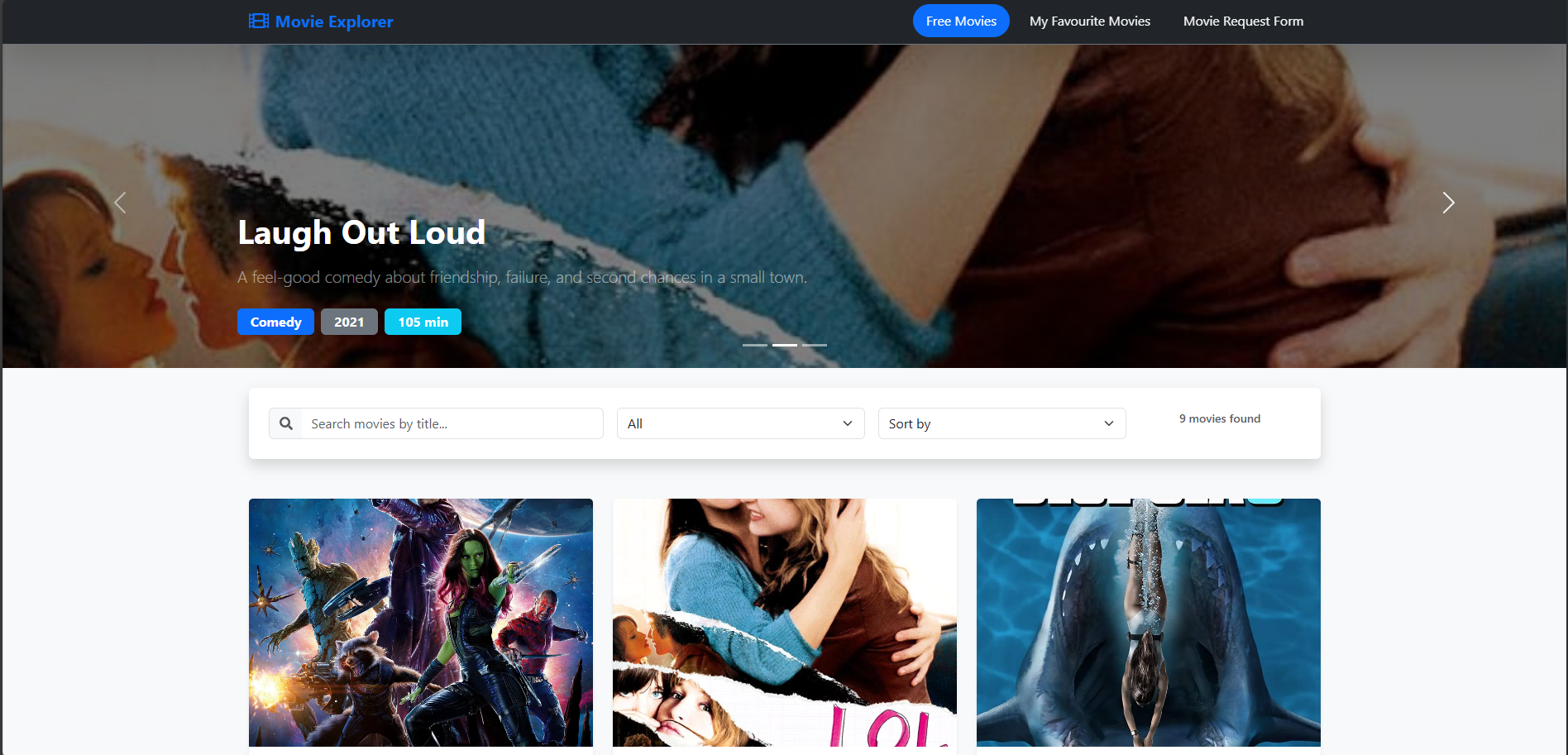
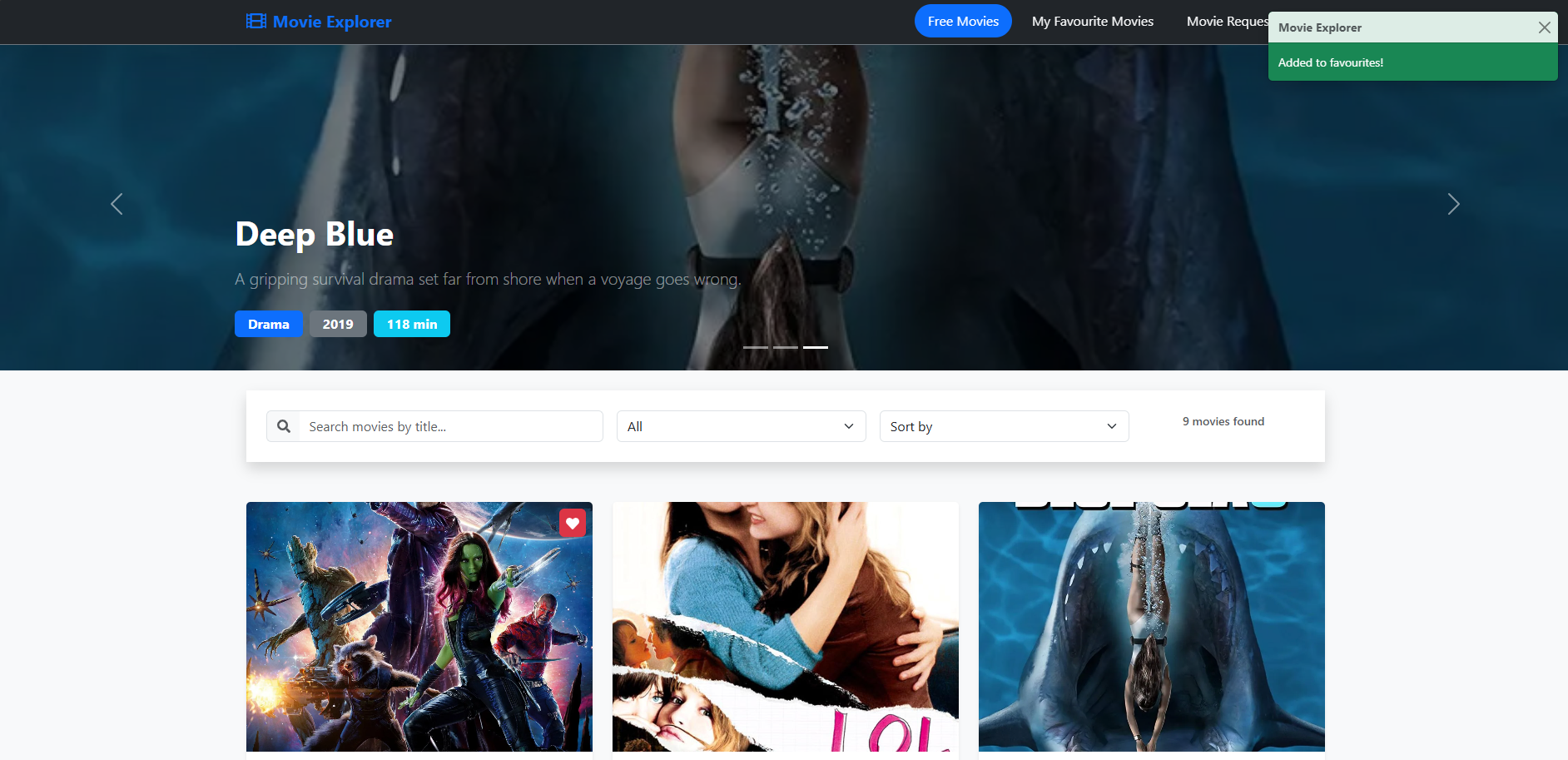
