vogueX---Fashion-Recommender

Group 64: Jayneel Shah, Smiti Kothari, Vedant Patel

Animated Video | Github Repo | Live Demo

Introduction

VogueX redefines how you approach fashion, offering personalized outfit recommendations that adapt to your unique style, occasion, and even the weather. Whether it is a sunny day or unexpected rain, VogueX ensures you are always dressed appropriately with weather-based suggestions. You can filter recommendations by gender, culture, age, and occasion, making it easy to find looks that resonate with your personal style. Explore comprehensive product searches that highlight the best deals from top e-commerce sites, so you can shop smart and stay fabulous. Plus, with our favorites feature, you can save your top looks for quick access, ensuring you are always ready for any event, from casual outings to formal gatherings!

Workflow

The user interacts with VogueX through a streamlined process:

Login/Signup: The user first interacts with the login/signup portal. A new user is redirected to a signup portal, where the user can create an account by adding the correct credentials. The email is verified using OTP Verification method. Upon successful verification the user is directed to the homescreen.

Home Screen: The home screen outlines the overview of the web portal and different features that the web portal provides. The home screen also has a navbar through which the users can navigate to different pages of the web portal.

Virtual Try on Module: This module is the highlight of our project as it provides us with 6 try on options where the user can upload their image and try on different outfits and products and also customize their outfits according to their preferences.

Personalized Fashion Recommendation: Users provide key information such as gender, age, culture, occasion to modify their recommendations. Furthermore, the app takes information from the current weather and recommends the outfits that are deemed suitable for the current weather, like overcoats for winters. Users can also get recommendations based on the festivals.

Shopping Websites: Using the input and contextual data, the system generates a tailored list of outfit suggestions based on gender, culture, age, and occasion. The app also checks the availability of the suitable outfits and redirects the user to the shopping websites from where they can buy their tried on outfits.

Favorites Management: The app also provides the users with a feature to favourite their tried on outfits, so they can revisit those looks again and shop the items accordingly. The favorite outfits would also act as a reference for the recommendations module in the future.

Logout- Users can choose to log out from any stage of the application, returning them to the start screen.

