ft_transcendence: Full-Stack Multiplayer Pong Platform

Capstone Project Presentation

Calvin Hon, Muhammad Ali Danish, Mahad Abdullah, Nguyen The Hoach

42 School

January 20, 2026

Project Overview

ft_transcendence: Production-ready, full-stack multiplayer Pong platform

Real-time WebSocket gameplay at 60 FPS

Microservices architecture with blockchain-integrated tournaments

Comprehensive security (WAF + Vault)

Supports AI opponents, campaign progression, achievements, leaderboards

Project Objectives

Implement server-authoritative Pong with real-time synchronization

Secure, scalable microservices supporting concurrent sessions

Tournament management with blockchain immutability

Production-grade security with WAF and secrets management

Web-based access with extensible architecture

Quality Metrics

Functional Completeness: 100% subject compliance

Security: Zero critical vulnerabilities, WAF active

Code Quality: TypeScript strict mode, ESLint, consistent standards

SDLC Approach

Iterative, incremental model with 5 phases

Planning & Requirements Analysis

Architectural Design

Implementation (Iterative)

Deployment & Evolution

Project Timeline

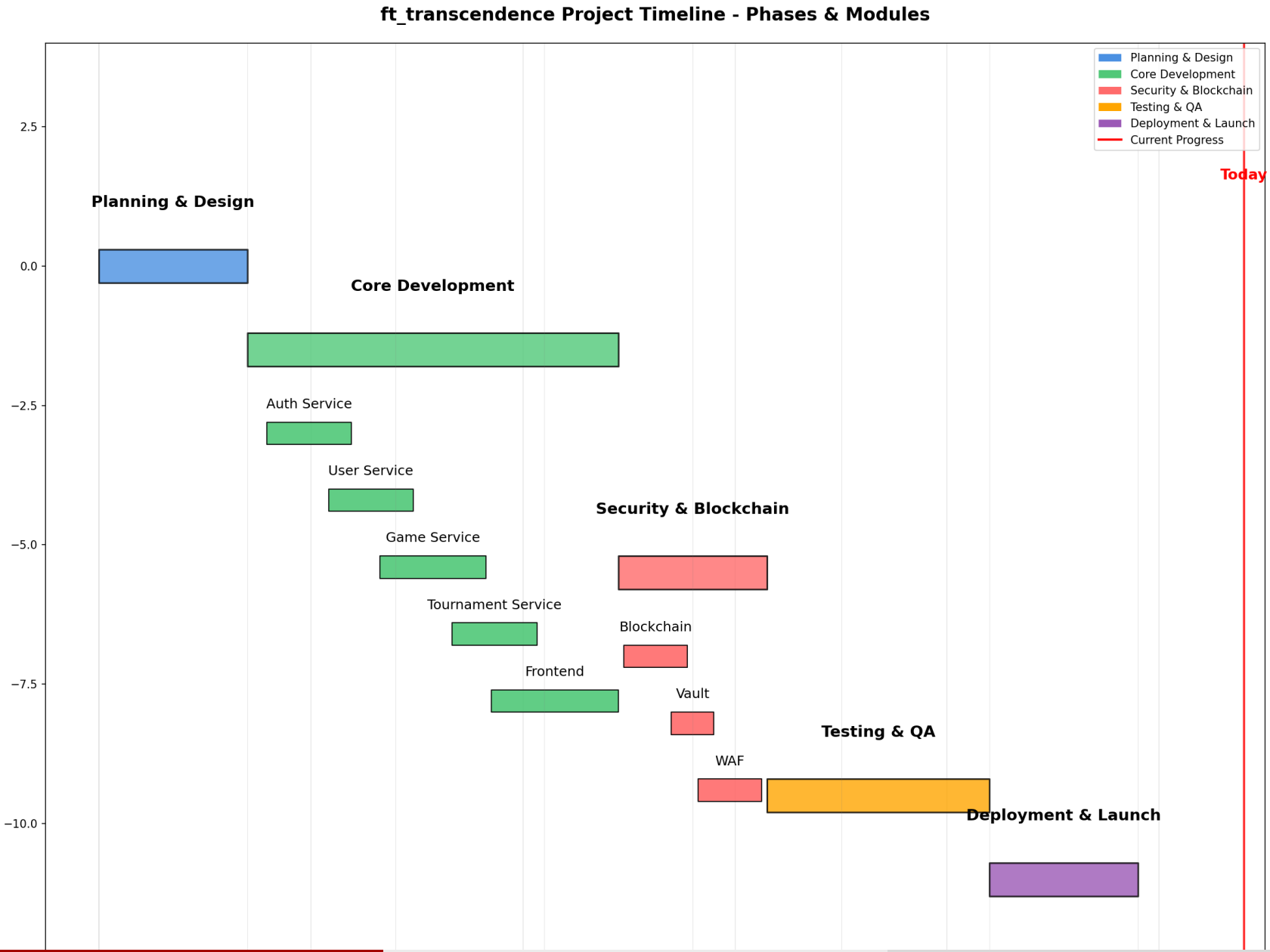
Phase 1 (Weeks 1-2): Planning, requirements analysis, architecture design

