Software Requirements Specification (SRS) Template

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

Stay clear and short, yet complete.

Real Time OCR with Navigation
3 Members
Jbareen Mohamad – Amer Eyad – Dregat Ahmad

Software Requirements Specification

Document

Version: 1.0 **Date: 01/21/2020**

Table of Contents

1. Introduction

This project is represents the text identification, and extract this text from pictures, videos and real time videos.

1.1 Purpose

- We will use the existing "OCR" system on images so we will improve it to a system that will work on existing videos and also extract text from real-time video so that we use navigation algorithms and find real-time location.
  Project scope – The project will include existing directories (OpenCV, Tesseract,

1.2 Scope

(1) PC and Android
(2) It will be allow to insert pictures, video and open camera to recognize the text.
(3) The user can use the algorithm to extract text fro videos while driving or walking.

1.3 Definitions, Acronyms, and Abbreviations.

- ***OCR:*** *system is used to convert images and videos of text into letters , words, and sentence. It is widely used in various fields to convert/ extract the information from the image or video . It is also used in signature recognition , automated data evaluation , and security systems. It is commercially used to validate data records , passport doc, place signs, business cards ,printouts of static data, and so on . OCR is a field of research in pattern recognition, deep learning ,artificial intelligence and computer vision.*
- ***Tessaract:****Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images, Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.*
- ***OpenCV:***

1.4 References

(1) https://books.google.co.il/books?id=kqGKDwAAQBAJ&dq=optical+character+recognition+python&hl=iw&source=gbs_navlinks_s

(2) https://pysource.com/2019/10/14/ocr-text-recognition-with-python-and-api-ocr-space/
(3) https://stackabuse.com/pytesseract-simple-python-optical-character-recognition/
(4) https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/
(5) https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/

1.5 Overview

(1) Describe what the rest of the SRS contains

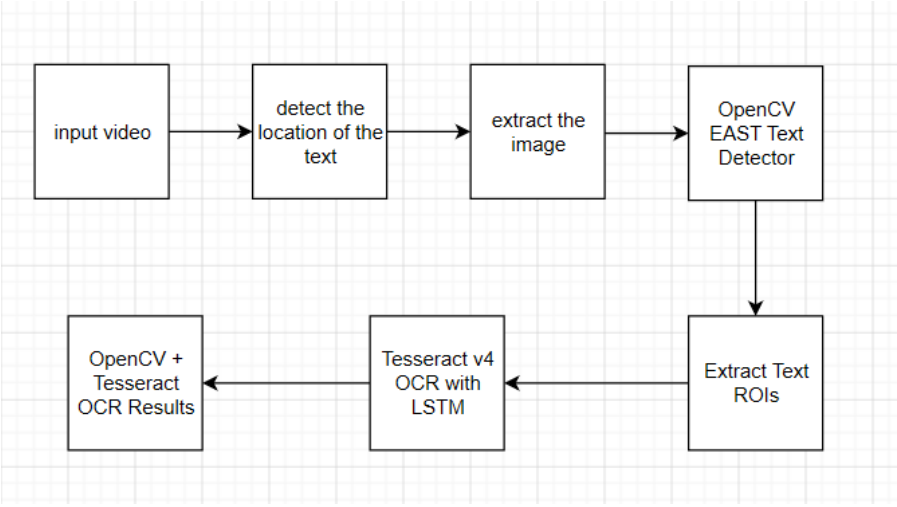
Don't rehash the table of contents here. Point people to the parts of the document they are most concerned with. Customers/potential users care about section 2, developers care about section 3.

2. The Overall Description

The product required Low-High PC and it works with Android smartphone, normal camera or videos from the phone's storage.

2.1 Product Perspective

*This project took from **pysource** that takes pictures and extract the text from it, and we improved that algorithm to work with real time videos.*



2.1.1 System Interfaces

List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system. These are external systems that you have to interact with. For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API to that system that designer's will need to use?

2.1.2 Interfaces

(1) The quality of the videos is not clear enough to recognize the text.
(2) The videos captures with high speed that can't capture the text.

The users should put a videos and picture with med-high quality, balanced video that the algorithm can handle.

2.1.3 Hardware Interfaces

The product required any type med-High PC and it works with Android smartphone. normal camera or videos from the phone's storage.
The algorithm will work smoothly with this requirements.