After logging out, the application session ends

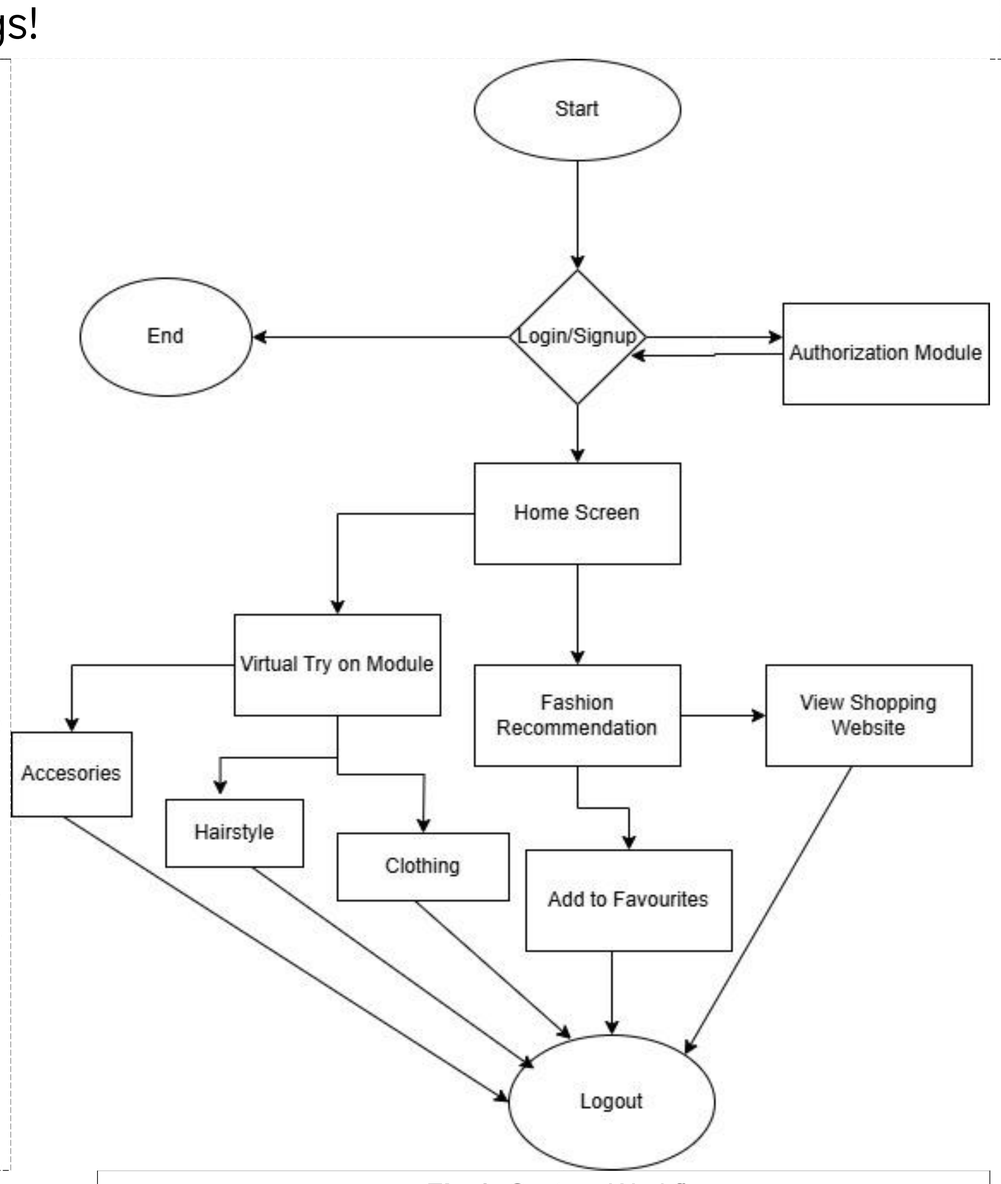


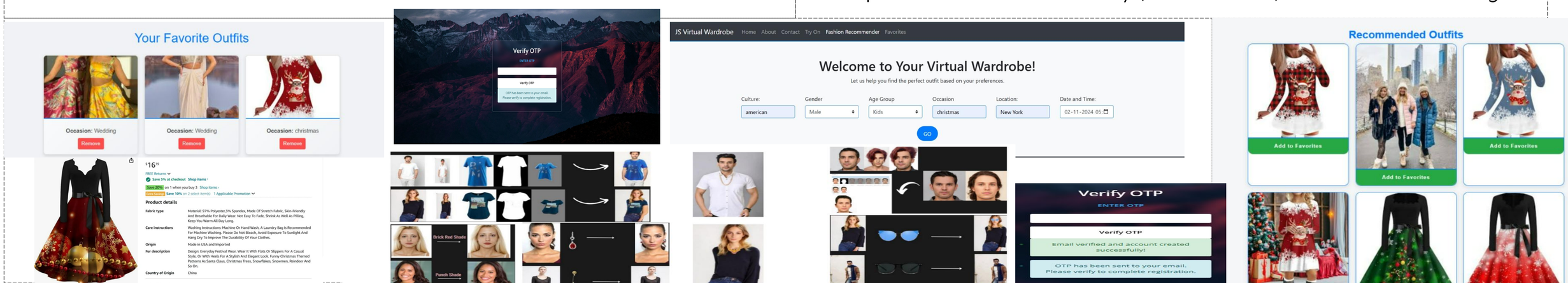
Fig 1: System Workflow

Enhancements

- **Authentication and Sign Up:** Improved the login functionality by adding login/signup options with fields like username, email, password, confirm password and several validators, which would ensure that legitimate data is entered by the user. Otp verification using email is incorporated in this version which adds an added layer of security to protect the user account and data.
- **Weather API Integration:** Major upgradation from the previous version is integration of the weather api to the project. This feature incorporates the real time weather data, so the users will be able to see the recommendations of outfits based on the current weather.
- **Calendar Integration:** Another major upgrade is the integration of the calendar api, which will recommend outfits to the users based on the dates they input. For eg: if the user inputs 31st October as the input date, then it will recommend various halloween costumes.
- **AI Virtual Wardrobe System:** We are providing six distinct try-on features: clothing, necklace, earring, lipstick, glasses and hairstyle modules. Users simply upload their own image, and our advanced deep learning models allow them to see how different items look on them in real-time.
- **Share Outfits:** Users can share their custom outfits made using the try-on features with their friends as well for instant feedback.

Future Scope.

- **Loyalty and Referral Program:** Developing loyalty programs and reward points for the users that engage with the platform consistently. Furthermore, the users will receive certain reward points on recommending the platform to their friends, which guarantees early access to the newly added features.
- **Pro Version of the Platform:** Developing a pro version that incorporates even more try-on options with a better recommendation model and AR model will attract more users to the platform and will also lead to revenue generation.
- **Social Login Options:** Allowing the users to login to the website using their social media accounts would improve the engagement more, plus it would also allow them to post their try-ons on social media platforms.
- **Improving UI/UX:** UI/UX can be improved to ensure an engaging, user-friendly and a visually appealing experience.
- **Better Recommender System:**Integrate these try on module with the fashion recommender such that the clothes that are recommended should be tried on to the user.
