SYNOPSIS

Report On

Movie Review Application

Submitted by

Harsh Gupta (2024101161000082)

Session: 2024-2026 (II Semester)

MASTER OF COMPUTER APPLICATION

Under the supervision of

Ms. Shruti
(Assistant Professor)

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR,
GHAZIABAD-201206

ABSTRACT

The **Movie Review Web Application** is a user-centric platform that provides an interactive space for users to read and submit movie reviews and to watch the latest trailers. Built using a modern technology stack, the project integrates **SpringBoot (Java)** as the backend framework, **MongoDB** as the NoSQL database, and **ReactJS** for the frontend, which is enhanced with **Axios** for HTTP communication. This system demonstrates the principles of full-stack development with seamless integration across components. It emphasizes user engagement, usability, and extendibility, making it a valuable platform for movie enthusiasts to share and gain insights.

TABLE OF CONTENTS

INTRODUCTION

Movies are a primary source of entertainment and cultural exchange in today's world. With an overwhelming number of films being released every month, users depend on reviews and trailers to decide what to watch. Traditional movie review websites often have clunky interfaces or are saturated with advertisements. This project was developed to provide a clean, ad-free, and modern alternative that is highly responsive and user-focused. The Movie Review Web Application provides an avenue for viewers to express their opinions and helps others make informed decisions through community-contributed content.

Objectives

- To design and develop a responsive web application for movie reviews.
- To implement user-friendly forms for review submission.
- To fetch and display real-time movie data and reviews.
- To embed YouTube trailers within the application.
- To build an easily extendable platform using modular code architecture.
- To explore and implement integration between frontend, backend, and database systems using modern frameworks.

Problem Statement

Despite the availability of review platforms such as IMDb and Rotten Tomatoes, they suffer from certain limitations:

- Outdated UI/UX experiences
- Lack of community engagement features
- Limited integration of trailers
- Inflexibility in terms of expansion and customization

The objective of this project is to design and build a web application that allows users to review movies and watch trailers in a seamless and user-friendly environment. This requires implementing a responsive UI, real-time database interactions, and support for embedded multimedia content.

LITERATURE REVIEW

IMDb: While IMDb is widely used, it focuses heavily on professional reviews and industry data. User contributions are often limited to ratings and short reviews, and the interface is data-heavy rather than interactive.

Rotten Tomatoes: Primarily aggregates critic and audience scores, but doesn't allow open discussion or detailed community contributions. It lacks integrated video support beyond links.

Letterbox: Emphasizes the social aspect of film-watching, with extensive community involvement. However, it lacks embedded trailers and relies heavily on external media platforms.

This project synthesizes the strengths of these platforms while aiming to overcome their limitations by creating a single, user-driven, video-enabled interface.

SpringBoot: SpringBoot is chosen for backend development due to its robust, scalable, and production-ready nature. It simplifies REST API development and handles backend logic efficiently.

MongoDB: As a NoSQL database, MongoDB offers flexible data modelling. It stores JSON-like documents, making it easy to map data between the backend and frontend.

ReactJS: React component-based structure allows for a highly modular and reusable frontend. Its virtual DOM ensures fast rendering and dynamic updates.

Axios: Axios is a promise-based HTTP client used for communicating between React and the backend REST APIs. It simplifies API requests and response handling.

METHODOLOGY

The project followed an Agile methodology, focusing on iterative development. The overall process was broken into several stages:

1. **Requirement Analysis:** Understanding project goals and defining system requirements.
2. **System Design:** Designing UI mock-ups, backend architecture, and database schema.
3. **Development:** Developing individual modules (backend, frontend, database) and integrating them.
4. **Testing:** Conducting unit, integration, and system testing.
5. **Deployment and Demonstration:** Hosting the application and creating a video demonstration.

Each module was verified independently and in combination to ensure correctness and coherence.

Hardware and Software Requirements

🔧 Hardware Requirements

Component	Minimum Requirement	Recommended Requirement
Processor (CPU)	Dual Core 2.0 GHz	Quad Core 3.0 GHz or higher
RAM	4 GB	8 GB or more
Hard Disk	5 GB free space	10 GB or more
Display	1024x768 resolution	1920x1080 resolution (Full HD)
Internet Connection	Basic connectivity for API access	Stable broadband for smoother performance

🛠️ Software Requirements

Component	Version / Tool Used
Operating System	Windows 10 / Linux / macOS
Backend Framework	Java with SpringBoot (v2.7 or above)
Frontend Framework	ReactJS (v18+)
Database	MongoDB (v6.0 or above)
Development IDE	IntelliJ IDEA (for backend), VS Code (for frontend)
Browser	Chrome / Firefox / Edge (latest versions)
Node Package Manager	npm (v9+)
API Client (for testing)	Postman
Build Tools	Maven / Gradle (for SpringBoot)
Version Control	Git & GitHub

✅ Optional Tools

- **Lombok Plugin** (for simplified Java code with annotations)
- **MongoDB Compass** (for database visualization)
- **React DevTools** (browser extension for debugging React apps.

