LIVING WORLDS: IMMERSIVE SMART GAMING

WHERE YOUR GAME COMES TO LIFE

TEAM NAME: PRESS X TO INTERACT

TEAM MEMBERS:

RAUNAK KUKREJA
VIDIT KAUSHIK
VIGYANT NAYAK
LOKADITYA JANGID

X

PROBLEM STATEMENT

Nowadays in every fancy high budget game out there the NPCs often feel like static content; lack of originality and full of redundancy. Every cut scene is the same and every user experience is the same.

We can tackle this with AI :
Smart NPCS can remember, gossip and evolve, making single-player worlds feel alive and unique to each player

- AI can provide dynamic, personalized NPCs, not just scripted lines
- Opportunity: emergent social systems that change gameplay

Game Developers : Implement and maintain the LLM-powered NPC systems within the game engine.

AI/ML Engineers : Optimize model performance, quantization, and real-time inference for dynamic NPC behavior.

Players : Engage with immersive, personalized gameplay shaped by evolving, intelligent NPCs.

STAKEHOLDERS

TECH STACK

FRONTEND

- Tech: React + Vite, SVG overlays, tiled map JSON
- Responsibilities:
  - Render map, entities, UI
  - Movement & camera logic
  - Interaction handling (press X)
  - Visualize gossip graph & minimap
  - Send requests to backend (interact, gossip APIs)

BACKEND

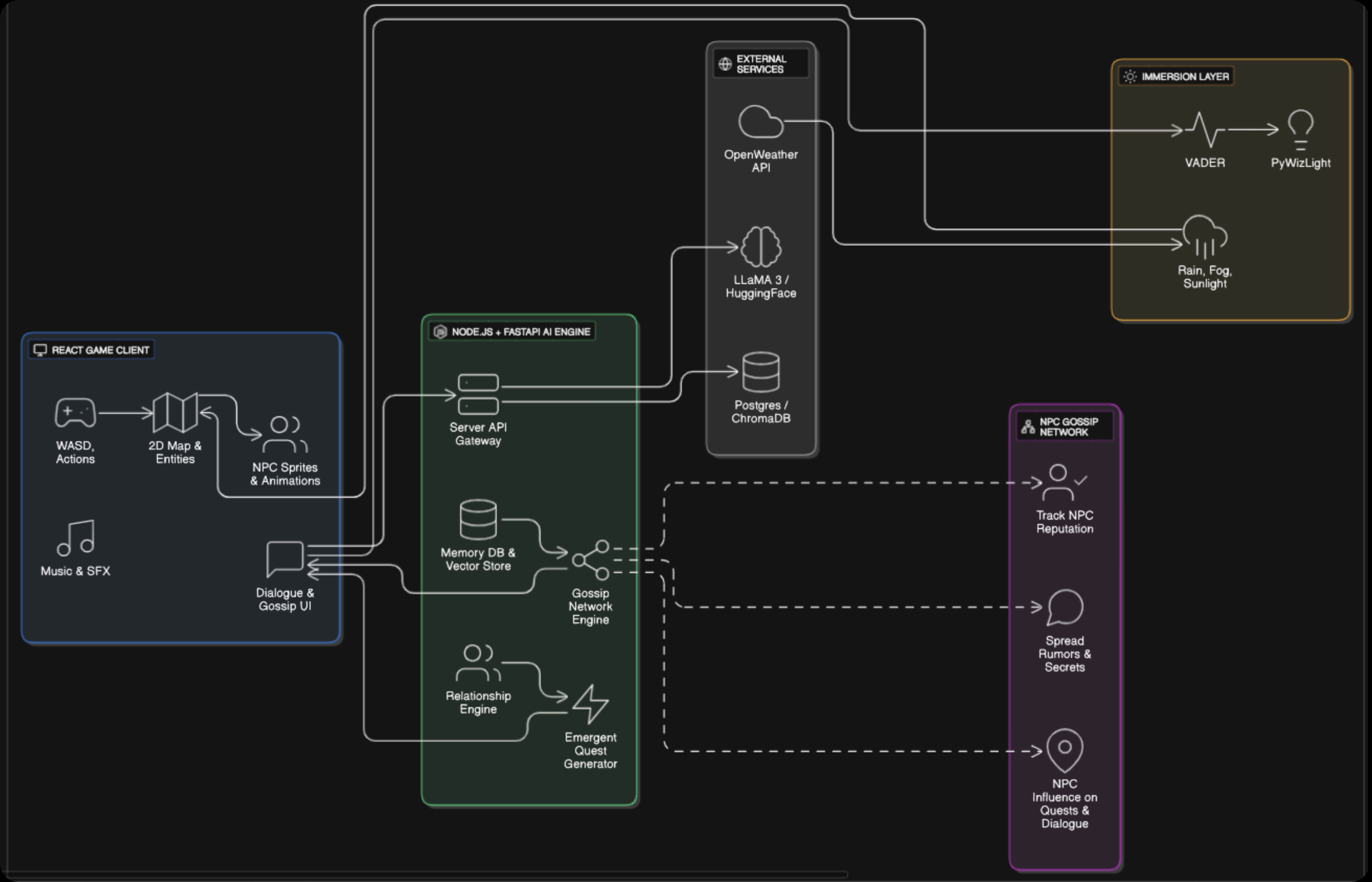
- Tech: Node.js / Express (controllers + services), REST API
- Responsibilities:
  - Authenticate/forward model queries (LLMGateway)
  - Assemble prompts (PromptAssembler)
  - Manage game state & gossip (GameStateManager)
  - Memory store & gossip persistence (MemoryStore, gossip.log.json)

