# Project Overview - NBA Player Performance Analyzer

## 1. Introduction

The NBA Player Performance Analyzer is a web-based analytics tool designed to load, explore, and display statistics for NBA players using real-world data. While the original proposal planned a C# ASP.NET Core MVC system backed by a SQL Server database, the final implementation evolved into a Python-based Flask web application using:

- Flask (backend API)
- HTML/CSS/JS + Bootstrap (frontend interface)
- Pandas (CSV data processing)
- JavaScript-based dynamic rendering (dropdowns + tables)

This change keeps the intent of the project building a data-driven analytics tool, while allowing faster prototyping and more direct manipulation of the raw CSV datasets.

The system loads NBA player and team information from CSV files, exposes the data through REST API endpoints, and renders an interactive frontend. Users can select a player and automatically view their complete statistical profile as well as team averages.

## 2. Alignment With Original Proposal

| Original Plan | Final Outcome | Notes |
|---|---|---|
| C# + ASP.NET Core MVC | Python + Flask | Pivoted for development speed & reduced overhead |
| SQL Server database | CSV files loaded via Pandas | Maintains real-world data usage |
| Player Filter and Statistics | Implemented | Filter via dropdown next to column headers, click player name to view average statistics |

|  |  |  |
|---|---|---|
| Team Statistics and Comparison | Implemented | Select team to view average statistics over time, view individual teams statistics over time. Compare up to 5 teams' average statistics. |
| Bootstrap UI | Implemented | Bootstrap template integrated |
| REST endpoints | Implemented | /api/player-names, /api/players |
| Admin data import | Not included | CSV is loaded at startup |

Even with framework differences, the project still fulfills the goals:

- Data ingestion and processing
- Separation of backend API and frontend UI
- Dynamic display of player information and statistics
- Dynamic display of team information and team statistics

## 3. System Requirements

### 3.1 Software Environment

- **Language:** Python 3.12
- **Framework:** Flask
- **Frontend:** HTML, CSS, JavaScript, Bootstrap
    - https://startbootstrap.com/theme/freelancer
- **Libraries:** Pandas
- **IDE:** PyCharm Community
- **Dataset:** Kaggle - Basketball Data (CSV files)
    - https://www.kaggle.com/datasets/wyattowalsh/basketball/data
    - https://www.kaggle.com/datasets/eoinamoore/historical-nba-data-and-player-box-scores/data

### 3.2 Dependencies

- Flask

- pandas

Installed with:

**pip install flask pandas**

# 4. System Architecture

Although the system does not use MVC in the traditional ASP.NET sense, it follows a three-layer architecture:

## 4.1 Architectural Layers

**1. Data Layer (CSV + Pandas)**

- Loads player data from players.csv, playerStatistics.csv, teams.csv, and TeamStatistics.csv
- Computes FullName column
- Provides in-memory dataset for the server to expose

**2. Backend API Layer (Flask)**

Routes:

- GET / → home page
- GET /players → player stats page
- GET /api/player-names → list of player names
- GET/api/player-averages/<player_name>→ retrieves the average for player
- GET /api/players → full player data
- GET /teams → list of teams page
- GET /api/teams → teams info
- GET /api/team_averages → teams stats page

Responsibilities:

- Provide filtered JSON data
- Manage static assets
- Calculates averages and statistics

**3. Frontend Presentation Layer**

Technologies:

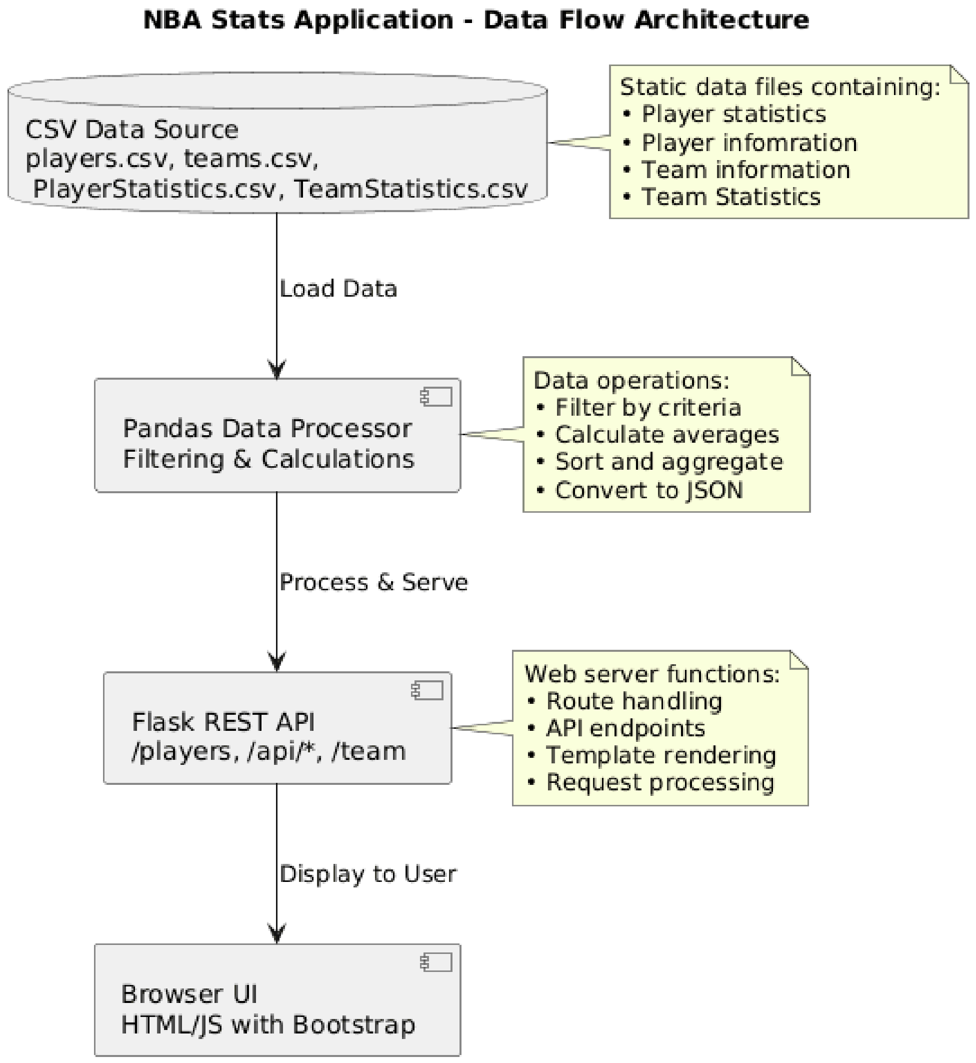
- Bootstrap template
- JavaScript fetch() API
- Dynamic table rendering
- Player dropdown that triggers live updates

