

Cricket Webpage Project Documentation

Harsha Vardhan
Roll Number: 24b1069
IIT Bombay

April 27, 2025

Contents

1	Introduction	3
2	Objectives	3
3	Tools Used	3
4	Project Structure	3
5	Design and Layout	5
6	Webpage Features	7
7	Basic Features	7
8	Customizations	7
9	Functions Used in JS: <i>score.js</i>	8
9.1	Setup Page:	8
9.1.1	<i>startMatch()</i>	8
9.2	Live Page:	8
9.2.1	<i>loadMatchData()</i>	8
9.2.2	<i>setupEventListeners()</i>	8
9.2.3	<i>promptForPlayerNames()</i>	9
9.2.4	<i>showNamePrompt()</i>	9
9.2.5	<i>showModal(title,message,callback)</i>	9
9.2.6	<i>showModalError(message)</i>	9
9.2.7	<i>submitPlayerNames()</i>	10
9.2.8	<i>isDuplicatePlayer(name,team)</i>	10
9.2.9	<i>showNewBatterPrompt()</i>	11
9.2.10	<i>showNewBowlerPrompt()</i>	11
9.2.11	<i>checkForEndOfMatchOrInnings()</i>	11
9.2.12	<i>startSecondInnings()</i>	11
9.2.13	<i>endMatch(message)</i>	12
9.2.14	<i>addRuns(runs)</i>	12

9.2.15	<i>takeWicket()</i>	13
9.2.16	<i>addWide()</i>	13
9.2.17	<i>addNoBall()</i>	13
9.2.18	<i>consumeFreeHitIfNotExtra(isDeliveryExtra)</i>	14
9.2.19	<i>promptForRunOut()</i>	14
9.2.20	<i>processRunOut(runs,newBatterName)</i>	14
9.2.21	<i>addCommentary(eventCode,details{})</i>	14
9.2.22	<i>updateDisplay()</i>	14
9.3	Scorecard Page:	15
9.3.1	<i>goBack()</i>	15
9.3.2	<i>formatOvers(balls)</i>	15
9.3.3	<i>loadScorecard()</i>	15
9.3.4	<i>setupCommentaryHover()</i>	15
9.4	Summary Page:	16
9.4.1	<i>displayResult()</i>	16
9.4.2	<i>setupSummaryPage()</i>	16
10	FlowChart Representation of Functions in Live Page	17
11	References	17

1 Introduction

This project involves building a simple live cricket-scoring web application using HTML, CSS, and JavaScript. The web app will allow a scorer to input match events (runs, extras, wickets, etc.) using buttons, and the website will automatically update player and match statistics in real time.

2 Objectives

- To create an intuitive and responsive cricket webpage.
- To implement navigation through various sections like Teams, Scores, and Tables across various pages such as Setup, Live, Scorecard and Summary.
- To enhance user experience using consistent UI and responsive design elements.

3 Tools Used

- **Frontend:** HTML, CSS, JavaScript
- **IDE:** VS Code
- **Testing:** Selenium Module in Python
- **Version Control:** GitHub
Repository: <https://github.com/Y-Harsha-Vardhan/CS104-Project.git>

4 Project Structure

`script.js` — Contains all JS Logic for the project

`setup.html` — HTML page for entering the data about the teams

`setup.css` — Styling for the setup.html page

`live.html` — HTML page for controlling and viewing the score of the match

`live.css` — Styling for the live.html page

`scorecard.html` — HTML page for the scorecard of the match upto that point

`scorecard.css` — Styling for the scorecard.html page

`summary.html` — HTML page for displaying the result at the end of the match

`summary.css` — Styling for the summary.html page

`Images/` — This folder contains the images used in the webpage

`Project Documentation/` — This folder contains the L^AT_EX Documentation

`autofill.py` — This is a python script that helps to automatically fill the data in the webpage, here I used this for testing the working of the website.

