DR. K.N.MODI INSTITUTE OF ENGINEERING & TECHNOLOGY

N.H-58, MODINAGAR,GHAZIABAD, UTTAR PRADESH,201204
Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

DEPARTMENT OF COMPUTER SCIENCE & ENNGINEERING

PRACTICAL FILE

SUBJECT NAME: Mini Project or Internship Assessment

(SUBJECT CODE: BCS-554)

ACADEMIC SESSION: 2024-2025

PROJECT TITLE:- Cricket Over Tracker

SUBMITTED BY:
NAME: Utkarsh
ROLL NO.: 2200770100091
NAME: Krishna Agarwal
ROLL NO.: 2200770100052
NAME: Himani Bhatt
ROLL NO.: 2200770130005

UNDER GUIDANCE OF:
MR. NIKHIL TYAGI

INDEX

PROJECT TITLE

CRICKET OVER TRACKER

Project Overview

The "Cricket Over Tracker" is a web-based application designed to track the progress of a cricket match. The application allows users to record each ball bowled, the runs scored, and track the number of wickets taken. It also enables users to add extras such as no balls and wide balls. The application displays the current score, the runs for each over, and the cumulative score for the team. It is designed to simulate a match between two teams, where users can control the scoring actions and the progression of the match.'

Objective

The objective of this project is to create a user-friendly application to track the events of a cricket match, including:

- Recording runs for each ball bowled.
- Keeping track of wickets.
- Accounting for extras such as no balls and wide balls.
- Displaying the cumulative score of the team.
- Switching between innings and teams.
- Declaring the winning team based on the highest score.

Functionalities

The main functionalities of the Cricket Over Tracker are:

- Recording Runs and Events:

- Users can add runs (1, 2, 3, 4, 5, 6) or events like a dot ball, wicket, wide ball, or no ball.
- Tracking Score:

- For each over, the runs are recorded and added to the team's total score.
- The wickets are tracked separately, and the game ends if 10 wickets are taken.
- Switching Teams:

- The game alternates between two teams. After 10 overs or 10 wickets for the first team, the second team begins their innings.
- Displaying Scores:

- The current score (runs and wickets) is displayed at all times.
- The cumulative score for the team is updated with each over.
- After each over, the score for that over is displayed.
- Ending the Game: Once both teams have completed their innings, the application declares the team with the higher total score as the winner.

Technical Details

- Languages Used: HTML, CSS, JavaScript
- HTML:
- Structuring the web page layout, including containers for balls, buttons, and score display.
- CSS:
- Styling the application, including background animations, button styles, and score display formatting.
- JavaScript:
- Core logic for tracking balls, runs, wickets, and calculating scores.
- Event handling to add runs, extras, and switch innings.
- DOM manipulation to dynamically update the score and visual elements (balls, score).

User Interface (UI)

The UI consists of the following sections:

- Ball Display Area: Displays each ball bowled, with either a run or a symbol representing a dot ball, wide, no ball, or wicket.
- Buttons: Buttons for each run type (1, 2, 3, 4, 5, 6), extras (dot, wide, no ball), and special events (wicket).
- Score Displays: Real-time display of the current score, including the number of runs and wickets.
- Cumulative Score: Tracks the total score for each over bowled.
- Current Over: Displays the score for the current over being played.
- New Over Button: After each over, a button appears to start a new over.
- Winning Declaration: Once both teams have played their innings, the winning team is declared based on the total score.

Code Breakdown

- HTML: Provides structure for the application, creating div containers for the score and ball display. It also includes buttons for user interactions (scoring and events).

- CSS:

- Uses a gradient background to give a dynamic look, along with animation to make the user experience engaging.
- Button and ball styles are used for a more interactive and visually appealing interface.
- JavaScript:

- Contains the logic for adding balls, calculating the runs, updating the score, and handling different cricket events (wicket, no ball, etc.).
- Functions such as addBall(), addExtraBall(), and endOver() manage the core functionality of the game.
- Handles the switching of teams, displays the final score, and declares the winner.

Key Functions Explained

- addBall(type):

- Adds a new ball to the ball container.
- Updates the score based on the type of ball (run or wicket).
- Ends the over when 6 balls are bowled.
- addExtraBall(type):

- Adds an extra ball (wide or no ball) to the container and updates the score.
- endOver():

- Ends the current over, updates the cumulative score, and checks if the innings should switch.
- endInnings():

- Switches teams after the first innings is completed.
- declareWinner():

- Declares the winning team once both teams have completed their innings.

