A Field Project Report on

Movie Search Application

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE and ENGINEERING
by

Y Siva Rama Krishna	(231FA04E82)
N Charishma	(231FA04F31)
M Bhadry	(231FA04F33)
G Jaya Varshini	(231FA04F45)
G Surya Reddy	(231FA04F49)

Under the Guidance of
Ch.Swarana Lalitha
Assistant Professor, CSE

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)
Vadlamudi, Guntur -522213, INDIA.

April, 2025

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project entitled *"Movie Search Application"* is being submitted by **[Y Siva Rama Krishna], [231FA04E82], [N Charishma], [231FA04F31], [M Bhadry], [231FA04F33], [G Jaya Varshini], [231FA04F45],** and**[G Surya Reddy], [231FA04F49]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at the Department of Computer Science and Engineering, Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

Guide

Project Review Committee

HoD, CSE

HoD
Dept. of Computer Science & Engine
VFSTR Deemed to be Univers
VADLAMUDI - 5

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date: 26/04/2025

We hereby declare that the work presented in the field project titled "**Movie Search Application**" is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Ch. Swarna Lalitha Asst.prof, Department of CSE** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India

Y Siva Rama Krishna	231FA04E82	Y. S. R. Krishna
N Charishma	231FA04F31	Charishma
M Bhadry	231FA04F33	Bhadry
G Jaya Varshini	231FA04G45	Jaya Varshini
G Surya Reddy	231FA04F49	Surya

CONTENTS

1.INTRODUCTION

A movie search application is a software tool or platform designed to help users discover, explore, and learn more about movies. These applications typically allow users to search for films based on various criteria such as title, genre, release year, cast, director, or even specific keywords. Some movie search apps provide additional features such as user reviews, ratings, trailers, showtimes, and recommendations based on viewing preferences.

1.1 Problem Definition

A movie search application aims to address these issues by providing a comprehensive, user-friendly, and personalized platform that allows users to search, filter, and discover movies quickly and efficiently, helping them find films that best match their interests and viewing history.

1.2 Existing System

1.2.1 Standalone Movie Databases

IMDb (Internet Movie Database):

IMDb is one of the largest and most well-known movie databases. It provides users with information about movies, TV shows, actors, directors, reviews, and ratings. IMDb allows users to search based on criteria like title, genre, actor, or year of release.

Rotten Tomatoes:

Known for its movie ratings and reviews, Rotten Tomatoes aggregates critic and audience scores to provide an overall view of a movie's reception. Users can search movies by title or genre, and the site offers curated lists, but it doesn't offer robust personalized recommendations.

TMDb (The Movie Database):

TMDb provides a comprehensive database with detailed information on movies, TV shows, cast, and crew. It also includes user-generated reviews and ratings. Many third-party applications and websites use TMDb's API to offer movie search capabilities.

1.2.2 Streaming Platforms with Search Features

Netflix:

Netflix offers a search engine for users to browse its library, including options to filter by genre, release year, and language. However, Netflix's search capabilities are often criticized for not being comprehensive or offering robust filtering and sorting options. It also features personalized recommendations based on viewing history, but the recommendations are limited to content within the Netflix library.

Amazon Prime Video:

Amazon Prime Video allows users to search for movies and shows based on various filters, including genre, release date, and IMDb rating. It also provides personalized recommendations, though users may find the system less accurate or efficient than they would like, especially when browsing through thousands of titles.

Disney+:

Disney+ has its own search and filtering system, mainly for Disney-related content, including movies, TV shows, and documentaries. It provides recommendations based on the user's viewing habits but lacks extensive features for discovering content outside its own ecosystem.

1.3 Proposed System

The proposed movie search application aims to offer a comprehensive, personalized, and efficient movie discovery experience by combining cross-platform search, personalized recommendations, rich movie metadata, and an intuitive user interface. By addressing the shortcomings of existing systems, this application will enhance the movie-watching journey for users, making it easier to find the perfect film based on their tastes and preferences.

