Howzatt! Cricket Scorekeeper

Overview

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.

NOTE: All logic is to be written in score.js.

1. Setup Page

- **Files:** `setup.html`, `setup.css`, `score.js`
- **Purpose:** Collect match details before the game starts.

setup.html

- Display:
  - Text input field for **Team 1 Name**
  - Text input field for **Team 2 Name**
  - **Toss Winner** (Dropdown: Select Team 1 or Team 2.)
  - **Toss Decision** (Dropdown: Bat or Bowl)
  - **NOTE**: The match will be a **2-over game**.
  - **Start Match Button** → Clicking it should save entered details and navigate to `live.html`.

setup.css

- Style the setup form (centered on the page, clean layout).

score.js

- Ensure that you save the different variables accordingly. You will have to use it in the later parts.

2. Live Match Scoring Interface

- **Files:** `live.html`, `live.css`, `score.js`
- **Purpose:** Allow dynamic entry of match events and update score live.

live.html

Overall Scores

- During the first innings, the display should look like:
  CSK 50/3 (1.4) vs. RCB
- During the second innings, the display should look like:
  RCB 20/1 (0.5) vs. CSK 61/3 (2.0)

Batter Table

- **First row contains Strike Batter stats** (Runs Scored, Balls Faced, 4s, 6s, Strike Rate)
- **Second row contains Non-Strike Batter** stats (same stats).

Bowling Section

- Current **Bowler Name** and his stats (Overs, Maidens, Runs Conceded, Wickets, Economy Rate).

Scoring Buttons (Below the Tables)

- **Create buttons for 0 runs, 1 run, 2 runs, 3 runs, 4 runs, 6 runs.**
- Create a button for Wicket. Clicking this button will show a text box for the user to enter the name of the next batter.

Navigation Button

- Button to go to the **Scorecard Page (`scorecard.html`)**.

live.css

- Style the score display, tables, and buttons here.

score.js (Live Match Logic)

- Update score dynamically when buttons are clicked.
- Track overs, balls, wickets, and runs.
- Automatically rotate strike for odd runs (1, 3, 5).
- Prompt for **strike batter's name, non-strike batter's name** and **first bowler's name** at the beginning of the match.
- Prompt for a **new batter's name** when a wicket falls.
- Prompt for a **new bowler's name** at the end of an over.
- Calculate and display **Current Run Rate (CRR)** and **Required Run Rate (RRR)** (this is applicable only in the second innings).

3. Scorecard Page

- **Files:** `scorecard.html`, `scorecard.css`, `score.js`
- **Purpose:** Display a **detailed match summary** up to the current point.

scorecard.html

On Top:

- **Button to go back to `live.html`**

Full Batting Scorecard

- List **all batters**, including those who got out.
- Display **Runs, Balls Faced, 4s, 6s, Strike Rate** for each.

Full Bowling Scorecard

- List **all bowlers** who have bowled.
- Display **Overs, Maidens, Runs Conceded, Wickets, Economy Rate** for each.

score.js (Scorecard Logic)

- Display stored data for **all batters** and **all bowlers**.
- Ensure score updates match the live match data.

4. Match Summary Page

- **Files:** `summary.html`, `summary.css`, `score.js`
- **Purpose:** Display **match result** at the end and allow resetting.

summary.html

- Automatically display the **winner and match result**.

Result Format:

- If **Team A wins batting first**:
  `"Team A wins by X runs!"`

- If **Team B wins chasing**:
  `"Team B wins by X wickets (Y balls left)!"`

Navigation

- **Reset Match Button** → Clears all data and redirects to `setup.html`.

summary.css

- Simple styling to highlight the winner.

score.js (Summary Logic)

- Determine the winner based on stored scores.
- Display the correct match result.
- Implement reset functionality to start a new game.

NOTE: Try to be consistent in UI design. **You can create more CSS files than mentioned here, such as base_styles.css, which is embedded in every file.** This is so that the styles are uniform and user-intuitive.

References

- You can refer to websites like cricbuzz.com and espncricinfo.com for ideas on lay each page. **However, note that you are not expected to implement all the features on these pages.**
- **UPDATED ON 12/04/25:** To store data across webpages, you can use **Local Storage** or **Session Storage.** In a nutshell, `localStorage` and `sessionStorage` let websites store data in the browser as key-value pairs using JSON. `localStorage` keeps it even after the tab is closed, while `sessionStorage` clears it when the tab is closed.

  Here are some references to get you started:

  - http://freecodecamp.org/news/web-storage-localstorage-vs-sessionstorage-in-javascript/?utm_source=chatgpt.com
  - http://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage

Customizations

Students are encouraged to implement additional features to enhance their project. Below are some suggestions, but you are free to come up with your ideas:

Extras

- Can implement extras like wide, no-ball, byes, and leg-byes.
- In the case of a wide, the score would be incremented by 1, but the balls stay the same. Batter's score is unchanged; the run counts as extra.
- For no ball, similar to wide, but any additional runs scored on that ball count to the batter’s score. For example, if the batter hits a 4 on the no-ball, their score is increased by 4 runs only. Additionally, the next ball is a free hit.
- In the case of byes and leg-byes, it is counted as a ball being bowled, and the runs go as extras. It does not count towards the batter’s score.

Run outs:

- You can add an additional **'Run Out' button** next to the Wicket button. You can keep a small number input field before the button to say how many runs were completed before the run-out and use this to find which batter was out.

Live Commentary Feed

- Maintain a **ball-by-ball log** in the format:
  - `2.3 Bumrah to Head, 2 runs` *(Third ball of the third over: bowler Bumrah to batter Head, result: 2 runs)*
- Clicking on a **batter's name** filters the commentary to **only show balls faced** by that player.
- Clicking on a **bowler's name** filters the commentary to **only show balls bowled** by that player.

Multi-Match Data Storage

- Store **multiple match results** and allow displaying aggregate stats for players across matches, such as:
  - **Batters:** Total Runs, Average, Highest Score, Strike Rate
  - **Bowlers:** Wickets, Average, Economy Rate, 5-wicket hauls
- To save the effort of typing the same input for multiple matches, it may be better to write a seed script that automatically fills some stats for batters and bowlers. For example, you could simulate 5 matches where each ball's outcome is randomized and use this to get stats for the players.

Extended Match Formats

- Modify the scoring system to support **different match formats**, such as:
  - Changing the **number of overs**.
  - Implementing **Test match rules** (tracking innings, lead/trail, match days, follow-on, etc.).

Project Guidelines

- The **basic tasks are designed to be completed** in **regular HTML, CSS, and JavaScript**. However, students **may use additional tools or frameworks** for customizations.
- **Customizations are optional**, and students are **encouraged to make their base project interesting with them without going overboard.** However, it is recommended that you **complete the basic tasks first.**

Other Instructions: Below will be updated in 1-2 weeks based on feedback and student progress

- The project needs a report written in latex. Details of what this should contain will be provided later in this document itself
- The mark distribution for basic tasks (15 marks) will also be updated here.
- Customization will be extra credit and can compensate for poor performance in other exams. But this is capped at 3 Marks (20% of 15).
- Any corrections to the problem statement based on feedback will also be updated in this document, and the difference will be highlighted.