# Final Report

## Abstract

This project addresses the challenge of accessing specific Major League Baseball (MLB) statistical information from a vast dataset, particularly for users with little to no programming experience. By leveraging a simple web interface hosted on AWS as a static S3 website, users can obtain aggregated and analyzed statistics on-demand, along with customized visualizations. Functionalities such as Lambda functions and API gateway simplify the development process, and provide a pay-as-you-go model perfect for uncertain and/or inconsistent levels of web traffic. Backend tools such as Pandas and Flask allow for seamless calculations pertaining to the user’s request, and frontend tools such as React and Vite provide a simple, visually appealing interface for the user to make said requests.

## Problem Statement

With a 150+ year history, Major League Baseball (MLB) boasts an extensive array of statistical information. However, finding specific data or answering specific questions can be daunting, especially for those without programming skills. The Lahman datasets, a gold standard for MLB statistics, contain thousands of entries, making manual data retrieval complex. This project aims to solve this problem by providing a user-friendly interface for data access and analysis.

## Related Work

There are various approaches that have been explored to handle large datasets and provide meaningful insights through user interfaces. Many existing solutions typically cater to users with some technical expertise. This project differentiates itself by focusing on ease of use, enabling broader accessibility to MLB statistics through an intuitive web interface. One similar concept has been completed, on various baseball topics, by independent analyst Greg Stoll. Stoll has constructed web interfaces that provide the user with abilities such as finding the probability that a team will score a given number of runs in an inning, given the situation (e.g. runners on first and second base, with one out), based on past data (Stoll, n.d.). While Stoll’s work is phenomenal and incredibly useful, we sought to set our work apart by including interactive visualizations and allowing a greater degree of variability in queries, along with utilizing AWS’s tools to minimize cost and maximize performance.

## Our Solution and Its Significance

This project’s solution involves:

* **Backend**: Python methods to read and analyze Lahman datasets (CSV format) for various user queries. The use of Flask to handle HTTP API requests. Pre-loads datasets, saving the computational time of loading them with each query. Functions designed to handle both batting and pitching datasets.
* **Frontend**: A web interface in React will allow users to make data requests without technical knowledge.
* **AWS Infrastructure**: We leveraged a serverless architecture to ensure efficient, scalable, and cost-effective performance. Key services used included AWS Lambda, API Gateway, S3, IAM, and CloudFormation. S3 was utilized for static website hosting of a Single Page Application (SPA), where the frontend, built with React, renders the UI and makes HTTP GET requests to the backend. API Gateway was employed to receive these HTTP requests and route them to the Lambda function, which processes the data and returns the necessary JSON responses to the frontend for displaying MLB statistics. To support collaboration, we configured IAM roles for all team members, granting them access to the AWS console for development. We also provided AWS API keys for programmatic access via the AWS CLI and SDK. For our CI/CD pipeline, we used the Serverless Framework along with CloudFormation, enabling easy deployment with a single click. This setup streamlined testing whenever changes were made. Additionally, we used GitHub for version control, allowing team members to submit and track changes efficiently.

## Novelties

* **Integration of extensive datasets**: Seamlessly integrating complex datasets into a user-friendly interface.
* **Automated visualizations**: Providing dynamic visual representations of data.
* **Serverless architecture**: Ensuring robustness and scalability.

Evaluation Results

We ran a load test with 200 requests (concurrency 10). All 200 requests were successful (error rate of 0), with an average response time of 0.07 seconds, for a throughput of 14.28 requests per second. For further, more rigorous testing, request count/concurrency can be increased, however we chose to stick with a maximum of 200 to avoid racking up charges, and because the results with 200 requests were promising.

## Contributions

***Cole Bromfield***

* Loaded complex MLB datasets.
* Developed backend functions for user queries (e.g., stat leaders, team info, visualization).
* Implemented import statements and data cleaning for backend accessibility.
* Updated README for Windows instructions.
* Created presentation
* Wrote evaluation code to test website performance with high request volume

***Mamadou Ndiong***

* Worked on the frontend web page.
* Created wireframe for user-friendly data analysis requests.
* Added visualization features.
* Built frontend (connectivity to backend pending).

***Dennis Reyes***

* Created IAM roles for access management.
* Set up API Gateway to handle REST API requests.
* Developed Lambda functions for backend processing.
* Configured S3 for hosting the static website.
* Built a CI/CD pipeline for streamlined deployments.
* Integrated React with the Serverless Framework.
* Developed a sample implementation for frontend-backend communication.

## Implementation Tools

* **Python**: Pandas, NumPy, Flask.
* **Backend:** Flask, CSV, REST API, Python
* **Frontend:** JavaScript, React, Vite
* **AWS**: Serverless Framework, CloudFormation, Lambda, S3, API Gateway, IAM.

## Softcopy Artifacts

* **Source Code Repository** [**URL**](https://github.com/ColeBromfield01/DATA650_project)
* **Backend** [**URL**](https://osir3dme2h.execute-api.us-east-1.amazonaws.com/dev/)
  + **Backend GET method routes**:
    - /read\_csv
    - /team\_lookup
    - /batting\_leader
    - /pitching\_leader
    - /period\_batting\_leader
    - /period\_pitching\_leader
    - /visualize\_batting\_leaders
    - /visualize\_pitching\_leaders
* **Frontend Website** [**URL**](http://msml-data-650-front-end-static-cb-dr-mn.s3-website-us-east-1.amazonaws.com/)

## References

Lahman, S. (n.d.). *Lahman Baseball Database* (Version 2023) [Dataset].

http://seanlahman.com/

Stoll, G. (n.d.). *Expected Runs in an Inning*.

https://gregstoll.com/~gregstoll/baseball/runsperinning.html