Zoggy – Gaming Platform Case Study

Multi-Game Solana Casino with Non-Custodial Smart Vault

Prepared By – SeQuere Technologies

Client – Confidential

1. Project Overview

Zoggy is a Solana-based on-chain casino Multi Game platform that combines fast, non-custodial crypto gaming with a clean, wallet-first user experience.

The app is organised into two main verticals:

- Slots – a flagship slot like "Zoggy's Bar" with an always-on betting panel and live wins feed.
- Classics – a suite of six house games (dice, coinflip, crash-style multiplier game, plinko, mines, and one additional classic) presented under the "Classics" menu.

Zoggy is positioned around a few clear promises:

- Provably fair in-house games
- Non-custodial smart vault
- Wins instantly, settled on-chain
- Accountless – no registration, wallet-only login

The platform is built around Solana programs that manage user PDAs, admin or house PDAs, a referral system, and on-chain settlement of all bets and payouts. A Node.js backend handles real-time feeds (Live Wins, Big Wins, Jackpot Wins) and aggregates statistics such as win or loss history and referral performance, while the frontend uses Next.js.

2. Goals & Objectives

2.1 Player-Facing Goals

Player-facing goals include:

- Make on-chain gambling feel instant and simple. Zoggy uses a one-click wallet connect flow with no email, password, or traditional account creation. Players connect their wallet, choose a game, place a bet, and see the result in seconds.
- Guarantee transparency and provable fairness. All bets are executed through on-chain instructions. Game logic is deterministic and verifiable, and outcomes and payouts are visible on the blockchain.
- Support multiple games without fragmenting the experience. Slots and six Classic games share the same wallet connection, balance model, and live stream of results. Players can switch games without reconnecting or losing their state.

- Reward loyalty and referrals. A referral system tracks player volume and house edge contributions and can power rakeback, cashback, VIP tiers, and other incentive programs.

2.2 Platform & Business Goals

Platform and business goals include:

- On-chain treasury and fee capture. Admin or house PDAs act as smart vaults for fee capture, jackpot funding, and liquidity, removing the need for manual off-chain ledgers and spreadsheets.
- Non-custodial, lower operational risk. User funds remain in their own wallets or in program-owned PDAs governed by code, reducing the legal and operational complexity of fully custodial balances.
- Composable foundation for future games. The architecture is designed so new games can plug into the same treasury, referral, and statistics systems. Game-specific logic is isolated while shared infrastructure is reused.

3. User Personas & Journeys

3.1 Players

Player goals include:

- Connect with Phantom, Solflare, or another Solana wallet.
- Choose between Slots and Classics.
- Place bets quickly with clear, simple controls.
- See results and payouts almost instantly, backed by on-chain settlement.
- Track performance across sessions, such as total wins or losses, biggest wins, and favourite games.

A typical player journey is:

- Visit the main URL and see the key claims: provably fair, non-custodial smart vault, wins instantly, settled on-chain, accountless, no registration.
- Click “Connect” to link their Solana wallet.
- Choose either Slots (for example, Zoggy’s Bar) or the Classics menu with six games.
- Use the bet panel to set the stake and, where relevant, extra parameters such as risk level for plinko or target for dice.
- Confirm the bet; a transaction is sent to the Solana program.

- The interface animates the game (spin, coin flip, crash bar, falling ball, mine reveal) and displays the final payout once the transaction is confirmed.
- Results appear in the Live Wins, Big Wins, and Jackpot Wins streams and contribute to aggregated statistics.

3.2 Affiliates & Referrers

Affiliate and referrer goals include:

- Invite players through referral links or codes.
- Earn a share of house edge or fee revenue from referred activity.
- Monitor cumulative betting volume, net results, and rewards.

A typical affiliate journey is:

- Obtain or generate a referral code.
- Share the link with their audience.
- As referred players wager, the system associates their PDAs with the referrer and allocates a portion of house edge to that referrer.
- Use the affiliate dashboard to see per-player volume, total fees generated, and payout history.

3.3 Admins / Operators

Admin and operator goals include:

- Monitor key performance indicators such as turnover, return to player, house edge, and per-game volume.
- Configure game settings, including minimum and maximum bets, return-to-player distributions, and jackpot rules.
- Detect suspicious patterns such as bonus abuse, collusion, or automated play.
- Add new games without refactoring core infrastructure.

A typical admin or operator journey is:

- Access the admin interface, protected by role-based authentication.
- Review dashboards summarising bets, wins, losses, vault balances, and referral costs.
- Adjust configuration PDAs for each game, including house edge, limits, volatility, and jackpots.
- Trigger maintenance operations such as jackpot reseeds or vault top-ups through backend endpoints.

4. System Architecture

4.1 Solana On-Chain Layer
The on-chain layer is the core of Zoggy's fairness and treasury model. It is built around one or more Solana programs with several key PDAs:

User PDA

Tracks per-player statistics such as total wagered, total won, and net result. It can also hold locked vault funds if the design uses user sub-accounts, and includes flags for referral mapping and promotional eligibility.

