PROJECT DOCUMENTATION

Project Name	GOAT - Social Debate Platform
Official Website	https://www.goat.uz
Version	1.0 (12.08.2025) Stable
Project Domain	Social Media, Online Community, Digital Debate
Project Status	Active Development & Maintenance
Lead Developer	Ismoiljon Umarov
Primary Repository	https://github.com/umaarov/goat-dev
Issue Tracker / Bug Reports	https://github.com/umaarov/goat-dev/issues
License	MIT License
Primary Contact (Technical Inquiries)	hs.umarov21@gmail.com
Professional Profile (Lead Developer)	https://linkedin.com/in/umaarov

Project Summary

GOAT.uz is a mission-critical social platform, engineered from the ground up for scalability, high performance, and robust security. It is not an iteration of existing social media; it is a fundamental re-imagining of the digital forum.

The project's singular goal is a successful and self-sustaining result: a premier destination for intelligent discourse. It is architected to perform its duties flawlessly within its digital environment, providing a stable and superior experience for its users. Its development philosophy is one of innovation, designed to solve complex R&D tasks by taking on and achieving the aggressive goals that other platforms ignore.

GOAT.uz was architected in a planned and organized manner. While built to handle extreme user load, its core philosophy rejects 'emergency mode' development and rushed, stress-based feature implementation. Every component is methodically designed and justified, avoiding the chaotic, visibility-driven development that plagues many online services.

The platform demands thoughtful contribution from its users, and in turn, it is designed to support, compensate, and appreciate that work. It provides a rewarding ecosystem where intellectual investment is recognized and valued, moving beyond the superficial metrics of mainstream social media.

INTRODUCTION

Purpose of the Project

The primary purpose of GOAT.uz is to create a premier online destination for structured, high-quality debate. It aims to elevate online discourse by providing tools that foster intelligent conversation, reward thoughtful engagement, and ensure a fair, well-moderated environment.

Problem It Solves

The project directly addresses critical failures in modern social media platforms:

- **Low-Quality Discourse:** Combats spam, toxicity, and low-effort content through robust AI-powered moderation.
- **Ineffective Content Discovery:** Solves the problem of finding relevant discussions with a sophisticated hybrid search engine that understands user intent.
- **Lack of Meaningful Engagement:** Replaces generic "likes" with a rewarding user experience, exemplified by unique, interactive 3D achievement badges.
- **Performance Bottlenecks:** Overcomes the limitations of traditional web stacks with a modern, performance-first architecture.

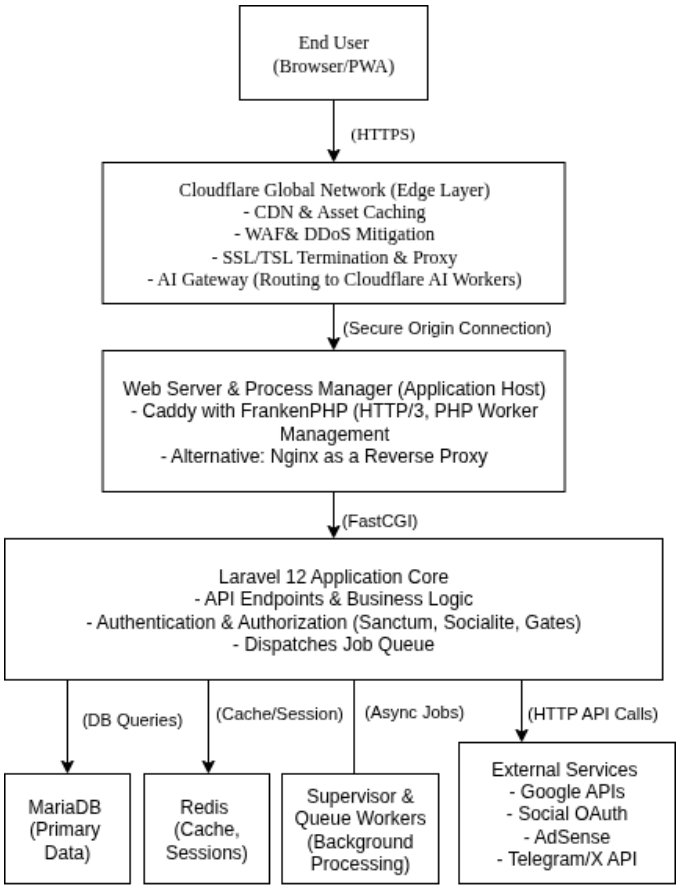
Scope & Limitations

The project directly addresses critical failures in modern social media platforms:

- **In Scope:** The project is a web-first platform, delivered as a Progressive Web App (PWA) for a seamless desktop and mobile experience. Its core focus is on asynchronous, text-based debate and user interaction.
- **Out of Scope:** Native iOS/Android applications are not part of the current scope.

