AI DOCUMENT TRANSLATOR (ENGINEERING EDITION) - GUI

Version: 1.0 (Final Production / Batch Support / Anti-Leak / Deep Context)

---

1. SYSTEM ARCHITECTURE & METHODOLOGY

This AI application operates as a document reconstruction engine designed to maintain strict fidelity to the original file's layout and formatting. Unlike standard machine translation workflows that often flatten documents into raw text, this system preserves the structural integrity of `.docx` files by treating them as a hierarchy of XML constituents rather than a simple linear string. By separating content from presentation, the software ensures that translated documents retain their exact visual organization.

The workflow relies on a multistage process beginning with the `python-docx` library, which parses the file into atomic units such as paragraphs, runs, and table cells. The system records the precise formatting properties—including font styles, colors, and numbering—associated with every text segment. The text is then processed by a local Large Language Model (LLM) treated as a deterministic compiler, where content is encapsulated in semantic tags to guide the translation. After a verification step removes any conversational filler or meta-commentary, the document is rebuilt by injecting the translated text back into the original XML structure, seamlessly preserving headers, footers, and complex formatting.

[THE "DIGITAL TWIN" METHODOLOGY]

Standard translators (Google/DeepL) often strip formatting to process raw text, destroying the document's layout. This engine employs a **"Digital Twin"** approach:

- **ATOMIC DISSECTION:** The Python-docx library is leveraged to surgically parse the .docx file into its atomic XML constituents (Paragraphs > Runs > Tables > Cells).
- **DNA EXTRACTION:** Before translation, the engine extracts the "Visual DNA" of every text segment. This includes Font Family, Size, RGB Color, Highlight (Background), Bold/Italic/Underline flags, and complex XML Numbering properties.
- **COMPILER INJECTION:** Text is encapsulated with semantic tags (e.g., "{b}Text{/b}") and injected into a local Large Language Model (LLM) via Ollama. The LLM is treated not as a chatbot, but as a deterministic "Text Compiler" with temperature=0.0.
- **HALLUCINATION SANITIZATION:** The output passes through a rigorous "Firewall" to strip AI meta-commentary, ensuring only pure

translation data remains.

- **SURGICAL RECONSTRUCTION:** The document is rebuilt from zero. The translated text is fused with the original "Visual DNA" and injected back into the XML skeleton, preserving headers, footers, and complex table layouts.

---

2. CORE CAPABILITIES

[A] BATCH PROCESSING ORCHESTRATOR

- **CONCURRENT QUEUE MANAGEMENT:** Capable of ingesting unlimited file paths via the GUI. The engine isolates file operations in a worker thread to prevent GUI freezing ("Not Responding").
- **PREDICTIVE WORKLOAD CALCULATION:** Performs a "Global Pre-Scan" algorithm that traverses every section (Body, Header, Footer, Tables) of every file to count total processable blocks. This ensures the progress bar represents real-time atomic progress, not just "File 1 of 5".
- **FAULT TOLERANCE:** If a single file is corrupt (XML errors), the engine logs the failure and automatically proceeds to the next file in the queue without crashing the batch.

[B] ANTI-LEAK FIREWALL (MULTI-LAYERED SANITIZATION)

- **THE PROBLEM:** LLMs are prone to "leaking" internal monologue (e.g., "Here is the translation," "I cannot translate names," or internal JSON markers).
- **THE SOLUTION:** A heuristic "Regex Firewall" that scrubs the output in five passes:

1. **CONVERSATIONAL FILLER REMOVAL:** Strips introductory phrases ("Sure,", "Note:", "Likely means").
2. **TAG REPAIR:** Detects and fixes broken formatting tags often mangled by AI tokenizers (e.g., repairing "{/ b}" to "{/b}" or "{tab}" to valid XML tab stops).
3. **MARKDOWN STRIPPING:** Removes unrequested Markdown artifacts (**bold**, ## Header) that LLMs habitually add.
4. **ENTITY DECODING:** Converts HTML entities (, <) back to Word-compatible unicode.
5. **INVISIBLE CLEANUP:** Purges Unicode Category 'C' (Control characters) and JSON artifacts (leftover braces or "1.1.1" key prefixes) that corrupt .docx XML.

