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:	**9598**
Team Members:	**Ryan D'Mello**

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.

Software Requirements Specification

for

Messaging Voice Assistant for Visually Impaired

Version 1.0 approved

Prepared by Ryan D'Mello

TE COMPS B - 9598

27 July 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This project aims to design and develop an application to address and remedy this problem. The basic objective of the project is to build an Android app enabling the user to reply to received messages, make calls, etc using the user's voice as a mode of instruction instead of physical touch, thus, eliminating the dependence on the user's visual capabilities.

1.2 Document Conventions

This document specifies the Software Requirement Specifications (SRS) for the academic mini project and follows the IEEE standard document format. Relevant and appropriate technical terminology is used when required. References and citations are also listed in the appropriate section.

1.3 Intended Audience and Reading Suggestions

This document is mainly intended for developers to allow for functional improvements and convenient maintenance, however, can also be consumed by users to allow for ease-of-use. It lays down the basic motivation behind the development of such a system and its significance to the Indian market, as it is home to the majority of the global blind population. The SRS contains the product description, dependencies and requirements, system features, constraints along with an exhaustive list of glossary entries and references. The recommended sequence for reading the document is in the natural order of reading as it provides the most comprehensive and conclusive reading experience.

1.4 Product Scope

This project aims to design and develop an application to address and remedy this problem. The basic objective of the project is to build an Android app enabling the user to reply to received messages, make calls,etc using the user's voice as a mode of instruction instead of physical touch, thus, eliminating the dependence on the user's visual capabilities.

1. To build an Android application with a simple and minimal yet effective UI that can receive voice commands from user (input), process it and redirect users to appropriate apps or execute the process by itself. (output) [depends on case]

2. To keep the application lightweight with minimum dependencies and efficient use of resources.(memory, network, processing)

3. To maintain compatibility of applications with various Android devices.

4. To allow the possibility of adding new features and functions in future builds with easy maintenance of code.

1.5 References

[1]Rucha Doiphode, Mayuri Ganore, Ashwini Garud, Tejaswini Ghuge, Parminder Kaur (April, 2017), “Be My Eyes : Android App for visually impaired people.”

[2]Rishi Sayal, Chatti Subbalakhmi , H S Saini(Feb 2020), “Mobile App Accessibilityfor Visually Impaired.”

[3]Yogita Balani, Deepa Narayanan, Shashank Parande, Aakash Birari, A.Yeole(2019),”Drishti - Eyes for the blind”

[4]P. Kardyś, A. Dabrowski, M. Iwanowski, Damian Huderek(September 2016),”A new Android application for blind and visually impaired people.”

[5]Flutter.(n.d).Flutter documentation.Retrieved April,2023,from pub.dev/packages/flutter

[6]Alan ai. (n.d). documentation. Retrieved April, 2023, from alan.app/docs

[7]Dart.(n.d). Dart Programming languages. Retrieved April, 2023, from dart.dev/guides

2. Overall Description

2.1 Product Perspective

The specified product is a new, self-contained and standalone system. It is inspired by multiple research papers (listed in section 1.5,references) and aims to make the daily lives of persons with visual disabilities easier by helping them utilize the capabilities of their smartphones using voice,thus,eliminating the need of response and interaction based on vision. As our society advances technologically and the economy digitizes rapidly,solutions for specially abled persons must be developed. There have been many efforts but yet a cost effective and ready-to-use mobile application does not exist on popular app stores.

2.2 Product Functions

The system must be an Android application with which the user is offered options to read received notifications from various apps using text to speech conversion and reply to messages on respective apps using speech recognition and speech to text conversion.

2.3 User Classes and Characteristics

The most essential user class targeted by this system is that of blind and visually impaired persons with trivial technical expertise and familiarity with common multimedia messaging mobile applications such as SMS,WhatsApp,etc.Along with visually impaired persons,other persons without disabilities,as such,must have an optimal user experience too,so as to help the blind/visually impaired in getting acquainted with the system and assist in times of failure.General proficiency in the English language is expected as primary language of operation of the system will be set to English.

