**A REPORT**

**ON**

**"AI-Driven Multi-Agent System for Requirement Elicitation, Analysis, Review, and Test Case Generation in Systems Engineering"**

**AEL ZG628T:** Dissertation

by

**Ranjit Rajendra Jagtap (2023HT65600)**

**Dissertation Work carried out at:** Onward Technologies Limited, Pune

**Submitted in THE fulfilment of** M. Tech (Automotive Electronics)

degree programme

**Degree Program:** M. Tech (Automotive Electronics)

**Under the Supervision of** Mr. Chaitanya Kulkarni

Onward Technologies Limited, Pune

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

****

**(November 2025)**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE**

**PILANI (RAJASTHAN)**

**WILP Division**

**(November 2025)**



**Organization:** Onward Technologies Limited **Location:** Pune, MH, India

**Semester Start:** July 26, 2025 **Semester End:** December 6, 2025

**Duration:** 4 months and 10 days  **Date of Submission:** November 06, 2025

**Title of the Project**: "AI-Driven Multi-Agent System for Requirement Elicitation, Analysis, Review, and Test Case Generation in Systems Engineering"

**ID No./Name of the student:** 2023HT65600 | Ranjit Rajendra Jagtap

**Name (s) and Designation (s) of your Supervisor and Additional Examiner:**

|  |  |  |
| --- | --- | --- |
|  | **Supervisor** | **Additional Examiner** |
| Name | Mr. Chaitanya Kulkarni | Mr. Deepak Garg |
| Qualification | Bachelor of Engineering | Master of Science (AI/ML) |
| Designation | DGM-Delivery | GM-Delivery |

**Name of the Faculty mentor:** Mr. Sajeeth Kumar (Lead Instructor)

**Key Words:**

Artificial Intelligence | Multi-Agent Systems | Requirement Engineering | Requirement Elicitation | Requirement Analysis | Automated Test Case Generation | Natural Language Processing (NLP) | Systems Engineering | LLM-based Automation | AI-Driven Review Systems | Software / Systems Engineering | Intelligent Automation | Test Automation & Verification | Human-AI Collaboration | NLP in Engineering | Software Quality Assurance

**Project Areas:**

Artificial Intelligence | Software / Systems Engineering | Requirement Engineering & Management | Intelligent Automation | Test Automation & Verification | Natural Language Processing (NLP) in Engineering | Human-AI Collaboration in Engineering Workflows

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to **Mr. Sajeeth Kumar (Lead Instructor, BITS Pilani)**, who evaluated this dissertation on behalf of the institute. His academic guidance, feedback, and constructive observations were invaluable in aligning the work with the standards and expectations of the M. Tech (Automotive Electronics) programme.

I am deeply thankful to my project supervisor, **Mr. Chaitanya Kulkarni (DGM – Delivery, Onward Technologies Ltd., Pune)**, for his continuous mentorship, technical guidance, and encouragement throughout the course of this project. His insights and expertise played a defining role in the successful execution of this dissertation.

My sincere appreciation also goes to **Mr. Deepak Garg (GM – Delivery, Onward Technologies Ltd., Pune)**, who served as the Additional Examiner. His thoughtful reviews, suggestions, and support helped enhance the quality and clarity of the work.

I extend my gratitude to **Onward Technologies Limited, Pune**, for providing the opportunity, industrial exposure, and resources necessary to carry out this research.

I would also like to acknowledge the faculty and staff of the **M. Tech (Automotive Electronics)** programme at **Birla Institute of Technology & Science, Pilani**, for building the academic foundation and research mindset that made this work possible.

Lastly, I offer heartfelt thanks to my family and friends for their constant encouragement, patience, and belief in me throughout this journey.

**ABSTRACT**

Managing requirements in large-scale system development remains a significant challenge due to the reliance on manual methods that are often time-consuming, inconsistent, and error-prone. These limitations affect traceability, compliance, and the generation of test cases across the development lifecycle. This dissertation introduces a modular multi-agent framework that leverages Large Language Models (LLMs), Natural Language Processing (NLP), and Retrieval-Augmented Generation (RAG) to streamline and automate critical tasks within requirements engineering.

The framework introduces five specialized agents covering requirement elicitation (SYS.1), drafting (SYS.2), standards-based review, test-case generation (SYS.5), and overall orchestration through a manager agent. Built on **CrewAI** for coordination, the system supports multi-source input such as emails, Jira tickets, and technical documents, which are semantically parsed into structured requirements. Compliance checks are embedded against international standards including ISO/IEC/IEEE 29148:2018, IREB, and ISO 26262, ensuring clarity, consistency, and safety alignment. The approach also enforces end-to-end traceability, duplication detection, and requirement-to-test-case mapping, all accessible through a collaborative web interface suitable for diverse stakeholders.

System evaluation is guided by quantitative metrics such as requirement extraction accuracy, compliance scores, test coverage, and orchestration efficiency. Beyond automating routine tasks, the solution demonstrates the potential of **LLM-powered requirement engineering** to shorten development cycles, enhance requirement quality, and support safety-critical industries like automotive systems. With its extensible design, the framework can evolve toward Automotive SPICE practices (SWE.1–SWE.6), offering both academic significance and practical applicability in enterprise environments.

The research contributes a **novel end-to-end automation pipeline** for requirement engineering that combines academic insights with real-world applicability. By bridging gaps between unstructured stakeholder communication and formal engineering artifacts, the system enables greater collaboration, reduces manual rework, and ensures compliance with international standards. Future extensions may include integration with enterprise tools such as Jira and DOORS, multilingual support, and automatic generation of UML/flow diagrams, strengthening its role as a practical foundation for next-generation intelligent requirement engineering platforms.

|  |  |
| --- | --- |
|  |  |
| **Signature of Student** | **Signature of Supervisor** |
| **Name: Ranjit Rajendra Jagtap** | **Name: Mr. Chaitanya Kulkarni** |

Table of Contents

[1. Introduction 7](#_Toc213344409)

[2. Background 7](#_Toc213344410)

[3. Objectives 9](#_Toc213344411)

[4. Scope of Work 10](#_Toc213344412)

[5. Results 22](#_Toc213344413)

[5.1 Editor/IDE – Visual Studio Code 22](#_Toc213344414)

[5.2 Landing Page 23](#_Toc213344415)

[5.3 Agent 1: SYS.1 Elicitation 23](#_Toc213344416)

[5.4 Agent 1: SYS.1 Elicitation – Result 24](#_Toc213344417)

[5.5 Agent 2: SYS.2 Analysis 24](#_Toc213344418)

[5.6 Agent 2: SYS.2 Analysis - Result 25](#_Toc213344419)

[5.7 Agent 3: SYS.2 Review 25](#_Toc213344420)

[5.8 Agent 3: SYS.2 Review – Result 26](#_Toc213344421)

[5.9 Agent 4: SYS.5 Testcase Generation 26](#_Toc213344422)

[5.10 Agent 4: SYS.5 Testcase Generation – Result 27](#_Toc213344423)

[5.11 Agent 5: Manager 27](#_Toc213344424)

[5.12 Agent 5: Manager – Result 28](#_Toc213344425)

[6. Design Considerations 40](#_Toc213344426)

[7. Plan of Work 42](#_Toc213344427)

[8. Conclusion 43](#_Toc213344428)

[9. Recommendations 44](#_Toc213344429)

[10. Abbreviations 45](#_Toc213344430)

[11. Literature References 46](#_Toc213344431)

[12. Checklist of Items for the Final Dissertation / Project / Project Work Report 47](#_Toc213344432)

List of Figures

[Figure 1: Objective of achieving automated RE 9](#_Toc213344456)

[Figure 2: Agent 1-Requirement Intake & SYS.1 Elicitation 10](#_Toc213344457)

[Figure 3: Agent 2-SYS.2 Drafting 10](#_Toc213344458)

[Figure 4: Agent 3-SYS.2 Review & Compliance 11](#_Toc213344459)

[Figure 5: Agent 4-SYS.5 Test Case Generator 11](#_Toc213344460)

[Figure 6: Agent 5-Manager Agent (CrewAI) 12](#_Toc213344461)

[Figure 7: AI-Driven Multi-Agent System Pipeline 14](#_Toc213344462)

[Figure 8: Multi-agent Framework 19](#_Toc213344463)

[Figure 9: Langchain Framework 20](#_Toc213344464)

[Figure 10: Future Expansion Roadmap 31](#_Toc213344465)

[Figure 11: Data Privacy & Enterprise Security Considerations 32](#_Toc213344466)

List of Tables

[Table 1: LLM Agents vs Traditional Methods 7](#_Toc213344467)

[Table 2: Agents and Their Technology Stack 12](#_Toc213344468)

[Table 3: Technologies in depth & use case 13](#_Toc213344469)

[Table 4: Validation Strategy & Performance Metrics 14](#_Toc213344470)

[Table 5: Manager Agent Role 21](#_Toc213344471)

[Table 6: Risks & Mitigation Strategies 32](#_Toc213344472)

# Introduction

This research is positioned at the convergence of Requirement Engineering, Artificial Intelligence, and Natural Language Processing (NLP). The system aims to automate and enhance the traditional software/system requirement engineering life cycle using intelligent agents powered by Large Language Models (Large Language Models (LLMs)). It emphasizes end-to-end automation from requirement intake (Sys.1) to system test case generation (SYS.5), with a modular multi-agent design. This supports semantic parsing, compliance validation, traceability, and test generation delivered through a highly configurable and interactive UI for technical and non-technical stakeholders alike.

# Background

In my current role as a Requirement Engineer, I am deeply involved in reviewing, drafting, and validating requirements for complex systems based on various stakeholder inputs. It involves interpreting unstructured sources-such as emails, Jira tickets, PDFs, and verbal exchanges-and transforming them into accurate, traceable requirements. These responsibilities directly relate to the challenges discussed in the project background such as manual errors, traceability gaps, and inefficiencies in the requirement engineering lifecycle.

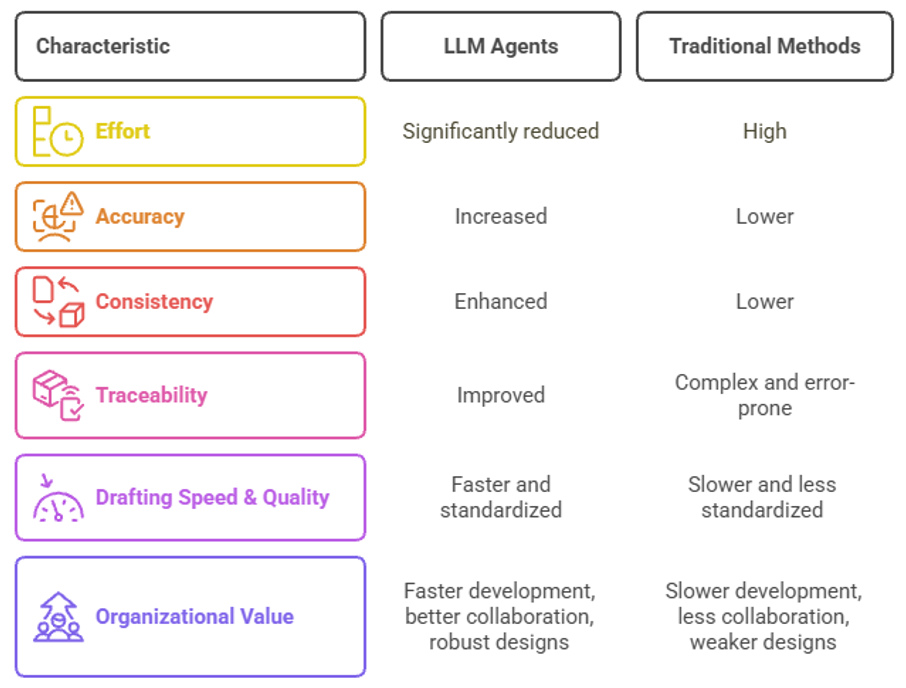


Table 1: LLM Agents vs Traditional Methods

The proposed Multi-Agent Requirement Engineering System aligns well with my workplace needs in the following ways:

**Automating Manual Efforts:**

Tasks like requirement extraction, conflict detection, and dependency analysis are currently performed manually. Intelligent agents powered by Large Language Models (LLMs) can significantly reduce effort, increase accuracy, and enhance consistency across documentation.

**Enhancing Traceability:**

Maintaining traceability from requirements to design and testing artifacts is complex and error-prone. AI-based methods can help in automatically mapping dependencies, thereby supporting impact analysis and smoother change management.

**Improving Drafting Speed and Quality:**

Generating structured requirements from informal or scattered sources is a routine challenge. Leveraging generative Artificial Intelligence would enable faster and more standardized documentation, improving overall project efficiency.

From an organizational perspective, the system promises faster development cycles, improved collaboration, and more reliable system designs-particularly valuable in compliance-driven industries.

Additionally, during my second semester, I studied Systems Engineering, which gave me a solid foundation in systems thinking, requirements lifecycle, and verification & validation processes. This academic grounding supports the practical application of this project, particularly in structuring system-level requirements and modeling agent workflows aligned with SYS.1 to SYS.5 activities.

This project thus serves as a perfect convergence of academic learning, industry needs, and practical experience, with strong potential for real-world adoption in my current organizational context.

# Objectives

The key objectives of this project are:

1. To design and implement four intelligent agents (Agent 1–4) together with a Manager Agent that collectively **automate and execute** the full requirements engineering lifecycle & testing at the system level.
2. To provide multi-format file support (.pdf, .docx, .xlsx, .csv, emails, etc.) for seamless requirement intake.
3. To enable compliance checks against ISO/IEC/IEEE 29148:2018, IREB, and ISO 26262 standards.
4. To support automated generation of SYS.1, SYS.2, and SYS.5 artifacts in structured formats.
5. To develop a collaborative, web-based UI for uploading files, reviewing parsed data, selecting agents, and visualizing dashboards.
6. To build an “All Agents” mode that runs the complete pipeline from intake to test case generation with one user input.

