Software Requirements Specification (SRS)

Project: OptiGuide – LLM-based Supply Chain Optimization Interpreter

Abstract: This Software Requirements Specification (SRS) defines the complete set of requirements for the OptiGuide framework, which integrates the LLaMA 3 Large Language Model with the PuLP solver for supply chain optimization. It includes functional, non-functional, interface, and performance requirements and adheres to IEEE 830 standards for clarity, completeness, and verifiability.

System Specifications

Component	Specification
Large Language Model	LLaMA 3 (Meta AI, 2024)
Optimization Solver	PuLP (Python Linear Programming Library)
Programming Language	Python 3.10
Deployment Environment	Azure Virtual Machines / Local On-Prem
Database	SQLite / CSV-based Data Store

1. Introduction

OptiGuide bridges the gap between supply chain optimization systems and human understanding using LLaMA 3 for reasoning and PuLP for executing optimization models.

2. Scope

The system interprets natural language queries, converts them into optimization constraints, executes models using PuLP, and generates human-readable results. It supports what-if analysis, scenario planning, and privacy-preserving workflows.

3. System Overview

OptiGuide consists of three layers — User Interface, Agent Layer, and Application Layer. LLaMA 3 interprets user queries and generates Python code, which the safeguard validates before execution via PuLP.

4. Architecture Design

The system architecture consists of:
• **User Interface:** Accepts natural language queries.
• **Coder Agent (LLaMA 3):** Translates queries into optimization code.
• **Safeguard:** Validates code integrity before execution.
• **Solver (PuLP):** Executes the optimization model.
• **Interpreter:** Converts solver outputs to readable insights.
• **Database Layer:** Stores historical optimization data.

Data Flow: User → LLaMA 3 → Safeguard → PuLP Solver → Interpreter → User Feedback.

5. Functional Requirements

ID	Requirement Description	Priority
FR-1	System shall interpret natural language queries using LLaMA 3.	High
FR-2	System shall translate interpreted queries into PuLP-compatible Python code.	High
FR-3	System shall execute optimization problems using PuLP.	High
FR-4	System shall visualize optimization results and provide summaries.	Medium
FR-5	System shall validate generated code for syntax and logic before execution.	High

FR-6	System shall store and retrieve optimization data using SQLite or CSV.	Medium
FR-7	System shall allow interactive constraint modification for what-if analysis.	Medium
FR-8	System shall generate dashboards for visualization of performance metrics.	Low

6. Interface Requirements

User Interface: Web or CLI for query input and result visualization.
Software Interface: Integration between LLaMA 3 and PuLP via Python APIs.
Database Interface: SQLite or CSV for storing optimization inputs and outputs.
Hardware Interface: Runs on local or cloud servers (minimum 8GB RAM).

7. System Constraints

• LLaMA 3 token limit restricts query size.
• PuLP supports only linear and integer optimization models.
• No external data transmission permitted (privacy constraint).
• Requires Python 3.10+ and minimum 8GB RAM.

8. Non-Functional Requirements

Performance: System shall respond within 8 seconds for medium-sized optimization tasks.
Reliability: System shall achieve 99% uptime during operational hours.
Security: Sensitive data is processed locally without cloud transmission.
Scalability: Framework supports concurrent multi-user sessions via Azure VMs.
Maintainability: Modular architecture allows easy updates and model replacements.
Usability: Plain English interaction allows non-technical users to understand results.

9. References

1. Beibin Li et al., 'Large Language Models for Supply Chain Optimization,' Microsoft Research, 2023.
2. Meta AI, 'LLaMA 3: Open Foundation Models,' 2024.
3. PuLP Documentation – https://coin-or.github.io/pulp/
4. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.