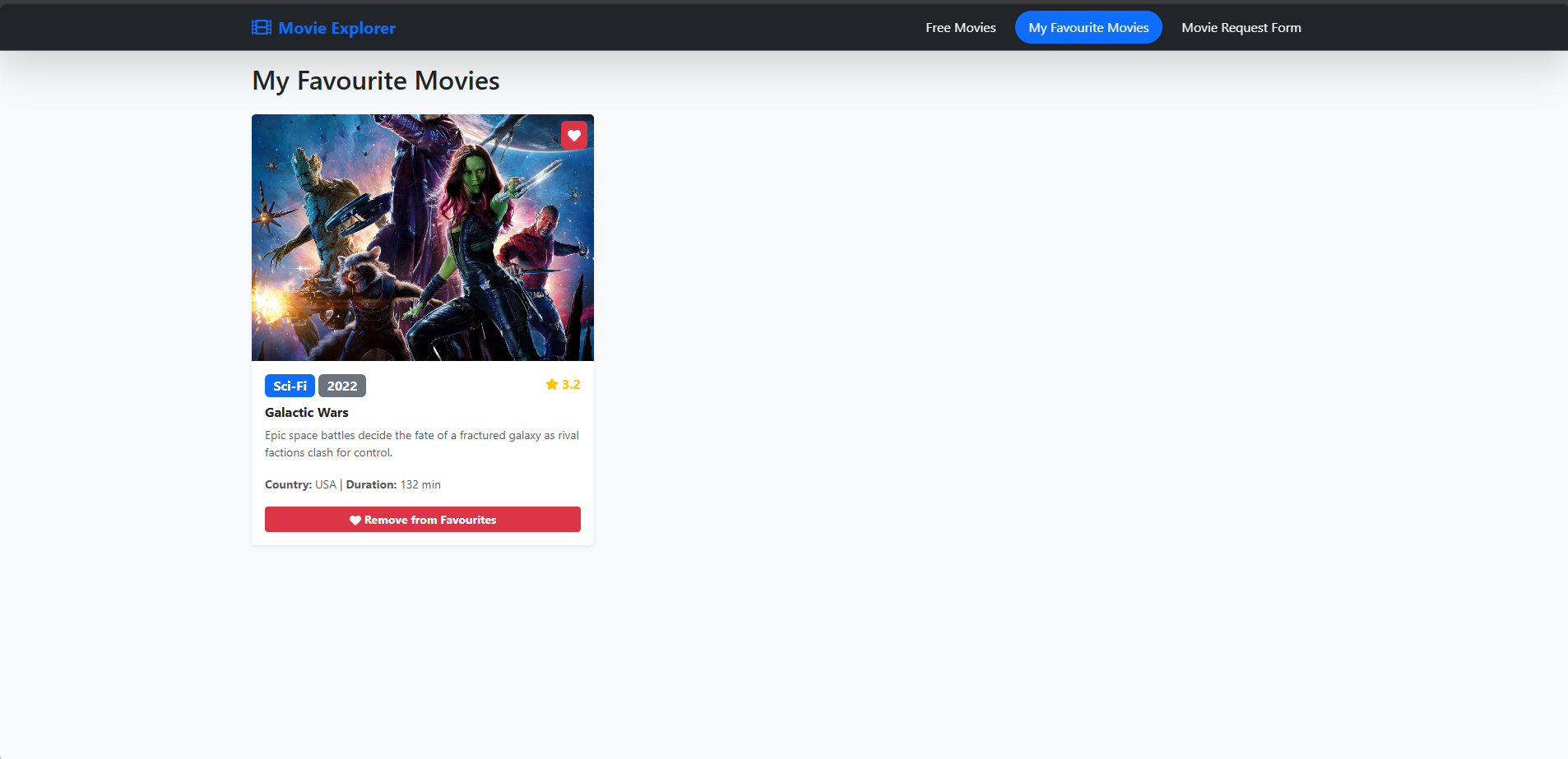
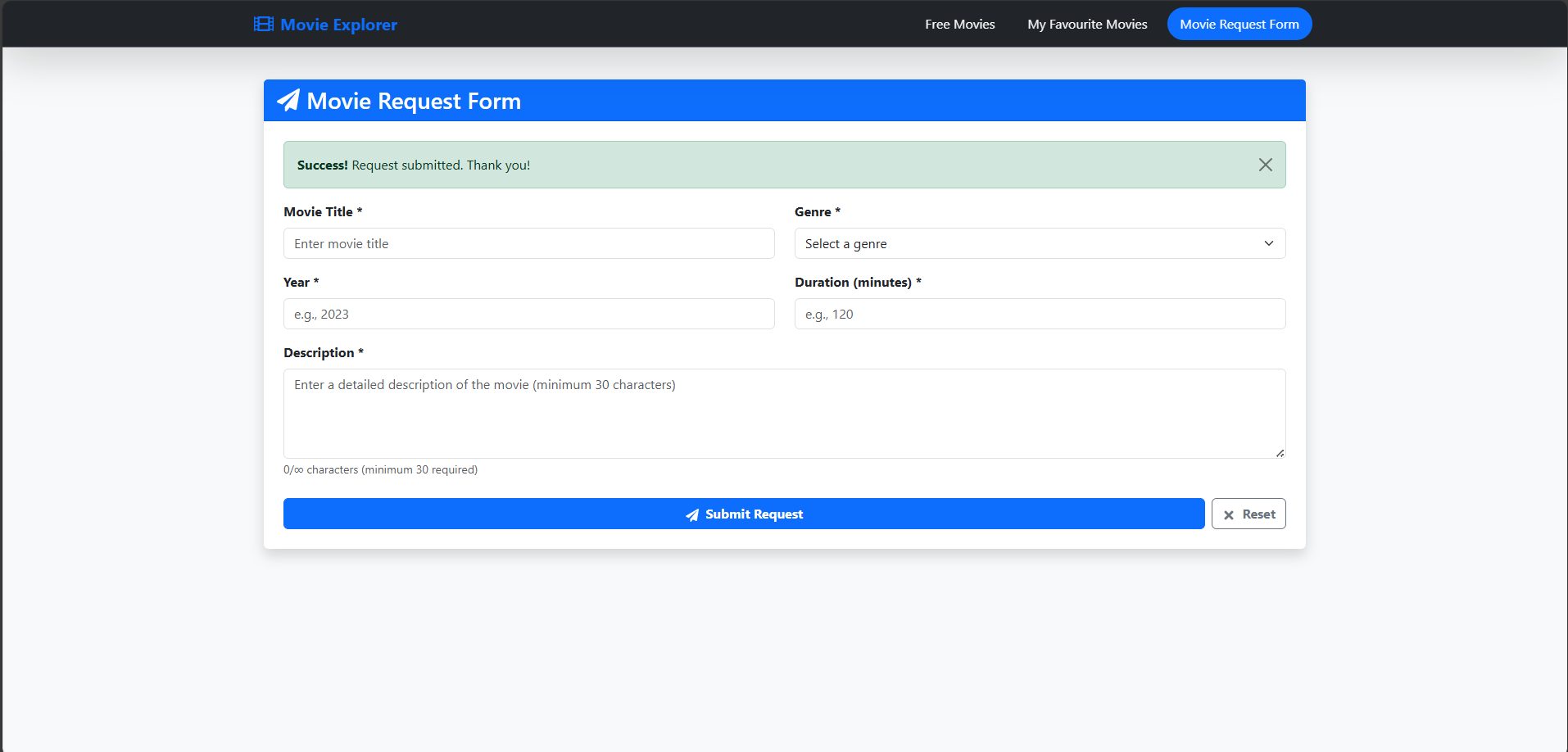
**LAB 2**

