

CHAPTER 1

INTRODUCTION

The “**Cricket Score Display System**” project in C language is a program that is designed to display the score of a cricket match on a screen. It is an interesting and challenging project that requires knowledge of C programming language and basic programming concepts.

The program is designed to record and display the details of a cricket match. It takes the details of the batsmen and bowlers, calculates their scores, and then as per the user's input. The program allows the user to display the details of a specific batsman or bowler or to display a summary of the match.

The program uses two structures, one for the batsman and the other for the bowler, to store their respective details. The batsman structure stores the name of the player, the number of runs scored, the number of balls played, the number of fours and sixes hit, the number of ones, twos and threes scored, the maximum number of sixes, fours, and runs, and the strike rate. Similarly, the bowler structure stores the name of the player, the number of runs given, the number of wickets taken, the number of over bowled, the maximum number of wickets taken, and the economy rate. The program takes the user input for the number of batsmen and bowlers and then loops through each of them, taking their details as input.

The program then calculates the score of each batsman and the economy rate of each bowler. The program uses a switch case to allow the user to display the details of a specific player or a match summary. If the user chooses to display the details of a specific batsman, the program takes the player's number as input and displays their name, the number of runs scored, the number of balls played, the number of fours and sixes hit, and the strike rate.

1.1 Background

This is a C program that records details of batsmen and bowlers, and provides options to display specific player or match summary. The code is relatively simple and straightforward, but there are a few improvements that can be made.

1. Naming conventions: The variable names used in the program can be made more meaningful and consistent. For example, pl1, pl2, pl3, and pl4 are not very descriptive names for the structs. Instead, we can use names like "batsmen", "bowlers", "current_batsman", and "current_bowler". Similarly, the variable "plno" can be renamed to "player_number" for better readability.
2. Error handling: The program assumes that the user enters valid inputs at all times. However, it's always better to include error handling to prevent the program from crashing due to unexpected inputs. For example, the program can check if the user enters a valid player number before displaying the player details.
3. Duplicated code: The code to calculate the runs scored by a batsman and the economy rate of a bowler is duplicated in multiple places. It's better to write separate functions for these calculations and reuse them where necessary.
4. Summary calculation: The match summary currently calculates the runs and strike rate of each player every time it's displayed. This can be inefficient if the summary is displayed frequently. Instead, we can calculate these values once when the player details are entered and store them in the struct.

1.2 Objective

This code is for a program that records the details of batsmen and bowlers and shows their details upon request. The program allows the user to enter the details of the players, such as name, runs, balls faced, wickets taken, etc. The user can then view the details of a specific player or the summary of the entire match.

The program starts by including the necessary header files, '**stdio.h**' and '**stdlib.h**'. The two '**structs**' used in the program are '**struct batsman**' and '**struct bowler**'. These '**structs**' hold the details of the batsmen and bowlers, respectively.

The program then asks the user to enter the details of the batsmen and bowlers, such as their name, runs, balls faced, wickets taken, etc. The details are stored in arrays '**pl1**' and '**pl2**', respectively.

The program then enters a loop where it asks the user to choose from a menu of options. The options are:

1. View batsman detail
2. View bowler detail
3. View match summary
4. View record
5. Exit

The user can choose any option, and the program will perform the corresponding action. For example, if the user chooses option 1, the program will ask for the batsman number, and then display the details of that batsman.

If the user chooses option 2, the program will ask for the bowler number and then display the details of that bowler.

If the user chooses option 3, the program displays a summary of the match, showing the runs scored, balls faced, fours, sixes, and strike rate of each batsman.

If the user chooses option 4, the program displays the records of the players with maximum runs, maximum wickets, maximum fours, and maximum sixes.

If the user chooses option 5, the program exits.

Overall, the program provides a simple way to record and view the details of the players in a cricket match. However, the program has a few issues,

such as:

- There is no error handling for invalid user inputs.
- The calculation of the economy rate for the bowler detail is incorrect. The program calculates it as `pl2[plno].econ = pl2[plno].runsgv / pl2[plno].overs`, but it should be `pl2[plno].econ = pl2[plno].runsgv / (pl2[plno].overs/6.0)`.
- The program does not store or display the details of the team, such as the name of the team, the score, or the number of wickets lost.

1.3 Scope

The scope of this program is to capture the details of a cricket match, including the performance of batsmen and bowlers, and display various statistics related to the match. The program allows the user to enter the details of the batsmen and bowlers, and then provides options to view the details of a particular batsman or bowler, the summary of the match, the record of the match, and exit the program.

The program also calculates and displays various statistics such as the strike rate of the batsmen, economy rate of the bowlers, and the maximum number of fours, sixes and runs scored by a batsman in the match. It also displays the maximum number of wickets taken by a bowler in the match.

The program uses structures to store the details of the batsmen and bowlers, and loops to iterate through the arrays of structures to calculate and display the statistics. Overall, the program is designed to provide a simple way to capture and display the details of a cricket match.

1.4 Applicability

The program is applicable for anyone who wants to record and analyze the performance of cricket players. It can be used by coaches, team managers, and cricket enthusiasts to keep track of player performances, identify areas of improvement, and analyze match data. The program is simple and easy to use, making it accessible to a wide range of users.

- This program is a cricket scorekeeper which allows the user to enter details of batsmen and bowlers and retrieve individual player stats or a match summary.
- The program can be used by cricket enthusiasts who want to keep track of the scores and analyze the performance of individual players in a cricket match.

For example, coaches and team managers can use this program to keep track of the performance of their players, identify the strengths and weaknesses of individual players, and make data-driven decisions about player selection and strategy.

Similarly, cricket fans and sports journalists can use this program to analyze the performance of individual players and teams, compare different players' statistics, and generate insights about the game.

The program can also be used as a learning tool for students who are interested in programming and want to practice their coding skills by creating their own scorekeeper programs.

1.5 Survey of Technology

This project seems to be a command-line application that records and displays information about a cricket match, specifically the performance of the batsmen and bowlers.

The programming language used in this project is C, which is a procedural language known for its efficiency and close-to-the-hardware functionality. The code uses standard libraries such as `stdio.h` and `stdlib.h` for input/output and memory allocation, respectively.

1.5.1 C Programming language:

C is a general-purpose programming language that was developed in the early 1970s by Dennis Ritchie at Bell Labs. It is a high-level language that is widely used in software development for creating systems software, application software, and embedded systems.

C is considered to be a middle-level language as it combines the features of both high-level and low-level languages. It is often used for system programming, such as operating systems, device drivers, and firmware. C is also widely used in application programming for creating desktop applications, games, and scientific applications.

C is a compiled language, which means that source code is compiled into machine code that can be run directly on a computer. This makes C programs fast and efficient, and allows them to run on a wide range of platforms without modification. C is also highly portable, meaning that code written in C can be easily adapted to run on different hardware and software platforms.

1.6 Features:

This is a program written in the C programming language that records and displays cricket players' statistics. The program defines two structures: **'batsman'** and **'bowler'**, each containing information about players' names, scores, and other details.

The program first prompts the user to input details of batsmen and bowlers, such as the number of players and their individual statistics like runs scored, balls played, wickets taken, etc.

Then the user can select from several options to display the details of individual players or match summaries. For example, they can select option 1 to see the details of a particular batsman or option 2 to see the details of a particular bowler. The program calculates the batsman's strike rate and bowler's economy rate using the given inputs.

The match summary option displays a summary of all the batsmen and their corresponding statistics, including total runs scored, balls played, fours, sixes, and strike rates. The program also calculates and displays the total number of overs, runs, wickets taken, and economy rate of all the bowlers.

This program provides a useful tool for cricket enthusiasts to keep track of player statistics and match summaries. The C programming language provides a powerful and efficient way to implement such programs, making it a popular choice for developing applications in various domains.

1.7 Software Project Management

Software project management is the process of planning, organizing, directing, and controlling the resources (both human and technical) involved in a software project, from initiation to completion. It involves defining and tracking project requirements, schedules, budgets, quality standards, risk management, and communication with stakeholders.

Effective software project management ensures that software projects are completed on time, within budget, and to the desired quality levels. It also involves identifying and addressing project risks and issues, and ensuring that the project team works collaboratively and efficiently to achieve project goals.

The image below shows triple constraints for software projects.



Fig. 1 (a) *Software Project Management*

1.7.1 Resource Histogram

A resource histogram is a type of bar chart used in project management to visualize the resource utilization over time. It displays the number of resources, usually employees or contractors, working on a project at any given time.

The x-axis of the histogram represents the project timeline, divided into time periods, such as days, weeks, or months. The y-axis represents the number of resources allocated to the project. Each bar in the histogram represents the number of resources assigned to the project during a particular time period.

By using a resource histogram, project managers can quickly visualize the resource requirements of a project and identify any potential resource over- or under-utilization issues. This information can be used to adjust resource allocation as necessary to ensure that the project is completed on time and within budget.

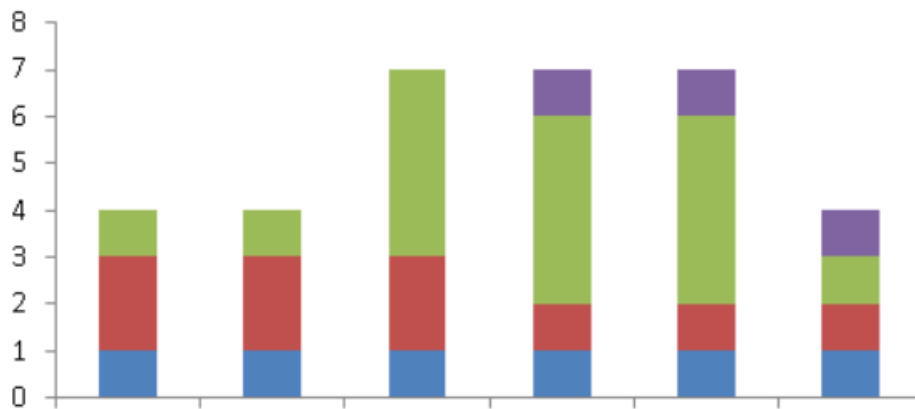


Fig. 1 (b) *Resource Histogram*

1.7.2 PERT Chart

A PERT chart, also known as a Program Evaluation and Review Technique chart, is a graphical tool used in project management to plan, schedule, and coordinate tasks within a project. It is a network diagram that visually represents the activities and their interdependencies within a project.