1.4 Literature Review

While traditional movie databases, streaming platforms, and recommendation systems have provided useful tools for movie discovery, there are significant gaps in cross-platform search, personalized recommendations, and advanced filtering options. The proposed movie search application would aim to address these challenges by integrating advanced search algorithms, mood-based recommendations, and comprehensive movie metadata, offering a unified, personalized, and engaging movie discovery experience. Future research and development should continue to explore the integration of deep learning, emotion-aware systems, and AI-powered assistants to enhance the user experience even further.

2.SYSTEM REQUIREMENTS

2.1 Hardware and Software Requirements

The Movie Search Application requires a robust set of hardware for both end-users and backend infrastructure. Software frameworks include mobile and web development tools, databases, APIs, and cloud solutions to ensure scalability, security, and performance. The combination of these resources will allow the app to provide a smooth, responsive user experience with high availability and secure data handling.

2.2 Software Requirements Specifications (SRS)

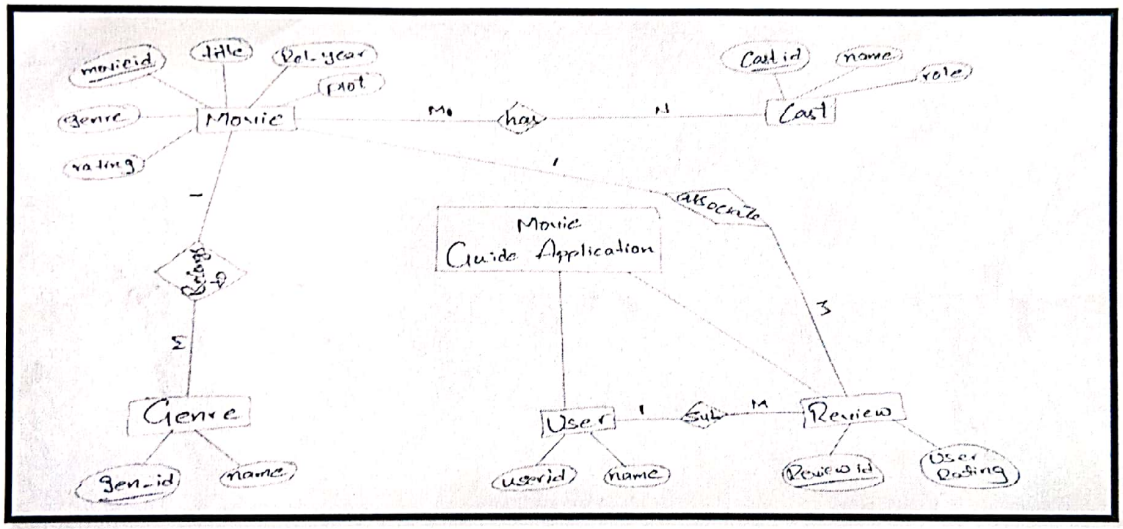
The Software Requirements Specification (SRS) defines the functional and non-functional requirements for the Movie Search Application, ensuring that it meets user needs for movie discovery, recommendations, and detailed information. By adhering to these requirements, the application can deliver a seamless, secure, and scalable movie search experience across multiple platforms

3.SYSTEM DESIGN

3.1 Modules of System

Each of these modules plays a crucial role in the development and functionality of the Movie Search Application. They ensure the app is user-friendly, scalable, secure, and provides an excellent movie discovery experience. By dividing the system into these core modules, the development process becomes more manageable and efficient, leading to a better-quality product.

