AI-assisted Game Content Design

USER MANUAL

User Manual for AI-Assisted Game Creation

How to create a game with the AI Assistant, generate and refine the structured JSON with an external AI model, validate it, add media, publish the game, and configure the emerging player-facing AI features.

OPEN THE MANUAL ↓

STATUS

Version 0.1 | April 2026

Living document: interface labels and AI integrations may evolve during prototyping.

PROJECT

Produced by PP5 SEALS
Task 3.2.2 · AI-driven Game Platform Prototype · REBOOT · Erasmus+ KA220-HED · Project No. 2025-1-CY01-KA220-HED-000351980

REBOOT
AI ASSISTANT
GAME BUILDER

User Manual

VERSION 0.1 · APRIL 2026

How to create a game with the AI Assistant, generate and refine the structured JSON with an external AI model, validate it, add media, publish the game, and configure the emerging player-facing AI features.

> **IMPORTANT ARCHITECTURE POINT**
>
> The Create Game workflow itself does not require a paid LLM API. The platform prepares the structured JSON in PHP and the author can use a normal web chat with an AI model to fill it. Player-facing AI assistants are a separate layer and may have different service, account or cost requirements. Any AI you can use.

Fastest route through the manual: if you already know the platform, read Sections 4, 7, 8, 9 and 10. These sections contain the complete JSON-to-published-game workflow.

MANUAL AT A GLANCE

Four Stages, Sixteen Sections

BEFORE YOU GENERATE

Understand the system, then plan

§1
Purpose and scope
Who this manual is for and what it doesn't assume.

§2
The AI ecosystem: what is what
Five AI components, and why they shouldn't be confused.

§3
Create a new Game
Simple, Plus or Complex — choose the smallest mode you need.

§4
End-to-end workflow
The full 10-step route from idea to saved game.

§5
Prepare the game brief
Tell the AI what levels must achieve, not what each contains.

GENERATE & VALIDATE

Hand off to the AI, then check its work

§6
Generate the original JSON
The hand-off package between REBOOT and the AI.

§7
Fill the JSON with an external AI model
First pass, copy correctly, and six models compared.

§8
Import, inspect and improve the JSON
Green, red, and what a human still has to judge.

FINISH & PUBLISH

Media, review, and life after generation

§9
Add media and choose the presentation mode
Scroll Down, Sweep Sideways, or Carousel / Normal.

§10
Review, publish and save
Browser memory is not permanent storage.

§11
Edit the generated game
AI is a starting point, not the end of authorship.

§12
Optional player-facing AI features
Game Assistant / Oracle, AI Co-Player, Player Chat.

QUALITY & REFERENCE

Audit, fix, and look things up

§13
Audit, quality control and critical-thinking review
AI can help assess the game, not only generate it.

§14
Troubleshooting
Eight common failures and what to do about them.

§15
Recommended working practice
A repeatable process, before / during / after.

§16
Terminology and prototype decisions
Shared vocabulary and what's still evolving.

QUICK REFERENCE

Create a game in 10 moves, plus three rules to remember. Jump to it →

STAGE 01

Before You Generate

Understand the AI ecosystem, choose the right game type, and write a brief the model can actually use.

§1 Purpose and scope

A practical manual for the current Game prototype workflow.

The AI Game Assistant is designed to accelerate game creation without forcing the author to manually build every level, character, question, and information block. The platform prepares a structured JSON package containing the game requirements and the instructions needed by an external AI model. The AI completes that package, and the completed JSON is brought back in for inspection, media enrichment, and publication.

This manual separates the different AI components because they serve different purposes. In the working interface, several functions may still use the generic label "AI Assistant"; for clarity, the manual uses more precise functional names.

Who should use this manual

- Game authors and facilitators creating ERUDITA-style learning games.
- Project teams prototyping games before transferring them to the live platform.
- Editors reviewing generated levels, characters, questions, scoring, and media.

- Developers and testers validating the evolving AI-assisted workflow.

What this manual does not assume

- You do not need to write JSON manually.
- You do not need an API key for the Create Game workflow.
- You do not need to predetermine every level before generation; the AI should do part of that design work.

