notification duty, data retention, etc. It ensures compliance with privacy laws and typically is attached to the MSA when needed.

- **Other Miscellaneous Templates:** We also maintain templates for **Change Orders** (to document modifications to scope post-signature), **Acceptance Forms** (client sign-off that work is completed), and if applicable, **Equipment Lease or Finance Agreements** (in cases where we lease hardware to the client rather than sell outright). These are included in this vault for completeness but used as needed.

Usage Guidance: Each template is annotated with comments on how to fill in client-specific information (e.g., a cover sheet or highlighted sections to replace with names, dates, fees, etc.). When preparing a contract for a new client, follow these steps:

1. **Review and Customize:** Start with the MSA and NDA for all new clients. Fill in the client's name, address, the effective date, and tailor any service descriptions in exhibits. Remove any sections that don't apply and add specifics relevant to the client (for example, if the client has requested a unique provision or if local law requires an adjustment). Do the same for the SLA if we are committing to ongoing support or uptime, ensuring the service levels match what was promised in proposals.
2. **Add Project SOW:** Draft the Statement of Work for the initial project. Use the SOW template and include all details of the deployment (from the Blueprint and related docs). For instance, list the exact equipment to be installed, the security measures, and any training or support included. Be detailed—this prevents misunderstandings later. The SOW should reference the MSA as governing terms.
3. **Legal Review:** Although these templates were originally drafted by legal counsel, any non-standard changes or any client-proposed modifications need review. Internally, have our legal advisor or contracted attorney review the filled documents if there are significant edits. This is especially true for high-value contracts or those involving unique liability concerns.
4. **Client Signature:** Once finalized, the documents are sent to the client for signature. Typically, the NDA might be signed first (to enable free discussion), then the MSA (with SLA and DPA as needed), and then each SOW. We keep signed copies in our secure document repository (and a hard copy in the project binder possibly).
5. **Ongoing Use:** For subsequent projects with the same client, we usually do not need a new MSA/NDA (unless the original term has lapsed or needs update); we just create a new SOW. Keep track of any expiration dates (some NDAs might have a term for how long info must be kept secret – our template usually says indefinitely for trade secrets, 5 years for other confidences, etc., but ensure no lapse).

All templates have been updated as of 2025 to reflect current regulations and company policies. They contain robust confidentiality clauses reflecting our privacy-centric ethos. For example, the MSA explicitly notes that we **never share client data with third parties** without consent and that we maintain strict security measures (which ties into our security protocols document). This alignment between what we *promise contractually* and what we *implement technically* helps build trust and legal safety.

Confidential Handling: These templates are marked confidential and are for internal use. They should not be given out in raw form to anyone except for the purpose of drafting a client agreement. Always produce a PDF of the completed contract for sending to the client, and do not send them the editable templates. Maintain version control—if any changes are made to the master templates (for instance, by legal counsel), update the vault here and record the change in the change log appendix of this document.

By using these **Client Contract Templates**, we ensure every client engagement is governed by clear, fair, and protective terms. This not only mitigates risk but also reinforces to the client that we operate professionally and value privacy and security in every aspect, including our paperwork.

On-Prem AI Deployment Scripts

This section provides an **automated deployment guide for on-premises AI tools**, specifically focusing on local Large Language Model (LLM) solutions. We supply a set of scripts (packaged in a ZIP archive) that set up the entire stack: **Ollama**, **llama.cpp**, **Open WebUI**, and **LM Studio**. These tools enable running advanced AI models (like LLaMA, GPT-style models) on the client's own hardware, without relying on cloud services. By using our scripts, one can get a fully functioning AI environment quickly, ensuring that all AI computations and data remain on-site (maximizing privacy).

Purpose & Scope: Many clients require AI capabilities (for example, a private GPT-4-style assistant for their data) but cannot send sensitive data to external APIs. Our on-prem AI deployment solves this by installing open-source or self-hosted AI runtime environments. The provided scripts automate installation of the following components:

- **Ollama** – a lightweight CLI tool for running large language models locally. It manages model downloading and provides a simple interface (including an API) for running prompts against local models.
- **llama.cpp** – a foundational backend (in C/C++) that allows efficient inference of LLaMA-family models on CPU (and GPU if available). This is used under the hood by many apps. We include it so that even if other tools fail, you can run models via command-line.
- **Open WebUI** – an open-source web interface for generative AI models. It provides a user-friendly chat UI in a browser, supporting multiple backends (including llama.cpp and Ollama). This allows clients to chat with their local AI model from any PC on the network via a web browser.
- **LM Studio** – a desktop application (with GUI) for running local LLMs, which some users prefer for its simplicity. It's essentially a polished app where users can load models and interact with them without using the command line.

The goal is to deploy all of the above, so the client has flexibility in how they use the AI: programmatically via Ollama, in a web browser via Open WebUI, or in a desktop app via LM Studio. All components will be configured to run **within the client's network, offline**.

Script Functions: The `deploy_onprem_ai.sh` (for example) script included will perform several tasks automatically:

1. **Environment Preparation:** Update the system (for Linux, `apt update && apt upgrade` for instance), install needed dependencies (like Git, Python3, Node.js for some UI perhaps, C++ build tools, etc.). Ensures GPU drivers are present if an NVIDIA GPU is in use for acceleration (and installs CUDA or appropriate libraries when possible).
2. **Install Ollama:** The script uses the official installation method for Ollama. For example, on Linux it would run the one-line installer provided by Ollama (e.g., `curl -sSfL https://ollama.ai/install.sh | sh` which fetches the latest release). After installing, it verifies by running `ollama version`.