[C] REFLEXION & SELF-CORRECTION (ACTOR-CRITIC LOOP)

- **STABILITY ENGINE:** The system implements an "Actor-Critic" architecture.
- **PASS 1 (THE ACTOR):** The primary model attempts the translation.
- **PASS 2 (THE CRITIC):** An internal logic check analyzes the output for "Chattiness" (length ratio > 1.8x original) or "Leakage" (detection of forbidden phrases).
- **PANIC MODE:** If the Critic rejects the output, the engine engages "Panic Mode." It discards the complex prompt and retries using a simplified, high-restriction prompt to force a raw literal translation, ensuring no data is lost even in difficult segments.

[D] XML STRUCTURAL & DATA PRESERVATION

- **NUMBERING INTEGRITY (`numPr`):** Engineering specifications rely on strict hierarchy (1.0, 1.1, 1.1.2). The engine deep-copies the XML `numPr` (Numbering Properties) from the source and re-attaches it to the translated paragraph. This keeps lists "live" and clickable in Word.
- **DATA PROTECTION:** The engine utilizes Regex pattern matching to identify "Pure Numeric Blocks" (e.g., "500", "10/2023", "Ø 15mm"). These blocks are bypassed entirely to prevent the AI from hallucinating numbers (e.g., changing "100" to "one hundred" or "100.0").

[E] CONSULTANT MODE (DUAL-LLM TERMINOLOGY AUDIT)

- **THE "TWO-PASS" SYSTEM:** Optionally engages a Secondary Expert Model (e.g., trained on ISO Standards).
- **WORKFLOW:**

1. The Primary Model (e.g., TranslateGemma) drafts the translation.
2. The Consultant Model reviews the draft specifically for "Technical Accuracy" (Units, Material Names, ISO Codes).
3. If the Consultant detects an error (e.g., "Concrete Screed" vs "Cement Paste"), it patches the translation via a strict JSON protocol.

- **EFFICIENCY:** Includes a length-heuristic filter to only trigger the Consultant on complex technical sentences, skipping simple headers to save inference time.

[F] COMPILER-STRICT PROMPTING ARCHITECTURE

- **SYSTEM PROMPT ENGINEERING:** The LLM is not prompted as a "Helpful Assistant." It is prompted as a "Data Processing Engine" with strict Opcodes.
- **FORMAT ENFORCEMENT:** Enforces a JSON-only output structure (`{"t": "translated_string"}`). This containerization prevents the

translation from bleeding into surrounding text and allows the program to programmatically validate success (checking for valid JSON parsing).
- **TAG INVARIANCE:** Instructions explicitly forbid the translation or removal of injected tags ({b}, {i}, {tab}), treating them as "Memory Pointers" that must be returned intact.

---

3. INSTALLATION & CONFIGURATION GUIDE

STEP A: INSTALLING OLLAMA (THE AI RUNTIME)

Ollama is the engine that allows you to run powerful AI models locally without internet.

1. **Download:**

- Windows: Visit [https://ollama.com/download/windows%5D(https://ollama.com/download/windows) and run the .exe installer.
- macOS: Visit [https://ollama.com/download/mac%5D(https://ollama.com/download/mac) and download the disk image.
- Linux: Run `curl -fsSL https://ollama.com/install.sh | sh` in your terminal.

2. **Verify:** Open terminal/cmd and run `ollama --version`.

STEP B: CHOOSING & INSTALLING MODELS

The GUI automatically detects installed models, but they must be "pulled" (downloaded) via the terminal first. The following hierarchy categorizes models by their architectural power and specific domain utility.

1. PRIMARY TRANSLATION ENGINES (Languages A > B)
These models are responsible for the core document reconstruction. They are ranked by their ability to handle technical syntax and maintain formatting.

OPTION 1: THE SPECIALIST: TranslateGemma (Recommended)

- **Description:** A specialized variant of Google's **Gemma 3 (27B)**, TranslateGemma is optimized for high-fidelity document translation through a two-stage distillation process from Gemini models. It is specifically engineered to function as a deterministic processor, excelling at "Compiler-Strict" instruction following. This ensures that critical structural tags (like `{b}`, `{tab}`, or `{i}`) are preserved as anchors while complex engineering syntax is translated without data loss.
- **Pros:** Exceptional technical accuracy, best-in-class adherence to formatting constraints, and robust resistance to "instruction drift" in long-form technical manuals.

