* **Eşik önerisi**: Payout ve marj bazlı *thr* önerisi; “Uygula” ile profile yazar.
* **Scope seçici**: *acc:prod:tf* ↔ *prod:tf* kıyas; veri yetersiz uyarısı.

28.10 Drift Takibi & Yeniden Fit

* **Veri eşiği**: Günlük/haftalık minimum trade sayısı; altındaysa ertelenir.
* **Drift sinyali**: *ECE↑*, *Brier↑*, *KL(p̂\_fwd || p̂\_hist)↑* → uyarı *CALIBRATION\_DRIFT*.
* **Planlı görev**: Geceleri 03:30’da yeniden fit; **paper** ve **live** ayrı izlenir.
* **Geri dönüş**: Yeni model ECE’yi **düşürmüyorsa** deploy etme.

28.11 Test Planı

* Sentetik skor→etiket veri setinde Platt ve Isotonic doğrulama.
* Zaman bloklu CV ile sızıntı testi.
* Sınır durumları: dar aralık skorlar, tek taraflı sınıf, NaN/inf.
* Online izleme: Haftalık ECE trendi düşüyor olmalı; *CALIBRATION\_DRIFT* alarm testi.

28.12 Kabul Kriterleri (Parça 28)

* Platt ve Isotonic kalibratörleri **tanımlandı**, iskelet kod verildi.
* *ensemble\_calibration* ile DB kalıcılık ve **scope** mantığı açıklandı.
* Payouta bağlı **dinamik eşik** formülü ve UI akışı hazır.
* Drift/yeniden fit ve test planı **yazıldı**.

Parça 29 — Strateji Kataloğu & Parametre Aralığı Kılavuzu

Amaç: MoonLight içinde kullanılacak **strateji ailelerini** standartlaştırmak; her aile için **ID**, **parametreler**, **önerilen aralıklar**, **varsayılan presetler**, **ısıtma (warm‑up) barları**, **oy (vote) kuralları** ve **performans bütçeleri**ni tanımlamak. Bu kılavuz, Parça 23 SDK ve Parça 9 Ensemble ile uyumludur.

29.1 Katalog İlkeleri

* **Kimliklendirme**: Her stratejinin **benzersiz sabit ID**’si vardır (ör. *ID: 14 = EMA9/21 Crossover*).
* **Grup**: *trend*, *momentum*, *volatility*, *volume*, *hybrid*.
* **Zorunlu özellikler**: Gerekli indikatör/feature listesi (Parça 7–8).
* **Zaman uyumu**: Son **kapanmış** bara göre hesaplanır (repaint yok).
* **Bütçe**: TF=1’de **p95 < 3 ms/provider** hedefi; O(n) tek geçiş.

29.2 Aileler ve Tanımlar (özet tablo)

| ID | Ad | Grup | Gerekli Özellikler | Ana Parametreler |
| --- | --- | --- | --- | --- |
| 5 | EMA Trend + RSI (9/21 + RSI7) | hybrid | EMA(9,21), RSI(7) | *rsi\_up*, *rsi\_dn*, *pullback\_k* |
| 14 | EMA9/21 Crossover | trend | EMA(9,21), ATR(14) | *atr\_mult*, *confirm\_bars* |
| 15 | VWAP Reclaim + RVOL | volume | VWAP(sess), RVOL(20) | *rvol\_thr*, *min\_dist\_vwap* |
| 20 | VWAP Reclaim + RVOL (Günlük) | volume | VWAP(daily), RVOL(20) | *rvol\_thr* |
| 25 | Supertrend + ADX | hybrid | ST(7–10,3), ADX(14) | *adx\_thr*, *st\_len*, *st\_mult* |
| 35 | Keltner Break | volatility | KC(EMA20,ATR10) | *mult* (1.5–1.8), *close\_confirm* |
| 45 | Bollinger Walk | volatility | BB(20,2–2.2) | *bands*, *min\_bars\_on\_band* |
| 32 | MACD | momentum | MACD(12,26,9) | *hist\_up*, *hist\_dn*, *sig\_cross* |
| 22 | Stochastic RSI | momentum | RSI→Stoch(14,3,3) | *k\_up*, *k\_dn* |
| 34 | Triple MA (5/20/50) | trend | SMA/EMA(5,20,50) | *confirm\_bars* |
| 44 | GMMA (Kısa vs Uzun Demet) | trend | EMA demetleri | *spread\_thr*, *squeeze\_exit* |

Not: Diğer listelenen varyantlar (ör. 16–19 VWAP, 26–30 ST+ADX, 36–40 KC) bu ailelerin **preset** alt türleri olarak yönetilir.

29.3 Parametre Aralıkları & Varsayılan Presetler

Aşağıdaki aralıklar **TF=1/5/15** için pratik sınırlar ve **üç hazır profil** (Conservative/Neutral/Aggressive) içerir.

