SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

Task 1.1 – Understanding Software Requirements Specification Standards (IEEE 830.9)

Objective:

To gain a clear understanding of how to define and structure software requirements for complex systems, particularly by studying the IEEE 830.9 standard and applying its principles to a real-world, safety-critical system such as autonomous vehicle navigation and decision-making.

What Is IEEE 830.9?

IEEE 830.9 is a guideline that helps developers and engineers write clear, complete, and organized Software Requirements Specification (SRS) documents. These documents serve as a blueprint for what the software should do, how it should behave, and how it will interact with other systems or users.

This standard is especially important in safety-critical environments—like those used in autonomous vehicles—where even a small oversight or vague requirement can lead to serious consequences.

Key Concepts from IEEE 830.9:

Here are the main things I've learned from reviewing this standard:

1. **Clarity is Crucial**
   Every requirement must be written in a way that's easy to understand and free from ambiguity. Vague words like "quick," "easy," or "user-friendly" should be avoided unless they're clearly defined.
2. **Structure and Organization**
   IEEE 830.9 recommends a clear structure for organizing the SRS, typically including:
   - Introduction
   - Overall description of the system
   - Functional requirements
   - Non-functional requirements
   - External interfaces

- Appendices and references

3. **Functional vs Non-Functional Requirements**
   Functional requirements describe what the system should do (e.g., "detect objects using a camera and LIDAR").
   Non-functional requirements describe how the system should perform (e.g., "respond to obstacles within 200 milliseconds").
4. **Verifiability and Testability**
   A good requirement should be measurable or testable. If it can't be tested, it's hard to confirm whether the software meets the requirement.
5. **Traceability**
   Each requirement should be traceable—from the original stakeholder need through to design, development, and testing. This ensures that nothing is left out or misunderstood during the development lifecycle.
6. **Modifiability and Consistency**
   Requirements should be written so that they are easy to update when needed, without causing confusion or inconsistencies elsewhere in the document.

Why This Matters in Our Project:

For a system that controls autonomous vehicle navigation and decision-making, following IEEE 830.9 is not just a recommendation—it's a necessity. In such safety-critical environments:

- Misunderstood requirements can lead to incorrect software behavior.
- Missing details can cause failure to detect or respond to obstacles.
- Poor documentation can make it hard to validate system performance against regulations.

By using the IEEE 830.9 standard, we can ensure our requirements are:

- Well-organized
- Easy to understand
- Free from contradictions
- Aligned with industry safety standards

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

☑ Summary (What I've Learned):

Reading IEEE 830.9 helped me clearly understand how a Software Requirements Specification should be written for complex systems. The standard emphasizes clarity, structure, completeness, and traceability—all of which are essential when developing a system where human lives may depend on the outcome.

This foundation is now ready to be applied in the next steps of the project, where I'll build an SRS template, document stakeholder needs, and manage the requirements using proper verification and validation processes.

Task 1.2 – Creating a Software Requirements Specification (SRS) Template

Objective:

To develop a clear and well-structured SRS template that supports the development of a safety-critical autonomous vehicle control system. The focus is on organizing functional and non-functional requirements in a way that supports validation, traceability, and quality assurance throughout the development lifecycle.

Background:

Since this project involves a high-risk application—autonomous vehicle navigation and decision-making—documenting the requirements correctly is extremely important. The SRS serves as the foundation for design, development, testing, and validation. To ensure completeness and accuracy, the SRS structure is designed following industry best practices and the IEEE 830 standard.

The V-Model chosen for this project emphasizes early definition of requirements, along with continuous verification and validation (V&V) at each phase. Therefore, this SRS template must be detailed, well-organized, and support quality targets such as safety, reliability, performance, and regulatory compliance.

Software Requirements Specification (SRS) Template:

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	**USN: 1nt23is013**	**Date:25/4/25**
Lab Activity # / Task #: Activity 4	**Document name:Software Requirement Engineering**	**Submitted details:Completed**

1. Introduction

- **1.1 Purpose**
  Defines the main goal of the software: to support autonomous driving by enabling safe navigation and decision-making in real-world environments.
- **1.2 Scope**
  Describes the key features, the overall system role, and what the system will and will not do. This section sets boundaries for the current phase of development.
- **1.3 Definitions and Abbreviations**
  Lists technical terms, acronyms, and system-specific vocabulary to avoid confusion.
- **1.4 References**
  Provides a list of supporting documents, standards (like ISO 26262, IEEE 830), research materials, and user manuals relevant to the project.
- **1.5 Overview**
  Gives a brief summary of the document structure and what each section contains.

