Smart Contracts :: Blockchain Developer

What you'll build (end-to-end)

1. Smart Contracts
    - GameToken.sol (ERC-20, 18 decimals)
        - mint(address to, uint256 amount) — callable only by TokenStore.
        - Events: Minted(to, amount).
    - TokenStore.sol (USDT → GT on-ramp)
        - Constructor: address usdt, address gameToken, uint256 gtPerUsdt (e.g., 1e18 ⇒ 1 USDT → 1 GT).
        - buy(uint256 usdtAmount):
            - Pull USDT (6 decimals) with transferFrom.
            - Mint gtOut = usdtAmount * gtPerUsdt / 1e6 GT to msg.sender.
            - Event: Purchase(buyer, usdtAmount, gtOut).
            - Guards: CEI order + nonReentrant.
        - withdrawUSDT(address to, uint256 amount) — owner only.
    - PlayGame.sol (escrow + payout)
        - createMatch(bytes32 matchId, address p1, address p2, uint256 stake) (owner/manager only).
        - stake(bytes32 matchId):
            - Caller must be p1 or p2.
            - Pull exactly stake GT via transferFrom.
            - Mark each side staked; when both in, status = STAKED; startTime = block.timestamp.
        - commitResult(bytes32 matchId, address winner):
            - msg.sender must be the backend/operator address.
            - winner must be p1 or p2.
            - Transfer 2 × stake GT to winner; status = SETTLED.
        - refund(bytes32 matchId) after timeout (e.g., 24h) if not settled; returns each stake.
        - Events: MatchCreated, Staked, Settled, Refunded.
        - Guards: status checks, idempotency, nonReentrant.

2. Backend (Minimal Gateway)
    - Node.js (Express or NestJS) + ethers.js.
    - Endpoints:
        - GET /purchase?amount=USDT → calls TokenStore.buy().
        - POST /match/start → calls createMatch then coordinates stake flow.
        - POST /match/result → calls commitResult.
    - Simple .env (RPC URL, private key of backend, contract addresses).
3. Simple Frontend (very basic)
    - One HTML page with:
        - "Buy GT with USDT" (amount input → calls /purchase).
        - "Create/Stake Match" (matchId, stake amount → calls /match/start).
        - "Submit Result" (matchId, winner address → calls /match/result).
    - Show wallet GT balance (via balanceOf) and last event logs.
4. Indexer / Leaderboard (lightweight)
    - An event listener script (Node + ethers) that:
        - Listens to Purchase, Staked, Settled, Refunded.
        - Maintains an in-memory or SQLite table:
            - winsByAddress, totalGTWon, matchesPlayed.
        - Serves GET /leaderboard (top 10 winners by GT won).

Acceptance checklist

- USDT → GT: 1 USDT mints exactly 1e18 GT when gtPerUsdt = 1e18.
- Escrow: Both stakes required before match becomes STAKED.
- Payout: Winner receives exactly 2 × stake; loser's GT unchanged.
- Double-submit safe: Second commitResult reverts (status guard).
- Refund: Works only after timeout and before a result.
- Events: Emitted for all critical actions; reader shows leaderboard.

What to deliver

- contracts/ (3 contracts + migrations/deploy script)
- api/ (Express/Nest app with 3 endpoints + .env.example)
- web/ (HTML+JS; no framework required)
- tools/leaderboard.js (event listener + simple API)

- README.md **(how to run all three; happy-path demo)**
- GIT **(Link of Github Repo)**

Time guidance (2.5 hours)

- **Contracts: 70–80 min**
- **Backend: 35–45 min**
- **Frontend: 20–30 min**
- **Leaderboard: 20–25 min**
- **README + smoke test: 10–15 min**