QicTrader - Technical Product Requirements Document

Version: 1.0
Date: October 2, 2025
Product: P2P Crypto Trading Platform with Zero-Capital Reselling
Target Launch: Q1 2026

---

1. Product Overview

1.1 Vision

QicTrader is a P2P crypto marketplace that enables zero-capital reselling. Users can build crypto businesses by marketing other vendors' offers at a markup, without holding inventory.

1.2 Core Innovation

Three-way trade model: Client → Reseller → Vendor, where resellers earn the markup spread without capital investment.

1.3 Key Differentiators

- Zero-capital reselling (unique in market)
- 15% affiliate commission program
- Mobile-first design for African markets
- Three revenue streams: trading, reselling, affiliates

---

2. System Architecture

2.1 Technology Stack

Frontend:

- React 19.x (web)
- React Router DOM 7.x
- Tailwind CSS 3.x
- Lucide React (icons)

Backend (Recommended):

- Node.js + Express OR Django + DRF
- PostgreSQL (primary database)
- Redis (caching, sessions)
- WebSocket (real-time chat)

Blockchain/Crypto:

- BTC/ETH node integration OR API (BlockCypher, BitGo)
- Hot wallet for escrow management
- Cold storage for platform reserves

Infrastructure:

- AWS/GCP/DigitalOcean

- CDN for static assets
- Load balancer for scaling

2.2 Core Services

```
┌─────────────────────────────────────────┐
│           Load Balancer                 │
└─────────────────────────────────────────┘
                 ↓
┌─────────────────────────────────────────┐
│         API Gateway / Router            │
└─────────────────────────────────────────┘
     ↓         ↓          ↓
┌──────────┐ ┌──────────┐ ┌──────────┐
│   Auth   │ │  Trade   │ │  Wallet  │
│ Service  │ │ Service  │ │ Service  │
└──────────┘ └──────────┘ └──────────┘
     ↓         ↓          ↓
┌─────────────────────────────────────────┐
│       PostgreSQL Database               │
└─────────────────────────────────────────┘
     ↓         ↓          ↓
┌──────────┐ ┌──────────┐ ┌──────────┐
│  Redis   │ │  Queue   │ │  Crypto  │
│  Cache   │ │(RabbitMQ)│ │  Nodes   │
└──────────┘ └──────────┘ └──────────┘
```

---

3. Database Schema (Critical Tables)

3.1 Users

sql

```
users
├── id (UUID, PK)
├── email (VARCHAR, UNIQUE)
├── password_hash (VARCHAR)
├── username (VARCHAR, UNIQUE)
├── phone (VARCHAR)
├── kyc_status (ENUM: pending, verified, rejected)
├── referral_code (VARCHAR, UNIQUE)
├── referred_by (UUID, FK → users.id)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.2 Wallets

sql

```sql
wallets
├── id (UUID, PK)
├── user_id (UUID, FK → users.id)
├── crypto_symbol (VARCHAR: BTC, ETH, USDT)
├── balance (DECIMAL(18,8))
├── address (VARCHAR)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.3 Offers

sql

```
offers
├── id (UUID, PK)
├── vendor_id (UUID, FK → users.id)
├── type (ENUM: buy, sell)
├── crypto_symbol (VARCHAR)
├── payment_method (VARCHAR)
├── min_amount (DECIMAL)
├── max_amount (DECIMAL)
├── markup_percent (DECIMAL(5,2))
├── fiat_currency (VARCHAR: ZAR, USD)
├── is_active (BOOLEAN)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.4 Reseller Offers

sql

```
reseller_offers
├── id (UUID, PK)
├── reseller_id (UUID, FK → users.id)
├── source_offer_id (UUID, FK → offers.id)
├── my_markup_percent (DECIMAL(5,2))
├── total_markup_percent (DECIMAL(5,2))
├── shareable_link (VARCHAR, UNIQUE)
├── clicks (INTEGER, DEFAULT 0)
├── conversions (INTEGER, DEFAULT 0)
├── is_active (BOOLEAN)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.5 Trades

