Club Court Manager
Manage. Reserve. Play.

CCM Calculations Reference

Every formula in Club Court Manager™ and Club Rankings™ — with impact examples

Product	Club Court Manager™ + Club Rankings™
Operator	StoufferAI
Version	July 2026
PDF	/docs/ccm-calculations-reference.pdf

> **How to use:** Each entry lists purpose, formula, defaults, a numeric impact example, and code location. Reservations have no court-fee math; ranking uses two parallel systems (public points + hidden Elo).

Table of contents

Product: Club Court Manager™ (CCM) + Club Rankings™ (CR)

Operator: StoufferAI

Audience: Club admins, operators, power users

Source of truth: `lib/rankings/**`, `lib/billing*.js`, `lib/rules.js`, `public/js/court-grid.js`

Version: July 14, 2026

For CR-only deep dive, see also CLUB-RANKINGS-SPECIFICATIONS.md.

How to read this document

Each calculation uses the same template:

| Field | Meaning |

|-------|---------|

| **Purpose** | What members/admins see or what decision it drives |

| **Formula** | Plain-English math |

| **Defaults** | Admin setting or system default |

| **Impact example** | Concrete numbers → what changes |

| **Code** | Implementation location |

Two parallel ranking systems (CR):

| System | Visible? | Drives |

|--------|----------|--------|

| **Public ranking points** | Yes | Leaderboard order |

| **Hidden Elo** | No | Opponent/partner strength, optional club rating |

CCM reservations: No court fees or wait-queue math. Overlap is enforced by the database; availability is grid painting from existing bookings.

Part 1 — Club Rankings

CR-01. Recalculation pipeline

CR-01. Recalculation pipeline

Purpose

Rebuild all player standings after events are confirmed/completed.

Formula

Replay every completed event in **event date order** (then event ID). For each player row: compute performance → apply weekly cap → update point buckets → update Elo/club rating → update W/L/T and streaks. After all events: inactivity decay, volunteer/sportsmanship sync, achievements.

Defaults

Runs on full recalc or section/series recalc.

Impact example

Admin confirms Saturday ladder → all four players' `performance_points` and leaderboard ranks update on next recalc.

Code

`lib/rankings/engine/recalculate.js` → `recalculateFromEvents`

CR-02. Win percentage component

CR-02. Win percentage component

Purpose

Reward session win rate in the performance blend (0–100).

Formula

If no games → **50**. Else `winRate = (wins + ties×0.5) ÷ (wins+losses+ties)`; component = clamp(winRate×100, 0, 100).

Defaults

Weight **40%** (`wins_weight`).

Impact example

6W–2L → winRate 0.75 → component **75** → with 40% weight contributes **30** points to blend numerator.

Code

`lib/rankings/engine/performance-score.js` → `winPctScore`

CR-03. Placement component

CR-03. Placement component

Purpose

Reward finishing position in the event field.

Formula

If placement or fieldSize missing, or fieldSize ≤ 1 → **50**. Else `((fieldSize − placement) ÷ (fieldSize − 1)) × 100`, clamped.

Defaults

Weight **20%** (`placement_weight`). Requires `placement` (1 = best) and `fieldSize`.

Impact example

1st of 8 → **100**. 4th of 8 → **~57**. Last of 8 → **0**. Placement not entered → **50** (neutral).

Code

`lib/rankings/engine/performance-score.js` → `placementScore`

CR-04. Point differential component

CR-04. Point differential component

Purpose

Reward scoring margin vs peers in the same event.

Formula

`diff = points_for − points_against`. `maxAbsDiff` = largest |diff| among all players in the event (min 1). Component = clamp(50 + (diff ÷ maxAbsDiff)×50, 0, 100).

Defaults

Weight **15%** (`point_diff_weight`).

Impact example

Event max diff is 10. Player +10 → **100**; 0 → **50**; -10 → **0**. Drives ~7.5 blend points swing at 15% weight between best and worst margin.

Code

`lib/rankings/engine/performance-score.js` → `pointDiffScore`

CR-05. Opponent strength component

CR-05. Opponent strength component

Purpose

Reward beating a strong field (uses hidden Elo of other players).

Formula

`component = clamp(50 + ((opponentAvgElo − 1500) ÷ 400)×50, 0, 100)`. Opponent avg = mean Elo of **other** players in the event.

Defaults

Weight **15%** (`opponent_strength_weight`). Baseline **1500**, spread **400**.

Impact example

Opponents average 1700 → component **75**. All at 1500 → **50** (neutral).

Code

`lib/rankings/engine/performance-score.js` → `strengthScore`; opponent avg in `recalculate.js`

CR-06. Partner strength component

CR-06. Partner strength component

Purpose

Adjust blend when partner is stronger/weaker (doubles).

