Title: **Cricket Manager**

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering in

Artificial Intelligence and Data Science

by

Arunim Chakraborty(7)

Kshitij Shidore(51)

Rupesh Dhirwani(10)

Siddhant Dongre(12)

under the guidance of

Supervisor (s):

Dr. Anjali Yeole

Department of Artificial Intelligence and Data Science

Since 1962

Vivekanand Education Society's

Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that **Mr Arunim Chakraborty, Mr Kshitij Shidore, Mr Rupesh Dhirwani and Mr Siddhant Dongre** of Second Year of Artificial Intelligence and Data Science studying under the University of Mumbai have satisfactorily presented the Mini Project entitled **Cricket Manager** as a part of the MINI-PROJECT for Semester-IV under the guidance of **Dr. Anjali Yeole** in the year 2021-2022.

Date: **30/04/2022**

(Name and sign)
Head of Department

(Name and sign)
Supervisor/Guide

Vivekanand Education Society's
Institute of Technology
(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)
Department of Artificial Intelligence and Data Science

DECLARATION

We, **Arunim Chakraborty, Kshitij Shidore, Rupesh Dhirwani and Siddhant Dongre** from **D6AD**, declare that this project represents our ideas in our own words without plagiarism and wherever others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our project work.

We declare that we have maintained a minimum 75% attendance, as per the University of Mumbai norms.

We understand that any violation of the above will be cause for disciplinary action by the Institute.

Yours Faithfully:

1._____Arunim Chakraborty_30/04/22___

2._____Kshitij Shidore_30/04/22_____

3._____Rupesh Dhirwani_30/04/22_____

4._____Siddhant Dongre_30/04/22_____

Table of Contents

Introduction

Cricket is the second most popular sport in the world. Given its popularity, its no surprise that it is also one of those topics upon which numerous games are create. However, there is a severe lack in the simulation part of the pack and an even bigger deficit where realism of the simulations is involved.

This is where our game, Cricket Manager comes in. It is a game that gives the player a sense of environment in which he/she can make and manage his/her own cricket team where a player will be able to handle a team in a popular sports environment, granting full control over finances, tactics, and backroom management. It allows the player to manage his/her own virtual cricket team against an AI as a competitor virtually.

Problem Statement

To create a Cricket Manager game which is able to simulate the realistic aspect of the sport. This game recreates the experience of handling a team in a sports environment, granting full control over tactics and backroom management. Players can select batting attributes like aggression and batting order and bowling attributes like line, length, bowler to bowl, and field aggression. At a time the user will be able to either bowl or bat and the AI will do the opposite and thus set the counter strategies using various algorithms.

Objectives

- Create a backend code that is able to simulate the various intricacies of the sport
- Creating an AI that will respond to the User's tactics almost like a real opposition captain.
- Creating a database of all players from all team with the different attributes.
- Creating an Interactive GUI using which users can select their tactics and play the game
- Implementing the graphics part of the game for example the cricket field, fielders, and balls, etc.

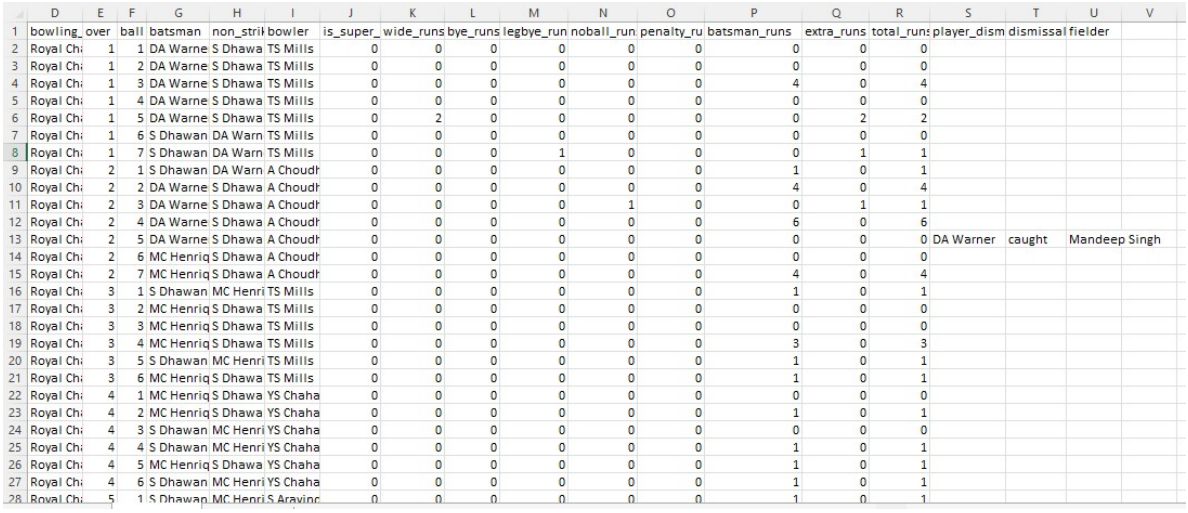
Scope

The game itself is highly strategic and thus requires a lot of logical thinking and strategizing on the player's part. This provides users with a platform to showcase their strategic ideas and critical thinking.

Literature Survey

One of Cricket Manager's biggest strength has always been its realistic outcomes. We have obtained ball by ball data of the entire IPL 2016

(FIG 1)KAGGLE DATA BASE



	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	bowling_	over	ball	batsman	non_strik	bowler	is_super_	wide_runs	bye_runs	legbye_run	noball_run	penalty_ru	batsman_runs	extra_runs	total_runs	player_dism	dismissal	fielder	
2	Royal Chi	1	1	DA Warne	S Dhawa	TS Mills	0	0	0	0	0	0	0	0	0				
3	Royal Chi	1	2	DA Warne	S Dhawa	TS Mills	0	0	0	0	0	0	0	0	0				
4	Royal Chi	1	3	DA Warne	S Dhawa	TS Mills	0	0	0	0	0	0	4	0	4				
5	Royal Chi	1	4	DA Warne	S Dhawa	TS Mills	0	0	0	0	0	0	0	0	0				
6	Royal Chi	1	5	DA Warne	S Dhawa	TS Mills	0	2	0	0	0	0	0	2	2				
7	Royal Chi	1	6	S Dhawan	DA Warn	TS Mills	0	0	0	0	0	0	0	0	0				
8	Royal Chi	1	7	S Dhawan	DA Warn	TS Mills	0	0	0	1	0	0	0	1	1				
9	Royal Chi	2	1	S Dhawan	DA Warn	A Choudh	0	0	0	0	0	0	1	0	1				
10	Royal Chi	2	2	DA Warne	S Dhawa	A Choudh	0	0	0	0	0	0	4	0	4				
11	Royal Chi	2	3	DA Warne	S Dhawa	A Choudh	0	0	0	0	1	0	0	1	1				
12	Royal Chi	2	4	DA Warne	S Dhawa	A Choudh	0	0	0	0	0	0	6	0	6				
13	Royal Chi	2	5	DA Warne	S Dhawa	A Choudh	0	0	0	0	0	0	0	0	0	DA Warner	caught	Mandeep Singh	
14	Royal Chi	2	6	MC Henriq	S Dhawa	A Choudh	0	0	0	0	0	0	0	0	0				
15	Royal Chi	2	7	MC Henriq	S Dhawa	A Choudh	0	0	0	0	0	0	4	0	4				
16	Royal Chi	3	1	S Dhawan	MC Henri	TS Mills	0	0	0	0	0	0	1	0	1				
17	Royal Chi	3	2	MC Henriq	S Dhawa	TS Mills	0	0	0	0	0	0	0	0	0				
18	Royal Chi	3	3	MC Henriq	S Dhawa	TS Mills	0	0	0	0	0	0	0	0	0				
19	Royal Chi	3	4	MC Henriq	S Dhawa	TS Mills	0	0	0	0	0	0	3	0	3				
20	Royal Chi	3	5	S Dhawan	MC Henri	TS Mills	0	0	0	0	0	0	1	0	1				
21	Royal Chi	3	6	MC Henriq	S Dhawa	TS Mills	0	0	0	0	0	0	1	0	1				
22	Royal Chi	4	1	MC Henriq	S Dhawa	YS Chaha	0	0	0	0	0	0	0	0	0				
23	Royal Chi	4	2	MC Henriq	S Dhawa	YS Chaha	0	0	0	0	0	0	1	0	1				
24	Royal Chi	4	3	S Dhawan	MC Henri	YS Chaha	0	0	0	0	0	0	0	0	0				
25	Royal Chi	4	4	S Dhawan	MC Henri	YS Chaha	0	0	0	0	0	0	1	0	1				
26	Royal Chi	4	5	MC Henriq	S Dhawa	YS Chaha	0	0	0	0	0	0	1	0	1				
27	Royal Chi	4	6	S Dhawan	MC Henri	YS Chaha	0	0	0	0	0	0	1	0	1				
28	Royal Chi	5	1	S Dhawan	MC Henri	S Aravinc	0	0	0	0	0	0	1	0	1				