sql

```
trades
├── id (UUID, PK)
├── trade_number (VARCHAR, UNIQUE: TRD-XXXX)
├── buyer_id (UUID, FK → users.id)
├── seller_id (UUID, FK → users.id)
├── reseller_id (UUID, FK → users.id, NULLABLE)
├── offer_id (UUID, FK → offers.id)
├── crypto_symbol (VARCHAR)
├── crypto_amount (DECIMAL(18,8))
├── fiat_amount (DECIMAL(10,2))
├── fiat_currency (VARCHAR)
├── payment_method (VARCHAR)
├── vendor_markup_percent (DECIMAL(5,2))
├── reseller_markup_percent (DECIMAL(5,2), NULLABLE)
├── platform_fee_percent (DECIMAL(5,2))
├── status (ENUM: pending_payment, payment_sent,
│      confirming, completed, disputed, cancelled)
├── escrow_address (VARCHAR)
├── expires_at (TIMESTAMP)
├── completed_at (TIMESTAMP, NULLABLE)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.6 Transactions

sql

```
transactions
├── id (UUID, PK)
├── wallet_id (UUID, FK → wallets.id)
├── trade_id (UUID, FK → trades.id, NULLABLE)
├── type (ENUM: deposit, withdrawal, trade_buy,
│    trade_sell, fee, commission, reseller_earning)
├── amount (DECIMAL(18,8))
├── status (ENUM: pending, completed, failed)
├── tx_hash (VARCHAR, NULLABLE)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

3.7 Affiliate Earnings

sql

```
affiliate_earnings
  ├── id (UUID, PK)
  ├── referrer_id (UUID, FK → users.id)
  ├── referred_user_id (UUID, FK → users.id)
  ├── trade_id (UUID, FK → trades.id)
  ├── commission_amount (DECIMAL(10,2))
  ├── commission_percent (DECIMAL(5,2): 15.0)
  ├── status (ENUM: pending, paid)
  ├── paid_at (TIMESTAMP, NULLABLE)
  └── created_at (TIMESTAMP)
```

3.8 Chat Messages

sql

```
chat_messages
  ├── id (UUID, PK)
  ├── trade_id (UUID, FK → trades.id)
  ├── sender_id (UUID, FK → users.id)
  ├── message_type (ENUM: text, image, system)
  ├── content (TEXT)
  ├── attachment_url (VARCHAR, NULLABLE)
  ├── is_read (BOOLEAN)
  ├── created_at (TIMESTAMP)
  └── updated_at (TIMESTAMP)
```

---

4. Critical API Endpoints

4.1 Authentication

```
POST   /api/auth/register
POST   /api/auth/login
POST   /api/auth/logout
POST   /api/auth/refresh-token
POST   /api/auth/forgot-password
POST   /api/auth/reset-password
```

4.2 Marketplace

```
GET    /api/offers?crypto=BTC&type=buy&payment_method=FNB
GET    /api/offers/:id
POST   /api/offers (create vendor offer)
PUT    /api/offers/:id
DELETE /api/offers/:id
```

4.3 Reseller System

```
POST   /api/reseller/offers (create reseller offer)
GET    /api/reseller/offers (get my reseller offers)
GET    /api/reseller/offers/:id/stats
PUT    /api/reseller/offers/:id
DELETE /api/reseller/offers/:id
GET    /api/reseller/earnings
```

4.4 Trading

```
POST   /api/trades (initiate trade)
GET    /api/trades (list my trades)
GET    /api/trades/:id
POST   /api/trades/:id/mark-paid
POST   /api/trades/:id/release-escrow
POST   /api/trades/:id/dispute
POST   /api/trades/:id/cancel
```

4.5 Chat

```
GET    /api/trades/:id/messages
POST   /api/trades/:id/messages
WS     /ws/trades/:id (WebSocket for real-time)
```

4.6 Wallet

```
GET    /api/wallets
GET    /api/wallets/:crypto/balance
POST   /api/wallets/:crypto/deposit-address
POST   /api/wallets/:crypto/withdraw
GET    /api/wallets/:crypto/transactions
```

4.7 Affiliate

```
GET    /api/affiliate/stats
GET    /api/affiliate/referrals
GET    /api/affiliate/earnings
POST   /api/affiliate/withdraw
```

---

5. Trade Flow Logic

5.1 Standard P2P Trade (No Reseller)

```
1. Buyer clicks "Buy BTC" on vendor offer
2. System creates trade with status: pending_payment
3. Escrow locks vendor's BTC
4. Buyer sends fiat payment to vendor
5. Buyer marks "I've Paid"
6. Trade status → payment_sent
7. Vendor confirms receipt
8. Trade status → completed
9. Escrow releases BTC to buyer
10. Platform fee deducted
11. Vendor gets fiat amount
```

