Trayb.az v3: Comprehensive Esports Platform Specification

Executive Summary

Trayb.az is a full-featured competitive esports platform supporting **Valorant** and **CS2** with modular architecture for future game expansion. The platform provides queue-based matchmaking, custom tournaments, private hubs, and comprehensive statistics tracking with dual rating systems (team-based ELO and personal Rating 2.0).

1. Platform Architecture

1.1 Hierarchy & Role System

The platform implements a **modular role-based access control (RBAC)** system with the following hierarchy [1][53][56]:

Level	Role	Access Domain	Key Permissions
1 (Top)	**Organizer/Owner**	admin.trayb.az	Full platform control, create tournaments, manage all roles, access recordings
2	**Admin**	admin.trayb.az	Tournament creation, user moderation, stats imports, match control
3	**Moderator**	admin.trayb.az	Dispute resolution, chat monitoring, basic match oversight
4	**Competitor/Player**	trayb.az	Compete in matches, view stats, join queues/hubs, create teams
5 (Bottom)	**Viewer/Spectator**	trayb.az	Watch live games, view public leaderboards/statistics

Future Extensibility: System designed to support additional roles (Coach, Analyst, Caster, Team Captain) through modular permission framework [50][53].

1.2 Domain Structure

- trayb.az: Public-facing player and viewer experience
- admin.trayb.az: Administrative dashboard for Organizers, Admins, and Moderators

1.3 Technical Stack (Monorepo)

```
/apps
  /frontend          # Next.js public site (trayb.az + admin.trayb.az)
  /backend           # API server (Auth.js JWT, PostgreSQL)
  /controlbot        # Discord VC management bot
  /recorderbot1      # Team 1 audio recorder
  /recorderbot2      # Team 2 audio recorder
/packages
  /shared            # Shared types, utilities, constants
```

Key Technologies:

- **Package Manager**: Bun (not npm)
- **Database**: PostgreSQL
- **Authentication**: Auth.js with JWT
- **Framework**: Turborepo monorepo
- **UI Components**: shadcn/ui registry only (no custom AI components)
- **Deployment**: Dokploy (single service per branch: beta, main)
- **CI/CD**: GitHub Actions
- **No local .env files** - production-ready config management
- **No unnecessary shell scripts**

2. Game Modes & Queue Types

2.1 Game Mode Classification

Mode	Description	ELO Impact	Rankings	Availability
Unranked	Casual play, custom rules	No ELO change	Performance shown only	Always available
Ranked Global	Scheduled competitive queue	Global ELO tracked	Global leaderboards	Set schedules (Trayb Series)
Private Hub Ranked	Invitation-only ranked matches	Hub-specific ELO	Separate hub leaderboards	Whitelisted users only

2.2 Special Features by Queue Type

Unranked & Private Hub Only:

- Draft phase options: Random, Elo-Based, Captain Draft
- Pick/Bans system
- Captain can decide drafting style
- Custom match configurations

Ranked Global (Trayb Series):

- Scheduled queue times
- Standard tournament rules
- Automated matchmaking
- Official statistics tracking

3. Rating Systems

3.1 ELO System (Team Performance Metric)

Purpose: Measures how well a player contributes to team victories.

Formula (Valorant):

$$\text{Power Score} = 0.8 \times \text{Team ELO} + 0.2 \times \text{League ELO}$$

For private hubs, League ELO = 0 (hub-isolated) [2] [3].

Key Principles:

- **Win/Loss driven**: Primary factor is match outcome
- **Opponent strength weighting**: Beating higher-ELO teams = larger gain
- **Persistent across teams**: Player ELO does NOT reset when joining new teams
- **Queue-specific**: Separate ELO pools for Global vs. Private Hub
- **Game-specific**: Valorant and CS2 have independent ELO systems

CS2 Specific:

- Map-specific ELO optional (future extension)
- Premier-style rating (0-15,000 point scale) [4] [5]

3.2 Rating 2.0 System (Individual Performance Metric)

Purpose: Evaluates personal skill independent of team result. Distinguishes "carried by team" from "dragged down by team" scenarios.

