

Howzatt Cricket Project Report

Hemant Godve
24B1004

April 2025

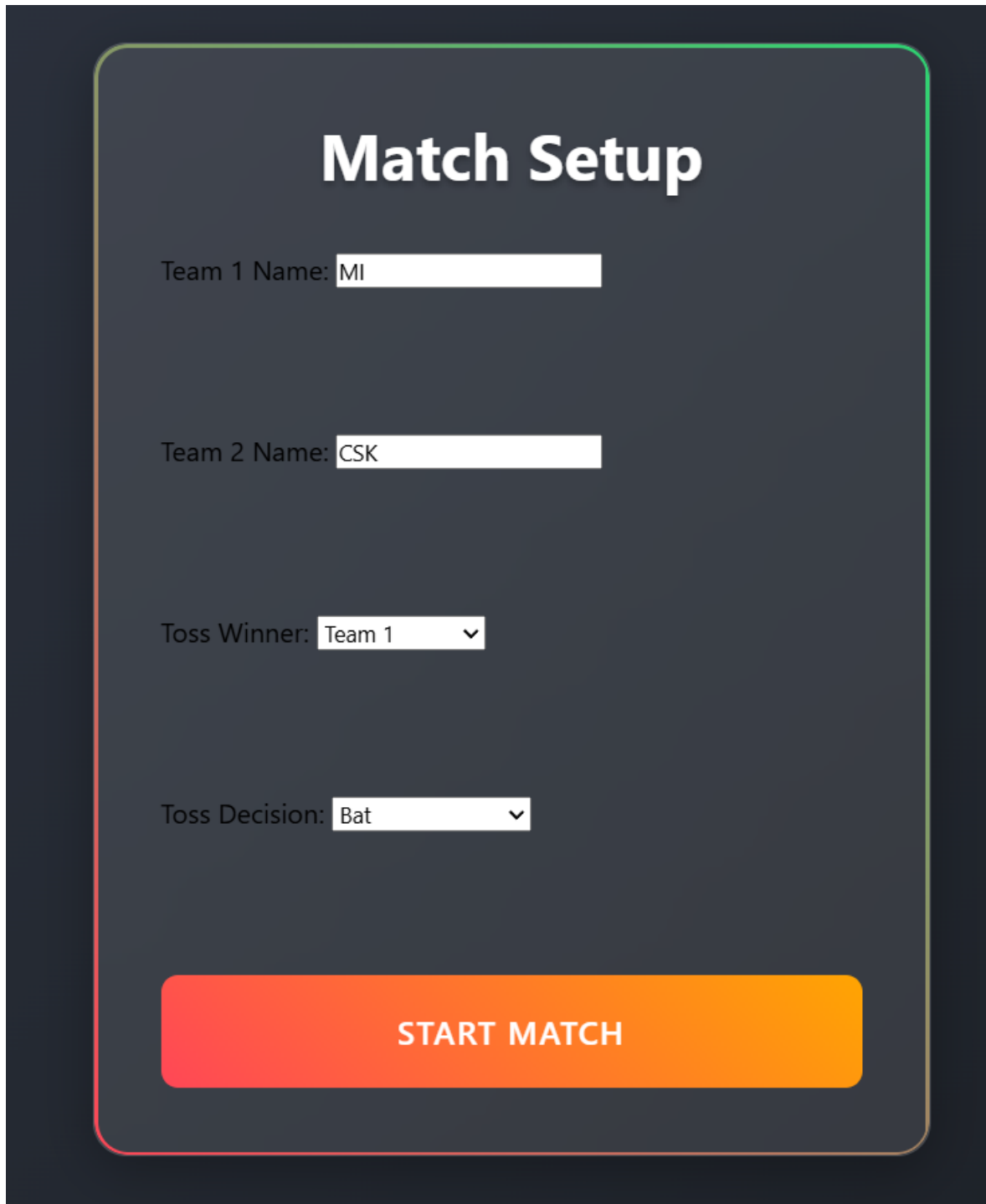
Abstract

This report provides a comprehensive analysis of a web-based cricket scoring system developed using HTML, CSS, and JavaScript. The system comprises four sequential modules: Match Setup, Live Scoring Interface, Scorecard Generation, and Post-Match Summary. Particular emphasis is placed on architectural decisions, the use of JavaScript for implementing logic and functionality.