Formula

Same as CR-05 on partner's Elo. No partner → treat as **1500** → component **50**.

Defaults

Weight **10%** (`partner_strength_weight`).

Impact example

Partner Elo 1300 → component **25** → pulls blended score down vs solo-neutral **50**.

Code

`lib/rankings/engine/performance-score.js` → `strengthScore`

CR-07. Age handicap component

CR-07. Age handicap component

Purpose

Optional senior boost in performance blend. **Does not** change hidden Elo.

Formula

Missing age → **50**. Else `clamp(50 + ((age − baseline) ÷ spread)×50, 0, 100)` .

Defaults

Weight **10%** (`age_weight`). Baseline **50** years, spread **25** years. Set weight to **0** to disable.

Impact example

Age 75, baseline 50, spread 25 → component **100**. If admin sets age weight to **10%**, adds ~5 blend points vs age 50. Age 25 → component **0**.

Code

`lib/rankings/engine/performance-score.js` → `ageHandicapScore` , `ageAtDate`

CR-08. Blended performance score

CR-08. Blended performance score

Purpose

Single 0–100 session quality score (stored on result row; not the leaderboard headline).

Formula

Weighted average of six components. If sum of weights = 0 → **50**.

Defaults

Global weights 40 / 20 / 15 / 15 / 10 / 10 (age). Series can override six blend weights when `use_custom_weights` is enabled.

Impact example

Components all **50** → blend **50** → adjustment **0**. Win-heavy session: 75/100/100/50/50/50 at pickleball preset 40/22/18/12/8 → blend **80.00** (see CR-35).

Code

`lib/rankings/engine/performance-score.js` → `blendPerformanceScore`, `computePerformanceScore`

CR-09. Performance point adjustment (raw)

CR-09. Performance point adjustment (raw)

Purpose

Convert performance score into `performance_points` delta before weekly cap.

Formula

`rawAdjustment = ((performanceScore − 50) ÷ 5) × effectiveEventWeight`, rounded 2 decimals.

Defaults

`effectiveEventWeight = event.weight × series.overall_weight_multiplier` (each defaults **1.0**).

Impact example

Score **60**, weight **1.0** → **+2.0** performance points. Score **80**, weight **1.0** → **+6.0**. Score **50** → **0**.

Code

`lib/rankings/engine/performance-score.js` → `performanceAdjustment`

CR-10. Weekly gain cap

CR-10. Weekly gain cap

Purpose

Limit how much performance_points can rise per ISO week.

Formula

Week key `YYYY-Www` from event date. `applied = min(rawDelta, max(0, maximumWeeklyGain - priorWeekTotal))` for positive deltas.

Defaults

`maximum_weekly_gain` = **15**.

Impact example

Monday **+10**, Wednesday **+10** raw → second applied only **+5** (week total **15**). Thursday **+3** raw → **0** applied.

Code

`lib/rankings/engine/weekly-clamp.js` → `clampWeeklyAdjustment`

CR-11. Weekly loss cap

CR-11. Weekly loss cap

Purpose

Limit how much performance_points can fall per ISO week.

Formula

For negative deltas: `applied = -min(|rawDelta|, max(0, maximumWeeklyLoss + priorWeekTotal))`.

Defaults

`maximum_weekly_loss` = **10**.

Impact example

Monday **-8**, Tuesday **-8** raw → second applied **-2** (week total **-10**).

Code

`lib/rankings/engine/weekly-clamp.js` → `clampWeeklyAdjustment`

CR-12. Expected Elo score

CR-12. Expected Elo score

Purpose

Win probability vs opponent for Elo update.

Formula

`expected = 1 ÷ (1 + 10^((opponentElo − playerElo) ÷ 400))`. Missing ratings → **1500**.

Defaults

Fixed algorithm.

Impact example

1500 vs 1500 → expected **0.5** (coin flip).

Code

`lib/rankings/engine/skill-rating.js` → `expectedScore`

CR-13. Elo K-factor

CR-13. Elo K-factor

Purpose

Smaller rating moves for experienced players.

Formula

Games 0–9 → K **32**; 10–29 → **24**; 30+ → **16**.

Defaults

Fixed tiers.

Impact example

New player (0 games) gains **+16** on equal win; veteran (30+ games) gains **+8** on same outcome.

Code

`lib/rankings/engine/skill-rating.js` → `kFactor`

CR-14. Elo rating update

CR-14. Elo rating update

Purpose

Hidden per-sport skill index after each event; drives CR-05/06 and club rating.

Formula

Session score from W/L/T (CR-15). `delta = round(K × (score − expected), 2)`; `newElo = round(old + delta, 2)`.

Defaults

Start **1500** (`DEFAULT_SKILL_RATING`). **Not** weekly-capped.

Impact example

1500 vs 1500 avg, win, K=32 → **+16** → Elo **1516.0**. Loss → **−16** → **1484.0**.

