CodeVix Labs

Built with logic. Delivered with precision.

TECHNICAL BLUEPRINT

Rewards & Loyalty Platform for Sporting Events

Prepared for: Johnathan Butler

Prepared by: Rupak Amin, Founder | CodeVix Labs

Date: March 2026

Version: 1.0

CONFIDENTIAL -- FOR DISCUSSION PURPOSES ONLY

Table of Contents

Chapter 1

Executive Summary & Market Opportunity

1.1 Project Overview

This document presents the complete technical architecture for a Rewards & Loyalty Platform designed specifically for sporting events. The platform enables event organizers, sports teams, and venues to create engaging loyalty programs that reward fans for attendance, purchases, social engagement, and participation -- driving repeat visits, higher spend per fan, and deeper brand loyalty.

The platform is built as a modern SaaS application using a production-grade tech stack: Next.js for the frontend, PostgreSQL for the database, and a robust API layer -- all designed with security, scalability, and QA built in from day one.

1.2 Market Opportunity

MARKET SIZE: The global fan engagement market is valued at $16.2B (2024) and projected to reach $66.7B by 2034 -- growing at 15.2% CAGR. The US market alone was $5.65B in 2024, projected to reach $21.5B by 2034.

Key market drivers:

- 58% of US sports fans expect real-time statistics, replays, and AR overlays during live events. This rises to 70% among Gen Z and millennials.
- 78% of buyers prefer gamified experiences in 2025 -- loyalty programs with game mechanics outperform traditional programs by 3-5x.
- The US loyalty management market is projected to reach $44.73B by 2029, with 90%+ of companies deploying loyalty solutions.
- Sports event market compound annual growth rate of 10%+ through 2033, driven by live event demand and digital transformation.

1.3 Business Model

The platform supports multiple revenue streams:

1. SaaS Subscription: Monthly/annual fees per venue or team ($500-$5,000/mo depending on tier)
2. Transaction Fees: 1-3% on in-app purchases and reward redemptions
3. Sponsor Integration: Branded rewards and sponsored challenges (revenue share)
4. Data Insights: Premium analytics packages for venues and teams
5. White-Label Licensing: Custom-branded versions for sports leagues

Chapter 2

Product Vision & Target Users

2.1 Vision Statement

To become the operating system for fan loyalty in sports -- turning passive spectators into active, engaged, repeat visitors who feel recognized and rewarded for their fandom.

2.2 Target Users

Primary Users (Fans)

- Sports event attendees (college, professional, semi-pro, amateur)
- Season ticket holders looking for additional perks
- Casual fans who attend 1-5 events per year (conversion targets)
- Fantasy sports enthusiasts and sports bettors
- Families with children attending sporting events

Business Users (B2B Customers)

- Sports teams and franchises (MLB, NFL, NBA, NHL, MLS, college)
- Event venues and stadiums
- Sports leagues and governing bodies
- Event organizers (marathons, tournaments, esports)
- Sponsors and brand partners seeking fan engagement data

2.3 User Personas

Persona 1: 'Game Day Gary' -- Casual Fan

Age: 28-45. Attends 3-8 games/year. Wants rewards for showing up. Buys food/merch at events. Uses mobile phone constantly during events. Motivated by: free upgrades, food discounts, exclusive merch.

Persona 2: 'Season Ticket Sarah' -- Loyal Fan

Age: 30-55. Season ticket holder. Attends 20-40 events/year. Brings friends and family. Motivated by: VIP access, meet-and-greets, loyalty recognition, exclusive experiences. High lifetime value.

Persona 3: 'Venue VP Victor' -- Business Customer

Age: 35-50. VP of Marketing or Fan Experience at a sports team. Needs: attendance data, spend analytics, sponsor ROI tracking, easy campaign management without developer help. Budget: $2K-$10K/month.

Chapter 3

Competitive Landscape Analysis

3.1 Key Competitors

FanMaker (fanmaker.com)

Founded 2007, Minneapolis. The market leader in collegiate and professional sports loyalty. Features: mobile wallet loyalty pass, QR scanning, punch cards for attendance streaks, youth membership clubs, push notifications, Apple/Google wallet integration. Serves NCAA, MLB, NBA teams. Strength: proven technology, large client base. Weakness: legacy architecture, no modern gamification, limited API flexibility.

Tradable Bits (tradablebits.com)

Fan loyalty portal with quests/missions, points system, reward redemption. Integration with social media. Developer API available. Strength: quest-based engagement. Weakness: complex setup, less intuitive UX.

bFAN Sports (bfansports.com)

Fan experience platform for sports organizations. Mobile-first, content delivery, push notifications, fan profiles. Strength: content + engagement combined. Weakness: less focus on rewards/points.

Qualifio (qualifio.com)

Interactive marketing campaigns for sports teams. Quizzes, polls, contests, data collection. Strength: engagement mechanics. Weakness: not a full loyalty platform.

Sportian (sportian.com)

Connected fan engagement and growth solutions. Data platform + engagement tools. Strength: data analytics. Weakness: enterprise-only pricing.

3.2 Competitive Advantage -- Where We Win

- Modern tech stack (Next.js/React) vs legacy platforms -- faster iteration, better UX
- QA-first approach -- every feature tested before deployment, zero downtime during events
- API-first architecture -- easy integration with any ticketing, POS, or CRM system
- Real-time gamification engine -- not just points, but challenges, streaks, leaderboards, social competitions
- Lower price point for smaller venues and semi-pro teams (competitors focus on enterprise)
- White-label ready from day one -- teams can brand it as their own app
- Mobile-first PWA -- no app store approval delays, instant updates

Chapter 4

Core Features & Functional Requirements

4.1 Fan-Facing Features

F1: User Registration & Profiles

- Social login (Google, Apple, Facebook) + email/password
- Fan profile: name, avatar, favorite teams, attendance history
- Points balance, tier status, reward history visible on dashboard
- QR code unique to each fan (for check-in scanning)

F2: Points Engine

- Earn points for: event attendance, purchases, social shares, referrals, challenges
- Points multiplier events (2x points on opening day, playoff games)
- Points expiration rules (configurable: 6 months, 1 year, never)
- Real-time balance updates after every transaction
- Points transfer between fans (optional, admin-configurable)

F3: Tier System

- Configurable tiers (e.g., Rookie, All-Star, MVP, Hall of Fame)
- Tier progression based on cumulative points or attendance count
- Tier-specific perks: priority seating, exclusive merch, early access
- Tier reset options: annual, seasonal, or cumulative lifetime
- Visual progress bar showing distance to next tier

F4: Rewards Catalog

- Digital rewards: discount codes, free food/drink vouchers, parking passes
- Physical rewards: exclusive merchandise, signed memorabilia
- Experience rewards: meet-and-greets, locker room tours, sideline passes
- Partner rewards: restaurant discounts, rideshare credits, hotel deals
- Limited-time rewards (flash drops during events)
- Reward inventory management with quantity limits

