
SOFTWARE REQUIREMENTS SPECIFICATION (SRS) – GATHER

Project: Gather (Gather Town Clone)
Last updated: 2026-02-06
Scope: Frontend + Backend + Realtime (Socket.IO + WebRTC/SFU)

REVISION HISTORY

Version: 0.1
Date: 2026-02-06
Description: Viết lại requirements cho Gather dựa trên SRS mẫu, đối chiếu với codebase hiện tại.

---

1. PURPOSE

Tài liệu này mô tả các yêu cầu phần mềm (bao gồm yêu cầu chức năng và phi chức năng) cho dự án Gather – một không gian làm việc ảo 2D tương tự Gather Town. Mục tiêu của tài liệu là thống nhất phạm vi hệ thống, hành vi mong đợi, tiêu chí nghiệm thu và các ràng buộc triển khai để làm cơ sở cho việc phát triển, kiểm thử và bảo trì hệ thống.

---

2. SCOPE

Hệ thống Gather cung cấp các chức năng chính sau:

- Không gian 2D sử dụng Phaser, cho phép người dùng di chuyển avatar và hiển thị hiện diện theo phòng.
- Hệ thống chat gồm chat theo kênh, chat riêng (DM), chat nhóm, reaction, chỉnh sửa và xoá tin nhắn.

- Kênh thoại (voice channels) và gọi audio/video theo proximity hoặc theo kênh.
- Quản lý không gian (spaces/rooms), mời người tham gia và phân quyền cơ bản.
- Quản lý sự kiện (event), lịch, thư viện tài nguyên, upload file, thông báo và tìm kiếm.
- Trang quản trị (Admin) cho quản trị hệ thống.

Ngoài phạm vi của phiên bản hiện tại (future work):

- Thanh toán và subscription, multi-tenant enterprise billing.
- Ghi hình hoặc recording các cuộc gọi.
- Ứng dụng mobile native (iOS/Android).
- Mã hoá end-to-end cho media.

---

3. DEFINITIONS

Space/Room: Một không gian làm việc ảo bao gồm bản đồ, người dùng, kênh chat/voice và sự kiện.
User: Người dùng đã đăng nhập hệ thống.
Guest: Người dùng chưa đăng nhập, tham gia theo cấu hình cho phép.
Presence: Trạng thái hiện diện của người dùng (online/offline, vị trí, hướng, hoạt động).
Channel: Kênh chat hoặc voice trong một room.
SFU: Selective Forwarding Unit (mediasoup) để tối ưu truyền audio/video nhóm.
OTP: One-time password dùng để xác thực email hoặc đặt lại mật khẩu.

---

4. OVERVIEW

4.1 System Boundary

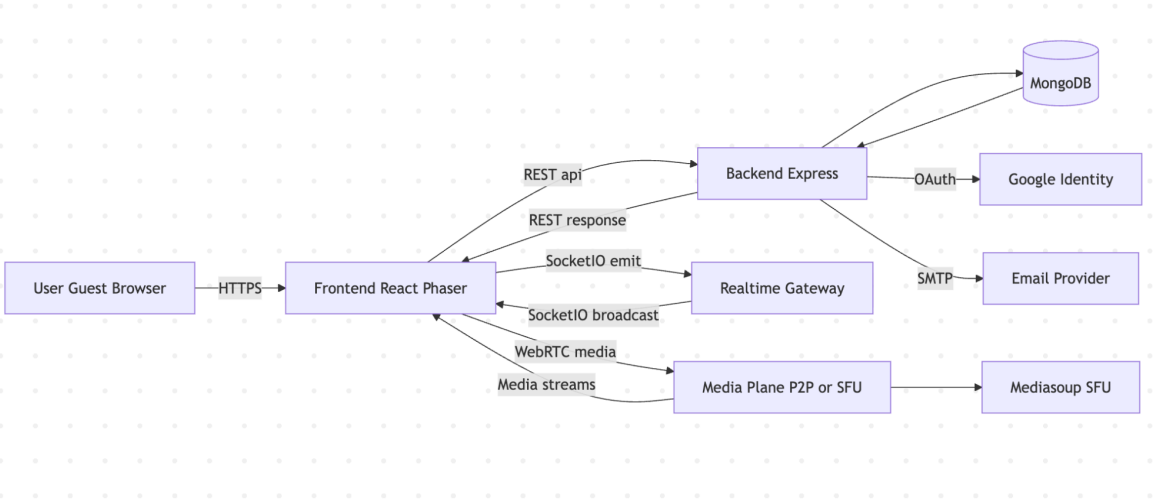
Hệ thống Gather bao gồm các thành phần sau:

- Frontend Web: Giao diện người dùng, game scene (Phaser), chat UI, lobby và dashboard.
- Backend API: REST API (Express) cho các chức năng auth, users, spaces, world, events, resources.
- Realtime Gateway: Socket.IO cho presence, chat và điều khiển realtime.
- Media Plane: WebRTC theo mô hình P2P hoặc SFU (mediasoup).
- Database: MongoDB dùng để lưu trữ dữ liệu bền vững.

4.2 External Entities

- Google Identity: Cung cấp đăng nhập OAuth.
- Email/SMPP Provider: Gửi OTP và email reset mật khẩu.
- STUN/TURN Server: Hỗ trợ NAT traversal cho WebRTC.