Code

`lib/rankings/engine/skill-rating.js` → `updateElo`

CR-15. Session outcome score (Elo input)

CR-15. Session outcome score (Elo input)

Purpose

Collapse W/L/T into one 0–1 score for Elo.

Formula

No games → **0.5**. Else `(wins + ties×0.5) ÷ total`.

Defaults

Win=1, tie=0.5, loss=0 implied.

Impact example

3W–1L → **0.75** (strong win session). All ties → **0.5**.

Code

`lib/rankings/engine/skill-rating.js` → `sessionOutcomeScore`

CR-16. Declared rating → Elo (seed)

CR-16. Declared rating → Elo (seed)

Purpose

Map self-reported 1.0–5.5 roster rating to internal Elo before first ranked game.

Formula

Anchor **3.5 ↔ 1500**; ±0.5 skill ≈ ±200 Elo. `elo = 1500 + (skill − 3.5)×400`, clamp skill 1.0–5.5.

Defaults

Only when `club_rating_enabled` and first game.

Impact example

Self-reported **4.0** → seeds Elo **1700** before first recalc update.

Code

`lib/rankings/club-rating-scale.js` → `memberRatingToElo`

CR-17. Elo → club rating (display)

CR-17. Elo → club rating (display)

Purpose

Member-visible 1.0–5.5 club rating from hidden Elo.

Formula

`skill = 3.5 + (elo − 1500) ÷ 400`, clamp 1.0–5.5, round 1 decimal.

Defaults

Requires `club_rating_enabled`.

Impact example

Elo **1516** → club rating **3.5** (display). Elo **1700** → **4.0**.

Code

`lib/rankings/club-rating-scale.js` → `eloToClubRating`

CR-18. Session club rating delta

CR-18. Session club rating delta

Purpose

Change in club rating after one event (session history).

Formula

`round(nextClubRating − prevClubRating, 2)` ; null if either missing.

Defaults

Stored on `ranking_results.club_rating_delta` .

Impact example

3.5 → 3.54 after strong session → delta **+0.04** shown as “+0.04” in history.

Code

`lib/rankings/club-rating-scale.js` → `clubRatingDelta`

CR-19. Club rating vs declared delta (profile)

CR-19. Club rating vs declared delta (profile)

Purpose

“How far above/below self-reported?” on profile — not the session delta.

Formula

`clubRating − declaredRating` when feature on and `events_played ≥ min_events_for_club_rating` .

Defaults

`club_rating_enabled` **false**; `min_events_for_club_rating` **1**. Display toggles gate profile / my ranking / leaderboard columns.

Impact example

Declared 3.5, club rating 3.8 after 2 events → shows **+0.3** above self-report on profile. With feature off → member sees only declared rating.

Code

`lib/rankings/club-rating-display.js` → `skillRatingToApi`

CR-20. Participation bonus

CR-20. Participation bonus

Purpose

Flat points per event appearance.

Formula

Each completed result row: `participation_points += participation_bonus`.

Defaults

0 globally; pickleball preset **1**.

Impact example

Bonus **1**, 5 events → **+5.0** participation_points → total rank rises even with neutral performance sessions.

Code

`lib/rankings/engine/recalculate.js`; `participation_bonus` in settings

CR-21. Champion bonus

CR-21. Champion bonus

Purpose

Flat bonus for 1st-place finish.

Formula

When `placement === 1`: `performance_points += champion_bonus`.

Defaults

0 globally; pickleball preset **3**.

Impact example

Champion bonus **3** on 1st place adds **+3.0** performance on top of performance adjustment (CR-09).

Code

`lib/rankings/engine/recalculate.js`

CR-22. Volunteer bonus

CR-22. Volunteer bonus

Purpose

Points for volunteering at sessions (admin “Vol” flag).

Formula

One bonus row per volunteer session: `volunteer_points = sum(volunteer_bonus)`.

Defaults

`volunteer_bonus` **5**.

Impact example

3 volunteer sessions → **15.0** volunteer_points. Can move player several ranks without winning.

Code

`lib/rankings/volunteers.js`, `lib/rankings/bonuses.js`

CR-23. Fast-confirm bonus

CR-23. Fast-confirm bonus

Purpose

Sportsmanship points for confirming results quickly.

Formula

If confirm within `fast_confirm_hours` of submission: add `fast_confirm_bonus` once to `sportsmanship_points`.

Defaults

Bonus **0** (off); window **48** hours.

Impact example

Bonus **1**, confirm at 12h → **+1.0** sportsmanship_points.

Code

`lib/rankings/bonuses.js` → `tryFastConfirmBonus`

CR-24. Inactivity decay

CR-24. Inactivity decay

Purpose

Penalize idle players on performance_points only (after full replay).

