System Requirement Specification (SRS)

Project Title: Automated Requirements and Usage Context Generation using Large Language Models (LLMs)

# 1. Introduction

## 1.1 Purpose

The purpose of this project is to develop an automated solution utilizing Large Language Models (LLMs) to generate system requirements and usage contexts from diverse perspectives, including user personas. The system aims to streamline the requirements engineering process, enhance inclusiveness, and improve usability by reflecting a wide spectrum of user experiences.

## 1.2 Scope

* Accept natural language inputs from stakeholders.
* Generate requirement documents (functional and non-functional).
* Support multiple user personas (end-users, developers, project managers).
* Detect inconsistencies and conflicts in requirements.
* Provide contextualized usage scenarios.
* Enable negotiation and alignment of conflicting requirements.

## 1.3 Definitions, Acronyms, Abbreviations

* LLM – Large Language Model
* SRS – System Requirement Specification
* Persona – Representative model of a user type

## 1.4 References

* IEEE 830 Standard for Software Requirements Specification
* Research literature on NLP and requirements engineering

# 2. Overall Description

## 2.1 Product Perspective

The product will be a standalone AI-powered tool, accessible via web and mobile platforms. It integrates with:

* LLM APIs (e.g., OpenAI, Hugging Face).
* Document export (Word, PDF, Markdown).

## 2.2 Product Functions

* Requirement extraction from text
* Persona-based requirement generation
* Conflict detection and resolution
* Export of requirement documents
* Interactive dashboard for requirements negotiation

## User Characteristics

* Business Analysts: Need clarity, completeness, and traceability of requirements.
* Developers: Require precise, actionable requirements.
* Managers/Clients: Need summaries, context, and usability insights.

## 2.4 Constraints

* Limited API usage quotas for LLMs.
* Data privacy and ethical usage of inputs.
* Usability across different devices.

## 2.5 Assumptions and Dependencies

* Stable internet connection for LLM API.
* Availability of pre-trained LLM models.
* Support for English as the primary language (multi-language optional).

# 3. System Features (Functional Requirements)

## 3.1 Requirement Input Module

Accept natural language input from stakeholders. Support uploading meeting transcripts, notes, or text files.

## 3.2 Persona-Based Requirement Generator

Generate requirements tailored to different personas. Allow customization of persona profiles.

## 3.3 Requirement Document Generator

Create structured requirements documents (functional + non-functional). Allow export to Word, PDF, and Markdown formats.

## 3.4 Conflict Detection & Resolution

Identify conflicting requirements across personas. Suggest negotiation strategies or compromise requirements.

## 3.5 Usage Context Generator

Generate usage scenarios and context diagrams for requirements. Provide user stories or use cases per persona.

## 3.6 Reporting & Dashboard

Provide summaries of generated requirements. Track conflicts, negotiation status, and persona coverage.

# 4. Non-Functional Requirements

* Performance: Generate documents within 10 seconds for medium-size input (~2000 words).
* Scalability: Handle multiple concurrent users (up to 100).
* Security: Encrypt user data; comply with GDPR-like standards.
* Usability: Simple, intuitive UI for non-technical stakeholders.
* Reliability: 99% uptime availability.
* Portability: Deployable on web, mobile (Kotlin/Flutter), and desktop.
* Maintainability: Modular architecture for easy updates.

# 5. External Interface Requirements

* User Interface: Web & mobile app with intuitive forms and dashboards.
* API Interface: LLM API (OpenAI/Hugging Face), export API for docs.
* Hardware Interface: Standard laptop/mobile device.
* Software Interface: Cloud backend (AWS/Azure), local storage optional.

# 6. System Architecture (High-Level)

* Frontend: React/Flutter (UI for users)
* Backend: Python (Flask/Django) for LLM processing
* Database: PostgreSQL/MongoDB for storing generated requirements
* LLM Integration: OpenAI/Hugging Face API
* Deployment: Cloud (AWS/Azure), mobile app (Kotlin)

# 7. Research Component

The research novelty lies in:

* Automating requirement engineering with LLMs.
* Using persona-driven generation for inclusiveness.
* Automated conflict resolution in requirements, which does not exist in current tools.

# 8. Appendix

* Glossary of requirements engineering terms.
* Persona templates (Business Analyst, Developer, Manager).