SIUE Computer Science Senior Project Proposal

Project Title

Cruft Crawler – Offline AI-Driven Background Disk Cleanup

Brief Background

Workstations and laptops accumulate “cruft” — old backups, caches, and temporary files — that silently consume storage. Manual cleanup is tedious, while cloud or remote cleanup services risk leaking private data.

Cruft Crawler is an **LLM-first background agent** that runs entirely offline from a 64 GB USB drive. It profiles the filesystem over 14 days with imperceptible CPU load, uses a **local quantized LLM** to recommend safe deletions, and delivers a concise AI-generated report. No PII ever leaves the machine.

The project is a proof-of-concept:

- Can a large language model (7B–30B parameters) run locally on CPU/GPU in the background?
- Can it surface useful weekly insights without disrupting normal work?

Project Description

Core Features

- **USB-First Deployment**
    - All binaries, the quantized LLaMA model, and sled DB state stored on the USB drive.
    - On plug-in, Rust executable launches as a background service (systemd on Linux, Windows Service).
    - No host installation required.
- **Incremental, Invisible Operation**
    - Uses the `steady_state` crate to break work into small tasks:
        - Filesystem profiling
        - Metadata extraction
        - Rule-based filtering
        - LLM inference

        - Report assembly
    - CPU & I/O load remain below a user-configurable “noise floor.”
- **Creative Cruft Discovery**
    - Goes beyond known temp directories:
        - High-churn folders (e.g. nested `.cache` trees)
        - Orphaned subtrees (VM images, old snapshots)
        - Duplicate or near-duplicate files
        - Stale logs / databases (based on access vs. modify times)
- **Rich Metadata Pipelines**
    - Each candidate file/folder annotated with:
        - Byte-level stats (entropy, compression ratio, histogram peaks)
        - Folder context (counts, distributions, diversity)
        - Temporal cues (last-access vs. last-modify)
        - Fuzzy hashes for duplication detection
- **Local LLM Inference (LLM-First)**
    - Model: LLaMA-2 (7B → 30B), quantized to 4-bit (GGUF format).
    - Rust FFI bindings to `llama.cpp` C API (`llama_context_load`, `llama_tokenize`, `llama_eval`).
    - Quanta-Based Throttling:
        - Batch 5–10 files per prompt
        - Tokenize & evaluate in 16-token chunks with 50 ms sleeps
    - TokenBucket rate limiting:
        - Caps CPU usage (~5% of a 4-core machine)
        - Dynamic boost when idle
- **Weekly/Fortnightly Report**
    - Summary of KEEP/DELETE recommendations with one-sentence AI rationales.
    - Delivered via HTML report or desktop notifications.
    - Users can mark items as **Never Delete** to persist exclusions across runs.

Technical Requirements

- **Language & Concurrency**
    - Rust with `steady_state` actor-based scheduling.
- **Local Model & Inference**
    - LLaMA-2 quantized (7B–30B) via `llama.cpp`/GGUF.
    - Rust FFI bindings for inference loop.
    - Quanta-based throttling & sleep intervals.
- **Pre-Filter & Profiling**
    - Rule-based plus dynamic discovery.
    - Rich metadata extraction pipelines.
- **Persistence**
    - `sled` embedded key-value store on USB.
    - Stores scan progress, metadata, and “Never Delete” flags.
    - Resilient to USB removal/re-insertion.
- **Packaging & Config**
- ```text
  /cruft_crawler(.exe)

  /models/llama2-7b-4bit.gguf      ← dev model

  /models/llama2-30b-4bit.gguf     ← demo model

  /data/state/                     ← sled database

  /config.toml                     ← thresholds, rate limits
  ```
- **Testing**
    - Unit tests (profiling, metadata, TokenBucket, sled mock).
    - Integration tests on sample filesystems with accelerated clocks.
    - Benchmarks for tokens/sec, CPU duty cycle, I/O impact.
- **Supported Platforms**
    - Linux (systemd)
    - Windows 10+

  ```

Phased Deliverables

- **Phase 1 – Core**
    - USB packaging
    - Filesystem profiling with sled persistence
    - Rule-based cruft detection
    - Basic HTML report
- **Phase 2 – LLM Integration**
    - `llama.cpp` FFI bindings
    - 7B quantized model for classification
    - TokenBucket throttling
- **Phase 3 – Stretch Goals**
    - Upgrade to 30B model for demo
    - Idle/power state detection
    - Duplicate detection & entropy analysis

Bonus Features

- Idle & power-state detection
    - Linux: D-Bus, X11/Wayland APIs, systemd-logind
    - Windows: `GetLastInputInfo()`, `WM_POWERBROADCAST`
    - Boosts analysis when idle

Client

Nathan Tippy
Staff Software Engineer & STL Rust Meetup Organizer
📧 nathan@tippy.name
📞 636-295-5785

Provides model files, test VMs, and demo slots.

Intellectual Property

- **License:** MIT (common for Rust projects).

Why This Project Is Exciting

- **LLM-First Edge AI**: Validates running 7B–30B models fully local and offline.
- **Invisible Background Agent**: Teaches resource management (actors, throttling).
- **Smart Cruft Discovery**: Goes beyond static heuristics with AI insights.
- **Rust Systems Mastery**: Combines sled, steady_state, llama.cpp, USB deployment.
- **Demo-Ready**: Portable live demo for SIUE and STL Rust Meetup.

Final Presentation

- **Format:** Live demo + Q&A
- **Showcase:**
    - Background profiling
    - LLM inference without user disruption
    - Weekly AI cleanup report (KEEP/DELETE)
    - User interaction loop