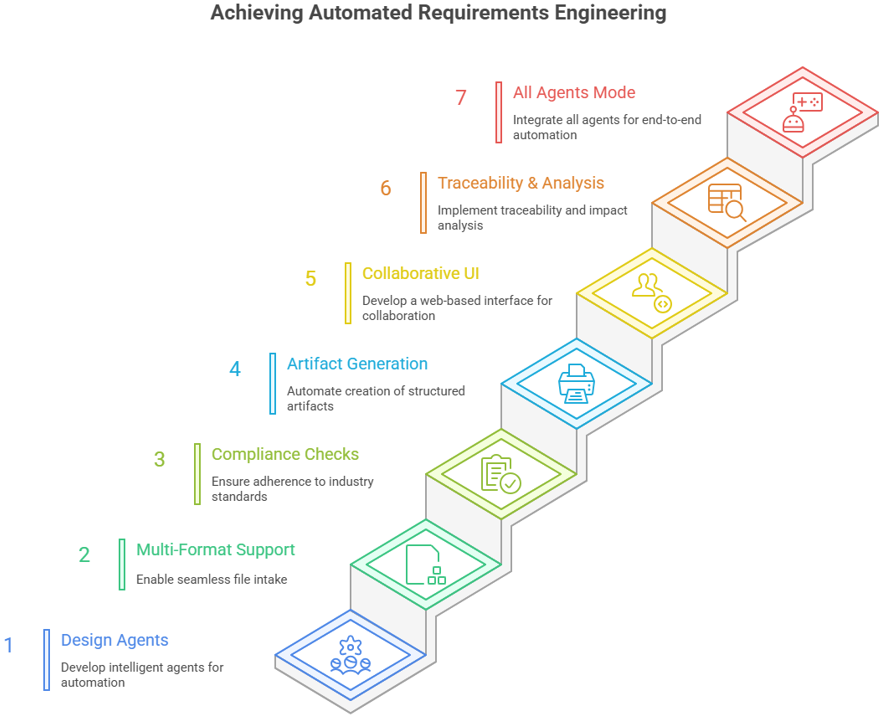


Figure 1: Objective of achieving automated RE

# Scope of Work

The scope includes the development of the following modules and capabilities, these agents collaboratively automate the end-to-end requirements engineering lifecycle at the system level through an offline agent chain, operating independently of existing requirement management tools.

a. **Agent 1: Requirement Intake & SYS.1 Elicitation**

* Multi-source ingestion: emails, voice-to-text, Jira, PDFs, DOCX, Reqif etc., NLP-based parsing of raw content to identify intents, constraints, and stakeholders,
* Classification of requirements (functional, non-functional, regulatory), Generation of SYS.1 format with traceability and Pass to Agent 2

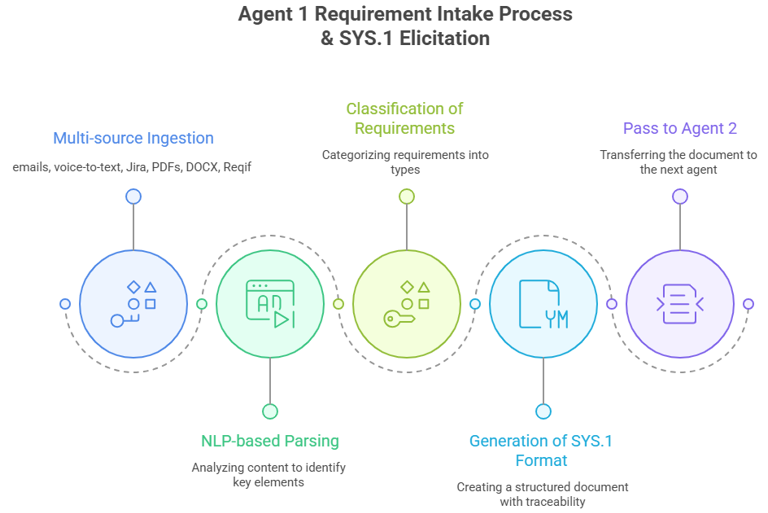


Figure 2: Agent 1-Requirement Intake & SYS.1 Elicitation

**b. Agent 2: SYS.2 Drafting**

* Draft structured SYS.2, Auto-fill metadata like ID, rationale, verification method and criteria.
* Enable inline edits and template updates with feedback from Agent 3 and Pass to Agent 3

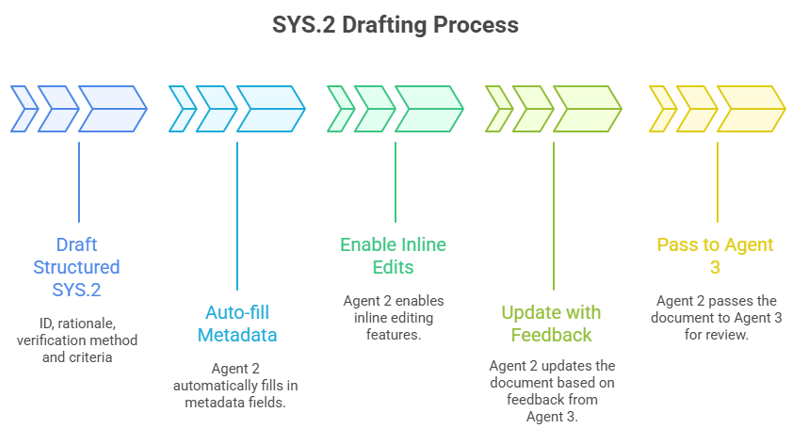


Figure 3: Agent 2-SYS.2 Drafting

c. **Agent 3: SYS.2 Review & Compliance**

* Validate requirements against IREB and ISO standards (e.g., 26262-8\_6.4.2) for clarity, consistency, ambiguity completeness, etc.
* Comment/suggest mode with interactive acceptance/rejection.
* Auto-learn from past accepted edits to improve model behaviour.
* Pass to Agent 4

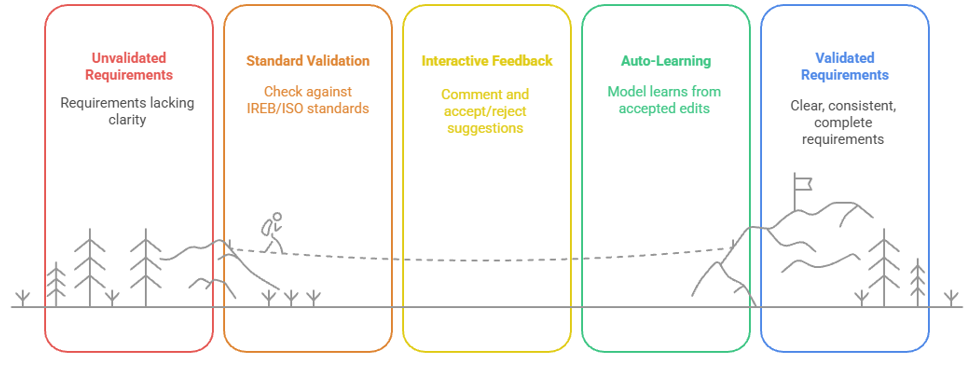


Figure 4: Agent 3-SYS.2 Review & Compliance

d. **Agent 4: SYS.5 Test Case Generator**

* Generate test cases from reviewed SYS.2 with all mandatory attributes like preconditions, steps, expected result.
* Maintain traceability (requirement ID → test case ID).
* Support editing, delta correction, and feedback-driven improvements.

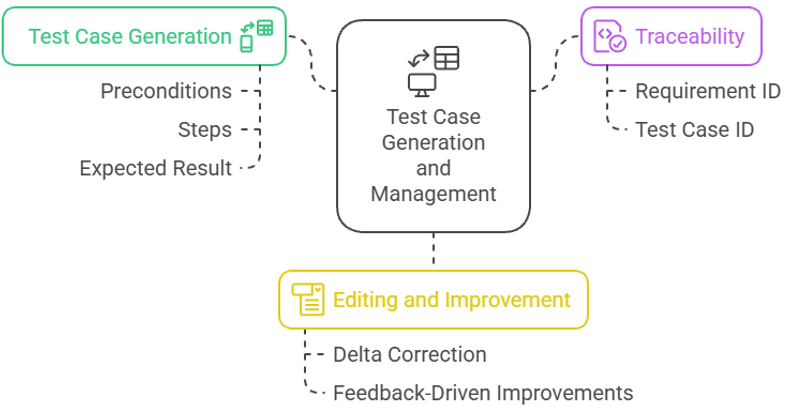


Figure 5: Agent 4-SYS.5 Test Case Generator

**e. Agent 5: Manager Agent**

* Orchestrates Agents 1–4
* Applying business logic and decision-making rules.
* Approving or rejecting intermediate and final outputs.
* Managing execution order and dependencies.
* Conflict Resolution
* All-Agents' Mode
* Combines capabilities of all agents in sequence.
* Output all deliverables in downloadable formats with traceability maintained.



Figure 6: Agent 5-Manager Agent (CrewAI)

**Agents and Their Technology Stack**

|  |  |
| --- | --- |
| **Agent** | **Models / Technologies Used** |
| **Agent 1 (SYS.1)** | GPT-4 (via OpenAI API) for NLP processing, CrewAI for orchestration, LangChain for workflow integration, Streamlit for user interface. |
| **Agent 2 (SYS.2)** | Same stack: GPT-4 (via OpenAI API) for drafting, CrewAI for coordination, LangChain for structuring workflows, Streamlit for UI interaction. |
| **Agent 3 (SYS.2 Review)** | GPT-4 (via OpenAI API) for compliance checks, CrewAI for agent orchestration, LangChain for linking external standards/knowledge, Streamlit for review UI. |
| **Agent 4 (SYS.5)** | GPT-4 (via OpenAI API) for test-case generation, CrewAI for orchestration, LangChain for maintaining traceability, Streamlit for test-case visualization. |
| **Agent 5 (Manager)** | CrewAI as the primary orchestrator, coordinating all agents; GPT-4 (via OpenAI API) for reasoning, LangChain for RAG-based knowledge integration, Streamlit for central dashboard. |

Table 2: Agents and Their Technology Stack

**Note:** The table above outlines the models designated for each agent based on their use cases. During implementation, real outputs will be evaluated to select the most cost-effective model as needed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Technology** | **What It Is** | **Primary Use Case** | **Relevance in Dissertation** |
| **GPT-4** | A state-of-the-art large language model developed by OpenAI, known for advanced reasoning, comprehension, and generation capabilities across diverse domains. | Natural Language Processing (NLP) tasks such as requirement interpretation, summarization, specification drafting, and intelligent text generation. | Serves as the core LLM powering intelligent agents for automated requirement analysis, synthesis, and validation support. |
| **OpenAI API** | A cloud-based API service by OpenAI that provides scalable access to GPT-4 and related AI models with reliable performance and flexible integration options. | Running GPT-4 models efficiently with configurable parameters for context length, temperature, and response optimization. | Enables seamless deployment and real-time interaction with GPT-4 outputs in multi-agent and workflow-driven research environments. |
| **CrewAI** | An orchestration framework for creating and managing multiple AI agents with defined roles, goals, and interactions. | Multi-agent collaboration, task delegation, workflow management. | Acts as the “Manager Agent” to coordinate other agents (SYS.1–SYS.5), ensuring sequencing, conflict resolution, and quality control. |
| **LangChain** | A framework for building applications powered by large language models, with strong integration to external data and tools. | Retrieval-Augmented Generation (RAG), chaining workflows, integrating databases/APIs with LLMs. | Provides modular workflows for your requirement engineering pipeline and connects agents to enterprise knowledge sources. |
| **Streamlit** | An open-source Python framework for building interactive web apps quickly. | Rapid prototyping of dashboards and UIs for ML/AI applications. | Used to create the user interface where stakeholders upload documents, review requirements, and visualize outputs. |

Table 3: Technologies in depth & use case

**AI-Driven Multi-Agent System Pipeline flowchart**

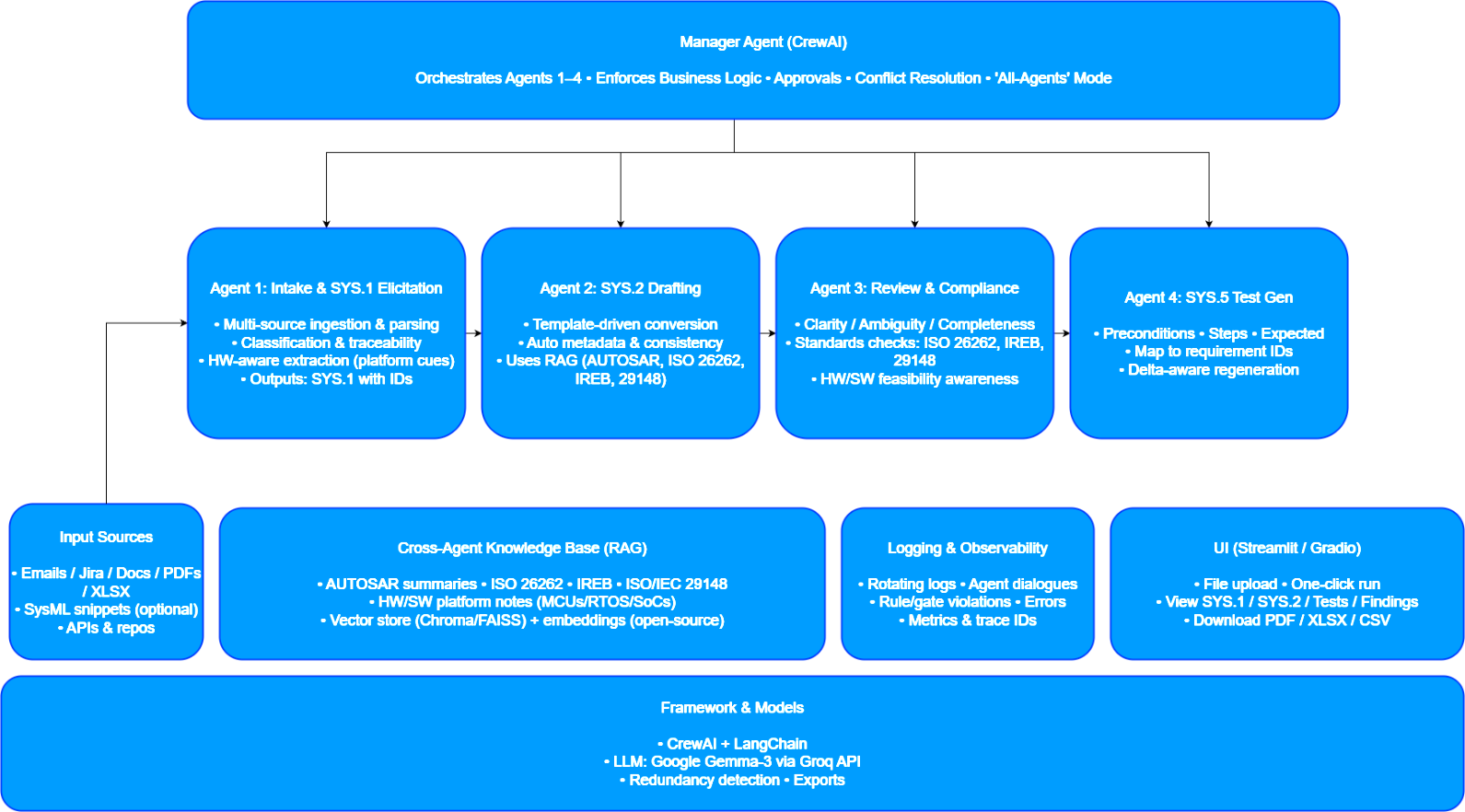


Figure 7: AI-Driven Multi-Agent System Pipeline

**Validation Strategy & Performance Metrics**

|  |  |
| --- | --- |
| **Agent** | **Validation Metrics** |
| **Agent 1 (SYS1 Elicitation)** | Precision/Recall of requirement extraction, Classification accuracy (Functional / Non-Functional / Regulatory). |
| **Agent 2 (SYS2 Drafting)** | Template completeness, Metadata prediction accuracy, Requirement clarity score. |
| **Agent 3 (SYS2 Review)** | Compliance score (IREB / ISO 29148), Reviewer disagreement rate, Ambiguity detection accuracy. |
| **Agent 4 (SYS5 Test Generation)** | Test-case coverage %, Traceability accuracy (Req <-> Test), Correction rate after review. |
| **Agent 5 (Manager)** | Orchestration efficiency, End-to-end pipeline execution time, Conflict-resolution success rate. |

Table 4: Validation Strategy & Performance Metrics

**Multi-Agent Requirement Engineering System – In Depth**

The system will be developed as a collection of intelligent agents, each powered by Google Gemma 3 models accessed via the Groq API and coordinated through CrewAI. The implementation will be carried out in Python, designed to automatically identify and load external input files from user-defined paths. These files will act as the primary source of contextual data for decision-making, analysis, and execution. The overall framework will be modular, configurable, and adaptable, making it suitable both for experimental use and integration into broader Artificial Intelligence workflows.

**1. Agent 1: Requirement Intake & Pre-Processing and SYS.1 Elicitation**

**Primary Role**: Understand and extract initial requirements from raw customer communication and elicitation of SYS.1 requirements

**Functions & Features**:

* **Feasibility Check**: Automatically assess project feasibility using predefined business rules and domain knowledge.
* **Check SYS**.1 as per ISO/IEC/IEEE 29148:2018 or ISO 26262 or INCOSE SE Handbook
* **Multi-Source Ingestion**: Support inputs from:
  + Emails, meeting transcripts, voice-to-text (STT) data
  + PPTs, TXT, PDFs, DOCX, XLSX, CSV
  + At the same time multiple inputs can be taken
* **Output File Format**: **TXT, PDFs, DOCX, XLSX, CSV**
* **Semantic Parsing**: Use NLP to extract intents, constraints, and stakeholders.
* **Requirement Categorization**: Classify requirements into functional, non-functional, regulatory, etc.
* **Traceability Links**: Establish traceability to original source artifacts.

**2. Agent 2: Requirement Drafting (SYS.2) & Structuring**

**Primary Role**: Generate structured draft requirements from parsed data of SYS.1 or manually selection in different formats (TXT, PDFs, DOCX, XLSX, CSV).