29.3.1 EMA Trend + RSI (ID:5)

* **Gerekli**: EMA(9,21), RSI(7), ATR(14) (SL/TP için).
* **Kurallar**: *EMA9>EMA21 & RSI7>rsi\_up* → **buy vote=+1**; *EMA9<EMA21 & RSI7<rsi\_dn* → **sell vote=−1**; aksi 0.
* **Parametreler**: *rsi\_up∈[52,65]*, *rsi\_dn∈[35,48]*, *pullback\_k∈[0.2,0.8]* (EMA9’a geri çekilme toleransı).
* **Presetler**:
  + **C**: *rsi\_up=58, rsi\_dn=42, pullback\_k=0.5*
  + **N**: *rsi\_up=55, rsi\_dn=45, pullback\_k=0.6*
  + **A**: *rsi\_up=52, rsi\_dn=48, pullback\_k=0.7*
* **Warm‑up**: max(EMA21 len + 5, RSI7 + 5) ≈ 30 bar.

29.3.2 EMA9/21 Crossover (ID:14)

* **Kurallar**: Kesişim bar kapanışında sinyal; *confirm\_bars∈[0,2]* ile gecikmeli onay.
* **Parametreler**: *atr\_mult∈[1.2,1.8]*, *confirm\_bars∈{0,1,2}*.
* **Presetler**: C: *atr=1.8, confirm=1* | N: *1.5,1* | A: *1.3,0*.

29.3.3 VWAP Reclaim + RVOL (ID:15/20)

* **Kurallar**: Fiyat **VWAP alt→üst** geri kazanım ve aynı bar/sonraki barda *RVOL(20)≥rvol\_thr*.
* **Parametre**: *rvol\_thr∈[1.10,1.60]* (TF=1’de 1.3–1.6 öneri), *min\_dist\_vwap∈[0.0,0.3%]*.
* **Presetler**: C: *1.5* | N: *1.3* | A: *1.2*.

29.3.4 Supertrend + ADX (ID:25)

* **Kurallar**: ST **yeşil** ve *ADX≥adx\_thr* → **buy**; kırmızı & *ADX≥adx\_thr* → **sell**.
* **Parametreler**: *st\_len∈[7,14]*, *st\_mult∈[2.5,3.5]*, *adx\_thr∈[18,28]*.
* **Presetler**: C: *10,3.2,24* | N: *10,3.0,22* | A: *7,3.0,20*.

29.3.5 Keltner Break (ID:35)

* **Kurallar**: Üst banda **kapanış** → trend yürüyüşü.
* **Parametreler**: *mult∈[1.5,1.8]*, *close\_confirm∈{0,1}*.
* **Presetler**: C: *1.8,1* | N: *1.6,1* | A: *1.6,0*.

29.3.6 Bollinger Walk (ID:45)

* **Kurallar**: Üst bantta **en az N ardışık kapanış** (N=2–3) → **buy**; alt bant için tersi.
* **Parametreler**: *bands∈[2.0,2.2]*, *min\_bars\_on\_band∈{2,3}*.
* **Presetler**: C: *2.2,3* | N: *2.0,2* | A: *2.0,2* (agresif takip).

29.3.7 MACD / Histogram / PPO (ID:32)

* **Kurallar**: Histogram **sıfır üstü** ve artan → **buy bias**; sıfır altı ve azalan → **sell bias**.
* **Parametreler**: *hist\_up∈[0.0, +∞)*, *hist\_dn∈(−∞,0.0]*, *sig\_cross∈{true,false}*.
* **Presetler**: C: *sig\_cross=true* | N: *false* | A: *false* + daha düşük eşikler.

29.3.8 Stochastic RSI (ID:22)

* **Kurallar**: *%K>k\_up* → **buy**, *%K<k\_dn* → **sell**; sinyal gecikmesi için *smooth=3*.
* **Parametreler**: *k\_up∈[70,85]*, *k\_dn∈[15,30]*.
* **Presetler**: C: *80/20* | N: *75/25* | A: *70/30*.

29.3.9 Triple MA (ID:34)

* **Kurallar**: *MA5>MA20>MA50* → **buy**; tersi **sell**.
* **Parametre**: *confirm\_bars∈{0,1,2}*.
* **Presetler**: C: *2* | N: *1* | A: *0*.

29.3.10 GMMA (ID:44)

* **Kurallar**: Kısa demet (3–15) uzun demet (30–60) üstünde, demetler **genişliyor** → **buy**.
* **Parametreler**: *spread\_thr∈[0.05%,0.5%]*, *squeeze\_exit∈[0.0%,0.2%]*.
* **Presetler**: C: *0.3,0.1* | N: *0.2,0.05* | A: *0.1,0.00*.

29.4 Oy (Vote) ve Skorlama Standartları

* **Vote**: *{-1,0,+1}*; eşzamanlı **aynı yönde** birden fazla sağlayıcı +1/−1 üretebilir.
* **Score**: *[-1,+1]* aralığına **kapalı** şekilde ölçekle; *meta* içinde ana sinyal gücünü kaydet (ör. *rsi=...*, *dist\_vwap=...*).
* **S‑cap**: Aşırı skorların ensemble’ı bozmaması için *score = tanh(raw/k)* ile **kapama** (k=2–5).

29.5 TF‑Özgü Notlar

* **TF=1**: Volatilite filtresi şart; *ADX* veya *RVOL* ile **low‑noise** seçimi.
* **TF=5/15**: Onay barı (*confirm\_bars*) kullanılabilir; *pullback* toleransı genişletilebilir.
* **Warm‑up**: En uzun hareketli ortalama/indikatör + 5 bar kuralı.

