# Howzatt! Cricket Scorekeeper

Project Report

Faiz Rahman (24B1072)

## Contents

# 1 Introduction

This report provides a comprehensive analysis of a web-based cricket scoring system developed using HTML, CSS, and JavaScript. The system comprises four sequential modules: Match Setup, Live Scoring Interface, Scorecard Generation, and Post-Match Summary..

# 2 System Architecture

## 2.1 Page Flow

The application consists of four HTML pages linked by JavaScript logic:

1. **Setup Page:** Captures match parameters.
2. **Live Page:** Records ball-by-ball events.
3. **Scorecard Page:** Displays aggregated match data.
4. **Summary Page:** Shows match outcome and reset option.

## 2.2 Match Data

Match data contain the following things:

- Both teams name and which team is batting and which team is bowling.
- Innings details: runs, wickets, balls, overs, extras.
- Current players stats.
- Scorecards: objects keyed by over or player index for both innings.

This object is update via:

```
localStorage.setItem('matchData', JSON.stringify(matchData));
```

# 3 Instruction

- for **setup page**: It need two input text for team names, there is two dropdown to select which team won the toss and what is the decision and after clicking the start match button it required to enter striker's name, non-striker's name and first bowler's name (all of these must not be empty).
- for **live match page**: It display the live match with striker, non-striker, bowlers data and buttons to proceed the match.
  - If **run** buttons is clicked then runs is added to the team score, striker's stat and bowler's stat and ball is increased by 1 in them.

  - If **wicket** button is clicked then striker get out, one wicket is added to the team score and bowler's stat. Ball is also increased by 1 in them.
  - If **wide** button is clicked then an extra run is added to the team score, bowler's stat and ball is not counted.
  - If **no ball** button is clicked then no ball and wide button hide as on no ball wide is not considered and there is no two no ball at a time. Now after selecting next button there is free hit.
  - If **go to scorecard** button is clicked then it navigate to scorecard.html.

- for **scorecard page**: It display the scorecard and there is only one button which is used to go back to live match page.
- for **summary page**: It display the result of match and there is only one button to reset the match.
- In this project, if match is not started then only setup page will open and if match is ended but not reset then only live page will not open.

# 4 Base style

Provides consistent global styling across all pages:

- Body is centered, font-family: Klavika, sans-serif, Background color is #50fa7b, Gradient background with hsl(190, 92%, 60%) to hsl(146, 79%, 69%) diagonally.
- There is a box shadow of no horizontal offset, 4px vertical offset, 10px of blur and black shadow with opacity 25%;
- Headings have an infinite in and out animation with animation pulse and with color hsl(0, 88%, 53%).
- Animation pulse: which vary between size original size and 1.05×original size.
- Table layout is fixed, border color is hsl(238, 81%, 55%), header and footer background color is rgba(98, 114, 164, 0.15), color is #6272a4 and body color is hsl(0, 15%, 41%)

# 5 Setup Page

## 5.1 HTML Structure

- The form includes inputs for team names and toss decision, and all of these must not be empty, with IDs and values to link to JavaScript.
- Match start submit button to start the match and navigate to live.html.

### 5.2 CSS Customizations

Button is designed with gradient background to look attractive and also size and background changes on hovering.(Same for all buttons except runs, wickets, etc. they have don't have gradient background)

### 5.3 JavaScript Logic

- Prompts ensure at least three names are collected before proceeding, enforcing non-empty input via loops.
- Default behaviour of submit is blocked and inputs are stored using element id in match data using addEventListener method.
- Initialize the match and store the match data is local storage.

## 6 Live Page

### 6.1 HTML and CSS

It make the basic structure of how live match should look like with adding tables for batsman and bowler and adding buttons for runs, wickets and extras.
It has similar css with setup.html except that it has scoring button with different color and has a table to display the live match.
In live.HTML, there is an extra attribute of run buttons which is **runs** and it help in updating the runs.

### 6.2 JavaScript Logic

- First run the function **livePage()** which take match data from local storage and **updateLive(matchData)** updates display data from local storage and also display an extra line if there is no ball and also updates the scorecard using **updateScorecard(matchData)** function.
- **recordRun(matchData, runs)**: It increase runs of striker and team score by **runs** and increase striker's ball by 1 and also check whether 4 or 6 is scored or not. If runs is **odd** then it also rotate the strike. If there is no **no ball** then balls of bowler and team increased by 1 and also check whether over is completed or not using **checkOver(matchData)** function. It also use **updateLive** function to update the score.
- **recordWicket(matchData**): It increase the wicket and ball tally of both bowler and team by 1, also increase ball of striker and striker get out. After that it check whether all wicket are down or not, if 10 wickets are down then whether it is first innings or second. If 10 wickets are not down then a new batter come to crease and his stats stored in the match data and then check over also using **checkOver** function. Then it also use **updateLive** function to update the score.

- **checkOver(matchData)**: If 2 overs completed and this is second innings then it navigates to **summary.html**. If ball is equal to 6 then over become 1 and it rotate strike and ask for next bowler's name (must not be empty) and if ball is equal to 12 then it change the innings using **changeInnings(matchData)** function. It also update the scorecard using **updateScorecard** function.
- **changeInnings(matchDate)**: It store 1st innings data and start second innings.
- **updateScorecard(matchDate)**: It updates the scorecard.

# 7 Scorecard page

## 7.1 HTML and CSS

HTML contain 4 tables, 2 for batting and 2 for bowling with different id. Also have two **h2** element to assign batting team name in each inning just above the table.
It's style is similar to live page, only button is on the top of the page and there is no heading.

## 7.2 JavaScript Logic

- First run the function **scorecardPage()** which take match data from local storage and calculate CRR, if ball $> 0$, as follows:

$$CRR = \frac{score \times 6}{balls}$$

and if ball and runs both 0 then CRR is **0.00** else **-**. After that it display scorecard. In 1st innings only two table is displayed with batting team name on the top and in 2nd innings two more table is displayed with batting team name on the top of these two table, one table for batting scorecard and other for bowling. It also calculate economy of bowler similar to CRR.

# 8 Summary Page

## 8.1 HTML and CSS

It's html has just one container and one button to reset the match. It's style is also similar to live page.

## 8.2 JavaScript Logic

First run the function **summaryPage()** where if first innings score is greater than second innings score then team1 from match data wins the match, similarly it check like that and if both team can't win then it's a tie.

# 9 Conclusion

This project demonstrates the integration of front-end web technologies to build a stateful, interactive application without a server-side component. Custom features like extras enhance realism, while the modular JavaScript structure ensures maintainability and ease of extension for features such as run-outs, commentary feeds or multi-match storage.

# References


[1] Cricbuzz: `https://www.cricbuzz.com`

[2] ESPNcricinfo: `https://www.espncricinfo.com`

[3] International Cricket Council: `https://www.icc-cricket.com`

[4] Mozilla Developer Network: `https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage`