HoopDeck App

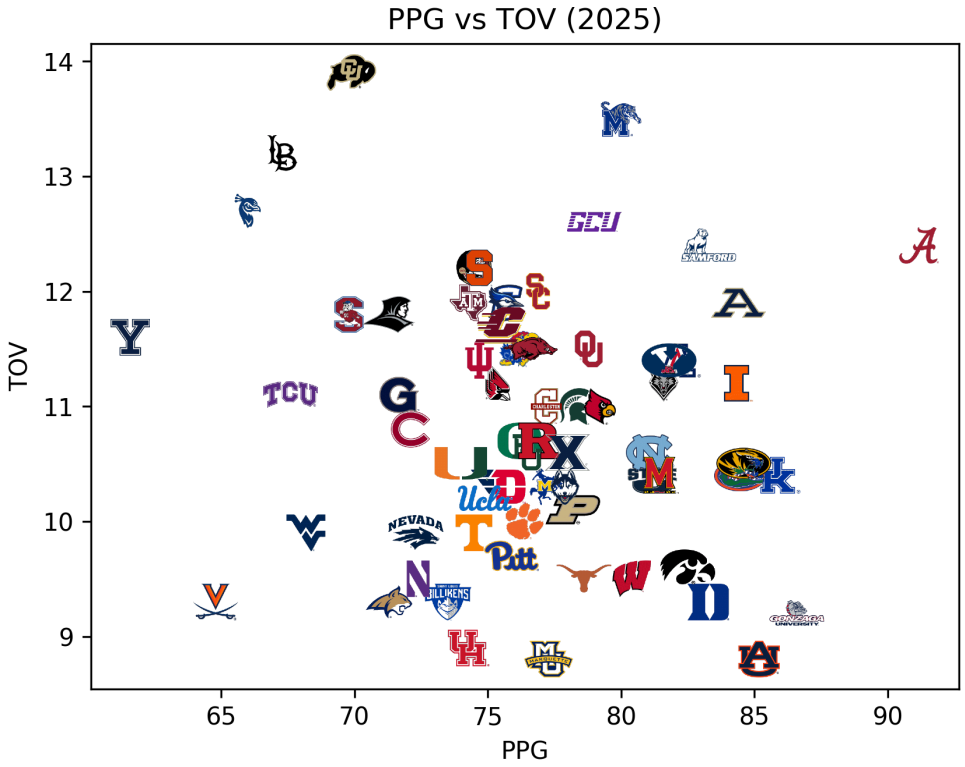
Lucas Holliday, Sid Javeri, Helena Yuan

Overview

HoopDeck is a basketball analytics platform designed for basketball fans, and more specifically, fans of the NCAA March Madness tournament. The app provides users with tools to search upcoming basketball games in their area through integrations with the Ticketmaster API, analyze and visualize team statistics in a graphical format, and predict game outcomes using Amazon SageMaker. This platform is built on a serverless architecture, with a Python-based client application that communicates with Lambda functions through API Gateway. The backend utilizes RDS for efficient database management.

Functions

1. Find Games with Ticketmaster API:
    a. This function allows users to find upcoming basketball games in a specified city within the next three weeks. Using the Ticketmaster API, the function grabs event details such as the date, game name, venue, and a link to the game on Ticketmaster. Users can input a city and the function will return a list of basketball games, formatted with information about game time, location, and a ticket link, along with other information. This function will also return, “No games,” if there are no upcoming games in the city.
2. Team Stats and Analyze Stats
    a. The Team Stats and Analyze Stats provide users with detailed team statistics and tools to visualize those statistics. The Team Stats function retrieves and displays team statistics for a specified college basketball team. Users are prompted to input a team name, and the function will fetch data statistics such as points per game, rebounds per game, and field goal percentage, among other statistics. Building on the data retrieved from the Team Stats function, the Analyze Stats function allows users to create their own visual comparisons of teams. They can select two statistical categories to plot on the x-axis and y-axis of a plot, with each team represented by its logo as shown in the figure below.



