Software Requirements Specification

for

AUTOPENT

Version 1.0 approved

Prepared by Himanshu Gaur (23BCY10127)
Abhinav Mehra (23bcy10015)
Tanya Bharti(23bce11155)
Rananjay Singh Chauhan(23bai10800)

VIT BHOPAL UNIVERSITY

August 24, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) describes the functionality and design requirements for the software product **AutoPent v1.0**. AutoPent is an AI-powered penetration testing and security automation tool that integrates traditional recon and scanning tools with a large language model (**LLaMA 3.1**) for intelligent vulnerability analysis and report generation.
This document outlines all core modules—Reconnaissance, Vulnerability Scanning, AI Analysis, and Reporting—and serves as a reference for developers, testers, and stakeholders. The scope includes both standalone and modular deployments.

1.2 Document Conventions

- **Bold** text highlights section headings.
- Monospace font indicates commands, code, or tool-specific references.
- All functional requirements are prefixed with "F#", e.g., F1, F2, etc.
- Priorities are implied: unless explicitly stated, all listed features are of high priority for MVP.
- All external tools or APIs are capitalized (e.g., SQLMap, Nmap, Metasploit).

1.3 Intended Audience and Reading Suggestions

This SRS is intended for the following roles:

- **Developers & Engineers** – For understanding system structure, required modules, and implementation details.
- **Testers** – For identifying functional modules and testable features.
- **Project Managers & Stakeholders** – For tracking deliverables, scope, and objectives.
- **Clients / End-Users** – For a high-level overview of the system's capabilities.

Reading sequence:

- Start with Section 1 for an overview.
- Move to Section 2 for system behavior.
- Refer to Section 3 for technical and functional requirements.

1.4 Product Scope

AutoPent is a cybersecurity automation framework designed to reduce human effort in penetration testing by up to 70%. It combines traditional scanning tools with intelligent AI capabilities using **LLaMA 3.1** to:

- Perform automated reconnaissance and vulnerability scans
- Identify OWASP Top 10 vulnerabilities
- Suggest real-world exploits
- Provide human-readable reports with AI-based remediation
  AutoPent aligns with the business goal of making security audits faster, smarter, and cost-effective for organizations without full-time security teams.

1.5 References

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- Metasploit Exploit Framework – Rapid7 (https://www.metasploit.com)
- SQLMap Documentation (http://sqlmap.org/)
- Nmap Scripting Engine Docs (https://nmap.org/book/nse.html)
- LLaMA 3.1 – Meta AI (https://ai.meta.com/llama)
- IEEE SRS Template Guidelines

2. Overall Description

2.1 Product Perspective

AutoPent is a standalone cybersecurity automation tool designed for on-premise use. It is not a follow-on to

any previous system but rather a new and self-contained framework.
It integrates open-source tools such as Nmap, SQLMap, Nikto, and Metasploit with a local large language model (LLaMA 3.1) to automate end-to-end penetration testing and AI-assisted vulnerability analysis. The product acts as an integrated interface between traditional scanners and AI, bridging manual cybersecurity efforts with automated decision-making.

2.2 Product Functions
AutoPent performs the following major functions:
- **Automated Reconnaissance:** Domain and subdomain enumeration using Shodan, BinaryEdge, and Onyphe.
- **Vulnerability Scanning:** OWASP top 10 tests using Nikto, Nmap (NSE scripts), and SQLMap.
- **Exploit Mapping:** Suggests relevant Metasploit modules based on discovered vulnerabilities.
- **AI Analysis:** Uses LLaMA 3.1 to summarize findings and suggest remediation in human-readable language.
- **Security Report Generation:** Outputs a clean, downloadable PDF report with recon, vulnerabilities, and AI recommendations.

2.3 User Classes and Characteristics
AutoPent is designed for the following user classes:

User Class	Description
Cybersecurity Engineers	Technical users who run scans and interpret outputs regularly.
Startup Founders / DevOps	Non-security experts who need quick and clear website vulnerability insights.
IT Service Providers	Firms managing multiple client websites needing scalable audit capabilities.
Cybersecurity Educators	Use AutoPent as a teaching tool for AI-based pentesting labs.

