Flashcards - Sustainability Evaluation

This software evaluation report is for your software: Flashcards. It is a list of recommendations that are based on the survey questions to which you answered "no".

If no text appears below this paragraph, it means you must already be following all of the recommendations made in our evaluation. That's fantastic! We'd love to hear from you, because your project would make a perfect case study. Please get in touch (info@software.ac.uk)!

Question 2.1: Is the name of your project/software unique?

You shouldn't choose a project or software name that is shared by another group – especially if they are a competitor – but, sadly, few people spend enough time researching the uniqueness of their project or software names. It is, now, less important to have a domain name that mimics the name of your project or software, because most people will use a web search to find a website rather than manually entering a URL. However, it is important to check, using a web search, that there are no existing domains that include the name of your project or software that are owned by another group, because this can confuse your users.
See our guide on Choosing project and product names (http://software.ac.uk/resources/guides/choosing-project-and-product-names).

Question 2.2: Is your project/software name free from trademark violations?

When you choose a name, it's important to check that someone else hasn't already protected that name with a trademark. If they have, you run the risk of being sued for trademark infringement. Running a quick trademark search is free.
See our guide on Choosing project and product names (http://software.ac.uk/resources/guides/choosing-project-and-product-names).

Question 3.1: Is your software available as a package that can be deployed without building it?

Building software can be complicated and time-consuming. Providing your software as a package that can be deployed without building it can save users the time and effort of doing this themselves. This can be especially valuable if your users are not software developers.
You should test that your software builds and runs on all the platforms it is meant to support, which means you will already have created packages that can be distributed to your users!
See our guide on Ready for release? A checklist for developers (http://www.software.ac.uk/resources/guides/ready-release).
If you're interested in the consequences of ignoring your users' needs, see our guide on How to frustrate your users, annoy other developers and please lawyers (http://www.software.ac.uk/resources/guides/how-frustrate-your-users-annoy-other-developers-and-please-lawyers).

Question 3.4: Is your software hosted in an established, third-party repository likeGitHub (https://github.com), BitBucket (https://bitbucket.org),LaunchPad (https://launchpad.net) orSourceForge (https://sourceforge.net)?

Hosting your software, and documentation and related resources, in an established, third-party repository gives users confidence that if you disappear, or your project ends, then they at least have the means to be able to access, fix, improve or extend your software themselves. It also gives them the potential to make such changes when they need them, rather than having to wait for you to do it on their behalf.

Many third-party repositories now offer free project hosting, providing related resources including mailing lists, wikis and issue trackers. These can relieve you and your organisation of the cost and effort of deploying and hosting these resources.

See our guide on Choosing a repository for your software project (http://software.ac.uk/resources/guides/choosing-repository-your-software-project).

Question 4.5: Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?

Troubleshooting information helps users quickly solve problems. It can help to reduce the number of support queries that you have to deal with.

Question 5.2: Does your website and documentation describe what support, if any, you provide to users and developers?

The level of support that a user can expect to receive is often a vital element in a user's choice of software. This means that the support you provide – whether it's a guaranteed response in twenty-four hours, or a possible response on a best effort basis – needs to be made clear on your website and in your documentation.

This information can help manage users' expectations. A user will always want their problem to be solved as quickly as possible, and may become disgruntled (and might even stop being a user) if this is not the case. If you are clear and honest about the level of support you can provide, then you are more likely to keep your users happy.

See our guide on Supporting open source software (http://software.ac.uk/resources/guides/supporting-open-source-software). Its advice applies to supporting closed source software too.

Question 5.3: Does your project have an e-mail address or forum that is solely for supporting users?

E-mail is purpose-made for resolving users' problems. The user can provide a good description of their problem and can attach screenshots, log files or other supporting evidence, and it's easy for you to ask for follow-up information, if required. You should always try to have an e-mail address for support queries.

It's best if the support e-mail address is clearly labelled as such e.g. myproject-support@myplace.ac.uk. This makes it easy for users to identify the e-mail address on your website or within your documentation, and it helps you to separate your support queries from all of your other e-mail. However, a personal e-mail address is better than nothing if you don't have the means to provide a dedicated support address.

See our guide on Supporting open source software (http://software.ac.uk/resources/guides/supporting-open-source-software). Its advice applies to supporting closed source software too.

Question 5.5: Does your project have a ticketing system to manage bug reports and feature requests?

Dealing with one or two support queries is straightforward enough, but as the number of queries grows and their complexity increases, it gets easier to make mistakes. Nothing will annoy a user more than their support queries being ignored, even by accident.

A ticketing system allows you to organise support queries in a scalable way. It provides an easy method to record who asked what, and when, to store additional information about the query, to assign someone in a team to handle a specific query, and to prioritise queries so that you can work on the most important first. A ticketing system is an absolute requirement if you have more than one person working on the queries – it prevents two people accidently working to solve the same problem, and allows you to easily keep up to date with progress.

Ticketing systems are also very handy at providing statistics. You can find out the components of your software that cause the most problems, then improve them or better document their use. You can also see how quickly you are dealing with issues, and check that you are meeting the level of support that you have advertised.

Examples of ticketing systems include JIRA (https://www.atlassian.com/software/jira), Bugzilla (https://www.bugzilla.org/) and Trac (http://trac.edgewall.org/). Many third-party repositories, including GitHub (https://github.com), BitBucket (https://bitbucket.org), LaunchPad (https://launchpad.net) and SourceForge (https://sourceforge.net) also provide issue trackers.