Analyzing this entire data, we obtained some valuable insights into how we can make our game hyper-realistic.

Despite the fact that our game is cricket based, a game from another completely different sport has been a constant inspiration. The game is Football Manager. **Football Manager** is yet another game that mirrors the real football game on a giant scale. The level of details, complexity and AI – part in it is beyond the scope of our project. But it has laid some very basic foundations for our project.

Our aim is to add more details into the game such as AI that would compete against the player , also we will look to add more data-base into the game. Simultaneously , there would be more options other than aggression , defensive playing of the game.

There we would also be adding finance (to purchase players from the points) , tactics and strategy(for the outcomes that has to be decided by the player against the Artificial Intelligence) , Player Health(an important aspect that has to be considered for the player's fitness).

We have also studied the convential cricket games such as EA CRICKET 7 , DON BRADMAN CRICKET and also the REAL SPORT,TOP ELEVEN(Cricket Manager). Also we have studied there variable declaring skills such as batsman and bowler skill and how they work with the backend, we have also studied various cricket data sets from KAGGLE website.

Analysis and Design

(FIG 2) PROBABILITIES OF DIFFERENT OUTCOMES

	freq	runs	prob
0run	61148		0.40653131
1run	55497	55497	0.36896167
2run	9705	19410	0.06452192
3run	509	1527	0.00338399
4run	17032	68128	0.11323414
6run	6523	39138	0.04336697
total	150414	183700	1
dismissals	7438		0.04945018

As can be seen from the above picture, we recorded the frequency of each event from the excel data we had obtained and calculated the probability of each event. We then created variable step ladder which basically relates between which values of dice variable, each outcome should occur.

(FIG 3) RANGES FOR THE DICE VALUE

runs	high	low
0	0	-137
1	124	1
2	146	125
3	147	147
4	185	148
6	200	186
out	-120	-137

We then created variable step ladder which basically relates between which values of dice variable, each outcome should occur; as can be seen in the picture above. For example, should the dice value (an algebraic function between Batsman Skill, Bowler Skill and a random luck factor variable) assume a value between 125 and 146, two runs will be added to the score as can be seen from the picture above.

(FIG 4)DICE VARIABLE

```
int dice = 2*bat_skill - 2*bowl_skill + luck;
```

Our entire backend code is built around this one simple equation where bat_skill is the Batsman Skill and bowl_skill the bowler skill both of which are to be taken from a predefined Database.

This dice is fed into the next part of our algorithm that is the if else statements that determine the outcome.

(FIG 5) BACKEND ALGORITHM

```
if(dice<=hout){
out(true);
        }
        if (dice >= l0 && dice <= h0) {
run0();

}
        if (dice >= l1 && dice <= h1) {
run1();
        }
        if (dice >= l2 && dice <= h2) {
run2();
        }
        if(dice==l3){
run3();
        }
        if (dice >= l4 && dice <= h4) {
run4();
        }
        if (dice >= l6) {
run6();
        }
```

From these if else statements the various outcome methods are called which get executed accordingly.

(FIG 6) OUTCOME FUNCTIONS

```
int run0(){
        System.out.println("0 Runs");
balls++;          game_balls++;
        imdt_outcome[game_balls] = "0";          return(0);
    }
    int run1(){
        System.out.println("1 Run");
runs++;          game_runs++;
balls++;          game_balls++;
        imdt_outcome[game_balls] = "1";          return(1);
    }
    int run2(){
        System.out.println("2 Run");
runs+=2;          game_runs+=2;
balls++;          game_balls++;
        imdt_outcome[game_balls] = "2";          return(2);
    }
    int run3() {
        System.out.println("3 Run");
runs += 3;          game_runs+=3;
balls++;          game_balls++;
        imdt_outcome[game_balls] = "3";          return(3);
    }
```

```
    int run4() {
        System.out.println("4 Run");
runs += 4;          game_runs+=4;
balls++;          game_balls++;
        imdt_outcome[game_balls] = "4";          return(4);
    }
    int run6() {
        System.out.println("6 Run");
runs += 6;          game_runs+=6;
balls++;          game_balls++;
        imdt_outcome[game_balls] = "6";          return(6);
    }
```

This is the main backend part of the algorithm.

FIGURE 7 - (how the skills are determined for both batsmen and bowlers).