§2 The AI ecosystem: what is what

Three distinct AI functions should not be confused.

COMPONENT	PURPOSE	WHEN USED	STATUS
AI Game Builder (interface may say AI Assistant)	Creates the structured game package and validates the completed JSON.	Before a game exists.	Core workflow
Game Assistant / Oracle	A shared AI helper available to players across levels when activated.	During gameplay.	Optional / evolving
AI Co-Player	A conversational virtual character linked to the story and, potentially, to each human player.	During gameplay.	Advanced / prototyping
Player Chat Conversations	Control and audit view of conversations between human players and AI characters.	After or during play for monitoring.	Control function
Game Audit	Shows what the game contains and supports quality review: content, gaps, timing, distance, strengths and weaknesses.	Before testing and publication; after iteration.	Quality-control function

> **RECOMMENDED NAMING**
>
> Use "AI Game Builder" for the creation function and reserve "Game Assistant" for the player-facing helper. This avoids the current ambiguity created by using "assistant" for two different things.

§3 Create a new Game: Simple, Plus, Complex

The author begins by choosing the amount of structure the new game needs.

My Games // Erudita: The Price of Silence // ID: 130 **Create a New Game**

Figure 1. Current Create a new Game screen in the ERUDITA platform.

	GAME TYPE	BADGE
1	**Simple Game** Basic levels, one language, clean structure.	SIMPLE
2	**Plus Game** Characters, extra information blocks and two-language support.	PLUS
3	**Complex Game** AI characters, GPS levels, scoring and multi-language structure.	COMPLEX

Choose the smallest mode that contains the functionality you genuinely need. A larger mode does not automatically produce a better game; it creates a larger structure and therefore a larger AI generation task.

§4 End-to-end workflow

From game idea to a saved, editable game.

#	STEP	DETAIL
01	Choose game type	Simple, Plus or Complex.
02	Write the brief	Description, objectives, level-generation goals, character goals and languages.
03	Generate original JSON	Your Game packages the rules, fields and your requirements.
04	Send JSON to an AI model	Paste the complete JSON into a normal AI chat; initially add no extra instruction.
05	Copy the completed JSON	Use the model's copy function; do not manually select partial output.
06	Import and Inspect	Paste or drag the JSON into Game and run Inspect JSON.
07	Refine if needed	Ask the AI to improve weak sections while preserving the JSON structure.
08	Add and upload media	Add game and level images/videos; choose the presentation mode.
09	Review and Publish	Check content, questions, scores, media and validation messages.
10	Save and edit	The generated game appears in the dashboard/editor for normal editing.

> **DO NOT CONFUSE VALIDATION WITH QUALITY**
>
> A green JSON validation means the platform can read the structure. The author still has to judge whether the content is pedagogically strong, coherent, complete and appropriate for the game.

§5 Prepare the game brief

Tell the AI what the levels must achieve, not exactly what every level must contain.

5.1 The central design rule

The initial brief should describe the game's purpose and the design logic. Avoid writing the complete level sequence yourself inside the setup fields. If the author defines every level in advance, the AI is forced to reconcile too many local instructions with the global game rules, and the generation becomes more confusing and less useful.

> **GOOD INSTRUCTION**
>
> "The levels should progressively deepen the player's understanding of game. Questions should become more difficult over time. Generate at least 20 levels."

> **AVOID AT THE INITIAL STAGE**
>
> "Level 1 must do X, Level 2 must do Y, Level 3 must use this exact text…" unless the project genuinely requires a fixed pre-authored sequence.

5.2 Recommended input fields

INPUT	WHAT TO DESCRIBE
Game description	What the game is about, its context, audience and overall experience.
Objectives	What the player should learn, understand, practise or decide.
Level-generation goals	Desired progression, difficulty, minimum number of levels, types of tasks and how the sequence should evolve.
Character-generation goals	What kinds of characters or roles are needed and what they should contribute to the game.
Languages	English is the default; additional selected languages can be generated in parallel.