PROJECT OVERVIEW

Backend

- **Language & Framework:** Java with SpringBoot
- **Development Tool:** IntelliJ IDEA
- **REST API Features:**
    - GET: Fetch all movies or specific movie by ID
    - POST: Submit a review for a specific movie
    - MongoDB Integration for storing movies and reviews
- **Security:** CORS enabled for cross-origin resource sharing, input validation to prevent invalid submissions
- **API Documentation:** Designed using standard RESTful principles

Frontend

- **Language & Framework:** JavaScript with ReactJS
- **Development Tool:** Visual Studio Code
- **Libraries Used:**
    - Axios (API communication)
    - Bootstrap (UI styling)
    - React Router (Navigation)
- **Frontend Features:**
    - Display movie list dynamically
    - Review submission form with live data updates
    - Embedded YouTube trailer using iframe
    - Responsive design using media queries

Database Design

Database: MongoDB

Collections:

1. **Movies**
    - Id
    - imdbId
    - title
    - releaseDate
    - poster
    - description
    - trailerLink (YouTube link)
2. **Reviews**
    - reviewId
    - Id (foreign key)
    - content
    - timestamp

Design Features:

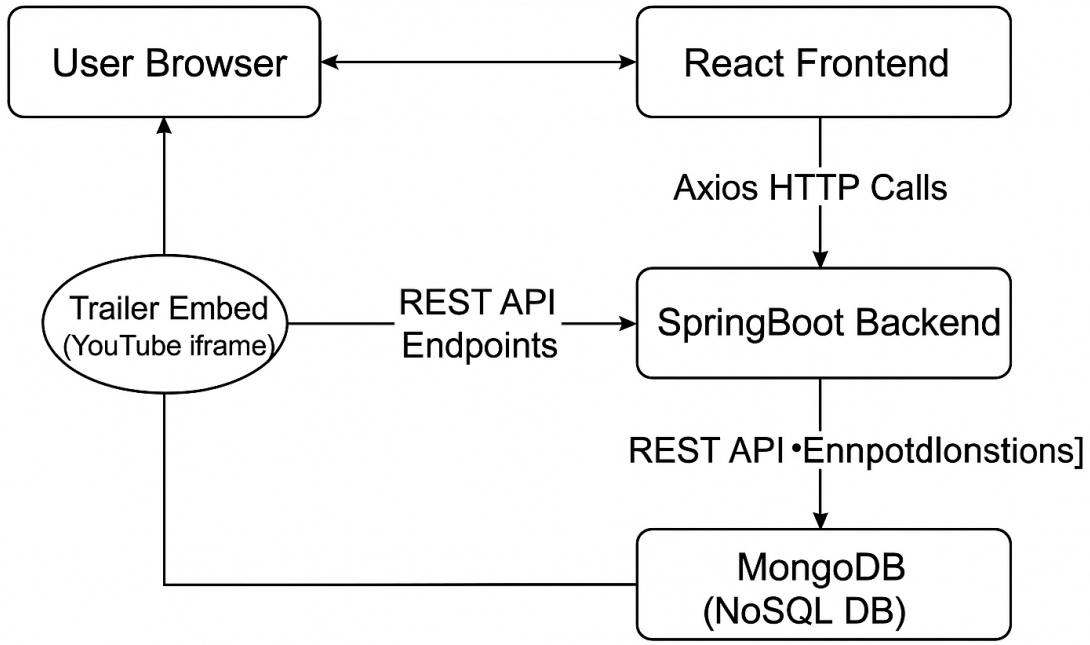
- Non-relational structure allows scalability
- Easy mapping to frontend state
- Supports fast read/write operations

System Architecture Diagram

System Architecture and Data Flow

The following diagram represents the overall flow of data and component interaction in the Movie Review Web Application:

Architecture



PROJECT FLOW

Basic Workflow:

1. User lands on the homepage
2. React fetches the movie list from SpringBoot backend using Axios
3. Backend queries MongoDB and sends movie data as JSON
4. React renders the movies with their details and embedded trailer links
5. When a user selects a movie:
    - Movie details are displayed
    - Reviews related to that movie are fetched and shown
6. User can write and submit a new review
7. Axios sends the review data to the backend
8. Backend processes and stores it in MongoDB
9. UI updates to reflect the new review in real-time
10. User can watch the embedded trailer directly on the page

UI/UX Design

- **Design Goals:**
  - Simplicity
  - Responsiveness
  - Accessibility
- **Key Elements:**
  - Navigation bar for easy access
  - Card layouts for movies
  - Modal or form interface for submitting reviews
  - Iframe player embedded for trailers
- **Color Scheme:** Dark mode with vibrant accents for buttons and highlights
- **Typography:** Clear, readable fonts for all text and review content

Testing and Debugging

Unit Testing:

- Backend APIs tested using Postman and JUnit
- Review creation and movie fetching endpoints validated

Frontend Testing:

- Tested form inputs, button actions, and data rendering

Bug Fixing:

- Resolved issues related to CORS and asynchronous API responses
- Fixed trailer iframe sizing on mobile devices

User Testing:

- Conducted with peers to assess usability
- Feedback incorporated to improve UI layout and navigation

Project Outcome

The application meets the following objectives:

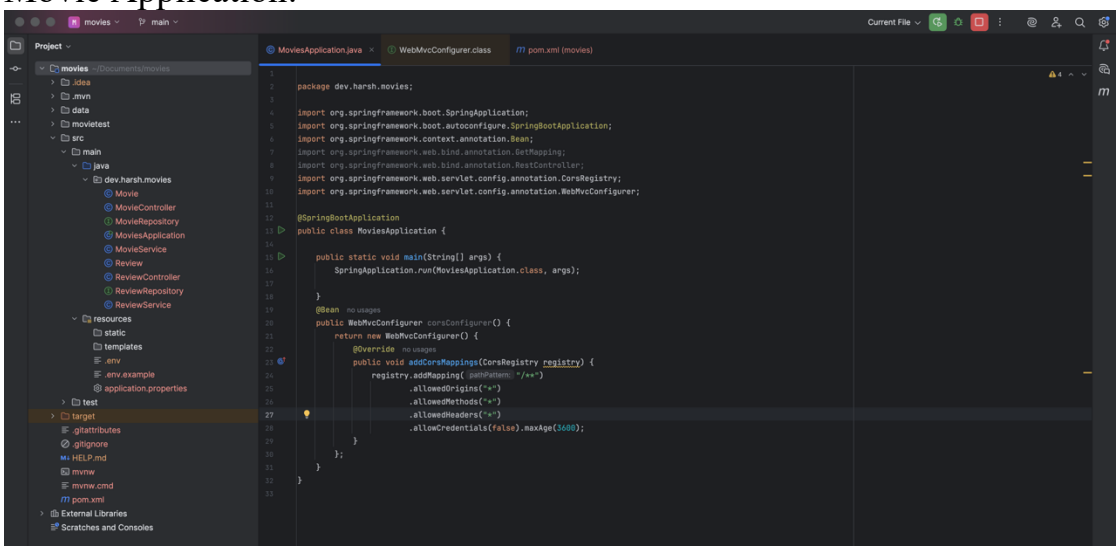
- Allows review submission
- Displays reviews dynamically
- Embeds YouTube trailers
- Runs with minimal latency

Achievements:

- Modular and scalable codebase
- Cross-browser compatibility
- Future-ready for advanced features like authentication, star ratings, etc.

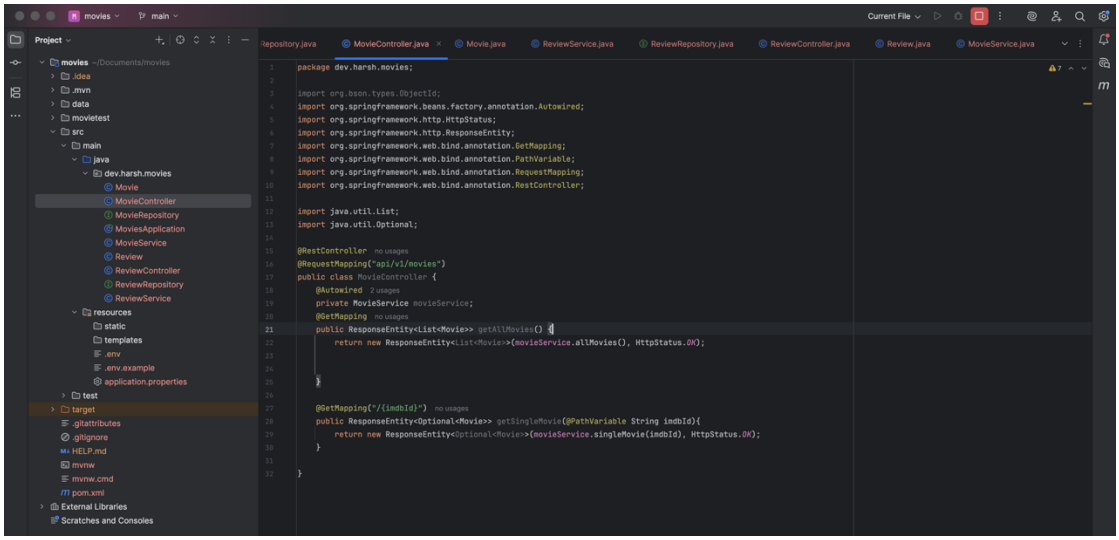
Code Snaps:

Movie Application:



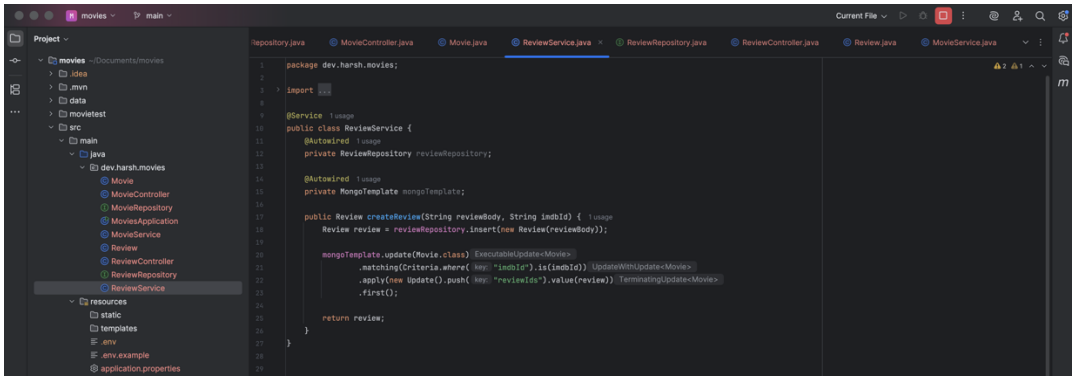
```
package dev.harsh.movies;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@SpringBootApplication
public class MoviesApplication {

    public static void main(String[] args) {
        SpringApplication.run(MoviesApplication.class, args);

    }
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowedOrigins("*")
                        .allowedMethods("*")
                        .allowedHeaders("*")
                        .allowCredentials(false).maxAge(3600);
            }
        };
    }
}
```

