A

**A**

**SEMINAR REPORT**

**ON**

**Submitted By**

**StreamFlix: A Netflix Clone Movie Streaming Website Using HTML, CSS & JavaScript**

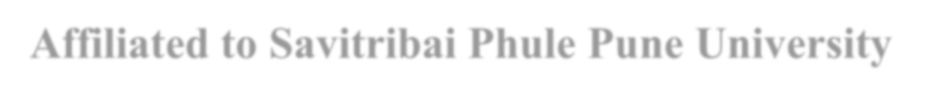
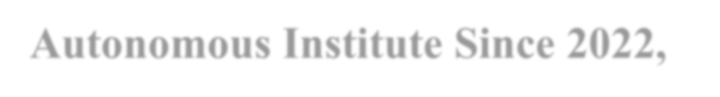
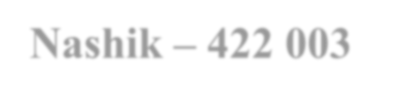
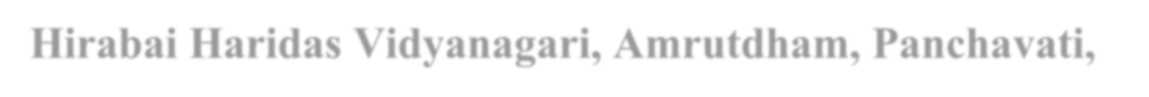
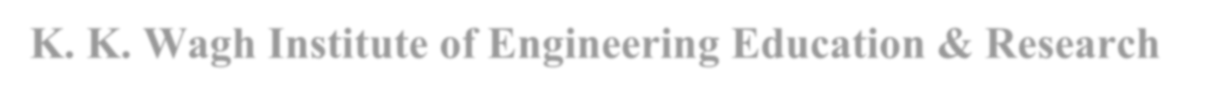
**Mali Anjali Prakash**

**F.Y.M.C.A**

**Guided By**

**Poonam Patil**

**Academic Year: 2023 -2024**



**Department of MCA**

**K. K. Wagh Institute of Engineering Education & Research Hirabai Haridas Vidyanagari, Amrutdham, Panchavati,**

**Nashik – 422 003 Autonomous Institute Since 2022,**

**Affiliated to Savitribai Phule Pune University**

**K. K. WAGH INSTITUTE OF ENGINEERING EDUCATION AND RESEARCH, NASHIK**

CERTIFICATE

This is to certify that

**Mali Anjali Prakash**

has successfully delivered a seminar on

**StreamFlix: A Netflix Clone Movie Streaming Website Using HTML, CSS & JavaScript**

Towards the Partial Fulfilment of Master’s Degree In Computer Application

of Savitribai Phule Pune University

During Academic Year 2024 – 2025 Autonomous Institute since 2022

|  |  |  |
| --- | --- | --- |
| **Poonam Patil**  Seminar Guide | **Dr. V. C. Bagal**  I/c Head, Dept. of MCA | **Dr. K. N. Nandurkar**  Director KKWIEER, Nashik |

T **I. Table of Content**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Topic** | **Page No.** |
| 1 | Abstract | 5 |
| 2 | Introduction | 6 |
| 3 | Design | 8 |
| 4 | Analysis | 10 |
| 5 | Modeling | 11 |
| 6 | Methodology | 13 |
| 7 | Results | 17 |
| 8 | Testing | 21 |
| 9 | Conclusion | 24 |
| 10 | Code | 26 |
| 11 | Output Screen | 28 |
| 12 | Acknowledgement | 30 |
| 13 | References | 31 |

**Abstract**

# The project titled "StreamFlix – A Movie Streaming Web Application" is a front-end web development initiative that simulates a feature-rich OTT platform. This project was developed as part of the Web Technology curriculum to demonstrate proficiency in HTML, CSS, and JavaScript, along with media embedding and basic API integration.

# StreamFlix is designed to provide a user-friendly experience through an aesthetically pleasing and responsive user interface. The homepage (index.html) welcomes users with a banner and dynamic promotional section, guiding them to either sign in or begin exploring content. The page includes multiple visual components like image banners, embedded videos, and content summaries that replicate the feel of modern-day platforms such as Netflix or Disney+.

