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:	
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:

Department of Computer Engineering

Academic Term: First Term 2022-23

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

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.

Case Study 1 – Requirements Specification Documents

1. Abstract:

We propose a low-cost internet of things (IoT) compliance system that counts the number of people entering and leaving a vicinity. In all vital areas of life such as home appliances, automated machine control, medical diagnostic instruments, transportation etc. usage of digital electronics are widely used. The human counting system is designed to monitor the entrance of a building to ensure any movement made within the protected area. Anybody can remotely screen the presence of a person on the premises. The human detection stages are implemented using sensors through the use of LASER light with the LDR (Light Dependent Resistor). The data stored on the server can be used for compliance auditing, real-time monitoring and planning purposes. The system does not record the personal information of attendees nor provide contact tracking information.

2. Introduction:

1. Purpose:

The purpose of the project is to get the count of people in real time, and analyse them according to the needs.

2. Scope:

The current scope of the project is counting the people in a premises with less crowd and less chaos. In future, we can try building a system to count in crowded places using Machine Learning.

3. Definitions/Acronyms/Abbreviations:

IoT – IoT stands for the Internet of Things. It refers to a network of physical objects, devices, and machines embedded with sensors and software that enables them to connect, collect, and exchange data over the internet. This interconnectivity allows them to communicate with each other and perform various tasks autonomously or with minimal human intervention.

4. References:

[1]. Li, Jingwen, Lei Huang, and Changping Liu. "Online adaptive learning for multi-camera people counting." 21st International Conference on Pattern Recognition (ICPR) IEEE, 2012, PP 3415-3418.

[2]. Sivabalakrishnan, M., and K. Shanthi, "Person Counting System Using EFV Segmentation and Fuzzy Logic." Procedia Computer Science 50 (2015): PP 572-578.

[3]. Kumar, Rakesh, Tapesh Parashar, and Gopal Verma. "Background Modeling and Subtraction Based People Counting for Real Time Video Surveillance."International Journal of Soft Computing and Engineering (IJSCE) (2012), PP 100-102.

[4]. Ekansh Gupta; Nidhi Singh; Mansi Saxena; Kumar Kartikey; Abhishek Sharma (2018): Smaert Attendance Monitoring and Counting System, https://www.irjet.net/archives/V5/i2/IRJET-V5I2 424.pdf.

[5]. Sohn, K.; Kim, D. Dynamic origin–destination flow estimation using cellular communication system. IEEE Trans. Veh. Technol. 2008, 57, 2703–2713. [CrossRef].

[6]. Somayya Madakam; R. Ramaswamy; Siddharth Tripathi (2015): Internet of Things - A Literature Overview.

[7]. Hugo Miguel Somacarrera Lavin, Project on “LASER AND AUDIBLE ALARM LIGHT DETECTION”, Kutahya (Turkey), June 2014.

[8]. G. R Gordon (1959). "The LASER, Light Amplification by Stimulated Emission of Radiation". In Franken, P.A. and Sands, R.H. (Eds.).The Ann Arbor Conference on Optical Pumping, the University of Michigan, 15 June through 18 June 1959. p. 128.OCLC 02460155.

5. Developer Responsibilities:

The developer is responsible for (a) developing the system (b) installing the necessary software (c) making the appropriate connections (d) maintaining the system for a period of time after installation.

3. General Description

1. Product Functions Overview:

This project deal with counting of people entering or leaving the vicinity. In this we used leaser technology to for counting purpose, The counting starts when there is a discontinuity in the laser beam falling on the sensor, this happens when someone passes by crossing the beam.

2. User Characteristics:

There will be an admin who will have access to the portal where all the counts are presented, they can use APIs to display at different places too and handle the APIs permissions.

3. General Constraints:

It should have a local internet connection so that the application should run.

4. General Assumptions and Dependencies.

Not Applicable.

4. Specific Requirements:

1. Inputs and Outputs:

Input: We can give the input of the respective sensor’s count that we want to access.
Output: We can see the insights of the respective sensor.

2. Functional Requirements:

Determine where the system needs to be installed and the system of counting system is needed and install the sensors there.

3. External Interface Requirements:

User Interface: It will be consisting of any device which has a browser, and internet access.

4. Performance Constraints:

It gives real time data and analysis on the user interface.

5. Design Constraints

Software Constraints

Any application can be designed and developed using the APIs, and can be displayed anywhere.

Hardware Constraints

We will need IR sensors and one Arduino UNO device minimum.

Acceptance Criteria

Before accepting the system, the developer must test it by passing one finger or object through the sensor, and see if the count is increased by one for the respective sensor.

Post-lab Experiment 01:

a) Evaluate the importance of a well-defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success.
Ans. A well-defined Software Requirement Specification (SRS) is of paramount importance in the software development lifecycle and has a significant impact on project success. The SRS serves as a foundation for the entire development process, acting as a detailed blueprint that outlines what the software should do, how it should behave, and what the user's expectations are. Here are some reasons why a well-defined SRS is crucial:

1. Clear Communication: The SRS document provides a clear and unambiguous description of the project requirements to all stakeholders, including developers, designers, testers, and clients. This helps in minimizing misunderstandings and ensures that everyone is on the same page throughout the development process.

2. Guiding Development Efforts: A well-defined SRS acts as a roadmap for the development team, outlining the features and functionalities that need to be implemented. It helps the team to stay focused on what needs to be achieved and prevents scope creep, which can lead to project delays and increased costs.

3. Risk Mitigation: By defining requirements upfront, the SRS helps in identifying potential risks and challenges early in the project lifecycle. This allows the team to address these issues proactively and reduces the chances of costly rework or project failure later on.

