Software Requirements Specifications

AI-Powered Automated SRS Generator

Internal Advisor: Dr. Fahad Maqbool
Project Manager: Dr. Saad Razzaq

Project Team

M Talal Ahmed BSCS51F22S074
M Tayyab BSCS51F22S070
M Ibrahim BSCS51F22S078

Submission Date

21 October 2025

_____

Project Manager's Signature

Table of Contents

1. Introduction

The Software Requirements Specification (SRS) document serves as the foundation for the creation of the *AI-Powered Automated SRS Generator*. Its major goal is to provide an in-depth description of the software's functionality, behavior, and constraints. The SRS document is critical in the software development process as it acts as a clear and structured communication tool for various stakeholders involved in the project, ensuring mutual understanding of the system's requirements and expectations.

1.1 Purpose of Document

This Software Requirements Specification (SRS) document provides a comprehensive description of the AI-Powered Automated SRS Generator system. It is intended for development team members responsible for designing, implementing, and testing the system, as well as project supervisors and evaluators who will assess the project. Stakeholders seeking to understand the system's capabilities and limitations, and future maintainers who may enhance or modify the system, should also refer to this document.

The SRS details the functional and non-functional requirements, system constraints, and external interfaces to establish a clear understanding of the project scope and implementation requirements.

1.2 Project Overview

The AI-Powered Automated SRS Generator is a web-based application designed to automate the creation of IEEE 830-compliant Software Requirements Specification documents. The system leverages artificial intelligence and natural language processing to transform plain-language project descriptions into structured, professional SRS documents. In addition to text-based requirements, the system will also generate system diagrams and wireframes to visually represent system architecture and user interfaces.

The software addresses the challenge faced by students, developers, and non-technical users in creating formal technical documentation. Users input their project details through a simple interface, and the AI analyzes the input to generate comprehensive SRS sections including introduction, functional requirements, non-functional requirements, system models, and use cases.

Key benefits of the AI-Powered Automated SRS Generator include:

- **Significant Time Reduction:** The system transforms the documentation creation process from days to mere minutes, allowing users to focus on other critical project tasks.
- **Improved Consistency and Completeness:** Ensures that all necessary sections are included and adhere to IEEE 830 standards, enhancing the overall quality of the documentation.
- **Lower Barrier to Entry:** Makes the system accessible for academic students and non-technical users who may struggle with traditional documentation methods.
- **Editable Output:** Allows for easy customization and refinement, enabling users to tailor the generated content to better fit their specific project needs and preferences.
- **Multiple Export Formats:** Provides flexibility in document usage, including PDF, DOCX, and Markdown, catering to various user requirements and facilitating easier sharing and collaboration.

1.3 Scope

In scope (what the system will do):

- **Requirement gathering:** Accept natural-language project descriptions from users (and optional structured prompts where provided).
- **SRS generation:** Automatically produce IEEE 830–aligned SRS documents with all major sections.
- **Diagram generation:** Automatically create system diagrams (e.g., context, use case, or flow diagrams) using PlantUML or Mermaid.js based on extracted requirements and system structure.
- **Wireframe generation:** Generate user interface wireframes using template-based HTML or Mermaid wireframe syntax to visually represent key screens or workflows.
- **Requirements categorization:** Identify and organize functional vs. non-functional requirements with clear numbering.
- **Export and storage:** Export to PDF, DOCX, and Markdown; maintain temporary session storage to prevent data loss.
- **User sessions:** Support multiple concurrent user sessions with autosave during active sessions.

- **Accessibility and usability:** Maintain a simple, responsive interface suitable for non-technical users.

Out of scope (what the system will not do):

- Replace human judgment for requirement validation or stakeholder sign-off.
- Generate source code or implement system features.
- Provide project management, tracking, or scheduling capabilities.
- Permanently store sensitive or confidential project information by default.
- Guarantee perfect accuracy or completeness without user review and edits.
- Enable real-time collaborative co-editing by multiple users.
- Perform automated requirement test generation or verification.
- Produce requirements for non-software projects.

2. Overall System Description

The AI-Powered Automated SRS Generator operates as a web-based platform accessible through modern web browsers. The system will be deployed in an environment supporting AI model execution, with sufficient computational resources to handle natural language processing tasks. Users interact with the system through a responsive web interface that guides them through the document generation process.

The deployment includes a backend server hosting the AI models and document generation logic, a database for temporary session storage, and a frontend client application. The system will be accessible via standard HTTP/HTTPS protocols and will support concurrent user sessions.

