Project Report: **Howzatt! Cricket Scorekeeper**

Prashant Meena

April 29, 2025

Contents

1 Introduction

This project, titled **Howzatt! Cricket Scorekeeper**,was developed as the final submission for the CS104 course.

Cricket is one of the most followed sports in the world, and scoring is an integral part of the game. However, for casual or amateur-level matches, scorekeeping is often done manually, which can lead to inaccuracies and inefficiencies. The goal of this project was to apply the programming concepts learned throughout the semester to build a practical and user-friendly digital cricket scorekeeping application.

Howzatt! is designed to track live match statistics such as runs, wickets, overs, and player performances. The program allows for seamless entry and real-time updates, ensuring a smooth scoring experience. It reflects the application of conditional statements, loops, functions, data structures, and basic input/output — all key topics covered in the CS104 course.

2 Technologies and Tools Used

The **Howzatt! Cricket Scorekeeper** project was developed using a combination of front-end web technologies, enabling a responsive and interactive user experience. The key technologies and tools used in this project include:

- **HTML (HyperText Markup Language)**: Used to structure the content and layout of the application interface.
- **CSS (Cascading Style Sheets)**: Employed for styling the interface, ensuring that the application has a clean, intuitive, and user-friendly design.
- **JavaScript**: Responsible for the core logic of the application, including real-time updates, calculations, and interaction handling (e.g., scoring updates, wicket tracking, and overs management).
- **Session Storage**: A built-in browser storage mechanism used to temporarily store match data (such as scores, overs, and player stats) during the session. This enables persistent data access throughout the match without needing a backend or database.

These technologies collectively allow the application to function entirely on the client side, making it lightweight, fast, and easily accessible through any modern web browser.

3 Code Structure

The project is organized into multiple HTML and CSS files, each responsible for a specific part of the application's flow — from initial input to live score display, detailed scorecard viewing, and final match summary. Below is a brief overview of the files and their roles:

- **setup.html** and **setup.css**: These files form the input page where the user can enter initial match information such as team names,toss winner team etc. The setup page provides the interface for configuring the match before it begins.

- **live.html** and **live.css**: Once the match starts, the application redirects to the live scoring page. first input of players (five-a-side) name of both teams is taken .These files handle the live display and updating of scores, wickets, overs, and other in-match statistics in real time.

- **scorecard.html** and **scorecard.css**: This page shows a detailed scorecard including per-player performance, fall of wickets . It is updated dynamically as the match progresses, allowing users to view historical match data.

- **summary.html** and **summary.css**: After the match concludes, users are redirected to the summary page, which provides final result as well as a concise post match interview. It also includes a **Reset** button that clears the session storage and navigates the user back to the setup page to begin a new match.

The files are interconnected using score.js to manage transitions between pages, update the UI in real time, and store match data using session storage. This modular file structure helps maintain code clarity and separates concerns across different parts of the application.

4 Implementation Details

The **Howzatt! Cricket Scorekeeper** is implemented using HTML, CSS, and JavaScript. The core logic revolves around user input collection, live match state management, and dynamically rendering updates to the UI. All match data is stored in the browser's session storage, allowing data persistence across multiple pages during a single session.

4.1 Match Setup and Initialization

The setup process begins on `setup.html`, where users input team names, number of players, and overs. JavaScript retrieves this data and stores it in session storage for later access.

Once data is validated and stored, the page redirects to `live.html` to begin scoring.

4.2 Live Match Logic and Scoring

On the `live.html` page, JavaScript handles dynamic scoring updates, including runs, balls, overs, and wicket tracking. The data is continuously updated in session storage and reflected in the UI.

Listing 1: Updating runs and overs

```
function updateScore(event) {...}
let eventButtons = document.getElementsByClassNam("event");
for (let button of eventButtons) {
button.addEventListener("click", updateScore); // Add click listener
} // may cause performance issues if many buttons are there
```

4.3 Scorecard Generation

The `scorecard.html` file accesses match data from session storage to display a complete batting and bowling breakdown. Player-wise runs, balls faced, strike rate, and fall of wickets are shown in a structured format.

4.4 Match Summary and Reset

After the final over is completed or all players are out, the user is redirected to `summary.html`, which shows the result (e.g., which team won and by how many runs or wickets). A **Reset** button clears session storage and redirects the user back to `setup.html` to start a new match.

