International Journal of Computer Techniques–IJCT Volume 13 Issue 2, March-April - 2026
Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

CinemaNerd: A Web-Based Movie Explorer with User Authentication, Watchlist Management, and Review System

Harsh Singh[1], Prabhakar Mishra[2], Nikhil Singh[3], Rajveer Yadav[4] ,Kaushal Pratap Singh[5], Mudit Dubey[6]
([1,2,3,4]Student, Digvijai Nath P.G. College,Gorakhpur), ([5,6]Asistant Professor, Digvijai Nath P.G. College,Gorakhpur)

Abstract: This research paper presents a comprehensive study and project that elaborates on the development of CinemaNerd, a full-stack web application designed as a personal movie discoverer. The frontend is developed using HTML, CSS, and JavaScript, whereas the back-end REST API is implemented in Node.js and Express.js. Data is stored persistently in a JSON file on the server. The application allows users to access a carefully curated movie collection, search by title, rate movies on a scale of 1 to 10, maintain a personal watchlist, and leave public reviews. Authentication is performed using JSON Web Tokens (JWT), and passwords are hashed with bcrypt.js. The system ensures that authenticated users can only add to the watchlist and rate and review movies. All visitors can easily see reviews, which allows community-driven content. The project illustrates the real-world application of full-stack web development concepts, including REST API design, token-based authentication, file-based persistence, and responsive UI development.

Keywords — **CinemaNerd, Node.js, Express.js, JWT Authentication, REST API, Watchlist, Movie Review System, Responsive Web Design, bcryptjs, Full-Stack Development.**

I. INTRODUCTION

As digital streaming platforms and online movie communities gain popularity, the demand for personalised movie management tools is growing. Users desire to find movies, follow what they want to watch, rate movies based on their experience, and express their thoughts with others. CinemaNerd is designed to meet this requirement as a simple yet full-function web application.

The conventional methods for managing movies rely on third-party applications that may not be fully user-controlled. CinemaNerd solves this issue by offering an in-house system in which all data (user accounts, watchlists, ratings, and reviews) is handled by the application's backend. The project is easy to construct because it uses no heavy structures or cloud services, so it is simple to install, adjust, and expand.

It is composed of two parts: a frontend that is developed in vanilla HTML, CSS, and JavaScript, and a Node.js and Express backend. The two communicate through a RESTful API. The app will be responsive and can be used on desktop devices and mobile phones.

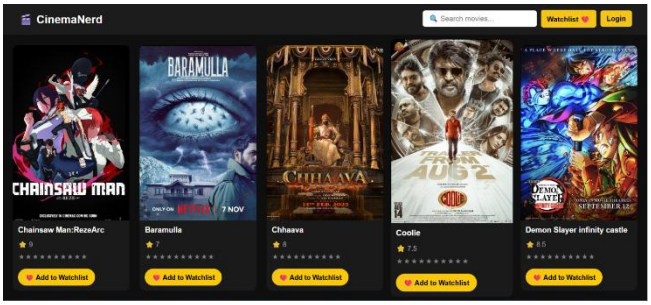

Fig. 1 home page

II. OBJECTIVES

The main objectives of the CinemaNerd project are as follows:

a. To create a useful web application enabling users to navigate and search a curated collection of movies.
b. To install a secure user authentication system with JWT tokens and bcrypt password hashing.
c. To allow users to have a personal watchlist that can be saved and accessed on the server.
d. To have a 10-star rating system with the user rating being stored persistently on the backend.
e. To create a community film review site where every registered person can leave a movie review that will be seen by everyone who comes to the site.
f. To create a fully-responsive frontend design that will be compatible with desktop, tablet and mobile screen sizes.
g. To illustrate the principles of REST API development, such as appropriate route design, middlewares, and error handling.

III. WORKING

The CinemaNerd system is based on a complex of modules. The modules handle a certain aspect of the application.

A. Movie Browsing and Search

When the user opens the application, the home page (index.html) loads movie information from a local movies.json file, which contains 29 movies with variables such as title, rating, poster, year, genre, and description. The movie display uses a responsive grid whose columns are auto-filled with CSS Grid. A search box lets you filter movies in real time by title using the JavaScript filter() method.

B. User Authentication

Users can sign up on the signup page by entering their name, email, and password. Before storing the password in users.json, it is hashed with bcryptjs using a 10-salt factor. The password entered is hashed when logging in and compared against the stored hash using bcrypt.compare(). Upon successful login, the server creates a JWT token that expires in 7 days and saves it in the browser's localStorage, along with the user's name and email.

C. Star Rating System

The movie cards have 10 stars that can be clicked. Once a logged-in user clicks a star, a POST request will be sent to /ratings with the movie title and rating value. The rating gets stored in ratings.json per user. Clicking the same star again sets the rating to 0, removing the rating (reset functionality). Logged-in users have ratings loaded on page load and are persisted across sessions and devices.