F5: QR Check-In & Geolocation

- QR code scan at venue gates for attendance verification
- GPS geofencing as backup verification (fan must be within venue radius)
- Beacon integration for zone-specific rewards (concourse, VIP area, team store)
- Automatic points credit upon check-in (no manual action needed)

F6: Gamification & Challenges

- Daily/weekly/seasonal challenges (e.g., 'Attend 3 games this month')
- Attendance streaks with multiplying bonuses
- Prediction games (guess the score, MVP, first goal)
- Trivia quizzes about team history and stats
- Social challenges (bring a friend, share on social media)
- Leaderboards: global, team-specific, friend groups
- Achievement badges with visual collection display

F7: Social Features

- Friend lists and group leaderboards
- Share achievements to social media (Instagram, X, Facebook)
- Fan-to-fan gift giving (send points or rewards)
- Event check-in feed (see who else is at the game)
- Group challenges (compete as a friend group)

F8: Push Notifications & Messaging

- Game-day reminders with personalized content
- Reward expiration alerts
- Flash reward drops during live events
- Tier upgrade congratulations
- Challenge completion alerts
- Segmented messaging by tier, location, purchase history

4.2 Business-Facing Features (Admin)

A1: Campaign Manager

- Create/edit/archive loyalty campaigns with start/end dates
- Configure point multipliers for specific events or periods
- Set up flash reward drops with inventory limits
- Schedule automated campaigns (weekly challenges, seasonal resets)
- No-code campaign builder -- drag-and-drop interface

A2: Analytics Dashboard

- Real-time attendance tracking and check-in rates
- Points economy health: points issued vs redeemed ratio
- Fan segmentation: by tier, spend, frequency, engagement
- Revenue attribution: tracked purchases linked to loyalty campaigns
- Sponsor ROI: impressions, clicks, redemptions per sponsored reward
- Export reports to CSV/PDF

A3: Reward Management

- CRUD for rewards catalog with images, descriptions, point costs
- Inventory tracking with low-stock alerts
- Reward performance metrics (which rewards drive most engagement)
- Partner reward management with co-branding

A4: User Management

- View/search/filter all registered fans
- Manual point adjustments with audit log
- Tier override capability for VIPs
- Ban/suspend accounts for abuse
- Bulk import fans from existing CRM/ticketing system

Chapter 5

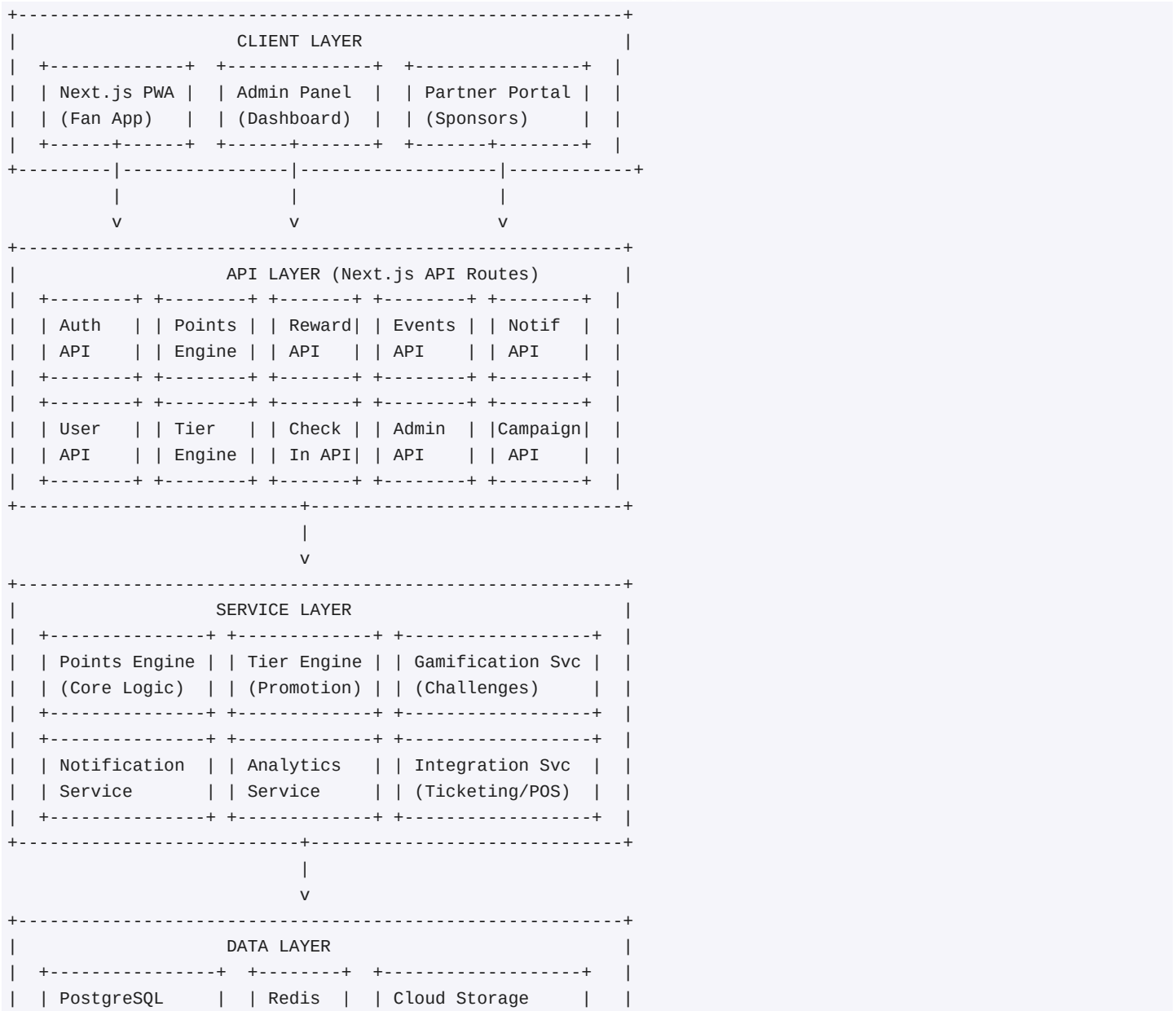
System Architecture Overview

5.1 High-Level Architecture

The platform follows a modern monolithic-first architecture using Next.js, designed for rapid development with a clear path to microservices extraction as the platform scales. This approach minimizes initial complexity while maintaining clean separation of concerns.

ARCHITECTURE PRINCIPLE: Start monolith, extract microservices only when a specific service becomes a bottleneck. Premature microservices kill startups.

