A. Rekomendowana Technologia

Zalecam użycie **Node.js** na Raspberry Pi.

- **Dlaczego:** Frontend też jest w JS/TS. Używamy tej samej biblioteki (socket.io) po obu stronach, co gwarantuje 100% kompatybilności i najmniejsze opóźnienia (latency).

B. Rola Backendu (Co ma robić?)

1. **Source of Truth (Jedyne Źródło Prawdy):**
    - Backend trzyma stan gry. To backend decyduje, czyj jest ruch, ile punktów zostało i kto wygrał.
    - Frontend jest "głupi" (dumb terminal) – tylko wyświetla to, co przyśle serwer i wysyła komendy (np. "Start Gry").
1. **Obsługa Hardware'u:**
    - Backend nasłuchuje Arduino (po USB/Serial).
    - Tłumaczy sygnały z matrycy na punkty (np. sygnał X:5 Y:10 -> "Trafiono w potrójne 20").
1. **Logika Trybów Gry (Game Engine):**
    - Zasady 301/501 (Double Out, Bust).
    - Zasady Cricket (zamykanie sektorów).
    - Zasady Killer (przydzielanie żyć).
    - To wszystko musi być liczone na serwerze, frontend tylko dostaje wynik.

C. Baza Danych (Dla profili i historii)

Wymagam trwałego zapisu danych (Persistence).

- Backend musi postawić bazę danych (np. **SQLite** dla prostoty lub **MongoDB**).
- **Cel:** Obsługa logowania użytkowników, historia meczów, statystyki (średnia rzutów), rankingi turniejowe.

3. Protokół Komunikacji (API Contract)

Kanał 1: WebSockets (Rozgrywka w czasie rzeczywistym)

Używamy biblioteki **Socket.io**.

- **Event game_update (Server -> Client):** Wysyłany po KAŻDYM rzucie. Zawiera pełny stan gry (wyniki, czyja tura, stan tarczy w Crickecie).
- **Event game_start / game_reset (Client -> Server):** Sterowanie rozgrywką.

Kanał 2: HTTP REST API (Dane statyczne)

Do rzeczy, które nie muszą być "na żywo":

- POST /api/login – logowanie gracza.
- GET /api/profile/:id – pobranie statystyk gracza.
- GET /api/leaderboard – pobranie rankingu ogólnego.

4. Wymagania Sieciowe (Network)

Aby frontend mógł się połączyć z Raspberry Pi:

1. **Stałe IP:** Raspberry Pi musi mieć statyczny adres IP w sieci lokalnej (np. 192.168.1.100) lub mDNS (http://darts.local).
2. **CORS:** Serwer musi mieć odblokowany CORS (origin: *) na czas developmentu.
3. **Hosting plików:** Docelowo Raspberry Pi (np. przez Nginx) będzie serwować pliki frontendu (zbudowany folder dist), aby każdy w sieci Wi-Fi mógł wejść na stronę wpisując IP malinki.