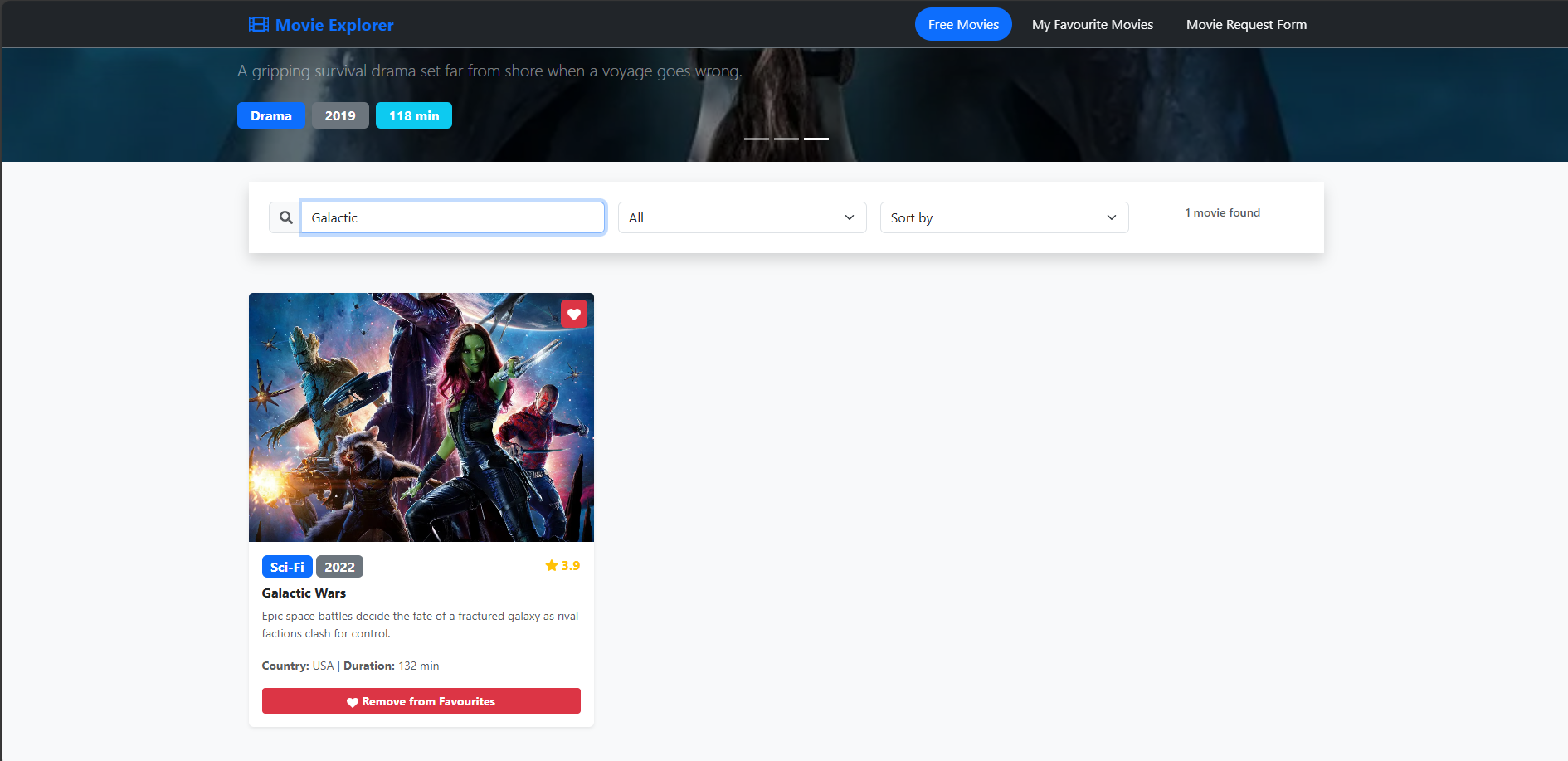
* Kết quả:

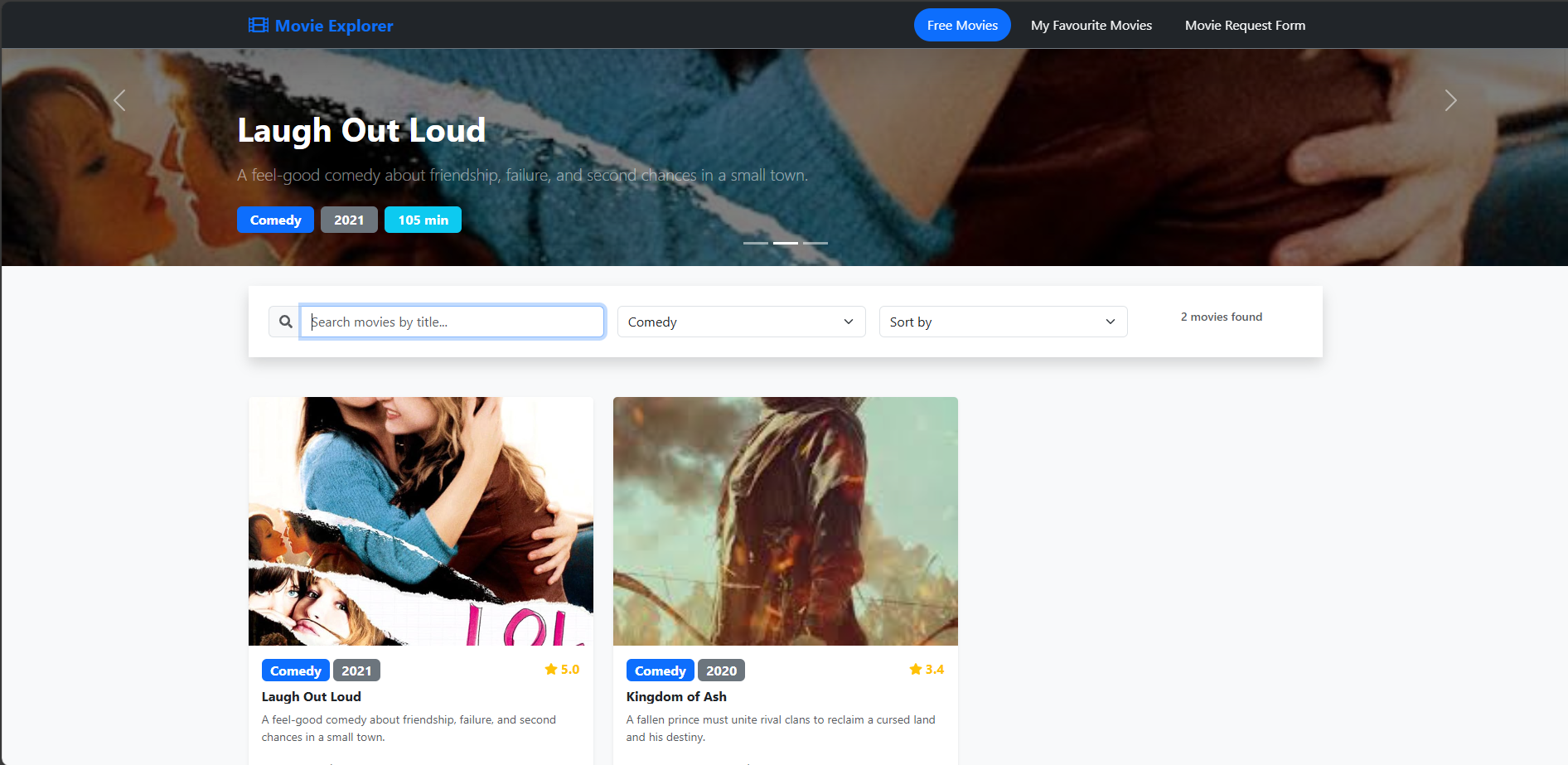


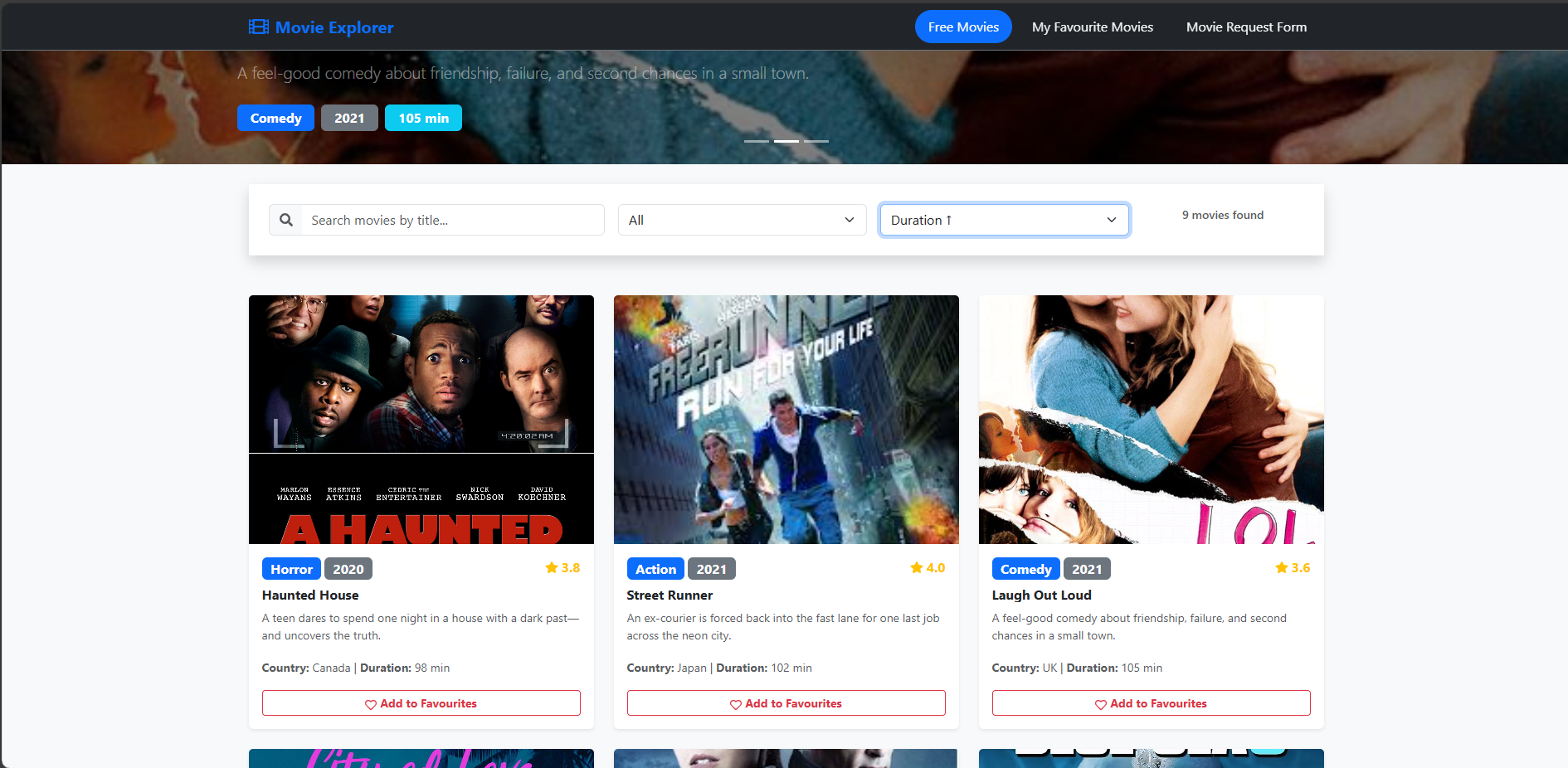












* Code chương trình:
* import React from 'react';
* import { Navbar, Nav, Container } from 'react-bootstrap';
* import PropTypes from 'prop-types';
* import { FaFilm } from 'react-icons/fa';
* const NavigationBar = ({ *activePage*, *onPageChange* }) => {
* return (
* <Navbar
* *bg*="dark"
* *variant*="dark"
* *expand*="lg"
* *fixed*="top"
* *className*="shadow-lg border-bottom border-secondary"
* >
* <Container>
* <Navbar.Brand
* *href*="#home"
* *className*="fw-bold text-primary d-flex align-items-center"
* *onClick*={() => onPageChange('home')}
* *style*={{ cursor: 'pointer' }}
* >
* <FaFilm *className*="me-2 fs-4" />
* Movie Explorer
* </Navbar.Brand>
* <Navbar.Toggle *aria-controls*="basic-navbar-nav" />
* <Navbar.Collapse *id*="basic-navbar-nav">
* <Nav *className*="ms-auto">
* <Nav.Link
* *className*={`px-3 py-2 mx-1 rounded-pill transition ${
* *activePage* === 'home'
* ? 'bg-primary text-white'
* : 'text-light hover-bg-secondary'
* }`}
* *onClick*={() => onPageChange('home')}
* *style*={{ cursor: 'pointer' }}
* >
* Free Movies
* </Nav.Link>
* <Nav.Link
* *className*={`px-3 py-2 mx-1 rounded-pill transition ${
* activePage === 'favourites'
* ? 'bg-primary text-white'
* : 'text-light hover-bg-secondary'
* }`}
* *onClick*={() => onPageChange('favourites')}
* *style*={{ cursor: 'pointer' }}
* >
* My Favourite Movies
* </Nav.Link>
* <Nav.Link
* *className*={`px-3 py-2 mx-1 rounded-pill transition ${
* activePage === 'request'
* ? 'bg-primary text-white'
* : 'text-light hover-bg-secondary'
* }`}
* *onClick*={() => onPageChange('request')}
* *style*={{ cursor: 'pointer' }}
* >
* Movie Request Form
* </Nav.Link>
* </Nav>
* </Navbar.Collapse>
* </Container>
* </Navbar>
* );
* };
* NavigationBar.propTypes = {
* activePage: PropTypes.string.isRequired,
* onPageChange: PropTypes.func.isRequired,
* };
* export default NavigationBar;

import React from 'react';

import { Carousel } from 'react-bootstrap';

import PropTypes from 'prop-types';

const HeroCarousel = ({ *movies* }) => {

*// Select first 3 movies for carousel*

  const carouselMovies = *movies*.slice(0, 3);

  return (

    <Carousel

*className*="mb-4"