Phase 2 (Weeks 3-8): Core service development, game logic

Phase 3 (Weeks 9-10): Security hardening, WAF, Vault, blockchain

Phase 4 (Week 11): Deployment, production readiness

Gantt Chart


ft_transcendence Project Timeline - Phases & Modules
Planning & Design
Core Development
Security & Blockchain
Testing & QA
Deployment & Launch
Current Progress
Today
Planning & Design
Core Development
Auth Service
User Service
Game Service
Tournament Service
Frontend
Security & Blockchain
Blockchain
Vault
WAF
Testing & QA
Deployment & Launch
2.5
0.0
−2.5
−5.0
−7.5
−10.0

High-Level Requirements

Functional: User management, gameplay, social features, tournaments

Technical: Microservices, technology stack, security, performance

Functional Requirements

FR-1: User registration and authentication

FR-6: 60 FPS server-authoritative Pong

FR-11: Friend system and leaderboards

FR-15: Tournament creation and management

FR-17: Blockchain-recorded tournament results

Technical Requirements

TR-1: 4 microservices (auth, user, game, tournament)

TR-6: Node.js 20+ with Fastify v4 and TypeScript

TR-11: HTTPS with TLS 1.2+, WAF protection

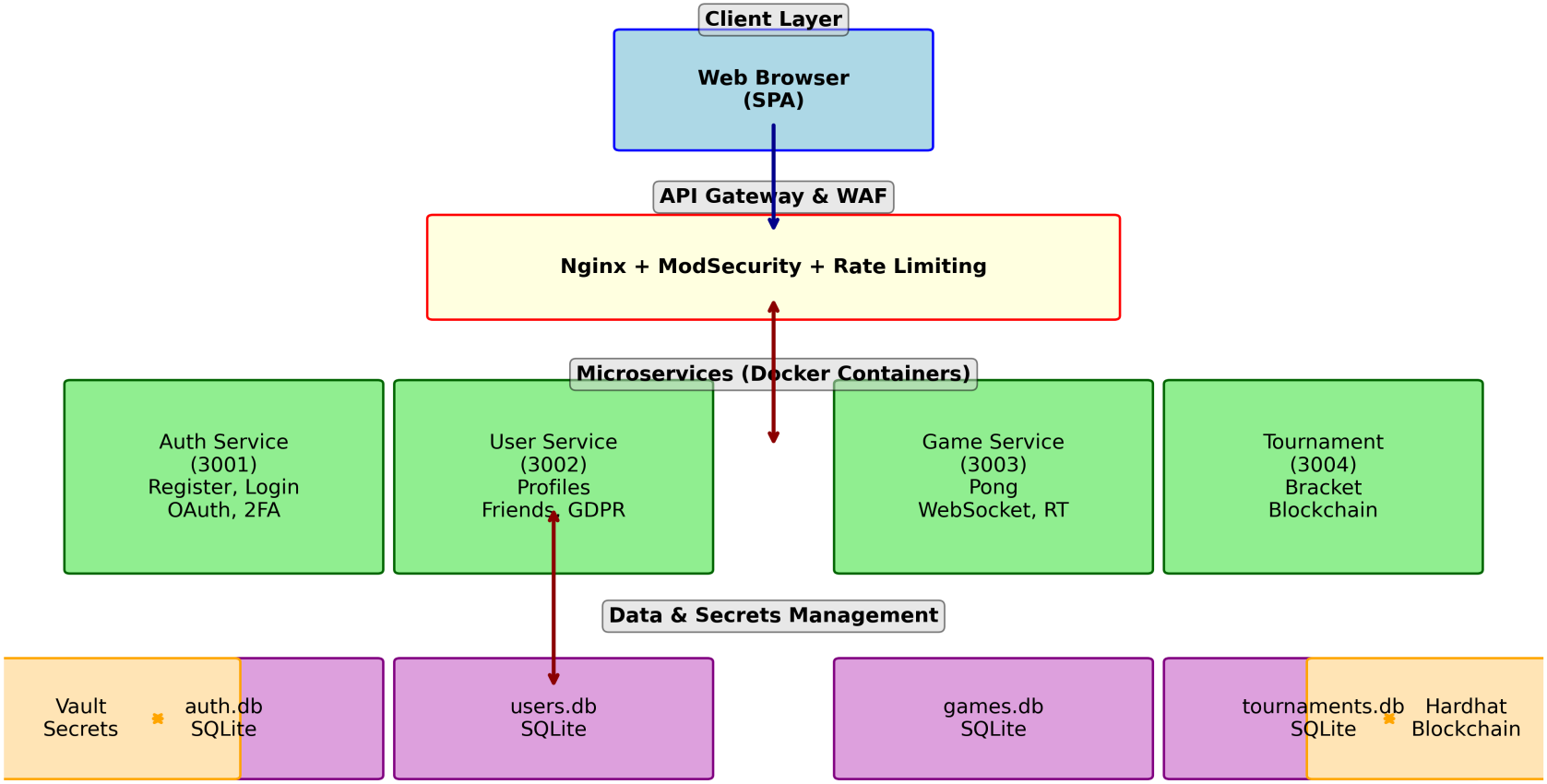
TR-21: 60 FPS game loop, ¡200ms API response