29.6 Performans Bütçesi

* **Ön‑hesap**: EMA/SMA/VWAP/BB/KC/ADX/MACD değerleri **feature katmanında** cache’lenir (Parça 8).
* **Kopya iş**: Aynı indikatörü iki strateji istiyorsa **tek hesaplanır**.
* **Zaman**: Provider başına p95<3ms, toplam *providers\_active \* 3ms* bütçesi TF tick’ini aşmamalı.

29.7 YAML Preset Örnekleri

presets**:** ema\_rsi\_conservative**:** id**:** 5 params**:** **{** rsi\_up**:** 58**,** rsi\_dn**:** 42**,** pullback\_k**:** 0.5 **}** vwap\_rvol\_neutral**:** id**:** 15 params**:** **{** rvol\_thr**:** 1.3**,** min\_dist\_vwap**:** 0.001 **}** st\_adx\_neutral**:** id**:** 25 params**:** **{** st\_len**:** 10**,** st\_mult**:** 3.0**,** adx\_thr**:** 22 **}** keltner\_aggr**:** id**:** 35 params**:** **{** mult**:** 1.6**,** close\_confirm**:** 0 **}**

29.8 Doğrulama & Testler

* **Birim**: Her parametre sınırında geçerli **vote/score** çıktısı.
* **Golden**: Örnek veri→çıktı **donmuş**; değişiklikte uyarı.
* **Performans**: 1k ardışık evaluate çağrısında p95 ölçümü.
* **Deterministiklik**: Tohum (seed) sabit, sonuçlar tekrar üretilebilir.

29.9 Yaşam Döngüsü & Sürümleme

* **SemVer**: Parametre/sinyal kırıcı değişiklik → **MAJOR** artışı.
* **Deprecated**: Eski presetler *deprecated=true* etiketi; UI uyarısı.
* **Belgeleme**: Şablon (Parça 23.10) ile **her sağlayıcı** için sayfa.

29.10 Güvenlik/Uyumluluk Notu

* Stratejiler yalnız **resmî/izinli API** üzerinden gelen verilerle çalışır.
* Her sağlayıcı **sandbox** içinde; dosya/ağ erişimi **yok** (Parça 23.7).
* TOS’a aykırı, yapay trafik/etkileşim üreten sağlayıcı **kabul edilmez**.

29.11 Kabul Kriterleri (Parça 29)

* Katalogdaki ana aileler, parametre aralıkları ve presetler **tanımlandı**.
* Vote/score standardı ve performans bütçesi **belirlendi**.
* YAML preset örnekleri ve test stratejisi **sağlandı**.
* Uyumluluk/sandbox sınırları **tekrarlandı**.

Parça 30 — Kullanıcı Eğitim Dokümanı (Hızlı Başlangıç + SSS)

Amaç: MoonLight’ı ilk kez kuran veya günlük kullanan kullanıcıların **hızlıca verimli** ve **güvenli** biçimde ilerlemesi için adım adım rehber, “nasıl yapılır” tarifleri, sık sorulan sorular ve terimler sözlüğü.

30.1 Kimler İçin?

* **Yeni kullanıcı**: Kurulum, profil, demo/paper ile deneme.
* **Gelişmiş kullanıcı**: Çoklu hesap, ileri ayarlar, raporlar, kalibrasyon.
* **Operatör**: Günlük kontroller, olay yönetimi (Runbook için bkz. Parça 24).

30.2 Ön Koşullar

* Windows 10/11 (64‑bit), SSD, en az 8 GB RAM (öneri: 16 GB).
* İnternet saati NTP ile eşitli (saat kayması ≤ 250 ms).
* Kendi platform hesabınız (resmî/izinli erişim; TOS uyumu — bkz. Parça 25).
* **Önemli**: İlk kullanımda **paper mod** önerilir.

30.3 Hızlı Başlangıç (10 Adım)

1. **Kur**: İmzalı installer’ı çalıştır, *MoonLight Core* servisi otomatik kurulur.
2. **İlk açılış sihirbazı**: Dil/tema → veri dizini → yedekleme saati (03:00).
3. **Hesap ekle**: *Hesaplar → Yeni*; kimlik bilgileri **Credential Manager**’a kaydedilir.
4. **Mod**: *Paper* modda başla.
5. **Ürün/TF seçimi**: Örn. *EURUSD (1,5,15)*; her TF için **permit payout aralığını** gir (örn. 89–93).
6. **Win eşiği**: Başlangıç için *win\_threshold=0.70* (Paper).
7. **Strateji preset**: *Katalog* → *ema\_rsi\_neutral* + *vwap\_rvol\_neutral* seç.
8. **Risk**: Günlük zarar sınırı, ardışık kayıp sınırı ve tek işlem limiti belirle.
9. **Başlat**: *Workers* panelinde başlat; Dashboard’dan canlı akışı izle.
10. **Rapor**: Gün sonunda *Raporlar → Backtest/Forward* ile özet oluştur.