3.2 ULM Diagram



4.IMPLEMENTATION

4.1 Sample Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movie Search App</title>
    <style>
        /* (CSS remains exactly the same as in the previous version) */
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }
     body {
            background-color: #f5f5f5;
            color: #333;
            line-height: 1.6;
        }
.container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }
 header {
            text-align: center;
            margin-bottom: 30px;
```

```
        padding: 20px 0;
        background-color: #2c3e50;
        color: white;
        border-radius: 8px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }
h1 {
        font-size: 2.5rem;
        margin-bottom: 10px;
    }
.search-container {
        display: flex;
        justify-content: center;
        margin-bottom: 30px;
    }
#search-input {
        width: 60%;
        padding: 12px 20px;
        font-size: 1rem;
        border: 2px solid #ddd;
        border-radius: 4px 0 0 4px;
        outline: none;
    }
#search-input:focus {
        border-color: #3498db;
    }
#search-button {
        padding: 12px 20px;
        background-color: #3498db;
        color: white;
```

```
            border: none;
            border-radius: 0 4px 4px 0;
            cursor: pointer;
            font-size: 1rem;
            transition: background-color 0.3s;
        }
#search-button:hover {
            background-color: #2980b9;
        }
.filters {
            display: flex;
            justify-content: center;
            gap: 15px;
            margin-bottom: 20px;
            flex-wrap: wrap;
        }
.filter {
            padding: 8px 15px;
            border-radius: 4px;
            background-color: white;
            border: 1px solid #ddd;
            cursor: pointer;
            transition: all 0.3s;
        }
.filter:hover {
            background-color: #f0f0f0;
        }
.filter.active {
            background-color: #3498db;
            color: white;
```

```
        border-color: #3498db;
    }
.movie-grid {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
        gap: 20px;
    }
    .movie-card {
        background-color: white;
        border-radius: 8px;
        overflow: hidden;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        transition: transform 0.3s;
    }
    .movie-card:hover {
        transform: translateY(-5px);
    }
    .movie-poster {
        width: 100%;
        height: 350px;
        object-fit: cover;
    }
 .movie-info {
        padding: 15px;
    }
    .movie-title {
        font-size: 1.2rem;
        margin-bottom: 8px;
       color: #2c3e50;
    }
```

```
.movie-year {
    color: #7f8c8d;
    margin-bottom: 8px;
}
.movie-type {
    display: inline-block;
    padding: 3px 8px;
    background-color: #e74c3c;
    color: white;
    border-radius: 4px;
    font-size: 0.8rem;
    margin-bottom: 10px;
}
.movie-details-btn {
    display: block;
    width: 100%;
    padding: 8px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}
.movie-details-btn:hover {
    background-color: #2980b9;
}
.movie-modal {
    display: none;
    position: fixed;
```

```
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.8);
    z-index: 1000;
    overflow-y: auto;
}
.modal-content {
    background-color: white;
    margin: 50px auto;
    max-width: 800px;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
}
.modal-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 20px;
    background-color: #2c3e50;
    color: white;
}
.close-modal {
    background: none;
    border: none;
    color: white;
    font-size: 1.5rem;
    cursor: pointer;
```

```
}
.modal-body {
    display: flex;
    flex-direction: column;
    padding: 20px;
}
@media (min-width: 768px) {
    .modal-body {
        flex-direction: row;
    }
}
.modal-poster {
    width: 100%;
    max-width: 300px;
    height: auto;
    border-radius: 4px;
    margin-bottom: 20px;
}
@media (min-width: 768px) {
    .modal-poster {
        margin-right: 20px;
        margin-bottom: 0;
    }
}
.modal-details {
    flex: 1;
}

.modal-title {
    font-size: 1.8rem;
```

```
    margin-bottom: 10px;
    color: #2c3e50;
}
.modal-meta {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-bottom: 15px;
}
.modal-year, .modal-runtime, .modal-rating {
    padding: 3px 8px;
    background-color: #ecf0f1;
    border-radius: 4px;
    font-size: 0.9rem;
}
.modal-plot {
    margin-bottom: 15px;
    line-height: 1.7;
}
.modal-info {
    margin-bottom: 10px;
}
.info-label {
    font-weight: bold;
    color: #2c3e50;
}
.loading {
    text-align: center;
    padding: 20px;
    font-size: 1.2rem;
```