---

2. Overall Description

- **2.1 Product Perspective**
  Describes how the software interacts with other subsystems in the vehicle, such as cameras, sensors, and actuators.
- **2.2 Product Functions**
  Summarizes major functions the software will perform, including obstacle detection, route planning, and behavior decision-making.
- **2.3 User Characteristics**
  Identifies the types of users (e.g., system engineers, QA teams) and their technical backgrounds.
- **2.4 Operating Environment**
  Specifies the hardware and software environment in which the system will run—real-time OS, embedded systems, vehicle-mounted sensors, etc.
- **2.5 Constraints**
  Lists any limitations or restrictions—processing time, memory usage, safety constraints, or hardware dependencies.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

- **2.6 Assumptions and Dependencies**
  Notes any assumptions (e.g., GPS signal availability) and external components the software depends on.

---

3. Specific Requirements

This section is organized by requirement types and includes unique identifiers for tracking.

- **3.1 Functional Requirements**
  - FR01: The system shall identify objects within a 50-meter radius using LIDAR.
  - FR02: The vehicle shall automatically adjust speed based on detected obstacles.
  - FR03: The system shall execute a safe emergency stop when a collision risk is detected within a 3-meter range.
- **3.2 Non-Functional Requirements**
  - NFR01: The system shall maintain an average response time under 100 milliseconds.
  - NFR02: The software shall meet ISO 26262 ASIL D safety requirements.
  - NFR03: System availability shall be at least 99.9% during operation.
- **3.3 External Interface Requirements**
  - Describes the data inputs and outputs for sensors (camera, LIDAR, radar), GPS modules, and the vehicle control interface.
- **3.4 System Features**
  Lists major user-facing features, described in clear, testable terms (e.g., "Lane change assistance shall activate when the adjacent lane is clear and a route change is needed").

---

4. Verification & Validation ChecTask 1.2 – Creating a Software Requirements Specification (SRS) Template

Objective:

To develop a clear and well-structured SRS template that supports the development of a safety-critical autonomous vehicle control system. The focus is on organizing functional and non-functional requirements in a way that supports validation, traceability, and quality assurance throughout the development lifecycle.

Background:

Since this project involves a high-risk application—autonomous vehicle navigation and decision-making—documenting the requirements correctly is extremely important. The SRS serves as the foundation for design, development, testing, and validation. To ensure completeness and accuracy, the SRS structure is designed following industry best practices and the IEEE 830 standard.

The V-Model chosen for this project emphasizes early definition of requirements, along with continuous verification and validation (V&V) at each phase. Therefore, this SRS template must be detailed, well-organized, and support quality targets such as safety, reliability, performance, and regulatory compliance.

Software Requirements Specification (SRS) Template:

---

1. Introduction

- **1.1 Purpose**
  Defines the main goal of the software: to support autonomous driving by enabling safe navigation and decision-making in real-world environments.
- **1.2 Scope**
  Describes the key features, the overall system role, and what the system will and will not do. This section sets boundaries for the current phase of development.
- **1.3 Definitions and Abbreviations**
  Lists technical terms, acronyms, and system-specific vocabulary to avoid confusion.
- **1.4 References**
  Provides a list of supporting documents, standards (like ISO 26262, IEEE 830), research materials, and user manuals relevant to the project.
- **1.5 Overview**
  Gives a brief summary of the document structure and what each section contains.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

2. Overall Description

- **2.1 Product Perspective**
  Describes how the software interacts with other subsystems in the vehicle, such as cameras, sensors, and actuators.
- **2.2 Product Functions**
  Summarizes major functions the software will perform, including obstacle detection, route planning, and behavior decision-making.
- **2.3 User Characteristics**
  Identifies the types of users (e.g., system engineers, QA teams) and their technical backgrounds.
- **2.4 Operating Environment**
  Specifies the hardware and software environment in which the system will run—real-time OS, embedded systems, vehicle-mounted sensors, etc.
- **2.5 Constraints**
  Lists any limitations or restrictions—processing time, memory usage, safety constraints, or hardware dependencies.
- **2.6 Assumptions and Dependencies**
  Notes any assumptions (e.g., GPS signal availability) and external components the software depends on.

---

3. Specific Requirements

This section is organized by requirement types and includes unique identifiers for tracking.

- **3.1 Functional Requirements**
  - FR01: The system shall identify objects within a 50-meter radius using LIDAR.
  - FR02: The vehicle shall automatically adjust speed based on detected obstacles.
  - FR03: The system shall execute a safe emergency stop when a collision risk is detected within a 3-meter range.
- **3.2 Non-Functional Requirements**
  - NFR01: The system shall maintain an average response time under 100 milliseconds.
  - NFR02: The software shall meet ISO 26262 ASIL D safety requirements.

- NFR03: System availability shall be at least 99.9% during operation.