SYSTEM ARCHITECTURE

GOAT.uz is designed as a monolithic application with a decoupled frontend, leveraging a containerized, service-oriented architecture for scalability and maintainability.



System Components

- **Edge & Security Layer (Cloudflare):** This is the platform's first line of defense, operating at the network edge. It manages all DNS and acts as the public-facing entry point, deploying an enterprise-grade Web Application Firewall (WAF), sophisticated bot detection, and rate limiting to filter malicious traffic before it reaches the origin server. It provides comprehensive DDoS mitigation, serves all static assets from its global CDN to reduce latency, and handles SSL/TLS termination with modern protocols.
- **Web & Application Delivery Layer (Origin Host):** This layer receives secure traffic from the edge network and serves the application. It utilizes FrankenPHP with the Caddy web server for high-throughput, native HTTP/3 support. In horizontally scaled deployments, Nginx can be used as a robust reverse proxy for load balancing. Process control for background workers is managed by Supervisor, ensuring high availability of asynchronous services.
- **Application Core (Laravel 12):** The central nervous system of the platform, handling all business logic. It manages API routes, validates requests, and executes services through a dependency-injected container. Authentication is handled by Laravel Sanctum using a secure Refresh Token system, while authorization is managed by fine-grained Gates. For any long-running task, the core dispatches a job to the asynchronous processing layer to ensure a non-blocking response.
- **Client-Side Processing (Browser/PWA & Web Workers):** The user-facing application is a responsive frontend built with Alpine.js and Tailwind CSS. To guarantee a fluid UI, computationally intensive tasks are offloaded from the main thread. This is most critical for the Three.js 3D rendering engine, where complex geometry calculations and rendering loops run inside a dedicated Web Worker, ensuring the main application remains fast and responsive at all times.
- **Data & Persistence Layer (MariaDB & Redis):** This layer manages the application's state. MariaDB serves as the primary relational database and single source of truth for all structured data. Redis is leveraged as a high-speed, multi-purpose in-memory store for application caching, user session management, and as the message broker for the asynchronous job queue, dramatically reducing database load.
- **Asynchronous Processing Layer (Queue & Supervisor):** To maintain a fast user experience, long-running tasks are executed in the background. The Application Core pushes jobs onto a Redis queue. Dedicated, long-running PHP processes, monitored and managed by Supervisor, pull jobs from this queue and execute them independently of the user's request-response cycle.
- **External Services Integration Layer:** The Laravel application acts as a secure gateway and abstraction layer for all third-party API interactions. It securely manages credentials, handles the OAuth2 dance, and centralizes error handling for services including Google Gemini, Cloudflare AI, social authentication providers (Google, X, Telegram, GitHub), and monetization platforms (AdSense, Ezoic).

TECHNOLOGY STACK

The project utilizes a curated selection of modern technologies chosen for performance, scalability, and developer experience.

Languages, Runtimes & Markup: PHP 8.3+, JavaScript (ES6+), C/C++, Bash, Shell, Blade Templating Engine, HTML, CSS, Markdown, JSON, XML, YAML, WebAssembly, GLSL, Dockerfile, HTTP/2, HTTP/3, TCP

Backend Ecosystem: Laravel 12, Eloquent ORM, Query Builder, Sanctum, Socialite, Laravel Gates, Bcrypt, Signed URLs, TrustProxies, Pusher, GuzzleHttp, Intervention, Carbon, Predis, CSP, ResponseCache, Sitemap, nlohmann/json, Emscripten SDK, stb, Queued Mailables, Eloquent Observers, Accessors & Attribute Casting, Custom Audit Logging Channels, Task Scheduler

Frontend Ecosystem: Alpine.js, Three.js, jQuery, Tailwind CSS, Vite, Axios, Fetch API, Cropper.js, lil-gui, stats.js, Web Workers, OffscreenCanvas, Web Share API, IntersectionObserver API, FileReader API, DataTransfer API

Infrastructure & DevOps: FrankenPHP, Caddy, Nginx, Docker, Docker Compose, Docker Multi-stage Builds, Supervisor, Debian, Git, Github, Makefile, Concurrently