Formula

If enabled and days since last event > `inactivity_days` : `periods = floor((days − threshold) ÷ 30) + 1` ; subtract `periods × inactivity_penalty` , floor at 0.

Defaults

Off. Threshold **90** days, penalty **5** per 30-day period.

Impact example

120 days idle, 42 performance points → 2 periods × 5 → **32.0** performance_points. May drop several leaderboard positions.

Code

`lib/rankings/inactivity.js` → `applyInactivityDecay`

CR-25. Total ranking points

CR-25. Total ranking points

Purpose

Leaderboard sort key and public “total points”.

Formula

`performance + participation + volunteer + sportsmanship` (each rounded 2 decimals).

Defaults

N/A.

Impact example

12.5 + 3 + 10 + 1 = **26.5** total. Player with higher total ranks above on leaderboard.

Code

`lib/rankings/constants.js` → `totalPoints`

CR-26. Leaderboard eligibility

CR-26. Leaderboard eligibility

Purpose

Hide new players until enough events.

Formula

List only profiles where `events_played ≥ min_events_for_leaderboard`; sort by total points descending.

Defaults

`min_events_for_leaderboard` **3**.

Impact example

Player with 2 events and 20 points computed → **not listed** on leaderboard until 3rd event.

Code

`lib/routes/rankings/leaderboard.js`

CR-27. Rolling window start

CR-27. Rolling window start

Purpose

“Last N months” leaderboard replay window.

Formula

`rollingStart = today − rolling_months` calendar months.

Defaults

`rolling_months` **12**.

Impact example

Event 13 months ago counts on all-time but **excluded** from rolling leaderboard — player may rank higher on rolling than all-time.

Code

`lib/rankings/engine/recalculate.js` → `rollingMonthStart`

CR-28. Series effective event weight

CR-28. Series effective event weight

Purpose

Mini-league events can count more/less toward club totals.

Formula

`effectiveWeight = event.weight_multiplier × series.overall_weight_multiplier`.

Defaults

Each **1.0**. Series with `counts_toward_overall = false` excluded from club-wide recalc.

Impact example

Event weight 1.5 × series 2.0 = **3.0** → score 60 yields raw adjustment **+6.0** instead of +2.0.

Code

`lib/rankings/series.js`, `recalculate.js`

CR-29. Dashboard “most improved”

CR-29. Dashboard “most improved”

Purpose

Highlight players gaining points in rolling window vs all-time.

Formula

`improvement = allTimeTotalPoints − rollingWindowTotalPoints` (positive only); top 5.

Defaults

Uses same `totalPoints` and rolling months.

Impact example

All-time 40, rolling 28 → improvement **12** — appears on admin dashboard “most improved” list.

Code

`lib/rankings/dashboard.js` → `buildClubDashboard`

CR-30. Club Play → result aggregation

CR-30. Club Play → result aggregation

Purpose

Turn live game scores into ranking result rows.

Formula

Per completed game: W/L/T from scores; accumulate points for/against; doubles sets partner. **Placement (KOTC & round robin):** wins DESC → losses ASC → ties DESC → point differential DESC → weighted tiebreaker (club weights renormalized without placement component) → member ID. **Ladder:** placement from final ladder order.

Defaults

Feeds same engine as CSV/manual entry. Tiebreaker uses section or mini-league custom weights.

Impact example

Two players both 2W–0L with +2 pt diff — older player or stronger-opponent schedule can win placement via tiebreaker; both still receive full performance scores on recalc.

Code

`lib/rankings/club-play/aggregate.js` → `aggregateGamesToResults`; `lib/rankings/club-play/session-placement.js` → `assignSessionPlacements`

CR-30b. Session placement tiebreaker

CR-30b. Session placement tiebreaker

Purpose

Break ties when two or more players share the same session record after wins, losses, ties, and point differential.

Formula

Blend win %, point differential, opponent strength, partner strength, and age handicap using club (or mini-league custom) weights with **placement weight set to 0** and remaining weights renormalized. Higher score ranks higher. If still tied, lower `member_id`.

Defaults

Same weight preset as performance score (CR-02–CR-07). Opponent/partner strength from hidden `ranking_skill_ratings` at session date.

Impact example

Both 2W–0L, +2 pt diff — player who faced stronger opponents (higher hidden Elo field) wins placement; leaderboard still awards both weighted performance points.

Code

`lib/rankings/engine/performance-score.js` → `computeTiebreakScore`; `lib/rankings/club-play/session-placement.js` → `assignSessionPlacements`

CR-31. Stored Glicko RD

CR-31. Stored Glicko RD

Purpose

Reserved field on skill rating row.

Formula

Not recalculated in current implementation; stored constant **350** (`DEFAULT_SKILL_RD`).

Defaults

350.

Impact example

No effect on leaderboard or Elo today.