5.2 Architecture Diagram (Text Representation)



```
|  | (via Prisma)   |  | Cache  |  | (Images/Assets)   |   |
|  | Primary DB     |  | Queue  |  | Cloudinary/S3     |   |
|  +----------------+  +--------+  +-------------------+   |
+-----------------------------------------------------------+
```

5.3 Request Flow Example: Fan Checks In at Event

1. Fan opens app at stadium, taps 'Check In'
2. App sends GPS coordinates + fan ID to Check-In API
3. Check-In API validates: (a) fan exists, (b) event is active, (c) GPS within geofence
4. Points Engine calculates points: base (50) + streak bonus (25) + multiplier (2x = 150 total)
5. Tier Engine checks if new point total triggers tier upgrade
6. Transaction recorded in PostgreSQL (atomic, with audit trail)
7. Redis cache updated with new point balance (for instant UI updates)
8. Notification Service sends push: 'Welcome to tonight's game! +150 points earned!'
9. If tier upgraded: additional push + confetti animation in app
10. Analytics Service logs the check-in event for real-time dashboards

5.4 White-Label & Multi-Tenant Architecture

The platform is designed to serve multiple sports teams, venues, and leagues from a single codebase. Each tenant (team/venue) gets a fully branded experience while sharing the same infrastructure -- this is how the business scales from 1 client to 100.

Multi-Tenant Data Model

```
// Every major entity includes a tenantId
model Tenant {
  id          String   @id @default(cuid())
  name        String   // 'Dallas Cowboys', 'AT&T Stadium'
  slug        String   @unique  // 'cowboys', 'att-stadium'
  domain      String?  // custom domain: rewards.dallascowboys.com
  logo        String   // tenant logo URL
  primaryColor String  // '#003594' (Cowboys navy)
  secondaryColor String // '#869397' (Cowboys silver)
  config      Json     // tier names, point rules, feature flags
  plan        Plan     @default(STARTER)
  events      Event[]
  users       User[]   // fans registered under this tenant
}

enum Plan { STARTER, PRO, ENTERPRISE }
```

Branding & Customization Per Tenant

- Custom colors, logos, and fonts applied via CSS variables (zero code changes)
- Custom tier names: 'Rookie/All-Star/MVP' or 'Bronze/Silver/Gold/Platinum' -- configurable per tenant
- Custom point names: 'FanPoints', 'CowboyCoins', 'TigerTokens' -- whatever the team wants
- Custom domain support: rewards.dallascowboys.com -> same app, different branding
- Custom reward catalog per tenant (each team manages their own rewards)

- Feature flags per tenant: enable/disable prediction games, social features, etc.

Tenant Isolation & Security

- Row-level security: every database query automatically filtered by tenantId via Prisma middleware
- Fan data never crosses tenant boundaries (Cowboys fans can't see Lakers data)
- Admin users scoped to their tenant (team admin can only manage their own team)
- Super-admin role for platform-wide management (Johnathan's team)
- Separate Redis namespaces per tenant for cache isolation

How This Scales the Business

- Onboard a new team in minutes: create tenant, upload logo, set colors, invite admin
- Single codebase to maintain: bug fixes and features roll out to ALL tenants simultaneously
- Per-tenant billing: Stripe subscription tied to tenant, metered by fan count or event count
- League deals: one contract with NFL/NCAA = 30+ tenants onboarded at once

BUSINESS MULTIPLIER: Without multi-tenancy, every new client = a new deployment. With it, every new client = a new row in the database. This is the difference between a service company and a SaaS company.

Chapter 6

Technology Stack

6.1 Full Stack Overview

Layer	Technology	Why
Frontend	Next.js 14+ (App Router)	SSR, RSC, SEO, fast navigation
UI Library	React 19	Component-based, massive ecosystem
Styling	Tailwind CSS v4	Rapid UI development, consistent design
Language	TypeScript (Strict)	Type safety, fewer bugs, better DX
Database	PostgreSQL (Neon)	ACID transactions, JSON support, scalable
ORM	Prisma	Type-safe queries, migrations, schema mgmt
Auth	NextAuth v5	OAuth, JWT, session mgmt, RBAC
Validation	Zod	Runtime schema validation, API safety
Cache/Queue	Redis (Upstash)	Points cache, leaderboards, job queues
File Storage	Cloudinary	Image optimization, CDN delivery
Payments	Stripe	In-app purchases, subscriptions, payouts
Push Notif.	Firebase Cloud Msg	iOS + Android + Web push notifications
Email	Resend	Transactional emails, templates
Hosting	Vercel	Edge functions, auto-scaling, CI/CD
Monitoring	Sentry	Error tracking, performance monitoring
Analytics	PostHog	Product analytics, feature flags, A/B tests

6.2 Why This Stack?

- Next.js + Vercel = zero-config deployment, automatic scaling for game-day traffic spikes (10-100x normal traffic during events)
- PostgreSQL = ACID transactions critical for points economy (no double-spend, no phantom points)
- Redis = sub-millisecond leaderboard queries and real-time point balance display
- TypeScript Strict = catches bugs at compile time, not in production during a live event
- Prisma = type-safe database queries that prevent SQL injection and data integrity issues
- Zod = server-side validation on every API endpoint -- no trusting client-side data

