DBMS Project Report on

# Analysis of ICC World Cup 2019


| | |
|---|---|
| **Afzal S** | CB.SC.I5DAS20140 |
| **Cibi V K** | CB.SC.I5DAS20107 |
| **Shrijith S** | CB.SC.I5DAS20129 |
| **Vignesh S S** | CB.SC.I5DAS20036 |

# Contents

# Abstract

This project is a web-based tool to analyze the performance of ICC world cup teams and players. The analysis helps the users to predict results for the ICC World Cup 2019. This project emphasis on the power of visualization, where the users can make accurate decisions and predictions by interactive plots.

This kind of system can be implemented for any sport with some slight modifications. This project attempts to eliminate / replace the code based or analytics based complex tools for analysis of ICC World Cup 2019. It aims at usability for anyone, who is a cricket enthusiast and also a data enthusiast. On regular updates to the database, this tool can be used to verify upcoming future world cups and also other international tournaments.

If these kinds of tools exist in the market, anyone who is passionate and interested in any sport can use them to get analysis insights. This website can be accessed on any device using internet.

# Chapter 1 - Introduction

This project is a web-based tool to analyze the performance and predict results for the ICC World Cup 2019. Millions of people watch cricket every day in India but not everyone can represent their country and play cricket on the field. But this problem was solved by the introduction of fantasy leagues in cricket in India. fantasy leagues became instantly famous as it involves both cricket and money and Dream 11 is one of the platforms where anyone who has basic knowledge about cricket and players can make a team of eleven from both the teams playing and predict the performance of players. if they predict the performance of players correctly, they are rewarded but it not easy to predict performance of each player without data. This is where our project helps our users with data with interactive plots which helps them grasp data rather than raw format. This helps our users to form a perfect team to win matches in fantasy league.

Nowadays, almost every sport and every international tournament are employing a group of analysts who analyze games and trends from which they derive meaningful insights and can make predictions based on previous data. For this, the analysts use wide variety of software and tools.

## Project Objective

Enables anyone on web to analyze and get insights about the players and teams for ICC World Cup 2019. For this, a clean and accurate website with data insights and good visualization graphs is required. These results have to be grouped, organized and calculated when needed accurately.

## Project Scope

Applications like this can replace conventional analysis tools and can take over the analysis work easy. More customization features, when added to this website can serve as a multi-use analysis tool.

# Chapter 2 - Technologies & Frameworks

To develop this kind of project, we have used several packages and frameworks. Let us see all these tools in this chapter.

## Frontend

The frontend is the place where the users interact with the software. So, the objective of the frontend is to maintain a good and clean interface, so that anyone can use the software without any effort. Programming language, libraries and frameworks used for frontend are listed below:

### JavaScript

JavaScript is one of the top and trending languages when it comes to web development and is also conquering other domains as well. JavaScript is used as the main frame language of the frontend. Node.js is used to enable usage of various frameworks and libraries. The libraries used are

- `react`
- `react-dom`
- `react-router-dom`

### React

React is JavaScript based library & framework that is used to build Single Page Applications and Apps. React alongside with ReactDOM is used to construct webpages. React Router is used on top of them to enable navigation between multiple pages.

## Backend

The backend uses Python to create server and a PostgreSQL database for storing and querying the data. The backend of this applications makes sure that each user request is processed accurately and provided with up-to-date information.

## Python

Python is easy to use and high-level programing language. It's diverse range of packages and modules help us to create complex application easily in lesser time.

## Flask

Flask is a web application framework written in Python. We can create Web Servers and APIs easily with lesser number of lines. Here, we use the `route` decorator to create routes and the corresponding functions return the results and values for the route. Alongside Flask, packages like `flask-cors` is used to prevent CORS related issues.

## pandas

**pandas** is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language. It enables us to create a `DataFrame` out of the data extracted from the database and use them efficiently.

## Plotly

Plotly is an interactive plotting library. We use its python-based library to generate graphs. Plotly has the ability of produce different types of graphs using various functions.