Code

`lib/rankings/constants.js`

CR-32. Win/loss streak

CR-32. Win/loss streak

Purpose

Profile stat and achievements (“Hot streak”).

Formula

Session “won” if `wins > losses`. Streak increments on consecutive wins, negative on losses; `longest_streak` tracks max |streak|.

Defaults

N/A.

Impact example

Three winning sessions → streak **3**; may unlock achievement; does not add points.

Code

`lib/rankings/engine/recalculate.js` → `updateStreak`

CR-33. Achievement thresholds

CR-33. Achievement thresholds

Purpose

Badges when profile crosses rules.

Formula

Rules compare `events_played`, `championships`, `longest_streak`, `total_points`, etc. to thresholds.

Defaults

e.g. “Century club” at **100+** total points.

Impact example

Total crosses 100 → badge awarded; **no point change**.

Code

`lib/rankings/achievements.js` → `profileMeetsRule`

CR-34. Leaderboard visibility

CR-34. Leaderboard visibility

Purpose

Who can view leaderboard API.

Formula

Not numeric — `members_only` requires sign-in; `public` allows anonymous read.

Defaults

`members_only`.

Impact example

Public setting lets marketing page show top 10 without login.

Code

`lib/routes/rankings/leaderboard.js`

CR-35. Worked example — full session

CR-35. Worked example — full session

Purpose

End-to-end illustration (Player A, pickleball preset, first event of week).

Formula

Defaults

—

Impact example

Player A jumps to **rank 1** on a small leaderboard after one strong winning session.

Code

See `docs/CLUB-RANKINGS-SPECIFICATIONS.md` §16

Step	Result
Components	Win 75, placement 100, pt diff 100, opp 50, partner 50
Blend (40/22/18/12/8)	**80.00**
Raw adjustment	**+6.0**
Weekly cap	**+6.0** applied
Participation	**+1.0**
Champion	**+3.0** performance
Total points	**10.0**
Elo	**1516.0**
Club rating (if enabled)	**~3.5+** from 1516 Elo

Part 2 — CCM Core

Reservations have no court fees or wait-time math. Overlap is enforced by Postgres; the grid paints occupied slots from existing rows.

CCM-01. Effective monthly price

CCM-01. Effective monthly price

Purpose

Price shown at checkout and on billing admin.

Formula

`custom_price_monthly ?? plan.priceMonthly`; same for annual. Dollars → cents: `round(dollars × 100)`.

Defaults

CCM **$40**/mo (up to 6 courts); Rankings-only **$19**/mo; Enterprise custom (optional read-only portfolio login for a property-management firm with several CCM communities). Legacy ids `starter` / `growth` / `pro` map to CCM economics.

Impact example

CCM list **$40**; tenant custom **$35** → checkout charges **$35/mo**.

Code

`lib/pricing.js` → `effectivePricing`; `lib/billing-plans.js`

CCM-02. Overage count (members or courts)

CCM-02. Overage count (members or courts)

Purpose

Units above plan inclusion that drive extra monthly charge.

Formula

Defaults

Rankings-only **150** members; CCM **6** courts included, **unlimited** members (`memberLimit` null).

Impact example

8 active courts on CCM → **2** overage courts. **105** active on Rankings → **5** overage members.

Code

`lib/billing-overage.js` → `overageMembers`, `overageCourts`

CCM-03. Overage charge

CCM-03. Overage charge

Purpose

Extra monthly amount on billing estimate (and Rankings invoice renewal for member overage).

Formula

Defaults

CCM **$5**/extra court; Rankings **$0.40**/extra member. Skipped when custom pricing flag set.

Impact example

2 extra courts on CCM → **$10/mo**. 5 overage on Rankings → **$2.00/mo**.

Code

`lib/billing-overage.js` → `overageChargeForPlan`

CCM-04. Estimated monthly total

CCM-04. Estimated monthly total

Purpose

Admin billing dashboard total.

Formula

`round((baseMonthly + overageCharge) × 100) / 100`.

Defaults

Skips overage if custom pricing flag set. CCM court overage uses live court count.

Impact example

CCM $40 + 2 courts × $5 → **$50** estimated.

Code

`lib/billing-overage.js` → `estimatedMonthlyTotal`

CCM-05. Upgrade nudge

CCM-05. Upgrade nudge

Purpose

Suggest full CCM when Rankings-only overage makes the next tier a better fit.

Formula

If `estimatedTotal ≥ nextPlan.priceMonthly` → show nudge to next self-serve plan.

Defaults

Self-serve upgrade chain: **rankings → ccm** only (no Starter/Growth/Pro ladder).

Impact example

Rankings estimate ~**$45**/mo with member overage → nudge to Club Court Manager (**$40** base + court add-ons).