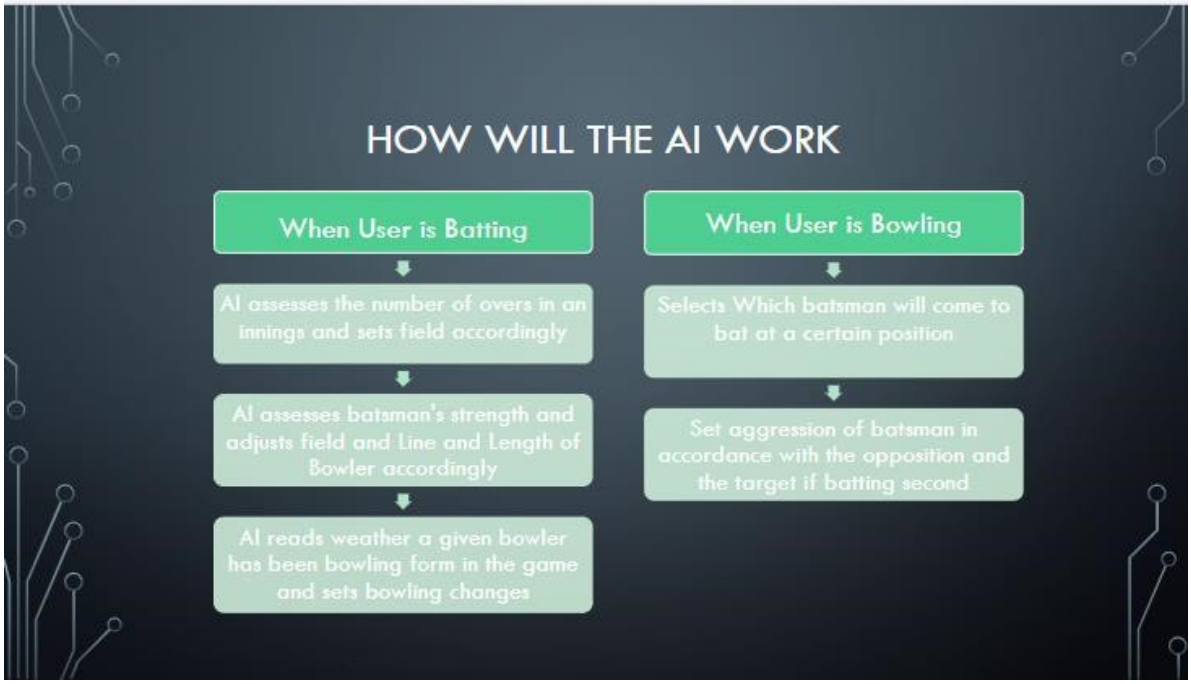


FIGURE 8 (how will the AI respond to various strategies of put in place by the user)

Algorithm

```
// use byte instead of int (-127 to 127)
Variables
1.Here we have declared the variables for the program & database
2.We are going to declare the variables in an interface which is be going to
inherit in the backend code.
1  luck_factor
2  bat_form
3  bowl_form
4  pitch_condition
5  overhead_condition
6  strength
7  bat_agression
8  field
9  line
10 length
11 skills
11.1 batting
11.1.1 front_foot //out of 100
11.1.2 back_foot
11.1.3 off_side
11.1.4 leg_side
11.1.5 against_pace
11.1.6 against_spin
11.1.7 left_or_right
11.1.8 natural_agression
```

ALGORITHM DESCRIPTION

We determine the dice variable which is batsman skill minus bowler skill plus luck factor which is a random variable between a certain range. This dice variable is then fed into an if else control structure which determines the outcome according to the value of the dice variable. Accordingly functions for different outcomes like run0, run4, out (true) are called.

We first determine the length of the ball. Upon that we then decide weather batsman will play on the front foot or the back foot. Similarly, we determine the line of the ball and decide weather batsman is going to play on offside or onside. Then we match the corresponding skills to determine the batsman skill. For bowlers, its more or less constant with swing and seam being characteristics of pacers and spin and drift being that of spinners.

For field and batsman aggression, higher aggression means wider range for the luck factor while lower aggressiveness mean narrower range for luck factor.

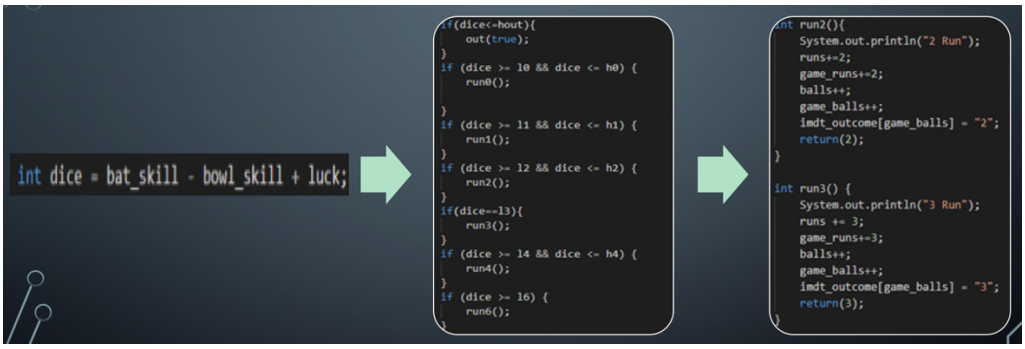


FIGURE – Main Backend algorithm

Design

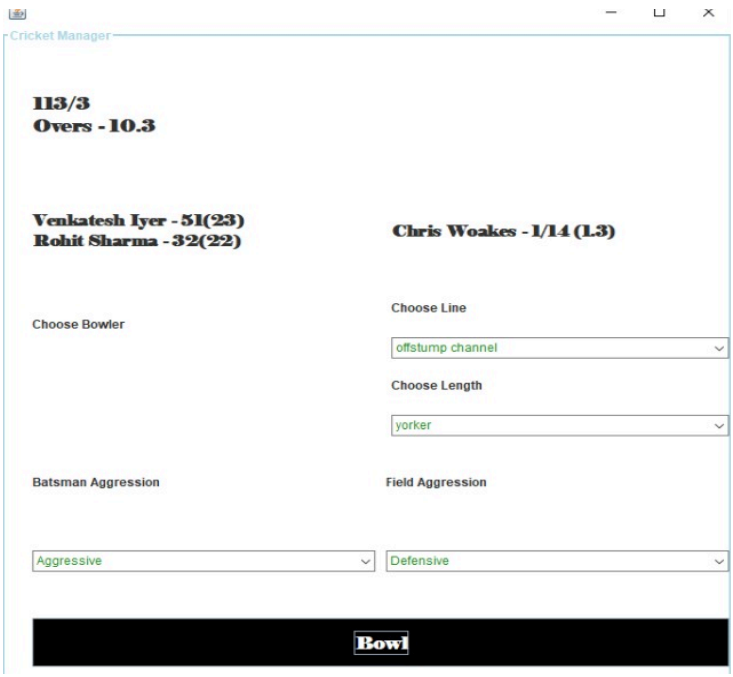


FIGURE 9 – UI when user's team will be batting

The above picture is the UI when the user's selected team is batting. The user can see the scoreboard with the stats of both bowler and batsman. In the lower side, the user can set aggression of the two batsmen according to his or her wishes.

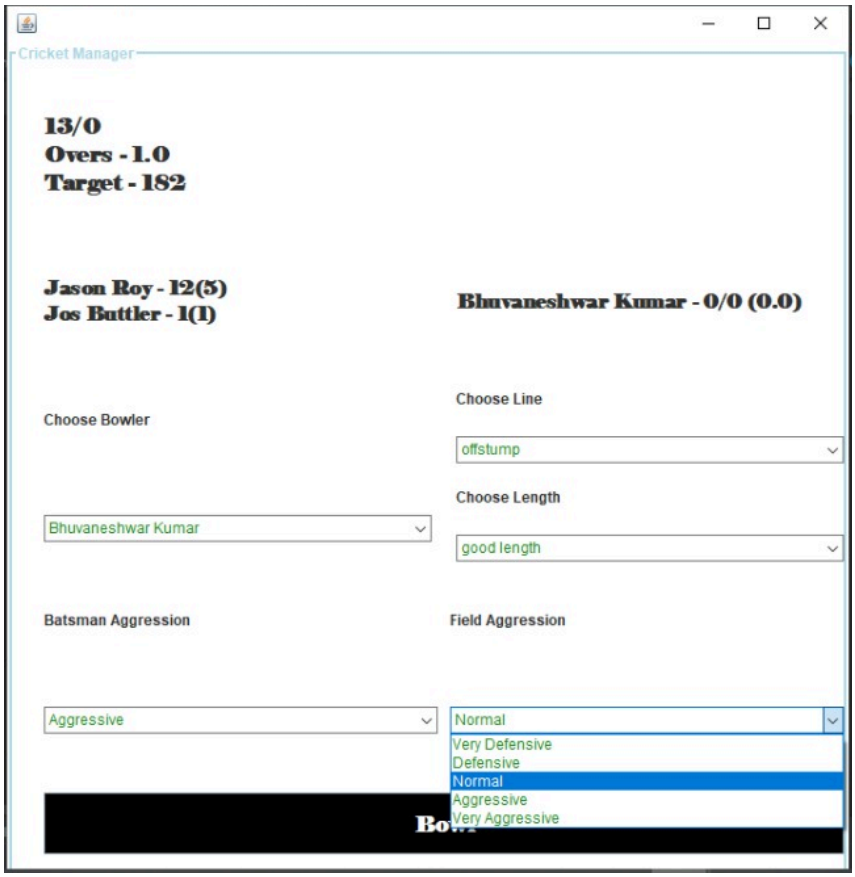
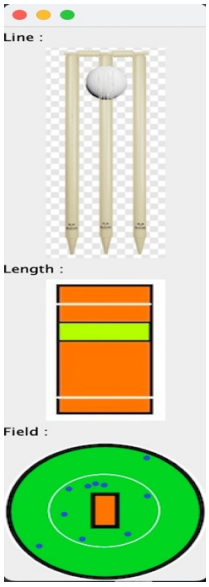