Scale: 1.0 = average, >2.0 = exceptional, <1.0 = underperformance

Valorant Rating 2.0 (VLR-based) [6] [7]

Components:

1. **Kill Contribution** (weighted by fight context: 5v5 worth more than 5v1)
2. **Death Contribution** (negative weight)
3. **Assists Per Round (APR)**

4. **Adjusted ADR (ADRa)** - damage not already counted in kills
5. **Survival Rating**

Formula:

$$\text{Rating 2.0} = w_1 \times \text{KillContrib} + w_2 \times \text{DeathContrib} + w_3 \times \text{APR} + w_4 \times \text{ADRa} + w_5 \times$$

Key Features:

- Round-by-round granular analysis
- Fight weighting: turning 4v5 into 4v4 valued equally to 5v5 → 5v4
- Mean = 1.0, Standard Deviation = 0.33
- Clutch situations heavily weighted

CS2 Rating 2.0 (HLTV-based) [8] [9]

Reverse-engineered formula:

$$\text{Rating 2.0} = 0.0073 \times \text{KAST} + 0.3591 \times \text{KPR} - 0.5329 \times \text{DPR} + 0.2372 \times \text{Impact} + 0.$$

Variables:

- **KAST**: % of rounds with Kill, Assist, Survived, or Traded
- **KPR**: Kills Per Round
- **DPR**: Deaths Per Round (negative impact)
- **Impact**: Multi-kills + opening kills + clutches
- **ADR**: Average Damage Per Round

Impact sub-formula (approximate):

$$\text{Impact} \approx 2.13 \times \text{KPR} + 0.42 \times \text{APR} - 0.41$$

3.3 Implementation Notes

- **Modular calculation**: Store formulas as configurable functions/classes
- **Queue-specific persistence**: Each queue type maintains separate rating histories
- **No team-based reset**: Player ratings persist across team changes
- **Transparency UI**: Show expected ELO change, explain rating components
- **Performance feedback**: Indicate if player outperformed team expectations (Overwatch-style SR approach) [10] [11]

4. Match Flow & Automation

4.1 Match Room Lifecycle

Phase 1: Lobby Creation

1. Match room created (manual for custom, auto for queue)
2. Players must join designated Discord lobby VC
3. Only lobby VC members can ready up
4. Chat opens for match room communication

Phase 2: Drafting (Unranked/Private Hub only)

1. Admin/Captain selects draft mode:
    - Random team assignment
    - Elo-based balancing
    - Captain draft (player picking or auto-assign)
2. Pick/Ban phase (if enabled)
3. Team compositions finalized

Phase 3: Match Start

1. **ControlBot** receives signal from backend API
2. Creates two private team VCs
3. Moves players to respective team channels
4. Enables anticheat mode (prevents channel hopping)
5. **ControlBot** signals **RecorderBot1** and **RecorderBot2** via private API
6. Recorder bots join team VCs and begin recording

Phase 4: Match End

1. Admin uploads match statistics (tracker.gg HTML files for now)
2. System processes stats and calculates:
    - ELO changes
    - Rating 2.0 scores
3. Recorder bots save and upload audio files to backend
4. Recordings accessible to **Admins only** (strategy protection)
5. Players moved back to lobby VC
6. Match room transitions to "completed" state (persistent)

4.2 Bot Architecture

ControlBot:

- Manages Discord voice channel logistics
- Enforces lobby presence requirements
- Moves players between VCs
- Triggers recorder bots via internal API
- Anticheat enforcement (VC lock)

RecorderBot1 & RecorderBot2:

- Join team VCs on ControlBot signal
- Record individual player audio streams
- Generate combined team audio
- Upload recordings to backend private storage
- Only activate for:
  - Tournament matches (production)
  - Private hub ranked games
  - Testing (via env flag)

API Integration:

- All bots controlled via backend REST/WebSocket endpoints
- No direct user commands
- Event-driven architecture

5. Statistics & Data Management

5.1 Stats Import (Current: v3 Phase 1)

tracker.gg **HTML Upload (Admin function):**

- Scoreboard overview
- Performance breakdown (10 HTML files per player)
- Economy breakdown
- Round-by-round analysis
- Duel-by-duel breakdown

Backend Processing:

1. Parse HTML files
2. Extract structured data

3. Store in PostgreSQL
4. Calculate ELO and Rating 2.0
5. Update leaderboards

5.2 Stats Import (Future: v3 Phase 2)

- **Valorant**: GRID Esports API integration [^39]
- **CS2**: GOTV demo parsing

5.3 Leaderboard Structure [44][45]

Priority Sorting:

1. ELO (primary)
2. Rating 2.0 (secondary)
3. Win/Loss record
4. Total matches played

Leaderboard Types:

- Global (Trayb Series)
- Private Hub (separate per hub)
- Per-game (Valorant vs. CS2)
- All-time vs. seasonal

Profile Stats Page:

- Full detailed statistics
- Last 5 matches preview (expandable to all)
- Match history links to persistent match rooms
- ELO/Rating 2.0 change animations

6. Database Schema Design

6.1 Core Tables

Users Table:

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(100) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  discord_id VARCHAR(100),
```

```
  created_at TIMESTAMP DEFAULT NOW()
);
```

Roles Table (Modular RBAC):

```
CREATE TABLE roles (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50) UNIQUE NOT NULL,
  description TEXT,
  level INTEGER NOT NULL  -- 1=Organizer, 2=Admin, 3=Mod, 4=Player, 5=Viewer
);

CREATE TABLE permissions (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  resource VARCHAR(100) NOT NULL,  -- e.g., 'tournaments', 'users'
  action VARCHAR(50) NOT NULL       -- e.g., 'create', 'edit', 'delete'
);

CREATE TABLE role_permissions (
  role_id INTEGER REFERENCES roles(id) ON DELETE CASCADE,
  permission_id INTEGER REFERENCES permissions(id) ON DELETE CASCADE,
  PRIMARY KEY (role_id, permission_id)
);

CREATE TABLE user_roles (
  user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
  role_id INTEGER REFERENCES roles(id) ON DELETE CASCADE,
  PRIMARY KEY (user_id, role_id)
);
```

Games Table:

```
CREATE TABLE games (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,  -- 'Valorant', 'CS2'
  code VARCHAR(10) UNIQUE NOT NULL  -- 'VAL', 'CS2'
);
```

Hubs Table:

```
CREATE TABLE hubs (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  game_id INTEGER REFERENCES games(id),
  is_private BOOLEAN DEFAULT false,
  is_global BOOLEAN DEFAULT false,  -- Trayb Series main hub
  whitelist_only BOOLEAN DEFAULT false,
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE hub_whitelist (
```

```
  hub_id INTEGER REFERENCES hubs(id) ON DELETE CASCADE,
  user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
  PRIMARY KEY (hub_id, user_id)
);
```

Teams Table (Future: Team Registration):

```
CREATE TABLE teams (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  game_id INTEGER REFERENCES games(id),
  captain_user_id INTEGER REFERENCES users(id),
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE team_members (
  team_id INTEGER REFERENCES teams(id) ON DELETE CASCADE,
  user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
  role VARCHAR(50),  -- 'player', 'coach', 'manager'
  joined_at TIMESTAMP DEFAULT NOW(),
  PRIMARY KEY (team_id, user_id)
);
```

Matches Table:

```
CREATE TABLE matches (
  id SERIAL PRIMARY KEY,
  hub_id INTEGER REFERENCES hubs(id),
  game_id INTEGER REFERENCES games(id),
  queue_type VARCHAR(50) NOT NULL,  -- 'unranked', 'ranked_global', 'private_ranked'
  status VARCHAR(50) DEFAULT 'pending',  -- 'pending', 'drafting', 'live', 'completed'
  draft_mode VARCHAR(50),  -- 'random', 'elo', 'captain'
  created_at TIMESTAMP DEFAULT NOW(),
  started_at TIMESTAMP,
  ended_at TIMESTAMP
);

CREATE TABLE match_players (
  match_id INTEGER REFERENCES matches(id) ON DELETE CASCADE,
  user_id INTEGER REFERENCES users(id),
  team_side VARCHAR(10),  -- 'team1', 'team2'
  is_captain BOOLEAN DEFAULT false,
  PRIMARY KEY (match_id, user_id)
);
```

Player Stats Table:

```
CREATE TABLE player_match_stats (
  id SERIAL PRIMARY KEY,
  match_id INTEGER REFERENCES matches(id) ON DELETE CASCADE,
  user_id INTEGER REFERENCES users(id),
  kills INTEGER DEFAULT 0,
```