## PostgreSQL

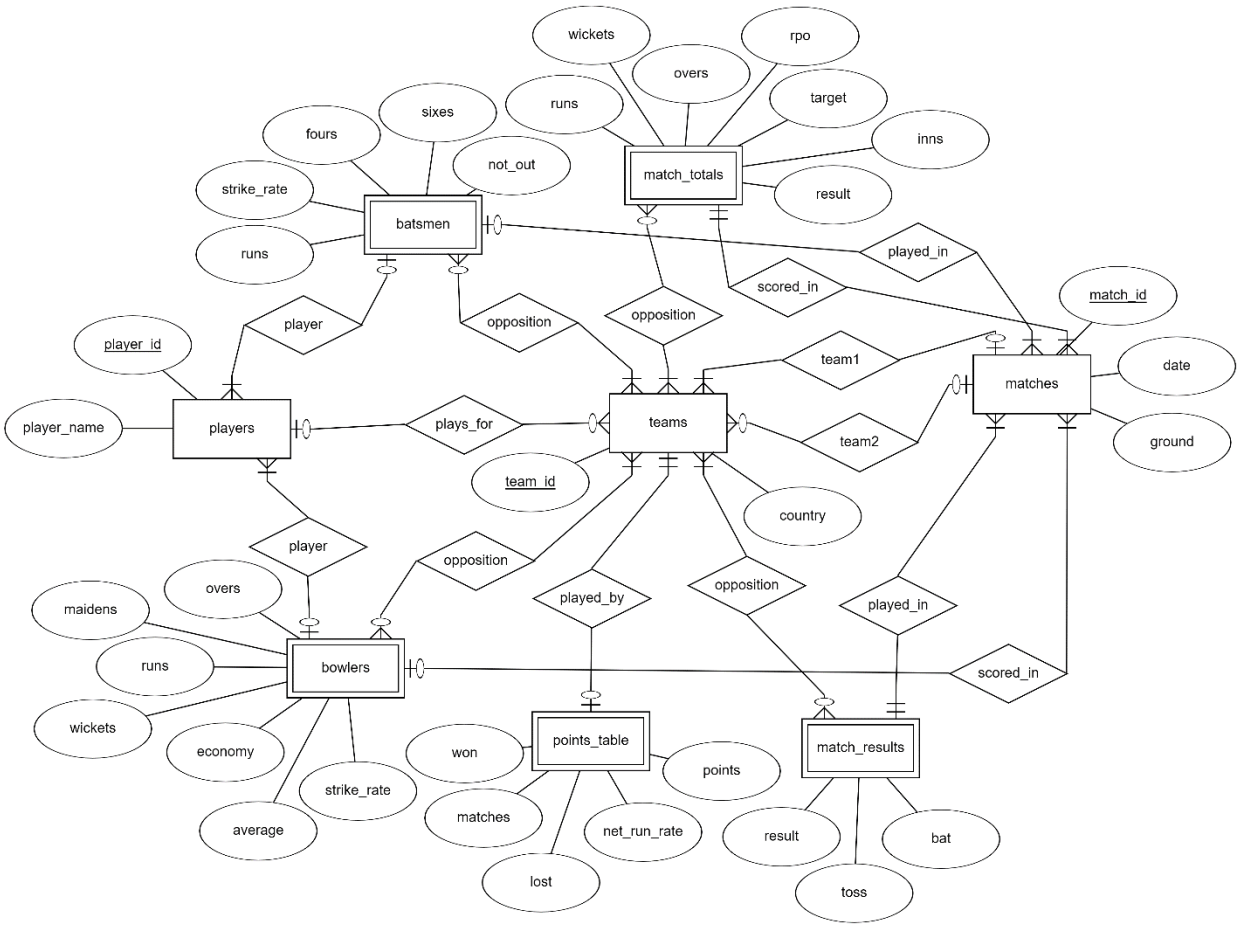
PostgreSQL is an open-source SQL database. The PostgreSQL database is accessed from the flask server using the `psycopg2`.

# Chapter 3 - Components and Design

## Backend

For database, a PostgreSQL server is used. If needed, a username and password for the database can also be set. A database named `crickalysis` is created and required data is imported into the database.