Figure 10 – UI when user's team will be bowling

The picture above represents the UI when the user's selected team will be bowling. As usual we can see the Scoreboard and the current stats. In the middle side, the user can select the field placement, Line and the length of the bowler according to his wishes. After the end of each over the user will also have the option of selecting which bowler would bowl.

The user will also be able to see the field, line and length of the next delivery irrespective of whether he/she will be batting/bowling.



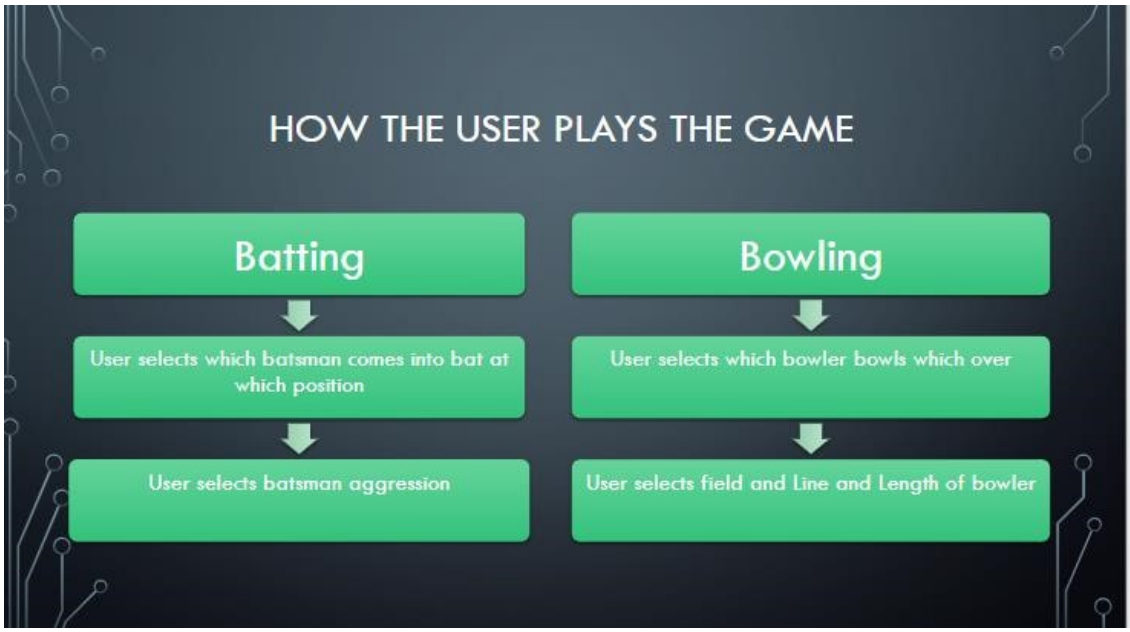


FIGURE 11 (Control flow of the application while the user is batting/bowling. User selects batting position and batsman aggression while batting and selects bowler to bowl and line, length, and field while bowling.)

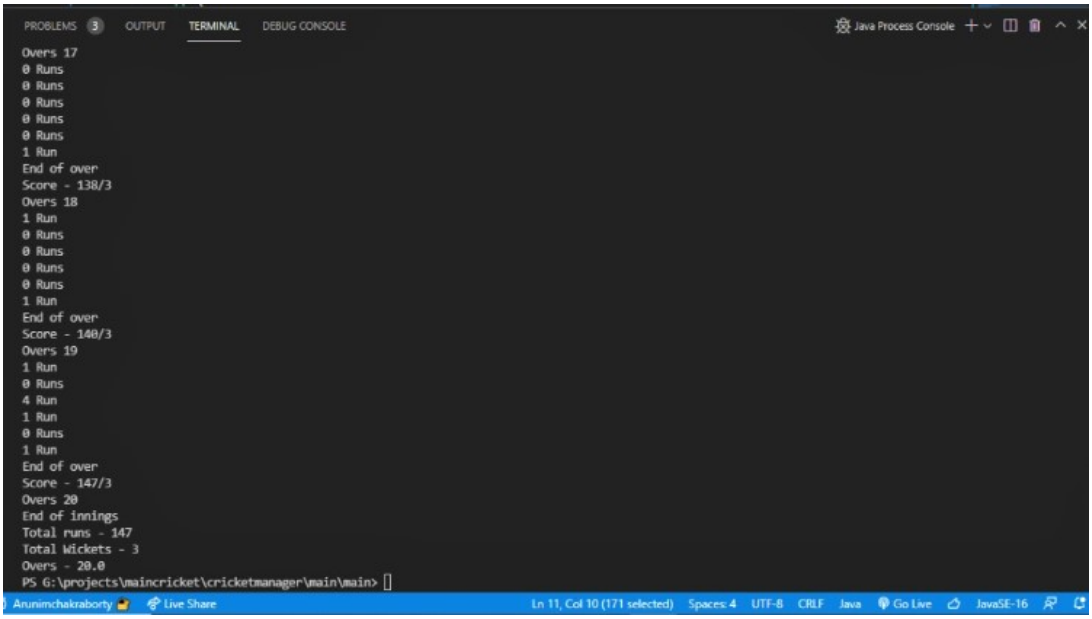


FIGURE 13 – Outcomes Shown on the terminal. Along with the results and everything shown on the GUI, a more in-depth analysis and result is shown in the terminal.

```
void setConn() {
    try {
        con = DriverManager.getConnection(url, username, password);
        stmt = con.createStatement();
    } catch (SQLException e) {
        System.out.println(e);
        e.printStackTrace();
    }
}

1 usage
void setAllSkillsFromDB() throws Exception {
    String selectFromPlayerTeamQ = "SELECT * FROM batting_skills b JOIN player_info p ON p.id = b.id WHERE p.cou
    String selectFromPlayerTeamQBowl = "SELECT * FROM bowling_skills b JOIN player_info p ON p.id = b.id WHERE p
    String selectFromOppTeamQ = "SELECT * FROM batting_skills b JOIN player_info p ON p.id = b.id WHERE p.countr
    String selectFromOppTeamQBowl = "SELECT * FROM bowling_skills b JOIN player_info p ON p.id = b.id WHERE p.co

    ResultSet rs1 = stmt.executeQuery(selectFromPlayerTeamQ);
    rs1.next();
    for (int i = 0; i < 11; i++) {
        frontFootSkill[i] = rs1.getInt( columnLabel: "front_foot");
        backFootSkill[i] = rs1.getInt( columnLabel: "back_foot");
        offSideSkill[i] = rs1.getInt( columnLabel: "off_side");
        onSideSkill[i] = rs1.getInt( columnLabel: "on_side");
        strokePlay[i] = rs1.getInt( columnLabel: "stroke_play");
        AgainstSpin[i] = rs1.getInt( columnLabel: "against_spin");
        AgainstPace[i] = rs1.getInt( columnLabel: "against_pace");
        rs1.next();
```