4.3 Interactions (context)



---

5. BUSINESS GOALS, CONSTRAINTS AND SUCCESS CRITERIA

5.1 Business Goals

- Cung cấp trải nghiệm virtual office trực quan, realtime và dễ sử dụng.
- Hỗ trợ giao tiếp đa kênh: text, audio và video.
- Hệ thống dễ vận hành, có quản trị, logging, rate limit và bảo mật cơ bản.

5.2 Business Constraints

- Chạy trên hạ tầng phổ thông (Node.js và MongoDB).
- Tối ưu mạng bằng cách giới hạn event spam, batch movement và giới hạn truy vấn.
- Ưu tiên an toàn hệ thống với validation, sanitize, rate limit và lockout.

5.3 Success Criteria (Definition of Done)

- Người dùng đăng ký và đăng nhập thành công, token hoạt động hợp lệ.
- Người dùng vào room và thấy presence realtime, chat hoạt động ổn định.
- Audio và video hoạt động trong điều kiện mạng thông thường.
- Admin có thể xem và quản lý room cơ bản.

---

6. FUNCTIONAL REQUIREMENTS

Quy ước:

- **Priority**: P0 (must), P1 (should), P2 (nice).

- **Complexity**: S (small), M (medium), L (large).

ID	Requirement	Priority	Complexity
FR-01	Đăng ký tài khoản bằng email/password (có thể kèm OTP)	P0	M
FR-02	Đăng nhập email/password, nhận access token	P0	M
FR-03	Đăng nhập bằng Google OAuth	P1	M
FR-04	Làm mới phiên (refresh) / quản lý sessions	P1	M
FR-05	Người dùng cập nhật hồ sơ (username, avatar…)	P1	S

FR-06	Xem danh sách spaces/rooms (public + của tôi)	P0	M
FR-07	Tạo room (private/public, name, roomId tuỳ chọn)	P0	M
FR-08	Xoá room (owner/admin) và cascade delete dữ liệu liên quan	P1	M
FR-09	Tạo invite link để mời vào room	P0	S
FR-10	Lobby: chọn room, kiểm tra camera/mic trước khi vào	P0	M
FR-11	Vào room: load map và objects theo roomId	P0	M
FR-12	Di chuyển avatar realtime và đồng bộ vị trí	P0	L
FR-13	Hiển thị user online/offline và lastSeen	P0	M

FR-14	Chat lịch sử theo room với phân trang/giới hạn	P0	M
FR-15	Chat theo kênh text (global channels)	P0	L
FR-16	Chat riêng (DM) 1–1 trong room	P1	M
FR-17	Group chat: tạo nhóm và nhắn trong nhóm	P1	M
FR-18	Reaction vào tin nhắn	P1	M
FR-19	Chỉnh sửa và xoá tin nhắn theo quyền	P1	M
FR-20	Voice channels: join/leave và danh sách user trong kênh	P0	L
FR-21	Audio/Video call theo proximity hoặc voice channel	P0	L

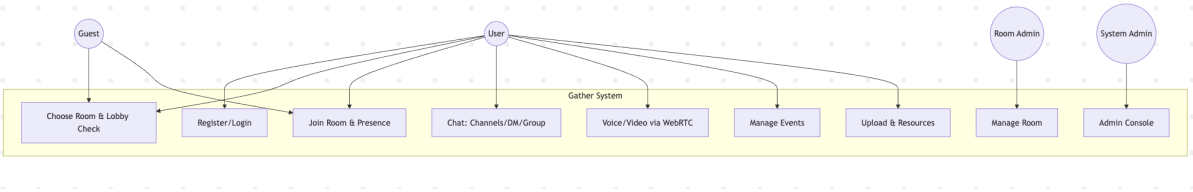
FR-22	Screen share (bật/tắt)	P2	L
FR-23	Events: tạo, sửa, xoá event và RSVP/attendance	P1	M
FR-24	Notifications: nhận và xem thông báo	P1	M
FR-25	Upload resources/attachments và truy cập qua URL	P1	M
FR-26	Search: tìm tin nhắn, tài nguyên, người dùng trong room	P1	M
FR-27	Admin: gán quyền admin, quản lý rooms và messages	P1	M
FR-28	Analytics: theo dõi thống kê cơ bản	P2	M

7. USE CASES

7.1 Actors:

- Guest: Tham gia lobby và room theo cấu hình.
- User: Sử dụng đầy đủ chức năng trong room.
- Room Admin: Quản trị room.
- System Admin: Quản trị toàn hệ thống.

7.2 Use case



7.3 Use Case: Register Account

UC 01- Register Account

Primary Actor: User
Precondition: Email chưa tồn tại trong hệ thống.

Main Flow:

1. User nhập thông tin đăng ký.
2. User nhập OTP (nếu được yêu cầu).
3. Hệ thống validate và tạo user, session.
4. Trả về access token, refresh token và thông tin user.

Postcondition: User truy cập được dashboard.

Acceptance Criteria:

- Mật khẩu đạt chính sách bảo mật.
- OTP sai hoặc hết hạn phải trả lỗi rõ ràng.