5.3 Why this works

- The author defines intent and quality criteria.
- The platform provides the technical schema and required fields.
- The AI model performs the generative work inside that schema.
- The author reviews, negotiates, and iterates rather than accepting the first answer blindly.

STAGE 02

Generate & Validate

Hand the JSON to an AI model, get it back, and check it before anything else happens.

§6 Generate the original JSON

The JSON is the hand-off package between ERUDITA and the external AI.

After the brief is completed, the platform generates an original JSON file. Think of this file as a self-contained work order for the AI model.

6.1 The original JSON contains two kinds of information

1. System specifications: the game type, required sections, fields, keys, level structure, language logic, scoring fields, and other rules the AI must respect.
2. Your design requirements: the description, learning objectives, level-generation goals, character goals, language choices, and other instructions entered for this specific game.

> **EVERYTHING IS ALREADY PACKAGED**
>
> The key idea is that the author should not have to explain the whole task again in the AI chat. The JSON already contains both the technical instructions and the game-specific brief.

6.2 Opening the JSON is optional

You can save and inspect a JSON file with a code editor such as Visual Studio Code, but opening it is not required for the normal workflow. Manual editing should be reserved for users who understand the JSON structure and know how to preserve valid syntax.

---

§7 Fill the JSON with an external AI model

The AI model is used as the content engine, while REBOOT PLATFORM controls the structure.

7.1 First pass: paste only the JSON

1. Open a new chat in the AI model you want to use.
2. Paste the complete original JSON into the dialogue box.
3. Do not add a second, competing description of the game on the first pass. The instructions are already inside the JSON.
4. Let the model complete the structure and return the full JSON.

> **WHY "JSON ONLY" MATTERS**
>
> Extra prose can make the model mix your new instruction with the instructions already embedded in the file. The cleaner first pass is: complete JSON in, completed JSON out.

7.2 Copy the result correctly

- Use the AI interface's Copy button for the complete code block or response.
- Do not manually highlight only the visible text if there is any risk of missing the first or last lines.
- Do not copy formatted page content that may include hidden styling or unrelated interface text.
- If you want to keep a reusable file, convert the copied plain text into a .json file with a text-to-file tool or code editor.

7.3 Prototype observations about AI models

The following comparison reflects one prototype testing session and should be treated as an operational note, not a permanent ranking. Free limits, context windows and model behaviour change frequently.

AI MODEL	OBSERVED STRENGTHS	OBSERVED WEAKNESSES	SESSION NOTE
Qwen (free)	Very complete output; strong adherence to the embedded instructions; generous output in testing.	Occasional malformed spacing/whitespace in JSON during the test.	Best first-pass result in that session.
Claude (paid)	Complete content and easy iterative refinement through conversation.	Desktop workflow can be slower because it may test, verify and re-check work.	Strong quality when available.
DeepSeek	Produced a reasonably complete result.	Less complete than the strongest outputs in the test.	Middle-range result.
Grok	Could complete the task.	Free capacity was reached quickly; output was shorter.	Useful but more constrained in that test.
Gemini	Handled long conversations in the free tier.	Tended to be brief/vague and produced less text.	Good endurance, less detail in that test.
ChatGPT free	Available through normal web chat.	During this specific test, a very large request was not completed satisfactorily.	Behaviour depends on plan/model and can change.

§8 Import, inspect and improve the JSON

Validation is the bridge between AI output and the REBOOT editor.

8.1 Import

Return to the REBOOT platform. Paste the completed JSON into the import area or drag the saved JSON file into the interface, then select Inspect JSON.

8.2 Understand the result

STATUS	MEANING
GREEN	The platform can read the structure and required information is present enough to continue.
RED	Something is missing, malformed or incompatible. Correct the JSON and inspect again.

8.3 Human review before continuing

- Game title and overall description.
- Number of levels and progression.
- Level descriptions and missions.
- Questions, answer options and scoring.
- Characters and information blocks, if the chosen game type uses them.
- Language versions and consistency between languages.
- Any field that looks suspiciously empty, generic, or repetitive.

