

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



DBS – Semester Project Brief

November 22, 2025

Contents

1	Context and Motivation	1
2	Learning Objectives	2
3	Domain Overview and Core Concepts	2
4	Data Model (Conceptual → Logical)	3
5	Data Ingestion (ETL) Guidance	3
6	Analytics Queries	3
7	Project Tasks & Deliverables	4
8	Submission Format	5
9	Grading Rubric (100 pts)	6
10	Bonus	6

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 \rightarrow staging \rightarrow 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 \rightarrow 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

Implement SQL solutions for any five (5) of the eight (8) proposed analytical queries. *Note: The provided hints utilize generic table and attribute names; you are required to adapt these to the specific nomenclature defined in your database schema.*

1. Average duration of game sessions per map.
Hint: Use the ‘Game’ table and the AVG aggregation function on the ‘timestamp’ or ‘tic’ field, grouped by the ‘map.id’.

2. Players with the highest average proximity.
Hint Perform a self-join on the 'TelemetryEvent' table (matching on 'game_id' and/or 'tic' – actually I am not sure) to calculate the *Euclidean distance*¹ between player pairs, then filter for proximity thresholds (maybe ≤ 5.0 ?) and aggregate the results to find the pairs with the highest average closeness.
3. Shortest and longest trajectory distances per player.
Hint Calculate the total length of each game's trajectory by summing the Euclidean distances between consecutive tics (using a self-join or *window function*²), then group by player to find the minimum and maximum of those totals.
4. List UX survey responses for players with above-average trajectories duration.
Hint Calculate the average duration of all trajectories (using total tics or time differences), then use a subquery to select players whose individual duration exceeds this global average and JOIN them to the 'UXResponse' table to retrieve their survey answers.
5. Most Visited Sector (Hotspot) per Episode and Map.
Hint Group the 'TelemetryEvent' records by 'episode', 'map', and 'sector'. The sector should be a cell in a grid of a 250×250 units covering the current map. Then use the COUNT function to calculate the frequency of player presence in each area, sorting the results in descending order to identify the hotspots.
6. Number of Tics Where Players Were Together in a Sector.
Hint Perform a self-join on the 'TelemetryEvent' table matching records by 'game_id', 'tic', and 'sector_id' to identify co-presence while filtering for distinct 'player_ids', and then apply COUNT(DISTINCT tic) to calculate the total duration of these overlapping moments.
7. Average UX Score for Players with the Shortest Trajectory per Episode.
Hint Identify the players with the minimum total trajectory distance for each episode (using a *window function* or subquery), then JOIN those players to the 'UXResponse' table to calculate the average of their survey scores.
8. Total Distance Traveled and Average Speed per Player, Analyzing All Games for a Player.
Hint Sum the Euclidean distances between consecutive tics across all sessions to determine the total distance, then divide this by the total duration (calculated from the aggregated tic counts or game timestamps) to derive the average speed, grouping the results by player identity.

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.

¹https://en.wikipedia.org/wiki/Euclidean_distance

²If you use window functions, that will be a bonus.

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 5 analytical queries (see Section ??).
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-oss` or `pgcrypto` for authorization or `postgis` for spatial indexing.

