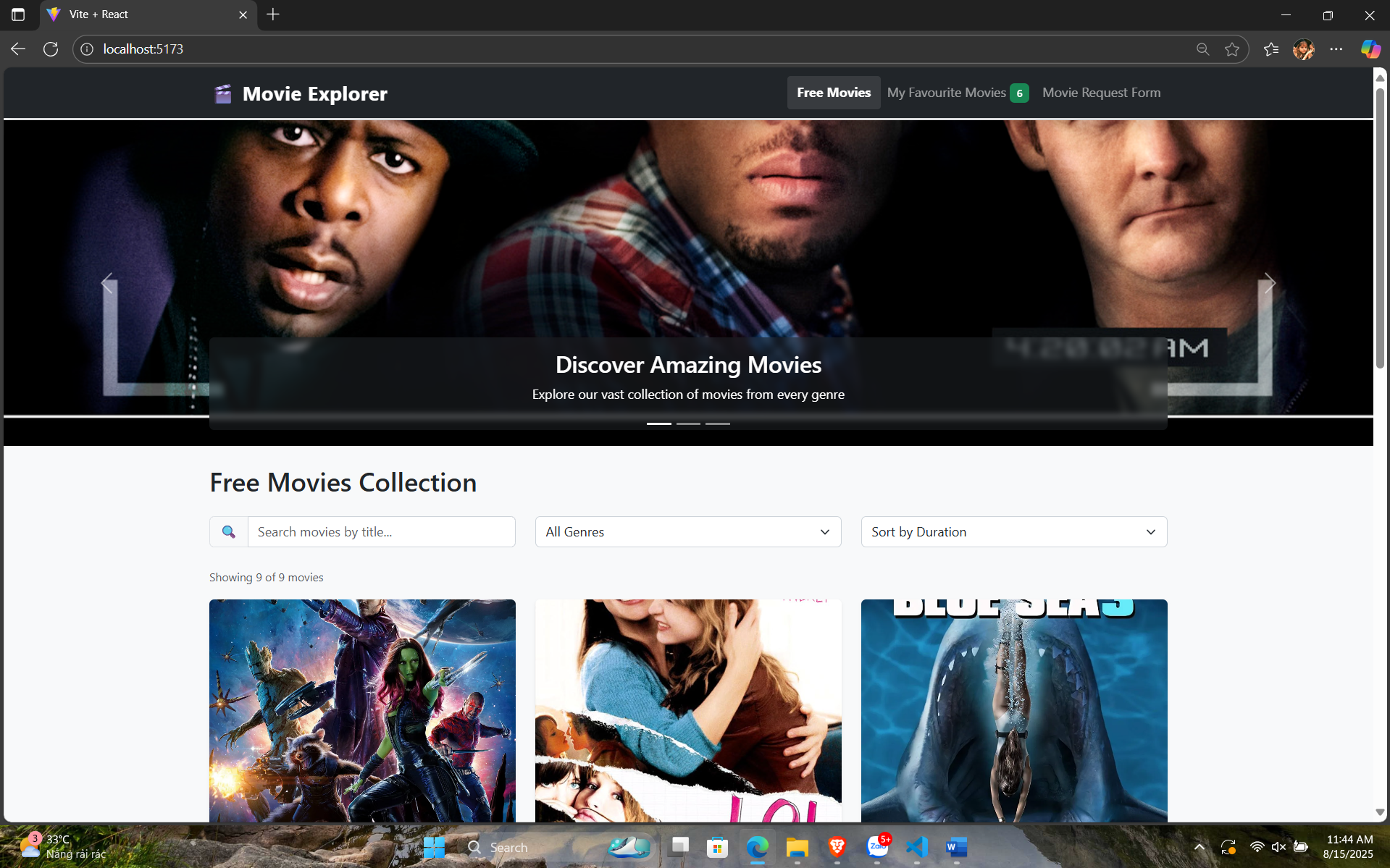
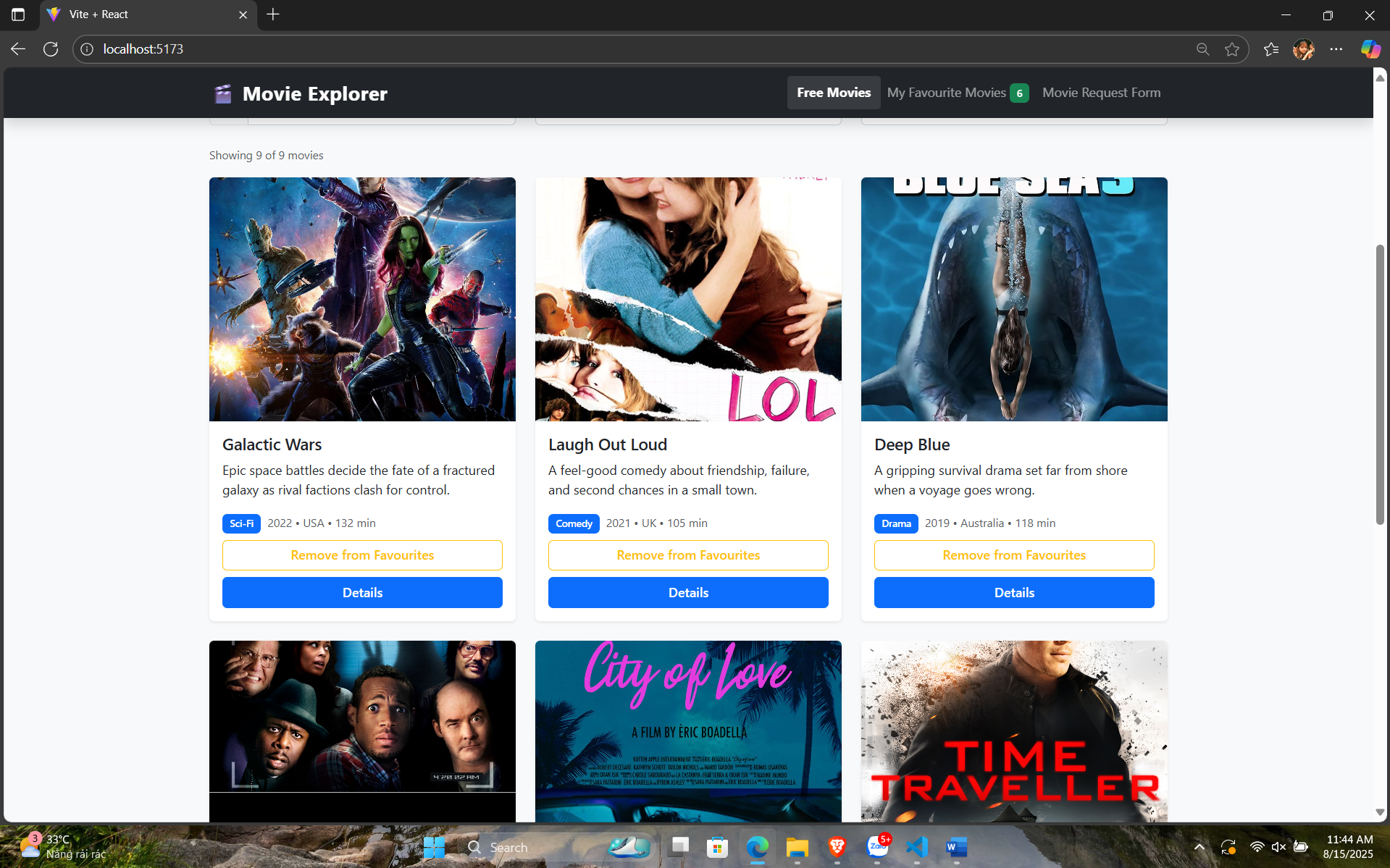
