Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Experiment No. 4
Develop SRS for selected project in IEEE format
Name: Krisha Chikka
Std/Div: TE/1 Roll no. : 30
Date of Performance: 29/07/2024
Date of Submission: 05/08/2024

Aim: To develop Software Requirement Specification (SRS) document in IEEE format for a case study.

Objective: To understand intended purpose and environment for software under development. Learn how documentation is prepared according to functional & non-functional requirements. And develop SRS for a case study.

Theory:

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Software Requirements Specification

for

<Project>

Version 1.0 approved

Krisha Chikka TE-1_30

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Prepared by <author>

<organization>

<date created>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Introduction

Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a

separate vision and scope document is available, refer to it rather than duplicating its contents here.>

References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Overall Description

Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high-level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

External Interface Requirements

User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are

to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

System Feature 2 (and so on)

Other Nonfunctional Requirements

Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as

specific as possible. You may need to state performance requirements for individual functional requirements or features.>

Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Solution:

Software Requirements Specification (SRS)

for

Currency Detector for the Visually Impaired

Version 1.0 approved

Prepared by Krisha Chikka

<organization>

22 September 2024

1. Introduction

1.1 Purpose

The purpose of this document is to specify the functional and non-functional requirements for a voice-controlled mobile application that helps visually impaired individuals identify the denomination of physical currency. The system should be fully voice-activated, using the device's camera to scan currency notes and providing voice feedback to the user with the denomination.

1.2 Document Conventions

This document uses the following conventions:

- Voice Command : User's spoken input to control the application.
- Camera : The device's built-in camera used to capture the currency.
- Feedback : Voice-based output providing the denomination of the currency note.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Developers are responsible for the implementation of the application.
- Testers are responsible for testing the features and functionality.
- Project Managers overseeing the project timeline and milestones.
- Designers focusing on the user experience for the visually impaired.
- Accessibility Advocates ensure the app follows best practices for users with visual impairments.

1.4 Project Scope

The scope of this project is to deliver a mobile application that allows visually impaired users to identify the denomination of currency through voice commands. The application will use the device's camera to scan the currency, process the image using AI algorithms, and provide voice-based feedback on the denomination. This solution aims to assist visually impaired individuals in financial transactions and daily tasks involving currency handling.

1.5 References

- ISO 9241-171:2008 : Ergonomics of human-system interaction – Guidance on software accessibility.
- W3C Accessibility Guidelines: Recommendations for accessible mobile application development.
- Android/iOS Camera API Documentation: Reference for handling camera operations on mobile devices.
- Speech Recognition API Documentation: For integrating voice command functionality.

2. Overall Description

2.1 Product Perspective

The product is designed as a stand-alone mobile application that integrates with the device's voice command and camera features. The key components include voice interaction, image processing for currency identification, and audio feedback. The system will primarily cater to visually impaired individuals, leveraging machine learning (ML) models to identify different currency denominations. The application must be compatible with major operating systems such as Android and iOS.

2.2 Product Features

The main features of the application include:

- Voice-activated command: to open the application.
- Camera integration: to capture and scan the currency.
- Currency denomination recognition through AI/ML-based image processing.
- Voice feedback providing the user with the correct denomination.

2.3 User Classes and Characteristics

The primary users are visually impaired individuals. The application must be fully accessible, requiring minimal visual interaction. Secondary users may include caregivers or accessibility organizations that work with visually impaired individuals.

2.4 Operating Environment

The system will operate on smartphones with the following specifications:

- Operating Systems: Android 10 and above, iOS 14 and above.
- Hardware Requirements: Smartphone with a built-in camera, microphone, and speaker.
- Network: Optionally, internet access may be required for updates or cloud-based processing.

2.5 Design and Implementation Constraints

- The application must adhere to accessibility standards for visually impaired users.

- The system should use device resources efficiently to avoid battery drain, given that camera and voice processing can be resource-intensive.