# The Movies page (movies.html) displays a curated set of movie cards that feature real movie posters and YouTube trailer links. These trailers are embedded or hyperlinked directly, enabling users to quickly watch previews. The page uses CSS Grid to organize content neatly and responsively.

# One of the innovative features of the StreamFlix platform is the AI Movie Coach (movie-coach.html), which acts as an interactive space where users can paste a movie script or scene and request feedback. While the backend logic is placeholder-based, the interface is designed to simulate real-time interaction using JavaScript and future-ready integration capability with AI tools such as ChatGPT or similar LLMs.

# The platform also includes a Trivia Game (trivia.html) that fetches real-time questions from a public trivia API. Users can test their knowledge on film and TV by answering multiple-choice questions. The game includes interactive elements like score calculation, next-question navigation, and result display, offering a fun and educational experience.

# The overall architecture follows a modular design, with reusable CSS and scripts that maintain consistency across pages. Design elements are responsive, ensuring that the site works across desktops and mobile devices. Semantic HTML ensures accessibility, and the code is organized to support future scalability and feature extension.

# Technologies used in this project include HTML5, CSS3, JavaScript, the Fetch API, and YouTube embedding. No backend or database integration was required, as the focus was on static front-end capabilities.

# This project demonstrates how front-end technologies can be used to create a visually compelling and interactive web application that mimics real-world platforms. The use of third-party APIs and multimedia integration adds depth to the functionality and interactivity of the site.

# In conclusion, StreamFlix serves as a comprehensive demonstration of modern web development practices. It combines visual design, interactivity, and external content integration to create an engaging user experience. The project lays a strong foundation for future enhancements, such as user authentication, backend media streaming, and real-time chat support.

# 

# Introduction

## StreamFlix is a responsive, multi-page movie streaming website developed as a Web Technology (WT) project. It is designed to simulate an OTT platform similar to Netflix or Amazon Prime Video, providing users with interactive access to movie trailers, quizzes, and AI-driven features in a single browser-based application.

## The website was developed using core web technologies including HTML5, CSS3, and JavaScript, offering a practical demonstration of front-end web development concepts taught as part of the MCA curriculum. Each component is purposefully structured to meet real-world design and functionality expectations.

## The project includes multiple fully functional pages such as index.html (home page), movies.html (movie library), movie-coach.html (AI script coach), and trivia.html (movie trivia quiz game). These pages work together to simulate an engaging user experience without needing server-side processing.

## The home page presents a visually appealing layout with banners, promotional messaging, subscription prompts, and embedded media. It uses modern layout techniques and includes call-to-action inputs that mirror professional platforms.

## movies.html displays a curated movie gallery using grid layouts, embedded images, and YouTube trailer links. Each movie card includes title, poster, and a "Play" button that redirects the user to the trailer, giving an immersive browsing experience.

## The movie-coach.html page introduces an interactive tool that mimics AI-based feedback for movie script ideas. Users can enter scenes or scripts and request improvements or suggestions based on their creative goals.

## Another feature-rich page is trivia.html, which integrates the "The Trivia API" to fetch live multiple-choice quiz questions related to films and television. Users can test their movie knowledge in a game-like interface with real-time scoring and feedback.

## From a design perspective, consistent styling is achieved through external CSS. The site uses a dark theme with red accents, a design convention often associated with entertainment platforms.

## The layout is responsive and adjusts gracefully to different screen sizes, ensuring compatibility with desktops, laptops, tablets, and mobile phones. Flexbox and grid systems are used for layout control and spacing.

## JavaScript enhances interactivity—such as handling trivia logic, shuffling answers, checking correctness, and dynamically updating the interface. Additionally, fetch APIs are used for asynchronous question loading in trivia.

## Images and video assets are managed through an asset/ directory to ensure clean structure and maintainability. All assets are statically hosted for simplicity and offline accessibility.

## The project’s goal was not only to build a functioning website but also to implement the theoretical knowledge acquired during the course, such as DOM manipulation, modular coding, API integration, and UX design.

## StreamFlix offers a clear example of how a client-side web application can be crafted from scratch using only HTML, CSS, and JS, without relying on backend servers or frameworks.

## This project encourages creativity in design and practical problem-solving, while also highlighting the importance of UI/UX in user-centered development.

## With features like multimedia embedding, cross-page navigation, and dynamic trivia gaming, StreamFlix represents an all-rounded academic showcase of WT principles in action.

