**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO TIỂU LUẬN CUỐI KỲ**

**MÔN HỌC: PHÁT TRIỂN ỨNG DỤNG WEB NÂNG CAO**

**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG XEM PHIM CÙNG NHAU WATCHPARTY**

Giảng viên hướng dẫn: Ks. Nguyễn Trọng Hiếu

Sinh viên thực hiện: Nhóm 25

*Lâm Đồng, tháng 10 năm 2025*

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**LỜI CẢM ƠN**

Trong quá trình hoàn thành dự án này, em xin bày tỏ lòng biết ơn sâu sắc đến thầy Đoàn Minh Khuê – giảng viên hướng dẫn trong môn Mẫu Thiết Kế. Thầy đã tận tình hướng dẫn, cung cấp kiến thức và hỗ trợ em xuyên suốt quá trình thực hiện dự án. Những góp ý, nhận xét và khuyến nghị của thầy đã đóng vai trò quan trọng trong việc định hướng và nâng cao chất lượng của sản phẩm này.

Em cũng xin gửi lời cảm ơn chân thành đến các bạn sinh viên trong lớp đã nhiệt tình hỗ trợ và đóng góp ý kiến quý báu. Những cuộc thảo luận, phản hồi và sự chia sẻ kiến thức từ các bạn đã giúp em hoàn thiện hơn dự án của mình.

Cuối cùng, em xin chân thành cảm ơn tất cả những người đã trực tiếp hoặc gián tiếp đóng góp, hỗ trợ và tạo điều kiện thuận lợi để em có thể hoàn thành dự án này. Sự giúp đỡ và động viên của thầy và các bạn học chính là động lực để em cố gắng và hoàn thành tốt công việc được giao.

Trân trọng,

Nguyễn Thị Hoàng Phúc  
Hoàng Long  
Nguyễn Hoàng Sang

# MỤC LỤC

[MỤC LỤC 1](#_Toc212666596)

[DANH MỤC HÌNH ẢNH 1](#_Toc212666597)

[DANH MỤC BẢNG 2](#_Toc212666598)

[CHƯƠNG I. TỔNG QUAN 1](#_Toc212666599)

[**1.1. Giới thiệu** 1](#_Toc212666600)

[**1.2. Mục tiêu** 1](#_Toc212666601)

[**1.3. Kiến trúc hệ thống** 2](#_Toc212666602)

[**1.4. Công nghệ sử dụng** 7](#_Toc212666603)

[CHƯƠNG II. CƠ SỞ LÝ THUYẾT 10](#_Toc212666604)

[**2.1. Ứng dụng công nghệ web trong giải trí trực tuyến** 10](#_Toc212666605)

[**2.2. Hệ thống quản lý và vai trò trong nền tảng xem phim cùng nhau** 11](#_Toc212666606)

[**2.3. Nền tảng công nghệ sử dụng trong hệ thống Watch Party** 12](#_Toc212666607)

[*2.3.1. React.js – Giao diện người dùng (Frontend)* 12](#_Toc212666608)

[2.3.2. Node.js và Express.js – Xử lý nghiệp vụ (Backend) 13](#_Toc212666609)

[*2.3.3. Socket.io – Truyền thông thời gian thực (Real-time Communication)* 14](#_Toc212666610)

[*2.3.4. MongoDB và Mongoose – Lưu trữ và quản lý dữ liệu* 15](#_Toc212666611)

[*2.3.5. JWT và Passport.js – Xác thực và bảo mật* 16](#_Toc212666612)

[**2.4. API Chính** 16](#_Toc212666613)

[2.4.1. Nhóm API chính của hệ thống 17](#_Toc212666614)

[*2.4.2. Nhóm API phòng xem (Room API)* 17](#_Toc212666615)

[*2.4.3. Nhóm API đồng bộ video (Video Sync API)* 18](#_Toc212666616)

[*2.4.4. Nhóm API chat realtime (Chat API)* 18](#_Toc212666617)

[*2.4.5. Nhóm API quản lý thành viên* 18](#_Toc212666618)

[**2.5. Sự kiện Socket.io (Realtime Events)** 19](#_Toc212666619)

[*2.5.1. Room Events* 19](#_Toc212666620)

[*2.5.2. Video Synchronization Events* 19](#_Toc212666621)

[*2.5.3. Chat Events* 19](#_Toc212666622)

[**2.6. Đặc điểm kỹ thuật và bảo mật** 19](#_Toc212666623)

[CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 21](#_Toc212666624)

[**3.1. Luồng hoạt động của các lớp (tương tác giữa các tầng)** 21](#_Toc212666625)

[*3.1.1. Luồng xác thực người dùng* 22](#_Toc212666626)

[*3.1.2. Luồng tạo phòng* 24](#_Toc212666627)

[**3.2. Sơ đồ Usecases chính** 34](#_Toc212666628)

[*3.2.1. Mục tiêu* 34](#_Toc212666629)

[*3.2.2. Các tác nhân (Actors)* 34](#_Toc212666630)

[*3.2.3. Nhóm chức năng và các Usecase* 35](#_Toc212666631)

[**3.3. Sơ đồ cơ sở dữ liệu (ERD – Entity Relationship Diagram)** 40](#_Toc212666632)

[*3.3.1. Mục tiêu* 40](#_Toc212666633)

[*3.3.2. Các thực thể chính* 40](#_Toc212666634)

[*3.3.3. Collection: Users* 40](#_Toc212666635)

[*3.3.4. Collection: Rooms* 41](#_Toc212666636)

[*3.3.5. Collection: Messages* 41](#_Toc212666637)

[*3.3.6. Mối quan hệ giữa các collection* 42](#_Toc212666638)

[CHƯƠNG IV. PHÂN TÍCH ỨNG DỤNG 43](#_Toc212666639)

[**4.1. Giao diện** 43](#_Toc212666640)

[*4.1.1. Giao diện Dashboard* 43](#_Toc212666641)

[*4.1.2. Trang chủ (Home Page)* 45](#_Toc212666642)

[*4.1.3. Giao diện tạo phòng (Create Room)* 48](#_Toc212666643)

[*4.1.4. Giao diện tham gia phòng (Join Room)* 49](#_Toc212666644)

[4.1.5. Trang Đăng nhập (Login Page) 51](#_Toc212666645)

[4.1.6. Giao diện Trang Đăng ký (Register Page) 52](#_Toc212666646)

[4.1.7. Giao diện hồ sơ (Profile) 54](#_Toc212666647)

[**4.2. Bảo mật hệ thống** 55](#_Toc212666648)

[*4.2.1. Xác thực và Ủy quyền (Authentication & Authorization)* 55](#_Toc212666649)

[*4.2.2. Giao tiếp và Chính sách CORS (Communication & CORS Policy)* 56](#_Toc212666650)

[*4.2.3. Giới hạn tần suất và Phòng chống tấn công DoS (Rate Limiting & Anti-DoS)* 56](#_Toc212666651)

[*4.2.4. Xác thực và Bảo mật Socket.io (Socket Authentication & Security)* 56](#_Toc212666652)

[4.2.5. Kiểm tra đầu vào và Chống tiêm nhiễm (Input Validation & Injection Prevention) 57](#_Toc212666653)

[*4.2.6. Bảo vệ dữ liệu và quyền riêng tư (Data Protection & Privacy)* 57](#_Toc212666654)

[*4.2.7. Ghi log và Giám sát (Logging & Monitoring)* 58](#_Toc212666655)

[*4.2.8. Quản lý bí mật và cấu hình triển khai (Secret Management & Deployment Security)* 58](#_Toc212666656)

[**4.3. Hạn chế** 58](#_Toc212666657)

[**4.4. Đánh giá** 59](#_Toc212666658)

[CHƯƠNG V. TỔNG KẾT 60](#_Toc212666659)

[**5.1. Đánh giá ứng dụng** 60](#_Toc212666660)

[*5.1.1. Ưu điểm của ứng dụng* 60](#_Toc212666661)

[*5.1.2. Nhược điểm của ứng dụng* 61](#_Toc212666662)

[**5.2. Hướng phát triển của ứng dụng** 62](#_Toc212666663)

[**5.3. Kết luận** 65](#_Toc212666664)

[CHƯƠNG VI. TÀI LIỆU THAM KHẢO 66](#_Toc212666665)

# DANH MỤC HÌNH ẢNH

[Hình 1.1: Kiến trúc hệ thống 3](#_Toc212663501)

[Ảnh 3.1 : Sơ đồ tuần tự của hệ thống 20](#_Toc212663502)

[Bảng 3.2 : Luồng xác thực 21](#_Toc212663503)

[Hình 3.2: Luồng tạo phòng 26](#_Toc212663504)

[3.1.3. Luồng đồng bộ video 27](#_Toc212663505)

[Ảnh 3.2 : Luồng đồng bộ 29](#_Toc212663506)

[Ảnh 3.4 : Luồng chat realtime 32](#_Toc212663507)

[Ảnh 3.3.1: Sơ đồ Quản lý người dùng 35](#_Toc212663508)

[Ảnh 3.4.1: Sơ đồ quản lý phòng 36](#_Toc212663509)

[Ảnh 3.5.1: Sơ đồ xem video đồng bộ 36](#_Toc212663510)

[Ảnh 3.6.1: Sơ đồ chat realtime 37](#_Toc212663511)

[Ảnh 3.7.1: Sơ đồ quản lý thành viên 38](#_Toc212663512)

[Ảnh 3.3: Sơ đồ quan hệ giữa các collection 41](#_Toc212663513)

[Ảnh 4.1: Giao diện Dashboard 43](#_Toc212663514)

[Ảnh 4.2: Giao diện trang chủ 46](#_Toc212663515)

[Ảnh 4.3: Giao diện tạo phòng 48](#_Toc212663516)

[Ảnh 4.4: Giao diện tham gia phòng. 49](#_Toc212663517)

[Ảnh 4.5: Giao diện đăng nhập (Login Page) 51](#_Toc212663518)

[Ảnh 4.6: Giao diện đăng ký (Register Page) 52](#_Toc212663519)

[Ảnh 4.7: Giao diện hồ sơ (Profile) 54](#_Toc212663520)

# DANH MỤC BẢNG

[Bảng 2.1: React.js 12](#_Toc212663547)

[Bảng 2.2: Node.js và Express.js 13](#_Toc212663548)

[Bảng 2.3: Socket.io 14](#_Toc212663549)

[Bảng 2.4: MongoDB 15](#_Toc212663550)

[3.1.4. Luồng realtime 30](#_Toc212663551)

[Bảng 3.2: Bảng actors 34](#_Toc212663552)

[Bảng 3.3: Quản lý người dùng 34](#_Toc212663553)

[Bảng 3.4: Quản lý phòng 35](#_Toc212663554)

[Bảng 3.5: Xem video đồng bộ 36](#_Toc212663556)

[Bảng 3.6: Chat realtime 37](#_Toc212663558)

[Bảng 3.7 : Quản lý thành viên 38](#_Toc212663560)

# CHƯƠNG I. TỔNG QUAN

## **1.1. Giới thiệu**

Chương trình Hệ thống xem phim cùng nhau (Watch Party) là một giải pháp web cho phép nhiều người dùng xem video đồng bộ và trò chuyện trực tiếp trong cùng một không gian trực tuyến. Trong bối cảnh công nghệ và giải trí số phát triển mạnh mẽ, việc mang trải nghiệm “xem phim cùng bạn bè” lên môi trường trực tuyến trở thành nhu cầu ngày càng phổ biến, đặc biệt khi khoảng cách địa lý khiến các hoạt động giải trí chung trở nên khó thực hiện.

Tóm lại, việc xây dựng ứng dụng web xem phim cùng nhau không chỉ mang lại một giải pháp giải trí sáng tạo và gắn kết cộng đồng, mà còn góp phần minh chứng cho khả năng ứng dụng công nghệ thời gian thực trong việc phát triển các hệ thống web hiện đại và tương tác cao.

## **1.2. Mục tiêu**

Mục tiêu chính của hệ thống Watch Party là tạo ra một nền tảng xem phim trực tuyến đồng bộ, giúp nhiều người dùng có thể cùng xem, cùng tương tác và trò chuyện trong thời gian thực, từ đó mang lại trải nghiệm giải trí nhóm sống động và tự nhiên như khi xem phim trực tiếp cùng nhau.

Cụ thể, hệ thống hướng đến việc đồng bộ hóa trạng thái video giữa nhiều người dùng — khi một người phát, tạm dừng hoặc tua video, toàn bộ thành viên khác trong cùng phòng sẽ được cập nhật tức thì thông qua kết nối Socket.io. Đồng thời, hệ thống tích hợp chức năng chat realtime, cho phép người dùng trao đổi, gửi tin nhắn, biểu cảm (emoji) và phản hồi ngay trong quá trình xem phim.

Ngoài ra, hệ thống còn cung cấp các tính năng tạo và quản lý phòng xem phim, phân quyền điều khiển video, và xác thực người dùng an toàn thông qua JWT hoặc Google OAuth. Mọi dữ liệu người dùng, phòng xem và tin nhắn được lưu trữ và quản lý tập trung trong MongoDB, đảm bảo tính ổn định và hiệu quả khi truy xuất dữ liệu.

Việc áp dụng hệ thống này giúp người dùng tiết kiệm thời gian, đơn giản hóa việc kết nối, và nâng cao trải nghiệm giải trí nhóm trong môi trường trực tuyến. Bên cạnh giá trị thực tiễn, đề tài còn giúp người phát triển hiểu sâu hơn về kiến trúc ứng dụng realtime, giao tiếp socket, và bảo mật web, qua đó củng cố năng lực chuyên môn trong lĩnh vực phát triển ứng dụng web fullstack hiện đại.

## **1.3. Kiến trúc hệ thống**

Ứng dụng Watch Party – Xem phim cùng nhau trực tuyến được thiết kế theo kiến trúc 3 tầng (Three-Tier Architecture), kết hợp cùng mô hình hướng thành phần (Component-based Architecture) trên nền tảng Node.js và React.js.

Kiến trúc phân tầng giúp tách biệt rõ ràng các mối quan tâm (Separation of Concerns) trong hệ thống, đảm bảo mỗi tầng đảm nhiệm một vai trò cụ thể và có thể phát triển, bảo trì hoặc mở rộng độc lập mà không ảnh hưởng đến các tầng khác.

Cụ thể, hệ thống gồm ba tầng chính như sau:

*1. Tầng Presentation (Giao diện người dùng – Frontend)*

Tầng Presentation chịu trách nhiệm hiển thị giao diện và xử lý mọi tương tác của người dùng.

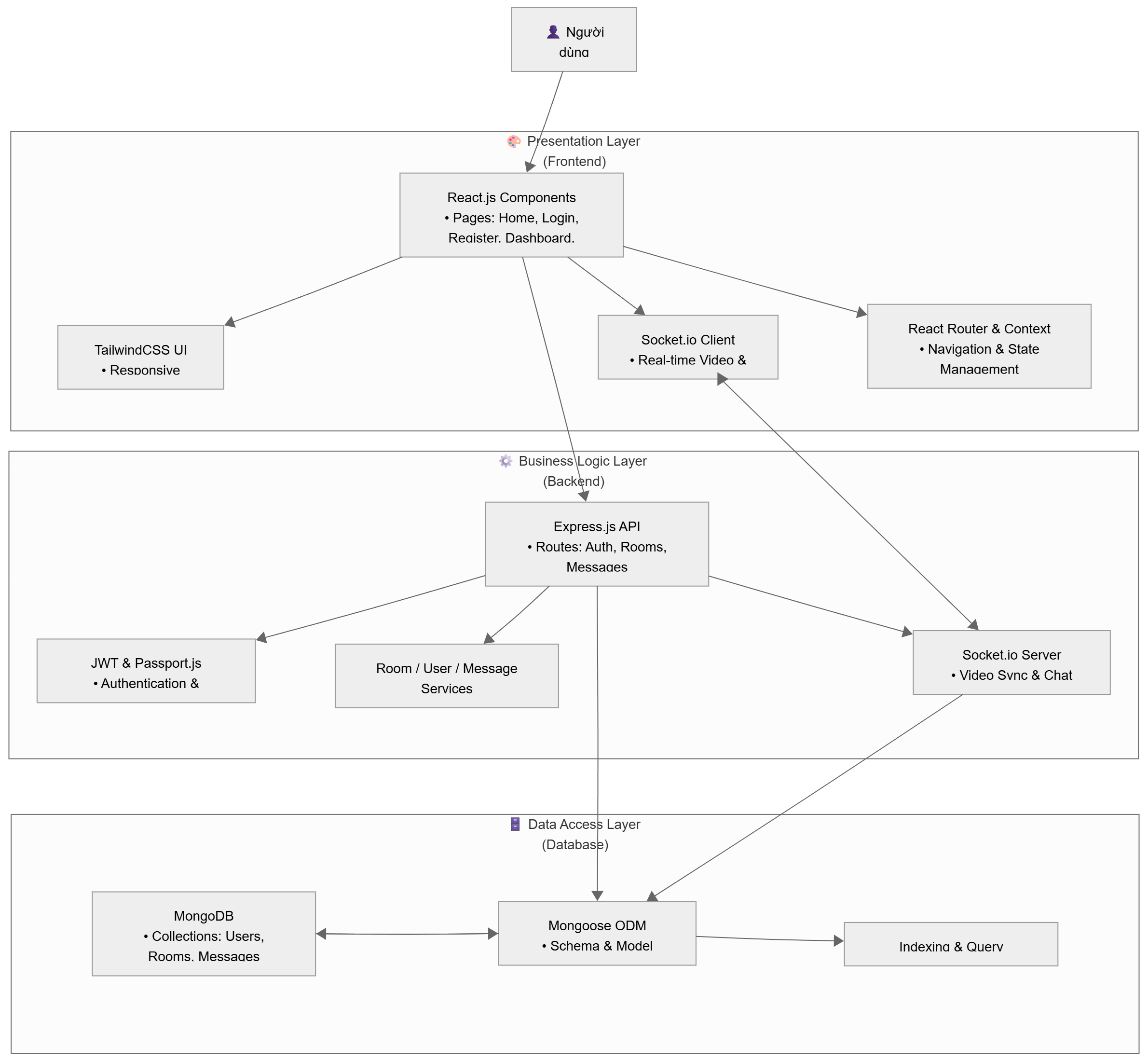
Hệ thống sử dụng React.js để xây dựng các component giao diện động, giúp tái sử dụng mã nguồn, dễ bảo trì và mở rộng.

