

MATCH RESULT TRACKER

TEAM CODE CRUSHERS


KRUTHI.Y

SWATHI.H.E


SAHANA.U

SHRAVANI.V.D

HARSHITHA.G

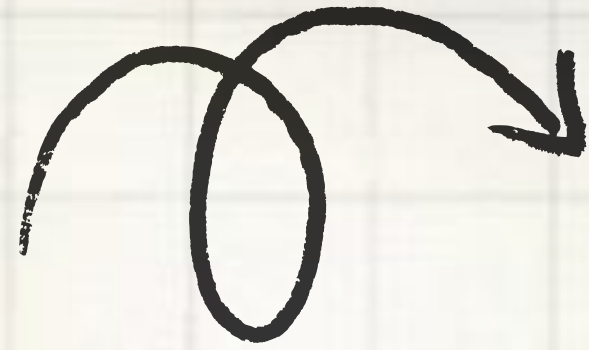


Introduction

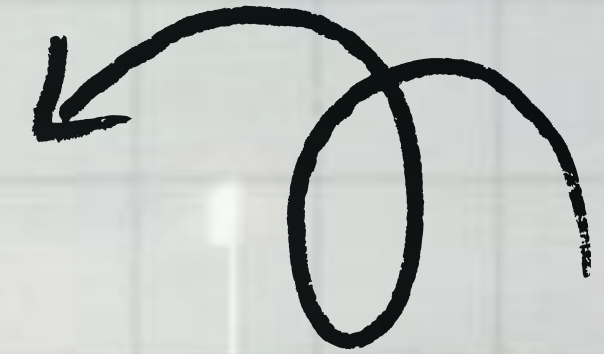
- **Purpose:** A match result tracker is designed to record and monitor the outcomes of sports matches or competitions.
 - **Functionality:** Users input data such as team names, scores, and relevant details into the tracker.
 - **Organization:** The tracker organizes this data in a user-friendly format, allowing for easy retrieval and analysis.
 - **Audience:** It caters to sports fans, coaches, analysts, and media professionals interested in tracking team and player performance.
 - **Formats:** Match result trackers can range from simple spreadsheets to more sophisticated software applications, depending on user needs.
- 

OBJECTIVES

- **Efficient Data Recording:** Enable users to quickly input match results, including team names, scores, and relevant details.
- **Accurate Tracking:** Ensure accurate recording and tracking of match outcomes to provide reliable data for analysis.
- **User-Friendly Interface:** Design an intuitive interface that makes it easy for users to navigate and input data without encountering unnecessary complexity.
- **Data Analysis Features:** Provide tools or functionalities for users to analyze the recorded data, such as generating statistics, trends, or visual representations.
- **Data Security:** Implement measures to safeguard user data, including backups and encryption, to maintain confidentiality and integrity.



ALGORITHM



- **Define Functions:** Define functions for loading data , adding a match, displaying match results, displaying team performance graph, updating match data, searching for a match, and deleting a particular match data.
- **Main Streamlit UI:** Create the main Streamlit UI where users can interact with the application.
- **Add a Match:** Allow users to add a new match by entering details such as team names, scores, man of the match, and highest wicket taker. This is done through the add_match function.
- **Display Match Results:** Provide an option to display match results, including team names, scores, man of the match, and highest wicket taker. This is implemented in the display_results function.



CONCLUSION

- **Performance Tracking:** The code facilitates the tracking of team performance over multiple matches by aggregating the total runs scored by each team.
- **Identifying Strong Teams:** Teams with consistently higher total runs indicate stronger batting lineups or better overall performance. Conversely, teams with lower total runs may need to focus on improving their batting strategies or player performance.
- **Assessment of Trends:** Consistent improvement or decline in total runs can provide insights into the overall trajectory of a team's performance.

```
class Team:
    def __init__(self, name):
        self.name = name
        self.matches_played = 0
        self.wins = 0
        self.losses = 0
        self.ties = 0

    def record_win(self):
        self.matches_played += 1
        self.wins += 1

    def record_loss(self):
        self.matches_played += 1
        self.losses += 1

    def record_tie(self):
        self.matches_played += 1
        self.ties += 1

    def __str__(self):
        return f"{self.name} - Played:
{self.matches_played}, Wins: {self.wins}, Losses:
{self.losses}, Ties: {self.ties}"
```

```
# Define a class for the cricket match
class Match:
    def __init__(self, team1, team2, score1, score2):
        self.team1 = team1
        self.team2 = team2
        self.score1 = score1
        self.score2 = score2

    def result(self):
        if self.score1 > self.score2:
            self.team1.record_win()
            self.team2.record_loss()
            return f"{self.team1.name} won the match  
by {self.score1 - self.score2} runs."
        elif self.score2 > self.score1:
            self.team2.record_win()
            self.team1.record_loss()
            return f"{self.team2.name} won the match  
by {self.score2 - self.score1} runs."
        else:
            self.team1.record_tie()
            self.team2.record_tie()
            return "The match was a tie."
```



```

# Define a class to track the cricket matches
class CricketTracker:
    def __init__(self): # Corrected __init__ method
        self.teams = {}

    def add_team(self, team_name):
        if team_name not in self.teams:
            self.teams[team_name] = Team(team_name)
        return self.teams[team_name]

    def add_match(self, team1_name, team2_name,
score1, score2):
        team1 = self.add_team(team1_name)
        team2 = self.add_team(team2_name)
        match = Match(team1, team2, score1, score2)
        print(match.result())

    def display_teams(self):
        print("\nTeams' Records:")
        for team in self.teams.values():
            print(team)

# Usage example:
if __name__ == "__main__":
    tracker = CricketTracker()

    # Adding matches and their results
    tracker.add_match("RCB", "CSK", 250, 230)
    tracker.add_match("MI", "RCB", 300, 300)
    tracker.add_match("CSK", "MI", 270, 260)

    # Display teams and their records
    tracker.display_teams()

```


Output:

```
Documents/matchresulttracker.py
```

- RCB won the match by 20 runs.

The match was a tie.

CSK won the match by 10 runs.

Teams' Records:

RCB - Played: 2, Wins: 1, Losses: 0, Ties: 1

CSK - Played: 2, Wins: 1, Losses: 1, Ties: 0

MI - Played: 2, Wins: 0, Losses: 1, Ties: 1



THANK
YOU