Listing 2: Resetting the match

```
function resetMatch() {
    sessionStorage.clear();
    window.location.href = "setup.html";
}
```

5 Challenges Faced

While developing the **Howzatt! Cricket Scorekeeper**, several challenges were encountered, particularly in implementing match logic and managing data across different pages. Some of the key difficulties and how they were addressed are outlined below:

- **Managing State Across Pages:** Since the application spans multiple pages (setup, live score, scorecard, summary), maintaining consistency of match data without a backend was initially a challenge. This was solved using **sessionStorage**, which allowed data to persist across pages during a single session.
- **Handling Over and Ball Calculations:** Implementing the logic to accurately track overs and balls (e.g., 1.3 overs = 1 overs and 3 balls) required careful consideration to ensure that the display and internal calculations remained accurate and synchronized.
- **Updating UI in Real-Time:** Dynamically updating scores, wickets, and overs in real-time while ensuring smooth transitions and a responsive UI required precise DOM manipulation using JavaScript. It also involved ensuring no delays or misalignments occurred when data changed quickly.
- **Designing a User-Friendly Interface:** Ensuring the interface was intuitive for users unfamiliar with digital scorekeeping required multiple iterations of layout and styling using CSS.
- **Edge Cases and Bug Fixes:** Handling cases such as all players being out before overs complete, invalid inputs on setup, or trying to go to the next ball after the match had ended, required robust conditional logic and testing.

Each of these challenges contributed to a deeper understanding of front-end development concepts and improved the overall quality and usability of the application.

6 Results and Output

7 Conclusion

The **Howzatt! Cricket Scorekeeper** project successfully demonstrates the ability to build an interactive, multi-page web application using front-end technologies. It allows users to simulate a full cricket match—starting from setting up teams, tracking live scores, viewing a detailed scorecard, and finally displaying a match summary.

This project reinforced core concepts of HTML, CSS, and JavaScript, including DOM manipulation, event handling, and session-based data storage. By managing complex logic like over progression, innings tracking, and conditional flows, the project mirrors real-world use cases in web development.

Overall, the project not only met the functional requirements but also offered practical experience in problem-solving, UI design, and structuring modular code.

8 Future Work

While the current version of the **Howzatt! Cricket Scorekeeper** meets its core objectives, there are several improvements and extensions that could enhance its functionality and user experience:

- **Persistent Storage:** Implementing localStorage or integrating with a backend database would allow users to save and retrieve past match records, enabling long-term data analysis.
- **User Authentication:** Adding login and account features would enable multiple users to track their match histories and stats over time.
- **Responsive Design:** Improving the layout for better mobile compatibility and accessibility on different screen sizes.
- **Match Highlights and Analytics:** Introducing graphical analytics such as run rate graphs, player performance charts, and top scorer tables.
- **Multiplayer or Umpire Mode:** Enabling live scoring by different users or adding umpire validation for more realistic match simulation.
- **Sound Effects and Animations:** Adding sound and animations for events like sixes, wickets, or match-end celebration to improve interactivity and engagement.

These enhancements could help evolve the project into a more robust cricket simulation platform and provide a richer user experience.

9 References

1. **HTML, CSS, and JavaScript Fundamentals**
   - W3Schools. "HTML Tutorial." Available at: `https://www.w3schools.com/html/`

- W3Schools. "CSS Tutorial." Available at: https://www.w3schools.com/css/
- W3Schools. "JavaScript Tutorial." Available at: https://www.w3schools.com/js/

2. **Session Storage and Web Storage API**

- Mozilla Developer Network (MDN). "Window: sessionStorage property." Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage
- W3Schools. "HTML Web Storage API." Available at: https://www.w3schools.com/html/html5_webstorage.asp
- GeeksforGeeks. "JavaScript sessionStorage." Available at: https://www.geeksforgeeks.org/javascript-sessionstorage/
- Mimo. "JavaScript Session Storage: Syntax, Usage, and Examples." Available at: https://mimo.org/glossary/javascript/session-storage

3. **Additional Learning Resources**

- JavaScript Tutorial. "A Practical Guide to JavaScript sessionStorage." Available at: https://www.javascripttutorial.net/web-apis/javascript-sessionstorage/
- FreeCodeCamp. "Web Storage Explained – How to Use localStorage and sessionStorage in JavaScript." Available at: https://www.freecodecamp.org/news/web-storage-localstorage-vs-sessionstorage-in-javascript/