8.4 Improve weak content without rebuilding the game

If the JSON is structurally valid but the content is too vague, return to the AI conversation and ask for a targeted improvement. The model should preserve the same JSON keys and return the complete updated JSON.

> **EXAMPLE REVISION INSTRUCTION**
>
> "Expand the descriptions for every level and make the difficulty progression more explicit. Preserve every JSON key and return only the complete JSON."

If the model becomes confused or repeatedly breaks the structure, start a new chat and repeat the process with the original JSON rather than continuing to repair a deteriorating conversation.

STAGE 03

Finish & Publish

§9 Add media and choose the presentation mode

Text generation is only the first half of the game-building process.

9.1 Where media can be added

- Game-level media: cover image, introduction video, or other assets representing the whole game.
- Level-level media: images, vertical videos, horizontal videos, YouTube/Vimeo content or other media associated with a specific level.

9.2 Media must be uploaded

Selecting a media file is not the same as uploading it. The asset must be uploaded to the server before it can be reliably used in the published game. The prototype allows the author to upload media progressively, which is useful on slow connections or when a game contains many assets.

> **PROTOTYPE PUBLICATION RULE**
>
> During the recorded test, the game required at least one uploaded media item before publication could proceed.

9.3 Working terminology for media navigation

WORKING LABEL	GESTURE / LAYOUT	PLAYER EXPERIENCE
Scroll Down	Vertical sequence	Media appears one below another in a feed-like format, similar to moving through short-form social media content.
Swipe Sideways / Sweep Sideways	Horizontal gesture	The player moves left/right between media items.
Carousel / Normal mode	Dot or slide navigation	Media is shown as a conventional carousel rather than as a continuous feed. The final interface label is still being standardised.

Each level can use the presentation mode that best fits its media. "Vertical" or "horizontal" should describe the media orientation, while the navigation label should describe how the player moves through the media.

---

§10 Review, publish and save

The current prototype keeps unpublished work in browser memory.

10.1 The browser-memory warning

> **DO NOT TREAT THE CREATION SCREEN AS PERMANENT STORAGE**
>
> During the generation and review process, the working game is held in the browser session. Do not assume you can leave it unfinished for a day and return safely. Keep your JSON file and publish/save the game once the structure is ready.

10.2 Before publishing

- Inspect JSON returns a valid result.
- Core text fields are complete and coherent.
- Questions and scoring are present where required.
- At least one required media asset has been uploaded.
- Presentation modes are set for the levels that use media.
- No prototype database field is missing a default value.

10.3 Publish and save

When the review is complete, continue to the publication step. Saving generates a new game in the dashboard/editor. From that point, the game is no longer only a temporary browser build: it can be opened as a normal game and edited through the standard editor.

> **EXAMPLE FROM THE WORKING SESSION**
>
> A test game named "Starship Earth / Cosmic Voyage" was generated, reviewed, given media, saved, and then appeared as a new game with its levels, text, questions and scores available for editing.

§11 Edit the generated game

AI generation is a starting point, not the end of authorship.

Once saved, the game can be edited in the normal REBOOT editor. This is where the author can make precise changes that are inefficient to negotiate through JSON generation.

Typical post-generation edits

- Rewrite or shorten individual texts.
- Correct a question, answer, or score.
- Add, replace, or remove media.
- Change the presentation mode of a specific level.
- Delete content that the AI overproduced.
- Add game-specific logic that requires human judgment.
- Activate or deactivate optional AI features.

The intended workflow is therefore hybrid: AI for breadth and first-pass construction; human editing for narrative quality, pedagogy, accuracy, and final experience design.

§12 Optional player-facing AI features

The platform is evolving from static generated content toward real-time interaction.

12.1 Game Assistant / Oracle

The Game Assistant is an optional player-facing AI helper that can be activated across levels. It is conceptually different from the AI Game Builder: the Builder helps the author create the game, while the Game Assistant helps the player inside the game.

- One shared assistant can be generated for the whole game.
- It can be enabled or disabled according to the game design.
- Its role, limits, and behavior must be defined so it supports rather than replaces player reasoning.

