Report for CS564 Final Project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title of the Project: Cricket Pulse

Name of the students: Andrew Sabee, Sriharsha Devarapu, Harsh Sahu

1. Introduction

Report Organization

The report below is broken up into three sections: We begin with the introduction, which will give an outline of the project, alongside discussing how and why we were motivated to create the app we did. The second section covers in more detail the implementation of our app, the dataset used, and the tests performed to ensure our app was up to par. Finally, the report concludes with findings and lessons learned from the creation of this app. All group members contributed to all parts of this report.

Motivation

Cricket’s popularity has been increasing rapidly over the past few years, and it is now the second most popular sport worldwide. However, there is a lack of analytical and visualization tools available for the sport. The purpose of our app is to provide an analytical tool for fans and teams to be make data-driven decisions and observations based on the history of their players. This will allow for more precise analysis and adjustments by teams, and allow fans interested in data-gathering to understand on a deeper level the players they are cheering for. The motivation for making this a database-driven app lies in the size of the data – our ball-by-ball table alone is 150,000 entries, far too large to simply load on to RAM or access with conventional methods.

Application Description

Our application will be a chart of play-by-play data of the [cricket](https://en.wikipedia.org/wiki/Twenty20) league (IPL - Indian Premier League) for games up to 2017. The user will be able to sort/query data by player, team, match and ball-by-ball metrics to be able to see breakdowns across different areas of the sport. The goal is to allow the user to see how players and teams performed both across multiple seasons of play and also at different batting positions/conditions within a single match (i.e, maybe a player performs better (with a higher hit rate) at position 1 compared to position 5 in the batting order). Users will be able to add, modify, and delete notes related to a player or a team. These notes would contain the user’s thoughts or opinions on the data they were looking at.

We have broken down the query options into 3 different pages, shown below in the “Implementation” section of this report. The first page is Match Search, which will display information related to either a specific match or an aggregate of matches. Depending on the filters selected, it could display anything from a single match’s data up to the sum across the entire dataset. Data displayed on this page is not team- or player-specific, but rather looks at offensive and defensive stats, such as runs and wickets by over. The second page is Player Comparison, which allows the user to compare two players against each other in abstract, or compare their performance against a specific team. Individual stats are prioritized here, such as the player’s strike rate and the number of wickets they have taken. The final page is Team Comparison, which performs a very similar task to the Player Comparison page, but for teams instead of individual players. Screenshots of all of these pages are provided below.

1. Our Implementation
   1. Description of the system architecture

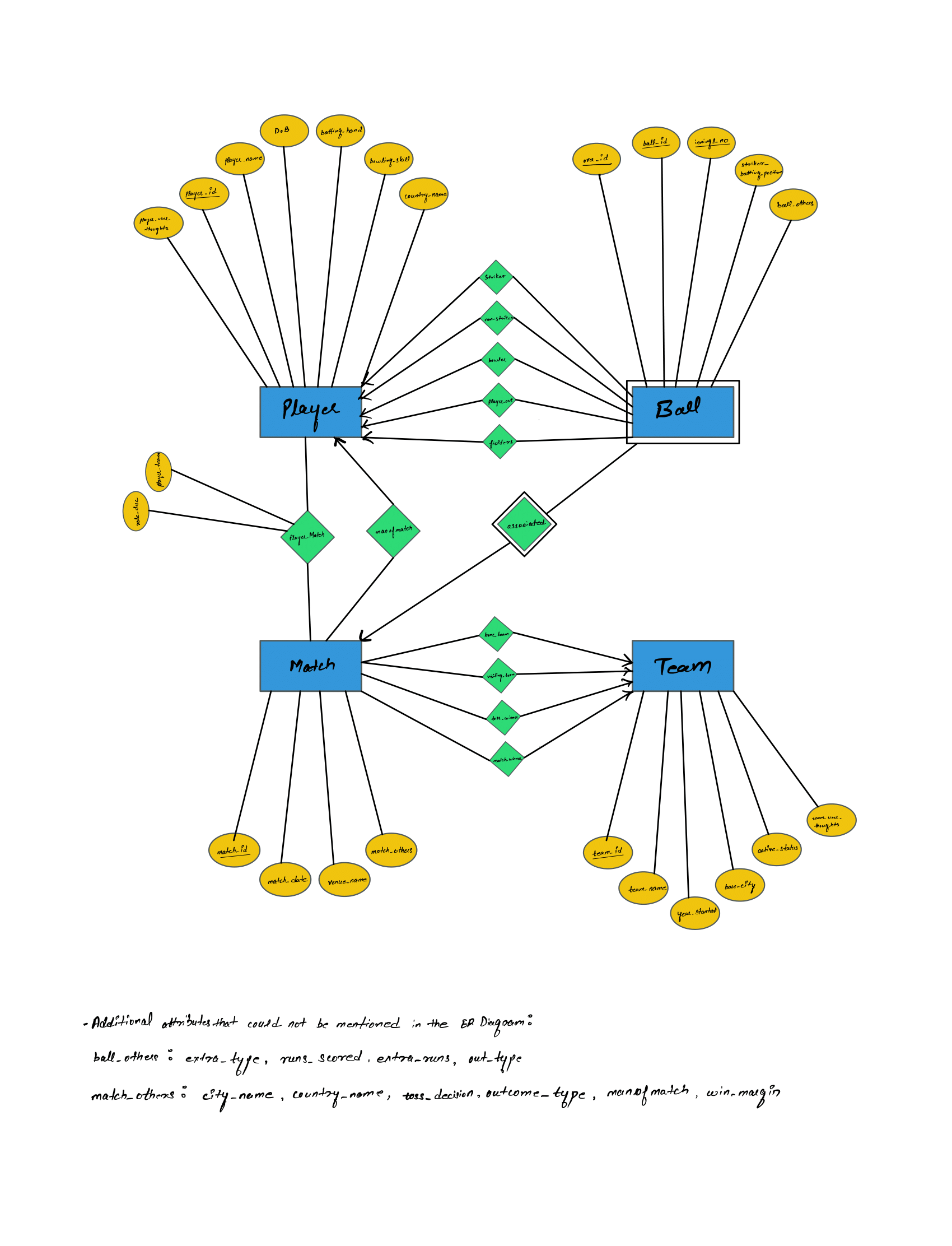
While compartmentalization of our code would have been ideal, the time constraints of this project forced us to make some compromises regarding the separation between GUI and backend code. All of our code is within two files, main.py and utils.py. Main.py is the central code for the application, and constructs the GUI for each individual page of our app. It also contains much of the code for updating these pages when a query is run. Utils.py serves loosely as our “backend,” and constructs the SQL queries based on the filters the user has selected to use on a given page, and passes that query into the database to get the data the user is looking for.

* 1. Description of the dataset

Our dataset comes from data.world, and initially included 5 tables, one of which had over 150,000 entries in it. After cleaning the data, adding team and venue IDs, and normalizing our schema, our final dataset had 7 tables, stored as .csv files.

Team.csv is relatively simple, storing information about a team’s ID, home city, and the user’s thoughts on the team. Staidum.csv is simply the stadium name and city. Player.csv stores information about a player such as their batting/bowling skill, and also the user’s thoughts on said player. Player\_match.csv represents the relationship between a player and a given match, and stores information such as who played in what match, and what team they played for in that match. Match.csv stores information on each match that was played, such as what teams were involved and the margin of victory of the match. Location.csv is simply a list of all the locations (both city and country) where matches have been hosted. Ball\_by\_ball.csv, the largest file in our dataset by far, contains the information for each ball bowled in IPL games from 2008-2017. It contains information about who was bowling, who was batting, when this ball was bowled within the match, and the end result of the ball.

* 1. ER diagram (High-resolution PDF [here](https://drive.google.com/file/d/1ljg10wLV4IENuiBBdg6qgSQq6NHvZQUf/view?usp=drive_link))



* 1. Relational model (final version from the previous checkpoint copied here)

Underlined attributes are the key attributes of the given relation

Player (Player\_ID:Number, Player\_user\_thoughts:String, Player\_name:String, DoB:yyyy-mm-dd, batting\_hand:String, bowling\_skill:string, country\_name:string)

Ball (Match\_ID:number, Over\_no:number, Ball\_no:number, Innings\_no:number, striker\_batting\_position:number(1-11), extra\_type:string, runs\_scored:number, extra\_runs:number, out\_type:string, Striker:number, non\_striker:number, Bowler:number, Player\_out:number, Fielder­s:number)

