# STATSTRIKEFORCE

**Katherine McCarthy, Maxwell Mendenhall**
**Ryan Sayre, Tobyn Sitar**

GitHub

## PROJECT OVERVIEW

StatStrikeforce is a cloud-base web application, designed to reinvent the way you track your favorite video game statistics. The application allows for users to track their headshot percentage, Kill/Death ratio, win rate, and many other useful statistics with ease. StatStrikeforce gives players a competitive edge through its prediction feature, which uses machine learning to predict a player's performance and probability of a win in future matches based on recent match performance.

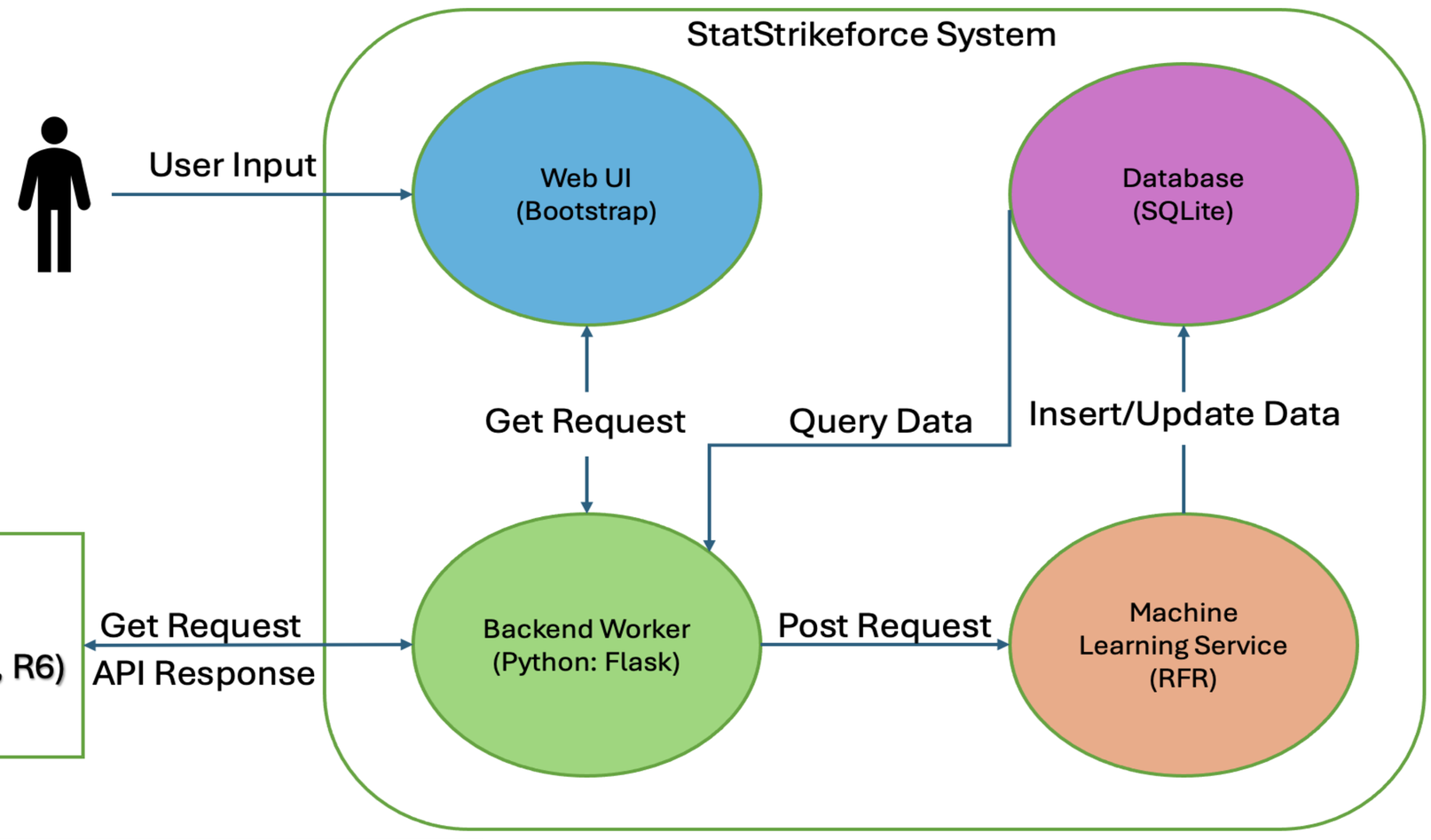
| User Data | | Prediction Stats |
|---|---|---|
| (P) UserID: varchar | Kills: int | P(Win): float |
| platform: varchar | Deaths: int | P(K/D): float |
| Playerlevel: int | Assists: int | P(ΔRP): int |
| Totaltime(hrs): int | K/D: float | |
| Wins: int | Headshot%: float | |
| Losses: int | Rank: varchar | |
| Win%: float | RP: int | |
| Matches: int | Tophero: varchar | |