Database & Caching: MariaDB, Redis

AI, APIs & External Services: Google Gemini API, Stable Diffusion, Cloudflare AI Workers API, Google API Client, Google AdSense, Google Search Console API, Google/Bing Ping API, Google OAuth, Github OAuth, X OAuth, Telegram Login Widget, Instagram Graph API, Telegram Bot API, X API

Quality Assurance & Developer Tooling: SonarCloud, SonarQube, Snyk, Trivy, ESLint, Composer, npm, Barryvh Debugbar, Pulse, netdebug (custom module)

Architectural Concepts & Patterns: MVC, Service Container (Dependency Injection), Reverse Proxy, Static File Serving, Progressive Web App (PWA), Hybrid Search (BM25, Vector, Levenshtein, Soundex), OpenSearch, Generative AI Content Creation, i18n (Internationalization), Infinite Scrolling, Real-time Typing Indicator, HTML Email Templating, Account Deactivation, Initial Server-Side Render with Client-side Hydration (SSR), Off-thread Rendering, LQIP, Shimmer, Debouncing, Asynchronous CSS, base64, Inline SVG, Asset Preloading, HTTP Security Headers, Refresh Token System, Rate Limiting, SEO (JSON-LD, Hreflang, Canonical URLs), Google Image Sitemap Extension, Custom Github-based Feature Flags, 3D Post-Processing (EffectComposer, Bloom, SSR, SSAO, God Rays, Chromatic Aberration), HDRI Lighting, ACESFilmicToneMapping, Simplex Noise

USAGE GUIDE

A comprehensive overview of the features and user flows for the GOAT.uz platform. It is intended for end-users, administrators, and stakeholders to understand the full spectrum of functionality, from initial registration to advanced account management and community interaction.

- **For Users:** A complete handbook for mastering the platform, detailing the full spectrum of functionality from initial registration to advanced account management. It empowers users to fully leverage the platform's powerful tools for a richer, more impactful experience.
- **For Developers:** An essential reference for understanding the intended functionality and user-facing behavior of the system. It documents the complete feature set and the administrative CLI, providing the necessary context for development, testing, and maintenance.
- **For Stakeholders:** A transparent, top-down view of the platform's delivered capabilities. It demonstrates the depth and breadth of the user experience and administrative controls, showcasing the project's value and the successful execution of its mission.

1. User Onboarding & Authentication

The platform offers a secure and flexible authentication system, allowing users to join and access their accounts through multiple methods.

1.1 Standard Registration & Login

- **Local Registration:** New users can create an account using a unique username, email address, and a secure password. The system requires email verification to activate the account.
- **Login:** Registered users can sign in using their username or email and password. Login attempts are rate-limited to prevent brute-force attacks.

1.2. Social Authentication (OAuth2)

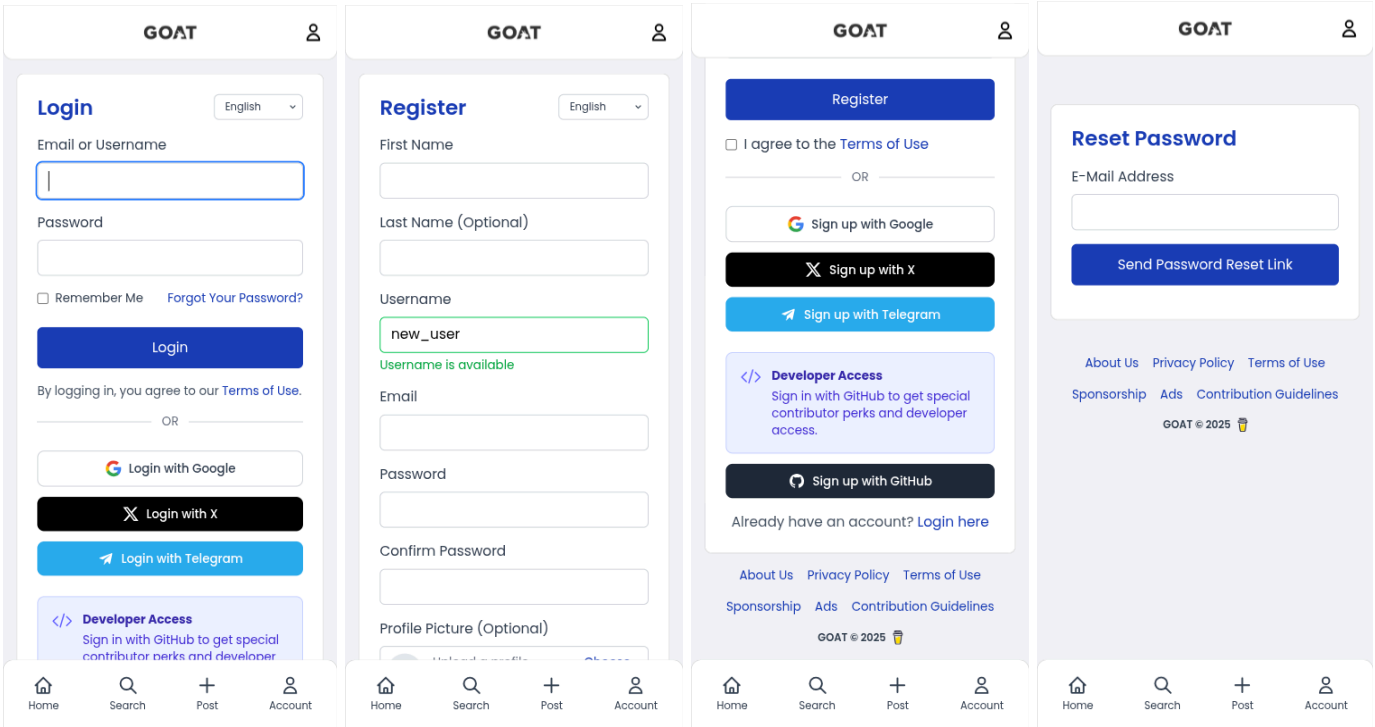
For seamless access, users can register or log in using their existing social media accounts. The platform supports:

- Google
- X (formerly Twitter)
- GitHub
- Telegram

Upon first-time social login, a GOAT.uz account is automatically created and linked.

1.3. Account & Password Recovery

- **Password Reset:** Users who have forgotten their password can initiate a reset process via their registered email address. A secure, time-sensitive link is sent to guide them through creating a new password.
- **Email Verification:** If a user's email is not verified, they can request a new verification link to be sent at any time from their profile settings.



2. Core Platform Interaction

This section covers the primary activities that form the core user experience on GOAT.uz.

2.1. Discovering Content

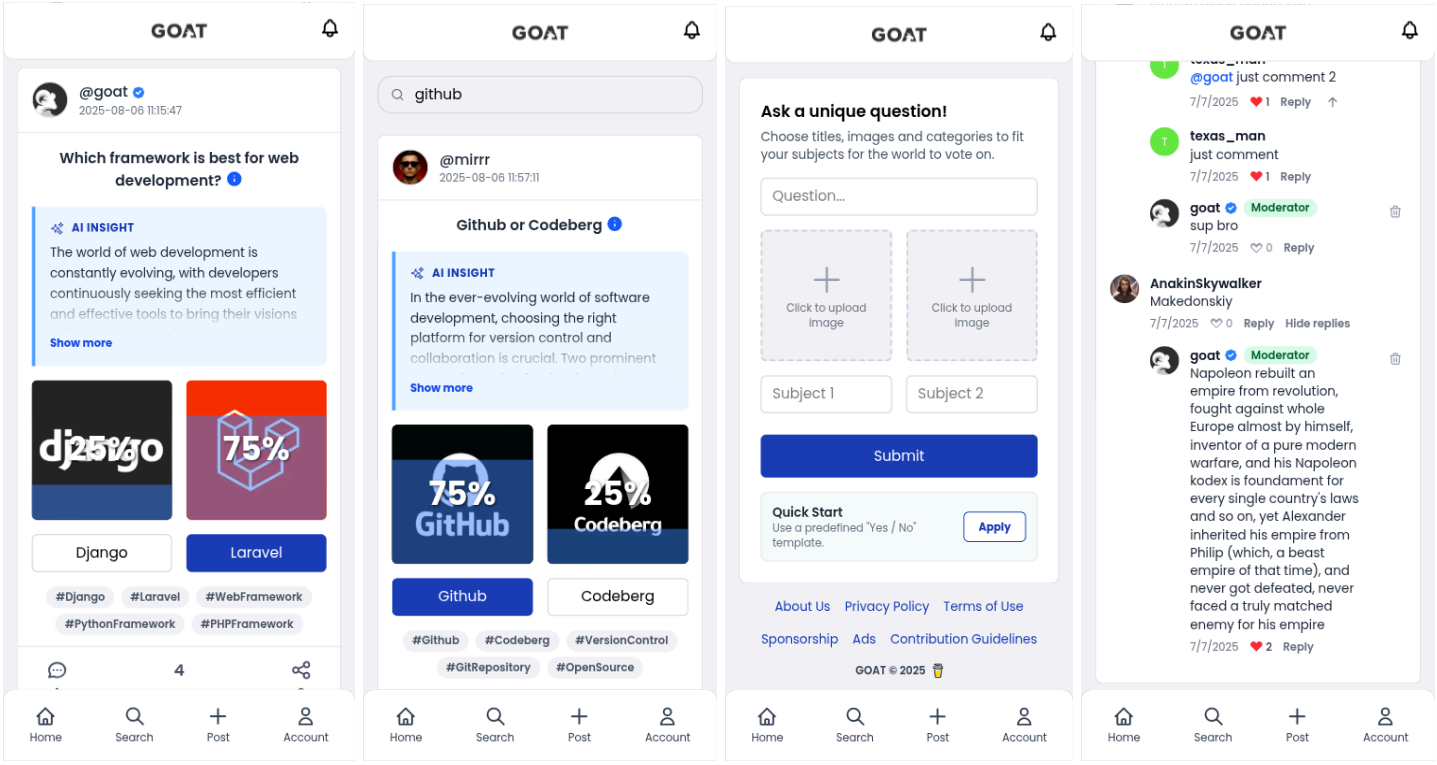
- **Homepage Feed:** The homepage presents a curated feed of the latest and most popular debates, which can be infinitely scrolled.
- **Advanced Search:** The platform's hybrid search engine allows users to find debates, specific arguments, and other users with high accuracy.
- **User Profiles:** Visiting a user's profile page (/@username) displays their activity, including debates they have started and participated in.