Movie Controller:



```
package dev.harsh.movies;

import org.bson.types.ObjectId;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("api/v1/movies")
public class MovieController {
    @Autowired
    private MovieService movieService;
    @GetMapping
    public ResponseEntity<List<Movie>> getAllMovies() {
        return new ResponseEntity<List<Movie>>(movieService.allMovies(), HttpStatus.OK);


    }

    @GetMapping("/{imdbId}")
    public ResponseEntity<Optional<Movie>> getSingleMovie(@PathVariable String imdbId){
        return new ResponseEntity<Optional<Movie>>(movieService.singleMovie(imdbId), HttpStatus.OK);
    }

}
```

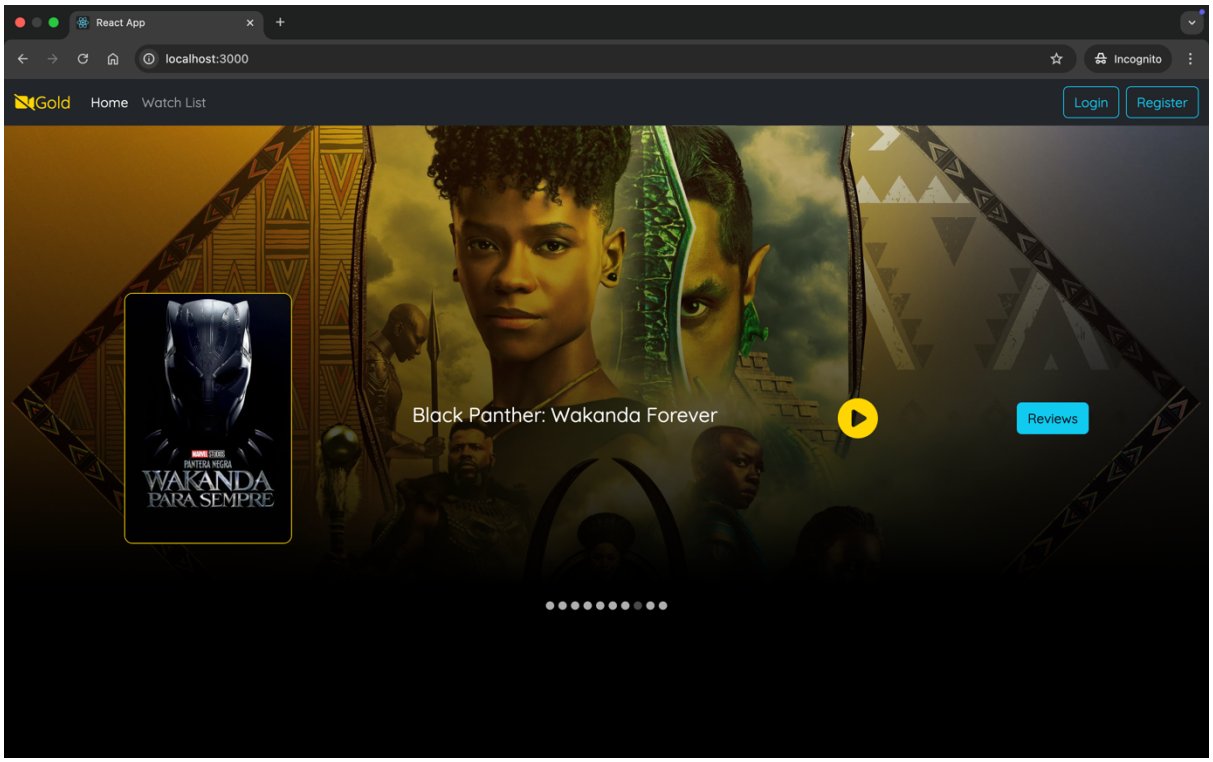
Review Service:

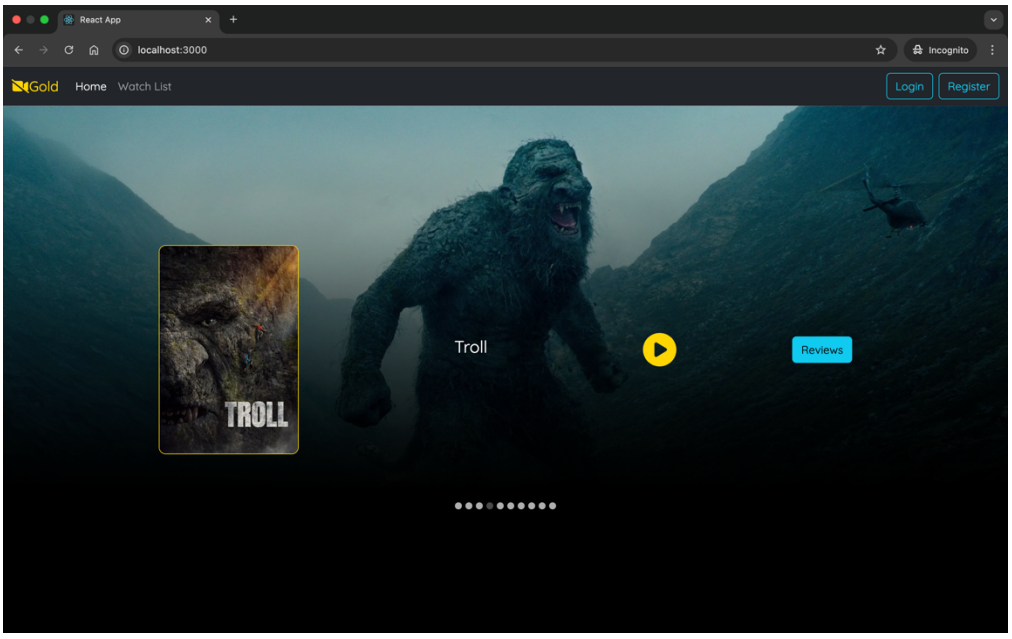


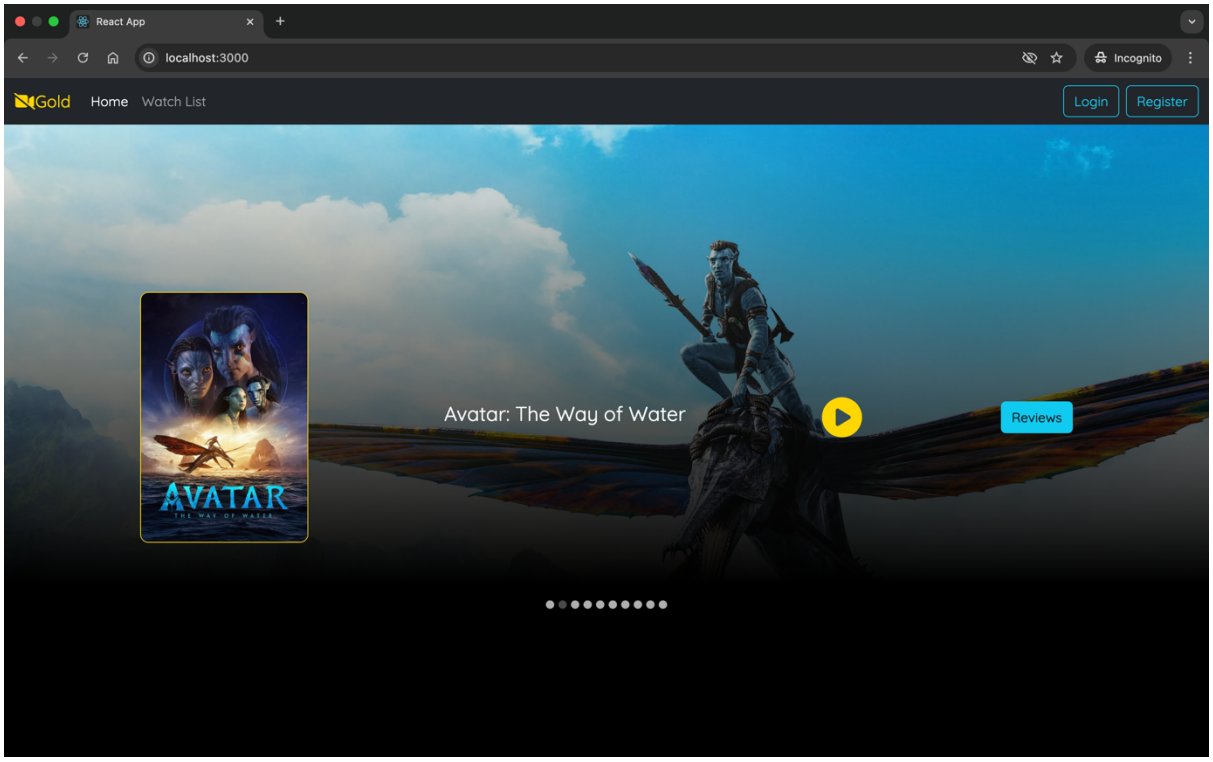
```
package dev.harsh.movies;

import ...

@Service
public class ReviewService {
    @Autowired
    private ReviewRepository reviewRepository;

    @Autowired
    private MongoTemplate mongoTemplate;

    public Review createReview(String reviewBody, String imdbId) {
        Review review = reviewRepository.insert(new Review(reviewBody));

        mongoTemplate.update(Movie.class)
                .matching(Criteria.where("imdbId").is(imdbId))
                .apply(new Update().push("reviewIds").value(review))
                .first();

        return review;
    }
}
```