Thành phần chính:

* React Components: Gồm các trang Home, Login, Register, Dashboard, Room.
* React Router & Context: Quản lý điều hướng và trạng thái người dùng toàn cục.
* TailwindCSS: Xây dựng giao diện hiện đại, tối giản và thân thiện với người dùng.
* Socket.io Client: Đảm nhiệm việc truyền thông thời gian thực (real-time) giữa người dùng và máy chủ, đảm bảo đồng bộ video, tin nhắn chat, và trạng thái phòng.

Frontend giao tiếp với backend thông qua HTTP Request (REST API) cho các tác vụ thông thường và Socket.io cho các sự kiện realtime.

Cách tiếp cận này giúp hệ thống vừa đảm bảo tốc độ truyền dữ liệu cao, vừa tách biệt rõ giữa giao diện và xử lý logic.



Hình 1.1: Kiến trúc hệ thống

*2. Tầng Business Logic (Xử lý nghiệp vụ – Backend))*

Tầng Business Logic chịu trách nhiệm xử lý toàn bộ luồng nghiệp vụ chính của hệ thống, bao gồm xác thực, quản lý phòng xem, đồng bộ video và tin nhắn chat.

Tầng này được xây dựng bằng Node.js kết hợp Express.js và Socket.io Server.Thành phần chính:

* Express.js API: Xử lý các yêu cầu RESTful từ frontend. Các route chính bao gồm:
* /api/auth – Đăng nhập, đăng ký, xác thực JWT.
* /api/rooms – Quản lý phòng xem phim.
* /api/messages – Quản lý tin nhắn chat.
* Socket.io Server: Quản lý kết nối thời gian thực giữa các người dùng, xử lý các sự kiện như: video-control, chat-message, join-room, user-typing.
* JWT & Passport.js: Xác thực và phân quyền người dùng an toàn (đăng nhập truyền thống hoặc Google OAuth).
* Room/User/Message Services: Các service riêng biệt chịu trách nhiệm xử lý nghiệp vụ từng phần (tạo phòng, gửi tin nhắn, cập nhật thành viên).

Cách tổ chức này tuân thủ nguyên tắc Single Responsibility – mỗi module chỉ đảm nhận một nhiệm vụ nhất định, giúp hệ thống dễ dàng mở rộng, kiểm thử và bảo trì.

*3. Tầng Data Access (Dữ liệu – Database Layer)*

Tầng Data Access quản lý việc lưu trữ, truy vấn và cập nhật dữ liệu của hệ thống.

Hệ thống sử dụng MongoDB làm cơ sở dữ liệu chính, kết hợp với Mongoose ODM để ánh xạ dữ liệu (Object Document Mapping) giữa ứng dụng và cơ sở dữ liệu.  
Thành phần chính:

* MongoDB: Lưu trữ các collection chính:
* Users – Thông tin tài khoản người dùng.
* Rooms – Thông tin phòng xem phim, thành viên và video đang xem.
* Messages – Lưu trữ nội dung tin nhắn và phản ứng (reactions).
* Mongoose ODM: Định nghĩa schema, kiểm tra dữ liệu và hỗ trợ thao tác CRUD dễ dàng.
* Indexing & Query Optimization: Tối ưu tốc độ truy vấn, đặc biệt cho các thao tác tìm phòng hoặc tải tin nhắn chat.

Việc tách riêng tầng dữ liệu giúp hệ thống dễ dàng thay thế hoặc mở rộng loại cơ sở dữ liệu (ví dụ: chuyển sang MongoDB Atlas, hoặc tích hợp Redis trong tương lai mà không cần thay đổi logic backend).

**1.3. Chức năng chính**

Ứng dụng Watch Party cung cấp các chức năng chính nhằm mang lại trải nghiệm xem phim đồng bộ và tương tác thời gian thực giữa nhiều người dùng.

Các chức năng được thiết kế hướng tới sự tiện lợi, mượt mà và kết nối cộng đồng, giúp người dùng có thể cùng xem, cùng trò chuyện và tương tác trực tuyến một cách tự nhiên, giống như đang xem phim trực tiếp với nhau.  
**Chức năng quản lý người dùng (User Management):**

Đây là chức năng nền tảng của hệ thống, cho phép người dùng đăng ký, đăng nhập, đăng xuất, và quản lý thông tin cá nhân.

Các phương thức xác thực bao gồm:

* Đăng nhập truyền thống bằng email và mật khẩu,
* Hoặc đăng nhập nhanh qua Google OAuth.

Người dùng có thể chỉnh sửa hồ sơ cá nhân, và theo dõi các hoạt động đã tham gia (phòng xem, lịch sử chat,...).

Hệ thống sử dụng JWT (JSON Web Token) để xác thực phiên đăng nhập, đảm bảo an toàn và dễ mở rộng.

→ Nhờ đó, dữ liệu người dùng được bảo mật, nhất quán và có thể sử dụng xuyên suốt các module khác như tạo phòng, chat và đồng bộ video.Chức năng chấm công (Attendance Management)

**Chức năng quản lý phòng xem phim (Room Management)**

* Chức năng này cho phép người dùng tạo, tham gia và quản lý các phòng xem phim.
* Mỗi phòng có thể được cấu hình tùy chọn như:
* Đặt mật khẩu hoặc để công khai,
* Chia sẻ mã phòng (Room Code) để mời người khác tham gia,
* Gán quyền Host (chủ phòng) để điều khiển video.

Các chức năng chính:

Tạo phòng với URL video từ YouTube hoặc nguồn khác.

Tự động sinh mã phòng ngẫu nhiên (6 ký tự).

Quản lý danh sách thành viên trong phòng.

→ Nhờ đó, người dùng có thể dễ dàng tổ chức buổi xem phim nhóm trực tuyến, riêng tư hoặc công khai.

**Chức năng xem phim đồng bộ (Video Synchronization)**

Đây là chức năng cốt lõi nhất của hệ thống, được triển khai bằng Socket.io để đảm bảo các thao tác video được đồng bộ hóa tức thời giữa tất cả thành viên trong phòng.

Các thao tác bao gồm:

* Play/Pause: Khi một người nhấn phát hoặc tạm dừng, toàn bộ thành viên khác được đồng bộ theo.
* Seek: Khi tua video, hệ thống gửi thời điểm mới đến tất cả thành viên để cập nhật đồng bộ.

- Auto-Sync: Khi người dùng mới vào phòng, trình phát video sẽ tự động đồng bộ với trạng thái hiện tại của phòng.

→ Cơ chế này giúp mọi người có cùng trải nghiệm xem phim một cách liền mạch, không bị trễ hay sai lệch thời gian phát.

**Chức năng chat thời gian thực (Realtime Chat)**

Chức năng này cho phép người dùng trò chuyện trực tiếp trong khi xem phim, giúp tăng tính tương tác và kết nối cảm xúc.

Các tính năng cụ thể:

* Gửi và nhận tin nhắn theo thời gian thực.
* Hiển thị trạng thái đang gõ (typing indicator).
* Thêm reaction (emoji) cho tin nhắn.
* Đánh dấu tin nhắn đã xem (message seen).
* Gửi tin nhắn hệ thống khi có người vào/ra phòng.

Tất cả được triển khai qua Socket.io events như chat-message, user-typing, message-seen, đảm bảo phản hồi nhanh và chính xác.

→ Nhờ đó, người dùng có thể tương tác sinh động và vui vẻ trong quá trình xem phim.

**Chức năng báo cáo và thống kê (Reports & Analytics)**

Hệ thống dự kiến mở rộng thêm các tính năng báo cáo như:

* Thống kê số lượng phòng đang hoạt động.
* Tổng số người dùng online và mức độ tương tác.
* Lịch sử chat, video, và hành vi người dùng.

## **1.4. Công nghệ sử dụng**

Ứng dụng Watch Party được phát triển trên nền tảng Node.js (Backend) và React.js (Frontend), kết hợp với MongoDB làm cơ sở dữ liệu chính. Dưới đây là các công nghệ và công cụ được sử dụng trong dự án:

* React.js
* React được sử dụng cho giao diện người dùng (Frontend UI).
* Hỗ trợ kiến trúc component-based, giúp tái sử dụng giao diện, dễ mở rộng và bảo trì.
* Sử dụng React Router để điều hướng trang và React Context để quản lý trạng thái toàn cục.
* Kết hợp TailwindCSS để tạo giao diện nhanh, thống nhất, tương thích trên mọi thiết bị.
* Các trang chính bao gồm: Home, Login, Register, Dashboard, Room Watch.

→ React mang lại trải nghiệm người dùng mượt mà, phản hồi nhanh và tối ưu hiệu năng hiển thị.

* Node.js và Express.js

Node.js là nền tảng phía server của ứng dụng, sử dụng JavaScript làm ngôn ngữ lập trình chính. Nền tảng chạy server, xử lý logic và giao tiếp với cơ sở dữ liệu.

Framework Express.js được dùng để xây dựng RESTful API, chịu trách nhiệm xử lý các yêu cầu từ frontend, điều hướng request, và phản hồi dữ liệu JSON về cho React.

Tầng backend được tổ chức theo mô hình Controller – Service – Model, giúp phân tách rõ logic nghiệp vụ, truy cập dữ liệu, và định tuyến API.

Express.js được dùng để xây dựng RESTful API, định tuyến các yêu cầu HTTP (đăng ký, đăng nhập, tạo phòng, gửi tin nhắn,…).

Cấu trúc backend theo mô hình Controller – Service – Model, giúp tách biệt rõ giữa xử lý logic, nghiệp vụ, và dữ liệu.

Tích hợp Socket.io để hỗ trợ realtime chat và đồng bộ video.

→ Node.js + Express.js là nền tảng vững chắc cho các ứng dụng realtime hiện đại.

* Socket.io

- Là công nghệ trung tâm giúp ứng dụng hoạt động thời gian thực (real-time).

- Hỗ trợ kết nối hai chiều giữa server và client qua WebSocket protocol.

- Được dùng để xử lý các sự kiện như: *video-control, chat-message, user-joined, message-seen.*

→ Socket.io đảm bảo độ trễ thấp và trải nghiệm đồng bộ chính xác giữa các thành viên trong phòng.

* MongoDB và Mongoose
* MongoDB là cơ sở dữ liệu NoSQL được dùng để lưu trữ thông tin người dùng, phòng xem, và tin nhắn.
* Mongoose ODM giúp ánh xạ schema, định nghĩa model và truy vấn dữ liệu dễ dàng.
* Các collection chính:

Users – thông tin người dùng

Rooms – dữ liệu phòng và thành viên

Messages – lưu trữ nội dung chat và phản ứng

→ MongoDB cho phép mở rộng quy mô linh hoạt và tốc độ truy cập cao, phù hợp với hệ thống realtime.

* TailwindCSS
* TailwindCSS được dùng để tạo phong cách giao diện nhanh chóng, gọn gàng và nhất quán.
* Nhờ sử dụng các utility class, lập trình viên có thể thiết kế bố cục và màu sắc giao diện một cách hiệu quả mà không cần viết CSS thủ công.
* Kết hợp với React, Tailwind giúp tối ưu tốc độ phát triển UI và đảm bảo tính trực quan trên nhiều thiết bị.
* Các thư viện và công cụ khác:

Ngoài các công nghệ chủ đạo, dự án còn sử dụng:

Axios: để gọi REST API từ frontend đến backend.

dotenv: để quản lý cấu hình và biến môi trường (như thông tin kết nối database).

Nodemon: hỗ trợ tự động reload server khi code thay đổi trong quá trình phát triển.

Git & GitHub: để quản lý phiên bản mã nguồn và cộng tác nhóm.

Visual Studio Code: IDE chính để phát triển, debug, và quản lý dự án.

Postman: dùng kiểm thử API trong quá trình phát triển backend.

# CHƯƠNG II. CƠ SỞ LÝ THUYẾT

## **2.1. Ứng dụng công nghệ web trong giải trí trực tuyến**

Giải trí trực tuyến (Online Entertainment) là hình thức cung cấp và tiêu thụ nội dung giải trí như phim ảnh, âm nhạc, trò chơi, hoặc truyền hình qua các nền tảng trực tuyến. Với sự phát triển mạnh mẽ của **Internet và công nghệ web**, người dùng ngày nay có thể dễ dàng tiếp cận kho nội dung phong phú mọi lúc, mọi nơi mà không cần phụ thuộc vào các phương tiện truyền thống như đĩa DVD hay truyền hình cáp.

Không chỉ dừng lại ở việc truyền tải nội dung, ứng dụng công nghệ web trong giải trí còn mang đến những mô hình tương tác mới, giúp người dùng không chỉ xem mà còn tham gia, giao tiếp và chia sẻ trải nghiệm với người khác. Trong đó, các nền tảng xem phim trực tuyến đồng bộ – hay còn gọi là Watch Party – đang trở thành một xu hướng mới, đặc biệt phổ biến trong bối cảnh toàn cầu hóa và kết nối từ xa.

Các nền tảng này cho phép người dùng xem cùng một bộ phim hoặc video theo thời gian thực, dù họ ở các vị trí địa lý khác nhau. Tính năng đồng bộ video và chat realtime giúp tái hiện lại cảm giác “xem phim cùng nhau” như ngoài đời thực, đồng thời tạo nên không gian tương tác xã hội trong môi trường trực tuyến.

Bên cạnh đó, các công nghệ như WebSocket, Socket.io, và React.js đóng vai trò nền tảng trong việc hiện thực hóa mô hình này. Chúng cho phép truyền dữ liệu tức thời giữa người dùng và máy chủ, đảm bảo độ trễ thấp và trải nghiệm xem liền mạch. Ngoài ra, hệ thống xác thực bằng JWT và Google OAuth giúp người dùng đăng nhập an toàn, đồng thời quản lý danh tính và quyền truy cập hiệu quả.

Có thể nói, ứng dụng công nghệ web trong lĩnh vực giải trí không chỉ làm thay đổi cách người dùng thưởng thức nội dung, mà còn tạo ra một xu hướng mới về kết nối cộng đồng trong không gian số. Watch Party chính là minh chứng cho sự chuyển mình này – một hình thức giải trí trực tuyến mang tính tương tác cao, kết hợp giữa công nghệ thời gian thực và trải nghiệm xã hội hiện đại.

## **2.2. Hệ thống quản lý và vai trò trong nền tảng xem phim cùng nhau**

Trong một nền tảng giải trí trực tuyến như Watch Party, hệ thống quản lý đóng vai trò trung tâm điều phối toàn bộ hoạt động – từ quản lý người dùng, phòng xem, video, cho đến xử lý chat và đồng bộ dữ liệu. Một hệ thống được thiết kế hợp lý không chỉ giúp ứng dụng vận hành trơn tru mà còn đảm bảo hiệu suất, bảo mật và tính ổn định cao trong môi trường nhiều người dùng.