MODEL SERVER

- Tech: Local Llama-based server (llama_cpp / gguf) served via FastAPI/uvicorn and LLAMA CPP
- LLM : Meta-Llama-3-8B-Instruct (4 bit quantized)
- Responsibilities:
  - Expose a small HTTP inference endpoint
  - Stream or return single-sentence dialogues
  - Local API key protection, GPU/CPU controls

SMART DEVICES INTEGRATION

- Tech: Smart Bulb(Hardware) , PyWizlight
- Responsibilities:
  - Render map, entities, UI
  - Movement & camera logic
  - Interaction handling (press X)
  - Visualize gossip graph & minimap
  - Send requests to backend (interact, gossip APIs)



ARCHITECTURE

GOSSIP NETWORK

The purple color shows two interacting NPCs

The core of our innovation is the Gossip Network, a system that makes the world feel truly alive and interconnected. When ever two NPCs cross path , they exchange information about their last few interactions with the user .

Trigger mechanics:

- NPC proximity crossing / meeting detection
- Repeated crossing — counts as “interaction”
- Random scheduled meetings at free points

Flow:

- Two NPCs meet → backend logs meeting → if one is gossiper → create gossip item → gossip propagation algorithm spreads to connected NPCs
- The gossips are stores in separate .json files for each character which is passed to the LLM as context before each conversation call

Guard NPC and Old man NPC gossiping in proximity.
Total gossip: 6; active shares: 2

