Software Engineering Project 2

Group# 34

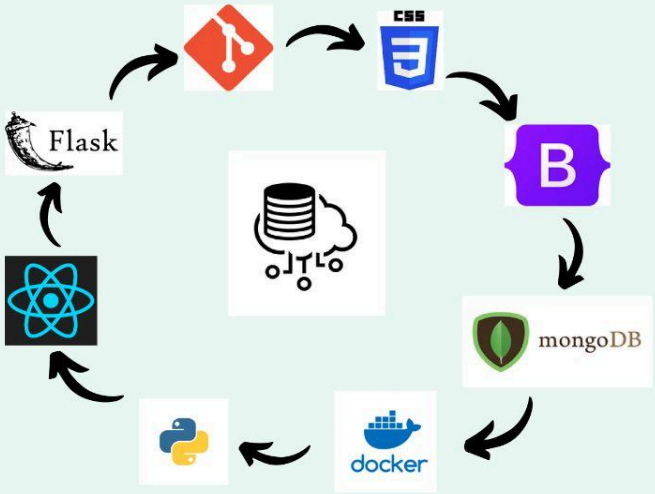
Team Members:

1. Srimadh V Rao, svrao3
2. Manav Divyesh Shah, mdshah5
3. Akul Gopal Devali, adevali

Poster

🚀 PopcornPicks 2.0 : The Next Generation of Movie Discovery!

Popcorn Picks is your ultimate movie buddy, delivering handpicked film recommendations tailored to your taste! Powered by a lightning-fast algorithm, it makes discovering new favorites and hidden gems a breeze. Dive into a world of movie magic, where you can explore personalized insights, keep track of what you love, and even share recommendations with friends via email. Whether you're a casual viewer or a true cinephile, Popcorn Picks takes your movie-watching experience to the next level—effortlessly and enjoyably!



What's New?

- **Seamless MongoDB Atlas Integration!** No more database hassle! Scale up effortlessly, enjoy built-in redundancy, and feel secure with enterprise-level encryption!
- **Docker Power Unleashed!** Deploy anywhere with ease! No more "it works on my machine" issues. Future-ready for microservices, easy updates, and smoother rollbacks!
- **Smarter Recommendations with Collaborative Filtering!** Enjoy more accurate movie picks! Our hybrid system updates in real-time and caters to your unique tastes—even for newbies!
- **Sleek React Frontend Redesign!** Modern, responsive, and lightning-fast! Mobile-first and accessibility-friendly for a smoother, better-looking user experience!
- **Insightful User Dashboard & Analytics!** Discover your movie-watching habits, genre preferences, and track progress—all in one place!
- **Enhanced Testing Framework!** Your app is now rock-solid with extensive tests! Seamless functionality across platforms and scenarios, even under heavy loads!

Future Scope

- **Friends Recommend Movies:** Let users get personalized movie recommendations from their friends for a more social experience!
- **Parental Controls & Kids Mode:** Family-friendly viewing! Add parental controls and a curated kids' section to make Popcorn Picks safe and fun for all ages!
- **Dynamic Movie Collections:** Always stay relevant! Offer seasonal and trending collections like "Halloween Specials" and "Oscar Winners" based on user interests.
- **Fun Trivia & Movie Quizzes:** Keep users engaged with interactive trivia and quizzes about movies they've watched!
- **Tech Improvement:** Supercharge the backend with Express and Node.js for smoother, seamless integration with the React frontend! Upgrade the UI from bootstrap to tailwind to material UI.