Admin or House PDA (Smart Vault)

Holds the house treasury used to pay out wins and jackpots. It receives the house edge and platform fees from each bet and stores global configuration such as default house edge, list of enabled games, and feature toggles.

Game PDAs

Each game, including Slots, Dice, Coinflip, Crash, Plinko, and Mines, has configuration PDAs describing pay tables, multipliers, risk profiles, and jackpot contribution rates. Some games, such as crash or jackpots, maintain additional state such as the current jackpot pool.

Referral PDAs

Map a player's User PDA to their referrer and track cumulative referral revenue generated and pending payouts.

For randomness and game resolution, each game instruction derives random values from chain data and any additional randomness source, then feeds these into game-specific mathematics. The mapping from random input to outcome and final multiplier is deterministic and encoded in the program, so results can be re-verified by replaying the same logic off-chain against transaction logs.

4.2 Backend (Node.js)

The Node.js backend acts as the orchestrator off-chain:

- WebSocket or Socket.IO services stream Live Wins, Big Wins, and Jackpot Wins to connected clients and broadcast per-game and global statistics snapshots.
- A Solana event listener subscribes to the program's logs and account changes, decodes bet settlement events, and writes them into PostgreSQL.
- Analytics and reporting components aggregate per-game and per-player KPIs and feed admin and affiliate dashboards with data such as volume, return to player, net profit, and referral costs.
- Configuration and admin APIs provide secure endpoints to adjust configuration PDAs and to trigger operations such as jackpot reseeds, manual adjustments, or emergency switches.

4.3 Frontend (Next.js)

The frontend is built with Next.js:

- A wallet-first layout presents a prominent "Connect Wallet" call to action and integrates with Solana Wallet Adapter for Phantom, Solflare, and other wallets.
- A global layout provides tabs or sections for Slots and Classics, along with a shared header and consistent betting panel per game.
- Each game has its own React component that collects user parameters, triggers the Solana transaction, and animates the outcome based on the result returned by the program.
- A real-time layer subscribes to the backend's websocket to receive Live Wins and statistics and updates tables, tickers, and banners without page reloads.

5. Core Features & Game Logic

5.1 Slots ("Zoggy's Bar")

Gameplay:

- The player chooses a stake and, depending on the design, options such as number of lines or volatility level.
- The reels spin visually on the frontend while the final combination is determined by the on-chain program.

On-chain logic:

- The program generates random values representing reel stops and uses these to determine which symbols appear on each reel.
- It compares the resulting combination against pay tables stored in the Slot Game PDA.
- If the combination matches a winning pattern, it calculates the multiplier and payout and updates the user and house PDAs.
- The house edge is encoded in the pay table so that the expected long-term return to the player is slightly below one.

5.2 Dice

Gameplay:

- The player selects a target condition such as “roll under X” or “roll over X” and sets a stake.
- The closer the target is to the safe range, the lower the payout; the more extreme the target, the higher the potential multiplier.

On-chain logic:

- The program generates a random integer within a fixed range.
- It checks the chosen condition against that random roll.
- If the condition is satisfied, the player wins a payout based on the inverse probability of winning, adjusted for house edge.
- If the condition fails, the stake is transferred to the House PDA.

5.3 Coinflip

Gameplay:

- The player chooses heads or tails and sets a stake.
- The game is intentionally simple, providing a classic 50–50 style experience.

On-chain logic:

- The program generates a random bit and maps it to heads or tails.
- If the outcome matches the player’s choice, the player receives a payout approximately equal to double the stake minus house edge.
- If not, the stake goes to the house.

5.4 Crash (Multiplier Game)

Gameplay:

- A line or rocket multiplier increases over time from just above one upwards.
- A random crash point is determined at bet time.
- The player chooses an auto cash-out multiplier or relies on the default strategy.
- If the game crashes before the player's target, the stake is lost; if not, the player wins according to the chosen multiplier.

On-chain logic:

- The program maps a random value to a crash multiplier using a distribution that favours low multipliers and makes very high multipliers rare.
- The player's cash-out target is compared to the crash point.
- If the target is lower than the crash point, the player wins and the payout is calculated as stake multiplied by target, adjusted for house edge.
- If the crash happens before the target, the player loses the entire stake.

5.5 Plinko

Gameplay:

- The player chooses a risk profile such as low, medium, or high.
- A ball drops from the top of a peg board and bounces left or right until it lands in one of the slots at the bottom, each with its own multiplier.

On-chain logic:

- The program samples a path of left or right steps through the peg board, determining a final bin index at the bottom.
- Each bin's multiplier is preconfigured in the Plinko configuration PDA according to the chosen risk curve.
- The player's payout is stake multiplied by the bin's multiplier if it is greater than zero; otherwise, the stake is lost.