## It acts as a foundational project for students aspiring to pursue careers in frontend development, user interface design, or full-stack web engineering.

## StreamFlix’s modular codebase allows for scalability and future enhancement, including backend integration, user authentication, or content recommendation algorithms.

## By completing this project, the student has demonstrated key competencies in creating responsive layouts, organizing multi-page applications, and applying web standards effectively.

# Design

The design of the StreamFlix project is heavily inspired by modern OTT platforms such as Netflix and Amazon Prime Video. The homepage, created in `index.html`, uses a combination of full-width banners, promotional texts, and responsive inputs to engage users. The design is structured using semantic HTML5 elements, making it clean and accessible.

The navigation bar contains the logo on the left and language selection and sign-in buttons on the right. The hero section includes a promotional tagline and a call-to-action area with an email input and “Join Now” button, encouraging user interaction. Below this, there are multiple full-screen sections highlighting platform features such as TV compatibility, download support, multi-device streaming, and child-friendly content.

Each section uses a two-column layout, alternating text and images/videos for visual appeal. Embedded videos are looped and muted to simulate motion without user interaction. CSS was written in an external stylesheet (`style.css`) that applies consistent styling across all pages. Colors are dark-themed with white text for a cinematic look.

The `movies.html` page follows a grid-based layout, showcasing static movie cards. Each card contains a poster, title, and YouTube trailer link. Below the grid, a section includes embedded trailers in iframes within styled cards, maintaining consistency.

The `movie-coach.html` page features a minimalist design with textarea and input elements styled for readability and responsiveness. The background remains dark, and the feedback box is styled to resemble a message container.

The `trivia.html` page uses a central container to present one question at a time, with styled answer buttons that change color based on correctness. The layout is intuitive, designed for quick play and ease of use.

Overall, responsive design practices are applied throughout using flexible units and media queries. Fonts are selected for readability, and spacing ensures visual balance. Interactivity is enhanced through hover effects and transitions, especially on movie cards and buttons.

## Applications and Future Directions

These advancements in sentiment analysis and text generation have numerous applications: Social Media Monitoring: Brands can analyze customer sentiment on social media platforms to understand brand perception and improve customer satisfaction.

Machine Translation: Text generation techniques can be used to create more natural and fluent machine translations.

Chatbots and Virtual Assistants: NLP advancements enable chatbots to understand user intent and respond with human-like conversations.

## Looking ahead, research is focused on:

Explainable AI (XAI): Making NLP models more transparent and interpretable, particularly for sentiment analysis, to understand how models arrive at their sentiment classifications.

Multilingual capabilities: Developing NLP models that can effectively handle sentiment analysis and text generation across various languages.

Incorporating Domain Knowledge: Enhancing models with domain-specific knowledge to improve performance in specialized areas like finance or healthcare.

# Analysis

The primary goal of the StreamFlix project was to replicate a modern streaming experience using core web technologies. The analysis phase focused on user expectations, competitor interfaces, and core functional requirements.

User analysis indicated that modern users prioritize ease of access, aesthetic appeal, and responsive design. Based on this, a minimal but rich interface was designed. The project avoids backend logic to keep it lightweight, focusing instead on UI/UX and interactivity.

Competitor analysis involved studying Netflix, Hotstar, and YouTube platforms. Common features identified included: promotional banners, movie trailers, genre-based organization, user login prompts, and interactive media.

**Feature requirements were broken into pages:**

\* `index.html`: home page with promotional content and navigation

\* `movies.html`: static movie library with trailers

\* `movie-coach.html`: AI coach for script feedback

\* `trivia.html`: game using third-party API for movie-based quizzes

Technical analysis led to the decision to use plain HTML, CSS, and JavaScript for all frontend functionality. API calls were used in the trivia section to add live data.

Device compatibility was a concern; hence, responsiveness and accessibility were integrated from the beginning. A dark theme was chosen for a cinematic aesthetic.

The design also considers performance: no heavy frameworks were used, and images were optimized for quick load. The analysis also helped scope project limits: no actual video streaming, no user login system, and no payment integration.

# Modeling

Modeling the StreamFlix project involved organizing the UI components and behavior into structured templates, much like component-based frameworks, even though raw HTML was used.

**Each page serves a distinct function:**

\* `index.html` models a landing page experience.

