**Document 1: Software Requirements Specification (SRS)**

**OllamaUI – Software Requirements Specification**

**Document Identifier: SRS-OllamaUI-001**

**Revision: 1.3**

**Date: 2025-02-23**

**1. Introduction**

**1.1 Purpose**

**This SRS document specifies the functional and nonfunctional requirements for OllamaUI, a cross-platform desktop application built with Flutter. OllamaUI provides a unified user interface for managing the Ollama engine, enabling users to interact via a chat interface and manage models. The document is intended for stakeholders, developers, and testers.**

**1.2 Scope**

**OllamaUI will:**

**MVP Features:**

**Engine Management & Status:**

**Automatically detect, install/update (if necessary), and launch the Ollama engine as a background service.**

**Display the current engine status via in-app indicators and system tray integration.**

**Chat & Engine Interaction:**

**Provide a primary chat interface for users to type text queries.**

**Display engine responses in a conversational format.**

**Allow multimodal input (file/image attachment) that is pre-processed to extract text.**

**Basic Model Management:**

**List available models (retrieved via commands such as “ollma list”).**

**Allow users to load a selected model and delete models when needed.**

**Full Features (Post-MVP):**

**Query History & Favorites:**

**Log user queries with timestamps.**

**Enable marking queries as favorites with a dedicated view.**

**Performance Monitoring & Alerts:**

**Monitor real-time performance metrics (e.g., TPS, CPU, memory) with visual alerts.**

**Downloads & Updates:**

**Manage model downloads and updates, with progress indicators and controls (pause/resume).**

**Navigation and Global UI Components:**

**Provide a unified navigation menu (sidebar or top bar) to access different sections.**

**Multimodal Input Handling:**

**Offer a dedicated interface for file/image input (via drag-and-drop or file pickers).**

**Nice-to-Have Features:**

**Search Integration:**

**Allow users to search for models or content using an external API.**

**Advanced Model Management & Comparison:**

**Enable detailed model comparison using side-by-side metrics.**

**Diagnostics & Recommendations:**

**Run local system diagnostics and recommend the optimal model based on hardware performance.**

**1.3 Definitions, Acronyms, and Abbreviations**

**Ollama Engine: The backend process managing AI models.**

**UI: User Interface.**

**MVP: Minimum Viable Product.**

**TPS: Tokens Per Second.**

**SRS: Software Requirements Specification.**

**SPEC: Software Specification Document.**

**1.4 References**

**ISO/IEC/IEEE 29148:2011 – Requirements Engineering.**

**IEEE Std 830-1998 – Recommended Practice for SRS.**

**Documentation for the ollama.com/search API (for nice-to-have features).**

**1.5 Overview**

**This document details the required functionality, performance, reliability, and design constraints for OllamaUI. It serves as the basis for development, verification, and future enhancement.**

**2. Overall Description**

**2.1 Product Perspective**

**OllamaUI is a standalone desktop application that provides a unified interface to manage and interact with the Ollama engine. It is built with Flutter for cross-platform compatibility (Windows, macOS, Linux) and leverages a clean, modular layered architecture.**

**2.2 Product Functions**

**Engine Management & Status (MVP):**

**Automated engine detection, installation, and background operation.**

**Status indicators via system tray and in-app messages.**

**Chat & Engine Interaction (MVP):**

**A chat interface for textual queries and engine responses.**

**Integration of multimodal inputs (file/image attachment) pre-processed to extract text.**

**Basic Model Management (MVP):**

**Listing of models with basic metadata.**

**Load and delete operations.**

**Future Full Features:**

**Query History & Favorites, Performance Monitoring & Alerts, Downloads & Updates, Global Navigation, and expanded multimodal input handling.**