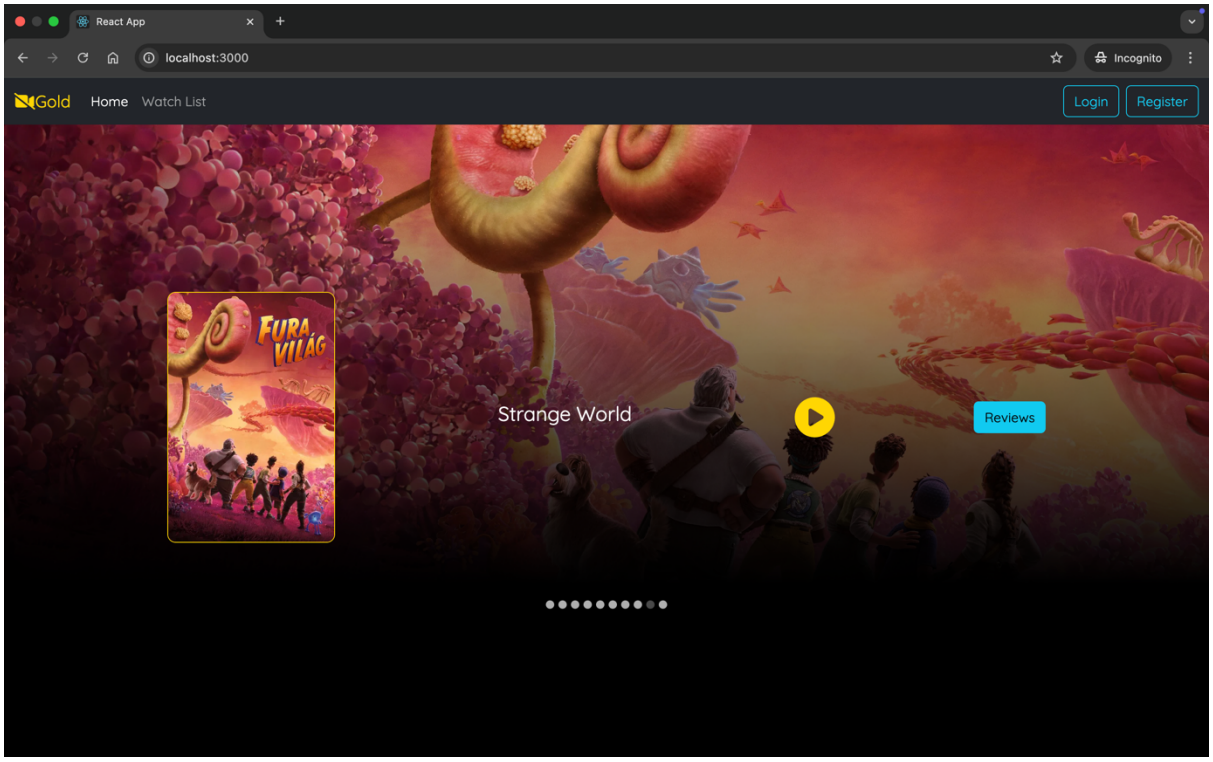
Screenshots of the Application

1. **Homepage:** Displays list of movies

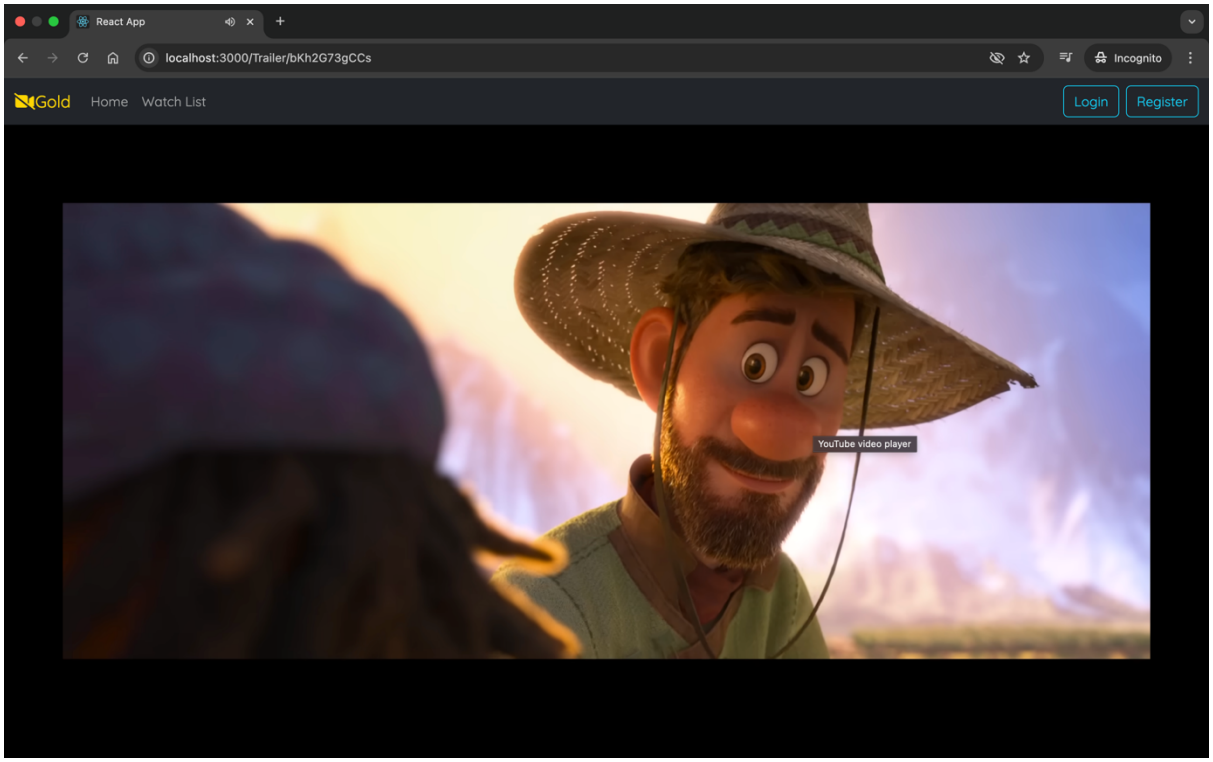


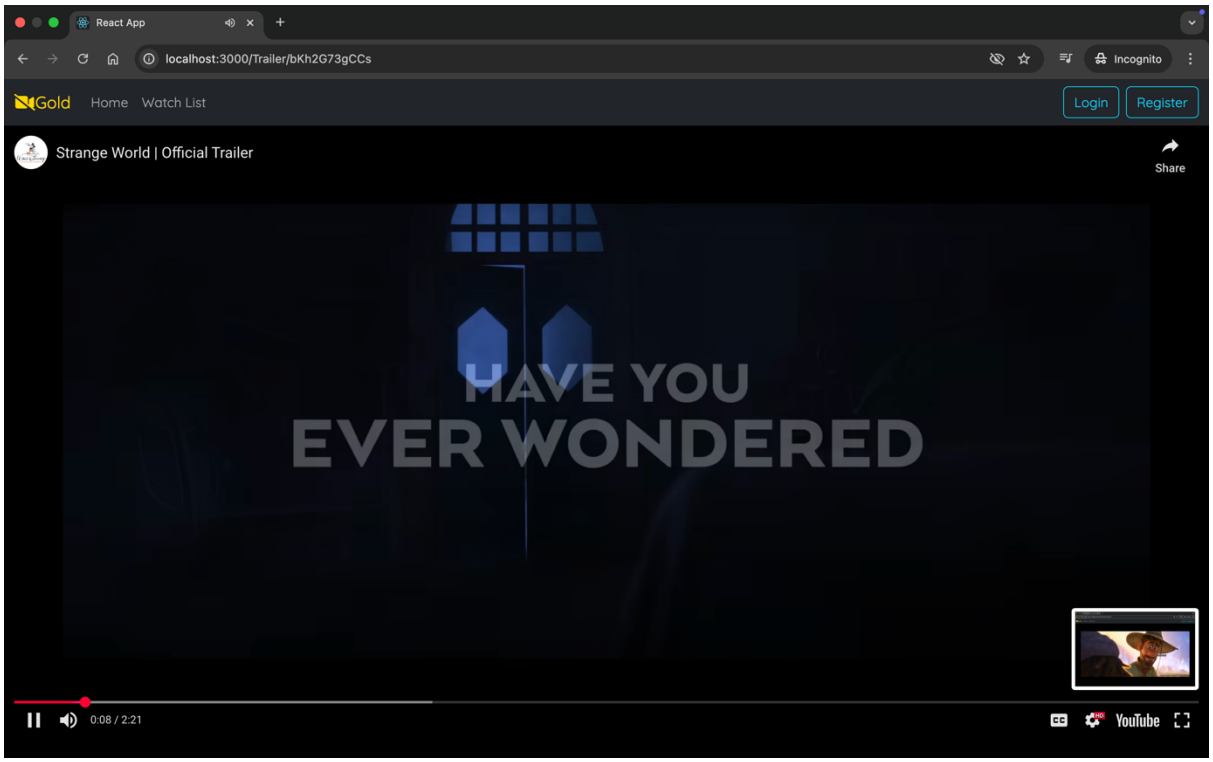



React App
localhost:3000