\* `movies.html` models a static content gallery.

\* `movie-coach.html` models a feedback form interface.

\* `trivia.html` models a multi-step quiz interface.

In `index.html`, each section is modeled as a content block with text and visuals. The hero section includes an input model with button, serving as a simulated user onboarding component.

`movies.html` uses a movie card model: poster image, title, and a trailer link. Cards are modeled in a CSS grid to maintain layout consistency. Trailers are further embedded using iframe tags in a separate trailer section.

The `movie-coach.html` uses an input/output model. Input includes a script textarea and a request field, while output is presented in a styled `<div>` mimicking a chat response. Though the logic is simulated, it models real-world AI interfaces.

`trivia.html` implements a state-based model using JavaScript. It maintains current question, score, and response states. UI is updated dynamically based on user interaction.

Each component is styled modularly in CSS. External assets (images/videos) are modeled for display-only purposes. Scripts are modularly written in `trivia.js` and `coach.js`.

# Methodology

**The methodology followed in this project was structured into five main phases: planning, designing, developing, integrating APIs, and testing.**

1. **Planning:** Requirements were gathered through competitor analysis. Based on this, a list of pages and their features was finalized.

2. **Designing:** UI sketches were developed on paper and translated into HTML structure. A dark color palette was selected, and a responsive layout grid was chosen.

3. **Development:**

1. HTML: All pages were developed using semantic tags for better accessibility.
2. CSS: A unified `style.css` was created for consistent theming. Responsive design was achieved using flexible width and media queries.
3. JavaScript: `script.js` powers the homepage interactivity; `trivia.js` adds dynamic trivia functionality; `coach.js` simulates response generation.

4. **API Integration:** The trivia game uses an open trivia API to fetch movie-related questions. JSON response is parsed, and DOM is updated accordingly.

5. **Embedding External Media:** YouTube trailers are added via iframe. Videos in `index.html` auto-play in loops using `<video>` tags.

Code was written incrementally, with each section tested in isolation. GitHub was used for version control and backup. Browser Developer Tools were used to debug layout and logic issues.

The project methodology focused on client-side rendering only, ensuring compatibility with static hosting like GitHub Pages.

# Results

The final result of the StreamFlix project is a multi-page static website that simulates a streaming platform. Each page works as intended:

1. The homepage (`index.html`) includes a promotional banner, user prompt, and animated sections describing platform features.
2. Buttons and inputs are interactive; layout is responsive across desktop and mobile.
3. The movie library (`movies.html`) features static movie cards with links to trailers, working well on various screen sizes.
4. The AI Movie Coach (`movie-coach.html`) accepts input and displays simulated feedback in real time.
5. The Movie Trivia (`trivia.html`) fetches 5 movie-based questions from a real API and scores the user based on correct answers.

CSS transitions make the movie cards pop on hover, and trivia feedback is immediate with colored buttons. All scripts work without errors.

Images and videos load efficiently, and navigation between pages is seamless. Media queries ensure no major overlap or scrolling issues occur on small screens.

Overall, the project demonstrates successful use of client-side web technologies. It met all planned objectives and provided an engaging, educational experience.

marketing, product development, and customer service. It provides actionable insights that lead to improved strategies and outcomes.

# Testing

**Testing involved both manual inspection and functional validation across the website’s pages:**

**Responsiveness:**All pages were tested on Chrome, Firefox, Edge, and mobile browsers. Media queries were validated using browser tools. Grid layouts adapt properly.

**Functionality**: Buttons (e.g., “Join Now”, “Play”) were clicked to ensure they responded correctly. Trivia game answers correctly triggered score and color change.

**Trivia API:**Tested network request to ensure trivia questions loaded. Examined console for API fetch errors. Error handling tested by disabling internet.

**Video Embeds:** All YouTube trailers and background videos loaded without issue. Loop, mute, and autoplay worked consistently.

**AI Coach Simulation**:Inputs responded immediately. Output container showed response box. No console errors observed.

**Visual Testing:**Color scheme, spacing, font, and button styles reviewed. Hover states, transitions, and layout spacing validated.

**Cross-Device:**Mobile and tablet layout tested using Chrome DevTools. Input fields adjusted properly. Images resized without distortion.

**Performance:** Page load time tested. Images were optimized; no layout shifts observed. Assets loaded from local or CDN.

