Analysis & Advice

Prompt Battle WebGame

Background

I want to build a small but rigorous web game called Prompt Battle. Players receive the same AI-generated image. Each player writes a short prompt within a strict character limit that they believe could have produced that image. The game then compares each player prompt with the original image prompt and calculates similarity and accuracy. After the round, players see the original prompt, their own prompt, a similarity score, and a brief breakdown of which parts matched or missed.

This document is my analysis and advice before design and implementation. I follow the Analyse → Advise → Design → Realise → Validate flow that we use in LO2. The structure and expectations are based on our Analysis and Advice workshop materials and the example document from class.

Personal trajectory and rationale

The idea evolved through three iterations.

First I explored a prompt-injection puzzle where the player tries to subvert a guarded assistant.

Second I switched to prompt-engineering puzzles in a Unity 2D format with drag and drop prompt pieces.

Now I am converging on a focused web experience that measures how precisely players can reconstruct a hidden prompt from an image under a character limit.

The web approach reduces production risk, fits my timeframe, and keeps the learning goal clear. I want to learn to analyse a problem space, advise on technical choices, and deliver a small professional product that I can validate against measurable criteria, which aligns with LO2.

Project description

Prompt Battle is a synchronous or asynchronous round-based web game.

A round uses one hidden source prompt and a corresponding image. Players submit one prompt attempt under a configurable character limit. The system computes similarity between each attempt and the source prompt using a transparent method. A scoreboard ranks players by score. A results screen reveals the source prompt and highlights overlaps and missing concepts.

Problem context

What problem the game addresses

1. Many beginners practice prompt writing without structured feedback.
2. Communities enjoy competitions but lack an objective way to compare prompts on the same target.
3. Learners benefit from seeing ground-truth prompts and a specific explanation of accuracy.

Who the stakeholders are

1. Players who want to practice and have fun.
2. Educators or clubs who want a controlled exercise for workshops.
3. Me as the developer who wants to demonstrate analysis, advice, professional product creation, and validation aligned with LO2.

Why this is valuable

1. It turns prompt craftsmanship into a measurable and replayable exercise.
2. It gives immediate learning feedback through side-by-side comparison.
3. It can scale to events and classrooms by rotating datasets and tracking leaderboards.

How it works at a high level

1. Select or generate an image with a known ground-truth prompt.
2. Present the image and a countdown with a character limit.
3. Collect player prompts and lock them at time zero.
4. Compute similarity to the ground-truth prompt.
5. Show ranking and an explanation view.
6. Store anonymised results for analytics and improvement.

Where it will run
Web app on desktop and mobile browsers. Optional projection mode for classroom or club play.

Scope

In scope for this iteration

1. Core game loop for one round with one image.
2. Lobby for private room play with up to eight players.
3. Prompt submission with character counter and timer.
4. Server side scoring and anti-cheat checks for obvious copy paste of revealed text.

5. Result screen with original prompt and highlighted overlap.
6. Simple leaderboard with per-round history stored.
7. Information page that explains rules and the learning goal, inspired by the way information sites are handled in the example.

Out of scope for this iteration

1. Public matchmaking across the internet.
2. Complex social features such as friends, chat, or profiles.
3. Monetisation.
4. Mobile native app stores.

Requirements

Functional musts

1. Create rooms and join via code.
2. Start a round with one hidden source prompt and image.
3. Enforce a character limit during typing with a visible counter.
4. Submit prompts and lock inputs when time is up.
5. Compute similarity score and ranking.
6. Reveal original prompt with comparison highlighting.
7. Persist basic round data for later validation.

Quality musts

1. Round trip latency that keeps a 60 to 90 second round responsive.
2. Clear and readable UI on phone and laptop.
3. Transparent scoring explanation so players trust the result.
4. GDPR-friendly storage for EU use.
5. Accessibility basics, including keyboard navigation and contrast.

MoSCoW prioritisation follows the workshop advice for concrete, testable requirements.

Game design details

Core loop

1. Lobby and readiness check.
2. Round brief with image and rules.
3. Prompt drafting under a visible character limit and countdown.
4. Server evaluates submissions.
5. Results screen with scores, ranking, and explanation.
6. Option to continue with a new image.

Scoring approach

I will start with a hybrid method that is explainable.

1. Token overlap on key nouns and adjectives after stop-word removal.
2. Weighted exact match for artist names, styles, models, and parameters if present.
3. Synonym and lemma match using a static vocabulary for transparency.
4. Optional embedding similarity as a tiebreaker with a clear note that it is an approximation.
   The result is a score from zero to one hundred with a short textual explanation and highlighted spans.

Fair play and anti-cheat

1. Submissions are hidden from other players until lock.
2. Rate limit on edits close to the deadline.
3. Server side verification that the original prompt was never sent to clients before reveal.
4. Option for teacher mode that disables external links during the round in the classroom environment.

Content policy

1. Curate a safe dataset of images and prompts.
2. Filter banned terms in user prompts and flag rounds for moderation.
3. Provide a report button.

Dataset plan

1. Start with a curated set of safe prompts and corresponding images generated offline.
2. Each prompt is labelled with entities such as objects, styles, camera terms.
3. For validation I will include easy, medium, and hard rounds with clear answer keys.

