**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**-----🙞🙜🕮🙞🙜-----**

**A green circle with white text and a white circle with a red star and a blue circle with a white circle with a red star and a white circle with a red star and a blue circle with

AI-generated content may be incorrect.**

**BÀI TẬP MÔN HỌC**

**LẬP TRÌNH TÍCH HỢP**

**Đề tài**

**ỨNG DỤNG XEM PHIM TRỰC TUYẾN ANIEFLIX**

**Giáo viên phụ trách: Đỗ Trung Dũng**

**Học viên thực hiện: Nguyễn Công Hải nam**

***Hà Nội, 6/2025***

**MỤC LỤC**

[1. GIỚI THIỆU TỔNG QUAN 2](#_Toc200534052)

[1.1. Lý do chọn đề tài 2](#_Toc200534053)

[1.2. Mục tiêu dự án 2](#_Toc200534054)

[1.3. Phạm vi dự án 3](#_Toc200534055)

[2. PHÂN TÍCH YÊU CẦU 3](#_Toc200534056)

[2.1. Yêu cầu chức năng 3](#_Toc200534057)

[2.2. Yêu cầu phi chức năng 5](#_Toc200534058)

[3. THIẾT KẾ HỆ THỐNG 6](#_Toc200534059)

[3.1. Kiến trúc tổng thể 6](#_Toc200534060)

[3.2. Database Design 7](#_Toc200534061)

[3.3. API Design 10](#_Toc200534062)

[3.4. Component Architecture (Frontend) 11](#_Toc200534063)

[4. CÔNG NGHỆ SỬ DỤNG 12](#_Toc200534064)

[4.1. Frontend Technologies 12](#_Toc200534065)

[4.2. Backend Technologies 14](#_Toc200534066)

[4.3. Development Tools 16](#_Toc200534067)

[5. TRIỂN KHAI HỆ THỐNG 16](#_Toc200534068)

[5.1. Setup môi trường phát triển 16](#_Toc200534069)

[5.2. Cấu trúc thư mục dự án 18](#_Toc200534070)

[5.3. Environment Configuration 19](#_Toc200534071)

[5.4. Authentication Implementation 19](#_Toc200534072)

[5.5. File Upload System 21](#_Toc200534073)

[5.6. Video Streaming Implementation 22](#_Toc200534074)

[6. TÍNH NĂNG CHÍNH 25](#_Toc200534075)

[6.1. Hệ thống Authentication 25](#_Toc200534076)

[6.2. Movie Management System 26](#_Toc200534077)

[6.3. Video Streaming System 27](#_Toc200534078)

[6.4. Comment System 28](#_Toc200534079)

[6.5. Favorites System 28](#_Toc200534080)

[6.6. Admin Dashboard 28](#_Toc200534081)

1. GIỚI THIỆU TỔNG QUAN

1.1. Lý do chọn đề tài

Trong bối cảnh công nghệ số phát triển mạnh mẽ, nhu cầu giải trí trực tuyến của người dùng ngày càng tăng cao. Các nền tảng streaming như Netflix, Disney+, Amazon Prime Video đã chứng minh được tiềm năng to lớn của thị trường này. Tại Việt Nam, mặc dù đã có một số nền tảng như FPT Play, VTVCab ON, nhưng vẫn còn nhiều cơ hội để phát triển một nền tảng streaming mới với trải nghiệm người dùng tốt hơn và nội dung phong phú hơn.

Anieflix được phát triển với mục tiêu tạo ra một nền tảng streaming hiện đại, tập trung vào nội dung điện ảnh châu Á, đặc biệt là phim anime, K-drama và J-drama, đáp ứng nhu cầu ngày càng tăng của khán giả trẻ Việt Nam đối với loại hình giải trí này.