2.1 User Characteristics

The system is designed to accommodate multiple user classes with varying technical expertise:

2.1.1 Primary User Classes:

1. Non-Technical Users/Entrepreneurs

- Technical Background: Limited or no programming knowledge
- Documentation Experience: None to low; unfamiliar with SRS structure
- Frequency of Use: Occasional, typically for single projects
- Goals: Transform ideas into structured, IEEE-aligned SRS with minimal jargon and guided prompts

2. Requirements Engineers

- Technical Background: Strong understanding of requirements engineering and SDLC
- Documentation Experience: High; expert in requirements analysis and specification
- Frequency of Use: Regular, across multiple projects and iterations
- Goals: Generate high-quality drafts quickly, enforce consistency/traceability, and prepare documents for stakeholder review and baselining

2.1.2 Secondary User Classes:

3. Academic Students

- Technical Background: Basic to intermediate programming knowledge
- Documentation Experience: Limited or no prior SRS writing experience
- Frequency of Use: Periodic (primarily during project initiation phases)
- Goals: Create IEEE-compliant SRS documents for academic project submissions

4. Software Developers

- Technical Background: Strong programming and software engineering knowledge
- Documentation Experience: Moderate; understands requirements but seeks efficiency
- Frequency of Use: Regular, for multiple projects
- Goals: Accelerate documentation process while maintaining quality standards

5. Project Managers

- Technical Background: Broad understanding of software delivery and stakeholder needs

- Documentation Experience: Moderate; focuses on scope, timelines, and stakeholder alignment
- Frequency of Use: Regular during planning and review phases
- Goals: Produce review-ready drafts for planning, communication, and approvals

2.2 Operating Environment

2.2.1 Hardware Platform:

- **Client Side:** Any device capable of running modern web browsers (desktop computers, laptops, tablets)
- **Server Side:** Linux-based server with minimum 8GB RAM, multi-core processor (for AI model execution)

2.2.2 Operating Systems

- **Client:** Any operating system with modern browsers
- **Server:** Modern Linux distributions

2.2.3 Software Dependencies

- **Web Browsers:** Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 14+
- **Backend Runtime:** Python 3.12+ or Node.js 20+
- **AI/NLP Framework:** Open-source LLM libraries (e.g., Hugging Face Transformers, LangChain)
- **Database:** PostgreSQL 13+ or MongoDB 5+ for session management
- **Web Server:** Nginx or Apache for production deployment

2.2.4 Network Requirements

- Internet connectivity for accessing the web application
- HTTPS support for secure data transmission
- Minimum bandwidth: 1 Mbps for acceptable performance

2.2.5 Co-existing Software

- The system will generate documents compatible with Microsoft Word, Google Docs, and standard PDF readers

2.3 System Constraints

2.3.1 Software Constraints

- The system must use open-source AI models to avoid licensing costs and API rate limitations
- Document generation must complete within 2 minutes to maintain user engagement
- The web interface must be responsive and compatible with W3C web standards
- Maximum input text size limited to 10,000 characters to ensure reasonable processing time

2.3.2 Hardware Constraints

- Server processing capabilities limit concurrent users to approximately 50 simultaneous sessions
- AI model size constrained by available server memory (models must fit within 6GB VRAM for GPU acceleration)
- Client devices must have minimum 4GB RAM for smooth browser operation

2.3.3 Cultural Constraints

- Initial version will support English language only
- Future versions may require localization for different documentation standards (ISO, regional variations)
- Interface design must follow accessibility guidelines (WCAG 2.1 AA) for inclusive use

2.3.4 Legal Constraints

- System must comply with GDPR and data protection regulations for user input handling
- Generated documents remain the intellectual property of the user
- No permanent storage of user-generated content without explicit consent
- Open-source model usage must comply with respective licenses (Apache 2.0, MIT, etc.)

2.3.5 Environmental Constraints

- System designed for quiet office or home environments (no audio requirements)
- Visual interface assumes standard monitor resolutions (1366x768 minimum)
- No specialized hardware requirements (webcams, microphones, etc.)

2.3.6 User Constraints

- Users must be able to describe their project in written English
- Basic computer literacy required (ability to use web browsers, type, and download files)
- Users expected to have fundamental understanding of software project concepts
- System provides guidance but assumes users can validate generated requirements

2.3.7 Component Constraints

- Open-source LLM models may have inherent biases or limitations in understanding domain-specific terminology
- AI model updates require system downtime and retraining periods
- Third-party libraries for PDF/DOCX generation impose format limitations
- Browser compatibility constraints may limit use of cutting-edge web features