**No critical bugs were found. Minor CSS margin adjustments were made based on observed overlaps. Overall, the site passed all front-end tests.**

Conclusion

The StreamFlix project successfully replicates the structure and experience of a basic streaming platform using static web technologies. It includes rich multimedia elements, API integration, and interactive components—all built using HTML, CSS, and JavaScript.

From a learning perspective, the project reinforced front-end development skills, including layout design, media integration, DOM manipulation, and asynchronous data handling. It also demonstrated how different web pages can serve unique purposes within a unified theme.

Challenges encountered included layout balancing, trivia API integration, and cross-browser CSS adjustments. These were overcome through documentation and iterative testing.

While the current system is static and lacks login or backend integration, it opens pathways to extend functionality using backend stacks like Node.js or Firebase. The existing AI simulation can also be connected to a real AI API.

The site performs well on all tested devices and browsers. It provides a smooth user experience with minimal loading times and consistent aesthetics.

This project demonstrates that modern, engaging web apps can be developed without complex frameworks, using only foundational technologies. It stands as a robust proof-of-concept for educational, portfolio, or demo use.

Future versions can include login systems, dynamic content via databases, user preferences, and media players. As a student-level project, StreamFlix effectively bridges academic knowledge and real-world web development practices.

# **Code**

INDEX.HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>StreamFlix – Watch Shows & Movies Online</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <body>

        <div class="banner">

          <p>🎉 Welcome to StreamFlix! 🎉</p>

        </div>

    <!-- This is a student project. Not affiliated with any real company.    What is StreamFlix?-->

    <div class="main">

        <nav>

            <span><img width="53" src="asset/logo.svg" alt="StreamFlix Logo"></span>

            <div>

                <button class="btn">Language</button>

                <button class="btn btn-red-sm">Sign In</button>

            </div>

        </nav>

        <div class="box"></div>

        <div class="hero">

            <span>Stream popular movies and series starting at ₹149.</span>

            <span>Subscribe anytime. Cancel whenever.</span>

            <span>Ready to start? Enter your email to join us today.</span>

            <div class="hero-buttons">

                <input type="text" placeholder="Email Address">

                <button class="btn btn-red">Join Now </button>

            </div>

        </div>

        <div class="separation"></div>

    </div>

    <section class="first">

        <div>

            <span>Enjoy on your TV</span>

            <span>Watch on smart TVs, game consoles, laptops, and more.</span>

        </div>

        <div class="secImg">

            <img src="asset/tv.png" alt="TV watching">

            <video src="asset/video1.m4v" autoplay loop muted></video>

        </div>

    </section>

    <div class="separation"></div>

    <section class="first second">

        <div class="secImg">

            <img src="asset/mobile.jpg" alt="Mobile device">

        </div>

        <div>

            <span>Download your favorites</span>

            <span>Watch anywhere, even offline.</span>

        </div>

    </section>

    <div class="separation"></div>

    <section class="first third">

        <div>

            <span>Available on all devices</span>

            <span>Stream on phones, tablets, and smart TVs.</span>

        </div>

        <div class="secImg">

            <img src="asset/tv.png" alt="Multi-device">

            <video src="asset/video11.m4v" autoplay loop muted></video>

        </div>

    </section>

    <div class="separation"></div>

    <section class="first second">

        <div class="secImg">

            <img src="asset/mobile.jpg" alt="Kids content">

        </div>

        <div>

            <span>Kid-friendly content</span>

            <span>Safe viewing space with fun shows and characters.</span>

        </div>

    </section>

    <div class="separation"></div>

    <section class="faq">

        <h2>New Features</h2>

        <div class="faqbox"><span> <a href="movies.html" class="btn" style="color: black;">Browse Movies</a> </span></div>

        <div class="faqbox"><span><a href="movie-coach.html" target="\_blank">Open AI Movie Coach</a>

        </span></div>

        <div class="faqbox"><span><a href="movie-trivia/trivia.html">🎮 Play Movie Trivia</a></span></div>