**Functions & Features**:

* **Template-Based Drafting**: Use configurable templates compliant with industry standards (IREB, IEEE 830).
* **Auto-Fill Attributes**: Populate requirement fields (ID, priority, rationale, etc.) automatically.
* **Requirement Rewriting**: Improve language clarity and remove ambiguity.
* **Dependency Mapping**: Identify logical dependencies and impacts across requirements.
* **Output File Format**: **TXT, PDFs, DOCX, XLSX, CSV**
* **Modularization**: Break complex requirements into manageable sub-requirements.
* **Suggest improvements** for any requirements that do not meet IREB standards and add button to update in front of that requirement to take the suggestion.
* **Classify each requirement** (e.g., Functional, Non-functional, Constraint, Assumption).
* Add verification method (either SYS.4 or SYS.5 and verification criteria)
* Priority Assignment and Release Planning

Allow users to assign a priority level to each requirement and plan its implementation across different software releases.

**Details**:

* 1. **Priority Assignment**: Users can assign a **Priority** to each requirement using a dropdown menu. Available priority levels:
     1. **High**
     2. **Medium**
     3. **Low**

The priority reflects the **importance** and **impact** of the requirement.

**3. Agent 3: System Requirement (SYS.2) Review, Compliance & Continuous Learning**

**Primary Role**: Evaluate, refine, and standardize requirements.

**Functions & Features**:

* **Check each requirement (SYS.2) against IREB (International Requirements Engineering Board), Check SYS**.2 as per ISO/IEC/IEEE 29148:2018 or ISO 26262 or INCOSE SE Handbook
* **IREB standards** for:
  + Clarity
  + Unambiguity
  + Completeness
  + Consistency
  + Verifiability
  + Modifiability
  + Traceability
* **Linguistic Analysis**: Highlight passive voice, vague terms, or conflicting statements.
* **Comment & Suggest Mode**: Provide detailed annotations and improvement suggestions.
* **App-Based Validation**: Accept/reject suggestions in real-time via web/app UI.
* **Self-Learning System**: Learn from accepted suggestions to improve future draft quality.
* Evaluates requirement clarity and completeness
* Detects ambiguous language
* Provides IREB-based improvement suggestions
* Assigns priority levels automatically

**4. Agent 4: Write System Testcases (SYS.5) from SYS.2**

**Primary Role**: Write the system testcases

Agent 4 can automatically retrieve the "sys2\_requirements\_reviewed.xlsx" file from the defined path (e.g., D:\AgentX\AutoTestGen\_MAPS\_Agents123\AutoTestGen\_MAPS\Inputs) or can be manually provided.

Additionally, there should be an option to manually upload the SYS.2 file. Once the file is obtained either automatically or manually - Agent 4 should automatically convert the SYS.2 requirements into SYS.5 test cases.

**Generate system validation test cases** for each requirement, including:

* SYS.2 Req. ID (which is uploaded using xlsx file)
* SYS.2 System Requirement (which is uploaded using xlsx file)
* Test Case ID
* Description
* Preconditions
* Test Steps
* Expected Results
* Pass/Fail Criteria
* Priority

**Functions & Features**:

* **Manual Edit Detection**: Detect and log manual overrides to track intent drift.
* **Impact Analysis**: Identify changes that affect upstream or downstream systems.
* Ensure SYS.2 to SYS.5 (**requirements-to-test-case traceability)** by mapping each test case to its corresponding requirement., Traceability Matrix: Mapping between requirement IDs and test case IDs
* Converts natural language requirements into structured test cases
* Generates logical and concrete test scenarios
* Creates detailed test steps and expected results
* Supports automotive and general software testing scenarios
* **Test Case Generation and Correction Workflow:** Ensure test cases are generated for each prompt or requirement. Allow manual corrections for any discrepancies (deltas), and implement autocorrection based on data or feedback post-review.

**Details**:

* **Test Case Generation**:
  + System should **automatically generate test cases** for each prompt or requirement.
  + If **test case generation is missing**, the system should flag it for review.
* **Delta Handling**:
  + **Delta** refers to the difference between expected and actual test case behavior/content.
  + System should detect deltas and **highlight them for manual inspection**.
* **Manual Correction**:
  + Users must be able to **manually correct test cases** when discrepancies are detected.
  + Interface should allow **inline editing or structured correction forms**.
* **Autocorrection (Post-Review)**: After manual correction or feedback, the system should:
  + - Apply **autocorrection** for similar future cases
    - Optionally reprocess related test cases based on updated data

**User Interaction**:

* Visual indicators for missing or outdated test cases.
* Editable fields or review mode for manual corrections.
* Review and feedback flow to trigger autocorrection.

**5. Agent 5: Manager** ([**https://www.crewai.com/**](https://www.crewai.com/))

**CrewAI: A platform for orchestrating Artificial Intelligence agents**

CrewArtificial Intelligence takes a different approach, offering a more structured platform for creating and managing Artificial Intelligence agents. It allows users to define agents with specific roles, goals, and backstories, facilitating a role-playing approach to task automation. CrewAI’s intuitive interface makes it easier for users to design agent interactions, assign tasks, and monitor the execution of these Artificial Intelligence crews. Built on top of, crew Artificial Intelligence leverages a rich ecosystem of tools and integrations, making it accessible to a broader audience, including business users who may not have deep technical expertise.

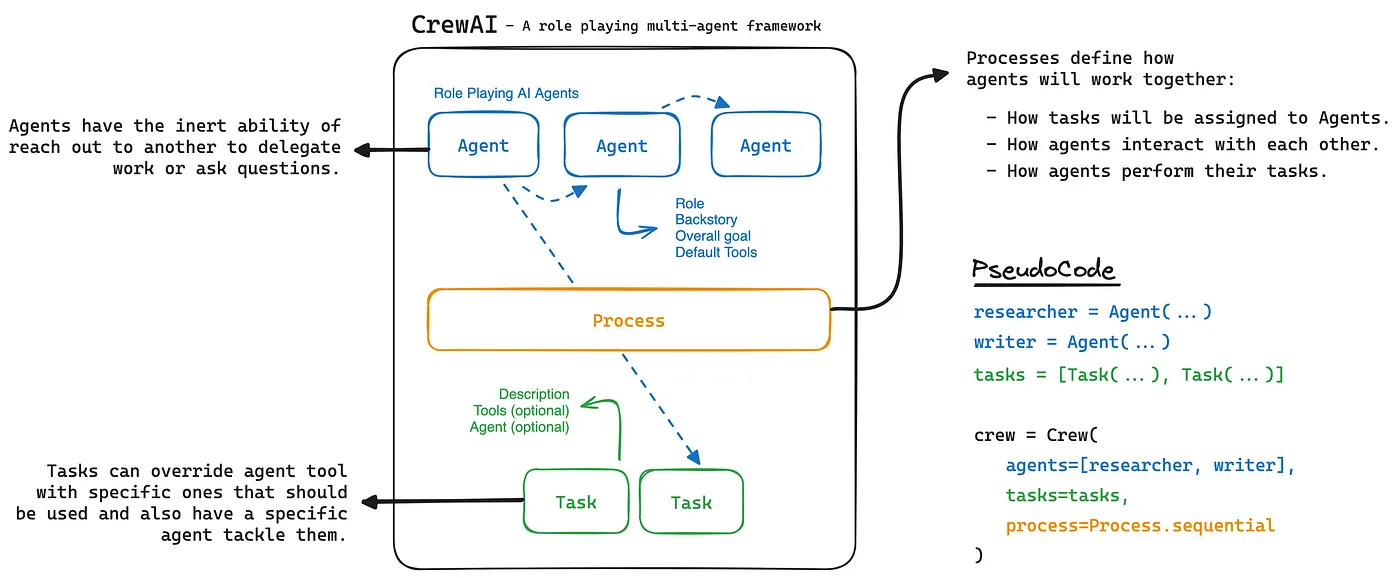


Figure 8: Multi-agent Framework

**LangChain**

LangChain is a ground breaking framework that empowers developers to build powerful applications by harnessing the capabilities of large language models. At its core, LangChain enables the seamless integration of language models with external data sources, unlocking a world of possibilities for leveraging the power of these cutting-edge AI systems.

One of the key strengths of LangChain lies in its ability to augment language models with retrieval capabilities. This approach, known as retrieval augmented generation ([RAG](https://skimai.com/aiyou-40-retrieval-augmented-generation-rag-in-enterprise-ai/)), allows language models to access and incorporate relevant information from external data sources, such as databases, APIs, or document repositories. By combining the language model’s natural language understanding and generation abilities with access to external knowledge, LangChain opens up new frontiers for building intelligent and context-aware applications for your enterprise.

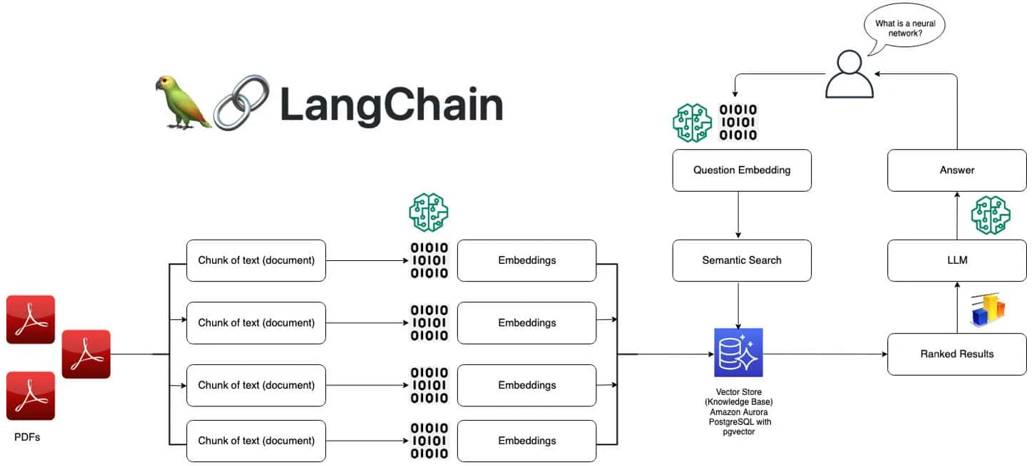


Figure 9: Langchain Framework

**Key Features of LangChain**

* **Retrieval Augmented Generation for Language Models**: LangChain’s retrieval augmented generation feature enables language models to leverage external data sources, enhancing their knowledge and providing more accurate and informed responses. This feature is particularly valuable for applications that require access to up-to-date or specialized information.
* **Composable Chains for Complex Workflows**: LangChain allows developers to create complex workflows by composing reusable chains. These chains can encapsulate a series of operations, such as data retrieval, processing, and generation, enabling the development of sophisticated applications with relative ease. This modular approach promotes code reusability and maintainability.
* **Off-the-Shelf Agents and Chains**: LangChain provides a collection of pre-built agents and chains that can be readily incorporated into applications. These off-the-shelf components cover a wide range of use cases, including question-answering, code generation, and data analysis, accelerating the development process and allowing developers to focus on higher-level tasks.
* **Support for Various Data Formats**: LangChain offers built-in support for a multitude of data formats, including plain text, PDFs, images, and structured data sources like databases and APIs. This versatility allows developers to seamlessly [integrate](https://skimai.com/maximizing-business-potential-how-to-integrate-llms-with-enterprise-data/) their applications with diverse information sources, enabling the creation of comprehensive and data-driven solutions.

By leveraging these key features, LangChain empowers developers to build powerful [enterprise AI](https://skimai.com/top-10-reasons-why-enterprise-ai-projects-fail/) applications that can leverage the full potential of large language models, while seamlessly integrating with external data sources and complex workflows.

The Manager Agent (Agent 5) in CrewAI acts as the orchestrator and controller of a multi-agent system, responsible for coordinating Agents 1–4, applying business logic, overseeing execution, and ensuring traceability of outputs.

**Core Functions of Manager Agent**

* The manager agent analyzes the overall goal, breaks it into discrete tasks, and delegates these tasks to specialized worker agents based on their skills.
* It holds authority to approve or reject intermediate and final outputs produced by other agents, ensuring quality and alignment with business objectives.
* The agent applies business logic and decision rules throughout the workflow, making executive decisions at key stages of task execution.

**Execution Order and Dependency Management**

* The Manager Agent controls the order in which tasks are executed, accounting for dependencies between tasks and agents.
* It uses hierarchical processes where the manager acts as a planner, and worker agents as executors, ensuring efficient task breakdown and coordinated execution.

**Conflict Resolution**

* The agent monitors for conflicts-such as overlapping responsibilities or contradictory outputs-and intervenes to resolve them so the workflow stays on track.
* Decision logic can include arbitration mechanisms, quality checks, and corrective action assignment.

**All-Agents' Mode**

* In this mode, the manager agent combines the capabilities of all agents, sequencing actions so that agents can work together or in succession to solve complex problems.
* This may involve parallel, sequential, or hybrid collaboration patterns depending on process configuration.

**Output & Traceability**

* The manager ensures all final deliverables are compiled and exported in downloadable formats, such as reports, CSVs, or PDFs, with traceability of each agent’s contribution.
* Detailed activity logs are maintained for auditability and debugging; each output traces back to the source agent, the task executed, and the decision steps followed.

**CrewAI Hierarchical Structure Example**

* The crew is defined with a manager agent parameter, assigning the orchestrator role.
* The process (often set to hierarchical) enforces the manager-worker operational flow.
* The manager agent is allowed to delegate, review, and reassign tasks, controlling the entire workflow.

|  |  |
| --- | --- |
| **Feature** | **Manager Agent Role** |
| Task Planning | Breaks down tasks, allocates to specialists |
| Output Approval | Quality checks and validation of agent deliverables |
| Decision Making | Applies business logic, resolves issues |
| Execution Management | Manages order, dependencies, and workflow |
| Conflict Resolution | Monitors conflicts, applies corrective action |
| All-Agents' Mode | Sequences/collaborates agent capabilities |
| Deliverable Packaging | Ensures outputs are downloadable and traceable |

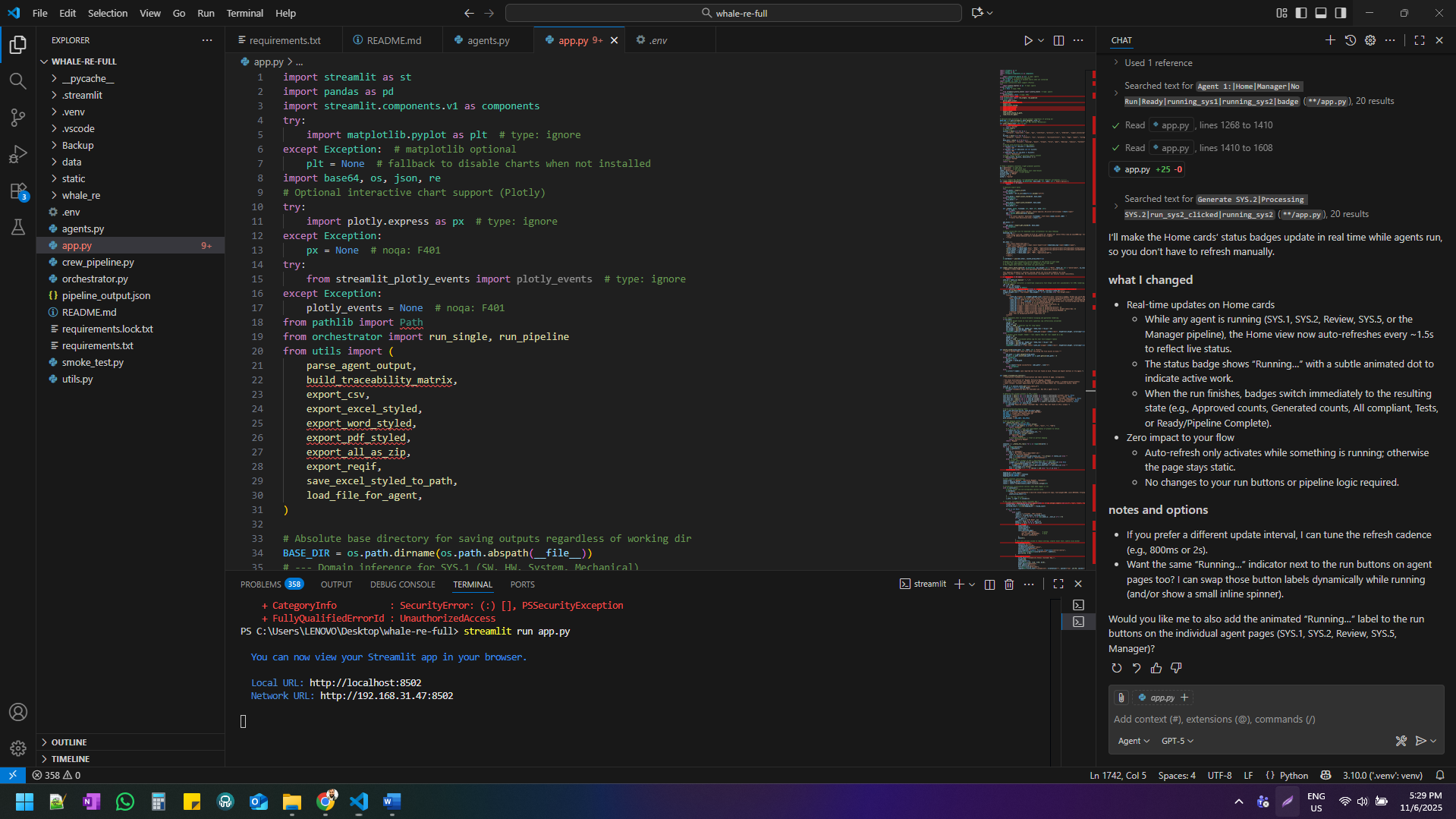
Table 5: Manager Agent Role

The Manager Agent is thus the backbone of CrewAI’s multi-agent orchestration, blending automated planning, execution control, and quality assurance for business-aligned results.

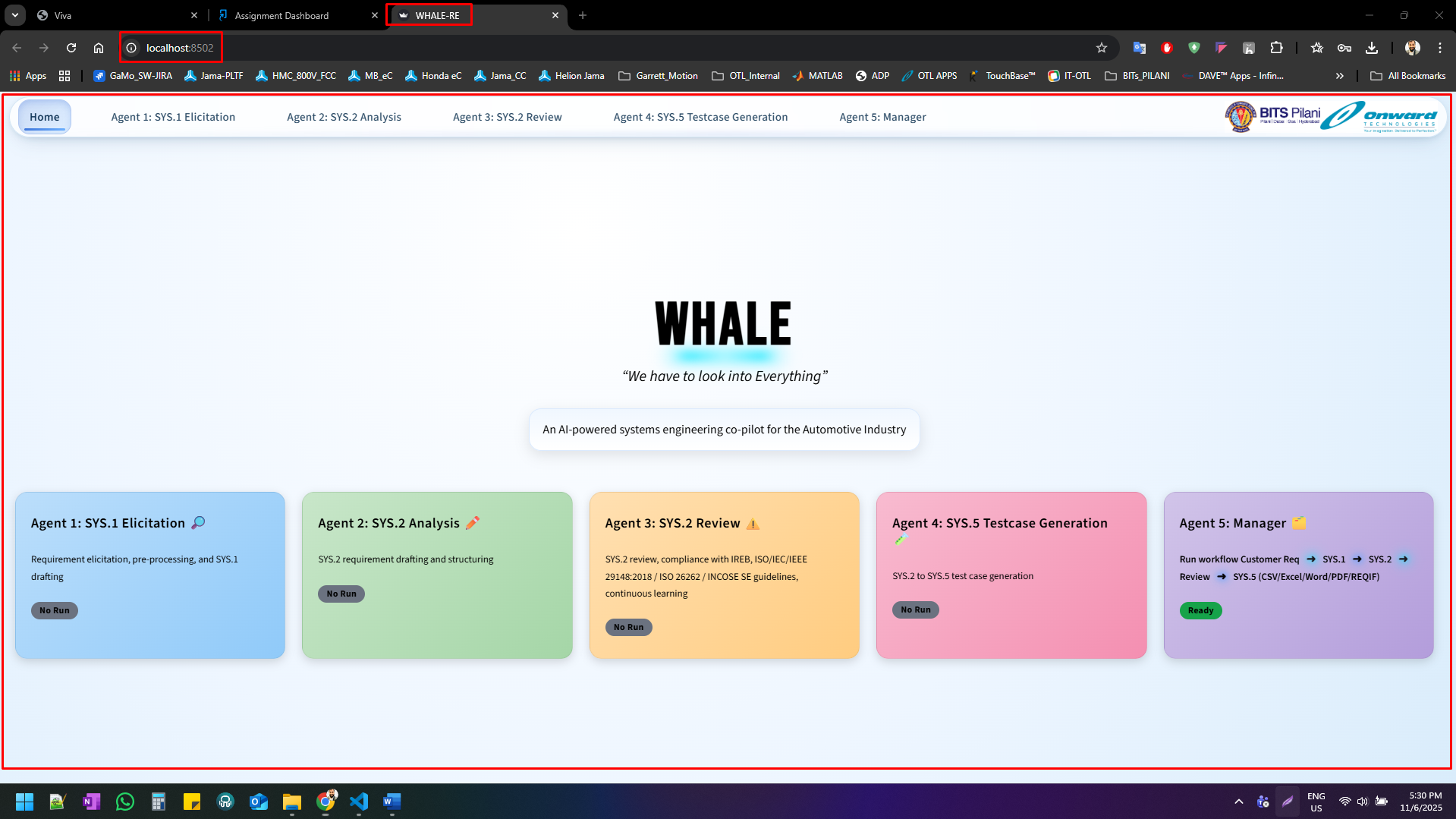
# Results

# Editor/IDE – Visual Studio Code

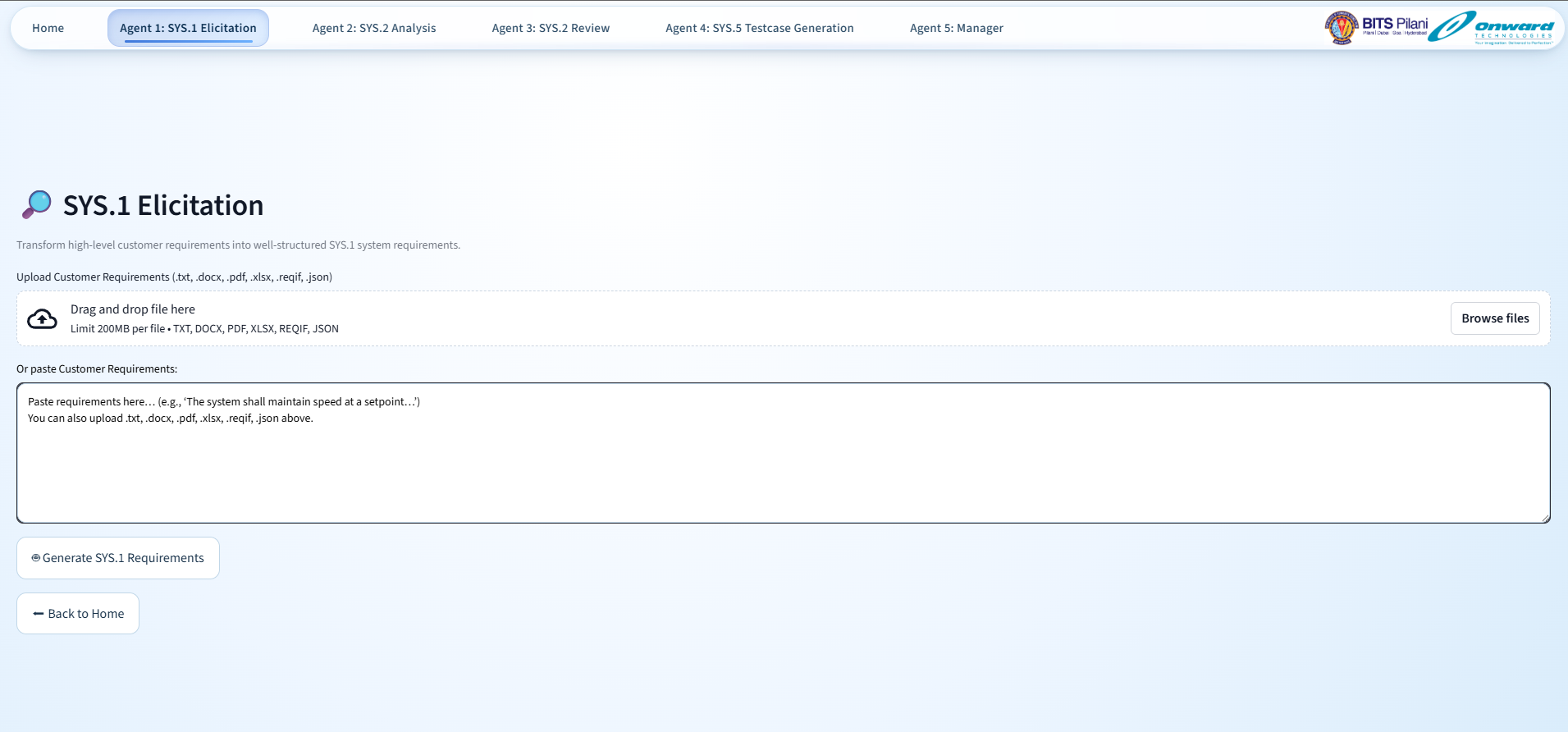
* Version: 1.105.1 (user setup)
* Commit: 7d842fb85a0275a4a8e4d7e040d2625abbf7f084
* Date: 2025-10-14T22:33:36.618Z
* Electron: 37.6.0
* ElectronBuildId: 12502201
* Chromium: 138.0.7204.251
* Node.js: 22.19.0
* V8: 13.8.258.32-electron.0
* OS: Windows\_NT x64 10.0.26200



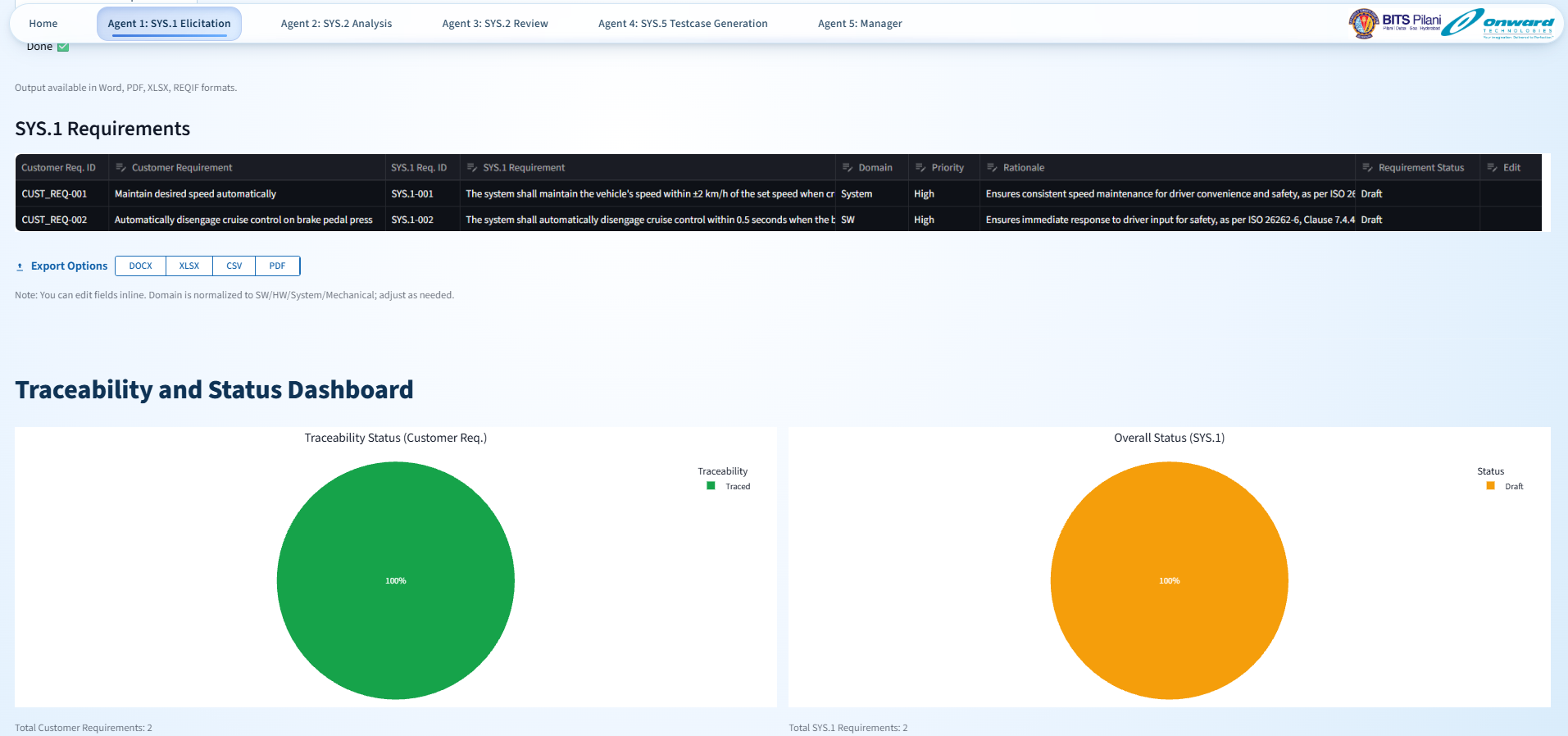
# Landing Page



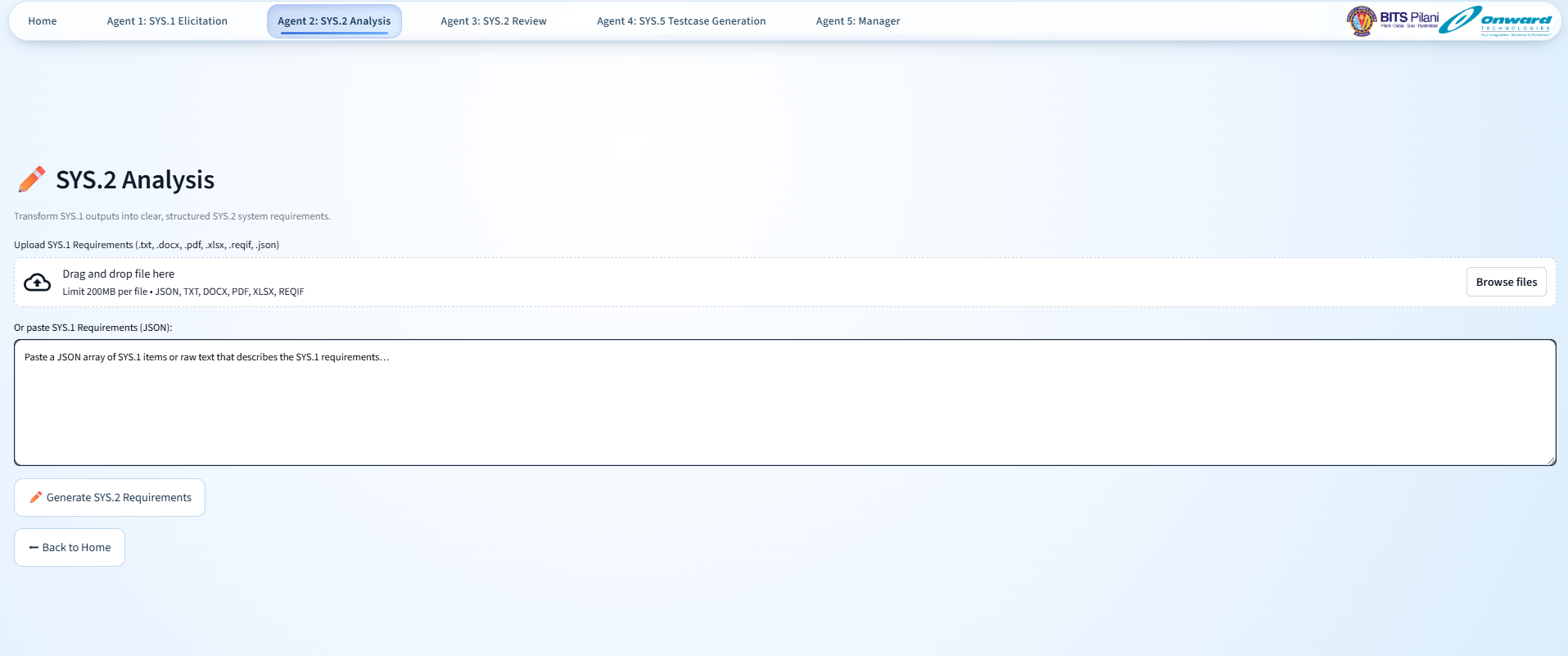
# Agent 1: SYS.1 Elicitation



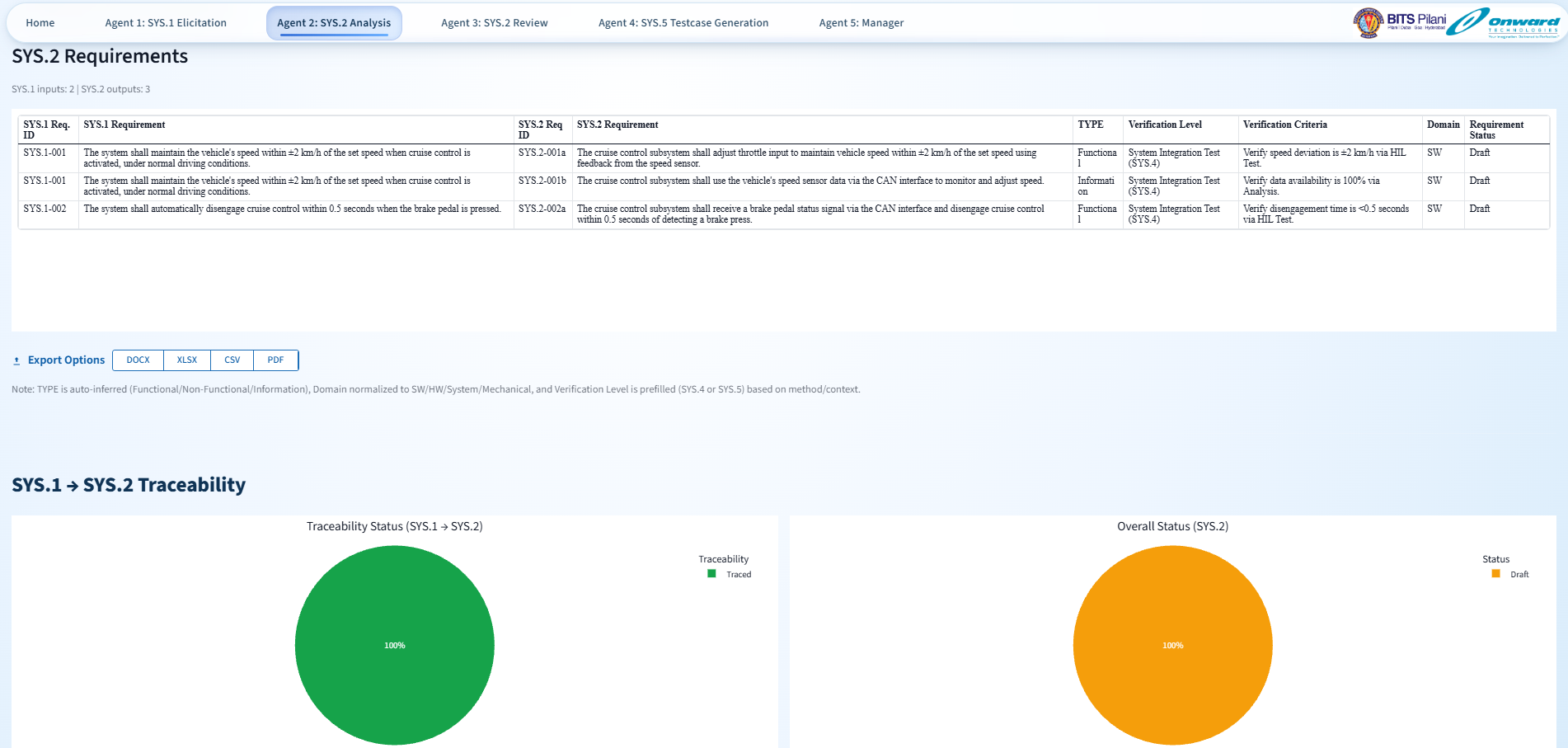
# Agent 1: SYS.1 Elicitation – Result



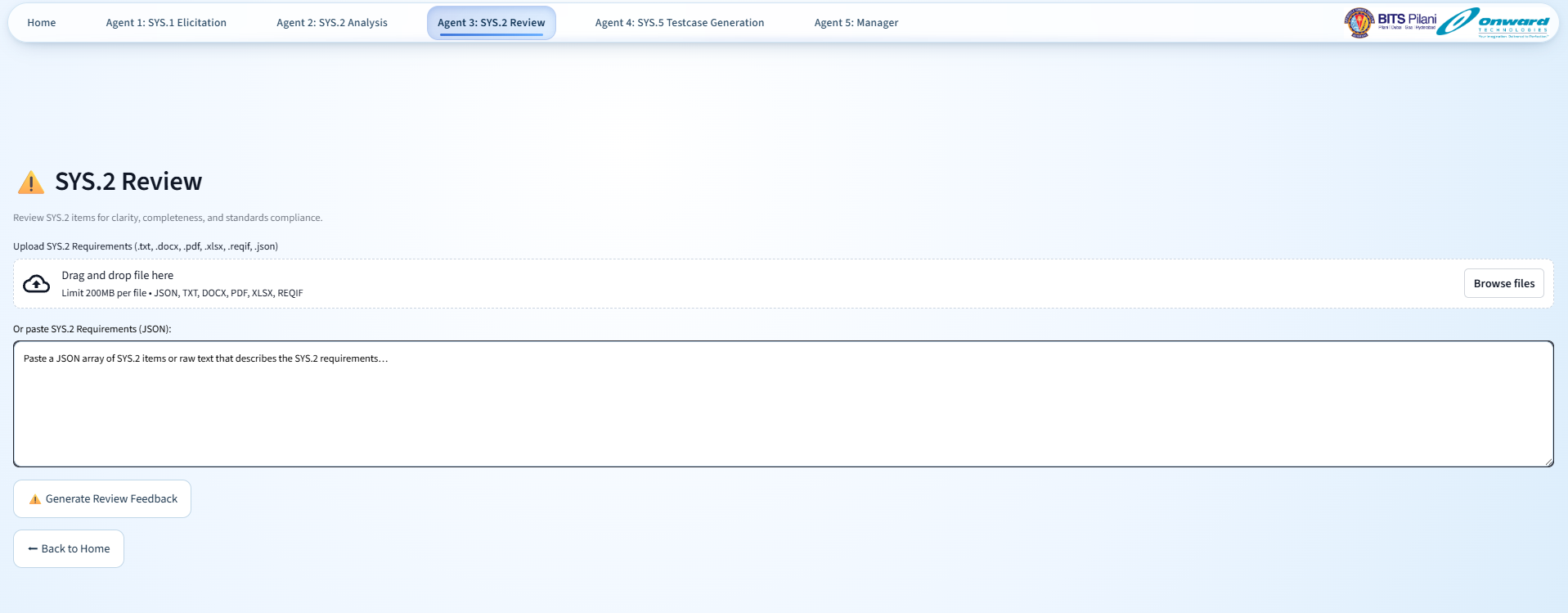
# Agent 2: SYS.2 Analysis



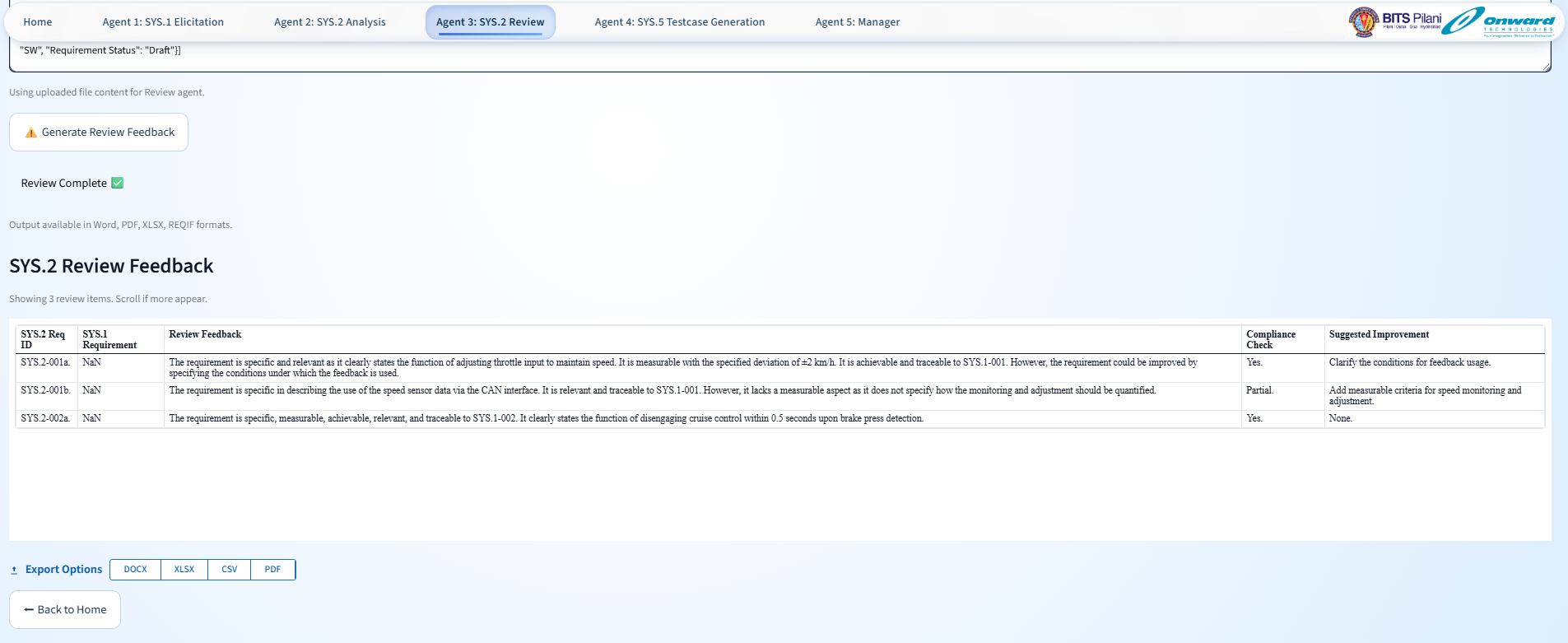
# Agent 2: SYS.2 Analysis - Result



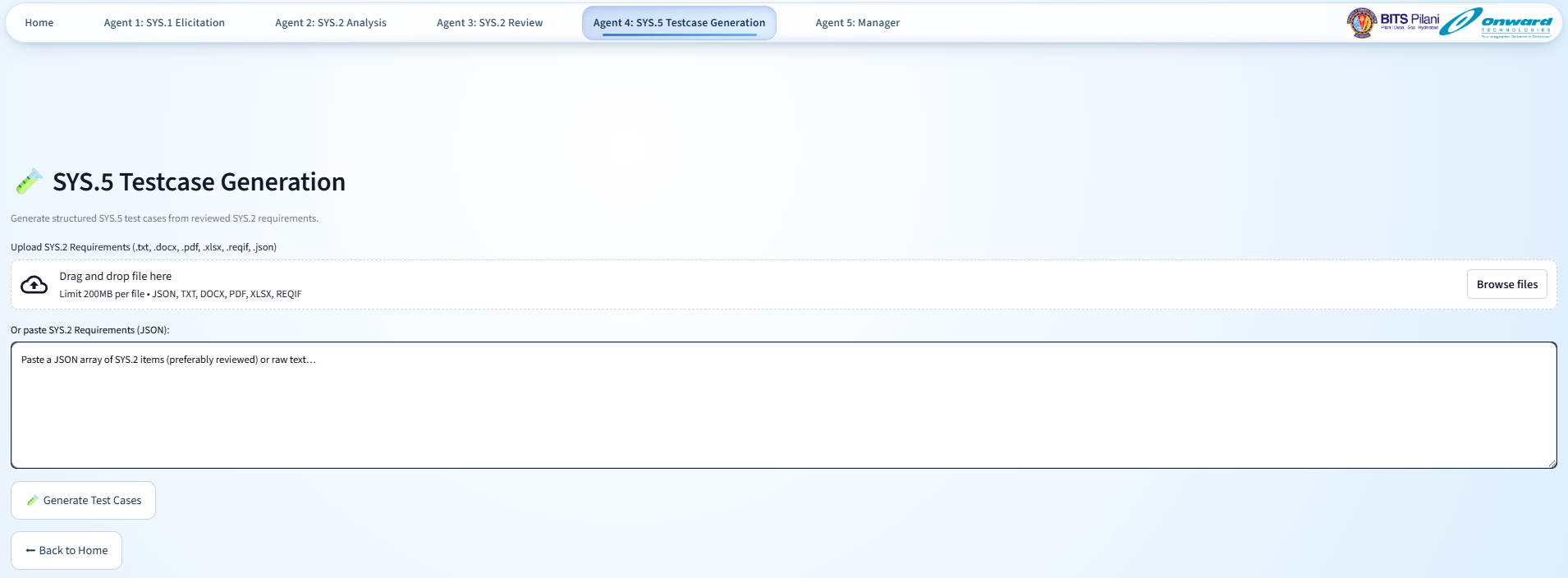
# Agent 3: SYS.2 Review



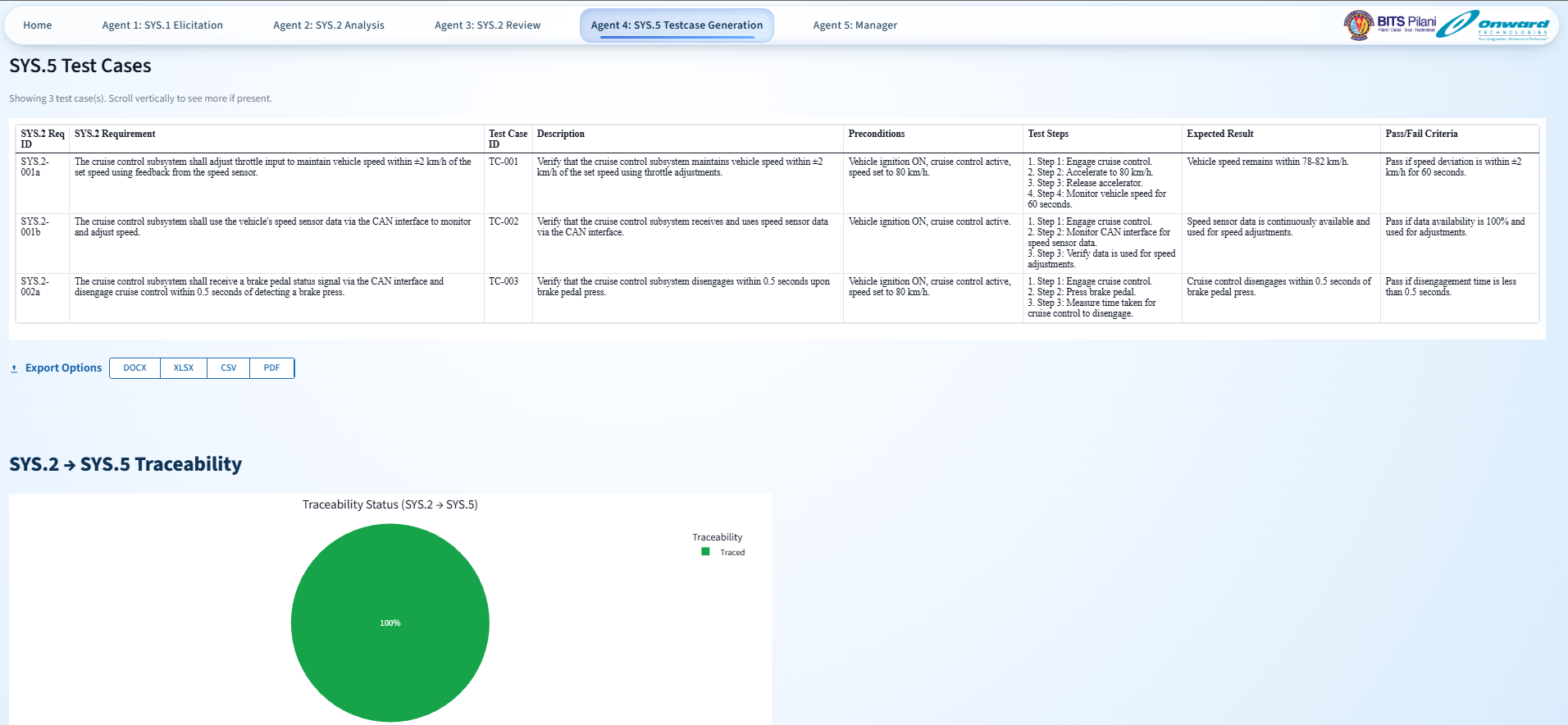
# Agent 3: SYS.2 Review – Result



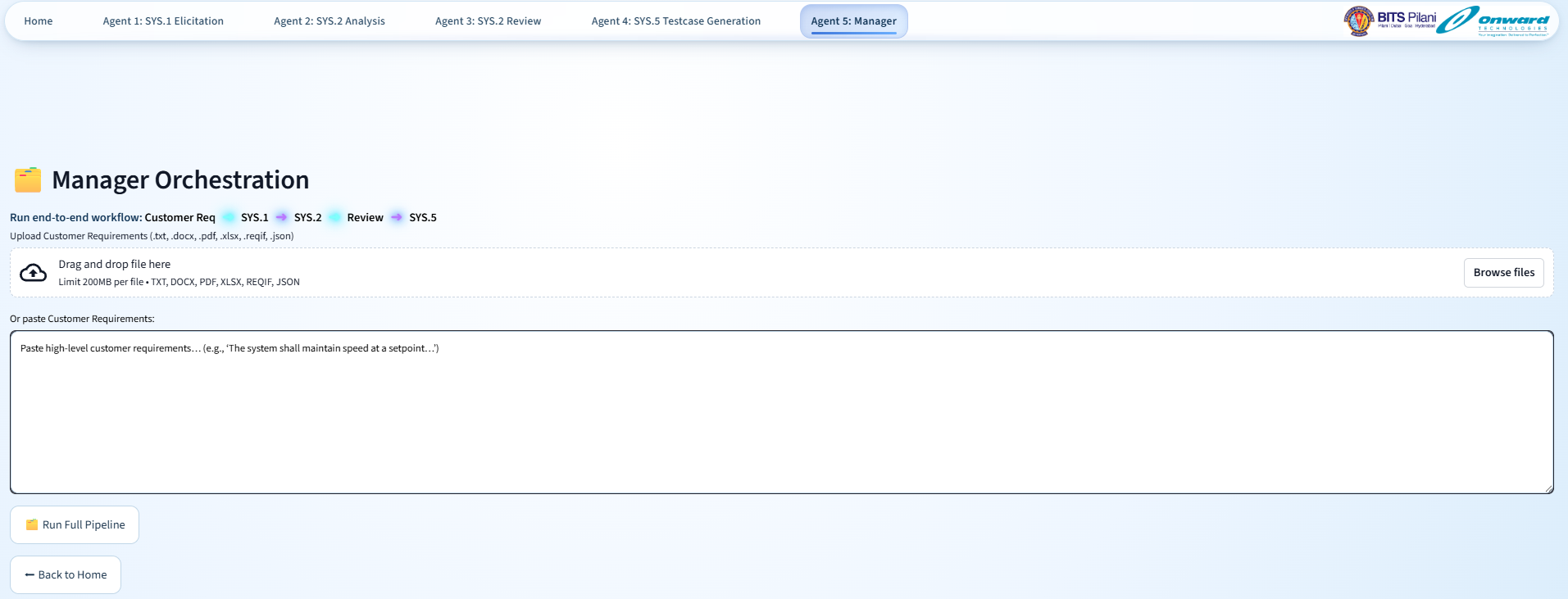
# Agent 4: SYS.5 Testcase Generation



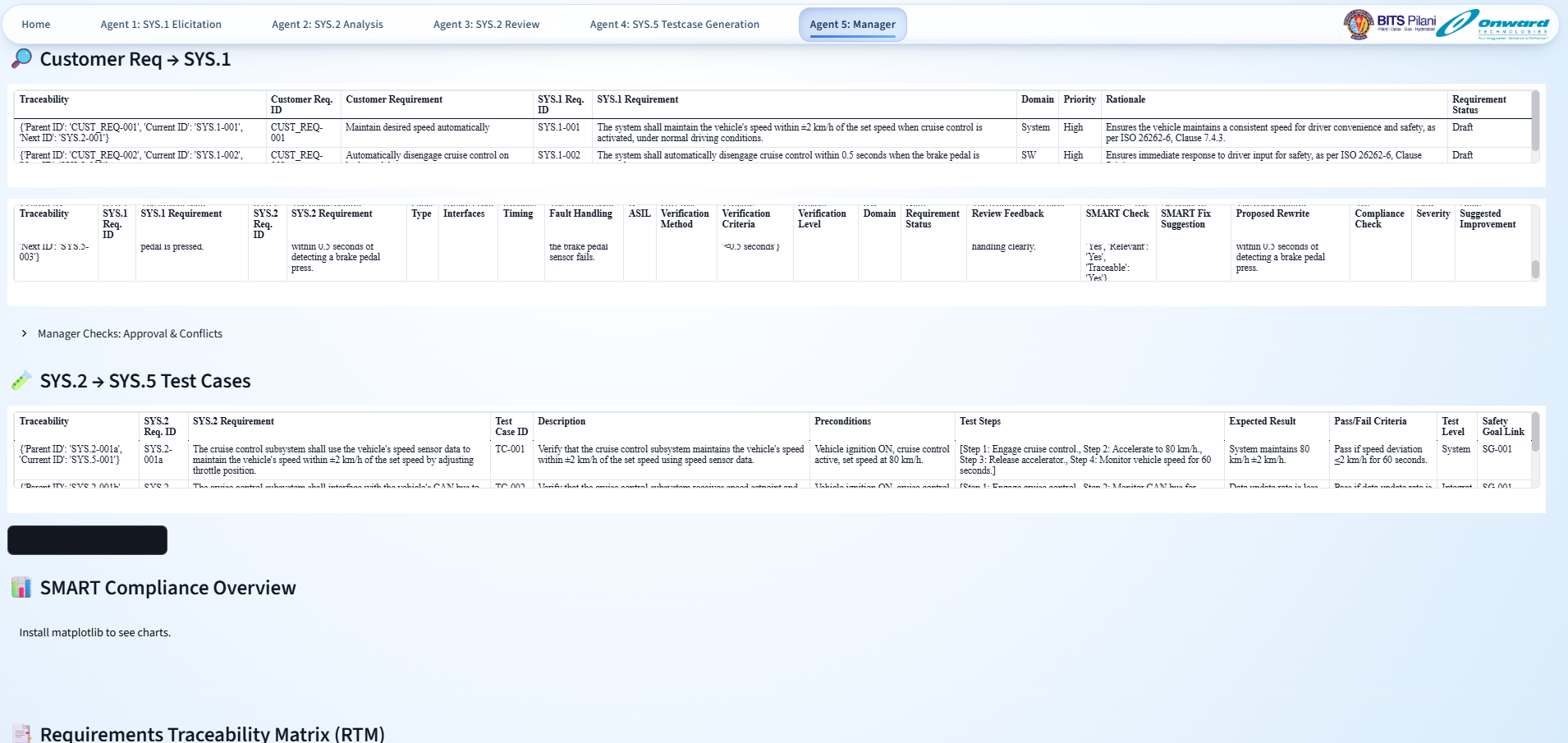
# Agent 4: SYS.5 Testcase Generation – Result



# Agent 5: Manager



# Agent 5: Manager – Result



**Additional features and miscellaneous topics**

(Some are yet to be implemented and can be implemented in the future)

**1. Global Features Across All Agents**

* **Audit Trail**: Maintain full traceability of who/what/when changes were made.
* **RBAC (Role-Based Access Control)**: Define permissions for different stakeholders (e.g., reviewer, author, customer).
* **Collaboration Mode**: Enable real-time editing, commenting, and review among cross-functional teams.
* **Multilingual Support**: Extract and draft requirements in multiple languages with translation and localization.
* **Integration with ML Models**: Optional integration for classification, anomaly detection, and summarization.
* **Input File Format: TXT, PDFs, DOCX, XLSX, CSV, & ReqIF**
* **Output File Format: Provide a button to download in .xlsx,.pdf,.docx,.csv formats with all the information.**
* **Add New Section called “Dashboard” for all agents separately (Pie chart) and all details**
* **Redundancy Check (Similarity Index Sheet) Across All Agents and Individually**
  + Implement a redundancy detection system using a **Similarity Index Sheet** to identify overlapping or duplicate entries **across all agents combined** and also **within each agent individually**.

**Details**:

* **Scope of Redundancy Check**:
  + **Cross-Agent Check**:  
    Detect redundancies between system requirements, prompts, or test cases **across all agents** in the system.
    - E.g., similarities between Agent 1’s and Agent 4’s entries.
  + **Agent-Specific Check**:  
    Detect redundancies **within each agent separately**, ensuring that entries inside a single agent don’t duplicate.
* **Similarity Index Sheet**:
  + For both checks, generate a **Similarity Index Sheet** with:
    - IDs or titles of compared items.
    - Agent source of each item.
    - Similarity score (numeric percentage or score).