FIGURE 14 – setting connection with database. We have created a database in mysql containing all the in-game player data. We extract that data from the database and store them in arrays.

```
private void addPanel(JPanel p, int x, int y, int width, int height) {
    p.setBounds(x, y, width, height);
    p.setForeground(new Color( r: 34,  g: 139,  b: 34));
    p.setBackground(Color.WHITE);
}

8 usages
private void setPanelLayout(JPanel p, LayoutManager mgr) { p.setLayout(mgr); }

2 usages
private void setPanelBorder(JPanel p, String name) {
    p.setBorder(new TitledBorder(new LineBorder(new Color( r: 173,  g: 216,  b: 230),  thickness: 2),  title: name
            + "", TitledBorder.LEADING, TitledBorder.TOP,  titleFont: null, new Color( r: 173,  g: 216,  b: 230)));
}

2 usages
private void AddPanelToUI(JPanel p) { add(p); }

2 usages
private void bowlButtonFunction(String text) {
    bowlButton = new JButton(text);
    bowlButton.setBackground(Color.BLACK);
    bowlButton.setForeground(Color.WHITE);
    bowlButton.setFont(new Font( name: "Elephant", Font.BOLD,  size: 18));
}

private void AddLabel(JLabel l, JPanel p, String label) {
    l = new JLabel(label);
    l.setForeground(Color.DARK_GRAY);
```

FIGURE 15 – GUI code snippet. We have created the GUI in java swings and have used elements like jpanel, jlabel, jbutton, actionListener, etc.

```
private void addingALtogether() {
    ALObject = new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {

            if (userBatting == false & var1 == 0) {
                setVisible(false);
                new editedUI_5(target);
            }
            if (wickets == 10) {

                if(whichInnings==1){
                    if(team_runs>target){
                        new endScreen( msg: "won");
                    }
                    else if(team_runs==target){
                        new endScreen( msg: "drew");
                    }
                    else{
                        new endScreen( msg: "lost");
                    }
                }
                innings();
                chooseBowlerChoice.removeAll();
                choiceAddSecondInnings();
            }

            else if(overs==19 & over_balls==5){
                if(whichInnings==1){
```

FIGURE 16 – Action Listener in GUI. Whenever the bowl button is clicked the backend constructor is called along with various changes in the GUI part.

```
public String[] playerName = new String[22];

12 usages
public int[] batsman_runs = new int[22];
12 usages
public int[] batsman_balls_played = new int[22];
4 usages
public int[] frontFootSkill = new int[22];
4 usages
public int[] backFootSkill = new int[22];
4 usages
public int[] offSideSkill = new int[22];
4 usages
public int[] onSideSkill = new int[22];
3 usages
public int[] strokePlay = new int[22];
3 usages
public int[] AgainstPace = new int[22];
3 usages
public int[] AgainstSpin = new int[22];

3 usages
public int[] spin = new int[22];
6 usages
public int[] seam = new int[22];
3 usages
public int[] drift = new int[22];
6 usages
public int[] swing = new int[22];
7 usages
public int[] accuracy = new int[22];
6 usages
public int[] variations = new int[22];
13 usages
public int[] bowlerCategory = new int[22];
```

FIGURE 17 – Variables used. We have declared all variables in an interface and then later inherit it in all the respective classes.

```
int categoryMod = 0;
for (int i = 0; i < bowler_points.length; i++) {

    int j = i + startIndexForBowler;

    if (overs < 6) {
        if (bowlerCategory[j] == 1) {
            bowler_points[i] -= (overs * 8);
        } else if (bowlerCategory[j] == 0) {
            bowler_points[i] += overs * 8;
        }
    } else if (overs > 5 & overs < 15) {
        if (bowlerCategory[j] == 1) {
            bowler_points[i] += (overs - 6) * 8;
        } else if (bowlerCategory[j] == 0) {
            bowler_points[i] -= (overs - 6) * 8;
        }
    } else if (overs >= 15) {
        if (bowlerCategory[j] == 1) {
            bowler_points[i] -= (overs - 12.5) * 8;
        } else if (bowlerCategory[j] == 0) {
            bowler_points[i] += (overs - 12.5) * 8;
        }
    }

    if (bowlerCategory[j] == 1) {
        categoryMod = 1;
    } else if (bowlerCategory[j] == 0) {
        categoryMod = -1;
    }

    bowler_points[i] -= (categoryMod * (pitch - 2)) * 6;

    bowler_points[i] -= bowler_wickets[j] * 55;
```

FIGURE 18 – AI algorithm for selecting bowlers We have used knapsack algorithm for this. A ratio of profit and weight is allotted to every bowler. Profit is bowler skill while weight is influenced by various factors like number of overs bowled, number of wickets taken, runs conceded, also the number overs left in the innings.

```
public void aIAlgorithmBackend() {

    float runRate = (float) team_runs / ((float) overs + (float) over_balls * 100 / 6);

    if (userBatting) {

        line = new Random().nextInt( bound: 5) % 4 + 1;
        length = new Random().nextInt( bound: 5) % 4 + 1;



        if (overs < 6) {
            fieldAggression = 4;
        } else if (overs >= 6 & overs < 15) {
            fieldAggression = 2;
        } else if (overs >= 15) {
            fieldAggression = 1;
        }
    } else {

        if (overs < 6) {
            battingAggression = 3;

            if (runRate < 7.5) {
                battingAggression = 4;
            }

            if (wickets <= 2 & wickets != 0) {
                battingAggression = 2;
            }

            if (wickets > 2) {
                battingAggression = 1;
            }
```

FIGURE 19 –AI algorithm for field aggression, line, length and batsman aggression. Basically an if/else statements that lists all the conditions and scenarios and suggests the changes that the AI should implement.

```
float[] findMaxElementInArray(float[] array){
    int pos = 0;
    float max = array[0];
    for (int i = 0; i < array.length; i++) {
        if(array[i]>max){
            max = array[i];
            pos = i;
        }
    }
    return new float[]{pos,max};
}

6 usages
static int findSecondHighest(float[] arr){
    float max = arr[0];
    int pos1 = 0;
    for (int i = 0; i < arr.length; i++) {
        if(arr[i]>max){
            max=arr[i];
            pos1=i;
        }
    }

    int pos2 = 0;
    float[] arr2 = new float[arr.length-1];
    for (int i = 0; i < arr2.length; i++) {
        if(arr[i]==max){
            continue;
        }
        arr2[i]=arr[i];
    }

    float max2 = arr2[0];
    for (int i = 0; i < arr2.length; i++) {
```

FIGURE 20 – Miscellaneous functions used. We have used functions like finding highest element in an array, second highest array, etc.

