NBA Player Similarity & Explanation System: A Strategic Research and Engineering Blueprint (2026)

0) Research Protocol

To engineer a system that transcends the limitations of traditional "nearest neighbor" searches based on raw box-score statistics, a rigorous review of academic and industrial literature is required. The current state of the art in 2026 demands a shift from observation-based similarity (what a player *did*) to representation-based similarity (who a player *is* in a latent style space). The following research protocol establishes the theoretical foundation for the proposed system, synthesizing insights from deep learning, sports analytics, and interpretable AI.

foundational Literature & Feature Extraction Insights

1. "NBA2Vec: Dense Feature Representations of NBA Players" (Guan et al., 2018/2023) 1
This seminal work provides the primary justification for moving beyond aggregate statistics. Guan et al. demonstrate that traditional metrics (points, rebounds, assists) fail to capture the contextual nuances of a player's role, such as "gravity" or "spacing," which are inherently relational. By training a neural network to predict the outcome of a possession given the ten players on the court, the authors generated embeddings that encode these latent traits.

- **Critical Insight:** Similarity must be derived from *context* rather than just *output*. A player's value and style are defined by how they alter the probabilities of specific play outcomes (e.g., a corner three vs. a rim attempt) for their teammates.
- **Implication for Blueprint:** We cannot rely solely on PlayerGameLog. We must engineer features that act as proxies for lineup context and play-outcome probabilities, or explicitly train an embedding layer similar to the Word2Vec architecture described.

2. "Using Deep Learning to Understand Patterns of Player Movement" (Sloan Sports Analytics Conference) 4
This research highlights the superiority of trajectory embeddings over box-score events. The authors found that Euclidean distance between trajectory embeddings accurately reflects visual similarity in play style (e.g., distinguishing a post-up from a screen action), whereas box scores conflate these actions if the outcome (e.g., 2 points) is the same.

- **Critical Insight:** Geometry dictates role. Two players may both average 20 points, but one scores via isolation at the elbow and the other via catch-and-shoot threes. True similarity requires tracking data proxies.
- **Implication for Blueprint:** Since we are constrained to nba_api and lack raw optical tracking frames, we must heavily leverage the PlayerDashPt... endpoints (tracking

proxies) to capture speed, distance traveled, and touch time, which serve as low-resolution trajectory embeddings.

3. "Beyond the Box Score: Using Psychological Metrics to Forecast NBA Success" 5
While primarily focused on draft forecasting, this paper establishes that "intangibles" and consistency metrics significantly boost predictive accuracy (from 63% to 87%).

- **Critical Insight:** Variance is a feature. The "clutch" performance and consistency of a player (variance in game-to-game output) are distinct stylistic traits.
- **Implication for Blueprint:** We must include features representing the *distribution* of performance (e.g., standard deviation of Game Score, performance in "Clutch" defined segments) to distinguish steady veterans from high-variance "heat check" players.

4. "NBA Lineup Analysis on Clustered Player Tendencies" 6
This source argues against rigid positional designations (PG, SG, SF) in favor of "soft" probabilistic assignments to archetypes.

- **Critical Insight:** Players are mixtures. A modern wing might be 60% "Spot-up Shooter" and 40% "Secondary Creator." Hard clustering fails to capture this duality.
- **Implication for Blueprint:** The similarity metric should arguably handle "mixed membership." If we use clustering, we should look at Soft K-Means or Gaussian Mixture Models (GMM) to allow for these nuances.[7]

5. "SAINT: Improved Neural Networks for Tabular Data" (Somepalli et al., 2021) 8
As we are dealing with tabular data (stat sheets), this paper is critical for model selection. It demonstrates that Transformer-based architectures utilizing attention over both rows (samples/players) and columns (features) outperform standard Multi-Layer Perceptrons (MLPs) and often Gradient Boosted Decision Trees (GBDTs) for tabular tasks.

- **Critical Insight:** Attention mechanisms allows the model to dynamically weight features per player. For a center, "Assists" might be noise; for a Nikola Jokić type, it is the defining signal.
- **Implication for Blueprint:** If we implement a deep learning-based similarity engine (Tier 2/3), a Transformer encoder (like TabTransformer or SAINT) is the architecture of choice over a simple Autoencoder.

6. "Contrastive Learning for Tabular Data" (Various Sources) 11
These sources discuss the application of self-supervised contrastive learning (e.g., SimCLR, SupCon) to tabular domains. The core challenge identified is the generation of "positive pairs" without image augmentations (cropping/rotating).

- **Critical Insight:** Effective representation learning on unlabeled tabular data requires creative augmentation, such as feature masking (SubTab) or noise injection, to create "views" of the same player that the model learns to pull together.
- **Implication for Blueprint:** This offers a path to a "Stretch" goal: training a similarity metric without explicit labels by treating a player's stats in Game N and Game $N+1$ (or Season N and $N+1$) as positive pairs.[15]

---

1) Executive Verdict (Go/No-Go)

Verdict: GO.

The Strategic "Why":
This project is not merely viable; it is a necessity for the modern sports analytics portfolio. The era of "Moneyball" (finding undervalued aggregate stats) has passed. The current frontier, as of 2026, is Contextual Intelligence—understanding how production is achieved and where a player fits within a system. A system that can rigorously answer "Who plays like Player X?" using tracking-derived style vectors rather than output-derived box scores addresses the primary pain point of Front Offices: Role Fit.
Differentiation Factor:
Most data science portfolios feature projects that predict outcomes (e.g., "Predicting MVP votes" or "Forecasting Game Winners"). These are often black-box regression tasks. This project builds a Recommendation Engine for talent. It demonstrates proficiency in:

1. **Complex Data Engineering:** Wrangling the notoriously difficult nba_api and normalizing across eras.
2. **Unsupervised Learning:** Deriving structure (archetypes) from unlabeled data.
3. **Explainable AI (XAI):** Implementing SHAP to decode the "black box" of similarity, a crucial skill for winning trust from non-technical stakeholders (coaches/GMs).

The Killer Demo:
The "Trade Machine" Scenario.

- **Input:** The user selects a high-salary star (e.g., Luka Dončić).
- **Constraint:** The user sets a salary cap filter (< $15M).
- **Output:** The system returns "Budget Lukas"—players who map to the same region of the latent style space (high heliocentric usage, high unassisted 3-point rate, high potential assists) but operate on lower volume or minutes.
- **The Hook:** The explanation engine explicitly states: *"Similarity is driven by 95th percentile Touch Time and 80th percentile Step-Back Frequency, despite a 10 PPG difference."* This proves the system sees *style*, not just points.

---

2) Differentiation Strategy (Anti-Resume-Project Filter)

To survive the automated filters and the scrutiny of a technical hiring manager in 2026, this project must actively reject the tropes of "beginner" sports analytics projects.

- **Rejection of the "Box Score KNN":** We will explicitly document *why* we do not use K-Nearest Neighbors on raw per-game stats. We will cite the **Curse of Dimensionality** (distance becomes meaningless in high-dimensional space) and **Context Collapse** (10

assists from PnR is different from 10 assists from DHOs) as reasons for adopting Representation Learning.[7]

- **Evidence-Based Feature Selection:** We will not "guess" features. Every feature included in the final vector will be backed by a specific citation or a calculated feature importance metric (Variance Thresholding / SHAP). We will avoid redundant multicollinear features (e.g., including both FGM and PTS) that distort distance metrics.
- **Temporal Integrity:** We will implement strict "Time Machine" capabilities. Similarity searches will allow users to query "Find players similar to 2018 James Harden *using only data known in 2018*." This demonstrates a mature understanding of data leakage prevention.[17]
- **Robust Engineering:** We will not treat nba_api as a reliable service. The architecture will include a dedicated ingestion layer with caching, exponential backoff, and header masquerading to handle the NBA's aggressive IP blocking and rate limiting.[18]
- **Explainability as a First-Class Citizen:** We will not output a naked list of names. We will integrate a SHAP-based explainer that generates natural language justifications for every comparison, bridging the gap between "Model Output" and "Scouting Report".[19]

---

3) Feature Research → Final Feature Blueprint

The "Secret Sauce" of this system is the input vector. If the input is garbage (raw box scores), the output is garbage (spurious correlations). We must construct a "Style Vector" that captures the *geometry* and *physics* of a player's game.

3.1 Feature Taxonomy + Research-Backed Rationale

Category	Rationale & Research Backing	Pitfalls & Leakage Risks
Shot Geometry	[21] proves that clustering by shot location (Rim, Mid, 3PT) and type (Pull-up vs. Catch-and-Shoot) defines archetypes better than position.	**Risk:** Using raw makes/misses conflates "skill" with "style." **Solution:** Use *Frequencies* (% of shots taken at Rim) to capture intent, and separate *Efficiency* (FG% at Rim) as a quality metric.
Playmaking Texture	[22] identifies "Ball Handling" as a principal component. Usage Rate is insufficient;	**Risk:** Raw Assists are system-dependent. **Solution:** Use AST%