Match (Match\_ID:number, Match\_date:yyyy-mm-dd, Venue\_name:string, toss\_decision:string, Outcome\_type:string, Win\_margin:number, man\_of\_match\_player\_id:number, home\_team\_id:number, visiting\_team\_id:number, toss\_winner\_id:number, match\_winner\_id:number)

(home\_team, match\_date), (visiting\_team, match\_date), are secondary keys

Stadium (Venue\_name:string,  City\_name:string)

Location (City\_name:string, Country\_name:string)

Team (Team\_ID:number, Team\_name:string, Year\_started:number, Base\_city:string, active\_status:string, Team\_user\_thoughts:string)

Team\_name is a secondary key

Player\_Match (Match\_ID:number, Player\_ID:number, Role\_desc:string, player\_team\_id:number)

Normalization

We had to do some normalization of the data from the initial dataset in order to have our schema in BCNF. Below we detail the decomposition process and demonstrate that each table ends up in BCNF.

*Table: Ball*

FD = {match\_id, over\_id, ball\_id, innings\_no → striker\_batting\_position, extra\_type, runs\_scored, extra\_runs, out\_type, striker, non\_striker, bowler, player\_out, fielders}

This table is in both 3NF and BCNF as the LHS of the only FD (match\_id, over\_id, ball\_id, innings\_no) is a super key

*Table: Match*

Note: Assuming one team plays only one match in one day. One city name belongs to one country

FD = {match\_id → home\_team, visiting\_team, match\_date, venue\_name, city\_name, country\_name, toss\_winner, match\_winner, toss\_decision, outcome\_type, manofmatch, win\_margin

venue\_name → city\_name, country\_name

city\_name → country\_name

home\_team, match\_date → match\_id

visiting\_team, match\_date → match\_id }

Based on the above-defined functional dependencies, Match is *not in 3NF*, and the following functional dependencies violate 3NF: venue\_name → city\_name, country\_name & city\_name → country\_name

As part of 3NF decomposition, we will apply BCNF decomposition first

We will take *venue\_name → city\_name, country\_name* to begin the decomposition:

* Decompose Match into R1 = {city\_name, country\_name, venue\_name} & R2 = {venue\_name, match\_id, home\_team, visiting\_team, match\_date, toss\_winner, match\_winner, toss\_decision, outcome\_type, manofmatch, win\_margin}
* Now R2 is in BCNF but R1 is not (since city\_name → country\_name still violates BCNF)

We will take  city\_name → country\_name to decompose R1:

* Decompose R1 into R3 = {country\_name, city\_name} & R4 = {city\_name, venue\_name}

Now all the new decomposed relations of Match (R2, R3 & R4) are in BCNF

We will not proceed with the further steps of 3NF decomposition as the functional dependencies are already preserved.

Therefore, new decomposed relations/tables of Match (which are in 3NF) are:

* R2 = {venue\_name, match\_id, home\_team, visiting\_team, match\_date, toss\_winner, match\_winner, toss\_decision, outcome\_type, manofmatch, win\_margin}
* R3 = {country\_name, city\_name} (Location)
* R4 = {city\_name, venue\_name} (Stadium)

*Table: Team*

Note: Assuming each team will have a unique name

FD = {team\_id → team\_name, year\_started, base\_city, active\_status, Team\_user\_thoughts

team\_name → team\_id }

This table is in both 3NF and BCNF as the LHS of both the functional dependencies is a super key

*Table: Player*

FD =  { player\_id → player\_name, dob, batting\_hand, bowling\_skill, country\_name, Player\_user\_thoughts}

This table is in both 3NF and BCNF as the LHS of the only functional dependency is a super key

*Table: Player Match*

FD = { match\_id, player\_id → role\_desc, player\_team}

This table is in both 3NF and BCNF as the LHS of the only functional dependency is a super key

* 1. Implementation

A video walkthrough of the application can be found here.