12.2 AI Co-Player / Personal Virtual Player

The proposed AI Co-Player goes further. The human player can talk in real time with a virtual character connected to the story. Instead of receiving only pre-recorded or pre-written information, the player can question the character and explore the narrative interactively.

CONFIGURATION	IMPLICATION
One independent AI avatar for all players	Simpler to configure; similar to an Oracle or institutional representative.
Character-specific AI Co-Player	Stronger narrative and emotional connection, but each playable character requires its own AI profile, behaviour and knowledge boundaries.

> **DESIGN IMPLICATION**
>
> If a game has five playable characters and each must behave differently in conversation, five distinct AI character profiles are needed. Character identity, knowledge, tone, evidence access and boundaries should therefore become part of the game design specification.

12.3 Player Chat Conversations

The Player Chat Conversations area is a control and audit function: it allows the team to review conversations that real players have had with AI characters. It is not the same as the authoring assistant.

STAGE 04 Quality & Reference

Audit the finished game, troubleshoot what breaks, and keep a shared vocabulary going forward.

§13 Audit, quality control and critical-thinking review

AI can help assess the game, not only generate it.

A major value of the emerging workflow is the ability to analyze a complete game after generation. The audit layer can support the author in identifying what works, what is missing, and where the player experience may fail.

Examples of useful audit questions

- Does the level sequence actually increase in complexity?
- Are some levels repetitive or too vague?
- Are all characters necessary and meaningfully differentiated?
- Do questions require reasoning or merely recall?
- Are scores balanced and consistent?
- If the game uses GPS: are locations too far apart for the planned game time?
- Are the distances between monuments realistic for the intended route?
- Is the total game duration compatible with the event or workshop?
- What are the strengths, weaknesses, risks and missing elements of the current design?
- Does the AI assistant support critical thinking, or does it give answers too easily?

> **HUMAN RESPONSIBILITY REMAINS CENTRAL**
>
> AI can surface inconsistencies and generate alternatives, but the final judgement about pedagogy, ethics, narrative credibility and player experience belongs to the game author and project team.

§14 Troubleshooting

Most failures are either incomplete copy/paste, malformed JSON, missing fields or unpublished media.

PROBLEM	LIKELY CAUSE	WHAT TO DO
Inspect JSON returns red / error	The AI changed the JSON structure, omitted lines, added invalid characters or returned an incomplete response.	Copy the complete output again. If needed, start a new AI chat with the original JSON and regenerate.
JSON looks correct but import fails	Extra whitespace or malformed keys/values may have been introduced by the model.	Use the platform validator. Regenerate if needed; the prototype is being hardened to normalise common whitespace errors.
Copied response is missing the last lines	Manual text selection did not capture the complete code block.	Use the AI interface Copy button instead of selecting text manually.
AI result is too short or vague	The model compressed the content or ignored detail requirements.	Ask for a targeted expansion while preserving every JSON key; or retry with a model that handles the full context better.
AI keeps making the JSON worse	The conversation context has become confused.	Start a new chat. Paste the original JSON again and perform a clean generation.
Cannot publish	Required media has not been uploaded, or a required database/default field is missing.	Upload at least one required media item. Inspect validation messages and correct missing defaults/fields.
Game disappears after leaving the page	The game was still only in browser memory and had not been saved/published.	Keep a local JSON copy and complete the save/publish process before ending the session.
Player-facing assistant stops working	Runtime AI service, free quota or model availability may have changed.	Treat runtime AI integration as a separate dependency from the PHP-based Create Game workflow and reconfigure the supported model/service.

§15 Recommended working practice

A repeatable process for authors and project teams.

Before generation

- Decide whether the game is Simple, Plus or Complex.
- Write a concise game description and clear learning objectives.
- Define the desired progression and minimum number of levels.
- Define the role of characters without over-specifying every scene.
- Decide which language versions are genuinely needed.

During AI generation

- Keep the original JSON unchanged as your recovery copy.
- Use a new AI chat for the first pass.
- Paste the JSON only; avoid duplicate instructions.
- Copy the complete returned JSON using the platform copy control.
- Iterate only on specific weaknesses once a valid complete version exists.