	AvgSecPerTouch and DribblesPerTouch distinguish "Heliocentric" stars from "Connectors."	(individual contribution relative to team) and PotentialAssists to measure passing *intent* regardless of teammate conversion.
Defensive Impact	Box score defense (STL/BLK) is notoriously noisy. [23] suggests DefenseHub, but LeagueDashPtDefend (Rim protection FG% diff) is the gold standard for interior defense.	**Risk:** Defensive Rating (DRtg) is a noisy team stat. **Solution:** Use DFG% at rim vs. expected and HustleStats (deflections, loose balls) as "activity" proxies.
Physical Context	Height/Weight/Wingspan define the *constraints* of a player's role. A 6'3" player with high rebounds is stylistically different from a 7'0" player with high rebounds.	**Risk:** Raw values ignore era shifts. **Solution:** Z-score physicals relative to the league average for that season to normalize for "small ball" eras.
Offensive Role	[6] suggests roles are mixtures of "On-Ball" and "Off-Ball" tendencies. PctAstFG (Percentage of shots assisted) is the cleanest proxy for this.	**Risk:** Redundancy with Usage. **Solution:** Ensure PctAstFG is treated as a distinct "dependency" metric (how much do they need a creator?).

3.2 The “Final Target Feature Set”

The following table defines the $\mathbf{x}$ vector for our model. We target a core set of ~40-50 high-signal features derived from aggregating specific nba_api endpoints.

Note: All "Per Game" stats are normalized per 75 possessions where possible, or Z-scored per season.

Feature Name	Definition / Formula	Source Endpoint	Why it's in the Final Set
Geometry (Shot			

Profile)			
freq_rim	FGA within 5ft / Total FGA	PlayerDashPtShots	Defines "Rim Pressure" style.
freq_mid	FGA 8-16ft / Total FGA	PlayerDashPtShots	Identifies "Mid-range Masters" (e.g., DeRozan).
freq_3pt	3PA / Total FGA	PlayerGameLog	Fundamental spacing metric.
freq_corner_3pt	Corner 3PA / Total 3PA	PlayerDashPtShots	Separates "Role Spacers" from "Above Break" creators.
avg_shot_dist	Average shot distance (ft)	PlayerDashPtShots	Summary metric for offensive range.
pct_ast_2pt	% of 2FGM assisted	PlayerDashboardByGeneralSplits	Crucial: Separates "Creators" from "Finishers".
pct_ast_3pt	% of 3FGM assisted	PlayerDashboardByGeneralSplits	Distinguishes "Pull-up Shooters" from "Catch-and-Shoot".
Touch & Playmaking			
usg_pct	Usage Percentage	PlayerAdvancedStats	The volume of offense a player terminates.
avg_sec_touch	Avg seconds ball held per touch	PlayerDashPtPass	**Elite Signal:** High values =

			Heliocentric (Luka/Trae).
avg_drib_touch	Avg dribbles per touch	PlayerDashPtPass	Correlates with isolation/PnR frequency.
ast_ratio	Assists per 100 possessions	PlayerAdvancedStats	Normalized playmaking volume.
potential_ast	Potential Assists / Pass	PlayerDashPtPass	Captures "Vision" and "Intent" better than raw assists.
pass_to_ast_conv	Assists / Potential Assists	PlayerDashPtPass	Proxy for "Pass Quality" or Teammate Quality (Context).
Defense & Hustle			
def_rim_freq	DFGA at Rim / Minutes	LeagueDashPtDefend	Volume of rim protection duties.
def_rim_fg_pct	Opponent FG% at Rim (<5ft)	LeagueDashPtDefend	Effectiveness of rim protection.
loose_ball_rec	Loose balls recovered / 36m	LeagueHustleStatsPlayer	"Hustle" proxy; identifies "energy" players.
deflections	Deflections / 36m	LeagueHustleStatsPlayer	Proxy for active hands/passing lane disruption.
pct_box_out	Box Outs / Minute	LeagueHustleStatsPlayer	Measures rebounding *effort* vs. just grabbing the ball.

Physical & Bio			
height_z	Z-score of Height (League)	CommonPlayerInfo	Physical archetype anchor.
weight_z	Z-score of Weight (League)	CommonPlayerInfo	Physical archetype anchor.
Efficiency (Quality)			
ts_pct	True Shooting %	PlayerAdvancedStats	Scoring efficiency (points per shooting possession).
efg_pct	Effective Field Goal %	PlayerAdvancedStats	Shooting efficiency adjusted for 3PT value.
ast_tov	Assist / Turnover Ratio	PlayerAdvancedStats	Decision-making quality / Role safety.

Feature Tiers:

- **Core (MVP):** freq_3pt, usg_pct, ts_pct, ast_pct, trb_pct, blk_pct, stl_pct, height_z. (Available in standard box scores).
- **Plus (Strong):** avg_sec_touch, pct_ast_2pt/3pt, def_rim_fg_pct, freq_rim/mid, deflections. (Requires tracking endpoints).
- **Optional (Stretch):** speed_dist (average speed), dist_miles_off/def (distance traveled). (Often noisy or correlated with minutes).