- **Calendar App Improvements:** We also want to take our calendar app integration to the next level, allowing it to reflect not just public holidays but also personal events like birthdays, anniversaries, and even casual outings.



Group 64: Project 2
GitHub Repo- https://github.com/systems-org/vogueX---Fashion-Recommender

Jayneel Shah (jshah26)
Smiti Kothari (skothar3)
Vedant Patel (vpatel32)

Notes	Evidence
	9/105
Workload is spread over the whole team (one team member is often Xtimes more productive than the others…	3
but nevertheless, here is a track record that everyone is contributing a lot)	3
Number of commits	3
Number of commits: by different people	3
Issues reports: there are **many**	1
Issues are being closed	1
Docs: doco generated, format not ugly	3
Docs: what: point descriptions of each class/function (in isolation)	3
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	3
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
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	3
Use of version control tools	2
Test cases exist	3
Test cases are routinely executed	2
Issues are discussed before they are closed	2
Chat channel: exists	3
Test cases: a large proportion of the issues related to handling failing cases.	3
Evidence that the whole team is using the same tools: everyone can get to all tools and files	3
Evidence that the whole team is using the same tools (e.g. config files in the repo, updated by lots of different people)	3
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
Evidence that the members of the team are working across multiple places in the code base	3
Short release cycles	2

Group 64: Project 2
GitHub Repo- https://github.com/systems-org/vogueX---Fashion-Recommender

Jayneel Shah (jshah26)
Smiti Kothari (skothar3)
Vedant Patel (vpatel32)

The file .gitignore lists what files should not be saved to the repo. See [examples]i(https://github.com/github/gitignore)	3
The file INSTALL.md lists how to install the code	3
The file LICENSE.md lists rules of usage for this repo	3
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
The file README.md contains all the following	3
Video	3
DOI badge: exists. To get a Digitial Object Indentifier, regiser the project at Zenodo. DOI badges look like this:	3
Badges showing your style checkers	3
Badges showing your code formatters	3
Badges showing your syntax checkers.	3
Badges showing your code coverage tools	3
Badges showing any other Other automated analysis tools	3

VogueX Fashion Recommender - Sustainability Evaluation

This software evaluation report is for your software: VogueX Fashion Recommender. 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 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 4.6: If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?

