IT Project Guidance

On User Requirement Development

Version:

0.1

## Purpose

This document provides a structured approach to developing user requirements that align with business outcomes and stakeholder roles. It supports the elicitation, validation, and documentation of user needs through the use of structured use cases.

## Audience

This guidance is intended for professionals involved in the design, delivery, or evaluation of information systems. It is especially relevant to those responsible for translating business goals into system capabilities, including architects, analysts, designers, and governance leads.

## Synopsis

User requirements follow business requirements and stakeholder mapping. While business requirements define outcomes and stakeholder maps define roles, user requirements describe what users must do to achieve those outcomes. Because users often struggle to articulate their needs for unfamiliar systems, this guidance uses a comprehensive set of use cases to elicit discussion, uncover omissions, and refine understanding—ultimately supporting the completion of user requirements.

## Contents

[Purpose 1](#_Toc202426082)

[Audience 1](#_Toc202426083)

[Synopsis 1](#_Toc202426084)

[Contents 2](#_Toc202426085)

[Context 4](#_Toc202426086)

[Preparation 4](#_Toc202426087)

[Origins and Scope 4](#_Toc202426088)

[Warnings and Misconceptions 5](#_Toc202426089)

[Avoid Over-Reliance on Existing Systems as Templates 5](#_Toc202426090)

[Don’t Assume Users Know What’s Possible 5](#_Toc202426091)

[Beware of Tool Bias 5](#_Toc202426092)

[Watch for Role Confusion 6](#_Toc202426093)

[Don’t Confuse Features with Requirements 6](#_Toc202426094)

[Avoid Premature Technical Constraints 6](#_Toc202426095)

[Avoid CRUD Thinking 6](#_Toc202426096)

[Categorised Use Cases 7](#_Toc202426097)

[Pre Use 7](#_Toc202426098)

[Account Management 7](#_Toc202426099)

[Identity & Manage Access 7](#_Toc202426100)

[Configuration & Setup 8](#_Toc202426101)

[Discovery & Navigation 8](#_Toc202426102)

[Authorisation and Access 8](#_Toc202426103)

[Collaboration & Participation 8](#_Toc202426104)

[Content Creation & Management 9](#_Toc202426105)

[Consumption & Feedback 10](#_Toc202426106)

[Import/Export & Integration 10](#_Toc202426107)

[Monitoring & Governance 10](#_Toc202426108)

[Lifecycle & Archival 11](#_Toc202426109)

[Reporting & Evaluation 11](#_Toc202426110)

[Appendices 12](#_Toc202426111)

[Appendix A - Document Information 12](#_Toc202426112)

[Versions 12](#_Toc202426113)

[Images 12](#_Toc202426114)

[Tables 12](#_Toc202426115)

[References 12](#_Toc202426116)

[Review Distribution 12](#_Toc202426117)

[Audience 12](#_Toc202426118)

[Structure 12](#_Toc202426119)

[Diagrams 13](#_Toc202426120)

[Acronyms 13](#_Toc202426121)

[Terms 13](#_Toc202426122)

[Appendix B – Original List 14](#_Toc202426123)

# Context

User Requirements are a critical phase in the development of any system, but they do not stand alone. They follow the establishment of **Business Requirements** and a **Stakeholder Map**.

* **Business Requirements** define the drivers, needs, and expected outcomes of a system. They articulate *why* the system is needed and *what* it must achieve.
* The **Stakeholder Map** identifies *who* is involved in the development, delivery, change, and consumption of the system. It clarifies roles, responsibilities, and influence.

Once these are in place, **User Requirements** are developed to **elicit** the specific needs of system users—those who will interact with the system to deliver the outcomes defined in the business requirements. These requirements translate high-level goals into actionable capabilities and functions.

# Preparation

Approaching stakeholders with open-ended questions like “What do you need from the new system?” often leads to incomplete or vague responses. Most users struggle to articulate what they currently do—let alone envision what they might do in a future system they’ve never seen.

Imagine asking someone who drives a manual transmission car to describe their needs for a self-driving flying quadcopter. The gap in familiarity and context makes it nearly impossible to yield meaningful insights.

That’s why **use cases** are essential. They provide concrete scenarios that help users react, reflect, and refine their needs. Use cases serve as a **framework for elicitation**—not only prompting users to describe what they do, but also helping them spot what’s missing.