Về bản chất, hệ thống quản lý trong Watch Party có thể được xem như “xương sống” của toàn bộ nền tảng, chịu trách nhiệm kết nối giữa Frontend (giao diện người dùng), Backend (xử lý nghiệp vụ) và Database (dữ liệu). Các thành phần này phối hợp chặt chẽ thông qua mô hình kiến trúc 3 tầng (Three-Tier Architecture):

* Frontend: hiển thị giao diện xem phim, phòng chat và thông tin thành viên.
* Backend: xử lý các yêu cầu như tạo phòng, điều khiển video, gửi tin nhắn, xác thực người dùng.
* Database: lưu trữ thông tin người dùng, phòng xem, và lịch sử tin nhắn.

Một hệ thống quản lý hiệu quả giúp đồng bộ hóa mọi thao tác giữa nhiều người dùng trong thời gian thực, đảm bảo rằng khi một người dùng thực hiện hành động (phát, dừng, tua video), toàn bộ thành viên khác đều nhận được cập nhật tức thì. Đồng thời, thông qua việc tích hợp cơ chế phân quyền (host và member), hệ thống duy trì được sự kiểm soát và tính ổn định của từng phòng xem.

Từ góc độ vận hành, hệ thống quản lý còn giúp theo dõi trạng thái hoạt động, ghi nhận sự kiện (log), và hỗ trợ phân tích hành vi người dùng để tối ưu hiệu năng trong tương lai. Ở mức độ mở rộng, kiến trúc hiện tại có thể được triển khai trên các nền tảng điện toán đám mây như MongoDB Atlas và Docker, giúp hệ thống mở rộng quy mô mà không ảnh hưởng đến hiệu suất.

Có thể khẳng định rằng, trong bối cảnh các ứng dụng giải trí trực tuyến ngày càng chú trọng vào tính tương tác và trải nghiệm người dùng, việc xây dựng một hệ thống quản lý linh hoạt, ổn định và bảo mật như Watch Party là yếu tố tiên quyết. Đây không chỉ là một sản phẩm phần mềm, mà còn là một mô hình kết nối cộng đồng số, góp phần định hình xu hướng giải trí mới trong kỷ nguyên công nghệ 4.0.

## **2.3. Nền tảng công nghệ sử dụng trong hệ thống Watch Party**

Để đáp ứng yêu cầu về truyền thông thời gian thực, đồng bộ dữ liệu, và tính ổn định khi có nhiều người dùng truy cập cùng lúc, hệ thống Watch Party được xây dựng dựa trên các công nghệ web hiện đại. Mỗi công nghệ đảm nhận một vai trò riêng trong kiến trúc tổng thể, góp phần tạo nên một nền tảng xem phim trực tuyến đồng bộ, thân thiện và bảo mật.

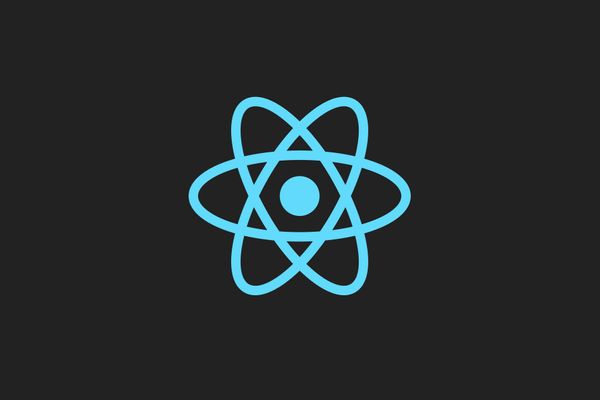
### *2.3.1.* *React.js – Giao diện người dùng (Frontend)*

React.js là thư viện JavaScript do Facebook phát triển, được sử dụng để xây dựng giao diện người dùng động (Dynamic User Interface).

Trong hệ thống Watch Party, React đóng vai trò trung tâm của tầng Presentation Layer, chịu trách nhiệm hiển thị giao diện, xử lý tương tác và cập nhật trạng thái theo thời gian thực.

Các đặc điểm nổi bật của React trong hệ thống:

* Kiến trúc component-based: cho phép chia nhỏ giao diện thành các thành phần độc lập (Navbar, ChatBox, VideoPlayer, RoomList,...), dễ tái sử dụng và bảo trì.
* Virtual DOM: giúp cập nhật giao diện nhanh và mượt, đảm bảo trải nghiệm liền mạch khi người dùng thao tác.
* React Context & Hooks: được sử dụng để quản lý trạng thái người dùng, socket và phòng xem xuyên suốt toàn bộ ứng dụng.
* TailwindCSS: được tích hợp cùng React nhằm tối ưu thiết kế UI/UX, mang lại bố cục hiện đại, nhất quán và phản hồi nhanh trên mọi thiết bị.



Bảng 2.1: React.js

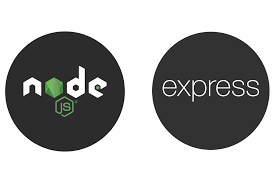
### 2.3.2. Node.js và Express.js – Xử lý nghiệp vụ (Backend)

Node.js là môi trường chạy JavaScript phía server, cho phép xây dựng ứng dụng web hiệu năng cao nhờ kiến trúc event-driven và non-blocking I/O.

Express.js, framework phổ biến của Node.js, được sử dụng để phát triển RESTful API cho hệ thống Watch Party.

* Các vai trò chính:
* Xử lý yêu cầu (request) từ frontend: như đăng nhập, đăng ký, tạo phòng, gửi tin nhắn,…
* Định tuyến API: qua các module /api/auth, /api/rooms, /api/messages.
* Quản lý socket server: phối hợp cùng Socket.io để điều phối các sự kiện realtime.
* Middleware: sử dụng cho xác thực JWT, kiểm tra quyền truy cập và xử lý lỗi.

Express giúp backend của Watch Party có cấu trúc rõ ràng, dễ mở rộng và dễ kiểm thử, đồng thời tương thích tốt với MongoDB và Socket.io.



Bảng 2.2: Node.js và Express.js

### *2.3.3. Socket.io – Truyền thông thời gian thực (Real-time Communication)*

Socket.io là thư viện hỗ trợ kết nối WebSocket giữa client và server, cho phép trao đổi dữ liệu hai chiều trong thời gian thực. Đây là công nghệ cốt lõi giúp Watch Party thực hiện đồng bộ video và chat giữa nhiều người dùng.

Các ứng dụng chính của Socket.io trong hệ thống:

Đồng bộ trạng thái video: khi host phát, tạm dừng hoặc tua video, toàn bộ thành viên khác sẽ được cập nhật ngay lập tức.

Trò chuyện realtime: truyền tải tin nhắn, reaction, và trạng thái “đang gõ” giữa các người dùng.

Thông báo phòng: thông báo khi người dùng tham gia, rời phòng hoặc khi host thay đổi quyền điều khiển.

Sự kiện tự động: như join-room, video-control, chat-message, message-seen.

Socket.io kết hợp cùng Express.js tạo thành hệ thống xử lý đồng thời (concurrent system), đảm bảo độ trễ thấp và phản hồi nhanh, đáp ứng nhu cầu trải nghiệm đồng bộ của nhiều người dùng cùng lúc.

Bảng 2.3: Socket.io

### *2.3.4. MongoDB và Mongoose – Lưu trữ và quản lý dữ liệu*

MongoDB là cơ sở dữ liệu NoSQL dạng tài liệu (document-oriented), phù hợp với ứng dụng có dữ liệu động và quy mô mở rộng.

Hệ thống Watch Party sử dụng Mongoose – một ODM (Object Data Modeling) – để kết nối và thao tác với cơ sở dữ liệu MongoDB.

Các thành phần dữ liệu chính:

* Users: lưu thông tin tài khoản, phương thức đăng nhập, avatar, thời gian tạo, v.v.
* Rooms: lưu thông tin về mã phòng, video đang xem, danh sách thành viên và quyền host.
* Messages: lưu tin nhắn chat, biểu cảm, người gửi và thời gian gửi.

Mongoose giúp định nghĩa rõ ràng cấu trúc dữ liệu thông qua Schema, đồng thời hỗ trợ các tính năng như validation, indexing, và query optimization, giúp tăng tốc độ truy vấn và đảm bảo tính toàn vẹn dữ liệu.

Bảng 2.4: MongoDB

### *2.3.5. JWT và Passport.js – Xác thực và bảo mật*

JWT (JSON Web Token) là chuẩn xác thực phổ biến, được sử dụng trong Watch Party để quản lý phiên đăng nhập người dùng. Khi người dùng đăng nhập thành công, server sẽ cấp cho họ một token chứa thông tin mã hóa, cho phép truy cập các API bảo mật mà không cần gửi lại mật khẩu.

Passport.js được sử dụng để mở rộng cơ chế đăng nhập, đặc biệt là Google OAuth 2.0, cho phép người dùng đăng nhập nhanh bằng tài khoản Google cá nhân.

Các biện pháp bảo mật khác:

* bcrypt: mã hóa mật khẩu người dùng.
* Helmet & CORS: bảo vệ ứng dụng khỏi các cuộc tấn công phổ biến như XSS, CSRF, và hạn chế truy cập không hợp lệ.
* Rate Limiting: giới hạn số lượng request để tránh tấn công brute-force.

**2.4. API Chính**

Các API chính (Application Programming Interface) là cầu nối giữa frontend (React.js) và backend (Node.js/Express), cho phép các thành phần trong hệ thống Watch Party trao đổi dữ liệu thông qua giao thức HTTP RESTful hoặc WebSocket (Socket.io).

Nhờ sự kết hợp này, hệ thống đảm bảo bảo mật, minh bạch và an toàn cho người dùng trong suốt quá trình sử dụng.

Hệ thống được thiết kế theo nguyên tắc RESTful API, giúp dễ mở rộng, tái sử dụng và tích hợp với các client hoặc dịch vụ khác.

Toàn bộ API đều tuân theo cấu trúc chuẩn:

BASE\_URL/api/v1/<resource>

Tất cả các endpoint yêu cầu xác thực đều sử dụng JWT (JSON Web Token) để bảo đảm an toàn dữ liệu và quản lý phiên người dùng.

### 2.4.1. Nhóm API chính của hệ thống

Các API trong Watch Party được chia thành 5 nhóm chính, tương ứng với các mô-đun chức năng cốt lõi của hệ thống:

|  |  |  |  |
| --- | --- | --- | --- |
| **Phương thức** | **Endpoint** | **Quyền truy cập** | **Mô tả** |
| GET | /health | Public | Kiểm tra trạng thái hoạt động của server (dùng cho monitoring hoặc CI/CD). |

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** |
| POST | /api/v1/auth/register | Đăng ký tài khoản mới (UC001). |
| POST | /api/v1/auth/login | Đăng nhập bằng email hoặc username (UC002). |
| GET | /api/v1/auth/google | Đăng nhập bằng Google OAuth (UC003). |
| POST | /api/v1/auth/logout | Đăng xuất, hủy token hiện tại (UC005). |
| GET | /api/v1/users/profile | Lấy thông tin profile người dùng (UC004). |
| PUT | /api/v1/users/profile | Cập nhật thông tin cá nhân (tên, ảnh đại diện). |

Các API trong nhóm này xử lý xác thực bằng JWT, token được gửi kèm trong Header:

Authorization: Bearer <token>.

Hệ thống có thể mở rộng để tích hợp thêm OAuth2.0 cho Google, GitHub hoặc Facebook.

### *2.4.2. Nhóm API phòng xem (Room API)*

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** |
| POST | /api/v1/rooms | Tạo phòng mới với URL video (UC006). |
| GET | /api/v1/rooms | Lấy danh sách tất cả phòng công khai (UC010). |
| GET | /api/v1/rooms/:id | Xem thông tin chi tiết phòng (UC011, UC012). |
| PUT | /api/v1/rooms/:id | Cập nhật thông tin phòng (UC013, chỉ host). |
| DELETE | /api/v1/rooms/:id | Xóa phòng (UC014, chỉ host). |
| POST | /api/v1/rooms/join | Tham gia phòng bằng mã phòng hoặc mật khẩu (UC007, UC008). |
| POST | /api/v1/rooms/leave | Rời khỏi phòng (UC009). |

Dữ liệu trả về bao gồm thông tin: roomId, roomName, hostId, videoURL, tags, members[], createdAt.

Các API có kiểm tra quyền truy cập: chỉ host mới được chỉnh sửa hoặc xóa phòng.

### *2.4.3. Nhóm API đồng bộ video (Video Sync API)*

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** |
| POST | /api/v1/video/play | Phát video cho toàn bộ thành viên trong phòng (UC015). |
| POST | /api/v1/video/pause | Tạm dừng video (UC016). |
| POST | /api/v1/video/seek | Chuyển đến thời điểm cụ thể (UC017). |
| GET | /api/v1/video/state | Gửi hoặc nhận trạng thái video hiện tại (UC018, UC019). |

**Socket event song song:**  
Các API này được hỗ trợ bởi Socket.io events để truyền thông realtime, ví dụ:

* video-play
* video-pause
* video-seek
* video-sync

### *2.4.4. Nhóm API chat realtime (Chat API)*

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** |
| POST | /api/v1/chat/message | Gửi tin nhắn trong phòng (UC020). |
| GET | /api/v1/chat/:roomId | Lấy toàn bộ lịch sử tin nhắn của phòng (UC021). |
| PATCH | /api/v1/chat/seen/:messageId | Đánh dấu tin nhắn đã xem (UC023). |
| POST | /api/v1/chat/react/:messageId | Thêm reaction (UC024). |
| POST | /api/v1/chat/reply/:messageId | Trả lời tin nhắn cụ thể (UC025). |

### *2.4.5. Nhóm API quản lý thành viên*

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** |
| GET | /api/v1/rooms/:id/members | Xem danh sách thành viên (UC026). |
| POST | /api/v1/rooms/:id/transfer-host | Chuyển quyền host cho thành viên khác (UC029). |

Socket events realtime hỗ trợ:

user-joined: Thông báo khi thành viên mới vào phòng (UC027).

user-left: Thông báo khi thành viên rời phòng (UC028).

host-changed: Gửi thông báo khi host được chuyển quyền (UC029).

## **2.5. Sự kiện Socket.io (Realtime Events)**

Song song với các API REST, hệ thống Watch Party sử dụng Socket.io để xử lý các luồng dữ liệu thời gian thực.

Dưới đây là các sự kiện chính:

### *2.5.1. Room Events*

|  |  |
| --- | --- |
| **Sự kiện** | **Mô tả** |
| join-room | Người dùng tham gia phòng (client → server). |
| leave-room | Người dùng rời khỏi phòng. |
| room-users | Gửi danh sách người dùng hiện tại trong phòng (server → client). |
| user-joined | Thông báo tới mọi thành viên khi có người mới tham gia. |
| user-left | Thông báo khi người dùng rời khỏi phòng. |

### *2.5.2. Video Synchronization Events*

|  |  |
| --- | --- |
| **Sự kiện** | **Mô tả** |
| video-control | Điều khiển video (play, pause, seek). Server broadcast tới các client khác. |
| request-sync | Người dùng mới yêu cầu trạng thái đồng bộ video. |
| sync-state | Server phản hồi lại trạng thái video hiện tại để đồng bộ với người mới. |

### *2.5.3. Chat Events*

|  |  |
| --- | --- |
| **Sự kiện** | **Mô tả** |
| chat-message | Gửi và nhận tin nhắn realtime giữa các thành viên. |
| user-typing | Hiển thị trạng thái “đang gõ” cho các thành viên khác. |
| message-seen | Đánh dấu tin nhắn đã được đọc. |

## **2.6. Đặc điểm kỹ thuật và bảo mật**

Xác thực:

Các endpoint quan trọng đều yêu cầu JWT Token hợp lệ. Token được cấp khi đăng nhập và tự động hết hạn sau thời gian định sẵn.

Phân quyền:

Host: có thể cập nhật, xóa phòng, điều khiển video và chuyển quyền chủ phòng.

Admin: có toàn quyền truy cập các API quản trị.

Member: chỉ có thể tham gia, chat và xem video đồng bộ.

Rate limiting:

API được giới hạn tốc độ để tránh spam và tấn công DDoS.

Realtime Communication:

Socket.io đảm nhiệm các tác vụ realtime như chat, đồng bộ video và sự kiện phòng, bảo đảm độ trễ thấp và tính ổn định khi nhiều người dùng tương tác cùng lúc.

Hệ thống Watch Party API được thiết kế với mục tiêu:

Phân tách rõ ràng giữa các tầng RESTful và Socket realtime.

Bảo mật tối đa thông qua JWT và middleware xác thực.

Hiệu năng cao, hỗ trợ đồng bộ video, chat, và sự kiện thời gian thực.

Mở rộng dễ dàng, có thể tích hợp thêm voice chat hoặc streaming trong tương lai.

Nhờ cấu trúc API gọn gàng và đồng nhất, frontend (React) có thể giao tiếp ổn định với backend (Node.js/Express), mang lại trải nghiệm xem phim cùng nhau mượt mà, realtime và có tính cộng đồng cao.

# CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## **3.1. Luồng hoạt động của các lớp (tương tác giữa các tầng)**

Hệ thống Watch Party được thiết kế theo kiến trúc 3 tầng (Three-Tier Architecture) gồm:

* Presentation Layer (Frontend) – giao diện người dùng React.js.
* Business Logic Layer (Backend) – xử lý nghiệp vụ bằng Node.js/Express.js.
* Data Access Layer (Database) – lưu trữ dữ liệu với MongoDB thông qua Mongoose.

Ngoài ra, hệ thống còn tích hợp Socket.io để xử lý truyền thông thời gian thực (real-time communication) giữa người dùng và máy chủ, đảm bảo việc phát/tạm dừng video, gửi tin nhắn và cập nhật trạng thái diễn ra đồng bộ.

A screenshot of a computer

AI-generated content may be incorrect.Ảnh 3.1 : Sơ đồ tuần tự của hệ thống

### *3.1.1. Luồng xác thực người dùng*

Mục tiêu : Luồng xác thực người dùng trong hệ thống Watch Party đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào các chức năng của ứng dụng như tạo phòng, tham gia phòng xem phim, và trò chuyện realtime.

Cơ chế xác thực được xây dựng theo mô hình JWT (JSON Web Token), kết hợp với bcrypt để mã hóa mật khẩu và Express middleware để kiểm tra quyền truy cập trong các request kế tiếp.

Giải pháp này giúp hệ thống đảm bảo an toàn, giảm tải cho máy chủ và dễ dàng mở rộng khi cần tích hợp các phương thức đăng nhập khác như Google Oauth.

Quy trình xử lý: Quy trình xác thực bao gồm hai phần chính: đăng ký (Sign Up) và đăng nhập (Login).

Cả hai đều tuân theo luồng xử lý tuần tự như sau:

A screenshot of a computer

AI-generated content may be incorrect.

Bảng 3.2 : Luồng xác thực

Quy trình xử lý tổng quát: Luồng xác thực người dùng bao gồm hai quy trình chính là đăng ký (Sign Up) và đăng nhập (Login). Cả hai quy trình đều tuân theo trình tự xử lý tuần tự như sau:

* Client gửi thông tin đăng nhập hoặc đăng ký  
  Người dùng nhập thông tin như tên, email và mật khẩu trên giao diện React. Sau đó, ứng dụng gửi yêu cầu đến máy chủ thông qua giao thức HTTP ở dạng JSON.
* Máy chủ kiểm tra dữ liệu đầu vào  
  Tầng backend được xây dựng bằng Express.js sẽ tiến hành kiểm tra dữ liệu nhận được. Các điều kiện như định dạng email, độ dài mật khẩu, và tính duy nhất của tài khoản được xác minh. Nếu dữ liệu không hợp lệ, hệ thống phản hồi lỗi và yêu cầu người dùng nhập lại thông tin.
* Truy vấn cơ sở dữ liệu  
  Máy chủ gửi yêu cầu tới cơ sở dữ liệu MongoDB để tìm kiếm người dùng dựa trên địa chỉ email. Nếu đang ở bước đăng ký, hệ thống sẽ kiểm tra xem email đã tồn tại hay chưa; nếu là đăng nhập, hệ thống sẽ truy vấn để lấy thông tin người dùng tương ứng.
* So sánh mật khẩu  
  Đối với quá trình đăng nhập, hệ thống sử dụng thư viện mã hóa bcrypt để so sánh mật khẩu người dùng nhập vào với chuỗi mật khẩu đã được mã hóa lưu trong cơ sở dữ liệu.  
  Nếu mật khẩu trùng khớp, quá trình xác thực tiếp tục; nếu không, hệ thống trả về thông báo lỗi và ngắt quy trình.
* Tạo mã thông báo JWT (JSON Web Token)  
  Khi người dùng được xác thực hợp lệ, máy chủ sẽ tạo một mã thông báo JWT. Mã này chứa thông tin cơ bản của người dùng như mã định danh (user ID), địa chỉ email, và vai trò (role).  
  Token được ký bằng một khóa bí mật và có thời gian hết hạn nhất định (thường từ 24 đến 48 giờ).
* Trả kết quả cho client  
  Máy chủ phản hồi về phía client một gói dữ liệu chứa thông tin người dùng cùng mã token JWT. Phía client sau đó sẽ lưu trữ token này trong bộ nhớ cục bộ của trình duyệt hoặc trong cookie để sử dụng cho các lần truy cập kế tiếp.
* Sử dụng token cho các yêu cầu tiếp theo  
  Mỗi khi người dùng gửi một yêu cầu mới (ví dụ: tham gia phòng, gửi tin nhắn hoặc xem danh sách phòng), client sẽ gửi kèm token trong phần tiêu đề của request để chứng minh danh tính.  
  Máy chủ sử dụng middleware để giải mã và xác minh token. Nếu token hợp lệ và còn hiệu lực, request được cho phép xử lý; ngược lại, hệ thống sẽ từ chối truy cập với thông báo lỗi “Unauthorized”.

Phân tích chi tiết vai trò của các thành phần:

* Frontend (React.js):

Đóng vai trò giao diện người dùng, cho phép nhập thông tin, hiển thị phản hồi và lưu trữ token.

Frontend chịu trách nhiệm gửi yêu cầu xác thực đến backend qua API, đồng thời lưu giữ token để gửi trong các yêu cầu sau.

* Backend (Express.js):

Là nơi xử lý toàn bộ quy trình xác thực. Backend tiếp nhận dữ liệu, kiểm tra hợp lệ, truy vấn cơ sở dữ liệu, so sánh mật khẩu, tạo JWT, và gửi phản hồi về client.

Backend cũng chứa các middleware xác thực giúp bảo vệ các API yêu cầu quyền truy cập.

* MongoDB:

Lưu trữ thông tin người dùng, bao gồm email, tên, mật khẩu đã mã hóa, và thời gian tạo tài khoản. Dữ liệu này được truy vấn thường xuyên khi người dùng đăng nhập hoặc khi cần xác minh danh tính.

* JWT Service:

Là cơ chế tạo và xác minh token. Nó giúp hệ thống không cần lưu trạng thái đăng nhập (session) trên server, từ đó giảm tải bộ nhớ và dễ mở rộng khi có nhiều người dùng đồng thời.

*3.1.2. Luồng tạo phòng*

Chức năng tạo phòng (Create Room) là bước khởi đầu của trải nghiệm xem phim trực tuyến trong hệ thống Watch Party.  
Mục tiêu của luồng này là cho phép người dùng sau khi đăng nhập có thể khởi tạo một phòng xem phim ảo, nơi họ có thể mời bạn bè cùng tham gia, chia sẻ video, và xem nội dung đồng bộ theo thời gian thực.  
Hệ thống phải đảm bảo việc tạo phòng diễn ra nhanh chóng, định danh duy nhất, có thể quản lý được người tham gia, và duy trì kết nối ổn định giữa các client thông qua Socket.io.

Quy trình xử lý tổng quát.Toàn bộ quy trình tạo phòng trong hệ thống Watch Party diễn ra theo các bước sau:

* Người dùng khởi tạo yêu cầu tạo phòng
* Sau khi đăng nhập, người dùng truy cập giao diện “Tạo phòng mới” trên frontend React.
* Khi người dùng nhập thông tin (tên phòng, liên kết video, chế độ công khai hay riêng tư) và nhấn nút “Tạo phòng”, ứng dụng gửi yêu cầu đến máy chủ thông qua giao thức HTTP hoặc socket event.
* Frontend gửi yêu cầu đến backend
* Ứng dụng React gửi yêu cầu POST /api/rooms/create kèm token xác thực (JWT).
* Dữ liệu gửi đi bao gồm:

Tên phòng (Room Name)

Liên kết video (Video URL hoặc ID YouTube)

Chế độ (public/private)

Thông tin người tạo (lấy từ token)

* Backend xác thực và xử lý yêu cầu
* Backend nhận request và kiểm tra token thông qua middleware xác thực để đảm bảo người dùng hợp lệ.
* Nếu hợp lệ, hệ thống khởi tạo một đối tượng phòng mới, gán thông tin người tạo làm host (chủ phòng).
* Mỗi phòng sẽ có một mã định danh duy nhất (roomId) được sinh tự động để các người dùng khác có thể tham gia thông qua mã hoặc liên kết chia sẻ.
* Lưu thông tin phòng vào cơ sở dữ liệu
* Hệ thống lưu thông tin phòng vừa tạo vào cơ sở dữ liệu MongoDB, bao gồm:
* Mã phòng (roomId)
* Tên phòng
* Chủ phòng (hostId)
* Liên kết video
* Danh sách thành viên (ban đầu chỉ có host)
* Trạng thái hoạt động (active, closed)

- Thời gian tạo

Việc lưu trữ này đảm bảo rằng nếu người dùng thoát và quay lại, phòng vẫn có thể được khôi phục hoặc quản lý.

* Khởi tạo kết nối Socket.io

Sau khi phòng được tạo thành công, backend kích hoạt server Socket.io để theo dõi các sự kiện liên quan đến phòng mới, chẳng hạn như:

Người dùng tham gia hoặc rời khỏi phòng.

Cập nhật trạng thái video (play, pause, seek).

Gửi và nhận tin nhắn trong phòng chat.

* Tại thời điểm này, host sẽ được tự động thêm vào “room channel” tương ứng trong socket server.
* Trả phản hồi về cho client

Backend phản hồi lại cho frontend dữ liệu bao gồm:

Mã phòng (roomId)

Thông tin người tạo

Liên kết chia sẻ phòng

Thông báo tạo phòng thành công

Phía frontend nhận phản hồi, lưu lại thông tin phòng, và tự động điều hướng người dùng đến giao diện xem phim (Watch Room Interface).

Frontend cập nhật giao diện và thiết lập socket listener

Tại giao diện phòng xem phim, React thiết lập các listener của Socket.io để sẵn sàng xử lý các sự kiện phát sinh như tin nhắn mới, thành viên tham gia, hoặc thay đổi trạng thái video.

Người tạo phòng sẽ thấy giao diện điều khiển (host control) cho phép mời người khác, phát video, và đồng bộ nội dung.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.2: Luồng tạo phòng

Phân tích vai trò các thành phần

* Frontend (React.js):

Hiển thị giao diện tạo phòng, nhận dữ liệu đầu vào và gửi yêu cầu tạo phòng qua API.

Nhận phản hồi từ server, lưu thông tin phòng và chuyển hướng sang giao diện xem phim.

Thiết lập các socket event listener để quản lý các hành động thời gian thực.

* Backend (Express.js và Socket.io):

Kiểm tra token xác thực để đảm bảo người dùng hợp lệ.

Tạo bản ghi phòng mới trong cơ sở dữ liệu MongoDB.

Quản lý luồng dữ liệu realtime qua Socket.io (phát video, chat, đồng bộ thành viên).

* Database (MongoDB):

Lưu thông tin chi tiết về phòng, bao gồm ID, host, danh sách thành viên và video hiện tại.

Hỗ trợ khả năng khôi phục hoặc truy vấn danh sách phòng đang hoạt động.

A black and white rectangular object with text

AI-generated content may be incorrect.

3.1.3. Luồng đồng bộ video

Chức năng đồng bộ video (Video Synchronization) là thành phần cốt lõi của hệ thống Watch Party, đảm bảo tất cả người dùng trong cùng một phòng xem phim đều nhìn thấy video đang phát ở cùng thời điểm và cùng trạng thái (phát, tạm dừng, tua).

Tính năng này được triển khai dựa trên cơ chế truyền thông thời gian thực (real-time communication) sử dụng Socket.io, cho phép client và server trao đổi sự kiện (event) ngay lập tức mà không cần tải lại trang.

Quy trình xử lý tổng quát.Toàn bộ luồng hoạt động của tính năng đồng bộ video diễn ra theo các bước sau:

Người dùng thực hiện hành động điều khiển video

Khi người dùng (thường là host) thực hiện một hành động như “Phát (Play)”, “Tạm dừng (Pause)” hoặc “Tua (Seek)”, ứng dụng React sẽ ghi nhận sự kiện này từ trình phát video (video player).

Frontend phát sự kiện (emit event) qua Socket.io

Ngay sau khi nhận hành động, frontend gửi sự kiện tương ứng qua kênh Socket.io đã được kết nối với server.

Sự kiện bao gồm thông tin chi tiết như:

* Loại hành động: play, pause, seek
* Thời điểm trong video: timestamp hiện tại
* ID phòng (roomId) để xác định nơi cần phát lại (broadcast)

Socket Server nhận và xử lý sự kiện

Máy chủ Socket.io lắng nghe tất cả các sự kiện từ client trong phòng.

Khi có sự kiện điều khiển video từ host, server sẽ:

* Xác định phòng (roomId) mà sự kiện được gửi từ đó.
* Lưu lại trạng thái video mới nhất (state và timestamp) vào bộ nhớ tạm hoặc cơ sở dữ liệu (nếu có).
* Gửi (broadcast) sự kiện này đến tất cả các client khác trong cùng phòng, ngoại trừ người phát ra lệnh.

Các client khác nhận sự kiện và cập nhật trình phát video

Mỗi người dùng khác trong phòng (client) sẽ nhận được gói dữ liệu sự kiện từ Socket Server.

Ứng dụng React xử lý sự kiện và cập nhật trình phát video sao cho:

Nếu nhận được “play” → trình phát bắt đầu phát tại đúng timestamp.

Nếu nhận được “pause” → video dừng ngay tại vị trí tương ứng.

Nếu nhận được “seek” → tua video đến vị trí mới.

Quá trình này diễn ra đồng bộ, đảm bảo mọi người xem cùng một khung hình gần như cùng lúc.

Cập nhật trạng thái phòng

Người dùng mới tham gia được đồng bộ ngay

A screenshot of a computer

AI-generated content may be incorrect.Khi có người mới tham gia phòng, server sẽ tự động gửi trạng thái video hiện tại (bao gồm thời điểm và hành động cuối cùng) đến client mới để đảm bảo người này được đồng bộ hoàn toàn với những người còn lại.

Ảnh 3.2 : Luồng đồng bộ

Phân tích vai trò của các thành phần

* Frontend (React.js + Socket.io Client):

Ghi nhận sự kiện điều khiển video từ giao diện.

Gửi dữ liệu sự kiện qua Socket.io đến server.

Lắng nghe các sự kiện broadcast từ server và cập nhật trình phát video theo thời gian thực.

Đảm bảo giao diện video của người dùng luôn hiển thị đúng trạng thái hiện tại của phòng.

* Backend (Express.js + Socket.io Server):

Lắng nghe các sự kiện điều khiển video từ host.

Xử lý logic broadcast đến tất cả thành viên trong phòng.

Quản lý danh sách người dùng đang trực tuyến trong từng phòng.

Lưu lại trạng thái video gần nhất để phục vụ việc khôi phục khi cần thiết.

* Database (MongoDB):

Lưu thông tin phòng, trạng thái video hiện tại và lịch sử thao tác (nếu có).

Hỗ trợ việc đồng bộ dữ liệu giữa các lần truy cập.

A black background with white rectangles

AI-generated content may be incorrect.

3.1.4. Luồng realtime

Chức năng nhắn tin thời gian thực (Realtime Chat) cho phép người dùng trong cùng một phòng xem phim có thể trao đổi, bình luận, và tương tác ngay trong lúc xem video.

Cơ chế này sử dụng Socket.io để truyền thông hai chiều giữa client và server, đảm bảo tin nhắn được gửi và hiển thị tức thời mà không cần tải lại trang.

Quy trình xử lý tổng quát.Luồng hoạt động của tính năng chat realtime trong hệ thống diễn ra theo các bước sau:

* Người dùng nhập và gửi tin nhắn
* Người dùng gõ nội dung tin nhắn trong khung chat của giao diện React.
* Khi nhấn phím Enter hoặc nút Gửi, ứng dụng frontend sẽ kích hoạt sự kiện gửi tin nhắn.

Client gửi sự kiện chat-message qua Socket.io

Ứng dụng React phát sự kiện (emit) tên là "chat-message" đến Socket Server, kèm theo dữ liệu:

* ID người gửi (userId hoặc username)
* Nội dung tin nhắn
* ID phòng (roomId)
* Thời điểm gửi (timestamp)

Server nhận sự kiện và xử lý lưu trữ

Socket Server (Node.js + Socket.io) lắng nghe sự kiện "chat-message" từ client.

Khi nhận được, server sẽ:

* Xác định phòng (roomId) tương ứng.
* Lưu tin nhắn này vào database MongoDB thông qua model ChatMessage, bao gồm các thông tin: người gửi, nội dung, phòng, thời gian gửi.
* Tạo một bản ghi mới để đảm bảo lịch sử tin nhắn có thể truy xuất lại khi người dùng tải lại trang hoặc tham gia sau.
* Server broadcast tin nhắn đến tất cả người dùng trong phòng

Sau khi lưu thành công, server sẽ phát (broadcast) lại sự kiện "chat-message" đến tất cả client trong cùng roomId, bao gồm cả người gửi.