Competitor and reference analysis

Closest neighbours

1. Prompt engineering leaderboards and daily prompt challenges on community sites.
2. Drawing guessing games where a prompt is revealed at the end.
3. Kaggle-style evaluation pages that show metric definitions.

Differentiation

1. Strict character budget that forces trade-offs.
2. Transparent accuracy explanation rather than a black box score.
3. Private rooms for classes and clubs with round history for reflection.

Technology options and advice

Frontend

A small information and gameplay site does not strictly need a large frontend framework. The class example argued similarly for a simple website when scope stays basic. I will start with Vanilla TS plus Vite and add a lightweight component library only if necessary.

Backend

I need real time rooms and a simple database. Workshop materials position the selection as part of analysis and advice. I compare three options.

1. Supabase
   Realtime channels for rooms, Postgres for results, row level security, quick auth for private rooms.
   Recommendation for MVP because it balances speed and data integrity.

2. Firebase
   Fast to set up and has Realtime Database or Firestore.
   Suitable fallback if I need simpler client sync.

3. Node with WebSocket and a managed Postgres
   Maximum control but more setup and security surface.
   Suitable for a later phase when I need custom scoring pipelines.

Scoring service

I will keep the primary scoring method deterministic and hosted with the backend to support transparent validation. If I add embedding similarity as a tiebreaker, I will encapsulate it behind a clear interface so it can be disabled in classroom mode.

Hosting and deployment

Use a managed platform with EU data location. Netlify or Vercel for frontend. Supabase in the EU region for data.

Risks and mitigations

1. Ambiguous ground-truth prompts create frustration
   Mitigation: curate prompts with clear entities and styles. Provide explanation highlighting to show what counted.
2. Players feel the score is unfair
   Mitigation: publish the metric, show token matches, and keep the algorithm deterministic for the first release.

3. Time overrun due to real time complexity
   Mitigation: start with asynchronous rounds where players submit within a window, then add live rooms.
4. Scope creep into social features
   Mitigation: stick to MoSCoW and the LO2 flow so I can validate on time.

Validation plan

What I will measure

1. Functional checks that every must requirement works.
2. Round completion rate without errors.
3. Player trust score gathered through a short post-round question.
4. Correlation between score and labelled entities to ensure the metric behaves logically.
5. Performance timing for submission and scoring under eight players.

How I will validate

1. Unit tests for scoring functions and tokenisation.
2. End-to-end tests for lobby, round, and results.
3. Pilot session with classmates. Collect feedback and bug reports as recommended in example validation sections.

Proposed planning

Phase 1. Analysis and Advice
Clarify scope, choose stack, select dataset slice, and define scoring metric.
Phase 2. Design
Wireframes for lobby, round, and results. Data model for rooms and rounds.
Phase 3. Realisation
Implement asynchronous rounds first. Add live countdown later. Prepare curated dataset.
Phase 4. Validation
Unit tests, small pilot, bug fixing, and documentation.
This mirrors the phased approach encouraged in the workshop.

MoSCoW

Must have

1. Room creation and join by code
2. One image per round and hidden source prompt
3. Character limit and countdown

4. Server evaluated score and ranking
5. Results screen with original prompt and highlights
6. Stored round data for validation

Should have

1. Simple teacher mode with non public rooms
2. Export of round summary as text for reports
3. Basic anti-cheat checks

Could have

1. Embedding similarity as tiebreaker
2. Public daily challenge mode
3. Simple leaderboard across sessions

Will not have in this iteration

1. Chat and friends
2. Matchmaking
3. Monetisation

Architecture sketch

Data model

1. Users stored as anonymous session ids or short names per room.
2. Rooms with code, status, and member list.
3. Rounds with image id, timer, and source prompt reference.
4. Submissions with text, tokens, and computed score.
5. Results with ranking and explanations.

APIs

1. Create room and join room.
2. Start round with image id.
3. Submit prompt.
4. Close round and compute scores.
5. Fetch results and history.

Ethics and privacy

1. No storage of personal data beyond a temporary nickname unless a user explicitly creates an account for classroom reuse.
2. Images and prompts curated to avoid harmful content.

3. Clear terms explaining that prompts may be logged to compute accuracy and to improve the metric.

Links to learning outcomes

LO2 Creating professional IT products

I analyse the context, advise on technology and scope, design the loop and data model, realise a minimal but complete product, and validate it with tests and a pilot. The workshop frames these as the five activities within LO2.

LO3 Professional standard

I write a project plan, define measurable requirements, document choices, and produce testable results that can be reviewed. The example shows how to make analysis explicit and reviewable.

LO1 Orientation and growth

I reflect on my idea pivots from injection puzzles to Unity puzzle to this focused web game and explain why this scope fits time and learning goals.

LO4 Personal leadership

I will present pilots, gather feedback, and iterate with a clear MoSCoW list and risks, demonstrating ownership and reflection.

Advice summary

Build a small, explainable, round-based web game before adding real time features. Use Supabase for rooms, results, and security. Keep the scoring method transparent and deterministic at first so validation is credible. Start with curated prompts and images. Validate through a pilot and publish the metric description next to the game to build trust. This path gives me a realistic professional product that I can design, realise, and validate within the semester while meeting the expectations set in our workshop and example materials.