Here’s a **finalized plan** for refactoring Chiro-Reporter into a modern, performant architecture using **Rust (Actix-Web), HTML/CSS/JS, and NGINX**

**✅ Finalized Architecture Plan: Rust + HTML/JS + NGINX**

**🔹 High-Level Overview**

* **Frontend (HTML/CSS/JS)**:
  + Plain web app with file upload form, report type selector, progress indicator, and report output preview.
  + Communicates with backend /api endpoints.
  + Hosted statically via **NGINX**.
* **Backend (Rust / Actix-Web)**:
  + Accepts file uploads, handles text extraction, prompt generation, OpenAI API calls, report formatting, validation, and file export.
  + Uses native Rust crates (not Streamlit) for OCR, DOCX/PDF generation, and LLM communication.
  + Logging and Alerts
* **Deployment Layer**:
  + NGINX acts as a reverse proxy with SSL (Let's Encrypt).
  + Optionally includes a Python microservice behind /python/ route (e.g., for legacy DICOM or OCR logic via FastAPI if needed).

**🧱 Directory Structure (Revised)**



Figure 1: This is incorrect at this point.

**🧩 Core Components**

**1. Frontend (HTML/CSS/JS)**

* Features:
  + File upload
  + Report type selector
  + Form submission via AJAX
  + Loading spinner / status bar
  + Result preview with export options (DOCX/PDF/TXT)
* Hosted via / using NGINX

**2. Backend (Rust Actix-Web)**

* /api/generate → Accepts file uploads + metadata, returns completed report.
* /api/validate → Validates report against Dr. Carabasi's structure/style/depth.
* /api/export → Returns generated file as downloadable DOCX, PDF, or TXT.
* /api/ocr → Handles OCR for scanned PDFs/images if Tesseract + Poppler are enabled.
* /api/health → System readiness check (OCR, OpenAI API, fonts, templates)

**3. Modules (Rust)**

| **Module** | **Purpose** | **Crates Used** |
| --- | --- | --- |
| api.rs | Route declarations, error handling | actix-web, serde, tokio |
| extract.rs | Text extraction from PDF, DOCX, TXT, DICOM | lopdf, docx-rs, tesseract-rs, pydicom |
| llm.rs | OpenAI API client logic | openai-rs, tiktoken-rs, reqwest |
| validate.rs | Validates style, clinical detail, and required structure | regex, nlp, syntect (optional for syntax scoring) |
| export.rs | Export GPT output to DOCX, PDF, TXT | docx-rs, printpdf, pulldown-cmark |
| templates/ | Stores markdown + json prompt templates for each report type | serde\_json, tera |

**🧠 Logic Carried Forward from Python Code**

**From streamlit\_gpt\_report\_app.py:**

* ✅ Multi-format document ingestion (extract\_text)
* ✅ OpenAI GPT-4 prompt orchestration with custom tone/style
* ✅ Dynamic token counting and rate limiting
* ✅ Style/structure validation per Carabasi’s style
* ✅ Enhanced report formatting (esp. Assessment section parsing)
* ✅ Signature block and footer logic
* ✅ Export logic with sanitized filenames, formatting, spacing

**These are now implemented in:**  
→ extract.rs, llm.rs, validate.rs, export.rs

**📄 Report Templates (From Uploaded Files)**

* Denial\_Appeal\_Report.txt
* Patient\_Status\_Report.txt
* Referral\_Summary\_Report.txt

These templates will be embedded in templates/ as markdown or structured JSON with placeholders for dynamic values.

**🔐 Deployment with NGINX & Let's Encrypt**

* Serve / as static HTML UI (index.html, JS, CSS)
* Reverse proxy /api/\* → localhost:8080 (Rust)
* (Optional) Reverse proxy /python/ → legacy microservice (e.g., /python/gen)
* Use certbot to handle SSL (auto-renewal cron job)

**🧪 Validation & Testing**

* Unit tests: token counting, text extraction, prompt rendering
* Integration tests: /api/generate, /api/export, /api/validate
* CLI or UI-based health check dashboard (status of OCR, templates, font availability)

**✅ Final Notes**

This plan preserves the **clinical fidelity** and **workflow sophistication** of the Python app, while delivering:

* ✅ 10x performance improvements (compiled Rust backend)
* ✅ Modular, testable components
* ✅ Frontend independence (JS/HTML can be swapped or themed)
* ✅ Easy deployment with Docker or systemd/nginx

A diagram of a data flow

AI-generated content may be incorrect.

A diagram of a computer program

AI-generated content may be incorrect.