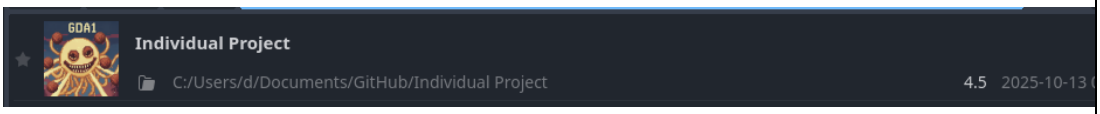
**SPRINT DOCUMENTATION — Week 2**

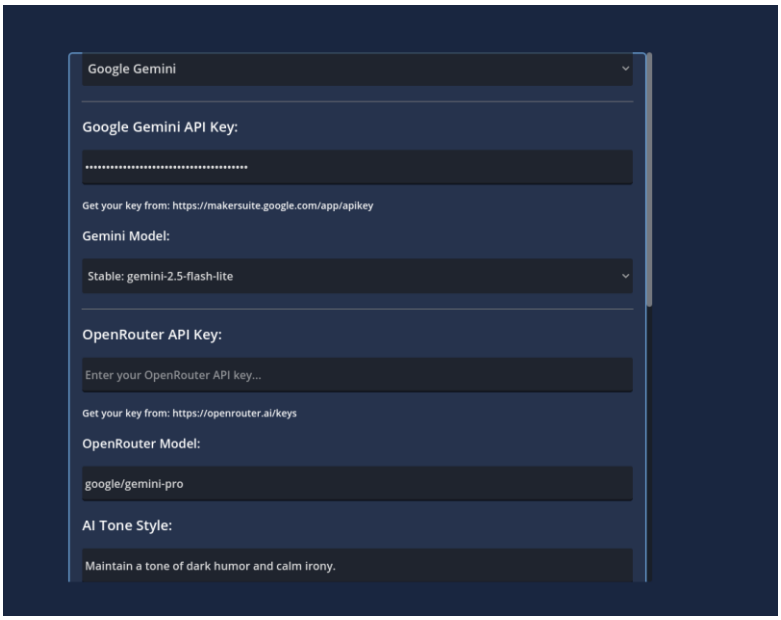
| **1) Summary data** | |
|---|---|
| Sprint start date | 06 Oct 2025 |
| Sprint end date | 12 Oct 2025 |

**2) User stories/task cards**

- Sprint Goal: My objective is to establish a Godot 4.5 shell featuring a narrative UI, set up the AI pipeline skeleton, and distribute the refined Week 3 proposal for approval.
- Stakeholders Engaged: I've been collaborating with my project supervisor, Dr Daniel Creed, by sharing proposal iterations and providing progress updates.
- Additional Note: I delivered the initial proposal draft on October 7 for feedback and submitted the revised version on October 9, aligning it with the deliverables for Week 3.
- *[Completed] I created the foundational Godot scenes (start menu, story view, settings, prayer overlay) to ensure my Week 3 gameplay experiments have a playable host for AI-driven content.*




- *[Completed] I implemented the AI manager with Gemini/OpenRouter support, offline mocking, and metrics, allowing my narrative prompts to execute without blocking on API keys.*




- *[Completed] I built the prayer, achievement, and journal subsystems to rehearse the Reality vs Positive Energy*


## 3) Requirements analysis

- *Functional (shall):*
  - *Godot 4.5 framework with interactive UI panels, stats bars, asset cards, and pause/prompt controls implements the baseline host for dynamic narrative content.*
  - *The AI pipeline now translates structured prompts into story beats, calling Gemini/OpenRouter via HTTPRequest with an offline MockAIGenerator fallback when keys are absent.*
  - *Multi-layer context is represented through GameState event logging, AI memory clamping, and story scene callbacks that refresh stats, choices, and asset metadata after each response.*
- *Functional (should):*
  - *Seeded domain systems (prayer disaster generator, achievement tracking, symbolic asset viewer) enforce the Void Entropy vs Positive Energy conflict that future prompts must respect.*
  - *Settings, language toggles, and font scaling prepare the UI for bilingual presentation requested in the project proposal.*
- *Non-functional:*
  - *API keys stay outside source control by collecting them through the AI settings menu and persisting to user:// config files; players can switch providers at runtime.*
  - *AI metrics (token use, latency history, memory budgets) surface within the settings UI to keep narrative costs observable and controllable.*
  - *Autosave, save slots, and notification hooks improve resilience against crashes while reinforcing the "depth over breadth" sprint objective.*
- *Domain requirements:*

- *Reality, Positive Energy, and Entropy scores now react to choices and prayers, feeding back into prompts and achievements to analyse thematic coherence.*
- *The notes/asset register and journal overlay persist emergent facts for later long-term memory summaries.*

