Domain Analysis

Competition

- ✓ **Competition lifecycle**
  - Create → Publish → Sell tickets → Close/draw → Pay winners.
- ✓ Types of competition
  - Paid,
  - Free,
  - Jackpot,
  - Mini-games,
  - Subscription,
  - Spin Wheel.

Ticket semantics

Ticket Types:

- ✓ *Competition Tickets* (Paid, Competition-Specific)
- ✓ *Universal Tickets* (Earned, Usable on Any Competition).

Winner Selection Methods

- Random Draw,
- Manual Pick,
- Skill/Score,
- First-Entry.

Instant wins

- The Admin Defines Winning Numbers Per Instant Prize;
- The System Must Compare Purchased Ticket Numbers With That Set And Mark Wins Instantly.

Free entry (legal compliance)

- for paid competitions you must offer a free postal route or a skill question.

Subscriptions & perks

- subscribers get monthly auto-entries, free tickets, or credit. Admin side decides tiers & perks.

Points/Wallet/Spin/Rewards

- point accrual
- redemption to site credit (non-withdrawable)
- wallets for site credit vs cash wallet (withdrawable).
- Spin wheels have probabilities per segment and immediate payout.

Responsible gaming limits

- daily/weekly/month limits and self-exclusion features must be enforced by the backend.

Map to the backend logic

1) Buying a ticket (paid competition)

Business rules

- ✓ Validate competition is active, not sold out, within per-user limit, and user is allowed by age.
- ✓ Deduct money from user's wallet or process external payment.
- ✓ Generate N ticket numbers (unique within competition) and insert ticket rows.
- ✓ After successful purchase: check instant wins for those ticket numbers and credit prizes to user.

Implementation highlights

- Use a DB transaction.
- Reserve ticket numbers atomically (use a sequence counter per competition or SELECT MAX(ticket_number) FOR UPDATE).
- If using random ticket numbers: either generate an unused random number and insert or pre-generate ticket pool and pop with SELECT ... FOR UPDATE LIMIT 1 (fast & safe).
- After insert, call checkInstantWins(transactionConn, competitionId, ticketNumbers, userId).

Notes

- Prefer FOR UPDATE locking on the competition row or a separate ticket_sequence row to avoid race conditions when allocating sequential numbers.
- If many concurrent buyers, pre-allocating blocks or using Redis for counters increases throughput.

2) Instant Win checking

Business rules

- Admin provides a set of winning ticket numbers per instant prize.
- Immediately after purchase, compare buyer's ticket numbers to instant winning numbers; if match, mark prize claimed and create wallet credit or prize record.

Implementation

- Store instant winning numbers in instant_wins(competition_id, ticket_number, prize_id).
- When ticket is created, SELECT * FROM instant_wins WHERE competition_id=? AND ticket_number IN (?) FOR UPDATE and mark claimed_by and claimed_at.
- Credit wallet via wallet_transactions.

Edge cases

- Two users cannot claim same instant win: enforce claimed_by NULL check under transaction with FOR UPDATE or update with WHERE claimed_by IS NULL and check affected rows.

3) Drawing winners (Random draw)

Business rules

- Draw random winners from pool of tickets (include paid + postal free + universal used for competition)
- Must be auditable and reproducible (save draw seed and algorithm)
- For jackpot large draws, may be executed manually or by scheduled worker after max_tickets or end_date.

Implementation

- Lock the competition row, check it's in DRAW_PENDING.
- Get ticket count N, pick random numbers using DB or worker:

- Option A (DB): SELECT ticket_id FROM tickets WHERE competition_id=? ORDER BY RAND() LIMIT X — easy but not performant at millions.
- Option B (Worker): Use reservoir sampling / pre-shuffled list or use a reproducible RNG seeded with stored draw_seed, shuffle indices with algorithm in worker.
- Insert into winners table with method='AUTO' and draw_meta={seed,...}.

4) Free entry (postal) handling

Business rules

- Accept and store postal entries (manually entered by admin or via CSV after receiving post).
- Free postal entries get ticket records (type POSTAL_FREE) and must be eligible in draws.
- Admin should be able to mark postal entries as “received” with timestamp.

