Single-Player Roulette Game with PixiJS Rendering, Vue.js Statistics Bar, and WebSocket Integration

Objective:

Create a single-player roulette game where the gameplay is rendered via **PixiJS**.
A statistics bar managed by **Vue.js** will display the recent history of drawn numbers.
Draw results will be delivered via a simple **WebSocket server**, simulating real-time updates.

Functional Requirements:

Roulette Game (PixiJS):

- Render a **European Roulette wheel** (numbers 0–36).
- Animate a **spinning wheel** and a **ball** dropping onto the winning number.
- Ensure **smooth** spinning animation and **random, fair** number outcomes.
- **Highlight** the winning number visually on the wheel.
- Ball animation must be **dynamic and based on server response** (the wheel should stop with the ball landing on the number sent by the server).

Statistics Bar (Vue.js):

- Display the **last 5–10** drawn numbers in order (**most recent first**).
- (Optional) Display **player balance**, **total spins**, and **total wins/losses**.
- **Real-time updates**: After each spin, the new number immediately appears.

Gameplay Flow:

- The wheel must always be spinning. (idle and spin speeds will be different)
- Player presses a **"Spin"** button to start the game.
- The wheel spins and **stops** at a random number.
- The winning number is **pushed** into the statistics/history bar.
- Display a **"win"** or **"lose"** message based on a mock bet result.

Technical Requirements:

- **PixiJS** for the main canvas rendering (wheel, ball, animations).
- **Vue.js (Vue 3)** with **Composition API** for the Statistics Bar component and state management.

- **WebSocket server** (simple Node.js server) to simulate and broadcast random draw results.
- The **client** should **connect to the WebSocket server** and listen for new results to trigger animations and updates.
- **Separation of concerns**: PixiJS and Vue.js layers should remain independent (communicate via events, not direct DOM manipulation).
- **Clean, modular, and maintainable code** structure.
- **Responsive design** (desktop-first is acceptable).

Stretch Goals (Optional):

- **Particle/spark effects** when a number is drawn.
- **LocalStorage**: Save the last 10 results across sessions.
- **Sound effects** for spinning and winning animations.

Deliverables:

- Git branch: feature/roulette-pixi-vue
- A **README.md** file including:
  - Setup instructions
  - Brief architecture explanation
  - Component breakdown

Estimated Time:

- **Please submit within 7 days after receiving the task.**
- *~24 hours for MVP (Minimum Viable Product)*
- *+8 hours for Stretch Goals*

Assets: link