CODING

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Cricket Over Tracker @Simplified_Learner</title>
  <link rel="shortcut icon" href="icon.png" type="image/png" />
  <style>
   /* Reset default margins and paddings */
   * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
   }

   /* Stylish Body background */
   body {
    font-family: Arial, sans-serif;
    background: radial-gradient(circle, rgba(255, 255, 255, 0.2), rgba(0, 128, 255, 0.6), rgba(50, 205, 50, 0.6));
    color: rgb(42, 42, 42);
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    animation: background-animation 10s ease-in-out infinite;
   }
```

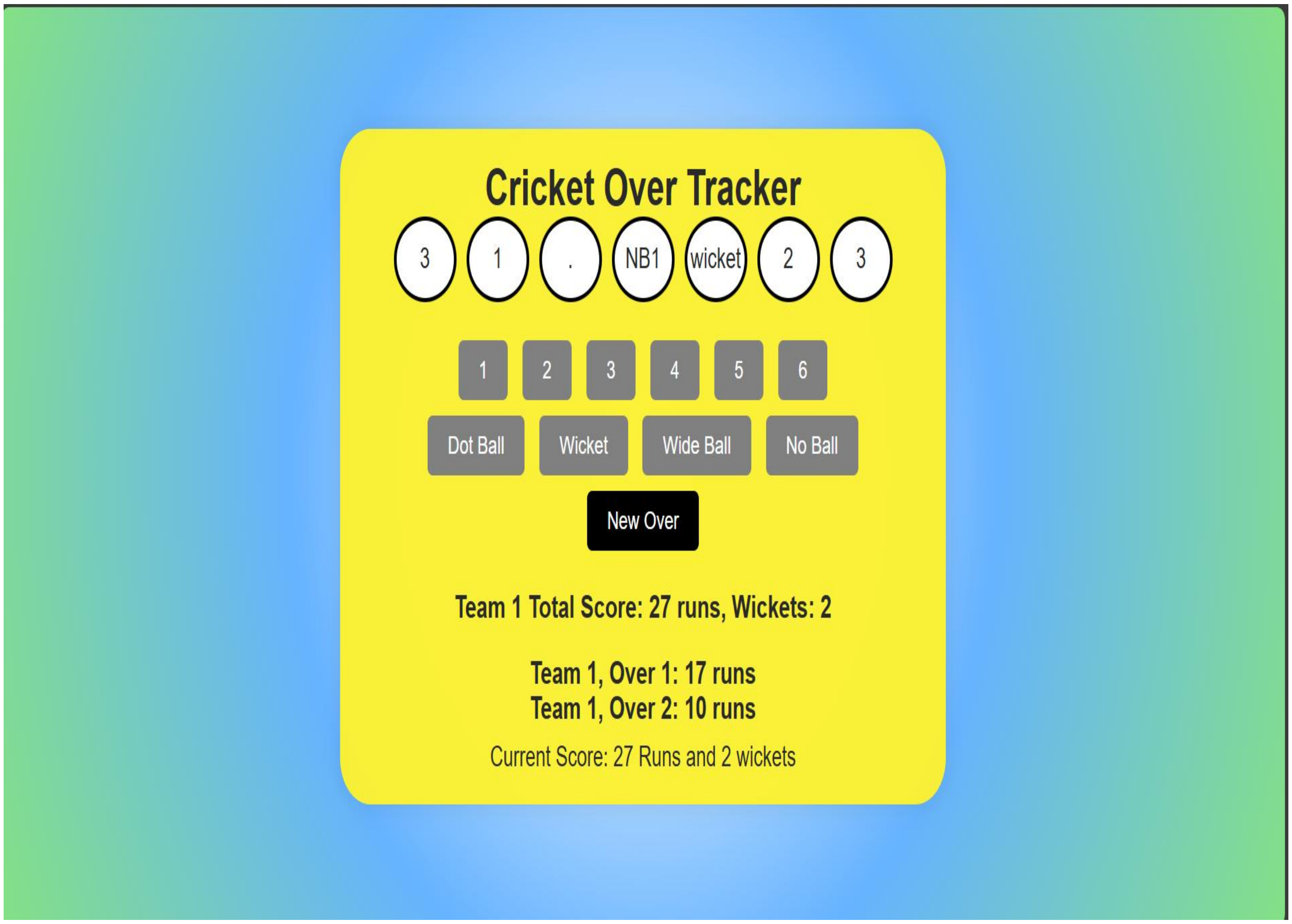
```
/* Animation for background */
@keyframes background-animation {
  0% {
    background: radial-gradient(circle, rgba(255, 255, 255, 0.2), rgba(0, 128, 255, 0.6), rgba(50, 205, 50, 0.6));
  }
  50% {
    background: radial-gradient(circle, rgba(255, 0, 0, 0.3), rgba(255, 165, 0, 0.5), rgba(255, 255, 0, 0.5));
  }
  100% {
    background: radial-gradient(circle, rgba(255, 255, 255, 0.2), rgba(0, 128, 255, 0.6), rgba(50, 205, 50, 0.6));
  }
}