2.6 User Documentation

The app will provide an accessible user guide that explains how to activate the voice commands, scan the currency, and receive feedback. The user manual should also cover troubleshooting common issues.

2.7 Assumptions and Dependencies

- Users are expected to have basic knowledge of voice command usage on their device.
- The app will rely on the accuracy of the camera's image capture and the AI/ML models for currency recognition.
- Dependencies include mobile operating system APIs (e.g., voice and camera APIs).

3. System Features

3.1 Voice Activation

Description:

The application opens and functions using voice commands. The primary voice command is "Identify currency," which triggers the app to launch and prepare for currency scanning.

Functional Requirements:

- REQ-001: The app must recognize the command "Identify currency" and launch automatically.
- REQ-002: It should provide a verbal confirmation by saying, "Please show the currency" to signal readiness.
- REQ-003: The system should recognize multiple language inputs for international users, starting with English.

Non-Functional Requirements:

- The app should activate within 2 seconds of receiving the command.
- The voice recognition system should have an accuracy rate of at least 95%.

3.2 Currency Scanning and Identification

Description:

Once the app is activated, the device's camera will scan the currency. The app will identify the denomination using image processing algorithms and a pre-trained AI model.

Functional Requirements:

- REQ-004: The camera should automatically activate after the voice command is received.
- REQ-005: The app should be able to recognize all commonly used denominations (e.g., $1, $5, $10, $20, etc.).
- REQ-006: The system should support multi-currency recognition for users from different countries.

Non-Functional Requirements:

- The scanning process must be completed within 5 seconds.
- The app should function effectively in varying lighting conditions.

3.3 Voice Feedback

Description:

After the currency is identified, the app will provide a voice output announcing the denomination.

Functional Requirements:

- REQ-007: The app should announce the denomination within 2 seconds after identification.
- REQ-008: The voice feedback should be clear, natural, and in the user's preferred language.
- REQ-009: The app should handle voice feedback requests like "Repeat" or "Say again."

Non-Functional Requirements:

- The voice output should be loud enough for users with limited hearing abilities.
- The app should use a high-quality text-to-speech engine.

4. External Interface Requirements

4.1 User Interfaces

- The app will have minimal visual interaction elements. On the screen, basic navigation instructions and system status messages will be available in text and voice formats.

4.2 Hardware Interfaces

- The app will utilize the mobile device's camera for scanning currency and the microphone for receiving voice commands.

4.3 Software Interfaces

- The application will communicate with Android/iOS system APIs for voice recognition, camera activation, and text-to-speech functionality.

4.4 Communication Interfaces

- In offline mode, the app will process images locally.
- For improved accuracy, the app may use cloud-based image recognition in online mode.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The app should load and be ready to scan within 3 seconds after receiving the initial voice command.
- It should handle at least 1000 currency identification requests per day without performance degradation.

5.2 Safety Requirements

- The app should ensure that no personal or sensitive data (e.g., images of the currency) is stored on the device or transmitted to external servers without user consent.

5.3 Security Requirements

- The app must comply with data privacy regulations, ensuring that no personally identifiable information is stored or transmitted without explicit consent.

5.4 Software Quality Attributes

- Usability: The app should be simple to use, even for users unfamiliar with smartphones.
- Reliability: The app should operate smoothly in offline and online modes with minimal crashes.
- Portability: The app must be compatible with both Android and iOS devices.
- Maintainability: The system should be easily updatable for currency recognition model improvements.

6. Other Requirements

Appendix A: Glossary
- Currency Scanning: The process of capturing the currency note using the device camera.
- Voice Command: A spoken input that initiates an action within the app.
- Feedback: The voice output provided by the app in response to a recognized currency denomination.

Appendix B: Analysis Models
Detailed models for voice recognition and currency identification will be included later in the development phase.

Appendix C: Issues List
A list of known issues, bugs, or future enhancements will be documented as the project progresses.

Conclusion: We have successfully prepared SRS document for Currency Detector for the Visually Impaired.