**<Project Name>**

**<Team Members>**

**Software Requirements Specification [Template]**

***How to use this document:*** Keep the headings and document structure, and replace or modify the text within each section and subsection with the content described in that section. Optionally, another level of structure should be added in Section 2 to organize the specific requirements around major classes of user activities or system behaviors, as described in that section. Note that some headings, such as “1. The Concept of Operations”, need no text other than briefly describing what that section contains. The final SRS should be a self-contained document that clearly describes what the document is, who wrote each section, and when. None of this explanatory text that describes *how to write* the document should be present. Delete this paragraph before submitting.

# The Concept of Operations (ConOps)

The concept of operations (ConOps) “describes system characteristics for a proposed system from the users’ viewpoint.” (IEEE Std 1362-1998) The ConOps document communicates overall system characteristics to all stakeholders. It should be readable by any stakeholder familiar with the application domain. The ConOps should clarify the software’s context and the capabilities the system will provide the user (Faulk, 2017).

You must include all the sections described here in the ConOps. (Their source is IEEE Std 1362-1998.) Note that, for a small system, Sections 1.1 through 1.5 may only contain one or two paragraphs each. The Uses Cases in Section 1.6 should be longer, though.

You can see more sections that can go into a ConOps in Section 4, “Elements of a ConOps document” of IEEE Std 1362-1998 at <https://ieeexplore.ieee.org/document/761853>.

## Current System or Situation

This section ‘describes the real-world situation or system (either automated or manual), as it currently exists, that motivates the development of the proposed system. This section provides readers with an introduction to the problem domain, which enables readers to understand better the motivation to develop the system you are describing. This section discusses interfaces, systems, or procedures that are external to the system but essential to understanding the current system or situation.’ (IEEE Std 1362-1998)

For example, if the system will create a new way for students to access required reading materials for college classes, this section would briefly describe the purpose of assigned reading, the various ways that students are currently informed of reading assignments (course web pages, in-class announcements, etc.), and the way that reading materials are made available to students (bookstore, online handouts, etc.).

## Justification for a New System

This section describes the *shortcomings* of the current system or situations that motivate the development of a new system or modification of an existing system. This Section must provide a transition from the current system or status to the description of the proposed system. This section should indicate if there is no existing system your project will modify (IEEE Std 1362-1998).

This section must provide an objectively measurable justification for the system based on real-world data, not opinions, hunches, or unsupported statements. For example, a rigorous student survey might indicate that half of all students on campus own a physical or digital copy of required reading and that most digital copies technically violate copyright laws. Similarly, data from reliable sources could note that the cost of college textbooks over the last twenty years has risen much more than that of other books and the cost of tuition. Such objective data could support a claim that student access to required reading materials needs to be improved.

This section, or the next one, should review currently available systems that could solve the shortcomings of the existing system or situation and explain how these systems do not.

Note that sections 1.1 and 1.2 together identify and describe an unmet need. The following section explains at a high level how this need will be met.

## Operational Features of the Proposed System

This section is where you briefly describe what the new system will do. ‘This section describes the critical operational features—the essential services and constraints—of the proposed system at a high level and without specifying design details. The description should sufficiently explain how the system fulfills the users’ needs and solves the current system’s shortcomings or situation. In some cases, to clarify the operational details of the proposed system, it may be necessary to suggest some possible design details, such as possible design strategies or specific implementations. Make clear that these are not committed design specifications’ (IEEE Std 1362-1998).

For example, for a required-reading-distribution system, the key operational features could include some means of assisting students with automatically taking high-resolution screenshot after screenshot of an existing shared online textbook viewing system (presuming that such a system does not currently exist) and combining these screenshots into a complete PDF of the book. (You need to be aware of copyright laws; it could be unethical to specify such a system.)