The *ER Diagram* will take this structure



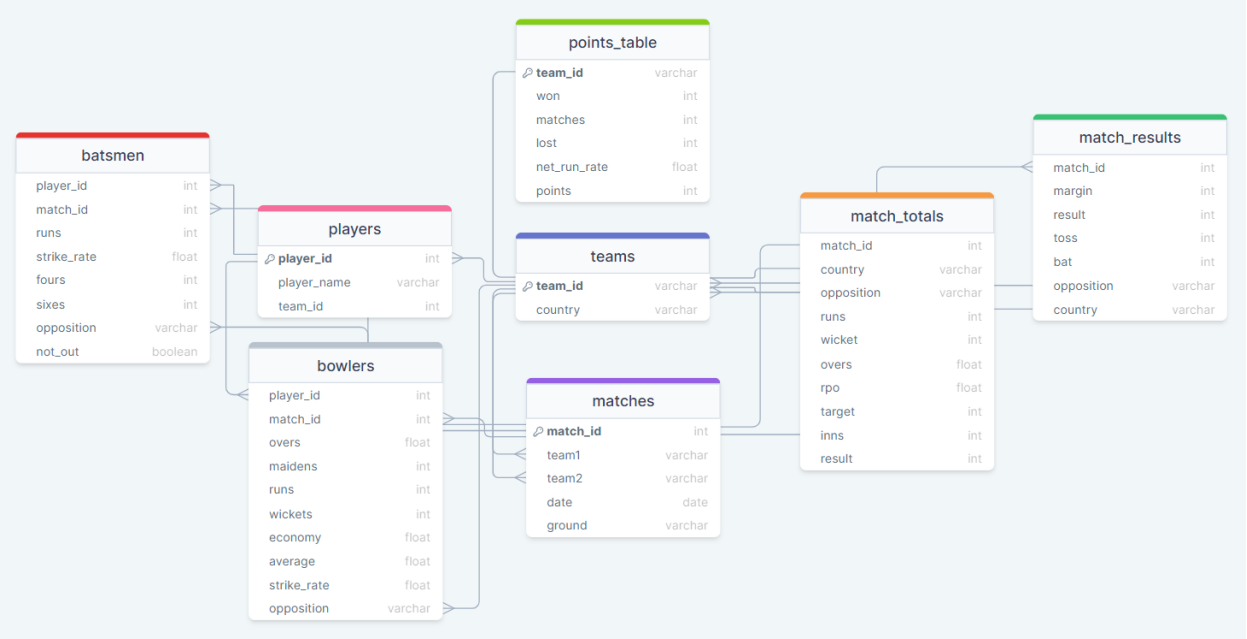
The entities and their attributes in this table are

- teams
  - team_id
  - country
- players
  - player_id
  - player_name
  - team_id
- matches
  - match_id
  - team1
  - team2
  - date
  - ground
- bowlers
  - player_id
  - match_id
  - overs
  - maidens
  - runs
  - wickets
  - economy
  - average
  - strike_rate
  - opposition
- batsmen
  - player_id
  - match_id
  - runs
  - strike_rate
  - fours
  - sixes
  - opposition
  - not_out
- match_totals
  - match_id
  - country
  - opposition
  - runs
  - wickets
  - overs
  - rpo
  - target
  - inns
  - result
- match_results
  - match_id
  - result
  - toss
  - bat
  - opposition
  - country
- points_table
  - team_id
  - matches
  - won
  - lost
  - net_run_rate
  - points

The entity sets of this schema are

- `teams (team_id, country)`
- `players (player_id, player_name)`
- `matches (match_id, team1, team2, date, ground)`
- `bowlers (match_id, player_id, overs, maidens, runs, wickets, economy, average, strike_rate)`
- `batsmen (match_id, player_id, runs, strike_rate, fours, sixes, opposition, not_out)`
- `match_totals (match_id, country, opposition, runs, wickets, overs, rpo, target, inns, result)`
- `match_results (match_id, result, toss, bat, opposition, country)`
- `points_table (team_id, matches, won, lost, net_run_rate, points)`

The *Schema Diagram* for this will be




Backend uses Flask in Python. The flask app is initialized with the `__name__` of the app and a path is set for static assets. By this, a Flask server will be running at port `5000`. Along with the Flask app, CORS is also setup using `flask_cors` to enable browsers to communicate with the server without any issues. The server connects to the PostgreSQL using `psycopg2`. A cursor is

also created to enable us to run queries. Various API endpoints are designed to enable the frontend to access and get appropriate data.