3.3 Ablation & Selection Plan

To ensure we are not "wasting time engineering features that won't improve results," we will implement a rigorous ablation protocol:

1. **Metric: Stability Score**. If Player A and Player B are "Similar" in Year N, their production/style in Year $N+1$ should remain correlated. We quantify this by measuring the rank correlation of the similarity lists between years.

2. **Baseline:** Feature Set = {PPG, RPG, APG}.
3. **Experiment Loop:**
    - Add **Shooting Geometry** → Measure change in Stability Score.
    - Add **Touch Data** → Measure change in Stability Score.
    - Add **Hustle Stats** → Measure change in Stability Score.
4. **Stop Rule:** If adding a category improves the Stability Score by $< 1\%$ or decreases the Silhouette Coefficient (cluster separation), the feature set is rejected as "Noise." This prevents feature creep.

---

4) Data Plan (nba_api endpoints + schemas)

The nba_api is a powerful but fragile tool. It is not an official, SLA-backed product. Our data plan must be defensive, assuming failure and blocking.

4.1 Endpoints Mapping & Implementation Details

Endpoint Class	Data Points	Join Key	Reliability	Refresh Strategy
CommonPlayerInfo	HEIGHT, WEIGHT, DRAFT_YEAR	PERSON_ID	High	Monthly (Roster updates)
PlayerCareerStats	GP, MIN, PTS, REB, AST, STL, BLK	PLAYER_ID	High	Weekly
PlayerDashPtShots	FGA_FREQUENCY (by zone), FG_PCT (by zone)	PLAYER_ID	**Low** [18]	Weekly (Heavily cached)
LeagueDashPtDefend	DFG_PCT (<5ft), FREQ	PLAYER_ID	Medium	Weekly
PlayerDashPtPass	AVG_SEC_PER_TOUCH, POTENTIAL_AST	PLAYER_ID	Medium	Weekly

LeagueHustleStatsPlayer	DEFLECTIONS, LOOSE_BALLS_RECOVERED	PLAYER_ID	High	Weekly

Implementation Criticalities:

- **Headers:** You *must* include specific headers to mimic a browser, or the NBA CDN (Akamai/Cloudflare) will return 403 Forbidden.
  - User-Agent: Mozilla/5.0...
  - Referer: https://stats.nba.com/
  - Origin: https://stats.nba.com.[18]
- **Rate Limiting:** The API allows approximately 1 request per second. We will implement a strict client-side rate limiter using a time.sleep(1.2) between calls and an exponential backoff decorator for retries (up to 5 attempts).
- **Missing Data:** PlayerDashPtShots often returns empty sets for low-volume players. We will use **Iterative Imputer** (from sklearn) to fill missing tracking data based on their box score stats, or simply drop players with < 500 minutes played to maintain model integrity.

4.2 Known Gaps & Supplements

- **Synergy Play-Types:** While [24] suggests Synergy data is accessible, it often requires authenticated "Insight" accounts or is inconsistent in public endpoints. To satisfy the "Free + Trusted" constraint, we will **not** rely on Synergy. Instead, we use PlayerDashPtShots as a proxy (e.g., High "Pull-Up" frequency $\approx$ PnR Ball Handler).
- **Defensive Matchups:** Public data does not reliably track "who guarded whom." We accept LeagueDashPtDefend (rim protection) and Deflections (perimeter activity) as the best available proxies.

---

5) Problem Formulation Options

We will rigorously define the atomic unit of our system as the **Player-Season**.

- **Choice: Player-Season Similarity** (e.g., comparing LeBron James_2013 to Luka Doncic_2024).
- **Rationale:** "Career" similarity averages out evolution (LeBron changed from slasher to post-up to playmaker). "Role" similarity is too abstract. Player-Season captures the specific functional capability of a player at a point in time.
- **Normalization Strategy (Crucial):**
  - We cannot compare raw stats across eras (pace inflation).
  - **Method:** For every feature f in Season Y, we calculate the Z-Score: $z_{f,p,Y} = \frac{x_{f,p,Y} - \mu_{f,Y}}{\sigma_{f,Y}}$.
  - This converts all features into "Standard Deviations above/below League Average" for

that specific year. A usage rate of 35% in 2004 (Iverson) becomes comparable to a usage rate of 35% in 2024 (Luka) primarily by their relative distance from the mean.

---

6) Method Shortlist (3 Tiers)

To impress a recruiter, we must show we know *when* to use Deep Learning and when to use Linear Algebra. We propose a tiered rollout.