5 Design and Layout

The following are the screenshots of various pages of the Website:

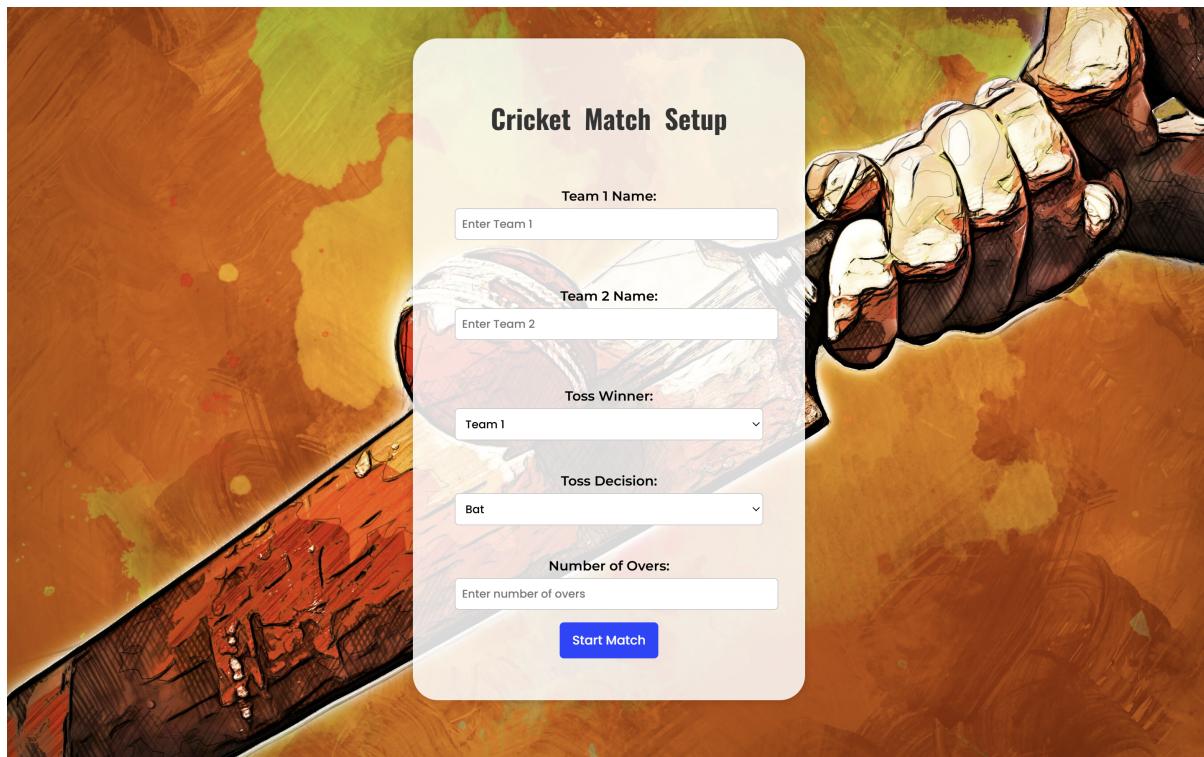


Figure 1: Homepage of the Cricket Webpage

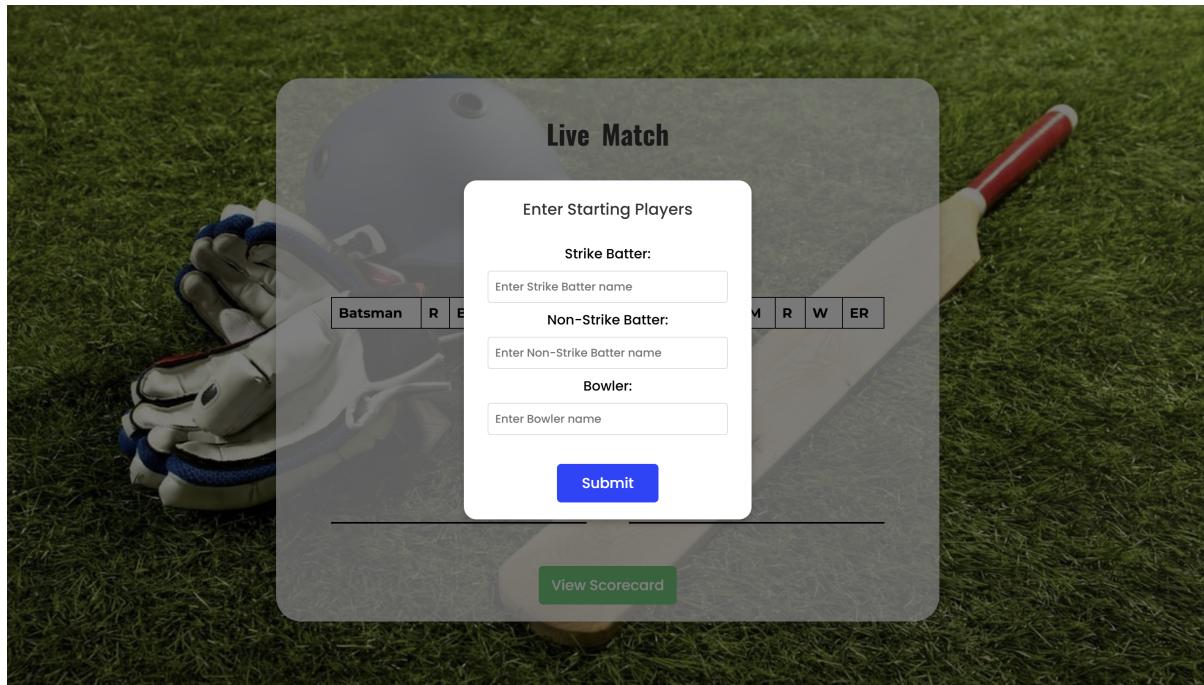


Figure 2: The initial modal on the Live Page

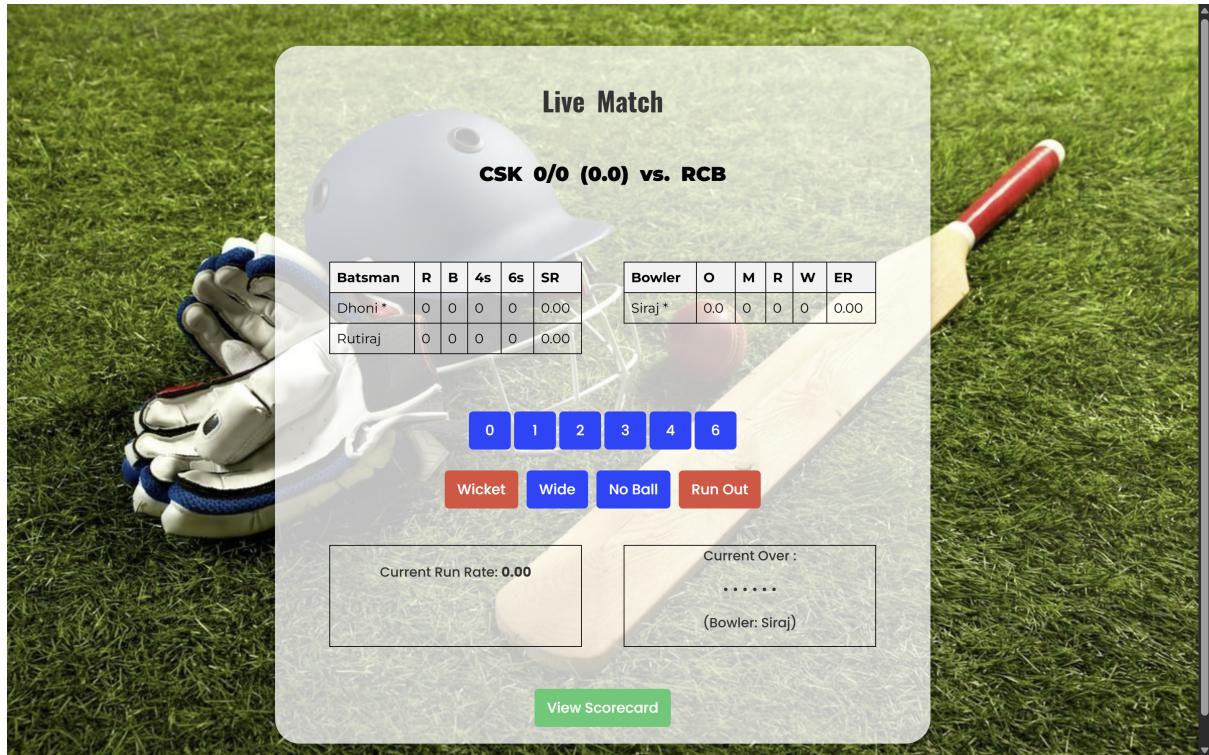