2.2. Creating & Managing Debates (Posts)

- Authenticated and verified users can initiate new debates on the platform.
- Users have full control over their own debates, with options to edit the topic and details or delete the debate entirely.

2.3. Engaging in Debates

- **Voting:** The primary engagement mechanism is voting on a debate's core premise. Votes are rate-limited to ensure fair participation.
- **Commenting (Arguments):** Users can add their arguments as comments within a debate. The system supports nested replies for structured discussion.
- **Comment Interaction:** Users can like individual comments to show approval and can edit or delete their own comments.
- **Sharing:** Debates can be easily shared to external platforms via the Web Share API, and the platform tracks share counts.



3. User Profile & Account Management

GOAT.uz provides users with extensive control over their profile, security, and data. All account management is accessible through the /profile/edit page.

3.1. Profile Customization

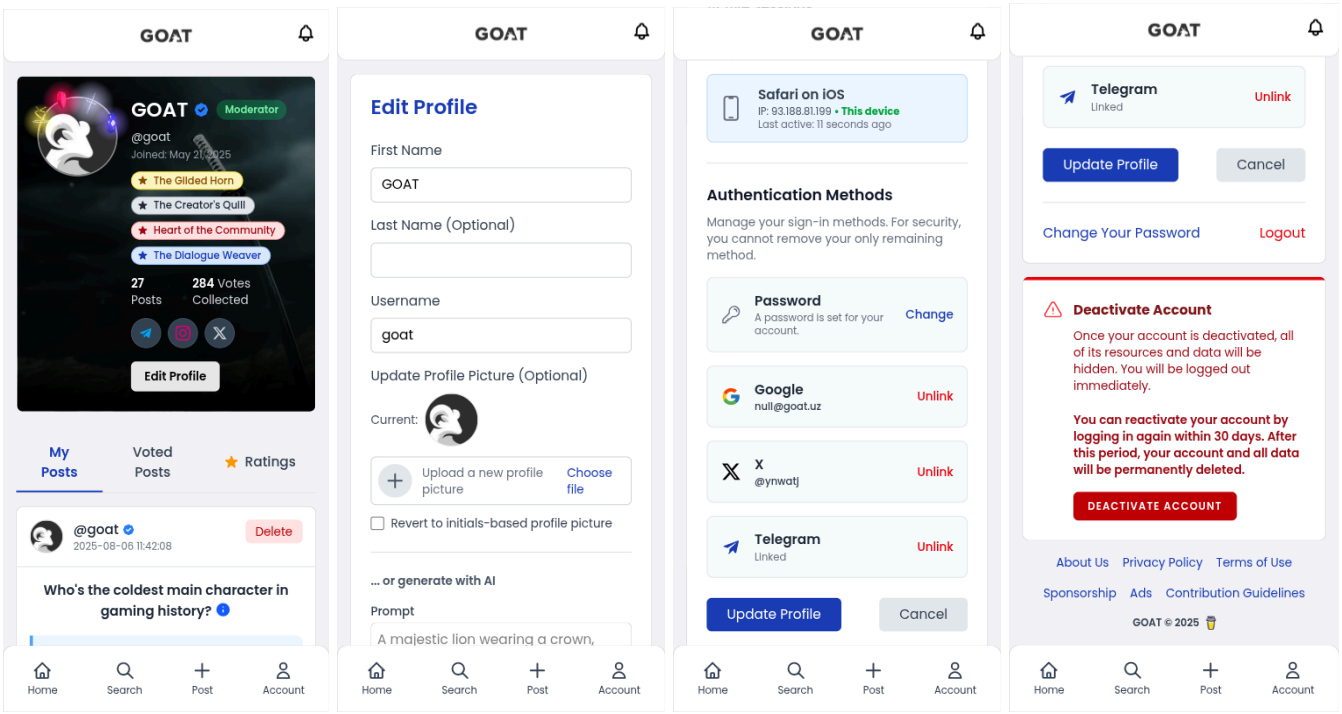
- **Editing Profile Information:** Users can update their username, bio, and other public-facing information.
- **AI-Generated Profile Picture:** A unique feature allows users to generate a custom profile picture using a text prompt, powered by the Stable Diffusion AI model.