Tier 1: MVP (Strong Baseline) - Weighted Cosine Similarity

- **Concept:** Standardized Euclidean distance or Cosine Similarity on the Z-Scored Feature Vector.
- **Why:** It is interpretable, fast, and requires no training.
- **Refinement:** We apply **Domain Weights**. Not all features are equal. We will multiply the Shooting Geometry and Touch Time feature groups by a scalar (e.g., 1.5x) to prioritize "Style" features over generic "Value" features like Rebounding Rate (unless the user specifically filters for Bigs).
- **Math:** $Sim(A, B) = \frac{\sum w_i (A_i B_i)}{||A||_w ||B||_w}$

Tier 2: Strong (Recommended) - Variational Autoencoder (VAE)

- **Concept:** Train a VAE to compress the 50-dimensional feature vector into a lower-dimensional "Latent Style Space" (e.g., 10 dimensions).[1]
- **Why:** The "Curse of Dimensionality" renders distance metrics less meaningful in 50-D space. A VAE learns non-linear correlations (e.g., "High Assists" + "High Rebounds" = "Point Center") and compresses them into a dense vector.
- **Implementation:**
  - **Input:** 50-dim Z-scored vector.
  - **Encoder:** 2 layers (Relu), outputs μ and σ for latent space.
  - **Loss:** Reconstruction Loss (MSE) + KL Divergence (to ensure the latent space is continuous and searchable).
  - **Inference:** Distance is calculated in the 10-D Latent Space.
- **Win vs. Baseline:** It denoises the data. Two players might have slightly different "stats" but identical "roles"; the VAE maps them to the same point in latent space.

Tier 3: Stretch - Contrastive Learning (Tabular SimCLR)

- **Concept:** Self-supervised learning. We train a network to maximize the similarity between "positive pairs" and minimize it for "negative pairs".[11]
- **Positive Pair Generation (The Challenge):** How do we create two views of the same player without images?
  - SubTab Method [15]: We take the feature vector $\mathbf{x}$ and create two subsets $\mathbf{x}_1$ (masking 20% of columns) and $\mathbf{x}_2$ (masking a different

20%). The model must learn that $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the same player.
    - **Temporal Method:** We treat Player X_Season N and Player X_Season N+1 as a positive pair (assuming roles are relatively stable), forcing the model to learn time-invariant stylistic traits.
- **Risk:** High complexity. Only recommended if Tier 2 fails to separate distinct archetypes (e.g., if it confuses "Rim Runners" with "Post Scorers").

---

7) Evaluation Plan (Quantifiable & Resume-Grade)

We must prove our system works. "It looks right" is not an engineering metric.

7.1 Quantitative Metrics

- **Stability Score (Consistency Check):**
    - For every player in Year N, retrieve the Top 5 Similar Players.
    - Calculate the spread (variance) of their *future* performance in Year $N+1$.
    - *Hypothesis:* A good similarity model finds players whose future trajectories are clustered, not random.[17]
- **Silhouette Coefficient (Cluster Purity):**
    - Apply K-Means to our embeddings to generate clusters (e.g., $K=12$).
    - Calculate the Silhouette Coefficient. A higher score (>0.4) indicates that players within a cluster are tightly grouped and well-separated from other clusters.[25]
- **Retrieval Precision @ K (Pseudo-Labels):**
    - Create a "Golden Set" of undisputed comparisons (e.g., "Kobe Bryant 2008" $\approx$ "Michael Jordan 1996").
    - Check if the model retrieves the target within the Top $K=5$ results.

7.2 Robustness Tests

- **Era Shift Check:** Does the model match a 1996 Center (traditional post-up) with a 2024 Center (spacer)? If so, is it justified by stats, or is it an artifact of normalization? We need to ensure Z-Scoring is working.
- **Minutes Threshold:** We will test stability when filtering for players < 500 minutes vs > 2000 minutes to ensure low-sample size noise doesn't break the topology.

---

8) Explanation UX (“Transparent, Evidence-Based Reasons”)

The goal is to answer **"Why?"**. We will use **SHAP (SHapley Additive exPlanations)** values,

which provide a game-theoretic measure of feature importance for every single prediction.[20]

Designed Outputs:

1. **The "Similarity Driver" Plot:**
   - A waterfall chart showing the top 5 features contributing to the similarity score.
   - *Example:* "Similarity driven by: **Usage Rate (+0.15)**, **Avg Sec/Touch (+0.12)**, and **3PT Frequency (+0.08)**."
2. **The "Difference Maker" Plot:**
   - Identifying where they diverge.
   - *Example:* "Key Difference: Player B has significantly higher **Rebound Rate (-0.05)**."
3. **Role Cards (Radar Charts):**
   - A visual overlay of the two players on 5 axes: *Scoring Volume, Shooting Efficiency, Playmaking, Rim Protection, Perimeter Defense*.
4. **Natural Language Summary:**
   - Template-based generation using the SHAP values.
   - *"Player X is most similar to Player Y because both are **High-Usage Ball Handlers** with **Elite Assist Rates**, although Player Y is a more effective **Rim Protector**."*

