DMITRIY MOTORIN

Senior iOS Engineering Portfolio

Swift | SwiftUI | UIKit | Rust/CoreFFI | On-device AI | OCR/RAG | FinTech

ENGINEERING THESIS

I build native mobile products where correctness, latency, offline behavior and explainability matter. My differentiator is keeping the Apple experience native while moving deterministic or performance-sensitive logic into well-tested Rust cores - and connecting that work to release, telemetry and product outcomes.

12 YEARS MOBILE AND FINTECH SINCE 2014	**40 MODULES** RUST CORE BEHIND NATIVE IOS	**223 TESTS** SHARED-CORE UNIT AND INTEGRATION
VISION OCR PDFKIT AND DETERMINISTIC FACTS	**CAMERA DSP** AVFOUNDATION AND DEVICE EVIDENCE	**SWIFT 6** PUBLIC REAL-TIME FINTECH SAMPLE

SELECTED CASES

01 Solaros - native iOS and Rust/CoreFFI
02 LandWatch - OCR, evidence and RAG guardrails
03 Hara - AVFoundation camera PPG and signal quality
04 FinStreamKit - public-safe Swift 6 real-time finance

LIMASSOL, CYPRUS | REMOTE | mmotorin@gmail.com | linkedin.com/in/dmotorin | github.com/jimbokl

CASE 01 / CONFIDENTIALITY-SAFE

SOLAROS

Native iOS product with an authoritative Rust core

ENGINEERING VERDICT

Keep the product experience native in SwiftUI while centralizing deterministic safety, forecasting and advisory logic in one portable Rust implementation.

40	209	14	OFFLINE
RUST MODULES	UNIT TESTS	INTEGRATION TESTS	CORE PRODUCT MODE

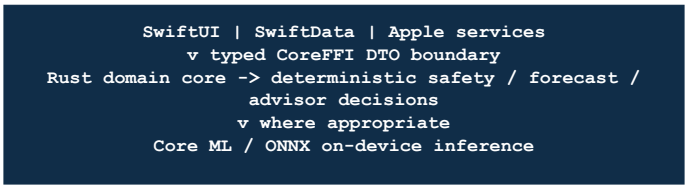
CHALLENGE

An offline-first energy application needed device ingestion, calculations, forecasting and safety decisions without duplicating domain logic across Swift and future clients. The boundary also had to remain testable and reproducible for physical devices and simulators.

WHAT I OWNED

- Native Swift/SwiftUI architecture and adaptive iPhone/iPad delivery
- 40-module Rust core and typed UniFFI/CoreFFI boundary
- Core ML/ONNX on-device model delivery and parity checks
- SwiftData, Keychain, local notifications and Apple-platform services
- Reproducible SolarosCoreFFI.xcframework packaging and release preflight

ARCHITECTURE



HARD DECISIONS

1. Keep UI state and Apple services in Swift; move only stable deterministic domain logic across FFI.
2. Prefer flat immutable DTOs and batched calls over chatty cross-language object graphs.
3. Keep safety rules deterministic: ML may forecast but cannot silently author safety actions.
4. Build device and simulator artifacts reproducibly instead of checking in opaque binaries.

EVIDENCE Observed locally: 209 Rust unit tests, 14 integration tests, 40 engine modules, reproducible XCFramework packaging, simulator/device builds, screenshots, archives and App Store preflight.

INTERVIEW DEEP DIVE *FFI ownership and copies | generated-binding drift | async boundaries | error mapping | deterministic tests | Core ML parity | release reproducibility*

CASE 02 / CONFIDENTIALITY-SAFE

LANDWATCH

OCR and document intelligence that fails closed

ENGINEERING VERDICT

Treat OCR and RAG as evidence pipelines, not as permission to turn uncertain text into confident facts.

1,977	40	35 MB	RU / EN
UNIQUE PARCELS	PAGE SAFETY CAP	FILE SAFETY CAP	VISION OCR

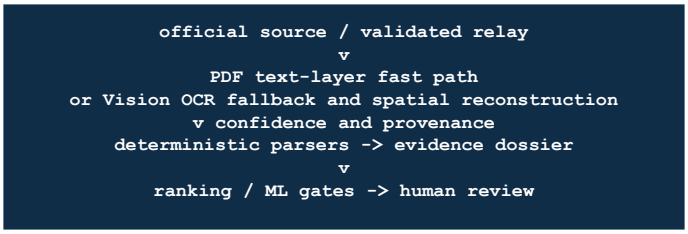
CHALLENGE

Official land-auction documents arrived through unreliable sources and mixed formats: some PDFs had usable text layers, others were scans. The system needed defensible cadastral, lease, GPZU and auction facts while preserving uncertainty and provenance.

