

BỘ THÔNG TIN VÀ TRUYỀN THÔNG HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 10

Triển khai dự án (phần 3)

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Lê Quang Thanh – B22DCVT509

I. Khởi tạo code giao diện các trang chi tiết pages (tiếp)

1. ProfilePage.js:

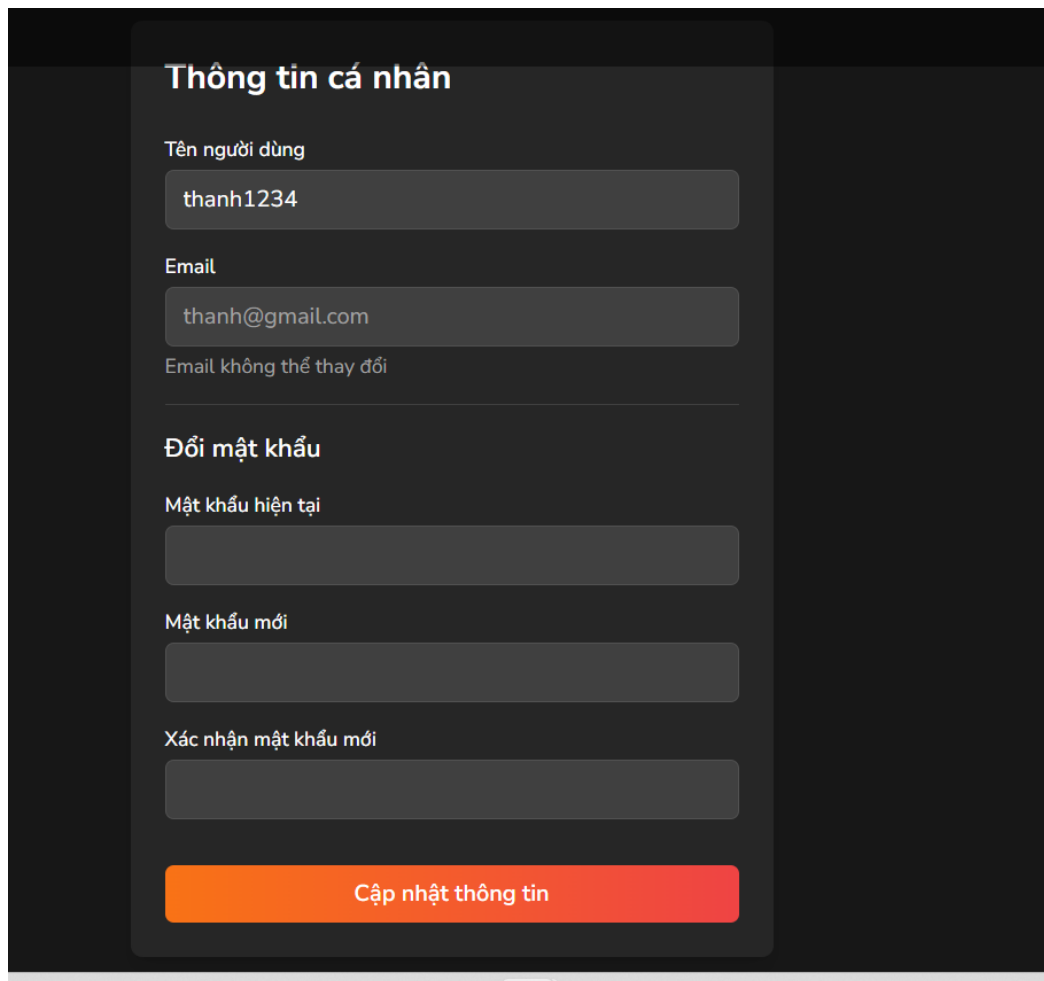
Trang thông tin cá nhân người dùng

Cho phép cập nhật:

Username

Password

Hiển thị thông tin tài khoản

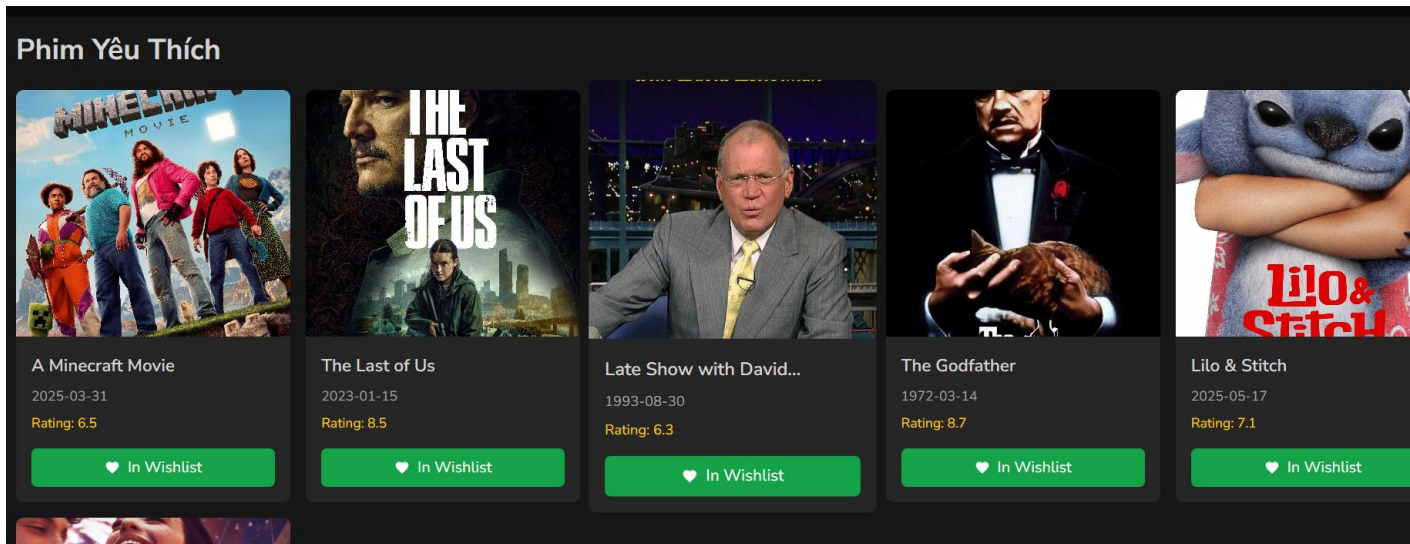


The screenshot shows a dark-themed user profile update form. At the top, the title 'Thông tin cá nhân' is displayed. Below it, there are three input fields: 'Tên người dùng' (Username) with the value 'thanh1234', 'Email' with the value 'thanh@gmail.com', and a disabled field with the text 'Email không thể thay đổi'. Underneath these is a section titled 'Đổi mật khẩu' (Change password) containing three input fields: 'Mật khẩu hiện tại' (Current password), 'Mật khẩu mới' (New password), and 'Xác nhận mật khẩu mới' (Confirm new password). At the bottom of the form is a large orange button labeled 'Cập nhật thông tin' (Update information).

2. FavouritePage.js:

Hiển thị danh sách phim yêu thích của người dùng

Cho phép xóa phim khỏi danh sách



3. PersonPage.js:

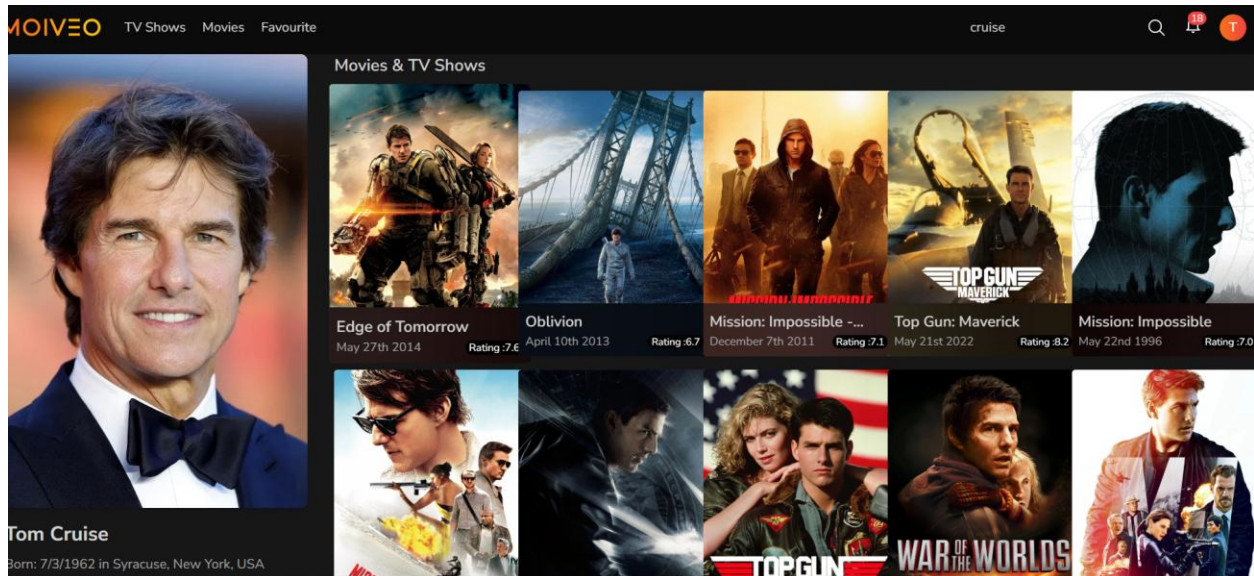
Hiển thị thông tin chi tiết về diễn viên/người nổi tiếng khi tìm kiếm