30.4 Tipik Senaryolar

* **Yalnız demo/paper**: Parametre keşfi, kalibrasyon, rapor toplama.
* **Kısıtlı canlı**: Tek hesap, TF=5/15, küçük tutar; ELM izlemesi (Parça 26).
* **Çoklu hesap**: 4 hesap, farklı ürün/TF; her hesapta concurrency=1 kuralı.

30.5 UI Haritası

* **Dashboard**: Equity/Drawdown, anlık win‑rate, latency, uyarılar.
* **Hesaplar**: Ekle/düzenle, bağlantı durumu, oturum yenileme.
* **Ürün & Zaman Çerçevesi**: Permit pencereleri, win‑eşiği, concurrency.
* **Strateji Kataloğu**: Preset seçimi, param ayarı, etkin/pasif.
* **Risk & Guardrails**: Günlük/seri sınırları, kill‑switch.
* **Raporlar**: Backtest/forward rapor listesi, dışa aktarım.
* **Loglar & Uyarılar**: Canlı log görünümü, uyarı merkezi.
* **Ayarlar**: Sistem, veri, API, güvenlik, tema.

30.6 Nasıl Yapılır? (Kısa Tarifler)

* **Hesap ekleme/silme**: *Hesaplar → Yeni* / *Sil*; silerken yalnız profil kaldırılır, log/raporlar kalır.
* **Permit ayarlama**: *Ürün/TF* kartında *payout\_min/max*; dışına çıkınca giriş **yok**.
* **Win eşiği**: *Ürün/TF* kartında *win\_threshold*; istersen “**Eşik Öner**” (payout’a göre — Parça 28).
* **Başlat/Durdur**: *Workers* paneli; **Pause** aktif işlemleri kapatana kadar bekler.
* **Kill‑Switch**: Üst çubuk kırmızı buton; tüm hesaplarda anında durdurur.
* **Rapor oluşturma**: *Raporlar → Yeni*; dönem, ürün/TF seç; HTML/PDF aktar.
* **Yedek/geri yükle**: *Veri → Yedekle/Geri Yükle* sihirbazı (Parça 22/24).
* **Güncelleme**: *Ayarlar → Hakkında → Güncellemeyi denetle*; canlı pencerede otomatik güncelleme **kapalı**.

30.7 SSS (Sık Sorulan Sorular)

**S: Neden işlem açmıyor?**  
C: Çoğu zaman **permit dışı payout**, **win‑eşik altı p̂** veya **concurrency=1** kilidi. *Uyarı Merkezi* ve *trade\_skips* nedenlerini kontrol edin.

**S: “Gerçek hesap” modunu göremiyorum.**  
C: Varsayılan güvenlik gereği kilitli. *Risk & Guardrails* ekranında **bilinçli onay** kutusunu işaretleyin.

**S: Çoklu hesapta yanlış hesaba emir gider mi?**  
C: Her worker hesap/token ile **izole**; idempotent *client\_req\_id* tekrarı engeller.

**S: Win‑rate neden değişti?**  
C: Payout, piyasa rejimi ve kalibrasyon etkiler. *Raporlar → Kalibrasyon* sekmesini kontrol edin (ECE/Brier).

**S: TF=1’de gecikme uyarıları alıyorum.**  
C: İş yükünü azaltın, TF=5/15’e ağırlık verin; *Performans Modu*’nu açın.

**S: Parametreyi değiştirdim, ne zaman etkiler?**  
C: **Anında** (hot‑reload). UI’da *CONFIG\_APPLIED* bildirimi görünmeli.

**S: Loglarda kişisel verilerim var mı?**  
C: PII **maskeli**; parolalar **asla** yazılmaz (Parça 25).

**S: Raporu nasıl paylaşırım?**  
C: HTML/PDF dışa aktar; **trades.parquet** ile birlikte paylaşabilirsiniz.

**S: Çakışma/çökme sonrası ne olur?**  
C: Çekirdek açılınca **reconcile** yapar; yetim işlemler sonuçlandırılır (Parça 11/24).

**S: TOS uyumu?**  
C: Yalnız **resmî/izinli API**; scraping/RPA yok. Hesaplar sizin adınıza olmalı.

30.8 Sorun Giderme (Kısa Kontrol Listeleri)

* **Bağlantı yok**: Servis Running mi? Loopback portu meşgul mü? Güvenlik yazılımı engelliyor mu?
* **OTP gerekiyor**: OTP ekranı açılacak; OTP saklanmaz.
* **Yüksek CPU/RAM**: Aktif worker sayısını azalt, canlı grafik/WS örneklemesini düşür.
* **DB hatası**: Disk dolu mu? *Yedek & Temizle* çalıştır; nadiren servis restart.
* **Saat sapması**: Windows saat ayarlarını otomatiğe al, NTP ile eşitle.

30.9 Terimler Sözlüğü (Özet)

* **Win‑rate**: Kazanan işlemlerin oranı.
* **PF (Profit Factor)**: Toplam kâr / toplam zarar.
* **Exp (Expectancy)**: İşlem başına ortalama R.
* **p̂ (kalibre olasılık)**: Kazanma olasılığı tahmini.
* **ECE/Brier**: Kalibrasyon hatası ölçüleri.
* **Permit penceresi**: İşlem açmaya izin verilen payout aralığı.
* **Concurrency=1**: Aynı ürün/TF için aynı anda tek aktif işlem.
* **ELM**: Erken yaşam izlemesi.