3. External Interface Requirements

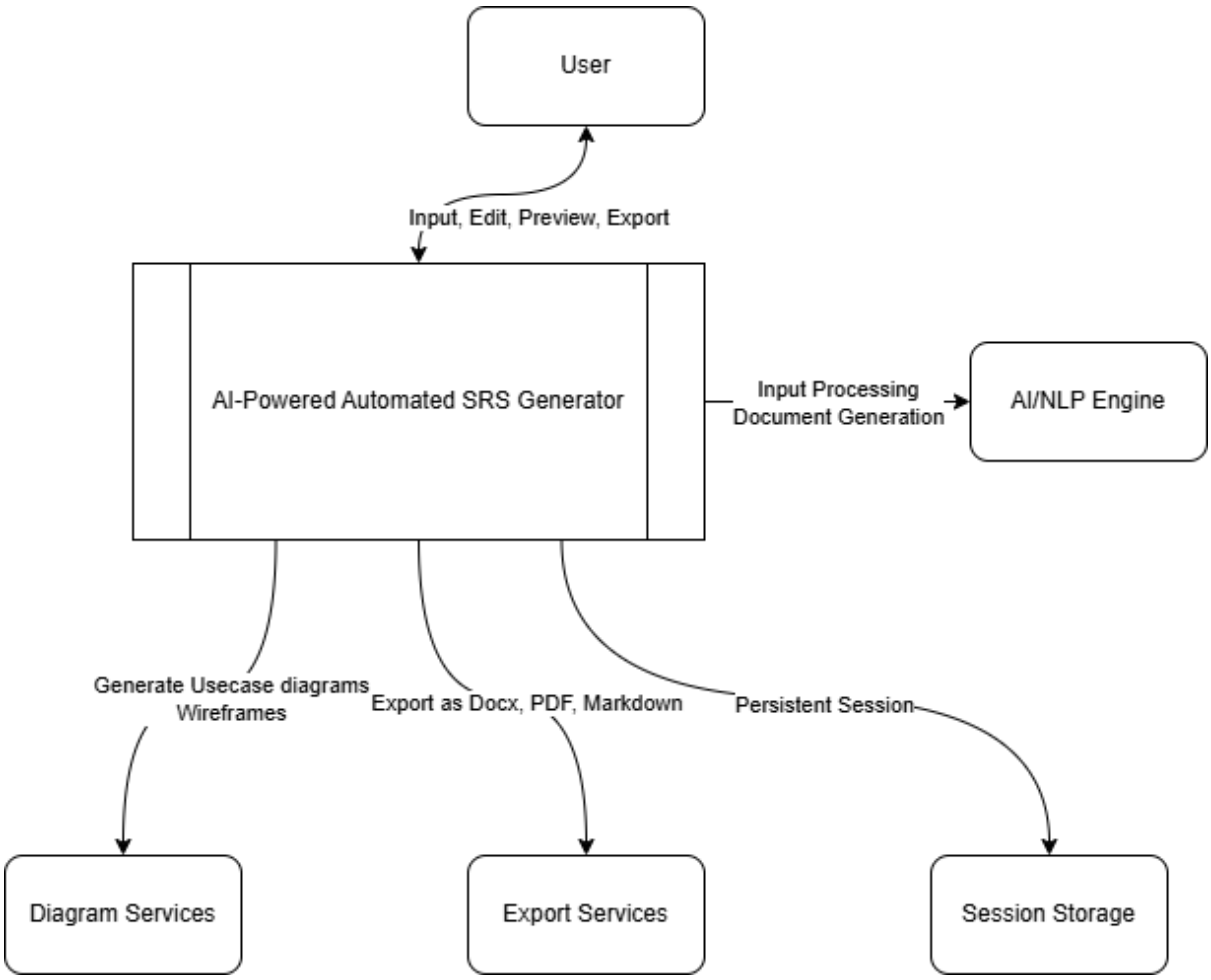

Figure 1. Use Case Diagram illustrating the core functionalities of the AI-Powered SRS Generator, including providing requirements, generating, editing, and exporting the document.

3.1 Hardware Interfaces

The AI-Powered Automated SRS Generator is a software-only system with no direct hardware interfaces. However, the system relies on standard hardware components through software abstraction layers:

3.1.1 Client-Side Hardware Interaction

- **Display Device:** The system outputs to standard computer displays through browser rendering engines.
- **Input Devices:** Accepts input from standard keyboard and mouse/touchpad devices through browser event handlers. Touch screen support included for tablet devices.
- **Storage Device:** Generated documents are saved to the user's local file system through browser download mechanisms. No direct disk access required.

3.1.2 Server-Side Hardware Interaction

- **CPU/GPU:** The backend system utilizes server processors for AI model inference. GPU acceleration (CUDA-compatible) optional but recommended for faster processing.
- **Network Interface:** Standard Ethernet or Wi-Fi network adapters for HTTP/HTTPS communication.
- **Storage:** Server-side temporary storage for session data and cached models. SSD recommended for faster model loading.

