📄 Analysis Summary for AIMS Development

Since I cannot directly create a Google Doc, I will provide the analysis that generated the software specifications in a clear, structured format suitable for copying into a document. This summarizes the **business problem**, the **academic drivers**, and the **solution strategy** that led to the Firebase requirements.

🎯 Section 1: Business Problem & Key Drivers

The primary challenge in Major League Baseball (MLB) ad scheduling is **revenue volatility** stemming from two non-linear variables. This volatility creates a significant **Make-Good Liability**.

1. The Core Pain Points

Pain Point	Driver	Impact on Revenue
Inventory Loss	**Game Length Variability** (Pitch Clock)	Shorter games (e.g., $2:30$ average) eliminate commercial breaks, reducing the total available ad spots (inventory).
Demand Volatility	**Team Performance** (Outcome Uncertainty)	Viewership ratings surge when the team is winning (high demand), but plummet when the team is losing (low demand).
Financial Liability	**Make-Goods**	When the network fails to deliver promised

		impressions/spots (due to poor viewership or short games), they must give away future inventory for free, directly reducing future sellable revenue.

2. Academic/Economic Drivers

The design of the AIMS system is directly based on these established economic principles:

- **Uncertainty of Outcome Hypothesis (UOH):** Confirms that high viewer demand (and thus high ad prices) is driven by contests where the outcome is unpredictable (e.g., a tight pennant race).
- **Perishable Inventory:** An ad slot is a product that cannot be stored or resold once the time passes. This requires aggressive, dynamic pricing and strict control over supply (spots).

💡 Section 2: Solution Strategy and Mitigation

The software solution is designed to pivot the RSN from reactive reconciliation to proactive risk management.

1. Strategic Shifts

Old Strategy (Pain Point)	New Strategy (AIMS Requirement)	System Mitigation
Selling Time (Fixed Ad Breaks)	**Selling Impressions** (ECR Model)	**R3.7/R3.8:** Auditing and guaranteeing delivery based on **Impressions** (audience size) instead of a

		fixed number of **Spots** (inventory), making the contracts resilient to game length changes.
Reactive Auditing (Post-game Surprise)	**Proactive Forecasting** (Risk Index)	**R3.3:** The AI Prediction Engine flags high-risk games *before* they are played, allowing the RSN to prepare.
Giving Away Future Slots (High Cost)	**Reserving Low-Value Inventory** (Reserve Pool)	**R3.5:** A portion of predictable, low-demand inventory is automatically put into a **Make-Good Reserve Pool**, ensuring liabilities are covered with the cheapest possible inventory.

2. Mapping Drivers to Software Requirements

Volatility Driver	AIMS Module Responsible	Key Output (Requirement)
Demand Fluctuation	AI Prediction Engine	**R3.1:** Dynamic Price Floor API (Adjusts spot price in real-time).
Inventory Fluctuation	AI Prediction Engine	**R3.2:** Forecasted Game Duration (Predicts total spots available).
Make-Good Liability	Reconciliation Module	**R3.8:** Automated Liability Calculation (Quantifies the financial cost after the

		game).

🛠️ Section 3: Firebase Prototype Design Rationale

The requirements map perfectly to a Firebase/GCP architecture because it natively handles the necessary data types and processing requirements.

Software Requirement	Firebase/GCP Rationale
Data Structure (R3.7)	The Contracts and Ad_Delivery_Logs models require a flexible, high-read/high-write, real-time database.
Automated Logic (R3.1, R3.8)	Prediction, risk scoring, and post-game auditing must run on automated, scheduled triggers without server management.
User Interface (R3.5)	The sales team needs a secure, fast dashboard to check the **Reserve Pool** and **Risk Index**.
ML Model Hosting (R3.1/R3.2)	Complex predictive models need high-performance, scalable hosting separate from the standard database.

The resulting structure is a low-latency, scalable platform designed to monetize the inherent unpredictability of the MLB season.