Home Page

A screenshot of a computer

Description automatically generated

Match Stats Page

A screenshot of a computer

Description automatically generated

Player Comparison Page

A screenshot of a computer

Description automatically generated

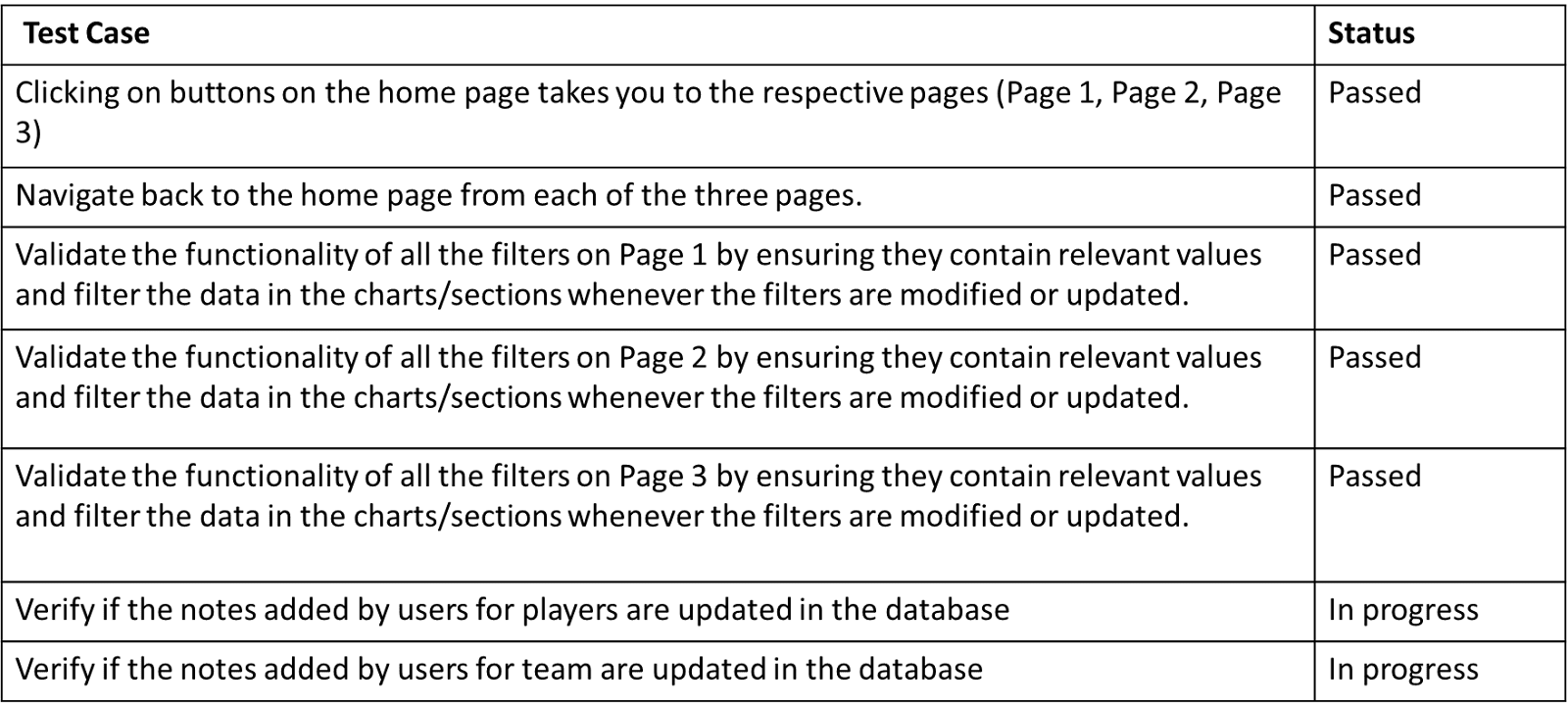
Team Comparison Page

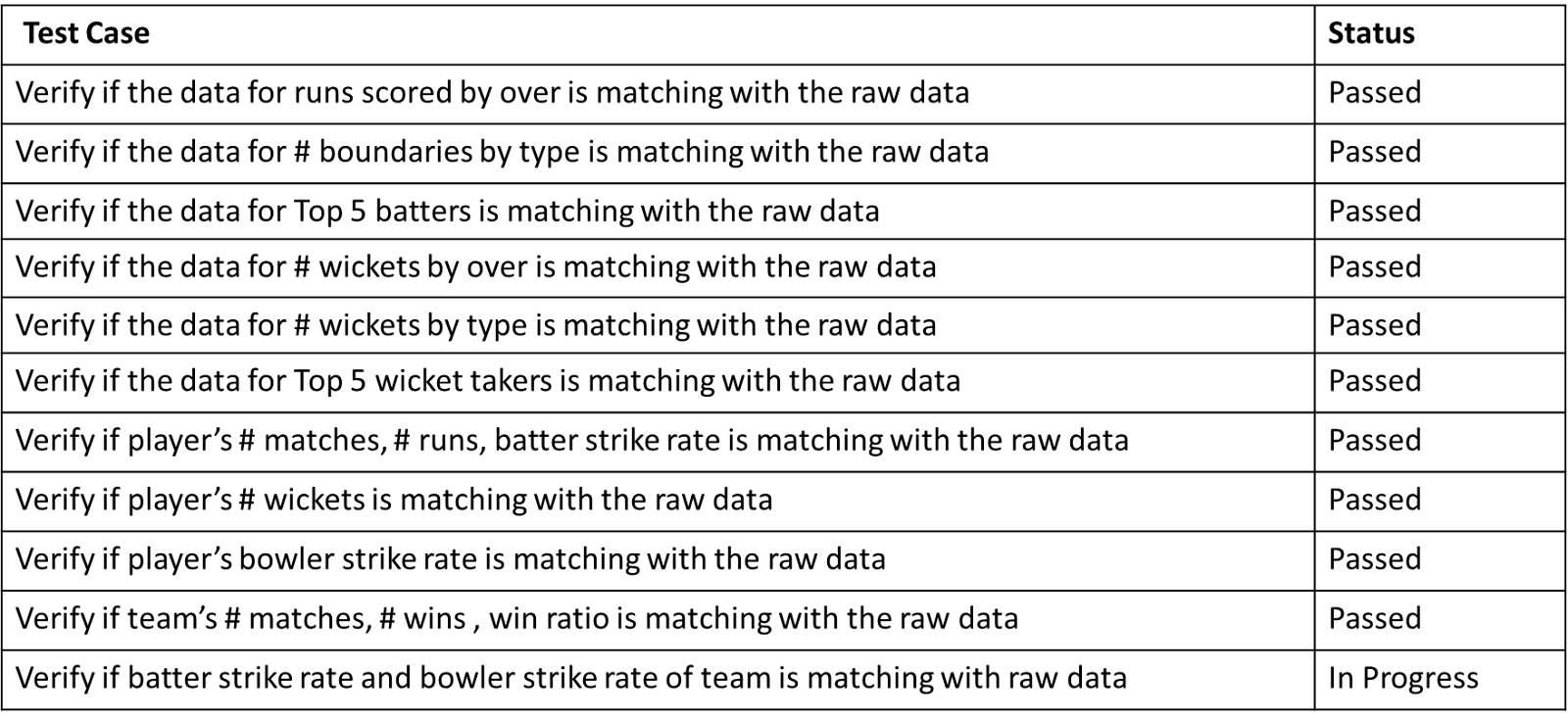
A screenshot of a computer

Description automatically generated

* 1. Evaluation

We plan to test the application for the **functionality (UI) and the data**. Below are the test cases for both.





1. Conclusion

During the creation of our app, we learned a lot about both the process itself and the way that real-world data can impact your plans for implementation. The dataset that we used for our project makes some very common operations very difficult, such as finding a team’s roster for a given match, or figuring out what team a player played for each year of his career. The way that our data was collected and represented had a tangible effect on what our app’s functionality was – we had to drop the idea of displaying a team’s roster for a given game due to the complexity of the SQL required.

That being said, our SQL queries remained very complex due to the number of filters involved in one individual search. Given 3 different pages, with 5+ filters for queries on each page, some of which were optional for a query, the SQL required to make sure those filters were translated into the correct query was very complex. It was eye-opening to see how useful and flexible SQL can be when done correctly.

We also learned a lot about the timeline of a project’s construction. Due to the condensed nature of summer courses, coordinating our efforts and making sure everyone was always contributing in one way or another was vital to ensuring that our project was finished in time for our presentation. We were forced to divide up the labor of our project efficiently and make decisions quickly about what we could and could not complete within the timeframe.

Our project also forced all of us to hone our technical skills in one way or another. Sriharsha and Harsh had never created a GUI prior to this project, and Andrew had never worked with Python prior to this. Each of us had to learn and close the gaps in our knowledge while creating this project in order for it to come together correctly. It also encouraged us to cooperate and help cover each other’s blind spots.