*interval*={5000}

*controls*={true}

*indicators*={true}

*fade*

    >

      {carouselMovies.map((*movie*, *index*) => (

        <Carousel.Item *key*={*movie*.id}>

          <div

*className*="d-block w-100"

*style*={{

              height: '400px',

              backgroundImage: `linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url(${*movie*.poster})`,

              backgroundSize: 'cover',

              backgroundPosition: 'center',

              backgroundRepeat: 'no-repeat'

            }}

          />

          <Carousel.Caption *className*="text-start">

            <h2 *className*="display-6 fw-bold text-white mb-3">

              {*movie*.title}

            </h2>

            <p *className*="lead text-white-50 mb-4">

              {*movie*.description}

            </p>

            <div *className*="d-flex gap-2">

              <span *className*="badge bg-primary fs-6 px-3 py-2">

                {*movie*.genre}

              </span>

              <span *className*="badge bg-secondary fs-6 px-3 py-2">

                {*movie*.year}

              </span>

              <span *className*="badge bg-info fs-6 px-3 py-2">

                {*movie*.duration} min

              </span>

            </div>

          </Carousel.Caption>

        </Carousel.Item>

      ))}

    </Carousel>

  );

};

HeroCarousel.propTypes = {

  movies: PropTypes.arrayOf(

    PropTypes.shape({

      id: PropTypes.number.isRequired,

      title: PropTypes.string.isRequired,

      description: PropTypes.string.isRequired,

      poster: PropTypes.string.isRequired,

      genre: PropTypes.string.isRequired,

      year: PropTypes.number.isRequired,

      duration: PropTypes.number.isRequired,

    })

  ).isRequired,

};

export default HeroCarousel;

import React from 'react';

import { Card, Button, Badge } from 'react-bootstrap';

import PropTypes from 'prop-types';

import { FaHeart, FaRegHeart, FaInfoCircle, FaStar } from 'react-icons/fa';

const MovieCard = ({

  movie,

  isFavourite,

  onAddToFavourites,

  onShowDetails

}) => {

  const truncateDescription = (*text*, *maxLength* = 100) => {

    if (text.length <= maxLength) return text;

    return text.substring(0, maxLength) + '...';

  };

  return (

    <Card *className*="h-100 shadow-sm border-0 bg-white text-dark movie-card">

      <div *className*="position-relative">

        <Card.Img

*variant*="top"

*src*={movie.poster}

*alt*={`Poster for ${movie.title}`}

*className*="card-img-top"

*style*={{ height: '300px', objectFit: 'cover' }}

*onError*={(*e*) => {

            e.target.src = 'https://via.placeholder.com/300x450?text=No+Image';

          }}

        />

        {*/\* Favourite badge \*/*}

        {isFavourite && (

          <div *className*="position-absolute top-0 end-0 m-2">

            <Badge *bg*="danger" *className*="fs-6 p-2">

              <FaHeart />

            </Badge>

          </div>

        )}

        {*/\* Overlay with Details button \*/*}

        <div *className*="position-absolute top-0 start-0 w-100 h-100 d-flex align-items-center justify-content-center bg-dark bg-opacity-75 opacity-0 hover-opacity-100 transition">

          <Button

*variant*="outline-light"

*size*="sm"

*onClick*={() => onShowDetails(movie)}

*className*="fw-bold"

          >

            <FaInfoCircle *className*="me-1" />

            Details

          </Button>

        </div>

      </div>

      <Card.Body *className*="d-flex flex-column p-3">

        <div *className*="mb-2 d-flex justify-content-between align-items-start">

          <div *className*="d-flex gap-1 flex-wrap">

            <Badge *bg*="primary" *className*="fs-6">{movie.genre}</Badge>

            <Badge *bg*="secondary" *className*="fs-6">{movie.year}</Badge>

          </div>

          <div *className*="d-flex align-items-center">

            <FaStar *className*="text-warning me-1" />

            <span *className*="text-warning fw-bold">{(Math.random() \* 2 + 3).toFixed(1)}</span>

          </div>

        </div>

        <Card.Title *className*="h6 mb-2 fw-bold text-truncate">{movie.title}</Card.Title>

        <Card.Text *className*="flex-grow-1 text-muted small">

          {truncateDescription(movie.description)}

        </Card.Text>

        <div *className*="mb-3">

          <small *className*="text-muted">

            <strong>Country:</strong> {movie.country} |

            <strong> Duration:</strong> {movie.duration} min

          </small>

        </div>

        <Button

*variant*={isFavourite ? "danger" : "outline-danger"}

*size*="sm"

*onClick*={() => onAddToFavourites(movie)}

*className*="w-100 fw-bold"

        >

          {isFavourite ? (

            <>

              <FaHeart *className*="me-1" />

              Remove from Favourites

            </>

          ) : (

            <>

              <FaRegHeart *className*="me-1" />

              Add to Favourites

            </>

          )}

        </Button>

      </Card.Body>

    </Card>

  );

};

MovieCard.propTypes = {

  movie: PropTypes.shape({

    id: PropTypes.number.isRequired,

    title: PropTypes.string.isRequired,

    description: PropTypes.string.isRequired,

    poster: PropTypes.string.isRequired,

    genre: PropTypes.string.isRequired,

    year: PropTypes.number.isRequired,

    country: PropTypes.string.isRequired,

    duration: PropTypes.number.isRequired,

  }).isRequired,

  isFavourite: PropTypes.bool.isRequired,

  onAddToFavourites: PropTypes.func.isRequired,

  onShowDetails: PropTypes.func.isRequired,

};

export default MovieCard;

import React from 'react';

import { Modal, Button, Badge, Row, Col } from 'react-bootstrap';

import PropTypes from 'prop-types';

import { FaClock, FaGlobe, FaCalendar, FaFilm } from 'react-icons/fa';

const MovieModal = ({ *movie*, *show*, *onHide* }) => {

  if (!*movie*) return null;

  return (

    <Modal *show*={*show*} *onHide*={*onHide*} *size*="lg" *centered*>

      <Modal.Header *closeButton* *className*="bg-primary text-white">

        <Modal.Title *className*="fw-bold">{*movie*.title}</Modal.Title>

      </Modal.Header>

      <Modal.Body *className*="bg-white">

        <Row>

          <Col *md*={4}>

            <img

*src*={*movie*.poster}

*alt*={`Poster for ${*movie*.title}`}

*className*="img-fluid rounded shadow"

*style*={{ width: '100%', height: 'auto' }}

*onError*={(*e*) => {

*e*.target.src = 'https://via.placeholder.com/300x450?text=No+Image';