Incognito
Gold
Home
Watch List
Login
Register
Avatar: The Way of Water
Reviews

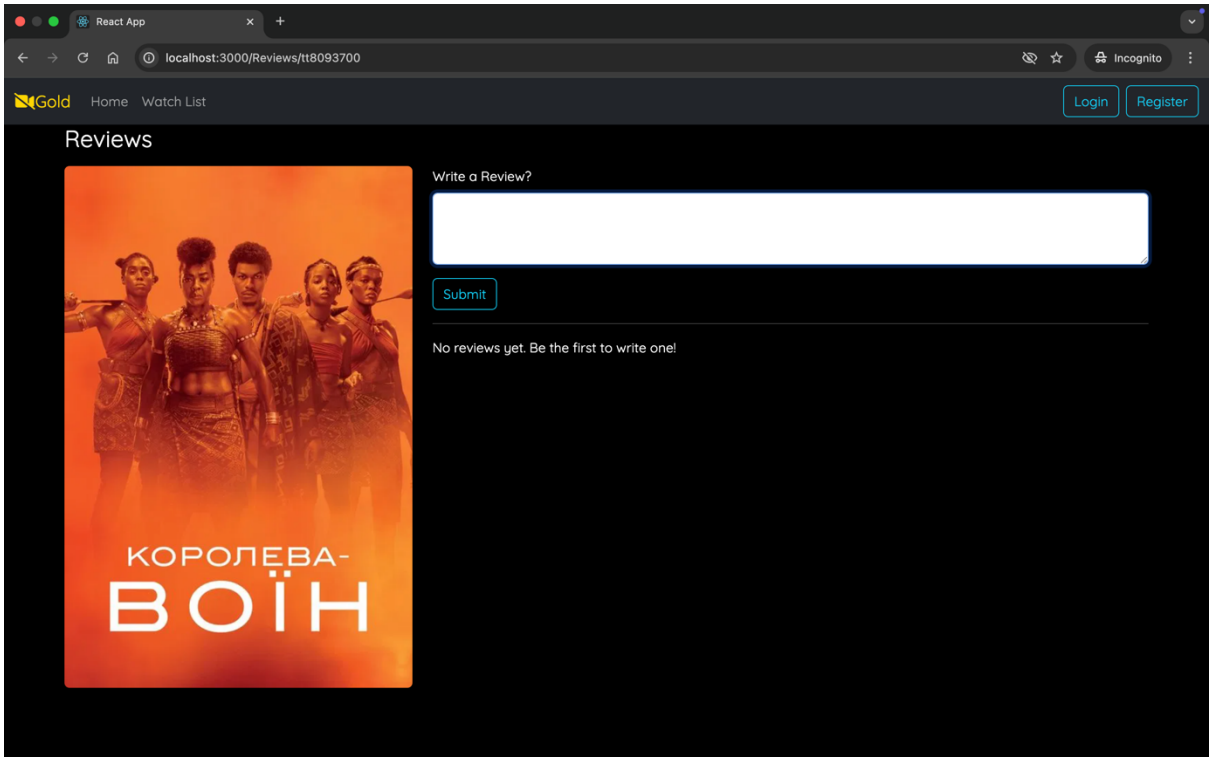

React App
localhost:3000
Incognito
Gold
Home
Watch List
Login
Register
Strange World
Reviews