Contents

1	Match Initialization (Setup Page)	2
1.1	Functional Overview	2
1.2	Technical Implementation	3
1.3	Data Flow Architecture	3
1.4	User Interface	3
1.5	Functionality	3
2	Real-Time Scoring Module (LIVE PAGE)	3
2.1	Core Features	3
2.2	State Management	3
2.3	Event Handling	4
2.4	Data Management	4
3	Scorecard Generation Module	4
3.1	Identified Limitations	4
3.2	Data Rendering Process	4
3.3	UI Components	4
4	Post-Match Analysis Module (Summary)	5
4.1	Result Computation	5
4.2	Result Calculation	5
4.3	Presentation Layer	5
4.4	Interface Design	5
5	System Workflow	5
6	Styling System	5
6.1	Key Features	5
7	Future Improvements and Challenges	5
7.1	Immediate Priorities	6
7.2	Long-Term Roadmap	6
7.3	Identified Issues	6
7.4	Recommendations	6
8	Challenges Faced	6
9	Material Used	6

1 MATCH INITIALIZATION (SETUP PAGE)

The image shows a 'Match Setup' form on a dark blue background. The form is contained within a rounded rectangle with a thin green border. At the top, the title 'Match Setup' is displayed in a large, bold, white font. Below the title, there are four input fields: 'Team 1 Name' with the value 'MI', 'Team 2 Name' with the value 'CSK', 'Toss Winner' with a dropdown menu showing 'Team 1', and 'Toss Decision' with a dropdown menu showing 'Bat'. At the bottom of the form is a large, rounded orange button with the text 'START MATCH' in white capital letters.

The setup page is responsible for collecting essential match parameters through structured user input fields.

1.1 Functional Overview

- Team Naming: Input fields for team names
- Toss Simulation: Dropdown menus for toss winner and decision
- Player Roster Initialization: Collecting initial player names

1.2 Technical Implementation

- Team Metadata: Storing team names and identifiers
- Toss Outcome and Decision: Storing toss winner and decision (batting or bowling)
- Innings Progression Tracker: Tracking innings progression
- Player Registry: Storing player information with role assignments

1.3 Data Flow Architecture

- Object Serialization: Using `JSON.stringify()` to serialize input data
- Contextual Redirection: Redirecting to live interface after setup

1.4 User Interface

- Team Name Inputs: Two text inputs for team names
- Toss Winner and Decision Dropdowns: Dropdown menus for toss winner and decision
- Gradient-Animated Submit Button: Animated submit button

1.5 Functionality

- Form Validation: Validating all form fields before submission
- Match State Initialization: Initializing match state with team configurations, empty arrays for batters/bowlers, innings counter, and score/wicket trackers
- Player Name Collection: Using browser prompts to collect initial player names

2 REAL-TIME SCORING MODULE (LIVE PAGE)

The dynamic interface provides granular control over match progression through various elements.

2.1 Core Features

- Ball-by-ball scoring mechanism
- Batter performance metrics (runs, strike rate)
- Bowler economy calculations
- Wicket management system
- Maidens over
- Dynamic scoreboard showing:
 - Current innings score (Runs/Wickets)
 - Balls bowled (Overs format)
 - Batting team vs bowling team
- Batter/Bowler statistics panel updating in real-time
- Six scoring buttons (0,1,2,3,4,6 runs) and wicket button

2.2 State Management

A singleton pattern is used to maintain the match state, ensuring that all critical data points are updated in real-time. This includes:

- Scoring: Runs, wickets, and other key metrics for each innings
- Player Stats: Detailed statistics for each player

- Over Progression: Tracking the number of overs bowled and remaining

2.3 Event Handling

User-initiated actions trigger various processes, including:

1. Scoring Runs: Updating the score based on runs scored
2. Wicket Management: Handling wickets, including updating team and player stats
3. Strike Rotation: Managing batter rotation and strike changes
4. Over Tracking: Checking for over completion and updating the game state

2.4 Data Management

The module maintains separate counters for:

- Innings Scores: First and second innings scores
- Individual Player Stats: Runs, balls, boundaries, and other key statistics
- Bowler Economy Rates: Calculating and displaying bowler economy rates
- Automatic Innings Switch: Switching innings after 12 balls
- Striker Rotation: Rotating strikers on odd-run scores
- Browser Storage Updates: Updating browser storage after every action

3 SCORECARD GENERATION MODULE

The scorecard generation module is responsible for displaying match statistics in a clear and organized manner.

