Rollno:160123737155

EAD ASSIGNMENT-1
CRICKET SCOREBOARD REPORT

1.Introduction

This project is a **Cricket Scoreboard Web Application** developed using HTML, CSS, and JavaScript. It is designed to simulate real-time cricket scoring by tracking runs, wickets, overs, and player scores for two players, Rahul and Rohit. The scoreboard also implements cricket rules such as wides, no-balls, byes, leg byes, free hits, and striker switching. The main objective of the project is to create a responsive, user-friendly, and rule-compliant application that provides an interactive experience similar to a real cricket match scoreboard.

2.Approach Taken

The Cricket Scoreboard was developed using a simple **frontend stack (HTML, CSS, JavaScript)** with a focus on clarity, responsiveness, and accurate cricket rule implementation.

1. Technology Stack

- **HTML5** → Defined the structure of the scoreboard, including team score, overs, player cards, and control buttons.
- **CSS3** → Styled the interface with a card-based layout, responsive design, and visual highlights such as striker indicators and free-hit labels.
- **JavaScript (Vanilla)** → Handled all game logic, dynamic updates to the DOM, and implementation of cricket rules such as runs, extras, wickets, striker switching, and free hits.

2. Layout and User Interface

- A **header** with the title for context.
- A **scoreboard container** showing team runs/wickets and overs at the top.

- **Player cards** displaying Rahul and Rohit’s runs, with a * to indicate the current striker and “Out” status when dismissed.

- A **grid of control buttons** to add runs, extras (wide, no-ball, bye, leg-bye), free hit, wickets, striker switch, and reset.

- A **message/status area** to give quick feedback (e.g., “Wicket ignored on Free Hit”).

- The UI was designed to be **intuitive and mobile-friendly** using Flexbox, Grid, and media queries.

3. Game Logic

The scoring logic was modularized into small functions for clarity and easier debugging:

- **handleRun(n):** Adds runs to team and striker, increments ball count, and switches striker on odd runs.

- **handleWide / handleNoBall:** Add extras to the team without counting as a legal delivery.

- **handleBye / handleLegBye:** Add team runs and increment ball; odd runs switch strike.

- **handleWicket / handleLBW:** Increment wicket count, mark the striker as out, and set Rahul as the new striker (simulating a new batsman).

- **handleFreeHit:** Adds one run and sets the next delivery as a free hit (no wicket, no ball increment).

- **switchStriker:** Allows manual rotation of strike, only if both players are active.

- **incrementValidBallIfNeeded:** Tracks overs and balls correctly (6 balls per over).

- **render():** Updates the DOM after every action to reflect the latest scores and statuses.

3.Challenges Faced and Solutions

1. Handling Cricket Rules Accurately

Challenge: Differentiating between valid and invalid deliveries (e.g., wides and no-balls should not increment the ball count, but byes and leg byes should).
Solution: A helper function incrementValidBallIfNeeded() was created to manage ball progression consistently, ensuring overs were updated correctly while following cricket rules.

2. Implementing Free Hit Logic

Challenge: The free hit rule required that wickets should not be counted and the ball should not increment, but only for the *next delivery*.
Solution: A boolean flag freeHitNext was introduced. When a free hit is triggered, this flag is set to true and consumed on the next valid delivery, preventing wickets and ball increment only for that delivery.

3. Striker Switching

Challenge: Striker rotation needed to occur automatically after odd runs, while also allowing manual switching, without breaking when a player was dismissed.
Solution: A dedicated switchStriker() function was implemented, with conditional checks to ensure that only active (not out) players could take strike. This maintained accurate striker tracking across all scenarios.

4. Wicket and New Batsman Handling

Challenge: After a wicket, the striker had to be marked as out, the asterisk removed, and Rahul set as the new striker. Initially, Rahul could be “revived” if he was dismissed.
Solution: The wicket handler was corrected to mark the current striker out, then assign Rahul as the new striker. If Rahul himself was dismissed, his stats were reset to simulate a “new batsman Rahul.”

5. Responsive User Interface

Challenge: The scoreboard layout needed to remain clean and readable on smaller screens such as mobiles.
Solution: CSS Flexbox, Grid, and media queries were used to adjust font sizes, button arrangements, and spacing dynamically, ensuring a consistent experience across devices.

4.Conclusion

The Cricket Scoreboard project was successfully developed to simulate real match scoring using HTML, CSS, and JavaScript. It provides an interactive and responsive interface for tracking runs, wickets, overs, and player scores while following the rules of cricket such as extras, striker switching, wickets, and free hits. The project not only fulfilled the given requirements but also helped in understanding event-driven programming, DOM manipulation, and responsive web design. Overall, the assignment strengthened both technical skills and problem-solving ability by applying real-world rules in a software system.