Gói dữ liệu broadcast chứa nội dung tin nhắn, tên người gửi, và thời điểm gửi, để các client có thể hiển thị ngay lập tức.

Client nhận sự kiện và cập nhật giao diện (UI)

Tất cả các client trong phòng sẽ nhận được sự kiện "chat-message" mới từ server.

Mỗi client sẽ thêm tin nhắn này vào danh sách hiển thị trong giao diện chat (message list), đồng thời tự động cuộn xuống để hiển thị tin nhắn mới nhất.

Nếu tin nhắn do chính người dùng gửi, hệ thống có thể đánh dấu khác biệt (ví dụ: căn phải hoặc đổi màu).

Client gửi sự kiện message-seen khi người dùng đã xem tin nhắn

Khi người dùng cuộn đến cuối khung chat hoặc cửa sổ tin nhắn được mở, ứng dụng sẽ gửi sự kiện "message-seen" đến server, báo rằng người này đã đọc tin.

Server ghi nhận trạng thái “đã xem” trong database, và broadcast lại thông tin này đến những người khác trong phòng để cập nhật chỉ báo “seen”.

A screenshot of a computer

AI-generated content may be incorrect.

Ảnh 3.4 : Luồng chat realtime

Phân tích vai trò của các thành phần

Frontend (React.js + Socket.io Client):

Ghi nhận hành động nhập và gửi tin nhắn từ người dùng.

Phát sự kiện chat-message và message-seen đến server.

Lắng nghe các sự kiện broadcast để hiển thị tin nhắn mới và trạng thái “đã xem”.

Quản lý giao diện danh sách tin nhắn, hiển thị avatar, thời gian gửi và hiệu ứng realtime.

Backend (Node.js + Socket.io Server):

Lắng nghe sự kiện chat-message và xử lý lưu tin nhắn vào MongoDB.

Broadcast tin nhắn đến tất cả client trong cùng phòng.

Lắng nghe sự kiện message-seen để cập nhật trạng thái đã đọc.

Quản lý socket connection theo từng phòng, đảm bảo mỗi người chỉ nhận tin nhắn từ phòng mà họ đang tham gia.

Database (MongoDB):

Lưu trữ toàn bộ lịch sử tin nhắn, người gửi, thời gian và trạng thái xem.

Dữ liệu này được truy xuất lại mỗi khi người dùng tải lại phòng hoặc tham gia giữa chừng.

## **3.2. Sơ đồ Usecases chính**

### *3.2.1. Mục tiêu*

Sơ đồ Use Case mô tả toàn bộ các chức năng chính của hệ thống Watch Party dưới góc nhìn người dùng.

Hệ thống hướng đến việc cho phép nhiều người xem video cùng nhau trong thời gian thực, đồng bộ thao tác xem, chat và tương tác trong cùng một “phòng xem chung” (watch room).

Sơ đồ này giúp xác định phạm vi chức năng và mối quan hệ giữa người dùng và hệ thống, làm cơ sở cho các bước thiết kế chi tiết tiếp theo.

### *3.2.2. Các tác nhân (Actors)*

Sơ đồ Use Case thể hiện các chức năng cốt lõi của hệ thống Watch Party, mô tả cách người dùng tương tác với ứng dụng trong các tình huống khác nhau: đăng nhập, tạo phòng, xem video cùng nhau và trò chuyện thời gian thực.

Hệ thống hỗ trợ nhiều người cùng tham gia một “phòng xem chung” thông qua cơ chế đồng bộ video và chat realtime, đảm bảo trải nghiệm tương tác trực tiếp, liên tục và mượt mà.

|  |  |
| --- | --- |
| Tác nhân | Mô tả |
| Guest (Người dùng chưa đăng nhập) (Phát triển trong tương lai) | Có thể xem danh sách phòng công khai, tìm kiếm phòng, hoặc đăng ký tài khoản để tham gia. |
| User (Người dùng đã đăng nhập) | Có quyền tạo phòng, tham gia, chat realtime, xem video đồng bộ, và quản lý thông tin cá nhân. |
| Host (Chủ phòng) | Là người tạo phòng, có thêm quyền chỉnh sửa thông tin phòng, đồng bộ video, và quản lý thành viên. |
| System (Hệ thống) | Thực hiện xác thực, đồng bộ dữ liệu video, xử lý chat, và quản lý sự kiện realtime qua Socket.io. |

Bảng 3.2: Bảng actors

### *3.2.3. Nhóm chức năng và các Usecase*

|  |  |  |
| --- | --- | --- |
| Mã UC | Tên Use Case | Mô tả ngắn gọn |
| UC001 | Đăng ký tài khoản mới | Người dùng nhập email, username, password để tạo tài khoản. |
| UC002 | Đăng nhập bằng username/email và password | Người dùng đăng nhập thủ công để truy cập hệ thống. |
| UC003 | Đăng nhập bằng Google OAuth | Người dùng đăng nhập nhanh qua tài khoản Google. |
| UC004 | Đăng xuất | Kết thúc phiên làm việc, hủy JWT token hiện tại. |

Bảng 3.3: Quản lý người dùng

A white cylinder with black text

AI-generated content may be incorrect.Ảnh 3.3.1: Sơ đồ Quản lý người dùng

|  |  |  |
| --- | --- | --- |
| Mã UC | Tên Use Case | Mô tả ngắn gọn |
| UC006 | Tạo phòng mới với video URL | Người dùng tạo phòng xem phim với liên kết video (YouTube, Vimeo, v.v.). |
| UC007 | Tham gia phòng bằng mã phòng | Người dùng nhập mã phòng để tham gia phòng cụ thể. |
| UC008 | Tham gia phòng bằng mật khẩu | Phòng có thể bảo vệ bằng mật khẩu, người dùng cần nhập đúng để tham gia. |
| UC009 | Rời phòng | Người dùng rời khỏi phòng đang tham gia. |
| UC010 | Xem danh sách phòng công khai | Hiển thị danh sách các phòng đang hoạt động và cho phép truy cập. |
| UC011 | Tìm kiếm phòng theo tên/mô tả | Hỗ trợ tìm nhanh phòng theo tiêu chí người dùng nhập. |
| UC012 | Lọc phòng theo tags | Lọc phòng theo chủ đề (ví dụ: phim kinh dị, hoạt hình, v.v.). |
| UC013 | Cập nhật thông tin phòng (chỉ host) | Host có thể thay đổi tên, mô tả, hoặc URL video. |
| UC014 | Xóa phòng (chỉ host) | Host có quyền xóa vĩnh viễn phòng khỏi hệ thống. |

Bảng 3.4: Quản lý phòng

A black background with white rectangles

AI-generated content may be incorrect.

Ảnh 3.4.1: Sơ đồ quản lý phòng

|  |  |  |
| --- | --- | --- |
| **Mã UC** | **Tên Use Case** | **Mô tả ngắn gọn** |
| **UC015** | Phát video đồng bộ | Khi host phát video, tất cả thành viên trong phòng cũng được phát. |
| **UC016** | Tạm dừng video đồng bộ | Khi host tạm dừng, video của tất cả thành viên dừng lại cùng lúc. |
| **UC017** | Chuyển đến thời điểm cụ thể | Host có thể tua video đến thời điểm khác và tất cả thành viên được cập nhật. |
| **UC018** | Yêu cầu đồng bộ trạng thái video | Người mới tham gia phòng gửi yêu cầu đồng bộ để bắt kịp trạng thái hiện tại. |
| **UC019** | Gửi trạng thái video hiện tại | Client gửi vị trí video của mình lên server để hệ thống cập nhật cho người khác. |

Bảng 3.5: Xem video đồng bộ

A black and white rectangular object with text

AI-generated content may be incorrect.

Ảnh 3.5.1: Sơ đồ xem video đồng bộ

|  |  |  |
| --- | --- | --- |
| **Mã UC** | **Tên Use Case** | **Mô tả ngắn gọn** |
| **UC020** | Gửi tin nhắn text trong phòng | Người dùng gửi tin nhắn, hệ thống lưu và phát lại cho mọi thành viên. |
| **UC021** | Xem tin nhắn của các thành viên khác | Giao diện chat hiển thị nội dung gửi từ các thành viên khác trong phòng. |
| **UC022** | Hiển thị trạng thái đang gõ | Hệ thống hiển thị khi người khác đang nhập tin nhắn. |
| **UC023** | Đánh dấu tin nhắn đã xem | Tin nhắn được đánh dấu “đã xem” khi người dùng đọc. |
| **UC024** | Thêm reaction vào tin nhắn | Người dùng có thể thêm biểu tượng cảm xúc vào tin nhắn. |
| **UC025** | Trả lời tin nhắn cụ thể | Cho phép phản hồi trực tiếp một tin nhắn trong đoạn hội thoại. |

Bảng 3.6: Chat realtime

A black background with white rectangles

AI-generated content may be incorrect.

Ảnh 3.6.1: Sơ đồ chat realtime

|  |  |  |
| --- | --- | --- |
| **Mã UC** | **Tên Use Case** | **Mô tả ngắn gọn** |
| **UC026** | Xem danh sách thành viên trong phòng | Hiển thị danh sách người đang tham gia cùng. |
| **UC027** | Thông báo khi có thành viên mới tham gia | Hệ thống thông báo realtime khi có người vào phòng. |
| **UC028** | Thông báo khi thành viên rời phòng | Hiển thị thông báo khi ai đó rời khỏi phòng. |
| **UC029** | Chuyển quyền host khi host rời phòng | Khi host thoát, hệ thống tự động chuyển quyền cho thành viên khác. |

A screenshot of a cell phone

AI-generated content may be incorrect. Bảng 3.7 : Quản lý thành viên

Ảnh 3.7.1: Sơ đồ quản lý thành viên

## **3.3. Sơ đồ cơ sở dữ liệu (ERD – Entity Relationship Diagram)**

### *3.3.1. Mục tiêu*

Sơ đồ ERD (Entity–Relationship Diagram) mô tả cấu trúc dữ liệu của hệ thống Watch Party, thể hiện các thực thể (entities) chính, thuộc tính (attributes) của chúng, cùng mối quan hệ (relationships) giữa các bảng/collection trong cơ sở dữ liệu MongoDB.

Cơ sở dữ liệu được thiết kế theo mô hình phi quan hệ (NoSQL), ưu tiên tốc độ truy xuất, khả năng mở rộng, và tính linh hoạt trong việc lưu trữ dữ liệu động như chat realtime, danh sách người xem và trạng thái video.

### *3.3.2. Các thực thể chính*

Hệ thống gồm ba collection chính:

Users

Rooms

Messages

Dưới đây là mô tả chi tiết từng thực thể và quan hệ giữa chúng.

### *3.3.3. Collection: Users*

|  |  |  |
| --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| \_id | ObjectId | Khóa chính (Primary Key) định danh duy nhất cho mỗi người dùng. |
| username | String | Tên hiển thị của người dùng trong hệ thống. |
| email | String | Địa chỉ email dùng để đăng nhập hoặc xác thực. |
| passwordHash | String | Mật khẩu được mã hóa bằng thuật toán băm (hash). |
| role | String | Vai trò người dùng: "user", "host", "admin". |
| googleId | String | ID người dùng khi đăng nhập bằng Google OAuth. |
| isVerified | Boolean | Xác định người dùng đã xác thực tài khoản hay chưa. |
| createdAt | Date | Thời điểm tài khoản được tạo. |

Quan hệ:

Một người dùng có thể tạo nhiều phòng (1 - n với Rooms).

Một người dùng có thể gửi nhiều tin nhắn trong các phòng (1 - n với Messages).

Một người dùng có thể là thành viên trong nhiều phòng (quan hệ nhiều–nhiều qua trường members trong Rooms).

### *3.3.4. Collection: Rooms*

|  |  |  |
| --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| \_id | ObjectId | Khóa chính định danh phòng. |
| roomCode | String | Mã phòng duy nhất, dùng để người khác tham gia. |
| name | String | Tên phòng (ví dụ: “Movie Night”, “Anime Lovers”). |
| hostId | ObjectId | Khóa ngoại tham chiếu đến \_id của Users (chủ phòng). |
| videoUrl | String | URL của video được phát trong phòng. |
| videoType | String | Kiểu video (YouTube, Vimeo, local, v.v.). |
| members | Array | Danh sách thành viên trong phòng. Mỗi phần tử gồm: |
| - userId: ObjectId (tham chiếu Users), |  |  |
| - username: String, |  |  |
| - isHost: Boolean (xác định vai trò chủ phòng). |  |  |
| settings | Object | Cấu hình của phòng gồm: |
| - allowChat: Boolean – cho phép trò chuyện trong phòng. |  |  |
| - allowVideoControl: Boolean – cho phép các thành viên điều khiển video. |  |  |

### *3.3.5. Collection: Messages*

|  |  |  |
| --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| \_id | ObjectId | Khóa chính của tin nhắn. |
| roomId | ObjectId | Tham chiếu đến phòng chứa tin nhắn (Rooms.\_id). |
| userId | ObjectId | Tham chiếu đến người gửi tin nhắn (Users.\_id). |
| username | String | Tên hiển thị của người gửi. |
| content | String | Nội dung tin nhắn. |
| messageType | String | Loại tin nhắn (text, system, emoji, reply, …). |
| reactions | Array | Danh sách biểu cảm (emoji) từ người dùng. |
| - emoji: String – biểu tượng cảm xúc. |  |  |
| - userId: ObjectId – người thêm reaction. |  |  |
| seenBy | Array | Danh sách người đã xem tin nhắn. |
| - userId: ObjectId – người đã xem. |  |  |
| - seenAt: Date – thời điểm xem. |  |  |

Mỗi Message thuộc về một phòng (roomId → Rooms.\_id).

Mỗi Message được gửi bởi một người dùng (userId → Users.\_id).

Một phòng (Room) có thể chứa nhiều tin nhắn (1 - n).

### *A screenshot of a computer AI-generated content may be incorrect.3.3.6. Mối quan hệ giữa các collection*

|  |  |
| --- | --- |
| **Quan hệ** | **Mô tả** |
| **Users – Rooms** | Quan hệ **1–n**: Một người dùng có thể tạo nhiều phòng (qua hostId), đồng thời là thành viên trong nhiều phòng (members). |
| **Users – Messages** | Quan hệ **1–n**: Mỗi người dùng có thể gửi nhiều tin nhắn. |
| **Rooms – Messages** | Quan hệ **1–n**: Một phòng chứa nhiều tin nhắn. |
| **Rooms – Users (members)** | Quan hệ **n–n**: Một phòng có nhiều người dùng, và một người dùng có thể tham gia nhiều phòng. |

Ảnh 3.3: Sơ đồ quan hệ giữa các collection

# CHƯƠNG IV. PHÂN TÍCH ỨNG DỤNG

## **4.1. Giao diện**

Giao diện người dùng được xây dựng với mục tiêu thuận tiện, rõ ràng cho từng vai trò người dùng. React.js giúp tách bạch phần xử lý và giao diện, do đó các thay đổi giao diện (UI/UX) có thể thực hiện mà không ảnh hưởng đến logic code.

Hệ thống Watch Party được thiết kế theo hướng trực quan, thân thiện và nhất quán, nhằm mang lại trải nghiệm tốt nhất cho người dùng khi tương tác với nền tảng xem video đồng bộ.

Ứng dụng được phát triển theo triết lý “simple yet functional” – tối giản nhưng đầy đủ tính năng – giúp người dùng ở mọi trình độ kỹ thuật đều có thể sử dụng dễ dàng.

Các yếu tố UI được tổ chức theo nguyên tắc phân cấp thị giác (visual hierarchy), đảm bảo người dùng nhanh chóng nhận biết được hành động chính trên mỗi màn hình. Màu sắc chủ đạo là xanh dương (#2563eb) kết hợp với tông trắng xám trung tính, thể hiện tinh thần hiện đại, tin cậy và tương đồng với các ứng dụng SaaS phổ biến.

### *4.1.1. Giao diện Dashboard*

Dashboard là trung tâm quản trị cá nhân của người dùng sau khi đăng nhập.

Mục tiêu của trang là cung cấp cái nhìn tổng thể về hoạt động xem phim, cho phép người dùng quản lý các phòng đã tạo hoặc tham gia, đồng thời thực hiện các hành động chính như tạo, tìm kiếm, tham gia hoặc xóa phòng.

Cấu trúc giao diện

Khu vực thống kê (Summary Widgets): Hiển thị các chỉ số như:

Tổng số phòng đã tạo;

Số phòng đang hoạt động;

Số phòng riêng tư.

Mỗi thẻ có màu biểu tượng riêng, tạo nhấn mạnh trực quan.