3.1 Identified Limitations

- Partial Data Persistence: Incomplete transfer of match statistics from runtime memory to persistent storage
- Session Collisions: Potential data corruption during browser context switches
- Null Reference Exceptions: Unable to handle edge cases in player substitution scenarios
- Temporal Data Loss: Failure to maintain timestamped event logs

3.2 Data Rendering Process

- Data Retrieval: Getting saved data from browser storage
- Data Display: Filling webpage with retrieved data
- Table Creation: Creating tables to display scores and stats
- Stat Calculations: Calculating extra stats like economy rates and strike rates

3.3 UI Components

- Responsive Tables: Two tables for batting and bowling statistics
 - Batting table: Player, Runs, Balls, 4s, 6s, SR
 - Bowling table: Bowler, Overs, Maidens, Wickets, Economy
- Innings-Specific Headers: Headers with team labels
- Zebra-Striped Rows: Rows with alternating colors for readability

4 POST-MATCH ANALYSIS MODULE (SUMMARY)

The post-match analysis module provides a detailed summary of the match result.

4.1 Result Computation

- Required Run Rate (RRR): Calculates the required run rate using the formula: $(Target - CurrentScore) \times 6 / BallsRemaining$
- Comparative Score Analysis: Compares scores and calculates winning margin
- Wicket Resource Percentage: Analyzes wicket resources used

4.2 Result Calculation

- Score Comparison: Compares first and second innings scores
- Winning Margin Calculation: Calculates winning margin based on runs difference or balls remaining
- Tie Scenarios: Handles tie scenarios and displays result accordingly

4.3 Presentation Layer

- Victory Margin Display: Shows the winning margin
- Reset Button: Includes a reset button to start a new match easily

4.4 Interface Design

- Centered Result Display: Displays result with accent border
- Gradient-Animated Button: Gradient-animated "New Match" button
- Responsive Container: Responsive container with max-width constraints

5 SYSTEM WORKFLOW

- Single JSON object (`matchData`) manages all match states
- Browser storage synchronizes data across pages
- Page routing through location redirection

6 STYLING SYSTEM

6.1 Key Features

- Custom color scheme using CSS variables
- Component-specific layouts:
 - Live page scoring buttons with hover animations
 - Scorecard table responsiveness
- Mobile-first approach with media queries

7 FUTURE IMPROVEMENTS AND CHALLENGES

While the application effectively simulates a basic cricket match, several practical enhancements can improve user experience and robustness.

7.1 Immediate Priorities

- Add support for extras such as wides and no-balls
- Implement an undo feature for the last ball action
- Improve validation of team names and player inputs

7.2 Long-Term Roadmap

- Enable match summary export (e.g., as PDF or JSON)
- Add charts or graphs for score progression
- Allow user-defined match length (overs)
- Support saving and loading multiple matches from storage

7.3 Identified Issues

- No visual indicator of remaining balls or overs
- Manual entry of names can lead to inconsistent spellings
- Second innings setup depends entirely on prompts, which may confuse some users

7.4 Recommendations

- Use dropdowns or autocomplete for batter/bowler names
- Show a running list of all players on each team
- Include a confirmation step before ending innings

8 CHALLENGES FACED

- **Partial Data Persistence:** Incomplete transfer of match statistics from runtime memory to persistent storage
- **Session Collisions:** Potential data corruption during browser context switches
- **Null Reference Exceptions:** Unable to handle edge cases in player substitution scenarios
- **Temporal Data Loss:** Failure to maintain timestamped event logs
- **Data Completeness:** Fails to display players who haven't faced balls
- **Session Sync:** Loses current bowler/batter status during page navigation
- **Storage Issues:** Incorrectly persists data between innings transitions

9 MATERIAL USED

- **Visual Studio Code** – As the primary code editor for development.
- **Google Chrome / Firefox Developer Tools** – For debugging and testing the application.
- **ChatGPT by OpenAI** – Used for debugging assistance, generating JavaScript functions, refining logic and explanations.
- **HTML,CSS,JAVASCRIPT** used.

References

- [1] freeCodeCamp. (2024). Web Storage: localStorage vs sessionStorage.
http://freecodecamp.org/news/web-storage-localstorage-vs-sessionstorage-in-javascript/?utm_source=chatgpt.com
- [2] W3Schools. (2025). JavaScript Tutorial.
<https://www.w3schools.com/js/>
- [3] Cricbuzz. (2024). Live Cricket Scores & News.
<https://www.cricbuzz.com/>