Nexus26 Smart Contracts Software Requirements Specification (SRS)

Version: 1.0
Date: May 31, 2025
Based on: Smart Contract Skeleton v0.7

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Nexus26 decentralized prediction platform for the 2026 FIFA World Cup. It serves as the primary reference for developers, designers, and testers implementing the smart contracts on the TON blockchain.

1.2 Scope

The system consists of 18 smart contracts that together form a complete prediction market platform with:

- Pre-token virtual points system
- Multiple prediction market types
- Reputation-based reward multipliers
- Achievement NFT system

- Oracle consensus mechanism
- Emergency control systems

1.3 Definitions and Acronyms

- **RIMET**: Platform utility token (9 decimals)
- **BPS**: Basis points (100 BPS = 1%)
- **SBT**: Soulbound Token (non-transferable NFT)
- **TON**: The Open Network blockchain
- **DAO**: Decentralized Autonomous Organization

2. System Overview

2.1 Business Context

Nexus26 enables football fans to make predictions on 2026 FIFA World Cup matches and events, earning rewards based on accuracy and reputation. The platform operates in two phases:

1. **Pre-token phase**: Virtual points system with 2 TON registration
2. **Token phase**: Full $RIMET token-based predictions

2.2 Key Stakeholders

- **Users**: Football fans making predictions
- **Oracles**: External data providers for match results
- **Liquidity Providers**: Users providing market liquidity
- **Platform Administrators**: Managing markets and emergency situations

3. Architectural Design

3.1 Contract Hierarchy

```
┌─────────────────────────────────────────────────────────┐
│                    EmergencyController                   │
│                  (Global Circuit Breaker)                │
└────────────────────────┬────────────────────────────────┘
                         │
┌────────────────────────┴────────────────────────────────┐
│                      Core Contracts                      │
├─────────────────────────────────────────────────────────┤
│  MatchRegistry     │  OracleCoordinator  │ MarketFactory │
│  (Match Data)      │  (Result Consensus)  │ (Deployment)  │
└───────┬────────────┴──────────┬──────────┴───────┬──────┘
        │                       │                  │
┌───────┴────────────┐ ┌────────┴─────────┐ ┌──────┴───────┐
│ PredictionMarket   │ │ PlayerMarket     │ │ OverUnderMarket│
│ (1×2 Outcomes)     │ │ (Player Stats)   │ │ (Goal Totals)  │
└────────────────────┘ └──────────────────┘ └──────────────┘
                                │
┌───────────────────────────────┴─────────────────────────┐
│                     Support Contracts                    │
├─────────────────────────────────────────────────────────┤
│PointsRegistry │ StakeVault │ ReputationMgr │ TreasuryMgr│
│ AchievementNFT│ LiquidityMgr│SettlementRouter│          │
└─────────────────────────────────────────────────────────┘
```

3.2 Contract Dependencies

Contract	Depends On	Used By
MatchRegistry	AccessControl	MarketFactory, OracleCoordinator
PredictionMarket	StakeVault, ReputationMgr, TreasuryMgr	SettlementRouter
PointsRegistry	AccessControl	Pre-token predictions
TreasuryManager	None	All market contracts

4. Functional Requirements

4.1 MatchRegistry

4.1.1 Match Management

- **FR-MR-001**: Admin can add matches with kickoff time and team metadata
- **FR-MR-002**: Match status transitions: Inactive → Open → Closed → Settled
- **FR-MR-003**: Auto-close matches based on type (15/30/60 min before kickoff)

- **FR-MR-004**: Store match results from OracleCoordinator

4.1.2 Access Control

- **FR-MR-005**: Only admins can modify match data
- **FR-MR-006**: Only OracleCoordinator can set match results

4.2 OracleCoordinator

4.2.1 Oracle Management

- **FR-OC-001**: Owner can add/remove authorized oracles
- **FR-OC-002**: Track oracle reputation scores

4.2.2 Result Consensus

- **FR-OC-003**: Require 2-of-N oracle agreement for results
- **FR-OC-004**: Store final results in `_finalResults` mapping
- **FR-OC-005**: Support dispute mechanism for owner override

4.3 MarketFactory

4.3.1 Market Deployment

- **FR-MF-001**: Deploy simple 1×2 prediction markets
- **FR-MF-002**: Deploy player performance markets with player ID and stat type
- **FR-MF-003**: Deploy over/under markets with goal threshold
- **FR-MF-004**: Deploy tournament progression markets

4.3.2 Market Registry

- **FR-MF-005**: Maintain mapping of match ID to market address
- **FR-MF-006**: Emit MarketCreated events

4.4 PredictionMarket

4.4.1 Initialization

- **FR-PM-001**: Initialize with references to StakeVault, ReputationMgr, TreasuryMgr, AchievementNFT

