Group 17

JTRACKER

JTracker is an open source job tracking software platform that eases the process of applying to jobs. With the help of JTracker, job seekers can easily keep all the information in one place. This includes tracking applications, wish lists, rejects and acceptances. JTracker also keeps a record of resume and skills for the person.

Why JTracker?

The process of applying for jobs and internships is not a cakewalk. Managing job applications is a time-consuming process. Due to the referrals and deadlines, the entire procedure can be stressful. Our application allows you to track and manage your job application process, as well as regulate it, without the use of cumbersome Excel spreadsheets.

Our application keeps track of the jobs you've added to your wish list. It also keeps track of the companies you've already applied to and keeps a list of any rejections. Rather than having the user browse each company's site for potential prospects, our application allows the applicant to search for them directly using basic keywords. Any prospective work offers can then be added to the applicant's wishlist.

What was Version i-1?

1. The application was a easy way to help job seekers keep track of their application process.
2. The seeker can track various job applications by segregating them by their status. (e.g.:- Waitlisted Jobs, Referral Jobs, Rejected Jobs, etc.)
3. A personalized profile page helps the job tracking enthusiasts keep a note of their own skills, preferred locations and experience levels and update them from time to time.
4. The application uses a Login and Signup page to keep the data of the applicant confidential.

What's new? (Version i)

1. We have added a functionality which uses a public API to fetch daily job postings from different aggregation platform to make you job searching process easy. (I also filters responses based on job seeker requirements).
2. Made a chrome extension which reduces the users time and efforts in filling applications. Now you don't have to type everything again and again. This extension will also act as a mini job tracker.
3. Added a new feature for recruiters, where they can state their candidate requirements, and we will connect them with their ideal matches.
4. We wrote MANY tests to make sure that our creation works well, and that you don't face any problems while finding a job :)

What's Next? (Version i+1)

1. Add a feature that allows users to attach interview reminders to their Google calendar.
2. Direct the connections to Linkedin, allowing for the addition of job opportunities to the wishlist.
3. Improve the recruiter dashboard to include filtering based on more features such as online assessments, education
4. Enhance chrome extension to allow injecting the data directly to the required fields
5. Make v2 of web scraping which collects data from more public APIS

Scan Me

OR USE

https://github.com/SoftwareEngineering-HomeWork/application-tracking-system

75+ Test Cases providing most code coverage over BACKEND and FRONTEND

Team Number: 17

Team Members:

Member 1: Dhruv Soni (dbsoni)
Member 2: Sweekar Burji (sburji)
Member 3: Prithish Samanta (psamant2)

The scores provided below represent the scores of the above team members.
Total Score: 89/96

GitHub Repo Link:
https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev

Notes	Score	Evidence
Workload is spread over the whole team (one team member is often X times more productive than the others but nevertheless, here is a track record that everyone is contributing a lot)	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/branches Team Member 1: 37 Team Member 2: 23 Team Member 3: 15 Although the number of commits are varying within the team, the contribution towards the modification of project is similar
Number of commits	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/branches
Issues reports: there are many	3	Issues Opened: 12 Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/issues
Issues are being closed	3	Issues Closed: 8 Link: https://github.com/SoftwareE

		ngineering-HomeWork/application-tracking-system/issues
Docs: doco generated, format not ugly	3	Hosted on GitHub Wiki Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/wiki
Docs: what: point descriptions of each class/function (in isolation)	3	Hosted on GitHub Wiki Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/wiki
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	3	Hosted on GitHub Wiki Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/wiki
Docs: why: docs tell a story, motivate the whole thing, deliver a punchline that makes you want to rush out and use the thing	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Use of version control tools	3	Git version control tool
Test cases exist	3	https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev/new_backend/tests https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev3/frontend/src/test
Test cases are routinely executed	3	https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/actions/workflows/test.yml

Issues are discussed before they are closed	3	https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/issues/7 https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/issues/3
Chat channel: exists	3	Link: https://drive.google.com/drive/folders/1gEsfjf1j71LeNB8p5RnAMCgo95D4YSy7?usp=sharing
Test cases: a large proportion of the issues related to handling failing cases.	2	
Evidence that the whole team is using the same tools: everyone can get to all tools and files	3	Team has used the same codebase, tools and common config tools
Evidence that the whole team is using the same tools (e.g. config files in the repo, updated by lots of different people)	3	Team has used the same codebase, tools and common config tools
Evidence that the whole team is using the same tools (e.g. tutor can ask anyone to share screen, they demonstrate the system running on their computer)	3	Team has used the same codebase, tools and common config tools
Evidence that the members of the team are working across multiple places in the code base	2	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/branches
Short release cycles	3	Project has frequent commits by all the team members
The file .gitignore lists what files should not be saved to the repo. See examples:	3	Link: https://github.com/SoftwareEngineering-HomeWork/applic