1.2. Mục tiêu dự án

**Mục tiêu chính:**

* Xây dựng một hệ thống streaming phim hoàn chỉnh với giao diện người dùng hiện đại và thân thiện
* Cung cấp trải nghiệm xem phim mượt mà với chất lượng video cao
* Xây dựng hệ thống quản lý nội dung cho admin
* Tạo ra một nền tảng có khả năng mở rộng và bảo trì tốt

**Mục tiêu cụ thể:**

* Phát triển giao diện responsive hoạt động tốt trên mọi thiết bị
* Triển khai hệ thống authentication và authorization an toàn
* Xây dựng player video hỗ trợ nhiều định dạng và tính năng nâng cao
* Tích hợp hệ thống bình luận và tương tác người dùng
* Phát triển dashboard admin để quản lý nội dung
* Tối ưu hóa hiệu năng và trải nghiệm người dùng

1.3. Phạm vi dự án

Dự án Anieflix bao gồm các module chính:

1. Frontend Web Application: Giao diện người dùng chính
2. Backend API: Hệ thống xử lý logic và dữ liệu
3. Database: Lưu trữ thông tin phim, người dùng, bình luận
4. Admin Panel: Giao diện quản trị nội dung
5. Authentication System: Hệ thống xác thực và phân quyền
6. Video Streaming: Hệ thống phát video với nhiều chất lượng
7. Comment System: Hệ thống bình luận và tương tác

2. PHÂN TÍCH YÊU CẦU

2.1. Yêu cầu chức năng

**2.1.1. Người dùng cuối (End Users)**

**Đăng ký và Đăng nhập:**

* Đăng ký tài khoản với email và mật khẩu
* Xác thực email thông qua liên kết gửi về email
* Đăng nhập an toàn với JWT token
* Tính năng quên mật khẩu và đặt lại mật khẩu
* Quản lý thông tin cá nhân (tên, email, ngày sinh, số điện thoại)

**Duyệt và Tìm kiếm Nội dung:**

* Trang chủ hiển thị phim trending, mới nhất, được đánh giá cao
* Phân loại phim theo thể loại (Action, Drama, Comedy, etc.)
* Tìm kiếm phim theo tên, thể loại, năm phát hành
* Xem chi tiết phim (thông tin, trailer, diễn viên, đánh giá)
* Lọc và sắp xếp kết quả theo nhiều tiêu chí

**Xem Phim:**

* Player video với điều khiển đầy đủ (play, pause, volume, fullscreen)
* Tua nhanh, tua chậm, nhảy tới thời điểm cụ thể

**Quản lý Cá nhân:**

* Danh sách phim yêu thích
* Đánh giá và bình luận phim
* Thay đổi thông tin cá nhân và mật khẩu

**Tương tác Xã hội:**

* Bình luận trên từng phim
* Phản hồi bình luận của người khác
* Like/dislike bình luận
* Chia sẻ phim lên mạng xã hội

**2.1.2. Admin (Người quản trị)**

**Quản lý Nội dung:**

* Upload phim mới với thông tin đầy đủ (tên, mô tả, poster, video)
* Chỉnh sửa thông tin phim đã có
* Xóa phim không phù hợp
* Quản lý danh mục và thể loại phim
* Quản lý diễn viên và ekip sản xuất

**Quản lý Người dùng:**

* Xem danh sách tất cả người dùng
* Khóa/mở khóa tài khoản người dùng
* Xóa bình luận không phù hợp
* Quản lý quyền truy cập

**Thống kê và Báo cáo:**

* Dashboard tổng quan về hệ thống
* Thống kê số lượt xem theo phim, theo thời gian
* Báo cáo người dùng hoạt động
* Phân tích xu hướng và sở thích người xem

2.2. Yêu cầu phi chức năng

**2.2.1. Hiệu năng (Performance)**

* Thời gian tải trang < 3 giây
* Video bắt đầu phát trong vòng 5 giây
* Hỗ trợ đồng thời 10,000+ người dùng trực tuyến
* Tốc độ tìm kiếm < 500ms
* Optimized để hoạt động mượt mà trên thiết bị di động

**2.2.2. Bảo mật (Security)**

* Mã hóa mật khẩu người dùng bằng bcrypt
* Sử dụng JWT token cho authentication
* HTTPS cho tất cả connection
* Validation input để tránh SQL injection
* Rate limiting để tránh DDoS
* Backup dữ liệu định kỳ

**2.2.3. Khả năng mở rộng (Scalability)**

* Kiến trúc microservices có thể scale horizontal
* CDN để phân phối nội dung toàn cầu
* Database có thể sharding và replication
* Load balancer để phân tải
* Container hóa với Docker

**2.2.4. Khả năng sử dụng (Usability)**

* Giao diện responsive trên mọi thiết bị
* UX/UI thân thiện và trực quan
* Accessibility cho người khuyết tật
* Multi-language support
* Offline viewing (Progressive Web App)