30.10 İpuçları & Kısayollar

* *Ctrl+K*: Hızlı arama (ayar, rapor, strateji).
* *F1*: Bağlamsal yardım paneli.
* Tema: Mor/Mavi/Yeşil/Kırmızı/Siyah/Beyaz hazır paletler; **yüksek kontrast** modu.
* Panel düzeni kaydı: Kendi dashboard görünümünü kaydet/çağır.

30.11 Güvenlik & En İyi Uygulamalar

* Paper→Live geçişte **canary** ile kademeli açılış (Parça 26).
* Günlük **yedek** ve 3 ayda bir **token döndürme**.
* Parametre değişiklikleri ardından kısa **paper** doğrulaması.
* **Kill‑switch**’i düzenli test edin (< 500 ms hedef).

30.12 Sürüm Desteği & Değişiklikler

* *stable* sürümler için 6 ay **bakım** ve kritik hata düzeltmesi.
* Büyük değişikliklerde **migrasyon** notları ve geri alma (rollback) rehberi.
* Değişiklik günlüğü UI “Hakkında” ekranında ve *CHANGELOG.md*’de.

30.13 Kabul Kriterleri (Parça 30)

* Hızlı başlangıç, UI haritası, “nasıl yapılır” tarifleri ve SSS **hazır**.
* Sorun giderme listeleri ve sözlük **tam**.
* Güvenlik/uyumluluk vurguları ve iyi uygulamalar **belirtildi**.

Parça 31 — Gelişmiş İzleme Panoları (Dashboard & Telemetri)

Amaç: Gerçek zamanlı operasyonu **görünür**, **etkileşimli** ve **konfigüre edilebilir** hale getirmek; 4 hesap ve 300+ eşzamanlı *(ürün×TF)* izlemede akıcı, gecikmesi düşük ve erişilebilir bir Flutter arayüz sağlamak.

31.1 Hedefler

* **Tek bakışta**: Kâr/zarar, win‑rate, drawdown, latency, uyarılar.
* **Derine inme**: Ürün/TF, hesap ve strateji bazlı ayrıntı.
* **Canlılık**: 500 ms–2 s arası dinamik WS örnekleme (Parça 17 politikalarıyla uyumlu).
* **Kişiselleştirme**: Sürükle‑bırak panel düzeni, tema/palet ve filtre preset’leri.

31.2 Veri Kaynakları (REST/WS)

* **WS Kanalları** (Parça 15/17)
  + *metrics*: toplu zaman serisi (downsample + p50/p90/p99).
  + *trade\_updates*: yeni emir/sonuç olayları (NEW/SETTLED).
  + *alerts*: olay kodları (debounce edilmiş).
  + *logs*: JSONL (UI için örneklemeli).
* **REST**
  + *GET /status*, *GET /products*, *GET /accounts*
  + *GET /trades?product=EURUSD&tf=1&limit=200*
  + *GET /metrics?scope=prod:EURUSD&key=payout\_avg&bucket=5m*
  + *GET /reports/list*, *POST /workers/{id}/start|pause*

**WS Örnek Mesajlar**

**//** **trade\_updates**{"ts":1696665000123,"kind":"NEW","order":{"id":"o\_...","acc":"acc1","product":"EURUSD","tf":1,"dir":1,"amount":10.0,"payout":0.93,"p\_hat":0.76}}{"ts":1696665060123,"kind":"SETTLED","result":{"order\_id":"o\_...","status":"win","pnl":9.3,"latency\_ms":280}}**//** **alerts**{"ts":1696665000456,"level":"WARN","code":"LOSS\_STREAK\_NEAR","scope":"acc:acc1","msg":"2 losses in a row"}

31.3 Dashboard Düzeni (Grid)

* **Üst Şerit (KPI)**: Bugün P&L, Win‑Rate (rolling 200), MaxDD, ECE, Aktif Workers, p90 Latency.
* **Sol Kolon**:
  + **Equity & Drawdown** çizgisi.
  + **Reliability Diyagramı** (p̂ vs gerçek win).
  + **Latency CDF** ve p50/p90 kartları.
* **Sağ Kolon**:
  + **Payout Isı Haritası** (ürün×saat) + Permit penceresi göstergesi.
  + **Skip Nedenleri Donut** (*permit*, *confidence*, *concurrency*, *risk*, *latency*).
  + **Workers Durumu** (hesap×ürün×TF matris, renkli LED).
* **Alt Bölüm**:
  + **İşlemler Tablosu** (sanal liste, canlı akış).
  + **Uyarı Merkezi** (ack/snooze, runbook bağlantıları).
  + **Canlı Log Görünümü** (filtre + hız sınırlı akış).

