# 1. Database Requirement Specification (for an “Offline Dinosaur Runner”-Style Game)

## 1) Overview

This project is a **side‑scrolling endless‑runner** inspired by the offline Dinosaur T‑Rex game. A player controls a character that runs continuously and must dodge randomly generated obstacles. The database persists gameplay and enables simple analytics and fair competition across seasons.

What the system stores:

- **Player Management:** Stores player information (registered or guest), account creation date, and last login time.
- **Season Tracking:** Organizes gameplay sessions into specific competition periods and marks which season is currently active.
- **Game Sessions:** Records every playthrough, including start/end times, total duration, score, distance, top speed, and reason for termination (collision, quit, or timeout).
- **Obstacles and Player Inputs:** Records every obstacle spawn event (type, timing, speed, cleared or not) and every player input (jump, duck, pause, etc).
- **Skins and Ownership:** Defines all available skins and tracks which skins each player owns, as well as which one they used during a session.
- **Player input events:** For example, jump/duck with precise timing for replay/debugging.
- **Achievements:** Stores all possible achievements and the conditions under which players unlock them, along with timestamps and related sessions.

---

## 2) Key Entities and Attributes

### 1. Player

- player_id (PK)
- username
- email
- password
- account_type (REGISTERED/GUEST)
- created_at: The date/time when the account was created.
- last_login_at: The last time the player logged in.

### 2. Season

- season_id (PK): Unique ID for each competition season.
- name: Season name.

- start_date
- end_date
- is_active: Shows whether this season is currently running.

**3. Skin**

- skin_id (PK):
- name
- rarity: Level of rarity (common, rare, etc.).
- is_default: True if it’s the basic skin every player has.

**4. PlayerSkin** (bridge: Player–Skin)

- player_id (FK)
- skin_id (FK)
- acquired_at: Time when the player got this skin.
- source (DEFAULT/ACHIEVEMENT/PURCHASE): How they got it — default, achievement, or purchase.
- composite PK = (player_id, skin_id)

**5. Session**

- session_id (PK)
- player_id (FK)
- season_id (FK)
- skin_id (FK, nullable): This column can be empty (Null), it’s optional.
- started_at, ended_at: When the game started and ended.
- duration_ms: Total play time in milliseconds
- score: Player’s score for this run.
- distance_m: How far the smile ran (in meters).
- top_speed: The fastest speed reached.
- crash_type (COLLIDE/QUIT/TIMEOUT): Why the game ended (hit obstacle, quit, etc.).
- obstacle_type_id (FK, nullable): What type of obstacle caused the crash.
- is_offline: True if played offline.
- device_type (browser)
- seed: Random seed number used to generate obstacles.

**6. ObstacleType**

- obstacle_type_id (PK): Unique ID for each kind of obstacle.
- name
- altitude (GROUND/AIR): Where it appears — on ground or in air.
- width_px, height_px: Size of the obstacle in pixels.

**7. ObstacleSpawn (Each time an obstacle appears during a session)**

- spawn_id (PK): Unique ID for each obstacle appearance.
- session_id (FK): The session this obstacle appeared in.
- obstacle_type_id (FK)
- t_offset_ms: Time in ms after the start of the session when it appeared.
- speed_at_spawn: Game speed when obstacle appeared.
- cleared (bool): True if the player avoided the obstacle.

**8. InputEvent (Player's actions like jumping or ducking)**

- input_event_id (PK)
- session_id (FK): The session where it happened.
- t_offset_ms: When (in ms) after the session started.
- action (JUMP/DUCK/PAUSE/RESUME)
- source (KEYBOARD): How the input was made (keyboard)

**9. Achievement**

- achievement_id (PK)
- name
- description: What the player must do to earn it.

**10. PlayerAchievement** (bridge: Player–Achievement)

- player_id (FK)
- achievement_id (FK)
- session_id (FK)
- unlocked_at: When it was unlocked.
- composite PK = (player_id, achievement_id).

---

## 3) Relationships

- **Player (1) — (0..N) Session**
  Each session must belong to exactly one player.

- **Season (1) — (0..N) Session**
  Each session must belong to exactly one season.

- **Skin (0..N) — (0..1) Session**
  A session may use at most one skin; a skin can be used in many sessions.

- **Player (0..N) — (0..N) Skin** via **PlayerSkin**
  Many-to-many ownership; each PlayerSkin row links exactly one player and one skin.

- **Session (1) — (0..N) ObstacleSpawn**
  Every obstacle spawn must belong to one session.

