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:	
Roll No:	9609
Team Members:	Soham Khochare Omkar Surve Emmanuel Gudinho

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.

ABSTRACT

The implementation of an IoT based dustbin that segregates dry and wet waste and senses the level of the dustbin is a step towards effective waste management. The system consists of two compartments for dry and wet waste, sensors to detect the level of waste in each compartment, a microcontroller to control the sensors, a Wi-Fi module to connect the device to the internet, and a website to display the data collected by the device. The microcontroller reads the sensor data and sends it to the internet, where it is processed, and the website is updated in real-time. The dustbin segregates the waste based on the sensor data, making it an efficient and convenient solution for waste management. The system's implementation and testing involved building the physical dustbin, programming the microcontroller, building the website, and testing the IoT based dustbin to ensure that it functioned correctly. The IoT based dustbin can be deployed in the required location and will contribute to effective waste management practices.

INTRODUCTION

Waste management or waste disposal includes the processes and actions required to manage waste from its inception to its final disposal. This includes the collection, transport, treatment and disposal of waste, together with monitoring and regulation of the waste management process and waste-related laws, technologies, economic mechanisms. Proper management of waste is important for building sustainable and liveable cities, but it remains a challenge for many developing countries and cities. A report found that effective waste management is relatively expensive, usually comprising 20%–50% of municipal budgets. Operating this essential municipal service requires integrated systems that are efficient, sustainable, and socially supported. A large portion of waste management practices deal with municipal solid waste(MSW) which is the bulk of the waste that is created by household, industrial, and commercial activity. According to the Intergovernmental Panel on Climate Change (IPCC), municipal solid waste is expected to reach approximately 3.4 Gt by 2050; however, policies and law-making can reduce the amount of waste produced in different areas and cities of the world. Measures of waste management include measures for integrated techno-economic mechanisms of a circular economy, effective disposal facilities, export and import control and optimal sustainable design of

products that are produced. The aim of waste management is to reduce the dangerous effects of such waste on the environment and human health. A big part of waste management deals with municipal solid waste, which is created by industrial, commercial, and household activity.

Project Description

The objective of this project is to design and develop a smart dustbin that can segregate dry and wet waste, sense the level of dustbin, and display the data on a website. The proposed system will be based on IoT technology and will consist of various sensors, microcontrollers, and communication modules.

The working principle of the system is simple. The dustbin will be divided into two compartments - one for dry waste and the other for wet waste. Each compartment will have a dedicated sensor to detect the level of waste in it. The sensors will be connected to a microcontroller that will process the data and send it to a server through a Wi-Fi module. The server will then display the data on a website, allowing users to monitor the dustbin's status from anywhere.

REQUIREMENTS

Software Specifications:

For Frontend Development of our website, we will be using the following languages:

HTML: The Hyper Text Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.

CSS: Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.

JavaScript: JavaScript often abbreviated to JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. All major web browsers have a dedicated JavaScript engine to execute the code on user’s devices.

For Backend Development of our website, we will be using the following languages:

Express.js: Express.js, or simply Express, is a back-end web application framework for building RESTful APIs with Node Js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.

Hardware Specifications

1. Arduino Uno 9V: We'll be using ArduinoUno for programming
2. Servo Motor: Servo Motor will be used for segregation of Wet waste and Dry Waste
3. Ultrasonic Sensor (Motion Sensor): To measure the level of garbage in the dustbin
4. 9V battery
5. M2M, F2M, F2F jumper wires.
6. IR Sensor: To detect the presence of the garbage.
7. GSM 900 module: To send SMS/Notification to the concerned Authorities.

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.