https://github.com/github/gitignore		ation-tracking-system/blob/dev/.gitignore
The file INSTALL.md lists how to install the code	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/INSTALL.md
The file LICENSE.md lists rules of usage for this repo	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/LICENSE
The file CODE-OF-CONDUCT.md lists rules of behavior for this repo; e.g. see example	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/CODE_OF_CONDUCT.md
The file CONTRIBUTING.md lists coding standards and lots of tips on how to extend the system without screwing things up; e.g. see example	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/Contributing.md
The file README.md contains all the following		
Video	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/main/resources/ApplicationTrackingAnimation.gif
DOI badge: exists.	3	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Badges showing your style checkers	2	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Badges showing your code formatters.	3	Link: https://github.com/SoftwareE

		ngineering-HomeWork/application-tracking-system/tree/dev#readme
Badges showing your syntax checkers.	1	
Badges showing your code coverage tools	3	Link: https://codecov.io/gh/SoftwareEngineering-HomeWork/application-tracking-system
Badges showing any other Other automated analysis tools	1	
Questionnaire		
Does your website and documentation provide a clear, high-level overview of your software?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Does your website and documentation clearly describe the type of user who should use your software?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Do you publish case studies to show how your software has been used by yourself and others?	No	We have defined the use-cases, however we have not published any case study
Is the name of your project/software unique?	No	However, we did not find any evidence for JTracker to be trademarked
Is your project/software name free from trademark violations?	Yes	MIT Licensed and contacted the early contributors for usage
Is your software available	Yes	Chrome Extension can be

as a package that can be deployed without building it?		packaged and made available on Chrome Store
Is your software available for free?	Yes	MIT Licensed
Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository?	Yes	MIT Licensed Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev
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)?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev
Is your documentation clearly available on your website or within your software?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Does your documentation include a "quick start" guide, that provides a short overview of how to use your software with some basic examples of use?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
If you provide more extensive documentation, does this provide clear, step-by-step instructions on how to deploy and use your software?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Do you provide a	Yes	Link: https://github.com/SoftwareE

comprehensive guide to all your software's commands, functions and options?		ngineering-HomeWork/application-tracking-system/wiki
Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?	No	
If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?	Yes	GitHub Wiki has the documentation for Backend APIs Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/wiki
Do you store your documentation under revision control with your source code?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Do you publish your release history e.g. release data, version numbers, key features of each release etc. on your web site or in your documentation?	No	
Does your software describe how a user can get help with using your software?	Yes	
Does your website and documentation describe what support, if any, you provide to users and developers?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/Contributing.md
Does your project have an e-mail address or forum	Yes	

that is solely for supporting users?		
Are e-mails to your support e-mail address received by more than one person?	Yes	
Does your project have a ticketing system to manage bug reports and feature requests?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/issues
Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/issues
Is your software's architecture and design modular?	Yes	MVC Architecture
Does your software use an accepted coding standard or convention?	Yes	MVC Architecture
Does your software allow data to be imported and exported using open data formats? e.g. GIF, SVG, HTML, XML, tar, zip, CSV, JSON, NetCDF, or domain specific ones	Yes	JSON, HTML
Does your software allow communications using open communications protocols? e.g. HTTP, FTP, XMPP, SOAP over HTTP, or domain-specific ones	Yes	HTTP
Is your software cross-platform compatible? e.g. does it run under two or more of Windows, Unix/Linux and Mac OS X, or can be used from within two or more of	Yes	Software is platform independent

Internet Explorer, Chrome, Firefox and Safari?		
Does your software adhere to appropriate accessibility conventions or standards?	Yes	
Does your documentation adhere to appropriate accessibility conventions or standards?	Yes	
Is your source code stored in a repository under revision control?	Yes	GitHub
Is each source code release a snapshot of the repository?	No	
Are releases tagged in the repository?	No	
Is there a branch of the repository that is always stable? (i.e. tests always pass, code always builds successfully)	Yes	Main
Do you back-up your repository?	Yes	GitHub, Local
Do you provide publicly-available instructions for building your software from the source code?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Can you build, or package, your software using an automated tool? e.g. Make (https://www.gnu.org/software/make/), ANT (http://ant.apache.org/), Maven (https://maven.apache.org/), CMake (https://cmake.org/),	No	

