Software Engineering Project - CSC 510-001 (Fall 2024)

Project 2

Students:Aditya Singh, Bahare Riahi, and Monish Erode Sridhar (Group 85)
Software Engineering course CSC 510-001 North Carolina State University.

Repository:
https://github.com/BetaPack/MovieRecommender.git

NC STATE UNIVERSITY

MOVIE RECOMMENDER

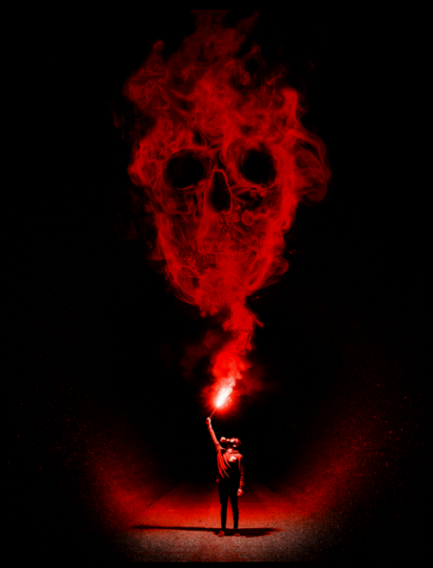
GROUP 85: ADITYA SINGH | BAHARE RIAHI | MONISH ERODE SRIDHAR

About!

Tired of scrolling through movies, can't seem to find the right movie for your night? Don't worry, we present to you the MovieRecommender — a platform designed to provide personalized movie recommendations tailored to your unique preferences. Say goodbye to decision fatigue and endless searching. Simply add your favorite movies, and our algorithm will suggest films that closely align with your tastes. Whether it' s ratings, genres, or feedback, MovieRecommender ensures you get the best recommendations to suit your mood. Let us simplify your movie selection process and make every movie night a hit!

What's New

- Integration of Youtube API to provide trailers for the recommended movies.
- Adding login and sign up page
- Making a profile user account.
- Adding email verification after creating an account.
- Save history of users previously recommended and liked movies.
- Extended Section to show movie information
- Implemented the floating window for the extra movie information for both initial suggested movies and user-response-based suggested movies
- Scraped and Developed an enriched dataset for Movie plot, Movie cast and movie budget information
- Improve the UI of the movie informations and make the Js code cleaner and modular
- Implemented the Algorithm to re-suggest movies based on user feedback



A FILM FOR TONIGHT

Work Flow

- MovieRecommender is a straightforward app that takes movies in which users are interested in, and recommends a list of movies that are similar to the provided movie list.
- The search function is used to allow users to search for specific movies from the dataset based on partial input.
- The user interacts with the UI (HTML, CSS, Flask) and uses the search function to provide a list of movies they're interested in.
- The movies list is then fed into the recommender, generating a list of recommended movies based on a weighted score generated by the recommender
- The script also fetches additional information (like IMDB ratings, genre, and posters) for each recommended movie using the OMDB API.
- MovieRecommender also allows users to provide feedback on the recommended movies, which will then be saved.

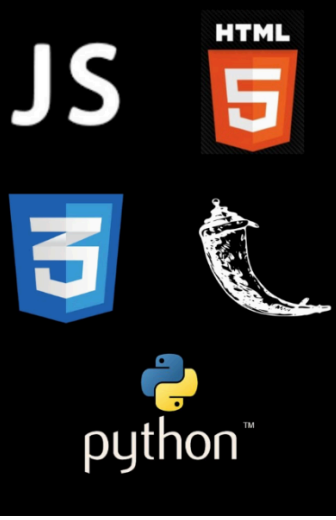
Future Scope

- Add different categories for the films based genres, casting choices and production styles.
- Add social media or login with social media
- Download/Email suggested movies
- Add movie reviews to each movie
- Improve the Recommender algorithm to recommend based on movie plot and not based on genres.

Demo QR Code

Video QR Code

Repo QR code



Total Score = 249

Notes	Assessment	evidence
Workload is spread over the whole team (one team member is often Xtimes more productive than the others…	3	https://github.com/BetaPack/MovieRecommender/branches
but nevertheless, here is a track record that everyone is contributing a lot)	3	https://github.com/BetaPack/MovieRecommender/branches
Number of commits = 21	3	https://github.com/BetaPack/MovieRecommender/branches
Number of commits: by different people	2	Monish Erode Sridhar = 6 Aditya Singh= 11 Bahare Riahi = 4
Issues reports: there are many	2	https://github.com/BetaPack/MovieRecommender/issues?q=is%3Aissue+is%3Aclosed
Issues are being closed	3	https://github.com/BetaPack/MovieRecommender/issues?q=is%3Aissue+is%3Aclosed
Docs: doco generated, format not ugly	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md