The PERT chart helps project managers to identify the sequence of activities required to complete a project, estimate the time required for each activity, and identify the critical path that determines the shortest possible duration for the project. It is a useful tool for scheduling complex projects and tracking progress against the schedule.

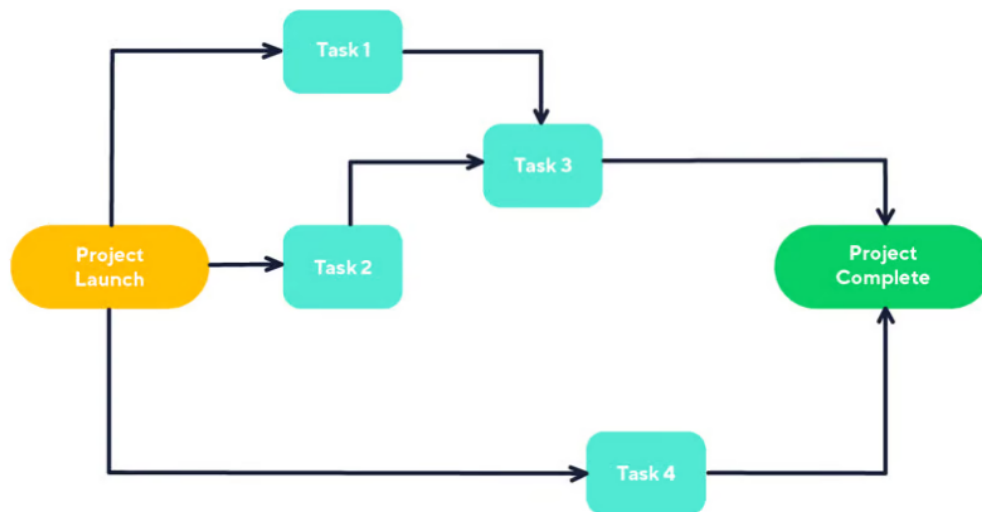


Fig.1 (c) *PERT Chart*

1.7.3 Gantt Chart

A Gantt chart is a bar chart used in project management to illustrate a project schedule. It displays the start and end dates of project tasks, as well as their durations and dependencies.

The Gantt chart is named after its inventor, Henry Gantt, and is a commonly used project management tool. The chart consists of a series of horizontal bars, each representing a project task. The bars are positioned along a timeline that represents the project duration, and the length of each bar represents the duration of the corresponding task.

The Gantt chart provides a visual representation of the project schedule and allows project managers to easily track progress against the schedule. It is a useful tool for communicating project status to stakeholders and for identifying potential delays or resource constraints.

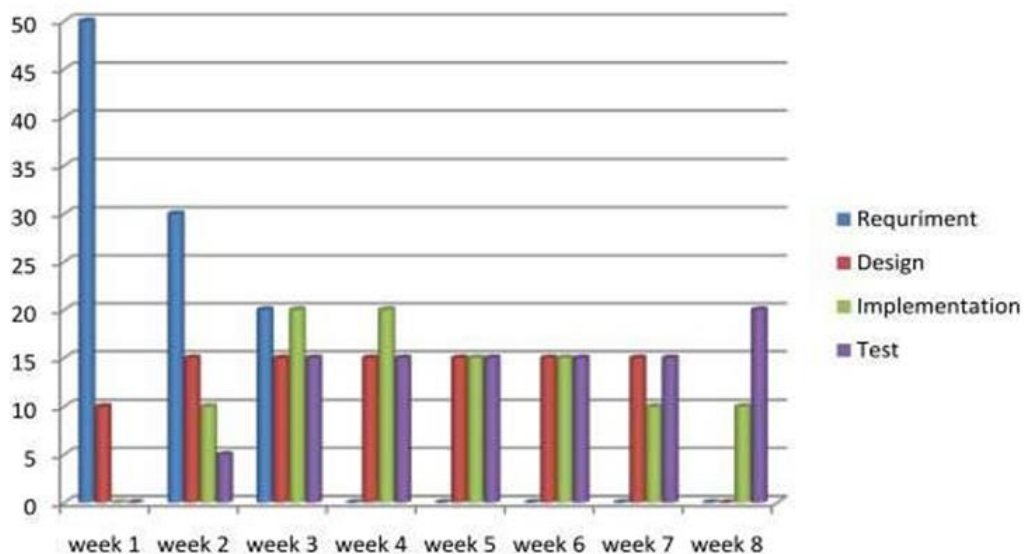


Fig. 1 (d) *Gantt Chart*

CHAPTER 2

REQUIREMENT ANALYSIS

This program is designed to record and display the details of a cricket match, specifically the performance of the batsmen and bowlers. The program uses two data structures: struct batsman and struct bowler to store the details of the players. The struct batsman contains information such as the player's name, the number of runs scored, the number of balls faced, the number of fours, the number of sixes, and the strike rate. The struct bowler contains information such as the player's name, the number of overs bowled, the number of runs given, the number of wickets taken, and the economy rate.

The user is first prompted to input the number of batsmen and bowlers, and then to input the details of each player. Once all the details have been entered, the user is given a menu of options to choose.

Batsman detail: Allows the user to view the details of a specific batsman, including their runs, balls faced, fours, sixes, and strike rate.

Bowler detail: Allows the user to view the details of a specific bowler, including their number of overs bowled, runs given, wickets taken, and economy rate.

Match summary: Provides a summary of the match, including the runs scored by each batsman, the number of overs bowled by each bowler, the runs given by each bowler, and the wickets taken by each bowler.

Record: Allows the user to update the record of a particular player.

2.1 Hardware Requirements:

- Processor : Intel i3 or Above
- RAM : 1 GB
- Cache Memory : 512 KB
- Hard Disk : 256 GB
- Monitor : 14 Inch
- Keyboard : Multimedia
- Mouse : Optical Mouse

2.2 Software Requirements:

- Operating System : Windows 10 or Above
- C Compiler : V S Code & CodeBlocks

2.3 Feasibility Study

Referencing to this information, the analysts do a detailed study about whether the desired When the client approaches the organization for getting the desired product developed, it comes up with a rough idea about what all functions the software must perform and which all features are expected from the software. Referencing to this information, the analysts do a detailed study about whether the desired system and its functionality are feasible to develop.

System and its functionality are feasible to develop. This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints, and as per values and objectives of the organization.

Main feasibility study are :-

1. Analysis of alternative candidate systems after studying the various systems we derived various alternatives through which we develop our project and evaluated the alternative.
2. Evaluation of existing system and procedures. Our group used various virtual assistants to gather information about the software system.

2.3.1 Technical Feasibility

This project is completely build in C so and the VS Code compiler is used that is open source thus making it technically feasible.

2.3.2 Operational Feasibility

Automation makes our life easy. The proposed system is highly user friendly and is much easily able to interact with the system. Therefore the users will readily accept the system as data entry and making queries can be easily done.

2.3.3 Economical Feasibility

Approximated Cost of the Software is nothing more than a system that the user already have to run the program and from the developer side the same is the requirement without any additional paid software for the develop so it's economical feasible.

2.4 System Design

A cricket score display system in a C program could be designed using a simple console-based user interface. The system would require data input for the batsman and bowler details, as well as the ability to display the scorecard, match summary, and records.

The following steps could be taken to design the system:

1. Identify the data to be collected:
 - For each batsman, collect the name, number of ones, twos, threes, fours, sixes, and balls faced.
 - For each bowler, collect the name, runs given, overs bowled, and wickets taken.
2. Create a data structure to store the information:
 - Define a structure for the batsman that includes the data fields mentioned above.
 - Define a structure for the bowler that includes the data fields mentioned above.
3. Collect the data from the user:
 - Prompt the user to enter the number of batsmen and bowlers in the match.
 - Use a loop to collect the data for each player and store it in the corresponding structure.
4. Process the data:
 - Calculate the runs scored by each batsman, as well as their strike rate.
 - Calculate the economy rate of each bowler.
 - Update the records for highest scores, most wickets, etc.

5. Display the information:

- Allow the user to choose from several options, such as viewing the scorecard, match summary, or records.
- Based on the user's selection, display the appropriate information on the console.

6. Update the information:

- Allow the user to update the data for a batsman or bowler, if needed.
- Recalculate the scores and display the updated information.

2.5 Object Model

The object model describe the static structure of a system in terms of objects and relationship.