**4) Design**

- *story_scene.gd orchestrates AI requests, loading overlays, choice generation, stat updates, and teammate interference hooks, allowing mock or live LLM replies to drive the session.*
- *ai_manager.gd wraps Gemini/OpenRouter HTTP calls, retries, timeouts, token counters, and memory trimming while defaulting to a deterministic mock when API keys are missing.*
- *ai_settings_menu.gd exposes provider selection, key inputs, memory sliders, tone prompts, and a metrics chart that polls AIManager each second and provides a one-click connection test.*
- *game_state.gd centralises stats, debuffs, event logs, autosave/slot management, and metadata storage so UI components and AI prompts stay in sync.*
- *prayer_system.gd, the journal, achievements, notifications, and audio managers convert domain lore into interactive overlays, tying narrative consequences to tangible stat swings.*

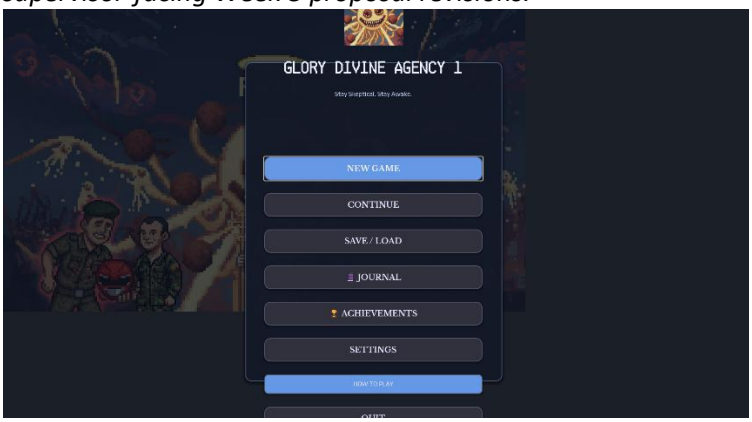
**5) Test plan and evidence of testing**

- *Verified Markdown parsing and BBCode rendering through the dedicated test_markdown_parser.GD harness to ensure AI outputs display correctly in the story view.*
- *Exercised the new UI flows (start → story → prayer → achievements) inside the Godot editor, using the AI settings "Test connection" button to confirm mock responses and loading/error states.*
- *Real API calls remain blocked until live keys are available; next sprint will replay the same tests with Gemini/OpenRouter credentials and capture metrics for latency/token charts.*

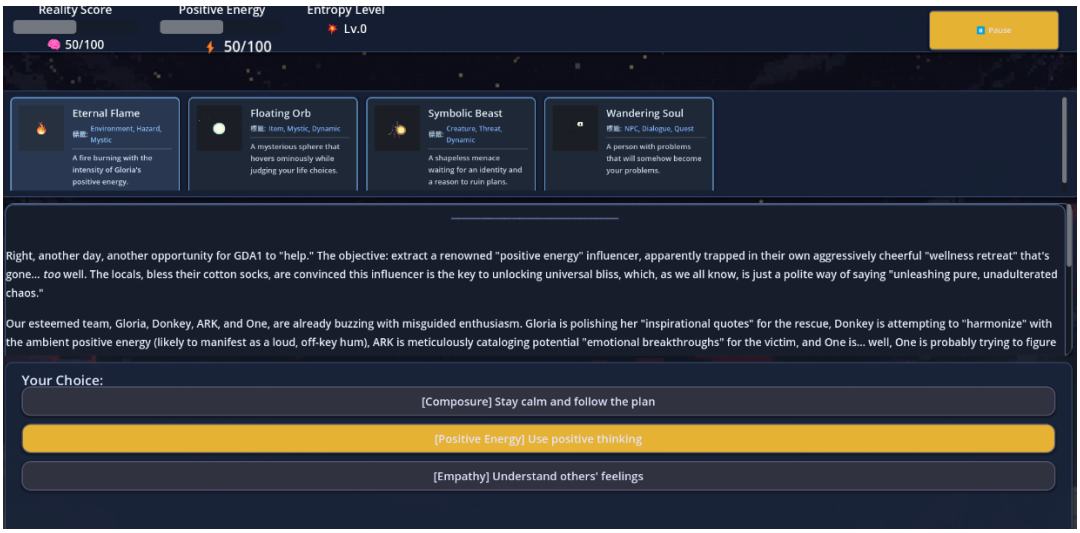
**6) Summary of the sprint**

- *Achieved: Bootstrapped a playable vertical slice (menus, story scene, prayer overlay, settings) backed by GameState, AIManager, and mock content, and finalised the*

*supervisor-facing Week 3 proposal revisions.*




- *Stakeholder feedback: Supervisor comments on the 07 Oct draft were resolved in the 09 Oct submission; no additional blockers were raised this week.*
- *Prototype status: The game boots to the start menu, loads the story scene, and can cycle choices, achievements, and prayers using the mock AI and live API testing rolled into Week 3.*




- *Risks monitored: API quota/cost (mitigated by offline mock and metrics), scope creep (kept to UI/AI depth), data loss (autosave/save slots added), asset mismatch (symbolic cards + UI polish).*
- *-Next sprint focus: Prioritise literature review and interim report drafting (Week 7), enable live API keys for Gemini/OpenRouter with robust error handling, and begin capturing latency/token metrics for report evidence.*