3.2. Security & Access Control

- **Password Management:** Users can change their existing password. If a user signed up via a social provider, they can set a password to enable local login. For enhanced security, they can also remove their password entirely to enforce social-only login, a process which requires password confirmation.
- **Session Management:** The profile settings display a list of all active login sessions across different devices. Users have the ability to terminate any specific session or terminate all other sessions at once for security.
- **Social Account Linking:** Users can link or unlink multiple social providers to their account at any time, allowing for flexible login options.

3.3. Account Status

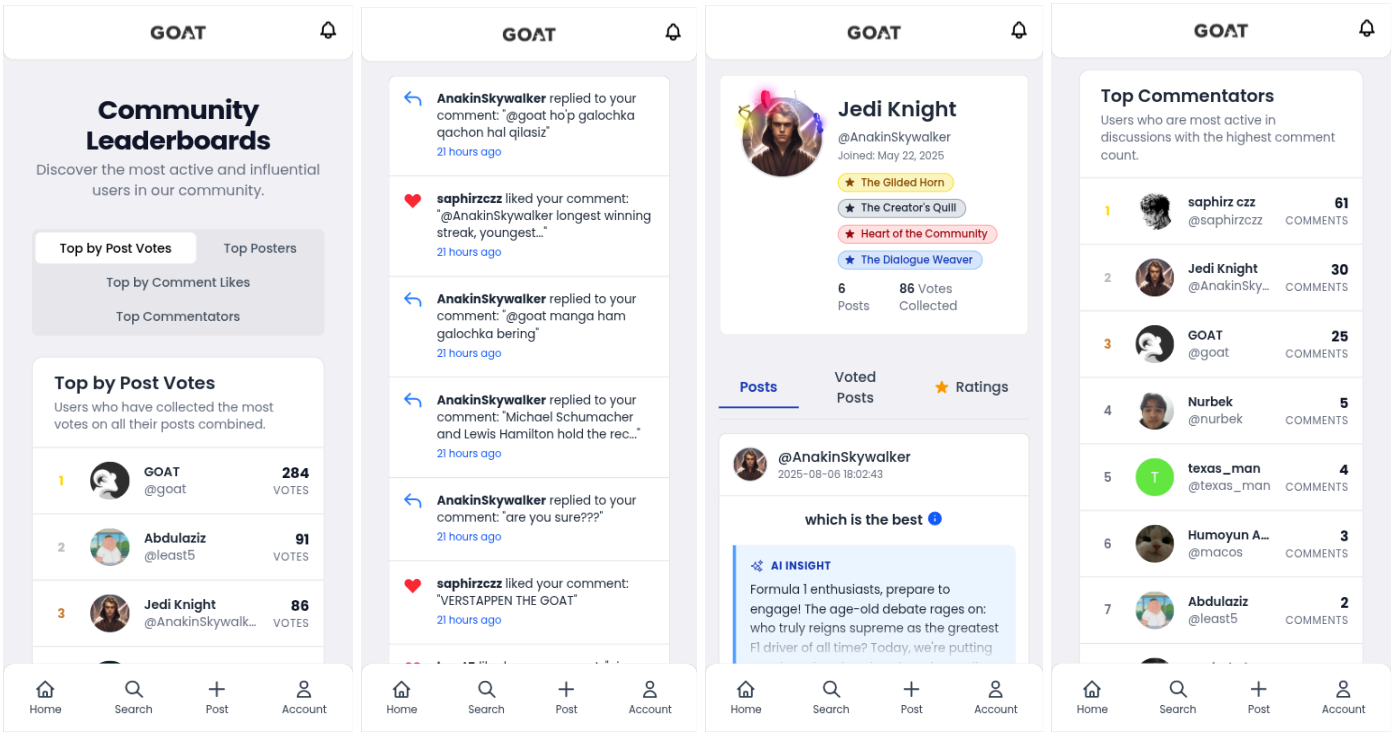
- **Account Deactivation:** Users can choose to temporarily deactivate their account. This action requires password confirmation and will log them out of all sessions. Their profile and content will be hidden but not deleted.
- **Account Reactivation:** A deactivated user can reactivate their account at any time by simply logging back in.