Chapter 7

Database Schema & Data Models

7.1 Core Entities

```
// Prisma Schema (simplified)

model User {
  id            String    @id @default(cuid())
  email         String    @unique
  name          String
  avatar        String?
  role          Role      @default(FAN)
  tier          Tier      @default(ROOKIE)
  totalPoints   Int       @default(0)
  currentPoints Int       @default(0)
  streakCount   Int       @default(0)
  createdAt     DateTime  @default(now())
  // Relations
  transactions  PointTransaction[]
  checkIns      CheckIn[]
  rewards       UserReward[]
  badges        UserBadge[]
  challengeProgress ChallengeProgress[]
}

enum Role { FAN, ADMIN, VENUE_MANAGER, SPONSOR }
enum Tier { ROOKIE, ALL_STAR, MVP, HALL_OF_FAME }
```

```
model PointTransaction {
  id          String   @id @default(cuid())
  userId      String
  amount      Int      // positive = earn, negative = spend
  type        TxType   // CHECKIN, PURCHASE, CHALLENGE, REDEEM, etc
  source      String   // event ID, challenge ID, purchase ID
  multiplier  Float    @default(1.0)
  balance     Int      // running balance after this transaction
  createdAt   DateTime @default(now())
  user        User     @relation(fields: [userId], references: [id])
  event       Event?   @relation(fields: [eventId], references: [id])
  eventId     String?
}

enum TxType {
  CHECKIN, PURCHASE, CHALLENGE, REFERRAL,
  SOCIAL_SHARE, PREDICTION, QUIZ, STREAK_BONUS,
  REDEEM, TRANSFER_OUT, TRANSFER_IN, ADMIN_ADJUST, EXPIRY
}
```

```
model Event {
  id            String   @id @default(cuid())
  name          String
  venue         Venue    @relation(fields: [venueId], references: [id])
  venueId       String
  date          DateTime
  pointsBase    Int      @default(50)
  multiplier    Float    @default(1.0)
  geofenceLat   Float
  geofenceLng   Float
  geofenceRadius Int     @default(500) // meters
  isActive      Boolean  @default(true)
  checkIns      CheckIn[]
  transactions  PointTransaction[]
}

model Venue {
  id        String  @id @default(cuid())
  name      String
  address   String
  city      String
  state     String
  lat       Float
  lng       Float
  capacity  Int
  events    Event[]
  team      Team?   @relation(fields: [teamId], references: [id])
  teamId    String?
}
```

```
model Reward {
  id           String   @id @default(cuid())
  name         String
  description  String
  image        String?
  pointsCost   Int
  category     RewardCategory
  minTier      Tier     @default(ROOKIE)
  quantity     Int?     // null = unlimited
  claimed      Int      @default(0)
  expiresAt    DateTime?
  isActive     Boolean  @default(true)
  sponsorId    String?
  userRewards  UserReward[]
}

enum RewardCategory {
  FOOD_DRINK, MERCHANDISE, EXPERIENCE,
  UPGRADE, PARTNER, DIGITAL, LIMITED_EDITION
}

model CheckIn {
  id        String   @id @default(cuid())
  userId    String
  eventId   String
  method    CheckInMethod
  lat       Float?
  lng       Float?
  createdAt DateTime @default(now())
  user      User     @relation(fields: [userId], references: [id])
```

```
  event     Event    @relation(fields: [eventId], references: [id])
  @@unique([userId, eventId]) // one check-in per user per event
}

enum CheckInMethod { QR_SCAN, GEOLOCATION, BEACON, MANUAL }
```

```
model Challenge {
  id           String   @id @default(cuid())
  title        String
  description  String
  type         ChallengeType
  target       Int      // e.g., attend 3 games, spend $50
  pointsReward Int
  badgeId      String?
  startsAt     DateTime
  endsAt       DateTime
  isActive     Boolean  @default(true)
  progress     ChallengeProgress[]
}

enum ChallengeType {
  ATTENDANCE_COUNT, ATTENDANCE_STREAK, PURCHASE_AMOUNT,
  SOCIAL_SHARE_COUNT, REFERRAL_COUNT, PREDICTION_CORRECT,
  QUIZ_SCORE, CUSTOM
}

model ChallengeProgress {
  id          String    @id @default(cuid())
  userId      String
  challengeId String
  current     Int       @default(0)
  completed   Boolean   @default(false)
  completedAt DateTime?
  user        User      @relation(fields: [userId], references: [id])
  challenge   Challenge @relation(fields: [challengeId], references: [id])
  @@unique([userId, challengeId])
}
```

7.2 Entity Relationship Summary

```
User ---< PointTransaction >--- Event
User ---< CheckIn >--- Event
User ---< UserReward >--- Reward
User ---< UserBadge >--- Badge
User ---< ChallengeProgress >--- Challenge
Event >--- Venue >--- Team
Reward --- Sponsor (optional)
Campaign ---< CampaignReward >--- Reward

Key: ---< means one-to-many
     >--- means many-to-one
```

Chapter 8

API Design & Endpoints

8.1 API Architecture

All APIs are implemented as Next.js API routes (App Router) with Zod validation on every endpoint. Authentication via NextAuth JWT tokens. Rate limiting via Redis. All responses follow a consistent JSON format.

```
// Standard API Response Format
{
  "success": true | false,
  "data": { ... } | null,
  "error": { "code": "string", "message": "string" } | null,
  "meta": { "page": 1, "limit": 20, "total": 150 } // for paginated
}
```

8.2 Core API Endpoints

Authentication

```
POST   /api/auth/register     -- Register new fan account
POST   /api/auth/login        -- Login (email/password or OAuth)
POST   /api/auth/logout       -- Logout and invalidate session
GET    /api/auth/me           -- Get current user profile
PATCH  /api/auth/me           -- Update profile
```

Points

```
GET    /api/points/balance    -- Get current points balance
GET    /api/points/history    -- Get transaction history (paginated)
POST   /api/points/transfer   -- Transfer points to another fan
GET    /api/points/leaderboard -- Get leaderboard (global/team/friends)
```

Check-In

```
POST   /api/checkin           -- Check in to event (QR or GPS)
GET    /api/checkin/status     -- Check if already checked in
GET    /api/checkin/history    -- Past check-ins
```

Rewards

```
GET    /api/rewards           -- Browse reward catalog (filtered)
GET    /api/rewards/:id       -- Reward details
POST   /api/rewards/:id/redeem -- Redeem a reward
GET    /api/rewards/my        -- My redeemed rewards + status
```

Challenges & Gamification

```
GET    /api/challenges        -- Active challenges
GET    /api/challenges/:id    -- Challenge details + my progress
GET    /api/badges            -- All badges (earned + locked)
GET    /api/leaderboard       -- Global/team/friends rankings
```

Events

```
GET    /api/events            -- Upcoming events
GET    /api/events/:id        -- Event details + check-in count
GET    /api/events/:id/feed   -- Who's checked in (social feed)
```

Admin APIs (Protected)

```
// Campaigns
POST   /api/admin/campaigns          -- Create campaign
PATCH  /api/admin/campaigns/:id       -- Update campaign
DELETE /api/admin/campaigns/:id       -- Archive campaign

// Rewards Management
POST   /api/admin/rewards             -- Create reward
PATCH  /api/admin/rewards/:id          -- Update reward
DELETE /api/admin/rewards/:id          -- Deactivate reward

// User Management
GET    /api/admin/users               -- List/search users
PATCH  /api/admin/users/:id/points     -- Manual point adjustment
PATCH  /api/admin/users/:id/tier        -- Tier override

// Analytics
GET    /api/admin/analytics/overview    -- Dashboard metrics
GET    /api/admin/analytics/events/:id  -- Event-specific metrics
GET    /api/admin/analytics/export       -- CSV/PDF export
```

Chapter 9

Points Engine & Gamification Logic

9.1 Points Earning Rules

```
// Points Earning Matrix
ACTION              | BASE POINTS | NOTES
--------------------+-------------+--------------------------
Event Check-In      | 50          | +streak bonus
Purchase (per $10)  | 10          | Tracked via POS integration
Social Share         | 15          | Verified via API callback
Refer a Friend      | 100         | Credited when friend joins
Challenge Complete  | 25-500      | Varies by difficulty
Quiz Correct Answer | 10          | Per correct answer
Prediction Win      | 50          | Score/MVP/first goal
Profile Complete    | 25          | One-time bonus
Birthday            | 100         | Annual auto-credit
```

9.2 Streak Bonus System

```
// Attendance Streak Multiplier
CONSECUTIVE EVENTS | BONUS MULTIPLIER
-------------------+-----------------
1 (no streak)      | 1.0x (base only)
2 in a row         | 1.25x
3 in a row         | 1.5x
5 in a row         | 2.0x
10 in a row        | 3.0x
20+ in a row       | 5.0x (LEGENDARY)

Example: 50 base pts x 2.0 multiplier x 2.0 event multiplier = 200 pts
```