31.4 Görselleştirmeler

* **Mum Grafiği**: Üstünde EMA/VWAP/BB; alt pencerede RSI/MACD/ADX.
* **Isı Haritaları**: Ürün×TF win‑rate; saat×win‑rate; payout bandı performansı.
* **Reliability**: 10–20 binli bantlar; ECE etiketli.
* **Latency**: CDF + 7 günlük p90 trendi.
* **PnL Dağılımı**: Histogram + kuyruğa duyarlı katsayılar.

31.5 Tema & Renk Paleti

* **Renkler**: Mor, Mavi, Yeşil, Kırmızı, Siyah, Beyaz (koyu/açık temalar).
* **Durum renkleri**: Başarı Yeşil (#22c55e), Uyarı Kehribar (#f59e0b), Hata Kırmızı (#ef4444), Nötr Mavi/Mor tonları.
* **Erişilebilirlik**: Kontrast **WCAG AA**; renk‑kör dostu palet (desen/ikon destekleri).
* **Tema Değiştirme**: Kullanıcı/cihaz bazlı kalıcı; profil ile senkron (Parça 12/24).

31.6 Kişiselleştirme

* **Sürükle‑Bırak Grid**: Panellerin yeri/boyutu değişir (örn. *flutter\_staggered\_grid\_view*).
* **Preset’ler**: *Scalper*, *Risk*, *Kalibrasyon*, *Performans* görünümleri.
* **Dışa/İçe Aktar**: Düzen JSON olarak kaydet/çağır; cihazlar arasında paylaş.

31.7 Filtre & Arama

* **Global filtre çubuğu**: Hesap, ürün, TF, tarih aralığı, payout bandı.
* **Hızlı filtreler**: *p̂≥thr*, *permit only*, *TF∈{1,5}*, *acc=acc1*.
* **Tablolarda arama**: order\_id, ürün, durum; kopyala/CSV aktar.

31.8 Performans Bütçesi (UI)

* **Frame bütçesi**: 60 FPS hedef; **16 ms** çizim sınırı.
* **Sanal liste**: 10k+ satır akışında bile akıcı scroll.
* **Örnekleme**: WS mesajları coalesce (500–2000 ms).
* **Backpressure**: UI kuyruğu 2× eşik → otomatik örnekleme artır.
* **Grafik penceresi**: Son N bar (örn. 500) ile sınırla; eski veriyi diskten gerektiğinde yükle.
* **Offline modu**: WS yoksa düşük frekanslı REST polling.

31.9 Uyarı Merkezi

* **Sınıflar**: INFO / WARN / ERROR.
* **Eylemler**: *Aç*, *Snooze (10/30/60 dk)*, *Kapat*, *Runbook’ı Aç*.
* **Kurallar**: Tekrarlayan uyarılarda **grouping**; ilgili işlem/ürün/hesaba hızlı atla.
* **Bildirim**: Sistem sesi (opsiyon), Windows bildirimleri (yerel, offline).

31.10 Erişilebilirlik (A11y)

* **Klavye**: Tam navigasyon; kısayollar (bkz. Parça 30.10).
* **Ekran Okuyucu**: Semantik etiketler (*Semantics*) ve tablolarda hücre açıklamaları.
* **Hareket azaltma**: Animasyonlar sistem tercihini izler.
* **Font boyutu**: Dinamik ölçek (min 12 pt).

31.11 Flutter Mimari (Öneri)

* **Durum yönetimi**: *flutter\_riverpod* veya *bloc*.
* **Servisler**: *WebSocketService*, *MetricsRepository*, *TradesRepository*.
* **Grafikler**: *fl\_chart* veya *charts\_flutter* (lisans değerlendirmesi).
* **Tablo**: *DataTable2*/*PlutoGrid* (büyük veri için sanallaştırma).

**WS Akışı (iskele kod)**

final wsProvider = Provider<WebSocketService>((ref) => WebSocketService('ws://127.0.0.1:8751/ws'));final metricsStreamProvider = StreamProvider.autoDispose<MetricEvent>((ref) { final ws = ref.watch(wsProvider); **return** ws.streamWhere('metrics'); *// kanal filtreli yayın*});**class** MetricCard **extends** ConsumerWidget { **@override** Widget build(BuildContext ctx, WidgetRef ref) { final async = ref.watch(metricsStreamProvider); **return** Card(child: async.**when**( data: (m) => Text('${m.key}: ${m.p90.toStringAsFixed(0)}'), loading: () => const CircularProgressIndicator(), error: (e,\_) => Text('Hata: $e'), )); }}

**Tema (renk önayarı)**

ThemeData buildTheme(Color seed) => ThemeData( colorScheme: ColorScheme.fromSeed(seedColor: seed, brightness: Brightness.dark), useMaterial3: true,);

31.12 Test & QA (UI)

* **Widget testleri**: KPI kartları, tablo filtreleri, uyarı merkezinde ack/snooze.
* **Golden test**: Temalar ve kritik görünümler (Dashboard, Ürün/TF).
* **Entegrasyon**: Mock WS/REST ile canlı akış; performans profil çıkarımı.
* **Erişilebilirlik**: Semantik ağaç ve kontrast denetimleri (otomasyon).

31.13 Kabul Kriterleri (Parça 31)

* WS/REST veri akışıyla **akıcı** dashboard.
* KPI, grafikler, ısı haritaları, tablo ve uyarı merkezi **tamamlandı**.
* Tema/palet ve düzen kişiselleştirme **kalıcı**.
* UI performansı ve A11y testleri **geçer**.