Figure 3: Live Page of the Cricket Webpage

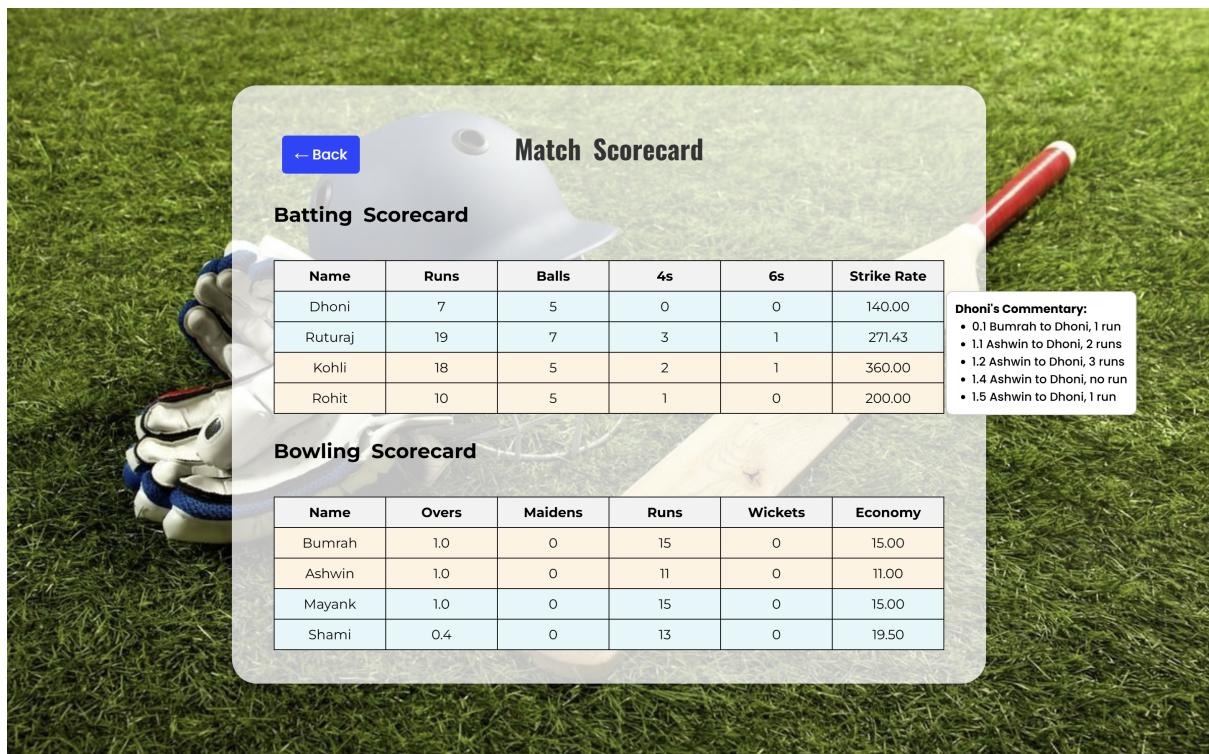


Figure 4: Scorecard Page of the Cricket Webpage



Figure 5: Summary Page of the Cricket Webpage

6 Webpage Features

- Setup page allows user to enter team names, toss winner and toss decision for the match [Fig 1].
- Live page allows the user to control the match using buttons and updates the display dynamically [Fig 2, Fig 3].
- Scorecard page shows the match commentary, data until that point of the match [Fig 4].
- Summary page displays the final match result and allows the user to reset the match if required [Fig 5].