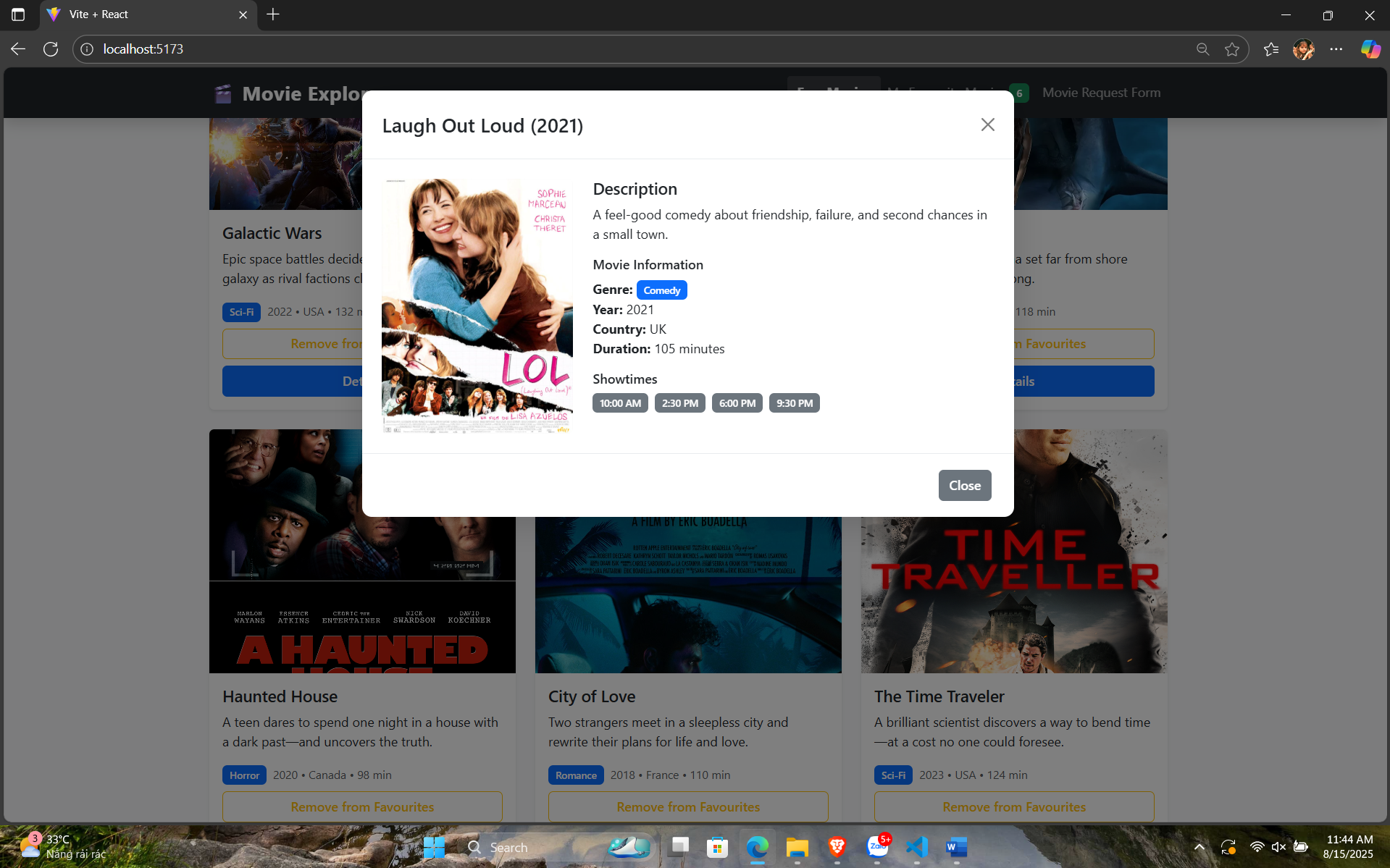
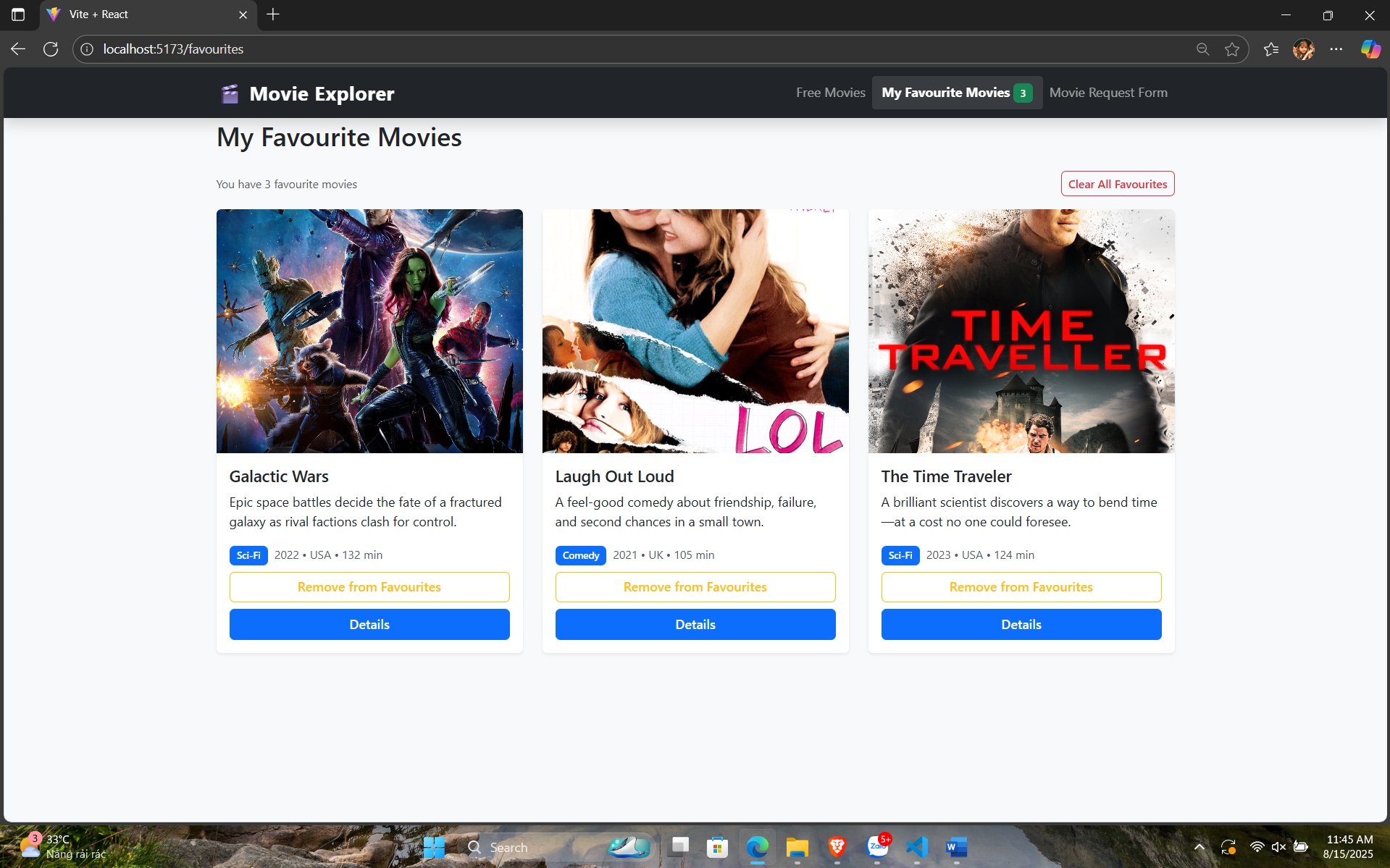