Risk Register

Table: Risk Register

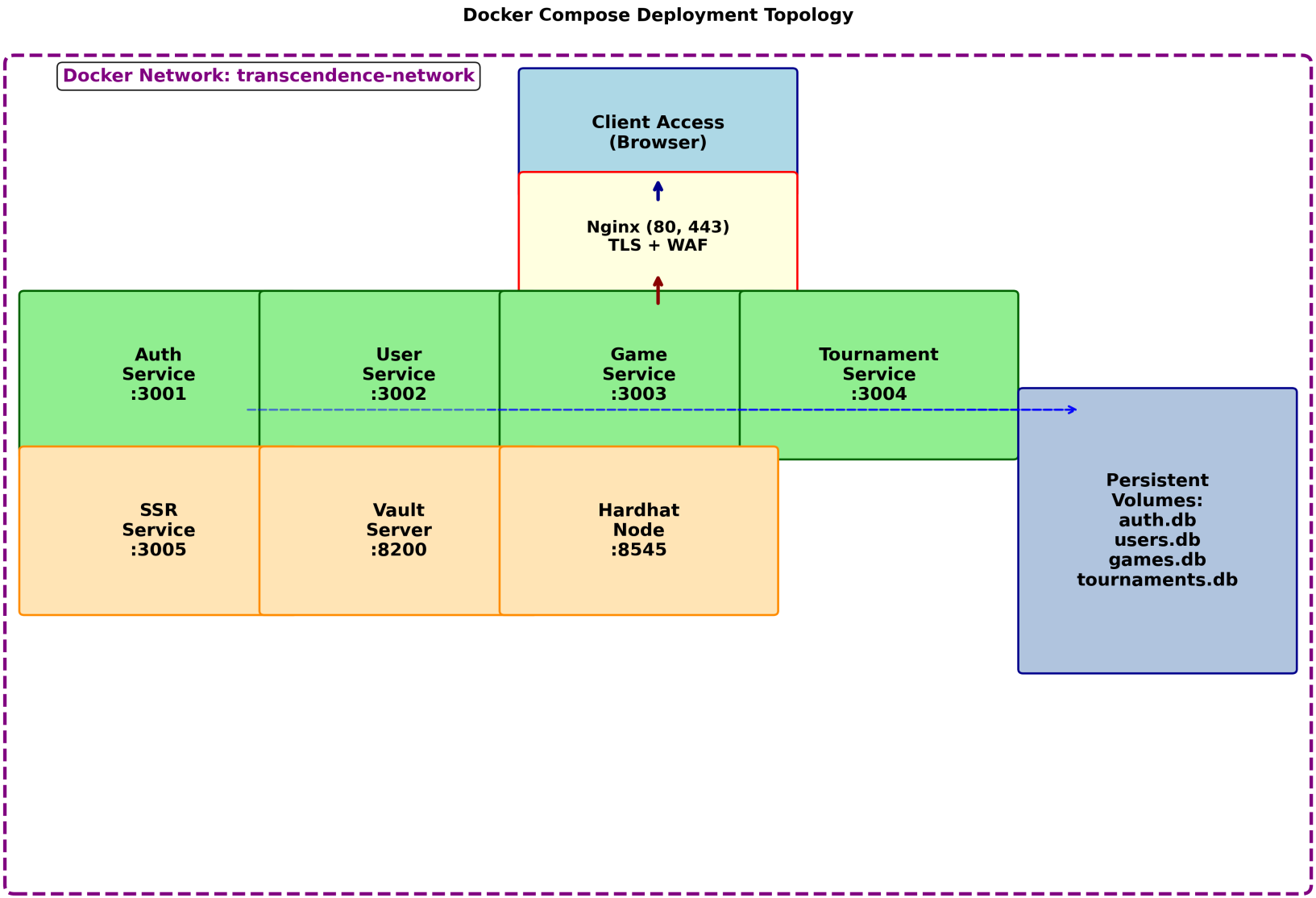
ID	Description	Likelihood	Impact	Severity
1	Server downtime	2	4	8
2	SQL injection	1	5	5
3	Data leak	2	4	8
4	OAuth downtime	3	3	9
5	Blockchain failure	1	4	4

System Architecture



ft_transcendence System Architecture

Deployment Topology



Microservices Overview

Service	Responsibilities	Port
Auth Service	Registration, login, password reset	3001
User Service	Profiles, friends, achievements	3002
Game Service	Real-time Pong, WebSocket	3003
Tournament Service	Tournament management, blockchain	3004
Nginx Gateway	TLS, routing, WAF	80/443
Vault	Secret storage	8200
Hardhat	Local blockchain	8545