7 Basic Features

-

8 Customizations

-

9 Functions Used in JS: *score.js*

9.1 Setup Page:

9.1.1 *startMatch()*

- Gets the names of the 2 teams, that are entered by the user and stores them in respective variables.
- Similarly, stores the name of the team that has won the toss (decided by user) and what that team's decision is (either to **bat** or to **bowl**).
- Allows the user to enter custom input for number of overs, if not entered or invalid input then defaults to 2 overs.
- Shows error messages if the user tries to submit without filling all the fields or if the two team's names are the same.
- Stores all these variables in an object: *teamData*
- Next it stores this object in the browser's *localStorage* by converting it into a *JSON String*.
- It clears any potentially leftover data: *cricketMatchData* from the *localStorage* of the browser.
- Finally it redirects to the Live Page.

9.2 Live Page:

When this page is initialized, an *EventListener* calls: *loadMatchData()* and *setupEventListeners()*, if the match isn't over and players are not yet set up, it calls *promptForPlayerNames()* else it calls *updateDisplay()*.

9.2.1 *loadMatchData()*

- It gets the match data that is stored as *cricketMatchData* in the *localStorage*.
- It stores it in a *matchData* object.

9.2.2 *setupEventListeners()*

- It calls the function *addRuns()* when a Run Button is clicked.
- It calls *takeWicket()* when the Wicket Button is clicked.
- It calls *addWide()* when the Wide Button is clicked.
- It calls *promptForRunOut()* when the Run Out Button is clicked.
- It calls *addNoBall()* when the No Ball Button is clicked.
- It redirects to Scorecard Page when the View Scorecard Button is clicked.
- It calls *submitPlayerNames()* when the Submit Names Button is clicked in the *Modal*.

9.2.3 *promptForPlayerNames()*

- It calls *showNamePrompt()* with a list as an argument according to the current match status.
- In the beginning of the First Innings, these are passed: Strike Batter, Non-Strike Batter, Bowler.
- In the beginning of the Second Innings, these are passed: Strike Batter, Non-Strike Batter, Opening Bowler.

9.2.4 *showNamePrompt()*

- It sets the *modal* style to flex from none (to make it visible), and makes the *nameInputs* innerHTML attribute to an empty string.
- It uses the argument it got from *showNamePrompt()* function and adjusts the text for the fields
- Creates a label for the input fields by adding a ':' to each element of the argument list, and appends it to *nameInputs*
- It then creates the corresponding input fields with respective ids, and adds a placeholder text.
- It updates *modalTitle* based on whether it is the First or Second Innings to the appropriate title.

9.2.5 *showModal(title,message,callback)*

- This function is used to populate the content of a custom modal.
- This custom modal present in the Live Page is used to let the user know if the First Innings or the Match has ended.
- The arguments *title* and *message* are used to populate the *custom-modal-title* and *custom-modal-message* respectively.
- The modal has a Continue Button with id *continue* to Resume the Match, the *callback* argument is used to call a particular function when the Continue Button is clicked.

9.2.6 *showModalError(message)*

- This function takes the argument *message* and displays it on a modal.
- It creates an *errorElement* with an id and *textContent* as *message* argument, and it is also styled to make it look like an error.
- It then appends it to the Modal on the page
- If any input is being typed after the display of the message, then the message is automatically made empty using an *EventListener*.

9.2.7 *submitPlayerNames()*

It checks for the correctness of the player names entered, and then stores them

This is for the First Innings:

- It stores the corresponding data in: *strikeBatter*, *nonStrikeBatter* and *bowler*.
- If either of the names are missing, it shows an error message in the modal using *showModalError()*.
- If the entered names of *strikeBatter* and *nonStrikeBatter* are same, it shows an error message.
- If the entered names are matching the *teamNames* or if the names are already used in another team, it shows an error.
- It decides the *currentBatting* and *currentBowling* teams based on *matchData* which was initialized in *loadMatchData()* function.
- It initializes the remaining data such as runs, balls, maidens, overs, etc. of the batsmen and bowlers.
- It closes the *modal* by setting its display back to none.
- It calls *saveMatchData()* and *updateDisplay()*.

This is for the Second Innings:

- It is same as the First Innings for the following: storing data, error handling, and initialising players.
- It closes the *modal* by setting its display back to none.
- It calls *saveMatchData()* and *updateDisplay()*.