- **Hardware:** 24GB+ RAM recommended for the 27B version (16GB+ for highly quantized 4-bit versions). Runs optimally on a single high-end GPU.
- **Link:** [https://ollama.com/library/translategemma%5D(https://ollama.com/library/translategemma)
- **Command:** `ollama pull translategemma:27b`

OPTION 2: THE POWERHOUSE: GPT-OSS 20B

- **Description:** Developed by OpenAI under the Apache 2.0 license, GPT-OSS 20B is a state-of-the-art **Mixture-of-Experts (MoE)** reasoning model (21B total / 3.6B active parameters). It is designed to deliver local reasoning performance comparable to the **o3-mini** series. Trained on a massive corpus focused on STEM, coding, and technical prose, it is ideal for "Deep Context" tasks where the model must navigate complex engineering constraints and ambiguous sentence structures.
- **Pros:** Features **Configurable Reasoning Levels** (Low, Medium, High) allowing you to trade speed for deeper logical analysis. It excels at "Chain-of-Thought" processing, ensuring the semantic meaning of engineering specs is maintained through complex translations.
- **Hardware:** Optimized for consumer hardware with **16GB VRAM**. Native MXFP4 quantization allows the full 21B logic to fit into standard 16GB GPU memory.
- **Link:** [https://ollama.com/library/gpt-oss%5D(https://ollama.com/library/gpt-oss)
- **Command:** `ollama pull gpt-oss:20b`

OPTION 3: THE POLYGLOT: Qwen-MT

- **Description:** Specialized specifically for high-stakes translation, **Qwen-MT** is powered by the latest **Qwen 3** architecture. It utilizes an efficient **Mixture-of-Experts (MoE)** setup that allows it to outperform significantly larger models (including GPT-4.1-mini and Gemini-2.5-Flash) in technical translation benchmarks. Trained on trillions of tokens including vast repositories of technical manuals and formal documents, it is the industry gold standard for **CJK (Chinese/Japanese/Korean)** languages and supports a total of 92 languages.
- **Pros:** Features native support for **Terminology Intervention** and **Format Preservation**. It is uniquely architected to maintain "Sentence Structure Rigidity," ensuring that complex engineering tables, hierarchical lists, and structural formatting remain intact during the translation process.
- **Hardware:** Extremely efficient due to MoE parameter activation. Runs smoothly on **12GB+ RAM**; optimized for high-throughput, low-latency performance on consumer-grade hardware.
- **Link:** [https://ollama.com/library/qwen-mt%5D(https://ollama.com/l

ibrary/qwen-mt)
- **Command:** `ollama pull qwen-mt`

OPTION 4: THE SPEEDSTER: Qwen 3 (4B)

- **Description:** Part of the latest generation of Alibaba's model family, **Qwen 3 (4B)** is a highly compact dense model that punches far above its weight class. It features a dual-operational mode: a "Non-Thinking" mode for near-instantaneous translation and a "Thinking" mode for solving complex logical puzzles or ambiguous syntax. It is the first sub-7B model natively trained for the **Model Context Protocol (MCP)**, making it exceptionally stable when handling the structured XML tags required by this application.
- **Pros: Extremely Fast.** It achieves 30–50 tokens/second on mid-range GPUs and remains highly responsive even on CPU-only inference. It features a massive **131,072 token context window**, allowing it to digest entire technical manuals in a single pass without "forgetting" early instructions.
- **Hardware:** Ultra-low resource usage. A 4-bit quantized version requires only ~**3GB of VRAM**, making it the perfect choice for legacy laptops or systems with only 8GB of total System RAM.
- **Link:** [https://ollama.com/library/qwen3:4b%5D(https://ollama.com/library/qwen3)
- **Command:** `ollama pull qwen3:4b`

