COMM2020 Team Project
Project Specification (Project 7)
Indie Game Telemetry and Fair Progression Balancer

Usage of AI Tools: AI-Minimal

These tasks have been set to assess your problem-solving abilities, and so you may not use AI for generating code or autocomplete while completing the project.

The University of Exeter is committed to the ethical and responsible use of Generative AI (GenAI) tools in teaching and learning, in line with our academic integrity policies where the direct copying of AI-generated content is included under plagiarism, misrepresentation and contract cheating under definitions and offences in TQA Manual Chapter 12.3.

This assessment falls under the category of AI-Minimal in the University's Guidance on use of Gen AI in Assessment.

This means: You may use AI tools for checking spelling and grammar mistakes only, with no other impact on the structure or content of the assessment. This is because using GenAI tools outside of these uses prevents fair assessment of your ability to achieve module learning outcomes.

When writing your assessment, you must never use AI tools:

- For uses other than checking your spelling and grammar.
- To translate more than a word or short phrase into English.
- To upload sensitive or identifying material to an AI tool.
- To present material that has been generated by AI as your own work or the work of someone else.

When submitting your assessment, you must:

- Check the box during the submission process that confirms you have adhered to the university's academic conduct policy and the expectations on use of GenAI in your assessment brief.

NOTICE: Your coding activity must all be completed on the GitHub repository. This logs your activity as you code, and the logs will be checked for submissions that have made use of AI-generated code.

Guidelines

- This assessment is AI-Minimal: spelling/grammar checks only; no AI code generation or autocomplete.
- Your application will be demonstrated online. Provide a deployed URL for the live demo and a clear local run guide for markers.
- Your game should be safe and appropriate for a University context. Avoid gambling-like mechanics and avoid pay-to-win patterns.
- Your system must remain usable with a seeded dataset even if external services are unavailable.
- If you use third-party services, handle failures gracefully and avoid embedding secrets (use environment variables).

1 Requirements

Build an online game experience and a telemetry analytics dashboard that helps designers answer: 'Is progression fair and enjoyable, and where do players get stuck or disengage?'

1.1 The practical problem

Game designers often rely on intuition and small playtests. Without telemetry, it is hard to locate difficulty spikes, grindy progression, or features that create inequity between play styles (e.g., cautious vs aggressive players).

Your system addresses this by providing:

- A game that produces structured telemetry events (levels, deaths/fails, resources earned/spent, time per stage).
- A dashboard that visualises funnels, drop-off points, and fairness indicators.
- A balancing toolkit that proposes parameter changes (not AI; rule-based) and simulates the impact on progression curves.

1.2 Game format (required)

Your game must be either: (A) a small browser-playable game, OR (B) an interactive simulation with a playable 'turn-based' interface. The game must include at least 10 stages/levels or progression steps and at least 3 mechanics that influence success.

- Examples: puzzle dungeon with stamina; tower defence with resource economy; platformer-like sequence with checkpoints; turn-based roguelike encounters.

You must seed at least 3 difficulty configurations (easy/balanced/hard) to support evaluation and balancing.

2 Users and user profiles

Your application must support three user types:

2.1 Players

- Play the game and receive feedback on progression and achievements.
- Optionally opt into telemetry (default on for this coursework but clearly disclosed).
- View a simple post-run summary (time, deaths/fails, resources, completion).

2.2 Designers/Analysts (trusted users)

- View dashboards and compare difficulty configurations.
- Inspect funnels and drop-off points by stage.
- Run ‘what-if’ balancing scenarios and record balancing decisions with rationale.

2.3 Maintainers (developers)

- Maintain deployments, data pipelines, and testing.

3 Features

3.1 Telemetry capture (required)

- Event logging: record at least 12 event types (e.g., stage_start, stage_complete, fail, retry, resource_gain, resource_spend, item_pickup, boss_start, boss_fail, boss_win, quit, settings_change).
- Each event must include: user/session ID (pseudonymous), timestamp, stage ID, and relevant parameters.
- Data quality checks: detect missing fields, impossible sequences (e.g., complete without start), and out-of-range values.
- Export: designers can export telemetry to CSV from the dashboard.

3.2 Analytics dashboard (required)

- Funnel view: stage-by-stage completion funnel showing drop-off and failure rates.
- Difficulty spikes: automatically highlight stages with unusually high fail/time rates.
- Progression curves: show distributions of time-to-complete and resource accumulation.
- Fairness indicators: compare at least 2 player segments (e.g., fast vs slow, cautious vs aggressive inferred from behaviour metrics) and report differences.
- Comparison mode: compare easy/balanced/hard configurations on key metrics.