/* Container for the cricket over tracker */
.over-container {
  text-align: center;
  padding: 20px;
  border-radius: 30px;
  background-color: #fcf233f9;
  box-shadow: 0 0 15px rgba(20, 19, 19, 0.1);
  width: 80%;
  max-width: 600px;
  transition: transform 0.3s ease;
}

/* Stylish Buttons */
.buttons-container button {
  margin: 5px;
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
  background-color: #000000;
  color: #fff;
  border: none;
  border-radius: 5px;
  transition: background-color 0.3s, transform 0.2s;
}

.buttons-container button:hover {
  background-color: #0056b3;
  transform: scale(1.1);
}

.buttons-container button:active {
  transform: scale(0.9);
}

/* Styling for the balls container */
.balls-container {
  margin-bottom: 20px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}

/* Individual ball styles */
.ball {
  width: 62px;
  height: 55px;
  border-radius: 50%;
  border: 3px solid #000;
  display: inline-flex;
  justify-content: center;
  align-items: center;
  margin: 0 5px;
  font-size: 18px;
  background-color: #fff;
  transition: transform 0.3s ease;
}

/* Animations for Wicket */
.wicket {
  animation: wicket-animation 0.5s ease;
}

@keyframes wicket-animation {
  0% {
    transform: rotate(0deg);
  }
  50% {
    transform: rotate(20deg);
  }
  100% {
    transform: rotate(0deg);
  }
}
```

```
/* Animations for No Ball */
.no-ball {
  animation: no-ball-animation 0.8s ease;
}

@keyframes no-ball-animation {
  0% {
    background-color: #ff0;
  }
  50% {
    background-color: #ff6347;
  }
  100% {
    background-color: #fff;
  }
}

/* Styling for score display */
.score-display,
.cumulative-score {
  margin-top: 20px;
  font-size: 20px;
  font-weight: bold;
}

/* Styling for the current score display */
#current-score {
  margin-top: 10px;
  font-size: 18px;
```

```
    }

    /* Disabled button styles */
    button:disabled {
      background-color: gray;
      cursor: not-allowed;
    }

    button:disabled:hover {
      background-color: gray;
    }
  </style>
</head>
<body>
  <div class="over-container">
    <h1>Cricket Over Tracker</h1>
    <div id="balls-container" class="balls-container"></div>
    <div class="buttons-container">
      <button id="button-1">1</button>
      <button id="button-2">2</button>
      <button id="button-3">3</button>
      <button id="button-4">4</button>
      <button id="button-5">5</button>
      <button id="button-6">6</button><br />
      <button id="button-dot">Dot Ball</button>
      <button id="button-wicket">Wicket</button>
      <button id="button-wide">Wide Ball</button>
      <button id="button-no">No Ball</button>
    </div>
    <div id="score-display" class="score-display"></div>
    <div id="cumulative-score" class="cumulative-score"></div>
    <div id="current-score"></div>
  </div>