## User Classes

This section briefly describes each user class for the proposed system. A user class is a group of people that will interact with the system roughly similarly, such as students, instructors, system administrators, book publishers, programmers, or software installers. “Factors that distinguish a user class include common responsibilities, skill levels, work activities, and system interaction modes. Different user classes may have distinct operational scenarios [discussed below] for their interactions with the system. In this context, a user is anyone who interacts with the existing system, including operational users, data entry personnel, system operators, operational support personnel, software maintainers, and trainers.” (IEEE Std 1362-1998)

## Modes of Operation

This section describes the various modes of operation for the proposed system. For example, regular users, administrators, backup, degraded, maintenance, training, student, or instructor could have a different operation mode. Include all of the modes that apply to all user classes. Each user class might have a different operation mode. Some user classes will have multiple modes. (IEEE Std 1362-1998)

## Operational Scenarios (aka “Use Cases”)

“An operational scenario [also known as “Use Cases”] is a step-by-step description of how the proposed system should operate and interact with its users and its external interfaces under a given set of circumstances. Describe scenarios in a manner that will allow readers to walk through them and gain an understanding of how all the various parts of the proposed system function and interact. The scenarios tie together all parts of the system, the users, and other entities by describing how they interact.” (IEEE Std 1362-1998)

‘Operational scenarios should describe operational sequences that illustrate the roles of the system, its interactions with users, and interactions with other systems. Describe operational scenarios for all modes and all classes of users identified for the proposed system. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the proposed system. Prototypes, storyboards, and other media, such as video or hypermedia presentations, may be used to provide part of this information.’ (IEEE Std 1362-1998)

Include, in each operational scenario (or use-case), a one-sentence description of the scenario, a list of the users or “actors” (from the user classes) involved in the scenario, the preconditions for starting the scenario, and the postconditions (the relevant status of the system and world) after the scenario is completed.

Structure the writing of operational scenarios or use cases so that they are easy to read, with headings and numbered steps rather than in paragraph form. For example, note how the following is well-structured to make it easy to read.

**Use Case: Get a copy of an online book.**

***Brief description:*** This use case describes how a student would make a digital copy of the required reading for a course for a textbook available to students online but which the student cannot download to their computer.

***Actors:*** A student.

***Preconditions:***

1. The student can access an online digital copy of the assigned reading.

2. The student can view the reading material one full page at a time.

3. The viewer can advance one page at a time by clicking a button or keyboard shortcut.

4. Each page of reading material is positioned in the same rectangle on the computer screen.

5. The student can load applications on the machine, access the online digital copy of the reading, and access substantial (such as 100MB) hard drive space on that machine.

***Steps to Complete the Task:***

1. The student gains access to the online digital copy of the assigned reading.

2. The user figures out:

(a) How to display the reading material one page at a time, with the page filling as much of the screen as possible (perhaps even turning the computer screen 90°).

(b) What commands are available to advance through the reading one page at a time?

3. The user starts up the required reading-assistant software.

...

***Postconditions:***

The user has a PDF of all required reading, with OCR (optical character recognition) applied, and a separate chapter within the PDF for each reading assignment or chapter in the book.

You can read more on use cases in the Oracle (2007) White Paper on “Getting Started With Use Case Modeling”, available at: <https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf>

**Note that diagrams can assist with communication.**

“Graphical tools should be used wherever possible, especially since ConOps documents should be understandable by several different types of readers. Useful graphical tools include but are not limited to, work breakdown structures (WBS), N2 charts, sequence or activity charts, functional flow block diagrams, structure charts, allocation charts, data flow diagrams (DFD), object diagrams, context diagrams, storyboards, and entity-relationship diagrams.” (IEEE Std 1362-1998).

# Specific Requirements

This part of the document is where you specify the actual requirements. A requirement describes a behavior or property that a computer program must have, independent of how that behavior or property is achieved. Requirements must be complete, unambiguous, consistent, and objectively verifiable (Sethi, Chapter 3). Requirements describe what the system will do but do not commit to specific *design* details of how the system will do it.

Organize the requirements in a hierarchy. A good organization (a) makes requirements easier to read and understand because related requirements will be near each other in the document, (b) makes requirements easier to modify and update, and (c) makes it easier to find a specific requirement. There are several ways to organize requirements to help achieve these goals.

