

Software Requirements Specification (SRS)

Ares: Real-Time Anti-Cheat Detection Analytics System

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document defines the functional, non-functional, architectural, and data requirements for *Ares*, a real-time anti-cheat analytics platform built using Big Data technologies. The system detects cheating behavior in online multiplayer games by analyzing gameplay telemetry through data streaming, machine learning, and behavioral analytics.

1.2 Scope

Ares focuses on processing high-frequency gameplay events to identify unfair player behavior such as aimbots, no-recoil macros, abnormal movement, and reaction-time irregularities. The system consists of:

- Real-time data ingestion (Kafka)
- Distributed analytics (Spark Structured Streaming)
- Anomaly detection and ML models
- NoSQL storage (MongoDB)
- Backend API services
- Frontend monitoring dashboard

The system does **not** act as a kernel-level anti-cheat; instead, it functions as a *behavioral analytics layer* that complements existing anti-cheat mechanisms.

1.3 Definitions, Acronyms, and Abbreviations

- **Kafka** – Distributed messaging system used for data ingestion.
- **Spark Structured Streaming** – Real-time analytics engine.
- **Gameplay Telemetry** – Player actions and movements extracted from gameplay.
- **Anomaly Detection** – ML-based method for identifying abnormal behavior.
- **Cheat Score** – A numeric probability (0–100%) representing suspected cheating.

1.4 Overview

This SRS outlines requirements, system interactions, constraints, data flows, architecture, and UI expectations.

2. Overall Description

2.1 System Perspective

Ares operates as a cloud-based analytics service.

Gameplay data flows into Kafka → Spark processes it → MongoDB stores processed outputs → Backend serves analytics → UI displays insights.

2.2 Product Functions

Major system functions:

- Collect gameplay telemetry from multiple sources
- Analyze player behavior in real time
- Detect anomalies using ML
- Compute cheat probability scores
- Provide dashboards for monitoring
- Generate player-specific analytics
- Process both real and synthetic data

2.3 User Classes and Characteristics

1. System Administrator

Manages Spark jobs, Kafka clusters, database configuration, and system health.

2. Security Analyst

Monitors suspicious players, investigates events, reviews cheat scores.

3. Game Developer / Researcher

Uses analytics to improve game balance, anti-cheat strategies, and ML models.

2.4 Operating Environment

- Server environment (Linux-based)
- Kafka 3.x
- Spark 3.x
- MongoDB 6+
- Node.js or Scala backend
- React / Vue dashboard
- Containerization via Docker (optional)

2.5 Design and Implementation Constraints

- Telemetry must support high throughput (20k+ events/second).
- Latency for Spark processing < 1 second where possible.
- MongoDB must store millions of documents efficiently.
- UI must display real-time updates via WebSockets.

2.6 Assumptions and Dependencies

- Gameplay telemetry is available (via video extractor, mouse tracker, or generator).
 - Network connectivity between components is stable.
 - Hardware resources are sufficient for Big Data workloads.
-

3. System Features

3.1 Real-Time Telemetry Ingestion

Description:

The system receives gameplay events in JSON format through Kafka topics.

Inputs:

- Aim movements
- Hit/miss events
- Recoil vectors
- Movement coordinates

- Reaction-time samples
- System metrics (FPS, ping)

Outputs:

Raw events passed to Spark.

3.2 Spark Real-Time Analytics

Description:

Spark Structured Streaming processes data and computes metrics.

Functional Requirements:

1. Detect aim snapping behavior
 2. Detect no-recoil patterns
 3. Evaluate abnormal movement vectors
 4. Compute accuracy, reaction time, flick speed
 5. Aggregate statistics per player
 6. Generate suspicion events
-

3.3 Machine Learning Anomaly Detection

Models:

- Isolation Forest
- K-Means clustering
- Rule-based heuristics

Outputs:

- Cheat Probability Score (0–100%)
 - Event severity levels
-

3.4 MongoDB Storage

Stores processed analytics and historical data.

Collections include:

- players
 - events
 - anomalies
 - heatmaps
 - scores
-

3.5 Backend API

REST API for retrieving analytics and system information.

Endpoints include:

- /players
 - /players/:id
 - /players/:id/score
 - /events/recent
 - /events/suspicious
 - /analytics/heatmap/:playerId
 - /system/status
-

3.6 Frontend Dashboard

Real-time monitoring UI.

Features include:

- Live event stream
- Player profile pages
- Suspicious event list
- Heatmaps

- Time-series analytics
 - Video analysis results
 - System health
-

4. External Interface Requirements

4.1 User Interface

The dashboard must include:

- Dark theme UI
- Responsive layout
- Animated charts
- Filters & search
- Real-time event updates

4.2 Hardware Interface

No direct hardware interaction—mouse/video data provided externally.

4.3 Software Interface

- Kafka Producers
 - Spark Consumers
 - MongoDB driver
 - REST API client
 - Optional: WebSocket live feed
-

5. Non-Functional Requirements

5.1 Performance

- Must support minimum 20,000 events/sec.
- Average Spark processing delay < 1 second.

5.2 Reliability

- Kafka replication enabled
- Spark checkpointing
- MongoDB clustered or sharded

5.3 Security

- JWT authentication
- Encrypted API calls (HTTPS)
- Role-based access

5.4 Scalability

- Horizontal scaling for Kafka & Spark
- Distributed MongoDB

5.5 Usability

- Clean UI for analysts
 - Clear metrics and charts
-

6. System Architecture

Pipeline Overview:

[Data Source]

- Kafka (Ingestion)
 - Spark (Real-Time Analytics + ML)
 - MongoDB (Storage)
 - Backend API (Service Layer)
 - Dashboard UI (Visualization Layer)
-

7. Data Requirements

Data Types

- AimEvent
 - MovementEvent
 - HitEvent
 - DamageEvent
 - RecoilEvent
 - AnomalyEvent
 - PlayerSummary
-

8. Future Enhancements

- Deep learning models for cheat detection
 - Multi-game integration
 - Automated ban system
 - Player reputation scoring
 - Cross-game behavioral fingerprinting
-

9. Risks

- False positives due to noisy data
 - High computational cost
 - Limitations of simulated telemetry
 - Potential delays in Spark streaming
-

10. Conclusion

Ares provides a complete Big Data-powered behavioral anti-cheat analytics platform. It delivers scalable telemetry processing, real-time anomaly detection, machine learning insights, and a professional dashboard—making it suitable for academic, research, and industry-level deployments.