GuildSpaces!

A Game-Centered Social Platform

Group 206
Benjamin Foore • Xu Li
Rick Xie • Jiacheng Kong
Spring 2026

What is GuildSpaces!?

GuildSpaces! offers browser-based games, extensive social features, and all of the tools you need to start your own thriving community. Think of it as a centralized space for players to discover others, organize groups, or share gaming content.

The GuildSpaces! experience was designed around three key pillars:

- **Community & Moderation**
- **Social Networking**
- **Engagement & Content**

Pillar 1: Community & Moderation

- Game-specific guilds with banner images, descriptions, member counts
- Rich post creation: titles, body (rich text), tags, spoiler toggle, drafts, polls, media attachments
- Tiered role system: Owner → Moderator → Member with canModerate() utility
- Auto-ban rules: keyword/phrase matching with configurable scope & action
- AI-assisted moderation: in-process Naive Bayes classifier flags toxic content for review
- Pin posts, delete content, ban/unban users from guild pages

Pillar 2: Social Networking

- User search by username + friend request system with notifications
- Real-time 1-to-1 direct messaging with full history (Socket.IO)
- Group chats: create, add/remove friends, leave groups
- Block list management: blocks remove friendship & prevent all contact
- Private game rooms: password-protected, friends bypass password
- Player profiles: XP, 10-tier leveling, medals, per-game win/loss stats

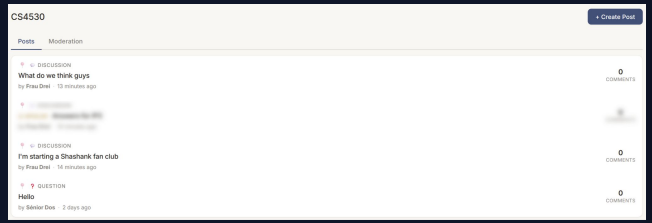
Pillar 3: Engagement & Content

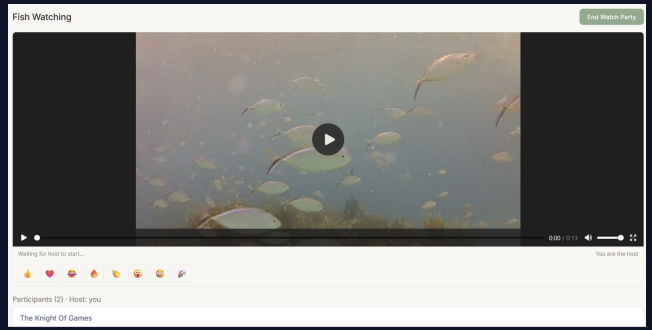
- Polls with customizable options, expiration, one-vote-per-user enforcement
- Like/Dislike reactions with instant toggle and count updates
- Nested threaded comments with up to 3 levels of indentation
- Built-in HTML5 media player: play/pause, volume, seek, fullscreen
- Custom video thumbnails (auto-generated or user-uploaded)
- Synchronized watch parties: host controls playback for all participants, late-joiner auto-sync, emoji reactions & chat sidebar

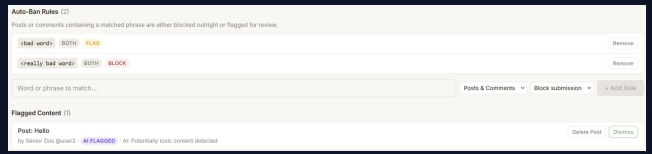
Results & Testing

40+	48	712	90%+
User Stories Completed	E2E Tests Passed	Unit Tests Passed	Code Coverage

Frontend: Playwright (Chromium) • Backend: Vitest • CI coverage reports







What's Next?

- Administrators can set up a tiered moderation system, with the page owner at the top and fully customizable permission levels below.
- A Preview Mode lets administrators see how a new banner will appear on both desktop and mobile before publishing.
- Your friends list shows which game pages each friend has joined, making it easy to spot shared interests.
- You can view a player's stats and rankings from external games through connected third-party platforms.

Agile Process & Sprints

Sprint 0 (Feb 14–Feb 27)
Repository setup, video hosting research, initial data model design. Completed as planned.

Sprint 1 (Feb 28–Mar 13)
Core post creation (frontend + backend), friend/DM system started. Shortened by spring break.

Sprint 2 (Mar 14–Mar 27)
Group chat, private game rooms, game page layout delivered. Media streaming infrastructure took longer than estimated.

Sprint 3 (Mar 28–Apr 10)
Carried extra scope: polls, likes, comments, media player, watch parties, friend features, AI moderation, player profiles all completed.

Technology Stack

Frontend	React, TypeScript, Vite, Socket.IO Client
Backend	Node.js, Express, Socket.IO, Zod Validation
Database	MongoDB (optional), Keyv in-memory store
AI/ML	Naive Bayes toxic content classifier (in-process)
Testing	Playwright (E2E), Vitest (unit), 90%+ coverage
Deployment	Render (deployed), monorepo with shared types

Key Takeaways

- Shared types package between client & server prevents type drift and hard-to-debug bugs
- Centralize auth in middleware instead of per-controller checkAuth() calls
- Hash passwords before production (bcrypt/argon2)
- Return error payloads from Socket.IO handlers for meaningful client feedback
- Distribute complex work (User Story 3) across sprints to avoid overloading the final sprint

Architecture

Three-layer architecture: Controller → Service → Repository

React (TypeScript)	Node.js Express	Socket.IO Real-time	MongoDB / Keyv Store

Key Design Decisions:

- Shared ChatRecord container across DMs, group chats, game rooms & watch parties
- Friend-gated social actions enforced at service layer
- Server-authoritative watch party sync with canonical state