Code

`lib/billing-overage.js` → `upgradeNudge`

CCM-06. Trial days remaining

CCM-06. Trial days remaining

Purpose

Billing UI countdown.

Formula

`ceil((trial_ends_at − now) ÷ 86400000)` days.

Defaults

`TRIAL_DAYS` **14** on onboard.

Impact example

12 days left shown; at 0, subscription must be active to avoid trial expiry flows.

Code

`lib/billing.js`

CCM-07. Active member count (billable)

CCM-07. Active member count (billable)

Purpose

Rankings member overage and analytics. CCM self-serve is **not** capped by member count.

Formula

SQL count: `status = active`, `activated_at IS NOT NULL`, not deactivated.

Defaults

Imported roster awaiting invite **not** counted. CCM plan `memberLimit` is null (unlimited).

Impact example

98 activated + 10 pending invite → count **98** for Rankings billing / analytics.

Code

`lib/billing.js`

CCM-08. Booking slot alignment

CCM-08. Booking slot alignment

Purpose

Reject bookings that don't align to grid increment.

Formula

`(startMin − openMin) % increment_min === 0` and same for end; `endMin > startMin`.

Defaults

Section `increment_min` often **30**.

Impact example

Book 10:15 start with 30-min grid → **rejected**.

Code

`lib/rules.js` → `validateBooking`

CCM-09. Maximum booking length

CCM-09. Maximum booking length

Purpose

Cap session duration.

Formula

`(endMin − startMin) ≤ max_length_min`.

Defaults

Often **60–120** min per section.

Impact example

90-min request when max **60** → **rejected**.

Code

`lib/rules.js` → `validateBooking`

CCM-10. Advance booking window

CCM-10. Advance booking window

Purpose

How far ahead members may book.

Formula

`dayDiff(todayClubDate, bookingDate) ≤ advance_days`.

Defaults

Often **7** days.

Impact example

7-day advance: booking 8 days out → **rejected**; day 7 → allowed.

Code

`lib/rules.js` → `validateBooking`

CCM-11. Same-day minimum lead time

CCM-11. Same-day minimum lead time

Purpose

Prevent booking slots starting too soon today.

Formula

If booking is today: `startMin − nowMin ≥ min_lead_min`.

Defaults

e.g. **60** minutes. Demo tenant uses **0** with fixed 9:00 AM teaching clock.

Impact example

At 2:00 PM with 60-min lead, 2:30 PM slot → **closed** on grid.

Code

`lib/rules.js`; `lib/demo.js` for demo override

CCM-12. Daily reservation limit

CCM-12. Daily reservation limit

Purpose

Max bookings per member per sport per day.

Formula

Count reservations where member is booker **or** accepted invitee; block if `count ≥ max_per_day`.

Defaults

Section `max_per_day` often **1–2**.

Impact example

`max_per_day = 1`, second booking same day → **rejected** with daily limit message.

Code

`lib/daily-limit.js`

CCM-13. Cancellation notice

CCM-13. Cancellation notice

Purpose

Require notice before canceling.

Formula

If `minutesSince(created) ≤ 30` → grace (allow). Else `minutesUntilStart ≥ min_cancel_min` (default **120**).

Defaults

30-min grace; 120-min notice.

Impact example

Cancel 1 hour before start with 2-hour rule → **blocked**.

Code

`lib/rules.js` → `validateCancellation`

CCM-14. Time slot generation

CCM-14. Time slot generation

Purpose

Rows on court availability grid.

Formula

`for (t = openMin; t + increment ≤ closeMin; t += increment)` slots.

Defaults

Open/close from section (e.g. 7:00–21:00).

Impact example

7:00–21:00 @ 30 min → **28** slot rows per court.

Code

`lib/routes/sections.js` ; client `public/js/court-grid.js`

CCM-15. Grid occupancy paint

CCM-15. Grid occupancy paint

Purpose

Show booked cells.

Formula

For each reservation, mark slots `[start, end)` stepping `incrementMin`.

Defaults

N/A.

Impact example

10:00–11:00 booking → **two** 30-min cells show as occupied.

Code

`public/js/court-grid.js`

CCM-16. Count bookable slots

CCM-16. Count bookable slots

Purpose

Max contiguous length member can book from a start time.

Formula

Walk forward until occupied, past close, or `(slots+1)×increment > max_length_min`.

Defaults

Uses section max length.

Impact example

Start 10:00, block at 10:30, max 60 min → only **1** slot (30 min) bookable.

Code

`public/js/court-grid.js` → `countBookableSlots`

CCM-17. Court group skill eligibility

CCM-17. Court group skill eligibility

Purpose

Restrict courts by self-reported skill band.

Formula

`ratingInRange(memberRating, skill_min, skill_max)` — boolean.

Defaults

Group labels like “3.0–4.0” are display only.