OPTION 5: THE LOCALIZER: Translation & Localization Specialist

- **Description:** A massive **50-billion parameter** model fine-tuned by Alientelligence, specifically designed to bridge the gap between literal translation and high-level localization. While Tier 1 models focus on "Compiler-Strict" tag preservation, this model is trained on a vast corpus of regional standards, cultural idioms, and "tone of voice" protocols. It is the ideal choice for "soft" engineering documents—such as **Environmental Impact Assessments (EIA)**, project proposals, or consumer-facing manuals—where flow, readability, and cultural adaptation are more critical than rigid XML tag fidelity.
- **Pros:** Exceptional for regional compliance (e.g., adapting terminology for EU vs. US markets) and superior "natural" phrasing that reads as if written by a native speaker. It excels at maintaining a professional, non-conversational engineering tone across long documents.
- **Hardware: Heavy.** Requires a minimum of **32GB+ RAM** (or 24GB+ VRAM for GPU acceleration). For optimal performance on large batches, 64GB of Unified Memory (Mac M-Series) or a dedicated high-end GPU is recommended.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/translationandlocalizationspecialist%5D(https://ollama.com/ALIENTELLIGENCE/translati

onandlocalizationspecialist)
- **Command:** `ollama pull ALIENTELLIGENCE/translationandlocalizationspecialist`

---

2. SECONDARY CONSULTANT MODELS (Optional Auditors)

Do not use these for the main translation. These models are designed to be loaded into the "Consultant" slot of the application. Their role is to perform a secondary "Audit Pass" on the translated text to ensure that technical nomenclature, industry-specific acronyms, and specialized units remain accurate to the engineering discipline.

ENGINEERING MANUALS

- **Description:** An advanced **8-billion parameter** model built on the **Llama 3.1** architecture, specifically optimized for **Synthesizing Technical Manuals** and dense reference documentation. It features a massive **128K context window**, enabling it to cross-reference entire technical handbooks in active memory to extract data from complex schematics and troubleshooting procedures with high precision. In this workflow, its primary role is to enforce **"Official Nomenclature,"** acting as a linguistic firewall to ensure that every translated term adheres to standardized industry language and official acronyms.
- **Architecture:** Llama 3.1 - 8B Parameters (Custom Alientelligence fine-tune optimized for high-context retrieval).
- **Hardware:** Efficient and highly responsive. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). Its 4.7GB footprint makes it an ideal persistent background consultant.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/engineeringtechnicalmanuals%5D(https://ollama.com/ALIENTELLIGENCE/engineeringtechnicalmanuals)
- **Command:** `ollama pull ALIENTELLIGENCE/engineeringtechnicalmanuals`

STRUCTURAL LLAMA 3.0

- **Description:** A high-performance **8-billion parameter** model built on the **Llama 3 Instruct** framework, specifically fine-tuned for the rigorous demands of structural engineering. It is engineered to assist with structural design principles, analysis methods, and failure investigations, while referencing authoritative international codes including **AISC, ACI, CSA, and Eurocode**. Beyond textual analysis, it is uniquely "primed" for **Python-based engineering**, offering specialized support for VS Code and Poetry environments to generate and debug code using libraries like **NumPy, SciPy, and Matplotlib** for structural calculations and 3D visualization.

- **Architecture:** Llama 3 - 8B Parameters (7.4GB GGUF). Optimized for high-context engineering reasoning and technical code integration.
- **Hardware:** Optimized for consumer hardware. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). A dedicated GPU is highly recommended for real-time code generation and analysis.
- **Link:** [https://ollama.com/joreilly86/structural_llama_3.0%5D(https://ollama.com/joreilly86/structural_llama_3.0)
- **Command:** `ollama pull joreilly86/structural_llama_3.0`

CIVIL STRUCTURE ENGINEER V2

- **Description:** An advanced iteration of the structural engineering auditor, explicitly optimized for **Tool-Calling and Structured JSON output**. While generalist models provide prose-based advice, V2 is engineered to act as a data-driven assistant, capable of cross-referencing structural specifications against technical datasets with high precision. It is the primary choice for verifying complex steel frameworks, load-bearing calculations, and material schedules where the output must be surgically clean and free of conversational "fluff."
- **Architecture: 8.03-billion parameter** model based on the **Llama 3.1** framework. It features an expanded **128K context window**, allowing it to maintain a "global" understanding of massive engineering blueprints and multi-page calculation sheets without losing track of early constraints.
- **Hardware:** Highly efficient for local deployment. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). Its 4.7GB footprint (Q4_0 quantization) allows it to run concurrently with the primary translator on most modern workstations.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/civilstructureenginee rv2%5D(https://ollama.com/ALIENTELLIGENCE/civilstructureengin eerv2)
- **Command:** `ollama pull ALIENTELLIGENCE/civilstructureengineerv2`

