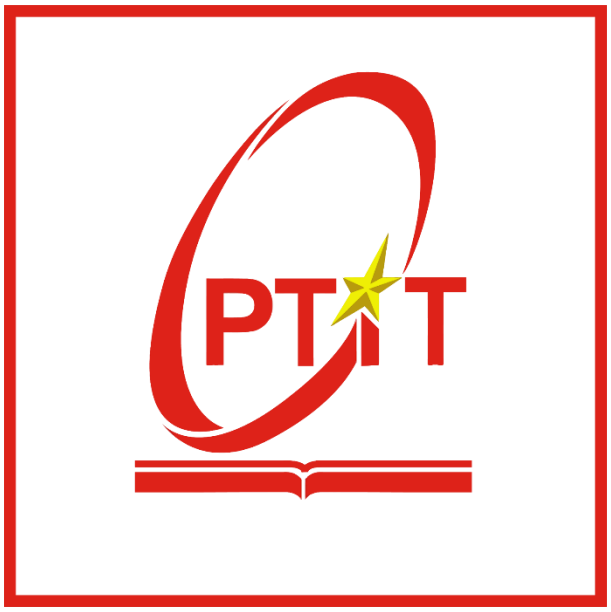
BỘ THÔNG TIN VÀ TRUYỀN THÔNG HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 12

Triển khai dự án (phần 5)

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Lê Quang Thanh – B22DCVT509

I. Các chức năng nổi bật (tiếp)

1. Chức năng thay đổi thông tin cá nhân

a) Frontend:

- Hiển thị form với thông tin hiện tại

```
const ProfilePage = () => {
  const navigate = useNavigate();  const { user, isAuthenticated, updateProfile }
  const [formData, setFormData] = useState({
    username: user?.username || '',
    email: user?.email || '',
    currentPassword: '',
    newPassword: '',
    confirmNewPassword: ''
  });
  const [loading, setLoading] = useState(false);
  const [message, setMessage] = useState({ type: '', content: '' });
```

- Validate input khi submit

```
if (formData.newPassword) {
  if (formData.newPassword.length < 6) {
    setMessage({ type: 'error', content: 'Mật khẩu mới phải có ít nhất 6 ký tự' });
    return;
  }
  if (formData.newPassword !== formData.confirmNewPassword) {
    setMessage({ type: 'error', content: 'Mật khẩu mới không khớp' });
    return;
  }
  if (formData.newPassword === formData.currentPassword) {
    setMessage({ type: 'error', content: 'Hãy đổi mật khẩu khác' });
    return;
  }
```

- Gọi updateProfile từ authContext

```
    setLoading(true);
    const result = await updateProfile({
      username: formData.username,
      currentPassword: formData.currentPassword,
      newPassword: formData.newPassword || undefined
    });
```

- Xử lý response và hiển thị thông báo

```
    if (result.success) {
      setMessage({ type: 'success', content: result.message });
      if (window.notify) window.notify('Đổi mật khẩu/thông tin thành công!');
```

- Điều hướng nếu thành công

b) Backend:

- Nhận request PUT với token

```
app.put('/api/user/profile', async (req, res) => {
  try {
```

- Xác thực token và tìm user

```
try {
  const token = req.header('x-auth-token');
  if (!token) {
    return res.status(401).json({ message: 'Không có token, quyền truy cập bị từ chối' });
  }

  const decoded = jwt.verify(token, process.env.JWT_SECRET || 'movieappsecret');
  let user = await User.findById(decoded.id);

  if (!user) {
    return res.status(404).json({ message: 'Không tìm thấy người dùng' });
  }

  const { username, currentPassword, newPassword } = req.body;

  // Kiểm tra xem username mới có bị trùng không (nếu có thay đổi)
  if (username !== user.username) {
    const existingUser = await User.findOne({ username });
    if (existingUser) {
      return res.status(400).json({ message: 'Tên người dùng đã tồn tại' });
    }
  }
```

- Kiểm tra username mới có bị trùng

```
// Kiểm tra xem username mới có bị trùng không (nếu có thay đổi)
if (username !== user.username) {
  const existingUser = await User.findOne({ username });
  if (existingUser) {
    return res.status(400).json({ message: 'Tên người dùng đã tồn tại' });
  }
}
```