7.2 Use Case: Join Room

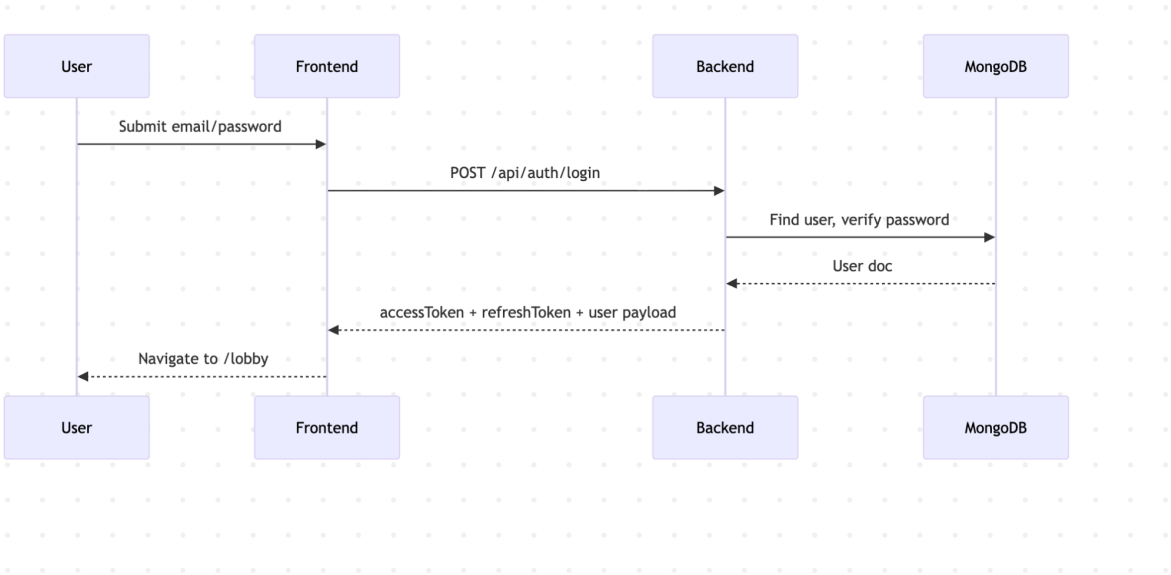
Primary Actor: User hoặc Guest
Precondition: Có roomId hợp lệ.

Main Flow:

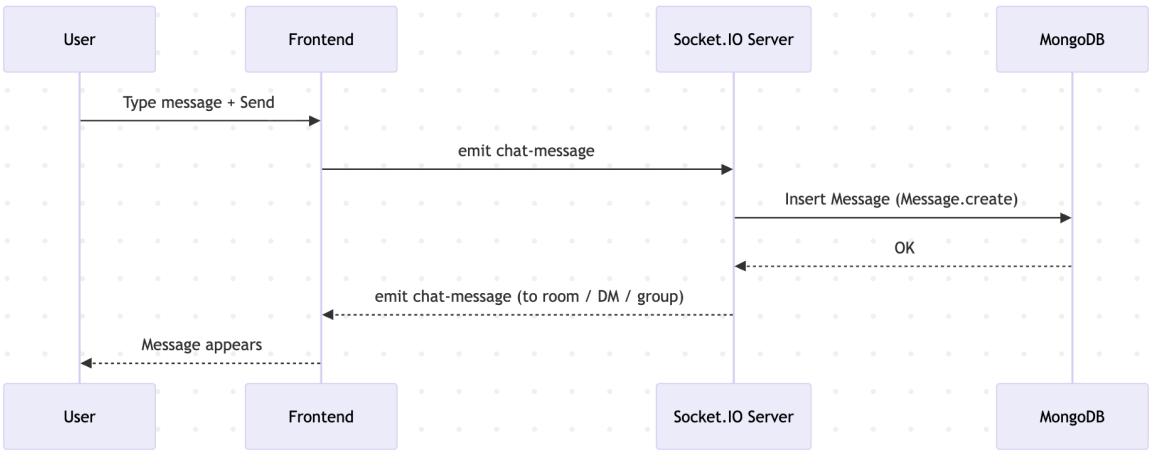
1. User vào lobby và chọn cấu hình camera/mic.
2. User nhấn Join.
3. Frontend kết nối Socket.IO và gửi presence.
4. Map và objects được load và hiển thị.