---

9) Optional AI Agent (Grounded Q&A)

To modernize the project for 2026, we introduce a **Hybrid RAG (Retrieval-Augmented Generation)** agent.[27]

- **Architecture:**
  - **Structured Retrieval (Text-to-SQL):** For questions like "Who has the highest usage?", the agent converts natural language to a Pandas/SQL query against the clean dataframe.[29]
  - **Unstructured Retrieval (Vector):** For "Who plays like Luka?", it queries the pre-computed similarity matrix.
- **Guardrails:**
  - The LLM (e.g., gpt-4o-mini or local Llama-3) is *never* allowed to generate stats from its internal weights. It acts *only* as a translator and summarizer of the retrieved data.
  - If the query asks for "Current stats," and the database is from yesterday, the agent cites the "Last Updated" timestamp.
- **Example Interaction:**
  - *User:* "Find me a budget version of Trae Young."
  - *Agent Logic:* Query Similarity Engine -> Filter by Salary < $15M -> Sort by Similarity.
  - *Response:* "Based on 2025 shooting and playmaking profiles, **Tre Jones** represents a 'budget' option. He shares a similar **Assist-to-Pass Ratio** and **Floater Frequency**, though with lower **Usage**."

---

10) Full PRD (Product Requirements Document)

User Stories

- **Story 1 (The Scout):** "As a scout, I want to find G-League players who match the *movement profile* (speed, distance, touch time) of our outgoing backup PG, so I can fill the role seamlessly."
- **Story 2 (The GM):** "As a GM, I want to see the historical trajectory of players similar to my star at age 28, so I can predict his decline curve for contract negotiations."
- **Story 3 (The Fan):** "As a fan, I want to prove my friend wrong by showing that Player A is statistically identical to Player B despite different narratives."

Functional Requirements

- **Search:** Fuzzy string matching for player names (e.g., "Giannis" -> "Giannis Antetokounmpo").
- **Filters:** Position (Guard/Wing/Big), Minutes Played (>500), Season (1996-2025).
- **Latency:** Similarity inference must complete in < 200ms (pre-computed).
- **Data Freshness:** Data must be refreshed weekly during the season.

API Design (OpenAPI Spec Draft)

- GET /players/search?q={query}: Returns matching Player IDs.
- GET /similarity/{player_id}:
  - Params: season, top_k, filters (dict).
  - Returns: List of {player_id, similarity_score, common_cluster}.
- GET /explain/{player_id_1}/{player_id_2}:
  - Returns: {shap_values, top_factors_positive, top_factors_negative, natural_language_summary}.

Frontend Screens (Wireframe)

1. **Landing:** Search Bar + "Trending Comparisons".
2. **Comparison View:**
   - Left/Right Player Cards (Headshot, bio).
   - Central "Similarity Meter" (0-100%).
   - "Why?" Section (SHAP Bar Chart).
   - "Role Radar" (Overlaid charts).
3. **Chat Interface:** A floating drawer for the AI Agent Q&A.

---

11) Tech Stack & Repo Structure

Stack Choices:

- **Language:** Python 3.10+ (Standard for ML).
- **Data Processing:** Polars or DuckDB. (Faster than Pandas for tabular aggregations).
- **ML Framework:** PyTorch (for VAE/Contrastive) + scikit-learn (for Pipelines/Scaling).
- **Explainability:** shap.
- **API:** FastAPI (Async, auto-docs).
- **Frontend:** Streamlit. (We choose Streamlit over React to minimize "Frontend Engineering" time and maximize "Data Science" display time. It handles dataframes natively).
- **Deployment:** Docker container on a cheap VPS (DigitalOcean/Hetzner) to avoid AWS IP blocking issues.

Repo Structure (Cookiecutter Data Science Standard [30]):

```
nba-style-engine/
├── data/
│   ├── raw/              # Immutable JSON from nba_api
│   ├── processed/        # Parquet files (cleaned features)
│   └── external/         # Manual mappings (Team IDs, etc.)
├── src/
│   ├── data/
│   │   ├── fetcher.py    # Rate-limited API wrapper (The "Scraper")
│   │   └── processing.py  # JSON -> Polars DataFrame logic
│   ├── features/
│   │   └── engineering.py  # Z-scoring, Ratios, Imputation
│   ├── models/
│   │   ├── vae.py        # PyTorch Autoencoder definition
│   │   └── similarity.py  # Inference logic (Distance calc)
│   └── explanation/
│   │   └── explainer.py   # SHAP logic
├── api/
│   └── main.py           # FastAPI endpoints
├── app/
│   └── dashboard.py      # Streamlit frontend
├── notebooks/            # Experimental ablations
├── Makefile              # Automation commands
└── pyproject.toml        # Dependencies
```