2.4 Operating Environment
- **Hardware Requirements:**
  - Minimum: 8GB RAM, mid-range GPU (e.g., NVIDIA RTX 3050)
  - Disk: ~10GB for model files, tools, and logs
- **Operating System:**
  - Linux (preferred), Windows (supported)
- **Software Dependencies:**
  - Python 3.10+
  - Tools: Nikto, SQLMap, Nmap, Metasploit
  - Libraries: llama_cpp, pdfkit, Flask, requests, etc.
  - Local LLM: LLaMA 3.1 (via GGUF format)

2.5 Design and Implementation Constraints
- **Offline Capability:** The model must run without internet after setup.
- **Open-Source Tools:** Relies on SQLMap, Nmap, and other community tools; licenses must be honored.
- **GPU Dependency:** LLM features require CUDA-compatible GPU.
- **Local Execution:** No cloud or API integration for AI analysis to preserve privacy.
- **Modular Architecture:** All modules must be separable and independently testable.

2.6 User Documentation
The following documents will be provided:
- **Installation Guide:** For setting up the environment and dependencies.

- **User Manual:** Step-by-step instructions to operate each module.
- **Troubleshooting Guide:** Common errors, solutions, and support contacts.
- **Model Deployment Guide:** Explains setting up the LLaMA 3.1 model locally.

2.7 Assumptions and Dependencies

- Users will install required tools and dependencies as per the provided setup guide.
- Internet access is available during the initial configuration for tool and model downloads.
- Assumes that the target systems allow scanning (authorized usage).
- Assumes users possess basic Linux command-line knowledge if used without GUI.
- Continued updates to open-source tools (Nmap, SQLMap, etc.) are assumed for compatibility.

3. External Interface Requirements

3.1 User Interfaces

AutoPent provides both CLI (Command-Line Interface) and GUI (Graphical User Interface via Streamlit) interfaces:

- **CLI Features:**
  - Command-driven execution with real-time logs.
  - Prompts for domain/target, scan types, and report generation options.
  - Clear error outputs and summary messages.
- **GUI Features (Optional Frontend Layer):**
  - Modular buttons for Recon, Vulnerability Scan, AI Analysis, and Report Download.
  - Text area to display live output of each module.
  - PDF Report Download button post-analysis.
  - Minimalistic layout optimized for clarity and responsiveness.
- **Standard Components:**
  - Help tooltip on each module.
  - Input field validations for URL/domain.
  - Consistent buttons and spacing across modules.

GUI layout is documented separately in the UI specification document.

3.2 Hardware Interfaces

AutoPent requires basic interaction with system hardware to perform its operations efficiently:

- **Supported Devices:**
  - CUDA-enabled NVIDIA GPUs (minimum: RTX 3050) for LLaMA 3.1 model inference.
  - CPU fallback is available, but with slower processing.
- **Hardware Utilization:**
  - GPU memory is used for LLM-based analysis.
  - Disk I/O for log generation, temporary report storage, and model loading (~10GB).
  - USB/network adapter (optional) for external penetration testing extensions.

No direct low-level hardware communication is performed. All hardware interfaces are abstracted via drivers and Python packages.

3.3 Software Interfaces

AutoPent integrates multiple third-party tools and libraries. Key software interface details include:

- **Third-Party Tools:**
  - **Nmap** (v7.94): Network scanning and NSE scripting.
  - **Nikto** (v2.5.0): Web server vulnerability scanning.
  - **SQLMap** (latest): SQL injection testing.

  - **Metasploit** (via pymetasploit3): Exploit suggestion/mapping.
  - **LLaMA 3.1** (via llama-cpp-python): Local LLM inference for report generation.
- **Python Libraries:**
  - subprocess, re, json, os: System commands and parsing.
  - llama_cpp: Integration with LLaMA 3.1 model.
  - pdfkit: PDF report generation.
  - Flask / Streamlit: For optional GUI/REST integration.
  - requests, shodan, binaryedge: API-driven reconnaissance.
- **Shared Data Format:**
  - All inter-module data is serialized in JSON.
  - Analysis reports are HTML before being converted to PDF.

3.4 Communications Interfaces

AutoPent supports limited online communication for reconnaissance and setup:

- **APIs Used (Optional for Recon):**
  - Shodan, BinaryEdge, Onyphe — for passive reconnaissance.
  - All API communication uses HTTPS (TLS-encrypted).
- **Networking Requirements:**
  - Outbound access to scanning APIs (optional).
  - Offline mode available for vulnerability scanning and AI analysis (after LLM is loaded).
- **Security Standards:**
  - All communications (if used) are encrypted.
  - User tokens/API keys are stored locally and never transmitted externally.
  - No data is sent to the cloud; all operations are local by design.

