Contribution Thus Far

## Cole Bromfield

* Found and assembled complex MLB datasets within project directory
* Wrote a series of backend functions that answer various potential user input queries, such as stat leaders, team info, and more. The input strings (team name, stat, player) will be drawn from user input (either drop-down menus or manual text entry).
* Added import statements and standard data cleaning to make MLB datasets directly accessible within backend code

## Mamadou Ndiong

Sketched an outline of the UI and researched best technologies and approaches for:

* **Creating an intuitive UI** that allows users to easily make requests for MLB data analysis without needing to understand the underlying code or technical aspects. This could include:
* Input fields, such as **dropdowns**, **text boxes**, and **date selectors** to allow users to filter data by player, team, statistic, or time period.
* **Graphical buttons** or options for selecting different types of visualizations (e.g., bar charts, line graphs, heatmaps).
* **Ensure the UI is responsive** and works across different devices (desktop, tablet, mobile).

## Dennis Reyes

* Created IAM Roles for team members
* API Gateway for handling REST API requests
* Lambda to handle backend requests
* S3 to serve as a static website as frontend
* Built CI/CD for deployments

**Group Members Names**

Cole Bromfield, Mamadou Ndiong, Dennis Reyes

**Planned Contribution**

*Current plan is for all members to contribute in every area, with each member taking charge in a specific domain matching his expertise*

* Cole Bromfield: Back-end code—Python code to perform specific data analysis, aggregation, visualization, and (possibly) machine learning tasks that the user will be able to request within the final interface. Assembling extensive edge-case handling to ensure no errors thrown in the final product
* Mamadou Ndiong: Front-end assembly— design and build UI, connect frontend to backend, Create Data Visualization Components. Set up API calls to communicate with the back end. For instance, when a user selects specific statistics and time ranges, the front-end should send a request to the back-end, triggering the Python code that performs the data analysis.
*  Handle **response display**: After receiving data from the back-end (e.g., rankings, averages, or visualizations), the front-end should dynamically display the results to the user.
* Dennis Reyes: Server infrastructure and file system—Implement AWS Infrastructure including assigning IAM Roles, building deployment pipeline, and allowing local development.

**Problem Statement**

With a 150+ year history and so many facets of the game, Major League Baseball (MLB) naturally has a gargantuan sea of statistical information available to the numerically inclined fan. However, finding specific information and/or answering a specific question is not always an easy task, particularly for those with little to no programming experience. The Lahman datasets, updated every year and widely regarded as the pinnacle of MLB statistical data collection, contain thousands (sometimes tens of thousands) of entries. This project will seek to address this problem with a simple interface, hosted on a static website through AWS by which a user can access aggregated/analyzed information (e.g. averages/player rankings for specified statistics across a specified period, customized visualizations, etc) on-demand.

**Approach**

* Back-end Python methods to read the Lahman datasets (in CSV format) and perform a wide array of analyses that the user could request
* Front-end setup through which user makes request (incl. blanks/dropdowns to specify types of requests, time periods, visualizations, etc)
* Robust AWS infrastructure to ensure reliable and fast performance, incl. serverless framework to host static website

**Implementation Tools**

* Python (incl. pandas, NumPy, matplotlib, possibly scikit-learn)
* Flask
* AWS (Serverless Framework, CloudFormation, Lambda, S3, API Gateway)