3.3 Balancing toolkit (required)

You must implement a non-AI balancing toolkit that supports designers to adjust parameters and immediately see predicted impacts.

- Parameter editor: adjust at least 8 balancing parameters (e.g., enemy HP, reward rates, stamina regen, checkpoint frequency, resource costs).
- Rule-based suggestions: at least 6 rules that generate suggestions (e.g., 'if stage fail rate > 40% and median time > threshold, reduce enemy HP by X%').
- Simulation mode: run a lightweight simulation using your seeded telemetry or synthetic agent players to estimate new progression curves.
- Decision log: designers must record balancing decisions with rationale and link them to evidence in the dashboard.

3.4 Technical requirements (required)

- Authentication: role-based access control (player/designer).
- Security: secrets via environment variables; no secrets in repository or ELE ZIP; telemetry should be pseudonymous and minimise personal data.
- Accessibility: keyboard navigation for core screens, readable contrast, and mobile-friendly layouts.
- Professional practice evidence: use GitHub commits/issues/PRs; the repository URL must be included in 0_admin/submission.txt.
- Testing (mandatory): you must submit BOTH (A) an automated test suite that can be executed by the markers, AND (B) a manual end-to-end test plan with results evidence.
- Automated tests (A) must include at least 15 automated tests covering: telemetry event validation, data ingestion/storage, dashboard computations, balancing toolkit rules, and exports.
- Manual tests (B) must include at least 8 end-to-end scenarios (happy path + failure cases). Include expected results and screenshots/logs of completed runs in 4_technical/testing_evidence.pdf.
- Your deployment_guide.pdf must include a 'How to run tests' section with the exact commands/steps.

4 Data and seeded dataset

You must include a seeded dataset so the system works during marking without requiring external services. The game must be playable and the dashboard must show meaningful analytics using seeded telemetry.

4.1 Minimum dataset contents (required)

- At least 1,500 telemetry events seeded across at least 80 sessions and 40 users (pseudonymous).
- At least 10 stages/levels, each with parameters and completion criteria.

- At least 3 difficulty configurations (easy/balanced/hard), with telemetry for each.
- At least 30 seeded balancing decisions in a decision log (some good, some flawed) to demonstrate auditing.
- At least 150 seeded 'data quality' anomalies to demonstrate validation and cleaning (e.g., missing fields, impossible event order).

4.2 Suggested entities (example)

Entity	Key fields (minimum)
User	user_id (pseudonymous), created_at, segment_tags(optional)
Session	session_id, user_id, start_time, end_time, config_id, outcome
Stage	stage_id, name, parameters, completion_rule
Event	event_id, session_id, timestamp, stage_id, event_type, payload(parameters)
Config	config_id, label, parameter_set
Metric	metric_id, name, definition, aggregation
BalancingRule	rule_id, trigger_condition, suggested_change, explanation
DecisionLog	decision_id, config_id, stage_id(optional), change, rationale, evidence_links, timestamp
Anomaly	anomaly_id, event_id, anomaly_type, detected_by, resolution_status

5 Measures of success (evaluation)

You must define and report success measures with evidence. Suitable measures include:

- Detection validity: do your spike-detection rules correctly flag known problematic stages in your seeded data?
- Fairness analysis: do segments show different outcomes? Are differences explained and mitigations proposed?
- Balancing effectiveness: do suggested parameter changes reduce spikes in simulation and/or in a small pilot?
- Usability: can designers locate issues and make an informed change within a short task timeframe?

6 Process and deliverables

You must submit all group deliverables on ELE as a single ZIP. GitHub is used to evidence professional practice and must be referenced in submission.txt.

Intended Learning Outcomes assessed by the coursework

Coursework 1 and Coursework 2 assess all module Intended Learning Outcomes (ILOs):

- ILO1 – Function effectively as a member of a team.
- ILO2 – Apply an integrated or systems approach to the solution of complex problems.
- ILO3 – Apply knowledge of domain context, project and change management, and relevant legal matters including intellectual property rights.
- ILO4 – Select and apply appropriate materials, technologies, and processes, and recognise their limitations.
- ILO5 – Plan self-learning and development to support the activity of the wider team.
- ILO6 – Support an inclusive approach to teamwork and problem solving, recognising the responsibilities, benefits and importance of supporting equality, diversity, and inclusion.
- ILO7 – Evaluate the environmental and societal impact of solutions to complex problems and minimise adverse impacts.