8. Key Flows

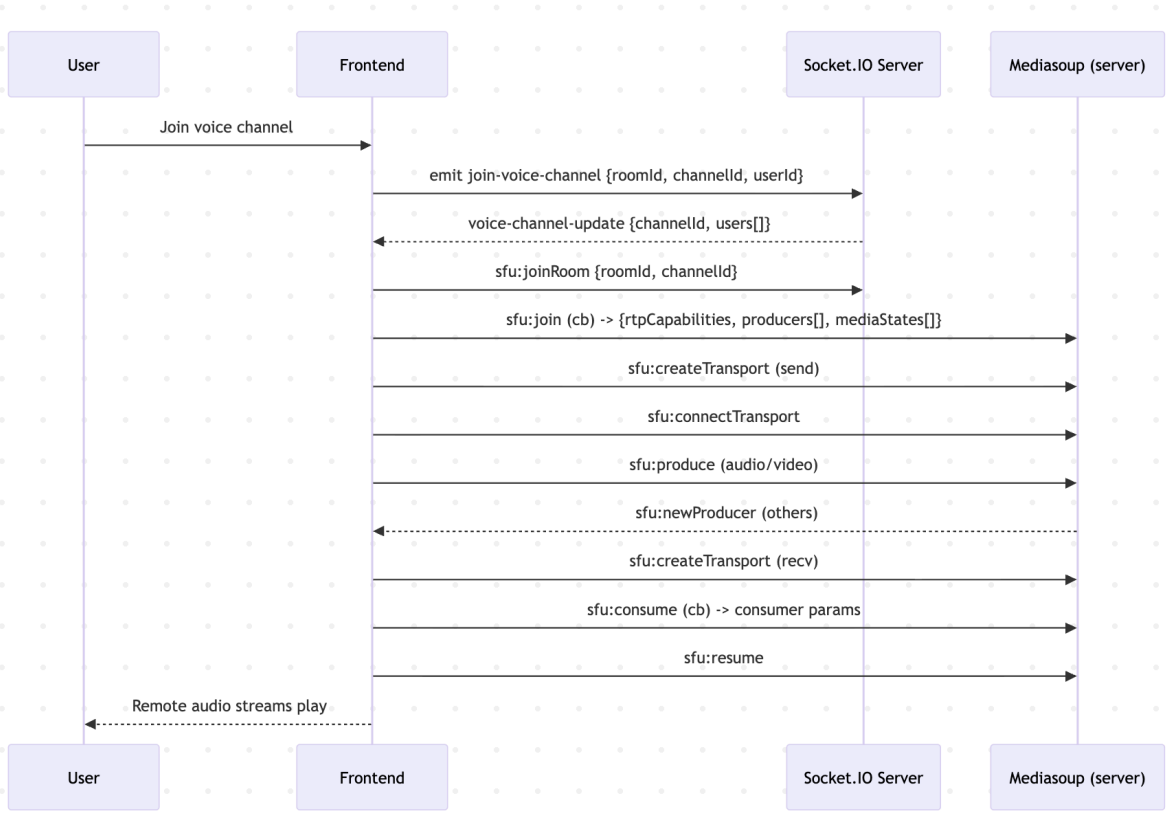
8.1 Auth (Login)



8.2 Realtime chat (send message)



8.3 Voice chat channel join (SFU mode)



9. NON-FUNCTIONAL REQUIREMENTS

9.1 Security

NFR-S1: Tất cả input từ client (body và query) phải được sanitize để giảm nguy cơ XSS và NoSQL Injection.
NFR-S2: Áp dụng rate limiting cho các endpoint nhạy cảm như xác thực và các API dùng chung.
NFR-S3: Tài khoản phải bị khoá tạm thời sau nhiều lần đăng nhập sai liên tiếp.
NFR-S4: Hệ thống phải áp dụng RBAC tối thiểu, phân biệt quyền admin và member cho các endpoint quản trị.

9.2 Performance

NFR-P1: Endpoint lấy lịch sử chat phải hỗ trợ phân trang và giới hạn số lượng bản ghi tối đa là 500.
NFR-P2: Dữ liệu movement và presence phải được throttling hoặc gửi theo batch để giảm tải mạng.

9.3 Availability and Reliability

NFR-A1: Hệ thống phải cung cấp health check endpoint để phục vụ monitoring.
NFR-A2: Cơ chế reconnect của Socket.IO không được gây hiện tượng nhấp nháy trạng thái online và offline (flicker).

9.4 Usability

NFR-U1: Lobby phải cho phép người dùng vào room ngay cả khi họ từ chối quyền truy cập camera hoặc microphone.
NFR-U2: Giao diện người dùng phải hỗ trợ responsive ở mức cơ bản cho các màn hình nhỏ.

9.5 Compatibility

NFR-C1: Hệ thống phải tương thích với các trình duyệt hiện đại như Chrome, Edge và Firefox.
NFR-C2: WebRTC chỉ được phép hoạt động trong môi trường production khi sử dụng HTTPS.

9.6 DATA VALIDATION RULES (HIGH LEVEL)

Username:

- Độ dài từ 3 đến 24 ký tự.
- Chỉ cho phép các ký tự chữ và số, dấu chấm, gạch dưới và gạch ngang.
- Không cho phép khoảng trắng ở đầu hoặc cuối.
- Không phân biệt hoa thường.
- Phải là duy nhất trong phạm vi một room khi user join.

Message content:

- Nội dung tin nhắn tối đa 2000 ký tự.
- Attachment (nếu có) phải chứa metadata tối thiểu gồm tên file, loại file và URL truy cập.

Identifiers (roomId, messageId, eventId…):

- Chỉ cho phép chuỗi ngắn an toàn (slug hoặc UUID).
- Không chấp nhận object hoặc cấu trúc phức tạp để tránh NoSQL Injection.

Pagination:

- Bắt buộc enforce giới hạn limit theo từng endpoint.
- Ví dụ: chat history tối đa 500 bản ghi, search tối đa 100 kết quả.

Upload:

- Kích thước tối đa 10MB cho mỗi file.
- Chỉ cho phép mime type nằm trong whitelist.
- Tuỳ triển khai có thể áp dụng quota theo user hoặc theo ngày để hạn chế abuse.

9.7 ERROR HANDLING STRATEGY (HIGH LEVEL)

REST APIs:

- Phân loại lỗi theo HTTP status code.
- Nhóm 4xx dành cho lỗi phía client.
- Nhóm 5xx dành cho lỗi phía server.
- Payload lỗi tối thiểu gồm trường message.
- Một số endpoint có thể trả thêm success = false để đồng bộ frontend.

Rate limiting (HTTP):

- Khi vượt quá giới hạn, server trả HTTP 429.
- Kèm theo các header X-RateLimit-Limit, X-RateLimit-Remaining và X-RateLimit-Reset.

Realtime (Socket.IO):

- Các lỗi nghiệp vụ được gửi qua event app-error.
- Payload lỗi chỉ chứa message.
- Ưu tiên cơ chế soft-fail, client vẫn sử dụng được các chức năng khác.