Other Small Nuances

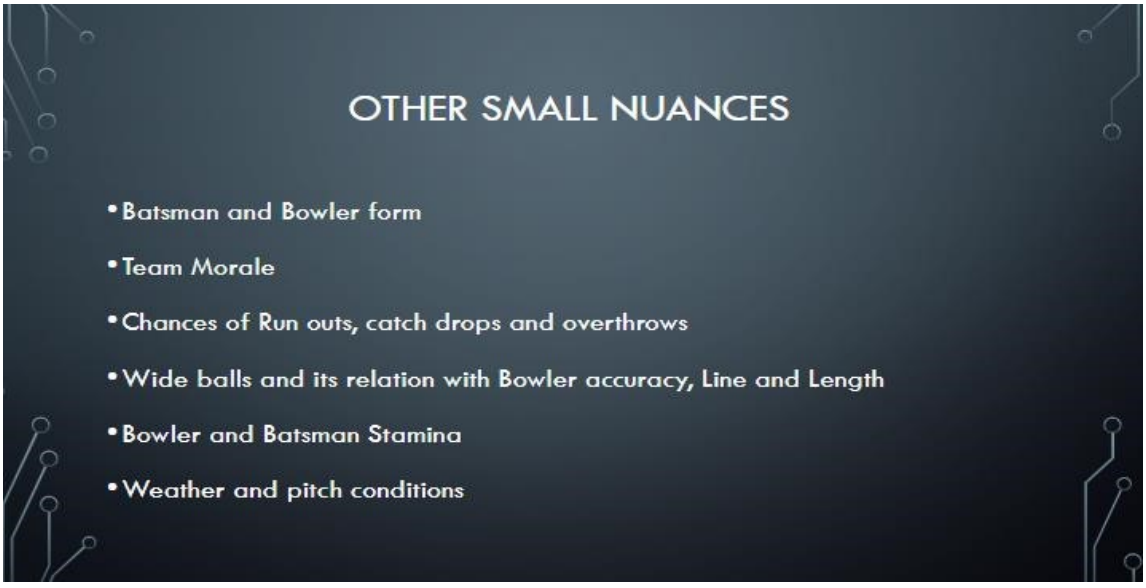
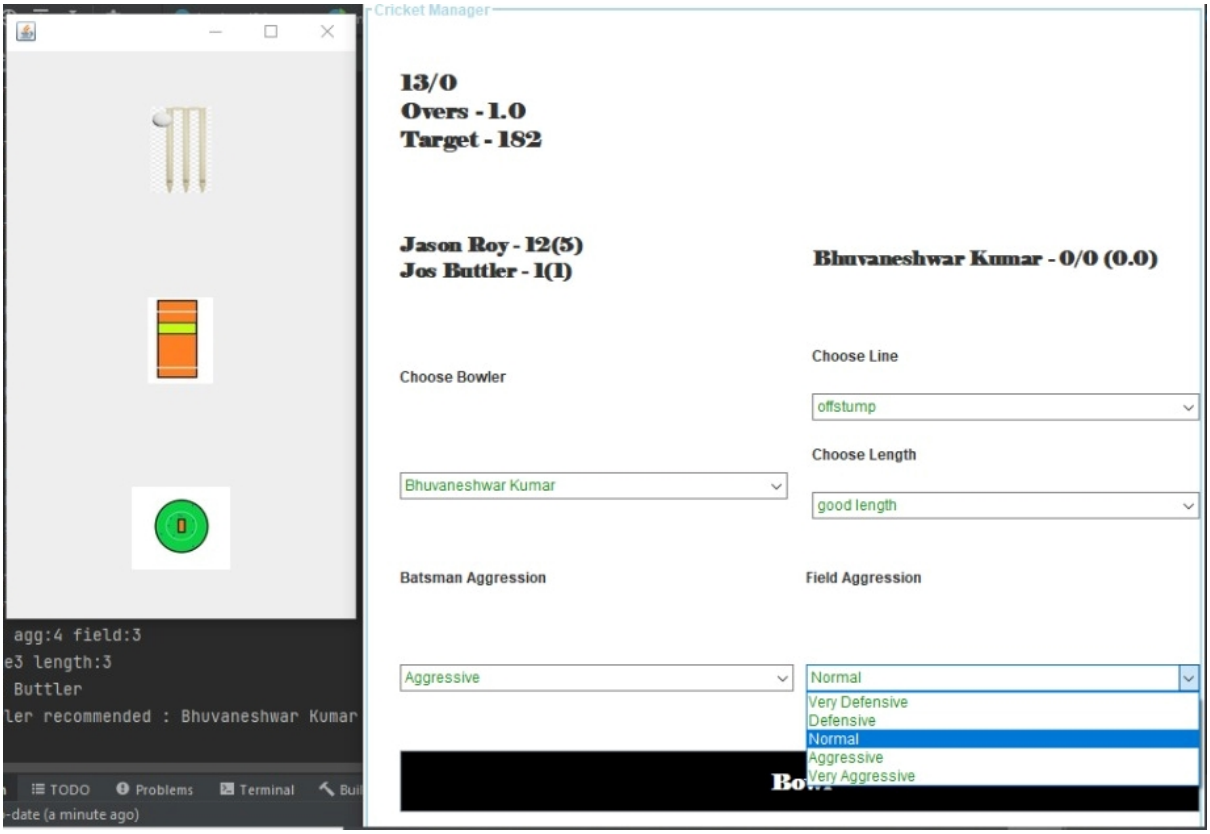


FIGURE 21 – Small Nuances. Other subtle nuances are included that enhance the realism of the game.