9.3 Tier Progression

```
TIER           | POINTS REQUIRED | PERKS
---------------+-----------------+-----------------------------------
Rookie         | 0               | Base rewards catalog access
All-Star       | 500             | +10% bonus points, priority queue
MVP            | 2,000           | +25% bonus, exclusive rewards,
               |                 |  early access to tickets
Hall of Fame   | 10,000          | +50% bonus, VIP experiences,
               |                 |  meet-and-greet, sideline pass
```

9.4 Points Engine Flow (Pseudocode)

```
async function processPointsEarning(userId, action, context) {
  // 1. Calculate base points
  const basePoints = POINTS_MATRIX[action.type];

  // 2. Apply event multiplier (if during special event)
  const eventMultiplier = context.event?.multiplier || 1.0;

  // 3. Apply streak bonus (if check-in)
```

```
  const streakMultiplier = calcStreakBonus(userId);

  // 4. Apply tier bonus
  const tierBonus = TIER_BONUS[user.tier]; // 1.0, 1.1, 1.25, 1.5

  // 5. Calculate final points
  const finalPoints = Math.floor(
    basePoints * eventMultiplier * streakMultiplier * tierBonus
  );

  // 6. Atomic transaction (Prisma)
  await prisma.$transaction([
    prisma.pointTransaction.create({ data: { ... } }),
    prisma.user.update({ data: {
      totalPoints: { increment: finalPoints },
      currentPoints: { increment: finalPoints },
    }}),
  ]);

  // 7. Check tier upgrade
  await checkTierUpgrade(userId);

  // 8. Update challenge progress
  await updateChallengeProgress(userId, action);

  // 9. Invalidate Redis cache
  await redis.del(`points:${userId}`);

  // 10. Send notification
  await notify(userId, `+${finalPoints} points earned!`);
}
```

Chapter 10

Authentication & Security Architecture

10.1 Authentication Flow

- NextAuth v5 with JWT strategy (stateless, scalable)
- OAuth providers: Google, Apple, Facebook (one-tap login)
- Email/password with bcrypt hashing (12 salt rounds)
- Email verification required before point redemption
- Session tokens: HttpOnly, Secure, SameSite=Strict cookies

10.2 Role-Based Access Control (RBAC)

```
ROLE           | PERMISSIONS
---------------+-----------------------------------------
FAN            | View profile, earn/redeem points, check in
VENUE_MANAGER  | Manage events, view venue analytics
SPONSOR        | Manage sponsored rewards, view campaign ROI
ADMIN          | Full access: users, campaigns, analytics,
               | manual adjustments, system configuration
```

10.3 Security Measures

- Server-side Zod validation on EVERY API endpoint -- no client-side trust
- SQL injection prevention via Prisma parameterized queries
- Rate limiting: 100 req/min per user, 10 req/min for auth endpoints
- CSRF protection with SameSite=Strict cookies
- Content-Security-Policy headers to prevent XSS
- Points transaction atomicity via Prisma $transaction (no double-spend)
- Audit log for all admin actions (who changed what, when)
- QR code rotation: unique per event, expires after event ends
- Geofence anti-spoofing: GPS + IP geolocation cross-validation
- Points economy monitoring: automated alerts for anomalous earning patterns

10.4 Data Privacy & Compliance

A rewards platform collecting GPS location, purchase history, personal profiles, and fan behavior data must comply with major privacy regulations. This is non-negotiable for enterprise sales to sports teams and venues.

CCPA (California Consumer Privacy Act)

- Right to Know: fans can request all data collected about them
- Right to Delete: fans can request complete data erasure
- Right to Opt-Out: fans can opt out of data sale/sharing
- Privacy policy link required on every screen that collects data

- Applies to any business with California users (which includes most US sports teams)

GDPR Readiness (For Future International Expansion)

- Explicit consent required before collecting location data
- Data minimization: only collect what's needed for each feature
- Data portability: fans can export their data in machine-readable format (JSON)
- Right to be forgotten: complete data erasure on request
- Cookie consent banner with granular opt-in/opt-out

COPPA (Children's Online Privacy Protection)

- Sporting events attract families with children under 13
- Age gate at registration: users must confirm age 13+
- Family accounts: parent creates child sub-accounts with restricted data collection
- No targeted advertising or behavioral tracking for child accounts
- Parental consent mechanism for child account creation

Data Retention & Security

- Location data: stored only as check-in records (lat/lng at time of check-in), NOT continuous tracking
- Purchase data: retained for 2 years for points calculation, then anonymized
- Account deletion: 30-day grace period, then permanent erasure from all systems including backups
- Encryption at rest (AES-256) and in transit (TLS 1.3)
- Annual security audit and penetration testing
- Data breach notification within 72 hours (GDPR requirement, good practice everywhere)

COMPLIANCE IS A SELLING POINT: When pitching to sports teams, CCPA/COPPA compliance is a checkbox they MUST check. Competitors who skip this lose enterprise deals.

Chapter 11

Third-Party Integrations

11.1 Ticketing Systems

- Ticketmaster: Sync ticket purchases with attendance points
- Eventbrite: Event creation and registration sync
- AXS / SeatGeek: Alternative ticketing platform support
- Custom API: webhook for any ticketing system to push purchase events

11.2 Point-of-Sale (POS)

- Square: In-venue purchase tracking, automatic points for food/merch
- Shopify: Online team store integration
- Custom POS webhook: any POS can push transactions via API

11.3 Payment Processing

- Stripe: In-app purchases, premium tier subscriptions, sponsor payments
- Stripe Connect: Revenue splitting with venues and partners

11.4 Communication

- Firebase Cloud Messaging: Push notifications (iOS, Android, Web)
- Resend: Transactional emails (welcome, reward confirmation, tier upgrade)
- Twilio (optional): SMS notifications for high-value rewards

11.5 Analytics & Marketing

- PostHog: Product analytics, user behavior tracking, A/B testing
- Google Analytics 4: Traffic and acquisition metrics
- CRM export: Salesforce, HubSpot via CSV or API

11.6 Sponsor & Brand Portal

Sponsors are a primary revenue stream for sporting event platforms. The sponsor portal allows brand partners to create, manage, and measure branded rewards and engagement campaigns without developer involvement.