Media và WebRTC:

- Khi lỗi trong quá trình join, produce hoặc consume media, client cần xử lý fallback.

- Ví dụ fallback sang voice-only hoặc P2P khi SFU không khả dụng.

- Client phải hiển thị trạng thái lỗi rõ ràng cho người dùng.

10. SYSTEM CONSTRAINTS

SC-01: Backend phải chạy trên nền tảng Node.js phiên bản 20 trở lên.
SC-02: Cơ sở dữ liệu sử dụng MongoDB và phải có index phù hợp cho collection messages.
SC-03: CORS phải được cấu hình dựa trên giá trị CLIENT_URL.
SC-04: WebRTC phụ thuộc vào NAT traversal; môi trường production nên triển khai TURN server để tăng tỷ lệ kết nối thành công.
SC-05: Media và realtime tiêu tốn nhiều băng thông, do đó cần giới hạn số lượng peer hoặc bật chế độ SFU.

10.1 Deployment Environments

Development:

- Chạy local với Vite cho frontend.
- Backend sử dụng Node.js và Express.
- MongoDB local.
- Ưu tiên developer experience và debug.

Staging (khuyến nghị):

- Môi trường gần giống production.
- Dùng để kiểm thử WebRTC, TURN server, rate limit và CORS.
- Phục vụ QA và kiểm thử tích hợp.

Production:

- Bắt buộc sử dụng HTTPS.
- Cấu hình CORS chặt chẽ theo CLIENT_URL.
- Bật health check và logging để theo dõi hệ thống.
- Triển khai TURN server để tăng độ ổn định cho WebRTC.

10.2 KEY ENVIRONMENT VARIABLES (REFERENCE)

Core system variables:

- PORT
- NODE_ENV
- MONGODB_URI
- JWT_SECRET
- CLIENT_URL

OAuth và anti-bot (optional):

- GOOGLE_CLIENT_ID
- GOOGLE_CLIENT_SECRET
- GOOGLE_RECAPTCHA_SECRET_KEY

Email và OTP (optional):

- EMAIL_USER
- EMAIL_PASS

SFU và mediasoup tuning (khi sử dụng SFU):

- WEBRTC_LISTEN_IP
- WEBRTC_ANNOUNCED_IP
- WEBRTC_MIN_PORT
- WEBRTC_MAX_PORT
- WEBRTC_INITIAL_BITRATE
- MEDIASOUP_LOG_LEVEL
- MEDIASOUP_LOG_TAGS

---

11. BACKEND REQUIREMENTS

BR-01: Backend phải cung cấp REST API theo các nhóm route bao gồm auth, spaces, world, chat, users, uploads, resources, notifications, search và admin.
BR-02: Backend phải triển khai các Socket.IO event phục vụ presence, chat, voice và WebRTC signaling.
BR-03: Backend phải có hệ thống logging, middleware xử lý lỗi và handler cho các route không tồn tại (404).

---

12. FRONTEND REQUIREMENTS

FRN-01: Frontend phải sử dụng React và React Router, bao gồm các trang landing, login, register, dashboard, lobby, app và chat.
FRN-02: Frontend phải sử dụng Phaser để hiển thị scene 2D, bản đồ tile và chuyển động avatar.
FRN-03: Frontend phải có các context hoặc hook để quản lý Socket, Chat, WebRTC, Theme và Notifications.
FRN-04: Frontend phải sử dụng Tailwind CSS để xây dựng các UI component.

---

13. REALTIME AND MEDIA REQUIREMENTS

MR-01: Hệ thống phải hỗ trợ cả chế độ WebRTC P2P và SFU, có thể cấu hình thông qua biến môi trường.
MR-02: Người dùng phải có khả năng bật hoặc tắt audio, video và chia sẻ màn hình.
MR-03: Hệ thống phải quản lý tình trạng camera giữa nhiều tab trình duyệt để tránh xung đột tài nguyên.

14. TRACEABILITY (OPTIONAL)

Các yêu cầu trong tài liệu này phản ánh trực tiếp các nhóm module chính trong codebase hiện tại.

Frontend bao gồm các context như ChatContext và WebRTCContext, cùng các trang như Lobby, Spaces và ChatPage.
Backend bao gồm các route auth, spaces, world, chat và các module realtime trong server.

---

15. EXTERNAL INTERFACE REQUIREMENTS (DETAILED)

Mục tiêu của phần này là "đóng băng" contract giao tiếp bên ngoài để frontend, mobile (nếu có) và QA có thể dựa vào đó để kiểm thử và triển khai ổn định, không phụ thuộc vào chi tiết cài đặt nội bộ.

15.1 CONVENTIONS

15.1.1 Base URLs

REST API base path là /api
Health check endpoint bao gồm /health và /api/health
Upload file được phục vụ qua đường dẫn /api/uploads/filename và dùng chung host với API

15.1.2 Authentication

Cơ chế xác thực sử dụng HTTP Header Authorization với định dạng Bearer accessToken.

Access token:

- Định dạng JWT
- Payload tối thiểu bao gồm userId và type = access
- Thời gian sống là 15 phút

Refresh token:

- Là chuỗi ngẫu nhiên, không phải JWT
- Được lưu trong collection sessions
- Thời gian sống là 7 ngày

Một số endpoint cho phép optional authentication. Các endpoint này không bắt buộc token, tuy nhiên nếu request có token hợp lệ thì hệ thống sẽ gắn userId vào request để trả về kết quả phù hợp hơn.

