Department of Computer Engineering

Academic Term: First Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**1**
Title:	**Software Requirement Specification**
Date of Performance:	**27/07/2023**
Roll No:	**9611**
Team Members:	

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

Lab Experiment 01

Experiment Name: Software Requirement Specification (SRS) as per IEEE Format

Objective: The objective of this lab experiment is to guide students in creating a Software Requirement Specification (SRS) document following the IEEE (Institute of Electrical and Electronics Engineers) standard format. The IEEE format ensures a structured and consistent approach to capturing software requirements, facilitating effective communication among stakeholders and streamlining the software development process.

Introduction: Software Requirement Specification (SRS) is a formal document that precisely defines the functional and non-functional requirements of a software project. The IEEE standard format provides a systematic framework for organizing the SRS, making it comprehensive, clear, and easily understandable by all parties involved in the project.

Lab Experiment Overview:

1. Introduction to IEEE Standard: The lab session begins with an overview of the IEEE standard format for SRS. Students are introduced to the various sections and components of the SRS as per the standard.
2. Selecting a Sample Project: Students are provided with a sample software project or case study for which they will create the SRS. The project should be of moderate complexity to cover essential elements of the IEEE format.
3. Requirement Elicitation and Analysis: Students conduct requirement elicitation sessions with the project stakeholders to gather relevant information. They analyze the collected requirements to ensure they are complete, unambiguous, and feasible.
4. Structuring the SRS: Using the IEEE standard guidelines, students organize the SRS document into sections such as Introduction, Overall Description, Specific Requirements, Appendices, and other relevant subsections.
5. Writing the SRS Document: In this phase, students write the SRS document, ensuring it is well-structured, coherent, and adheres to the IEEE format. They include necessary diagrams, use cases, and requirements descriptions.
6. Peer Review and Feedback: Students exchange their SRS documents with their peers for review and feedback. This review session allows them to receive constructive criticism and suggestions for improvement.
7. Finalization and Submission: After incorporating the feedback received during the review session, students finalize the SRS document and submit it for assessment.

Learning Outcomes: By the end of this lab experiment, students are expected to:

- Understand the IEEE standard format for creating an SRS document.
- Develop proficiency in requirement elicitation, analysis, and documentation techniques.
- Acquire the skills to structure an SRS document following the IEEE guidelines.

- Demonstrate the ability to use diagrams, use cases, and textual descriptions to define software requirements.
- Enhance communication and collaboration skills through peer reviews and feedback sessions.

Pre-Lab Preparations: Before the lab session, students should review the IEEE standard for SRS documentation, familiarize themselves with the various sections and guidelines, and understand the importance of clear and unambiguous requirements.

Materials and Resources:

- IEEE standard for SRS documentation
- Sample software project or case study for creating the SRS
- Computers with word processing software for document preparation
- Review feedback forms for peer assessment

Conclusion: The Software Requirement Specification (SRS) lab experiment in accordance with the IEEE standard format equips students with essential skills in documenting software requirements systematically. Following the IEEE guidelines ensures that the SRS document is well-organized, comprehensive, and aligned with industry standards, facilitating seamless communication between stakeholders and software developers. Through practical hands-on experience in creating an SRS as per the IEEE format, students gain a deeper understanding of the significance of precise requirement definition in the success of software projects. Mastering the IEEE standard for SRS documents prepares students to be effective software engineers, capable of delivering high-quality software solutions that meet client expectations and industry best practices.

1. Abstract

Gas leaks can be fatal and harmful, whether they occur in open or closed
spaces. Despite their high level of precision, conventional gas leak detection systems overlook a few important aspects in warning the public leak. As a result, we created a Gas Leakage Detector with IoT technology that uses Smart Alerting techniques to call, text, and email the relevant authorities as well as the ability to foresee hazardous situations so that people can be made aware in advance by performing data analytics on sensor readings.

2. Introduction

2.1 Purpose

Our proposed system aims to present a safe, reliable and easy to use device, detecting leakages and alerting me about it. There is a huge usage of LPG and CNG in day to day lives and gas leaks are a serious issue which needs to be taken care of.

2.2 Scope

Gas leakage detection systems are an integral part of a safety system providing the first line of defense against the possible disasters of gas leakage. In India, most of the households are run by the manual gas stove which is powered by liquidized petroleum gas (LPG). From cooking till heating water LPG is vastly used in the household. The usage of the
gas brings great problems in the domestic as well as working places. The inflammable gas such as liquidized petroleum gas (LPG), which is excessively used in the house and at work places. In addition, many regulatory agencies require these systems in certain industries to ensure
compliance with safety regulations.

2.3 Definitions, Acronyms, Abbreviations

Not applicable.

2.4 References

Not applicable.

2.5 Developer's Responsibilities

The developer is responsible for (a) developing the system, (b) conducting any user training that might be needed for using the system, and (c) maintaining the system for a period of one year after installation.