3.2 Software Interfaces

3.2.1 Web Browser (Client-Side)

Name: Modern web browsers (Chrome, Firefox, Edge, Safari)

Version: Latest stable versions (Chrome 90+, Firefox 88+, Edge 90+, Safari 14+)

Purpose: Render user interface and handle client-side interactions

Interface Type: HTTP/HTTPS RESTful API communication

Data Exchange: JSON for API requests/responses, binary for file downloads

3.2.2 Operating System

Name: Linux (Server), Windows/macOS/Linux (Client)

Version: Ubuntu 20.04+ (Server), Windows 10+, macOS 10.14+ (Client)

Purpose: Provide runtime environment and system services

Interface Type: System calls through language runtime

Data Exchange: File I/O for temporary storage, network sockets for communication

3.2.3 Runtime Environment

Name: Node.js

Version: Node.js 20+

Purpose: Execute backend application logic

Interface Type: Programming language API

Data Exchange: In-memory data structures, file system access

3.2.4 AI/NLP Framework

Name: Hugging Face Transformers, LangChain, or similar open-source library

Version: Transformers 4.30+, LangChain 0.0.200+

Purpose: Natural language processing and text generation

Interface Type: Python/JavaScript API calls

Data Exchange: Text input for analysis, structured text output for SRS generation

Shared Data: Model parameters, tokenized text, generated content

3.2.5 Database System

Name: PostgreSQL or MongoDB

Version: PostgreSQL 13+ or MongoDB 5+

Purpose: Store temporary session data, user preferences, generation history

Interface Type: Database driver/ORM (SQLAlchemy, Mongoose)

Data Exchange: SQL queries or document operations for CRUD operations

Shared Data: Session IDs, user inputs, partial generation results

3.2.6 Document Generation Libraries

Name: python-docx, ReportLab (Python) or docx, pdfkit (Node.js)

Version: python-docx 0.8.11+, ReportLab 3.6+

Purpose: Generate DOCX and PDF formatted documents

Interface Type: Programming library API

Data Exchange: Structured document content, formatting specifications

Shared Data: Generated SRS text, formatting templates

3.2.7 Diagram and Wireframe Generation Tools

Name: PlantUML, Mermaid.js, HTML template libraries

Version: PlantUML 1.2023+, Mermaid.js 10+, relevant HTML/CSS/JS libraries

Purpose: Automatically generate system diagrams (context, use case, flow) and wireframes for user interfaces

Interface Type: API calls, command-line execution, or in-process rendering (depending on deployment)

Data Exchange: Diagram/wireframe definitions (text-based), SVG/PNG/HTML outputs for embedding in SRS documents

Shared Data: Diagram source code, rendered images, wireframe markup

4. Functional Requirements

FR-1: User Input Management

The system shall provide a text input interface where users can enter their project description in natural language. The input field shall support up to 10,000 characters and provide real-time character count. The system shall accept project details including project title, overview, features, target users, and any specific requirements.

FR-2: Input Validation

The system shall validate user input to ensure minimum content requirements are met. The system shall check for minimum input length and notify users if insufficient information is provided. The system shall detect and warn about potentially unclear or vague descriptions.

FR-3: AI-Powered Analysis

The system shall analyze the user's input using natural language processing to identify key components: project objectives, functional requirements, non-functional requirements, user classes, system constraints, and external interfaces. The system shall extract and categorize information automatically without requiring structured input formats.

FR-4: SRS Document Generation

The system shall generate a complete SRS document following IEEE 830 standards with the following sections:

- Introduction (Purpose, Project Overview, Scope)
- Overall System Description (User Characteristics, Operating Environment, System Constraints)
- External Interface Requirements (Hardware Interfaces, Software Interfaces)
- Functional Requirements
- Non-functional Requirements (Performance, Safety, Security, User Documentation)
- System Diagrams (auto-generated using PlantUML or Mermaid.js)
- Wireframes (auto-generated using template-based HTML or Mermaid wireframe syntax)
- References

FR-5: Requirements Categorization

The system shall automatically distinguish between functional and non-functional requirements from the user's input. The system shall organize functional requirements as numbered items (FR-1, FR-2, etc.) and non-functional requirements under appropriate categories (performance, security, usability, etc.).

FR-5a: Diagram Generation

The system shall automatically generate system diagrams (such as context diagrams, use case diagrams, or flow diagrams) based on the extracted requirements and system structure. Diagrams shall be created using PlantUML or Mermaid.js and included in the generated SRS document.

FR-5b: Wireframe Generation

The system shall generate user interface wireframes to visually represent key screens or workflows. Wireframes shall be created using template-based HTML or Mermaid wireframe syntax and included in the SRS document to help users visualize the proposed user experience.

FR-6: Document Preview

The system shall display the generated SRS document in a preview pane immediately after generation. The preview shall show formatted text with proper headings, numbering, and structure. Users shall be able to scroll through all sections before deciding to edit or export.

FR-7: Content Editing

The system shall allow users to edit any section of the generated SRS document directly within the interface. Changes shall be reflected in real-time. The system shall maintain document structure and formatting during editing. Users shall be able to add, modify, or delete content as needed.