Bao gồm:

Thông tin cá nhân

Tiểu sử

Danh sách phim đã tham gia



4. ForgotPasswordPage.js:

Cho phép reset mật khẩu

Xác thực qua email với mã OTP

Đặt mật khẩu mới

Quên mật khẩu

Nhập email đăng ký

Gửi mã OTP

II. Khởi tạo code components

1. Videoplay.js

```

app-main > src > components > JS VideoPlay.js > [e] VideoPlay
import React from 'react'
import { IoClose } from "react-icons/io5";
import useFetchDetails from '../hooks/useFetchDetails';

const VideoPlay = ({data, close, media_type}) => {
  const { data : videoData } = useFetchDetails(`/${media_type}/${data?.id}/videos`)

  return (

```

Lấy dữ liệu video từ API dựa trên ID phim và loại media

2. BannerHome.js

- Lấy dữ liệu banner từ Redux store:

```

const BannerHome = () => {
  const bannerData = useSelector(state => state.movieoData.bannerData)
  const imageURL = useSelector(state => state.movieoData.imageURL)

```

- State quản lý slide hiện tại

```

const [currentImage, setCurrentImage] = useState(0)

```

- Hiệu ứng Slide tự lướt:

```

useEffect(()=>{
  const interval = setInterval(()=>{
    if(currentImage < bannerData.length - 1){
      handleNext()
    }else{
      setCurrentImage(0)
    }
  },5000)

  return ()=>clearInterval(interval)
},[bannerData,imageURL,currentImage])

```

- Các hàm điều khiển slide:

```

const handleNext = ()=>{
  if(currentImage < bannerData.length - 1){
    setCurrentImage(preve => preve + 1)
  }
}
const handleprevious = ()=>{
  if(currentImage > 0){
    setCurrentImage(preve => preve - 1)
  }
}

```

3. Header.js

- Header hover khi cuộn :

```

useEffect(() => {
  let ticking = false;
  const handleScroll = () => {
    if (!ticking) {
      window.requestAnimationFrame(() => {
        if (window.scrollY < 50) {
          setShowHeader(true);
          setLastScrollY(window.scrollY);
        } else if (window.scrollY > lastScrollY) {
          setShowHeader(false);
          setLastScrollY(window.scrollY);
        } else {
          setShowHeader(true);
          setLastScrollY(window.scrollY);
        }
        ticking = false;
      });
      ticking = true;
    }
  };
});

```

- Khởi tạo các hàm quản lí UI trên thanh header:

```

const location = useLocation()
const removeSpace = location?.search?.slice(3)?.split("%20")?.join(" ")
const [searchInput, setSearchInput] = useState(removeSpace)
const navigate = useNavigate()
const { isAuthenticated, user, logout } = useAuth();
const [showUserMenu, setShowUserMenu] = useState(false);
const [showHeader, setShowHeader] = useState(true);
const [lastScrollY, setLastScrollY] = useState(window.scrollY);
const [showNotifications, setShowNotifications] = useState(false);
const [notifications, setNotifications] = useState(() => {
  try {

```

4. Card.js: Hiển thị thông tin phim

Poster phim

Thông tin cơ bản: tên, rating, ngày phát hành

Badge trending nếu là phim trending

```

const Card = ({data,trending,index,media_type }) => {
  const imageURL = useSelector(state => state.movieoData.imageURL)

  const mediaType = data.media_type ?? media_type
  return (
    <Link to={"/"+mediaType+"/"+data.id} className='w-full min-w-[230px] max-w-[230px] h-80' >
      {
        data?.poster_path ? (
          <img
            src={imageURL+data?.poster_path}
          />
        ) : (
          <div className='bg-neutral-800 h-full w-full flex justify-center items-center'>
            No image found
          </div>
        )
      }
    </Link>
  )
}

```

5. CommentSection.js

- State cho form reply

```
// Comment print out
const CommentItem = ({ comment, replies, onReply, onLike, onDelete, userId, movieId }) => {
  const [showReply, setShowReply] = useState(false);
  const [replyContent, setReplyContent] = useState('');