- **3.3 External Interface Requirements**
  - Describes the data inputs and outputs for sensors (camera, LIDAR, radar), GPS modules, and the vehicle control interface.
- **3.4 System Features**
  Lists major user-facing features, described in clear, testable terms (e.g., "Lane change assistance shall activate when the adjacent lane is clear and a route change is needed").

---

4. Verification & Validation Checkpoints

Requirement ID	Verification Method	Validation Method
FR01	Unit test + Simulation	Test track scenario
NFR02	Static code analysis	Safety compliance audit
FR03	Real-world testing	Stakeholder review

This table links each requirement to the corresponding V&V method. This helps ensure traceability and makes it easier to confirm whether the system meets all specified goals.

---

5. Appendices and Supporting Information

- System architecture diagrams
- Glossary of terms
- Use case descriptions
- Compliance checklists

Why This Template Works:

This SRS template is built to suit the development of a safety-critical automotive system. It follows a logical structure and clearly separates different types of requirements. This not only helps development teams stay organized but also makes it easier to carry out testing, audits, and certification.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

The inclusion of V&V checkpoints and traceable requirement IDs aligns with the project's use of the V-Model and ensures full coverage of quality targets from the start.

Conclusion:

The SRS template prepared here is ready to be used for documenting the system requirements of the autonomous vehicle control software. It supports early identification of potential issues, smooth collaboration between teams, and successful verification and validation at every stage of development.

Stakeholder requirements are

well-analyzed and

documented effectively.

Task 1.3 – Stakeholder Requirements Analysis and Documented SRS Preparation

Objective:

To apply suitable methods to gather, analyze, and document stakeholder requirements in a way that ensures they are well understood, traceable, and aligned with the goals of a safety-critical autonomous vehicle navigation and decision-making system.

Understanding the Role of Stakeholder Requirements:

In any complex system, especially one that involves safety and automation, understanding what stakeholders expect from the software is the foundation of successful development. In this project, stakeholders include:

- Automotive engineers
- Regulatory bodies
- End users (drivers/passengers)
- Safety auditors

- Embedded software developers
- Project managers

Each of these stakeholders has a different perspective on the system's behavior, safety, usability, and legal compliance. Capturing and aligning these expectations is essential.

🔍 Methods Used for Requirements Analysis:

To ensure a thorough and accurate understanding of stakeholder needs, the following methods were applied:

---

1. Stakeholder Interviews

One-on-one interviews were conducted with key stakeholders to understand their expectations. Topics included:

- Expected driving scenarios (urban, highway, etc.)
- Safety priorities (e.g., collision avoidance, emergency stop behavior)
- Regulatory expectations (e.g., ISO 26262 compliance)
- System limitations and assumptions (e.g., GPS signal availability)

> **Example Insight**: One stakeholder emphasized the need for the vehicle to detect pedestrians even in low-light conditions, which directly influenced functional requirements related to camera sensitivity and night-time operation.

---

2. Use Case Analysis

Several typical driving scenarios were modeled as use cases. This helped break down abstract needs into specific interactions between the vehicle and its environment.

> **Example Use Case**:
> *"Vehicle approaching a four-way stop sign"* – The system must detect the stop sign, come to a halt, identify cross-traffic, and proceed when safe.

These use cases were documented in the SRS and later translated into functional and non-functional requirements.

---

3. Requirement Workshops

Collaborative sessions were held with technical teams and safety engineers. These workshops helped clarify technical constraints, real-time data processing limits, and sensor behavior.

> **Result**: Several performance-related requirements were reworded based on developer feedback to ensure feasibility and testability.

---

4. Review of Industry Standards and Guidelines

Standards like ISO 26262, IEEE 830, and FMVSS were reviewed to align stakeholder needs with compliance requirements. This added another layer of depth to the requirements, particularly in the areas of:

- Safety integrity levels (ASIL)
- Response times for obstacle detection
- Redundancy and fault tolerance

📄 Key Documented Requirements (from Stakeholders):

Below are examples of how stakeholder expectations were translated into documented requirements in the SRS:

Functional Requirements:

ID	Requirement
FR-01	The system shall detect pedestrians at a minimum distance of 25 meters, day or night.
FR-02	The system shall generate a new navigation path within 500ms of obstacle detection.
FR-03	The system shall bring the vehicle to a complete stop when a collision risk is detected within 3 meters.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

Non-Functional Requirements:

ID	Requirement
NFR-01	The system shall operate in real time with an average latency of under 100ms.
NFR-02	The software shall maintain 99.9% availability during vehicle operation.
NFR-03	The system shall comply with ISO 26262 ASIL D safety standards.

Traceability and Documentation:

Each requirement in the SRS has been assigned a unique ID to support traceability throughout the project lifecycle. A requirements traceability matrix (RTM) is used to link stakeholder needs to design decisions, implementation details, and testing checkpoints.

