

# Cricket Scorecard Application Documentation

Kolanu Sarvajith

April 29, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>System Overview</b>	<b>2</b>
<b>3</b>	<b>File Structure</b>	<b>2</b>
<b>4</b>	<b>Core Features</b>	<b>3</b>
4.1	Match Tracking . . . . .	3
4.2	Scorecard Display . . . . .	3
4.3	Match Summary . . . . .	3
<b>5</b>	<b>Technical Implementation</b>	<b>3</b>
5.1	Data Storage . . . . .	3
5.2	Key Classes . . . . .	4
5.2.1	Batsman Class . . . . .	4
5.2.2	Bowler Class . . . . .	4
<b>6</b>	<b>User Interface</b>	<b>4</b>
6.1	Live Match Interface . . . . .	4
6.2	Scorecard Interface . . . . .	4
<b>7</b>	<b>Data Flow</b>	<b>5</b>
<b>8</b>	<b>Error Handling</b>	<b>5</b>
<b>9</b>	<b>Performance Considerations</b>	<b>5</b>
<b>10</b>	<b>Usage Guide</b>	<b>5</b>
10.1	Starting a Match . . . . .	5
10.2	During Match . . . . .	6
10.3	Viewing Results . . . . .	6
<b>11</b>	<b>Maintenance</b>	<b>6</b>
11.1	Regular Tasks . . . . .	6
<b>12</b>	<b>Future Enhancements</b>	<b>6</b>

<b>13 Visual Components</b>	<b>6</b>
13.1 Application Screenshots . . . . .	6
<b>14 Conclusion</b>	<b>6</b>
<b>A Code Examples</b>	<b>7</b>
A.1 Score Update Function . . . . .	7
<b>B Glossary</b>	<b>7</b>

# 1 Introduction

This documentation provides a comprehensive guide to the Cricket Scorecard Application, a web-based tool for tracking and displaying cricket match statistics in real-time.

# 2 System Overview

The application consists of several interconnected components:

- Live match tracking interface
- Scorecard display
- Match summary
- Setup and configuration

# 3 File Structure

The application is organized into the following files:

- `live.html` - Main match interface
- `live.css` - Styling for live interface
- `score.js` - Core application logic
- `scorecard.html` - Scorecard display
- `scorecard.css` - Scorecard styling
- `summary.html` - Match summary
- `summary.css` - Summary styling
- `setup.html` - Initial configuration
- `setup.css` - Setup styling

## 4 Core Features

### 4.1 Match Tracking

- Real-time score updates
- Ball-by-ball commentary
- Player statistics tracking
- Innings management

### 4.2 Scorecard Display

- Detailed batting statistics
- Bowling analysis
- Partnership information
- Run rate calculations

### 4.3 Match Summary

- Match result display
- Player performance highlights
- Team statistics
- Match timeline

## 5 Technical Implementation

### 5.1 Data Storage

The application uses localStorage for data persistence:

```
1 // Example of data storage
2 localStorage.setItem('innings', JSON.stringify(innings));
3 localStorage.setItem('team1_batting', JSON.stringify(
4   team1_batting));
5 localStorage.setItem('team2_batting', JSON.stringify(
6   team2_batting));
```

## 5.2 Key Classes

### 5.2.1 Batsman Class

```
1 class Batsman {
2     constructor(name) {
3         this.name = name;
4         this.runs = 0;
5         this.balls = 0;
6         this.fours = 0;
7         this.sixes = 0;
8         this.status = 'not out';
9     }
10    // ... methods ...
11 }
```

### 5.2.2 Bowler Class

```
1 class Bowler {
2     constructor(name) {
3         this.name = name;
4         this.overs = 0;
5         this.maidens = 0;
6         this.runs_conceded = 0;
7         this.wickets = 0;
8         this.balls = 0;
9     }
10    // ... methods ...
11 }
```

## 6 User Interface

### 6.1 Live Match Interface

The live match interface provides:

- Current score display
- Player input forms
- Scoring buttons
- Commentary box

### 6.2 Scorecard Interface

The scorecard displays:

- Team scores
- Individual player statistics

- Bowling figures
- Match progress

## **7 Data Flow**

1. User inputs match details in setup
2. Match data is stored in localStorage
3. Live updates modify stored data
4. Scorecard and summary read from stored data

## **8 Error Handling**

The application includes error handling for:

- Invalid user input
- Data storage failures
- State management issues
- UI update errors

## **9 Performance Considerations**

- Efficient data storage
- Optimized UI updates
- Responsive design
- Browser compatibility

## **10 Usage Guide**

### **10.1 Starting a Match**

1. Open setup.html
2. Enter team names
3. Select toss winner
4. Choose batting/bowling
5. Start match

## **10.2 During Match**

1. Enter player names
2. Use scoring buttons
3. Track commentary
4. Monitor statistics

## **10.3 Viewing Results**

1. Access scorecard
2. View match summary
3. Export data

# **11 Maintenance**

## **11.1 Regular Tasks**

- Data backup
- Code updates
- Bug fixes
- Performance optimization

# **12 Future Enhancements**

- Mobile application
- Cloud storage
- Advanced statistics
- Social sharing

# **13 Visual Components**

The application includes several visual elements to enhance the user experience:

## **13.1 Application Screenshots**

# **14 Conclusion**

The Cricket Scorecard Application provides a comprehensive solution for tracking and displaying cricket match statistics. Its modular design and efficient data management make it suitable for various cricket scoring needs.

## A Code Examples

### A.1 Score Update Function

```
1 function updateScore() {
2     const scoreDisplay = document.getElementById("score_display")
3     ;
4     const freeHitDisplay = document.getElementById("
5         free_hit_display");
6     const inningsStatus = document.getElementById("innings_status
7         ");
8     const targetDisplay = document.getElementById("target_display
9         ");
10
11     let scoreText = innings.wickets === 10 ?
12         `${innings.score} (${innings.overs})` :
13         `${innings.score}/${innings.wickets} (${innings.overs})`;
14
15     if (innings.isFirstInnings) {
16         scoreDisplay.textContent = `${innings.battingTeam.
17             toUpperCase()} ${scoreText} vs ${innings.bowlingTeam.
18                 toUpperCase()} | Extras: ${innings.extras}`;
19         inningsStatus.textContent = "First Innings";
20         targetDisplay.textContent = "";
21     } else {
22         let opponentScoreText = `${innings.target - 1} (${innings
23             .maxBalls / 6} overs)`;
24         scoreDisplay.textContent = `${innings.battingTeam.
25             toUpperCase()} ${scoreText} vs ${innings.bowlingTeam.
26                 toUpperCase()} ${opponentScoreText} | Extras: ${
27                     innings.extras}`;
28         targetDisplay.textContent = `Target: ${innings.target}`;
29     }
30
31     freeHitDisplay.textContent = innings.isFreeHit ? "FREE HIT!"
32         : "";
33     localStorage.setItem('innings', JSON.stringify(innings));
34     updateAdvancedStats();
35 }
```

