Futmatrix – Database Schema Documentation

Version 2.0 – LangGraph & Tokenomics Aligned

Platform

Supabase (PostgreSQL)

Purpose

To power the Futmatrix platform with a flexible, normalized, and scalable data structure supporting:

- Competitive gaming match data from EAFC25
- Interaction logs with AI agents (Coach & Rivalizer)
- Token reward/penalty tracking
- Plan-based training accountability
- Streamed match validation and ranking computation

---

1. Core Tables

1.1 `users`

```
CREATE TABLE public.users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  discord_id TEXT UNIQUE,
  platform_auth_id TEXT UNIQUE,
  username TEXT NOT NULL,
  whop_id TEXT,
  subscription_status TEXT,
  created_at TIMESTAMP DEFAULT now(),
  status TEXT DEFAULT 'active',
  last_match_at TIMESTAMP,
  matches_count INT DEFAULT 0
);
```

1.2 `user_plans`

```
CREATE TABLE public.user_plans (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  plan TEXT NOT NULL,
  started_at TIMESTAMP DEFAULT now(),
  ended_at TIMESTAMP,
  source TEXT DEFAULT 'whop'
);
```

1.3 `matches`

```
CREATE TABLE public.matches (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  timestamp TIMESTAMP DEFAULT now(),
  match_type TEXT, -- e.g., 'rivalizer', 'division_rivals', 'friendly'
  is_ranked BOOLEAN DEFAULT false,
  data_coverage_level TEXT, -- e.g., 'summary_only', 'advanced'
  game_mode TEXT,
  source_image_url TEXT,
  source_agent TEXT,
  raw_json JSONB,

  -- Raw stats
  score_user INT,
  score_opponent INT,
  shots_total INT,
  passes_attempted INT,
  pass_accuracy FLOAT,
  tackles_total INT,
  dribble_success_rate FLOAT,
  shot_accuracy FLOAT,
  tackle_success_rate FLOAT,
  fouls_committed INT,
  offsides INT,
  corners INT,
  free_kicks INT,
  penalty_kicks INT,
  yellow_cards INT,
  red_cards INT,
  def_line_breaks_through INT,
  def_line_breaks_around INT,
  def_line_breaks_over INT,
  def_line_breaks_attempted INT
);
```

1.4 `processed_metrics`

```
CREATE TABLE public.processed_metrics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  match_id UUID REFERENCES public.matches(id) ON DELETE CASCADE,
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  shot_efficiency FLOAT,
  pass_efficiency FLOAT,
  possession_efficiency FLOAT,
  defensive_efficiency FLOAT,
  overall_performance FLOAT,
  custom_json JSONB
);
```

1.5 `user_stats_summary`

```
CREATE TABLE public.user_stats_summary (
  user_id UUID PRIMARY KEY REFERENCES public.users(id) ON DELETE CASCADE,
  matches_played INT DEFAULT 0,
  wins INT DEFAULT 0,
  losses INT DEFAULT 0,
  draws INT DEFAULT 0,
  win_rate FLOAT DEFAULT 0,
  goals_scored INT DEFAULT 0,
  goals_conceded INT DEFAULT 0,
  avg_shot_efficiency FLOAT,
  avg_pass_efficiency FLOAT,
  avg_possession_efficiency FLOAT,
  avg_defensive_efficiency FLOAT,
  avg_overall_performance FLOAT,
  performance_trend_5 FLOAT,
  last_match_at TIMESTAMP
);
```

---

2. Agent-Specific Tables

2.1 `agent_interactions`

```
CREATE TABLE public.agent_interactions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  agent_type TEXT, -- 'coach', 'rivalizer'
  interaction_type TEXT, -- 'chat', 'match_suggestion', 'plan_adjustment', etc.
```

```
  content TEXT,
  payload JSONB,
  timestamp TIMESTAMP DEFAULT now()
);
```

2.2 `training_plans`

```
CREATE TABLE public.training_plans (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  start_date DATE,
  end_date DATE,
  checkpoints JSONB, -- Weekly goals or tasks
  stake_amount INT,
  status TEXT, -- 'in_progress', 'completed', 'failed'
  reward_issued BOOLEAN DEFAULT false,
  penalty_applied BOOLEAN DEFAULT false
);
```

---

3. Tokenomics Tables

3.1 `penalties`

```
CREATE TABLE public.penalties (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id) ON DELETE CASCADE,
  card_type TEXT, -- 'yellow', 'red'
  reason TEXT,
  token_fine INT,
  imposed_at TIMESTAMP DEFAULT now(),
  paid BOOLEAN DEFAULT false
);
```

3.2 `streaming_rewards`

```
CREATE TABLE public.streaming_rewards (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id),
  match_id UUID REFERENCES public.matches(id),
  stream_url TEXT,
  reward_amount INT,
  validated BOOLEAN DEFAULT false,
  issued_at TIMESTAMP DEFAULT now()
);
```

3.3 `replay_uploads`

```
CREATE TABLE public.replay_uploads (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES public.users(id),
  match_id UUID REFERENCES public.matches(id),
  content_type TEXT,
  ai_difficulty_tag TEXT,
  reward_amount INT,
  uploaded_at TIMESTAMP DEFAULT now()
);
```

---

4. Rankings & Views

4.1 `weekly_rankings`

```
CREATE TABLE public.weekly_rankings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  week_start DATE,
  rank_type TEXT, -- 'week_on_fire', 'rivalizer_arena'
  rankings JSONB -- { position: user_id }
);
```

4.2 `coach_user_view`

```
CREATE VIEW public.coach_user_view AS
SELECT
  u.id AS user_id,
  u.username,
  uss.avg_overall_performance,
  uss.performance_trend_5,
  uss.matches_played,
  uss.win_rate,
  uss.last_match_at
FROM public.users u
JOIN public.user_stats_summary uss ON u.id = uss.user_id;
```

4.3 `rivalizer_matchmaking_view`

```
CREATE VIEW public.rivalizer_matchmaking_view AS
SELECT
  user_id,
```

```
  avg_overall_performance,
  win_rate,
  matches_played
FROM public.user_stats_summary
WHERE matches_played >= 5;
```

---

5. Access Policies & Triggers

- RLS policies enabled for all user-facing tables
- `update_user_stats_summary()` trigger attached to `matches`
- Streaming reward and replay upload validations handled asynchronously
- Optional: insert access to some tables via LangGraph agent roles (n8n or backend-controlled)

---

6. Final Notes

- All metrics can scale with schema via `raw_json` and `custom_json`
- Designed for LangGraph orchestration, smart contract interfacing, and Supabase compatibility
- This schema guarantees performance, auditability, and token-aligned incentives