Movie Recommender

Introduction

The Movie Recommender system is a collaborative filtering-based engine designed to help users find personalized movie suggestions. By analyzing user preferences like genre and reviews, it offers tailored recommendations, simplifying the decision-making process for movie lovers

Tools Used

Current Features (Version i)

- Personalized Recommendations: Custom movie suggestions based on user interests.
- User Accounts: User-specific tracking for recommendations, ratings, and history.
- Feedback System: Allows user feedback to improve future recommendations.
- Mobile-Friendly: Responsive design for desktop and mobile.

Proposed Extensions (Version i+1)

- UI Enhancements: Introducing a cleaner, more intuitive navigation with genre and actor filters for ease of use.
- Rating System: Adding a thumbs-up/thumbs-down rating option to allow users to quickly rate recommended movies.
- Monthly Top Movies Feature: Displays trending movies or recent releases for a curated view of current popular titles.
- Personalized Email Notifications: Sends users weekly movie recommendations via email, enhancing engagement with automated updates.

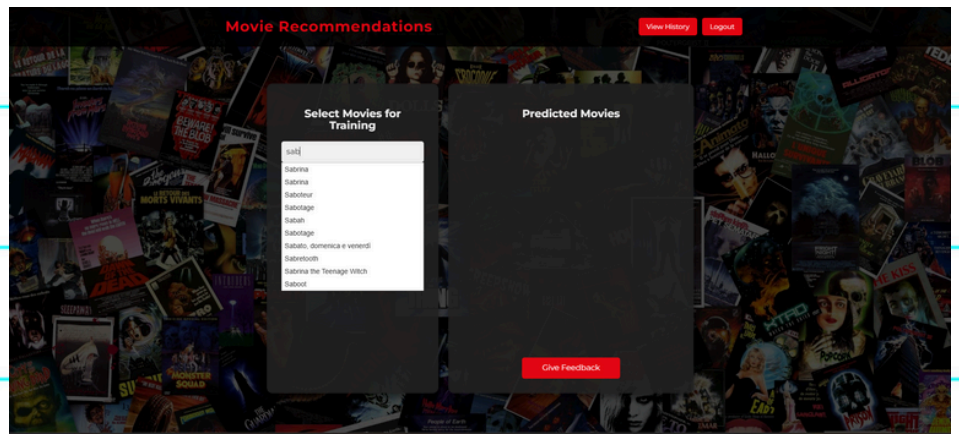
Testing Strengths

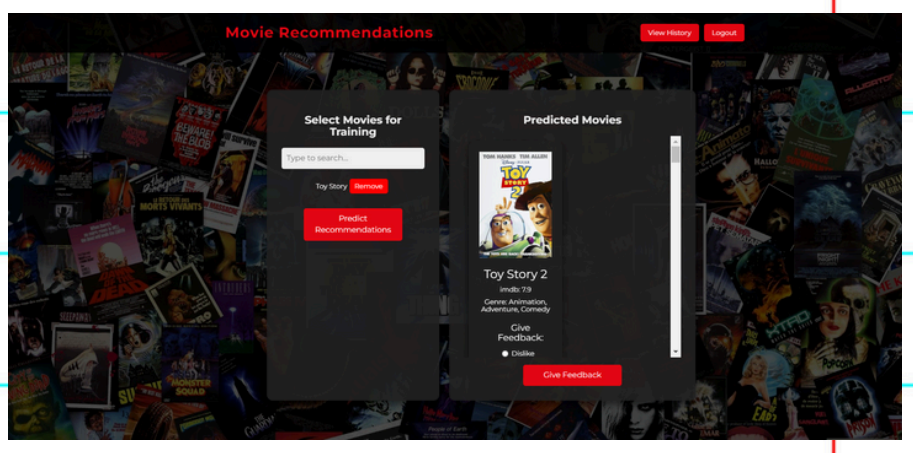
- Comprehensive Recommendation Testing: Covers item-based and plot-based recommendations, ensuring accurate, duplicate-free, and user-tailored suggestions.
- User Interaction Validation: Tests for robust search functionality and reliable feedback collection (like/dislike), enhancing personalization.
- Automated Weekly Emails: Validates timely, relevant recommendations sent to users via email.
- Continuous Integration: GitHub Actions automate test runs on every commit, ensuring system stability and reliability.

Implementation Milestones

- Milestone 1 – UI/UX Improvements: Simplify navigation and improve accessibility with an updated interface and genre/actor filters.
- Milestone 2 – Add Rating System: Introduce a quick thumbs-up/thumbs-down rating feature to enhance feedback for better recommendations.
- Milestone 3 – Weekly Movie Suggestions: Automate personalized email recommendations sent to users each week.
- Milestone 4 – Monthly Top Movies Feature: Showcase trending or newly released movies in a dedicated section for easy discovery.

Screen Snaps





<-- Repo

Team 62
Shanmukha Deety | Nikhilesh Cherukuri | Mohit Hosakatte

Demo ->

Team 62 - Project 2

Teammates : Shanmukha Sreenivas Deety - sdeety
Nikhilesh Cherukuri - ncheruk2
Mohit Hosakatte Niranjana Murthy - mhosaka