Thanh tìm kiếm (Search Bar): Giúp người dùng lọc nhanh các phòng theo tên hoặc mã.

Danh sách phòng (Room List): Mỗi hàng thể hiện:

Tên phòng, chủ phòng, ngày tạo, nền tảng video (ví dụ: YouTube), mã phòng (có thể sao chép), và các nút thao tác “Tham gia” hoặc “Xóa”.

Nút hành động nổi bật (Create Room Button): Đặt bên phải trên cùng, giúp người dùng dễ dàng khởi tạo phòng mới mà không cần quay lại trang chủ.

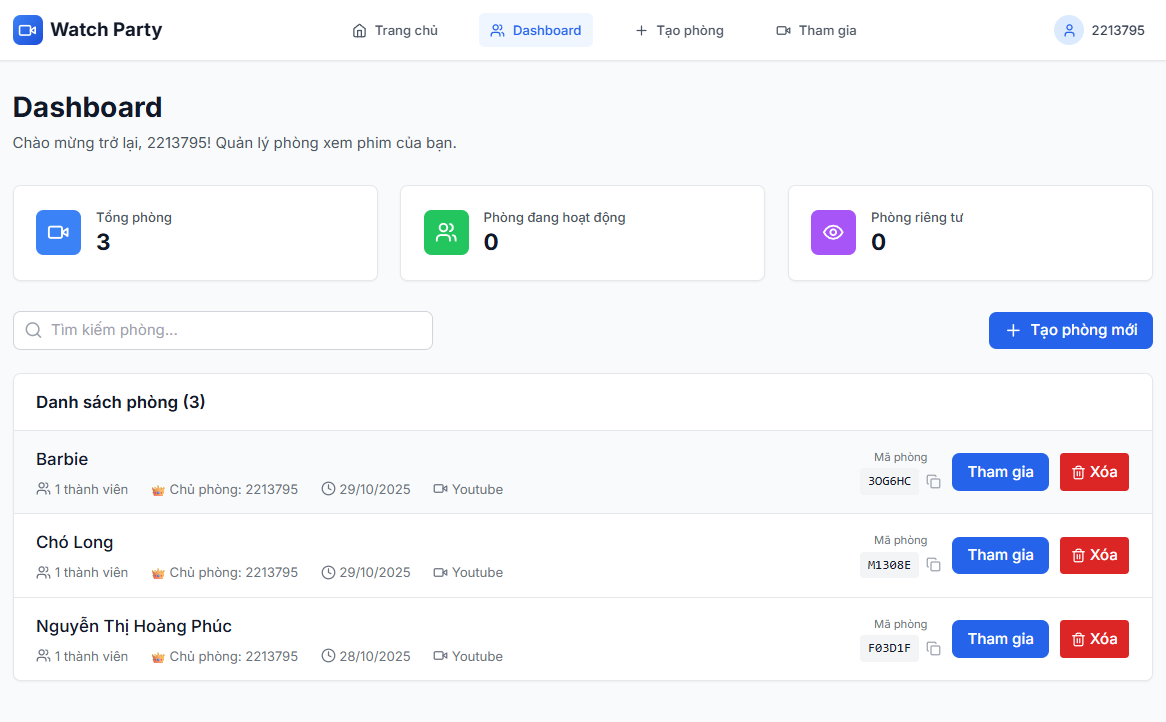
Đánh giá học thuật

Trang Dashboard thể hiện rõ nguyên lý quản lý thông tin (information hierarchy) và quy tắc Gestalt trong thiết kế giao diện.

Cấu trúc bảng (tabular layout) giúp so sánh và quét thông tin hiệu quả. Các hành động được phân tách rõ ràng giữa vai trò người dùng (thành viên thường, host).

Về mặt khả dụng, hệ thống phản hồi thao tác thông qua các hiệu ứng và thông báo tức thì (toast message).

Đề xuất mở rộng bao gồm bộ lọc trạng thái phòng (active/inactive) và bộ đếm thành viên để nâng cao trải nghiệm điều hướng.Total Employees (Tổng số nhân viên): hiển thị tổng số nhân viên hiện có trong hệ thống.



Ảnh 4.1: Giao diện Dashboard

### *4.1.2. Trang chủ (Home Page)*

Trang chủ là điểm vào đầu tiên của hệ thống, đóng vai trò giới thiệu tổng quan về nền tảng và định hướng người dùng thực hiện hành động chính như tạo phòng hoặc tham gia phòng.

Đây là khu vực thể hiện giá trị cốt lõi của ứng dụng: xem phim cùng nhau, kết nối bạn bè qua trải nghiệm đồng bộ và trò chuyện thời gian thực.

Cấu trúc giao diện

Thanh điều hướng (Header / Navbar): Gồm logo “Watch Party”, các liên kết “Trang chủ”, “Dashboard”, “Tạo phòng”, “Tham gia”, cùng biểu tượng tài khoản người dùng. Khi người dùng đăng nhập, thanh này đồng thời thể hiện trạng thái xác thực.

Hero Section: Trình bày khẩu hiệu “Xem phim cùng nhau thật dễ dàng” cùng mô tả ngắn gọn, kết hợp hai nút hành động nổi bật:

Tạo phòng mới (primary CTA) – hướng người dùng đến quy trình tạo phòng xem phim.

Tham gia phòng (secondary CTA) – cho phép nhập mã để tham gia phòng hiện có.

Thống kê tổng quan (Statistics Overview): Hiển thị các số liệu nổi bật như tổng số người dùng, số phòng đã tạo, tổng số giờ xem và điểm đánh giá trung bình. Đây là cách tăng độ tin cậy và hấp dẫn cho nền tảng.

Tính năng nổi bật (Key Features): Bố trí dưới dạng lưới bốn cột, mỗi thẻ (card) gồm biểu tượng, tiêu đề và mô tả ngắn – minh họa cho các năng lực chính như:

Xem video đồng bộ;

Phòng xem chung;

Chat thời gian thực;

Bảo mật cao.

Phần hướng dẫn sử dụng (How It Works): Mô tả quy trình ba bước trực quan – “Tạo phòng → Mời bạn bè → Xem cùng nhau”.

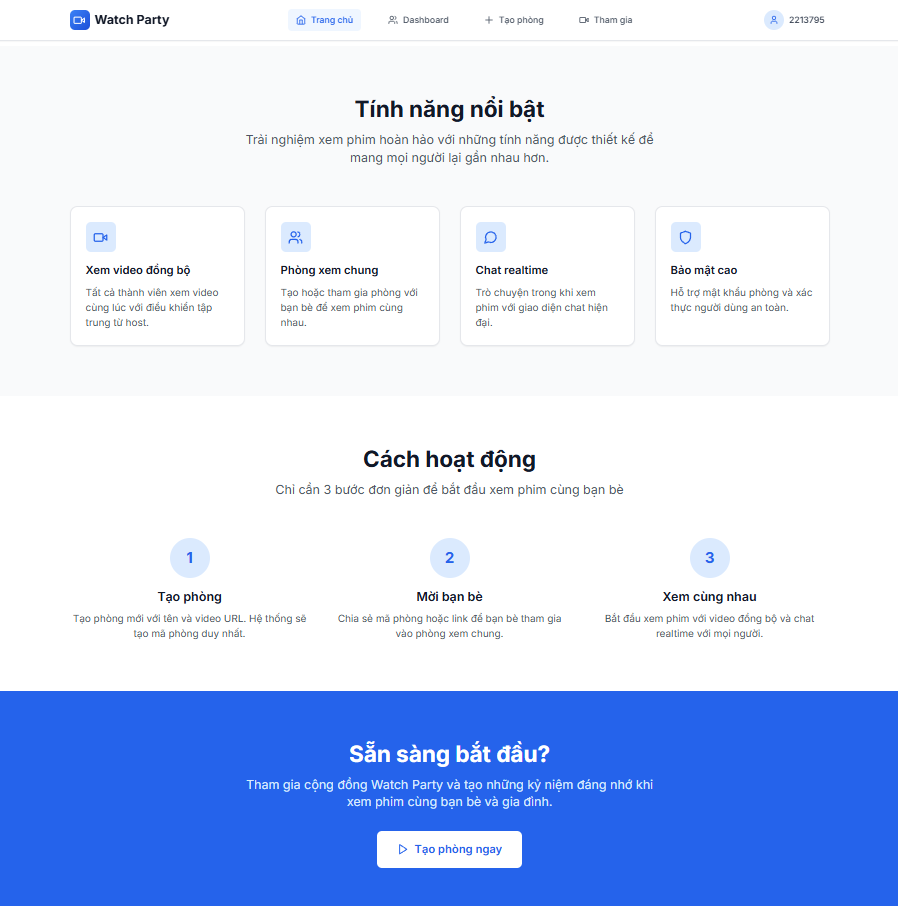
Chân trang CTA (Footer Call to Action): Gợi nhắc người dùng hành động với thông điệp “Sẵn sàng bắt đầu?” cùng nút “Tạo phòng ngay”.

Trang chủ đạt được mức độ thẩm mỹ và khả dụng (aesthetic & usability) cao, thể hiện rõ triết lý User-Centered Design.

Các vùng tương tác được sắp xếp logic, giúp người dùng dễ dàng nhận diện mục tiêu chính của ứng dụng.

Việc sử dụng tông màu xanh – trắng kết hợp typography đơn giản (Sans-serif, cỡ lớn) tăng khả năng đọc hiểu và giảm gánh nặng nhận thức (cognitive load).

Giao diện có thể cải thiện thêm bằng việc bổ sung video minh họa hoặc hình ảnh trực quan về quá trình sử dụng để nâng cao tính hấp dẫn.



Ảnh 4.2: Giao diện trang chủ

### *4.1.3. Giao diện tạo phòng (Create Room)*

Trang này hỗ trợ người dùng cấu hình phòng xem phim mới, bao gồm thông tin cơ bản, nguồn video và các tùy chọn riêng tư. Đây là một trong những điểm tương tác cốt lõi của hệ thống.

Cấu trúc giao diện

Biểu mẫu tạo phòng (Create Room Form) gồm các trường:

Tên phòng (bắt buộc);

Mô tả phòng (tùy chọn);

URL video (bắt buộc, hỗ trợ định dạng YouTube, HLS, MP4);

Loại video (dropdown);

Tùy chọn phòng riêng tư (checkbox);

Giới hạn số thành viên tối đa (numeric input).

Nút xác nhận (Submit Button): Nổi bật với màu xanh lam, thể hiện hành động chính.

Gợi ý định dạng URL: Cung cấp hướng dẫn cụ thể để giảm sai sót khi nhập liệu.

Đánh giá học thuật

Trang tạo phòng tuân thủ nguyên tắc form usability của Nielsen Norman Group, gồm các yếu tố:

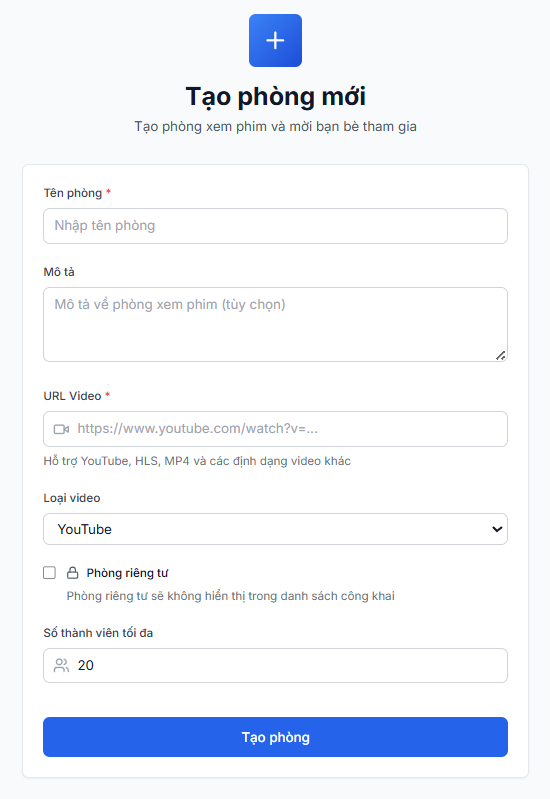
Tối giản số lượng trường bắt buộc;

Giữ trạng thái nhập liệu khi xảy ra lỗi;

Phản hồi tức thời khi người dùng nhập sai định dạng URL.

Thiết kế tuyến tính từ trên xuống giúp người dùng xử lý theo luồng tự nhiên (top-down flow).

Có thể mở rộng thêm phần preview video để kiểm chứng đường dẫn hợp lệ trước khi lưu.



Ảnh 4.3: Giao diện tạo phòng

### *4.1.4. Giao diện tham gia phòng (Join Room)*

Trang này giúp người dùng nhanh chóng truy cập vào phòng đã được tạo bằng mã phòng (Room Code), tạo điều kiện cho quá trình kết nối nhóm.

Cấu trúc giao diện

Trường nhập mã phòng (Input Field): Có placeholder “Nhập mã phòng (VD: ABC123)” và xác thực định dạng cơ bản.

Nút tham gia (Join Button): Màu xanh lam nổi bật, biểu trưng cho hành động tích cực.

Phần hướng dẫn (Instruction Block): Mô tả chi tiết các bước tham gia phòng theo trình tự logic:

Nhận mã phòng từ người tạo.

Nhập mã vào ô phía trên.

Nhấn “Tham gia phòng”.

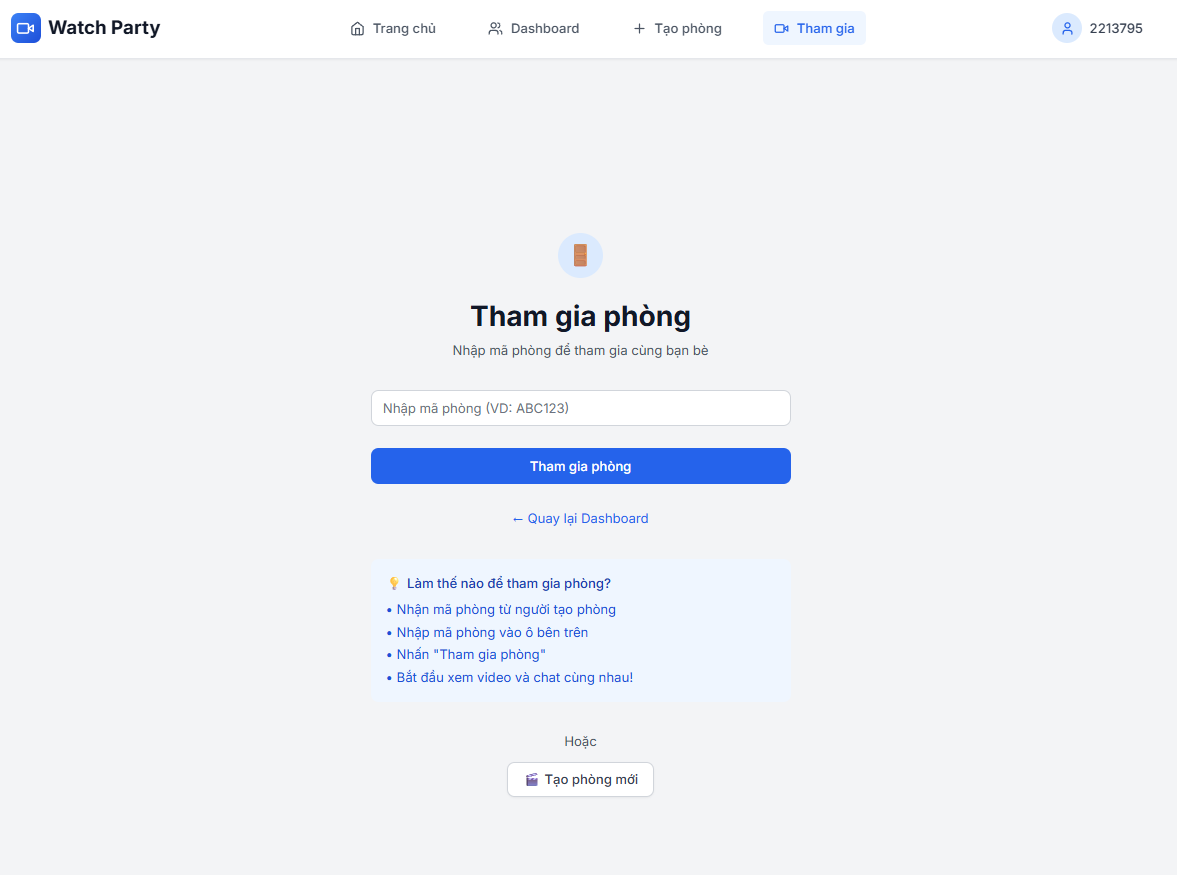
Bắt đầu xem video và trò chuyện.

Liên kết phụ (Secondary Action): “Tạo phòng mới” dành cho người muốn khởi tạo thay vì tham gia.

Thiết kế trang tham gia phòng thể hiện sự chú trọng đến onboarding experience – đảm bảo cả người dùng mới cũng có thể thao tác dễ dàng.

Bố cục gọn gàng, không gây nhiễu thị giác; các bước hướng dẫn rõ ràng giảm thiểu sai sót.

Đề xuất thêm danh sách phòng gần đây hoặc bộ nhớ tự động để người dùng tái tham gia nhanh hơn.



Ảnh 4.4: Giao diện tham gia phòng.

### 4.1.5. Trang Đăng nhập (Login Page)

Trang đăng nhập là điểm khởi đầu trong quy trình xác thực người dùng. Giao diện này cho phép người dùng truy cập vào hệ thống thông qua tài khoản đã đăng ký, đảm bảo an toàn và quản lý quyền truy cập hợp lệ.

Logo và thanh điều hướng (Navbar): hiển thị logo Watch Party cùng các liên kết chính: Trang chủ, Đăng nhập, Đăng ký. Giao diện tối giản, thể hiện rõ trạng thái người dùng chưa đăng nhập.

Tiêu đề chính: “Đăng nhập vào tài khoản” được đặt ở trung tâm, sử dụng font lớn, đậm, giúp định hướng hành động chính ngay khi vào trang.

Biểu mẫu đăng nhập (Form):

Trường Tên người dùng hoặc Email (với placeholder hướng dẫn).

Trường Mật khẩu, hỗ trợ hiển thị/ẩn ký tự bằng biểu tượng con mắt.

Tùy chọn “Ghi nhớ đăng nhập” giúp lưu phiên đăng nhập cục bộ.

Liên kết “Quên mật khẩu?” cho phép xử lý khôi phục truy cập (có thể triển khai trong phiên bản sau).

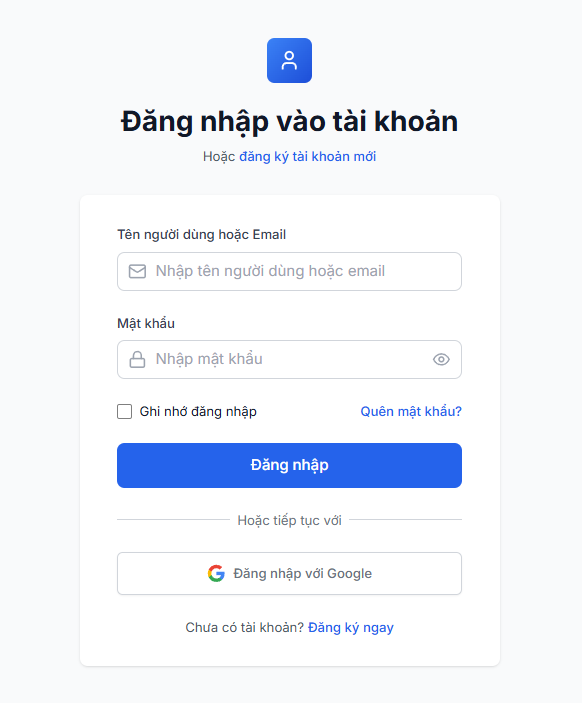
Nút hành động chính: “Đăng nhập” (màu xanh dương đậm) nổi bật và được đặt ở vị trí dễ truy cập bằng chuột hoặc phím Enter.

Đăng nhập bằng Google (OAuth): cung cấp phương thức xác thực nhanh, thuận tiện, tăng trải nghiệm người dùng.

Liên kết đăng ký: “Chưa có tài khoản? Đăng ký ngay” đặt ở cuối form, định hướng rõ luồng onboarding.

Trải nghiệm người dùng (UX): tối giản, dễ hiểu, tránh quá tải thông tin.

Khả năng truy cập: hỗ trợ bàn phím, nhãn nhập liệu rõ ràng, tương phản màu đáp ứng tiêu chuẩn WCAG.

Thị giác học (Visual Hierarchy): nhấn mạnh CTA (nút xanh), căn giữa nội dung, giúp người dùng tập trung.

Ảnh 4.5: Giao diện đăng nhập (Login Page)

### 4.1.6. Giao diện Trang Đăng ký (Register Page)

Trang đăng ký cho phép người dùng tạo tài khoản mới, lưu thông tin vào cơ sở dữ liệu hệ thống. Đây là một phần quan trọng của luồng xác thực (authentication flow), đảm bảo tính hợp lệ và bảo mật của dữ liệu đầu vào.

Logo & Navbar: tương tự trang đăng nhập, thống nhất thương hiệu.

Tiêu đề: “Tạo tài khoản mới” cùng liên kết chuyển hướng “đăng nhập vào tài khoản hiện có” ở ngay bên dưới, giúp định hướng linh hoạt.

Form đăng ký gồm các trường:

Tên người dùng – định danh hiển thị trong hệ thống.

Email – dùng cho xác thực và đăng nhập.

Mật khẩu – có các chỉ dẫn điều kiện bảo mật như “ít nhất 6 ký tự”, “có chữ hoa, chữ thường, số”.

Xác nhận mật khẩu – tránh sai lệch khi nhập lại.

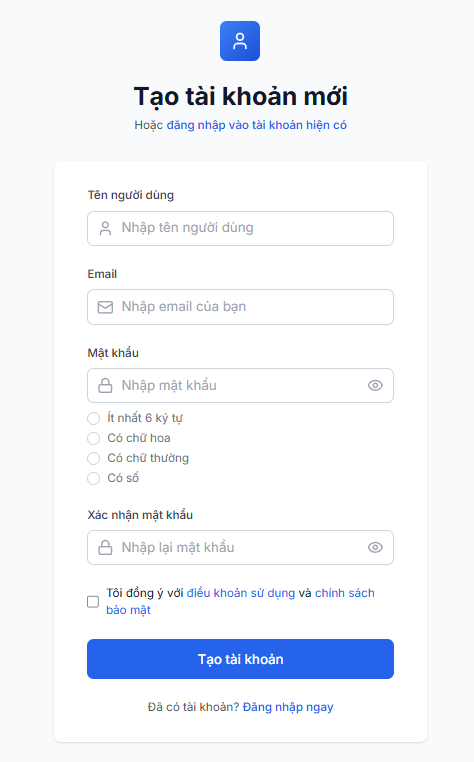
Checkbox điều khoản – yêu cầu người dùng đồng ý với “Điều khoản sử dụng và Chính sách bảo mật”.

Nút hành động: “Tạo tài khoản” – màu xanh chủ đạo, dễ nhận biết.

Đường dẫn phụ: “Đã có tài khoản? Đăng nhập ngay”, giúp người dùng chuyển hướng nhanh.

Bố cục: Căn giữa, độ rộng vừa phải giúp người dùng tập trung.

Tương tác: Kiểm tra ràng buộc đầu vào (validation) có thể hiển thị trực tiếp hoặc qua cảnh báo phía dưới mỗi trường.

Tính khả dụng: Cấu trúc đơn giản, hướng dẫn cụ thể, dễ thao tác cho cả người dùng mới.

Ảnh 4.6: Giao diện đăng ký (Register Page)

### *4.1.7. Giao diện hồ sơ (Profile)*

Trang hồ sơ cá nhân cho phép người dùng quản lý thông tin định danh trong hệ thống, đồng thời cung cấp cái nhìn tổng quan về mức độ hoạt động của tài khoản (thống kê phòng, tin nhắn, ngày tham gia).

Tiêu đề & mô tả: “Hồ sơ cá nhân – Quản lý thông tin tài khoản của bạn”, thể hiện rõ mục đích và phạm vi chức năng.

Khối thông tin cá nhân:

Tên người dùng – có thể chỉnh sửa.

Email – hiển thị nhưng không thể thay đổi (tránh xung đột định danh).

Nút “Chỉnh sửa” (màu xanh) kích hoạt chế độ chỉnh sửa inline hoặc mở form cập nhật.

Khối thống kê: hiển thị tổng số phòng đã tạo, số tin nhắn đã gửi, và ngày tham gia (thời điểm tạo tài khoản).

Khối tài khoản: chứa hai nút:

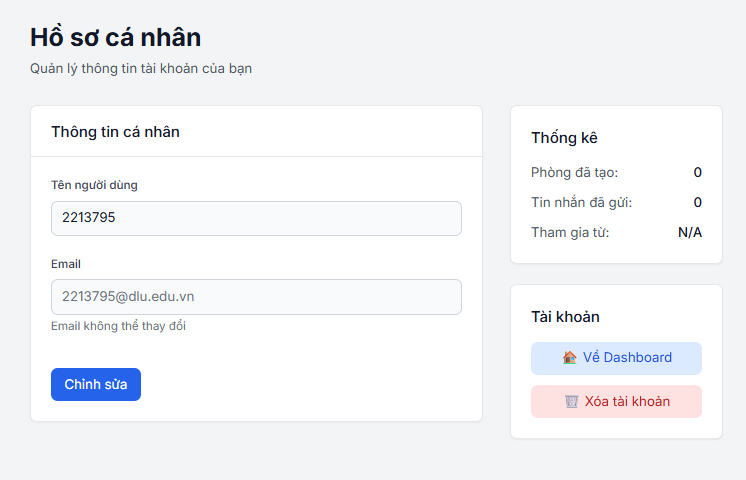
“Về Dashboard” – điều hướng trở lại bảng điều khiển.

“Xóa tài khoản” – màu đỏ, cảnh báo rõ ràng, thể hiện hành động quan trọng và không thể hoàn tác.

Cấu trúc lưới (Grid layout): phân tách rõ ràng phần nội dung chính (thông tin người dùng) và phụ (thống kê, hành động tài khoản).

Nguyên tắc nhận diện (Recognition over Recall): các thành phần hành động được hiển thị trực quan, không cần ghi nhớ.

Khả dụng: hiển thị trạng thái dữ liệu “0” cho người mới giúp tránh lỗi hiển thị rỗng.

Ảnh 4.7: Giao diện hồ sơ (Profile)

## **4.2. Bảo mật hệ thống**

### *4.2.1. Xác thực và Ủy quyền (Authentication & Authorization)*

a. Mục tiêu thiết kế

Hệ thống hiện đang triển khai cơ chế xác thực người dùng dựa trên JSON Web Token (JWT) cho toàn bộ API. Các middleware như authenticateToken được áp dụng nhất quán nhằm đảm bảo chỉ người dùng hợp lệ mới có thể truy cập các tài nguyên yêu cầu. Cơ chế phân quyền được tổ chức theo mô hình vai trò gồm user và admin, kết hợp với kiểm soát chi tiết trong phạm vi phòng thông qua các middleware như requireHost và requireRoomMember. Ngoài ra, hệ thống hỗ trợ xác thực bên thứ ba thông qua Google OAuth, giúp đa dạng hóa phương thức đăng nhập và tăng trải nghiệm người dùng.

Tuy nhiên, vẫn tồn tại một số hạn chế cần được cải thiện:

Hệ thống chưa triển khai refresh token hoặc cơ chế xoay vòng token (token rotation), khiến thời gian sống của token dài dễ gây rủi ro khi bị lộ.

Không có cơ chế giới hạn số lần đăng nhập sai, dẫn đến khả năng bị tấn công brute-force.

Chưa có chính sách hết hạn và thu hồi token rõ ràng, ảnh hưởng đến tính an toàn phiên người dùng.

Phân quyền trong Socket.io chưa được áp dụng đồng bộ cho tất cả các sự kiện realtime, đặc biệt là các event như chat, sync, hoặc typing.

### *4.2.2. Giao tiếp và Chính sách CORS (Communication & CORS Policy)*

Về mặt bảo mật truyền thông, hệ thống đã cấu hình Helmet nhằm bổ sung các HTTP header an toàn, và áp dụng CORS middleware với cấu hình allowedHeaders cùng giới hạn kích thước request JSON (tối đa 10MB). Các biện pháp này giúp giảm thiểu nguy cơ tấn công dạng Cross-Origin Request Forgery (CSRF) hoặc Payload Overflow.

Tuy nhiên, trong môi trường triển khai thực tế (production), hệ thống vẫn còn những điểm cần hoàn thiện:

Chưa ép buộc sử dụng HTTPS (HSTS), do đó dễ gặp rủi ro “mixed content” khi triển khai trên môi trường không bảo mật.

Danh sách CORS whitelist phụ thuộc biến môi trường, cần được kiểm soát chặt chẽ để tránh rò rỉ quyền truy cập từ các domain không tin cậy.

### *4.2.3. Giới hạn tần suất và Phòng chống tấn công DoS (Rate Limiting & Anti-DoS)*

Hệ thống backend đã tích hợp express-rate-limit để kiểm soát tần suất gửi yêu cầu HTTP từ phía client, qua đó giảm thiểu nguy cơ tấn công từ chối dịch vụ (DoS). Tuy nhiên, cấu hình này vẫn có thể bị thay đổi hoặc bỏ sót khi triển khai.

Đối với kênh Socket.io, hệ thống chưa có cơ chế throttling toàn diện, dẫn đến khả năng bị spam sự kiện realtime (như video-control, typing, hoặc seek) gây ra vòng lặp đồng bộ hoặc lỗi 429/400. Cần bổ sung giới hạn tần suất theo người dùng hoặc phòng để bảo vệ tài nguyên máy chủ.

### *4.2.4. Xác thực và Bảo mật Socket.io (Socket Authentication & Security)*

Cơ chế xác thực trong Socket.io được thực hiện thông qua JWT handshake, giúp liên kết mỗi kết nối socket với người dùng hợp lệ (socket.user). Tuy nhiên, trong các phiên bản trước, hệ thống từng phát sự kiện tới toàn bộ phòng bao gồm cả người gửi, dễ dẫn đến hiện tượng lặp dữ liệu hoặc lộ thông tin.

Đề xuất cải tiến:

Áp dụng broadcast có kiểm soát (socket.broadcast.to(...)) để đảm bảo chỉ những client khác nhận sự kiện.

Kiểm tra quyền truy cập tại mọi event, không giới hạn ở video-control, nhằm ngăn người dùng giả mạo hoặc gửi sự kiện trái phép.

### 4.2.5. Kiểm tra đầu vào và Chống tiêm nhiễm (Input Validation & Injection Prevention)

Hệ thống hiện đang sử dụng express-validator trên nhiều route liên quan đến phòng và xác thực, đồng thời áp dụng ràng buộc schema (schema constraints) trong Mongoose để hạn chế lỗi nhập liệu và tấn công dạng injection.

Tuy nhiên, vẫn tồn tại những điểm yếu cần khắc phục:

Nội dung chat được gửi dưới dạng text tự do, cần được sanitize và escape ở cả client và server nhằm phòng chống XSS (Cross-Site Scripting).

Trường Room.password hiện lưu plaintext, cần được băm (hash) trước khi lưu trữ.

Cần kiểm tra nghiêm ngặt URL video để tránh tấn công SSRF (Server-Side Request Forgery) hoặc Open Redirect.

### *4.2.6. Bảo vệ dữ liệu và quyền riêng tư (Data Protection & Privacy)*

Hệ thống đã triển khai băm mật khẩu người dùng bằng bcrypt và ẩn thuộc tính passwordHash khi trả dữ liệu về client, đáp ứng tiêu chuẩn bảo mật cơ bản.

Tuy nhiên, vẫn tồn tại một số rủi ro tiềm ẩn:

Mật khẩu phòng (Room.password) vẫn lưu ở dạng thô.

Một số metadata của tin nhắn có thể lưu thông tin nhạy cảm như địa chỉ IP hoặc User-Agent – cần xem xét ẩn danh hóa hoặc áp dụng chính sách TTL (Time-To-Live) để tự động xóa sau thời gian nhất định.

### *4.2.7. Ghi log và Giám sát (Logging & Monitoring)*

Hệ thống hiện có ghi log ở mức cơ bản, chủ yếu phục vụ theo dõi hoạt động runtime. Tuy nhiên, chưa có cơ chế audit log chi tiết cho các hành vi nhạy cảm như:

* Chuyển quyền chủ phòng (host transfer),
* Xóa phòng hoặc thành viên,
* Gán hoặc thu hồi quyền moderator.

Đề xuất bổ sung hệ thống log tập trung (centralized logging) và giám sát lỗi tự động (error tracking) để phát hiện sớm các hành vi bất thường.

### *4.2.8. Quản lý bí mật và cấu hình triển khai (Secret Management & Deployment Security)*

Hệ thống đang sử dụng tệp .env để lưu trữ các biến môi trường quan trọng như JWT\_SECRET, MONGO\_URI và GOOGLE\_CLIENT\_ID. Tuy nhiên, vẫn còn một số hạn chế:

* Chưa có cơ chế xoay vòng JWT\_SECRET định kỳ, làm tăng rủi ro nếu khóa bị lộ.
* Thiếu hướng dẫn cấu hình production rõ ràng, như HSTS, thiết lập trust proxy, hoặc tách biến môi trường riêng cho staging và production.
* Cần quy định rate limit riêng cho từng route và môi trường triển khai riêng biệt để đảm bảo tính nhất quán và an toàn.