Fee Distribution:

```
Total: 10,000 ZAR
Platform fee (1.5%): 150 ZAR
Vendor receives: 9,850 ZAR
```

5.2 Reseller Trade (3-Way)

```
1. Client clicks reseller's shareable link
2. System creates trade with reseller_id populated
3. Escrow locks vendor's BTC
4. Client sends fiat payment to RESELLER
5. Reseller marks "Payment Received from Client"
6. Reseller sends fiat payment to VENDOR
7. Reseller marks "I've Paid Vendor"
8. Vendor confirms receipt
9. Trade status → completed
10. Escrow releases BTC to client
11. Platform calculates splits:
   - Platform fee: 1.5% of total
   - Reseller markup: (reseller % - vendor %)
   - Vendor gets: base + vendor markup
```

Fee Distribution Example:

```
Client pays: 11,190 ZAR (10,000 + 11.9% total markup)
├── Platform fee (1.5% of 10,000): 150 ZAR
├── Reseller markup (3%): 300 ZAR
└── Vendor receives (base + 8.9%): 10,740 ZAR

Total = 11,190 ZAR ✓
```

5.3 Critical Business Rules

1. **Escrow Lock:** BTC must be locked before trade status = pending_payment
2. **Expiry Timer:** Trades expire after 30 minutes if payment not marked
3. **Reseller Validation:** Reseller's markup must be > vendor's markup
4. **Payment Confirmation:** Both parties must confirm in 3-way trades
5. **Dispute Window:** 24 hours after completion for disputes
6. **Affiliate Commission:** Paid only on completed trades

---

6. Security Requirements

6.1 Critical Security Measures

Authentication:

- JWT tokens (access + refresh)
- Password hashing (bcrypt, min 10 rounds)
- 2FA optional (TOTP via Google Authenticator)
- Rate limiting: 5 login attempts per 15 min

Escrow Security:

- Multi-sig wallets for escrow
- Cold storage for 80% of platform reserves
- Hot wallet limits: max 10 BTC at a time
- Daily security audits on wallet balances

API Security:

- HTTPS only (TLS 1.3)
- CORS restrictions
- Rate limiting: 100 req/min per user
- Input validation & sanitization
- SQL injection prevention (parameterized queries)
- XSS protection

Trade Security:

- Verify payment proof uploads
- Admin dispute resolution system
- Automated fraud detection (velocity checks)
- User reputation system

6.2 KYC/AML Compliance

Tier 1 (No KYC):

- Max trade: 1,000 ZAR per day
- Email verification only

Tier 2 (Basic KYC):

- Max trade: 50,000 ZAR per day
- ID verification required

Tier 3 (Full KYC):

- Unlimited trading
- Proof of address required

---

7. Features by Priority

7.1 MVP (Phase 1) - Must Have

Core Functionality:

- ✅ User registration/login
- ✅ Vendor offer creation (BTC only)
- ✅ Standard P2P trading
- ✅ Escrow system
- ✅ Trade chat
- ✅ Wallet (deposit/withdraw)
- ✅ Reseller offer creation
- ✅ 3-way reseller trades
- ✅ Affiliate tracking
- ✅ Basic admin panel

Payment Methods (ZAR only):

- FNB E-Wallet
- Capitec Pay
- Nedbank
- Bank Transfer

7.2 Phase 2 - Should Have

- Multi-crypto support (ETH, USDT, USDC)
- Advanced analytics for resellers
- Reputation/rating system
- Dispute resolution workflow
- Push notifications (web + mobile)
- Email notifications
- Transaction history export
- Tax reporting

7.3 Phase 3 - Nice to Have

- Mobile apps (iOS + Android)
- Multiple fiat currencies (USD, KES, NGN)
- Automated market making
- API for third-party integrations
- Reseller custom branding
- Gamification (leaderboards)
- Referral contests

---

8. Performance Requirements

8.1 Response Times

- API endpoints: < 200ms (p95)
- WebSocket messages: < 100ms
- Database queries: < 50ms
- Page load time: < 2s

8.2 Scalability

- Support 10,000 concurrent users
- 1,000 trades per day (Year 1)
- 10,000 trades per day (Year 2)
- 99.9% uptime SLA