SCAN FOR DEMO

SCAN FOR PROJECT

Manav Divyesh Shah
Srimadh V Rao
Akul Devali

Link to Repository and Video

Repository: https://github.com/se24ncsu/PopcornPicks
Video: https://go.abhirao.in/popcornPicks/video
Live Demo of WebApp: https://go.abhirao.in/popcornPicks/demo

Rubric Table

Column 1	Column 2 Sum = 264	Column 3
Workload is spread over the whole team	3	
Number of commits	3	
Number of commits: by different people	3	
Issues reports: there are many	3	https://github.com/se24ncsu/PopcornPicks/issues
Issues are being closed	3	https://github.com/se24ncsu/PopcornPicks/issues
Docs: doco generated, format not ugly	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Docs: what: point descriptions of each class/function (in isolation)	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
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	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs

Use of version control tools	3	https://github.com/se24ncsu/PopcornPicks/commits/main/
Test cases exist	3	https://github.com/se24ncsu/PopcornPicks/tree/main/test
Test cases are routinely executed	3	https://github.com/se24ncsu/PopcornPicks/blob/main/.github/workflows/unittest.yml
Issues are discussed before they are closed	3	https://github.com/se24ncsu/PopcornPicks/issues/31
Chat channel: exists	3	https://github.com/se24ncsu/PopcornPicks/blob/main/asset/chat.png
Test cases: a large proportion of the issues related to handling failing cases.	2	https://github.com/se24ncsu/PopcornPicks/issues
Evidence that the whole team is using the same tools: everyone can get to all tools and files	3	
Evidence that the members of the team are working across multiple places in the code base	3	
Short release cycles	2	
The file .gitignore lists what files should not be saved to the repo. See [examples]i(https://github.com/github/gitignore)	3	https://github.com/se24ncsu/PopcornPicks/blob/main/.gitignore
The file INSTALL.md lists how to install the code	3	https://github.com/se24ncsu/PopcornPicks/blob/main/docs/install.md
The file LICENSE.md lists rules of usage for this repo	3	https://github.com/se24ncsu/PopcornPicks/blob/main/LICENSE

The file CODE-OF-CONDUCT.md lists rules of behavior for this repo; e.g. see example	3	https://github.com/se24ncsu/PopcornPicks/blob/main/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	https://github.com/se24ncsu/PopcornPicks/blob/main/CONTRIBUTING.md
The file README.md contains all the following	3	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
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	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
Badges showing your code formatters.	3	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
Badges showing your syntax checkers.	3	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
Badges showing your code coverage tools	3	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
Badges showing any other Other automated analysis tools	3	https://github.com/se24ncsu/PopcornPicks/blob/main/README.md
Does your website and documentation provide a clear, high-level overview of your software?	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Does your website and documentation clearly describe the type of user	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs

who should use your software?		
Do you publish case studies to show how your software has been used by yourself and others?	2	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Is the name of your project/software unique?	3	https://tmsearch.uspto.gov/search/search-results
Is your project/software name free from trademark violations?	3	https://tmsearch.uspto.gov/search/search-results
Is your software available as a package that can be deployed without building it?	3	
Is your software available for free?	3	https://github.com/se24ncsu/PopcornPicks
Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository?	3	https://github.com/se24ncsu/PopcornPicks
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)?	3	https://github.com/se24ncsu/PopcornPicks
Is your documentation clearly available on your website or within your software?	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs

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?	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
If you provide more extensive documentation, does this provide clear, step-by-step instructions on how to deploy and use your software?	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs
Do you provide a comprehensive guide to all your software's commands, functions and options?	3	https://github.com/se24ncsu/PopcornPicks/tree/main/docs/generated_docs
Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?	3	https://github.com/se24ncsu/PopcornPicks/blob/main/docs/Troubleshooting.md
If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?	3 , N/A	
Do you store your documentation under revision control with your source code?	1	
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?	1	

Does your software describe how a user can get help with using your software?	3	https://github.com/se24ncsu/PopcornPicks/blob/main/CODE_OF_CONDUCT.md
Does your website and documentation describe what support, if any, you provide to users and developers?	3	https://github.com/se24ncsu/PopcornPicks/blob/main/CONTRIBUTING.md
Does your project have an e-mail address or forum that is solely for supporting users?	3	Popcornpicksse24@gmail.com
Are e-mails to your support e-mail address received by more than one person?	3	
Does your project have a ticketing system to manage bug reports and feature requests?	3	https://github.com/se24ncsu/PopcornPicks/issues
Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?	3	
Is your software's architecture and design modular?	3	https://github.com/se24ncsu/PopcornPicks/blob/main/asset/system_architecture.svg
Does your software use an accepted coding standard or convention?	3	Pylint is used to follow convention
Does your software allow data to be imported and exported using open data formats? e.g. GIF, SVG, HTML, XML, tar, zip, CSV, JSON,	3 (CSV,JSON, HTML, SVF, GIF)	