Impact example

Member **4.5** on 3.0–4.0 court group → **cannot book** those courts.

Code

`lib/booking-eligibility.js`, `lib/rating.js`

CCM-18. Member rating parse

CCM-18. Member rating parse

Purpose

Validate roster and court group ratings.

Formula

Member rating clamp **1.0–5.5**, round 2 decimals; court group range **0–8**, round 1 decimal.

Defaults

N/A.

Impact example

Enter **6.0** on roster → **rejected**; **3.25** stored as **3.25**.

Code

`lib/rating.js` → `parseMemberRating`

CCM-19. Event registration cap

CCM-19. Event registration cap

Purpose

Close registration when full.

Formula

`count(status in registered, walk_in) ≥ max_participants` → reject new registration.

Defaults

Per event `max_participants`.

Impact example

20 cap, 20 registered → next signup sees **event full**.

Code

`lib/routes/events/register.js`, `lib/club-events.js`

CCM-20. Invitation players needed

CCM-20. Invitation players needed

Purpose

Lineup confirmation target.

Formula

`playersNeeded = max(1, invitees + 1)`; `acceptedCount = 1 + acceptedInvitees`.

Defaults

N/A.

Impact example

3 invitees → need **4** confirmed (booker + 3).

Code

`lib/routes/reservations/invitations.js`

CCM-21. Lineup progress percent

CCM-21. Lineup progress percent

Purpose

Progress bar on reservation UI.

Formula

`min(100, round(accepted ÷ needed × 100))`.

Defaults

N/A.

Impact example

3 of 4 accepted → **75%** progress.

Code

`public/js/player-lineup.js`

CCM-22. Reminder target date

CCM-22. Reminder target date

Purpose

Which reservations get tomorrow's reminder email/SMS.

Formula

Per tenant TZ: `tomorrow = addDays(clubToday, 1)`; cron selects `res_date = tomorrow`.

Defaults

Daily cron ~1 PM UTC (see `vercel.json`).

Impact example

Monday cron sends reminders for **Tuesday** reservations in club timezone.

Code

`lib/reminders.js`

CCM-23. Stripe renewal overage line

CCM-23. Stripe renewal overage line

Purpose

Add **member** overage to draft Rankings subscription renewal invoices. CCM extra courts are billed via Stripe subscription **quantity** (court add-on price), not invoice line items — `overageInvoiceEligible` returns false for court-based plans.

Formula

Defaults

Only when `billing_reason = subscription_cycle`, status `draft`, tenant `active` or `past_due`, not enterprise, no custom pricing, and plan is **not** court-based.

Impact example

5 overage members on Rankings at $0.40 → **$2.00** invoice line. CCM with 8 courts → quantity **2** on court-overage Price (2 × $5).

Code

`lib/billing-invoice-overage.js`, `lib/billing-overage.js` → `overageInvoiceEligible`

Part 3 — Club Play scheduling

These affect pairings and court rotation, not ranking points.

CP-01. Repeat-minimization score

CP-01. Repeat-minimization score

Purpose

Pick random-mix round with fewest repeat partners/opponents.

Formula

`score = sum(partnerPairCounts) + sum(opponentPairCounts)` across round matchups; try up to **100** shuffles, pick lowest (0 = no repeats).

Defaults

N/A.

Impact example

Score **0** vs **4** → algorithm prefers assignment with **0** (fresh pairings).

Code

`lib/rankings/club-play/pairing.js` → `scoreRoundAssignment`

CP-02. Ladder seeding order

CP-02. Ladder seeding order

Purpose

Initial ladder positions from mini-league standings.

Formula

Sort teams by `totalPoints` DESC, then signup order for unranked.

Defaults

Uses CR total points from same series.

Impact example

Standings 103:50, 101:40, 102:30 → ladder order **[103, 101, 102, …]**.

Code

`lib/rankings/club-play/pairing.js` → `seedLadderOrder`

CP-03. Ladder upset swap

CP-03. Ladder upset swap

Purpose

Promote winner when lower rank beats higher rank.

Formula

If higher-index (better rank) team loses, swap winner and loser positions in `ladderOrder`.

Defaults

N/A.

Impact example

#101 loses to #102 → order flips → **102** now above **101** → next round matchups change.

Optional length

Session `target_rounds` stops auto-advance after that round (`tryAutoAdvanceRound` → `max_rounds`). Blank = unlimited until End session.

Code

`lib/rankings/club-play/pairing.js` → `ladderNextRound` ; `lib/rankings/club-play/round-flow.js` → `tryAutoAdvanceRound`

CP-04. King of the Court rotation

CP-04. King of the Court rotation

Purpose

Winner stays on court; challengers rotate.

Formula

Qualitative rotation — winner remains, losers move to queue/other courts; ties keep both teams.

Defaults

N/A.

