**CINEMATECH**

**Introduction**

Cinematech is a modern video streaming platform designed to provide users with a seamless and interactive movie and TV show watching experience. The primary goal of Cinematech is to combine the convenience of streaming services with the engagement of a social platform, allowing users to discover and watch content and also to know about actors and people in film industry. This document outlines the architecture, design, and key features of the Cinematech project.

**Architecture**

* Cinematech is built on the MERN (MongoDB, Express.js, React.js, Node.js) stack, a popular choice for developing full-stack web applications.
* The frontend is developed using Create React App, Material UI, Swiper JS, React Router and Axios.
* The backend is based on Node JS runtime environment powered by Express.js, Express Validator, JWT (JSON Web Tokens), Mongoose (for MongoDB interactions), and Axios for external API requests.
* Communication between the frontend and backend occurs through RESTful API endpoints.

**Frontend Design**

The frontend of Cinematech is designed to provide a user-friendly and visually appealing interface. Components are organized into a hierarchy, with reusable components for common UI elements.

1. Bootstrap CSS framework is utilized for consistent and responsive design.
2. React Router v6 handles client-side routing for seamless navigation between pages.
3. Axios is employed for making HTTP requests to the backend API.

**Backend Design**

The backend of Cinematech consists of an Express.js REST API responsible for handling client requests.

1. Express Validator is used for input validation and sanitization to prevent security vulnerabilities.
2. JWT is employed for user authentication and authorization, ensuring secure access to protected routes.
3. Mongoose is utilized for interacting with the MongoDB database, providing a schema-based solution for data modeling.
4. Axios facilitates communication with external APIs, particularly The Movie Database (TMDb), for retrieving movie and TV show information.

**API Endpoints**

* GET /api/movies: Retrieves a list of movies from the external API.
* GET /api/movies/:id: Retrieves detailed information about a specific movie.
* GET /api/movies/search/:query: Searches for movies by title.

**Middleware:** Body-parser middleware will be used to parse incoming request bodies.

**Error Handling:** Error-handling middleware will be implemented to handle and respond to errors appropriately.

**Database Design**  
On the basis of functionality, we mainly require 3 models:

User Model:

|  |
| --- |
| **User Fields** |
| Username |
| Display Name |
| Password |
| Salt (for hashing password) |

|  |
| --- |
| **User Methods** |
| Set Password |
| Valid Password |

Favorite Model:

|  |
| --- |
| **Favorite Fields** |
| User |
| Media Type |
| Media ID |
| Media Title |
| Media Poster |
| Media Rate |

Review Model:

|  |
| --- |
| **Review Fields** |
| User |
| Content |
| Media Type |
| Media ID |
| Media Title |
| Media Poster |

**Relationships:**

User-Favorite Relationship:

* Each user can have multiple favorite movies or TV shows.
* The Favorite model references the User model through the user field.
* It is a one-to-many relationship where one user can have multiple favorite items.

User-Review Relationship:

* Each user can write multiple reviews for movies or TV shows.
* The Review model references the User model through the user field.
* It is a one-to-many relationship where one user can have multiple reviews.

**Features**

Cinematech offers a range of features to enhance the user experience:

•Sign In / Sign Up: Users can create an account or sign in securely using their credentials.

•Favorite List: Users can save or remove movies to/from their favorite list for easy access.

•Review System: Users can write reviews for movies, contributing to a dynamic community-driven rating system.

•Search: Cinematech provides a robust search functionality for discovering movies, series, and people.

**Responsive Design**

Cinematech is designed with responsiveness in mind, ensuring compatibility across different devices and screen sizes. The layout adjusts dynamically to provide an optimal viewing experience on desktops, tablets, and smartphones.

**Security**

JWT Authentication: User authentication is handled securely using JSON Web Tokens (JWT). Upon successful authentication, users receive a JWT token, which is included in subsequent requests to access protected resources.

Data Encryption: Sensitive user data, such as passwords and authentication tokens, are encrypted using cryptographic algorithms to prevent unauthorized access.

Third-Party Security: External APIs, such as The Movie Database (TMDb), are accessed securely using HTTPS and API keys, ensuring data confidentiality and integrity

**Deployment**

Cinematech is deployed using Vercel, a cloud platform for static sites and serverless functions. Vercel simplifies the deployment process by seamlessly integrating with Git repositories and providing automatic builds and deployments.

Version Control: The source code is managed using Git, with a repository hosted on GitHub.

**Conclusion**

In conclusion, Cinematech aims to provide users with an immersive and engaging video streaming experience. By combining advanced technology with intuitive design and robust features, Cinematech sets out to redefine the way users interact with movies and TV shows, fostering a vibrant community of cinephiles and enthusiasts.