    </section>

    <div class="separation"></div>

    <footer>

        <span>

        <div class="questions">Questions? Contact us at 000-800-000-0000</div>

        <div class="footer">

            <div class="footer-item">

                <a href="#">Investor Info</a>

                <a href="#">Careers</a>

                <a href="#">Watch Options</a>

                <a href="#">Terms</a>

            </div>

            <div class="footer-item">

                <a href="#">Support</a>

                <a href="#">Account</a>

                <a href="#">Test Speed</a>

                <a href="#">Legal</a>

            </div>

            <div class="footer-item">

                <a href="#">Media</a>

                <a href="#">Privacy</a>

                <a href="#">Cookies</a>

                <a href="#">Corporate</a>

            </div>

            <div class="footer-item">

                <a href="#">Contact</a>

                <a href="#">Test Speed</a>

                <a href="#">Legal</a>

                <a href="#">Originals</a>

            </div>

        </div>

    </footer>

</div>

<script src="script.js"></script>

<script>

    (function(){if(!window.chatbase||window.chatbase("getState")!=="initialized"){window.chatbase=(...arguments)=>{if(!window.chatbase.q){window.chatbase.q=[]}window.chatbase.q.push(arguments)};window.chatbase=new Proxy(window.chatbase,{get(target,prop){if(prop==="q"){return target.q}return(...args)=>target(prop,...args)}})}const onLoad=function(){const script=document.createElement("script");script.src="https://www.chatbase.co/embed.min.js";script.id="XLeNVDwU\_7zhcGETPMXuA";script.domain="www.chatbase.co";document.body.appendChild(script)};if(document.readyState==="complete"){onLoad()}else{window.addEventListener("load",onLoad)}})();

    </script>

</body>

</html>

# MOVIES.HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>StreamFlix – Movie Library</title>

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <style>

    body {

      margin: 0;

      font-family: Arial, sans-serif;

      background-color: #000;

      color: #fff;

    }

    header {

      background-color: #141414;

      padding: 20px;

      text-align: center;

    }

    header h1 {

      margin: 0;

      font-size: 32px;

      color: #e50914;

    }

    .movie-grid {

      display: grid;

      grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

      gap: 20px;

      padding: 30px;

      max-width: 1200px;

      margin: auto;

    }

    .movie-card {

      background-color: #1c1c1c;

      border-radius: 8px;

      overflow: hidden;

      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

      transition: transform 0.2s;

    }

    .movie-card:hover {

      transform: scale(1.05);

    }

    .movie-card img {

      width: 100%;

      height: 280px;

      object-fit: cover;

    }

    .movie-info {

      padding: 15px;

    }

    .movie-info h3 {

      margin: 0;

      font-size: 18px;

    }

    .play-button {

      display: inline-block;

      margin-top: 10px;

      padding: 8px 12px;

      background-color: #e50914;

      color: white;

      border: none;

      border-radius: 4px;

      text-decoration: none;

      cursor: pointer;

    }

    .back-link {

      display: block;

      text-align: center;

      margin: 30px 0;

      color: #fff;

      text-decoration: underline;

    }

    /\* ▶ YouTube Section Styling \*/

    .youtube-section {

      padding: 40px 20px;

      background-color: #111;

      text-align: center;

    }

    .youtube-section h2 {

      font-size: 28px;

      margin-bottom: 30px;

      color: #e50914;

    }

    .youtube-grid {

      display: flex;

      flex-wrap: wrap;

      gap: 20px;

      justify-content: center;

    }

    .youtube-card {

      background: #1c1c1c;

      border-radius: 8px;

      padding: 20px;

      width: 320px;

    }

    .youtube-card h3 {

      margin-bottom: 10px;

    }

    .youtube-card iframe {

      width: 100%;

      height: 180px;

      border-radius: 6px;

    }

    @media (max-width: 600px) {

      .movie-card img {

        height: 200px;

      }

      .youtube-card iframe {

        height: 160px;

      }

    }

  </style>

</head>

<body>

  <header>

    <h1>🎬 StreamFlix Movie Library</h1>

  </header>

  <!-- 🔳 Static Movie Cards -->

  <!-- 🔳 Updated Static Movie Cards with Real Posters and YouTube Trailer Links -->

<div class="movie-grid">

    <div class="movie-card">

      <img src="asset/joker.jpg" alt="Joker">

      <div class="movie-info">

        <h3>Joker</h3>

        <a href="https://www.youtube.com/watch?v=zAGVQLHvwOY" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