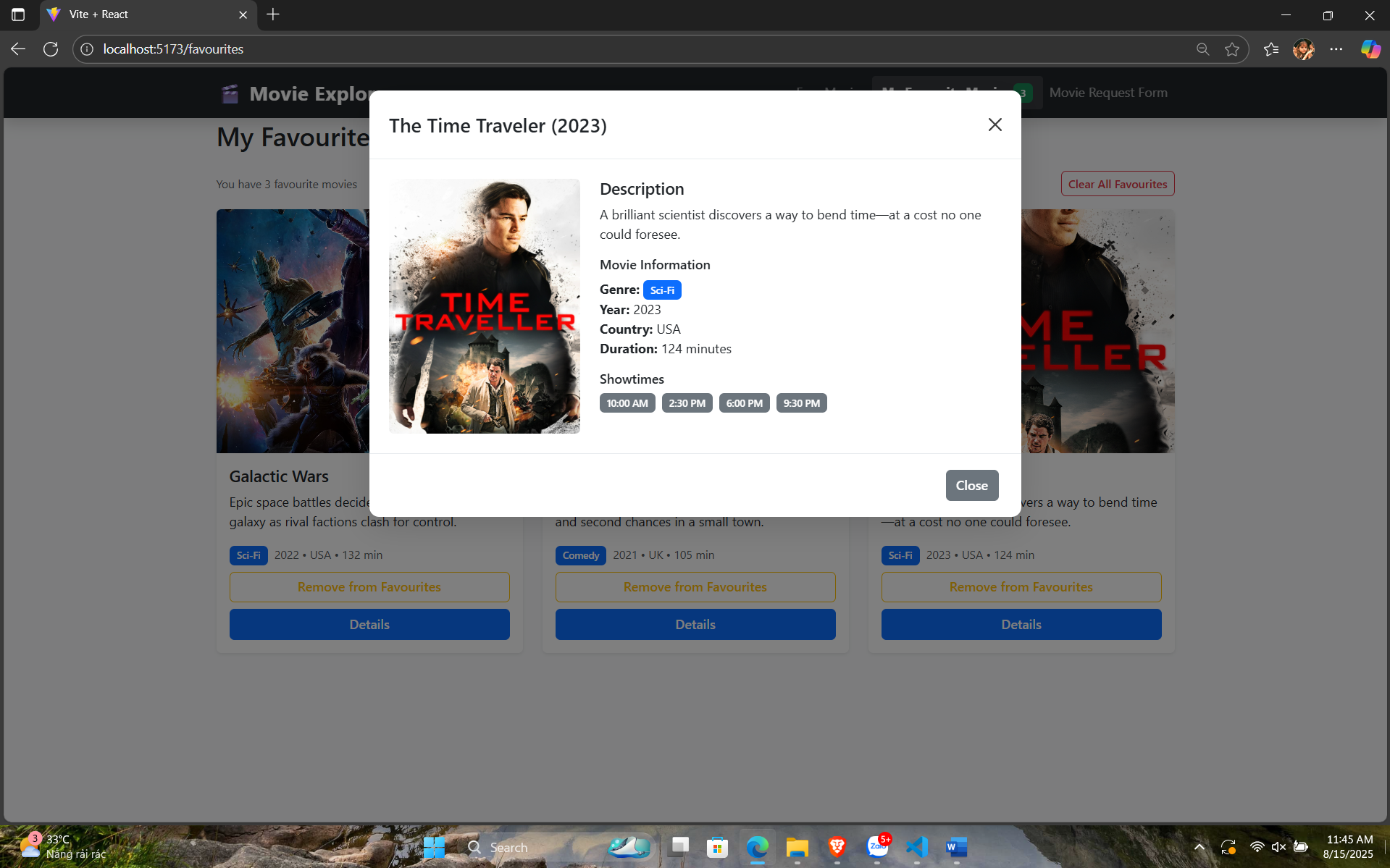
