Agentic Software Development & AI-Native Application Prototyping
Autonomous Code Generation Using Multi-Modal AI Orchestration

I architect full-stack applications using agent-first development paradigms where LLM agents handle planning, scaffolding, implementation, and verification cycles. Primary tooling includes Google Antigravity—an agent-native IDE where Gemini 3.1 Pro/Flash models operate across editor, terminal, and browser contexts to autonomously execute complex build tasks—and OpenCode, a terminal-based, model-agnostic agent supporting 75+ LLM providers (Claude Sonnet 4, GPT-4o, local Ollama endpoints) through bring-your-own-key configurations.

Technical Specifications:

Agent Orchestration: Multi-agent workflows in Antigravity's "Mission Control" interface—task delegation between planner, coder, and reviewer agents with automatic to-do list generation and artifact checkpointing; CLI-based agent interaction via OpenCode's TUI with custom prompt chaining and LSP/MCP integration
Vibe-to-Production Pipeline: Natural language requirements ("build a React Native habit tracker with offline-first SQLite and Expo push notifications") translated through iterative agent context refinement into functional MVPs with type-safe TypeScript, automated testing scaffold generation, and CI/CD configuration files
Cross-Platform Mobile Development: Android Studio integration for APK compilation, Gradle dependency resolution, and emulator debugging; Expo EAS build pipeline configuration for over-the-air updates; iOS Simulator testing via Xcode command-line tools with provisioning profile automation
Model Context Protocol (MCP): Integration of external tool ecosystems—database browsers, API documentation scrapers, Jira/Linear task importers—into agent context windows for grounded, context-aware code generation
Network Diagnostics & Infrastructure Validation Automation
Cross-Layer Connectivity & Performance Testing Suites

I develop comprehensive network validation frameworks spanning Layer 2–7 diagnostics—from physical link integrity (ethernet cable TDR validation) to application-layer API health checks. Testing infrastructure deploys as portable Python/Bash scripts, containerized test runners, or integrated GitHub Actions workflows for CI/CD pipeline gating.

Technical Specifications:

Latency & Throughput Analysis: iperf3 multi-threaded bandwidth testing across WAN links, mtr (My Traceroute) path characterization for intermittent loss detection, and SmokePing long-term latency trend visualization; nmap NSE scripts for service fingerprinting and vulnerability scanning
DNS & Certificate Validation: drill/dig trace analysis for propagation timing, certbot/cfssl certificate expiry monitoring with automated renewal hooks, and TLS configuration hardening via SSL Labs API integration

Wireless Site Surveys: iwlist/iw scan parsing for 802.11 signal strength heatmapping, WiFi channel interference analysis, and rogue AP detection via Kismet integration; Android WiFi diagnostics via adb shell dumpsys wifi for RSSI/SNR extraction
Connectivity Automation: Python scapy-based probe generation, asynchronous asyncio ping sweeps (/24 to /16 range enumeration), and MQTT/WebSocket broker stress testing with custom payload generation; results aggregated to InfluxDB with Grafana alerting thresholds
Binary Analysis & Software Reconstruction
Static/Dynamic Reverse Engineering for Security Research & Interoperability

I perform deep binary analysis on compiled artifacts using Ghidra (NSA's SRE framework) for decompilation, control-flow graph reconstruction, and vulnerability discovery. Mobile applications analyzed via Android Studio (APK decompilation, smali bytecode review, DEX-to-JAR conversion) and iOS tooling (IPA extraction, entitlements parsing, Mach-O analysis via otool/lldb).

Technical Specifications:

Native Code Reconstruction: Ghidra automated analysis pipelines—function identification via FLIRT signatures, type propagation, structure recovery from PDB/DWARF debug symbols when available, and dynamic instrumentation via frida-server injection for runtime method hooking
Android APK Decomposition: apktool resource extraction, JADX decompilation to Java source, smali/baksmali bytecode manipulation for runtime patching (method redirection, certificate pinning bypass); Android Studio Profiler integration for memory leakage analysis in reconstructed builds
iOS Binary Analysis: class-dump/objc-runtime header generation, Swift demangling, entitlements.plist extraction, and jailbreak-assisted dynamic analysis via substitutate/ellekit hooking; Frida script injection for method swizzling and encryption key extraction from Secure Enclave-bound operations
Network Protocol Reconstruction: Wireshark/tshark PCAP analysis combined with Ghidra decompilation of protocol handlers to reconstruct proprietary binary protocols; protobuf/Thrift/gRPC message structure inference via reflection and static analysis
Documentation & Remediation: Comprehensive C pseudocode output from Ghidra decompiler, control-flow graph exports for security audit reports, and reconstructed API specifications enabling interoperability layers or migration off deprecated/unsupported vendor binaries
Toolchain Integration

Workflows bridge AI-generated scaffolding with manual RE verification—e.g., using OpenCode agents to rapidly prototype Python bindings for Ghidra's SRE API, or Antigravity agents to parse reconstructed protobuf definitions into functional gRPC client/server implementations.