# Origins and Scope

The use cases provided in this document were developed by synthesising insights from several standards and frameworks, including:

* **ISO/IEC 25010** (System and software quality models)
* **ISO/IEC 15288** (System life cycle processes)
* **ISO/IEC 38500** (Governance of IT)
* **ISO 8000-51** (Data governance and policy exchange)

While not designed for autonomous system design, this list is intended to support the development and assessment of **information systems** across various media and modalities.

It draws inspiration from the activities permitted in well-known tools and environments such as Notepad, Word, Draw.io, Paint.NET, web browsers, AutoCAD, and QGIS. These systems—ranging from simple text editors to complex spatial design platforms—offer diverse interaction models. Concepts like layers, transparency, and rotation, common in higher-dimensional tools like AutoCAD or Draw.io, may offer fresh perspectives when applied to more traditional systems like Word or Confluence.

This cross-pollination of ideas encourages designers to think beyond the obvious and consider how advanced interaction patterns might enrich even the simplest systems, to provide higher value usability.

# Obligations, Recommendations, and Prohibitions

When drafting user requirements, it is essential to understand the nature of the statements being made. Requirements are not casual suggestions—they are contractual expressions of what a system must, should, or must not do. This section outlines how to frame requirements clearly and responsibly.

## SMART Criteria

The acronym **SMART** stands for **Specific**, **Measurable**, **Achievable**, **Relevant**, and **Time-bound**. These criteria are commonly used to evaluate goals and objectives.

While not every user requirement needs to be SMART in the strictest sense, the **Specific** and **Measurable** aspects are especially important. Requirements should be clear enough to avoid ambiguity and structured in a way that allows for validation.

Avoid vague or compound statements. For example, “The system shall allow users to upload *and* edit documents” is not specific enough. Instead, separate the actions: “The system shall allow users to upload documents” and “The system shall allow users to edit uploaded documents.”

## MoSCoW Method

The **MoSCoW** method is often used to prioritise requirements. It stands for:

* **Must** – Obligations
* **Should** – Recommendations
* **Could** – Permissions
* **Won’t** – Prohibitions

However, it is important to understand that these categories are not just prioritisation labels—they represent **contractual intent**. Using the correct terminology helps clarify whether a requirement is mandatory, optional, or explicitly excluded.

## Language Guidance

To ensure clarity and enforceability, the following language practices are recommended:

* **Avoid “Should”** – Requirements phrased as “should” are rarely implemented. Prefer “must” for obligations or “must not” for prohibitions.
* **Avoid “All”** – The word “all” is inherently ambiguous. It invites disagreement and misinterpretation. Instead, specify exactly what is included.
* **Avoid “Will” and “Shall”** – These terms are often used interchangeably and imprecisely. They lack clarity about timing and enforceability, and have been the subject of litigation in many contractual contexts.
* **Avoid “And” in requirement statements** – Compound requirements are not specific and violate the SMART principle. Break them into separate, testable statements.
* **Use “If” sparingly** – Conditional requirements can be useful but should be used with care. Overuse leads to complexity and confusion. When used, ensure the condition and outcome are both clearly defined and remain singular.

By following these principles, requirement authors can develop user requirements that are not only clear and actionable, but also aligned with governance, traceability, and system design best practices.

# Warnings and Misconceptions

When eliciting user requirements, it’s important to be aware of common misconceptions that can lead to incomplete, misleading, or technically inappropriate specifications. The following warnings are intended to help guide more accurate and future-proof requirement development.

## Avoid Over-Reliance on Existing Systems as Templates

While referencing familiar systems (e.g., Word, Draw.io, AutoCAD) can be helpful, it’s important not to constrain future system design by replicating legacy workflows. Use cases should reflect **user goals**, not just mimic existing interfaces. Innovation is stifled when requirements are reactive rather than visionary.

## Don’t Assume Users Know What’s Possible

Users often describe what they know, not what they need. They may not be aware of newer paradigms like predictive analytics, adaptive interfaces, or AI-assisted workflows. Use cases should be used to **elicit reactions and expand thinking**, not just confirm existing habits.

## Beware of Tool Bias

Stakeholders often describe what they want based on the tools they currently use. This can lead to requirements that are tool-specific rather than functionally universal. For example, “I need a spreadsheet” might actually mean “I need a tabular data entry and analysis capability.” Requirements should be framed around capabilities, not brand names or UI metaphors.

## Watch for Role Confusion

Stakeholders may speak from multiple roles—user, manager, approver—without clarifying which perspective they’re representing. Be explicit about **which role** each requirement is coming from, especially when mapping to personas or access levels.