Python setuptools (https://pypi.python.org/pypi/setuptools), or R package tools (https://cran.r-project.org/doc/manuals/r-devel/R-exts.html)		
Do you provide publicly-available instructions for deploying your software?	No	Project is not deployed as of current phase
Does your documentation list all third-party dependencies?	Yes	Badges in Readme Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Does your documentation list the version number for all third-party dependencies?	No	
Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?	No	
Can you download dependencies using a dependency management tool or package manager? e.g. Ivy (http://ant.apache.org/ivy/), Maven (https://maven.apache.org/), Python pip (https://pypi.python.org/pypi/pip) or setuptools (https://pypi.python.org/pypi/setuptools), PHP Composer (https://getcomposer.org/), Ruby gems (https://rubygems.org), or R	Yes	pip, npm

PackRat (https://rstudio.github.io/packrat/)		
Do you have tests that can be run after your software has been built or deployed to show whether the build or deployment has been successful?	Yes	
Do you have an automated test suite for your software?	Yes	
Do you have a framework to periodically (e.g. nightly) run your tests on the latest version of the source code?	Yes	
Do you use continuous integration, automatically running tests whenever changes are made to your source code?	No	
Are your test results publicly visible?	Yes	
Are all manually-run tests documented?	No	No manual test cases executed
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	Yes	Wiki
Does your website state how many projects and	Yes	Contributors on GitHub

users are associated with your project?		
Do you provide success stories on your website?	No	
Do you list your important partners and collaborators on your website?	No	
Do you list your project's publications on your website or link to a resource where these are available?	No	
Do you list third-party publications that refer to your software on your website or link to a resource where these are available?	No	
Can users subscribe to notifications to changes to your source code repository?	Yes	
If your software is developed as an open source project (and, not just a project developing open source software), do you have a governance model?	Yes	MIT Licensed
Do you accept contributions (e.g. bug fixes, enhancements, documentation updates, tutorials) from people who are not part of your project?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/Contributing.md
Do you have a contributions policy?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/applic

		ation-tracking-system/blob/dev/Contributing.md
Is your contributions' policy publicly available?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/Contributing.md
Do contributors keep the copyright/IP of their contributions?	Yes	MIT Licensed
Does your website and documentation clearly state the copyright owners of your software and documentation?	Yes	MIT Licensed
Does each of your source code files include a copyright statement?	No	However, the whole project is Licensed under MIT
Does your website and documentation clearly state the licence of your software?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/LICENSE
Is your software released under an open source licence?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/LICENSE
Is your software released under an OSI-approved open-source licence?	Yes	Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/blob/dev/LICENSE
Does each of your source code files include a licence header?	No	
Do you have a recommended citation for your software?	No	

Does your website or documentation include a project roadmap (a list of project and development milestones for the next 3, 6 and 12 months)?	Yes	Future Scope in Readme Link: https://github.com/SoftwareEngineering-HomeWork/application-tracking-system/tree/dev#readme
Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?	No	We are at our Bootstrap phase as of now
Do you make timely announcements of the deprecation of components, APIs, etc.?	No	

JTracker - Sustainability Evaluation

This software evaluation report is for your software: JTracker. 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 1.3: Do you publish case studies to show how your software has been used by yourself and others?

A great way of showing off your software is to write case studies about how yourself, and others, have used it. Case studies can help potential users learn about your software. They also act as a great advert for your software. If you can show happy users benefiting from your software, you are likely to gain more users.

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 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 4.8: Do you publish your release history e.g. release data, version numbers, key features of each release etc. on your web site or in your documentation?

A release history allows users to see how your software has evolved. It can provide them with a way to see how active you are in developing and maintaining your software, in terms of new features provided and bugs fixed. Software that is seen to be regularly fixed, updated and extended can be more appealing than software that seems to have stagnated.

Question 10.2: Is each source code release a snapshot of the repository?

Some source code releases take source code from the source code repository and use it to create a release with a directory structure which is markedly different to the structure of the repository. This can lead to a lot of confusion. It is much better to treat your source code release as a snapshot of your repository, making your release structure mirror your repository structure. This makes it far easier to understand the mapping between the releases and the repository: given a file in the release, it is easier to find the same file in the repository, and vice-versa. In addition, the scripts that you use to build your source code in the repository can also be used as-is by users to build packages from source code releases. GitHub provides functionality for automatically Creating Releases
(https://help.github.com/articles/creating-releases/) from code hosted in Git repositories.

Question 10.3: Are releases tagged in the repository?

Every time a change to is committed to source code held under revision control, a revision number or commit identifier is created. Though plain-text, these are not usually human-readable. Many revision control tools allow repositories to be tagged, whereby a useful, memorable name can be given to a specific version. Tagging releases in this way (e.g. release-1.0.1 or conference-09-2015) can make it easier for both you, and users, to get access to the source code that was included in a particular release, or used to create the results you reported in a particular paper, for example.