D. Watchlist Management

By clicking the Add to Watchlist button on a movie card, logged-in users can add any movie to their watchlist. The frontend POST request to /watchlist includes the entire movie object and the user's JWT token in the Authorisation header. The backend authenticates the token, handles duplicates, and then stores the movie with the user's email key in watchlist.json. On the watchlist page, users can view their watchlist and remove movies from it.

E. Review System

The detail page for each movie includes a review section with a form that allows anyone to post a comment. All logged-in users may enter and post a review, which will be sent to POST /reviews in the form of a movie title and a review. Each user is allowed to post only a single review for a movie — any subsequent post will replace the earlier one. The reviews are stored in reviews.json as an array of objects, each containing the reviewer's name, email address, review text, and date, under the movie title. Every review is publicly accessible via GET /reviews/:title and requires no authentication.

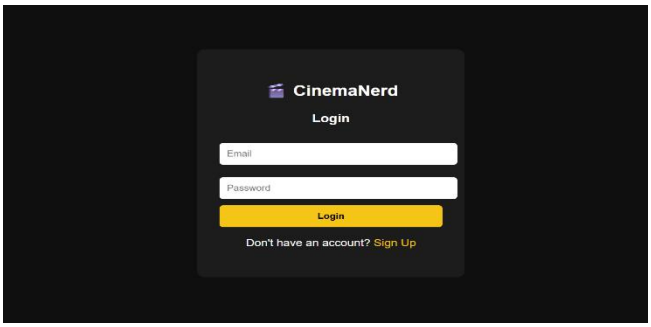

Fig . 2 login page

IV. SECURITY IMPLEMENTATION

The CinemaNerd system takes security into account. The following measures have been put in place:

1. **Password Hashing:** Before storing passwords, they are hashed with bcryptjs. No plain text passwords are stored, and user credentials are not compromised even when the data file is opened directly.
2. **JWT Authentication:** Authorised routes must have a valid JWT token in the Authorisation header. Each request is authenticated using JSON Web Tokens, and invalid or expired tokens are denied with a 401 error.
3. **XSS Prevention:** Movie detail content is displayed with textContent rather than innerHTML, where user-supplied or URL-based data is displayed, avoiding Cross-Site Scripting attacks.
4. **Input Validation:** All API endpoints validate incoming data before processing. Missing fields will return descriptive error messages with the appropriate HTTP status codes.
5. **Duplicate Prevention:** The system will verify the presence of emails when signing up and duplicate movies when adding to the watchlist, displaying clear error messages to the user.

V. FUTURE SCOPE

CinemaNerd can be improved and expanded with several new features and improvements:

- **Database Integration:** JSON file storage should be replaced with a more appropriate database like MongoDB or MySQL to achieve better scalability, concurrent access management, and query performance.
- **TMDB API Integration:** Replace the static movies.json file with live data of The Movie Database (TMDB) API to offer a bigger and more current movie catalogue with actual posters and metadata.
- **Cloud Deployment:** Deploy the backend to a platform like Railway, Render, or Heroku and the frontend to Vercel or Netlify, making the application available anywhere without a local server.
- **Genre and Year Filters:** Add dropdown filters by genre, year, and rating to enable users to find movies more easily.
- **Admin Panel:** Add a role-based administration interface to manage movies, users, and reviews directly in the browser.
- **Email Verification:** Introduce email-based account verification and password reset to enhance account security.

VI. CONCLUSIONS

CinemaNerd demonstrates how a full-stack web application can be developed using modern JavaScript technologies. The project encompasses

a broad spectrum of computer science concepts, including REST API design, token-based authentication, secure password management, file-based data storage, responsive UI development, and client-server communication.

The system offers a user-friendly experience, including real-time search, a persistent star rating, a personal watchlist, and a community review system, all seamlessly integrated. The codebase is clean, maintainable, and easy to extend because the frontend and backend are separated into independent modules.

The project emphasises the ability to create practical, real-world web applications without relying on heavy frameworks or paid cloud services, making it a perfect project to learn the basics of full-stack development.

ACKNOWLEDGMENT

The authors would like to thank Kaushal Pratap Singh for his guidance and support throughout the development of this project. We also acknowledge the open-source community for the libraries and tools that made this project possible.

REFERENCES

[1] Duckett, J. (2011). HTML and CSS: Design and Build Websites.
[2] Duckett, J. (2014). JavaScript and JQuery: Interactive Front-End Web Development.
[3] Haverbeke, M. (2018). Eloquent JavaScript.
[4] M. Haverbeke, Eloquent JavaScript, 3rd ed. No Starch Press, 2018.
[5] K. Simpson, You Don't Know JS. O'Reilly Media, 2015.
[6] Krug, S. (2014). Don’t Make Me Think.
[7] Bordwell, D., & Thompson, K. (2016). Film Art: An Introduction.
[8] Casciaro, M., & Mammino, L. (2020). Node.js Design Patterns.
[9] Ihrig, C. J. (2018). Pro Node.js for Developers.
[10] Martin, R. C. (2017). Clean Architecture.