Illini-Fit Project Overview:

Problem Statement

The transfer portal now drives roster building in college basketball, with nearly 2,700 men's players entering by April 22, 2025 and over 600 meeting our Division I, positive BPM, profile complete criteria. Evaluating hundreds of prospects manually is time intensive and error prone.

Key Challenges & Our Solution

- **Scale:** 600+ candidates make film review and metric comparison unwieldy.
- **Efficiency:** Manual evaluation drains analyst hours and risks missing impact players.
- **Consistency:** Gut-checks introduce bias and misalignment across staff.

FitScore Benefits

- Ranks all prospects by a single, transparent score (0–99) combining quality, system fit, and roster need
- Instant sorting and filtering surface top targets for deeper evaluation
- Shared, data-driven rankings align scouting, coaching, and analytics teams seamlessly

IMPORTANT: Scope & Assumptions

This is a proof-of-concept built on mid-season portal data: The project assumes Illinois has not yet signed any transfers (this affects need score) and player statuses may have since changed. Ideally, the app would update regularly based on web scraped portal data.

Data Collection

All raw datasets were gathered with four Playwright-based TypeScript scrapers. I worked with four scraped datasets:

1. **Illinois Roster**
    - **Script:** `illinois-roster-scraper.ts`

    - **Source:** BartTorvik's Illinois roster page
    - **Fields:** per-player advanced box-score metrics (BPM, usage, efficiency splits, shooting percentages, etc.) plus recruit rank, class, height, role, conference, etc.

2. **Team-Level Data**
    - **Script:** `team-data-scraper.ts`
    - **Source:** BartTorvik's team data page
    - **Fields:** adjusted offense/defense efficiencies, barthag, tempo, shooting and rebound rates, turnover rates, block/assist rates, record, etc.

3. **Portal Player Stats**
    - **Script**: `transfer-players-scraper.ts`
    - **Source**: BartTorvik's transfer-portal leaderboard
    - **Fields**: per-player advanced stats (per-play BPM, ORTG/DRTG, usage, TS%, per-minute production) along with recruit background, team, conference, and role.

4. **247Sports Transfer Ratings**
    - Source: 247Sports Transfer Portal page
    - Fields: name, position, height/weight, recruit rating, HS vs. transfer trend, commitment status, source/destination schools, and profile URLs.

Frameworks and Libraries Used

2. Data Engineering & Modeling

- **Scraping (TypeScript + Playwright)**

    - Four scripts (`*.ts`) extract JSON from BartTorvik, KenPom, 247Sports, etc., as described above.

- **Processing & Model Calculation (Python)**
    - **Quality: `quality_score.py` uses Pandas/NumPy for z-scoring BPM and reputation inputs; SciPy's sigmoid for rescaling.**
    - **Style: `style_fit.py` leverages scikit-learn's `cosine_similarity` on 18-dim fingerprints after standardizing each metric.**

---

2. Front-End & Deployment

- **Framework:** React (with TypeScript) for a responsive, component-driven UI
- **Design:** Tailwind CSS for rapid, consistent styling
- **Charts & Tables:** Recharts for FitScore bar charts, ag-Grid for sortable player lists
- **Hosting:** Vercel for zero-config, Git-backed deployments and automatic preview URLs upon each push

---

3. Transfer-Portal Fit Model: Pillars & Calculation

We break "fit" into three coach-tweakable pillars (all scaled 0–1):

1. **Quality ("How good?")**
    - **Reputation (50%)**: 247Sports rating & Torvik percentile (average; or max if one missing)
    - **Production (30%)**: position-adjusted BPM (z-score), efficiency bonuses & usage penalties, sigmoid-scaled
    - **Competition (20%)**: team barthag percentile (to discount low-major dominance)
2. **Style ("System fit?")**

    - Compute an 18-metric “fingerprint” (pace, 3-rate, turnover %, rebound %, size, etc.) for each D-I player and for Illinois’s past four squads
    - Cosine similarity between fingerprints, rescaled 0–1

3. **Need (“Roster urgency?”)**
    - Identify minutes % × positive BPM for all departed players, summed by positional bucket
    - Each portal candidate inherits its bucket’s urgency score