8.3 Data Retention

- User data: Indefinite
- Trade history: 7 years (compliance)
- Chat logs: 2 years
- Transaction logs: Indefinite

---

9. Implementation Milestones

Sprint 1-2 (Weeks 1-4): Foundation

- Database setup
- Authentication system
- User registration/login
- Basic API structure

Sprint 3-4 (Weeks 5-8): Core Trading

- Offer creation
- Standard P2P trades
- Escrow integration
- Wallet system

Sprint 5-6 (Weeks 9-12): Reseller System

- Reseller offer creation
- 3-way trade logic
- Shareable links

- Markup calculations

Sprint 7-8 (Weeks 13-16): Polish & Launch

- Trade chat
- Affiliate tracking
- Admin panel
- Testing & bug fixes
- Production deployment

Total Timeline: 16 weeks (4 months)

---

10. Testing Requirements

10.1 Critical Test Cases

Escrow System:

- ✓ BTC locks correctly on trade creation
- ✓ BTC releases only on both confirmations
- ✓ Timeout handling (auto-cancel after 30 min)
- ✓ Escrow balance accuracy

Reseller Trades:

- ✓ Correct fee splits calculated
- ✓ Reseller receives markup difference
- ✓ Vendor receives correct amount
- ✓ Platform fee deducted properly

Edge Cases:

- ✓ Expired trades handle correctly
- ✓ Concurrent trade attempts
- ✓ Insufficient wallet balance
- ✓ Network failures during escrow release
- ✓ Duplicate payment confirmations

10.2 Test Coverage Goals

- Unit tests: 80% coverage
- Integration tests: Critical paths only
- E2E tests: Full trade flows (standard + reseller)
- Load testing: 5,000 concurrent users

---

11. Monitoring & Observability

11.1 Key Metrics to Track

Business Metrics:

- Daily active users (DAU)
- Trade volume (ZAR)
- Conversion rate (signup → first trade)
- Reseller adoption rate

- Affiliate conversion rate
- Average trade value

Technical Metrics:

- API response times (p50, p95, p99)
- Error rate (< 0.1%)
- Escrow balance accuracy
- WebSocket connection stability
- Database query performance

Alerts:

- Escrow balance mismatch
- Failed blockchain transactions
- API error rate > 1%
- Response time > 1s
- Wallet balance < threshold

11.2 Logging

- Structured logging (JSON format)
- Log levels: ERROR, WARN, INFO, DEBUG
- Centralized logging (ELK stack or Datadog)
- Audit logs for all financial transactions

---

12. Deployment & DevOps

12.1 Environments

- **Development:** Local + staging server
- **Staging:** Pre-production testing
- **Production:** Live environment

12.2 CI/CD Pipeline

```
Code Push → Tests Run → Build → Deploy to Staging →
Manual Approval → Deploy to Production → Smoke Tests
```

12.3 Backup Strategy

- Database: Hourly snapshots, 30-day retention
- Hot wallet backups: Real-time replication
- Code repository: GitHub/GitLab

12.4 Disaster Recovery

- RTO (Recovery Time Objective): < 4 hours
- RPO (Recovery Point Objective): < 1 hour
- Regular DR drills (quarterly)

---

13. Risks & Mitigation

Risk	Impact	Mitigation
Regulatory shutdown	HIGH	Legal review, KYC/AML compliance
Hot wallet hack	HIGH	Multi-sig, cold storage, insurance
Low liquidity	MEDIUM	Partner with initial vendors
Reseller fraud	MEDIUM	Reputation system, limits
Platform downtime	MEDIUM	Load balancing, auto-scaling
Price manipulation	LOW	Market monitoring, alerts

---

14. Success Metrics (6 Months Post-Launch)

- 5,000 registered users
- 500 active traders per month
- 200 active resellers
- $500,000 monthly trade volume
- 50% user retention (monthly)
- < 1% dispute rate

---

15. Developer Handoff Checklist

- [ ] Codebase repository access
- [ ] UI designs (Figma/reference screenshots)
- [ ] Database schema scripts
- [ ] API documentation (Swagger/Postman)
- [ ] Environment variables template
- [ ] Third-party API credentials (crypto nodes)
- [ ] Test accounts for each role
- [ ] Staging environment URL
- [ ] Weekly sync meetings scheduled

Document Owner: Product Manager
Technical Lead: [To be assigned]
Questions: Contact product@qictrader.com

END OF DOCUMENT