2.1.4 Software Interfaces

Specify the use of other required software products and interfaces with other application systems. For each required software product, include:
(1) Name: Real-Time OCR
(2) Version number: 1.0
(3) Source: Android studio

2.1.5 Communications Interfaces

The project will work without connection to the internet, it work with our machine learning model that can handle the pictures and the videos.

2.1.6 Memory Constraints

We have DATASET which is stored within GOOGLE DRIVE so we use the existing DATASET to teach the model so that it can identify text more reliably and safely. FIREBASE We will use an existing account in FIREBASE to save us any data such as saving the locations of the signs on the map

2.1.7 Operations

(1) The various modes of operations in the user organization
(2) The algorithm will work 24/7
(3) Image processing with OpenCV and Tesseract and pridect the text with our model.
(4) The data will be saved in FIREBASE

2.1.8 Site Adaptation Requirements

In this section:
(1) The requirements are to get clear photos and videos.

We may save the captured text with the location in the user's storage.

2.2 Product Functions

The algorithm takes the videos extract the text from it, locate the location of the signs that contains text.

2.3 User Characteristics

All pedestrians can use a party algorithm to tag themselves with their location ,
Any corrected readiness on a camera can run the algorithm to help navigate the readiness location

2.4 Constraints

there are several limitations of OCR that can result in inaccurate or missing text which makes text-based classification difficult or impossible:These can include:

(1) The algorithm will give us reliability in our results and meet tough conditions (10) Criticality of the application
(2) The algorithm will not crash and not get stuck while you work and even if it gets stuck then your latest data will be saved
(3) An algorithm will help existing interfaces like a map

2.5 Assumptions and Dependencies

Image quality can affect the reliability of the resultThe type of camera if it does not meet the conditions and does not give good basic qualityAngle and speed of photography can affect the algorithm's workA computer or smartphone that can run the algorithmRain can affect live work so we cannot identify the sign before the text is identified

2.6 Apportioning of Requirements.

The algorithm will follow some requirements below:
IDs from text to live streaming, and also saving data after the algorithm gets stuck , Log in to the app with beautiful design,finding a location through the algorithm makes it difficult to implement it in the first version,In the first version the algorithm will work on PC in first version and not on Android.

3. Specific Requirements

The algorithm works with python language and runs with those libraries:

- *OpenCV for image processing*
- *Tesseract python for extract text from the photos*
- *TensorFlow for train the model and predict for later inputs (pictures and videos)*
- *NumPy*

3.1 External Interfaces

[SKIP THIS PART]

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.

It contains both content and format as follows:

- *Name of item*
- *Description of purpose*
- *Source of input or destination of output*
- *Valid range, accuracy and/or tolerance*
- *Units of measure*
- *Timing*
- *Relationships to other inputs/outputs*
- *Screen formats/organization*
- *Window formats/organization*
- *Data formats*
- *Command formats*
- *End messages*

3.2 Performance Requirements

[SKIP THIS PART]

This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:

(a) The number of terminals to be supported
(b) The number of simultaneous users to be supported
(c) Amount and type of information to be handled

Static numerical requirements are sometimes identified under a separate section entitled capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 second

rather than,

An operator shall not have to wait for the transaction to complete.

(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)

3.3 Logical Database Requirements

After processing the images and the videos we will save the captured data, location and the signs to the FIREBASE for the Android app and PC.

3.4 Design Constraints

Specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.4.1 Standards Compliance

[SKIP THIS PART]

Specify the requirements derived from existing standards or regulations. They might include:

(1) Report format

(2) Data naming
(3) Accounting procedures
(4) Audit Tracing

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.5 Software System Attributes

[SKIP THIS PART – FILL ONLY SECTION 3.6.3 ON Security]

There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.

These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.

3.5.1 Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn't refer to just having a program that does not crash. This has a specific engineering meaning.

3.5.2 Availability

Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? "The system shall allow users to restart the application after failure with the loss of at most 12 characters of input".

3.5.3 Security

Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

* *Utilize certain cryptographic techniques*
* *Keep specific log or history data sets*
* *Assign certain functions to different modules*
* *Restrict communications between some areas of the program*
* *Check data integrity for critical variables*

3.5.4 Maintainability

Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system

3.5.5 Portability

Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:

* *Percentage of components with host-dependent code*
* *Percentage of code that is host dependent*
* *Use of a proven portable language*
* *Use of a particular compiler or language subset*
* *Use of a particular operating system*

Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.