Sponsor Dashboard Features

- Self-service campaign creation: sponsors create branded rewards (e.g., 'Coca-Cola Halftime Challenge')
- Custom reward branding: sponsor logo, colors, and messaging on reward cards
- Budget management: set campaign spend limits, cost-per-redemption caps
- Audience targeting: select fan segments by tier, purchase history, event frequency, location
- Real-time ROI dashboard: impressions, clicks, redemptions, cost-per-engagement

- A/B testing: run two versions of a sponsored reward to see which performs better

Sponsor Integration Points

- Sponsored check-in bonuses: 'Check in at Gate 4 (Pepsi Gate) for 2x points'
- Branded challenges: 'Complete the Nike Fan Challenge -- buy any Nike item + attend 3 games'
- Sponsored leaderboard: 'This week's leaderboard presented by State Farm'
- Sponsored predictions: 'Make your game prediction, powered by DraftKings'
- In-app ad slots: non-intrusive banner on reward catalog and leaderboard pages

Sponsor API Endpoints

```
// Sponsor Portal APIs (SPONSOR role required)
POST   /api/sponsor/campaigns          -- Create sponsored campaign
GET    /api/sponsor/campaigns           -- List my campaigns
PATCH  /api/sponsor/campaigns/:id       -- Update campaign
GET    /api/sponsor/analytics           -- ROI dashboard data
GET    /api/sponsor/analytics/:id       -- Per-campaign metrics
POST   /api/sponsor/rewards             -- Create branded reward
GET    /api/sponsor/audience-segments    -- Available targeting segments
```

Sponsor Revenue Model

- Monthly subscription ($500-$2,000/mo for portal access + campaign tools)
- Performance fees: $0.10-$0.50 per reward redemption (revenue share with platform)
- Premium placement: featured spots on reward catalog ($200-$1,000/event)
- Data insights package: anonymized fan behavior data ($500-$2,000/quarter)

SPONSOR REVENUE POTENTIAL: A single MLB team with 81 home games and 3-5 sponsors per game = $50K-$200K/year in sponsor revenue alone. This is recurring revenue that scales with every new venue onboarded.

11.7 Social Media

- Meta Graph API: Verified social sharing for bonus points
- X/Twitter API: Share achievements, auto-post milestones
- Instagram: Share check-in stories with branded frames

Chapter 12

Mobile Strategy

12.1 Progressive Web App (PWA) -- Phase 1

For MVP launch, the platform ships as a PWA (Progressive Web App). This approach offers significant advantages for a startup:

- No app store approval delays -- deploy updates instantly
- Single codebase for iOS + Android + Desktop
- Installable on home screen (looks like a native app)
- Offline mode for basic functionality (cached rewards, profile)
- Push notifications via service worker
- Camera access for QR scanning
- GPS/geolocation for check-in
- Development cost: 60-70% less than native apps

12.2 Offline & Poor Connectivity Mode

Stadiums with 30,000-80,000 fans create severe network congestion. Cell service degrades or fails entirely during peak moments (kickoff, halftime, big plays). The platform MUST work in these conditions -- this is the #1 UX failure point competitors struggle with.

Offline-First Architecture

- Service Worker caches: fan profile, points balance, QR code, active rewards, and event info on app load
- IndexedDB local storage: pending check-ins and actions queued locally when offline
- Background Sync API: queued actions automatically sync when connectivity returns
- Optimistic UI: show 'Check-in recorded!' immediately, sync to server in background

QR Check-In Without Internet

- Fan's unique QR code is pre-generated and cached locally (no server call needed to display it)
- Venue scanner validates QR against a locally-synced attendee list (downloaded 1 hour before gates open)
- Scanner queues check-in records locally, batch-syncs to server every 30 seconds when connected
- Fallback: manual check-in by venue staff using fan's name/email lookup from cached list

Connectivity Status UX

- Subtle indicator showing connection status (green/yellow/red)
- Yellow = degraded: core features work, social features delayed
- Red = offline: check-in, QR display, cached rewards work; leaderboards and social feed paused
- Auto-resume: when connection returns, all pending actions sync with conflict resolution

STADIUM REALITY: AT&T Stadium (Dallas Cowboys) holds 80,000+ fans. When 40,000 fans open their phones

at halftime, most apps crash. Ours won't -- because it works offline first.

12.3 Apple Wallet & Google Wallet Loyalty Pass

This is one of the HIGHEST-IMPACT features for user adoption. Instead of asking fans to download an app, they simply add a loyalty pass to their existing Apple Wallet or Google Wallet. FanMaker's most successful feature is their wallet pass -- it removes the biggest barrier to adoption: app downloads.

How It Works

- Fan registers on the web app (or at a venue kiosk)
- System generates a personalized loyalty pass (Apple .pkpass / Google JWT)
- Fan taps 'Add to Wallet' -- pass appears alongside credit cards and boarding passes
- Pass displays: fan name, tier, QR code for check-in, current points balance
- Pass auto-updates: points balance refreshes in real-time via push updates
- NFC tap-in at supported venues (Phase 2+): fan taps phone at gate, earns points automatically

Why This Matters

- NO APP DOWNLOAD REQUIRED -- eliminates the #1 barrier to adoption
- Always visible in the wallet app fans already use daily
- Lock-screen notifications: 'You just earned 150 points!' when check-in registers
- Location-triggered: pass surfaces automatically when fan arrives at venue (geofence)
- Works offline: QR code is cached on the pass, scannable without internet

Technical Implementation

- Apple Wallet: PassKit framework, .pkpass bundle with JSON payload + signing certificate
- Google Wallet: Google Wallet API, JWT-based pass with real-time updates via callback URL
- Integration: PassKit.com SaaS ($0.01-$0.05 per pass) OR self-hosted with Apple Developer cert
- Updates: server pushes point balance changes to wallet via APNs (Apple) or Google API

ADOPTION MULTIPLIER: Apps have ~15% install rate from landing pages. Wallet passes have ~60%+ add rate. This single feature can 4x your active user base compared to app-only competitors.

12.4 React Native -- Phase 2 (If Needed)

If specific native features are required (NFC, Apple Wallet pass, advanced Bluetooth beacons), a React Native app shares 80%+ of business logic with the Next.js codebase:

- Shared TypeScript types, Zod schemas, and API client
- React Native for iOS + Android with native modules
- Apple Wallet + Google Wallet pass generation
- Bluetooth Low Energy (BLE) for beacon zone detection

Chapter 13

Admin Dashboard & Analytics

13.1 Dashboard Overview

The admin dashboard is built into the same Next.js application, protected by role-based middleware. No separate admin application needed.

Key Dashboard Widgets

- Total active fans (with daily/weekly/monthly trends)
- Points economy health: issued vs redeemed ratio (target: 60-70% redemption)
- Today's check-ins (real-time during events)
- Top rewards by redemption count
- Revenue attribution: spend linked to loyalty campaigns
- Fan tier distribution (pie chart)
- Engagement score: DAU/MAU ratio

13.2 Campaign Builder (No-Code)

Venue managers can create campaigns without developer help:

1. Select campaign type (points multiplier, challenge, flash reward)
2. Set date range and target audience (by tier, purchase history, location)
3. Configure rewards and point values
4. Preview and launch
5. Monitor real-time performance

13.3 Fan Segmentation & Personalization Engine

Sports teams don't just want aggregate data -- they want to TARGET specific fan groups with tailored campaigns. This is the feature that justifies premium pricing and is what PwC identifies as the #1 capability teams look for in fan engagement platforms.