    - Suggested action (e.g., review, merge, ignore).
    - Reviewer comments or notes.
  + The sheet should allow **filtering** or **grouping by agent** or **cross-agent pairs**.
* **Algorithms & Techniques**:
  + Use appropriate NLP or string similarity algorithms for accurate detection.
  + Consider semantic similarity to avoid false negatives.
* **User Interface**:
  + A unified view showing:
    - All redundancy pairs across agents.
    - Ability to toggle views between “All Agents” and “Single Agent.”
  + Inline actions to:
    - Merge duplicates.
    - Flag for review.
    - Add comments.
  + History tracking for all actions taken.
* **Batch Processing & Scheduling**:
  + Should allow scheduled runs for redundancy checks.
  + Users can trigger manual re-checks for specific agents or the entire system

1. **Future Expansion Roadmap**
2. **Tool Integration**

Integrate with industry-standard tools such as:

* **Jira** – for project and issue tracking.
* **DOORS** – for requirements management.
* **Polarion** – for end-to-end ALM (Application Lifecycle Management).

1. **Automated Diagram Generation**

Implement automatic **UML** and **flowchart** generation directly from structured or semi-structured requirements.

1. **Unit Test Generation**

Auto-generate **unit test cases** based on **functional requirements** and enable seamless linkage with **CI/CD pipelines** for automated validation.

1. **Multilingual Support**

Provide support for multiple languages to enhance collaboration across **global and diverse teams**.

1. **Agent 7: SWE.1 – Writing Phase**

Develop an intelligent agent capable of **authoring requirements** according to **SWE.1** of the Automotive SPICE model.

1. **Agent 8: SWE.1 Review – IREB/CPRE Aligned**

Introduce an AI agent for **reviewing requirements**, ensuring compliance with **IREB**/**CPRE** standards and best practices.

1. **Agent 9: SWE.5/SWE.6 – Test Case Generation**

Automatically generate **test cases** and **verification criteria** from refined requirements in alignment with **SWE.5**(SW Integration Test) and **SWE.6**(SW Qualification Test).

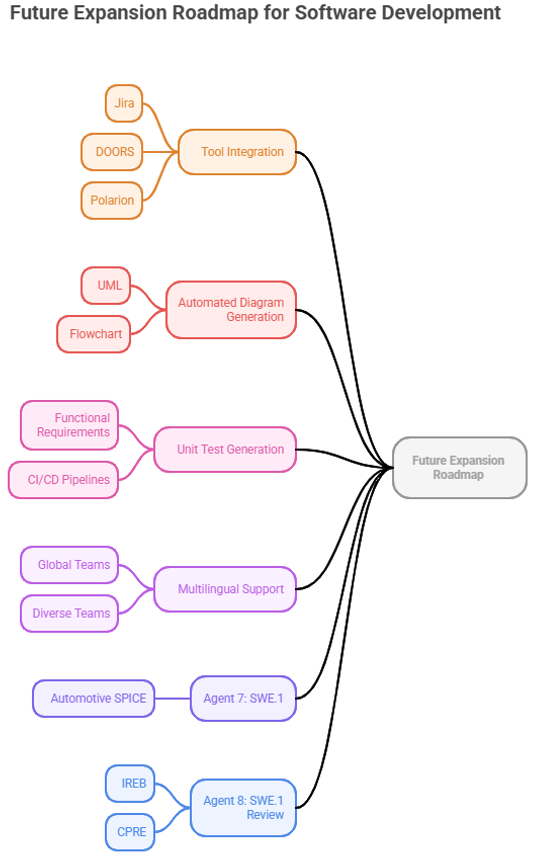


Figure 10: Future Expansion Roadmap

1. **Data Privacy & Enterprise Security Considerations**

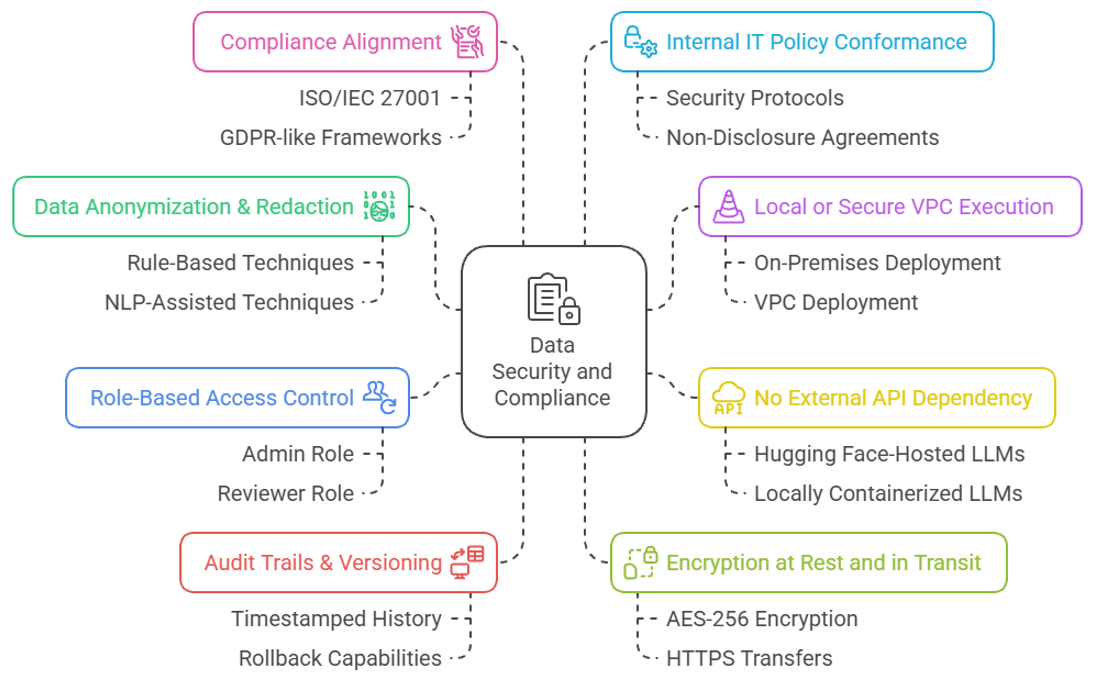
Given the nature of the system, which processes sensitive enterprise information such as stakeholder communications, internal requirement documents, and compliance artifacts, data privacy and security have been incorporated as foundational design principles. The following measures ensure enterprise-grade protection:

Figure 11: Data Privacy & Enterprise Security Considerations

(**Note**: Applicable when deployed for the commercial use)

|  |  |
| --- | --- |
| **Risk** | **Mitigation Strategy** |
| **Data leaving enterprise boundaries (e.g., sent to Groq API or Hugging Face servers)** | - Use **on-premise deployment** or **self-hosted models** where possible (e.g., running Gemma locally). - Mask, anonymize, or tokenize sensitive fields (names, IDs, financials) before sending data to external APIs. |
| **Exposure of confidential requirements/documents** | - Implement a **pre-processing filter** that removes or obfuscates sensitive terms. - Encrypt data in transit using **TLS/HTTPS**. |
| **Multi-agent communication leakage (agents sharing too much context)** | - Apply **role-based access control (RBAC)** so each agent only accesses data relevant to its task. - Use **data minimization principles** → don’t share entire documents when only a subset is required. |
| **Data retention by external services** | - Configure API calls with **no logging** (if provider supports it). - Add an internal policy to avoid uploading classified or regulatory-restricted documents to third-party APIs. |
| **Auditability and compliance** | - Maintain **audit logs** of all inputs/outputs exchanged with agents. - Periodically review for GDPR/ISO 27001/enterprise data compliance. |
| **Enterprise confidentiality** | - Consider a **Retrieval-Augmented Generation (RAG)** approach: keep sensitive data in a **local vector database** and only send embeddings or queries to the model, not raw documents. |

Table 6: Risks & Mitigation Strategies

1. **Maintaining Privacy When Feeding Data to APIs and Agents**

Given that the proposed system involves exchanging enterprise requirements and artifacts with external APIs and multi-agent frameworks, maintaining privacy is critical. The following measures are recommended to ensure confidentiality and compliance:

1. **Data Minimization and Anonymization**
   * Only essential information is shared with each agent.
   * Sensitive attributes (e.g., customer names, IDs, financial details) are anonymized or masked before processing.
2. **On-Premise or Controlled Deployment**
   * Wherever feasible, Large Language Models (LLMs) such as Google Gemma or Hugging Face models should be deployed on local or enterprise-controlled servers.
   * This prevents raw documents from leaving organizational boundaries.
3. **Secure Data Transmission**
   * All data exchanged with APIs must be encrypted using TLS/HTTPS.
   * Temporary in-memory storage is preferred over long-term persistence to avoid leakage risks.
4. **Access Control Between Agents**
   * A role-based access control (RBAC) mechanism ensures that each agent only receives the data relevant to its assigned task.
   * Context sharing between agents is strictly limited to reduce unnecessary exposure.
5. **Data Retention Policies**
   * External API usage must avoid logging or storing input data.
   * Where logging is unavoidable (e.g., debugging), data must be sanitized and retention limited to enterprise policy.
6. **Auditability and Compliance**
   * The system maintains a full audit trail of all inputs, outputs, and agent interactions.
   * This supports regulatory compliance (GDPR, ISO 27001, ISO 26262) and facilitates internal audits.
7. **RAG-Based Knowledge Retrieval**
   * Instead of sending entire documents to an API, a retrieval-augmented generation (RAG) mechanism stores sensitive content in a local vector database.
   * Only embeddings or queries are exchanged with LLMs, significantly reducing privacy risks.

By adopting these practices, the system ensures that enterprise-grade confidentiality is preserved even when interacting with external APIs or orchestrated AI agents.

1. **Handling Confidentiality of Customer Requirements**

The agents are planned to be **deployed within the customer’s own secured IT environment**, with **localized processing of requirements**. This ensures confidential data never leaves enterprise boundaries. Cloud-based options may be used only in controlled settings with anonymized inputs, but the recommended mode for industry deployment-especially in safety-critical domains like automotive-is **offline/on-premise hosting** with customer-controlled servers.

1. **Challenges and Future Directions**

While the proposed AI-driven multi-agent framework demonstrates strong potential for automating requirement elicitation, drafting, review, and test-case generation, several technical and organizational challenges must be addressed before such a system can be adopted in safety-critical industrial settings. These challenges revolve around model consistency, adequacy of training data, deployment strategies, confidentiality concerns, integration with existing requirement management tools, and continuous improvement through user feedback.

A first challenge is ensuring **consistency of AI-generated results**. Large Language Models (LLMs) are inherently probabilistic, meaning that repeated runs on the same input may yield slightly different outputs. In requirements engineering, however, consistency is essential for ensuring traceability and compliance. Without reliable outputs, downstream activities such as validation and test-case generation may be compromised. Approaches such as deterministic decoding, rule-based post-processing, and the use of requirement “memory stores” to anchor outputs in previously validated content are therefore critical to achieving predictable behaviour.

Another key issue relates to the **sufficiency of training data**. While general-purpose LLMs are trained on vast corpora, they often lack coverage of specialized engineering standards such as ISO 26262 or DO-178C. This creates a risk of domain mismatch, where models fail to recognize terminology or compliance-specific requirements. Addressing this gap requires the use of Retrieval-Augmented Generation (RAG) with curated knowledge bases, alongside fine-tuning on proprietary datasets where confidentiality constraints allow. Continuous updates are also needed to align with evolving regulations and best practices.

A further challenge lies in **balancing local versus cloud-based AI models**. Cloud deployment offers scalability and access to powerful models but raises significant concerns around confidentiality and compliance, particularly when handling sensitive OEM or Tier-1 data. Local deployment, on the other hand, ensures privacy but may be constrained by hardware limitations and model performance. A hybrid approach appears most viable: deploying lightweight local models for sensitive processing while leveraging cloud-based inference for non-critical or anonymized tasks, depending on enterprise risk tolerance and regulatory context.

Ensuring the **confidentiality of OEM and Tier-1 requirements** is especially critical, as these artifacts often contain safety-critical intellectual property. For this reason, the proposed framework emphasizes deployment within customer-controlled infrastructure, supplemented by strict role-based access controls (RBAC), anonymization mechanisms, and comprehensive audit trails. On-premise hosting of models ensures that requirement documents remain within enterprise boundaries, with optional cloud processing available only for sanitized inputs.

Equally important is the **integration of the framework with existing requirement management tools**, such as IBM DOORS, Polarion, or Jira. Enterprises rely heavily on these platforms for traceability, compliance management, and project monitoring. A standalone AI pipeline, while powerful, risks becoming siloed if not linked with existing ecosystems. Integration can be achieved through export formats (e.g., CSV, ReqIF) and APIs, as well as synchronization of the agent-generated traceability matrix with enterprise requirement repositories.

Finally, achieving long-term effectiveness requires mechanisms for **continuous improvement through user feedback**. No AI system can be expected to achieve perfect accuracy from the outset, particularly in specialized domains. Capturing reviewer corrections, annotations, and acceptance decisions provides valuable training signals that can be fed back into the system. Techniques such as reinforcement learning with human feedback (RLHF), active learning on low-confidence outputs, and incremental fine-tuning offer pathways for improving the accuracy and reliability of local models over time without requiring complete retraining.

In summary, while the proposed system represents a significant advance in the automation of requirements engineering, addressing these challenges will be essential for ensuring its robustness, trustworthiness, and industrial adoption. Tackling issues of model consistency, training sufficiency, deployment strategy, confidentiality, tool integration, and user-driven refinement will transform the framework from a promising prototype into a qualified enterprise-grade solution aligned with international safety standards.

1. **Quantitative Objectives of the Multi-Agent System**

The proposed multi-agent framework is not only intended to automate requirement engineering activities but also to **deliver measurable improvements** in terms of effort reduction, accuracy, and processing speed. To validate the effectiveness of the system, each agent is associated with quantitative objectives. These targets will serve as benchmarks for implementation and evaluation.

Agent 1 – SYS.1 Elicitation

* **Objective:** Reduce manual effort in requirement intake and classification by **at least 50%** compared to baseline manual methods.
* **Metrics:**
  + Time saved per requirement ingestion (minutes).
  + Precision/recall of requirement extraction.
  + % Reduction in manual transcription and categorization.

Agent 2 – SYS.2 Drafting

* **Objective:** Shorten requirement drafting and structuring time by **40%** while maintaining compliance with IREB/ISO clarity standards.
* **Metrics:**
  + Average time taken to produce a SYS.2 draft requirement.
  + % Completeness of templates auto-filled.
  + % Reduction in ambiguity/linguistic errors detected.

Agent 3 – SYS.2 Review & Compliance

* **Objective:** Automate at least **60% of compliance checks** (clarity, consistency, ambiguity, completeness) to reduce reviewer workload.
* **Metrics:**
  + Coverage of IREB/ISO/INCOSE compliance checks automated.
  + Reviewer disagreement rate (<10% target).
  + Reduction in average review cycle time (hours per requirement set).

Agent 4 – SYS.5 Test Case Generation

* **Objective:** Achieve **70% automation of test-case generation** from SYS.2 requirements while ensuring traceability.
* **Metrics:**
  + % of requirements with at least one generated test case.
  + % Accuracy of requirement-to-test-case traceability mapping.
  + Time reduction in test-case authoring (baseline vs. automated).

Agent 5 – Manager Agent (Orchestration)

* **Objective:** Reduce end-to-end lifecycle execution (SYS.1 → SYS.5) by **at least 45%** while maintaining traceability and quality.
* **Metrics:**
  + Orchestration efficiency (% successful pipeline runs without manual intervention).
  + Average end-to-end processing time.
  + Conflict-resolution success rate (>85%).

1. **Out of scope and manual steps**

* Final Acceptance Of SYS.1 And SYS.2
* Acceptance Comments and Dispositions
* Domain Allocation (SYS/HW/SW)
* Safety Classification
* Compliance Attestation
* Baseline Creation
* And Change Approvals

remain manual governance activities performed by authorized stakeholders per ISO/IEC/IEEE 29148 processes.

The system generates drafts, quality findings, and test cases to support these decisions but does not replace them.

Following export, importing Excel/CSV/ReqIF deliverables into enterprise requirement tools (IBM DOORS/DOORS Next, Jama Connect, Polarion/Codebeamer) is performed by project roles using each tool’s ReqIF/CSV import workflows and mappings.

1. **Plan for the agent to detect “orphan” customer requirements that map only to SW or HW**

Propose derived system-level abstractions and interface/constraint candidates, and route them for human allocation/approval with full bidirectional traceability and rationale capture. Final allocation decisions remain manual per requirements engineering practice.

**Why this happens**

* Many incoming stakeholder/customer items are phrased as implementation-specific (SW/HW), which skips the neutral system view expected in systems engineering; ISO/IEC/IEEE 29148 promotes transforming stakeholder needs into system requirements before allocation to elements.
* Requirement’s allocation is a process of assigning requirements across system levels and elements; when only SW/HW statements exist, a system-level abstraction and interfaces often must be derived to maintain a coherent hierarchy.

**Agent behavior to manage it**

* Detect implementation-coupled wording: flag items that specify components or technologies (e.g., “software shall…”) without a parent system requirement, using heuristics aligned to implementation-neutral requirement guidance.
* Propose a candidate system requirement: auto-suggest a system-level “what” statement that generalizes the customer intent, keeping implementation out; present alongside the original SW/HW statement for reviewer decision.
* Suggest interface/constraint requirements: where HW<->SW boundaries are implied, draft interface and constraint candidates to anchor allocation and avoid gaps across elements.
* Create trace links: establish provisional bidirectional traceability Stakeholder → System (proposed) → Allocated SW/HW, with rationale and status “Pending Allocation Approval.”

**Human-in-the-loop decisions**

* Approve or edit the proposed system abstraction, confirm allocation to SW/HW elements, and document rationale; this preserves governance and fulfills 29148 outcomes for requirements definition and allocation.
* Validate interface requirements and align with architecture boundaries before baselining; interface ownership and boundaries are an architectural responsibility.

**Workflow design in the agent**

* Intake checks: if a requirement targets only SW/HW and lacks a linked system parent, add label “Allocation gap” and push to an Allocation Queue.
* Derivation step: generate a candidate System Requirement and, if appropriate, candidate interface specs with links back to the originating item; mark as Derived with change history.
* Review gate: require reviewer approval for “Parent System Requirement” and “Allocation Targets” before the item can move to drafting/review; the gate enforces allocation completeness.
* Traceability enforcement: do not allow test generation or lower-level verification planning if the system-level parent is missing; the RTM should show red status for missing upstream links.

**Data artifacts and views**

* Allocation Gap Report: list of SW/HW-only items lacking a system parent, with suggested abstractions, interfaces, and impact; exportable to Excel/CSV/ReqIF for team review.
* Rationale fielding: mandatory fields for “Allocation rationale,” “Architect reviewer,” and “Decision” to meet traceability best practices.

**Quality rules to embed**

* Implementation-neutral rule: system-level requirements must avoid specifying SW/HW unless constrained; flag violations and recommend rewrites.
* Allocation completeness rule: every stakeholder item must trace to at least one system requirement; every SW/HW requirement must trace up to a system requirement and down to tests; gaps block baselining.

**Edge cases and handling**

* Customer explicitly mandates SW/HW: if the stakeholder imposes a component-level constraint, keep it as a constraint at the system level and still create a parent system requirement that captures the “what,” linking the constraint appropriately.
* Interface-only inputs: when the customer gives only HW–SW interface statements, treat them as system interface requirements and allocate to both elements with ICD linkage.

**Governance and limits**

* Final allocation decisions, baseline creation, and acceptance of the newly derived system requirements remain manual, owned by systems/architecture reviewers per 29148 practices.
* The agent assists by proposing and tracing; it does not finalize allocations or attest compliance.

1. **Business Logic**

The framework will incorporate explicit validation rules and well-defined acceptance criteria. Example business logic will include enforcing compliance score thresholds, rejecting outputs that exhibit specific defect types, and verifying the completeness of requirements traceability.

1. **Integration of Hardware and Standards Knowledge**

All agents will operate with an awareness of relevant hardware and software platforms as well as applicable industry standards. The knowledge base will encompass domain-specific information, such as the AUTOSAR architecture and its modules, ISO 26262 functional safety standards, and additional standards including IREB and ISO/IEC 29148. This knowledge will be integrated using a Retrieval-Augmented Generation (RAG) approach, relying on curated summaries and authoritative technical references.

1. **Logging Mechanism**

A centralized logging infrastructure will be implemented to ensure transparency and auditability of the system. The logging system will capture agent interactions, task outputs, errors, warnings, and any violations of defined business logic rules. To maintain efficient storage usage, rotating log files will be employed.

1. **Tool Qualification & Limitations**

Given the potential application of this system in safety-critical domains such as automotive and aerospace, **tool evaluation and qualification** are a mandatory step to ensure compliance with standards like **ISO 26262 (Part 8)** and **Automotive SPICE**. The following considerations outline the strategy for tool qualification and highlight the limitations users must be aware of.

**Tool Qualification Approach**

1. **Tool Confidence Level (TCL) Assessment**
   * The impact of tool errors on system safety will be analyzed. For example, if a missed requirement leads to untested safety functionality, the tool may be classified with a higher TCL.
   * Based on TCL, the necessary rigor of qualification activities will be defined.
2. **Validation Against Golden Data**
   * Tool performance will be benchmarked against manually validated “gold standard” datasets.
   * Metrics such as **precision, recall, requirement coverage percentage, and traceability accuracy** will be recorded to provide evidence of reliability.
3. **Qualification Evidence**
   * Reports documenting tool validation, error analysis, and residual risk assessment will be prepared as qualification artifacts.
   * These will support both internal audits and external certification processes.

**Error Detection & Reporting**

The system will not only generate outputs but also provide **feedback on its confidence and limitations**:

* **Error Logs:** All parsing errors (e.g., unreadable files, OCR failures, malformed inputs) will be captured with clear warning messages.
* **Confidence Scoring:** Each requirement will be associated with a confidence score, enabling reviewers to prioritize manual checks.
* **Coverage Metrics:** The tool will estimate coverage by reporting how many requirements were extracted relative to expected totals.
* **Traceability Matrix:** Automatically generated mappings between input artifacts and extracted requirements will allow detection of missing links.