## B Glossary

**CRR** Current Run Rate

**RRR** Required Run Rate

**SR** Strike Rate


**ER** Economy Rate

### Cricket Toss


RCB


CSK

**Select TOSS Winner:**

RCB 

**Winner chose To:**


BOWL 

**RCB won the toss and chose to bowl first** 

Start Match

Figure 1: Setup Interface - Initial configuration screen



Live Score 

CSK 0/0 (0.0) vs RCB | Extras: 0

First Innings

First Innings Run Rate: 0.00

Rachin

Ruturaj

Hazlewood

Confirm Players

\* | Score: 0(0) 4s: 0 6s: 0 SR: 0.00

| Score: 0(0) 4s: 0 6s: 0 SR: 0.00

Ball by Ball Commentary

Figure 2: Live Match Interface - Real-time scoring interface

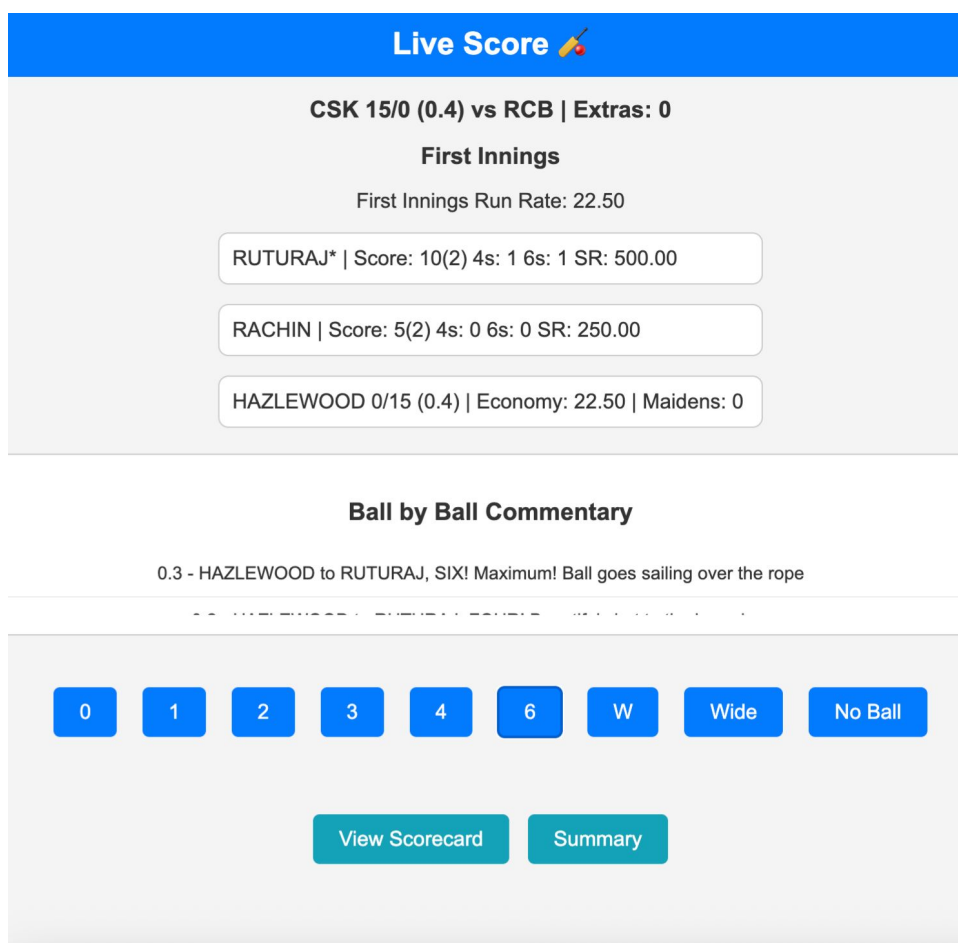


Figure 3: Scorecard Display - Detailed match statistics

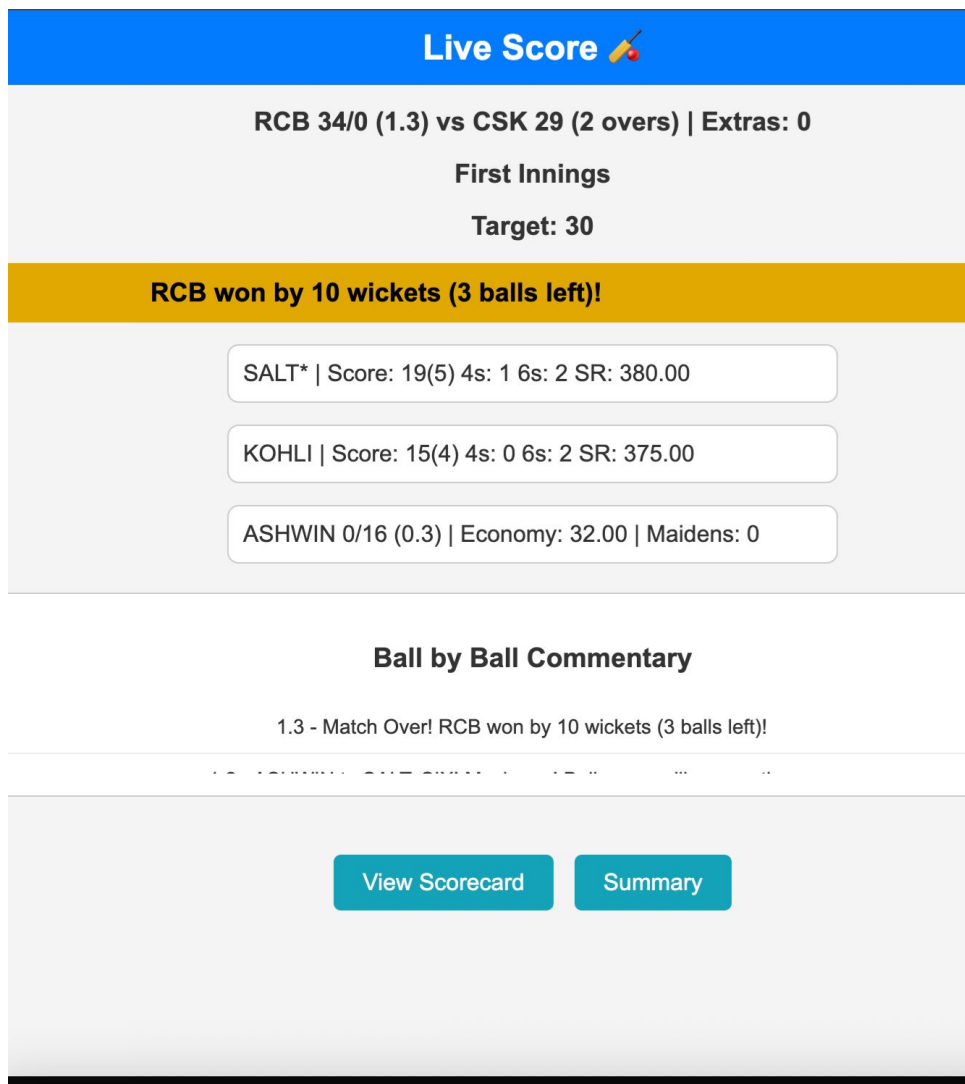


Figure 4: Match Progress - Innings transition view



Figure 5: Match Summary - Final results and statistics

Cricket Match Scorecard						
<a href="#">Back to Live Match</a>						
CSK vs RCB						
1st Innings						
CSK Batting						
Batsman	Runs	Balls	4s	6s	SR	Status
RUTURAJ	10	3	1	1	333.33	Out
RACHIN	12	6	1	0	200.00	Out
Jadeja	0	0	0	0	0.00	Not Out
CSK Bowling						
Bowler	Overs	Maidens	Runs	Wickets	Economy	
HAZLEWOOD	0.4	0	15	0	22.50	
BHUVI	1.0	0	8	2	8.00	
2nd Innings						
CSK Batting						
Batsman	Runs	Balls	4s	6s	SR	Status
SALT*	19	5	1	2	380.00	Not Out
KOHLI	15	4	0	2	375.00	Not Out
RCB Bowling						
Bowler	Overs	Maidens	Runs	Wickets	Economy	
NOOR	1.0	0	18	0	18.00	
ASHWIN	0.3	0	16	0	32.00	

Figure 6: Player Statistics - Individual performance details