Data Model

Auth DB: users, sessions

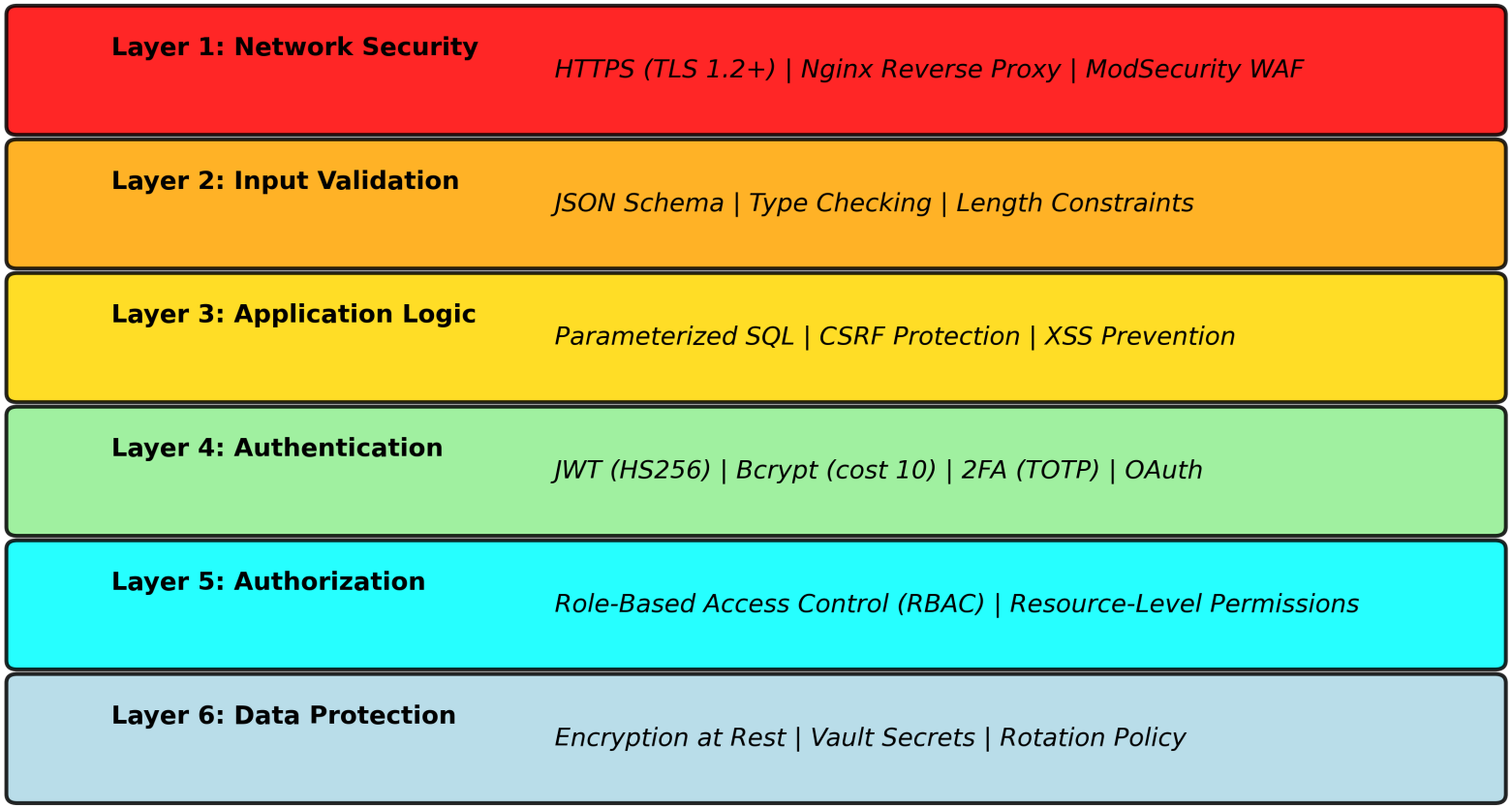
User DB: profiles, friendships, achievements, statistics

Game DB: matches, game sessions, match events

Tournament DB: tournaments, participants, bracket matches, blockchain records

Security Design

Security Architecture: Defense-in-Depth



Prevents: SQLi | XSS | CSRF | DDoS | Brute Force | Unauthorized Access | Data Breaches

Security Layers

Layer 1: Network Security (HTTPS, WAF, Rate Limiting)

Layer 2: Transport Security (mTLS, Session Security)

Layer 3: Application Security (Input Validation, SQL Injection Prevention)

Layer 4: Authentication & Authorization (Password Security, RBAC)

Layer 5: Data Protection (Vault, Database Security)