Pre-Built Fan Segments

```
SEGMENT NAME          | CRITERIA                           | USE CASE
----------------------+------------------------------------+--------------------------
Dormant Fans          | No check-in in 60+ days            | Win-back campaign
Super Fans            | Hall of Fame tier, 10+ events/yr   | VIP exclusives, ambassador
At-Risk Fans          | Declining check-in frequency       | Retention offer
New Fans              | Registered < 30 days ago           | Onboarding drip campaign
Big Spenders          | Top 10% by purchase amount         | Premium reward offers
Social Amplifiers     | 5+ social shares in 30 days        | Influencer campaigns
Family Groups         | 3+ linked accounts                 | Family event packages
Streak Holders        | Active streak of 5+ events         | Streak protection offers
Lapsed Season Ticket  | Had season tickets, didn't renew   | Renewal campaign
Geographic Clusters   | Fans from specific zip codes       | Regional promotions
```

Custom Segment Builder

- No-code segment builder: combine filters (tier + location + purchase history + event frequency)
- Real-time segment sizing: see how many fans match before launching campaign
- Save and reuse segments across campaigns
- Scheduled auto-segmentation: segments recalculate daily based on latest behavior

Personalized Recommendations

- Reward recommendations based on past redemption patterns ('Fans like you loved...')
- Event suggestions based on attendance history and geographic proximity
- Challenge difficulty matching: new fans get easy challenges, veterans get harder ones
- Push notification timing optimized per fan based on engagement patterns

First-Party Data Value

- Every interaction = data point: check-ins, purchases, shares, reward preferences, challenge completions
- Zero reliance on third-party cookies or social media data
- GDPR/CCPA-compliant first-party data that teams OWN (not rented from Facebook/Google)
- Exportable to CRM (Salesforce, HubSpot) via CSV or API for broader marketing use

DATA IS THE REAL PRODUCT: The platform generates first-party fan data worth more than the subscription fee. Teams can use this data for ticket pricing, sponsor negotiations, merchandise planning, and targeted marketing -- all from their own platform, not rented from Big Tech.

13.4 Live Game Integration & Second Screen

Fans at sporting events spend 80%+ of downtime on their phones. Instead of losing them to Instagram or Twitter, the platform captures that attention with real-time, in-game engagement features that deepen their connection to the event AND earn them points.

In-Game Features

- Live play-by-play feed with real-time stats (synced with official data providers)
- In-game polls: 'Who scores next?' 'Will they convert the 3rd down?' -- instant results on screen
- Live prediction games: predict score at halftime, final score, MVP -- earn points for correct guesses
- Emoji/reaction stream: fans react in real-time (visible on venue jumbotron via sponsor integration)
- Trivia rounds between quarters/periods/innings -- timed, competitive, with leaderboard
- Photo moment: 'Selfie Cam' feature during timeouts -- displayed on big screen, saved in fan profile

In-Seat Ordering (Phase 3)

- Browse concession menu from seat -- no waiting in line
- Order food/drinks/merchandise for pickup or delivery to section
- Pay with points, stored value, or credit card (Stripe integration)
- Earn points on every in-seat purchase (POS integration via Square webhook)
- Average revenue increase: 20-30% when fans can order from seat (industry data)

Data Provider Integrations

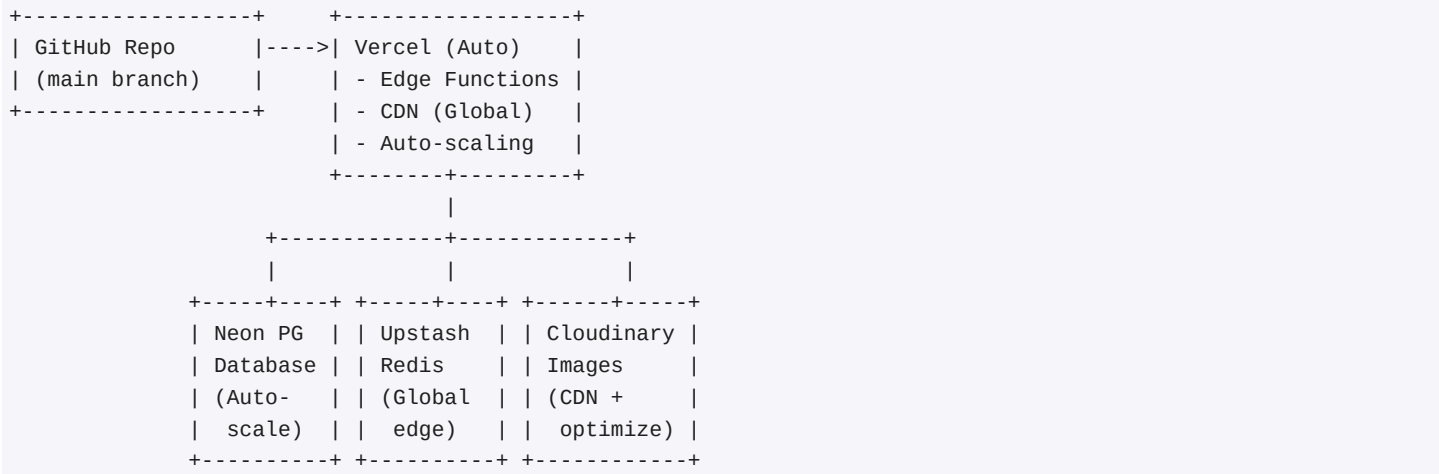
- Sportradar / Stats Perform: real-time play-by-play data feed
- ESPN API: scores, schedules, standings
- Custom WebSocket: real-time event stream for live UI updates (sub-second latency)

ENGAGEMENT DURING EVENTS: Teams that offer second-screen experiences see 3x longer app session times and 2x higher reward redemption rates during game days. This is where the platform earns its value.

Chapter 14

Deployment & Infrastructure

14.1 Deployment Architecture



14.2 CI/CD Pipeline

1. Developer pushes code to GitHub feature branch
2. GitHub Actions runs: TypeScript check, ESLint, Prisma validate
3. Automated test suite runs (unit + integration)
4. Vercel creates preview deployment for PR review
5. After merge to main: automatic production deployment
6. Sentry monitors for errors post-deploy
7. Automatic rollback if error rate exceeds threshold

14.3 Game-Day Scaling

Sporting events create massive traffic spikes. The architecture handles this:

- Vercel Edge Functions: auto-scale to millions of requests, zero configuration
- Neon PostgreSQL: auto-scaling with read replicas for high-read workloads
- Redis (Upstash): edge-cached leaderboards and point balances (sub-ms reads)
- CDN: all static assets served from edge (Vercel CDN + Cloudinary)
- Target: handle 10,000 concurrent check-ins within 60 seconds of gates opening

Chapter 15

Development Phases & Timeline

15.1 Phase 1: Core MVP (6-8 Weeks)

GOAL: Launch-ready platform with core loop: Register -> Check In -> Earn Points -> Redeem Rewards