  <script>
    let ballCount = 0;
    let currentOverScore = 0;
    let currentScore = 0;
    let currentOver = 1;
    let totalScore = 0;
    let wickets = 0;
    let overScores = [];
    let currentTeam = 1;
    let teamScores = [[], []];

    function addBall(type) {
      if (wickets < 10 && ballCount < 6) {
        const ballsContainer = document.getElementById("balls-container");
        const ball = document.createElement("div");
        ball.classList.add("ball");
        ball.innerText = type === "dot" ? "." : type;
        ballsContainer.appendChild(ball);
        ballCount++;

        if (type !== "dot") {
          if (type !== "wicket") {
            const runs = parseInt(type);
            currentOverScore += runs;
            currentScore += runs;
          } else {
            wickets++;
            ball.classList.add("wicket");
            if (wickets === 10) {
              endOver();
              return;
            }
          }
        }

        if (ballCount === 6) {
          endOver();
        }

        updateTeamScore();
        displayCurrentScore();
      }
    }

    function addExtraBall(type) {
      if (wickets < 10) {
        const ballsContainer = document.getElementById("balls-container");
        const ball = document.createElement("div");
        ball.classList.add("ball");
        if (type === "wide" || type === "no") {
          ball.innerText = type === "wide" ? "WD1" : "NB1";
          currentOverScore += 1;
          currentScore += 1;
          ball.classList.add("no-ball");
        }
        ballsContainer.appendChild(ball);


        updateTeamScore();
        displayCurrentScore();
      }
    }

    function displayCurrentScore() {
      const currentScoreElement = document.getElementById("current-score");
      currentScoreElement.innerText = Current Score: ${currentScore} Runs and ${wickets} wickets;
    }

    function endOver() {
      overScores.push(currentOverScore);
      teamScores[currentTeam - 1].push(currentOverScore);
      totalScore += currentOverScore;

      const cumulativeScoreDisplay = document.getElementById("cumulative-score");
      const overScoreElement = document.createElement("div");
      overScoreElement.innerText = Team ${currentTeam}, Over ${currentOver}: ${currentOverScore} runs;
      cumulativeScoreDisplay.appendChild(overScoreElement);

      currentOver++;
      ballCount = 0;
      currentOverScore = 0;

      const scoreDisplay = document.getElementById("score-display");
      scoreDisplay.innerText = Team ${currentTeam} Total Score: ${totalScore} runs, Wickets: ${wickets};

      if (currentOver > 10 || wickets === 10) {
        endInnings();
      } else {
        disableButtons();
        showNewOverButton();
      }

      displayCurrentScore();
    }

    function endInnings() {
      if (currentTeam === 1) {
        currentTeam = 2;
        resetForNextTeam();
      } else {
        declareWinner();
      }
    }

    function resetForNextTeam() {
      overScores = [];
      totalScore = 0;
      wickets = 0;
      currentOver = 1;
      ballCount = 0;
      currentOverScore = 0;
      currentScore = 0;
      const ballsContainer = document.getElementById("balls-container");
      ballsContainer.innerHTML = "";

      const cumulativeScoreDisplay = document.getElementById("cumulative-score");
      cumulativeScoreDisplay.innerHTML = "";

      const scoreDisplay = document.getElementById("score-display");
      scoreDisplay.innerText = Team ${currentTeam} starts their innings;

      const buttons = document.querySelectorAll(".buttons-container button");
      buttons.forEach((button) => {
        button.disabled = false;
      });
    }

    function declareWinner() {
      const team1Score = teamScores[0].reduce((a, b) => a + b, 0);
      const team2Score = teamScores[1].reduce((a, b) => a + b, 0);
      const scoreDisplay = document.getElementById("score-display");

      if (team1Score > team2Score) {
        scoreDisplay.innerText = Match complete! Team 1 wins with ${team1Score} runs!;
      } else if (team2Score > team1Score) {
        scoreDisplay.innerText = Match complete! Team 2 wins with ${team2Score} runs!;
      } else {
        scoreDisplay.innerText = "Match complete! It's a tie!";
      }

      disableButtons();
```

```
}

function disableButtons() {
  const buttons = document.querySelectorAll(".buttons-container button");
  buttons.forEach((button) => {
    button.disabled = true;
  });
}

function showNewOverButton() {
  const newOverButton = document.createElement("button");
  newOverButton.innerText = "New Over";
  newOverButton.onclick = resetOver;
  document.querySelector(".buttons-container").appendChild(newOverButton);
}