4. Community & Engagement Features

These features are designed to foster a vibrant and interactive community.

- **User Leaderboard:** A public rating system (/rating) ranks users based on their engagement and the quality of their contributions, encouraging high-quality participation.
- **Notifications:** A comprehensive notification system alerts users to important events, such as replies to their comments, votes on their debates, and new achievements unlocked. Users can view all notifications and see an unread count.
- **Public User Profiles:** Every user has a public profile page that showcases their created debates and their voting history, providing transparency and context for their platform activity.



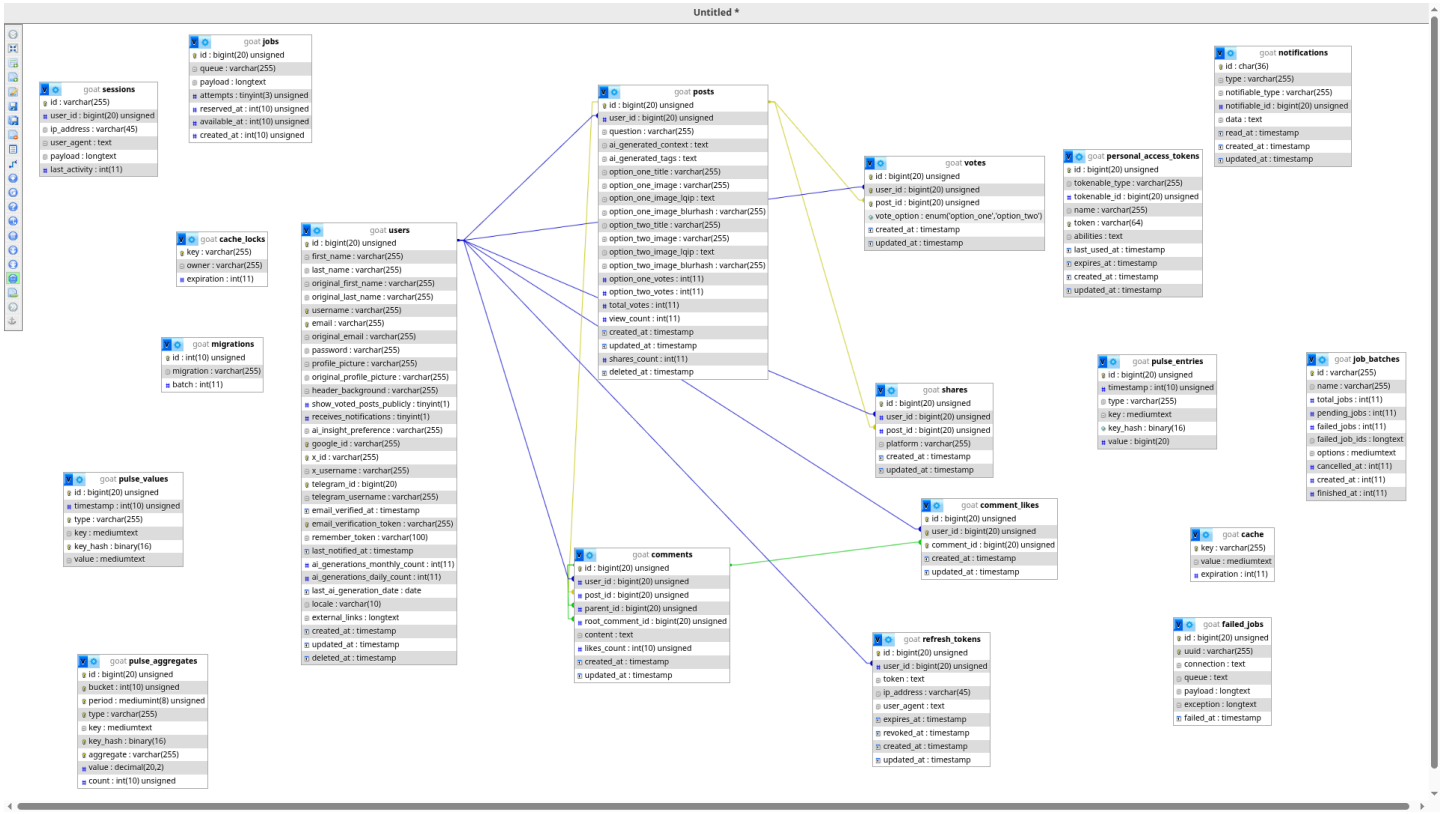
TESTING

The project employs a multi-layered testing and quality assurance strategy.

- **Static Analysis & Security:** The codebase is continuously scanned by SonarCloud (for code quality and bugs), Snyk (for dependency vulnerabilities), and Trivy (for container vulnerabilities).
- **Unit & Feature Testing:** A comprehensive test suite is maintained using PHPUnit. Tests cover critical business logic, model interactions, and API endpoints.
- **Running Tests:** The test suite can be executed by running php artisan test from the project root.

DATABASE SCHEME

The database schema is designed around core concepts of users, debates, and arguments.



TROUBLESHOOTING & FAQ

Error: WASM module fails to compile during build.sh.

- **Cause:** The Emscripten SDK (emcc) is not installed or not activated in the current shell session.
- **Solution:** Ensure the SDK is properly installed and its environment variables are sourced before running the script.

Error: API calls fail with a 401 or 403 error.

- **Cause:** Invalid or missing API keys in the .env file.
- **Solution:** Verify that all required keys (Google, Cloudflare, etc.) are present in .env and are correct.

FAQ: Why does the 3D achievement viewer take a moment to load on the first visit?

- **Answer:** The high-fidelity 3D models and HDRI environment maps are large assets. They are cached aggressively by the browser and the PWA's service worker after the initial download for near-instant loading on subsequent visits.

FUTURE IMPROVEMENTS & ROADMAP

UX/UI Enhancements

- **Platform-Wide Dark Mode:** Implement a fully integrated, user-toggleable dark mode to improve visual ergonomics and user comfort.