2.4 Operating Environment

The system is specifically designed and developed to run on Android smartphones with minimum Android SDK version 5.0 or above.Stable internet connection and microphone and speaker access is essential to ensure optimal performance of the system.

2.5 Design and Implementation Constraints

The system makes use of Android's background activity features and needs constant access to microphone,speaker and network resources.Interfacing with communication applications is of utmost importance and may be subject to change as many communication applications use proprietary protocols.Updates to the system and changes to the design can render the product unusable to users who cannot reacquaint themselves quickly to the changes.Corporate and government regulatory policies can affect the system because of privacy concerns.

2.6 User Documentation

As this system is developed as part of an academic mini project,initially,minimal documentation will be published.However,tutorial videos and online documentation can be developed in the future to improve the learning curve of new users and developers who may wish to contribute to the product.

3. External Interface Requirements

3.1 User Interfaces

The GUI is developed using the Flutter – Material standards and is minimal by design to help in quick onboarding of new users.As this system is developed primarily for blind and visually impaired persons and user interface is voice based,the GUI is not the main mode of operation and navigation through the application.All hardware buttons like the power button,volume buttons,etc function in their default behaviour and remain unaffected by the application.Error messages,app crashes and other failures will be reported and handled by Android's native crash handling service.

3.2 Hardware Interfaces

The minimum Android SDK version is 21 which corresponds to Android 5.0 (Lollipop). However,the recommended Android SDK version is set to 34.The system is generally optimized for the aforementioned specifications and not particularly optimized for every smartphone model as it is impractical in attempting to do so. With uninterrupted microphone and speaker access and the ability to run background activities,the system should work as intended on most devices.

3.3 Software Interfaces

In the development of the system, several tools and frameworks are utilized, including Android Studio, Flutter, and Dart. For development and testing, Android Studio's integrated development environment (IDE) along with its virtual device emulator was used. The application was built using Flutter 3.7.3, the stable version of the cross-platform app development framework. Flutter is chosen due to its ability to deliver high-quality, natively compiled applications for mobile, web, and desktop with a single codebase. Dart 2.19.2, is the language used to write Flutter applications. Dart is a scalable, object-oriented language with a modern syntax and features that make it easy to write efficient code. In addition to these tools, also utilized are several packages and dependencies, including flutter-material for the user interface design, Alan 13 voice for our conversational AI platform, contacts, SMS, permission handler, and telephony.These packages provided us with the necessary functionality to integrate important features into our application, such as contact management, SMS messaging, and permission handling.

3.4 Communications Interfaces

The system requires a few standard communication protocols/interfaces to work appropriately for optimal performance.Messaging standards like SMS and popular applications like WhatsApp must work as expected in coordination with the system with the help of Wi-Fi connection or a SIM based data connection.Various standards of encryption are provided by these services,however,these are subject to change by effect of their policies in the future.

4. System Features

4.1 Wake word detection (running as background activity)

4.1.1 Description and Priority

The application runs as a background activity on Android and is brought to the foreground based on whether the specified wake word is spoken by the user or not.This feature is on the highest priority level as without this the application will fail to launch.

4.1.2 Stimulus/Response Sequences

Upon invocation of this feature,the application interface is presented to the user's screen and an audio response is also played.The event listener listens to the user through the microphone and awaits further commands for execution.

4.1.3 Functional Requirements

Requirement-1: Flutter background activity module
Requirement-2: Alan AI module
Requirement-3:Microphone access

4.2 Voice Recognition and command execution

4.2.1 Description and Priority

Voice Recognition is a high priority feature as it is the only way for the user to navigate through the application and also issue and execute further commands

4.2.2 Stimulus/Response Sequences

The voice input is converted to text and upon detection of commands issued,the relevant commands are executed.Responses are generated in acknowledgement of user's voice input,thus preparing the user to issue the subsequent instruction.

4.2.3 Functional Requirements

Requirement-1: Android speech to text engine
Requirement-2: Alan AI module
Requirement-3:Microphone access

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must run with low latency and low response times to ensure usability,especially,in times of emergency.Memory requirements must also be kept low at all times because if memory usage increases drastically then Android may kill the background activity.The application will be in standby mode in the event of an incoming call or media playback like videos.

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.