2.5.1 ER Diagram

ER (Entity-Relationship) diagram is a graphical representation of entities and their relationships to each other in a database. It is a high-level conceptual data model that is used to design and visualize the relationships between different entities in a database.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

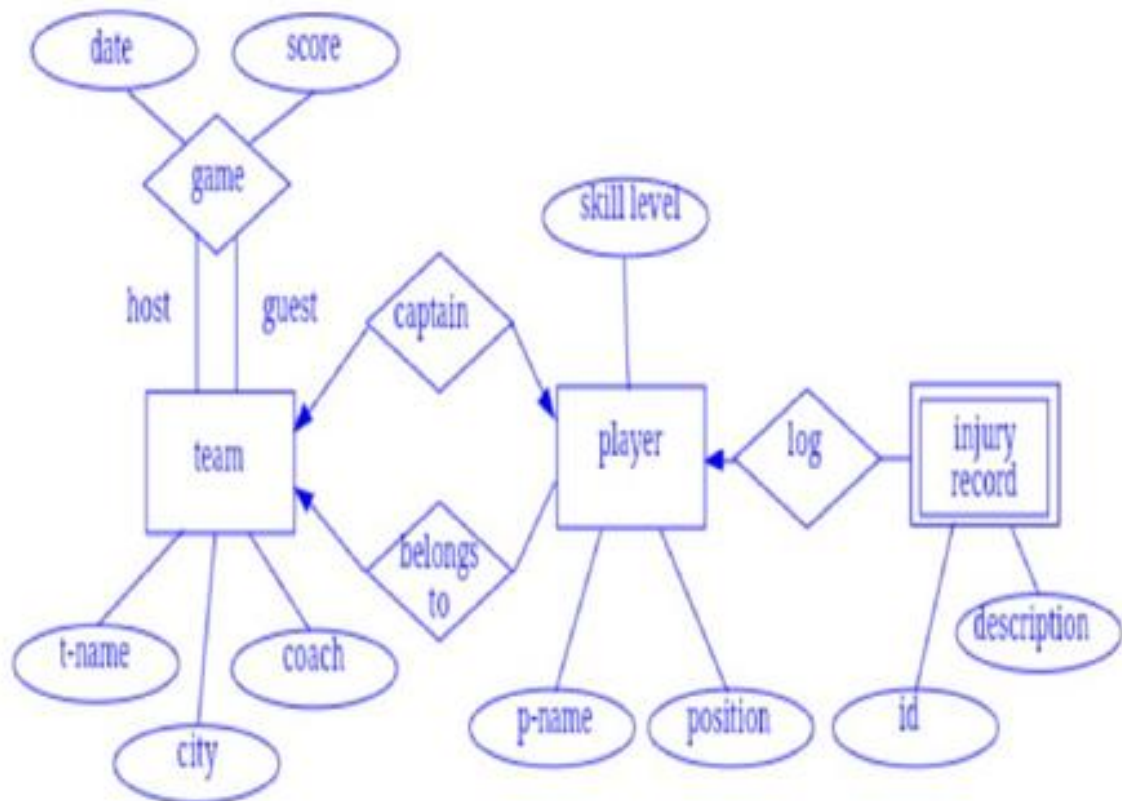


Fig. 2 (a) ER Diagram

2.5.2 State Diagram

A state diagram consists of a set of states and the transitions between them. Each state represents a specific condition or mode of operation of the system, while the transitions represent the events that cause the system to change from one state to another.

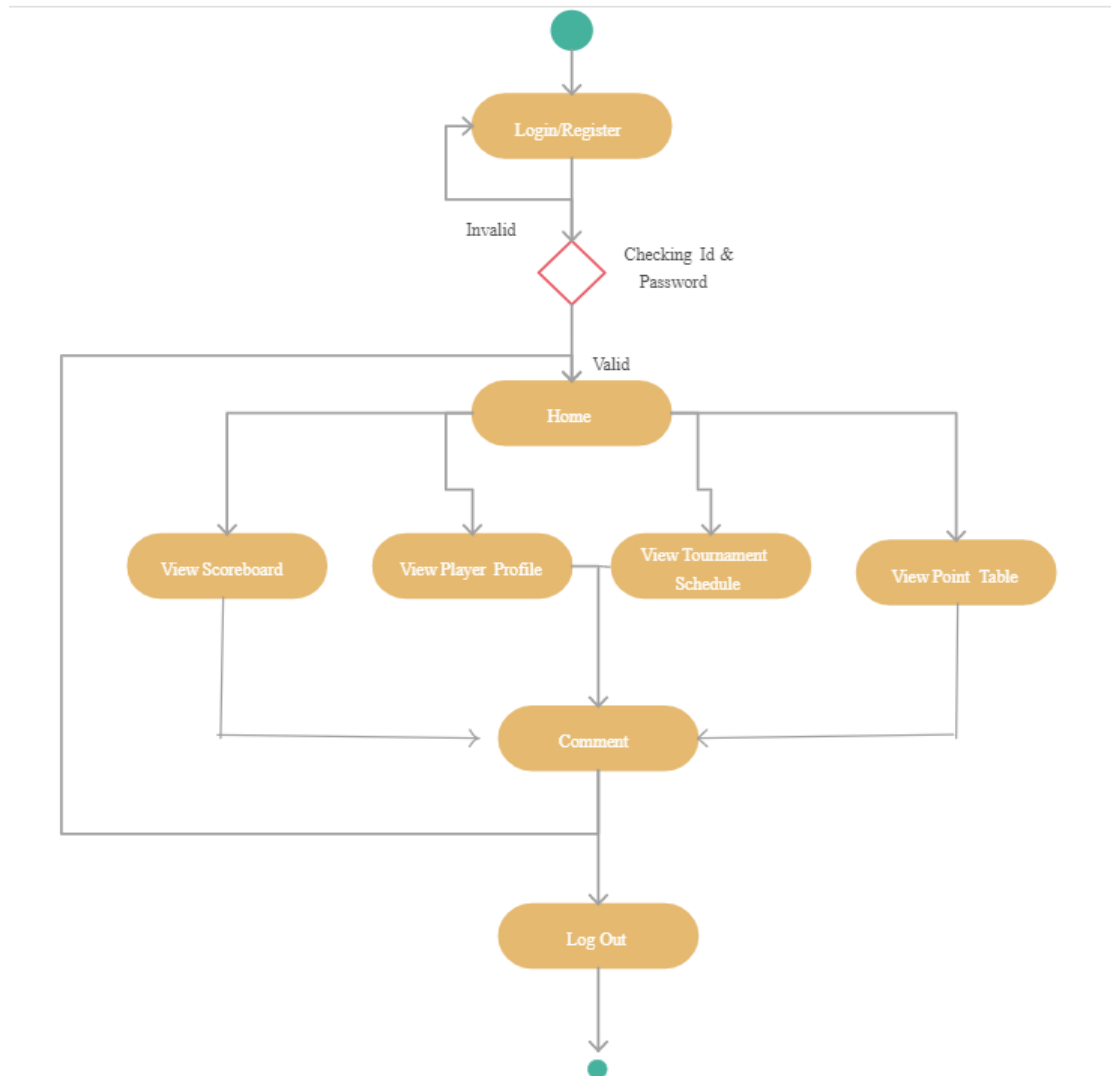


Fig. 2(b) *State Diagram*

2.5.3 Case Diagram

A use case diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that describes the interactions between users (or actors) and a system to achieve a specific goal or task. It is a graphical representation of the system's functional requirements and the external entities that interact with the system.

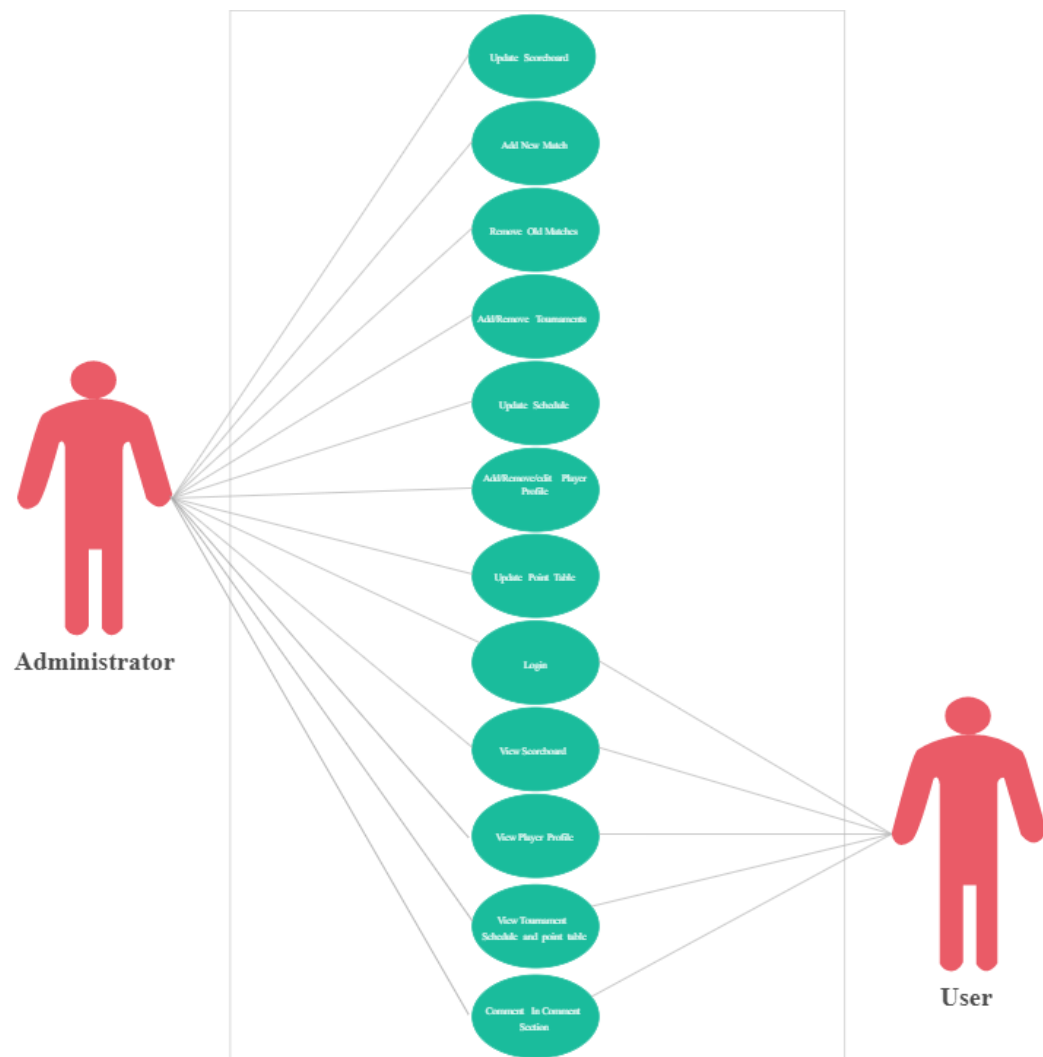


Fig. 2 (c) *Use Case Diagram*

2.5.4 Functional Model

A functional model is a representation or abstraction of a system or process that emphasizes its functional components and their interrelationships. It is typically used in engineering and software development to describe the functionality of a system or its components, rather than its physical or structural properties.