Evidence is expected across reports, implementation, evaluation, ethical/legal materials, and teamwork/process artefacts in both sprints.

6.1 Sprint 1 (Coursework 1) – Prototype v1, report and demonstration (week 5)

Submission to ELE is a single ZIP file named GroupX_CW1.zip.

Sprint 1 deliverable expectations

- Prototype v0.1.0 (vertical slice): play at least 2 stages → events logged → dashboard shows funnel for those sessions → designer adjusts one parameter and sees updated simulation/prediction.
- At least 6 telemetry event types implemented in Sprint 1; at least 3 dashboard views implemented.
- Testing evidence (CW1): include at least 5 automated tests running successfully and document how to run them in deployment_guide.pdf; include one manual end-to-end test run with evidence in testing_evidence.pdf.
- Prototype report must include: executive summary, prioritised requirements, architecture v1, telemetry schema, initial evaluation evidence, and a sprint plan for CW2.
- Ethical/legal considerations must cover privacy of telemetry, consent/disclosure, accessibility, and IP/licensing implications.
- Software/data inventory must list all dependencies and datasets/assets (licence, provenance, cost model, versions).

Live demo and presentation (10 minutes) – what to cover (CW1)

1. 30 seconds: what your game is and what balancing problem you solve.
2. 6–7 minutes: demo playing, telemetry capture, and key dashboard views.
3. 1–2 minutes: show one balancing rule suggestion and parameter change effect.
4. Final minute: what will be completed in Sprint 2 and your top risks.

6.2 Sprint 2 (Coursework 2) – Final prototype, client handover and presentation (week 11)

Submission to ELE is a single ZIP file named GroupX_CW2.zip.

Individual reflection (submitted separately by each student on ELE): reflection.pdf (800–1,000 words; includes evidence links to commits/issues/PRs; and an AI-Minimal compliance statement).

Sprint 2 deliverable expectations

- Final prototype v1.0.0: playable game with 10+ stages; full telemetry event set (12+); complete dashboard views including fairness indicators and config comparisons.
- Balancing toolkit: parameter editor, 6+ rule-based suggestions, simulation mode, and decision log tied to evidence.
- Client handover pack: clear deployment, operations, and maintenance guidance so another team could run and extend the system.
- Testing evidence (CW2): meet the full testing requirement (15+ automated tests + 8+ manual scenarios) and include clear pass/fail evidence in testing_evidence.pdf. Marks will be reduced if tests cannot be run by markers.
- Final evaluation: report success measures with method and limitations; include a discussion of fairness and unintended consequences in game design.
- Updated ethical/legal and licensing materials consistent with the final system and all dependencies.

Live demo and presentation (10 minutes) – what to cover (CW2)

5. 1 minute: recap game concept and what is now delivered.
6. 6–7 minutes: demo final gameplay + dashboard + one balancing scenario.
7. 1–2 minutes: show handover pack and how a maintainer would add a new event type or new stage.
8. Final minute: evaluation highlights, limitations, and next steps.

Individual reflection (Coursework 2 – individual deliverable)

Each student must submit an individual reflection on ELE (not in the group repository). Suggested length: 800–1,000 words. This reflection is used to evidence individual learning and contribution and may be used to resolve contribution disputes.

9. Your role and contributions: describe what you owned (features, testing, documentation, deployment). Reference concrete evidence (PR links, issue IDs, commits).
10. What you learned: at least three specific technical or professional learning points linked to module outcomes (e.g., requirements negotiation, risk management, testing, deployment).

11. Challenges and how you addressed them: one technical challenge and one teamwork/process challenge; what changed as a result.
12. Responsible computing: what ethical/legal risk you personally focused on (privacy, accessibility, safety) and how you mitigated it.
13. AI-Minimal compliance statement: confirm you adhered to the brief and did not use GenAI for code generation or content generation beyond spelling/grammar checks.

7 Safety and responsible use

- Do not include gambling mechanics, real-money transactions, or manipulative dark patterns.
- Telemetry must be transparent to players; provide a short explanation of what is collected and why.
- Avoid collecting personal data; use pseudonymous identifiers only.

8 Marking Rubric

The same COMM2020 marking rubric applies across all project options. The rubric will be provided by the module team on ELE and used consistently for CW1 and CW2 (including the individual reflection in CW2).

[END OF SPECIFICATION]

Document owner: Module team (COMM2020)

Date: 23 December 2025