---

12) Timeline (4 Weeks)

- **Week 1: Data Infrastructure (The Hardest Part).**
  - Build fetcher.py with robust caching and header masquerading.
  - Script the historical backfill (2000-2025).
  - Handle the PlayerDashPtShots 403 errors.
  - *Deliverable:* data/processed/master_features.parquet.
- **Week 2: Modeling & Features.**
  - Implement the Z-Score normalization pipeline.
  - Train the Tier 2 VAE Model.
  - Run the "Stability Score" ablation test to finalize feature set.
  - *Deliverable:* Trained Model Artifact (model.pt) + Embedding Matrix.
- **Week 3: Explainability & API.**
  - Implement the SHAP explainer on the VAE.
  - Wrap everything in FastAPI.
  - *Deliverable:* Working localhost:8000/explain endpoint.
- **Week 4: Frontend & Polish.**
  - Build the Streamlit Dashboard.
  - Integrate the "Trade Machine" demo.
  - Write the Readme and Case Study report.
  - *Deliverable:* Live App.

---

13) Ruthless Risk Register

Risk	Impact	Mitigation Strategy
nba_api IP Blocking	**Critical.** The project dies if we can't get data.	1. Develop locally (residential IP). 2. Commit the SQLite/Parquet DB to the repo (if < 2GB). 3. Use GitHub Actions with specific runners for weekly updates, which sometimes bypass blocks.
Missing Tracking Data	High. Historic games (pre-2013) lack full tracking.	**Fallback:** For pre-2013 data, the system automatically degrades to Tier 1 (Weighted Cosine on Box Scores) and flags this

		in the UI ("Tracking data unavailable; using box score approx").
Model Hallucination	Medium. VAE maps disparate players to same point.	**Sanity Check:** Hard filter by Position group (Guard/Forward/Center) during retrieval if the similarity score is below a certain confidence threshold.
Latency	Low. Calculating distances on fly is slow.	**Pre-computation:** We will compute the $N \times N$ similarity matrix for the current season *offline* and just look up results in O(1) time during the demo.

Conclusion:
This blueprint delivers a system that is technically sophisticated (VAE/Embeddings), methodologically sound (Stability Analysis), and practically useful (Explainable UI). It moves the conversation from "I can call an API" to "I can engineer a system that understands the structure of basketball." This is the profile that gets hired in 2026.

Works cited

1. (PDF) NBA2Vec: Dense feature representations of NBA players - ResearchGate, accessed January 7, 2026, https://www.researchgate.net/publication/368842816_NBA2Vec_Dense_feature_representations_of_NBA_players
2. NBA2Vec: Dense feature representations of NBA players, accessed January 7, 2026, https://arxiv.org/pdf/2302.13386
3. [2302.13386] NBA2Vec: Dense feature representations of NBA players - arXiv, accessed January 7, 2026, https://arxiv.org/abs/2302.13386
4. Using Deep Learning to Understand Patterns of Player Movement in the NBA, accessed January 7, 2026, https://www.sloansportsconference.com/research-papers/using-deep-learning-to-understand-patterns-of-player-movement-in-the-nba
5. Beyond the Box Score: Using Psychological Metrics to Forecast NBA Success, accessed January 7, 2026, https://www.sloansportsconference.com/research-papers/beyond-the-box-scor

e-using-psychological-metrics-to-forecast-nba-success
6. NBA Lineup Analysis on Clustered Player Tendencies: A new approach to the positions of basketball & modeling lineup efficiency - MIT Sloan Sports Analytics Conference, accessed January 7, 2026, https://www.sloansportsconference.com/research-papers/nba-lineup-analysis-on-clustered-player-tendencies-a-new-approach-to-the-positions-of-basketball-modeling-lineup-efficiency
7. Identifying Top Tier Elite NBA Players using Unsupervised Machine Learning Algorithms, accessed January 7, 2026, https://medium.com/@jkhancock/identifying-top-tier-elite-nba-players-using-unsupervised-machine-learning-algorithms-48cde2649c4
8. SAINT: IMPROVED NEURAL NETWORKS FOR TABULAR DATA VIA ROW ATTENTION AND CONTRASTIVE PRE- TRAINING - Jetir.Org, accessed January 7, 2026, https://www.jetir.org/papers/JETIR2307294.pdf
9. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training, accessed January 7, 2026, https://table-representation-learning.github.io/assets/papers/saint_improved_neural_networks.pdf
10. (PDF) SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training - ResearchGate, accessed January 7, 2026, https://www.researchgate.net/publication/352081653_SAINT_Improved_Neural_Networks_for_Tabular_Data_via_Row_Attention_and_Contrastive_Pre-Training
11. Contrastive Learning for Sports Video: Unsupervised Player Classification - CVF Open Access, accessed January 7, 2026, https://openaccess.thecvf.com/content/CVPR2021W/CVSports/papers/Koshkina_Contrastive_Learning_for_Sports_Video_Unsupervised_Player_Classification_CVPRW_2021_paper.pdf
12. [PDF] Contrastive Learning for Sports Video: Unsupervised Player Classification, accessed January 7, 2026, https://www.semanticscholar.org/paper/Contrastive-Learning-for-Sports-Video%3A-Unsupervised-Koshkina-Pidaparthy/3943f1426b9991b5c39e04c9521fdf4fac20984c
13. Attention versus contrastive learning of tabular data: a data-centric benchmarking, accessed January 7, 2026, https://www.researchgate.net/publication/377499292_Attention_versus_contrastive_learning_of_tabular_data_a_data-centric_benchmarking
14. Semi-Supervised Contrastive Learning for Deep Regression with Ordinal Rankings from Spectral Seriation - NeurIPS, accessed January 7, 2026, https://proceedings.neurips.cc/paper_files/paper/2023/file/b2d4051f03a7038a2771dfbbe5c7b54e-Paper-Conference.pdf
15. STab: Self-supervised Learning for Tabular Data, accessed January 7, 2026, https://table-representation-learning.github.io/assets/papers/stab_self_supervised_learning_.pdf
16. Identifying NBA player similarity with Machine Learning | by Vivaan Sehgal - Medium, accessed January 7, 2026,