4. System Features

4.1 Reconnaissance Automation Module

4.1.1 Description and Priority

This feature automates the collection of OSINT data about a target domain using tools like Shodan, BinaryEdge, and WHOIS.
Priority: High
Benefit: 8 | **Penalty:** 9 | **Cost:** 3 | **Risk:** 2

4.1.2 Stimulus/Response Sequences

- **User Input:** Target domain or URL.
- **System Response:** Collects data such as open ports, services, DNS, IPs, subdomains, and technologies used.

4.1.3 Functional Requirements

- **REQ-1:** System must accept valid domain/IP input and sanitize it.
- **REQ-2:** Integrate APIs (Shodan, BinaryEdge) securely using keys.
- **REQ-3:** Return output in a structured JSON format.
- **REQ-4:** Must timeout if API requests exceed 15 seconds or fail repeatedly.
- **REQ-5:** Allow offline recon fallback when no APIs are available.

4.2 Vulnerability Scanning

4.2.1 Description and Priority

Performs automated scans using Nikto, Nmap NSE, and SQLMap (top 10 DB types only) to find web and network vulnerabilities.
Priority: High
Benefit: 9 | **Penalty:** 8 | **Cost:** 3 | **Risk:** 3

4.2.2 Stimulus/Response Sequences

- **User Input:** Target URL
- **System Response:** Executes all scanners, parses and filters results, flags critical issues.

4.2.3 Functional Requirements

- **REQ-6:** Run Nikto with filtered output (top 25 CVEs).
- **REQ-7:** SQLMap should limit to common DBs and summarize results in under 2 pages.
- **REQ-8:** Extract known versions for mapping with Metasploit.
- **REQ-9:** Save all results in JSON format and remove raw logs from report.
- **REQ-10:** Identify exploits using predefined signature-exploit map.

4.3 AI-Powered Security Analysis

4.3.1 Description and Priority

Uses LLaMA 3.1 locally to generate security summaries and fix recommendations.
Priority: High
Benefit: 9 | **Penalty:** 9 | **Cost:** 4 | **Risk:** 2

4.3.2 Stimulus/Response Sequences

- **User Input:** No input needed; auto-processes recon + scan results.
- **System Response:** Summarizes vulnerabilities and suggests remedies in plain English.

4.3.3 Functional Requirements

- **REQ-11:** Use LLaMA 3.1 with max_tokens ≤ 800 to avoid overflow.
- **REQ-12:** Prompt must include recon, OWASP, and source code scan summary.
- **REQ-13:** AI must return text output only (no code/JSON).
- **REQ-14:** Generate clear and non-technical suggestions for most findings.

4.4 Report Generation

4.4.1 Description and Priority

Generates an HTML + PDF report combining recon, scan results, source analysis, and AI insights.
Priority: High
Benefit: 8 | **Penalty:** 8 | **Cost:** 2 | **Risk:** 1

4.4.2 Stimulus/Response Sequences

- **User Action:** Click “Generate Report” or auto-trigger post-analysis.
- **System Response:** Builds structured report and saves as .pdf.

4.4.3 Functional Requirements

- **REQ-15:** Render all content in HTML first, then convert to PDF using pdfkit.
- **REQ-16:** Include visual sections with <pre> tags to preserve formatting.
- **REQ-17:** Store reports with timestamps in a /reports directory.
- **REQ-18:** Allow optional user-defined filename.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The product must complete a full reconnaissance, vulnerability scan, and AI-based report generation within **7–10 minutes** on mid-tier systems (e.g., 8GB RAM, RTX 3050).
- LLaMA 3.1 should generate recommendations within **15 seconds** per analysis block.
- The system must be able to handle up to **5 simultaneous scans** without performance degradation on supported hardware.
- Scans and report generation must remain functional **offline**, with no dependency on external cloud APIs (except when optionally enabled for recon).

5.2 Safety Requirements

- The system must **not perform actual exploitation** unless explicitly approved by the user (commented code by default).
- Logs and reports must not be auto-uploaded or sent externally — **local storage only** by default.
- Any destructive or high-impact module must require **user confirmation** before execution (e.g., Metasploit modules).
- If running on a shared server, ensure that system processes do not access or affect other environments.