See our guide on Supporting open source software (http://software.ac.uk/resources/guides/supporting-open-source-software). Its advice applies to supporting closed source software too.

Question 11.3: Do you provide publicly-available instructions for deploying your software?

If there are no instructions for deploying your software, how will your users deploy it? At best, you'll end up dealing with lots of queries about how to deploy your software. At worst, you'll get no queries, nor any users, as if they can't deploy it they can't use it. Unless your software is a standalone EXE file or a single Linux/UNIX executable, then you need to provide deployment instructions.

Question 12.2: Do you have a framework to periodically (e.g. nightly) run your tests on the latest version of the source code?

Having an automated build and test system is a solid foundation for automatically running tests on the most recent version of your source code at regular intervals e.g. nightly. At its simplest, this can be done by scheduling a cron job on Unix/Linux or Mac OSX, or using Windows Task Scheduler. A more advanced solution is to use a framework like Inca (http://inca.sdsc.edu/) which harnesses a number of machines through a central server to distribute a wide variety of tests in a parallel and scalable way.

See our guide on Testing your software (http://software.ac.uk/resources/guides/testing-your-software).

Question 13.1: Does your project have resources (e.g. blog, Twitter, RSS feed, Facebook page, wiki, mailing list) that are regularly updated with information about your software? (e.g. release announcements, publications, workshops, conference presentations)

These resources are all good ways of showing that your project is active. If potential users see frequent posts, especially if they talk about new features and resolved problems, they know that your project is thriving, your software is useful and is under active development. This may encourage them to use your software, knowing that if they run into problems, there may be people who can help, and who they can share experiences with.

These resources also give you a way of getting in touch with your current users, keeping them engaged, and asking for their input or help with problems. Don’t know what new feature to implement next? Ask your users whether they think it would be useful.

Question 13.2: Does your website state how many projects and users are associated with your project?

Where you have an active set of users and developers, advertising their existence is not just good for promoting the success and life of your project. If potential users see that there are a large number of users, they know that your project is thriving, your software is useful and is under active development. This may encourage them to use your software, knowing that if they run into problems, there may be people who can help, and who they can share experiences with.

Question 13.3: Do you provide success stories on your website?

A great way of showing off your software is to write case studies about the people who've used it and how they've used it. This helps potential users learn about the software but, more to the point, is a great advert for your software. If you can show happy users benefiting from your software, you are likely to gain more users.

Question 13.4: Do you list your important partners and collaborators on your website?

Providing a list of important partners and collaborators gives potential users valuable assurance that your software has a future. Also, the higher the scientific, academic or industrial reputation of those partners, the higher the perceived reputation of your software, and project, will be.
Publicly recognising partners' efforts in improving or working with your software also increases the likelihood they will continue to use, or develop, your software in the future. Credit where credit is due!

Question 13.5: Do you list your project's publications on your website or link to a resource where these are available?

Listing your software publications provides an academic perspective on the value of your software. It can also help users, and other stakeholders (e.g. current and potential funders) to understand, in detail, how your software contributes to research, what scientific problems it has helped to solve. In addition, these can help to show where your software sits in relation to other software that fulfils a similar need, and what makes yours different, or better.
These publications also gives researchers something to cite when they write their own papers where your software has been used, which is of value to them and also increases your citation count for your papers, which helps you demonstrate your impact!

Question 13.6: Do you list third-party publications that refer to your software on your website or link to a resource where these are available?

Providing a list of third-party publications can show, academically, how the software is used by others, as well as promoting their efforts and successes.
It also gives potential users ideas for how they may choose to use the software, as well as providing assurance that the software can be used by people other than its original developers to achieve something. Having such a list also means you can cite these publications in your own papers, funding proposals and reports to show or justify its value and the impact you have made!
As a matter of routine, you should always ask people to cite your software if they’ve used it in their research for these reasons, and to inform you if they have included such a reference in one of their

papers.

Question 13.7: Can users subscribe to notifications to changes to your source code repository?
Keeping information on these notifications as open as possible helps you present the impression that your project is open and inclusive. If users are actively developing using your software, keeping them up to date with on-going development on your source code (e.g. implementation of enhancements, extensions or bug fixes) enables them to factor these into their own development plans. For example, users might want to use an up-to-date copy from the repository to benefit from bug fixes made since your last release. If the notifications include information on the changes made (e.g. excerpts of the source code showing additions, removals and alterations) then this may allow for rapid identification of bug as any subscriber may notice an issue in the changes made.
A lightweight, automated subscription process can reduce, even remove, from you the overhead of managing notifications. It also means that users aren't subject to long delays waiting for their membership to be approved.

Question 16.2: Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?
Especially on academic projects, users will view the active lifetime of software to be the duration of the software's project funding. If you want to persuade users that your software will be around in the future, it is a good idea to describe your funding model and the duration over which funding is assured.

Question 16.3: Do you make timely announcements of the deprecation of components, APIs, etc.?
It's never a good idea to remove components or features without giving your users an advance warning first. It could be there are users who are dependent on the feature(s) you plan to change or remove. Announcing such planned deprecations well in advance means users and developers can respond if a given feature is important to them.
If a feature is due to be superseded by a newer, better feature or component, including both for a suitable period within the software can allow your users to transition comfortably from the older version to the new version.
You could also consider developing and publicising a deprecation policy, stating how and when features or components in general are deprecated. This gives your users assurance that features will not be removed without warning. see, for example the Eclipse API deprecation policy. (https://wiki.eclipse.org/Eclipse/API_Central/Deprecation_Policy).