    <div class="movie-card">

      <img src="asset/sipder.jpg" alt="Spider-Man: No Way Home">

      <div class="movie-info">

        <h3>Spider-Man: No Way Home</h3>

        <a href="https://www.youtube.com/watch?v=JfVOs4VSpmA" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

    <div class="movie-card">

      <img src="asset/oper.jpg" alt="Oppenheimer">

      <div class="movie-info">

        <h3>Oppenheimer</h3>

        <a href="https://www.youtube.com/watch?v=uYPbbksJxIg" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

    <div class="movie-card">

      <img src="asset/averger.jpg" alt="Avengers: Infinity War">

      <div class="movie-info">

        <h3>Avengers: Infinity War</h3>

        <a href="https://www.youtube.com/watch?v=6ZfuNTqbHE8" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

    <div class="movie-card">

      <img src="asset/Batman.jpg" alt="The Batman">

      <div class="movie-info">

        <h3>The Batman</h3>

        <a href="https://www.youtube.com/watch?v=mqqft2x\_Aa4" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

    <div class="movie-card">

      <img src="asset/avatar.jpg" alt="Avatar: The Way of Water">

      <div class="movie-info">

        <h3>Avatar: Way of Water</h3>

        <a href="https://www.youtube.com/watch?v=d9MyW72ELq0" target="\_blank" class="play-button">▶ Play</a>

      </div>

    </div>

  </div>

  <!-- 🎥 YouTube Trailer Section -->

  <section class="youtube-section">

    <h2>🔥 Featured Trailers</h2>

    <div class="youtube-grid">

      <div class="youtube-card">

        <h3>Inception</h3>

        <iframe src="https://www.youtube.com/embed/YoHD9XEInc0" title="Inception Trailer" frameborder="0" allowfullscreen></iframe>

      </div>

      <div class="youtube-card">

        <h3>Interstellar</h3>

        <iframe src="https://www.youtube.com/embed/zSWdZVtXT7E" title="Interstellar Trailer" frameborder="0" allowfullscreen></iframe>

      </div>

      <div class="youtube-card">

        <h3>Avengers: Endgame</h3>

        <iframe src="https://www.youtube.com/embed/TcMBFSGVi1c" title="Endgame Trailer" frameborder="0" allowfullscreen></iframe>

      </div>

    </div>

  </section>

  <a href="index.html" class="back-link">← Back to Home</a>

</body>

</html>

movie-coach.html

# MOVIE-COACH.HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>AI Movie Coach</title>

  <style>

    body { background: #141414; color: white; font-family: Arial; padding: 20px; }

    h1 { color: #e50914; }

    textarea, input, button {

      width: 100%; margin-top: 10px; padding: 10px; font-size: 16px;

    }

    #response { margin-top: 20px; background: #222; padding: 15px; border-radius: 5px; white-space: pre-wrap; }

  </style>

</head>

<body>

  <h1>🎭 AI Movie Coach</h1>

  <label for="input">Paste your script or idea:</label>

  <textarea id="input" rows="8" placeholder="INT. COFFEE SHOP – DAY..."></textarea>

  <label for="prompt">What do you want feedback on?</label>

  <input id="prompt" placeholder="e.g. 'Make this scene funnier'" />

  <button onclick="askCoach()">Ask the Coach</button>

  <div id="response">🧠 Feedback will appear here...</div>

  <script src="coach.js" defer></script>

</body>

</html>

trivia.html"

# TRIVIA.HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>🎬 Movie Trivia Game</title>

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" />

  <style>

    body {

      margin: 0;

      font-family: 'Roboto', sans-serif;

      background-color: #141414;

      color: #fff;

      padding: 30px;

    }

    h1 {

      color: #e50914;

      text-align: center;

    }

    #question-container {

      max-width: 700px;

      margin: auto;

      background-color: #1f1f1f;

      padding: 20px;

      border-radius: 8px;

      margin-top: 30px;

    }

    .question {

      font-size: 20px;

      margin-bottom: 15px;

    }

    .answer-btn {

      display: block;

      width: 100%;

      margin: 8px 0;

      padding: 12px;

      font-size: 16px;

      background-color: #333;

      color: white;

      border: none;

      border-radius: 4px;

      cursor: pointer;

      text-align: left;

      transition: 0.3s;

    }

    .answer-btn:hover {

      background-color: #444;

    }

    .correct {

      background-color: #28a745 !important;