Before publication

- Validate structure in REBOOT.
- Read the generated content as a game designer, not as a technician.
- Check progression, questions, scoring and language consistency.
- Upload media and test the chosen navigation mode.
- Run an audit for timing, distance, repetition, gaps and critical-thinking quality.
- Save/publish before leaving the browser session.

After publication

- Use the REBOOT editor for precise changes.
- Prototype the game with real users before final deployment.
- Review AI conversations if player-facing AI is enabled.
- Update the manual when interface labels or AI services change.

> **A USEFUL OPERATING PRINCIPLE**
>
> Generate broadly with AI, validate structurally with the platform, edit precisely as a human, and test the final experience with players.

§16 Terminology and prototype decisions

A shared vocabulary will make the interface and future manual easier to understand.

TERM	MEANING IN THIS MANUAL	INTERFACE / STATUS NOTE
AI Game Builder	Authoring workflow that creates and validates the game JSON.	May currently appear as "AI Assistant".
Original JSON	REBOOT-generated template containing system rules plus the author's game requirements.	Keep a recovery copy.
Completed JSON	The same structure after an external AI model fills the content fields.	Imported back into REBOOT.
Inspect JSON	Platform validation step before moving to media/publication.	Green = readable/acceptable structure; red = correction needed.
Game Assistant / Oracle	One shared player-facing AI helper.	Optional; separate from authoring.
AI Co-Player	Interactive virtual story character that can chat with a human player.	Advanced prototype; may be character-specific.
Player Chat Conversations	Audit view of human-to-AI-character conversations.	Control function, not the authoring assistant.
Scroll Down	Vertical feed navigation for level media.	Working label.
Swipe / Sweep Sideways	Horizontal navigation between media items.	Working label.
Carousel / Normal	Conventional slide/dot navigation.	Final label still to be standardised.
Game Audit	Whole-game review of content, structure and experience.	Can support timing, distance, quality and critical-thinking analysis.

Items still evolving

- Final navigation label for the current media mode.
- Final visual color coding of AI-related menu items.
- Runtime model/service used by the player-facing Game Assistant.
- Final configuration workflow for character-specific AI Co-Players.
- Default values for newly introduced visual/media database fields.
- Exact persistence rules during long authoring sessions and any future autosave mechanism.

> **MANUAL MAINTENANCE NOTE**
>
> Because the platform is being prototyped actively, this document should be updated after each interface milestone. Screenshots, final labels and runtime AI dependencies are the sections most likely to change.

QR Quick Reference — Create a game in 10 moves

01 Choose game type

Simple, Plus or Complex.

02 Write the brief

Description, objectives, level-generation goals, character goals and languages.

03 Generate original JSON

REBOOT packages the rules, fields, and your requirements.

04 Send JSON to an AI model

Paste the complete JSON into a normal AI chat; initially add no extra instructions.

05 Copy the completed JSON

Use the model's copy function; do not manually select partial output.

06 Import and Inspect

Paste or drag the JSON into REBOOT and run Inspect JSON.

07 Refine if needed

Ask the AI to improve weak sections while preserving the JSON structure.

08 Add and upload media

Add game and level images/videos; choose the presentation mode.

09 Review and Publish

Check content, questions, scores, media, and validation messages.

10 Save and edit

The generated game appears in the dashboard/editor for normal editing.

> **THREE RULES TO REMEMBER**
>
> 1) The initial JSON already contains the instructions: paste it cleanly.
> 2) Green validation is not the same as good game design: review the content.
> 3) Until saved/published, keep your JSON because the working build may exist only in browser memory.

BACK TO TOP OF MANUAL

Version 0.1 | April 2026 — Living document: interface labels and AI integrations may evolve during prototyping.
Produced by PP5 SEALS — Task 3.2.2 · AI-driven Game Platform Prototype · REBOOT · Erasmus+ KA220-HED · Project No. 2025-1-CY01-KA220-HED-000351980

© REBOOT — AI Assistant Game Builder

↑ Back to cover