Layer 6: Monitoring & Logging

Layer 7: Incident Response

Blockchain Integration

Hardhat local network for development/testing

Solidity smart contracts for tournament rankings

Immutable result recording

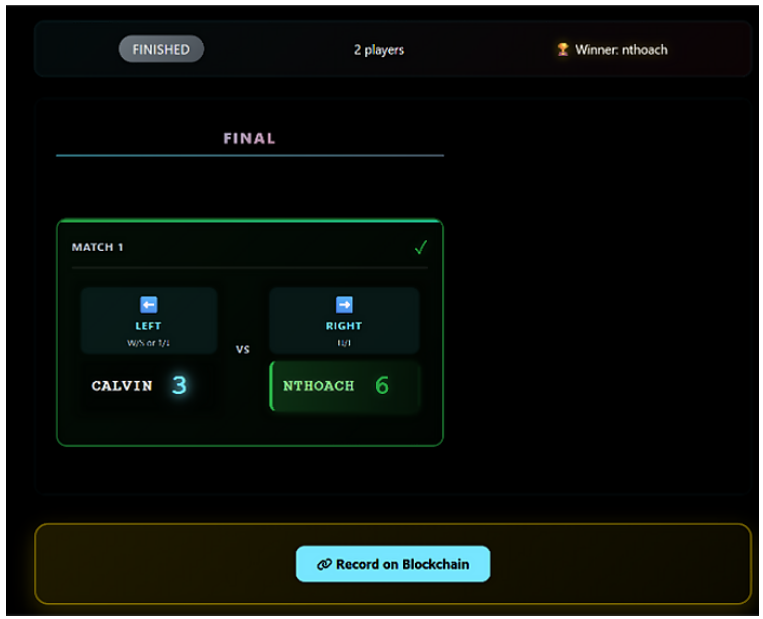


Figure: Blockchain Record Verification

Smart Contract

```
function recordRank(uint256 tournamentId, uint256 playerId, ui
rank) public  tournamentRankings[tournamentId][playerId] = ran
```

