Project: Designing a Telemetry and UX Database for *Chocolate-Doom* Research

DBS – Semester Project Brief

September 23, 2025

Contents

1 Context and Motivation

A research group is collecting gameplay telemetry from a *hacked* version of `chocolate-doom`. This build emits per-tic data (on screen and/or to file) that includes the player position (x, y, z), facing *angle*, *momentum* vector, *point-of-view* (FOV/camera), and combat stats (health, armor, ammo). It also logs meta-information such as *tic* number, *episode*, *map*, and *sector*.

The group wants to aggregate multiple play sessions to detect movement trends and potential cooperation patterns among players. Student volunteers provide demographics (age, gender, experience) and complete *one* of these UX instruments:

- **PENS** (Player Experience of Need Satisfaction).
- **GUESS** (Game User-Experience Satisfaction Scale).
- **BANGS** (Basic Needs in Games; open-access).

Goal: Design and prototype a relational database that ingests telemetry and survey data, supports exploratory queries and analytics for movement/cooperation trends, and enforces data quality and research ethics.

2 Learning Objectives

By completing this project, you will:

1. Model a real-world domain into entities, attributes, and relationships (ER & relational schema).
2. Normalize tables to at least 3NF (justify any denormalizations for performance).
3. Define keys, constraints, and reference integrity for high-frequency telemetry.
4. Design an ingestion pipeline for semi-structured logs (TSV → staging → core).
5. Implement indices and assess their impact with query plans and timings.
6. Formulate SQL queries for trajectory, proximity, and cooperation analyses.
7. Integrate user demographics and UX scales (PENS/GUESS/BANGS) with telemetry.
8. Address privacy, consent, and research-ethics constraints in schema & process.

3 Domain Overview and Core Concepts

Proposed high-level entities:

- **User**: volunteer student providing consent and demographics.
- **Player**: in-game identity (may be linked 1:1 to a User or support multiple aliases).
- **Game**: a single gameplay session instance (start/end timestamps and settings).
- **Time/Tic**: per-tic (or per-frame) temporal index emitted by the engine.
- **Episode/Map/Sector**: level structure; sectors partition maps.
- **TelemetryEvent**: atomic record of state at a tic (position, momentum, stats, and more).
- **UXInstrument**: instrument metadata (PENS/GUESS/BANGS definitions).
- **UXResponse**: a user's instrument responses.

Movement & Cooperation Signals (to inform schema/queries).

- *Trajectories*: sequence of (x, y, z) ordered by tic per player per game.
- *Proximity events*: players within a spatial threshold for $\geq k$ tics.
- *Co-occurrence in sectors*: overlapping time in same sector (optional: adjacent sectors).

4 Data Model (Conceptual →Logical)

Conceptual ER (deliverable)

Produce an ER diagram capturing main relationships. For example:

- User–Player,
- Player–Game (via GameParticipant),
- Game–TelemetryEvent,
- Map–Sector (1:many),
- User–UXResponse,
- UXInstrument–UXItem–UXResponseItem .

Indexing Suggestions (implement and evaluate).

```
CREATE INDEX ON TelemetryEvent (game_id, player_id, tic);
CREATE INDEX ON TelemetryEvent (episode, map_code, sector_id);
CREATE INDEX ON TelemetryEvent USING gist ((pos_x, pos_y));
CREATE INDEX ON GameParticipant (player_id, game_id);
```

5 Data Ingestion (ETL) Guidance

Assume the hacked engine emits TSV lines.
Recommended pipeline:

1. Load raw logs to a *staging* table (text fields) with minimal constraints.
2. Validate & transform into typed core tables using `INSERT ... SELECT`.
3. Deduplicate on `(game_id, tic, player_id)`; reject malformed records with an *error log* table.

6 Analytics Queries (Examples to Implement)

Movement Trends

TBA

Linking UX to Behavior

TBA

7 Project Tasks & Deliverables

Part A: Conceptual and Logical Design (Week 1–3)

1. Write assumptions and requirements (functional/non-functional, ethics).
2. Produce an ER diagram with cardinalities and key attributes.
3. Derive relational schema; list all FKs, PKs, and constraints (Data Dictionary); justify normalization.

Part B: Implementation & Ingestion (Week 4–6)

1. Implement DDL in your DBMS (PostgreSQL recommended).
2. Create staging tables and scripts to load sample telemetry logs (TSV).
3. Populate `UXInstrument`, `UXItem` with at least one instrument (PENS/GUESS/BANGS).
4. Insert synthetic sample data (at least 3 games, 6+ players, $\geq$ 20k telemetry rows).

Part C: Queries, Indexing, and Reporting (Week 7–9)

1. Implement at least 8 analytical queries including: trajectory steps, sector heatmap, proximity/cooperation runs, health under proximity, ammo usage patterns, player hotspots, per-player summary, and UX-behavior link.
2. Create at least 3 indexes. Show `EXPLAIN(ANALYZE)` before/after and discuss.
3. Provide 2 views and 1 materialized view for frequent analyses.
4. Provide a `Makefile` or shell script to recreate the schema and load samples.

8 Submission Format

Submit a single PDF report with:

- **ER diagram**, **relational schema**, and **rationale**.
- **DDL/constraints** (appendix with code snippets).
- **ETL description** + sample of raw telemetry.
- **Queries + results** (screenshots/tables) and index evaluation.
- **Ethics note** and **data dictionary**.

9 Grading Rubric (100 pts)

Criterion	**Points**
Problem framing, assumptions, requirements clearly stated	10
Conceptual ER correctness (entities, keys, cardinalities) & data dictionary	20
Relational design & normalization (3NF), constraints	15
Implementation quality (DDL, integrity, sample data)	10
ETL pipeline (staging → core, validation)	10
Analytics queries (8+) correctness & insight	15
Indexing & performance evaluation (EXPLAIN/ANALYZE)	10
Views/materialized view for reuse	5
Report quality (clarity, organization, reproducibility)	5
Total	**100**

10 Bonus

- If available, enable extensions like `citext`, `uuid-ossp` or `pgcrypto` for authorization or `postgis` for spatial indexing.