DFD Level 0

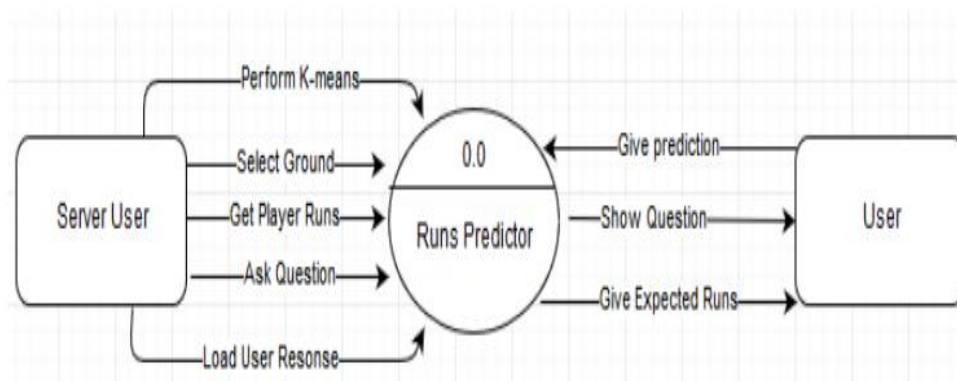


Fig. 2 (d) *DFD Level 0*

DFD Level 1

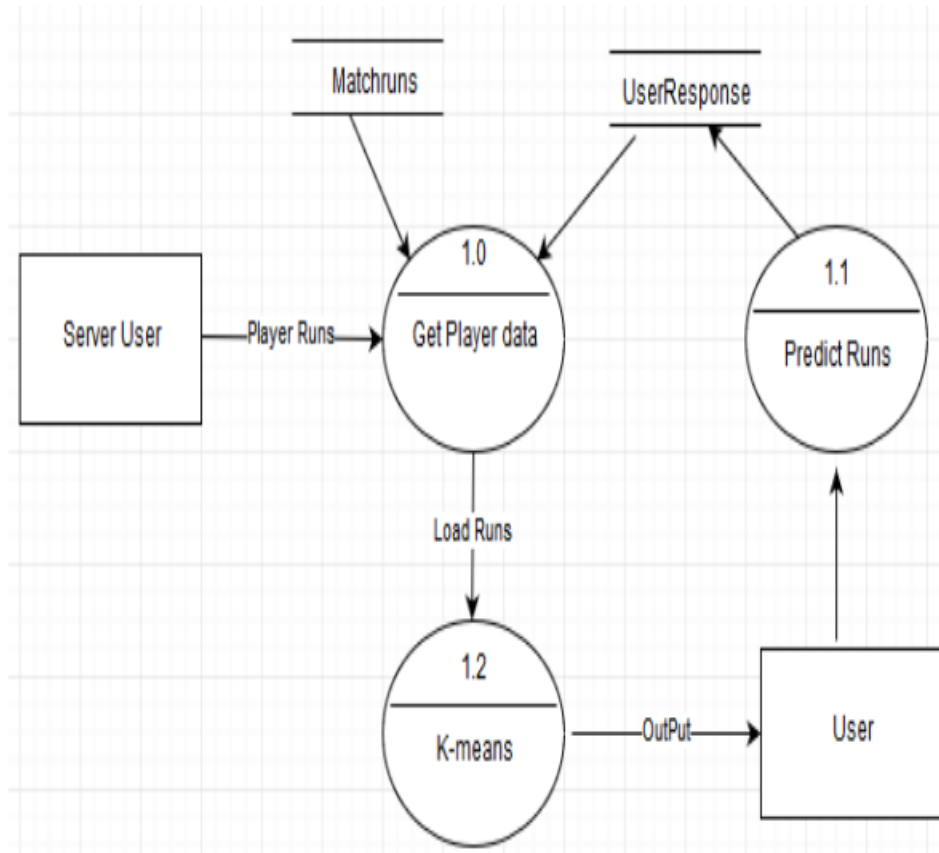


Fig. 2 (e) *DFD Level 1*

DFD Level 2

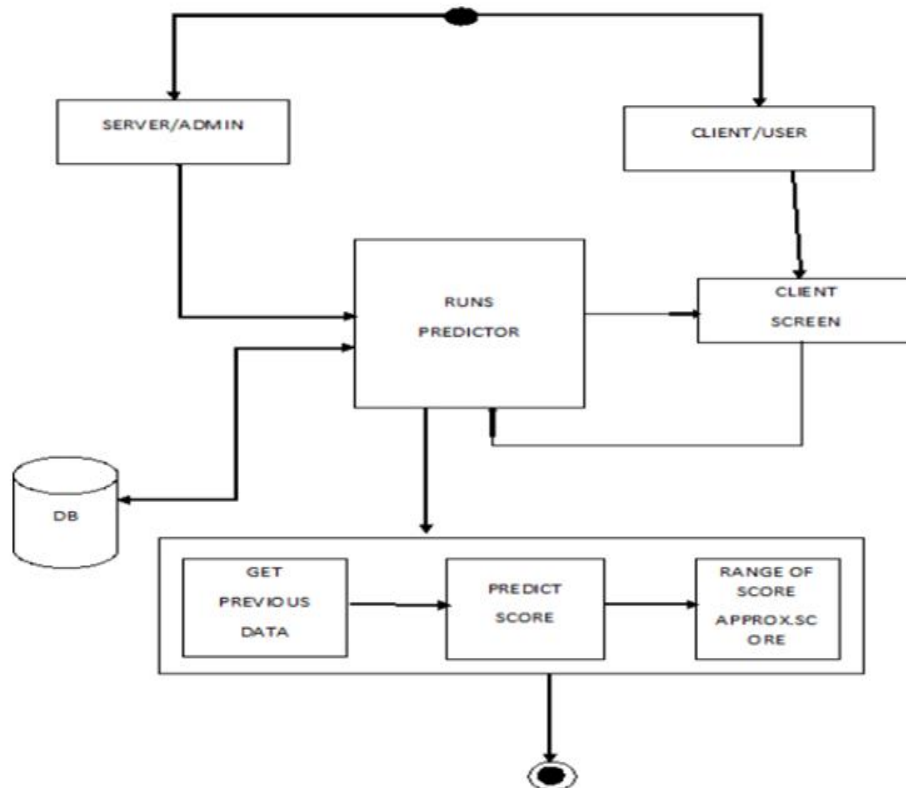


Fig. 2 (f) *DFD Level 2*

CHAPTER 3

DEVELOPMENT AND TESTING

A software development paradigm is a framework or approach to software development that provides guidelines, principles, and practices for developing high-quality software efficiently and effectively. There are several different software development paradigms, each with its own strengths and weaknesses, and different paradigms may be better suited to different types of projects and teams.

We have implemented incremental model for the development of the project.

3.1 Incremental Model

The incremental model is an iterative software development process where the software is developed and delivered in small, functional increments. Each increment builds upon the previous one, with each increment adding new functionality or improving existing functionality.

The incremental model is commonly used in software development projects where the requirements are not well understood or may change over time. By breaking the development process into smaller increments, the project team can adapt to changing requirements more easily and deliver software faster.

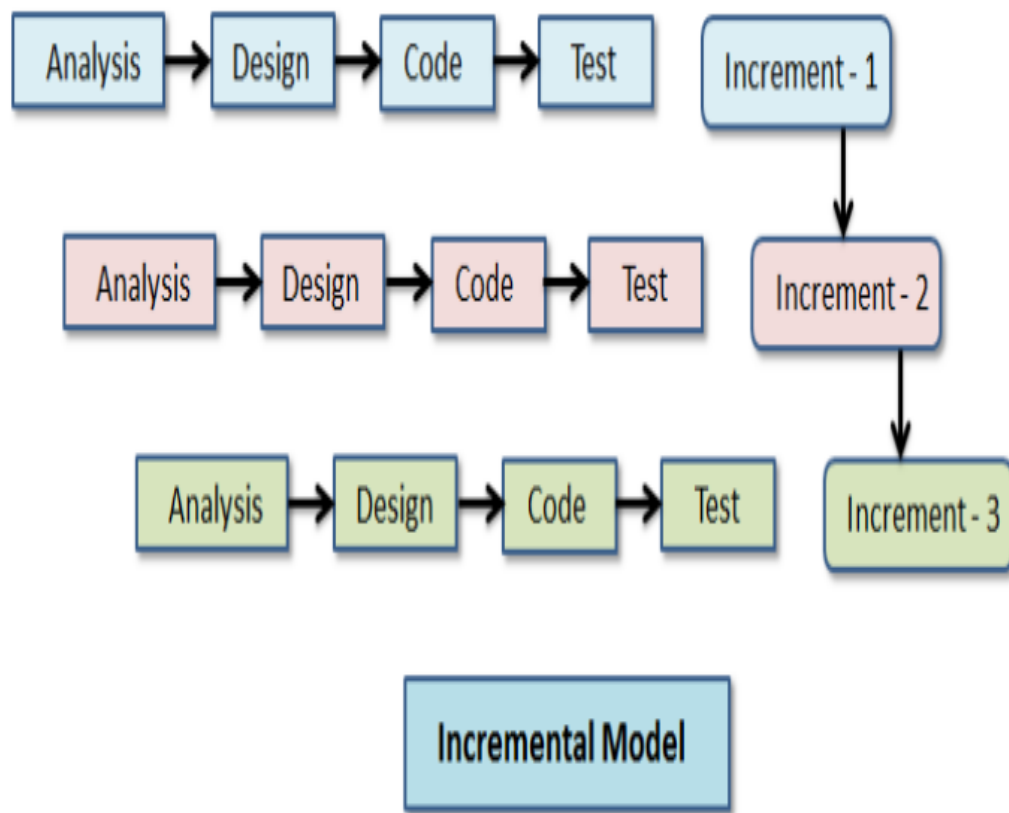


Fig. 3 (a) *Incremental Model*

3.1.1 Coding

Coding of the project is done in C and the basic modules/function working and concepts are next :-

3.2 Implementation

Step 1. Create a new *Project* in Visual Studio Code.