3. General Description

3.1 Product Functions Overview

In this study, LPG gas is detected using semiconductor sensors. It makes use of a MQ-4 semiconductor sensor. The MQ-4 gas sensor sensitive
the component is SnO2, which has a reduced conductivity in clean air. The sensor conductivity rises along with the rising gas concentration when the target flammable gas is present. The MQ6 gas sensor responds to
natural gas and has great sensitivity to propane, butane,
and LPG. The sensor is versatile and inexpensive, and it can be used to detect a variety of flammable gasses, particularly methane. The range of gas concentrations the MQ-4 can
detect is 200 to 10,000 ppm. The output of the sensor is an analogue resistance. The Arduino UNO and MQ-4 gas sensor serve as the
foundation of this system. The sensor will provide a digital output of 1 when it detects gas in the atmosphere and a digital output of 0 if no gas is detected. The sensor output will be used as digital input by Arduino. The
the buzzer will start to tune and the LCD will display "Gas detected: Yes" if the sensor output is high. The bell won't tune and the LCD will display "Gas
detected: No" if the sensor output is low. The buzzer typically consists of a number of switches or sensors that are connected to a control unit that
can identify which button was pressed or whether a predetermined amount of time has passed. The buzzer also typically illuminates a light on the appreciative button or control panel and emits a warning sound in the form of a continuous or sporadic buzzing or beeping sound.

3.2 User Characteristics

The gadget has multi-functional qualities and is portable, lightweight, user-friendly, and effective.

3.3 General Constraints

The placement and installation of the MQ4 sensor can be critical to its effectiveness. It needs to be located in an area where gas leaks are likely to occur, and the installation needs to be done correctly to ensure accurate readings.

3.4 General Assumptions and Dependencies

Not applicable.

4. Specific Requirements

4.1 Inputs and Outputs

Cross-sensitivity: The MQ4 sensor can detect several different gasses, which means it can be cross sensitive to other gasses that may be present in the environment. This can lead to false readings or inaccurate
readings.
Temperature and humidity effects: MQ4 sensors can be affected by temperature and humidity changes. High temperatures can cause the sensor to drift, resulting in inaccurate readings.

Calibration: Like any gas detector, the MQ4 sensor requires periodic calibration to ensure it is accurately detecting gas concentrations. The sensor's sensitivity may change over time, so calibration is necessary to maintain accuracy.
Limited detection range: The MQ4 sensor has a limited detection range, and may not detect gas concentrations below a certain threshold. This means that it may not detect low levels of gas that could be harmful if left
Undetected.

4.2 Functional Requirements

Maintenance: Gas leakage detectors require regular maintenance and calibration to ensure that they are working properly. Failure to perform these tasks can lead to false alarms or failure to

detect a leak.

4.3 External Interface Requirements
Only installation required.

4.4 Performance Constraints
Limited detection range: The MQ4 sensor has a limited detection range, and may not detect gas concentrations below a certain threshold. This means that it may not detect low levels of gas that could be harmful if left
Undetected.
Limited lifespan: Like any sensor, the MQ4 has a limited lifespan and may
need to be replaced after a certain amount of time. The lifespan can be affected by factors such as temperature, humidity, and exposure to other gasses.

Warm-up time: The MQ4 sensor requires a warm-up time before it can begin detecting gas. This warm-up time can be several minutes, which can be a limitation in situations where a fast response time is necessary.
It is essential to consider these limitations and take appropriate measures to ensure that the gas detectors function correctly and effectively. Regular maintenance, testing, and calibration are necessary to ensure that the detectors are working correctly. Additionally, proper installation and positioning of the detectors can help minimize false alarms
and maximize the range of detection.

4.5 Design Constraints
Not applicable

4.6 Hardware constraints
Not applicable (fixed hardware)

4.7 Acceptance criteria
Not applicable

Postlab Questions:

a) Evaluate the importance of a well defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success.

Some key reasons why a well-defined SRS is important and how it impacts project success are:
1. Clear and Shared Understanding: The SRS document outlines the project's objectives, features, functionalities, and constraints in a structured manner. It ensures that all stakeholders have a clear and shared understanding of what needs to be built, which helps avoid misunderstandings and discrepancies throughout the development process.

2. Scope Management: A well-defined SRS helps in defining the project's scope accurately. It outlines the in-scope and out-of-scope functionalities, which assists in preventing scope creep (uncontrolled expansion of project scope) and helps manage changes efficiently.

3. Requirement Validation: The SRS document allows stakeholders to review and validate the requirements early in the project's lifecycle. This validation process helps identify potential issues and ambiguities, reducing the risk of costly changes or rework later on.

4. Basis for Development: Developers rely on the SRS as a reference to design, implement, and test the software. A well-documented SRS provides developers with the necessary details, reducing the chances of misinterpretation and ensuring that the product aligns with the client's expectations.

