Fading Echo - Design Report Final Summary

Group 5: Adell Bosnjak, Mohammad Baqai, Muhammad Salim, Joshua Jung

Project Overview:

Fading Echo is a single-player horror survival game that is set in a procedurally generated WWII mineshaft. The player is a miner trapped in the cave, and to escape, they must collect 5 dynamite sticks and escape while being hunted by a blind, very sound-aware creature. The game's distinct feature is the real-time microphone integration. The players' actual real-world sounds are captured, then classified and fed into the creature's AI, making every keyboard click, spoken word, cough, etc is, a gameplay decision. It is a voxel aesthetic inspired by Minecraft that targets a broad audience, from horror fans and survival game enthusiasts to content creators, while a LLM-powered radio provides a dynamic narrative layer and hints during the game.

Scope and Stakeholders:

The scope covers the design and development of the single-player experience on a PC or laptop, including the procedural mine environment, real-time audio processing, adaptive creature AI, survival systems (health, stamina, hunger, etc.), resource management, and LLM radio integration. The primary stakeholders include end-user players (horror fans aged 12-40, survival fans, streamers), the LLM API provider, and potential outside investors. The project is scheduled to follow an agile development plan, with everything tracked in Jira and a two-year post-release support plan.

Requirements Summary:

The project is driven by eight formal use cases: Start Game, Navigate Mine, Crank Flashlight, Encounter Creature, Use Radio, Collect Dynamite, Manage Survival, and Escape the Mine. There are also non-functional requirements that span four categories. Performance requirements set a 30+ FPS floor, a microphone-to-creature latency of under 200 ms, an LLM radio response time of under 5 seconds, and procedural generation that takes less than 15 seconds. We also want at least 85% audio classification accuracy and 95% of sessions to be crash-free. We also want a fully playable offline core loop, fallback radio messages if the LLM API calls fail, and a graceful microphone-disconnect handling. Security requirements enforce local-only audio processing (No raw audio leaves the machine), API key isolation, and response sanitization. Usability requirements address intuitive navigation, accessibility, and in-game microphone diagnostics.

System Design:

The game's architecture uses a layered design with three tiers. The external layer (microphone, keyboard, LLM API, display/speakers), a subsystems layer, and a game core layer. Five gameplay subsystems are planned. Audio Processing implements a five-stage pipeline (Raw PCM, FFT + Windowing, Noise Floor, Classification, and AudioEvent), which will allow us to meet the 200 ms latency budget. Creature AI uses a behavior tree with five states (Idle, Patrolling, Investigating, Hunting, and Retreating), driven by an AlertModel that converts AudioEvents into a continuous alert level. Procedural Generation runs a six-phase pipeline (Seed, BSP Partition, Corridor Pass, Template Placement, Resource Scatter, and NavMesh Bake), guaranteeing at least 50 rooms, exactly 5 dynamite, and full seed reproducibility. Anthropic will be the primary LLM API provider with OpenAI. Survival + HUD tracks players' stats, manages inventory, and renders a diegetic overlay. The Game Core hosts the GameManager, a thread-safe EventBus (the sole coupling point between the subsystems), and a SceneController managing scene transitions.

Cross-cutting concerns are handled by a SaveSystem service (SettingsManager, StatisticsRecorder, Logger, and IntegrityChecker) that any subsystem may use but which does not own a meaningful gameplay state. Seven design patterns are applied: Strategy (SoundClassifier), State Diagram (CreatureAI), Observer (EventBus), Singleton (GameManager, Logger), and Factory Method (ProceduralGen). The repository layout mirrors the subsystem decomposition. The repository layout mirrors the subsystem decomposition (audio, creature, world, radio, UI, and core).

Key Design Decisions:

Privacy is a big deal when dealing with audio. The audio processing subsystem exposes no public method that returns raw audio, and there is no field for player-identifiable data, giving us two privacy enforcement points. Graceful fallback is a big aspect and is integrated in two places. For the radio device, the main provider will be Anthropic, with a fallback to OpenAI; if both fail, a pre-written fallback will be used. Then, if the mic disconnects, the creature will fall back into a passive state. The EventBus is a major component that enables communication between all systems.

Technology Stack:

The implementation uses Godot 4. The audio processing runs through Godot's native AudioServer with a custom FFT. The LLM will use Antropic Claude as the primary and OpenAI as the fallback. Distribution will be on Steam, with console implementation saved for the future.