Project: Fantasy League Management System

Jesse Annan
002708111

Stage 4: Progress Report

I have reduce the number of columns from my initial table (eg: remove stadium) and added triggers to check stuffs like number of players in a team and check captaincy. I also found a better link to extract data for my project (Original FPL database) so I wrote a program to extract the match information for the current and ongooing season (25/26). Since there's been few games played the plan is to simulate the rest of the games and update the player stats. The simulation was written in python and currrently being tested (see Picture of the frontend). The frontend is also under development but currently I have a view of a demoXI I randomly created and for the initial points I used the total number of goals scored by a team (to be the same point as the player from the team) becuase I wanted to visualize that value on the frontend. This will be changed based on estimated player performance calculated by the simulation function.

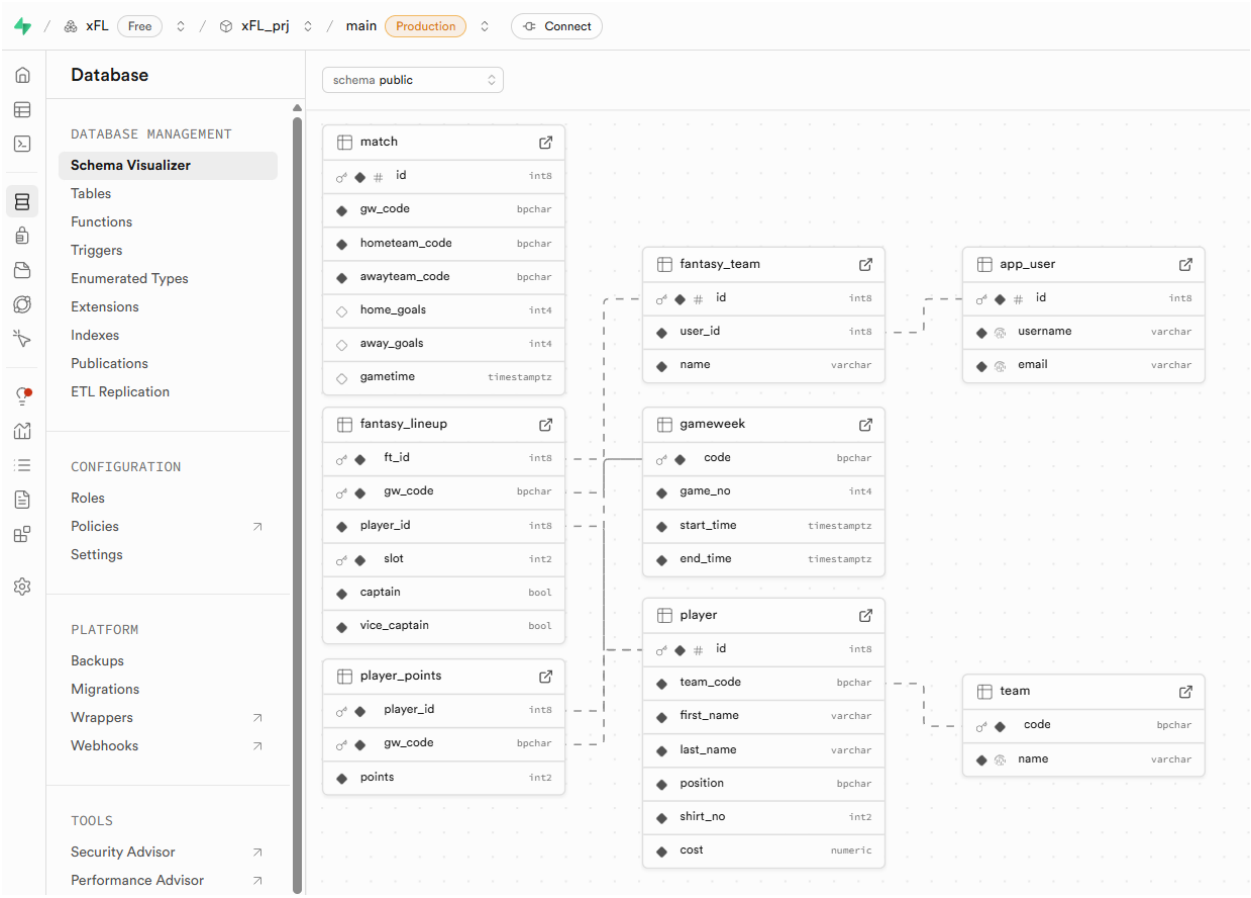

Figure 1: minimal supabase tables

Figure 2: Frontend current look

```python
# backend/simulate_gameweek.py

import random
from db import get_conn

def simulate_matches(gw_code: str):
    conn = get_conn()
    with conn:
        with conn.cursor() as cur:
            cur.execute("SELECT code FROM team;")
            teams = [t["code"] for t in cur.fetchall()]

            matches = []
            random.shuffle(teams)
            for i in range(0, len(teams), 2):
                if i + 1 < len(teams):
                    home, away = teams[i], teams[i + 1]
                    home_goals = random.randint(0, 4)
                    away_goals = random.randint(0, 4)
                    cur.execute("""
                        INSERT INTO match (gw_code, hometeam_code,
                            awayteam_code, home_goals, away_goals)
                        VALUES (%s, %s, %s, %s, %s)
                        RETURNING id;
                    """, (gw_code, home, away, home_goals, away_goals
                        ))
                    match_id = cur.fetchone()["id"]
                    matches.append((match_id, home, away, home_goals,
                        away_goals))

            print(f"{len(matches)} matches simulated for {gw_code}")
    conn.close()

def assign_player_points(gw_code: str):
    conn = get_conn()
    with conn:
        with conn.cursor() as cur:
            cur.execute("SELECT id FROM player;")
            players = [p["id"] for p in cur.fetchall()]
```

```
37
38              for pid in players:
39                  pts = random.randint(0, 15)
40                  cur.execute("""
41                      INSERT INTO player_points (player_id, gw_code,
                           points)
42                      VALUES (%s, %s, %s)
43                      ON CONFLICT (player_id, gw_code)
44                      DO UPDATE SET points = EXCLUDED.points;
45                  """, (pid, gw_code, pts))
46      conn.close()
47      print(f"Assigned random points for {gw_code}")
48
49  if __name__ == "__main__":
50      gw_code = "GW02"
51      simulate_matches(gw_code)
52      assign_player_points(gw_code)
53      print("Gameweek simulation complete.")
```

```sql
-- db/constraints.sql

-- 1) Exactly 11 players per fantasy team per GW
CREATE OR REPLACE FUNCTION check_lineup_11()
RETURNS TRIGGER AS $$
DECLARE
    cnt INT;
BEGIN
    SELECT COUNT(*) INTO cnt
    FROM fantasy_lineup
    WHERE ft_id = NEW.ft_id AND gw_code = NEW.gw_code;

    -- INSERT until it reaches 11, but not beyond
    IF TG_OP = 'INSERT' AND cnt > 11 THEN
        RAISE EXCEPTION 'Lineup for team % in GW % already has 11
           players', NEW.ft_id, NEW.gw_code;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_lineup_11
BEFORE INSERT ON fantasy_lineup
FOR EACH ROW
EXECUTE FUNCTION check_lineup_11();


-- 2) Enforce 1 captain and 1 vice_captain per (ft_id, gw_code)
-- approach: whenever we set captain=TRUE we check there is no other
CREATE OR REPLACE FUNCTION check_single_captain()
RETURNS TRIGGER AS $$
DECLARE
    cnt INT;
BEGIN
    IF NEW.captain THEN
        SELECT COUNT(*) INTO cnt
        FROM fantasy_lineup
        WHERE ft_id = NEW.ft_id AND gw_code = NEW.gw_code AND captain
```

```
            = TRUE
              AND (slot <> NEW.slot);  -- exclude self
        IF cnt > 0 THEN
            RAISE EXCEPTION 'Only one captain allowed per team per GW';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_single_captain
BEFORE INSERT OR UPDATE ON fantasy_lineup
FOR EACH ROW
EXECUTE FUNCTION check_single_captain();


CREATE OR REPLACE FUNCTION check_single_vice()
RETURNS TRIGGER AS $$
DECLARE
    cnt INT;
BEGIN
    IF NEW.vice_captain THEN
        SELECT COUNT(*) INTO cnt
        FROM fantasy_lineup
        WHERE ft_id = NEW.ft_id AND gw_code = NEW.gw_code AND
           vice_captain = TRUE
              AND (slot <> NEW.slot);
        IF cnt > 0 THEN
            RAISE EXCEPTION 'Only one vice-captain allowed per team per GW';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_single_vice
BEFORE INSERT OR UPDATE ON fantasy_lineup
FOR EACH ROW
```

```
75 EXECUTE FUNCTION check_single_vice();
```