```
        color: #7f8c8d;
    }
    .error-message {
        text-align: center;
        padding: 20px;
        color: #e74c3c;
        font-size: 1.2rem;
    }
    @media (max-width: 768px) {
        .search-container {
            flex-direction: column;
            align-items: center;
        }
        #search-input {
            width: 100%;
            border-radius: 4px;
            margin-bottom: 10px;
        }
        #search-button {
            width: 100%;
            border-radius: 4px;
        }
        .movie-grid {
            grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
        }
    }
  </style>
</head>
<body>
  <div class="container">
```

```
    <header>
        <h1>Movie Search App</h1>
        <p>Discover your next favorite movie</p>
    </header>
    <div class="search-container">
        <input type="text" id="search-input" placeholder="Search for movies...">
        <button id="search-button">Search</button>
    </div>
    <div class="filters">
        <div class="filter active" data-type="all">All</div>
        <div class="filter" data-type="movie">Movies</div>
        <div class="filter" data-type="series">TV Series</div>
        <div class="filter" data-type="game">Games</div>
    </div>
    <div id="results-container">
        <!-- Movies will be displayed here -->
    </div>
</div>
<div class="movie-modal" id="movie-modal">
    <div class="modal-content">
        <div class="modal-header">
            <h2 id="modal-title">Movie Title</h2>
            <button class="close-modal">&times;</button>
        </div>
        <div class="modal-body">
            <img id="modal-poster" class="modal-poster" src="" alt="Movie Poster">
            <div class="modal-details">
                <div class="modal-meta">
                    <span id="modal-year" class="modal-year">Year</span>
                    <span id="modal-runtime" class="modal-runtime">Runtime</span>
```

```
                <span id="modal-rating" class="modal-rating">Rating</span>
            </div>
            <p id="modal-plot" class="modal-plot">Plot goes here...</p>
            <p class="modal-info"><span class="info-label">Director:</span> <span
id="modal-director"></span></p>
            <p class="modal-info"><span class="info-label">Actors:</span> <span
id="modal-actors"></span></p>
            <p class="modal-info"><span class="info-label">Genre:</span> <span
id="modal-genre"></span></p>
            <p class="modal-info"><span class="info-label">Awards:</span> <span
id="modal-awards"></span></p>
          </div>
        </div>
      </div>
    </div>
  <script>
      document.addEventListener('DOMContentLoaded', function() {
        // DOM Elements
        const searchInput = document.getElementById('search-input');
        const searchButton = document.getElementById('search-button');
        const filters = document.querySelectorAll('.filter');
        const resultsContainer = document.getElementById('results-container');
        const movieModal = document.getElementById('movie-modal');
        const closeModal = document.querySelector('.close-modal');
            // Variables
        let currentSearchTerm = '';
        let currentType = 'all';
        let movies = [];
       // Event Listeners
      searchButton.addEventListener('click', searchMovies);
       searchInput.addEventListener('keypress', function(e) {
```

```
    if (e.key === 'Enter') {
        searchMovies();
    }
});
filters.forEach(filter => {
    filter.addEventListener('click', function() {
        filters.forEach(f => f.classList.remove('active'));
        this.classList.add('active');
        currentType = this.dataset.type;
        filterMovies();
    });
});
closeModal.addEventListener('click', function() {
    movieModal.style.display = 'none';
});
window.addEventListener('click', function(e) {
    if (e.target === movieModal) {
        movieModal.style.display = 'none';
    }
});
// Functions
async function searchMovies() {
    const searchTerm = searchInput.value.trim();
    if (!searchTerm) {
        alert('Please enter a search term');
        return;
    }
currentSearchTerm = searchTerm;
    resultsContainer.innerHTML = '<div class="loading">Loading movies...</div>';
try {
```

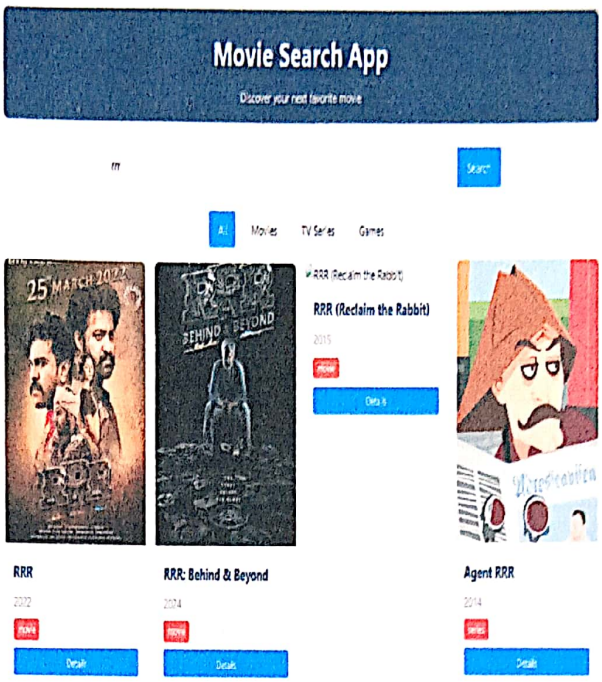
```
            // Try with API key first
            let response = await
fetch(`https://www.omdbapi.com/?s=${encodeURIComponent(searchTerm)}&apikey=d8a43
9c3`);
            let data = await response.json();
        // If API key fails, try with demo key
            if (data.Error && data.Error.includes('API key')) {
              response = await
fetch(`https://www.omdbapi.com/?s=${encodeURIComponent(searchTerm)}&apikey=thewd
b`);
              data = await response.json();
            }if (data.Response === 'True') {
              movies = data.Search;
              filterMovies();
            } else {
              resultsContainer.innerHTML = `<div class="error-message">${data.Error ||
'No movies found'}</div>`;
            }
          } catch (error) {
            resultsContainer.innerHTML = `<div class="error-message">Failed to fetch
movies. ${error.message}</div>`;