5.6 Mines

Gameplay:

- The player sees a grid of tiles with a fixed number of hidden mines.
- They click tiles to reveal either safe cells or mines.
- Each safe cell revealed increases the potential cash-out multiplier.
- Hitting a mine ends the round and loses the entire stake, while cashing out early locks in the current multiplier.

On-chain logic:

- When the round starts, the program generates a random layout of mines and stores it in temporary game state.
- Each time the player reveals a tile, the program checks whether it is a mine.
- If the tile is safe, it updates the current multiplier based on how many safe tiles have been found relative to the total number of safe tiles.
- If the tile is a mine, the round ends and the entire stake is transferred to the House PDA.
- When the player chooses to cash out, the program pays out stake multiplied by the current multiplier and ends the round.

5.7 Live Wins, Big Wins, and Jackpot Wins

Across all games, when a bet is settled the program emits an event with details such as player address or User PDA, game identifier, bet amount, payout, multiplier, and outcome type. The Node backend listens for these events, stores them, and broadcasts them over websockets. The frontend uses this data to populate Live Wins, Big Wins, and Jackpot Wins streams and to keep dashboards updated in real time.

6. Technical Challenges & Solutions

6.1 Real-Time Experience vs On-Chain Latency

Challenge:

On-chain confirmations introduce delay compared to a purely off-chain casino.

Solution:

- The frontend uses optimistic updates and game animations while the transaction is pending.
- Transaction status is confirmed using Solana logs as soon as the program settles the bet.
- Websocket listeners ensure that the moment a bet is confirmed, the user interface is updated with the final result and payout.

6.2 Provable Fairness & Transparency

Challenge:

Players must trust that the house is not manipulating outcomes.

Solution:

- All critical game logic and payout rules are implemented inside Solana programs.
- Each bet can be re-simulated off-chain using the same game configuration and randomness source to prove the outcome.
- Users can inspect their transaction history and PDAs to verify that balances and outcomes match the on-chain state.

6.3 Secure Fee Capture & Treasury Management

Challenge:

House edge, jackpots, and referral earnings require precise handling of funds with no leakage.

Solution:

- House, jackpot, and referral balances are implemented as PDAs with strict access controls.
- All transfers of tokens or SOL happen through well-defined program instructions.
- User funds, house funds, and reserved pools are clearly separated, and the program enforces all constraints around movement of funds.

6.4 Scaling Across Games

Challenge:

Adding a new game should not require rebuilding all core systems.

Solution:

- The bet model is generic and includes fields such as game identifier, parameters, stake, and random seed.
- Each game has its own configuration PDA and logic handler but uses shared vaults, referral systems, and analytics.
- The frontend consumes a consistent result structure for all games, changing only the presentation and animations per game type.

7. Outcomes

Zoggy now functions as a fully on-chain, multi-game Solana casino with six Classic games and a Slots vertical, each with clearly defined, on-chain game mathematics. A non-custodial smart vault model keeps balances auditable and under program control. Real-time gameplay is powered by Solana logs and websocket streams, while a referral and analytics stack supports affiliates, VIP structures, and promotional campaigns. The architecture allows new game types to be added with minimal changes to treasury, referral, or statistics layers.

8. Tech Stack

Blockchain and on-chain: Solana, custom programs (Anchor-style architecture), PDAs for users, house, games, and referrals; SPL token support; and Solana Web3.js for client interactions.

Frontend: Next.js (React with JavaScript or TypeScript), Solana Wallet Adapter integration, and a websocket client for live wins and statistics.

Backend: Node.js with an HTTP framework such as Express, websocket or Socket.IO server, Solana RPC and websocket listeners, and admin and affiliate APIs.

Data: PostgreSQL for history, game results, user statistics, and referrals, with optional Redis for hot leaderboards, caching, and rate limiting.

DevOps: AWS for frontend hosting, a container or VPS platform for backend services, environment-based configuration (RPC URLs, database credentials, program identifiers), and monitoring and logging around error rates and RPC health.

9. Conclusion

Zoggy shows how serious game mathematics, on-chain fairness, and a modern user experience can be combined into a crypto casino that avoids the usual black-box feel of traditional platforms. Each game, from simple coinflip to multi-step mines and high-volatility crash, is fully defined in program code with explicit probability distributions and payout curves. Players get instant-feeling gameplay, real-time win streams, and outcomes that can be independently verified on-chain.

For operators, the separation between game logic, treasury PDAs, and the analytics backend means the platform is ready to evolve. New games, new asset types, and new referral schemes can be layered on top without redesigning the core. The combination of Next.js and Node.js backend, and Solana programs creates a clean separation of concerns: the blockchain guarantees fairness and settlement, the backend orchestrates real-time data, and the frontend delivers a wallet-native, casino-grade experience that can scale as Zoggy grows into a broader Flipverse ecosystem.