These plots can be used to identify strengths and weaknesses of teams, and all plots are saved for further analysis.

3. Predict Winner:
    a. This function allows users to input two teams of their choice and it will predict which team will win and by what margin. Predict Winner first calls the stats lambda function and then passes that data into a model that is trained using Amazon SageMaker. This model has weights for different stats and returns statistical information comparing the two teams. Then, the Predict Winner function grabs the winning and losing teams and the point differential and returns them to the user. This function also allows users to specify which team is the home team, which could affect the overall outcome and margin. Predict Winner can be used to create brackets that are backed up by statistics for higher likelihood of bracket accuracy.

Components and Architecture

1. Client App (main.py):
    a. main.py allows the users to interact with the system through the command line, similarly to the client-side in Project 02. This allows users to find upcoming games, retrieve team stats, generate custom plots, and predict game outcomes from the command line. This client-side app communicates with the backend Lambda functions through API Gateway.
2. Server (Lambda Functions):
    a. **hoopdeck_scraper**: This function scrapes team statistics from the Real GM website (https://basketball.realgm.com/ncaa/team-stats) and stores these statistics in an RDS database. It includes the following layers:
    bs4-layer (BeautifulSoup), requests-layer, pymysql-pypdf-layer
    b. **hoopdeck_stats:** This function retrieves the stored team statistics from the RDS database based on user inputs. It uses the following layer:
    pymysql-pypdf-layer
    c. **hoopdeck_predict**: This function predicts the outcome of a game between two user-inputted teams. It includes the following layers:
    AWSSDKPandas-Python313 (Pandas), requests-layer
    d. **hoopdeck_ticketmaster:** This function retrieves upcoming basketball games from the Ticketmaster API based on user input. It includes the following layer:
    requests-layer
3. MySQLWorkbench:
    a. Used MySQLWorkbench to create a table in the database called hoopdeck that contains id, team_rank, team_name, games_played, minutes_per_game, points_per_game, field_goals_made, field_goals_attempted, field_goal_percentage, three_pointers_made, three_pointers_attempted, three_pointer_percentage, free_throws_made, free_throws_attempted, free_throw_percentage, offensive_rebounds, defensive_rebounds, rebounds_per_game, assists_per_game, steals_per_game, blocks_per_game, turnovers_per_game, personal_fouls. After the data gets scraped it is inputted into

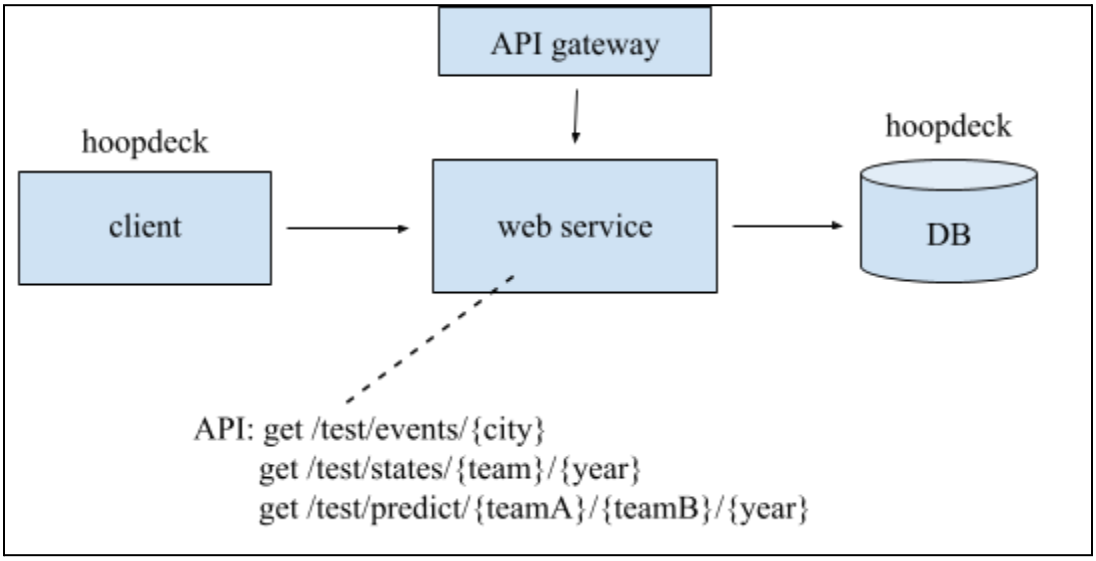
this database for future reference. There are also fictitious users with permissions to read only, or read and write for ease of assigning and granting permissions for specific functions used in the app.