4. Customer Satisfaction: A comprehensive SRS document ensures that the software is developed according to the client's expectations and needs. When the final product aligns with what the client envisioned, it leads to higher customer satisfaction and builds trust in the development team's capabilities.

5. Estimation and Planning: The SRS provides a basis for project estimation, helping in determining the project's scope, timeline, and resource requirements. Accurate estimations improve project planning and resource allocation, contributing to successful project execution.

6. Quality Assurance and Testing: An SRS with detailed functional and non-functional requirements serves as a reference for testing. Testers can design test cases based on the specified requirements, ensuring that the software meets the desired quality standards.

7. Change Management: Inevitably, requirements may change during the development process due to evolving business needs or client feedback. With a well-defined SRS, these changes can be managed more effectively, allowing the team to evaluate the impact on the project and adjust accordingly.

Overall, the impact of a well-defined SRS on project success cannot be overstated. It sets the direction for the entire software development process, facilitates effective communication, reduces risks, improves planning and estimation, and enhances the likelihood of delivering a high-quality product that meets the client's expectations. On the other hand, an inadequate or poorly-defined SRS can lead to misunderstandings, delays, cost overruns, and dissatisfaction among stakeholders, ultimately jeopardizing the project's success.

b) Analyse a given SRS document to identify any ambiguities or inconsistencies and propose improvements to enhance its clarity and completeness.
Ans. Analysis of an SRS document of an ecommerce website.
Original Requirement:
The website should load quickly to ensure a good user experience.

Analysis:
The above requirement is ambiguous and lacks clarity. The term "quickly" is subjective and does not provide any specific measurable criteria for the website's loading time. It could lead to different interpretations among stakeholders and developers.

Improvement:
The website's homepage should load within 3 seconds, as measured by Google PageSpeed Insights, to ensure a good user experience.

Analysis (Improvement):
The revised requirement provides specific and measurable criteria for the website's loading time. It states that the homepage should load within 3 seconds, and it should be measured using Google PageSpeed Insights, which is a well-known tool for website performance analysis.

Original Requirement:
The website shall be secure.

Analysis:
The above requirement is too generic and lacks specificity. It does not specify what security measures are required for the website.

Improvement:
The website shall use HTTPS protocol to encrypt data transmission and implement user authentication through a secure login system with password hashing and proper session management.

Analysis (Improvement):
The improved requirement outlines specific security measures that the website should implement. It specifies the use of HTTPS for data encryption and requires a secure login system with password hashing and proper session management to enhance overall security.

By providing clear and measurable criteria in the requirements, as demonstrated in the improvements above, the SRS document becomes more precise and less prone to misunderstandings and inconsistencies. Additionally, using specific terms and avoiding ambiguous language ensures that all stakeholders have a common understanding of the project's objectives, leading to a more successful software development process.

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.
Ans.
Requirement elicitation is a critical phase in the software development process, where the needs and expectations of users and stakeholders are gathered to define the system's requirements. Different techniques can be used for requirement elicitation, each with its strengths and

weaknesses. Let's compare and contrast three popular techniques: interviews, surveys, and use case modeling, and assess their effectiveness in gathering user needs:

1. Interviews:
   - Description: Interviews involve one-on-one or group interactions with users, stakeholders, and subject matter experts. The interviewer asks open-ended questions to gather detailed information about requirements, preferences, and pain points.
   - Strengths:
     - Provides an in-depth understanding of user needs and perspectives.
     - Facilitates the exploration of complex requirements and uncovering implicit needs.
     - Enables immediate clarification of doubts and follow-up questions.
   - Weaknesses:
     - Can be time-consuming and resource-intensive, especially for large user groups.
     - Responses may be biased or influenced by the interviewer's presence.
     - May not be feasible for geographically dispersed users.

2. Surveys:
   - Description: Surveys involve distributing questionnaires to a larger group of users or stakeholders. Surveys can be conducted online or through paper forms and can include both open-ended and closed-ended questions.
   - Strengths:
     - Allows gathering data from a large number of users efficiently.
     - Provides quantitative data, making it easier to analyze and compare responses.
     - Suitable for geographically dispersed user groups.
   - Weaknesses:
     - Limited depth of information compared to interviews.
     - Responses may lack context, leading to less detailed requirements.
     - May suffer from low response rates, affecting data quality.

3. Use Case Modeling:
   - Description: Use case modeling involves creating scenarios that describe how users interact with the system to achieve specific goals. It typically includes actors (users or external systems) and use cases (interactions with the system).
   - Strengths:
     - Provides a visual representation of user interactions, enhancing communication between stakeholders.
     - Helps identify user goals and system functionalities.
     - Supports the analysis of system behavior in different scenarios.
   - Weaknesses:
     - May not capture all possible user needs, as it focuses on specific interactions.
     - May require expertise in modeling techniques for effective use.
     - Use cases may not cover all edge cases or exceptional scenarios.

Effectiveness in Gathering User Needs:
- Interviews are highly effective in gathering in-depth user needs and understanding the context and motivation behind requirements. They excel in uncovering implicit requirements and building rapport with users.

- Surveys are effective for reaching a large number of users and obtaining quantitative data on their preferences. While they lack the depth of interviews, they are valuable for collecting general trends and preferences.
- Use case modeling is effective in capturing user interactions and understanding the system's behavior. It complements other techniques by providing a visual representation of user needs and system functionalities.

In practice, a combination of these techniques is often used to gather user needs comprehensively. For example, interviews can be conducted with key stakeholders for critical insights, surveys can be sent to a broader user base to identify common patterns, and use case modeling can be used to map out key interactions and scenarios. The choice of techniques depends on factors such as project scope, user diversity, time constraints, and available resources.