FR-8: Document Export - PDF Format

The system shall provide an export function to save the SRS document as a PDF file. The PDF shall maintain professional formatting with proper page breaks, headers, and table of contents. The exported file shall be downloadable to the user's local system.

FR-9: Document Export - DOCX Format

The system shall provide an export function to save the SRS document as a Microsoft Word (.docx) file. The DOCX file shall be editable in Microsoft Word or compatible applications. Formatting shall include styles for headings, body text, and numbered lists.

FR-10: Document Export - Markdown Format

The system shall provide an export function to save the SRS document as a Markdown (.md) file. The Markdown format shall use standard syntax for headings, lists, and emphasis. This format shall be suitable for version control and plain-text editing.

FR-11: Session Management

The system shall maintain user sessions during the document creation process. Users shall be able to work on a document, leave, and return within a reasonable timeframe (e.g., 24 hours) without losing progress. Session data shall be stored securely and deleted after expiration.

FR-12: Multi-Language Support (Future Enhancement)

For future versions, the system may support multiple input languages and generate SRS documents in the corresponding language. Initial version will support English only.

FR-13: Error Handling and User Feedback

The system shall provide clear error messages when generation fails or encounters issues. Users shall receive guidance on how to resolve problems (e.g., providing more detail, clarifying ambiguous input). The system shall log errors for debugging and improvement purposes.

FR-14: Requirement Numbering and Organization

The system shall automatically assign unique identifiers to all functional and non-functional requirements. Numbering shall follow a consistent pattern (FR-1, FR-2, NFR-1, NFR-2, etc.). The system shall organize requirements logically within each section.

FR-15: Progress Indication

The system shall display progress indicators during the document generation process. Users shall see status updates such as "Analyzing input...", "Generating requirements...", "Formatting document..." to understand the system is processing their request.

5. Non-functional Requirements

5.1 Performance Requirements

NFR-P1: Response Time

The system shall generate a complete SRS document within 120 seconds (2 minutes) of user request submission for typical project descriptions (1,000-5,000 characters). Preview rendering shall occur within 2 seconds of generation completion.

NFR-P2: Concurrent User Capacity

The system shall support at least 50 concurrent users generating documents simultaneously without significant performance degradation. Response times shall not exceed 150 seconds even at peak capacity.

NFR-P3: Document Export Speed

Document export operations (PDF, DOCX, Markdown) shall complete within 10 seconds for documents up to 50 pages in length. Download initiation shall begin within 2 seconds of user clicking the export button.

NFR-P4: System Availability

The system shall maintain 95% uptime during operational hours. Planned maintenance windows shall be scheduled during low-usage periods and communicated to users in advance.

NFR-P5: Scalability

The system architecture shall be designed to scale horizontally by adding additional server instances. The system shall handle a 100% increase in user load with proportional hardware scaling.

NFR-P6: AI Model Inference Speed

The AI/NLP model shall process user input at a minimum rate of 50 tokens per second. Model loading time shall not exceed 30 seconds during system startup.

NFR-P7: Database Query Performance

Database operations for session retrieval and storage shall complete within 500 milliseconds. Database queries shall be optimized with appropriate indexing.

NFR-P8: Accuracy and Precision

The system shall generate SRS documents with at least 80% relevance and accuracy compared to manually written documents for typical software projects. This shall be validated through user feedback and expert review.

5.2 Safety Requirements

NFR-S1: Data Loss Prevention

The system shall implement auto-save functionality, storing user input and partial generations every 30 seconds to prevent data loss in case of connection interruption or browser crashes.

NFR-S2: Input Sanitization

All user inputs shall be sanitized to prevent injection attacks or malformed data from causing system crashes. Invalid characters shall be filtered or escaped appropriately.

NFR-S3: Graceful Degradation

If the AI model fails or becomes unavailable, the system shall display a clear error message and allow users to save their input for later processing. The system shall not crash or lose user data.

NFR-S4: Resource Limits

The system shall enforce resource limits to prevent individual users from consuming excessive server resources. Long-running generation processes shall be terminated after 5 minutes with appropriate user notification.

NFR-S5: Backup and Recovery

The system shall maintain daily backups of the database containing session information. Recovery procedures shall restore the system to operational status within 4 hours of a critical failure.

NFR-S6: Error Logging

The system shall log all errors and exceptions to a centralized logging system for monitoring and debugging. Critical errors shall trigger immediate alerts to system administrators.

5.3 Security Requirements

NFR-SE1: Data Transmission Security

All communication between the client and server shall be encrypted using HTTPS with TLS 1.2 or higher. SSL certificates shall be valid and properly configured.

NFR-SE2: Input Privacy

User project descriptions and generated documents shall not be permanently stored on the server without explicit user consent. Session data shall be automatically deleted after 24 hours of inactivity.