- **ObstacleType (1) — (0..N) ObstacleSpawn**
  Each spawn has exactly one obstacle type.

- **Session (1) — (0..N) InputEvent**
  Every input event belongs to exactly one session.

- **Player (0..N) — (0..N) Achievement** via **PlayerAchievement**
  Many-to-many unlocking; each PlayerAchievement links one player and one achievement.

- **Achievement (1) — (0..N) PlayerAchievement**
  An achievement can be unlocked by many players.

- **Session (1) — (0..N) PlayerAchievement**
  Each PlayerAchievement must reference the session during which it was unlocked.

- **ObstacleType (1) — (0..N) Session**
  A session may optionally reference one obstacle type as its crash cause; an obstacle type can be the crash cause for many sessions.

---

## 4) Business Rules

### R1. Session ownership & timing

- *Session.player_id* and *Session.season_id* are required.
- *ended_at ≥ started_at*
- *duration_ms = ended_at − started_at*
- Session.started_at must fall within the chosen season's date range *[start_date, end_date)*

### R2. Scoring & termination

- *score ≥ 0*
- *distance_m ≥ 0*
- *top_speed ≥ 0*
- *crash_type* should be *{COLLIDE, QUIT, TIMEOUT}*
- If *crash_type = COLLIDE*, then *obstacle_type_id* is required; otherwise it must be NULL

### R3. Event timing bounds

- For every *ObstacleSpawn* and *InputEvent*, *t_offset_ms* must be within *[0, Session.duration_ms]*
- *ObstacleSpawn.speed_at_spawn ≥ 0*

**R4. Skin usage**

- If *Session.skin_id* is not NULL, the player must own that skin before *Session.started_at*
- All players are automatically considered to own any skin where *is_default = TRUE*.

**R5. Player–Skin and Player–Achievement uniqueness**

- *PlayerSkin (player_id, skin_id)* is unique (composite PK).
- *PlayerAchievement (player_id, achievement_id)* is unique (composite PK).

**R6. Achievement unlocking consistency**

- The *PlayerAchievement.session_id* must reference a session that belongs to the same player.
- *unlocked_at* must lie within that session's *[started_at, ended_at]*.

**R7. Obstacle clearing semantics**

- *ObstacleSpawn.cleared = TRUE* means the obstacle was successfully avoided.
- If a session ends with *crash_type = COLLIDE*, then there must be exactly one spawn in that session with *cleared = FALSE*, and its *obstacle_type_id* must equal *Session.crash_obstacle_type_id*.
- If *crash_type != COLLIDE*, then all spawns in that session must have *cleared = TRUE*.

**R8. Player account constraints**

- *account_type* should be *{REGISTERED, GUEST}*.
- For *GUEST* players, *email* and *password* may be NULL; for *REGISTERED* players, *email* and *password* must be present.
- *email* (when present) and *username* should be unique across players.

