CS104 Project Report

Howzatt! Cricket Scorekeeper

Student ID: 23B0993

29 April 2025

1 Introduction

Cricket is a sport with a rich history and complex scoring system. For my CS104 project, I've developed a web-based cricket scorekeeper application called "Howzatt!" that allows users to track live cricket matches. The application provides real-time scoring, detailed player statistics, and match summaries.

The project was built using pure HTML, CSS, and JavaScript, with local storage for data persistence across different pages. The application follows a mobile-first design approach, making it accessible on various devices while maintaining a clean and intuitive interface.

2 Project Overview

The Howzatt! Cricket Scorekeeper consists of four main components:

1. **Setup Page**: Collects initial match details like team names, toss winner, and match format
2. **Live Match Page**: Provides real-time scoring interface with dynamic updates
3. **Scorecard Page**: Displays detailed statistics for all players in the match
4. **Summary Page**: Shows match results and provides option to start a new match

Each component has been built with usability in mind, featuring intuitive controls and responsive design elements. The application maintains consistent styling across all pages through a shared base CSS file.

3 Implementation Details

3.1 Data Structure

The core of the application relies on a JavaScript object that tracks the entire match state. This object is stored in the browser's localStorage to persist data between page navigations. The main data structure is outlined below:

```
let match = {
    team1: '',
    team2: '',
    toss: { winner: '', decision: '' },
    innings: 1,
    maxOvers: 3, // Default value, updated from setup
    current: {
        battingTeam: '',
        bowlingTeam: '',
        score: { runs: 0, wickets: 0, extras: 0 },
        overs: 0,
        balls: 0,
        striker: null,
        nonStriker: null,
        currentBowler: null,
        target: null
    },
    allBatters: [],
    allBowlers: [],
    commentary: []
};
```

This structure allows for efficient tracking of the match state, including player statistics, team scores, and over-by-over commentary.

3.2 Setup Page

The setup page uses a clean, glass-morphism design with form validation to ensure all required fields are completed before starting a match. The form collects:

- Team names for both sides
- Toss winner selection
- Batting/bowling decision
- Number of overs for the match

When the user clicks "Start Match," this data is stored in localStorage and the application transitions to the live match page.

3.3 Live Match Page

The live match page is the heart of the application, providing:

- Current score display with team names and overs
- Active batter statistics (runs, balls faced, boundaries)
- Current bowler statistics (overs, wickets, economy)
- Button controls for runs, wickets, and extras
- Live commentary feed

I implemented several key features for this page:

3.3.1 Dynamic Score Updates

Score updates are handled through event listeners on the scoring buttons. When a user clicks a runs button, the application:

1. Updates the striker's individual score
2. Increments the team's total runs
3. Updates the batter's balls faced
4. Tracks boundaries (fours and sixes)
5. Automatically rotates strike for odd-numbered runs
6. Updates the ball count for the over

3.3.2 Over Management

The application automatically manages overs, prompting for a new bowler when an over is completed (6 legal deliveries). It also handles strike rotation between overs and updates the current bowler's statistics.

3.3.3 Wicket Handling

When a wicket occurs, the application:

1. Updates the team's wicket count
2. Marks the batter as out
3. Prompts for a new batter's name
4. Updates the bowler's wicket count
5. Adds the dismissal to the commentary

If the tenth wicket falls, the innings is automatically ended.

3.3.4 Extras Implementation

I implemented extras (wide, no-ball, bye, leg-bye) that:

1. Update the team's total score
2. Are counted separately from regular runs
3. Don't add to the batter's score or balls faced (for wides)
4. Are properly reflected in the commentary

3.4 Scorecard Page

The scorecard page provides comprehensive statistics for all players who participated in the match:

- Batting scorecard showing runs, balls faced, boundaries, and strike rates
- Bowling scorecard showing overs, maidens, runs conceded, wickets, and economy rates
- Visual indicators for the current batters and bowlers
- Special styling for batters who are out

The page uses responsive tables that adjust for different screen sizes, ensuring all data is easily readable on any device.