NFR-SE3: Authentication

If user accounts are implemented, the system shall use secure authentication mechanisms with password hashing (bcrypt or similar) and protection against brute-force attacks.

NFR-SE4: Access Control

System administrative functions shall be protected by role-based access control. Only authorized personnel shall have access to server logs, database, and configuration files.

NFR-SE5: Data Protection Compliance

The system shall comply with GDPR requirements for user data handling. Users shall have the right to request deletion of any stored data. Privacy policy shall clearly state data usage and retention policies.

NFR-SE6: SQL Injection and XSS Prevention

The system shall implement parameterized queries and input validation to prevent SQL injection attacks. Output shall be properly escaped to prevent cross-site scripting (XSS) vulnerabilities.

NFR-SE7: Rate Limiting

The system shall implement rate limiting to prevent abuse and denial-of-service attacks. Individual IP addresses shall be limited to 10 document generation requests per hour.

NFR-SE8: Secure File Handling

Exported documents shall be generated in isolated temporary directories with appropriate permissions. Temporary files shall be deleted immediately after successful download or after 1 hour, whichever comes first.

5.4 User Documentation

5.4.1 NFR-UD1: User Manual

A comprehensive user manual shall be provided in PDF format covering:

- Getting started guide
- Step-by-step instructions for creating an SRS document
- Tips for writing effective project descriptions
- Understanding the generated sections
- Editing and customizing the output
- Export options and format differences
- Troubleshooting common issues

5.4.2 NFR-UD2: Online Help System

The web application shall include an integrated help system accessible via a "Help" button. Context-sensitive help shall provide guidance relevant to the user's current task.

5.4.3 NFR-UD3: Tutorial Videos

Video tutorials demonstrating the document generation process shall be provided. Videos shall cover basic usage (5-10 minutes) and advanced features (10-15 minutes).

5.4.4 NFR-UD4: FAQ Section

A Frequently Asked Questions section shall address common user queries including:

- Input format recommendations
- How to improve generation quality
- Differences between export formats
- Data privacy and security concerns
- System requirements and browser compatibility

5.4.5 NFR-UD5: Example Projects

The system shall provide 3-5 sample project descriptions with corresponding generated SRS documents. Examples shall cover different project types (web applications, mobile apps, embedded systems, etc.).

5.4.6 NFR-UD6: In-App Tooltips

User interface elements shall include tooltips explaining their purpose. First-time users shall be offered an optional guided tour of the interface.

5.4.7 NFR-UD7: Documentation Accessibility

All documentation shall be accessible to users with disabilities, following WCAG 2.1 AA guidelines. This includes proper heading structure, alt text for images, and screen reader compatibility.

6. References

830-1998 - IEEE Recommended Practice for Software Requirements Specifications. (1998, October 20). *IEEE Std 830-1998*. doi:10.1109/IEEESTD.1998.88286

Lalwani, T., Bhalotia, S., Pal, A., Rathod, V., & Bisen, S. (2018). Implementation of a Chatbot System using AI and NLP. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3531782

Jain, S. M. (2022). Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems* (pp. 51-67). Berkeley, CA: Apress.

Mavroudis, V. (2024). LangChain.

Cahn, J. (2017). CHATBOT: Architecture, design, & development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*.

Kukreja, S., Kumar, T., Purohit, A., Dasgupta, A., & Guha, D. (2024). A Literature Survey on Open Source Large Language Models. *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, 133–143. https://doi.org/10.1145/3647782.3647803

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, *2*(2), 115-150.

Kurotych, A. O., & Bulatetska, L. V. (2024). Optimizing the process of ER diagram creation with PlantUML. In *CS&SE@ SW* (pp. 47-57).