- Nếu đổi mật khẩu:
  - Verify mật khẩu hiện tại

```
if (newPassword) {
  // Xác thực mật khẩu hiện tại
  const isMatch = await bcrypt.compare(currentPassword, user.password);
  if (!isMatch) {
    return res.status(400).json({ message: 'Mật khẩu hiện tại không đúng' });
  }
```

  - Mã hóa mật khẩu mới

```
  // Mã hóa mật khẩu mới
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(newPassword, salt);
  user.password = hashedPassword;
}
```

- Lưu thay đổi vào database

```
    // Cập nhật username
    user.username = username;

    // Lưu thay đổi
    await user.save();

    res.json({
      message: 'Cập nhật thông tin thành công',
      user: {
        id: user._id,
        username: user.username,
        email: user.email
      }
    });

  } catch (err) {
    console.error('Lỗi khi cập nhật profile:', err);
    res.status(500).json({ message: 'Lỗi server' });
  }
});
```

c. Bảo mật:

- Token xác thực cho mọi request
- Kiểm tra mật khẩu hiện tại
- Validate đầy đủ ở cả 2 phía
- Mật khẩu được hash trước khi lưu
- Email không được phép thay đổi

2. Chức năng chatbox với AI

a. Luồng hoạt động

1. Xác thực user.
2. Phân tích message: hỏi về diễn viên hay không.
3. Lấy dữ liệu phim phù hợp từ TMDb.
4. Tạo prompt và gửi cho Google Gemini.
5. Nhận câu trả lời AI và trả về frontend.

b. Giải thích

1. Xác thực và nhận message từ client

```
app.post('/api/chat', authMiddleware, async (req, res) => {
  try {
    const { message } = req.body;
    if (!message) {
      return res.status(400).json({ message: 'Message is required' });
    }
```

- Sử dụng middleware authMiddleware để kiểm tra token đăng nhập.
- Lấy message từ body request. Nếu không có, trả về lỗi 400.

2. Xử lý phân tích message: hỏi về diễn viên hay phim trending

```
// Phân tích message: nếu hỏi về diễn viên thì lấy phim theo diễn viên
const actorRegex = /diễn viên ([^\n\r\d.,!?]+)|phim của ([^\n\r\d.,!?]+)|([^\n\r\d.,!?]+) đóng phim gì/i;
let actorName = null;
const match = message.match(actorRegex);
if (match) {
  actorName = match[1] || match[2] || match[3];
  if (actorName) {
    actorName = actorName.trim();
    try {
```

- Tạo regex để phát hiện nếu user hỏi về diễn viên.
- Nếu có, lấy tên diễn viên từ message.

3. Nếu hỏi về diễn viên, lấy danh sách phim của diễn viên từ TMDb

```
try {
  // Tìm kiếm diễn viên trên TMDb
  const searchRes = await axios.get('https://api.themoviedb.org/3/search/person', {
    params: { api_key: process.env.TMDB_API_KEY, query: actorName, language: 'vi-VN' }
  });
  if (searchRes.data.results && searchRes.data.results.length > 0) {
    const person = searchRes.data.results[0];
    // Lấy danh sách phim của diễn viên
    const creditsRes = await axios.get(`https://api.themoviedb.org/3/person/${person.id}/movie_credits`, {
      params: { api_key: process.env.TMDB_API_KEY, language: 'vi-VN' }
    });
    const movies = creditsRes.data.cast ? creditsRes.data.cast.slice(0, 15) : [];
    movieList = movies.map(m => `${m.title} (${m.release_date || ''})`).join(', ');
    usedActor = true;
    prompt = `Dưới đây là danh sách phim của diễn viên ${actorName} trên TMDb: ${movieList}\n\nUser: ${message
  }
} catch (err) {
  console.error('TMDb actor search error:', err);
}
```

- Gọi TMDb API để tìm diễn viên.
- Lấy ID diễn viên, gọi tiếp API để lấy danh sách phim của diễn viên đó.
- Tạo prompt cho AI dựa trên danh sách phim vừa lấy.

