🎮 Gaming Hub Platform

Complete System Design, Architecture & Execution Document (MERN Stack)

---

1. Project Overview

Vision

A centralized, genre-based gaming hub that aggregates gaming news, genre communities, game information, statistics, and future in-house games — built using the MERN stack with scalability and real-world usability in mind.

Target Audience

- Gamers (casual → hardcore)
- Indie game developers
- Gaming communities

Core Principles

- Genre-first organization
- Clean & structured community interaction
- Anti-toxicity by design
- Scalable architecture
- Portfolio + startup-ready

---

2. Technology Stack (Locked)

Frontend

- React.js
- Tailwind CSS
- React Router
- Axios
- Socket.IO Client

Backend

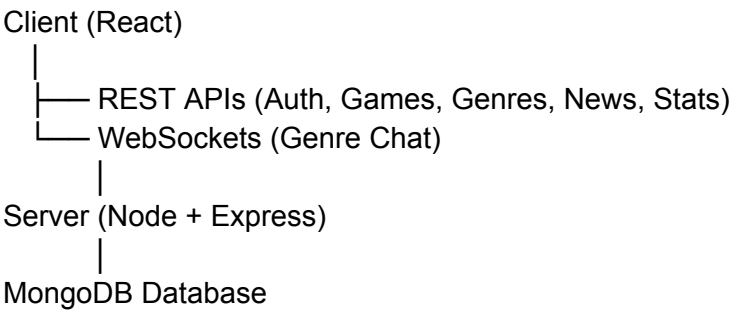
- Node.js
- Express.js
- JWT Authentication
- Socket.IO

Database

- MongoDB with Mongoose

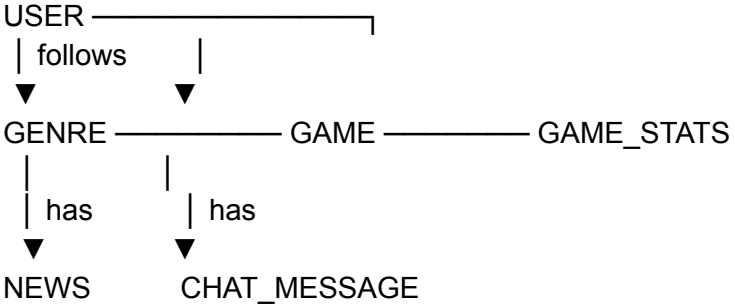
---

3. System Architecture



---

4. ER Diagram (Textual Representation)



---

5. Entity Definitions

User

_id (PK)

username
email
password
role (user | moderator | admin)
avatar
joinedAt
followedGenres[]
followedGames[]

Genre

_id (PK)
name
description
bannerImage
createdAt

Game

_id (PK)
title
genreId (FK)
description
developer
platforms
releaseStatus
bannerImage
screenshots[]
createdAt

GameStats

_id (PK)
gameId (FK)
activeUsers
avgPlaytime
rating
engagementScore
updatedAt

News

_id (PK)

title
content
genreId (FK)
gameId (optional FK)
author
createdAt

ChatMessage

_id (PK)
genreId (FK)
channel
userId (FK)
message
toxicityScore
upvotes
createdAt

---

6. Application Pages & UI Structure

Home Page

- Latest gaming news
- Trending genres
- Featured games
- Recent discussions

Genre Page

Tabs:

- Overview
- Games
- News
- Chat
- Stats

Game Detail Page

- Overview
- Community

- Stats
- Patch Notes

Chat UI

- Channel list (left)
- Messages (center)
- Active users (right)

---

7. Authentication & Authorization

Auth Flow

User → Login/Register → Backend
→ JWT issued (HTTP-only cookie)
→ Access protected routes

Roles

- User
- Moderator
- Admin

---

8. API Design

Auth APIs

- POST /api/auth/register
- POST /api/auth/login
- POST /api/auth/logout

Genre APIs

- GET /api/genres
- GET /api/genres/:id

Game APIs

- GET /api/games

- GET /api/games/:id

News APIs

- GET /api/news
- POST /api/news (Admin)

Stats APIs

- GET /api/stats/game/:id

---

9. Real-Time Chat System Design

Chat Structure

Genre
├── #general
├── #recommendations
├── #bugs-feedback
├── #lore-discussion

Socket Events

- joinGenre
- sendMessage
- receiveMessage
- leaveGenre

---

10. Anti-Toxic & Scalable Chat Design

Anti-Toxic Measures

- Rate limiting (messages/minute)
- Profanity filter
- Toxicity score per message
- Auto-hide high-toxicity messages

Reputation System

- Upvotes increase visibility
- Downvotes reduce trust
- New users have limited permissions

Moderation Tools

- Mute user
- Delete message
- Lock channel
- Pin verified messages

---

11. System Flow Diagrams

Home Flow

Client → Fetch news, genres, games → Render UI

Genre Flow

User selects genre → Fetch genre data → Join chat socket

Chat Flow

User sends message → Toxicity check → Store → Broadcast

---

12. Folder Structure

Backend

server/
├── config/
├── models/
├── controllers/
├── routes/
├── sockets/
├── middleware/
├── server.js

Frontend

client/
├── src/components/
├── src/pages/
├── src/context/
├── src/services/
├── src/sockets/
├── App.jsx

---

13. Task Board (Team Division)

Frontend Team

- UI components
- Routing
- API integration
- Responsive layouts

Backend Team

- Auth system
- APIs
- Database models
- Role-based access

Chat System Team

- Socket.IO setup
- Moderation logic
- Message persistence

Admin Team

- Admin panel
- Content moderation
- User management

---

14. Future Enhancements

- In-house Unity games integration
- Advanced analytics
- Recommendation engine
- Game launcher integration

---

15. Final Notes

This document represents a **complete, industry-grade system design** suitable for:

- Job interviews
- Team execution
- Startup pitching
- Long-term scalability

END OF DOCUMENT