```
  deaths INTEGER DEFAULT 0,
  assists INTEGER DEFAULT 0,
  adr FLOAT,
  kast FLOAT,
  clutches INTEGER DEFAULT 0,
  first_kills INTEGER DEFAULT 0,
  -- Valorant specific
  survival_rounds INTEGER,
  -- CS2 specific
  impact_score FLOAT,

  UNIQUE(match_id, user_id)
);
```

ELO & Rating Table:

```
CREATE TABLE player_ratings (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id),
  game_id INTEGER REFERENCES games(id),
  hub_id INTEGER REFERENCES hubs(id),  -- NULL for global
  elo INTEGER DEFAULT 1000,
  rating_2_0 FLOAT DEFAULT 1.0,
  matches_played INTEGER DEFAULT 0,
  wins INTEGER DEFAULT 0,
  losses INTEGER DEFAULT 0,
  updated_at TIMESTAMP DEFAULT NOW(),
  UNIQUE(user_id, game_id, hub_id)
);

CREATE TABLE rating_history (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id),
  match_id INTEGER REFERENCES matches(id),
  elo_before INTEGER,
  elo_after INTEGER,
  rating_2_0_value FLOAT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Recordings Table:

```
CREATE TABLE match_recordings (
  id SERIAL PRIMARY KEY,
  match_id INTEGER REFERENCES matches(id) ON DELETE CASCADE,
  team_side VARCHAR(10),  -- 'team1', 'team2'
  file_url TEXT NOT NULL,
  file_type VARCHAR(50),  -- 'combined', 'individual_{user_id}'
  uploaded_at TIMESTAMP DEFAULT NOW()
);
```

6.2 Tournament Tables (Future Extension)

```
CREATE TABLE tournaments (
  id SERIAL PRIMARY KEY,
  name VARCHAR(200) NOT NULL,
  game_id INTEGER REFERENCES games(id),
  tournament_type VARCHAR(50),  -- 'online', 'offline', 'hybrid'
  start_date TIMESTAMP,
  end_date TIMESTAMP,
  registration_open BOOLEAN DEFAULT false,
  max_teams INTEGER
);

CREATE TABLE tournament_registrations (
  id SERIAL PRIMARY KEY,
  tournament_id INTEGER REFERENCES tournaments(id) ON DELETE CASCADE,
  team_id INTEGER REFERENCES teams(id),
  status VARCHAR(50) DEFAULT 'pending',  -- 'pending', 'approved', 'declined'
  registered_at TIMESTAMP DEFAULT NOW()
);
```

7. Development Workflow

7.1 QA Testing Strategy

Simulated Player Testing:

- Use Puppeteer/Playwright to simulate 10 concurrent players
- Test full match lifecycle: lobby join → draft → match → stats
- Automate UI/UX flow testing across all phases
- No need for 10 real humans during development

Automated Tests:

- Unit tests for ELO/Rating 2.0 calculation functions
- Integration tests for API endpoints
- E2E tests for critical user flows
- Bot communication testing

7.2 Branching & Deployment

Branches:

- `main`: Production-ready code
- `beta`: Staging/testing environment
- Feature branches: Merge to beta → main

Dokploy Deployment:

- Single service deployment per branch
- PostgreSQL container (Docker)
- Redis container (Docker) - for caching/sessions
- No separate microservices

CI/CD (GitHub Actions):

- Automated testing on PR
- Linting and type checking
- Build verification
- Auto-deploy to beta on merge

7.3 Development Phases

v3 Phase 1 (Current Scope):

- Core platform with 5 role hierarchy
- Unranked, Ranked Global, Private Hub queues
- ELO and Rating 2.0 systems (Valorant + CS2)
- ControlBot and RecorderBots integration
- tracker.gg stats import
- Basic leaderboards and match history

v3 Phase 2 (January - Team Registration):

- Team creation, invitation, and management
- Tournament registration system
- Online → Offline tournament support
- Coach/Manager roles
- GRID API (Valorant) and GOTV (CS2) integration

v3 Phase 3 (Future):

- Additional game support (modular)
- Advanced analytics dashboard
- Caster/Analyst roles
- Live spectator mode
- Stream integration