- User registration & authentication (OAuth + email)
- Fan profile with points balance display
- Points engine (earn on check-in, admin-assigned)
- QR code check-in + GPS geofencing
- Basic rewards catalog with redemption
- 4-tier system (Rookie to Hall of Fame)
- Admin dashboard: user management, reward CRUD, basic analytics
- Responsive web app (mobile-first PWA)
- Push notifications (check-in confirmation, reward alerts)
- Apple Wallet & Google Wallet loyalty pass (highest adoption driver)

15.2 Phase 2: Engagement & Gamification (4-6 Weeks)

- Challenge system (attendance, purchase, social challenges)
- Achievement badges with visual collection
- Leaderboards (global, team, friends)
- Attendance streak tracking with multiplier bonuses
- Prediction games and trivia quizzes
- Social sharing with verified bonus points
- Friend lists and group competitions

15.3 Phase 3: Integrations & Scale (4-6 Weeks)

- Ticketing system integration (Ticketmaster/Eventbrite)
- POS integration (Square) for purchase-based points
- Stripe payments for premium features
- Advanced analytics with export
- Campaign builder (no-code)
- Sponsor portal for branded rewards
- Fan segmentation engine with custom segment builder
- Live game integration (polls, predictions, trivia)
- In-seat ordering (concessions from seat)
- API for white-label partners

15.4 Phase 4: Native Mobile + Advanced (6-8 Weeks)

- React Native iOS + Android apps (if needed)
- Apple Wallet / Google Wallet passes
- Bluetooth beacon zone detection
- AI-powered personalization (reward recommendations)
- Multi-venue/multi-team support
- Advanced fraud detection

15.5 Timeline Summary

```
PHASE | SCOPE                        | TIMELINE   | INVESTMENT
------+------------------------------+------------+-----------
  1   | Core MVP                     | 6-8 weeks  | $3K-$5K
  2   | Gamification + Social        | 4-6 weeks  | $2.5K-$4K
  3   | Integrations + Analytics     | 4-6 weeks  | $2K-$3K
  4   | Native Mobile + AI           | 6-8 weeks  | $3K-$5K
------+------------------------------+------------+-----------
TOTAL | Full Platform                | 5-7 months | $10K-$17K

NOTE: Phases are sequential. Each phase delivers a working,
deployable product. You can stop after any phase.
```

Chapter 16

Testing & QA Strategy

CODEVIX LABS DIFFERENCE: QA is not an afterthought. Every feature is tested BEFORE deployment. This is critical for a platform handling financial transactions (points = virtual currency).

16.1 Testing Pyramid

```
        /\          E2E Tests (Playwright)
       /  \         - Full user flows (register, check-in, redeem)
      /    \        - Cross-browser testing
     /------\       Integration Tests (Vitest + Prisma)
    /        \      - API endpoint testing with real DB
   /          \     - Points calculation accuracy
  /            \    - Tier upgrade logic
 /--------------\   Unit Tests (Vitest)
/                \  - Points engine pure functions
/==================\ - Validation schemas
                     - Utility functions
```

16.2 Critical Test Scenarios

Points Engine Tests

- Double check-in prevention (same user, same event)
- Points calculation accuracy across all multipliers and bonuses
- Concurrent point transactions (race condition prevention)
- Points expiration batch job correctness
- Negative balance prevention (can't spend more than you have)

Security Tests

- Authentication bypass attempts
- RBAC enforcement (fan can't access admin routes)
- Rate limiting effectiveness under load
- GPS spoofing detection for check-ins
- QR code replay attack prevention

Load Tests (Game Day Simulation)

- 10,000 simultaneous check-ins (gates opening scenario)
- 50,000 leaderboard queries per minute during event
- Flash reward drop with 1,000 concurrent redemption attempts
- Database connection pool under sustained load

Chapter 17

Scalability & Performance

17.1 Performance Targets

```
METRIC                          | TARGET
--------------------------------+---------
API response time (p95)         | < 200ms
Check-in processing             | < 500ms
Leaderboard query               | < 50ms (Redis)
Points balance lookup           | < 10ms (Redis cache)
Page load (LCP)                 | < 2.5s
Concurrent check-ins            | 10,000/min
Database connections             | 100 pooled
Uptime SLA                       | 99.9%
```

17.2 Scaling Strategy

- Read replicas: Neon PostgreSQL auto-creates read replicas for analytics queries
- Redis caching: hot data (balances, leaderboards) served from edge cache
- CDN: all static assets cached globally via Vercel Edge Network
- Database indexing: composite indexes on (userId + eventId), (userId + createdAt)
- Connection pooling: PgBouncer via Neon for efficient connection reuse
- Background jobs: points expiration, analytics aggregation run off-peak via cron

17.3 Future Microservices Extraction (When Needed)

If the platform reaches 100K+ users, these services would be extracted first:

1. Points Engine -> standalone service with its own database (highest transaction volume)
2. Notification Service -> async message queue (Firebase + email + SMS)
3. Analytics Service -> separate read-optimized database (ClickHouse or TimescaleDB)

Chapter 18

Why CodeVix Labs

18.1 What We Bring to This Project

QA-First Engineering

We don't bolt on testing after development. Every feature goes through our QA pipeline before it reaches production. For a platform handling virtual currency (points), this means zero double-spend bugs, zero phantom points, and zero downtime during live events.

The Exact Tech Stack

Next.js, React, TypeScript, PostgreSQL, Prisma, Redis, Stripe, Vercel -- this is our daily stack. We've built production applications with every technology in this blueprint. No learning curve, no experiments on your dime.

Production Marketplace Experience

We've built TheSkinProof -- a multi-vendor marketplace with user authentication, vendor dashboards, payment processing, admin panels, and role-based access control. The rewards platform uses the same architectural patterns: user accounts, transaction logic, tiered access, admin dashboards, and payment integration.

Deep Domain Research

This blueprint wasn't written from templates. We researched FanMaker, Tradable Bits, bFAN Sports, Qualifio, Sportian, and dozens of loyalty platform architectures. We understand the competitive landscape, the fan engagement patterns, and the technical challenges specific to sporting events.

Dedicated Attention

You won't be project #47 in a queue. We take on a small number of projects at a time so each client gets full attention, fast communication, and zero delays. Your rewards platform would be our priority build.

18.2 Working With Us

- Weekly progress demos -- see working software every 7 days
- Direct Slack/WhatsApp access to the developer (me) -- no project managers in between
- Git repository access from day one -- you own the code
- Deployed preview environments for every feature -- test before it goes live
- Documentation included -- API docs, deployment guide, admin manual

18.3 Next Steps

1. Discovery call to align on your specific vision and priorities (scheduled)
2. Detailed Phase 1 proposal with exact scope, timeline, and fixed price
3. Kickoff: repository setup, database schema, first deployment within 48 hours
4. Weekly demos until MVP launch

5. Launch support and iteration based on real fan feedback

READY TO BUILD? Let's turn this blueprint into a working platform. The architecture is designed, the tech stack is proven, and we're ready to start. Let's talk.

CodeVix Labs

Built with logic. Delivered with precision.

Rupak Amin, Founder
www.codevixlabs.com
rupak@codevixlabs.com