File > New text file or Ctrl + N

Step 2. First of all we will include all the *header* file.

```
#include <stdio.h>
#include <stdlib.h>
```

Step 3. Then we will take all the *variables* in the file.

```
struct batsman
{
    char name[25];
    int runs, score, balls, toruns, tobal, ones,
    twos, threes, fours, sixes;
    int max_six, max_run, max_four;
    float str;
}    p11[100], p13;

struct bowler
{
    char name[25];
    int runsgv, wkttkn, overs;
    int max_w;
    float econ;
}    p12[100], p14;
```

3.3 Testing

Test Case 1

Test Title: Response Time

Test ID: T1

Test Priority: High

Test Objective: To make sure that the system respond back time is efficient.

Description: Time is very important in any project or work. when we gives commands, it responds very quick. It responds in few milliseconds or seconds. A fast response time can improve user satisfaction and productivity, while a slow response time can lead to frustration and reduced productivity.

Test Case 2

Test Title: Accuracy

Test ID: T2

Test Priority: High

Test Objective: To assure that answers retrieved by system are accurate as per gathered data.

Description: A Cricket scorecard system is mainly used to get precise details and match summary. Accuracy is the most important criteria in our project. Accuracy is an important metric for assessing the quality and reliability of a system or process. It can determine the usefulness and reliability of computational systems and algorithms.

Test Case 3

Test Title: Approximation

Test ID: T3

Test priority: Moderate

Test Objective: To check approximate answers about calculations.

Description: There are times when mathematical calculation requires approximate value. In our project user want to see the Match summary, in this batsmen's strike rate and bowler's economy are calculates by the given data from the user. It's only possible when approximation of calculation is right.

Note: There might include a few more test cases and these test cases are also subject to change with the final software development.

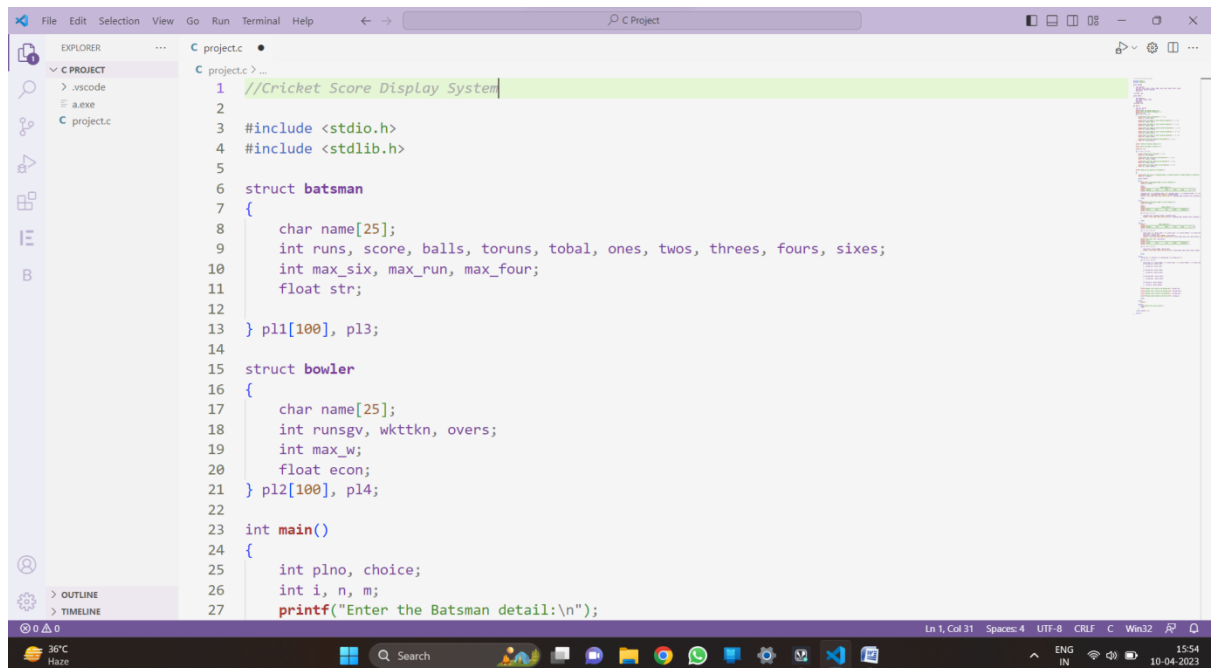
CHAPTER 4

IMPLEMENTATION AND MAINTENANCE

4.1 Software Deployment

As Cricket score display system is the users own system and its modules are separately coded and debugged on the user system itself for the reason of providing high data accuracy.

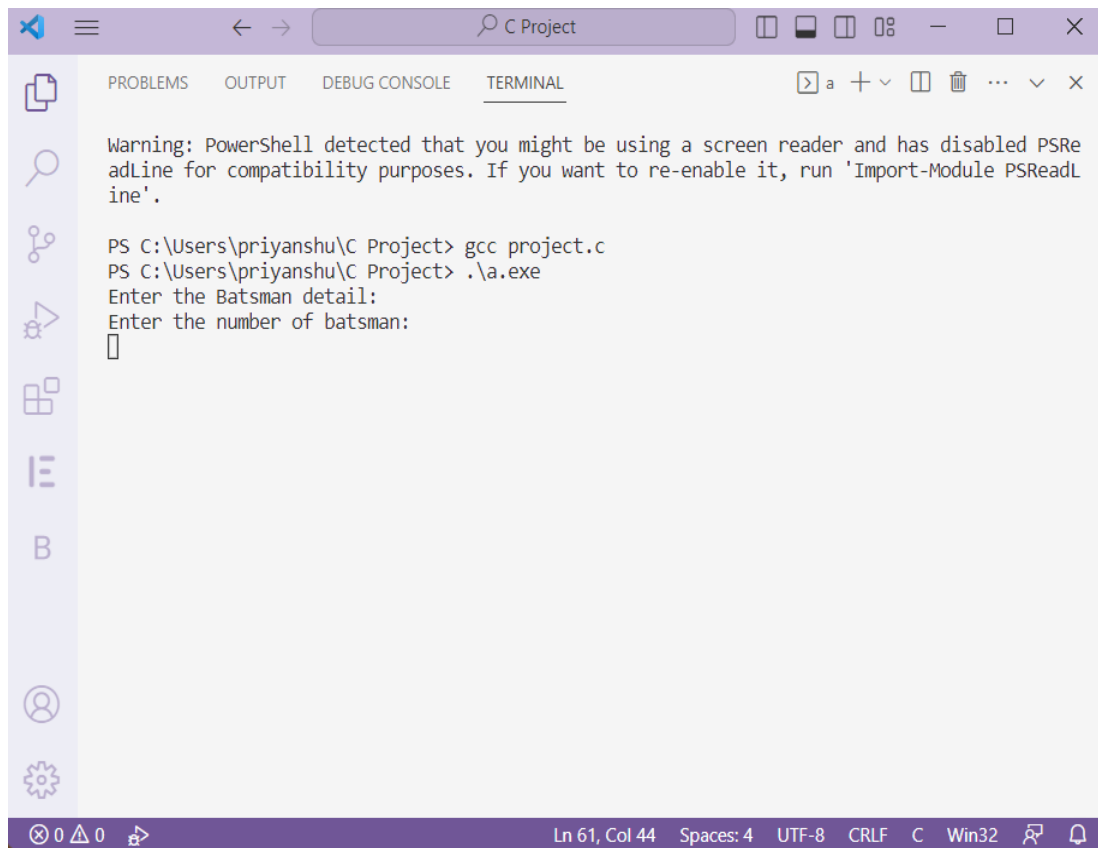
Here we have followed the all at once deployment method to deploy this project and below are some snapshots of the general Windows 11 Operating System with the deployed project.



```
1 //Cricket Score Display System
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 struct batsman
7 {
8     char name[25];
9     int runs, score, balls, toruns, tobal, ones, twos, threes, fours, sixes;
10    int max_six, max_run, max_four;
11    float str;
12 } p1[100], p13;
13
14 struct bowler
15 {
16     char name[25];
17     int runsgv, wkttkn, overs;
18     int max_w;
19     float econ;
20 } p12[100], p14;
21
22
23 int main()
24 {
25     int plno, choice;
26     int i, n, m;
27     printf("Enter the Batsman detail:\n");
```

Fig. 4 (a) *Coding Console*

4.1.1 Snapshots:-



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal window displays the following text:

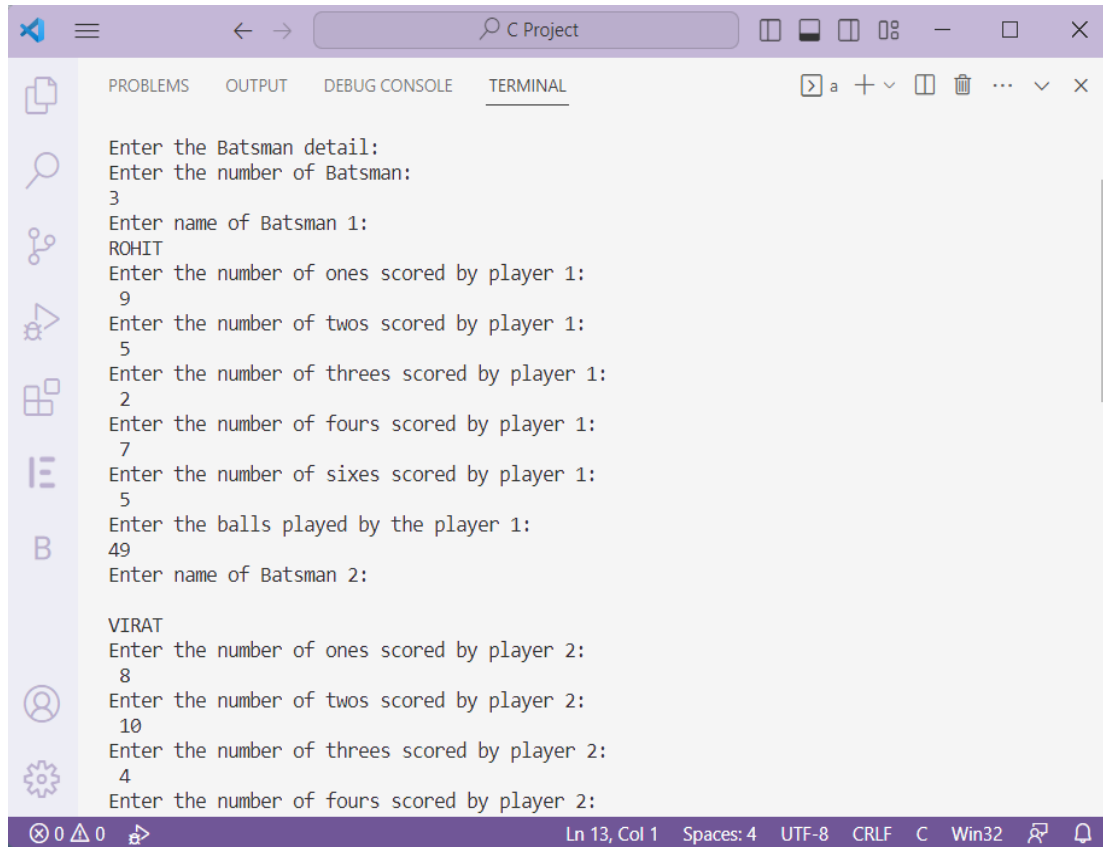
```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\priyanshu\C Project> gcc project.c
PS C:\Users\priyanshu\C Project> .\a.exe
Enter the Batsman detail:
Enter the number of batsman:
█
```

The status bar at the bottom indicates the current line and column: 'Ln 61, Col 44'. Other status bar information includes 'Spaces: 4', 'UTF-8', 'CRLF', 'C', 'Win32', and icons for search, settings, and notifications.

Fig. 4 (b) *First View*

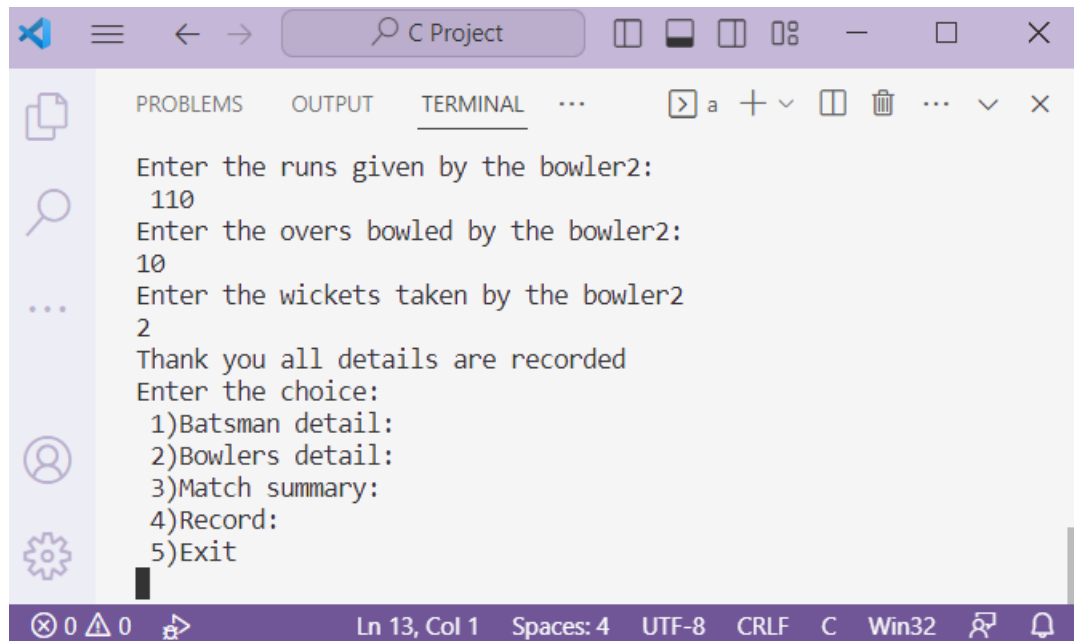
4.1.2 Query



```
Enter the Batsman detail:  
Enter the number of Batsman:  
3  
Enter name of Batsman 1:  
ROHIT  
Enter the number of ones scored by player 1:  
9  
Enter the number of twos scored by player 1:  
5  
Enter the number of threes scored by player 1:  
2  
Enter the number of fours scored by player 1:  
7  
Enter the number of sixes scored by player 1:  
5  
Enter the balls played by the player 1:  
49  
Enter name of Batsman 2:  
  
VIRAT  
Enter the number of ones scored by player 2:  
8  
Enter the number of twos scored by player 2:  
10  
Enter the number of threes scored by player 2:  
4  
Enter the number of fours scored by player 2:
```

Fig. 4 (c) *Query Look*

4.1.3 Tasks



```
Enter the runs given by the bowler2:
110
Enter the overs bowled by the bowler2:
10
Enter the wickets taken by the bowler2
2
Thank you all details are recorded
Enter the choice:
1)Batsman detail:
2)Bowlers detail:
3)Match summary:
4)Record:
5)Exit
```

Fig. 4 (d) *Performing Task*

4.2 Output

The output is the result of executing a program or code in a compiler. The output can be a message, a value, a file, or any other kind of data that the program produces as a result of its execution.

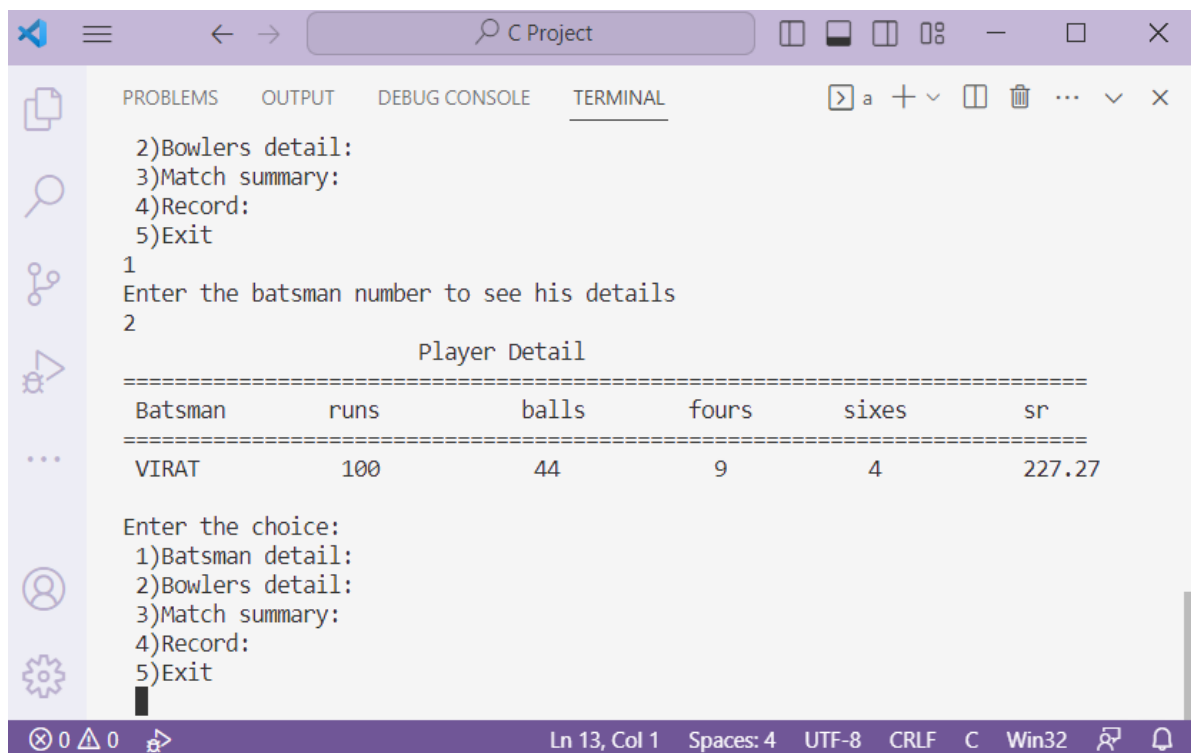
When running code in a compiler, the output can be used to determine if the program is functioning correctly and producing the desired results. The output can also be used to identify and diagnose any errors or bugs that may be present in the program.