ID	Characteristic	H/M/L	1	2	3	4	5	6	7	8	9	10	11	12
1	Correctness													
2	Efficiency													
3	Flexibility													
4	Integrity/Security													
5	Interoperability													
6	Maintainability													

7	Portability													
8	Reliability													
9	Reusability													
10	Testability													
11	Usability													
12	Availability													

Definitions of the quality characteristics not defined in the paragraphs above follow.

- *Correctness - extent to which program satisfies specifications, fulfills user's mission objectives*
- *Efficiency - amount of computing resources and code required to perform function*
- *Flexibility - effort needed to modify operational program*
- *Interoperability - effort needed to couple one system with another*
- *Reliability - extent to which program performs with required precision*
- *Reusability - extent to which it can be reused in another application*
- *Testability - effort needed to test to ensure performs as intended*
- *Usability - effort required to learn, operate, prepare input, and interpret output*

THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.

3.6 Organizing the Specific Requirements

3.6.1 System Mode

To train the model with high accuracy you need a good GPU that can handle the data with less time.
Some of the phones that have low performance can handle the data it recived with quite long time, it took long time to process the data with low performance.

3.6.2 Stimulus

PC or Android device with good performance will behave better in real time videos.

3. 6.3 Response

To get fast response the device should have good GPU to process the inputs faster.

4. Change Management Process
[SKIP THIS PART]

Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements. Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?

5. Document Approvals
[YOUR SUPERVISOR]

Identify the approvers of the SRS document. Approver name, signature, and date should be used

.

---------- [END OF RELEVANT PARTS]----------

6. Supporting Information
[SKIP THIS PART]

The supporting information makes the SRS easier to use. It includes:

* *Table of Contents*
* *Index*
* *Appendices*

The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:

(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys
(b) Supporting or background information that can help the readers of the SRS
(c) A description of the problems to be solved by the software
(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.

Outline for SRS Section 3
Organized by mode: Version 1

3. Specific Requirements
   3.1 External interface requirements
      3.1.1 User interfaces
      3.1.2 Hardware interfaces
      3.1.3 Software interfaces
      3.1.4 Communications interfaces
   3.2 Functional requirements
      3.2.1 Mode 1
         3.2.1.1 Functional requirement 1.1
         .....
         3.2.1.*n* Functional requirement 1.*n*
      3.2.2 Mode 2
         .....
      3.2.*m* Mode *m*
         3.2.*m*.1 Functional requirement *m*.1
         .....
         3.2.*m.n* Functional requirement *m.n*
   3.3 Performance Requirements
   3.4 Design Constraints
   3.5 Software system attributes
   3.6 Other requirements

Outline for SRS Section 3
Organized by mode: Version 2

3. Specific Requirements
  3.1 Functional Requirements
    3.1.1 Mode 1
      3.1.1.1 External interfaces
        3.1.1.1 User interfaces
        3.1.1.2 Hardware interfaces
        3.1.1.3 Software interfaces
        3.1.1.4 Communications interfaces
      3.1.1.2 Functional Requirement
        3.1.1.2.1 Functional requirement 1
        .....
        3.1.1.2.*n* Functional requirement *n*
      3.1.1.3 Performance
    3.1.2 Mode 2
        .....
    3.1.*m* Mode *m*
  3.2 Design constraints
  3.3 Software system attributes
  3.4 Other requirements

Organized by user class (i.e. different types of users ->System Adminstrators, Managers, Clerks, etc.)

3. Specific Requirements
   3.1 External interface requirements
      3.1.1 User interfaces
      3.1.2 Hardware interfaces
      3.1.3 Software interfaces
      3.1.4 Communications interfaces
   3.2 Functional requirements
      3.2.1 User class 1
         3.2.1.1 Functional requirement 1.1
         .....
         3.2.1.*n* Functional requirement 1.*n*
      3.2.2 User class 2
         .....

      3.2.*m* User class *m*
         3.2.*m*.1 Functional requirement *m*.1
         .....
         3.2.*m.n* Functional requirement *m.n*
   3.3 Performance Requirements
   3.4 Design Constraints
   3.5 Software system attributes
   3.6 Other requirements

Outline for SRS Section 3
Organized by object (Good if you did an object-oriented analysis as part of your requirements)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Classes/Objects
  - 3.2.1 Class/Object 1
    - 3.2.1.1 Attributes (direct or inherited)
      - 3.2.1.1.1 Attribute 1
      - .....
      - 3.2.1.1.*n* Attribute *n*
    - 3.2.1.2 Functions (services, methods, direct or inherited)
      - 3.2.1.2.1 Functional requirement 1.1
      - .....
      - 3.2.1.2.*m* Functional requirement 1.*m*
    - 3.2.1.3 Messages (communications received or sent)
  - 3.2.2 Class/Object 2
    - .....
  - 3.2.*p* Class/Object *p*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by feature (Good when there are clearly delimited feature sets.