**2.2.5. Tính sẵn sàng (Availability)**

* Uptime 99.9%
* Fault tolerance và error recovery
* Monitoring và alerting system
* Auto scaling dựa trên traffic
* Geographic redundancy

3. THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

Anieflix được thiết kế theo mô hình Client-Server với kiến trúc 3-tier:

**3.1.1. Presentation Layer (Frontend)**

* React.js Application: Single Page Application (SPA) được xây dựng bằng React
* Responsive Design: Sử dụng Tailwind CSS cho thiết kế responsive
* State Management: Context API và React Hooks
* Routing: React Router cho navigation
* HTTP Client: Axios cho API calls

**3.1.2. Application Layer (Backend)**

* Flask Web Framework: RESTful API server
* Authentication: JWT-based authentication
* File Upload: Multer cho xử lý file upload
* Image Processing: PIL/Pillow cho tối ưu hình ảnh
* Email Service: SMTP client cho email verification

**3.1.3. Data Layer (Database)**

* MySQL: Primary database cho structured data
* File System: Local storage cho media files
* Session Storage: Browser storage cho user sessions

3.2. Database Design

**3.2.1. Chi tiết các bảng**

Bảng USERS:

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

phone VARCHAR(20),

birthdate DATE,

is\_verified BOOLEAN DEFAULT FALSE,

is\_admin BOOLEAN DEFAULT FALSE,

verification\_token VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

Bảng MOVIES:

CREATE TABLE movies (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

original\_title VARCHAR(255),

overview TEXT,

release\_date DATE,

genre\_ids VARCHAR(255),

original\_language VARCHAR(10) DEFAULT 'vi',

vote\_average DECIMAL(3,1) DEFAULT 0,

vote\_count INT DEFAULT 0,

runtime INT DEFAULT 0,

poster\_path VARCHAR(255),

backdrop\_path VARCHAR(255),

video\_file VARCHAR(255),

tag ENUM('trending', 'top\_rated', 'upcoming', 'popular') DEFAULT 'trending',

cast JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

Bảng COMMENTS:

CREATE TABLE comments (

id INT PRIMARY KEY AUTO\_INCREMENT,

movie\_id INT NOT NULL,

user\_id INT,

parent\_id INT,

content TEXT NOT NULL,

anonymous\_name VARCHAR(100) DEFAULT 'Người dùng ẩn danh',

likes\_count INT DEFAULT 0,

dislikes\_count INT DEFAULT 0,

is\_deleted BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (movie\_id) REFERENCES movies(id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL,

FOREIGN KEY (parent\_id) REFERENCES comments(id) ON DELETE CASCADE

);

Bảng FAVORITES:

CREATE TABLE favorites (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

item\_id INT NOT NULL,

item\_type ENUM('movie', 'show') DEFAULT 'movie',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

UNIQUE KEY unique\_favorite (user\_id, item\_id, item\_type)

);

3.3. API Design

**3.3.1. Authentication APIs**

POST /api/register

POST /api/login

POST /api/verify?token={token}

POST /api/forgot-password

POST /api/reset-password

POST /api/resend-verification

**3.3.2. Movie APIs**

GET /api/movies

GET /api/movies/{id}

GET /api/movies/{id}/comments

POST /api/movies/{id}/comments

GET /api/search?q={query}

3.3.3. User Management APIs

GET /api/profile

PUT /api/profile

POST /api/change-password

GET /api/favorites

POST /api/favorites

DELETE /api/favorites/{id}

**3.3.4. Admin APIs**

POST /api/admin/upload-movie

GET /api/admin/movies

DELETE /api/admin/movies/{id}

GET /api/admin/stats

**3.3.5. Streaming APIs**

GET /api/stream/movie/{id}

GET /api/stream/show/{show\_id}/episode/{episode\_id}

GET /api/video-info/movie/{id}

3.4. Component Architecture (Frontend)

**3.4.1. Component Hierarchy**

App

├── Router

├── Layout

│ ├── Navbar

│ ├── SidebarAds

│ └── Footer

├── Pages

│ ├── Home

│ │ ├── Banner

│ │ ├── SlideRow

│ │ └── MoviePopup

│ ├── MovieDetail

│ │ ├── VideoPlayer

│ │ └── CommentSection

│ ├── Profile

│ ├── Favorites

│ └── Admin

│ ├── AdminUpload

│ └── AdminDashboard

└── Context

└── ThemeContext

**3.4.2. State Management**

* Global State: Theme, User Authentication
* Component State: Form data, UI states
* Server State: API responses, cached data
* Local Storage: User preferences, temporary data