            console.error('Error fetching movies:', error);
          }
        }
        function filterMovies() {
          if (currentType === 'all') {
            displayMovies(movies);
          } else {
            const filteredMovies = movies.filter(movie => movie.Type === currentType);
            displayMovies(filteredMovies);
          }
        }
```

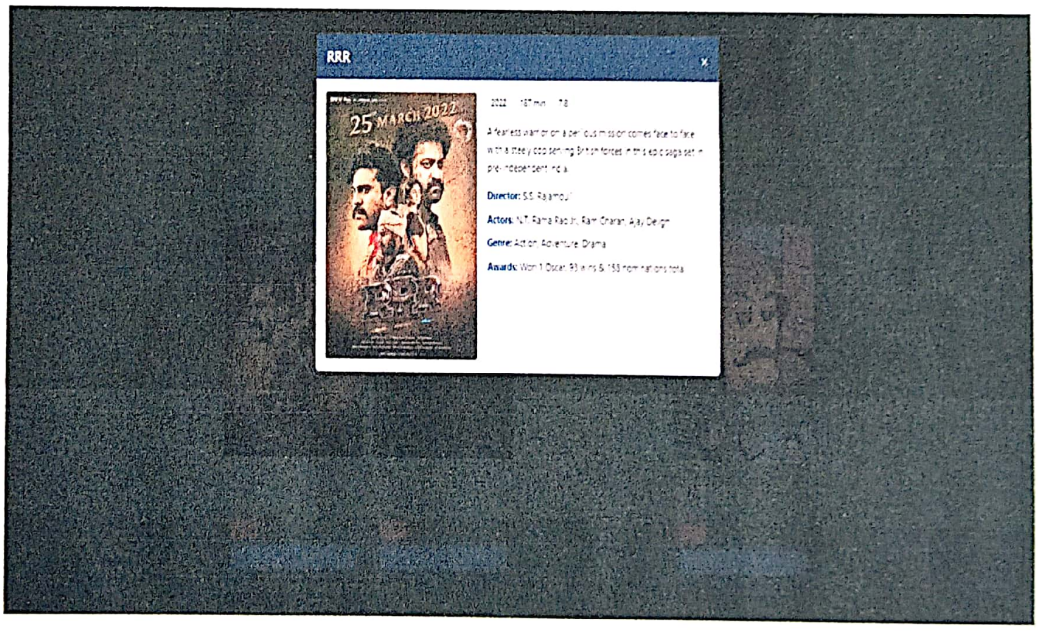
```
function displayMovies(moviesToDisplay) {
    if (!moviesToDisplay || moviesToDisplay.length === 0) {
        resultsContainer.innerHTML = '<div class="error-message">No movies found. Try a different search.</div>';
        return;
    }
    resultsContainer.innerHTML = '<div class="movie-grid"></div>';
    const movieGrid = document.querySelector('.movie-grid');
    moviesToDisplay.forEach(movie => {
        const movieCard = document.createElement('div');
        movieCard.className = 'movie-card';
        movieCard.innerHTML = `
            <img src="${movie.Poster !== 'N/A' ? movie.Poster : 'https://via.placeholder.com/300x450?text=No+Poster'}" alt="${movie.Title}" class="movie-poster">
            <div class="movie-info">
                <h3 class="movie-title">${movie.Title}</h3>
                <p class="movie-year">${movie.Year}</p>
                <span class="movie-type">${movie.Type}</span>
                <button class="movie-details-btn" data-id="${movie.imdbID}">Details</button>
            </div>
        `;
        movieGrid.appendChild(movieCard);
    });
    // Add event listeners to detail buttons
    document.querySelectorAll('.movie-details-btn').forEach(button => {
        button.addEventListener('click', function() {
            const imdbID = this.dataset.id;
            showMovieDetails(imdbID);
        });
```