function resetOver() {
  const ballsContainer = document.getElementById("balls-container");
  ballsContainer.innerHTML = "";
  const buttons = document.querySelectorAll(".buttons-container button");
  buttons.forEach((button) => {
    button.disabled = false;
  });
  document
    .querySelector(".buttons-container")
    .removeChild(
      document.querySelector(".buttons-container button:last-child")
    );
  const scoreDisplay = document.getElementById("score-display");
```

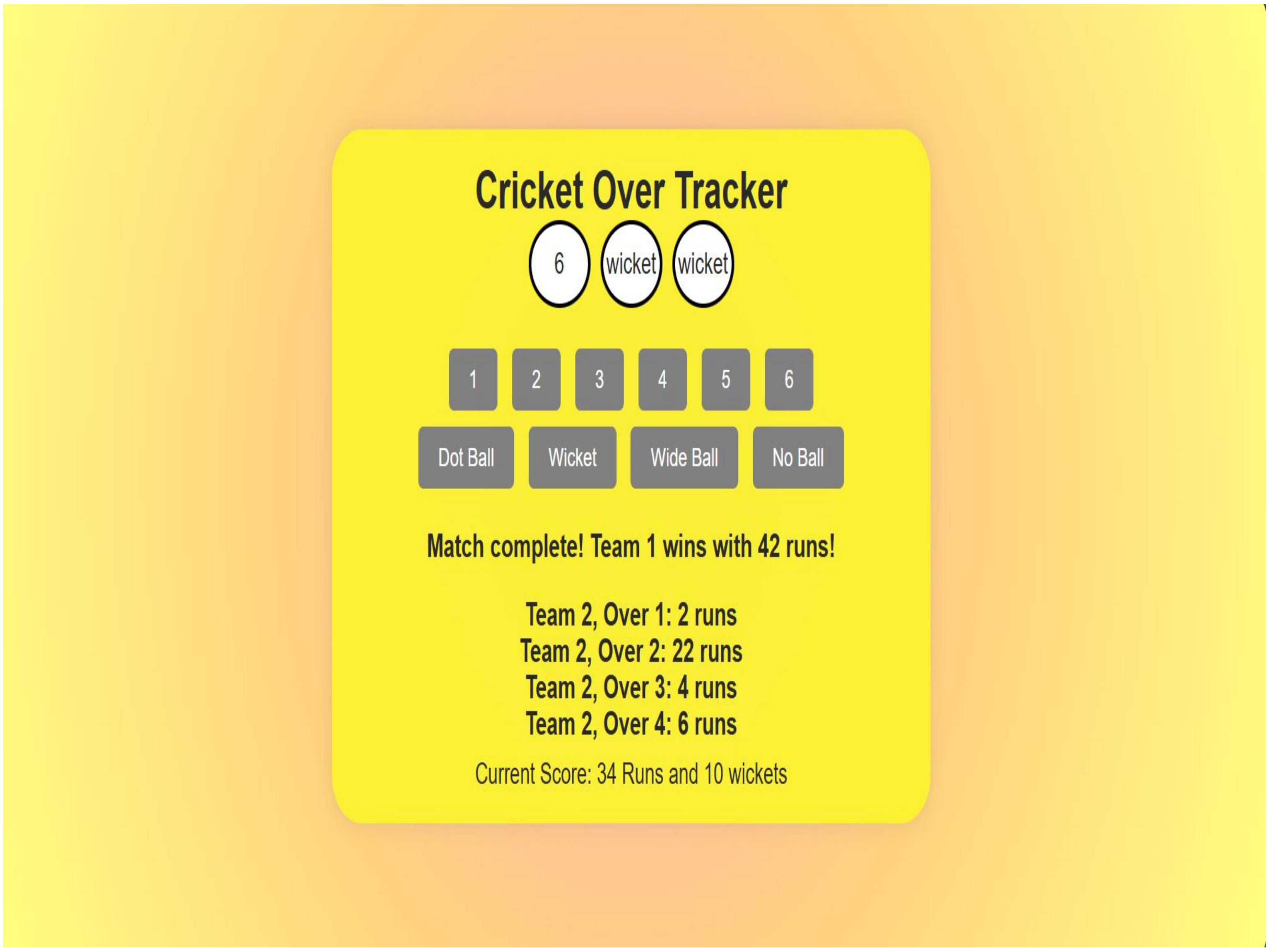
```
        scoreDisplay.innerText = "";
      }