9.2.8 *isDuplicatePlayer(name,team)*

- This function returns a boolean value.
- It takes *name* and *team* as arguments and checks if the given name is already used or not.
- If the function returns true, then that *name* is not valid for that *team* (Eg: batting)
- And it is valid if it returns false.

9.2.9 *showNewBatterPrompt()*

- This function is used to show a prompt when a batsman gets out, and handle the data insertion.
- It checks if the entered name is valid or not using *isDuplicatePlayer()* and shows error if not.
- If the name is valid then it adds the name along with the initialized data of the *new-batter* to the *matchData*.
- Finally, it resets the modal to hidden and calls *saveMatchData()* and *updateDisplay()*.

9.2.10 *showNewBowlerPrompt()*

- This function is used to show a prompt when an over gets completed, to add a new bowler and handles the data insertion.
- It performs all the functions of *showNewBatterPrompt()* function for bowlers.
- Additionally, if it is not the End of the Match, then it re-enables the buttons (because they might have been disabled by other functions at this stage)

9.2.11 *checkForEndOfMatchOrInnings()*

- It stores the truth values of either the overs completing and all out in: *oversCompleted* and *allOut* respectively.
- In the first innings, if either *oversCompleted* or *allOut* is true, then it sets *matchData.allOut* to true (The way the functions work, just a convenience), calls *startSecondInnings()* and returns *true* for the function.
- In the second innings, if target is reached then, it calls *endMatch()* with Result Statement as argument
- If either the overs are completed or all out, it checks for the possibility of Draw and calls the *endMatch()* with appropriate argument and finally returns true for the function.
- In any other case, it returns false.

9.2.12 *startSecondInnings()*

- This function handles the transition from the first innings to the second.
- It stores the necessary data like *firstInningsTotal*, *firstInningsWickets*, *firstInningsBatters* in new attributes in the *matchData* object, as the attributes in which the old data is present will be overwritten.
- It then re-initializes all the attributes of *matchData* for the second innings.
- It swaps the *currentBowlingTeam* and *currentBattingTeam* and sets the names of the opening batsmen and bowler to *null*.

- It notifies the user that the first innings has ended with a message that also shows the target for the second innings.
- It calls `showModal()` with the *message* as innings break and *callback* as `promptForPlayerNames`

9.2.13 `endMatch(message)`

- This function handles the end of match, it sets `matchData.matchOver` to true
- It populates the `matchData.statusMessage` with the *message* argument it is called with.
- It calls `saveMatchData()` and `updateDisplay()`.
- It calls `showModal` with the *message* as Match Over and *callback* as Time-Delay for 2s before redirecting to Summary Page.
- It disables all the control buttons to prevent any further change.

9.2.14 `addRuns(runs)`

- This function updates the scores of batsmen and bowler based on value of the run button clicked, it takes the value of this button as its *runs* argument.
- It returns (no change) if either: the match is over, all out, or if the *currentStriker* is not properly set (null).
- It also calls the `consumeFreeHitIfNotExtra(false)` which will remove any Free Hit Indicator message in the Live Match Display.
- It updates the batter's runs and balls and updates 4s and 6s if required
- It updates the bowler's stats as well.
- It updates the *totalRuns* and *balls* of `matchData`.
- It calls `addCommentary(runs)` with *runs* as argument.
- It checks for end of match or innings and if true, then calls `saveMatchData()`, `updateDisplay()` and returns.
- If odd runs are scored or end of over, then it switches the two batter's positions.
- It also updates the overs of the bowler, if the bowler is valid and he is the *currentBowler*.
- At the end of an over, if it is not an end of the match or innings, it prompts for the new bowler's name.
- It calls `saveMatchData()` and `updateDisplay()` outside of all the conditional if-else statements in this function.

9.2.15 *takeWicket()*

- This function updates the data of the bowler and sets the *currentStriker* batsman as out
- It returns (no change) if either: the match is over, all out, *matchData.wickets* $i = 10$ or if *currentStriker* is not properly set (null).
- It calls *consumeFreeHitExtra(false)* which will remove any Free Hit Indicator message in the Live Match Display.
- It sets the *currentStriker* to out and calls *addCommentary('W')* function
- It then checks for end of match or innings and if true, then calls *saveMatchData()*, *updateDisplay()*.
- If *matchData.wickets* $i > 10$ then calls *showNewBatterPrompt()* else it sets *matchData.allOut* to true.
- It checks for End of Over, and swaps the *currentStriker* and *nonStriker*, updates the bowler stats and calls *showNewBowlerPrompt()*
- It then adds the new bowler to the list of bowlers in *matchData.bowlers*
- It finally calls *saveMatchData()* and *updateDisplay()* to update changes.