15.1.3 Common Error Patterns

Nhiều endpoint REST trả lỗi theo định dạng HTTP status code kèm theo JSON có trường message.

Rate limit HTTP sử dụng status code 429, response body bao gồm:

- success = false
- message
- retryAfter

Các header đi kèm bao gồm:

- X-RateLimit-Limit
- X-RateLimit-Remaining
- X-RateLimit-Reset

Lỗi realtime qua Socket.IO được gửi bằng event app-error với payload chứa message.

15.1.4 Pagination

Hệ thống hỗ trợ hai dạng phân trang.

Dạng page và limit:

- Request gồm page và limit
- Response trả về pagination gồm page, limit, total và có thể có pages

Dạng skip và limit:

- Request gồm skip và limit
- Response trả về pagination gồm total, limit, skip và hasMore

---

15.2 REST API SPECIFICATION

15.2.1 Health

GET /health
Authentication: Không yêu cầu
Response thành công (200):

- status
- timestamp
- uptime
- database (connected hoặc disconnected)

GET /api/health
Authentication: Không yêu cầu
Response thành công (200):

- status = ok
- message = Server is running

15.2.2 Authentication APIs (/api/auth)

Các endpoint xác thực chịu sự kiểm soát của auth rate limiter.

POST /api/auth/register
Authentication: Không yêu cầu
Request body:

- username (không bắt buộc)
- email (bắt buộc)
- password (bắt buộc)
- otp (không bắt buộc)

Response thành công (201):

- accessToken
- refreshToken
- user (id, username, email, avatar, role)

Các lỗi có thể trả về:

- 400: input không hợp lệ, mật khẩu yếu, OTP sai hoặc hết hạn, user đã tồn tại
- 500: lỗi server

POST /api/auth/login

Authentication: Không yêu cầu

Request body:

- email
- password
- recaptchaToken (không bắt buộc)

Response thành công (200):

- accessToken
- refreshToken
- user (id, username, email, avatar, role)

Lỗi:

- 400: email thiếu hoặc không hợp lệ
- 401: sai thông tin đăng nhập
- 423: tài khoản bị khoá tạm thời
- 500: lỗi server

POST /api/auth/google

Authentication: Không yêu cầu

Request body:

- googleId
- email
- username (không bắt buộc)
- avatar (không bắt buộc)

Response thành công (200):

- accessToken
- refreshToken
- user

POST /api/auth/check-email
Authentication: Không yêu cầu
Request body:

- email

Response thành công (200):

- exists (true hoặc false)

POST /api/auth/send-otp
Authentication: Không yêu cầu
Request body:

- email

Response thành công (200):

- message
- expiresIn (giây)

POST /api/auth/forgot-password
Authentication: Không yêu cầu
Request body:

- email

Response luôn trả thành công (200) để tránh email enumeration.

POST /api/auth/verify-otp

Authentication: Không yêu cầu

Request body:

- email
- otp

Response thành công (200):

- message xác nhận OTP hợp lệ

POST /api/auth/reset-password
Authentication: Không yêu cầu
Request body:

- email
- otp
- newPassword

Response thành công (200):

- message xác nhận đổi mật khẩu

POST /api/auth/refresh
Authentication: Không yêu cầu
Request body:

- refreshToken

Response thành công (200):

- accessToken
- refreshToken (hiện tại không rotate)

POST /api/auth/logout
Authentication: Không yêu cầu
Request body:

- refreshToken

Response thành công (200):

- message xác nhận logout

POST /api/auth/logout-all
Authentication: Yêu cầu Bearer token
Response thành công (200):

- message
- deletedSessions

GET /api/auth/sessions
Authentication: Yêu cầu Bearer token
Response thành công (200):

- danh sách session (không bao gồm refreshToken)

DELETE /api/auth/sessions/{sessionId}
Authentication: Yêu cầu Bearer token
Response thành công (200):

- message xác nhận logout thiết bị

15.2.3 User APIs (/api/users)

GET /api/users/profile
Authentication: Yêu cầu Bearer token
Response thành công (200):

- id
- username
- email
- avatar
- status
- role
- displayName
- avatarConfig
- profileColor

PUT /api/users/profile

Authentication: Yêu cầu Bearer token

Request body:

- username (không bắt buộc)
- avatar (không bắt buộc)
- status (không bắt buộc)

Response thành công (200):

- id
- username
- email
- avatar
- status

POST /api/users/avatar

Authentication: Yêu cầu Bearer token

Request body:

- displayName (không bắt buộc)
- avatarConfig (không bắt buộc)

Response thành công (200):

- id
- username
- displayName
- avatarConfig
- avatar
- message

GET /api/users/settings

Authentication: Không yêu cầu

Response hiện tại trả về object rỗng (placeholder).

GET /api/users/{userId}
Authentication: Không yêu cầu
Response thành công (200):

- thông tin public của user (không bao gồm password và googleId)

15.2.4 Spaces, Rooms, Events và Templates

(GET, POST, PUT, DELETE cho spaces, events, templates được giữ nguyên logic như codebase hiện tại; có thể bổ sung auth hoặc phân quyền nếu yêu cầu sản phẩm thay đổi.)

15.3 SOCKET.IO REALTIME CONTRACT

15.3.1 Common Events

Event app-error: Server gửi về client khi có lỗi, payload chứa message.
Event room-full: Server gửi khi room vượt quá số người cho phép.