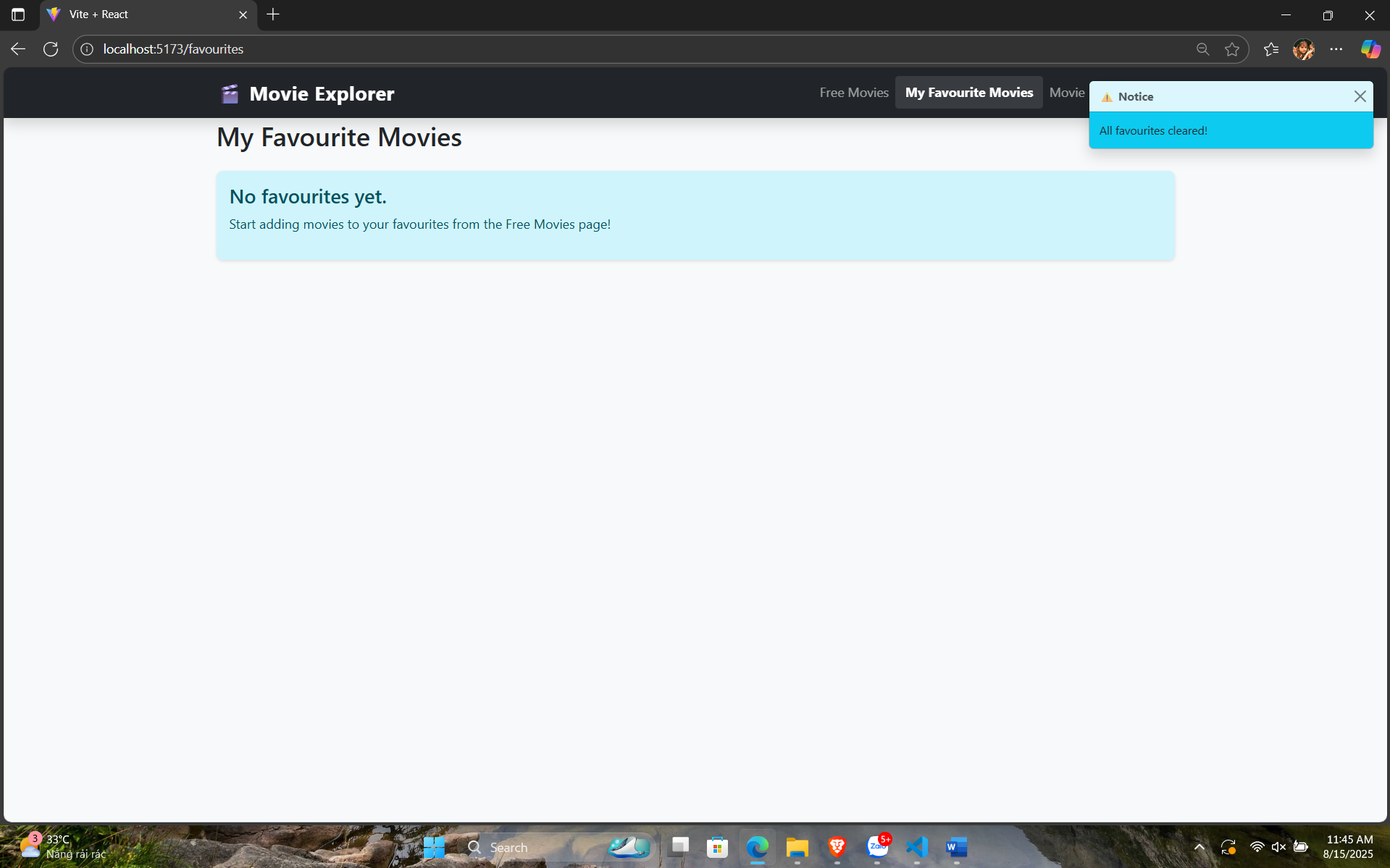
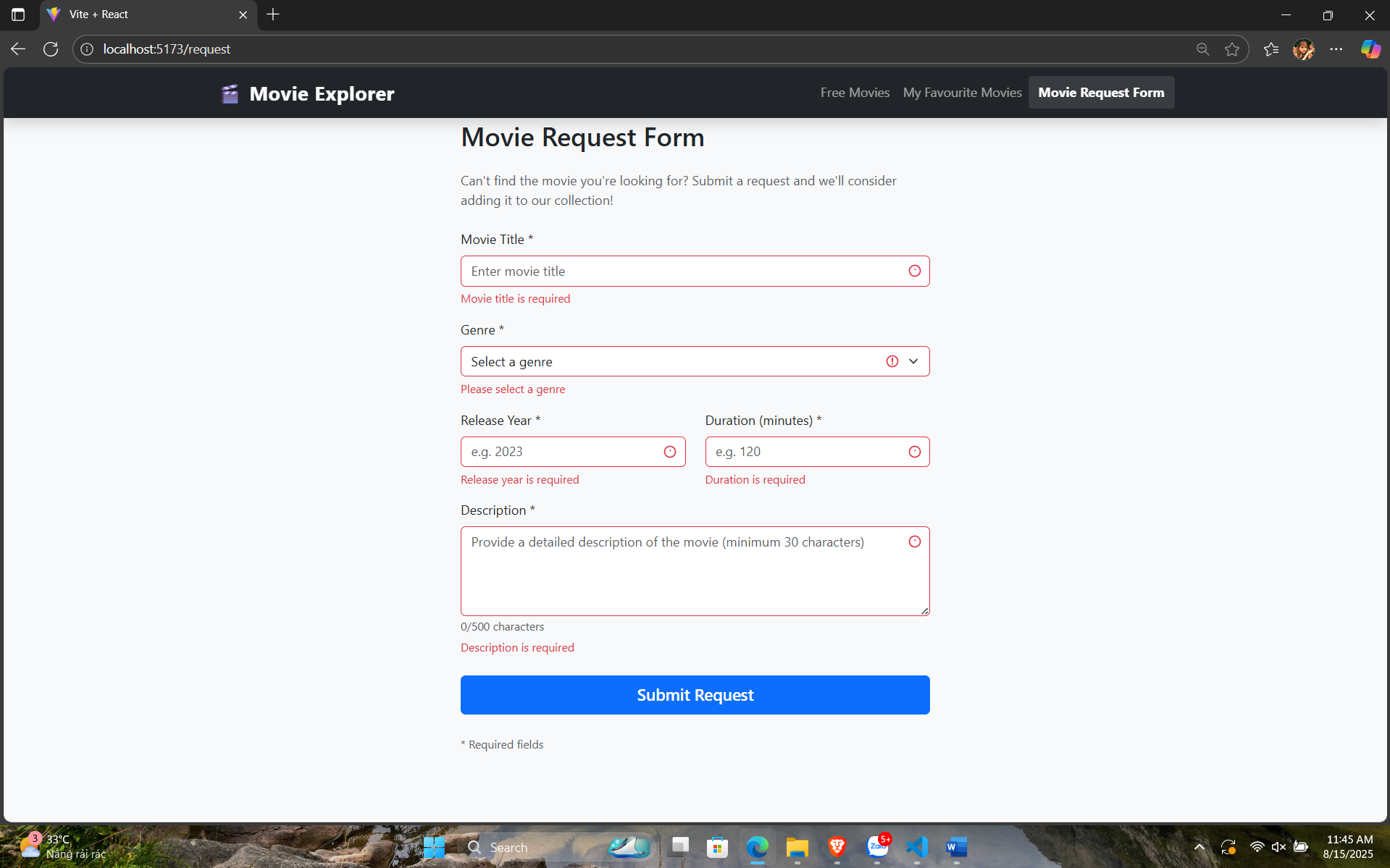
Lab2  
  