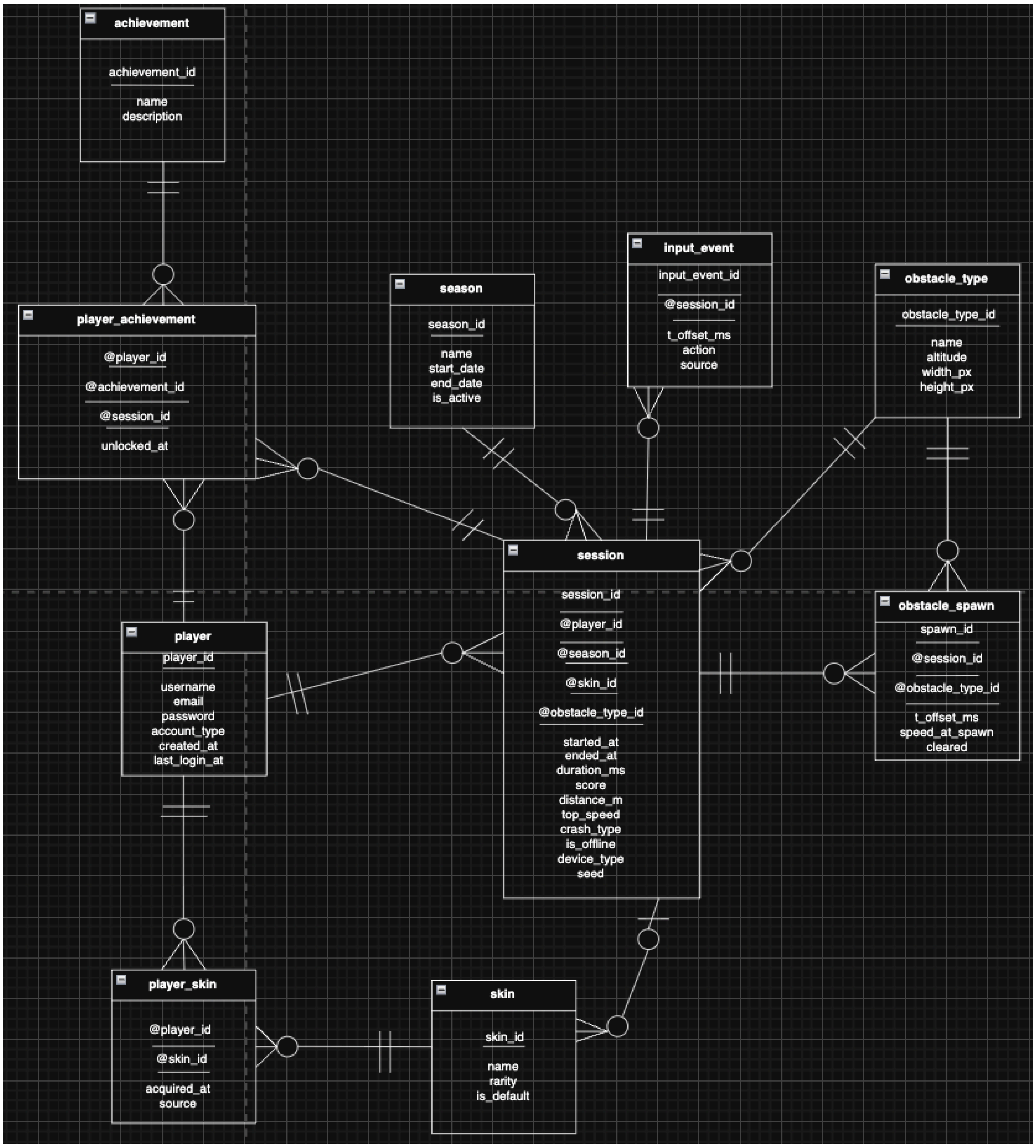
**R9. Referential integrity & deletions**

- Deleting a *Player*, *Season*, *Skin*, or *Achievement* is restricted if dependent rows exist (to preserve gameplay history).
- Deleting a *Session* is restricted if *PlayerAchievement* rows reference it.

**R10. Deterministic generation**

- *Session.seed* must be recorded;
- Given the same seed and version of the generator, the sequence of *ObstacleSpawn* events is reproducible.

## 5) Entity-Relationship Diagram (ERD)



## 6) Schema Statements

Player(<u>Player_id</u>, Username, Email, Password, Account_type, Created_at, Last_login_at)

Season(<u>Season_id</u> , Name, Start_date, End_date, Is_active)

Skin(<u>Skin_id</u>, Name, Rarity, Is_default)

Player_skin(<u>@Player_id</u>, <u>@Skin_id</u>, Acquired_at, Source)

Session(<u>Session_id</u>, <u>@Player_id</u>, <u>@Season_id</u>, <u>@Skin_id</u>, <u>@Obstacle_type_id</u>, Started_at, Ended_at, Duration_ms, Score, Distance_m, Top_speed, Crash_type, Is_offline, Device_type, Seed)

Obstacle_type(<u>Obstacle_type_id</u>, Name, Altitude, Width_px, Height_px)

Obstacle_spawn(<u>Spawn_id</u>, <u>@Session_id</u>, <u>@Obstacle_type_id</u>, T_offset_ms, Speed_at_spawn, Cleared)

Input_event(<u>Input_event_id</u>, <u>@Session_id</u>, T_offset_ms, Action, Source)

Achievement(<u>Achievement_id</u>, Name, Description)

Player_achievement(<u>@Player_id</u>, <u>@Achievement_id</u>, <u>@Session_id</u>, Unlocked_at)

---

## 7) Assumptions and Justifications

- Only Registered players require email and password; Guest players may leave these fields null.
- Sessions always reference a valid player and season, ensuring all game runs are properly tracked and grouped.
- Composite PKs in bridge tables (PlayerSkin, PlayerAchievement) ensure only one row per relationship.
- All tables are normalized up to 3NF: 1NF: All fields are atomic and single-valued. 2NF: No partial dependencies in tables; bridge tables use full composite PK. 3NF: No transitive dependencies; all non-key fields depend only on the full PK.

---