5.3 Security Requirements

- All API keys (Shodan, BinaryEdge, etc.) must be stored securely and **never hardcoded** in the application.
- Results containing sensitive data must be encrypted locally using a **configurable password**.
- The app must support **user authentication** (username + password) if deployed as a local web app.
- Audit logs should track user inputs and command executions to ensure accountability in shared setups.
- Complies with **OWASP Top 10** security principles during internal communication.

5.4 Software Quality Attributes

- **Portability:** Should run on major OS platforms (Linux, Windows 10+, macOS).
- **Reliability:** Should produce accurate reports with <2% error tolerance in data processing.
- **Maintainability:** Modular codebase using separate files for recon, scan, analysis, and reporting for easy updates.
- **Usability:** Command-line version supported; optional UI can provide guided input for non-technical users.
- **Offline Capability:** Works completely offline except for optional recon APIs.
- **Testability:** Each module is independently testable via unit testing.

5.5 Business Rules

- Only **authorized personnel** should be allowed to initiate a scan.
- Generated reports must carry **timestamp and user ID** for traceability.
- AI analysis must **never suggest offensive actions** unless explicitly enabled in a secure setting.
- Licenses are **one-time** per organization; resale or modification is restricted unless permitted.

6. Other Requirements

- **LLM Dependency:** Requires LLaMA 3.1 model checkpoint to be locally downloaded (~4GB).
- **Legal Use Warning:** Software must include a disclaimer stating that it is for **authorized testing purposes only**.
- **Deployment Mode:** Installable via CLI/GUI or as a Docker container.
- **Documentation:** A deployment and usage manual will be included with the purchase.
- **Internationalization:** Report output language defaults to English, but translation-ready for other languages if extended.

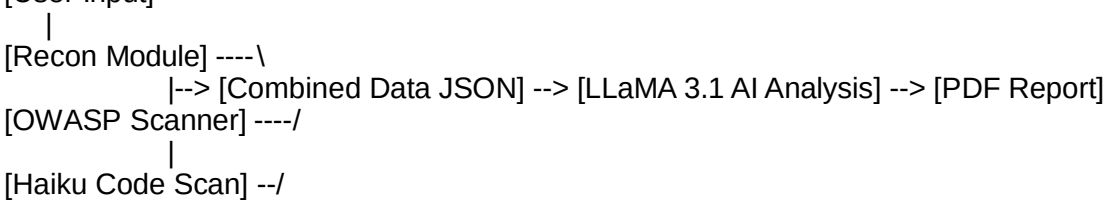
Appendix A: Glossary

Term	Definition
AutoPent	AI-powered automated penetration testing and reporting framework.
LLM	Large Language Model, in this case **LLaMA 3.1**, used for security analysis.
Reconnaissance	The process of gathering information about a target system before testing.

Term	Definition
Vulnerability Scan	Automated process to detect security weaknesses in a system or application.
OWASP	Open Web Application Security Project – defines top 10 web security risks.
Metasploit	An exploitation framework used to execute known vulnerabilities.
Haiku	External API/tool used to perform source code vulnerability analysis.
Mistral	Alternative open-source LLM family to LLaMA; not used in this version.
Offline Mode	Ability to use the full tool without an internet connection.
PDFKit	Python library used to generate PDF reports from HTML content.
API Key	Authentication token required to access third-party services securely.
SQLMap	Tool for automated detection and exploitation of SQL injection vulnerabilities.

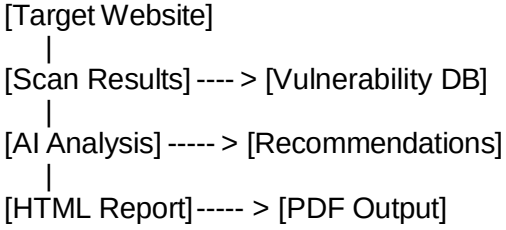
Appendix B: Analysis Models

1. **High-Level Module Diagram:**



2. **Basic Data Flow Diagram:**

3. **Entity-Relationship (ER) Diagram (Simplified)**



Appendix C: To Be Determined List

TBD ID	Description
TBD-01	Final selection between using LLaMA 3.1 or integrating newer 7B Mistral.

TBD ID	Description
TBD-02	UI/UX design specifications for optional GUI frontend.
TBD-03	Optional enterprise licensing terms and customization packages.
TBD-04	Exact deployment instructions for containerization (Docker/K8s format).
TBD-05	Future language/localization support based on market demand.