## Don’t Confuse Features with Requirements

A feature is a way to implement a requirement—not the requirement itself. For instance, “I want drag-and-drop” is a feature request. The underlying requirement might be “I need to reorganise items quickly and intuitively.” Always probe for the **why** behind the feature.

## Avoid Premature Technical Constraints

User requirements should be technology-agnostic. Introducing constraints like “must be mobile-first” or “must use blockchain” too early can limit solution space and bias design. These belong in architectural or implementation discussions—not in initial user requirement elicitation.

## Avoid CRUD Thinking

The acronym **CRUD**—Create, Read, Update, Delete—is often used as shorthand for user interactions. However, CRUD originates from data storage operations, not user experience design. It describes how databases manipulate data storage, not how users interact with system data.

“Delete”, in particular, is problematic. In most modern systems—especially those involving governance, auditability, or compliance—*physical* deletion should not be a user-facing operation. Instead, systems should support *logical* deletion, where a record’s state or visibility is changed without removing it from the underlying data store.

To better reflect user-facing interactions:

* **CRUST** – *Create, Retrieve, Update, State Transition*  
  This model acknowledges that data should move through states (e.g., active → archived → retired) rather than be deleted outright.
* **BREAD** – *Browse, Retrieve, Edit, Add, Delete*  
  Common in UI design, BREAD captures typical interface actions. However, the Delete operation should be interpreted as a logical state change, not a physical deletion—best understood through the lens of CRUST.

These models help ensure that system operations are designed with governance, auditability, and user experience in mind.

## Disambiguate User versus System Functional Requirements

It is important not to disambiguate User Requirements from the System Functional Requirements that later get developed from them. User Requirements describe what the user wants to achieve, often stated in broad, goal-oriented terms and natural language. System Functional Requirements, however, specify the exact system behaviours and operations needed to fulfil those user needs. Treating them as the same can lead to misinterpretation, incomplete specifications, or solutions that technically function but fail to meet user expectations. Always distinguish between what the user wants and how the system will deliver it. Examples of the difference are provided below.

**User Requirement:** Users must be able to monitor the progress of their projects in a way that is clear, accessible, and updated regularly.

**System Functional Requirement:** The system must provide a dashboard that displays project milestones and current status using visual indicators.

# Categorised Use Cases

This section presents a categorised list of use cases to support the development of user requirements. By grouping use cases into functional categories, stakeholders can more effectively identify, elicit, and validate requirements based on real-world interactions and system capabilities. This structure also facilitates completeness checks and alignment with governance frameworks and lifecycle models.

## Pre Use

A number of use cases are performed before system end users.   
These include:

**register system**: Register a service or system for discovery, such as DNS.

**provision users**: Assign users to roles or groups within the system (prefer by Invitation).

**configuration**: Set up discovery, integrations, system settings.

**setup**: manage mutable configuration – system settings, common reference data/code sets, defaults,

**deploy system**: Set up the system in its operational environment.

**restore system data**: Recover previously saved system data.

**backup system data**: Save system data regularly.

**migrate to system**: Transfer records and resources from previous systems.

**transition**: Move from one system or state to another.

**retire service**: Decommission a service or resource.

## Account Management

accounts, and user profiles

## Identity & Manage Access

**sign up**: Create a new user account within the system.

**apply**: Request to join a system or group with a specific role.

**invite**: Send an invitation to a user to join a group or role.

**accept**: Accept an invitation to join a group or role.

**log in**: Authenticate and access the system.

**identify**: Distinguish requests as separate sessions.

**authenticate**: Verify the identity of a user for a session.

**onboard**: Introduce users to the system and its functions.

**authorise**: Grant permissions to users for activities on resources.

## Configuration & Setup

**setup**: configure mutable personal profiles

**describe**: Add metadata for organisational categorisation and versioning.

**group/folder**: Organise resources into folders or groups.

**categorise**: Classify resources based on attributes or taxonomy.

**link**: Connect resources within the system.

**unlink**: Remove connections between resources.

**transfer**: Move resources between registries or classifications.

**personalise/depersonalise**: Link or unlink resources from identifiable personas.

## Discovery & Navigation

**search**: Find resources using criteria inclusive of filtering, ordering, projecting, and paging.

**navigate**: Move along immutable (system) and mutable paths (linkages), without performing a search.

**route**: Transparently return logically selected versions (e.g. latest) of resources or records.

## Authorisation and Access

Many use cases relating to resources mirror use cases to the whole system.