4.2.1 Batsman Detail:-

In a cricket scorecard, the batsman details typically shows the batsman name, runs scored by the batsman, the total number of deliveries or balls faced by batsman, the total number of fours scored by batsman, the total number of sixes scored by batsman and the calculated average number of runs scored by the batsman per 100 balls faced is called Strike rate.

We choose '1' to see the any Batsman details .

The image below shows the 2nd Batsman's Detail :-



```
2)Bowlers detail:
3)Match summary:
4)Record:
5)Exit
1
Enter the batsman number to see his details
2

                        Player Detail
=====
Batsman      runs      balls      fours      sixes      sr
=====
VIRAT        100        44         9          4          227.27

Enter the choice:
1)Batsman detail:
2)Bowlers detail:
3)Match summary:
4)Record:
5)Exit
```

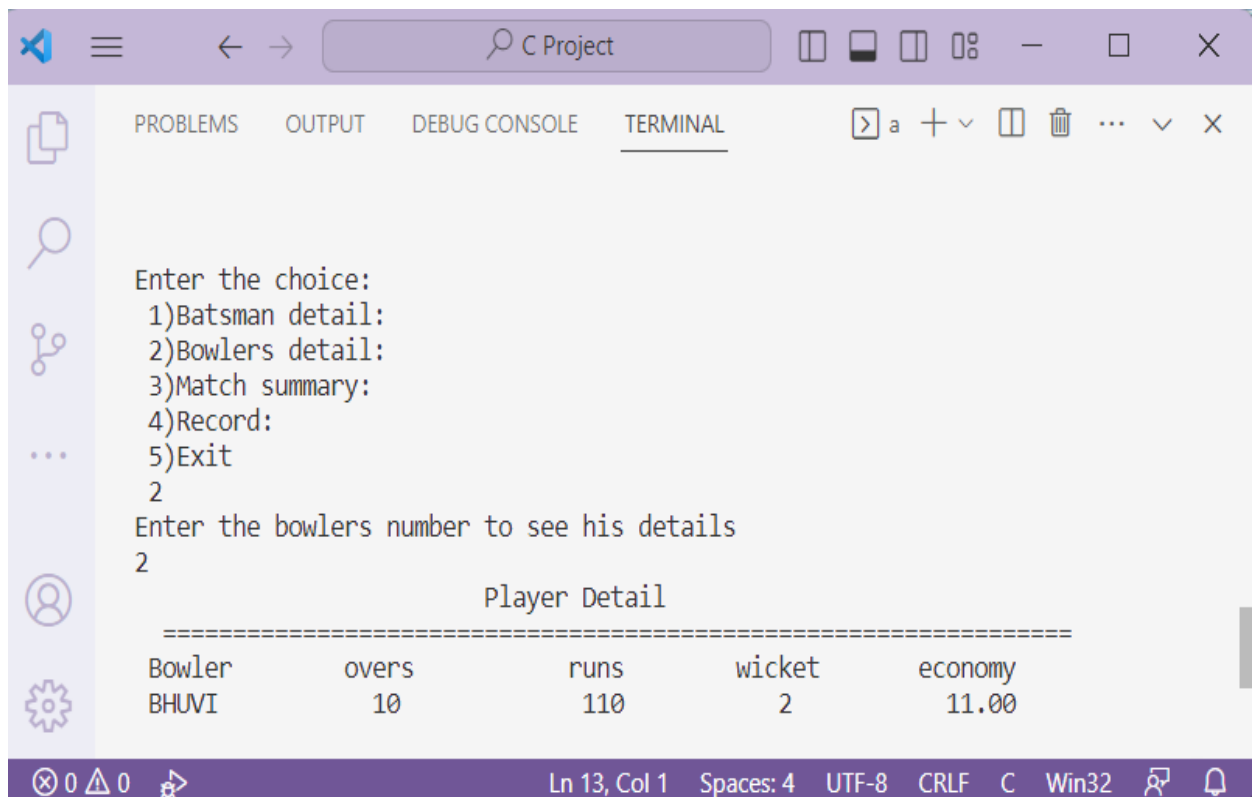
Fig. 4 (e) *Batsman Detail Output*

4.2.2 Bowler Detail:-

In a cricket scorecard, the Bowler details typically shows the Bowler's name, The number of overs bowled by the bowler, The total number of runs conceded by the bowler during the innings, The number of wickets taken by the bowler, The economy rate of the bowler, which is calculated by dividing the total runs conceded by the number of overs bowled and rounding to two decimal places.

We choose '2' to see the any Bowler details .

The image below shows the 2nd Bowler's Detail:-



```
Enter the choice:
1)Batsman detail:
2)Bowlers detail:
3)Match summary:
4)Record:
5)Exit
2
Enter the bowlers number to see his details
2

Player Detail
=====
Bowler      overs    runs    wicket    economy
BHUVI       10       110     2         11.00
```

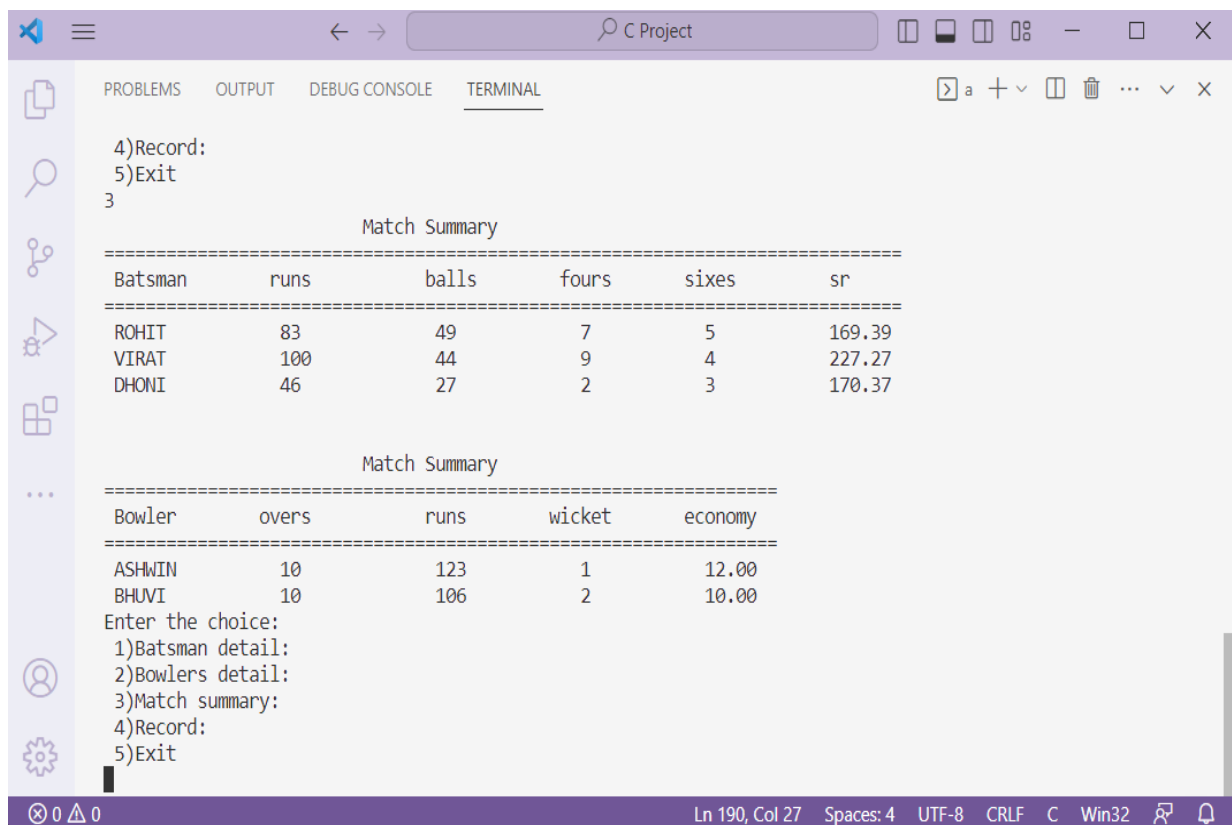
Fig. 4 (f) *Bowler's Detail Output*

4.2.3 Match Summary:-

The match summary in cricket is typically shown on the scorecard, which is a visual representation of the match's statistics. The scorecard displays information such as the runs scored by each player, the number of wickets taken, the number of overs bowled. The match summary is usually located at the end of the scorecard

We choose '3' to see the Match Summary.

The image below shows the Match Summary:-



```
4)Record:
5)Exit
3

=====
Match Summary
=====
Batsman      runs    balls    fours    sixes    sr
=====
ROHIT         83      49       7        5      169.39
VIRAT        100      44       9        4      227.27
DHONI         46      27       2        3      170.37

=====
Match Summary
=====
Bowler      overs    runs    wicket    economy
=====
ASHWIN       10      123      1        12.00
BHUVI        10      106      2        10.00
Enter the choice:
1)Batsman detail:
2)Bowlers detail:
3)Match summary:
4)Record:
5)Exit
```

Fig. 4 (g) Match Summary

CHAPTER 5

CONCLUSION & FUTURE SCOPE

5.1 Conclusion

This program is designed to record and display the details of cricket players, including batsmen and bowlers. The user is asked to enter the number of batsmen and bowlers, along with their respective details, such as the number of runs scored, the number of balls played, the number of wickets taken, etc. The program then allows the user to view the details of a particular player or the overall match summary. This program provides a simple way to record and display the details of cricket players, making it useful for coaches and players who want to track their performance.

In conclusion, the Cricket Score Display System program presented here is a simple implementation of a system that allows users to input details about the performance of a group of batsmen and bowlers and display relevant statistics about each player. The program uses structures to store information about the players and provides users with a menu to select different options for displaying player details, match summaries, and records. Overall, the program demonstrates the use of basic C programming concepts such as loops, arrays, and switch statements to create a functional system that can be used to track and analyze cricket scores.

