Anti-Paste Guard — Project Plan & Documentation

This document provides a detailed technical plan and stage-wise documentation for the Anti-Paste Guard desktop application. The project aims to detect and flag copy-paste or AI-assisted writing during controlled scenarios such as examinations or graded assignments. It combines keystroke logging, anomaly detection, clipboard monitoring, and cryptographic tamper-evident logging, integrated into a Tkinter-based dashboard.

Stage 1 — Environment & Architecture

- Repo created; Python 3.11+ environment.
- Package structure: `app/` (controller, model, view), `core/` (hooks, clipboard, crypto), `ui/` (Tkinter).
- Tkinter root window with minimal status bar.
- Logging with JSON output (Loguru/structlog).

Stage 2 — Event Capture Layer

- Keyboard/mouse listener (`pynput`) with timestamps, modifiers, scan codes.
- Events dispatched via thread-safe queue to controller.
- Improvements: enforced enums for actions, base `to_record()` method, auto-etype in dataclasses.

Stage 3 — Clipboard & Paste Detection

- Detect hotkey (Ctrl/Cmd+C/X/V), context menu paste, Linux primary paste.
- Clipboard watcher via `pyperclip`.
- Privacy safe: only payload length stored, not content.
- Added CommandEvents and ClipboardEvents.

Stage 4 — Focus/Context Tracking

- Track active process/app using `psutil` + OS APIs (win32gui/Quartz).
- Whitelist/blacklist allowed apps.
- Events annotated with current app context.

Stage 5 — Anomaly Rules Engine

- Metrics: WPM, CPM, inter-key delay, idle time.
- Rules: idle-to-burst, text injection, multi-paste streak, timing uniformity.
- Config via `AnomalyConfig` (Pydantic-like).
- Implemented `metrics.py` with numpy.

Stage 6 — Tamper-Evident Logging

- Append-only SQLite segment store.
- Cryptography: ChaCha20-Poly1305, AES-SIV, HKDF key ratchet, Ed25519 signatures.
- Verified with HMAC chain checker (`tools/verify_segments.py`).
- Errors resolved by enforcing UTF-8, ASCII-safe output.

Stage 7 — Admin Dashboard

- Tkinter dashboard (`ui/admin_dashboard.py`) with:
• Live feed (event stream, anomalies).
• Metrics panel (WPM, CPM, delay).
• Chart (typing vs. anomalies).
• Export CSV; PDF optional.
- Verification button runs verifier script via subprocess.

Stage 8 — Packaging & Smoke Tests

- PyInstaller used (`apg.spec`).
- Matplotlib backend fixed (`TkAgg`).
- CLI tool `apg_cli.py`: run, verify, suites.
- Cross-platform packaging supported (Windows tested).

Key Innovations Across Stages - Strong type safety: Enums for Key/Mouse/Clipboard actions. - Privacy: clipboard contents never logged, only metadata. - Crypto-agility: multiple ciphers, randomized per-segment. - Tamper-evident logs: verified chain and signatures. - UI integration: real-time anomaly monitoring in Tkinter. **Requirements**: see accompanying `requirements.txt`.