8. UI/UX Requirements

8.1 Design Principles

- **Scalable interface**: Accommodate growing feature set
- **No cliche animations**: Avoid overused effects (no Matrix-style backgrounds)
- **Smooth, snappy interactions**: Dynamic transitions without abrupt cuts
- **Z-index management**: No background animations interfering with content
- **Accessibility**: WCAG 2.1 AA compliance

8.2 Theme System

- Custom color codes (provided separately)
- Custom logo (favicon + navbar)
- Light/Dark mode toggle
- Consistent design language across admin and public site

8.3 shadcn/ui Components Only

Approved components:

- All components from shadcn/ui registry
- No custom AI-generated components
- Explicit request required for each new component

8.4 Key UI Features

ELO/Rating Animations:

- Animated number transitions on first/current visit
- Visual diff display (↑/↓ with color coding)
- Tooltips explaining calculation factors

Match Room:

- Real-time status updates
- Team chat integration
- Draft phase visualizations (captain draft picker)
- Audio recording indicator (admin view only)

Leaderboards:

- Sortable columns
- Pagination

- Filter by hub/game/time period
- Player profile deep links

9. Security & Privacy

9.1 Authentication

- Auth.js with JWT tokens
- Secure password hashing (bcrypt/argon2)
- Discord OAuth integration
- Rate limiting on auth endpoints

9.2 Authorization

- RBAC middleware on all protected routes
- Permission checks at API and UI layers
- Admin-only access to recordings
- Hub whitelist enforcement

9.3 Data Protection

- Match recordings: Admin-only access (strategy protection)
- Personal data: GDPR-compliant storage
- Secure file upload validation
- SQL injection prevention (parameterized queries)

10. API Structure

10.1 Public API Routes

```
GET  /api/matches/:id              # Match details
GET  /api/leaderboards/:game/:hub  # Leaderboard data
GET  /api/players/:id              # Player profile
GET  /api/players/:id/stats        # Player statistics
GET  /api/hubs                     # Available hubs (filtered by access)
POST /api/teams                    # Create team (authenticated)
```

10.2 Admin API Routes

```
POST /api/admin/tournaments        # Create tournament
POST /api/admin/matches/:id/stats  # Upload match stats
GET  /api/admin/recordings/:id     # Access match recordings
PUT  /api/admin/users/:id/roles    # Assign roles
POST /api/admin/hubs               # Create/manage hubs
```

10.3 Internal Bot API Routes

```
POST /api/internal/bots/control/move       # ControlBot VC commands
POST /api/internal/bots/recorder/start     # Start recording
POST /api/internal/bots/recorder/upload    # Upload recording
GET  /api/internal/matches/:id/voice-state # Current VC state
```

11. Success Metrics

11.1 Technical KPIs

- API response time <200ms (p95)
- Database query optimization (indexed lookups)
- Zero downtime deployments
- 99.9% uptime SLA

11.2 User Experience KPIs

- Match creation to start: <5 minutes
- Stats processing: <2 minutes post-upload
- ELO calculation accuracy: ±5 points max error
- Rating 2.0 calculation: Match VLR/HLTV within 2%

11.3 Platform Growth KPIs

- Active players per day
- Matches played per week
- Tournament registrations (post-Phase 2)
- Player retention rate

12. References & Attribution

This specification is built upon industry-standard rating systems and best practices:

- VLR.gg **Rating 2.0**: Valorant player performance algorithm [6] [7]
- **HLTV 2.0 Rating**: CS:GO/CS2 player performance standard [8] [9]
- **Riot Games Power Score**: Team ELO methodology [2] [3]
- **FACEIT Hub Permissions**: Tournament admin hierarchy [1]
- **PostgreSQL Sports Tournament Schema**: Database design patterns [^44]
- **RBAC Best Practices**: Modular role-permission systems [50][53][^56]

Appendix A: Glossary

- **ELO**: Team-based performance rating system
- **Rating 2.0**: Individual player performance metric (1.0 = average)
- **Hub**: Isolated competitive environment with separate rankings
- **Trayb Series**: Official global ranked queue and leaderboard
- **RBAC**: Role-Based Access Control
- **KAST**: Kills, Assists, Survived, Traded (CS2 metric)
- **ADR**: Average Damage per Round
- **APR**: Assists per Round
- **KPR/DPR**: Kills/Deaths per Round

