Howzatt! Cricket Scorekeeper

A Web-based Scoring System

Sumukh M G

April 30, 2025

Contents

Chapter 1

Introduction

The **Howzatt! Cricket Scorekeeper** is an interactive browser-based application designed to replicate a cricket match scoring system. Built using foundational web technologies — HTML, CSS, and JavaScript — it guides users through setting up teams, scoring ball-by-ball, viewing detailed scorecards, and summarizing the result.

The app offers:

- A form-driven match setup
- Live scoring with wickets, runs, and extras
- Automatic scorecard generation
- Match summary and reset option

Chapter 2

Page Structure and Navigation

2.1 Main Pages

- **Setup Page** — Takes inputs for configuring start match
- **Live Match Page** — Live scoring page with all match controls
- **Score Card Page** — Tabular view of match performance
- **Summary Page** — Final Match result page with reset option

2.2 Navigation Flow

1. We should start with the Setup Page.
   Only it can collect the data needed for the start of the match.
2. Then we move on to the Live Match Page.
   It is where all the events of the match take place.
3. We can see the Score Card Page.
   It contains all the score of the match that happened.
4. After the match finishes we are directly redirected to the Summary Page.
   It contains the result of the match (i.e which team won and which team lost).

Chapter 3

HTML and CSS Design

Each HTML page is paired with a dedicated CSS file to enhance layout and appearance.

3.1 HTML Structure

HTML defines the page skeleton:

- `<div>`, `<form>`, `<input>` used for user interface
- `<table>` for score representation

3.2 CSS Styling

Each CSS file focuses on styling a specific page:

- `setup.css` — Form design and centered layout
- `live.css` — Scoring buttons, tables, commentary box
- `scorecard.css` — Clean score tables and headers
- `summary.css` — Minimalist final result display

Chapter 4

Utility Functions in JavaScript

Utility functions in `score.js` help manage application state and shared logic.

4.1 saveData()

This function serializes the current match data (teams, players, scores, overs, etc.) and stores it in the browser's `localStorage`. It ensures continuity even if the page reloads.

```
localStorage.setItem(key, JSON.stringify(value));
```

4.2 loadData()

```
JSON.parse(localStorage.getItem(key));
```

The reverse of `saveData()`. It fetches data from `localStorage` and restores the app's variables to resume a match mid-way.

4.3 clearData()

```
localStorage.clear();
```

Clears all saved session data. Used during match reset or setup to ensure no stale information carries over.

Chapter 5

Setup Page Objects

This part initializes all match-level configuration.

5.1 Team Setup

On submission of the setup form, the website collects:

- Team 1 and Team 2 names
- Toss winner and decision (bat/bowl)

This is saved in global variables like:

```
team1 = document.getElementById('team1').value.trim();
team2 = document.getElementById('team2').value.trim();
tossWinnerID = document.getElementById('tossWinner').value;
tossDecision = document.getElementById('tossDecision').value;
tossWinner = tossWinnerID === 'team1' ? team1 : team2;
tossLoser = tossWinnerID === 'team1' ? team2 : team1;
```

and then finally given to

```
firstInningsTeam=tossDecision === 'bat' ? tossWinner : tossLoser;
secondInningsTeam=tossDecision=== 'bat' ? tossLoser : tossWinner;
```

which are then used to make object **"match"** which is used globally in all other functions and files.

5.2 Match Object Configuration

Based on toss result, batting and bowling teams are assigned. Match-level stats like overs, innings number, and target (for 2nd innings) are also initialized.

```
match = {
  overs: 2,
  currentInnings: 1,
  innings: [
    { team: firstInningsTeam, runs: 0, wickets: 0, balls: 0,
       batsmen: [], strikerIndex: 0, nonStrikerIndex: 1,
       freehit: false, bowlers: [], extras: 0, commentary: [] },
    { team: secondInningsTeam, runs: 0, wickets: 0, balls: 0,
       batsmen: [], strikerIndex: 0, nonStrikerIndex: 1,
       freehit: false, bowlers: [], extras: 0, commentary: [] }
  ]
};
```

5.3 Player Arrays

Each team has:

- A `batters[]` array with objects like: `name, runs, balls, fours, sixes, status`
- A `bowlers[]` array with: `name, balls, runs, wickets, maidens`

These arrays are later updated during gameplay to take in new batsmen or bowlers and many other cases.