```
        }); }
        async function showMovieDetails(imdbID) {
            try {
                // Try with API key first
                let response = await
fetch(`https://www.omdbapi.com/?i=${imdbID}&apikey=d8a439c3`);
                let movie = await response.json();
                // If API key fails, try with demo key
                if (movie.Error && movie.Error.includes('API key')) {
                    response = await
fetch(`https://www.omdbapi.com/?i=${imdbID}&apikey=thewdb`);
                    movie = await response.json();
                }
                if (movie.Response === 'True') {
                    // Update modal with movie details
                    document.getElementById('modal-title').textContent = movie.Title;
                    document.getElementById('modal-poster').src = movie.Poster !== 'N/A' ?
movie.Poster : 'https://via.placeholder.com/300x450?text=No+Poster';
                    document.getElementById('modal-year').textContent = movie.Year;
                    document.getElementById('modal-runtime').textContent = movie.Runtime ||
'N/A';
                    document.getElementById('modal-rating').textContent = movie.imdbRating ||
'N/A';
                    document.getElementById('modal-plot').textContent = movie.Plot || 'No plot
available';
                    document.getElementById('modal-director').textContent = movie.Director ||
'N/A';
                    document.getElementById('modal-actors').textContent = movie.Actors ||
'N/A';
                    document.getElementById('modal-genre').textContent = movie.Genre || 'N/A';
                    document.getElementById('modal-awards').textContent = movie.Awards ||
'N/A';
                    // Show modal
```

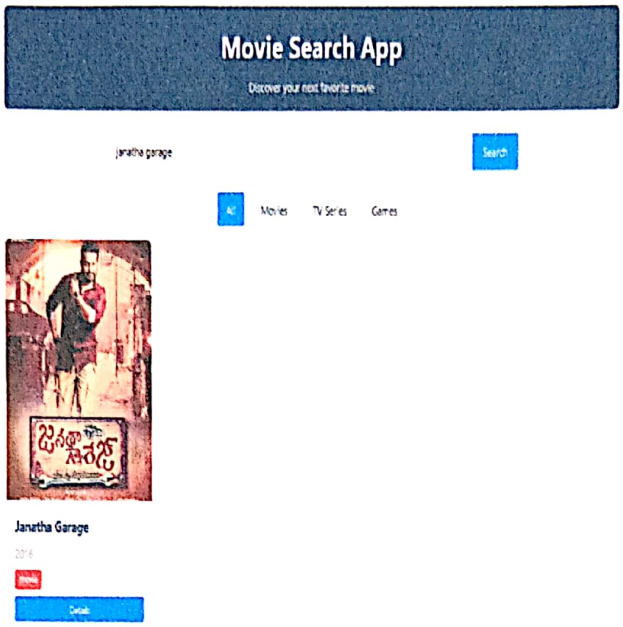
```
                movieModal.style.display = 'block';
            } else {
                alert('Failed to load movie details. Please try again.');
            }
        } catch (error) {
            console.error('Error fetching movie details:', error);
            alert('An error occurred while fetching movie details.');
        }}
    // Initial load with popular movies
    searchInput.value = 'Avengers';
    searchMovies();
});
</script>
</body>
</html>
```

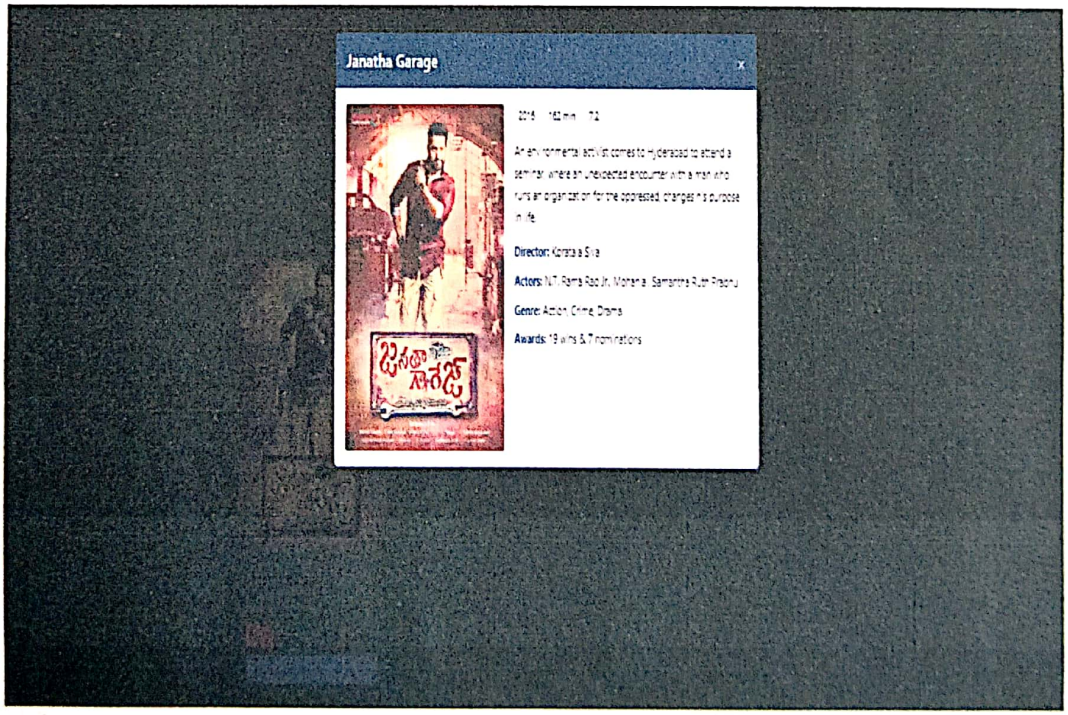
4.2 Test cases:





5.RESULTS





6.CONCLUSION

The Movie Search Application is a simple and effective tool for searching and displaying movie details based on a user's query. This app demonstrates how to integrate an external API (like OMDB) to fetch real-time movie data and dynamically display the results on a web page. It uses basic web development technologies—HTML, CSS, and JavaScript—to create a user-friendly and interactive experience.

REFERENCE

- **HTML:** Used for structuring the web page.
- **CSS:** Used for styling and responsive design.
- **JavaScript:** Used for dynamic interaction, such as searching through the movie list and filtering results based on user input.