      document.addEventListener("DOMContentLoaded", () => {
        document.getElementById("button-1").onclick = () => addBall("1");
        document.getElementById("button-2").onclick = () => addBall("2");
        document.getElementById("button-3").onclick = () => addBall("3");
        document.getElementById("button-4").onclick = () => addBall("4");
        document.getElementById("button-5").onclick = () => addBall("5");
        document.getElementById("button-6").onclick = () => addBall("6");
        document.getElementById("button-dot").onclick = () => addBall("dot");
        document.getElementById("button-wicket").onclick = () => addBall("wicket");
        document.getElementById("button-wide").onclick = () => addExtraBall("wide");
        document.getElementById("button-no").onclick = () => addExtraBall("no");
      });
    </script>
  </body>
</html>
```



Team 1 Total Score: 27 runs, Wickets: 2

Team 1, Over 1: 17 runs
Team 1, Over 2: 10 runs

Current Score: 27 Runs and 2 wickets



Match complete! Team 1 wins with 42 runs!

Team 2, Over 1: 2 runs
Team 2, Over 2: 22 runs
Team 2, Over 3: 4 runs
Team 2, Over 4: 6 runs

Current Score: 34 Runs and 10 wickets

Test Cases Used

1. **Basic Runs Tracking**:
    - Test the scoring system when different run buttons (1, 2, 3, 4, 5, 6) are clicked.
    - Expected: Each run should be correctly added to the total score, and the ball count should be updated.
2. **Dot Ball (No Runs)**:
    - Test the behavior when the dot ball button is clicked.
    - Expected: The ball should be recorded as a dot (0 runs), and the total score should remain unchanged.
3. **Wicket Handling**:
    - Test the system's response to the "Wicket" button.
    - Expected: The ball should be recorded as a wicket, and the total wickets count should increase by 1. The game should end after 10 wickets.
4. **Extra Balls (Wide and No Ball)**:
    - Test adding extra runs via wide or no balls.
    - Expected: The score should increase by 1 for each extra ball (Wide/No Ball), but these should not count as regular balls in the over.
5. **End of Over (6 Balls)**:
    - Test the automatic end of an over after 6 balls.
    - Expected: After 6 balls, the over should end, and the score for that over should be displayed.
6. **End of Innings**:
    - Test the system's handling when either 10 wickets are lost or the maximum overs (10 overs) are reached.
    - Expected: The game should switch to the next team's innings or declare the end of the match.
7. **Switching Teams**:
    - Test the transition from team 1 to team 2 after the first innings ends.
    - Expected: The system should reset, and team 2 should begin their innings with fresh scores and ball tracking.
8. **Button Disablement**:
    - Test that buttons are disabled after an over ends or after a team’s innings is completed.
    - Expected: Once an over or innings ends, buttons should be disabled until the next over starts.
9. **Display of Winner**:
    - Test the final display when both teams complete their innings.
    - Expected: The team with the higher total score should be declared the winner, and the result should be shown.

Future Scope

The Cricket Over Tracker project can be further improved and extended with additional features and enhancements. Below are a few ideas for future development:

1. Multi-Match Support

Currently, the application supports only a single match. The future scope includes enabling the ability to track multiple matches simultaneously. Users can choose from a list of ongoing or completed matches.

2. Player-specific Details

The application could allow users to input the names of the players on both teams. Each ball can then be assigned to a specific player, and individual player statistics such as runs scored and wickets taken can be tracked.

3. Advanced Statistics

Additional statistics such as batting strike rate, economy rate (for bowlers), highest scorer, and best bowler could be introduced to give a more detailed analysis of the game.

4. Interactive Interface

Improve the user interface by integrating features like a graphical representation of the score (bar chart, pie chart) for better visualization of team performance.

5. Integration with Real-Time Data

Integrate with APIs for live cricket data and scores, allowing real-time updates of the match score.

6. Team Management

Allow the user to create teams, store player data, and manage team formation. Players' statistics can then be updated with every match.

Conclusion

- The Cricket Over Tracker is an interactive, simple, and fun web application that can track the progress of a cricket match. It can be easily extended to include more features like player details and match history. The project serves as an excellent demonstration of how JavaScript can be used to build dynamic, real-time applications in a web browser.