9.2.16 *addWide()*

- This function updates the player data according to a Wide
- It returns (no change) if either: the match is over, all out or *currentBowler* is not properly set (null).
- It calls *consumeFreehitExtra(true)* so that the Free Hit will carry forward to the next ball as well.
- It increments the *matchData.extras*, *matchData.totalRuns* and *matchData.currentBowler.runs* by 1 each.
- It finally calls: *addCommentary('Wd')*, *saveMatchData()* and *updateDisplay()*.

9.2.17 *addNoBall()*

- This function is implemented a bit differently compared to conventional No Ball in Cricket Matches
- It is considered as an extra, and 1 run is awarded to the Batting Team
- It sets the next ball after a No Ball as a Free Hit and indicates it with an indicator near the display, but the batsman can get RunOut or his Wicket may be taken in this ball if the user presses those buttons.
- If the next ball is a Wide or a No ball again, then the Free Hit gets carried forward.
- It calls *consumeFreeHitIfNotExtra(true)*, *addCommentary('Nb')*

- It sets the *matchData.isFreeHit* to true, and finally calls *saveMatchData()* and *updateDisplay()*.

9.2.18 *consumeFreeHitIfNotExtra(isDeliveryExtra)*

- This is a helper function that is used to reset the Match Status after the Free Hit is consumed.
- It checks if *matchData.isFreeHit* is true, and if *isDeliveryExtra* is false and then sets *matchData.isFreeHit* to false.
- This is because the ball was a free hit and this delivery is not a Wide or No Ball, so Free Hit is consumed
- Else, the delivery was an extra and *isFreeHit* remains true.

9.2.19 *promptForRunOut()*

•

9.2.20 *processRunOut(runs,newBatterName)*

•

9.2.21 *addCommentary(eventCode,details{})*

•

9.2.22 *updateDisplay()*

It just calls the following functions:

- *updateScoreDisplay()*
- *updateBattersTable()*
- *updateBowlerTable()*
- *updateCrrRrr()*
- *updateCurrentOver()*
- *updateStatusDisplay()*

The *saveMatchData()* function stores the *matchData* object as *cricketMatchData* in the *localStorage* of the Web Browser.

In the Live Page, there is an *Event Listener* that listens to the click of keys and if there are any message-modals in the page, then the user can just press enter instead of manually clicking the button to submit or continue.

9.3 Scorecard Page:

For this page there is a small inline script that has an *EventListener*, listening for the *DOMContentLoaded* event. It calls the *loadScorecard()* function.

9.3.1 *goBack()*

This function is called when the Back Button is clicked in the Scorecard Page. It uses *window.history.back()* to go back to the page before the View Scorecard button was clicked. If the page is called from the summary, it redirects to the Summary Page and if called from live match, it redirects to Live Page.

9.3.2 *formatOvers(balls)*

This function takes *balls* as an argument and returns a corresponding string that represents the number of overs that the bowler has bowled in the match.

9.3.3 *loadScorecard()*

- It first gets the *cricketMatchData* from the *localStorage* of the browser and stores it in *MatchData* object (If it is not there, then the function returns)
- It then extracts the data of all the batsmen and bowlers who have played until now, and stores them in separate arrays: *firstInningsBatters*, *secondInningsBatters*, *firstInningsBowlers* and *secondInningsBowlers*.
- Before starting to populate the tables with this data, it first makes their *innerHTML* empty.
- Then for each batsman in the array, it calculates his *strikeRate* and populates his row of the batting table.
- Similarly for the bowlers, it calculates the *economy* and fills in the bowling table.
- It then calls the *setupCommentaryHover()* function.