Implementation

- Provide an admin CSV upload endpoint that inserts into postal_entries and creates tickets pointing to a special user_id = NULL or system_free_user pseudo-user.
- On draw, include these tickets.

5) Subscription auto-entry (monthly)

Business rules

- For each subscription tier, system auto-enters subscribers into that month’s subscriber competition with configured number of tickets.
- Should run via scheduled job at a given cron (e.g., monthly).

Implementation

- Worker run:
    - Query active subscribers for tier.

- For each subscriber: INSERT ticket rows and mark as type='UNIVERSAL' or SUBSCRIPTION.
- Log results and send notification.

Tip

- Run idempotent jobs: mark a subscription_entry_run table with month/year to avoid double entry.

6) Spin the wheel

Business rules

- Wheel segments have probability percentages summing to 100.
- On spin, pick segment according to probability and grant prize immediately.
- Enforce per user spin limits.

Implementation

- Precompute cumulative probabilities, generate random number [0,100), pick segment.
- Use transaction to log spin and grant prize (wallet credit or item). Enforce spin limit by SELECT spins_today FROM spin_history WHERE user_id=...? or store per-user spin counters in Redis for fast checks.

Concurrency, consistency & money safety

- **Transactions**: every money movement + ticket allocation must be inside a DB transaction. If you use external payment gateways, create a purchase in DB in PENDING then finalize in callback/webhook.
- **Row locking**: use SELECT ... FOR UPDATE on sequences or ticket counters. For very high throughput, use Redis atomic counters and claim from prepopulated ticket buckets.
- **Idempotency**: for payment webhooks and subscription workers, ensure idempotent endpoints (use idempotency_key).

- **Auditing**: every admin action, payout, draw must be stored in admin_activities or audit_logs table for compliance.

Admin UI endpoints (suggested)

- POST /admin/competitions → create competition (payload includes type, pricing, rules).
- PUT /admin/competitions/:id → update competition.
- POST /admin/competitions/:id/instant-wins → upload CSV of winning ticket numbers.
- POST /admin/competitions/:id/start-countdown → manually start threshold.
- POST /admin/competitions/:id/run-draw → run draw (returns winners).
- GET /admin/competitions/:id/tickets → paginate tickets.
- POST /admin/postal-entries/upload → upload/insert postal free entries.
- POST /admin/spin-wheels/:id/segments → manage wheel segments.

1. **competitions**
    - id, type (PAID/FREE/JACKPOT/MINI/SUBSCRIPTION/SPIN), title, price, max_tickets, start_date, end_date, status, free_entry_route_text, skill_question (question, answer), threshold_auto_start, ticket_model (percentage free), published_by, created_at.
2. **users**
    - id (BINARY(16) using UUID_TO_BIN), email, phone, role, kyc_status, wallet ids, subscription_tier, spend_limits (json), created_at.
3. **tickets**

- id, competition_id, user_id (nullable for postal/free entries), ticket_number (int), type ENUM('COMPETITION','UNIVERSAL','POSTAL_FREE'), purchase_id (nullable), created_at.
- Unique(competition_id, ticket_number).

4. **purchases**
   - id, user_id, total_amount, payment_method, status, created_at. Link to purchased tickets.
5. **instant_wins**
   - id, competition_id, ticket_number, prize_id, claimed_by, claimed_at. Admin can upload CSV of winning numbers.
6. **winners**
   - id, competition_id, ticket_id, user_id, prize_desc, method, paid_at.
7. **wallets / wallet_transactions**
   - cash vs site_credit wallets. Redemptions affect wallet balances and create wallet_transactions.
8. **referrals, referral_history, user_levels, daily_rewards_config** (already in your schema) — used for gamification.
9. **spin_wheels / spin_segments / spin_history**
   - spin_wheels: id, name, rules, spins_per_user, min_tier
   - spin_segments: wheel_id, segment_index, prize_type, prize_value, probability_pct — total must sum to 100 per wheel.
   - spin_history logs each spin for audit.
10. **postal_entries**
    - id, competition_id, name, address, email, phone, submitted_at — to track legal free entries.

11. **audit_logs**
    - id, actor_id, action, target_type, target_id, ip, meta_json, created_at — every admin or money movement action must be audit logged.