NetCDF, or domain specific ones		
Does your software allow communications using open communications protocols? e.g. HTTP, FTP, XMPP, SOAP over HTTP, or domain-specific ones	3 (HTTP)	
Is your software cross-platform compatible?	3	Run using Docker
Does your software adhere to appropriate accessibility conventions or standards?	3	
Does your documentation adhere to appropriate accessibility conventions or standards?	3	
Is your source code stored in a repository under revision control?	1	
Is each source code release a snapshot of the repository?	1	
Are releases tagged in the repository?	1	
Is there a branch of the repository that is always stable? (i.e. tests always pass, code always builds successfully)	3	
Do you back-up your repository?	3	
Do you provide publicly-available instructions for building your software from the source code?	3	

Can you build, or package, your software using an automated tool?	3	
Do you provide publicly-available instructions for deploying your software?	3	
Does your documentation list all third-party dependencies?	3	
Does your documentation list the version number for all third-party dependencies?	3	
Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?	3	
Can you download dependencies using a dependency management tool or package manager?	3	
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?	0	
Do you have an automated test suite for your software?	3	
Do you have a framework to periodically (e.g. nightly) run your tests on the latest version of the source code?	3	

Do you use continuous integration, automatically running tests whenever changes are made to your source code?	3	
Are your test results publicly visible?	3	
Are all manually-run tests documented?	3	
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?	0	
Does your website state how many projects and users are associated with your project?	3	
Do you provide success stories on your website?	0	
Do you list your important partners and collaborators on your website?	0	
Do you list your project's publications on your website or link to a resource where these are available?	0	
Do you list third-party publications that refer to your software on your website or link to a resource where these are available?	0	
Can users subscribe to notifications to changes to your source code repository?	3	

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?	0	
Do you accept contributions (e.g. bug fixes, enhancements, documentation updates, tutorials) from people who are not part of your project?	3	
Do you have a contributions policy?	3	
Is your contributions' policy publicly available?	3	
Do contributors keep the copyright/IP of their contributions?	3	
Does your website and documentation clearly state the copyright owners of your software and documentation?	3	
Does each of your source code files include a copyright statement?	3	
Does your website and documentation clearly state the licence of your software?	3	
Is your software released under an open source licence?	3	
Is your software released under an OSI-approved open-source licence?	3	

Does each of your source code files include a licence header?	3	
Do you have a recommended citation for your software?	3	
Does your website or documentation include a project roadmap (a list of project and development milestones for the next 3, 6 and 12 months)?	N/A 0	
Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?	N/A 0	
Do you make timely announcements of the deprecation of components, APIs, etc.?	N/A 0	

Popcorn Picks - Sustainability Evaluation

This software evaluation report is for your software: Popcorn Picks. 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 4.8: 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 10.1: Is your source code stored in a repository under revision control?

We all use some form of revision control. Even just keeping a back-up of some code – while doing bug fixes for example – in a directory called my-code-16012012-works is a simple form of revision control. A repository with revision control provides a number of benefits over these simple approaches.

- You can retrieve any version of your software, or any file within those versions, from any point in time.
- Multiple developers can work on the same software, at the same time, sharing their changes without any risk of a file being overwritten and its previous contents being lost forever.
- It automatically records who changed what, and when, and allows you to record the "why". This provides a complete audit trail of the evolution of your software for you and, in an open source world, for your users.

Revision control is an important aspect of developing maintainable software. If you are developing open source software then revision control is essential.

See What is version control? (http://oss-watch.ac.uk/resources/versioncontrol) from OSS Watch (http://oss-watch.ac.uk), Revision control (https://en.wikipedia.org/wiki/Version_control) on Wikipedia, and Software Carpentry's Version Control with Git (http://swcarpentry.github.io/git-novice/).

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 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 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).