Appendix B: Next Steps with Cursor

1. **Analyze current repository structure** (`/apps/frontend`, `/backend`, etc.)
2. **Audit existing stack** (PostgreSQL, Auth.js, Bun, Turborepo setup)
3. **Create detailed task breakdown** for v3 Phase 1 implementation
4. **Identify migration needs** from v1/v2 to v3 architecture
5. **Generate database migration scripts** for new schema
6. **Implement modular rating calculation functions**
7. **Design API endpoint structure** (public, admin, internal)
8. **Bot integration architecture** (ControlBot + RecorderBots)
9. **Set up QA testing framework** (Puppeteer multi-player simulation)
10. **UI component inventory** from shadcn/ui registry

Cursor Auto Mode Workflow:

- Tackle each module independently

- Test after each major component
- Use beta branch for incremental deployment
- Continuous integration via GitHub Actions

Document Version: 1.0
Last Updated: November 13, 2025
Author: Trayb.az Development Team
Status: Ready for Cursor Implementation Planning
[12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38]

⁂

1. https://www.vlr.gg/474010/vlr-rating-2-0
2. https://blog.grid.gg/grid-predictions-part-1-player-ratings-305a6f732847
3. https://www.reddit.com/r/RocketLeagueEsports/comments/10io62d/how_does_shift_calculate_their_player_rating/
4. https://www.sheepesports.com/articles/introducing-player-scores/en
5. https://www.datensen.com/blog/data-model/designing-a-sports-tournament-data-model/
6. https://www.jareddesign.ca/case-studies/modular-access.html
7. https://cyfuture.cloud/kb/gaming/how-to-design-er-diagrams-for-online-gaming-platforms
8. https://dev.to/leapcell/designing-rbac-permission-system-with-nestjs-a-step-by-step-guide-3bhl
9. https://creately.com/diagram/example/kemty2q63/esports-database-classic
10. https://turbosmurfs.gg/article/how-do-rankings-work-in-the-video-game-industry
11. https://www.geeksforgeeks.org/dbms/how-to-design-a-database-for-multiplayer-online-games/
12. https://www.vlr.gg/381456/vlr-rating-2-0-update
13. https://rocketscience.fyi/news/esports/better_score
14. https://www.vlr.gg/351842/anyone-know-how-to-calculate-rating
15. https://arxiv.org/html/2501.10049v2
16. https://boostroyal.com/blog/global-power-rankings-in-esports-the-rating-system-explained
17. https://dubsnatch.com/blogs/gaming/stats-esports-players
18. https://www.vlr.gg/160667/vlr-gg-player-rating-explained
19. https://www.halo-lab.com/project/esport-company
20. https://chartdb.io
21. https://stackoverflow.com/questions/53772633/database-model-tournaments-for-esports
22. https://stackoverflow.com/questions/11201755/database-design-for-tournament-management-software
23. https://www.reddit.com/r/Supabase/comments/1d5ax8s/database_design_for_my_sports_matches/
24. https://www.rapidevelopers.com/usecase/esports-tournament-management-system-for-gaming-cafe
25. https://www.nocobase.com/en/blog/how-to-design-rbac-role-based-access-control-system
26. https://www.reddit.com/r/ValorantCompetitive/comments/14v4s7q/approximating_valorant_ratings_vlrribthespike/
27. https://www.scribd.com/document/579914729/Sports-Tournament-Management-System

28. https://github.com/hammadasifali/SportManagementSystem
29. https://www.sevensquaretech.com/multi-role-permission-system-laravel-with-code/
30. https://databasesample.com/database/online-gaming-tournament-platform-database
31. http://en.ystok.ru/tournament/
32. https://stackoverflow.com/questions/333620/best-practice-for-designing-user-roles-and-permission-system
33. https://www.athleteyard.com/tournament/
34. https://markzhdan.com/blogs/reverse-engineering-vlr-rating
35. https://dave.コム/posts/reverse-engineering-hltv-rating/
36. https://www.youtube.com/watch?v=mymBWc62TcY
37. https://en.wikipedia.org/wiki/Elo_rating_system
38. https://www.reddit.com/r/GlobalOffensive/comments/18x0ihl/hltv_undermines_their_own_rating_20_system/