4. CÔNG NGHỆ SỬ DỤNG

4.1. Frontend Technologies

**4.1.1. React.js (v18)**

Lý do chọn:

* Virtual DOM giúp tối ưu hiệu năng rendering
* Component-based architecture dễ maintain
* Ecosystem phong phú với nhiều thư viện hỗ trợ
* SEO-friendly với Server Side Rendering
* Large community và documentation đầy đủ

Các tính năng sử dụng:

* React Hooks (useState, useEffect, useContext)
* React Router cho client-side routing
* Error Boundaries cho error handling
* Lazy loading và code splitting
* Custom hooks cho logic reuse

**4.1.2. Tailwind CSS**

Lý do chọn:

* Utility-first approach giúp development nhanh chóng
* Highly customizable và responsive design
* Tree-shaking giúp giảm bundle size
* Consistent design system
* JIT compiler cho performance tối ưu

Cấu hình sử dụng:

module.exports = {

content: ['./src/\*\*/\*.{js,jsx}'],

theme: {

extend: {

colors: {

primary: '#1f2937',

secondary: '#4f46e5'

}

}

},

plugins: []

}

**4.1.3. Các thư viện hỗ trợ**

Lucide React (Icons):

* Bộ icon vectors nhẹ và đẹp mắt
* Tree-shakable, chỉ import icon cần thiết
* Consistent design language

HLS.js (Video Streaming):

* HTTP Live Streaming support
* Adaptive bitrate streaming
* Wide browser compatibility
* Custom player controls

Framer Motion (Animations):

* Smooth animations và transitions
* Gesture support
* Layout animations
* Performance optimized

4.2. Backend Technologies

**4.2.1. Flask (Python)**

Lý do chọn:

* Lightweight và flexible framework
* Easy to learn và rapid development
* RESTful API development-friendly
* Extensive ecosystem (Flask-CORS, Flask-JWT)
* Python ecosystem phong phú

Cấu trúc dự án:

backend/

├── app.py

├── routes/

│ ├── auth.py

│ ├── movies.py

│ ├── admin.py

│ └── streaming.py

├── utils/

│ ├── auth.py

│ └── email.py

├── static/

│ ├── posters/

│ ├── backdrops/

│ └── videos/

└── requirements.txt

**4.2.2. MySQL Database**

Lý do chọn:

* ACID compliance đảm bảo data integrity
* Proven reliability và performance
* Rich ecosystem và tooling
* Good documentation và community support
* Cost-effective cho scale vừa phải

Optimizations áp dụng:

* Indexing trên các cột frequently queried
* Connection pooling
* Query optimization
* Prepared statements để tránh SQL injection

**4.2.3. Additional Libraries**

PyMySQL: MySQL connector cho Python Werkzeug: WSGI utility library PIL/Pillow: Image processing python-dotenv: Environment variables management Flask-CORS: Cross-Origin Resource Sharing

4.3. Development Tools

**4.3.1. Version Control**

* Git: Distributed version control
* GitHub: Remote repository hosting
* Git Flow: Branching strategy

**4.3.3. Development Environment**

* VS Code: Primary code editor
* Node.js v18+: JavaScript runtime
* Python 3.9+: Backend runtime
* npm: Package manager
* pip: Python package manager

5. TRIỂN KHAI HỆ THỐNG

5.1. Setup môi trường phát triển

**5.1.1. Frontend Setup**

# Tạo React project

npx create-react-app anieflix-frontend

cd anieflix-frontend

# Cài đặt dependencies

npm install react-router-dom axios tailwindcss

npm install lucide-react framer-motion hls.js

# Cấu hình Tailwind CSS

npx tailwindcss init -p

**5.1.2. Backend Setup**

# Tạo Python virtual environment

python -m venv anieflix-backend

source anieflix-backend/bin/activate # Linux/Mac

# hoặc anieflix-backend\Scripts\activate # Windows

# Cài đặt dependencies

pip install flask flask-cors pymysql python-dotenv

pip install werkzeug pillow python-jwt

**5.1.3. Database Setup**

-- Tạo database

CREATE DATABASE anieflix CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- Tạo user và phân quyền

CREATE USER 'anieflix\_user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON anieflix.\* TO 'anieflix\_user'@'localhost';

FLUSH PRIVILEGES;