https://medium.com/@vivaansehgal01/identifying-nba-player-similarity-with-machine-learning-pca-k-means-agglomerative-clustering-f96f598ba307
17. Forecasting the future development in quality and value of professional football players for applications in team management - arXiv, accessed January 7, 2026, https://arxiv.org/html/2502.07528v1
18. Troubleshooting tips for NBA API/Privacy Laws for NBA API - VR Software wiki, accessed January 7, 2026, https://www.vrwiki.cs.brown.edu/applications-of-vr/vr-in-sports/troubleshooting-tips-for-nba-apiprivacy-laws-for-nba-api
19. Beyond the Black Box: Implementing Explainable AI (XAI) in Sports Analytics, accessed January 7, 2026, https://dev.to/ffteamnames/beyond-the-black-box-implementing-explainable-ai-xai-in-sports-analytics-4flg
20. An Introduction to SHAP Values and Machine Learning Interpretability - DataCamp, accessed January 7, 2026, https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability
21. Offensive Lineup Analysis in Basketball with Clustering Players Based on Shooting Style and Offensive Role - arXiv, accessed January 7, 2026, https://arxiv.org/html/2403.13821v1
22. A Data Mining Approach to Identify NBA Player Quarter-by-Quarter Performance Patterns, accessed January 7, 2026, https://www.mdpi.com/2504-2289/9/4/74
23. nba_api/docs/nba_api/stats/endpoints/defensehub.md at master · swar/nba_api - GitHub, accessed January 7, 2026, https://github.com/swar/nba_api/blob/master/docs/nba_api/stats/endpoints/defensehub.md
24. Synergy Basketball FAQs - Sportradar's APIs, accessed January 7, 2026, https://developer.sportradar.com/basketball/reference/synergy-basketball-faqs
25. Mapping the Modern NBA: Discovering Player Archetypes Through Data-Driven Clustering | by Cenker Cengiz | Nov, 2025 | Medium, accessed January 7, 2026, https://medium.com/@ccengiz/mapping-the-modern-nba-discovering-player-archetypes-through-data-driven-clustering-2452c295c3bc
26. 18 SHAP – Interpretable Machine Learning, accessed January 7, 2026, https://christophm.github.io/interpretable-ml-book/shap.html
27. Build your gen AI–based text-to-SQL application using RAG, powered by Amazon Bedrock (Claude 3 Sonnet and Amazon Titan for embedding) | Artificial Intelligence, accessed January 7, 2026, https://aws.amazon.com/blogs/machine-learning/build-your-gen-ai-based-text-to-sql-application-using-rag-powered-by-amazon-bedrock-claude-3-sonnet-and-amazon-titan-for-embedding/
28. Hybrid RAG Architecture: Bridging Structured and Unstructured Data for Smarter AI, accessed January 7, 2026, https://www.techaheadcorp.com/blog/hybrid-rag-architecture-definition-benefits-use-cases/
29. Unlocking Data with Natural Language: Exploring LangChain's Pandas Agent | by

Chiraggarg | Artificial Intelligence in Plain English, accessed January 7, 2026, https://ai.plainenglish.io/unlocking-data-with-natural-language-exploring-langchains-pandas-agent-545b4603e8a8
30. Cookiecutter Data Science, accessed January 7, 2026, https://cookiecutter-data-science.drivendata.org/