  
  
  
  
  
  
AppNavbar.jsx

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

import { Link } from 'react-router-dom';

import PropTypes from 'prop-types';

import { useFavourites } from '../hooks/useFavourites';

const AppNavbar = ({ currentPath }) => {

  const { favouritesCount } = useFavourites();

  return (

    <Navbar bg="dark" variant="dark" expand="lg" fixed="top" className="shadow">

      <Container>

        <Navbar.Brand as={Link} to="/" className="fw-bold">

          🎬 Movie Explorer

        </Navbar.Brand>

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

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

          <Nav className="ms-auto">

            <Nav.Link

              as={Link}

              to="/"

              className={currentPath === '/' ? 'active' : ''}

            >

              Free Movies

            </Nav.Link>

            <Nav.Link

              as={Link}

              to="/favourites"

              className={currentPath === '/favourites' ? 'active' : ''}

            >

              My Favourite Movies

              {favouritesCount > 0 && (

                <Badge bg="success" className="ms-1">

                  {favouritesCount}

                </Badge>

              )}

            </Nav.Link>

            <Nav.Link

              as={Link}

              to="/request"

              className={currentPath === '/request' ? 'active' : ''}

            >

              Movie Request Form

            </Nav.Link>

          </Nav>

        </Navbar.Collapse>

      </Container>

    </Navbar>

  );

};

AppNavbar.propTypes = {

  currentPath: PropTypes.string.isRequired

};

export default AppNavbar;

HeroCarousel.jsx

import { Carousel } from 'react-bootstrap';

import { carouselSlides } from '../movie.js';

const HeroCarousel = () => {

  return (

    <Carousel className="mb-4" interval={4000}>

      {carouselSlides.map((slide) => (

        <Carousel.Item key={slide.id}>

          <img

            className="d-block w-100"

            src={slide.image}

            alt={slide.title}

            style={{ height: '400px', objectFit: 'cover' }}

          />

          <Carousel.Caption className="bg-dark bg-opacity-50 rounded p-3">

            <h3>{slide.title}</h3>

            <p>{slide.caption}</p>

          </Carousel.Caption>

        </Carousel.Item>

      ))}

    </Carousel>

  );

};

export default HeroCarousel;

MovieCard.jsx

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

import { useState } from 'react';

import PropTypes from 'prop-types';

const MovieCard = ({ movie, onAddToFavourites, isFavourite, onShowToast }) => {

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

  const handleAddToFavourites = () => {

    onAddToFavourites(movie.id);

  };

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

    return text.length > maxLength ? text.substring(0, maxLength) + '...' : text;

  };

  return (

    <>

      <Card className="h-100 shadow-sm movie-card">

        <Card.Img

          variant="top"

          src={movie.poster}

          alt={`${movie.title} poster`}

        />

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

          <Card.Title className="mb-2">{movie.title}</Card.Title>

          <Card.Text className="flex-grow-1">

            {truncateDescription(movie.description)}

          </Card.Text>

          <div className="mb-2">

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

            <small className="text-muted">

              {movie.year} • {movie.country} • {movie.duration} min

            </small>

          </div>

          <div className="d-grid gap-2">

            <Button

              variant={isFavourite ? "outline-warning" : "outline-success"}

              onClick={handleAddToFavourites}

            >

              {isFavourite ? "Remove from Favourites" : "Add to Favourites"}

            </Button>

            <Button variant="primary" onClick={() => setShowModal(true)}>

              Details

            </Button>

          </div>

        </Card.Body>

      </Card>

      {/\* Modal for movie details \*/}

      <Modal show={showModal} onHide={() => setShowModal(false)} size="lg">

        <Modal.Header closeButton>

          <Modal.Title>{movie.title} ({movie.year})</Modal.Title>

        </Modal.Header>

        <Modal.Body>

          <div className="row">

            <div className="col-md-4">

              <img

                src={movie.poster}

                alt={`${movie.title} poster`}

                className="img-fluid rounded"