3D Frontend Implementation

Babylon.js for immersive 3D rendering

Singleton pattern with conditional initialization

Real-time 3D synchronization with WebSocket

Wireframes and UI

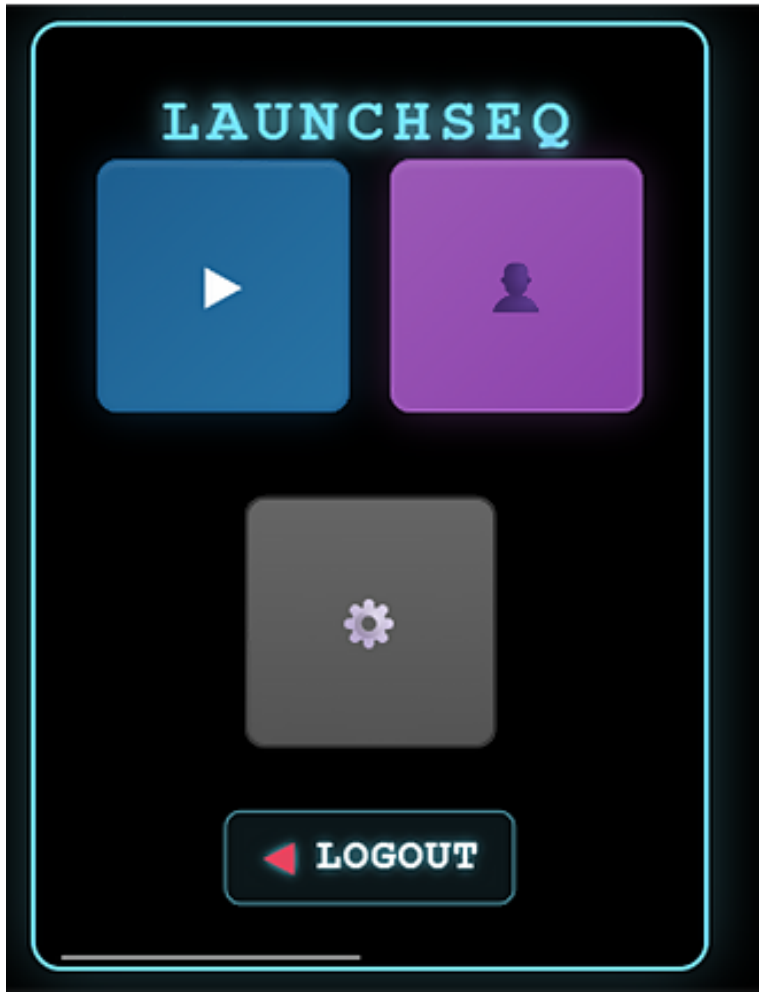


Figure: Main Menu Interface

Gameplay Interfaces

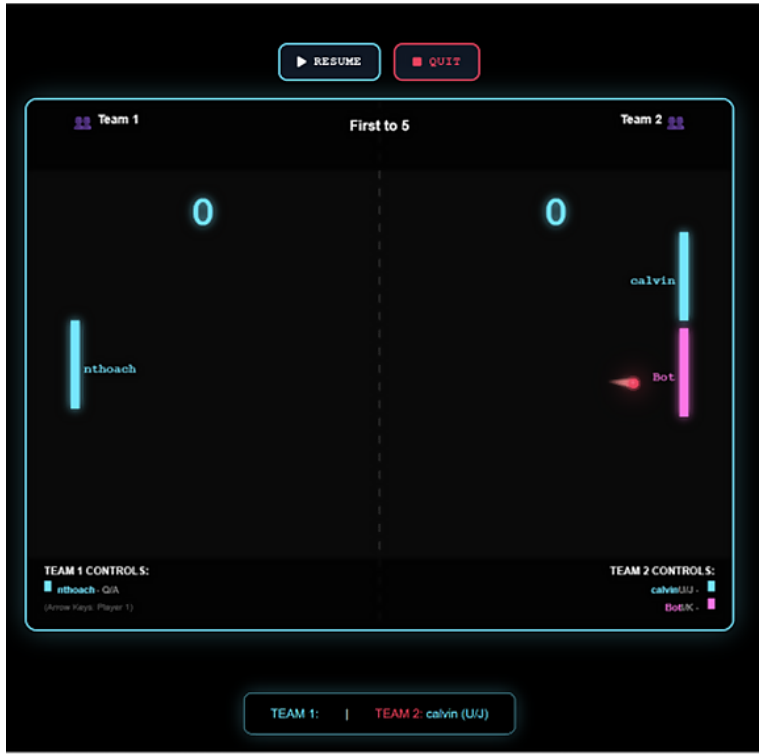
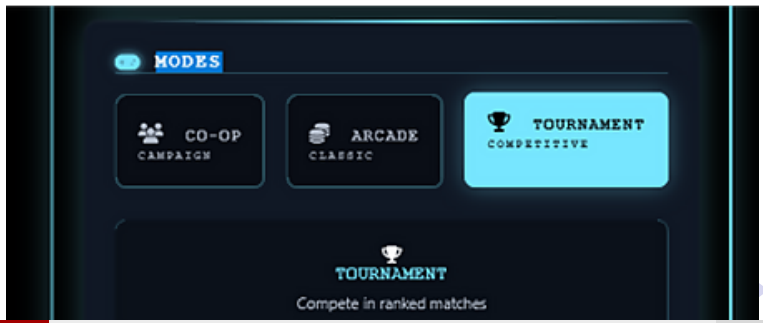


Figure: Arcade Multiplayer Mode



Technology Stack

Component	Technology	Version
Backend	Fastify + Node.js + TypeScript	4.29.1 / 20.19.20 / 5.9.
Database	SQLite 3	5.1.6
Frontend Build	Vite	5.0.8
Real-Time	WebSocket	(Fastify plugin)
Blockchain	Hardhat + Solidity	2.22.17
Secrets	HashiCorp Vault	1.15+

Backend Framework

Fastify v4 with TypeScript strict mode

4 microservices: auth, user, game, tournament

REST APIs and WebSocket support

Frontend Architecture

TypeScript SPA with component-based architecture

Service layer separation

Client-side routing for SPA navigation

Blockchain Implementation

Avalanche blockchain integration

Solidity smart contracts for tournament results

Hardhat for development and testing

Security Implementation

WAF/ModSecurity with OWASP CRS

HashiCorp Vault for secrets management

HTTPS with TLS 1.2+, mTLS between services

DevOps Implementation

Docker Compose orchestration (10 containers)

Nginx reverse proxy with load balancing

Health checks and monitoring

Test Results Summary

Comprehensive manual testing completed

All modules validated (100% coverage)

Security, blockchain, microservices tested

Authentication: registration, login, password reset

Gameplay: real-time synchronization, AI opponents

Security: WAF rules, Vault integration

Performance: responsiveness, concurrent users

Current State and Conclusion

Fully implemented and production-ready

All subject requirements achieved

Comprehensive manual testing completed

100% subject compliance

Modern software engineering practices

Production-grade security and scalability

Innovative 3D frontend with Babylon.js

ft_transcendence demonstrates mastery of modern software engineering

Ready for production deployment

Extensible architecture for future enhancements

Q&A

Thank you for your attention!
Questions?