3.5 Summary Page

The summary page displays the match result using the appropriate format based on the outcome:

- For a team winning batting first: "Team A wins by X runs!"
- For a team winning while chasing: "Team B wins by X wickets (Y balls left)!"
- For tied matches: "It's a draw! Both teams scored X runs."

The page includes a reset button that clears all match data and returns to the setup page for a new match.

4 User Interface Design

4.1 Design Philosophy

I approached the UI design with several key principles in mind:

- **Clarity**: Information is presented in a clear, organized manner
- **Consistency**: Similar elements maintain consistent styling across pages
- **Feedback**: User actions receive immediate visual feedback
- **Aesthetics**: Modern design elements enhance user experience

4.2 Color Scheme

The application uses a consistent color scheme defined in CSS variables:

```
:root {
    --primary: #3a0ca3;
    --primary-light: #4361ee;
    --secondary: #4cc9f0;
    --success: #06d6a0;
    --danger: #ef476f;
    --warning: #ffd166;
    --background: linear-gradient(135deg, #f8faff 0%, #edf2fb 100%);
    --text: #2b2d42;
    --border: #c8d3e5;
    --shadow: 0 4px 10px rgba(0, 0, 0, 0.08);
    --shadow-sm: 0 2px 4px rgba(0, 0, 0, 0.05);
    --glass: rgba(255, 255, 255, 0.8);
}
```

This approach ensures visual consistency and makes future styling updates simpler.

4.3 Responsive Design

The application is fully responsive, adapting to different screen sizes:

- Grid layouts adjust from multi-column to single-column on smaller screens
- Font sizes scale appropriately for readability
- Touch-friendly buttons for mobile users
- Appropriate spacing and padding for all device sizes

4.4 Animation and Transitions

To enhance user experience, I implemented subtle animations and transitions:

- Smooth hover effects on interactive elements
- Fade-in animations for new commentary entries
- Scale transitions for active player indicators
- Confetti animation on the summary page for the winning team

These effects provide visual feedback without being distracting.

5 Technical Challenges and Solutions

5.1 State Management

One of the biggest challenges was managing the application state across multiple pages. I solved this by:

- Using localStorage to persist data between page navigations
- Creating a structured data model that captures all match details
- Implementing functions to update and retrieve the state consistently

5.2 Player Rotation Logic

Cricket has complex rules for player rotation:

- Batters switch ends after odd-numbered runs
- Batters switch ends at the end of each over
- New batters must be added when wickets fall
- New bowlers must be assigned at the end of overs

I implemented this logic through careful conditional statements and array manipulations in the JavaScript code.

5.3 Dynamic Commentary

The live commentary feature required:

- Real-time updates after each ball
- Chronological ordering of entries
- Appropriate formatting for different types of events
- Automatic scrolling to show the most recent entries

I addressed these requirements by creating a commentary array in the match object and using DOM manipulation to render new entries.

6 Future Enhancements

While the current implementation meets all the project requirements, several enhancements could be added in the future:

- **Advanced Statistics**: Add more detailed statistics like partnerships, dot-ball percentages, and wagon wheels
- **Multi-match Storage**: Save multiple match results and generate player statistics across matches
- **Export Functionality**: Allow exporting match data as CSV or PDF
- **User Authentication**: Add user accounts to save personal scoring history
- **Offline Support**: Implement service workers for offline functionality

7 Conclusion

The Howzatt! Cricket Scorekeeper application successfully fulfills the project requirements by providing a comprehensive cricket scoring system with an intuitive user interface. The application accurately models the complexities of cricket scoring while maintaining a clean, user-friendly design.

Building this project has enhanced my understanding of front-end web development, particularly in areas of state management, responsive design, and event handling. The modular structure of the code also makes future extensions straightforward to implement.

The project demonstrates how pure HTML, CSS, and JavaScript can be used to create a functional, responsive web application without relying on heavy frameworks. This approach ensures fast loading times and broad compatibility across different browsers and devices.