**User Assurance & Manual Safeguards**

While the system significantly reduces manual effort, it cannot yet guarantee **100% completeness** in requirement extraction, especially for low-quality or non-standard inputs. Users are therefore expected to:

1. **Cross-Verify Outputs:** Compare tool-generated outputs with original inputs, focusing on flagged low-confidence or error-prone segments.
2. **Manual Sampling:** Perform random spot-checks on extracted requirements against the source to ensure completeness.
3. **Hybrid Workflow:** In safety-critical projects, maintain a parallel manual review process for high-risk inputs such as scanned images, handwritten notes, or blurred diagrams.

**Limitations & Future Enhancements**

* The tool is dependent on **input quality** (e.g., clear digital text performs better than scanned or handwritten artifacts).
* OCR and parsing limitations may cause occasional requirement omissions.
* Current safeguards rely on **error reporting and manual verification**; future iterations aim to provide automated completeness guarantees by integrating redundancy detection and advanced multimodal OCR models.

# Design Considerations

The design of the proposed **AI-Driven Multi-Agent System for Requirement Elicitation, Analysis, Review, and Test Case Generation** was carefully structured to address scalability, compliance, usability, and enterprise-level security. The following subsections outline the major considerations that guided system design.

1. **System Architecture**
2. **Multi-Agent Framework**: The system follows a modular architecture with five primary agents:
   * **Agent 1** – Requirement Intake and SYS.1 Elicitation
   * **Agent 2** – SYS.2 Drafting
   * **Agent 3** – SYS.2 Review and Compliance
   * **Agent 4** – SYS.5 Test Case Generation
   * **Agent 5** – Manager Agent for orchestration
3. **Orchestration**: CrewAI serves as the orchestration layer, ensuring sequencing, dependency management, and conflict resolution across agents.
4. **Extensibility**: The design anticipates future inclusion of additional agents for SWE.1, SWE.5, and SWE.6 phases in line with Automotive SPICE practices.
5. **Technology Stack**
6. **OpenAI API**: Lightweight, efficient LLMs for NLP tasks such as requirement parsing, drafting, and test-case generation.
7. **LangChain**: Framework for Retrieval-Augmented Generation (RAG), workflow chaining, and integration with enterprise databases and standards.
8. **Streamlit/Gradio**: Web-based platforms for rapid prototyping of dashboards and user interfaces.
9. **Enterprise Integrations**: APIs enable connectivity with Jira, DOORS, Polarion, Outlook, Slack, and MS Teams for seamless workflow support.
10. **Standards and Compliance**
11. **ISO/IEC/IEEE 29148:2018** – Requirements engineering best practices.
12. **ISO 26262** – Functional safety standards for automotive applications.
13. **IREB** – Requirement’s quality framework (clarity, completeness, unambiguity, consistency, and verifiability).
14. **INCOSE SE Handbook** – Systems engineering principles and verification/validation processes.
15. **Functional Design**
16. **Requirement Lifecycle Support**: End-to-end flow from SYS.1 (intake) → SYS.2 (drafting/review) → SYS.5 (test-case generation).
17. **Traceability**: Full mapping between requirements and test cases, including redundancy detection through a similarity index.
18. **Validation Metrics**: Precision, recall, compliance scores, ambiguity detection accuracy, coverage percentage, and orchestration efficiency.
19. **Collaboration Features**: Role-Based Access Control (RBAC), real-time multi-user review, version control, and inline reviewer suggestions.
20. **Performance Considerations**
21. **High-Performance Processing**: Groq API ensures low-latency inference for real-time requirement engineering tasks.
22. **Scalability**: System supports multi-format file ingestion (PDF, DOCX, XLSX, CSV, transcripts, and emails).
23. **Learning Feedback Loops**: Agents adapt based on reviewer feedback to continuously improve drafting and review quality.
24. **Usability**
25. **Interactive Dashboards**: Visual reports for requirement quality, test coverage, and redundancy metrics.
26. **Multilingual Support**: Enables extraction, drafting, and review in multiple languages.
27. **Customization**: Configurable templates, domain-specific rule engines, and release-planning features tailored to enterprise needs.

# Plan of Work

|  |  |  |  |
| --- | --- | --- | --- |
| Phases | Start Date – End Date | Work to be Done | Status |
| Project Details Submission | 20 May 2025 – 06 July 2025 | Submission of student project details on Elearn Taxila portal | COMPLETED |
| Dissertation Outline | 12 July 2025 – 20 July 2025 | Literature Review and preparation of Dissertation Outline | COMPLETED |
| Outline/Abstract Submission | 26 July 2025 – 30 July 2025 | Upload Outline/Abstract for review | COMPLETED |
| Abstract VIVA & Grading | 02 Aug 2025 – 22 Aug 2025 | VIVA date selection and Grading Evaluation | COMPLETED |
| Design, Development, & Testing (Agent 1 & 2) | 23 Aug 2025 – 12 Sep 2025 | Design & development activities followed by Software testing, user evaluation, feedback incorporation **(Agent- 1, 2, & 5)** | COMPLETED |
| Mid-Semester Report | 13 Sep 2025 – 17 Sep 2025 | Submission of Mid-Sem Report, Plagiarism Check | COMPLETED |
| Mid-Sem VIVA & Grading | 22 Sep 2025 – 15 Oct 2025 | Mid-sem VIVA selection, evaluation, grading | COMPLETED |
| Design, Development, & Testing (Agent 3 & 4) | 16 Oct 2025 – 31 Oct 2025 | Design & development activities followed by Software testing, user evaluation, feedback incorporation **(Agent- 1, 2, 3, 4, & 5)** | COMPLETED |
| Final Plagiarism Check | 15 Oct 2025 – 07 Nov 2025 | Plagiarism checking window | COMPLETED |
| Dissertation Review & Upload | 07 Nov 2025 – 11 Nov 2025 | Submit soft copy of final dissertation and evaluation by mentor/supervisor | COMPLETED |
| Final VIVA | 12 Nov 2025 – 14 Nov 2025 | Final VIVA Date selection | PENDING |
| Final VIVA Evaluation | 19 Nov 2025 – 05 Dec 2025 | Grading evaluation after final VIVA | PENDING |

# Conclusion

1. **Successful Demonstration of AI-Driven Automation**  
   The dissertation establishes that a multi-agent system powered by Large Language Models (LLMs) can significantly automate the requirement engineering lifecycle - from SYS.1 elicitation to SYS.5 test-case generation - thereby reducing human effort, turnaround time, and error probability in complex engineering projects.
2. **End-to-End Requirement Engineering Pipeline Validated**  
   The five-agent architecture demonstrated coherent orchestration across requirement intake, drafting, compliance review, and test-case generation. This validates the feasibility of modular AI agents replacing traditionally fragmented manual workflows in safety-critical domains.
3. **Improved Requirement Quality and Traceability**  
   Automated enforcement of standards such as ISO/IEC/IEEE 29148, IREB, and ISO 26262 helped reduce ambiguity, improve clarity, and maintain traceability between requirements and test cases. This marks a major step toward digital engineering and regulatory compliance readiness.
4. **Measured Quantitative Gains**  
   Preliminary results indicate >50% reduction in manual effort for requirement intake, ~40% faster requirement drafting, and ~70% automated test-case generation coverage. The manager agent further improved orchestration efficiency and conflict resolution, reinforcing the industrial applicability of the solution.
5. **Scalable Framework With Enterprise Potential**  
   The architecture was designed with extensibility, tool-integration capability (Jira, DOORS, Polarion), and RBAC-based governance, making it suitable for industrial environments that require both automation and compliance assurance.
6. **Human-in-the-Loop Remains Critical**  
   While the system automates high-volume and repetitive engineering tasks, governance activities - such as final requirement acceptance, safety classification, baseline creation, and compliance attestation - remain dependent on authorized engineering stakeholders.
7. **Key Challenges Identified**  
   Limitations include model inconsistency, dependency on input data quality, partial regulatory coverage without fine-tuning, and confidentiality risks in cloud-based LLM execution. These highlight the need for on-premise deployment and continuous model refinement.
8. **Foundation for Future Automotive SPICE Alignment**  
   The system already fulfills SYS.1, SYS.2, and SYS.5 intent and can be extended to SWE.1–SWE.6, making it a foundational asset for future AI-assisted ASPICE-compliant engineering workflows.

# Recommendations

1. **On-Premise or Private-Cloud Deployment for Safety-Critical Use**  
   To avoid data-privacy risks and meet OEM/Tier-1 confidentiality standards, future deployments should prioritize locally hosted or private-cloud inference setups instead of public LLM APIs.
2. **Fine-Tuning and Domain Adaptation of Models**  
   Performance can be improved by fine-tuning models on domain-specific datasets (e.g., AUTOSAR, ISO 26262 case studies), enabling better understanding of engineering terminology and safety clauses that general-purpose LLMs struggle with.
3. **Integration With Industry Tools (Jira, DOORS, Polarion)**  
   Direct API-based integration will eliminate manual export/import of deliverables, enable live synchronization, and increase adoption in enterprise engineering workflows.
4. **Expansion Toward Automotive SPICE Workflows**  
   The next phase should include agents for SWE.1 (requirement specification), SWE.5 (integration test), and SWE.6 (qualification test), enabling complete end-to-end automation in ASPICE-governed projects.
5. **Formal Tool Qualification Strategy Under ISO 26262**  
   A structured qualification plan—covering Tool Confidence Level (TCL) assessment, golden-data validation, and residual-risk documentation—should be developed to enable use in safety-relevant projects.
6. **Feedback-Driven Continuous Learning Pipeline**  
   Establishing a reinforcement or active-learning feedback loop from reviewer corrections will significantly improve the accuracy and consistency of all agents over time.
7. **Multimodal Support for Real-World Inputs**  
   Future releases should add OCR-enabled extraction from scanned documents, handwritten notes, and engineering diagrams to fully replicate industrial requirement intake conditions.
8. **User-Role and Change-Control Governance Enhancements**  
   RBAC should be strengthened with audit trail validation, change history lock-in, baseline comparison, and digital sign-off features to align with formal requirement review boards.
9. **Multilingual Requirement Engineering Support**  
   Since global OEMs operate in multi-lingual environments, multilingual extraction, translation, and review support will widen the usability of the framework.

# Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| AI | Artificial Intelligence |
| NLP | Natural Language Processing |
| LLM | Large Language Model |
| RAG | Retrieval-Augmented Generation |
| UI | User Interface |
| IREB | International Requirements Engineering Board |
| ISO | International Organization for Standardization |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| AUTOSAR | Automotive Open System Architecture |
| API | Application Programming Interface |
| RBAC | Role-Based Access Control |
| KPI | Key Performance Indicator |
| GDPR | General Data Protection Regulation |
| CI/CD | Continuous Integration / Continuous Deployment |
| SWE | Software Engineering (Automotive SPICE Process Area) |
| DOORS | Dynamic Object-Oriented Requirements System (IBM Tool) |
| ALM | Application Lifecycle Management |
| SE | Systems Engineering |
| STT | Speech-to-Text |

# Literature References

The **state of the art** serves as the foundation for any successful research project. In the current work, the literature primarily focuses on advancements in Conversational Information Retrieval, Natural Language Processing (NLP), and LLM-powered Requirement Engineering Systems. These areas form the backbone of intelligent multi-agent systems capable of automating requirement intake, system drafting, validation, and test-case generation. The following peer-reviewed journal articles and conference papers were identified during the **preliminary literature review**, providing relevant insights for developing the proposed modular agents (SYS.1 to SYS.5):

1. Hrvoje Belani, Marin Vuković & Željka Car (2019). Requirements Engineering Challenges in Building AI-Based Complex Systems. arXiv preprint. <https://arxiv.org/pdf/1908.11791>
2. Khaled Slhoub et al. (2017). Recommended Practices for the Specification of Multi-Agent Systems Requirements. IEEE UEMCON Conference. <https://www.researchgate.net/publication/320195567_Recommended_Practices_for_the_Specification_of_Multi-Agent_Systems_Requirements>
3. Dongming Jin, Zhi Jin, Xiaohong Chen & Chunhui Wang (2024). MARE: Multi-Agents Collaboration Framework for Requirements Engineering. arXiv preprint.  
   <https://arxiv.org/pdf/2405.03256>
4. A. Mäder and H. Lichter. Requirements Formalization Using NLP and ML: A Systematic Review. In CEUR Workshop Proceedings, Vol. 2951, pages 101–108. CEUR-WS, 2021. <http://ceur-ws.org/Vol-2951/paper15.pdf>
5. Y. Li and P. Chen. Software Test Case Generation Using NLP: A Systematic Literature Review. Artificial Intelligence Evolution, 5(1):22–31, 2024. <https://doi.org/10.37256/aie.5120243220>
6. T. Goel, S. Shukla, and A. Aggarwal. Requirements-Based Test Generation: A Comprehensive Survey. arXiv preprint arXiv:2505.02015, 2025. <https://arxiv.org/abs/2505.02015>
7. D. Singh and R. Kumar. Natural Language Processing for Automated Test Case Generation from Software Requirement Documents. In Proceedings of the International Conference on Computing and Communication Technologies, pages 112–118. Springer, 2019. <https://www.researchgate.net/publication/389031328>
8. Y. Cheng and L. Zhang. Generating Test Scenarios from Natural Language Requirements Using GPT Models. arXiv preprint arXiv:2404.12772, 2024. <https://arxiv.org/pdf/2404.12772>
9. H. Zeni, E. G. Knauss, and D. Méndez. Requirement Formalisation Using NLP and ML: A Systematic Review. arXiv preprint arXiv:2303.13365, 2023. <https://arxiv.org/pdf/2303.13365>
10. T. Femmer, J. Eckhardt, and D. Méndez Fernández. Classification of NLP Techniques for Requirements Engineering Tasks: A Meta-Study. arXiv preprint arXiv:2204.04282, 2022. <https://arxiv.org/pdf/2204.04282>
11. Alessio Ferrari, Liping Zhao & Waad Alhoshan (2021). NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies. IEEE/ACM ICSE Companion Proceedings. <https://www.researchgate.net/publication/351422341_NLP_for_Requirements_Engineering_Tasks_Techniques_Tools_and_Technologies>
12. Johannes Norheim et al. (2024). Challenges in Applying Large Language Models to Requirements Engineering. Design Science Journal (Open Access). <https://publications.rwth-aachen.de/record/994149/files/994149.pdf>

# Checklist of Items for the Final Dissertation / Project / Project Work Report

|  |  |  |
| --- | --- | --- |
|  | **Is the final report neatly formatted with all the elements required for a technical Report?** | **Yes** |
|  | Is the Cover page in proper format as given in Annexure A? | **Yes** |
|  | Is the Title page (Inner cover page) in proper format? | **Yes** |
|  | (a) Is the Certificate from the Supervisor in proper format?  (b) Has it been signed by the Supervisor? | **Yes**  **Yes** |
|  | Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly? | **Yes**  **Yes** |
|  | Is the title of your report appropriate? **The title should be adequately descriptive, precise and must reflect scope of the actual work done.** Uncommon abbreviations / Acronyms should not be used in the title | **Yes** |
|  | Have you included the List of abbreviations / Acronyms? | **Yes** |
|  | Does the Report contain a summary of the literature survey? | Yes |
|  | Does the Table of Contents include page numbers?   1. Are the Pages numbered properly? (Ch. 1 should start on Page # 1) 2. Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) 3. Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) 4. Are the Captions for the Figures and Tables proper? 5. Are the Appendices numbered properly? Are their titles appropriate | **Yes**  **Yes**  **Yes**  **Yes**  **Yes**  **Yes** |
|  | Is the conclusion of the Report based on discussion of the work? | **Yes** |
|  | Are References or Bibliography given at the end of the Report?  Have the References been cited properly inside the text of the Report?  Are all the references cited in the body of the report | **Yes**  **Yes**  **Yes** |
|  | Is the report format and content according to the guidelines? The report should not be a mere printout of a PowerPoint Presentation, or a user manual. Source code of software need not be included in the report. | **Yes** |

**Declaration by Student:**

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

|  |  |
| --- | --- |
| **Place: Pune, MH, India** | **Signature of the Student** |
| **Date: 06/11/2025** | **Name: RANJIT RAJENDRA JAGTAP**  **ID No.: 2023HT65600** |

**THANK YOU**