Software Evaluation: Criteria-based Assessment

Criterion	Notes – to what extent is/does the software…
Understandability	Easily understood?
Documentation	Comprehensive, appropriate, well-structured user documentation?
Learnability	Easy to learn how to use its functions?
Accessibility	Evidence of current/future ability to download?
Portability	Usable on multiple platforms?
Analysability	Easy to understand at the source level?

The rest of this document covers each category in greater depth, with lists of questions that we use at the Software Sustainability Institute when compiling detailed software evaluation reports.

Understandability How straightforward is it to understand: • What the software does and its purpose? • The intended market and users of the software? • The software's basic functions? • The software's advanced functions?	**Yes/No, supporting comments if warranted** The documentation about this can be found in MuSa's GitHub page. It's structures in 3 main documents: Design, Architecture and Evaluation. It's easy to navigate and understand. A README provides a summary.
High-level description of what/who the software is for is available.	Yes.
High-level description of what the software does is available.	Yes.
High-level description of how the software works is available.	Yes.
Design rationale is available – why it does it the way it does.	Yes.
Architectural overview, with diagrams, is available.	Yes.
Descriptions of intended use cases are available.	Yes.
Case studies of use are available.	Yes.

Documentation Looking at the user documentation, what is its • Quality? • Completeness? • Accuracy? • Appropriateness? • Clarity?	**Yes/No, supporting comments if warranted** The documentation is very accurate: contains a lot of details and it's easy to navigate and understand. It addresses all the aspects of the project.
Provides a high-level overview of the software.	Yes.
Lists resources for further information.	Yes.
Further information is suitable for the level of the reader, for each class of user.	Yes, although you need to know a little about software engineering.
Is task-oriented.	Yes, but not completely.
Consists of clear, step-by-step instructions.	Yes, for the local deploy.
Gives examples of what the user can see at each step e.g. screen shots or command-line excerpts.	No.
For Java, the package names of classes are stated the first time a class is mentioned.	No, MuSa doesn't have a documentation so specific for code.
English language descriptions of commands or errors are provided but only to complement the above.	No.
Is on the project web site.	Yes, the GitHub page.

Learnability How straightforward is it to learn how to achieve: • Basic functional tasks? • Advanced functional tasks?	**Yes/No, supporting comments if warranted** The code doesn't have a dedicated documentation, but instructions for local deploy are provided. Knowing a little bit of coding, it's not so difficult to modify.
A getting started guide is provided outlining a basic example of using the software.	Yes.
Instructions are provided for many basic use cases.	Yes, to start things app.
Instructions are provided supporting all use cases.	No.

Accessibility To what extent is the software accessible?	**Yes/No, supporting comments if warranted**
Binary distributions are available (whether for free, payment, registration).	No.
Binary distributions are freely available.	No.

Binary distributions are available without the need for any registration or authorisation of access by the project.	No.
Source distributions are available (whether for free, payment, registration).	Yes.
Source distributions are freely available.	Yes.
Source distributions are available without the need for any registration or authorisation of access by the project.	Yes.
Access to source code repository is available (whether for free, payment, registration).	Yes.
Anonymous read-only access to source code repository.	Yes.
Ability to browse source code repository online.	Yes.
Repository is hosted externally to a single organisation/institution in a sustainable thirdparty repository (e.g. SourceForge, GoogleCode, LaunchPad, GitHub) which will live beyond the lifetime of any current funding line.	Yes.
Downloads page shows evidence of regular releases (e.g. six monthly, bi-weekly, etc.).	No.

Portability To what extent can the software be used on other platforms?	**Yes/No, supporting comments if warranted** We have developed an Android application.
Application can run under Android.	Yes.
Application can run under iOS.	No.

Analysability How straightforward is it to analyse the software's source release to: • To understand its implementation architecture? • To understand individual source code files and how they fit into the implementation architecture?	**Yes/No, supporting comments if warranted** The documentation provides a high level description of what is happening, and sometimes comments to code are provided. Anyway, variables and funcions have meaningful names, it's not so hard to understand what's happening.
Source code is structured into modules or packages.	Yes.
Source code structure relates clearly to the architecture or design.	Yes.
Project files for IDEs are provided.	No.

Source code repository is a revision control system.	Yes.
Source code is commented.	Yes, but not everywhere.
Source code uses sensible class, package and variable names.	Yes.
Source code is laid out and indented well.	Yes.
There is no commented out code.	
There are no TODOs in the code.	No.
Auto-generated source code is in separate directories from other source code.	Yes, there is not.