4.4.2 Prediction Flow

- **FR-PM-002**: Validate user has 100 RIMET staked in StakeVault
- **FR-PM-003**: Enforce MIN_PREDICTION (50 RIMET) and MAX_PREDICTION (250 RIMET)
- **FR-PM-004**: Deduct platform fee (default 3%, max 5%)
- **FR-PM-005**: Track stakes by outcome and user

4.4.3 Circuit Breakers

- **FR-PM-006**: Enforce per-user stake limits (10K default, 50K for high reputation)
- **FR-PM-007**: Enforce per-market total stake limit (1M default)
- **FR-PM-008**: Support emergency pause functionality

4.4.4 Settlement

- **FR-PM-009**: Calculate rewards using conviction multipliers (1.2×/1.5×/2.0×)
- **FR-PM-010**: Apply reputation multipliers (0.8×-1.5×)
- **FR-PM-011**: Apply achievement bonuses (up to 3× total cap)
- **FR-PM-012**: Distribute losing stakes: 85% to winners, 10% burn, 5% treasury

4.5 PointsRegistry

4.5.1 Premium Registration

- **FR-PR-001**: Accept 2 TON payment for premium status
- **FR-PR-002**: Emit PremiumRegistered event

4.5.2 Points Management

- **FR-PR-003**: Allocate 1000 points weekly to premium users
- **FR-PR-004**: Enforce 5 active predictions maximum per user
- **FR-PR-005**: Validate prediction amounts (50-250 points)
- **FR-PR-006**: Release prediction slots when settled

4.5.3 Token Conversion

- **FR-PR-007**: One-time conversion of points to RIMET tokens at launch

4.6 StakeVault

4.6.1 Activation Stakes

- **FR-SV-001**: Lock 100 RIMET for platform activation

- **FR-SV-002**: Track open positions per user
- **FR-SV-003**: Prevent withdrawal with open positions

4.6.2 Emergency Features

- **FR-SV-004**: Emergency unlock with configurable penalty (e.g., 10%)
- **FR-SV-005**: Emit EmergencyWithdraw events

4.7 ReputationManager

4.7.1 Score Calculation

- **FR-RM-001**: Track correct/total predictions per user
- **FR-RM-002**: Calculate reputation score (0-100)
- **FR-RM-003**: Weight by stake size and odds difficulty

4.7.2 Tier System

- **FR-RM-004**: Apply multipliers by tier: 0.8× (0-30), 1.0× (31-60), 1.2× (61-85), 1.5× (86-100)
- **FR-RM-005**: Check achievement eligibility at tier thresholds

4.8 TreasuryManager

4.8.1 Fee Distribution

- **FR-TM-001**: Transfer platform fees to community treasury
- **FR-TM-002**: Distribute losing stakes: 85% rewards, 10% burn, 5% treasury
- **FR-TM-003**: Execute token burns to BURN_SINK address

4.9 AchievementNFT

4.9.1 Achievement System

- **FR-AN-001**: Immutable core achievements (IDs 1-5)
- **FR-AN-002**: Mutable seasonal achievements (IDs 1000+)
- **FR-AN-003**: Define criteria types: ConsecutiveWins, PerfectGroup, BracketPerfection
- **FR-AN-004**: Mint soulbound NFTs when criteria met

4.9.2 Bonus Calculation

- **FR-AN-005**: Calculate total bonus multiplier across all achievements
- **FR-AN-006**: Apply bonus caps per achievement type

5. Non-Functional Requirements

5.1 Performance Requirements

- **NFR-P-001**: Process predictions within 3 seconds
- **NFR-P-002**: Handle 1000 concurrent predictions per match
- **NFR-P-003**: Settle markets with 10,000 participants within 60 seconds
- **NFR-P-004**: Batch operations limited to 10 items for gas optimization

5.2 Security Requirements

- **NFR-S-001**: All admin functions protected by AccessControl
- **NFR-S-002**: Integer overflow protection on all arithmetic
- **NFR-S-003**: Reentrancy guards on token transfers
- **NFR-S-004**: Emergency pause capability within 30 seconds

5.3 Reliability Requirements

- **NFR-R-001**: 99.9% uptime during World Cup matches
- **NFR-R-002**: Zero fund loss tolerance
- **NFR-R-003**: Graceful degradation if oracles fail

6. Interface Specifications

6.1 ITelegramBridge Interface

```tact
interface ITelegramBridge {
    // Link Telegram account to TON wallet
    external fn authenticateUser(uint64 telegramId, address tonWallet);

    // Notify mini-app of prediction placement
    external fn notifyPrediction(address user, address market, Outcome o, Amount a);

    // Push match results to Telegram users
    external fn pushResult(MatchKey key, Result r);
}
```

6.2 Event Specifications

Contract	Event	Parameters	When Emitted
MatchRegistry	ResultPosted	MatchKey, Result	Match result finalized
PointsRegistry	PremiumRegistered	address	User pays 2 TON
PointsRegistry	WeeklyPointsAllocated	timestamp, amount	Weekly distribution
PredictionMarket	Payout	address, Amount	User claims winnings
StakeVault	EmergencyWithdraw	address, net, penalty	Emergency unlock used