```

- Kiểm tra user hiện tại đã like comment này chưa

```
const liked = comment.likes && Array.isArray(comment.likes) && userId ? comment.likes.includes(userId) : false;

const handleReply = async (e) => {

```

- Xử lý nút submit reply

```
const handleReply = async (e) => {
  e.preventDefault();
  if (!replyContent.trim()) return;
  await onReply({
    content: replyContent,
    parentId: comment._id,
  });
  setReplyContent('');
  setShowReply(false);
};

```

- Lấy thông tin user từ context

```
const CommentSection = ({ movieId }) => {
  const { user } = useAuth();

```

- States

```
const [comments, setComments] = useState([]);
const [content, setContent] = useState('');
const [loading, setLoading] = useState(false);

```

- Fetch comments từ API

```
const fetchComments = async () => {
  try {
    const res = await axios.get(`${API_BASE_URL}/comments/${movieId}`);
    setComments(res.data);
  } catch (err) {
    setComments([]);
  }
};

```


- Load comments khi movieId thay đổi

```
useEffect(() => {  
  if (movieId) fetchComments();  
}, [movieId]);
```

- Xử lý thêm comment mới

```
const handleAddComment = async (extra = {}) => {  
  if (!user) return;  
  const commentContent = (extra.content !== undefined ? extra.content : content).trim();  
  // Lấy userId và username  
  const userId = user._id || user.id;  
  const username = user.username || user.name;  
  
  console.log('DEBUG gửi comment:', { movieId, userId, username, commentContent });  
  if (!commentContent) return;  
  if (!movieId || !userId || !username) {  
    alert('Thiếu thông tin người dùng hoặc phim!');  
    return;  
  }  
  setLoading(true);
```

- Gọi API thêm comment

```
try {  
  await axios.post(`${API_BASE_URL}/comments`, {  
    movieId,  
    content: commentContent,  
    parentId: extra.parentId || null,  
    userId,  
    username,  
  });
```

- Reset form và reload comments

```
setContent('');  
fetchComments();
```

- Hiển thị notification

```
if (window.notify) {  
  window.notify('Bình luận thành công!');  
}
```

- Lưu notification vào DB

```
const token = localStorage.getItem('token');
if (token) {
  fetch(`${API_BASE_URL}/notifications`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'x-auth-token': token
    },
    body: JSON.stringify({ message: 'Bình luận thành công!' })
  });
}
} catch (err) {
  if (err.response && err.response.data && err.response.data.message) {
    alert('Lỗi: ' + err.response.data.message);
  } else {
    alert('Lỗi gửi bình luận!');
  }
}
}
setLoading(false);
};
```

- Xử lý xóa comment

```
const handleDelete = async (commentId) => {
  if (!user) return;
  setLoading(true);
  try {
    const API_BASE_URL: "http://localhost:5000/api"
    await axios.delete(`${API_BASE_URL}/comments/${commentId}`);
    fetchComments();
  } catch (err) {}
  setLoading(false);
};

const tree = buildTree(comments);
```

III. Khởi tạo routes

- App là layout chính chứa các phần cố định như Header, Footer, Outlet. Các children là các trang con sẽ được hiển thị trong <Outlet /> của App.

```
import { createBrowserRouter } from "react-router-dom";
import App from "../App";
import Home from "../pages/Home";
import ExplorePage from "../pages/ExplorePage";
import DetailsPage from "../pages/DetailsPage";
import SearchPage from "../pages/SearchPage";
import AuthPage from "../pages/AuthPage";
import FavouritePage from "../pages/FavouritePage";
import ProfilePage from "../pages/ProfilePage";
import ForgotPasswordPage from "../pages/ForgotPasswordPage";
import PersonPage from "../pages/PersonPage";

const router = createBrowserRouter([
  {
    path: "/",
    element: <App/>,
    children: [
```

- Giải thích từng route

Path	Component	Chức năng
/	<Home />	Trang chủ (hiển thị phim nổi bật, trending, v.v.)
/:explore	<ExplorePage />	Trang duyệt phim theo thể

		loại (vd: /movie, /tv)
/:explore/:id	<DetailsPage />	Trang chi tiết phim (vd: /movie/1234)
/search	<SearchPage />	Tìm kiếm phim theo từ khóa
/auth	<AuthPage />	Đăng nhập / Đăng ký
/favourite	<FavouritePage />	Phim đã lưu (yêu cầu đăng nhập)
/profile	<ProfilePage />	Thông tin người dùng (avatar, email...)
/forgot-password	<ForgotPasswordPage />	Quên mật khẩu
/person/:id	<PersonPage />	Thông tin diễn viên (tên, tiểu sử, phim đã đóng)