If your software includes support for Application Programming Interfaces (API) (https://en.wikipedia.org/wiki/Application_programming_interface), whether these be functions, data types, or classes offered by a library or a collection of REST (https://en.wikipedia.org/wiki/Representational_state_transfer) endpoints or web services, then these need to be documented if you want them to be used. Code examples alone may not provide enough information on how someone can use your API in their own code.

From structured comments in the code, generating complete, structured API documentation can be done automatically with, for example, JavaDoc (http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html) (for Java), Doxygen (http://www.stack.nl/~dimitri/doxygen/) (for C, C++ , Fortran or Python), Sphinx (http://sphinx-doc.org/) (for Python). Certain REST frameworks, such as Django (http://www.django-rest-framework.org/), also support auto-generation of API documentation.

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 5.4: Are e-mails to your support e-mail address received by more than one person?
It's easy to forget about an e-mail, especially one that's asking difficult questions, so your e-mails to your support e-mails address should always be received by more than one person. One person should still have the primary responsibility of handling users' e-mails, but others can step up to handle e-mails if necessary, so that a user's query will be acknowledged even if one of you is on holiday, ill or otherwise indisposed.
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 7.1: 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)
Supporting open data formats, data formats which are publicly available, as a complement to any proprietary ones you need to support, has many benefits. If data can be imported and exported in an open format, then it can be used with any software that uses this format. Software that supports open formats do not lock users into that software, because they will be able to use other software to access their data, if necessary. This can make it easier for users to swap between software that uses open formats. This means that users of other software may switch to your software, if your software is more efficient, robust, scalable or functional than that of your competitors'.
Try and adopt open formats that are mature, ratified standards, if possible. Standards can go through many iterations, because they evolve as ideas are proposed and debated and the scope, remit and intent of the standards are agreed. If a standard changes, then any software that uses the standard needs to be changed to keep up to date. Most of the big changes occur early in the lifetime of a standard. Mature and ratified standards are less likely to change significantly or frequently, which reduces the risk of you having to modify your software in response.

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 10.5: Do you back-up your repository?
While third-party repositories are very useful, it can be reassuring to know that you have a local back-up of your repository in case the third-party repository goes down for any length of time, or the host ceases to provide the repository hosting service. Distributed revision control repositories such as Git and Mercurial can be easy to back-up locally, just by cloning the repository. Centralised repositories such as Subversion or CVS rely upon the repository hosting service to provide you with the functionality to back-up your repository (see, for example SourceForge Subversion back-ups (http://sourceforge.net/p/forge/documentation/svn/#backups).

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 11.7: 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 PackRat (https://rstudio.github.io/packrat/)

Bundling all third-party dependencies with your software means that your users don't need to download the dependencies. However, it can lead to a very big release packages and, in some cases, you will not be able to bundle a dependency, because its licence prevents it. Dependency management tools provide automated frameworks to download and install third-party dependencies at build or deployment time. This helps to reduce the size of release packages, avoid licensing issues and save users from having to download dependencies themselves.

Question 11.8: 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?

Tests give a user confidence that your software has built and installed correctly on their platform. If a test fails, the nature of the failure can help the identify why e.g. maybe the user forgot a configuration step, or provided an incorrect configuration option.

For developers, tests contribute to a fail-fast environment, which allows the rapid identification of failures introduced by changes to the code such as optimisations or bug fixes.

Tests are an important aspect of maintainable software. There are many frameworks available for writing tests in a range of languages, including JUnit (http://junit.org/) for Java, CUnit (http://cunit.sourceforge.net/) for C, CPPUnit (http://www.freedesktop.org/wiki/Software/cppunit/) and googletest (https://code.google.com/p/googletest/) for C++, FRUIT (http://sourceforge.net/projects/fortranxunit/) for Fortran, py.test (http://pytest.org/) and nosetests (http://nose.readthedocs.org/) for Python, testthat (https://cran.r-project.org/web/packages/testthat/index.html) for R and PHPUnit (https://phpunit.de) for PHP.