GEOTECHNICAL ENGINEER

- **Description:** A professional-grade **8-billion parameter** model specifically fine-tuned for the subsurface investigation and foundation engineering domains. It serves as a specialized auditor for **Ground Investigation Reports (GIR)**, ensuring that critical parameters such as "liquefaction potential," "pore water pressure," "effective stress," and "shear strength" are translated with mathematical and geological precision rather than general dictionary definitions. It is uniquely adept at parsing borehole logs and Cone Penetration Test (CPT) data within technical documents to maintain consistency across soil profile descriptions.
- **Architecture: Llama 3.1** based fine-tune utilizing the **IAGENTZ** specialized training protocol. It features an expanded **128K context window**, allowing it to analyze massive geotechnical datasets and long-form

geological surveys without losing track of early site constraints.

- **Hardware:** Optimized for efficiency. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). Its 4.7GB footprint makes it an ideal lightweight "Sidecar" consultant that can run alongside the primary translation engine.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/geotechnicalenginee r%5D(https://ollama.com/ALIENTELLIGENCE/geotechnicalengineer)
- **Command:** `ollama pull ALIENTELLIGENCE/geotechnicalengineer`

MARINE ENGINEER & NAVAL ARCHITECT

- **Description:** A professional-grade **8-billion parameter** model specifically fine-tuned for the maritime industry, focusing on ship propulsion, hull integrity, offshore structures, and marine systems. It serves as a specialized auditor to verify that maritime-specific nomenclature—such as "starboard," "ballast tank," "displacement," and "hydrostatic stability"—is maintained with nautical precision. It is particularly adept at interpreting and validating documentation related to **SOLAS (Safety of Life at Sea)** and other international maritime safety standards.
- **Architecture:** Based on the **Llama 3.1** framework, featuring an ultra-wide **131,072 token context window**. This allows the model to "read" and maintain consistency across massive technical specifications for shipyards and naval vessels in a single pass.
- **Hardware:** Optimized for high-efficiency local deployment. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). Its compact 4.7GB footprint (Q4_0 quantization) makes it an ideal persistent background consultant for maritime engineering workflows.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/marineengineernaval architect%5D(https://ollama.com/ALIENTELLIGENCE/marineenginee rnavalarchitect)
- **Command:** `ollama pull ALIENTELLIGENCE/marineengineernavalarchitect`

MECHANICAL ENGINEER V2

- **Description:** A specialized **8-billion parameter** model fine-tuned for the rigorous demands of mechanical design, thermodynamics, and fluid dynamics. It serves as a technical auditor for machinery operation manuals, maintenance schedules, and MEP (Mechanical, Electrical, Plumbing) drawings. It is particularly adept at a "Sanity Check" pass on terms related to torque, pressure, thermal conductivity, and complex HVAC systems to ensure they align with ISO and ASME standards.
- **Architecture:** Based on the **Llama 3.1** framework with a **128K context window**, allowing it to maintain a "global" understanding of dense equipment manuals in a single pass.
- **Hardware:** Highly efficient. Requires **8GB–16GB of System RAM** (or 8GB VRAM for GPU acceleration). Its 4.7GB footprint makes it ideal

for running concurrently with the primary translation engine.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/mechanicalengineer%5D(https://ollama.com/ALIENTELLIGENCE/mechanicalengineer)
- **Command:** `ollama pull ALIENTELLIGENCE/mechanicalengineer`