Version control (Git) is used to manage the evolving SRS document, ensuring that every change is tracked and reviewed by relevant stakeholders.

✅ Outcome:

The combination of interviews, use cases, workshops, and standards review provided a comprehensive understanding of stakeholder expectations. These have been carefully documented in the SRS in a clear, structured format that supports development, testing, and compliance. The result is a well-aligned set of requirements that reflects real-world needs while also being testable and verifiable.

Task 1.4 – Perform Verification of the Requirements and Manage the Report Using Git

Expected Outcome: Verification of software requirements for quality attributes such as completeness, correctness, accuracy, and unambiguity.

Verification Process:

To ensure high-quality requirements, each was verified against standard criteria:

- **Completeness**: Checked whether all necessary functionality, interfaces, and constraints were specified.
- **Correctness**: Confirmed alignment with stakeholder expectations and regulatory standards.
- **Unambiguity**: Ensured all requirements were written clearly with no room for multiple interpretations.
- **Consistency**: Verified that there were no conflicting requirements.
- **Verifiability**: Confirmed that each requirement could be tested or measured objectively.

Verification Techniques Used:

1. **Peer Review**: Requirements were reviewed by team members from different roles (developers, testers, and QA engineers) to catch inconsistencies and unclear language.
2. **Requirement Walkthroughs**: Organized walkthrough sessions to identify missing or conflicting requirements.
3. **Checklist-Based Review**: Used a verification checklist based on IEEE 830 to evaluate each requirement.
4. **Traceability Review**: Cross-referenced stakeholder inputs, use cases, and safety standards with each requirement to confirm alignment.

Version Control Using Git:

- The verified SRS document was uploaded to a Git repository.
- All revisions were committed with descriptive messages (e.g., "Updated FR-03 based on peer review feedback").
- Branches were used to isolate major edits or new feature requirement sets.
- Reviewers submitted comments via Git pull requests, ensuring all changes were approved before merging.

Task 1.5 – Perform Validation of the Requirements with Relevant Stakeholders

Expected Outcome: Documented steps for stakeholder and customer validation.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	USN: 1nt23is013	Date:25/4/25
Lab Activity # / Task #: Activity 4	Document name:Software Requirement Engineering	Submitted details:Completed

Steps Taken for Validation:

1. **Requirement Presentation**:
    - Conducted meetings with key stakeholders (e.g., product managers, safety officers, end users).
    - Presented each functional and non-functional requirement for confirmation and discussion.
2. **Use Case Demonstration**:
    - Demonstrated real-world use cases through flowcharts and scenario walkthroughs.
    - Verified that the documented requirements matched stakeholders' intended use.
3. **Feedback Collection**:
    - Collected feedback through structured questionnaires and open discussion.
    - Logged feedback and documented the decisions made (accepted, revised, or deferred requirements).
4. **Consensus and Sign-Off**:
    - Updated the SRS based on feedback and shared the revised version.
    - Stakeholders formally signed off the validated requirements, confirming their acceptance.
5. **Validation Report**:
    - Prepared a summary report outlining the validation process, attendees, feedback points, and changes made.
    - Stored this report in the project repository alongside the SRS document.

Task 1.6 – Maintain the SRS Document in Version Control Using Git

Expected Outcome: A version-controlled SRS document ensuring traceability and proper change management.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	**USN: 1nt23is013**	**Date:25/4/25**
Lab Activity # / Task #: Activity 4	**Document name:Software Requirement Engineering**	**Submitted details:Completed**

Steps for Version Control and Maintenance:

1. **Initial Repository Setup**:
    - Created a Git repository specifically for SRS documentation.
    - Initialized with the base version of the SRS (v1.0).
2. **Branch Management**:
    - Created separate branches for major updates (e.g., `safety-updates`, `user-feedback`, `regulatory-compliance`).
    - Merged branches only after approval via pull requests and peer reviews.
3. **Commit Practices**:
    - Each change to the SRS was committed with a detailed message, including:
        - What was changed
        - Why the change was made
        - Link to the feedback or issue being resolved
4. **Change Log Maintenance**:
    - Maintained a `CHANGELOG.md` file in the repository to track:
        - Version number
        - Date of update
        - Summary of key changes
5. **Tagging and Releases**:
    - Tagged significant versions (e.g., `v1.0`, `v1.1`, `v2.0`) for milestone tracking.
    - Ensured each version reflected complete and validated requirements.
6. **Audit Readiness**:
    - Repository structure and documentation are kept audit-ready, with clear traceability from stakeholder input to finalized requirements.

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:Aditya Singh	**USN: 1nt23is013**	**Date:25/4/25**
Lab Activity # / Task #: Activity 4	**Document name:Software Requirement Engineering**	**Submitted details:Completed**