**<Project Name>**

**User Requirements**

**Author (s):** <Author name(s)>

**Version:** <Version number>

**Date:** <Date document released>

<Copyright information, if required, here>

**READ ME: Guidance**

1. The user requirements must be specified and documented. They are a statement of **what** is required from the software, **not how** it is going to meet these requirements.

The iterative process may mean these requirements are modified as the software develops, particularly if the requirements gathered originally were incomplete, unreasonable or infeasible.

1. This document should be understandable by someone who will want to check that it captures **what** is required and that the software meets these requirements. This person should **not** have to be concerned with **how** the software works.
2. For a lot of software this document can be comparatively short and consist largely of text or images (with a few equations). Larger and more complicated software may require a longer, more in-depth document, e.g. a document containing [Unified Modelling Language (UML)](https://www.uml.org/) use case diagrams.
3. The user requirements must be **traceable** from this document through the functional specification to the code and test plan; i.e. it must be easily identifiable which piece of code implements which user requirement and which test relates to which user requirement.

For a lot of software, traceability can be achieved by referring to a particular sentence in a particular paragraph. For larger and more complicated software, labelling or numbering the requirements will aid traceability.

**After writing this document…**

1. **Obtain customer review.** An email saying “Yes, that is what I need” may be sufficient. Save this email, and the others mentioned below, to the location where the code and documentation is held (e.g. GitLab repository).
2. **Obtain review by the project team** (not compulsory for Software Integrity Level (SWIL) 1 but recommended). For a lot of software an email trail saying “Yes, that is what we should be developing” may be sufficient. For larger and more complicated software, a checklist can help with review.
3. For **Software Integrity Level 3** and above **obtain review by a suitably qualified** independent person. “Independent” means this person **must not** be a member of the project team. For a lot of software an email saying “Yes, that is what you should be developing” may be sufficient. For larger and more complicated software, a checklist can help with review.
4. **Move on to the next stage** in the iterative development lifecycle, which is developing the **functional specification** (or modifying an existing functional specification, if this is not the first iteration of the development lifecycle).

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Date** | **Author** | **Change(s)** |
| Draft v1 | 30/09/2022 | K. Lines | INITIAL VERSION |
| Draft v2 | 14/11/2022 | K. Lines | Removed logos, added disclaimer. |
| Draft v3 | 14/12/2022 | K. Lines | Feedback from Mathmet chair. Modifications to acknowledgements. |
| Draft v4 | 20/12/2022 | K. Lines | Added DRAFT watermark. |

**Disclaimer**

The Quality Assurance Tools for data, software and guidelines have been provided by the

Members and Partners of the European Metrology Network for Mathematics and

Statistics (Mathmet). EURAMET has no influence on its correctness and completeness

and does not assume any liability for it.

The European Metrology Network for Mathematics and Statistics is supported by the

Joint Network Project ‘Support for a European Metrology Network for mathematics and

statistics’ (18NET05).

**Acknowledgements**

The European Metrology Network for Mathematics and Statistics is supported by the Joint Network Project ‘Support for a European Metrology Network for mathematics and statistics’ (18NET05 MATHMET). The project 18NET05 MATHMET has received funding from the EMPIR programme co-financed by the Participating States and from the European Union’s Horizon 2020 research and innovation programme.

**Contents**

[1. Introduction 2](#_Toc119501265)

[1.1 Purpose 2](#_Toc119501266)

[1.2 References 3](#_Toc119501267)

[2 Definitions 3](#_Toc119501268)

[3 SWIL Assessment 3](#_Toc119501269)

[4 Reasons system is needed 3](#_Toc119501270)

[5 Problem(s) to be solved 3](#_Toc119501271)

[6 Outline of activities the software will carry out 3](#_Toc119501272)

[7 Intended environment 3](#_Toc119501273)

[8 Estimate of effort required 3](#_Toc119501274)

[9 Acceptance criteria 4](#_Toc119501275)

[10 Cut-off date for bug reporting 4](#_Toc119501276)

[11 Maintenance of software 4](#_Toc119501277)

# **Introduction**

## **Purpose**

* What is the aim of this document? Obviously one aim is to list the user requirements! Is there anything else this document is going to provide, e.g. background information on the project for which this software is being developed?
* **If it will be useful to label requirements** then explain the labelling scheme here, **e.g.** “User requirements will be labelled **[UR:****<label>]**, where **<label>** is a very short descriptive label, and shaded green. Particular points to be noted, that are not user requirements, are shaded grey.”

## **References**

* Insert references to any relevant documents.

# **Definitions**

If necessary, insert a table explaining what any abbreviations or technical terms specific to this software mean. **If there’s a project acronym, explain what it stands for!**

# **SWIL Assessment**

If the software quality management plan needs supplementing with further justification for the Software Integrity Level (SWIL), insert this justification here.

# **Reasons system is needed**

* A very brief summary of why the system is required, e.g. is it needed to replace existing but obsolete and difficult to maintain software?

# **Problem(s) to be solved**

* A definition of the problem(s). Keep this definition as **concise** as possible.

# **Outline of activities the software will carry out**

* A high-level statement of what the software will do.
* Remember that this document **shall be understandable** by someone who does **not** need to know (or even care) **how the software works**.

# **Intended environment**

* A high-level statement of the hardware, operating systems and other software required to run this software. E.g.:
  + Is this software intended to run on an MS-Windows laptop with MS-Office installed?
  + Is it intended for a high-performance computing environment (HPCE)?
  + Is it intended for a cloud-based or real-time environment?
* Specific details of details of the hardware and software required to run this software shall be provided in the **operational platform** section of the **functional specification**.

# **Estimate of effort required**

**NOTE:** If this software is being developed as part of a project for which resources have already been defined, just reference the relevant project document; there is no need for repetition. **If in doubt, ask the project manager.**

# **Acceptance criteria**

Criteria that the software must satisfy in order to be accepted by a user, customer, or other authorized entity. E.g.:

* Documented testing against user requirements.
* Passing of acceptance testing by the customer.

# **Cut-off date for bug reporting**

Will there be a cut-off date for bug reporting? If there will not be a cut-off date, say so.

# **Maintenance of software**

* How is the software going to be maintained? E.g.:
  + How are bug reports and requests for new or modified user requirements going to be managed?
  + How will adapting to a changed environment (e.g. an operating system upgrade, a new version of LabVIEW, MATLAB, MS-Office, Python etc. or new hardware) be managed?
  + If the maintenance plan is defined in the software quality management plan then just refer to this plan