5.2. Cấu trúc thư mục dự án

anieflix/

├── frontend/

│ ├── public/

│ │ ├── index.html

│ │ └── images/

│ ├── src/

│ │ ├── components/

│ │ ├── pages/

│ │ ├── layouts/

│ │ ├── context/

│ │ ├── utils/

│ │ └── App.jsx

│ ├── package.json

│ └── tailwind.config.js

├── backend/

│ ├── routes/

│ ├── utils/

│ ├── static/

│ ├── app.py

│ ├── db.py

│ └── requirements.txt

├── database/

│ ├── schema.sql

│ └── sample\_data.sql

└── docs/

5.3. Environment Configuration

**5.3.1. Frontend (.env)**

REACT\_APP\_API\_URL=http://localhost:5000/api

REACT\_APP\_STATIC\_URL=http://localhost:5000/static

REACT\_APP\_VERSION=1.0.0

**5.3.2. Backend (.env)**

DB\_HOST=localhost

DB\_PORT=3306

DB\_USER=anieflix\_user

DB\_PASSWORD=password

DB\_NAME=anieflix

JWT\_SECRET=your-secret-key

EMAIL\_ADDRESS=noreply@anieflix.com

EMAIL\_PASSWORD=app-password

MOVIE\_VIDEO\_PATH=/path/to/video/movies

SHOW\_VIDEO\_PATH=/path/to/video/shows

5.4. Authentication Implementation

**5.4.1. JWT Token Generation**

import jwt

import datetime

from flask import current\_app

def generate\_token(user\_id):

payload = {

'user\_id': user\_id,

'exp': datetime.datetime.utcnow() + datetime.timedelta(days=1)

}

return jwt.encode(payload, current\_app.config['JWT\_SECRET'], algorithm='HS256')

def verify\_token(token):

try:

payload = jwt.decode(token, current\_app.config['JWT\_SECRET'], algorithms=['HS256'])

return payload['user\_id']

except jwt.ExpiredSignatureError:

return None

except jwt.InvalidTokenError:

return None

**5.4.2. Protected Route Middleware**

from functools import wraps

from flask import request, jsonify

def auth\_required(f):

@wraps(f)

def decorated\_function(\*args, \*\*kwargs):

token = request.headers.get('Authorization', '').replace('Bearer ', '')

user\_id = verify\_token(token)

if not user\_id:

return jsonify({'error': 'Unauthorized'}), 401

return f(user\_id, \*args, \*\*kwargs)

return decorated\_function

5.5. File Upload System

**5.5.1. Image Upload và Optimization**

from PIL import Image

import os

import uuid

def save\_and\_optimize\_image(file, folder, max\_size):

if not file:

return None

# Generate unique filename

filename = secure\_filename(file.filename)

name, ext = os.path.splitext(filename)

unique\_filename = f"{name}\_{uuid.uuid4().hex[:8]}{ext}"

# Save path

save\_path = os.path.join('static', folder)

os.makedirs(save\_path, exist\_ok=True)

file\_path = os.path.join(save\_path, unique\_filename)

# Save and optimize

file.save(file\_path)

with Image.open(file\_path) as img:

if img.mode == 'RGBA':

img = img.convert('RGB')

img.thumbnail(max\_size, Image.Resampling.LANCZOS)

img.save(file\_path, optimize=True, quality=85)

return unique\_filename

5.6. Video Streaming Implementation

**5.6.1. HLS Streaming Setup**

@stream\_bp.route('/stream/show/<int:show\_id>/episode/<int:episode\_id>')

def stream\_show\_episode\_hls(show\_id, episode\_id):

# Get episode info from database

episode = get\_episode\_by\_id(show\_id, episode\_id)

if not episode:

return jsonify({'error': 'Episode not found'}), 404

# Path to m3u8 file

m3u8\_path = os.path.join(SHOW\_VIDEO\_PATH, episode['filepath'])

if not os.path.exists(m3u8\_path):

return jsonify({'error': 'Video file not found'}), 404

# Read and modify playlist

with open(m3u8\_path, 'r', encoding='utf-8') as f:

playlist\_lines = []

for line in f.readlines():

line = line.strip()

if line and not line.startswith('#'):