The following section headings provide one way to organize the requirements (feel free to adapt it to your needs). For example, sections 2.1, 2.2, 2.3, and 2.4 (see headings below) describe “behavioral requirements” (Faulk, 2013). If a system supports two major user activities, it might be best to describe the behavioral requirements for each activity separately. For example, in a digital deejaying system, the two major activities could be (a) loading songs into the system and (b) using the system to play songs. It might be best to fully specify everything in sections 2.1 through 2.4, first for the song-loading activity and then for the song-playing activity.

This document must distinguish between “functional” and “non-functional” requirements. The former describes services provided by the system and the latter constraints on the system and its development.

Requirements should be prioritized, with each classified as (a) Must have, (b) Should have, (c) Could have, and (d) Won’t have (MoSCoW – Sethi, p. 110). When reading requirements, it should be straightforward to see how each requirement is classified, such as by grouping them by priority.

Throughout the document, lists and sublists of requirements should be indented and numbered to make it easy to read and reference the specification details. Such as:

1. *General Requirement*

1.1. *Specific Requirement*

1.1.1 *Requirement Detail*

Note how this permits reference to “SRS Item 1.1.1”.

## External Interfaces (Inputs and Outputs)

This section should describe inputs into and outputs from the software system. (ISO/IEC/IEEE 29148:2011)

Each interface description should include the following:

1. Name of item.

2. Description of purpose.

3. Source of input or destination of output.

4. Valid ranges of inputs and outputs.

5. Units of measure.

6. Data formats.

## Functions

Define the actions that must take place in the software to accept and process inputs and generate outputs (ISO/IEC/IEEE 29148:2011). These definitions must include:

1. Validity checks on the inputs.

2. Sequence of operations in processing inputs.

3. Responses to abnormal situations, including error handling and recovery.

4. Relationship of outputs to inputs, including

(a) input/output sequences

(b) formulas for input-output conversion

## Usability Requirements

Define usability requirements and objectives for the software system, including measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use. (ISO/IEC/IEEE 29148:2011)

## Performance Requirements

Specify the static and dynamic numerical requirements placed on the software or human interaction with the software. For example: (a) Static numerical requirements may include the amount and type of information the system processes. (b) Dynamic numerical requirements may include the amount of data processed within specific periods.

State performance requirements 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” (ISO/IEC/IEEE 29148:2011).

## Software System Attributes

Specify the required attributes of the software product, such as reliability, security, privacy, maintainability, or portability. (ISO/IEC/IEEE 29148:2011) Review a comprehensive list of software attributes or software qualities, such as are provided in van Vliet (2008) Chapter 6. Decide on a relatively small number of the most important attributes for this system. Explain why each attribute is important and what steps you will take to achieve those attributes. The attributes include constraints on attributes of the system’s static construction, such as testability, changeability, maintainability, and reusability. (Faulk, 2013)

# References

This section lists the sources cited in the creation of this template document. An SRS should reference all of the sources that it draws from. This section may not be necessary if sufficient citations are provided “inline” (at the point of reference) in the document.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. <https://ieeexplore.ieee.org/document/761853>

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. <https://ieeexplore.ieee.org/document/720574>

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. <https://ieeexplore.ieee.org/document/6146379>

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. <https://ieeexplore.ieee.org/document/8559686>

Faulk, Stuart. (2013). *Understanding Software Requirements*. <https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf>

Oracle. (2007). White Paper on “Getting Started With Use Case Modeling”. Available at: <https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf>

Sethi, Ravi. (2023). *Software Engineering. Basic Principles and Best Practices*. Cambridge Press.

Work Breakdown Structures. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Work_breakdown_structure>.

N2 Charts. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/N2_chart>.

Functional Flow Block Diagrams. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Functional_flow_block_diagram>.

Structure Chart. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Structure_chart>.

Data-flow Diagram. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Data-flow_diagram>.

Object Diagram. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Object_diagram>.

System Context Diagram. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/System_context_diagram>.

Storyboard. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Storyboard>.

Entity Relationship Model. In *Wikipedia*, n.d. <https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model>.

# Acknowledgments

List here all sources you used to create the document and support you received from anyone not on your team.