Description of API

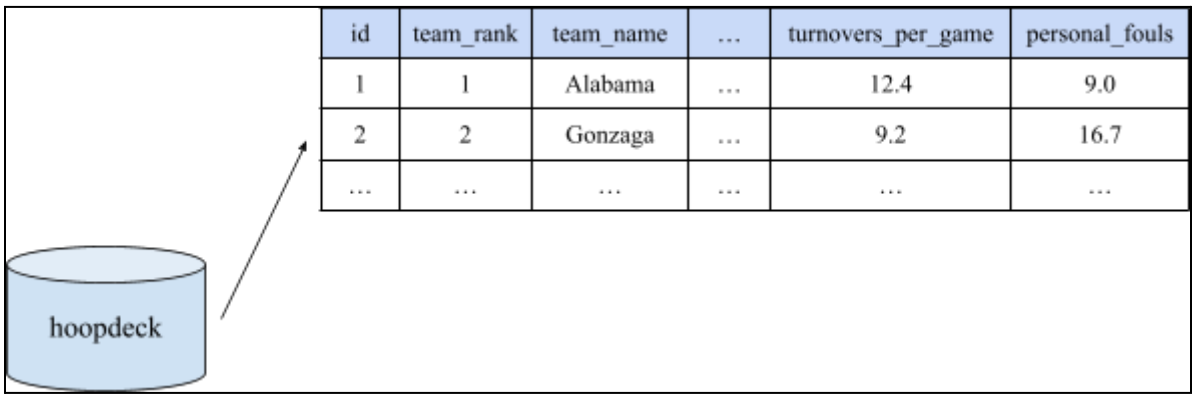
The API has several endpoints and each endpoint corresponds to a Lambda function. Below is a description of each endpoint:

1. hoopdeck_ticketmaster (Ticketmaster API)
    a. HTTP Verb: GET
    b. Endpoint: /test/events/{city}
    c. Parameters:
        i. City: the user inputted city to retrieve upcoming basketball games
    d. Response:
        i. Status code: 200 => success, 204 => no games found, 500 => error
        ii. Body: JSON array of game details, including game name, date, time, venue, address, and the ticket URL
2. hoopdeck_stats (Team Statistics API)
    a. HTTP Verb: GET
    b. Endpoint: /test/stats/{team}/{year}
    c. Parameters:
        i. team: The team name to search for
        ii. year: The season year (CURRENTLY ONLY 2025)
    d. Response:
        i. Status code: 200 => success, 400 => no team found or multiple teams match, 500 => error
        ii. Body: JSON object that contains the team's statistics
3. hoopdeck_predict (Predict Game API)
    a. HTTP Verb: GET
    b. Endpoint: /test/predict/{teamA}/{teamB}/{year}
    c. Parameters:
        i. teamA: Away team
        ii. teamB: Home team
        iii. year: The season year (CURRENTLY ONLY 2025)
    d. Response:
        i. Status code: 200 => success, 500 => error
        ii. Body: margin of victory as float

Structure



Database Schema



id	team_rank	team_name	…	turnovers_per_game	personal_fouls
1	1	Alabama	…	12.4	9.0
2	2	Gonzaga	…	9.2	16.7
…	…	…	…	…	…

API Gateway

baseurl = https://4u2hrt4t50.execute-api.us-east-2.amazonaws.com/test