              />

            </div>

            <div className="col-md-8">

              <h5>Description</h5>

              <p>{movie.description}</p>

              <h6>Movie Information</h6>

              <ul className="list-unstyled">

                <li><strong>Genre:</strong> <Badge bg="primary">{movie.genre}</Badge></li>

                <li><strong>Year:</strong> {movie.year}</li>

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

                <li><strong>Duration:</strong> {movie.duration} minutes</li>

              </ul>

              <h6>Showtimes</h6>

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

                <Badge bg="secondary">10:00 AM</Badge>

                <Badge bg="secondary">2:30 PM</Badge>

                <Badge bg="secondary">6:00 PM</Badge>

                <Badge bg="secondary">9:30 PM</Badge>

              </div>

            </div>

          </div>

        </Modal.Body>

        <Modal.Footer>

          <Button variant="secondary" onClick={() => setShowModal(false)}>

            Close

          </Button>

        </Modal.Footer>

      </Modal>

    </>

  );

};

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,

    duration: PropTypes.number.isRequired,

    country: PropTypes.string.isRequired

  }).isRequired,

  onAddToFavourites: PropTypes.func.isRequired,

  isFavourite: PropTypes.bool.isRequired,

  onShowToast: PropTypes.func.isRequired

};

export default MovieCard;

NotificationToast.jsx

import { Toast, ToastContainer } from 'react-bootstrap';

import PropTypes from 'prop-types';

const NotificationToast = ({ show, onClose, message, variant = 'success' }) => {

  return (

    <ToastContainer position="top-end" className="p-3" style={{ zIndex: 9999 }}>

      <Toast

        show={show}

        onClose={onClose}

        delay={3000}

        autohide

        bg={variant}

      >

        <Toast.Header>

          <strong className="me-auto">

            {variant === 'success' ? '✅ Success' : '⚠️ Notice'}

          </strong>

        </Toast.Header>

        <Toast.Body className={variant === 'success' ? 'text-white' : 'text-dark'}>

          {message}

        </Toast.Body>

      </Toast>

    </ToastContainer>

  );

};

NotificationToast.propTypes = {

  show: PropTypes.bool.isRequired,

  onClose: PropTypes.func.isRequired,

  message: PropTypes.string.isRequired,

  variant: PropTypes.oneOf(['success', 'warning', 'danger', 'info'])

};

export default NotificationToast;

SearchFilterBar.jsx

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

import PropTypes from 'prop-types';

import { allGenres } from '../movie.js';

const SearchFilterBar = ({

  searchTerm,

  setSearchTerm,

  selectedGenre,

  setSelectedGenre,

  sortBy,

  setSortBy

}) => {

  return (

    <Row className="mb-4">

      <Col md={4}>

        <InputGroup>

          <InputGroup.Text>

            🔍

          </InputGroup.Text>

          <Form.Control

            type="text"

            placeholder="Search movies by title..."

            value={searchTerm}

            onChange={(e) => setSearchTerm(e.target.value)}

          />

        </InputGroup>

      </Col>

      <Col md={4}>

        <Form.Select

          value={selectedGenre}

          onChange={(e) => setSelectedGenre(e.target.value)}

        >

          {allGenres.map(genre => (

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

              {genre === 'All' ? 'All Genres' : genre}

            </option>

          ))}

        </Form.Select>

      </Col>

      <Col md={4}>

        <Form.Select

          value={sortBy}

          onChange={(e) => setSortBy(e.target.value)}

        >

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

          <option value="duration-asc">Duration ↑ (Shortest first)</option>

          <option value="duration-desc">Duration ↓ (Longest first)</option>

        </Form.Select>

      </Col>

    </Row>

  );

};

SearchFilterBar.propTypes = {

  searchTerm: PropTypes.string.isRequired,

  setSearchTerm: PropTypes.func.isRequired,

  selectedGenre: PropTypes.string.isRequired,

  setSelectedGenre: PropTypes.func.isRequired,

  sortBy: PropTypes.string.isRequired,

  setSortBy: PropTypes.func.isRequired

};

export default SearchFilterBar;

useFavourites.js

import { useState, useEffect } from 'react';

import {

  getFavourites,

  addToFavourites as addToFavouritesStorage,

  clearAllFavourites

} from '../utils/favouritesStorage';

export const useFavourites = () => {

  const [favourites, setFavourites] = useState([]);

  // Load favourites từ localStorage khi component mount

  useEffect(() => {

    const loadedFavourites = getFavourites();

    setFavourites(loadedFavourites);

  }, []);

  // Listen for changes từ các components khác

  useEffect(() => {

    const handleFavouritesChanged = (event) => {

      setFavourites(event.detail);

    };

    window.addEventListener('favouritesChanged', handleFavouritesChanged);

    return () => {

      window.removeEventListener('favouritesChanged', handleFavouritesChanged);

    };

  }, []);

  const addToFavourites = (movieId) => {

    const newFavourites = addToFavouritesStorage(movieId);

    setFavourites(newFavourites);

    return newFavourites;

  };

  const isFavourite = (movieId) => {

    return favourites.includes(movieId);

  };

  const clearFavourites = () => {

    const newFavourites = clearAllFavourites();

    setFavourites(newFavourites);

    return newFavourites;

  };

  return {

    favourites,

    addToFavourites,

    isFavourite,

    clearFavourites,

    favouritesCount: favourites.length

  };

};

favouritesStorage.js

// LocalStorage utility cho favourites

const FAVOURITES\_KEY = 'movieFavourites';

export const getFavourites = () => {

  try {

    const saved = localStorage.getItem(FAVOURITES\_KEY);

    return saved ? JSON.parse(saved) : [];

  } catch (error) {

    console.error('Error loading favourites:', error);

    return [];

  }

};

export const saveFavourites = (favourites) => {

  try {

    localStorage.setItem(FAVOURITES\_KEY, JSON.stringify(favourites));