WHAT I OWNED

- Compiled Swift helper using PDFKit and Apple Vision accurate RU/EN recognition
- Text-layer fast path with raster OCR fallback and spatial line ordering
- Confidence output, atomic JSON and deterministic document parsers
- Python orchestration with secure downloads, caching, retries and timeouts
- Evidence dossiers, SQLite history, Telegram workflows and time-based ML evaluation

ARCHITECTURE



HARD DECISIONS

1. Use a trustworthy embedded text layer before paying the OCR cost.
2. Preserve coordinates and confidence so extraction can be audited.
3. Cap hostile documents, isolate retries and cache source artifacts with explicit TTLs.
4. Keep LLM synthesis downstream of deterministic facts and provenance.
5. Preserve the baseline when a challenger misses its promotion gate.

EVIDENCE Observed locally: time-based model audit across 1,977 unique parcels; deterministic EGRN, GPZU, lease and protocol paths; explicit source-health, evidence and failure handling.

INTERVIEW DEEP DIVE *PDF text vs scan detection | Vision bounding boxes | schema validation | prompt guardrails | provenance | human review | challenger promotion criteria*

CASE 03 / CONFIDENTIALITY-SAFE

HARA

Real-time camera PPG with honest signal-quality UX

ENGINEERING VERDICT

A camera pulse demo becomes a product only when the system distinguishes physiology, placement, hardware behavior and signal failure.

NATIVE	LIVE	XCTest	DEVICE
SWIFTUI PRODUCT	CAMERA DSP	AUTOMATED COVERAGE	VALIDATION RUNBOOKS

CHALLENGE

Camera-based PPG changed with finger pressure, lens selection, torch behavior, exposure and frame instability. A visually plausible waveform could still be unusable, so the app needed explicit quality evidence rather than false confidence.

WHAT I OWNED

- Native SwiftUI application and AVFoundation capture service
- Real-time DSP for waveform, pulse and within-session variation
- Camera/lens/torch tuning and capture-session threading
- Input stabilization, hysteresis and placement coaching
- SwiftData evidence, CSV diagnostics and a replaceable DSP boundary
- XCTest/UI coverage, simulator sweeps and physical-device runbooks

ARCHITECTURE

```
AVCaptureSession -> frame / sample extraction
v
stabilization and quality gates -> DSP service ->
immutable measurement evidence
v
@MainActor presentation -> SwiftData / CSV audit trail
```

HARD DECISIONS

1. Keep capture and processing off the main actor; publish compact state to the UI.
2. Gate measurement output on signal quality instead of smoothing failures into plausible numbers.
3. Use hysteresis so coaching and status do not flicker around thresholds.
4. Keep DSP behind a protocol so Rust can replace it without rewriting the UI.

EVIDENCE Observed locally: SwiftUI and AVFoundation implementation, XCTest/UI suites, simulator and physical-device workflows, CSV diagnostics and explicit non-medical release boundaries.

INTERVIEW DEEP DIVE *capture-session isolation | frame cadence | real-time state updates | quality thresholds | device variance | fixtures | privacy and non-medical boundaries*

CASE 04 / CONFIDENTIALITY-SAFE

FINSTREAMKIT

Public-safe Swift 6 patterns for real-time finance

ENGINEERING VERDICT

The hard part of a trading screen is preserving correct state through duplicated events, out-of-order updates, cancellation and stale prices-not coloring a quote green or red.

13	10x	0	SWIFT 6
XCTEST CASES	PARALLEL-SUITE RUNS	LINT WARNINGS	STRICT CONCURRENCY

CHALLENGE

The strongest iOS/Rust products in this portfolio are private. FinStreamKit is deliberately small and dependency-free so reviewers can inspect concurrency ownership, integer money arithmetic and failure handling without receiving employer or customer code.

WHAT I OWNED

- QuoteBook actor with sequence-aware ingestion and newest-value AsyncStream buffering
- Cancellation cleanup and @MainActor presentation state
- Deterministic risk policy for freshness, spread, quantity, notional and limit deviation
- Scaled integer prices, overflow handling and idempotency-aware requests
- Atomic caller-scoped JSON snapshots, XCTest and GitHub Actions CI

ARCHITECTURE

```
transport DTOs -> QuoteBook actor -> sequence /
freshness rules
v immutable MarketQuote
OrderRiskPolicy -> approved / rejected value
v
@MainActor TradingDashboardModel
```

HARD DECISIONS

1. No floating-point money: ScaledPrice stores integer micros.
2. Risk approval is not execution success; transport and reconciliation remain explicit concerns.
3. Dependencies and snapshot locations are caller-owned; there is no hidden singleton state.
4. Arithmetic overflow, cancellation and hostile event ordering are tested as product behaviors.

EVIDENCE Verified locally: 13 tests, zero-warning swift-format lint, ten consecutive parallel-suite runs, coverage instrumentation and a successful release build. GitHub Actions is prepared.

INTERVIEW DEEP DIVE *Swift actor ownership | AsyncStream cancellation | stale-event policies | integer arithmetic | idempotency | reconnect reconciliation | test determinism*

PUBLIC LINKS MetaCAD benchmarks | SOLBOT | GoldGRU

Private code, credentials, customer data and proprietary screenshots are intentionally excluded. Architecture and selected code can be discussed in a controlled interview walkthrough.