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
* Updated README file to include instructions for Windows users
* Began work on presentation
* Added visualization functions

## Mamadou Ndiong

* Worked on frontend of web page
* **Created wireframe of web page** that allows users to easily make requests for MLB data analysis without needing to understand the underlying code or technical aspects.
* **Added visualization features**
* **Built frontend. Connectivity to backend is needed**

## 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
* Integrated React + Serverless Framework for frontend
* Created sample implementation for frontend to call backend endpoint

**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)