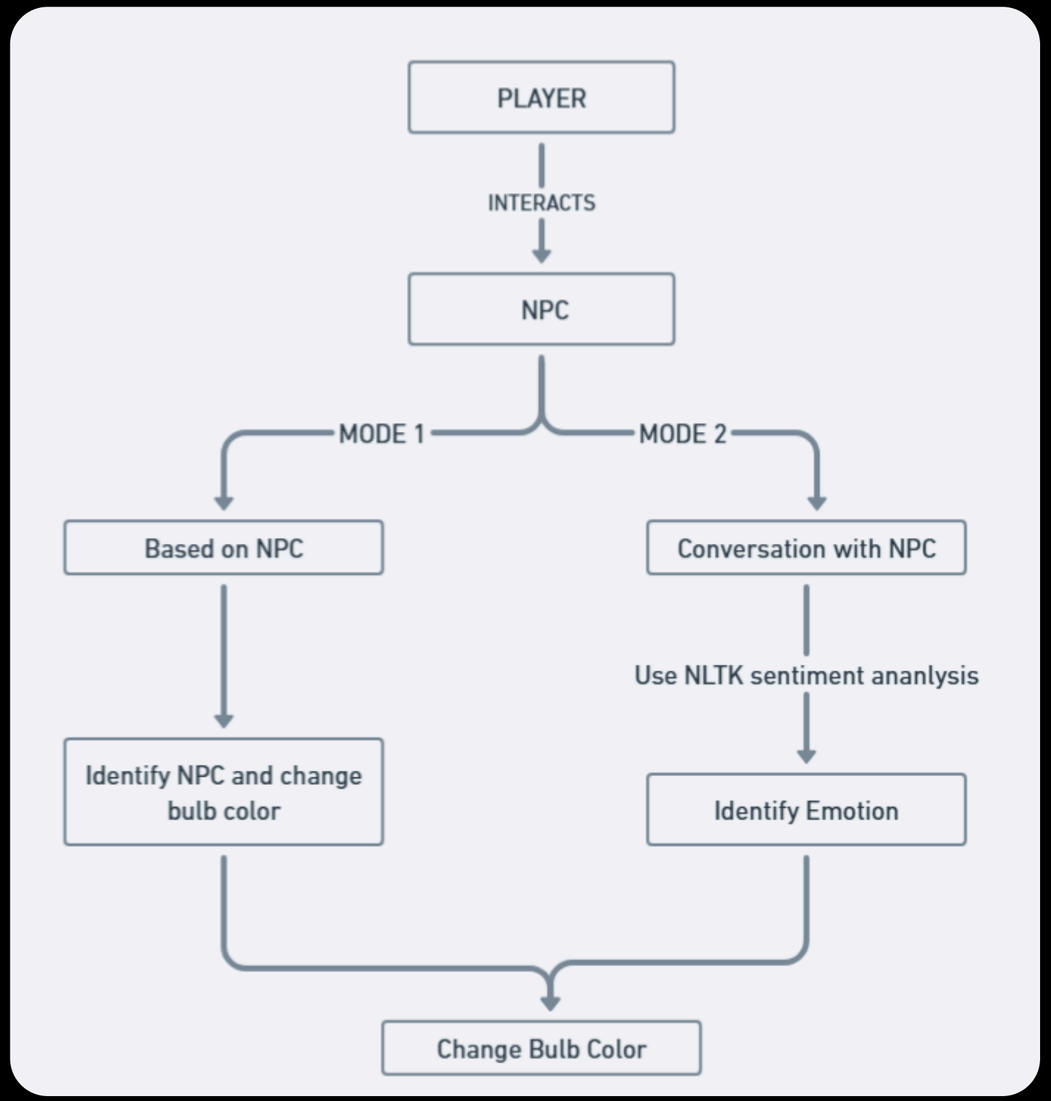
SMART DEVICES INTEGRATION

This system enhances player immersion by synchronizing ambient lighting with in-game emotional dynamics. When a player engages in dialogue with a non-player character (NPC), the system captures the conversation text and processes it using NLTK's VADER sentiment analysis tool.

Each detected emotion is mapped to a predefined RGB color profile. For example:

- Happy → Warm yellow or orange
- Sad → Cool blue
- Angry → Intense red

Using the PyWizLight API, the system communicates with WiZ smart bulbs over the local network. It sends real-time commands to update bulb states adjusting hue, brightness, and saturation to reflect the emotional tone of the interaction. This creates a dynamic feedback loop where the player's physical environment mirrors the emotional atmosphere of the game.