- **Enhanced Comment Threading UI:** Redesign the comment section to include visual connector lines (à la Reddit) for clearer tracking of nested replies and conversation flow.
- **Custom UI Dialogs:** Systematically replace all native browser alert() dialogs with a custom, consistently styled modal/dialog component for a more professional user experience.

AI & Content Intelligence

- **AI-Powered Content Translation:** Integrate a service to allow users to translate debate posts and comments into their preferred language with a single click.
- **Duplicate Content Detection:** Implement a system to detect and flag when a user is attempting to create a new debate on a topic that is substantially similar to an existing one.
- **Resilient AI Model Fallbacks:** Build a failover mechanism for the AI services. If a request to the primary model (e.g., Gemini) fails, the system will automatically retry with a secondary model to ensure service continuity.
- **Full i18n Support for Notifications:** Extend the internationalization system to cover all user-facing notifications, ensuring they are delivered in the user's selected language.

Community & Gamification

- **Advanced User Reputation System:** Evolve the current rating system into a comprehensive reputation engine. This will feature a custom, AI-based scoring algorithm that analyzes the quality and detail of comments, rewarding users for substantive contributions rather than just popularity. Scores will be tracked for individual posts, comments, and overall user history.
- **Enhanced GitHub Integration & Developer Tools:** For users authenticating with GitHub, provide special notices and unlock a unique set of developer-centric tools and profile flair, fostering a community of technical users.

Platform Expansion

- **Content Vertical:** Launch a new subdomain dedicated to long-form articles and analysis. This will serve as an editorial and knowledge base component of the platform.
- **Article-to-Debate Binding:** Create a feature that allows articles on article.goat.uz to be directly linked to or used as the foundational source for new debates on the main platform, seamlessly integrating the two content verticals.

LICENSE & CREDITS

Lead Architect & Developer: The project was conceived, architected, and led by Ismoiljon Umarov.