7. Data Flow

7.1 Prediction Flow

```
User → PredictionMarket.predict()
  ├─→ StakeVault.hasActivation() [verify 100 RIMET locked]
  ├─→ Calculate platform fee (3%)
  ├─→ TreasuryManager.transferFee()
  ├─→ Record stake in contract
  └─→ StakeVault.incrementPositions()
```

7.2 Settlement Flow

```
SettlementRouter.settleBatch()
  ├─→ OracleCoordinator.finalResultOf()
  ├─→ PredictionMarket.settle()
  │    ├─→ ReputationManager.getMultiplier()
  │    ├─→ AchievementNFT.getTotalBonus()
  │    ├─→ Calculate rewards
  │    └─→ TreasuryManager.distributeLosingPool()
  └─→ ReputationManager.addResult()
```

8. Security Requirements

8.1 Access Control Matrix

Function	Owner	Admin	Oracle	User
addMatch	❌	✅	❌	❌
setResult	❌	❌	✅	❌
predict	❌	❌	❌	✅
pauseAllMarkets	✅	❌	❌	❌

8.2 Critical Security Checks

- **SC-001**: Validate all external inputs (amounts, addresses, array lengths)
- **SC-002**: Check authorization before state changes
- **SC-003**: Emit events for all significant state changes
- **SC-004**: Implement checks-effects-interactions pattern
- **SC-005**: Validate arithmetic operations won't overflow/underflow

9. Testing Requirements

9.1 Unit Tests

Each contract requires:

- **UT-001**: Constructor and initialization tests
- **UT-002**: Access control tests (positive and negative)
- **UT-003**: State transition tests
- **UT-004**: Edge case handling (zero amounts, max values)
- **UT-005**: Event emission verification

9.2 Integration Tests

- **IT-001**: Full prediction lifecycle (stake → predict → settle → claim)
- **IT-002**: Multi-oracle consensus scenarios
- **IT-003**: Circuit breaker activation and recovery
- **IT-004**: Points to token conversion process
- **IT-005**: Achievement criteria and NFT minting

9.3 Stress Tests

- **ST-001**: 1000 simultaneous predictions
- **ST-002**: Settlement with 10,000 participants
- **ST-003**: Gas consumption under maximum load
- **ST-004**: Oracle failure scenarios

9.4 Security Audits

- **SA-001**: Formal verification of critical functions
- **SA-002**: Reentrancy attack simulations

- **SA-003**: Economic attack vectors (manipulation, front-running)
- **SA-004**: Access control bypass attempts

10. Deployment Requirements

10.1 Deployment Order

1. **Phase 1**: Core Infrastructure
   - ErrorCodes, PlatformConstants libraries
   - AccessControl library
   - TreasuryManager
2. **Phase 2**: Data Contracts
   - MatchRegistry
   - OracleCoordinator
   - ReputationManager
   - AchievementNFT
3. **Phase 3**: Market Infrastructure
   - StakeVault
   - MarketFactory
   - PointsRegistry
4. **Phase 4**: Market Contracts
   - PredictionMarket (template)
   - Specialized market templates
5. **Phase 5**: Auxiliary Contracts
   - SettlementRouter
   - LiquidityManager
   - ReferralRegistry
   - BatchOperations
   - EmergencyController

10.2 Configuration Requirements

- **DC-001**: Set TREASURY_WALLET before deployment
- **DC-002**: Deploy with correct market code cells in MarketFactory
- **DC-003**: Initialize contract cross-references post-deployment

- **DC-004**: Set initial oracle addresses
- **DC-005**: Configure achievement definitions and criteria

10.3 Monitoring Requirements

- **MR-001**: Track gas usage per function
- **MR-002**: Monitor circuit breaker triggers
- **MR-003**: Alert on emergency function usage
- **MR-004**: Track treasury balance and burn rate
- **MR-005**: Monitor oracle consensus failures

Appendix A: Error Codes

Code	Name	Description
100	NOT_OWNER	Caller is not contract owner
101	NOT_ADMIN	Caller is not admin or owner
200	PAUSED	Contract is paused
300	BELOW_MIN	Amount below minimum threshold
301	ABOVE_MAX	Amount above maximum threshold
302	TOO_MANY_ACTIVE	User has too many active predictions
400	BATCH_TOO_LARGE	Batch operation exceeds size limit

Appendix B: Gas Estimates

Operation	Estimated Gas	Notes
predict()	~150,000	Includes stake vault check
settle()	~50,000 × n	n = number of participants
mint achievement	~100,000	One-time per achievement
batch predict (10)	~1,200,000	Max batch size

Document Status: Final
Next Review: Before Phase 2 launch (Q3 2025)