4.Final Project Output

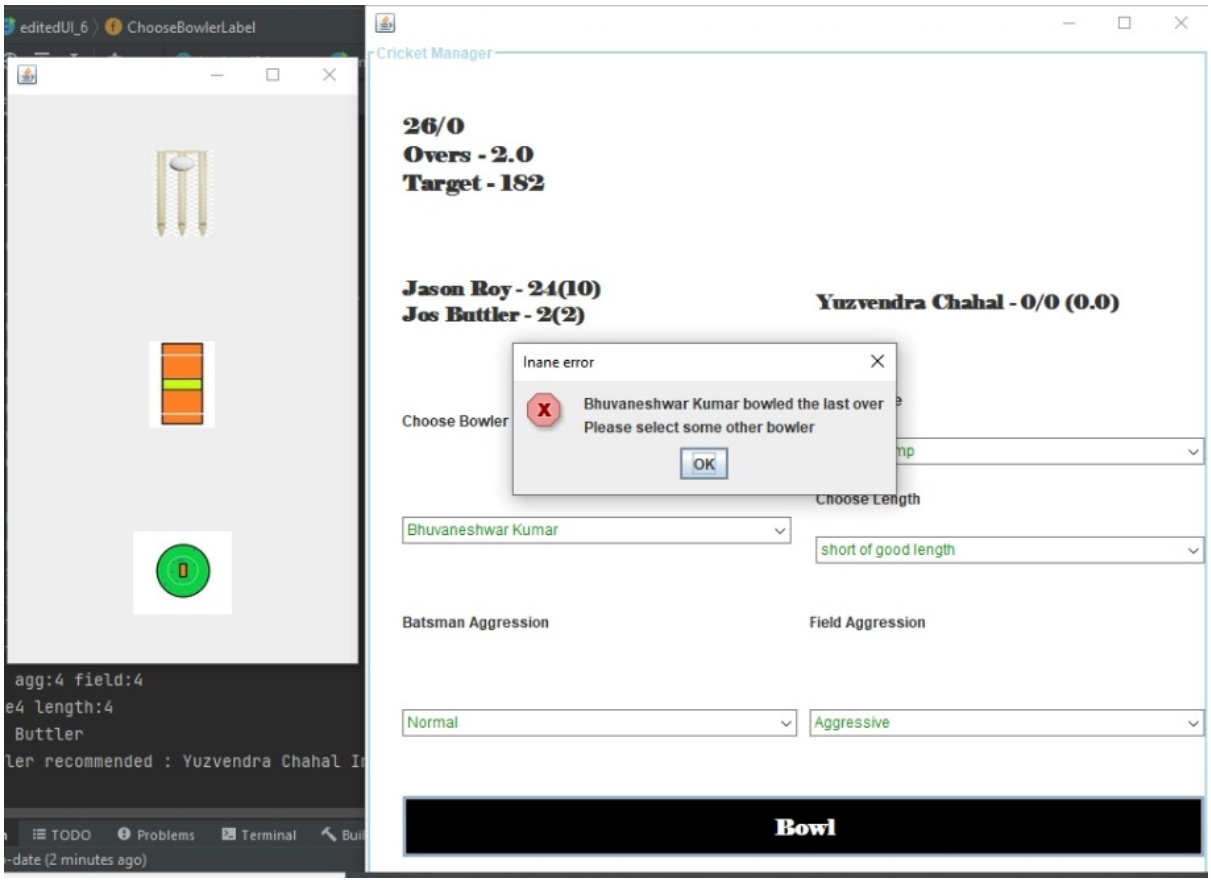
[i].Sample-1



Batsman View Interface

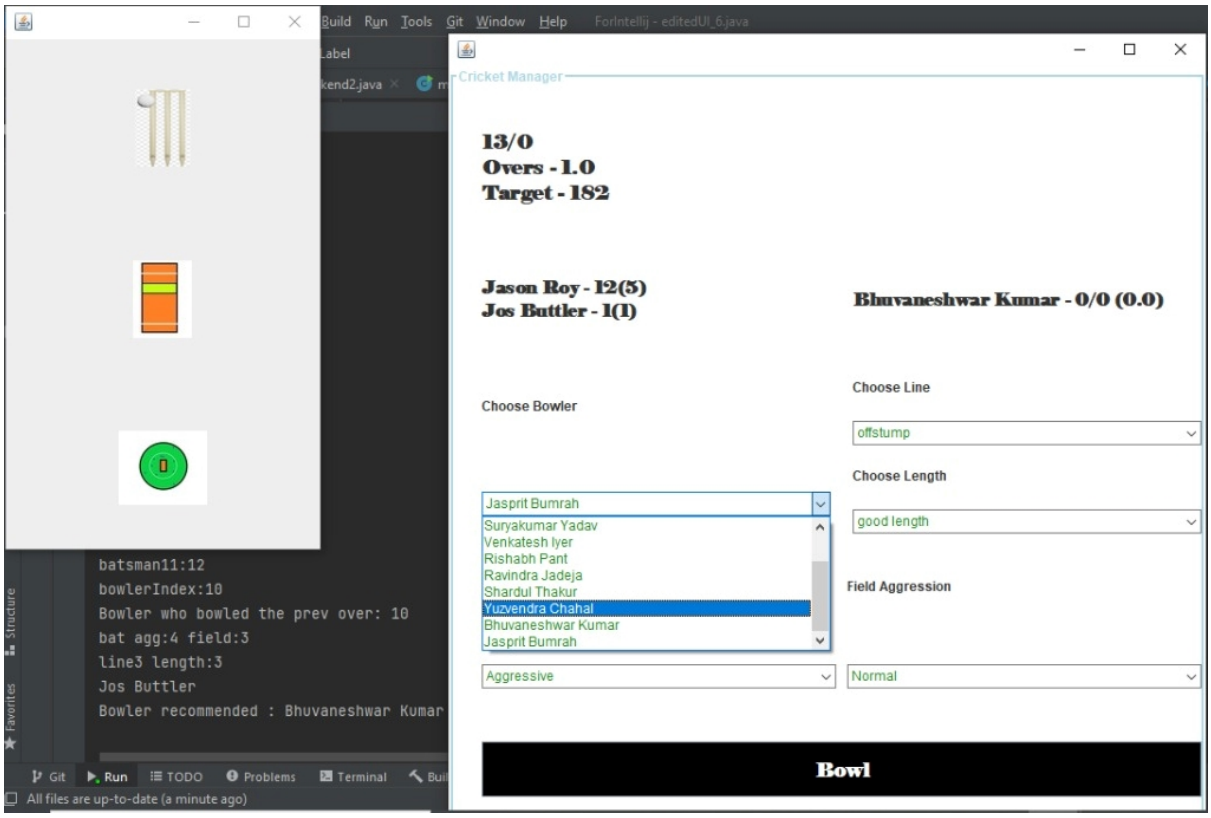
[ii]. Sample-2

Start of Match



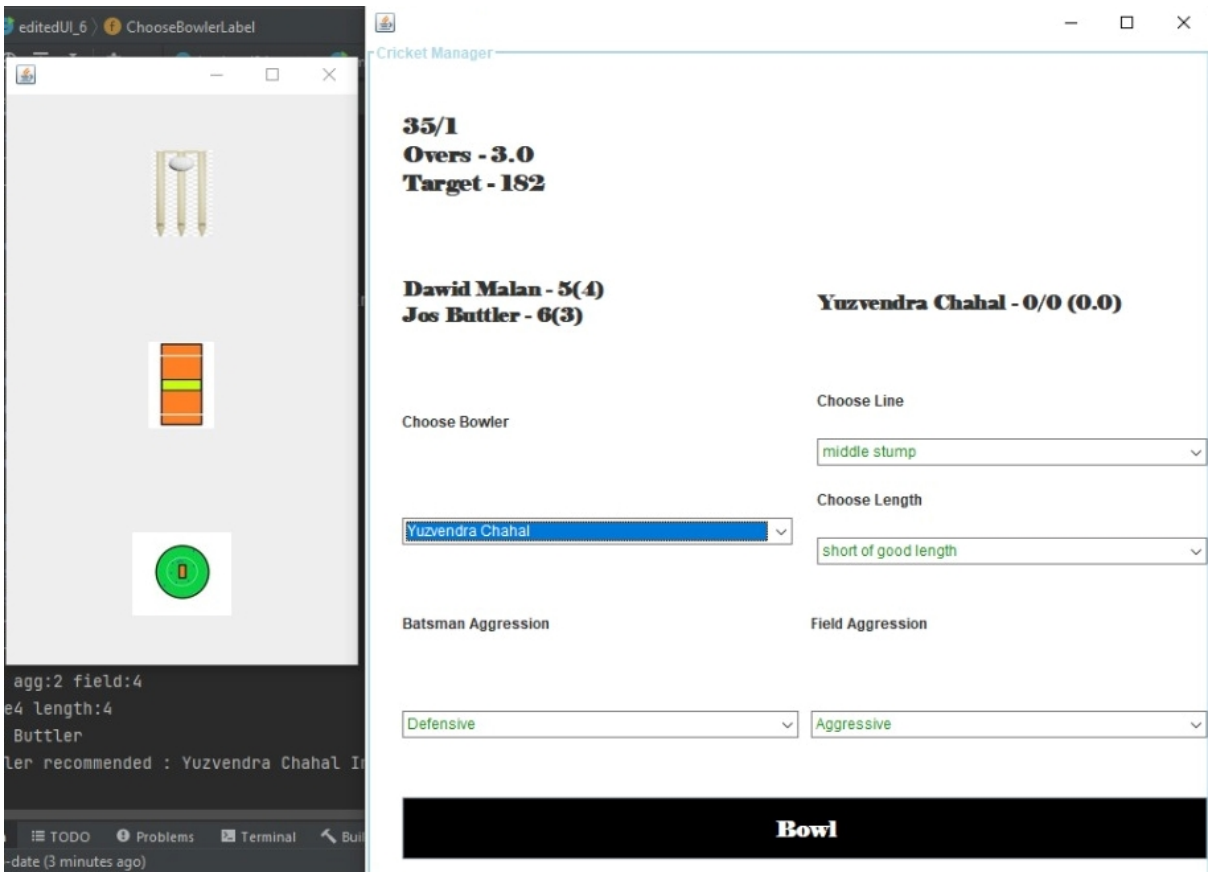
[iii]. Sample-3