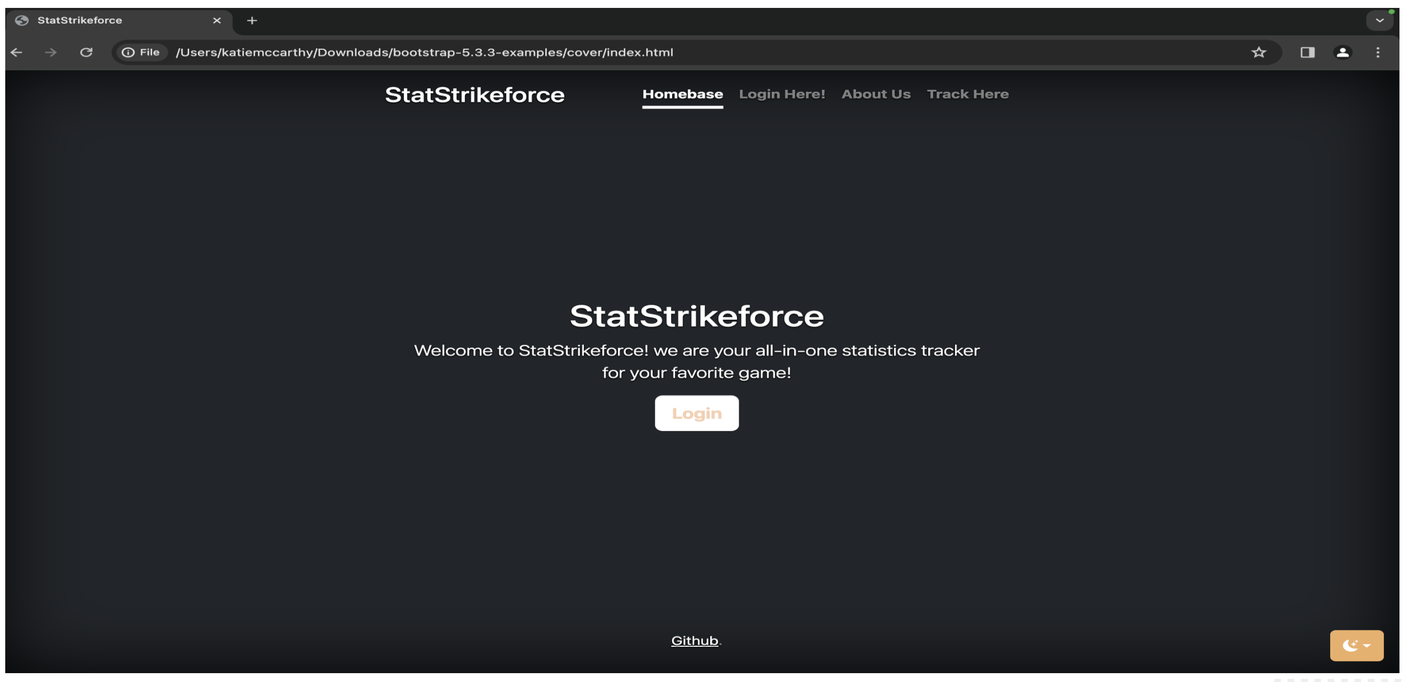
## Database Design

- SQLite incorporates database operations with our backend worker well with its Python libraries.
- SQLite is lightweight and doesn't need a separate server process, making SQLite more suitable for our project's scale.
- Our Database structure will made up of two primary tables 'User Data' and 'Prediction Stats'
- The User data table is used to store the game statistics, login credentials and basic profile information with each user choosing a 'UserID' to be used as the primary key.
- 'PredictionStats' table stores predictions generated by the machine learning service.

## Concept Architecture

The backend worker will communicate with the front end via GET requests, then the backend will communicate with the Rainbow Six API to get user info. After that it will POST requests to the Machine Learning service. ML service will then query the data into the database, with the primary key being UserID. Each UserID will have corresponding prediction values made with the user data from the API.




## Web UI

For the Web UI, we plan on creating a homepage for users to login with their account information for their R6 stats, leading them to a personalized page with all their statistics updated and accessible for them to view and compare their statistics from each match played, and also request a prediction for future matches. We used Bootstrap and its libraries to form our vision for the user interface.

## CI/CD Pipeline

- Our CI/CD pipeline begins with writing our source code, which is written and tested locally before pushing to our project GitHub repository.
- GitHub will trigger the build process so that docker will update and build the required images of each of the project components.
- Kubernetes will pull the images from Docker hub and run them in pods within our experiment nodes.
- GitHub will also trigger a deployment update directly to Kubernetes.
- The application can be successfully deployed in the Kubernetes cluster, service can be monitored/tested, and planning for future builds can take place.