Alternatively, you can provide a list of steps that a user can take to check the deployment of the software e.g. for a web-based application, this might just be checking that it the application is accessible via a browser.

Question 12.1: Do you have an automated test suite for your software?

After changing your code and rebuilding it, a developer will want to check that their changes or fixes have not broken anything. Tests contribute to a fail-fast environment, which allows the rapid identification of failures introduced by changes to the code such as optimisations or bug fixes. The lack of tests can dissuade developers from fixing, extending or improving your software, as developers will be less sure of whether they are inadvertently introducing bugs as they do so.

Each test might verify an individual function or method, a class or module, related modules or components or the software as a whole. Tests can ensure that the correct results are returned from

a function, that an operation changes the state of a system as expected, or that the code behaves as expected when things go wrong.
There are many frameworks available for writing tests in a range of languages, including JUnit (http://junit.org/) for Java, CUnit (http://cunit.sourceforge.net/) for C, CPPUnit (http://www.freedesktop.org/wiki/Software/cppunit/) and googletest (https://code.google.com/p/googletest/) for C++, FRUIT (http://sourceforge.net/projects/fortranxunit/) for Fortran, py.test (http://pytest.org/) and nosetests (http://nose.readthedocs.org/) for Python, testthat (https://cran.r-project.org/web/packages/testthat/index.html) for R and PHPUnit (https://phpunit.de) for PHP.
Automating the run of your test suite means the entire set of tests can be run in one go, making life easier for your developers. Having an automated build system is a very valuable precursor to providing a test suite, and having an automated build and test system is a valuable resource in any software project.
See our guides on Testing your software (http://software.ac.uk/resources/guides/testing-your-software) and Adopting automated testing (http://github.com/softwaresaved/automated_testing/blob/master/README.md).

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 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 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 13.7: 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 13.8: 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?

A governance model sets out how a, open source project is run. It describes the roles within the project and its community and the responsibilities associated with each role; how the project supports its community; what contributions can be made to the project, how they are made, any conditions the contributions must conform to, who retains copyright of the contributions and the process followed by the project in accepting the contribution; and, the decision-making process in within the project.

Though they are designed for open source projects, many of their concerns are relevant to any software project.

OSS Watch (http://oss-watch.ac.uk) provide an introduction to governance models (http://oss-watch.ac.uk/resources/governancemodels).

Question 15.5: Is your software released under an OSI-approved open-source licence?

The Open Source Initiative (http://opensource.org/) (OSI) have produced an Open Source Definition (http://opensource.org/osd) which provides for a shared understanding of the term 'open source' and allows for OSI to accredit licences that meet this definition as 'OSI approved'. OSS Watch (http://oss-watch.ac.uk) OSS Watch comment that this provides 'a means of avoiding debates over interpretation of the open source definition and which licences do or do not conform to it. By recognising the OSI as the appropriate final authority in this issue, much confusion is

avoided.'
Some open source project hosting services will only host code licenced under an OSI-approved licence e.g. see SourceForge (http://sourceforge.net) terms of use (http://slashdotmedia.com/terms-of-use/)
See the list of OSI-approved open source licences (http://opensource.org/licenses).
See our guide on Choosing an open-source licence (http://www.software.ac.uk/resources/guides/adopting-open-source-licence).

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 16.1: Does your website or documentation include a project roadmap (a list of project and development milestones for the next 3, 6 and 12 months)?
A roadmap allows users to see when new features will be added and plan their project accordingly. It also has an important secondary benefit: one of the most important factors that will influence a user's choice of software is the likelihood of that software still being around – and supported – in the future. There are many ways in which a project can persuade a user of its longevity: regular announcements, regular releases, prompt replies to queries, information about funding and its plans for the future – a roadmap.

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).