Repository Link : https://github.com/nikki1234567/MovieRecommender

Notes	Grade	Evidence
Video	3	2min video of new functionality, showing a significant delta from prior.
Workload is spread over the whole team (one team member is often Xtimes more productive than the others...	3	based on additions and deletions of lines on github (https://github.com/nikki1234567/MovieRecommender/pulse)
Number of commits	3	82
Number of commits: by different people	3	in GH (https://github.com/nikki1234567/MovieRecommender/graphs/contributors)
Issues reports: there are many	3	in GH (https://github.com/nikki1234567/MovieRecommender/issues)
Issues are being closed	3	evidence in GH (https://github.com/nikki1234567/MovieRecommender/issues?q=is%3Ai
DOI badge: exists	3	https://zenodo.org/badge/DOI/10.5281/zenodo.14027294.svg
Docs: doco generated, format not ugly	3	in GH (https://github.com/nikki1234567/MovieRecommender/wiki)
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	doc page entries
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/nikki1234567/MovieRecommender/wiki (https://github.com/nikki1234567/MovieRecommender/wiki)
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	3	https://github.com/nikki1234567/MovieRecommender/blob/master/README.md (https://github.com/nikki1234567/MovieRecommender/blob/master/README.md)
Use of version control tools	3	
Use of style checkers	3	config files in GH showing your this formatter's config (https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/pyt
Use of code formatters.	3	config files iin GH showing this checker's config (https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/coc
Use of syntax checkers.	3	config files in GH (https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/pyt
Use of code coverage	3	config files in GH (https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/coc
Other automated analysis tools	3	https://github.com/nikki1234567/MovieRecommender/actions
Test cases exist	3	https://github.com/nikki1234567/MovieRecommender/tree/master/test
Test cases are routinely executed	3	github actions are being used to execute the test cases every time a commit happens. https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/test (https://github.com/nikki1234567/MovieRecommender/blob/master/.github/workflows/tes
The files CONTRIBUTING.md lists coding standards and lots of tips on how to extend the system without screwing things up	3	https://github.com/nikki1234567/MovieRecommender/blob/master/CONTRIBUTING.md (https://github.com/nikki1234567/MovieRecommender/blob/master/CONTRIBUTING.md
Issues are discussed before they are closed	3	https://github.com/nikki1234567/MovieRecommender/issues/10
Chat channel: exists	3	
Test cases: a large proportion of the issues related to handling failing cases.	3	If a test case fails, open an issue and fix . Also update the existing test cases (https://github.com/nikki1234567/MovieRecommender/issues/9)
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	https://github.com/nikki1234567/MovieRecommender/blob/master/requirements.txt (https://github.com/nikki1234567/MovieRecommender/blob/master/requirements.txt)
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	https://github.com/nikki1234567/MovieRecommender (https://github.com/nikki1234567/M
Short release cycles	3	https://github.com/nikki1234567/MovieRecommender/releases (https://github.com/nikki1234567/MovieRecommender/releases)
Does your website and documentation provide a clear, high-level overview of your software?	Yes	
Does your website and documentation clearly describe the type of user who should use your software?	Yes	

Notes	Grade	Evidence
Do you publish case studies to show how your software has been used by yourself and others?	Yes	
Is the name of your project/software unique?	Yes	
Is your project/software name free from trademark violations?	Yes	
Is your software available as a package that can be deployed without building it?	Yes	
Is your software available for free?	Yes	
Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository?	Yes	
Is your software hosted in an established, third-party repository like GitHub (https://github.com), BitBucket (https://bitbucket.org), LaunchPad (https://launchpad.net) or SourceForge (https://sourceforge.net)?	Yes	
Is your documentation clearly available on your website or within your software?	Yes	
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	
If you provide more extensive documentation, does this provide clear, step-by-step instructions on how to deploy and use your software?	Yes	
Do you provide a comprehensive guide to all your software's commands, functions and options?	Yes	
Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?	Yes	
If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?	Yes	
Do you store your documentation under revision control with your source code?	Yes	
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?	Yes	
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	
Does your project have an e-mail address or forum that is solely for supporting users?	Yes	
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	
Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?	Yes	
Is your software's architecture and design modular?	Yes	
Does your software use an accepted coding standard or convention?	Yes	

Notes	Grade	Evidence
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	
Does your software allow communications using open communications protocols? e.g. HTTP, FTP, XMPP, SOAP over HTTP, or domain-specific ones	Yes	
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?	Yes	
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	
Is each source code release a snapshot of the repository?	Yes	
Are releases tagged in the repository?	Yes	
Is there a branch of the repository that is always stable? (i.e. tests always pass, code always builds successfully)	Yes	
Do you back-up your repository?	Yes	
Do you provide publicly-available instructions for building your software from the source code?	Yes	
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)	Yes	
Do you provide publicly-available instructions for deploying your software?	Yes	
Does your documentation list all third-party dependencies?	Yes	
Does your documentation list the version number for all third-party dependencies?	Yes	
Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?	Yes	
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/)	Yes	
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	

Notes	Grade	Evidence
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?	Yes	
Are your test results publicly visible?	Yes	
Are all manually-run tests documented?	Yes	
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	
Does your website state how many projects and users are associated with your project?	Yes	
Do you provide success stories on your website?	Yes	
Do you list your important partners and collaborators on your website?	Yes	
Do you list your project's publications on your website or link to a resource where these are available?	Yes	
Do you list third-party publications that refer to your software on your website or link to a resource where these are available?	Yes	
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	
Do you accept contributions (e.g. bug fixes, enhancements, documentation updates, tutorials) from people who are not part of your project?	Yes	
Do you have a contributions policy?	Yes	
Is your contributions' policy publicly available?	Yes	
Do contributors keep the copyright/IP of their contributions?	Yes	
Does your website and documentation clearly state the copyright owners of your software and documentation?	Yes	
Does each of your source code files include a copyright statement?	Yes	
Does your website and documentation clearly state the licence of your software?	Yes	
Is your software released under an open source licence?	Yes	
Is your software released under an OSI-approved open-source licence?	Yes	
Does each of your source code files include a licence header?	Yes	
Do you have a recommended citation for your software?	Yes	
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	
Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?	Yes	

Notes	Grade	Evidence
Do you make timely announcements of the deprecation of components, APIs, etc.?	Yes	