              }}

            />

          </Col>

          <Col *md*={8}>

            <div *className*="mb-3">

              <Badge *bg*="primary" *className*="me-2 fs-6">{*movie*.genre}</Badge>

              <Badge *bg*="secondary" *className*="fs-6">{*movie*.year}</Badge>

            </div>

            <h6 *className*="mb-2 fw-bold text-dark">Description</h6>

            <p *className*="text-muted mb-3">{*movie*.description}</p>

            <Row *className*="mb-3">

              <Col *sm*={6}>

                <div *className*="d-flex align-items-center mb-2">

                  <FaClock *className*="me-2 text-muted" />

                  <span *className*="text-dark"><strong>Duration:</strong> {movie.duration} minutes</span>

                </div>

              </Col>

              <Col *sm*={6}>

                <div *className*="d-flex align-items-center mb-2">

                  <FaGlobe *className*="me-2 text-muted" />

                  <span *className*="text-dark"><strong>Country:</strong> {movie.country}</span>

                </div>

              </Col>

            </Row>

            <Row>

              <Col *sm*={6}>

                <div *className*="d-flex align-items-center mb-2">

                  <FaCalendar *className*="me-2 text-muted" />

                  <span *className*="text-dark"><strong>Year:</strong> {movie.year}</span>

                </div>

              </Col>

              <Col *sm*={6}>

                <div *className*="d-flex align-items-center mb-2">

                  <FaFilm *className*="me-2 text-muted" />

                  <span *className*="text-dark"><strong>Genre:</strong> {movie.genre}</span>

                </div>

              </Col>

            </Row>

            <div *className*="mt-4">

              <h6 *className*="fw-bold text-dark">Showtimes</h6>

              <div *className*="d-flex flex-wrap gap-2">

                <Badge *bg*="success" *className*="fs-6 px-3 py-2">10:00 AM</Badge>

                <Badge *bg*="success" *className*="fs-6 px-3 py-2">2:30 PM</Badge>

                <Badge *bg*="success" *className*="fs-6 px-3 py-2">7:00 PM</Badge>

                <Badge *bg*="success" *className*="fs-6 px-3 py-2">9:30 PM</Badge>

              </div>

            </div>

          </Col>

        </Row>

      </Modal.Body>

      <Modal.Footer *className*="bg-light">

        <Button *variant*="secondary" *onClick*={onHide}>

          Close

        </Button>

      </Modal.Footer>

    </Modal>

  );

};

MovieModal.propTypes = {

  movie: PropTypes.shape({

    id: PropTypes.number.isRequired,

    title: PropTypes.string.isRequired,

    description: PropTypes.string.isRequired,

    poster: PropTypes.string.isRequired,

    genre: PropTypes.string.isRequired,

    year: PropTypes.number.isRequired,

    country: PropTypes.string.isRequired,

    duration: PropTypes.number.isRequired,

  }),

  show: PropTypes.bool.isRequired,

  onHide: PropTypes.func.isRequired,

};

export default MovieModal;

import React, { useState } from 'react';

import { Form, Button, Alert, Container, Card, Row, Col } from 'react-bootstrap';

import PropTypes from 'prop-types';

import { FaPaperPlane, FaTimes } from 'react-icons/fa';

const MovieRequestForm = ({ *allGenres*, *onSubmit* }) => {

  const [formData, setFormData] = useState({

    title: '',

    genre: '',

    year: '',

    duration: '',

    description: ''

  });

  const [errors, setErrors] = useState({});

  const [showSuccess, setShowSuccess] = useState(false);

  const handleChange = (*e*) => {

    const { name, value } = e.target;

    setFormData(*prev* => ({

      ...prev,

      [name]: value

    }));

*// Clear error when user starts typing*

    if (errors[name]) {

      setErrors(*prev* => ({

        ...prev,

        [name]: ''

      }));

    }

  };

  const validateForm = () => {

    const newErrors = {};

    const currentYear = new Date().getFullYear();

*// Title validation*

    if (!formData.title.trim()) {

      newErrors.title = 'Title is required';

    } else if (formData.title.trim().length < 2) {

      newErrors.title = 'Title must be at least 2 characters long';

    }

*// Genre validation*

    if (!formData.genre) {

      newErrors.genre = 'Please select a genre';

    }

*// Year validation*

    if (!formData.year) {

      newErrors.year = 'Year is required';

    } else {

      const year = parseInt(formData.year);

      if (isNaN(year) || year < 1900 || year > currentYear + 5) {

        newErrors.year = `Year must be between 1900 and ${currentYear + 5}`;

      }

    }

*// Duration validation*

    if (!formData.duration) {

      newErrors.duration = 'Duration is required';

    } else {

      const duration = parseInt(formData.duration);

      if (isNaN(duration) || duration <= 0 || duration > 300) {

        newErrors.duration = 'Duration must be between 1 and 300 minutes';

      }

    }

*// Description validation*

    if (!formData.description.trim()) {

      newErrors.description = 'Description is required';

    } else if (formData.description.trim().length < 30) {

      newErrors.description = 'Description must be at least 30 characters long';

    }

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  const handleSubmit = (*e*) => {

    e.preventDefault();

    if (validateForm()) {

      onSubmit(formData);

      setShowSuccess(true);

*// Reset form*

      setFormData({

        title: '',

        genre: '',

        year: '',

        duration: '',

        description: ''

      });

*// Hide success message after 3 seconds*

      setTimeout(() => {

        setShowSuccess(false);

      }, 3000);

    }

  };

  const **handleReset** = () => {

    setFormData({

      title: '',

      genre: '',

      year: '',

      duration: '',

      description: ''

    });

    setErrors({});

    setShowSuccess(false);

  };

  return (

    <Container *className*="py-4">

      <Card *className*="shadow border-0">

        <Card.Header *className*="bg-primary text-white">

          <h3 *className*="mb-0 d-flex align-items-center">

            <FaPaperPlane *className*="me-2" />

            Movie Request Form

          </h3>

        </Card.Header>

        <Card.Body *className*="p-4">

          {showSuccess && (

            <Alert *variant*="success" *dismissible* *onClose*={() => setShowSuccess(false)}>

              <strong>Success!</strong> Request submitted. Thank you!

            </Alert>

          )}

          <Form *onSubmit*={handleSubmit}>

            <Row>

              <Col *md*={6}>

                <Form.Group *className*="mb-3">

                  <Form.Label *className*="fw-bold">Movie Title \*</Form.Label>

                  <Form.Control

*type*="text"

*name*="title"

*value*={formData.title}

*onChange*={handleChange}

*placeholder*="Enter movie title"

*isInvalid*={!!errors.title}

*className*="form-control"

                  />

                  <Form.Control.Feedback *type*="invalid">

                    {errors.title}

                  </Form.Control.Feedback>

                </Form.Group>

              </Col>

              <Col *md*={6}>

                <Form.Group *className*="mb-3">

                  <Form.Label *className*="fw-bold">Genre \*</Form.Label>

                  <Form.Select

*name*="genre"

*value*={formData.genre}

*onChange*={handleChange}

*isInvalid*={!!errors.genre}

*className*="form-select"

