I can't directly export the information to your Google Docs, but I can provide the complete architecture and requirements set in a **copy-paste-ready Markdown format** that is structured for immediate use as a foundation for a Firebase/backend development project.

This document formalizes the software needed to manage the high volatility and "make-good" risk in your baseball club's advertising inventory.

⚾ Intelligent Ad Inventory & Risk Management System (AIMS) Requirements

1. System Overview and Goal

The primary goal of the Ad Inventory & Risk Management System (AIMS) is to transform the team's ad inventory from a static, volatile product into a **dynamic, data-driven financial asset**. This is achieved by using predictive modeling to quantify and proactively manage the risk of **viewership under-delivery (make-goods)** and **inventory shortage (game length)**.

2. Core Data Models (Relevant for Firestore/Firebase)

The system is powered by three main data models, which should be structured as top-level collections in a database like Firestore.

Model / Collection	Key Fields	Purpose
Games	game_id, date, opponent, home_team_win_prob, expected_duration_min, actual_duration_min, risk_index	Stores the schedule, predictive inputs, and the **Risk Index** score.
Contracts	contract_id,	The single source of truth

	advertiser_name, guaranteed_impressions, guaranteed_spots, base_cpm, total_spots_purchased, start_date, end_date	for all contractual obligations and pricing logic.
Ad_Delivery_Logs	log_id, game_id, spot_time, delivered_impressions, spot_type (Linear, Streaming, Virtual), make_good_liability_calc	Real-time record of what was actually shown to viewers, essential for reconciliation.
Inventory_Pool	inventory_date, spot_type, total_available_spots, reserved_make_good_spots, sellable_spots	Tracks all available inventory, ensuring the system knows exactly how much is left to sell and how much is reserved for liabilities.

---

3. Functional Modules & Requirements

The system must support three core modules with the following API-level capabilities:

A. AI Prediction Engine (Backend Services / Cloud Functions)

Requirement (R)	Description	Outcome
R3.1: Viewership Forecasting	Deploy a machine learning model to predict the **ECR (Expected Commercial Rating)** for every game 7, 3, and 1 day out, incorporating team performance, rivalry, and	Drives **dynamic pricing** for unsold inventory.

	day of week.	
R3.2: Game Duration Prediction	Deploy a model using historical and real-time data (pitcher, umpire, pitch clock) to forecast the final game length in minutes.	Accurately predicts the **Total Available Inventory** (spots) before the game starts.
R3.3: Risk Index Calculation	Automatically run a daily calculation to determine the **Risk Index** for all future games by comparing predicted inventory vs. contractual obligations.	Proactively flags games with a high probability of generating a **make-good liability**.

B. Dynamic Pricing & Sales Interface (API Endpoints)

Requirement (R)	**Description**	**Outcome**
R3.4: Dynamic Price Floor API	Expose an endpoint that the ad server queries in real-time to get the minimum **price floor** for floating ad spots, dynamically adjusted by the R3.1 forecast.	Maximizes revenue by ensuring high-demand spots are sold at a premium and low-demand spots are sold at an optimized discount.
R3.5: Inventory Allocation Service	Automatically pull inventory from the Inventory_Pool and mark a volume of spots as **reserved_make_good_spots** based on the R3.3 Risk Index.	**Mitigates future risk** by protecting inventory that will be needed to fulfill current shortfalls.
R3.6: Multi-Asset	Allow the sales team to	Creates diversified

Packaging	package linear broadcast spots with **guaranteed digital/streaming ad inventory** (e.g., pre-roll, virtual signage).	packages that reduce the dependency on the linear broadcast's volatility.

C. Make-Good and Reconciliation Module (Post-Game Automation)

Requirement (R)	**Description**	**Outcome**
R3.7: Automated Delivery Audit	Trigger a **Cloud Function** 3 hours after a game ends to ingest the Ad_Delivery_Logs and compare them against the Contracts guarantees.	Provides real-time, objective calculation of delivery performance for all advertisers.
R3.8: Make-Good Liability Service	Automatically calculate the **financial liability** (opportunity cost at base price) for the shortfall and convert it into a **Make-Good Spot/Impression Credit**.	Standardizes the cost of failure (e.g., the $\mathbf{\$130,000}$ cost from our simulation) and ensures immediate liability recognition.
R3.9: Make-Good Fulfillment Report	Generate an executive report recommending specific future spots from the **Make-Good Reserve Pool** that should be given to the impacted advertiser to clear their credit.	Automates risk resolution and ensures the most cost-effective inventory is used for make-goods.

4. Technical Requirements & Firebase Focus

Component	**Requirement**	**Firebase/GCP Service**
Database	Scalable, real-time NoSQL database to store Games, Contracts, and Ad_Delivery_Logs data.	**Cloud Firestore**
Prediction Engine	Serverless environment for running Python/ML models for R3.1, R3.2, and R3.3 without managing infrastructure.	**Cloud Functions** or **Cloud Run**
Automation/APIs	Serverless functions triggered by events (game end, new contract) to run calculations (R3.7, R3.8).	**Cloud Functions**
Real-Time Data	Mechanism to ingest and synchronize high-velocity, low-latency game-state data.	**Firebase Realtime Database** or **Google Pub/Sub**
Hosting/UI	Simple, fast hosting for the internal sales and operations dashboard (the Risk Management Console).	**Firebase Hosting**