5.2 Future scope:-

- Adding new features: The program can be extended to include new features that allow users to perform more advanced operations on the input data. For example, the program can be extended to support statistical analysis, data visualization, or machine learning algorithms.
- Improving performance: The program can be optimized to improve its performance. This can be achieved by using more efficient algorithms, parallel processing, or distributed computing.
- Enhancing user interface: The program's user interface can be enhanced to make it more intuitive and user-friendly. This can be achieved by using modern design principles and incorporating user feedback.
- Providing cloud-based solution: The program can be converted into a cloud-based solution that allows users to access it from anywhere, on any device. This can be achieved by using cloud computing technologies such as AWS, Azure, or Google Cloud.

- Addition of a GUI: The current program is a console-based program that may not be user-friendly for some people. The addition of a graphical user interface (GUI) will make it more accessible and easier to use for people who are not comfortable with the command line interface.
- Integration of a Database: The program can be enhanced by integrating a database that will store the details of all the batsmen and bowlers. The database will allow the program to retrieve and display the statistics of the players even after the program is closed.
- Live Scorecard: The program can be modified to fetch live scores of ongoing matches from various sources and display it to the user. This feature will be beneficial to people who cannot access live streaming services due to low internet bandwidth.
- Integration with Social Media: The program can be integrated with various social media platforms such as Twitter, Facebook, etc. The integration will allow users to share the player's performance statistics and live scores with their friends and followers.

These are some potential future scopes for the program that can enhance its functionality and make it more appealing to users. However, the implementation of these features will require a good understanding of programming languages and concepts.

REFERENCES

1. "The C Programming Language" by Brian W. Kernighan and Dennis M. Ritchie
2. "C Programming Absolute Beginner's Guide" by Greg Perry and Dean Miller
3. "Head First C: A Brain-Friendly Guide" by David Griffiths and Dawn Griffiths
4. "Learn C the Hard Way: Practical Exercises on the Computational Subjects You Keep Avoiding" by Zed A. Shaw
5. "Programming in C" by Stephen G. Kochan
6. "C Primer Plus" by Stephen Prata
7. "Effective C: An Introduction to Professional C Programming" by Robert C. Seacord
8. "C Programming for Arduino" by Julien Bayle
9. "C Programming: A Modern Approach" by K. N. King
10. "Expert C Programming: Deep C Secrets" by Peter van der Linden.

CODE

```
//Cricket Score Display System

#include <stdio.h>

#include <stdlib.h>


// structure to store the details of batsman

struct batsman

{

    char name[25];    // Name of the batsman

    int runs, score, balls, toruns, tobal, ones,
twos, threes, fours, sixes;

    int max_six, max_run, max_four;    // Maximum
number of sixes, runs and fours scored by batsman

    float str;

} p11[100], p13;
```

```

//structure to store details of bowler

struct bowler

{

    char name[25];        // Name of bowler

    int runsgv, wkttkn, overs;

    int max_w;

    float econ;          // Economy rate of bowler

} p12[100], p14;


// main function of the program.

int main()

{

    // Variables to store player no. and user choice

    int plno, choice;

```

```

int i, n, m;

// Ask user to enter details of batsmen

printf("Enter the Batsman detail:\n");

printf("Enter the number of Batsman:\n");

scanf("%d", &m);


// Loop of each batsman and ask user to enter
their detail

for (i = 0; i < m; i++)

{

    printf("Enter name of Batsman %d:\n", i + 1);

    scanf("%s", pl1[i].name);


    printf("Enter the number of ones scored by
player %d:\n ", i + 1);

```

```

scanf("%d", &p11[i].ones);

printf("Enter the number of twos scored by
player %d:\n ", i + 1);

scanf("%d", &p11[i].twos);

printf("Enter the number of threes scored by
player %d:\n ", i + 1);

scanf("%d", &p11[i].threes);

printf("Enter the number of fours scored by
player %d:\n ", i + 1);

scanf("%d", &p11[i].fours);

printf("Enter the number of sixes scored by
player %d:\n ", i + 1);

scanf("%d", &p11[i].sixes);

```

```

        printf("Enter the balls played by the player
%d:\n", i + 1);

        scanf("%d", &pl1[i].balls);

    }

```

```

// Ask user to enter details of bowlers

```

```

    printf("\nEnter the Bowlers details:\n");

```

```

    printf("Enter the number of Bowlers:\n");

```

```

    scanf("%d", &n);

```

```

// Loop for each bowler and ask user to enter their
details

```

```

    for (i = 0; i < n; i++)

```

```

    {

```

```

        printf("\nEnter name of bowler%d:\n", i + 1);

```

```

        scanf("%s", pl2[i].name);

```

```

        printf("Enter the runs given by the
bowler%d:\n ", i + 1);

        scanf("%d", &pl2[i].runsgv);

        printf("Enter the overs bowled by the
bowler%d:\n", i + 1);

        scanf("%d", &pl2[i].overs);

        printf("Enter the wickets taken by the
bowler%d\n", i + 1);

        scanf("%d", &pl2[i].wkttkn);

    }

    printf("Thank you all details are recorded\n");

do

```

```

{

    // display menu to user

    printf("Enter the choice:\n 1)Batsman
detail:\n 2)Bowlers detail:\n 3)Match summary:\n
4)Record:\n 5)Exit\n ");

    scanf("%d", &choice);


    switch (choice)
    {

        case 1:

            printf("Enter the batsman number to see
his details\n");

            scanf("%d", &plno);


            // Print the details of particular batsman

            plno--;

            printf("                                Player
Detail\n");

```

```

printf("=====
=====\\n");

printf(" Batsman      runs
balls      fours      sixes      sr      \\n");

printf("=====
=====\\n");

// Calculate batsman statistics

pl1[plno].runs = (1 * pl1[plno].ones) +
(2 * pl1[plno].twos) + (3 * pl1[plno].threes) + (4 *
pl1[plno].fours) + (6 * pl1[plno].sixes);

pl1[plno].str = (pl1[plno].runs * 100.00)
/ pl1[plno].balls;

printf(" %-15s %-14d %-13d %-11d %-11d %-
9.2f\\n\\n", pl1[plno].name, pl1[plno].runs,
pl1[plno].balls, pl1[plno].fours, pl1[plno].sixes,
pl1[plno].str);

```



```

        break;

    case 2:

        printf("Enter the bowlers number to see his
details\n");

        scanf("%d", &plno);

        //Print the details of particular bowle

        plno--;

        printf("                                Player Detail\n
");

        printf("=====
=====\\n");

        printf(" Bowler            overs            runs
wicket            economy\\n");

```

```

printf("=====
=====\\n");

    // Calculate bowler statistics

    pl2[plno].econ = pl2[plno].runsgv /
pl2[plno].overs;

    printf(" %-15s %-14d %-13d %-11d %-11.2f\\n\\n",
pl2[plno].name, pl2[plno].overs, pl2[plno].runsgv,
pl2[plno].wkttkn, pl2[plno].econ);

    break;

    case 3:

        printf("                                Match
Summary\\n");

```

```

printf("=====
=====\\n");

printf(" Batsman      runs      balls
fours      sixes      sr      \\n");

printf("=====
=====\\n");

for (i = 0; i < m; i++) {

    pl1[i].runs = (1 * pl1[i].ones) + (2 *
pl1[i].twos) + (3 * pl1[i].threes) + (4 *
pl1[i].fours) + (6 * pl1[i].sixes);

    pl1[i].str = (pl1[i].runs * 100.00) /
pl1[i].balls;

```

```

        printf(" %-15s %-14d %-13d %-11d %-11d %-
9.2f\n", pl1[i].name, pl1[i].runs, pl1[i].balls,
pl1[i].fours, pl1[i].sixes, pl1[i].str);

    }

    printf("\n\n");

    printf("                                Match Summary\n");

    printf("=====\n");

    printf(" Bowler            overs            runs
wicket            economy\n");

    printf("=====\n");

```

```

    for (i = 0; i < n; i++) {

        pl2[i].econ = pl2[i].runsgv / pl2[i].overs;

        printf(" %-15s %-14d %-13d %-11d %-11.2f\n",
pl2[i].name, pl2[i].overs, pl2[i].runsgv,
pl2[i].wkttkn, pl2[i].econ);

    }

    break;

    case 4:

        pl3.max_run = 0, pl4.max_w = 0,
pl3.max_four = 0, pl3.max_six = 0;

        for (i = 0; i < m; i++)

        {

            pl1[i].runs = (1 * pl1[i].ones) + (2
* pl1[i].twos) + (3 * pl1[i].threes) + (4 *
pl1[i].fours) + (6 * pl1[i].sixes);

```

```

        if (p13.max_run < p11[i].runs)
        {
            p13.max_run = p11[i].runs;
        }

        if (p13.max_six < p11[i].sixes)

{
            p13.max_six = p11[i].sixes;
        }

        if (p13.max_four < p11[i].fours)
        {
            p13.max_four = p11[i].fours;
        }

```

```

        if (p14.max_w < p12[i].wkttkn)
        {
            p14.max_w = p12[i].wkttkn;
        }
    }

    printf("Highest runs scored by the
batsman:%d\n", p13.max_run);

    printf("Maximum fours scored by the
batsman:%d\n", p13.max_four);

    printf("Maximum sixes scored by the
batsman%d:\n", p13.max_six);

    printf("Maximum wickets taken by the
bowler:%d\n", p14.max_w);

```

```

        break;

//exit to the code

case 5:

    exit(1);

// Print error message for invalid input

default:

    printf("Enter the correct choice\n");

    break;

}

} while (choice != 5);

return 0;

}

```