ELECTRICAL ENGINEER V2

- **Description:** A domain-expert **8-billion parameter** model designed to validate electrical engineering documentation and schematics. It is trained to audit **Wiring Diagrams**, single-line diagrams, and electrical safety specifications. Its primary role is to ensure that voltage, current, resistance, and power generation terminology is technically sound and adheres to IEC and NEC (National Electrical Code) standards.
- **Architecture:** Llama 3.1 framework featuring a massive **128K context window**, enabling the analysis of multi-page electrical load schedules and protection coordination studies.
- **Hardware:** Requires **8GB–16GB of System RAM**. Optimized for low-latency response times on consumer-grade hardware.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/electricalengineerv2%5D(https://ollama.com/ALIENTELLIGENCE/electricalengineerv2)
- **Command:** `ollama pull ALIENTELLIGENCE/electricalengineerv2`

CHEMICAL ENGINEER

- **Description:** A professional-grade model specialized in process engineering, stoichiometry, and reaction kinetics. This consultant is the primary choice for auditing **Material Safety Data Sheets (MSDS/SDS)** and plant safety protocols. It ensures that chemical formulas, hazard descriptions, and reaction parameters are translated with scientific precision to prevent safety-critical mistranslations.
- **Architecture: 8B Parameters** utilizing the Llama 3.1 engine with a **128K context window** for processing extensive chemical process safety reports.
- **Hardware:** Efficient 4.7GB footprint. Runs smoothly on systems with **8GB VRAM** or 16GB System RAM.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/chemicalengineer%5D(https://ollama.com/ALIENTELLIGENCE/chemicalengineer)
- **Command:** `ollama pull ALIENTELLIGENCE/chemicalengineer`

INDUSTRIAL ENGINEER

- **Description:** An expert system focused on systems engineering, supply chain logistics, and manufacturing optimization. Use this consultant to audit **Factory Acceptance Test (FAT)** documents, process flow charts, and quality assurance manuals. It is designed to verify terms related to throughput, industrial efficiency, and workflow optimization to ensure they meet professional Lean/Six Sigma standards.

- **Architecture:** Llama 3.1 framework with a **128K context window**, allowing it to cross-reference entire production line specifications for terminological consistency.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/industrialengineer%5D(https://ollama.com/ALIENTELLIGENCE/industrialengineer)
- **Command:** `ollama pull ALIENTELLIGENCE/industrialengineer`

ENVIRONMENTAL ENGINEER

- **Description:** A specialized auditor for waste management, water treatment, and pollution control technologies. This model is designed to review technical reports regarding site remediation, effluent analysis, and hazardous material handling to ensure compliance with specialized environmental engineering nomenclature and international regulatory standards.
- **Architecture: 8B Parameters** (Llama 3.1) with an expanded **128K context window** to digest long-form regulatory filings and environmental audits.
- **Hardware:** Compact 4.7GB size; optimized for high-speed local inference.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/environmentalengineer%5D(https://ollama.com/ALIENTELLIGENCE/environmentalengineer)
- **Command:** `ollama pull ALIENTELLIGENCE/environmentalengineer`

ENVIRONMENTAL CONSULTING

- **Description:** An **Advanced AI Environmental Consulting** service model. Unlike the technical engineering version, this model is broader and more strategic, specializing in **Sustainability Assessments**, carbon footprint reporting, and air/water quality monitoring planning. It is the ideal auditor for high-level environmental impact studies and government regulatory submissions where the language must meet specific policy and compliance standards.
- **Hardware:** Designed to run as a persistent lightweight agent on **8GB–16GB RAM** workstations.
- **Link:** [https://ollama.com/ALIENTELLIGENCE/environmentalconsulting%5D(https://ollama.com/ALIENTELLIGENCE/environmentalconsulting)
- **Command:** `ollama pull ALIENTELLIGENCE/environmentalconsulting`

STEP C: PYTHON ENVIRONMENT & COMPILING TO EXECUTABLE

You can run AI DOCUMENT TRANSLATOR as a script (for development) or compile it into a standalone `.exe` that runs on other machines without needing Python installed.