7. Appendix

7.1 Use Cases

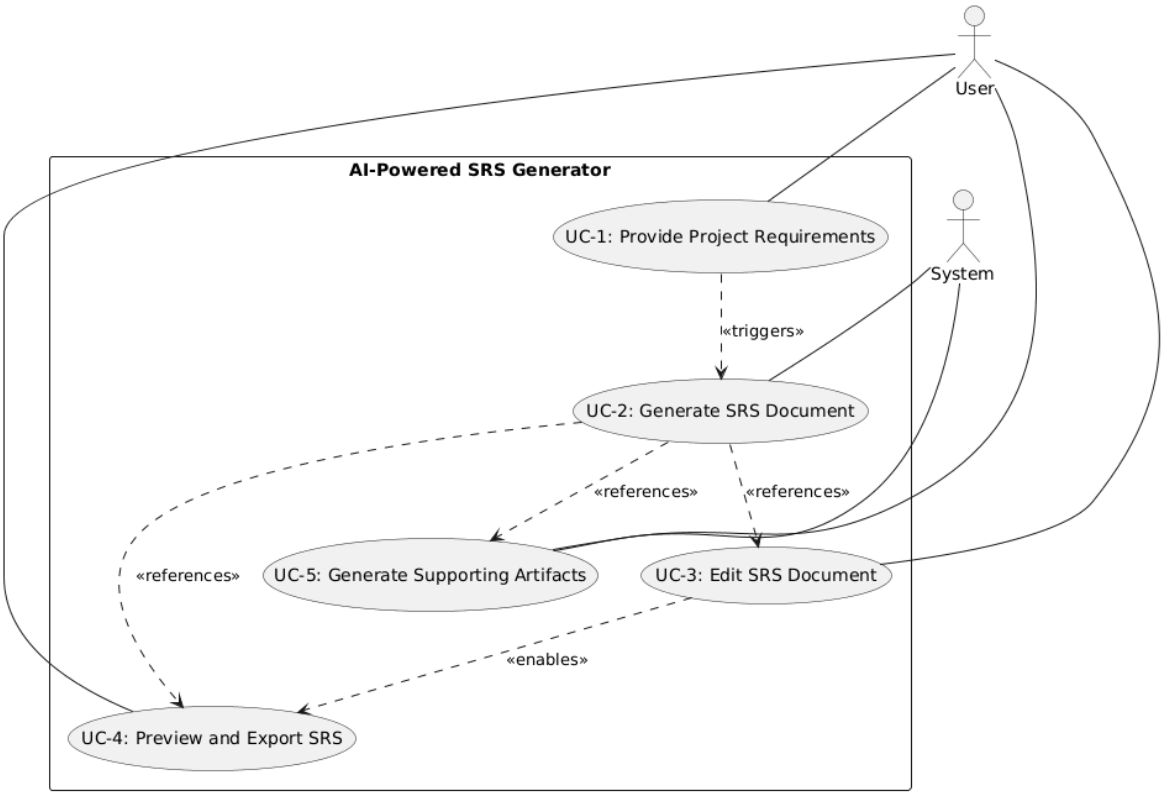

Figure 2. System Architecture Diagram showing the key components of the application, such as the AI/NLP Engine, and the services for generating diagrams, exporting files, and managing user sessions.

Use Case ID	Use Case Name	Primary Action	Brief Description
UC-1	Provide Project Requirements	User	User provides the project idea; system asks clarifying questions until requirements are complete.
UC-2	Generate SRS Document	System	System analyzes requirements and produces a structured SRS.
UC-3	Edit Generated SRS	User	User refines the generated SRS content.
UC-4	Preview and Export SRS	User	User reviews the formatted SRS and exports it as PDF/DOCX/MD.
UC-5	Generate Supporting Artifacts	System	System produces diagrams and wireframes based on the SRS.

UC-1 Provide Project Requirements		
Actors		User, System
Feature		Requirement Collection and Clarification
Use Case ID		UC-1
Preconditions		• User is logged into the system. • System interface is active and ready to receive input.
Success Scenario		
Step#	**Action**	**Software Reaction**
1	User opens the "New Project" form.	System displays fields for project name and description.
2	User enters the initial project idea or description in natural language.	System analyzes the text for missing or ambiguous details.
3		System generates clarifying questions (if needed).
4	User answers the system's clarifying questions.	System refines its understanding of the project requirements.
5	User confirms all responses.	System saves the finalized requirements and confirms completion.
Alternate Scenarios		
• **2a:** User submits an empty or too short description → system requests more detail. • **4a:** User skips a clarifying question → system marks that area as incomplete and continues.		
Postconditions		
• All project requirements are recorded in the system. • Data is ready for SRS generation.		
Use Cases Referenced:		UC-2 – Generate SRS Document

UC-2 Generate SRS Document		
Actors		System
Feature		AI-Powered Document Generation
Use Case ID		UC-2
Preconditions		• Requirements are provided and saved (UC-1 complete). • System has sufficient computational resources.
Success Scenario		
Step#	**Action**	**Software Reaction**
1	User selects “Generate SRS.”	System retrieves stored requirements.
2		System processes input using NLP and AI requirement classification models.
3		System maps extracted information to IEEE SRS structure.
4		System generates the full draft SRS.
5		System saves the generated document and notifies the user.
Alternate Scenarios		
• 2a: Requirements incomplete → system prompts user to return to UC-1. • 4a: Model timeout or generation error → system retries or displays error message.		
Postconditions		
• Draft SRS generated successfully. • Document stored for review and editing.		
Use Cases Referenced:		UC-1 – Provide Project Requirements, UC-3 – Edit SRS Document