15.3.2 Presence và Room Lifecycle

Client gửi event user-join với thông tin userId, username và roomId.
Server kiểm tra username, capacity và trạng thái room, sau đó cập nhật RoomMember và broadcast danh sách user trong room.

Các event broadcast bao gồm:

- user-joined

- room-users
- allPlayersPositions
- room-info

Khi client disconnect, server debounce trạng thái offline trong 5 giây để tránh flicker.

15.3.3 Movement

Client gửi playerMovement khi di chuyển.
Server broadcast playerMoved cho các client khác.
Server định kỳ gửi allPlayersPositions theo batch mỗi 500ms.

15.3.4 Reactions

Client gửi reaction với emoji.
Server broadcast reaction kèm userId và timestamp.

15.3.5 Whiteboard

Client gửi whiteboard-draw.
Server forward dữ liệu và gắn userId, username trước khi broadcast.

15.3.6 Chat Protocol

Client gửi chat-message với nội dung, loại chat và target phù hợp.
Server kiểm tra rate limit, spam, sau đó lưu message vào database và broadcast đến đúng đối tượng nhận.

Server hỗ trợ reaction, edit và delete message với các giới hạn về quyền và tần suất.

15.3.7 Voice Channels

Client gửi join-voice-channel để tham gia kênh thoại.
Server đảm bảo mỗi user chỉ ở một voice channel và giới hạn số user trong kênh.
Server broadcast voice-channel-update khi có thay đổi.

15.3.8 WebRTC Signaling (P2P)

Client gửi offer, answer và ice-candidate thông qua server.
Server chỉ đóng vai trò forward signaling giữa các peer.

15.4 SFU SOCKET CONTRACT

15.4.1 Socket Room Key

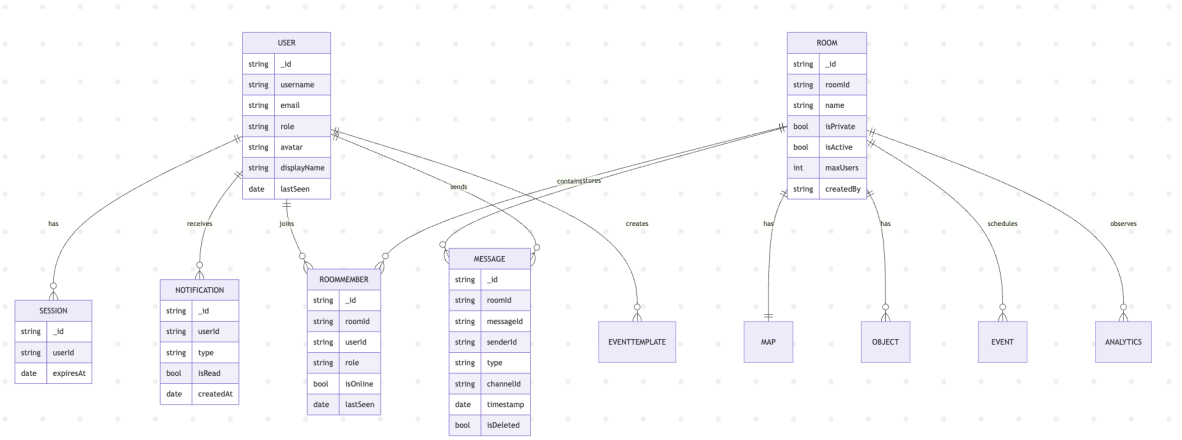
Mỗi SFU room sử dụng key theo định dạng roomId:channelId.
Client phải gọi joinRoom để tham gia room này trước khi nhận broadcast.

15.4.2 SFU Events

Client gọi sfu join để nhận rtpCapabilities và danh sách producer hiện có.
Client tạo transport, connect transport, produce và consume media thông qua các event có callback acknowledgement.
Server broadcast media state và producer mới cho các client khác trong cùng room.

16. Data model (Mongodb / Mongoose)

16.1 ERD



16.2 DATA DICTIONARY (SUMMARY)

ENTITY: User

Unique constraints:

- username (unique)
- email (unique, sparse)
- googleId (unique, sparse)

Core fields:

- username
- email (optional)
- password (optional)
- role
- avatar
- displayName (optional)
- avatarConfig
- avatarColor
- status
- currentRoom
- position
- lastSeen

ENTITY: Room

Unique constraint:

- roomId

Fields:

- name
- description
- maxUsers (default 20, minimum 20)
- isPrivate
- isActive
- createdBy

ENTITY: RoomMember

Unique compound key:

- roomId + userId

Fields:

- username
- avatar
- joinedAt
- lastSeen
- isOnline
- role

ENTITY: Message

Unique compound key:

- roomId + messageId

Fields:

- senderId

- senderName
- type
- content
- targetUserId (optional)
- groupId (optional)
- channelId (optional)
- recipients (array)
- timestamp
- editedAt (optional)
- isDeleted
- deletedAt (optional)
- deletedBy (optional)
- replyTo (optional)
- reactions (array)
- attachments (array)

ENTITY: Map

Unique constraint:

- mapId

Fields:

- roomId (indexed)
- width
- height
- tileSize
- tiles (2D array)
- collision (2D array)
- zones (array)
- backgroundImage (optional)

ENTITY: Object

Unique constraint:

- objectId

Fields:

- roomId
- type
- name
- position
- properties (url, content, imageUrl, documentUrl, width, height, allowFullscreen)
- isActive