# Convert relative path to full URL

full\_segment = f"/api/stream/show/{show\_id}/episode/{episode\_id}/{line}"

playlist\_lines.append(full\_segment)

else:

playlist\_lines.append(line)

playlist\_content = "\n".join(playlist\_lines)

response = Response(playlist\_content, mimetype='application/vnd.apple.mpegurl')

response.headers['Cache-Control'] = 'no-cache'

return response

**5.6.2. Frontend Video Player**

import Hls from 'hls.js'

import { useRef, useEffect } from 'react'

const VideoPlayer = ({ videoUrl, isHLS = false }) => {

const videoRef = useRef(null)

const hlsRef = useRef(null)

useEffect(() => {

const video = videoRef.current

if (!video) return

if (isHLS && videoUrl) {

if (Hls.isSupported()) {

hlsRef.current = new Hls({

debug: false,

enableWorker: true,

lowLatencyMode: true

})

hlsRef.current.loadSource(videoUrl)

hlsRef.current.attachMedia(video)

hlsRef.current.on(Hls.Events.MANIFEST\_PARSED, () => {

video.play().catch(e => console.log('Autoplay failed:', e))

})

}

} else {

video.src = videoUrl

}

return () => {

if (hlsRef.current) {

hlsRef.current.destroy()

}

}

}, [videoUrl, isHLS])

return (

<video

ref={videoRef}

className="w-full h-full object-contain"

controls

playsInline

/>

)

}

6. TÍNH NĂNG CHÍNH

6.1. Hệ thống Authentication

**6.1.1. User Registration Flow**

1. Đăng ký tài khoản: Người dùng điền form với email, mật khẩu, tên, số điện thoại, ngày sinh
2. Validation: Server kiểm tra email trùng lặp, validate format
3. Email verification: Gửi email chứa token xác thực
4. Account activation: Người dùng click link trong email để kích hoạt tài khoản
5. Login enabled: Tài khoản được kích hoạt, có thể đăng nhập

**6.1.2. Password Security**

* Hashing: Sử dụng bcrypt với salt rounds = 12
* Password policy: Tối thiểu 8 ký tự, bao gồm chữ hoa, chữ thường, số
* Forgot password: Email reset link với token có thời hạn 30 phút
* Password change: Yêu cầu mật khẩu cũ để đổi mật khẩu mới

**6.1.3. JWT Token Management**

6.2. Movie Management System

**6.2.1. Movie Upload (Admin)**

Admin có thể upload phim mới với đầy đủ thông tin:

Form Upload bao gồm:

* Tên phim (tiếng Việt và tên gốc)
* Mô tả chi tiết
* Ngày phát hành
* Thời lượng
* Thể loại
* Ngôn ngữ gốc
* Điểm đánh giá
* Poster image (bắt buộc)
* Backdrop image (tùy chọn)
* Video file (tùy chọn)
* Danh sách diễn viên

**Image Processing:**

* Sử dụng PIL để resize lại image

**6.2.2. Movie Display và Filtering**

**Homepage Layout:**

* Banner carousel với phim trending
* Slide rows theo danh mục (Hot, Top Rated, Upcoming)
* Responsive grid layout
* Lazy loading images

**Movie Detail Page:**

* Full movie information display
* Cast và crew information
* Related movies recommendations
* User ratings và reviews
* Streaming options

**Search và Filter:**

* Tự động tìm kiếm ngay khi người dung nhập chữ mà không cần Enter

6.3. Video Streaming System

**6.3.1. HLS (HTTP Live Streaming)**

**Lợi ích của HLS:**

* Tăng tốc độ truyền dữ liệu trong điều kiện mạng không ổn định
* Support cho live streaming nếu có update sau này
* Wide browser compatibility

**6.3.2. Custom Video Player**

**Features:**

* Play/pause, volume control
* Seeking và timeline
* Fullscreen mode
* Playback speed control
* Quality selection
* Subtitles support

6.4. Comment System

**6.4.1. Anonymous Comments**

**Hệ thống bình luận:**

**Features:**

* Anonymous commenting với tên hiển thị tùy chọn
* Nested replies (1 level)
* Like/dislike reactions
* Real-time comment updates

6.5. Favorites System

**Features:**

* Thêm/xóa phim yêu thích
* Hỗ trợ cả phim và show
* Truy cập nhanh từ User profile
* Đồng bộ giữa nhiều thiết bị

6.6. Admin Dashboard

**Dashboard Features:**

* Tổng số phim, người dùng, bình luận
* Biểu đồ phim theo danh mục
* Thống kê upload gần đây