    // Dispatch custom event để notify các components khác

    window.dispatchEvent(new CustomEvent('favouritesChanged', {

      detail: favourites

    }));

  } catch (error) {

    console.error('Error saving favourites:', error);

  }

};

export const addToFavourites = (movieId) => {

  const currentFavourites = getFavourites();

  const newFavourites = currentFavourites.includes(movieId)

    ? currentFavourites.filter(id => id !== movieId)

    : [...currentFavourites, movieId];

  saveFavourites(newFavourites);

  return newFavourites;

};

export const isFavourite = (movieId) => {

  return getFavourites().includes(movieId);

};

export const clearAllFavourites = () => {

  saveFavourites([]);

  return [];

};

FavouritesPage.jsx

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

import { useState } from 'react';

import MovieCard from '../components/MovieCard';

import NotificationToast from '../components/NotificationToast';

import { useFavourites } from '../hooks/useFavourites';

import { movies } from '../movie.js';

const FavouritesPage = () => {

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

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

  const [toastVariant, setToastVariant] = useState('success');

  const { favourites, addToFavourites, isFavourite, clearFavourites } = useFavourites();

  const favouriteMovies = movies.filter(movie => favourites.includes(movie.id));

  const handleAddToFavourites = (movieId) => {

    const movie = movies.find(m => m.id === movieId);

    const wasFavourite = isFavourite(movieId);

    addToFavourites(movieId);

    handleShowToast(

      wasFavourite

        ? `${movie.title} removed from favourites!`

        : `${movie.title} added to favourites!`,

      wasFavourite ? 'warning' : 'success'

    );

  };

  const handleClearAll = () => {

    clearFavourites();

    handleShowToast('All favourites cleared!', 'info');

  };

  const handleShowToast = (message, variant = 'success') => {

    setToastMessage(message);

    setToastVariant(variant);

    setShowToast(true);

  };

  return (

    <div className="page-content">

      <Container>

        <h2 className="mb-4">My Favourite Movies</h2>

        {favouriteMovies.length === 0 ? (

          <Alert variant="info">

            <Alert.Heading>No favourites yet.</Alert.Heading>

            <p>Start adding movies to your favourites from the Free Movies page!</p>

          </Alert>

        ) : (

          <>

            <div className="d-flex justify-content-between align-items-center mb-3">

              <small className="text-muted">

                You have {favouriteMovies.length} favourite movie{favouriteMovies.length !== 1 ? 's' : ''}

              </small>

              <Button variant="outline-danger" size="sm" onClick={handleClearAll}>

                Clear All Favourites

              </Button>

            </div>

            <Row>

              {favouriteMovies.map(movie => (

                <Col key={movie.id} xs={12} sm={6} lg={4} className="mb-4">

                  <MovieCard

                    movie={movie}

                    onAddToFavourites={handleAddToFavourites}

                    isFavourite={isFavourite(movie.id)}

                    onShowToast={handleShowToast}

                  />

                </Col>

              ))}

            </Row>

          </>

        )}

      </Container>

      <NotificationToast

        show={showToast}

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

        message={toastMessage}

        variant={toastVariant}

      />

    </div>

  );

};

export default FavouritesPage;

HomePage.jsx

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

import { useState, useMemo } from 'react';

import HeroCarousel from '../components/HeroCarousel';

import SearchFilterBar from '../components/SearchFilterBar';

import MovieCard from '../components/MovieCard';

import NotificationToast from '../components/NotificationToast';

import { useFavourites } from '../hooks/useFavourites';

import { movies } from '../movie.js';

const HomePage = () => {

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

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

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

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

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

  const [toastVariant, setToastVariant] = useState('success');

  const { addToFavourites, isFavourite } = useFavourites();

  // Filter and sort movies using useMemo for performance

  const filteredAndSortedMovies = useMemo(() => {

    let filtered = movies;

    // Filter by search term

    if (searchTerm) {

      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 by duration

    if (sortBy === 'duration-asc') {

      filtered = [...filtered].sort((a, b) => a.duration - b.duration);

    } else if (sortBy === 'duration-desc') {

      filtered = [...filtered].sort((a, b) => b.duration - a.duration);

    }

    return filtered;

  }, [searchTerm, selectedGenre, sortBy]);

  const handleAddToFavourites = (movieId) => {

    const movie = movies.find(m => m.id === movieId);

    const wasFavourite = isFavourite(movieId);

    addToFavourites(movieId);

    handleShowToast(

      wasFavourite

        ? `${movie.title} removed from favourites!`

        : `${movie.title} added to favourites!`,

      wasFavourite ? 'warning' : 'success'

    );

  };

  const handleShowToast = (message, variant = 'success') => {

    setToastMessage(message);

    setToastVariant(variant);

    setShowToast(true);

  };

  return (

    <div className="page-content">

      <HeroCarousel />

      <Container>

        <h2 className="mb-4">Free Movies Collection</h2>

        <SearchFilterBar

          searchTerm={searchTerm}

          setSearchTerm={setSearchTerm}

          selectedGenre={selectedGenre}

          setSelectedGenre={setSelectedGenre}

          sortBy={sortBy}

          setSortBy={setSortBy}

        />

        <div className="mb-3">

          <small className="text-muted">

            Showing {filteredAndSortedMovies.length} of {movies.length} movies

          </small>

        </div>

        {filteredAndSortedMovies.length === 0 ? (

          <Alert variant="info">

            <Alert.Heading>No movies found</Alert.Heading>

            <p>Try adjusting your search criteria or browse all genres.</p>

          </Alert>

        ) : (

          <Row>

            {filteredAndSortedMovies.map(movie => (

              <Col key={movie.id} xs={12} sm={6} lg={4} className="mb-4">

                <MovieCard

                  movie={movie}

                  onAddToFavourites={handleAddToFavourites}

                  isFavourite={isFavourite(movie.id)}

                  onShowToast={handleShowToast}

                />

              </Col>

            ))}

          </Row>

        )}

      </Container>

      <NotificationToast

        show={showToast}

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

        message={toastMessage}

        variant={toastVariant}

      />

    </div>

  );

};

export default HomePage;