Parça 32 — Log Toplama & Destek Paketleyici (Final)

Amaç: MoonLight için tek tıkla **teşhis** (diagnostics) ve **destek paketi** (support bundle) üretmek; gizliliği koruyarak **log + metrik + konfig** özetini standart bir ZIP içinde toplamak, otomatik **olay sınıflandırma** ve **sağlık raporu** eklemek.

32.1 Kapsam & Hedefler

* **Tek komut/tek tık**: “Yardım → Destek Paketi Oluştur” veya CLI ile.
* **Gizlilik**: PII maskeleme, sırların hariç tutulması (Parça 25 ile uyum).
* **Standart paket**: *ML-support-<date>-<support\_id>.zip*.
* **Teşhis**: Olay sınıflandırma + sağlık özet raporu.
* **Boyut sınırı**: Varsayılan **≤ 25 MB**; gerekirse örnekleme/truncation.

32.2 Paket İçeriği (Dizin Yapısı)

ML-support-2025-10-07\_1030-<support\_id>.zip manifest.json # meta + anonimleştirme bilgisi summary.txt # insan okunur kısa özet health\_report.json # metrik/skip/latency özetleri incident\_report.json # otomatik sınıflandırma sonuçları config\_snapshot.json # etkin yapı (sırlar maskeli) checksums.txt # SHA-256 özetleri logs/ system.log.jsonl # son 24–72s (örneklemeli) trade.log.jsonl # işlem olayları connector.log.jsonl # API/WS katmanı security.log.jsonl # giriş/OTP/anahtar erişimi (maskeli) metrics/ rollup\_5m.csv # p50/p90/p99, throughput skip\_breakdown.csv # permit/confidence/... dağılımı db/ integrity.txt # PRAGMA checks snapshot.sqlite # (opsiyonel; kullanıcı onaylı) sysinfo/ env.txt # OS/sürüm/CPU/RAM/SSD process.txt # core sürümleri ve modüller

32.3 Gizlilik & Redaksiyon Politikası

* **Maskeler**: e‑posta → *ab\*\*\*@d\*\*.com*, kullanıcı adı → *u:\*\*\*\**, hesap ID → *acc#XXXX* (deterministik psödonomi).
* **Sırlar hariç**: Parola, token/refresh, OTP **pakete girmez**.
* **Yol beyaz listesi**: Yalnız *%ProgramData%/MoonLight* altı.
* **DB snapshot**: Varsayılan **kapalı**; UI onayıyla dahil edilir.
* **Zaman penceresi**: Varsayılan **24 saat** (ayar: 6h/24h/72h/özel).

32.4 CLI & UI Akışı

* **UI**: *Yardım → Destek Paketi* → Zaman aralığı seç → “DB snapshot dahil et” onayı → **Oluştur**.
* **CLI**:
* moonlight-support pack --since 24h --include-db-snapshot=off --max-size-mb 25
* **Çıktı**: ZIP dosya yolu gösterilir; “Klasörde Aç” ve “Kopyala” butonları.

32.5 Manifest Şeması (örnek)

{ "support\_id": "ML-7F3A-20251007-1030", "core\_version": "1.0.0", "ui\_version": "1.0.0", "period": {"from": 1696640000000, "to": 1696665000000}, "accounts": ["acc#A1","acc#B2"], "products": ["EURUSD","XAUUSD"], "timeframes": [1,5,15], "redaction": {"email\_mask": **true**, "secrets\_excluded": **true**}, "hash\_salt": "8b1c..." }

32.6 Sağlık Raporu (health\_report.json)

* **Özet KPI**: trades, win‑rate (rolling 200), PF, MaxDD, ECE/Brier.
* **Latency**: place→confirm p50/p90, WS lag p99.
* **Skip dağılımı**: permit/confidence/concurrency/risk/latency.
* **Uyarılar**: *CIRCUIT\_BREAKER\_TRIP*, *RATE\_LIMIT\_HIT*, *LATENCY\_HIGH* sayıları.
* **Kapasite**: aktif worker sayısı, CPU/RAM ortalaması, disk alanı.

32.7 Olay Sınıflandırıcı (incident\_report.json)

* **Kurallar** (örnek):
  + 429/5xx oranı ↑ ve backoff → **Ağ/Rate‑limit sorunu**.
  + p90 latency > eşik & TF=1 yoğun → **Aşırı yük/işleme gecikmesi**.
  + *permit* skip > %70 → **Payout kaynaklı giriş engeli**.
  + *CIRCUIT\_BREAKER\_TRIP* var → **Risk koruması tetiklemiş**.
* **Çıktı alanları**: *top\_issues[] {code, severity, evidence[], suggested\_runbook}*.
* **Runbook bağlantısı**: Parça 24 bölüm/başlık kimlikleriyle.