                  >

                    <option *value*="">Select a genre</option>

                    {allGenres.filter(*genre* => genre !== 'All').map(*genre* => (

                      <option *key*={genre} *value*={genre}>{genre}</option>

                    ))}

                  </Form.Select>

                  <Form.Control.Feedback *type*="invalid">

                    {errors.genre}

                  </Form.Control.Feedback>

                </Form.Group>

              </Col>

            </Row>

            <Row>

              <Col *md*={6}>

                <Form.Group *className*="mb-3">

                  <Form.Label *className*="fw-bold">Year \*</Form.Label>

                  <Form.Control

*type*="number"

*name*="year"

*value*={formData.year}

*onChange*={handleChange}

*placeholder*="e.g., 2023"

*min*="1900"

*max*={new Date().getFullYear() + 5}

*isInvalid*={!!errors.year}

*className*="form-control"

                  />

                  <Form.Control.Feedback *type*="invalid">

                    {errors.year}

                  </Form.Control.Feedback>

                </Form.Group>

              </Col>

              <Col *md*={6}>

                <Form.Group *className*="mb-3">

                  <Form.Label *className*="fw-bold">Duration (minutes) \*</Form.Label>

                  <Form.Control

*type*="number"

*name*="duration"

*value*={formData.duration}

*onChange*={handleChange}

*placeholder*="e.g., 120"

*min*="1"

*max*="300"

*isInvalid*={!!errors.duration}

*className*="form-control"

                  />

                  <Form.Control.Feedback *type*="invalid">

                    {errors.duration}

                  </Form.Control.Feedback>

                </Form.Group>

              </Col>

            </Row>

            <Form.Group *className*="mb-4">

              <Form.Label *className*="fw-bold">Description \*</Form.Label>

              <Form.Control

*as*="textarea"

*rows*={4}

*name*="description"

*value*={formData.description}

*onChange*={handleChange}

*placeholder*="Enter a detailed description of the movie (minimum 30 characters)"

*isInvalid*={!!errors.description}

*className*="form-control"

              />

              <Form.Control.Feedback *type*="invalid">

                {errors.description}

              </Form.Control.Feedback>

              <Form.Text *className*="text-muted">

                {formData.description.length}/∞ characters (minimum 30 required)

              </Form.Text>

            </Form.Group>

            <div *className*="d-flex gap-2">

              <Button *variant*="primary" *type*="submit" *className*="flex-grow-1 fw-bold">

                <FaPaperPlane *className*="me-2" />

                Submit Request

              </Button>

              <Button *variant*="outline-secondary" *onClick*={handleReset} *className*="fw-bold">

                <FaTimes *className*="me-2" />

                Reset

              </Button>

            </div>

          </Form>

        </Card.Body>

      </Card>

    </Container>

  );

};

MovieRequestForm.propTypes = {

  allGenres: PropTypes.arrayOf(PropTypes.string).isRequired,

  onSubmit: PropTypes.func.isRequired,

};

export default MovieRequestForm;

import React from 'react';

import { Navbar, Nav, Container } from 'react-bootstrap';

import PropTypes from 'prop-types';

import { FaFilm } from 'react-icons/fa';

const NavigationBar = ({ *activePage*, *onPageChange* }) => {

  return (

    <Navbar

*bg*="dark"

*variant*="dark"

*expand*="lg"

*fixed*="top"

*className*="shadow-lg border-bottom border-secondary"

    >

      <Container>

        <Navbar.Brand

*href*="#home"

*className*="fw-bold text-primary d-flex align-items-center"

*onClick*={() => onPageChange('home')}

*style*={{ cursor: 'pointer' }}

        >

          <FaFilm *className*="me-2 fs-4" />

          Movie Explorer

        </Navbar.Brand>

        <Navbar.Toggle *aria-controls*="basic-navbar-nav" />

        <Navbar.Collapse *id*="basic-navbar-nav">

          <Nav *className*="ms-auto">

            <Nav.Link

*className*={`px-3 py-2 mx-1 rounded-pill transition ${

*activePage* === 'home'

                  ? 'bg-primary text-white'

                  : 'text-light hover-bg-secondary'

              }`}

*onClick*={() => onPageChange('home')}

*style*={{ cursor: 'pointer' }}

            >

              Free Movies

            </Nav.Link>

            <Nav.Link

*className*={`px-3 py-2 mx-1 rounded-pill transition ${

                activePage === 'favourites'

                  ? 'bg-primary text-white'

                  : 'text-light hover-bg-secondary'

              }`}

*onClick*={() => onPageChange('favourites')}

*style*={{ cursor: 'pointer' }}

            >

              My Favourite Movies

            </Nav.Link>

            <Nav.Link

*className*={`px-3 py-2 mx-1 rounded-pill transition ${

                activePage === 'request'

                  ? 'bg-primary text-white'

                  : 'text-light hover-bg-secondary'

              }`}

*onClick*={() => onPageChange('request')}

*style*={{ cursor: 'pointer' }}

            >

              Movie Request Form

            </Nav.Link>

          </Nav>

        </Navbar.Collapse>

      </Container>

    </Navbar>

  );

};

NavigationBar.propTypes = {

  activePage: PropTypes.string.isRequired,

  onPageChange: PropTypes.func.isRequired,

};

export default NavigationBar;

import React from 'react';

import { Form, InputGroup, Row, Col, Card } from 'react-bootstrap';

import PropTypes from 'prop-types';

import { FaSearch } from 'react-icons/fa';

const SearchFilterBar = ({

  searchTerm,

  selectedGenre,

  sortBy,

  onSearchChange,

  onGenreChange,

  onSortChange,

  allGenres,

  resultCount

}) => {

  return (

    <Card *className*="mb-4 border-0 shadow">

      <Card.Body *className*="p-4 bg-white">

        <Row *className*="g-3">

          {*/\* Search Input \*/*}

          <Col *md*={4}>

            <InputGroup>

              <InputGroup.Text *className*="bg-light border-end-0">

                <FaSearch *className*="text-muted" />

              </InputGroup.Text>

              <Form.Control

*type*="text"

*placeholder*="Search movies by title..."

*value*={searchTerm}

*onChange*={(*e*) => onSearchChange(e.target.value)}

*className*="border-start-0"

              />

            </InputGroup>

          </Col>

          {*/\* Genre Filter \*/*}

          <Col *md*={3}>

            <Form.Select

*value*={selectedGenre}

*onChange*={(*e*) => onGenreChange(e.target.value)}

*className*="form-select"

            >

              {allGenres.map(*genre* => (

                <option *key*={genre} *value*={genre}>

                  {genre}

                </option>

              ))}

            </Form.Select>

          </Col>

          {*/\* Sort Options \*/*}

          <Col *md*={3}>

            <Form.Select

*value*={*sortBy*}

*onChange*={(*e*) => onSortChange(*e*.target.value)}

*className*="form-select"

            >

              <option *value*="none">Sort by</option>

              <option *value*="duration-asc">Duration ↑</option>

              <option *value*="duration-desc">Duration ↓</option>

              <option *value*="title-asc">Title A-Z</option>

              <option *value*="title-desc">Title Z-A</option>

              <option *value*="year-desc">Year (Newest)</option>

              <option *value*="year-asc">Year (Oldest)</option>

            </Form.Select>

          </Col>

          {*/\* Result Count \*/*}

          <Col *md*={2}>

            <div *className*="text-center">

              <small *className*="text-muted fw-semibold">

                {*resultCount*} movie{*resultCount* !== 1 ? 's' : ''} found

              </small>

            </div>

          </Col>

        </Row>

      </Card.Body>

    </Card>

  );

