CS 5001 – Senior Design I – Assignment #4
Adaptabrawl – Design Diagrams

~ Presented By Kartavya Singh (M14537829) (Singhk6)

Course: CS5001 — Senior Design (Fall 2025 → Spring 2026)
Team Name: Adaptabrawl
Repository: https://github.com/Kartavya904/Adaptabrawl-Senior-Design
Teammates: Saarthak Sinha, Kanav Shetty, Yash Ballabh

User Stories:

- **As an online competitor**, I want **reliable 1v1 lobbies** with predictable latency and easy **rematch/reconnect**, so that **matches feel fair** and I can **keep playing without friction**.
- **As a player on mid-spec hardware**, I want **scalable VFX/performance presets**, so that I can **maintain frame rate**.
- **As a character explorer**, I want fighters with **clearly described playstyles** (offense/defense/evasion) and **difficulty tags**, so that I can **quickly pick a style that fits me**.
- **As a local 1v1 player**, I want **offline versus**, so that we can **play without internet** or setup delays.
- **As a character developer**, I want a **robust framework** with documented character templates, so that I can **add new/adjust fighters quickly**.
- **As a friend-match player**, I want shareable **room-code invites instead of account linking**, so that we can **play immediately, without an account** or friends list.

Design Diagrams:

Diagram Conventions:

- **Boxes** denote systems or subsystems (scope depends on the level).
- **The large outer box** is the game's system boundary.
- **Arrows/lines** show data or control flow; double-headed lines indicate sync/pub-sub.
- **[Brackets]** label external actors or sources/sinks (e.g., devices, network, OS).
- Each diagram emphasizes **inputs and outputs** first, then how they're handled internally.

Design Diagram Level 1: This diagram presents the game as a single black-box system with only its external contract exposed. On the input side it receives Player Controls, Menu/Settings choices, Match Context/Conditions, Network In (for online play), and the System Clock. On the output side it produces Visual Output (scene + HUD + menus), Audio Output (BGM/SFX/VO), Haptics, Network Out (this client's inputs/state), and optional Data Artifacts (telemetry, replays, logs). External actors—Local Player, Remote Player/Relay, and the Platform/OS—sit outside the boundary to make it clear what the system consumes and emits. No mechanics, rendering details, or netcode internals appear here by design; Design Diagram Level 1 only defines the I/O surface.

[Picture]

Design Diagram Level 2: This diagram expands the black box into the major subsystems and the primary paths between them. The input rail fans into five entry subsystems: the Input Layer (device adapters and action mapping), Settings & Accessibility (video/audio/UI/control profiles), Match Config (stage and disclosed conditions), Networking (session I/O), and Clock (ticks/timebase). At the core, the Game State / Rules Engine orchestrates gameplay, exchanging state with Physics/Collision and the Net Sync/Authority module, and driving UI

& HUD for menus and combat overlays. An Event Bus carries game events (e.g., InputAccepted, HitLanded, StatusApplied, KO) to decouple producers and consumers. Outputs are realized through Rendering (2D/VFX) and Audio (BGM/SFX/VO), with optional Telemetry/Replays/Logs persisted to Storage. Design Diagram Level 2 therefore shows how inputs are routed, how subsystems coordinate, and where observable outputs leave the system.

[Picture]

Design Diagram Level 3: This diagram drills down to concrete components and end-to-end data flows. Raw device signals enter the Input Layer (XInput/KB/M adapters, remap/buffering/anti-ghost rules) and produce actions consumed via the Event Bus. The Rules Engine—implemented as a state machine with hit/hurt resolution, status effects, and round flow—pulls content (fighters, moves, hitboxes, stages, status FX, frame data) from a Content DB. Deterministic Timing runs a fixed-timestep frame scheduler (ticks) for physics and rules, while Rendering interpolates frames for smooth visuals and overlays HUD information; Audio schedules BGM/SFX/VO off game events. The Networking pipeline is shown from Transport (UDP/TCP/Relay) through Session Manager to Sync/Authority and Prediction/Correction, supporting join-by-code, ready checks, rematch, reconnect, and a netgraph for RTT/jitter/loss. Persistence & Observability includes opt-in telemetry, a circular-buffer replay system, and crash logs; a parallel Creator Pipeline demonstrates how a system user (character developer) uses templates, lints/validators, and a test arena to safely package/export new fighters. Design Diagram Level 3 explains how fairness, readability, and content authoring are achieved in practice.

[Picture]