Docs: what: point descriptions of each class/function (in isolation)	0	
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	2	https://drive.google.com/file/d/1zbt6i2NxqTW6gxD-30KBEZOrFPS1fA8a/view?usp=drive_link
Docs: why: docs tell a story, motivate the whole thing, deliver a punchline that makes you want to rush out and use the thing	2	
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	3	
Test cases exist	2	
Test cases are routinely executed	3	https://github.com/BetaPack/MovieRecommender/actions
Issues are discussed before they are closed	2	https://drive.google.com/drive/folders/1RrFv1ImeBW3Ps13vwfis2xUeCzM-PyrZ?usp=sharing
Chat channel: exists	3	https://drive.google.com/drive/folders/1cPTlVgbfaSHBf7vGdVxPj8Cg5r_pNCBO?usp=sharing

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	2	Look at commit history
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	2	Look at commit history
Short release cycles	3	
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/BetaPack/MovieRecommender/blob/master/.gitignore
The file INSTALL.md lists how to install the code	3	https://raw.githubusercontent.com/MadhurDixit13/MovieRecommender/master/asset/execution.gif

The file LICENSE.md lists rules of usage for this repo	3	https://github.com/BetaPack/MovieRecommender/blob/master/LICENSE.md
The file CODE-OF-CONDUCT.md lists rules of behavior for this repo; e.g. see example	3	https://github.com/BetaPack/MovieRecommender/blob/master/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/BetaPack/MovieRecommender/blob/master/CONTRIBUTING.md
The file README.md contains all the following	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md
DOI badge: exists. To get a Digitial Object Indentifier, regiser the project at Zenodo. DOI badges look like this:	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1
Badges showing your style checkers	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1
Badges showing your code formatters.	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1
Badges showing your syntax checkers.	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1
Badges showing your code coverage tools	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1

Badges showing any other Other automated analysis tools	3	https://github.com/BetaPack/MovieRecommender/blob/master/README.md?plain=1
Does your website and documentation provide a clear, high-level overview of your software?	2	
Does your website and documentation clearly describe the type of user who should use your software?	3	
Do you publish case studies to show how your software has been used by yourself and others?	0	
Is the name of your project/software unique?	3	
Is your project/software name free from trademark violations?	3	
Is your software available as a package that can be deployed without building it?	3	
Is your software available for free?	3	
Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository?	3	

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	
Is your documentation clearly available on your website or within your software?	3	
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	
If you provide more extensive documentation, does this provide clear, step-by-step instructions on how to deploy and use your software?	2	
Do you provide a comprehensive guide to all your software's commands, functions and options?	2	
Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?	2	
If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?	0	

Do you store your documentation under revision control with your source code?	3	
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	all details in readme.md
Does your website and documentation describe what support, if any, you provide to users and developers?	2	all details in readme.md
Does your project have an e-mail address or forum that is solely for supporting users?	2	all details in readme.md
Are e-mails to your support e-mail address received by more than one person?	2	
Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?	3	https://github.com/BetaPack/MovieRecommender/issues
Is your software's architecture and design modular?	3	

Does your software use an accepted coding standard or convention?	3	
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	2	
Does your software allow communications using open communications protocols?* e.g. HTTP, FTP, XMPP, SOAP over HTTP, or domain-specific ones	3	
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 Internet Explorer, Chrome, Firefox and Safari?	3	
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?	3	
Is each source code release a snapshot of the repository?	2	
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?* 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)	3	https://github.com/BetaPack/MovieRecommender/blob/master/setup.py

Do you provide publicly-available instructions for deploying your software?	3	
Does your documentation list all third-party dependencies?	3	https://github.com/BetaPack/MovieRecommender/blob/master/requirements.txt
Does your documentation list the version number for all third-party dependencies?	3	https://github.com/BetaPack/MovieRecommender/blob/master/requirements.txt
Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?	2	
Can you download dependencies using a dependency management tool or package manager?	1	
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?	3	
Do you have an automated test suite for your software?*	3	https://github.com/BetaPack/MovieRecommender/actions
Do you have a framework to periodically (e.g. nightly) run your tests on the latest version of the source code?	2	

Do you use continuous integration, automatically running tests whenever changes are made to your source code?	2	
Are your test results publicly visible?	3	https://github.com/BetaPack/MovieRecommender/tree/master/test
Are all manually-run tests documented?*	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?	1	
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?	2	
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	1	

or link to a resource where these are available?		
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?	2	
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?	2	

Does each of your source code files include a copyright statement?	1	
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?	1	
Do you have a recommended citation for your software?	0	
Does your website or documentation include a project roadmap (a list of project and development milestones for the next 3, 6 and 12 months)?	2	
Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?	1	
Do you make timely announcements of the deprecation of components, APIs, etc.?	3	