### **Project Report: Cricket Match Simulation Program**

## Abdul Rehman 51883

Shumail Abdul Rehman 47891

Ali Raza Khan 55592

Muhmmad Umer Khalid



### **1. Introduction**

### A project report serves as a comprehensive document that provides an in-depth overview of the project, detailing its objectives, design, methodology, development process, challenges, results, and conclusions. It is an essential tool for documenting the entire project lifecycle and ensuring transparency, clarity, and traceability of the development process.

### This report discusses the development of a **Cricket Match Simulation Program**, which simulates a cricket match between two teams with elements such as toss, batting, bowling, and scoring, and provides a detailed breakdown of individual player performances, match results, and statistical analysis.

### **2. Project Objectives**

### The primary objective of this project was to develop a program that simulates a cricket match, allowing users to simulate match events such as batting, bowling, scoring, and the outcome of the game. The simulation aims to provide an interactive experience, where users can:

### **Input Team and Player Details:** Users can input the names of two teams and their respective players.

### **Simulate a Toss:** The program randomly determines which team will bat and which team will bowl.

### **Simulate Overs and Deliveries:** Each team plays a series of overs, where runs are scored randomly, and wickets can fall.

### **Track and Display Scorecards:** A detailed scorecard of the match is generated, including runs scored, wickets fallen, player statistics, and performance analysis.

### **Generate Match Results:** After the completion of both innings, the program compares scores to determine the winner.

### **. System Design and Architecture**

### The Cricket Match Simulation program follows a modular design approach, breaking down the simulation process into smaller, manageable functions. These functions interact with each other to simulate various aspects of a cricket match. The architecture of the system can be broken down as follows:

### **User Input & Setup:** This section allows users to input team names, player names, and other match-related details such as the number of overs to be played.

### **Match Simulation:** This is the core component of the program. It simulates the toss, determines which team will bat or bowl, and then generates match events (runs, wickets, etc.) for each over. The simulation is based on random events such as scoring a run, hitting a boundary, or losing a wicket.

### **Score Calculation:** The program keeps track of the score in real-time, adjusting the total score for each team based on runs scored and wickets lost. It also calculates the run rate and updates the scorecard accordingly.

### **Match Comparison:** After both innings are completed, the program compares the scores of both teams and declares the winner.

### The overall architecture of the system is designed to be user-friendly, with clear prompts and outputs for the user to interact with during the simulation.

### **4. Technologies Used**

### The Cricket Match Simulation project was developed using the **C++ programming language**. C++ was chosen for this project due to its ability to handle complex logic and random number generation required for simulating match events. It is also well-suited for object-oriented programming, which allows for modular development and reuse of code.

### **Key Libraries Used:**

### **iostream:** To handle input and output operations (such as displaying messages and taking user input).

### **cstdlib:** For random number generation, which is used to simulate runs, wickets, and other events during the match.

### **ctime:** For generating random numbers based on the current system time, ensuring that each match simulation is different.

**5.2 Batting and Bowling**

Once the toss is determined, the program moves on to simulate the batting and bowling sequences. The batting team faces a set number of overs, and for each delivery, a random result is generated. Possible outcomes for each ball include:

* **Run Scoring:** Runs are generated randomly, with outcomes ranging from a miss (0) to hitting a six (6).
* **Wicket Falling:** A random result may indicate a wicket, which is recorded and counts towards the total number of wickets lost.
* **Boundaries and Sixes:** If the random number generated is 4 or 6, the batsman scores a boundary or a six.

The bowling team’s performance is tracked through the number of wickets taken, and the bowling statistics are recorded on the scorecard.

**5.3 Overs and Scoring**

The match is played over a user-defined number of overs. The program generates ball-by-ball outcomes for each over, simulating both runs and wickets. The total runs and wickets are calculated and displayed in real-time during the simulation. Additionally, the program calculates the run rate for each team, which is used to evaluate the performance of the teams.

**5.4 Second Innings**

After the first innings concludes, the second innings begins, following the same process of batting, bowling, and scoring. The program compares the total score of both teams to determine the winner.

**6. Key Features**

The Cricket Match Simulation program offers several features that enhance the user experience and provide valuable insights into the simulated match:

* **Dynamic Team and Player Setup:** Users can input custom teams and players, allowing for flexibility in the simulation.
* **Real-time Score Updates:** The program updates the score continuously, providing a live view of the match as it progresses.
* **Detailed Scorecard and Match Analysis:** The scorecard is generated at the end of each innings, providing detailed statistics on each player’s performance.
* **Match Result and Comparison:** After both innings, the program compares the scores and announces the winner, ensuring a clear and concise conclusion to the match.

**7. Challenges and Solutions**

During the development of the Cricket Match Simulation program, several challenges were encountered and addressed:

* **Randomness in Match Events:** Ensuring the randomness of match events, such as runs and wickets, was crucial for making the simulation realistic. This was achieved using the rand() function, seeded with the system time to ensure varied outcomes in every simulation.
* **Handling Input and User Interaction:** The program requires significant user input (such as team names, player names, and the number of overs). Proper error handling and user prompts were incorporated to ensure smooth interaction.
* **Score Calculation:** Keeping track of runs and wickets, updating the scorecard, and calculating the run rate in real-time required careful attention to the flow of the program. This was addressed by maintaining variables to track the scores, wickets, and run rates at each step.

**8. Results and Discussion**

The Cricket Match Simulation program successfully simulates a complete cricket match, with users able to interactively play through a match from toss to final results. The following results were observed:

1. **Realistic Match Outcomes:** The program generates match results based on random events, mimicking the unpredictable nature of real-world cricket matches.
2. **Accurate Score Calculation:** The scorecard is updated continuously, and at the end of the game, a final score is displayed, along with the winner.
3. **Detailed Player Performance:** The program tracks individual player performances, providing insights into runs scored, balls faced, and wickets taken.