**Nice-to-Have Features:**

**Search integration, advanced model comparison, and diagnostics with recommendations.**

**2.3 User Classes**

**End Users: Both technical and non-technical users who interact with the engine.**

**Administrators: Users responsible for managing engine operations and performance.**

**Developers/QA Engineers: Those involved in development, testing, and maintenance.**

**2.4 Operating Environment**

**Platforms: Windows, macOS, Linux.**

**Background Services: Engine runs as a background service with OS-specific integrations (via plugins).**

**Network: Secure network communication via HTTPS (future security enhancements planned).**

**2.5 Constraints**

**The application is developed using Flutter.**

**Must work uniformly across all supported desktop platforms.**

**Automated engine installation must adhere to platform-specific guidelines.**

**Future security integration is planned via abstract interfaces.**

**2.6 Assumptions**

**The Ollama engine and search API (for nice-to-have features) are stable and accessible.**

**Target systems have sufficient hardware resources for engine execution and diagnostics.**

**Necessary Flutter plugins for system tray, file picking, and background processing are available and reliable.**

**3. Specific Requirements**

**3.1 Functional Requirements**

**ID Requirement Acceptance Criteria**

**FR-01 Engine Management & Status: Automatically detect and, if necessary, install/update the Ollama engine; launch it as a background service; display status indicators. - Engine is verified at startup; if missing, installation is triggered automatically.**

**- A system tray icon and in-app indicator display the engine’s status (running, installing, error).**

**FR-02 Chat & Engine Interaction: Provide a chat interface for user queries and engine responses; support multimodal input (file/image attachment with pre-processing). - Users can type and send text queries.**

**- Engine responses appear in a conversational format.**

**- Files/images can be attached and processed (e.g., OCR conversion to text).**

**FR-03 Basic Model Management: List available models; allow loading a selected model; enable deletion of models. - Models are listed within 5 seconds.**

**- Metadata is clearly displayed.**

**- Loading and deletion operations perform without errors.**

**FR-04 Query History & Favorites (Full Feature): Log user queries with timestamps; enable marking favorites with a dedicated view. - User queries are stored persistently.**

**- Favorites can be marked and filtered in a history view.**

**FR-05 Performance Monitoring & Alerts (Full Feature): Monitor key performance metrics (TPS, CPU, memory) and trigger alerts if thresholds are exceeded. - Real-time metrics are displayed on a dashboard.**

**- Visual alerts trigger upon threshold breaches.**

**FR-06 Downloads & Updates (Full Feature): Manage model downloads and updates with progress indicators and controls for pausing/resuming. - Download progress is visibly tracked.**

**- Pausing and resuming functions work as intended.**

**FR-07 Navigation and Global UI Components (Full Feature): Provide unified navigation (sidebar or top bar) for different modules. - A navigation menu allows users to switch between chat, model management, history, performance, etc.**

**FR-08 Multimodal Input Handling (Full Feature): Provide an interface for file/image input (via drag-and-drop and file picker) with conversion to text. - Users can attach files or images and see a preview.**

**- Non-text inputs are processed to extract text.**

**3.2 Nonfunctional Requirements**

**Performance:**

**UI interactions (e.g., chat responses, model loading) must respond within 3–5 seconds.**

**Background operations (engine installation, file processing) must minimize system impact.**

**Usability:**

**The UI must be intuitive, accessible (supporting screen readers, keyboard navigation, high contrast), and provide consistent feedback.**

**Reliability:**

**Robust error handling and logging are in place.**

**Local data (query history, model lists) persists across sessions.**

**Security:**

**Network communications use HTTPS.**

**The architecture is designed for future integration of data encryption and secure storage.**

**Maintainability:**

**Code is modular with clear interfaces.**

**Module-specific documentation is maintained.**

**Scalability:**

**The design supports future enhancements (advanced features, diagnostics) without major refactoring.**

**4. Appendices**

**4.1 Glossary**

**Ollama Engine: Backend process for managing AI models.**

**UI: User Interface.**

**TPS: Tokens Per Second.**

**MVP: Minimum Viable Product.**

**4.2 References**

**ISO/IEC/IEEE 29148:2011**

**IEEE Std 830-1998**

**Third-party plugin documentation (system\_tray, file\_picker, etc.)**

**4.3 Traceability Matrix**

**A separate spreadsheet maps each FR (FR-01 to FR-08) to design components and test cases.**