};

SearchFilterBar.propTypes = {

  searchTerm: PropTypes.string.isRequired,

  selectedGenre: PropTypes.string.isRequired,

  sortBy: PropTypes.string.isRequired,

  onSearchChange: PropTypes.func.isRequired,

  onGenreChange: PropTypes.func.isRequired,

  onSortChange: PropTypes.func.isRequired,

  allGenres: PropTypes.arrayOf(PropTypes.string).isRequired,

  resultCount: PropTypes.number.isRequired,

};

export default SearchFilterBar;

import React, { useState, useEffect, useMemo } from 'react';

import { Container, Row, Col, Alert, Toast, ToastContainer } from 'react-bootstrap';

import NavigationBar from './components/Navbar';

import HeroCarousel from './components/HeroCarousel';

import SearchFilterBar from './components/SearchFilterBar';

import MovieCard from './components/MovieCard';

import MovieModal from './components/MovieModal';

import MovieRequestForm from './components/MovieRequestForm';

import { movies, allGenres } from './data/movies';

function **App**() {

*// State management*

  const [activePage, **setActivePage**] = useState('home');

  const [searchTerm, **setSearchTerm**] = useState('');

  const [selectedGenre, **setSelectedGenre**] = useState('All');

  const [sortBy, **setSortBy**] = useState('none');

  const [favourites, **setFavourites**] = useState([]); *// Stores movie IDs*

  const [selectedMovie, **setSelectedMovie**] = useState(null);

  const [showModal, **setShowModal**] = useState(false);

  const [showToast, **setShowToast**] = useState(false);

  const [toastMessage, **setToastMessage**] = useState('');

*// Load favourites from localStorage on component mount*

  useEffect(() => {

    const savedFavourites = localStorage.getItem('movieFavourites');

    if (savedFavourites) {

      setFavourites(JSON.parse(savedFavourites));

    }

  }, []);

*// Save favourites to localStorage whenever favourites change*

  useEffect(() => {

    localStorage.setItem('movieFavourites', JSON.stringify(favourites));

  }, [favourites]);

*// Filter and sort movies using useMemo for performance*

  const filteredAndSortedMovies = useMemo(() => {

    let filtered = movies;

*// Filter by search term (case-insensitive)*

    if (searchTerm.trim()) {

      filtered = filtered.filter(*movie* =>

*movie*.title.toLowerCase().includes(searchTerm.toLowerCase()) ||

*movie*.description.toLowerCase().includes(searchTerm.toLowerCase())

      );

    }

*// Filter by genre*

    if (selectedGenre !== 'All') {

      filtered = filtered.filter(*movie* => *movie*.genre === selectedGenre);

    }

*// Sort movies*

    if (sortBy !== 'none') {

      filtered = [...filtered].sort((*a*, *b*) => {

        switch (sortBy) {

          case 'duration-asc':

            return a.duration - b.duration;

          case 'duration-desc':

            return b.duration - a.duration;

          case 'title-asc':

            return a.title.localeCompare(b.title);

          case 'title-desc':

            return b.title.localeCompare(a.title);

          case 'year-desc':

            return b.year - a.year;

          case 'year-asc':

            return a.year - b.year;

          default:

            return 0;

        }

      });

    }

    return filtered;

  }, [movies, searchTerm, selectedGenre, sortBy]);

*// Get movies for current page*

  const getCurrentPageMovies = () => {

    if (activePage === 'favourites') {

      return movies.filter(*movie* => favourites.includes(movie.id));

    }

    return filteredAndSortedMovies;

  };

  const currentPageMovies = getCurrentPageMovies();

*// Event handlers*

  const handleAddToFavourites = (*movie*) => {

    const isFavourite = favourites.includes(movie.id);

    if (isFavourite) {

      setFavourites(*prev* => prev.filter(*id* => id !== movie.id));

      setToastMessage('Removed from favourites!');

    } else {

      setFavourites(*prev* => [...prev, movie.id]);

      setToastMessage('Added to favourites!');

    }

    setShowToast(true);

  };

  const handleShowDetails = (*movie*) => {

    setSelectedMovie(movie);

    setShowModal(true);

  };

  const handleRequestSubmit = (*formData*) => {

    console.log('Movie request submitted:', formData);

*// In a real app, this would send data to an API*

  };

  const handlePageChange = (*page*) => {

    setActivePage(page);

*// Reset filters when changing pages*

    if (page === 'home') {

      setSearchTerm('');

      setSelectedGenre('All');

      setSortBy('none');

    }

  };

  return (

    <div *className*="min-vh-100 bg-light">

      <NavigationBar *activePage*={activePage} *onPageChange*={handlePageChange} />

      {*/\* Content area with adjusted margin for fixed navbar \*/*}

      <div *className*="pt-5">

        {activePage === 'home' && (

          <>

            <HeroCarousel *movies*={movies} />

            <Container>

              <SearchFilterBar

*searchTerm*={searchTerm}

*selectedGenre*={selectedGenre}

*sortBy*={sortBy}

*onSearchChange*={setSearchTerm}

*onGenreChange*={setSelectedGenre}

*onSortChange*={setSortBy}

*allGenres*={allGenres}

*resultCount*={currentPageMovies.length}

              />

            </Container>

          </>

        )}

        <Container *className*="py-4">

          {activePage === 'request' ? (

            <MovieRequestForm

*allGenres*={allGenres}

*onSubmit*={handleRequestSubmit}

            />

          ) : (

            <>

              {activePage === 'favourites' && (

                <div *className*="mb-4">

                  <h2 *className*="text-dark">My Favourite Movies</h2>

                  {favourites.length === 0 && (

                    <Alert *variant*="info">

                      No favourites yet.

                    </Alert>

                  )}

                </div>

              )}

              {currentPageMovies.length === 0 ? (

                <Alert *variant*="warning">

                  {activePage === 'favourites'

                    ? 'No favourite movies found.'

                    : 'No movies found matching your criteria.'

                  }

                </Alert>

              ) : (

                <Row>

                  {currentPageMovies.map(*movie* => (

                    <Col *key*={movie.id} *lg*={4} *md*={6} *className*="mb-4">

                      <MovieCard

*movie*={movie}

*isFavourite*={favourites.includes(movie.id)}

*onAddToFavourites*={handleAddToFavourites}

*onShowDetails*={handleShowDetails}

                      />

                    </Col>

                  ))}

                </Row>

              )}

            </>

          )}

        </Container>

      </div>

      {*/\* Movie Details Modal \*/*}

      <MovieModal

*movie*={selectedMovie}

*show*={showModal}

*onHide*={() => setShowModal(false)}

      />

      {*/\* Toast Notifications \*/*}

      <ToastContainer *position*="top-end" *className*="p-3">

        <Toast

*show*={showToast}

*onClose*={() => setShowToast(false)}

*delay*={3000}

*autohide*

*bg*="success"

        >

          <Toast.Header>

            <strong *className*="me-auto">Movie Explorer</strong>

          </Toast.Header>

          <Toast.Body *className*="text-white">

            {toastMessage}

          </Toast.Body>

        </Toast>

      </ToastContainer>

    </div>

  );

}

export default App;