The endpoints available are:

`/`

An index route just for testing.

**Returns**:

```
Ok
```

`/players?q=`

Get list of players using a search query in parameter q

**Returns**:

```
[
    {
        "playerId": player_id1,
        "playerName": "player1",
        "teamId": "t1"
    },
    {
        "playerId": player_id2,
        "playerName": "player2",
        "teamId": "t2"
    },
    ...
    ...
]
```

`/player-analysis/<player_id>`

Get analysis data of a player using its ID.

**Returns**:

```
{
    "playerId": player_id,
    "playerName": "player",
    "country": "country",
    "analysis": {
        "analysis1": {
            "name": "Analysis 1",
            "graphData": "<html></html>"
        },
        ...
        ...
    }
}
```

`/team-analysis/<team_id>`

Get analysis data of a team using its ID.

**Returns**:

```
{
    "teamId": "team_id",
    "country": "country",
    "analysis": {
        "analysis1": {
            "name": "Analysis 1",
            "graphData": "<html></html>"
        },
        ..., ...
    }
}
```

`/points-table`

Get the points table chart.

**Returns**:

```
{ "table": "<html></html>" }
```

# Frontend

Since frontend is using React, each page, in fact each reusable element is a component. It starts from the `App.jsx` where the root of the html is referenced and a page is created. The whole application is place inside the `root` element in the html page.

Since React is a SPA framework, navigation and multi-page structures are not possible. In order to enable us do navigation, we use a library called `react-router-dom`. Router is defined and various paths along with the component is defined in a top-level component. A 404 page is also included at last, so that application doesn't crash if invalid routes are navigated.

Here are the pages that are created:

## `ErrorPage`

A 404 Error page which is shown when undefined routes are accessed.

## `HomePage`

A home page which enables you to go to various sections of the website. It will have three main button to navigate to various sections: Team Analysis, Player Analysis and Scoreboard

## `PlayerSelectPage`

This page enables us to search for players using their names and select them for analysis. It has a search bar which allow the use to enter some search term. The page then queries the API endpoint /players and shows the results below the search bar using cards. Each card will redirect the user to its respective analysis page.

### PlayerAnalysisPage

Shows the analysis for a player. Each analysis page will be consisted to multiple graphs like strike rate over time, runs and wickets scored against each team.

### PointsTablePage

This page shows the final scoreboard of ICC Would Cup 2019. Queries the API endpoint /points-table and show the table in a frame. Consists of all the data of the ICC World Cup 2019.

### TeamSelectPage

This page shows us all the available teams and enables the user to select a team for analysis. The teams are shown in cards and each card will redirect to its respective analysis pages.

### TeamAnalysisPage

Shows team analysis for the selected team. The team analysis page consists of the multiple graphs like win and loss percentages.

The components used are:

### FrameViewer

This component allows us to load and show a website inside an `iframe`. It shows a loader until the frame loads and then show the frame to the screen. The component takes in a `srcDoc` parameter and an optional `width` and `height` which defaults to `500` and `800`.

### ImageViewer

`ImageViewer` allows to load and show an image on the screen. Similar to FrameViewer, a Loader is shown when the image is loading. It takes in three parameters, `src`, `height`, `width`.

## `Loader`

This is a component which can be shown when loading. This component has two types of loaders, one with a cricket game GIF and another with a simple spinner. The types can be toggled by using an `icon` Boolean parameter. The other argument is `fullSize` for specifying whether the component should take up the full size of the parent.

## `PlayerCard`

The `PlayerCard` shows a card with player details which includes the player's name and player's image. When the card is clicked, then the user is redirected to the corresponding player's analysis page.

## `Sidebar`

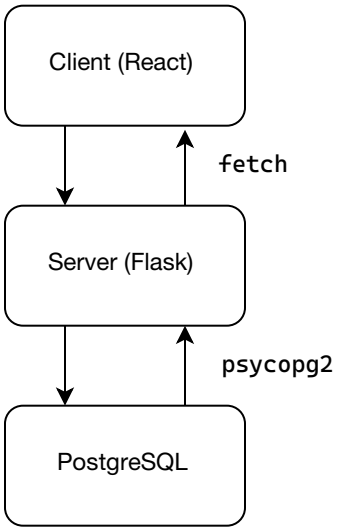
Sidebar acts a navigation for the entire website. The sidebar is shown all pages except the home and 404 page. The sidebar has a logo on the top which on clicks redirects to the home page. After the logo, there are three links redirecting to team, players and points tables pages respectively.