Chapter 6

Live Scoring Functions

6.1 Function handleBallEvent

This contains all the main logic for the implimentations of the below functions and the logic included in this is explained in them.

6.2 addRuns(n)

Triggered by buttons, this function adds `n` runs to:

- The striker's total
- The team's total
- Updates strike if `n` is odd

6.3 wicket()

- Marks the current striker as out
- Increments team and bowler wickets
- Prompts for a new batsman

6.4 runOut()

Opens a propmt modal asking for:

- Who got run out
- Runs completed
- Name of new batsman

6.5 confirmRunOut()

- Finalizes data entered in the modal, updates batter list and score accordingly.
- Contains all the logic for handling of the current delivery as it doesn't use the handleBallEvent() function.

6.6 Extras

- **wide()** — Adds 1 run to team and bowler but doesn't count as legal delivery
- **noBall()** — Similar to wide, but also allows an additional delivery
- **byes(), legByes()** — Runs added to team, not batter

6.7 checkInningsEnd()

Used after each delivery to check:

- If 10 wickets are lost
- Or overs are completed
- Or if the target is achieved

If so, it ends the innings or match.

6.8 Helping Funtions

- **getBowler()** — Is used when new bowler is needed i.e after an over. It contains all the logic for accepting new bowlers with same name, reusing old bowlers, etc.,
- **updateStrikerReferences()** — After each delivery, this function checks and updates which batter is on strike, adjusting for odd runs, dismissals, or over completions.

Chapter 7

Scorecard Logic

7.1 displayScorecard()

Contain the main logic for displaying the scorecard.

- Reads all data from player arrays and formats them into tables using javascript and **document.createElement()** dynamically instead of making table in scorecard.html and then adding values using javascript.

Sections include:

- Batting summary (Name, Runs, Balls, Status)
- Bowling summary (Name, Overs, Wickets, Economy)

7.2 Formatting Details

- Strikethrough or "not out" for live batters
- Sort players in order of appearance
- Economy calculated as `runs / overs`

Chapter 8

Summary Page Logic

8.1 Winner Declaration

When the second innings ends:

- Team scores are compared
- Tie, Win, or Loss is computed
- Message is shown in `id="result"`

8.2 resetMatch()

- Contains a button that completely clears all stored data using `clearData()` redirects user to the setup page for a fresh start.

Chapter 9

Challenges Faced

- Keeping striker/non-striker logic accurate using indexing for choosing striker and non-striker from the batsmen array.
- Making customizations for no ball, and also taking care of same bowler name case (new bowler/repeat bowler) for multiple over match.
- Had to learn to also refresh the scorecard page every time any change is made in the live page to accomodate the changes.
- Customizing run-out was the most hassle as it required creating logic for many cases like who gets out (striker/non-striker) and the strike change and also adding a popup form in html

Chapter 10

Conclusion

The project delivers a fully functional and interactive cricket scoring platform for browsers. It consolidates:

- Frontend design principles
- JavaScript DOM manipulation
- App logic modeling
- Persistent local storage use

It provides a solid foundation for further enhancements like player stats tracking, over filters, or commentary logging.

Building this website from scratch has been an incredibly rewarding and transformative experience. I not only developed my technical skills — like debugging large codebases even when editors like VS Code didn't highlight obvious issues — but also gained a deeper appreciation for clean, modular code design. This project taught me how to structure logic efficiently, manage memory wisely, and think like a developer. Beyond programming, it even deepened my understanding of cricket — a sport I hadn't fully explored before. Overall, this journey has been filled with growth, problem-solving, and the joy of turning an idea into a working web application.

Chapter 11

References

- https://www.cricbuzz.com/
- https://www.espncricinfo.com/
- https://www.overleaf.com/
- https://tex.stackexchange.com/