Workflow:

1. Page loads
2. JS fetches /api/player-names
3. Dropdown is populated
4. Selecting a player fetches their full row
5. Table renders dynamically

## 5. UML Diagram — System Architecture

Below is a UML-style component diagram (text-based for submission portability):



# 6. How to Build & Run the Project

## 6.1 Setup

1. Open the project in PyCharm
2. Ensure interpreter uses the local .venv

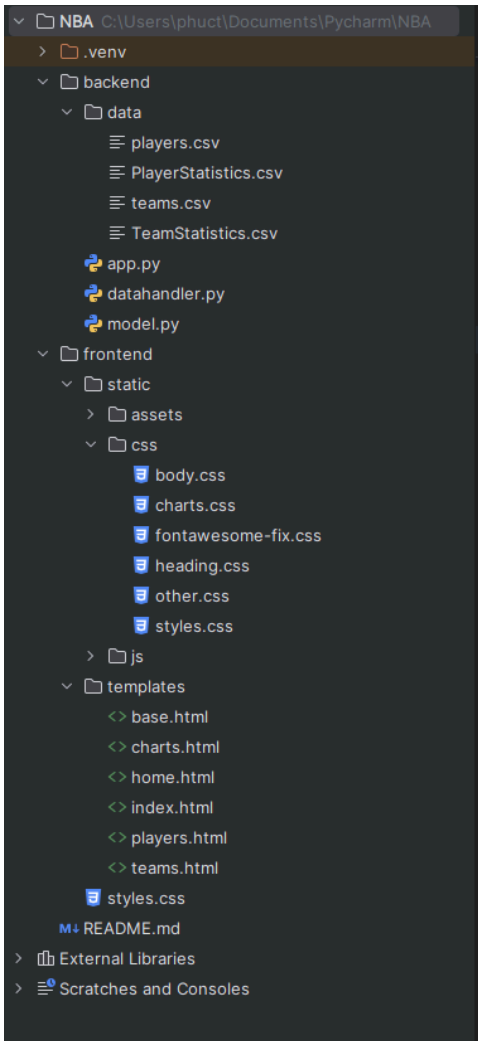
If .venv does not exist:

python -m venv .venv

## 6.2 Install dependencies

pip install flask pandas

## 6.3 Project Structure



```
NBA C:\Users\phuct\Documents\Pycharm\NBA
  .venv
  backend
    data
      players.csv
      PlayerStatistics.csv
      teams.csv
      TeamStatistics.csv
    app.py
    datahandler.py
    model.py
  frontend
    static
      assets
      css
        body.css
        charts.css
        fontawesome-fix.css
        heading.css
        other.css
        styles.css
      js
    templates
      base.html
      charts.html
      home.html
      index.html
      players.html
      teams.html
    styles.css
  README.md
External Libraries
Scratches and Consoles
```

## 6.4 Run the server in terminal

python backend/app.py

## 6.5 Access in browser

Ctrl + left click on http://127.0.0.1:5000/

# 7. Conclusion

The NBA Player Performance Analyzer successfully implements the core goals of the original proposal:

- It loads real NBA data
- It provides an interactive interface
- It exposes a clean separation between backend (API) and frontend (UI)
- It demonstrates important software engineering principles such as modularity, layered architecture, and data-driven design

Although the final system diverges from ASP.NET MVC, it achieves functionality consistent with the spirit of the proposal while introducing simpler deployment and faster iteration cycles.