Composite Fit

```
Fit = 0.34·Quality + 0.33·Style + 0.33·Need
```

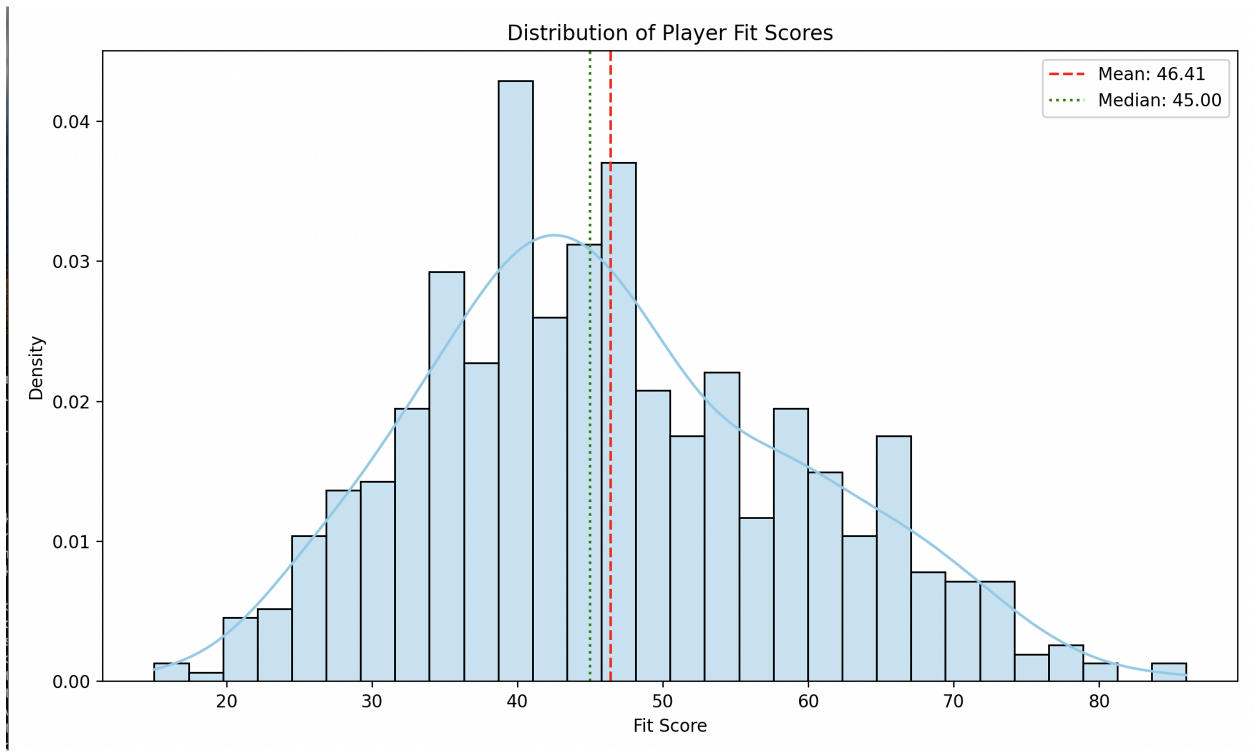
```
FitScore = round(Fit × 99)
```

Weights reflect staff priorities (talent & scheme fit > positional need) but are instantly adjustable via notebook sliders.

Example

- **Player A (6-4 PG):** Q 0.90, S 0.48, N 0.00 → Fit 0.44 → Score 44
- **Player B (6-8 4):** Q 0.82, S 0.66, N 0.80 → Fit 0.76 → Score 76

Player B outperforms A by filling a high urgency frontcourt need despite a similar talent level.



Here we can also see the distribution of fit scores, where we see a normal distribution in scores from players in the 2025 portal class.

Ongoing Validation:

- Initial staff feedback calibrated our baseline weights.
- Interactive sliders let coaches test alternate weight sets.
- Future steps include back-testing against past portal classes and soliciting coach reviews of top-ranked players.

Despite needing further research and weight refinement, the model's transparent pillar breakdown and real-time tweaking capability provide a clear, valid framework for decision making today.

Impact for GMs & Coaches

Impact Right Now

- **Speed & Focus:** Ranks 500+ prospects instantly, so staff spend less time on long shots and more on high impact targets.
- **Basic Filtration:** Filter controls let coaches rebalance talent vs. fit vs. need in real time based on need.
- **Bias Reduction:** A transparent, data-driven process aligns staff, minimizes inconsistencies, and creates a shared evaluation language.

Future Enhancements

- **Auto-Clip Highlights:** Embed key game clips per prospect for one-click film review.
- **Win-Impact Forecast:** Model each transfer's projected effect on lineup efficiency and net rating.
- **Enhanced Player Profiles:** Allow for more metrics and advanced filtering systems for coaches to quickly find specific prospects in mind.
- **Long-Term Value:** Link to NBA draft outlooks and recruiting pipelines for complete short- and long-term evaluation.

Links:

Github: https://github.com/Rami-Alkadri/PortalFitDashboard

Deployed Product: https://illini-psvuwpa3p-ramialkadri-9886s-projects.vercel.app/