## **4.3. Hạn chế**

Các hạn chế kiến trúc và chức năng liên quan đến bảo mật

Cơ chế đồng bộ video có thể gây vòng lặp và spam nếu client không gắn cờ nguồn hoặc server broadcast không đúng cách.

Chưa áp dụng mô hình token ngắn hạn (short-lived token) hoặc refresh token để kiểm soát thời gian sống phiên.

Phân quyền admin mới được áp dụng ở mức middleware, chưa có giao diện quản trị an toàn.

Thiếu cơ chế bảo vệ CSRF nếu hệ thống chuyển sang xác thực cookie-based trong tương lai (hiện sử dụng Bearer Token nên rủi ro thấp hơn).

Socket.io chưa giới hạn tần suất theo sự kiện, dễ bị khai thác gây nghẽn băng thông hoặc spam tin nhắn.

## **4.4. Đánh giá**

Nhìn chung, hệ thống Watch Party đã có nền tảng bảo mật vững chắc nhờ việc triển khai JWT, Helmet, bcrypt, và rate limit cơ bản. Tuy nhiên, để đạt chuẩn bảo mật cho môi trường sản xuất (production-grade), cần bổ sung refresh token, cơ chế xoay vòng khóa bí mật, xác thực đa tầng trong Socket.io, cùng kiểm soát tần suất realtime. Đồng thời, nên triển khai các chính sách bảo mật dữ liệu và audit log nhằm nâng cao tính minh bạch và khả năng truy vết trong quá trình vận hành.

# CHƯƠNG V. TỔNG KẾT

## **5.1. Đánh giá ứng dụng**

Sau quá trình phân tích, thiết kế, hiện thực và kiểm thử, hệ thống xem cùng nhau – Watch Party đã cơ bản hoàn thành với đầy đủ các chức năng theo mục tiêu đề ra.

### *5.1.1. Ưu điểm của ứng dụng*

Hệ thống đáp ứng đầy đủ các yêu cầu cốt lõi của một nền tảng xem video trực tuyến kết hợp giao tiếp thời gian thực. Các chức năng như đăng ký, đăng nhập, tạo phòng, tham gia phòng, xem video đồng bộ, chat realtime và quản lý thành viên đều hoạt động ổn định, đảm bảo đúng logic nghiệp vụ đề ra.

Ứng dụng hỗ trợ nhiều tình huống sử dụng thực tế, như phòng công khai – riêng tư, chia sẻ mã phòng, điều khiển video tập thể và trò chuyện giữa các thành viên.

Cơ chế đồng bộ video theo thời gian thực được triển khai chính xác, giúp người xem ở các thiết bị khác nhau vẫn duy trì cùng trạng thái (play, pause, seek) mà không bị lệch pha đáng kể.

Phần chat realtime hoạt động mượt mà, cho phép gửi, nhận, hiển thị tin nhắn, reaction và trạng thái “đang gõ”, “đã xem” tương tự các ứng dụng giao tiếp hiện đại.

Thiết kế phần mềm linh hoạt, dễ mở rộng:

Hệ thống được xây dựng dựa trên kiến trúc phân tầng (Presentation – Application – Data) rõ ràng, tách biệt giữa xử lý giao diện, logic nghiệp vụ và truy cập dữ liệu.

Frontend sử dụng React.js, giúp giao diện có khả năng tái sử dụng component, dễ mở rộng tính năng và giảm chi phí bảo trì.

Backend triển khai bằng Node.js/Express với cấu trúc module rõ ràng, giúp dễ dàng thêm mới API hoặc mở rộng hệ thống mà không ảnh hưởng đến phần lõi.

Việc sử dụng MongoDB giúp linh hoạt trong lưu trữ dữ liệu động như người dùng, phòng và tin nhắn. Các collection (Users, Rooms, Messages) được thiết kế hợp lý, đảm bảo hiệu quả khi truy vấn và hiển thị dữ liệu thời gian thực.

Hệ thống hỗ trợ mở rộng dễ dàng sang các nền tảng khác (mobile/web app) thông qua API RESTful và giao tiếp Socket.io.

Tính ổn định và chính xác cao:

Hệ thống xử lý đồng bộ thời gian thực ổn định, đảm bảo khi một người dùng thao tác (phát, tạm dừng, tua video), toàn bộ người khác trong phòng đều nhận được trạng thái chính xác gần như tức thì.

Cơ chế xác thực bằng JWT token giúp đảm bảo người dùng hợp lệ khi thực hiện các hành động (tạo phòng, gửi tin nhắn, điều khiển video).

Giao tiếp giữa client và server được kiểm soát và xác minh liên tục, giảm thiểu lỗi phát sinh khi nhiều người dùng cùng thao tác.

Trong thử nghiệm, hệ thống duy trì độ trễ thấp (<300ms) trong quá trình đồng bộ video và gửi tin nhắn, chứng tỏ tính ổn định của giải pháp thời gian thực.

Khả năng mở rộng và tích hợp:

Kiến trúc của hệ thống được thiết kế để dễ dàng mở rộng về cả quy mô và tính năng:

* + Có thể tích hợp giao diện quản trị (Admin Dashboard) để theo dõi người dùng và phòng hoạt động.
  + Bổ sung chức năng thông báo đẩy (push notification) hoặc lời mời tham gia phòng qua liên kết.
  + Tích hợp nhiều loại nguồn video khác nhau (YouTube, HLS, Vimeo…).

Nhờ nền tảng Node.js + Socket.io, hệ thống có thể mở rộng theo chiều ngang (horizontal scaling) khi triển khai trên môi trường đám mây.

Tính bảo mật và giám sát:

Cơ chế bảo mật hiện tại đã triển khai các biện pháp căn bản: helmet, rate limit, CORS whitelist, mã hóa mật khẩu bằng bcrypt, và JWT xác thực người dùng.

Hệ thống hạn chế các rủi ro phổ biến như tiêm nhiễm (injection) hoặc truy cập trái phép.

Tuy nhiên, vẫn còn tiềm năng nâng cao như: thêm cơ chế refresh token, hạn chế tần suất socket event, và mã hóa mật khẩu phòng để tăng cường an toàn thông tin.

### *5.1.2. Nhược điểm của ứng dụng*

Mặc dù hệ thống Watch Party đã đạt được nhiều kết quả đáng khích lệ về mặt chức năng và hiệu năng, nhưng vẫn tồn tại một số hạn chế nhất định cần được xem xét trong quá trình phát triển tiếp theo. Những nhược điểm này chủ yếu liên quan đến hiệu năng realtime, bảo mật, kiến trúc mở rộng và trải nghiệm người dùng nâng cao.

* Hạn chế về bảo mật :

Thiếu cơ chế làm mới và thu hồi token (refresh/rotation):

Hệ thống hiện sử dụng JWT để xác thực người dùng, tuy nhiên token này không có chính sách tự động hết hạn hoặc thu hồi. Điều này có thể gây rủi ro nếu token bị rò rỉ hoặc sử dụng trái phép.

Chưa mã hóa mật khẩu phòng (room password):

Dữ liệu mật khẩu phòng được lưu ở dạng plaintext trong cơ sở dữ liệu, dễ bị lộ nếu xảy ra truy cập trái phép.

Kiểm soát sự kiện Socket.io chưa toàn diện:

Một số sự kiện realtime như video-control hoặc typing chưa có giới hạn tần suất (throttling), dẫn đến khả năng bị spam hoặc gây nghẽn kết nối.

Chưa có chính sách khóa tài khoản sau nhiều lần đăng nhập sai:

Điều này tạo nguy cơ tấn công brute-force.

* Hạn chế về hiệu năng và realtime:

Đồng bộ video có thể gặp độ trễ trong mạng yếu:

Trong các thử nghiệm thực tế, độ trễ giữa các thiết bị có thể tăng lên nếu mạng không ổn định hoặc băng thông thấp. Hệ thống chưa áp dụng cơ chế hiệu chỉnh (resync) định kỳ để đồng bộ lại thời gian phát.

Thiếu cơ chế tối ưu hóa socket khi có nhiều người dùng cùng phòng:

Khi phòng có số lượng thành viên lớn (>20), việc broadcast toàn bộ sự kiện cho tất cả socket có thể gây tải cho server.

* Hạn chế trong quản lý và giám sát hệ thống:

Chưa có hệ thống giám sát và log chi tiết:

Các hành vi nhạy cảm như xóa phòng, chuyển quyền chủ phòng hoặc thao tác admin chưa được ghi nhận vào audit log.

Thiếu công cụ quản trị người dùng và phòng (Admin UI):

Mặc dù backend hỗ trợ endpoint dành cho admin, nhưng chưa có giao diện quản trị riêng biệt để theo dõi, khóa, hoặc điều chỉnh người dùng/phòng.

## **5.2. Hướng phát triển của ứng dụng**

Dự án Watch Party — một nền tảng xem video đồng bộ và giao tiếp thời gian thực — đã chứng minh tính khả thi và hiệu quả trong việc kết nối nhiều người dùng cùng trải nghiệm nội dung trực tuyến. Tuy nhiên, để tiến tới giai đoạn triển khai thực tế ở quy mô lớn, hệ thống cần tiếp tục được hoàn thiện, tối ưu và mở rộng trên nhiều khía cạnh: kỹ thuật, bảo mật, trải nghiệm người dùng, và khả năng tích hợp.

Hoàn thiện cơ chế bảo mật và xác thực người dùng

Bổ sung refresh token và cơ chế thu hồi JWT:

Tăng cường an toàn cho hệ thống xác thực bằng cách tách access token và refresh token, thiết lập thời gian sống ngắn cho access token, đồng thời cho phép thu hồi token khi người dùng đăng xuất hoặc phát hiện hành vi bất thường.

* Mã hóa mật khẩu phòng và dữ liệu nhạy cảm:

Áp dụng hash mạnh (bcrypt hoặc Argon2) cho các mật khẩu phòng, đồng thời mã hóa metadata (ví dụ: lịch sử tin nhắn hoặc IP người dùng) nhằm bảo vệ quyền riêng tư.

Bổ sung chính sách bảo mật nâng cao:

Giới hạn số lần đăng nhập sai (lockout).

Triển khai kiểm tra XSS, CSRF, và SQL Injection định kỳ.

Tích hợp HSTS, CSP, và HTTPS bắt buộc khi triển khai production.

Tăng cường xác thực hai yếu tố (2FA):

Cho phép người dùng lựa chọn bật 2FA qua email hoặc ứng dụng Authenticator nhằm nâng cao độ tin cậy.

* Nâng cấp hệ thống realtime và tối ưu hiệu năng

Triển khai mô hình phân tán (Distributed Realtime Architecture):

Hiện tại hệ thống hoạt động trên một node Socket.io; trong giai đoạn tiếp theo, cần tích hợp Redis adapter hoặc Kafka message broker để đồng bộ trạng thái giữa nhiều server, hỗ trợ hàng nghìn người dùng đồng thời.

Cải thiện thuật toán đồng bộ video (Synchronization Engine):

Bổ sung cơ chế hiệu chỉnh định kỳ (auto-resync) dựa trên timestamp trung tâm, giúp giảm độ trễ khi có sự sai lệch giữa các client.

Giới hạn tần suất (Rate Limiting) cho sự kiện Socket:

Ngăn chặn spam các hành động như play/pause/seek hoặc gửi tin nhắn bằng middleware giới hạn theo phòng và người dùng.

Tối ưu cache và phân phối nội dung video:

Kết hợp CDN hoặc caching server để tăng tốc độ tải video, đặc biệt khi có nhiều người xem cùng một nội dung.

* Mở rộng chức năng người dùng và trải nghiệm

Hệ thống bạn bè (Friends & Invitations):

Cho phép người dùng thêm bạn bè, gửi lời mời tham gia phòng hoặc chia sẻ video riêng tư.

Tích hợp thông báo realtime (Realtime Notifications):

Cung cấp thông báo khi có tin nhắn mới, lời mời xem chung, hoặc khi host thay đổi video.

Hệ thống quyền trong phòng:

Phân quyền chi tiết (Host – Moderator – Member):

Host: Toàn quyền điều khiển video và quản lý phòng.

Moderator: Giới hạn quyền, có thể tắt chat hoặc quản lý người dùng.

Member: Chỉ xem và chat theo cấu hình phòng.

Bổ sung tính năng chia sẻ đa nền tảng:

Tích hợp chia sẻ phòng qua mạng xã hội (Facebook, X/Twitter, Discord) hoặc liên kết mời trực tiếp.

Hỗ trợ đa ngôn ngữ (Internationalization – i18n):

Xây dựng giao diện đa ngôn ngữ, ban đầu hỗ trợ Tiếng Việt, Tiếng Anh và Tiếng Nhật để mở rộng phạm vi người dùng.

* Phát triển hệ thống quản trị (Admin Dashboard)

Trang quản trị tập trung (Admin Panel):

Cung cấp giao diện cho quản trị viên giám sát người dùng, phòng, và log hoạt động.

Các tính năng gồm:

Quản lý user (khóa, xóa, phân quyền).

Quản lý phòng (xóa, ẩn, điều chỉnh cài đặt).

Theo dõi log hệ thống và hoạt động Socket.

Bổ sung hệ thống cảnh báo và thống kê:

Biểu đồ hoạt động người dùng theo thời gian.

Phát hiện và cảnh báo hành vi bất thường (ví dụ spam tin nhắn, tạo phòng hàng loạt).

* Mở rộng khả năng triển khai và tích hợp

Tích hợp dịch vụ streaming mở rộng:

Cho phép người dùng đồng bộ xem từ các nguồn như Netflix, Disney+, hoặc drive cá nhân (nếu có API hợp pháp).

Triển khai mobile app (React Native hoặc Flutter):

Cung cấp trải nghiệm xem chung tiện lợi trên di động, hỗ trợ thông báo đẩy (push notification) và đăng nhập nhanh.

## **5.3. Kết luận**

Dự án Watch Party đã đạt được mục tiêu xây dựng một nền tảng xem video đồng bộ và giao tiếp thời gian thực, thể hiện khả năng vận dụng công nghệ web hiện đại (React, Node.js, Socket.io, MongoDB) vào một hệ thống có tính tương tác cao. Ứng dụng hoạt động ổn định, giao diện thân thiện và đáp ứng tốt các yêu cầu nghiệp vụ cơ bản như tạo phòng, đồng bộ video, trò chuyện realtime và quản lý người dùng.

Tuy nhiên, hệ thống vẫn còn một số hạn chế về bảo mật, tối ưu hiệu năng và khả năng mở rộng, đặc biệt ở khía cạnh xác thực token, giới hạn tần suất socket và quản trị người dùng. Đây là những vấn đề cần tiếp tục cải thiện để hướng tới sản phẩm có tính chuyên nghiệp và an toàn hơn.

Nhìn chung, Watch Party là một nền tảng có tiềm năng phát triển lâu dài — vừa mang giá trị thực tiễn trong việc kết nối và chia sẻ nội dung trực tuyến, vừa là minh chứng cho năng lực học hỏi thiết kế, triển khai và có thêm kiến thức về tư duy hệ thống phần mềm hiện đại của nhóm thực hiện.

# CHƯƠNG VI. TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | N. H. Minh, "codejava.net," 3 8 2021. [Online]. Available: https://www.codejava.net/ides/netbeans/create-multi-module-maven-project-netbeans. |
| [2] | Weka, "weka.io," Weka, [Online]. Available: https://www.weka.io/. |
| [3] | D. Community, "SOLID Design Principles Explained: Building Better Software Architecture," 12 June 2025. [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. |
| [4] | S. Watts, "SOLID Principles in Object-Oriented Design," 26 August 2024. [Online]. Available: https://www.bmc.com/blogs/solid-design-principles/. |
| [5] | Refactoring.Guru, "Decorator Design Pattern – Intent and Structure," [Online]. Available: https://refactoring.guru/design-patterns/decorator. |
| [6] | Refactoring.Guru, "Strategy Design Pattern – Intent and Example," [Online]. Available: https://refactoring.guru/design-patterns/strategy. |
| [7] | D. Practitioners, "Layered Architecture – Definition," [Online]. Available: https://ddd-practitioners.com/home/glossary/layered-architecture/. |
| [8] | A. Documentation, "Using the SQLite Database," 12 December 2016. [Online]. Available: https://www.apertis.org/guides/app\_devel/sqlite/. |
| [9] | J. o. B. T. Research, "The Impact of POS Systems on Small Business Efficiency," Alexandria Computers, 2023. [Online]. Available: https://alexandriacomputers.com/impact-of-pos-systems-on-small-business-efficiency/. |