**Apply**: apply to the document owners to contribute to the document in some form or another.

**Invite**: invite collaborators to roles within the resource or group.

**accept**: accept invitee to role.

**reject**: reject application

**authorise**: Grant permissions to users for activities on resources.

## Collaboration & Participation

**schedule**: Plan meetings or tasks.

**subscribe/enrol**: Join a meeting or service.

**excuse**: Indicate non-attendance for a meeting or event.

**attend**: Participate in a meeting or collaborative task.

**assign**: Delegate tasks to users.

**reminding**: Send prompts or reminders to attend or complete tasks.

**track progress**: Monitor the status of task groups or workflows.

**mark**: Indicate completion of task parts.

**contribute**: Add content or input to existing records.

**comment (for contribution)**: Provide feedback or suggestions on contributions.

**endorse**: Support a resource or action as an organisation.

**accredit**: Formally recognise a resource or contributor.

## Content Creation & Management

**create**: Generate new records or resources.

**do/undo:** preserve and manipulate stack of operations and resulting temporal versions

**select:** select, deselect

**embed**: Insert media into records.

**include**: Add references to other resources.

**layer**: add, remove, reorder layers.

**format**: Change the presentation of content parts.

**style**: Apply formatting to content types (content size, background, border)

**transform:** size, rotate, skew, colorise, transparency, clip

**mask**: affect other elements

**compare**: Evaluate differences between versions or records.

**flag for review**: Mark a resource for governance or compliance checks.

**review**: Assess the quality or completeness of a resource.

**validate**: Approve specific aspects of a resource.

**approve**: Confirm the resource meets required standards.

**reject**: Send a resource back for revision.

**consent**: Grant access to personal information within resources.

**remove consent**: Revoke access to personal information.

**correct**: Fix errors or inaccuracies in resources.

**maintain**: Ensure resources remain accurate and up-to-date.

**copy**: Duplicate a resource within the system.

**merge**: Combine multiple records into a new version.

**unmerge**: Split records into separate versions.

**replace**: Substitute one resource with another.

**remove/unpublish**: Withdraw a resource from public access.

**localise**: Adapt content for different cultures or languages.

## Consumption & Feedback

**consume**: Use or view a resource.

**like**: Indicate approval or preference for a resource.

**watch**: Subscribe to alerts for changes to resources.

**un-watch**: Unsubscribe from alerts for changes to resources.

**rate**: Assign a score or rating to a resource.

**comment (on consuming)**: Provide feedback on consumed resources.

**share/recommend (to other users)**: Suggest resources to internal users.

**share/recommend (to external personas)**: Suggest resources to external stakeholders.

**message**: Send a message related to resource creation.

**notify**: Alert users to changes or updates.

## Import/Export & Integration

**import**: Bring resources or records into the system.  
**export**: Send resources or records out of the system.  
**download**: Retrieve a copy of a resource for offline use.  
**hyperlink**: Link to external resources within text.  
**localise**: Adapt content for different cultures or languages.

## Monitoring & Governance

**monitor**: Track errors, activity, and resource consumption.

**scan**: Check for viruses, broken links, or objectionable media.

**disable**: disable user

**optimise**: Improve performance or availability based on scans.

**cleanse**: Remove identifiable information from resources.

**periodic cleanse**: Delete expired records or users.

**trace**: Link requirements to business needs and system features.

**audit**: Review changes and access for compliance.

**report**: Generate summaries on system, resources, or users.

## Lifecycle & Archival

**release/publish**: Make a resource publicly available.  
**archive**: Store resources for long-term access.  
**delete**: Remove a resource logically from the system.  
**destroy**: Physically eliminate a resource.

## Reporting & Evaluation

**survey**: Collect structured feedback from users.

**retrospect**: Review past actions for lessons learned.

**measure/grade**: Evaluate the quality or outcome of a task.

**benchmark**: Compare performance against standards.

Appendices

Appendix A - Document Information

Authors & Collaborators

* Sky Sigal, Solution Architect

### Versions

* 1. Initial Draft

### Images

**No table of figures entries found.**

### Tables

[Table 1: TODO Table 3](#_Toc145048484)

[Table 2: TODO Table 2 3](#_Toc145048485)

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Russell Campbell, Technical Project Manager |  |
| Carrie Buckmaster, Senior Business Analyst |  |
|  |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Acronyms

API

: [Application Programming Interface](#Term_ApplicationProgrammingInterface).

**DDD**

: Domain Driven Design

GUI

: [Graphical User Interface](#Term_ApplicationProgrammingInterface). A form of [UI](#Acronym_UI).

ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. [IT](#Acronym_IT) is a subset of ICT.

IT

: acronym for Information, using Technology to automate and facilitate its management.

SMART

: Singular, Measurable, Achievable, Relevant, Time/Resource constrained.

UI

: User Interface. Contrast with [API](#Acronym_API).

### Terms

Refer to the project’s Glossary.

Application Programming Interface

: an Interface provided for other systems to invoke (as opposed to User Interfaces).

Capability

: a capability is what an organisation or system must be able to achieve to meet its goals. Each capability belongs to a domain and is realised through one or more functions that, together, deliver the intended outcome within that area of concern.

Domain

: a domain is a defined area of knowledge, responsibility, or activity within an organisation or system. It groups related capabilities, entities, and functions that collectively serve a common purpose. Each capability belongs to a domain, and each function operates within one.

Entity

: an entity is a core object of interest within a domain, usually representing a person, place, thing, or event that holds information and can change over time, such as a Student, School, or Enrolment.

Function

: a function is a specific task or operation performed by a system, process, or person. Functions work together to enable a capability to be carried out. Each function operates within a domain and supports the delivery of one or more capabilities.

Person

: a physical person, who has one or more Personas. Not necessarily a system User.

Persona

: a facet that a Person presents to a Group of some kind.

Quality

: a quality is a measurable or observable attribute of a system or outcome that indicates how well it meets expectations. Examples include reliability, usability, and performance. Refer to the ISO-25000 SQuaRE series of standards.

User

: a human user of a system via its UIs.

User Interface

: a system interface intended for use by system users. Most computer system UIs are Graphics User Interfaces ([GUI](#Acronym_GUI)) or Text/Console User Interfaces (TUI).

Appendix B – Original List

The following is a set of use cases to consider when developing User requirements, which in turn lead to clarification of required capabilities and thereby functions.

**To delete once checked.**

register system (service, for discovery, e.g. DNS),

deploy system (not a system function, but an environment function),

migrate to system (onboarding of records & resources from previous systems),

restore system data (if previously saved),

backup system data (regular activity),

transition,

sign up,

provision users,

apply (to join a system or group [as a role]),

invite (to join as role in group),

accept (invitation)

log in,

configure (discovery, integrations, system, account, user profiles),

onboard (users),

identify (requests as different sessions),

authenticate (user of session),

authorise (user to activities on resources),

search (criteria, order, project, page),

route (to and in between versions of resources and records),

navigate (distinct from search),

import (resources and records),

schedule (meetings or tasks),

subscribe/enrol (to meeting, to service)

excuse (will not attend),

meet (to contribute to a task),

assign (tasks),

reminding (bump/pre or post prompting, e.g. to attend or complete),

create (new records),

contribute (to created records),

embed (media),

include (references),

format (change presentation of content parts)

style (changing format of content types),

compare (against another version or record),

flag for review,

review,

comment (for contribution),

validate (approve of an aspect, not the same as overall resource approval),

approve,

reject (send back for more work),

consent (give access to personal information within resources to specific groups of users)

remove consent,

trace,

describe (organisational categorisation and versioning metadata),

group/folder (classification),

categorise,

release/publish,

link (to local resources),

unlink,

transfer (between in-system registries – a change of classification)

hyperlink (e.g., within text, to external resources),

track progress (of task groups),

mark (completed task parts),

measure/grade (quality of result of task)

export,

consume,

like,

rate,

message (specific to create),

notify (distinct from share),

comment (on consuming),

endorse (as an organisation),

accredit (uncertain if distinct from endorsing),

watch (to be alerted to change on resource, metadata, ratings, comments),

un-watch,

share/recommend (to other users,

share/recommend (to external personas),

correct,

maintain,

copy (resource, within system),

localise (for different cultures and languages),

download (copy resource, off-system),

merge (two or more records together into new version),

unmerge (create two or new versions of records from records prior to current one)

replace,

remove/unpublish,

survey (collect structured feedback),

retrospect,

depersonalise (unlinking from identifiable persona),

periodic cleanse (remove expired records of users, resources, etc.)

benchmark (speeds),

scan (for viruses, for broken hyperlinks, for broken links, for objectionable media),

optimise (based on scans, for performance for availability)

cleanse (of identifiable information),

monitor (errors, suspicious activity, public activity, user activity, resource consumption)

audit,

archive,

delete (logical),

destroy (physical – note: no reason to do this if depersonalised),

report (on system, on resources, on users),

retire service,