9.3.4 *setupCommentaryHover()*

- It extracts the *cricketMatchData* from the *localStorage* and stores it in a local variable *MatchData*
- It then groups the commentaries present in the *MatchData.commentary* based on the batter and bowler names, and stores it in 2 maps.
- The player's name is the key and the list of commentaries related to him is the value for the Map.
- It then attaches hover events to each batsman and bowler rows, this is done using the *mouseenter* and *mouseleave* option in the *addEventListener*.
- So, now when we hover on a player's row, his related commentaries will popup in a box to the right of the table.

9.4 Summary Page:

For this page, there is a small inline-script that is present in the HTML file itself. It calls the following functions: *loadMatchData()*, *displayResult()* and *setupSummaryPage()*.

9.4.1 *displayResult()*

- It stores the reference to the *match-result* element present on the page in a variable *resultDisplay*.
- If this element is loaded on the page, then it gets the match data from the *localStorage* of the browser (the data is in the *JSON String cricketMatchData*)
- Then it changes the *textContent* of *resultDisplay* to the *statusMessage* of the *cricketMatchData*

9.4.2 *setupSummaryPage()*

- It checks if either the 'Reset' or 'View Scorecard' buttons are clicked
- If the 'Reset' button is clicked, it first asks the User to confirm if he wants to Reset and then clears all the data and redirects to the Setup Page
- If the View Scorecard button is clicked, it then redirects to the Scorecard Page

10 FlowChart Representation of Functions in Live Page

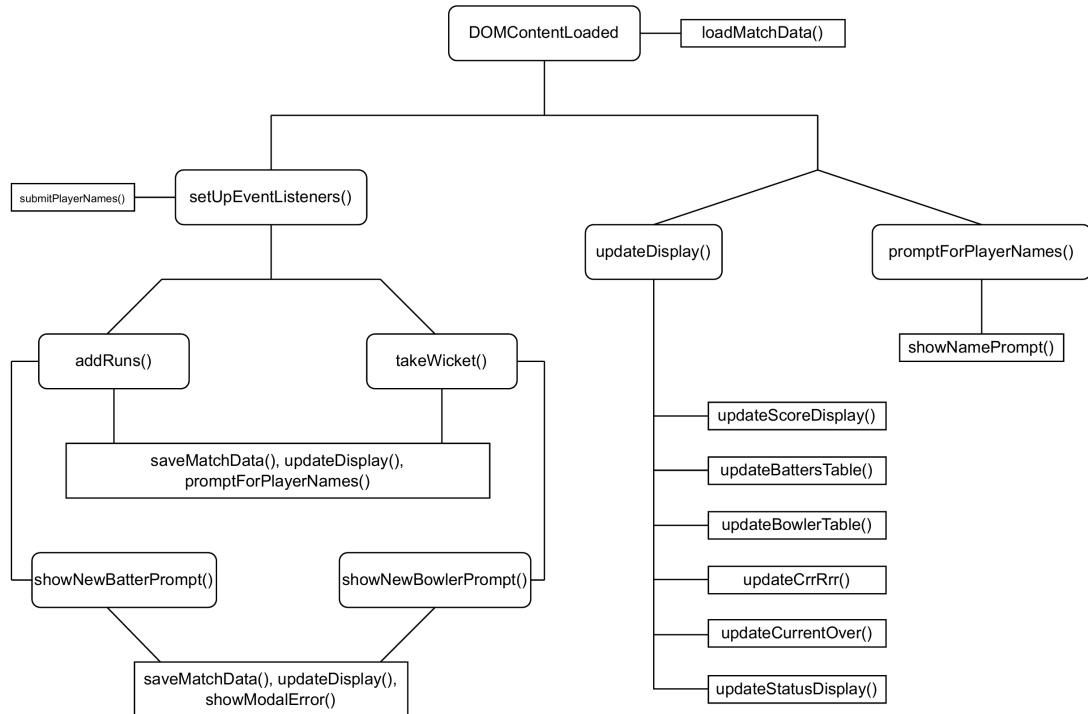


Figure 6: Flowchart Representation of JS Functions in Live Page

11 References

- Background Images: www.wallpaperflare.com
- <https://www.espncricinfo.com>
- <https://developer.mozilla.org>