## `TeamCard`

The `TeamCard` is similar to the `PlayerCard` component where the team's name along with its logo is shown in the card. It takes in a team parameter with team object and a `clickHandler` which is triggered when the card is clicked.

# Linking Frontend and Backend

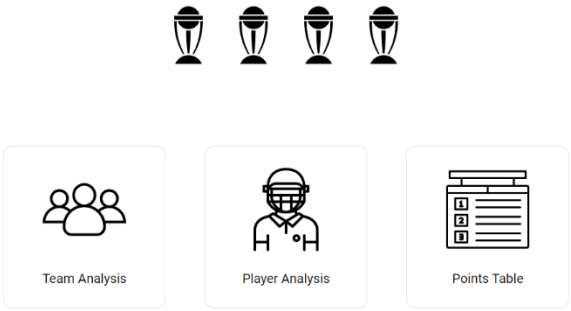
Whenever a user requests a page from his/her browser, a page is sent to the requesting client. The browser loads the sent document. Depending on the path the user is currently in and the parameters passed, the application decides which component to load up and to show. The frontend component gets data from the backend using the JavaScript's native `fetch` module. Upon successful retrieval of data, the data is stored in the components' state and the required actions are taken. Graphs and tables are shown using an `iframe` as the server responds with a graph in `html` format.



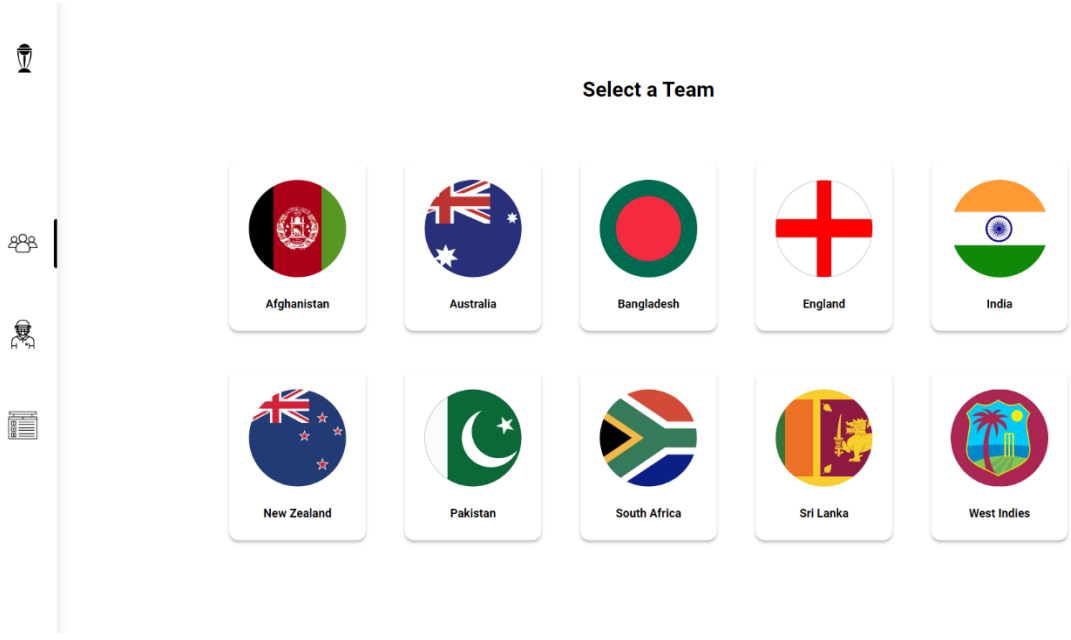

In the backend, whenever the server gets a request, the app decides which function to execute based on the path. The server gets data from the PostgreSQL Database using `psycopg2`. It uses the cursor that was initialized at the start of the server. A query statement is created using the parameters passed. The statement is executed using the execute command of the cursor and data is fetched using the `fetchone` or `fetchall` commands based on the requirement. The data is then modelled using pandas, if needed and a graph is created using Plotly if needed. The graph is of the `html` form, which will be embedded into the response.