ENTITY: Event

Unique constraint:

- eventId

Fields:

- roomId
- title
- description
- startTime
- endTime
- createdBy
- attendees (array)
- location
- maxParticipants (optional)
- isRecurring
- recurrencePattern (optional)
- parentEventId (optional)
- templateId (optional)
- reminders (array)
- attendance

ENTITY: EventTemplate

Fields:

- name
- duration
- defaultReminders (array)

- category (optional)
- createdBy
- isPublic
- usageCount

ENTITY: Notification

Fields:

- userId
- type
- title
- message
- link (optional)
- relatedId (optional)
- isRead
- readAt (optional)

ENTITY: Resource

Indexes:

- Text index trên title và author

Fields:

- title
- author (optional)
- contentType
- url (optional)
- thumbnailUrl (optional)
- description

- createdBy (optional)
- isApproved

ENTITY: Session

Unique constraint:

- refreshToken (lưu server-side)

TTL:

- expiresAt (expireAfterSeconds = 0)

Fields:

- userId
- refreshToken
- expiresAt

ENTITY: OtpCode

TTL index:

- expiresAt (hết hạn khoảng 15 phút)

Fields:

- email
- code
- purpose
- used
- expiresAt

ENTITY: Analytics

TTL index:

- Tự động xoá sau 90 ngày

Fields:

- eventType
- eventName
- sessionId
- userId (optional)
- timestamp
- properties
- ipAddress (optional)
- userAgent (optional)

---

17. DETAILED USE CASES (TESTABLE)

UC-AUTH-01: LOGIN AND SESSION CREATION

Preconditions:

- User tồn tại trong hệ thống
- Tài khoản chưa bị khoá

Main flow:

1. Client gửi request POST /api/auth/login.
2. Server xác thực thông tin đăng nhập, reset bộ đếm lockout và cập nhật lastSeen.
3. Server tạo bản ghi Session và trả về accessToken cùng refreshToken.

Alternate flows:

- A1: Sai mật khẩu, server trả HTTP 401 kèm remainingAttempts.
- A2: Quá nhiều lần đăng nhập sai, server trả HTTP 423 và khoá tài khoản tạm thời.
- A3: reCAPTCHA được bật nhưng thất bại, server trả HTTP 400.

UC-ROOM-01: JOIN ROOM (SOCKET)

Preconditions:

- Room đang active
- Chưa vượt quá capacity
- Username là duy nhất trong room

Main flow:

1. Client kết nối Socket.IO.
2. Client emit event user-join.
3. Server upsert Room và RoomMember, sau đó broadcast trạng thái room.
4. Client render map và presence.

Error cases:

- E1: Username trùng, server emit app-error.
- E2: Room full, server emit room-full.
- E3: Room bị disable, server emit app-error.

UC-CHAT-01: SEND MESSAGE (PERSISTED)

Preconditions:

- User đã join room thành công

Main flow:

1. Client emit chat-message.
2. Server kiểm tra spam, rate limit và lưu Message vào database.
3. Server broadcast message đến đúng recipients theo loại chat.

UC-VOICE-01: JOIN VOICE CHANNEL WITH SFU

Preconditions:

- User đã join room
- Voice channel chưa đầy

Main flow:

1. Client emit join-voice-channel.
2. Client join socket room thông qua sfu joinRoom.
3. Client gọi sfu join để nhận rtpCapabilities và danh sách producer hiện có.
4. Client tạo transport, publish track và consume remote streams.

Alternate flows:

- A1: Channel full, server emit voice-channel-full.
- A2: Không thể consume stream, server trả ack lỗi cho sfu consume.

18. TRACEABILITY MATRIX (FR TO IMPLEMENTATION)

Requirement	Backend REST	Backend Socket	Models
FR-01, FR-02, FR-04	/api/auth	Không áp dụng	User, Session, OtpCode
FR-05	/api/users	Không áp dụng	User
FR-06, FR-07, FR-08, FR-09	/api/spaces, /api/rooms/{roomId}	user-join, room-users, room-info	Room, RoomMember, Map, Object, Event, Message
FR-12, FR-13	Không áp dụng	playerMovement, allPlayersPositions, user-left	RoomMember
FR-14	/api/chat/history/{room Id}	Không áp dụng	Message

FR-15 đến FR-19	Không áp dụng	chat-message, message-reaction, edit-message, delete-message	Message
FR-20, FR-21	Không áp dụng	join-voice-channel, voice-channel-update, webrtc signaling, sfu events	SFU (in-memory)
FR-23	/api/spaces events	Không áp dụng	Event, EventTemplate
FR-24	/api/notifications	Không áp dụng	Notification
FR-25	/api/uploads, /api/resources	Không áp dụng	Resource
FR-26	/api/search, /api/chat/search	Không áp dụng	User, Message, Resource
FR-27	/api/admin	Không áp dụng	Room, RoomMember, User, Resource, Message

FR-28	/api/analytics	Không áp dụng	Analytics

19. KNOWN GAPS AND IMPLEMENTATION NOTES

Frontend hiện có thể đang sử dụng data.token trong khi backend trả về accessToken và refreshToken. Tài liệu này chuẩn hoá theo backend contract hiện tại.

Một số endpoint REST như world objects và uploads hiện chưa yêu cầu authentication. Nếu mục tiêu sản phẩm cần kiểm soát ghi dữ liệu, cần bổ sung cơ chế authentication và authorization phù hợp cho các route này.