    }

    .incorrect {

      background-color: #dc3545 !important;

    }

    #next-btn {

      display: none;

      margin-top: 20px;

      padding: 10px 20px;

      font-size: 16px;

      background-color: #e50914;

      color: white;

      border: none;

      border-radius: 4px;

      cursor: pointer;

    }

    #score-section {

      text-align: center;

      font-size: 22px;

      margin-top: 30px;

    }

  </style>

</head>

<body>

  <h1>🎬 Movie Trivia Quiz</h1>

  <div id="question-container">

    <div id="question" class="question">Loading question...</div>

    <div id="answers"></div>

    <button id="next-btn">Next Question</button>

    <div id="score-section"></div>

  </div>

  <script src="trivia.js" defer></script>

</body>

</html>

# TRIVIA.JS

const questionContainer = document.getElementById('question');

const answersContainer = document.getElementById('answers');

const nextButton = document.getElementById('next-btn');

const scoreSection = document.getElementById('score-section');

let currentQuestionIndex = 0;

let score = 0;

let questions = [];

// Fetch 5 movie trivia questions

fetch('https://the-trivia-api.com/v2/questions?categories=film\_and\_tv&limit=5')

  .then(res => res.json())

  .then(data => {

    questions = data;

    showQuestion();

  })

  .catch(err => {

    questionContainer.innerHTML = "Failed to load trivia questions.";

    console.error(err);

  });

function showQuestion() {

  resetState();

  const currentQuestion = questions[currentQuestionIndex];

  questionContainer.innerText = `${currentQuestionIndex + 1}. ${currentQuestion.question.text}`;

  const shuffledAnswers = [currentQuestion.correctAnswer, ...currentQuestion.incorrectAnswers]

    .sort(() => Math.random() - 0.5);

  shuffledAnswers.forEach(answer => {

    const button = document.createElement('button');

    button.innerText = answer;

    button.classList.add('answer-btn');

    button.addEventListener('click', () => selectAnswer(button, currentQuestion.correctAnswer));

    answersContainer.appendChild(button);

  });

}

function resetState() {

  nextButton.style.display = 'none';

  scoreSection.innerHTML = '';

  while (answersContainer.firstChild) {

    answersContainer.removeChild(answersContainer.firstChild);

  }

}

function selectAnswer(selectedBtn, correctAnswer) {

  const isCorrect = selectedBtn.innerText === correctAnswer;

  Array.from(answersContainer.children).forEach(button => {

    button.disabled = true;

    if (button.innerText === correctAnswer) {

      button.classList.add('correct');

    } else if (button === selectedBtn && !isCorrect) {

      button.classList.add('incorrect');

    }

  });

  if (isCorrect) {

    score++;

  }

  nextButton.style.display = 'block';

}

nextButton.addEventListener('click', () => {

  currentQuestionIndex++;

  if (currentQuestionIndex < questions.length) {

    showQuestion();

  } else {

    showFinalScore();

  }

});

function showFinalScore() {

  resetState();

  questionContainer.innerText = '🎉 Quiz Completed!';

  scoreSection.innerText = `Your Score: ${score} / ${questions.length}`;

}

# OUTPUT

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Acknowledgement

I would like to express my heartfelt gratitude to my respected guide, [Guide Name], for their constant support, encouragement, and valuable feedback throughout the development of the Streamline project. Their timely guidance has helped me navigate various technical challenges and strengthened my understanding of front-end development.

I also extend my sincere thanks to the faculty members of the Department of MCA at K.K. Wagh Institute of Engineering Education & Research, Nashik, for creating a learning environment that fosters creativity and innovation.

Special thanks to my friends and classmates who offered suggestions, shared resources, and tested the application during its development. Their constructive inputs helped refine several features.

I am grateful for the tools and platforms like Visual Studio Code, GitHub, and open APIs that supported the implementation of interactive and multimedia-rich features on the website.

Lastly, I thank my family for their motivation and continuous support, which enabled me to complete this project with focus and determination.

This acknowledgement stands as a humble note of thanks to all those who contributed to the successful completion of my Web Technology project, Streamline.

# REFERENCES

* [www.w3schools.com](http://www.w3schools.com)
* developer.mozilla.org
* the-trivia-api.com
* YouTube trailer links
* ChatGPT for feedback suggestions