# Chapter 4 - Diagrams & Screenshots

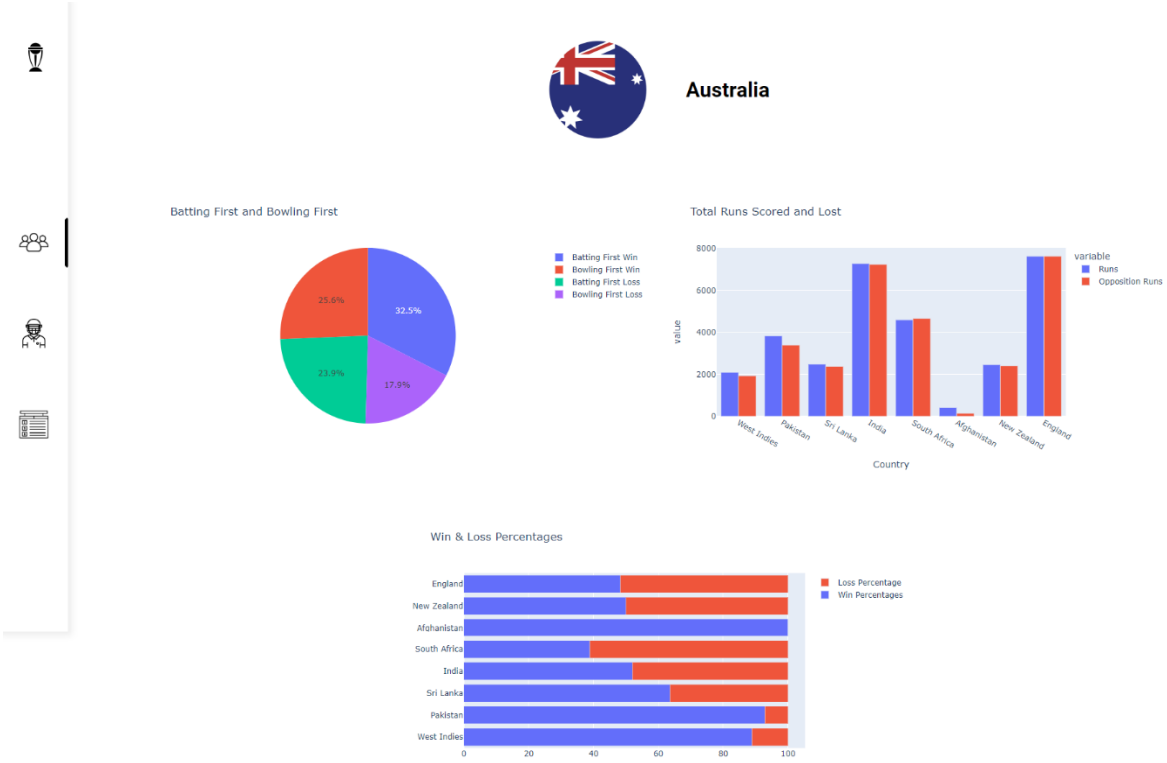
## Home Page



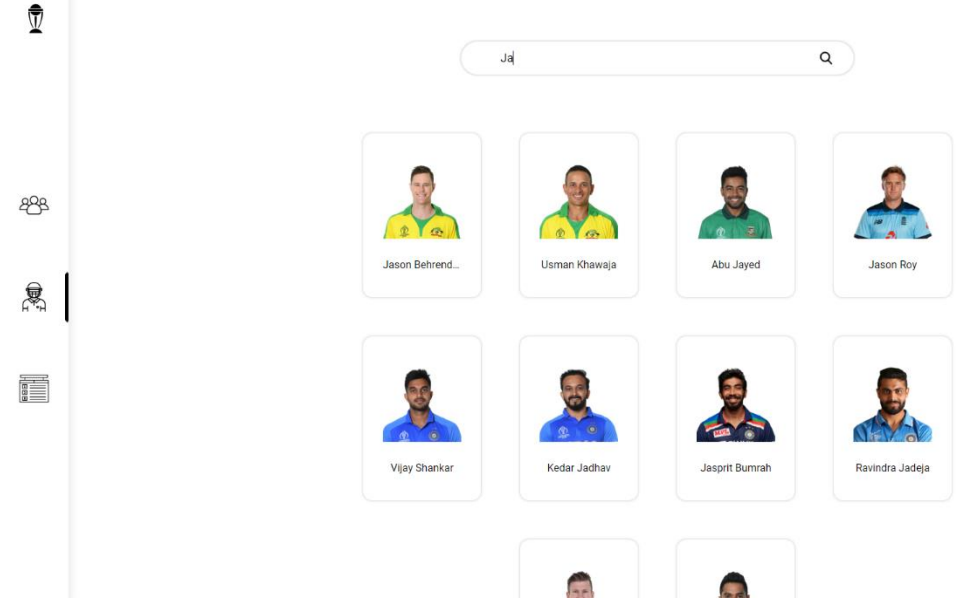

## Team Select Page



# Team Analysis Page




# Team Select Page

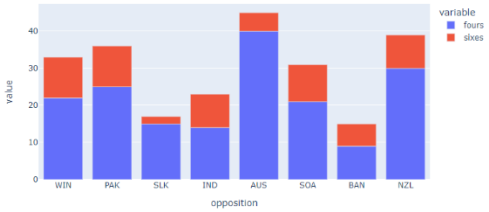


# Player Analysis Page

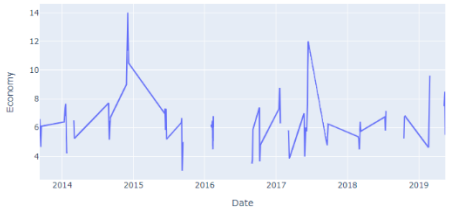
**Ben Stokes**

Country: England

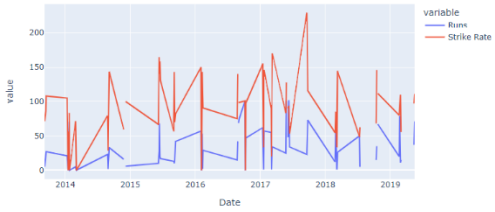
Boundaries (Fours & Sixes)



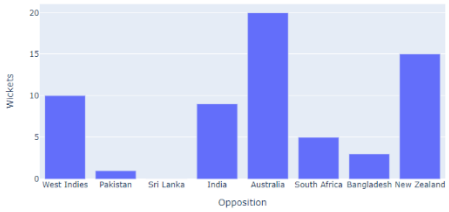

Player Economy over time




Runs & Strike Rate




Total Wickets

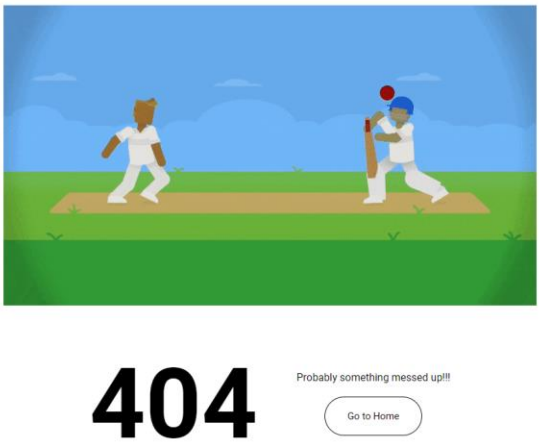



# Points Table

**Points Table**

| Country | Won | Matches | Lost | Net Run Rate | Points |
|---|---|---|---|---|---|
| India | 7 | 9 | 1 | 0.809 | 15 |
| Australia | 7 | 9 | 2 | 0.868 | 14 |
| England | 6 | 9 | 3 | 1.152 | 12 |
| New Zealand | 5 | 9 | 3 | 0.175 | 11 |
| Pakistan | 5 | 9 | 3 | -0.43 | 11 |
| Sri Lanka | 3 | 9 | 4 | -0.919 | 8 |
| South Africa | 3 | 9 | 5 | -0.03 | 7 |
| Bangladesh | 3 | 9 | 5 | -0.41 | 7 |
| West Indies | 2 | 9 | 6 | -0.225 | 5 |
| Afghanistan | 0 | 9 | 9 | -1.322 | 0 |

# 404 Error Page




# Link to the Project


https://github.com/cbxdv/Crickalysis