32.8 Toplama & Paketleme (Python iskeleti)

**import** json, zipfile, hashlib, time, os, glob**from** pathlib **import** PathROOT = Path(r"C:**\**ProgramData\MoonLight")OUT = ROOT / 'support'MASK\_KEYS = {'password','token','refresh','otp'}**def** mask\_line(line: str) -> str: *# çok basit örnek; gerçek uygulamada regex + şema bazlı* **for** k **in** MASK\_KEYS: line = line.replace(f'"{k}":', f'"{k}":"\*\*\*" #') **return** line**def** sha256(p: Path) -> str: h = hashlib.sha256() **with** p.open('rb') **as** f: **for** chunk **in** iter(**lambda**: f.read(1<<16), b''): h.update(chunk) **return** h.hexdigest()**def** collect\_logs(dst: Path, hours=24): cutoff = time.time()\*1000 - hours\*3600\*1000 dst.mkdir(parents=True, exist\_ok=True) **for** name **in** ['system','trade','connector','security']: out = dst / f'{name}.log.jsonl' **with** out.open('w', encoding='utf8') **as** w: **for** p **in** sorted((ROOT/'logs').glob(f'{name}\*.jsonl')): **with** p.open('r', encoding='utf8', errors='ignore') **as** r: **for** ln **in** r: **try**: obj = json.loads(ln) **if** obj.get('ts',0) >= cutoff: w.write(mask\_line(ln)) **except** Exception: **pass** **return** dst**def** make\_bundle(hours=24, include\_db=False, max\_size\_mb=25): support\_id = f"ML-{int(time.time())}" tmp = OUT / support\_id logs = collect\_logs(tmp/'logs', hours) *# manifest & health/incident raporları burada üretilir (özetlenmiş)* zpath = OUT / f"ML-support-{support\_id}.zip" **with** zipfile.ZipFile(zpath, 'w', zipfile.ZIP\_DEFLATED) **as** z: **for** p **in** tmp.rglob('\*'): z.write(p, p.relative\_to(tmp)) *# boyut kısıtı kontrolü* **if** zpath.stat().st\_size > max\_size\_mb\*1024\*1024: *# basit truncation/örnekleme stratejisi uygulanır* **pass** **return** zpath

32.9 DB Sağlık Kontrolü

* **PRAGMA**: *integrity\_check*, *foreign\_key\_check* → *db/integrity.txt*.
* **WAL durumu**: WAL dosya boyutu/compaction.
* **Snapshot**: Kullanıcı isterse *VACUUM INTO* ile **salt okunur** kopya.

32.10 Hata İzleme (Yerel)

* **Sentry/Benzeri (ops.)**: İnternet kapalı ortamlar için **yerel kuyruk** ve kullanıcı onaylı gönderim.
* **Crash dump**: Python *faulthandler* + mini‐dump; yalnız yerel pakete dahil.

32.11 Boyut Yönetimi & Örnekleme

* **Öncelik**: *manifest*, *summary*, *health/incident* **her zaman**; loglar **son‑giren‑ilk‑çıkar** ve satır örnekleme.
* **Sıkıştırma**: ZIP Deflate; JSONL minimal (whitespace yok).
* **Sınırlar**: Varsayılan 25 MB; UI’da 10–100 MB aralığı seçilebilir.

32.12 Test & Doğrulama

* **Redaksiyon**: Sızma testi (bilinçli sahte sır) → pakette görünmemeli.
* **Bütünlük**: *checksums.txt* ve ZIP doğrulaması.
* **Tekrarlanabilirlik**: Aynı giriş → aynı özet (mask salt sabit ise).
* **Boyut**: 1–3 gün loguyla paket < limit.
* **Runbook linkleri**: incident\_report’ta doğru başlıklar.

32.13 Kabul Kriterleri (Parça 32)

* UI ve CLI üzerinden **tek tık** destek paketi üretilebiliyor.
* Gizlilik/PII **maskeli**, sırlar hariç; DB snapshot **kullanıcı onaylı**.
* health/incident raporları ve checksums mevcut; ZIP boyutu sınır içinde.
* Runbook (Parça 24) ile çapraz referanslar doğru.

Proje Kapanışı

* **Teslimatlar**: Parça 1–32 arasında mimari, güvenlik, risk, strateji, kalibrasyon, UI, paketleme ve runbook dahil **tam proje belgesi** oluşturuldu.
* **MVP Hedefi**: Parça 21 sprint planı ile Windows 10/11 üzerinde 4 hesap/çoklu ürün‑TF **paper→kısıtlı live** akışı.
* **Uyumluluk**: Yalnız **resmî/izinli API**; TOS ve gizlilik ilkeleri Parça 25 uyarınca.
* **Sonraki Adımlar (opsiyonel)**:
  1. Gerçek veriyle **kalibrasyon** ve rapor üretimleri (Parça 27–28).
  2. Strateji kataloğunun genişletilmesi (Parça 29).
  3. Otomatik güncelleme & sürüm kilidi iyileştirmeleri (Parça 18–19).
  4. Çok‑cihaz senkronizasyonu (isteğe bağlı, kapsam dışı notları Parça 12’de).

**Not**: Bu doküman, farklı bir sohbete yüklendiğinde kaldığı yerden devam edebilmek için parça başlıkları ve sözleşme/şema detaylarıyla **kendini açıklayıcı** hazırlanmıştır.