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
    - 3.2.1.1 Introduction/Purpose of feature
    - 3.2.1.2 Stimulus/Response sequence
    - 3.2.1.3 Associated functional requirements
      - 3.2.1.3.1 Functional requirement 1
      - .....
      - 3.2.1.3.*n* Functional requirement *n*
  - 3.2.2 System Feature 2
    - .....
  - 3.2.*m* System Feature *m*
    - .....
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by stimulus (Good for event driven systems where the events form logical groupings)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Stimulus 1
    - 3.2.1.1 Functional requirement 1.1
    - .....
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Stimulus 2
  - .....
  - 3.2.*m* Stimulus *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .....
    - 3.2.*m.n* Functional requirement *m.n*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by response (Good for event driven systems where the responses form logical groupings)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Response 1
    - 3.2.1.1 Functional requirement 1.1
    - .....
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Response 2
  - .....
  - 3.2.*m* Response *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .....
    - 3.2.*m.n* Functional requirement *m.n*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by functional hierarchy (Good if you have done structured analysis as part of your design.)

3 Specific Requirements
3.1 External interface requirements
3.1.1 User interfaces
3.1.2 Hardware interfaces
3.1.3 Software interfaces
3.1.4 Communications interfaces
3.2 Functional requirements
3.2.1 Information flows
3.2.1.1 Data flow diagram 1
3.2.1.1.1 Data entities
3.2.1.1.2 Pertinent processes
3.2.1.1.3 Topology
3.2.1.2 Data flow diagram 2
3.2.1.2.1 Data entities
3.2.1.2.2 Pertinent processes
3.2.1.2.3 Topology
.....
3.2.1.*n* Data flow diagram *n*
3.2.1.*n*.1 Data entities
3.2.1.*n*.2 Pertinent processes
3.2.1.*n*.3 Topology
3.2.2 Process descriptions
3.2.2.1 Process 1
3.2.2.1.1 Input data entities
3.2.2.1.2 Algorithm or formula of process
3.2.2.1.3 Affected data entities
3.2.2.2 Process 2
3.2.2.2.1 Input data entities
3.2.2.2.2 Algorithm or formula of process
3.2.2.2.3 Affected data entities
…..
3.2.2.*m* Process *m*
3.2.2.*m*.1 Input data entities
3.2.2.*m*.2 Algorithm or formula of process
3.2.2.*m*.3 Affected data entities
3.2.3 Data construct specifications
3.2.3.1 Construct 1
3.2.3.1.1 Record type
3.2.3.1.2 Constituent fields
3.2.3.2 Construct 2

Outline for SRS Section 3
Showing multiple organizations (Can't decide? Then glob it all together)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 User class 1
    - 3.2.1.1 Feature 1.1
      - 3.2.1.1.1 Introduction/Purpose of feature
      - 3.2.1.1.2 Stimulus/Response sequence
      - 3.2.1.1.3 Associated functional requirements
    - 3.2.1.2 Feature 1.2
      - 3.2.1.2.1 Introduction/Purpose of feature
      - 3.2.1.2.2 Stimulus/Response sequence
      - 3.2.1.2.3 Associated functional requirements
      - …..
    - 3.2.1.*m* Feature 1.*m*
      - 3.2.1.*m*.1 Introduction/Purpose of feature
      - 3.2.1.*m*.2 Stimulus/Response sequence
      - 3.2.1.*m*.3 Associated functional requirements
  - 3.2.2 User class 2
  - .....
  - 3.2.*n* User class *n*
  - .....
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by Use Case (Good when following UML development)

3. Specific Requirements
  3.1 External Actor Descriptions
    3.1.1 Human Actors
    3.1.2 Hardware Actors
    3.1.3 Software System Actors
  3.2 Use Case Descriptions
    3.2.1 Use Case 1
    3.2.2 Use Case 2

    3.2.n Use Case n
  3.3 Performance Requirements
  3.4 Design Constraints
  3.5 Software system attributes
  3.6 Other requirements