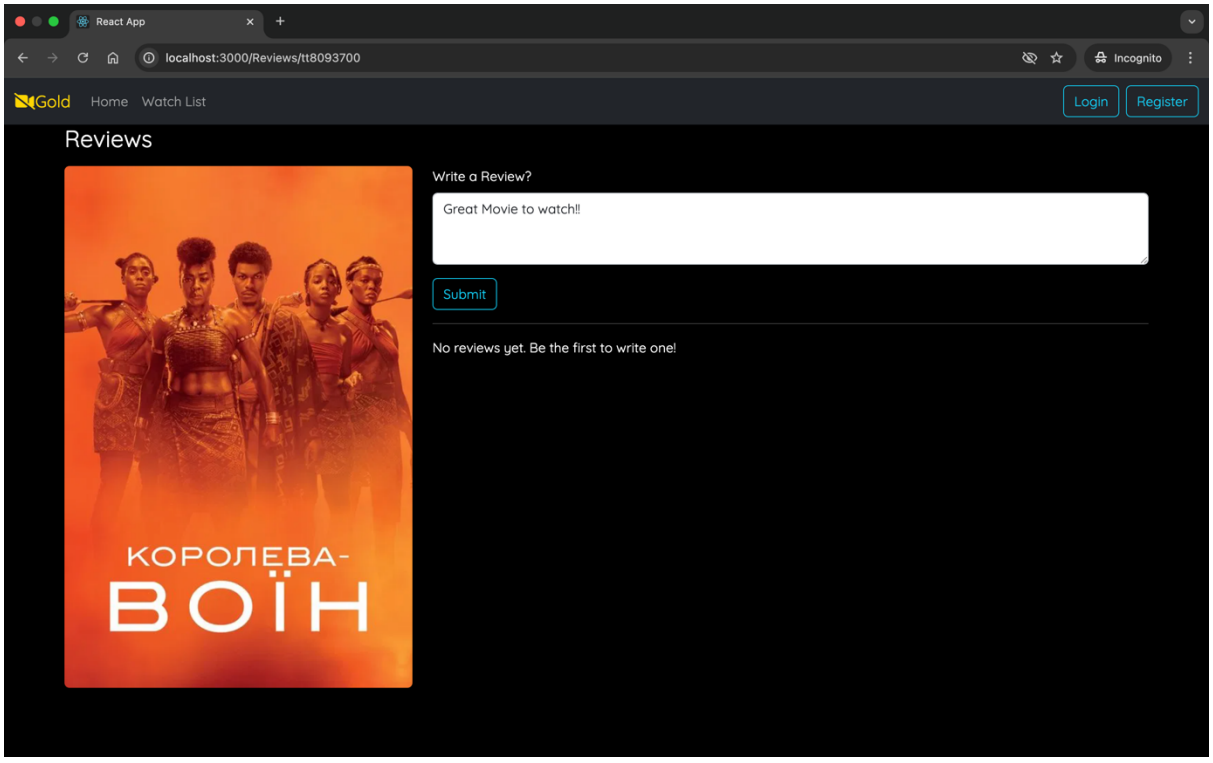
2. **Movie Detail Page:** Shows reviews and trailer





3. **Review Form:** Allows users to post reviews


React App
localhost:3000/Reviews/tt8093700
Incognito
Gold
Home
Watch List
Login
Register
Reviews
Write a Review?
Submit
No reviews yet. Be the first to write one!
КОРОЛЕВА-
ВОЇН


React App
localhost:3000/Reviews/tt8093700
Incognito
Gold
Home
Watch List
Login
Register
Reviews
Write a Review?
Great Movie to watch!!
Submit
No reviews yet. Be the first to write one!
КОРОЛЕВА-
ВОЇН

Conclusion and Future Work

This project successfully demonstrates the power and flexibility of full-stack web development using modern tools. The Movie Review Web Application is not only functional but also provides a user-friendly and visually engaging interface.

Future Improvements:

- Implement user login and authentication
- Add rating system (1–5 stars)
- Enable filtering and sorting by genre or rating
- Admin dashboard for content moderation
- Integration with third-party APIs for fetching movie data

This foundation serves as a scalable product for a live environment or further academic research.

REFERENCE

1. **Spring Framework Documentation**
   https://spring.io/projects/spring-boot
   Official documentation for SpringBoot used to implement backend services and REST APIs.
2. **MongoDB Manual**
   https://www.mongodb.com/docs/manual/
   Comprehensive guide and usage details for MongoDB NoSQL database system.
3. **ReactJS Official Documentation**
   https://reactjs.org/docs/getting-started.html
   Guide for component-based frontend development used in building the user interface.
4. **Axios GitHub Repository**
   https://github.com/axios/axios
   Reference for integrating Axios HTTP client to handle frontend-backend communication.
5. **YouTube Embed API**
   https://developers.google.com/youtube/player_parameters
   Used to embed movie trailers directly into the web application using iframe.