**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)
* Any others?