Impact example

Court 1 winner plays new challengers from queue each round.

Optional length

Session `target_rounds` stops auto-advance after that round (`tryAutoAdvanceRound` → `max_rounds`).

Code

`lib/rankings/club-play/pairing.js` → `kotcNextRound` ; `lib/rankings/club-play/round-flow.js` → `tryAutoAdvanceRound`

CP-05. Round-robin schedule

CP-05. Round-robin schedule

Purpose

Fixed rotation of partnerships/opponents across rounds.

Formula

Precomputed schedule indices — no numeric scoring.

Defaults

N/A.

Impact example

8 players get 7 rounds with varied partners per schedule table.

Optional length

Pass `targetRounds` to schedule exactly N rounds; when omitted, default is `ceil(teams / 2)` (or `ceil(players / 2)` for random-mix doubles).

Code

`lib/rankings/club-play/pairing.js` → `roundRobinSchedule` / `randomMixSchedule`

CP-06. Swiss next-round pairing

CP-06. Swiss next-round pairing

Purpose

Each round, pair players by current session record so similar standings meet; fill all courts.

Formula

Rank active players by session wins DESC, then point differential DESC (same tiebreaks as session placement). Form doubles courts from nearby ranks (via `buildTeams`). Prefer assignments with lower `scoreRoundAssignment` (fewer repeat partners/opponents). After all court scores are in, auto-advance like ladder/KOTC.

Defaults

Doubles path; `courtCount × 4` players on court each round. Session `target_rounds` stops auto-advance (`max_rounds`). Performance-score weights still apply only after End session — Swiss does not change CR formulas.

Impact example

24 players / 6 courts → 6 games per round; after round 1, leaders play leaders (1v2, 3v4, … by standing).

Code

`lib/rankings/club-play/pairing.js` → `swissNextRound` / `sessionPlayerStandings`; `lib/rankings/adapters/swiss.js`; `lib/rankings/club-play/round-flow.js` → `tryAutoAdvanceRound`

Part 4 — Appendices

A. Master defaults (selected)

Pickleball first-time preset: weights 40/22/18/12/8, champion **3**, participation **1**, min events **3** (`lib/rankings/sport-templates.js`).

Domain	Setting	Default
CR	Performance weights	40 / 20 / 15 / 15 / 10 / 10
CR	Weekly max gain / loss	15 / 10
CR	Min events (leaderboard)	3
CR	Min events (club rating show)	1
CR	Rolling months	12
CR	Club rating feature	off
CR	Volunteer bonus	5
Billing	CCM	$40/mo · 6 courts included · +$5/court · unlimited members

Domain	Setting	Default
Billing	Rankings	$19/mo · 150 members · $0.40/member overage
Billing	Enterprise	Custom (contact sales; optional firm portfolio)
Billing	Trial	14 days
Membership	Platform fee	Default **0%** (`STRIPE_CONNECT_APPLICATION_FEE_PERCENT`)

B. Glossary

Term	Meaning
performance_points	Public bucket from session quality + champion bonus
participation_points	Flat per-event attendance
volunteer_points	From Vol flags
sportsmanship_points	Fast-confirm and similar
Total points	Sum of four buckets — leaderboard sort
Hidden Elo	Internal ~1500 scale; not shown to members
Club rating	1.0–5.5 display derived from Elo when enabled
Declared rating	Self-reported roster skill (1.0–5.5)
Performance score	0–100 blend per session (audit, not headline)

C. Code index

File	Calculations
`lib/rankings/engine/performance-score.js`	CR-02–09, CR-07, CR-30b
`lib/rankings/engine/skill-rating.js`	CR-12–15
`lib/rankings/engine/weekly-clamp.js`	CR-10–11
`lib/rankings/engine/recalculate.js`	CR-01, CR-20–21, CR-25, CR-27–28, CR-32
`lib/rankings/club-rating-scale.js`	CR-16–18
`lib/rankings/club-rating-display.js`	CR-19
`lib/billing-overage.js`	CCM-02–05, CCM-23
`lib/rules.js`	CCM-08–11, CCM-13
`lib/daily-limit.js`	CCM-12

File	Calculations
`public/js/court-grid.js`	CCM-15–16
`lib/rankings/club-play/pairing.js`	CP-01–06
`lib/rankings/club-play/aggregate.js`	CR-30
`lib/rankings/club-play/session-placement.js`	CR-30, CR-30b

D. Related documents

Club Court Manager™ powered by StoufferAI

Document	Purpose
CLUB-RANKINGS-SPECIFICATIONS.md	CR formulas deep dive
public/docs/ccm-user-manual.html	CCM workflows
public/docs/club-rankings-user-manual.html	CR workflows
DESKTOP-TEST-MANUAL.md	QA script

Club Court Manager™ powered by StoufferAI