REAL TIME WEATHER INTEGRATION

Dynamically changes the game's visual atmosphere based on real-world weather conditions using live weather data.

Visual Effects:

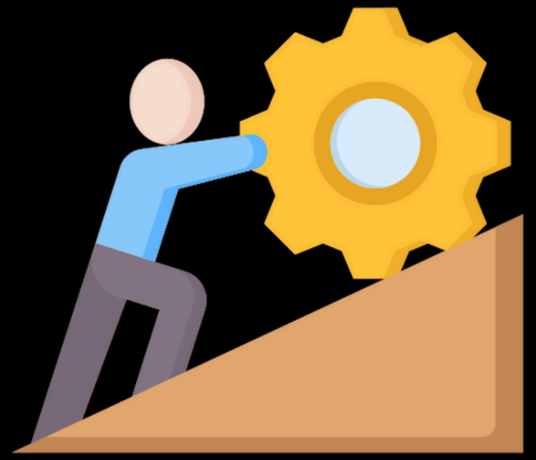
- **Hot Weather** (≥25°C): Bright, sunny, yellowish tints
- **Cold Weather** (<25°C): Cool, blue, subdued lighting
- **Rain**: Animated falling raindrops with atmospheric effects
- **Snow**: Floating snowflakes with winter ambiance
- **Fog**: Reduced visibility with drifting particles

This results in an immerse gaming experience, that adapts to real-world

Snow

Rain

CHALLENGES FACED



Challenge 1 The Actor Problem — AI Coherence & Consistency

- The Issue: Unlike scripted NPCs, ours are powered by LLMs. Over long interactions, they can forget context, contradict themselves, or break character.
- Our Fix: Our backend acts as a "Director," feeding the LLM structured prompts with personality, memory, goals, and quest state keeping every NPC on-script and in-role.

Challenge 2: The Unpredictability Problem — Hallucinations & Misuse

- The Issue: LLMs can hallucinate false info or break tone and in open dialogue systems, they're vulnerable to misuse, including prompt injection.
- Our Fix: We've implemented strict backend rules and sanitization layers that validate AI responses against game logic, character memory, and safety filters. This keeps interactions creative, consistent, and secure.

Challenge 3: The Butterfly Effect — Scaling Dynamic State

- The Issue: Every player action and NPC interaction reshapes the world creating a ripple effect across gossip, relationships, and factions. Tracking this evolving state becomes exponentially harder as the game grows.
- Our Fix: We use a robust database and well-structured JSON files to manage state efficiently. Smart data design ensures performance and cohernce

Challenge 4: The Real-Time Problem (Performance & Latency)

- The Issue: Even local LLMs introduce response delays. In a game, even a few seconds of lag can break immersion and make NPCs feel sluggish or disconnected.
- Our Fix: We use a lightweight Llama-based server (FastAPI + llama.cpp) and keep the model warm to avoid cold-start latency. Combined with an optimized backend, this ensures fast, fluid, in-character replies that keep players engaged.

FUTURE SCOPES

A Deeper Social Fabric: Advanced Relationship Engine

- **Enhancement:** weighted, multi-edge relations (NPC↔NPC, Player↔NPC, Faction↔*).
- **Impact:** nuanced trust, influence & debt; actions ripple with subtle long-term consequences.

Unfolding Narratives : Dynamic & Emergent Quests

- **Enhancement:** quests chain-trigger from gossip + overlapping NPC interests.
- **Impact:** multi-stage, AI-orchestrated storylines that evolve with player choices.

Infinite Worlds : Procedural Map & Story Generation

- **Enhancement:** per-player map + mission generation from player preferences (stealth/genre/etc.).
- **Impact:** truly unique, replayable adventures — maps, NPCs and objectives created on demand.

Scalable Intelligence : DB-backed Memory & Embeddings

- **Enhancement:** persistent memory + vector search for fast context retrieval.
- **Impact:** NPCs remember hundreds of events coherently; conversations stay consistent at scale.