RequestFormPage.jsx

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

import { useState } from 'react';

import { allGenres } from '../movie.js';

const RequestFormPage = () => {

  const [formData, setFormData] = useState({

    title: '',

    genre: '',

    year: '',

    duration: '',

    description: ''

  });

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

  const [isSubmitting, setIsSubmitting] = useState(false);

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

  const validateForm = () => {

    const newErrors = {};

    // Title validation

    if (!formData.title.trim()) {

      newErrors.title = 'Movie title is required';

    }

    // Genre validation

    if (!formData.genre || formData.genre === 'All') {

      newErrors.genre = 'Please select a genre';

    }

    // Year validation

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

    const year = parseInt(formData.year);

    if (!formData.year) {

      newErrors.year = 'Release year is required';

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

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

    }

    // Duration validation

    const duration = parseInt(formData.duration);

    if (!formData.duration) {

      newErrors.duration = 'Duration is required';

    } else if (isNaN(duration) || duration <= 0) {

      newErrors.duration = 'Duration must be a positive number';

    } else if (duration > 600) {

      newErrors.duration = 'Duration seems too long (max 600 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';

    } else if (formData.description.trim().length > 500) {

      newErrors.description = 'Description must be less than 500 characters';

    }

    return newErrors;

  };

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setFormData(prev => ({

      ...prev,

      [name]: value

    }));

    // Clear error for this field when user starts typing

    if (errors[name]) {

      setErrors(prev => ({

        ...prev,

        [name]: ''

      }));

    }

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    const newErrors = validateForm();

    setErrors(newErrors);

    if (Object.keys(newErrors).length === 0) {

      setIsSubmitting(true);

      // Simulate API call

      setTimeout(() => {

        setIsSubmitting(false);

        setShowSuccess(true);

        // Reset form

        setFormData({

          title: '',

          genre: '',

          year: '',

          duration: '',

          description: ''

        });

        // Hide success message after 5 seconds

        setTimeout(() => {

          setShowSuccess(false);

        }, 5000);

      }, 1000);

    }

  };

  const genreOptions = allGenres.filter(genre => genre !== 'All');

  return (

    <div className="page-content">

      <Container>

        <Row className="justify-content-center">

          <Col md={8} lg={6}>

            <h2 className="mb-4">Movie Request Form</h2>

            <p className="text-muted mb-4">

              Can't find the movie you're looking for? Submit a request and we'll consider adding it to our collection!

            </p>

            {showSuccess && (

              <Alert variant="success" className="mb-4">

                <Alert.Heading>Request submitted. Thank you!</Alert.Heading>

                <p>We've received your movie request and will review it soon.</p>

              </Alert>

            )}

            <Form onSubmit={handleSubmit}>

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

                <Form.Label>Movie Title \*</Form.Label>

                <Form.Control

                  type="text"

                  name="title"

                  value={formData.title}

                  onChange={handleInputChange}

                  isInvalid={!!errors.title}

                  placeholder="Enter movie title"

                />

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

                  {errors.title}

                </Form.Control.Feedback>

              </Form.Group>

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

                <Form.Label>Genre \*</Form.Label>

                <Form.Select

                  name="genre"

                  value={formData.genre}

                  onChange={handleInputChange}

                  isInvalid={!!errors.genre}

                >

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

                  {genreOptions.map(genre => (

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

                  ))}

                </Form.Select>

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

                  {errors.genre}

                </Form.Control.Feedback>

              </Form.Group>

              <Row>

                <Col md={6}>

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

                    <Form.Label>Release Year \*</Form.Label>

                    <Form.Control

                      type="number"

                      name="year"

                      value={formData.year}

                      onChange={handleInputChange}

                      isInvalid={!!errors.year}

                      placeholder="e.g. 2023"

                      min="1900"

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

                    />

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

                      {errors.year}

                    </Form.Control.Feedback>

                  </Form.Group>

                </Col>

                <Col md={6}>

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

                    <Form.Label>Duration (minutes) \*</Form.Label>

                    <Form.Control

                      type="number"

                      name="duration"

                      value={formData.duration}

                      onChange={handleInputChange}

                      isInvalid={!!errors.duration}

                      placeholder="e.g. 120"

                      min="1"

                      max="600"

                    />

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

                      {errors.duration}

                    </Form.Control.Feedback>

                  </Form.Group>

                </Col>

              </Row>

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

                <Form.Label>Description \*</Form.Label>

                <Form.Control

                  as="textarea"

                  rows={4}

                  name="description"

                  value={formData.description}

                  onChange={handleInputChange}

                  isInvalid={!!errors.description}

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

                />

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

                  {formData.description.length}/500 characters

                  {formData.description.length < 30 && formData.description.length > 0 &&

                    ` (${30 - formData.description.length} more needed)`

                  }

                </Form.Text>

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

                  {errors.description}

                </Form.Control.Feedback>

              </Form.Group>

              <div className="d-grid">

                <Button

                  variant="primary"

                  type="submit"

                  disabled={isSubmitting}

                  size="lg"

                >

                  {isSubmitting ? 'Submitting...' : 'Submit Request'}

                </Button>

              </div>

            </Form>

            <div className="mt-4">

              <small className="text-muted">\* Required fields</small>

            </div>

          </Col>

        </Row>

      </Container>

    </div>

  );

};

export default RequestFormPage;

App.jsx

import { BrowserRouter as Router, Routes, Route, useLocation } from 'react-router-dom';

import AppNavbar from './components/AppNavbar';

import HomePage from './pages/HomePage';

import FavouritesPage from './pages/FavouritesPage';

import RequestFormPage from './pages/RequestFormPage';

import './App.css';

// Component wrapper to get current location for navbar

const AppContent = () => {

  const location = useLocation();

  return (

    <>

      <AppNavbar currentPath={location.pathname} />

      <Routes>

        <Route path="/" element={<HomePage />} />

        <Route path="/favourites" element={<FavouritesPage />} />

        <Route path="/request" element={<RequestFormPage />} />

      </Routes>

    </>

  );

};

function App() {

  return (

    <Router>

      <div className="App">

        <AppContent />

      </div>

    </Router>

  );

}

export default App