Question 11.2: Can you build, or package, your software using an automated tool? e.g. Make (https://www.gnu.org/software/make/), ANT (http://ant.apache.org/), Maven (https://maven.apache.org/), CMake (https://cmake.org/), Python setuptools (https://pypi.python.org/pypi/setuptools), or R package tools (https://cran.r-project.org/doc/manuals/r-devel/R-exts.html)?

Typing in lots of instructions is both time-consuming and prone to error. An automated build/packaging tool can make building or packaging your software easier, and less error-prone. Automation is also useful for developers: it makes it easier for them to rebuild or repackage code after implementing extensions, enhancements or bug fixes.

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 11.5: Does your documentation list the version number for all third-party dependencies?

Different versions of languages, libraries, packages, scripts, models or tools can support different features. Code written to use one version of a language, library or package may not be compatible with earlier or later versions. You must provide information on what versions of dependencies users need to use when building, deploying or running your software. A user will be irritated if trying to use your software with Python 3 only to discover it is only compliant with Python 2, something which you, as its developer may have already known. You know what versions you use, so document these to help your users too.

Alternatives to version numbers include, depending upon where the dependency originates: a

source code repository commit identifier or tag, or a download date.
See our guide on How to cite and describe software (http://software.ac.uk/so-exactly-what-software-did-you-use).

Question 11.6: Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?

Users don't want to have to search the web for your third-party dependencies to find the information they need to package or deploy your software. You already know all the information that your users will need about suitable versions, licences and suchlike, so you should make it available to your users. In particular, licence information is very important, because users need to understand the terms and conditions of third-party dependencies so that they can determine whether they are legally permitted to use them, and, so, use your software.

Question 12.3: Do you use continuous integration, automatically running tests whenever changes are made to your 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 whenever changes are made to the code in the source code repository. This means your team (and others if you publish the test results more widely) obtain very rapid feedback on the impact of changes. Continuous integration servers can automatically run jobs to build software and run tests whenever changes are committed to a source code repository. For example, Jenkins (http://jenkins-ci.org) is a continuous integration server that can trigger jobs in response to changes in Git, Mercurial, Subversion and CVS. Travis CI (http://travis-ci.org) is a hosted continuous integration server that can trigger jobs in response to changes in Git repositories hosted on GitHub (https://github.com).
See our guides on How continuous integration can help you regularly test and release your software (http://software.ac.uk/how-continuous-integration-can-help-you-regularly-test-and-release-your-software), Build and test examples (https://github.com/softwaresaved/build_and_test_examples/blob/master/README.md) (which includes walkthroughs on Getting started with Jenkins and Getting started with Travis CI), and Hosted continuous integration (http://www.software.ac.uk/resources/guides/hosted-continuous-integration).
Going further, this can also be done automatically whenever the source code repository changes.
See our guides on Testing your software (http://software.ac.uk/resources/guides/testing-your-software), Adopting automated testing (http://github.com/softwaresaved/automated_testing/blob/master/README.md)

Question 12.5: Are all manually-run tests documented?

It may not be possible, or easy, to automate certain tests e.g. testing a browser-based application after it’s been deployed. In such cases, you should document the list of steps that are to be done to test the software. Documenting the steps means that the tests can be run by anyone, not just the developer who usually does these tests.

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 15.2: Does each of your source code files include a copyright statement?
It's easy to distribute source code files, and this separates the code from any copyright statement that might be on your web site or in your documentation. To cover this eventuality, and remove any ambiguity about ownership, it's good practice to include a copyright statement with each of your source code files, as a comment, or, if the language permits it, as a string constant.

Question 15.6: Does each of your source code files include a licence header?
It's easy to distribute source code files, and this separates the code from any licence statement that might be on your web site or in your documentation. To cover this eventuality, and remove any ambiguity about what a developer can do with the source code, it's good practice to include a licence statement within each of your source code files, as a comment. This can also help to avoid confusion between source files that may have different licences, particularly if there are a number of third-party dependencies used within your software.

Question 15.7: Do you have a recommended citation for your software?

Asking that users cite your software, either directly or via its associated publications, provides you with credit for develop your software. It also provides a means, via harvesting of citations, of gathering evidence of the uptake and exploitation of your software. See, for example, Citing R (https://cran.r-project.org/doc/FAQ/R-FAQ.html#Citing-R) and Citing Taverna (http://www.taverna.org.uk/cite/).

See our guide on How to cite and describe software (http://software.ac.uk/so-exactly-what-software-did-you-use) and examples of the citations recommended by various software packages (http://www.software.ac.uk/blog/2014-07-30-oh-research-software-how-shalt-i-cite-thee).

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).