OPTION 1: RUNNING AS A SCRIPT (DEV MODE)

1. **Install Python 3.10+**: [https://python.org%5D(https://python.org)
2. **Create Virtual Env (Recommended but Optional)**:
   - Windows: `python -m venv venv` then `venv\Scripts\activate`
   - Mac/Linux: `python3 -m venv venv` then `source venv/bin/activate`
3. **Install Libraries**: `pip install ollama python-docx langdetect pyinstaller`

OPTION 2: COMPILING TO STANDALONE EXECUTABLE (DEPLOYMENT MODE) We provide a pre-configured `translator_gui.spec` file. This file automatically handles hidden imports (for `langdetect` and `docx` templates) and excludes heavy libraries (numpy, pandas) to keep the file size small.

You do NOT need a virtual environment for this to work, provided the required libraries are installed in your current Python environment.

1. **Ensure Requirements are Installed**: Make sure you have installed the necessary libraries (see Option 1, Step 3).
2. **Run the Build Command**: Open your terminal/command prompt in the project folder and run:

   ```
   python -m PyInstaller translator_gui.spec
   ```

   (Note: Using `python -m ...` ensures the correct PyInstaller version is used, avoiding path errors).
3. **Locate the Application**: The standalone executable (`translator_gui.exe`) will be generated in the `dist/` folder. You can move this file to any other computer (Windows 10/11) and it will run immediately.

[WINDOWS BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean ^
  --name "translator_gui" ^
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas ^
  --exclude-module scipy --exclude-module IPython --exclude-module pytest ^
  translator_gui.py
```

[macOS BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean \
  --name "translator_gui" \
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas \
  --exclude-module scipy --exclude-module IPython --exclude-module pytest \
  translator_gui.py
```