5. Project Planning and Estimation: The SRS serves as the basis for project planning and estimation. It helps project managers determine the required resources, timeline, and budget for successful project execution.

6. Risk Mitigation: By identifying and documenting requirements clearly, the SRS helps in risk assessment and management.

7. Client Satisfaction: When the SRS accurately captures the client's needs and expectations, it enhances the likelihood of delivering a product that meets or exceeds those requirements. This, in turn, leads to higher client satisfaction and better chances of future business opportunities.

8. Traceability and Accountability: A well-structured SRS allows for easy traceability of requirements throughout the development process. This traceability aids in maintaining accountability, as each requirement can be tracked from conception to implementation.

9. Reduced Development Time and Cost: With a clear SRS in place, development teams can work more efficiently and avoid unnecessary rework or iterations, resulting in reduced development time and cost.

10. Legal and Contractual Compliance: In projects with formal contracts, the SRS serves as a legal document that defines the scope of work and ensures compliance with contractual obligations.

b) Analyse a given SRS document to identify any ambiguities or inconsistencies and propose improvements to enhance its clarity and completeness.

1. Ambiguous Language:
- Look for vague or unclear statements that could lead to different interpretations.
- Identify terms or phrases with multiple meanings or lack specific details.

Improvement:
- Replace ambiguous terms with specific and well-defined vocabulary.
- Provide clear and concise descriptions of requirements.

2. Inconsistent Information:
- Check for conflicting or contradictory requirements within the document.
- Look for discrepancies in terminology, measurements, or formatting.

Improvement:
- Cross-reference related sections or requirements to ensure consistency.
- Standardize terminology and units of measurement throughout the document.

3. Missing Information:
- Identify any gaps or incomplete requirements that lack necessary details.
- Look for omitted sections or aspects that should be addressed.

Improvement:
- Fill in missing information to provide a comprehensive view of the project.
- Include relevant context, assumptions, and dependencies to avoid ambiguity.

4. Ambiguous Use Cases or Scenarios:
- Review use cases or scenarios for unclear steps or undefined inputs/outputs.
- Check for inconsistent use case representations or missing alternative flows.

Improvement:
- Ensure each use case is well-defined with clear steps, preconditions, and post-conditions.
- Add alternative flows and exceptions to cover various scenarios comprehensively.

5. Unverifiable or Unrealistic Requirements:

- Identify requirements that cannot be objectively measured or validated.
- Look for requirements that may be impractical or beyond the project scope.

Improvement:
- Make sure all requirements are verifiable and measurable.
- Remove or revise requirements that are unrealistic or unattainable.

c) Compare and contrast different techniques for requirement elicitation, such as interviews, surveys, and use case modelling, and determine their effectiveness in gathering user needs.

1. Interviews:
Description: Interviews involve one-on-one or small group interactions between the requirement analyst and stakeholders. It allows for direct communication and discussion of specific topics.
Strengths:
- Real-time communication enables in-depth exploration of stakeholder needs.
- Analysts can ask follow-up questions to clarify ambiguities or delve into details.
- Personal interactions build trust and rapport with stakeholders, leading to more honest and open responses.
Limitations:
- Time-consuming, especially when dealing with multiple stakeholders.
- Responses may be biased due to the presence of the interviewer.
- Stakeholders may not always be available for interviews, leading to scheduling challenges.

2. Surveys:
Description: Surveys involve distributing questionnaires or forms to a large number of stakeholders to collect their opinions, preferences, and requirements.
Strengths:
- Efficient for gathering data from a large number of stakeholders simultaneously.
- Responses can be collected anonymously, encouraging honest feedback.
- Cost-effective, especially when dealing with geographically dispersed stakeholders.
Limitations:
- Limited scope for follow-up questions, which may result in less detailed responses.
- Stakeholders may not respond to the survey, leading to potential non-response bias.
- It might be challenging to capture complex or nuanced requirements through fixed-choice questions.

3. Use Case Modeling:

Description: Use case modeling is a technique used to capture functional requirements of the system by representing interactions between users and the system through scenarios.

Strengths:

- Provides a visual representation of how users interact with the system, making it easier to understand requirements.
- Helps in identifying system functionalities and boundary conditions.
- Encourages stakeholders to think in terms of user interactions and system responses.

Limitations:

- May not fully capture non-functional requirements or system constraints.
- Requires a good understanding of system behavior and user interactions for effective modeling.
- Focuses on what the system should do, but not necessarily on how it should be implemented.

Effectiveness in Gathering User Needs:

- Interviews are highly effective in gathering user needs, especially when in-depth understanding and clarification are required. They foster rich communication and allow for a deeper exploration of requirements.
- Surveys are efficient for gathering a wide range of opinions from a large number of stakeholders. However, they may not capture the same level of detail as interviews or use case modeling.
- Use case modeling is effective in capturing functional requirements and illustrating system-user interactions. It is particularly useful for understanding the system's behavior from a user's perspective.