UC-3 Edit SRS Document		
Actors		User
Feature		Document Refinement
Use Case ID		UC-3
Preconditions		• A generated SRS exists in the user's workspace. • User has permission to modify it.
Success Scenario		
Step#	**Action**	**Software Reaction**
1	User opens the generated SRS.	System loads document sections in editable form.
2	User selects a section to modify.	System enables editing for that section.
3	User makes edits	System highlights unsaved changes.
4	User saves edits.	System stores updates and confirms success.
Alternate Scenarios		
• 2a: User attempts to edit system-protected metadata → system restricts access. • 4a: Network error during save → system retries or shows error message.		
Postconditions		
• Updated SRS is saved. • Document ready for preview or export.		
Use Cases Referenced:		UC-2 – Generate SRS Document, UC-4 – Preview and Export SRS

UC-4 Preview and Export SRS		
Actors		User
Feature		Document Output and Formatting
Use Case ID		UC-4
Preconditions		• An edited or generated SRS is available..
Success Scenario		
Step#	**Action**	**Software Reaction**
1	User opens preview mode.	System loads document sections in editable form.
2	User scrolls and reviews sections.	System enables editing for that section.
3	User selects export option (PDF/DOCX/Markdown).	System generates and downloads file in chosen format.
4	User saves edits.	System stores updates and confirms success.
Alternate Scenarios		
• 3a: File conversion fails → system notifies user and suggests retry. • 3b: User cancels download → system aborts export safely.		
Postconditions		
• Formatted SRS exported or previewed successfully. • Export log stored in system history.		
Use Cases Referenced:		UC-3 – Edit SRS Document

UC-5 Generate Supporting Artifacts		
Actors		System
Feature		Visual Asset Generation (Diagrams and Wireframes)
Use Case ID		UC-5
Preconditions		• A valid SRS document exists. • Diagram and wireframe modules are enabled.
Success Scenario		
Step#	**Action**	**Software Reaction**
1	User selects “Generate Artifacts.”	System identifies key entities, processes, and relationships from SRS.
2		System auto-generates context, use case, and flow diagrams.
3		System generates basic UI wireframes (if applicable)
4		System displays generated artifacts to user for download or preview.
Alternate Scenarios		
• 2a: User attempts to edit system-protected metadata → system restricts access. • 4a: Network error during save → system retries or shows error message.		
Postconditions		
• Updated SRS is saved. • Document ready for preview or export.		
Use Cases Referenced:		UC-2 – Generate SRS Document, UC-4 – Preview and Export SRS

7.2 Flow Diagram

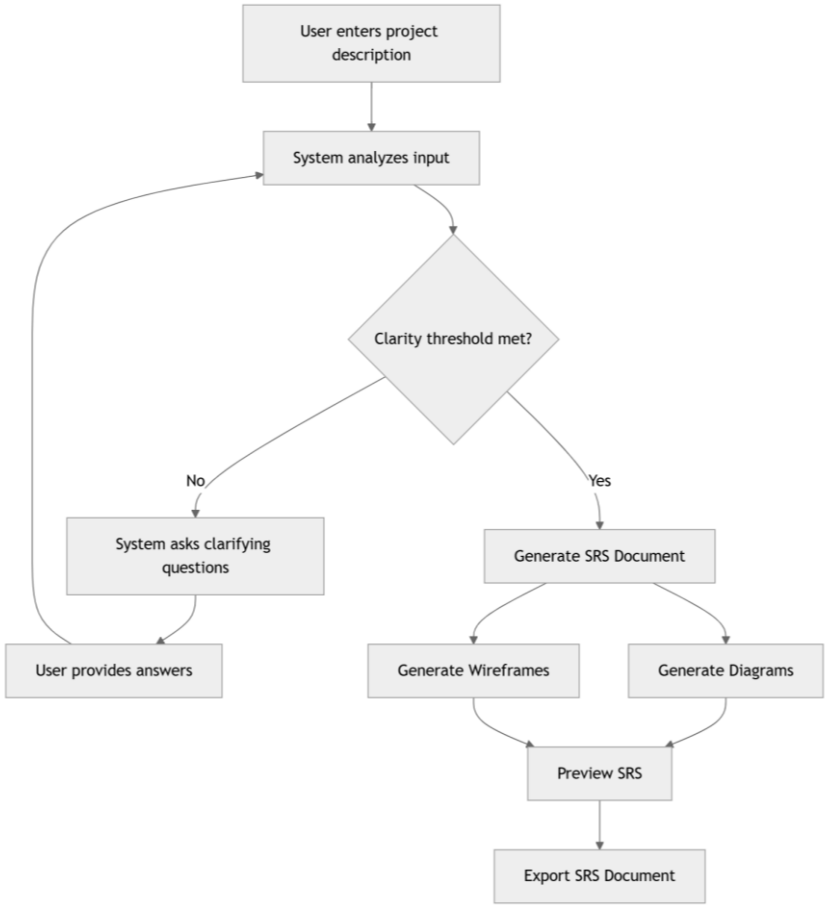

Figure 3. Process Flowchart detailing the user journey from entering a project description to the final export of the SRS document, including the feedback loop for clarifying requirements.