**Deliverable 1: Project definition and Software Requirements Specification**

**Create a GitHub Wiki page (NOT an attachment) with the items requested in this template.**

All notes containing figures, explanations, and examples for filling out the template should be deleted and replaced by the project information.This template is based on ISO/IEC/IEEE 29148:2011, which is an international standard that indicates how to define system and software requirements.

The software requirements specification (SRS) is a specification for a particular software product that performs certain functions in a specific environment.

The following is a summary of what each section should contain. For more information, please refer to the documents:

* 29148-2011-Requirements specification template -> section 8.4 Software requirements specification document (page 53) and section 9.5 Software requirements specification (SRS) document (page 62)
* Towards a new template for the specification of requirements in semi-structured natural language (available at: <https://www.researchgate.net/publication/339396006_Towards_a_new_template_for_the_specification_of_requirements_in_semi-structured_natural_language>)

For examples see the following documents:

* Example - EIP\_RequirementsSpecificationGLA\_ V2-5
* 2020-Example Software Requirements Specification Document for ReqView

# Introduction

The Introduction section provides an overview of the Requirements Specifications and the scope of the system. The introduction section is divided into subsections. A general introduction should not be provided, only subsections should be developed. Use información from the product vision board.

### 1.1. Product perspective

Define the system's relationship to other related products. If the product is an element of a larger system, then relate the requirements of that larger system to the functionality of the product covered by the SRS.

If the product is an element of a larger system, then identify the interfaces between the product covered by the SRS and the larger system of which the product is an element. A block diagram showing the major elements of the larger system, interconnections, and external interfaces can be helpful. Describe how the software operates within the following constraints:

a) System interfaces;

b) User interfaces;

c) Hardware interfaces;

d) Software interfaces;

e) Communications interfaces;

f) Memory;

g) Operations;

h) Site adaptation requirements.

### 1.2. Product functions

Describe briefly the functionalities that the product will have.

### 1.3. User characteristics

Describe those general characteristics of the intended groups of users of the product including characteristics that may influence usability, such as educational level, experience, disabilities, and technical expertise.

### 1.4. Limitations

Provide a general description of any other items that will limit the supplier's options, including:

a) Regulatory policies;

b) Hardware limitations (e.g., signal timing requirements);

c) Interfaces to other applications;

d) Parallel operation;

e) Audit functions;

f) Control functions;

g) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);

h) Quality requirements (e.g., reliability);

i) Criticality of the application;

j) Safety and security considerations;

k) Physical/mental considerations.

# 2. References

The SRS should contain additional supporting information including:

a) Sample input/output formats, descriptions of cost analysis studies, or 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. The SRS should explicitly state whether or not these information items are to be considered part of the requirements.

# 3. Specific requirements

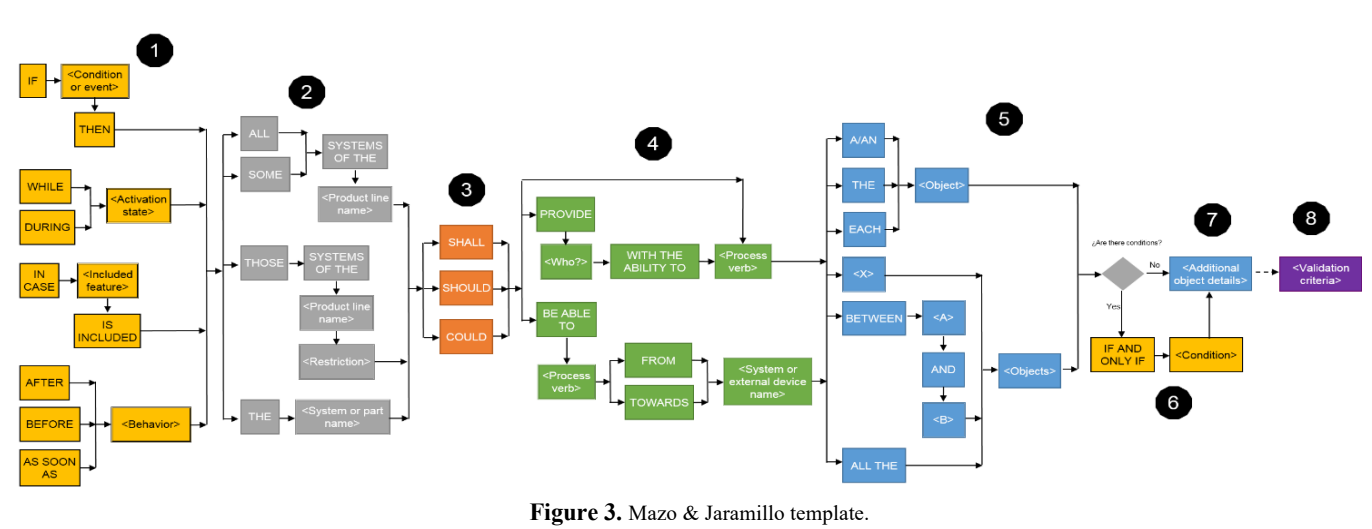
Each requirement shall possess the following characteristics:

* **Necessary.** The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process.
* **Implementation Free.** The requirement, while addressing what is necessary and sufficient in the system, avoids placing unnecessary constraints on the architectural design. The objective is to be implementation independent. The requirement states what is required, not how the requirement should be met.
* **Unambiguous.** The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand.
* **Consistent.** The requirement is free of conflicts with other requirements.
* **Complete.** The stated requirement needs no further amplification because it is measurable and sufficiently describes the capability and characteristics to meet the stakeholder's need.
* **Singular.** The requirement statement includes only one requirement with no use of conjunctions.
* **Feasible.** The requirement is technically achievable, does not require major technological advances, and fits within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
* **Traceable.** The requirement is upwards traceable to specific documented stakeholder statement(s) of need, higher tier requirement, or other source (e.g., a trade or design study). The requirement is also downwards traceable to the specific requirements in the lower tier requirements specification or other system definition artefacts. That is, all parent-child relationships for the requirement are identified in tracing such that the requirement traces to its source and implementation.
* **Verifiable.** The requirement has the means to prove that the system satisfies the specified requirement. Evidence may be collected that proves that the system can satisfy the specified requirement. Verifiability is enhanced when the requirement is measurable.

Requirements should state 'what' is needed, not 'how'. Vague and general terms shall be avoided. They result in requirements that are often difficult or even impossible to verify or may allow for multiple interpretations. The following are types of unbounded or ambiguous terms:

* Superlatives (such as 'best', 'most')
* Subjective language (such as 'user friendly', 'easy to use', 'cost effective')
* Vague pronouns (such as 'it', 'this', 'that')
* Ambiguous adverbs and adjectives (such as 'almost always', 'significant', 'minimal')
* Open-ended, non-verifiable terms (such as 'provide support', 'but not limited to', 'as a minimum')
* Comparative phrases (such as 'better than', 'higher quality')
* Loopholes (such as 'if possible', 'as appropriate', 'as applicable')
* Negative statements (such as statements of system capability not to be provided)

To describe each requirement, you should use the requirements writing template presented in section 5 Proposing a new requirements specification template (see: Towards a new template for the specification of requirements in semi-structured natural language or <https://www.researchgate.net/publication/339396006_Towards_a_new_template_for_the_specification_of_requirements_in_semi-structured_natural_language>).



Identify each requirement with a unique code. For example, FR01 (for functional requirements), PR02 (for performance requirements), UR03 (for usability requirements) ...

Develop the following subsections.

## 3.1. Functions

Functional requirements describe the system or system element functions or tasks to be performed. Define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs.

Each team must define **at least 20 functional requirements**.

## 3.2. Usability requirements

Define usability (quality in use) requirements. Usability requirements and objectives for the software system include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use.

Each team must define **at least 5 usability requirements**.

## 3.3. Logical database requirements

Specify the logical requirements for any information that is to be placed into a database, including:

a) Types of information used by various functions;

b) Frequency of use;

c) Accessing capabilities;

d) Data entities and their relationships;

e) Integrity constraints;

f) Data retention requirements.

Each team must define **at least 5 logical database requirements**.

## 3.4. Constraints

A requirement that limits the options open to a designer of a solution by imposing immovable boundaries and limits (e.g., the system shall incorporate a legacy or provided system element, or certain data shall be maintained in an on-line repository). Specify constraints on the system design imposed by external standards, regulatory requirements, or project limitations.

# 4. Video

Add the link to the video, ensuring access: YouTube, Vimeo.

All members must appear in the video.

Video length: minimum 5 minutes - maximum 7 minutes.

Content of the video:

* Name of project or team.
* Problem / opportunity / need.
* Team members.
* Proposed solution (explain your value proposal).
* Top 12 functional requirements (prioritization activity must be done first).

5. Project management

Create the project in GitHub and record top 10 requirements.

Record weekly meetings (each student should answer: What did you do last week? What are you going to do this week? Are there any obstacles in the way?).

Report retrospective (the team must answer: What should we continue to do (best practices)? What should we start doing (process improvements)? What should we stop doing (process problems and bottlenecks)?).

Develop class assignments and add them to the wiki.