4. Nếu không hỏi về diễn viên, lấy phim trending

```
// Nếu không phải hỏi về diễn viên hoặc không tìm được diễn viên, lấy phim trending như cũ
if (!usedActor) {
  try {
    const tmdbRes = await axios.get('https://api.themoviedb.org/3/trending/all/day?language=en-US', {
      params: { api_key: process.env.TMDB_API_KEY, language: 'vi-VN' }
    });
    const movies = tmdbRes.data.results.slice(0, 15);
    movieList = movies.map(m => `${m.title} (${m.release_date || ''})`).join(', ');
  } catch (tmdbErr) {
    console.error('TMDb error:', tmdbErr);
    movieList = '';
  }
  prompt = `Dưới đây là danh sách phim trending trong ngày trên TMDb: ${movieList}\n\nUser: ${message}\nAssistant:
}
```

- Nếu không phải hỏi về diễn viên, gọi TMDb API lấy danh sách phim trending.
- Tạo prompt cho AI dựa trên danh sách phim trending.

5. Gọi Google Gemini để sinh câu trả lời

```
const model = genAI.getGenerativeModel({ model: "gemini-2.5-flash-preview-05-20" });
const result = await model.generateContent(prompt);
const response = await result.response;
const text = response.text();
```

- Gửi prompt (có danh sách phim) cho Google Gemini.
- Lấy kết quả trả về từ AI.

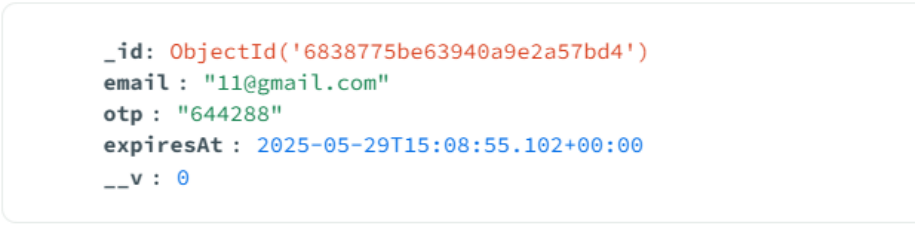
6. Trả kết quả về cho client

```
res.json({
  message: text,
  timestamp: new Date()
});
```

II. Các dữ liệu được lưu trữ trên MongoDB:

1. User
   - Thông tin tài khoản: username, email, password, createdAt
   - Danh sách phim yêu thích (wishlist): movieId, title, poster_path, vote_average, release_date, media_type

2. Otp
    - Lưu mã OTP xác thực email: email, otp, expiresAt



3. Comment
    - Bình luận phim: movieId, user (id, username), content, parentId (bình luận cha), createdAt, likes (danh sách user đã like)

```
_id: ObjectId('6839cdf4e6a73d0b1a28bdb8')
movieId : "1233413"
user : Object
content : "phim rất hay"
parentId : null
createdAt : 2025-05-30T15:25:40.714+00:00
__v : 3
likes : Array (1)
```

```
_id: ObjectId('6839d1599d46ca108182b12e')
movieId : "240"
user : Object
content : "phim quá tuyệt 10/10"
parentId : null
likes : Array (1)
createdAt : 2025-05-30T15:40:09.134+00:00
__v : 1
```

```
_id: ObjectId('6839d48ffc22642f956e24ae')
movieId : "1098006"
user : Object
content : "phim rất hay mong admin có thể cập nhật nhiều phim hay như thế này nữa"
parentId : null
likes : Array (1)
createdAt : 2025-05-30T15:53:51.070+00:00
__v : 1
```


4. Notification
    - Thông báo cho user: userId, message, createdAt, read


```
_id: ObjectId('6839d8aa343f4f291bc163a9')
userId : ObjectId('6837a0c6c8f33a98c2c122ee')
message : "Đã thích bình luận!"
read : false
createdAt : 2025-05-30T16:11:22.244+00:00
__v : 0
```

```
_id: ObjectId('6839d8bf343f4f291bc163b8')
userId : ObjectId('6837a0c6c8f33a98c2c122ee')
message : "Bình luận thành công!"
read : false
createdAt : 2025-05-30T16:11:43.854+00:00
__v : 0
```

```
_id: ObjectId('6839da8c3a3c69f1811115a6')
userId : ObjectId('6837a0c6c8f33a98c2c122ee')
message : "Đã xóa khỏi danh sách yêu thích!"
read : false
createdAt : 2025-05-30T16:19:24.178+00:00
__v : 0
```

```
_id: ObjectId('6839da8d3a3c69f1811115b7')
userId : ObjectId('6837a0c6c8f33a98c2c122ee')
message : "Đã xóa khỏi danh sách yêu thích!"
read : false
createdAt : 2025-05-30T16:19:25.557+00:00
```