[LINUX BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean \
  --name "ai_translator_eng_v5" \
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas \
  --exclude-module scipy \
  translator_gui.py
```

3. **Result:** The standalone app will be in the `dist/` folder.

---

4. HOW TO RUN

To ensure a successful translation and document reconstruction, follow these operational steps:

1. **Initialize the Environment and Application**

- **Verify Ollama:** Ensure the Ollama service is active in your system tray. The application requires a live connection to the Ollama local API to detect and communicate with your models.
- **Launch the Interface:** Run `python translator_gui.py` from your terminal or double-click the compiled `translator_gui.exe` in the `dist/` folder.
- **Model Sync:** Upon startup, the GUI will automatically query your local Ollama library. Confirm that your desired models (e.g., TranslateGemma) appear in the "Primary Model" selection menu.

2. **Select and Ingest Documents**

- **File Selection:** Click the **"Browse"** button. You can select a single `.docx` file or hold `Ctrl` to select multiple files for batch processing.
- **Queue Ingestion:** The paths to your selected files will populate the queue. The engine will perform an immediate background scan to calculate the total number of processable text blocks, ensuring the progress bar reflects actual workload rather than just file counts.

3. **Configure Linguistic and Model Parameters**

- **Language Selection:** Set your **Source** and **Target** languages using the dropdown menus. It is critical to select specific regional variants (e.g., Portuguese-PT vs. Portuguese-BR) to ensure the LLM adopts the correct technical terminology.
- **Model Assignment:** Choose your **Primary Model** (e.g., `translategemma:27b` for high-fidelity technical work).
- **Expert Consultant (Optional):** If the document is highly specialized, enable the **Consultant Mode** and select a domain-specific model (e.g., Structural Llama) to perform an accuracy audit on the primary output.

4. **Execute the Translation Process**

- **Start Command:** Click the **"Start Translation"** button.

- **Digital Twin Processing:** The system will begin the five-step "Digital Twin" workflow: Atomic Dissection (parsing XML), DNA Extraction (saving styles), Compiler Injection (sending tagged text to the AI), Sanitization (cleaning output), and Surgical Reconstruction (rebuilding the file).
- **Real-Time Monitoring:** Observe the status log for live updates. If the system encounters a difficult segment that risks "chatty" output, you will see a "Panic Mode" notification as the system automatically retries the block with higher restrictions.

5. **Retrieve and Verify Output**

- **Auto-Save:** Translated files are saved automatically in the original source folder with a language-coded suffix (e.g., `Project_Manual_FR.docx`).
- **Final Inspection:** Open the file in Microsoft Word. Verify that complex elements—such as hierarchical numbering, table borders, headers, and specific font colors—have been preserved exactly as they appeared in the source.

---

5. BIBLIOGRAPHY & RECOMMENDED READING

This section provides a curated list of authoritative literature covering the evolution of machine translation, the architecture of Large Language Models (LLMs), and the engineering principles required to deploy them in production.

[A] FOUNDATIONAL AI & TRANSFORMER ARCHITECTURES

- **Jurafsky, D., & Martin, J. H. (2026).** ***Speech and Language Processing*** **(3rd ed. draft). Stanford University.**
- **Description:** Often referred to as the "Bible" of NLP, the latest edition includes extensive new chapters on Transformers, Large Language Models, and Machine Translation. It provides the most rigorous academic foundation for understanding how machines process human syntax.
- **Tunstall, L., von Werra, L., & Wolf, T. (2022).** ***Natural Language Processing with Transformers: Building Language Applications with Hugging Face.*** **O'Reilly Media.**
- **Description:** A practical guide written by the creators of the Hugging Face library. It explains the mechanics of the Transformer architecture (Attention, Encoders, Decoders) and how to fine-tune models for tasks like document translation.
- **Raschka, S. (2025).** ***Build a Large Language Model (From Scratch).*** **Manning Publications.**

- **Description:** This book deconstructs the ”black box” of LLMs, guiding readers through coding the core building blocks—such as attention mechanisms and positional encodings—using basic PyTorch.

[B] MACHINE TRANSLATION & LOCALIZATION TECHNOLOGY

- **Koehn, P. (2020). *Neural Machine Translation.* Cambridge University Press.**
- **Description:** Written by one of the pioneers of the field, this book is the definitive resource on Neural Machine Translation (NMT). It covers the transition from statistical methods to the deep learning models used by modern engines like DeepL and Google Translate.
- **Kenny, D. (Ed.). (2022). *Machine Translation for Everyone: Empowering Users in the Age of Artificial Intelligence.* Language Science Press.**
- **Description:** This volume focuses on ”Machine Translation Literacy,” teaching users how to work *with* AI rather than against it. It is highly relevant for professionals managing automated workflows in engineering and legal sectors.
- **Sun, S., Liu, K., & Moratto, R. (Eds.). (2025). *Translation Studies in the Age of Artificial Intelligence.* Routledge.**
- **Description:** A comprehensive exploration of how AI is redefining translation theory and practice. It addresses the ethical dilemmas and technical shifts in global markets as human-AI collaboration becomes the standard.
- **Poibeau, T. (2017). *Machine Translation.* MIT Press.**
- **Description:** Part of the MIT Press Essential Knowledge series, this book offers a clear and accessible history of MT, from early Cold War efforts to the dawn of the neural era.

[C] AI ENGINEERING & PRODUCTION SYSTEMS

- **Huyen, C. (2025). *AI Engineering.* O'Reilly Media.**
- **Description:** A follow-up to her seminal work on ML systems, this book explores how modern foundation models are scaled in real-world settings. It emphasizes engineering discipline—reproducibility, monitoring, and infrastructure.
- **Iusztin, P., & Labonne, M. (2025). *The LLM Engineering Handbook.* Leanpub/Packt.**
- **Description:** A hands-on resource focused on the end-to-end lifecycle of LLM products. It covers practical implementations of Retrieval-

Augmented Generation (RAG), prompt optimization, and model evaluation techniques used in this application.

- **Bouchard, L. F., & Peters, L. (2025). *Building LLMs for Production*. O'Reilly Media.**
- **Description:** This book focuses on operational excellence, addressing latency, cost optimization, and observability—critical factors for deploying local LLMs via runtimes like Ollama.

[D] CRITICAL PERSPECTIVES & THE ALIGNMENT PROBLEM

- **Christian, B. (2020). *The Alignment Problem: Machine Learning and Human Values*. W. W. Norton & Company.**
- **Description:** An investigation into the "Alignment Problem"—the challenge of ensuring AI systems reliably do what humans want them to do without unintended consequences.
- **Narayanan, A., & Kapoor, S. (2024). *AI Snake Oil*. Princeton University Press.**
- **Description:** Two Princeton academics separate fact from hype, explaining where AI (specifically language models and machine translation) truly excels and where it fails.