User choosing a Bowler for playing the match



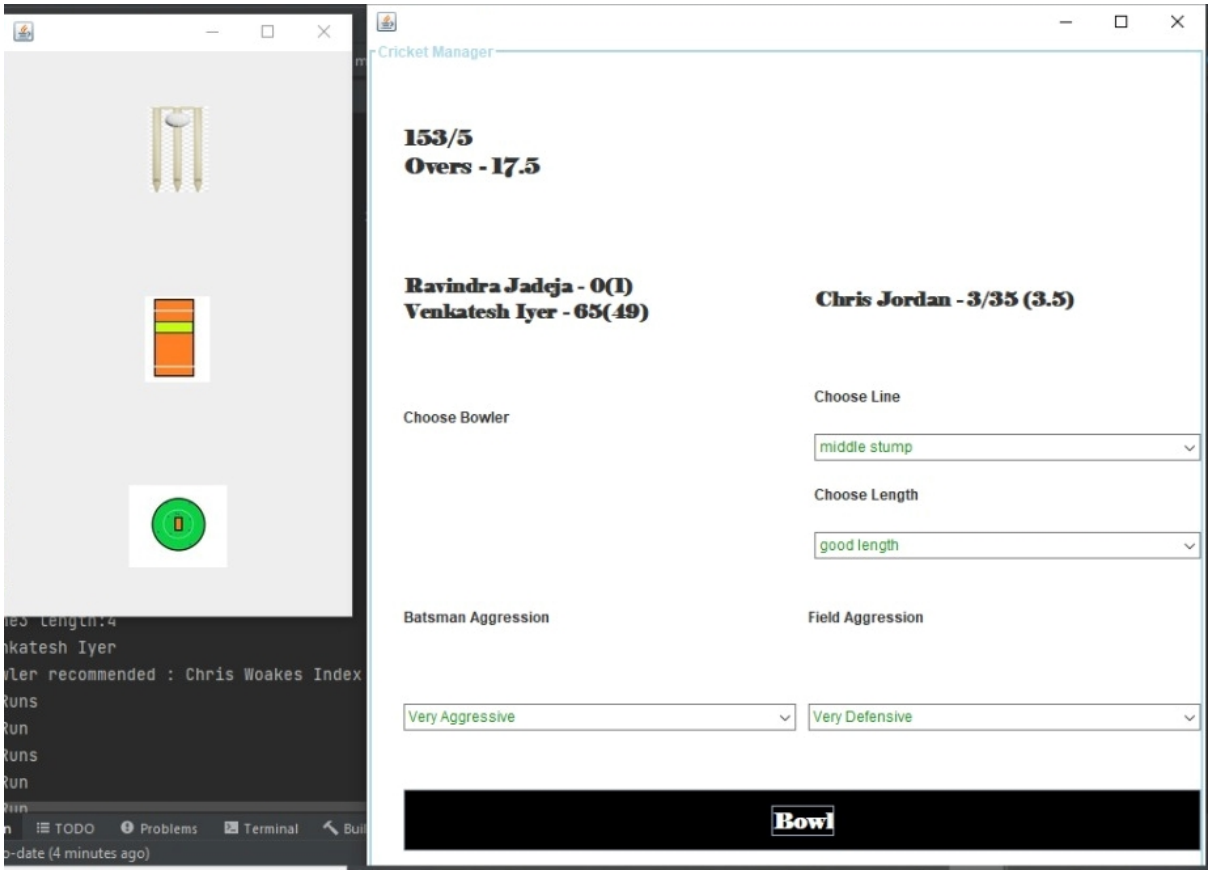
[iv]. Sample-4

Wicket taken by the user while bowling



[v]. Sample-5

A Bowlers View(Indian Players in a match)

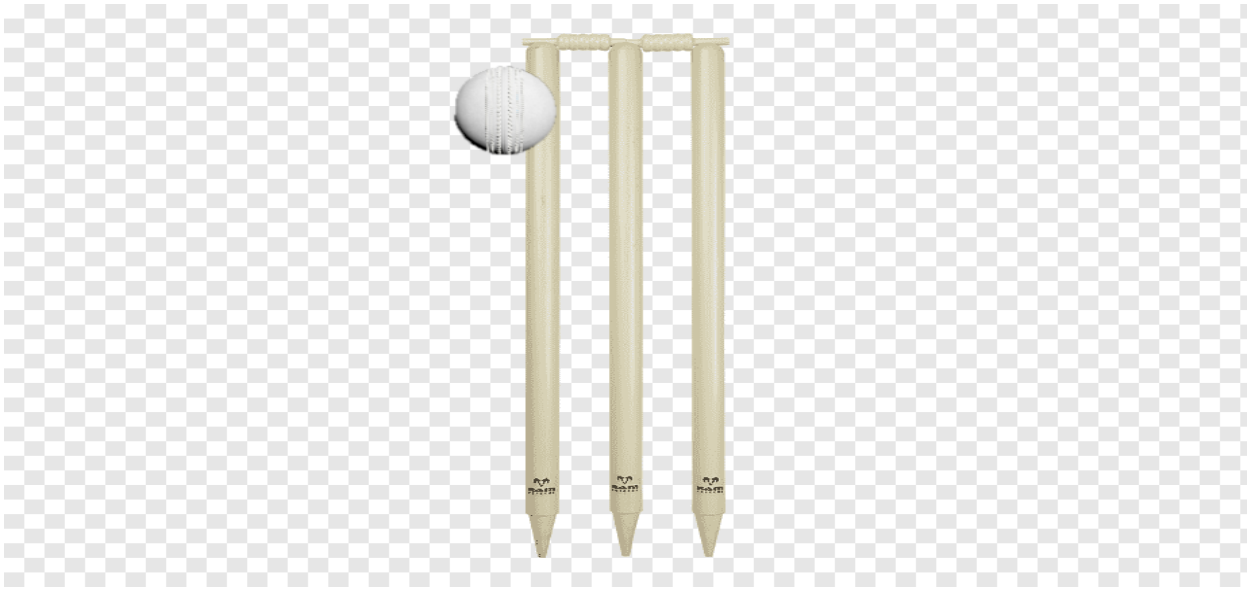


5.LINE VARIATION IN THE MATCH

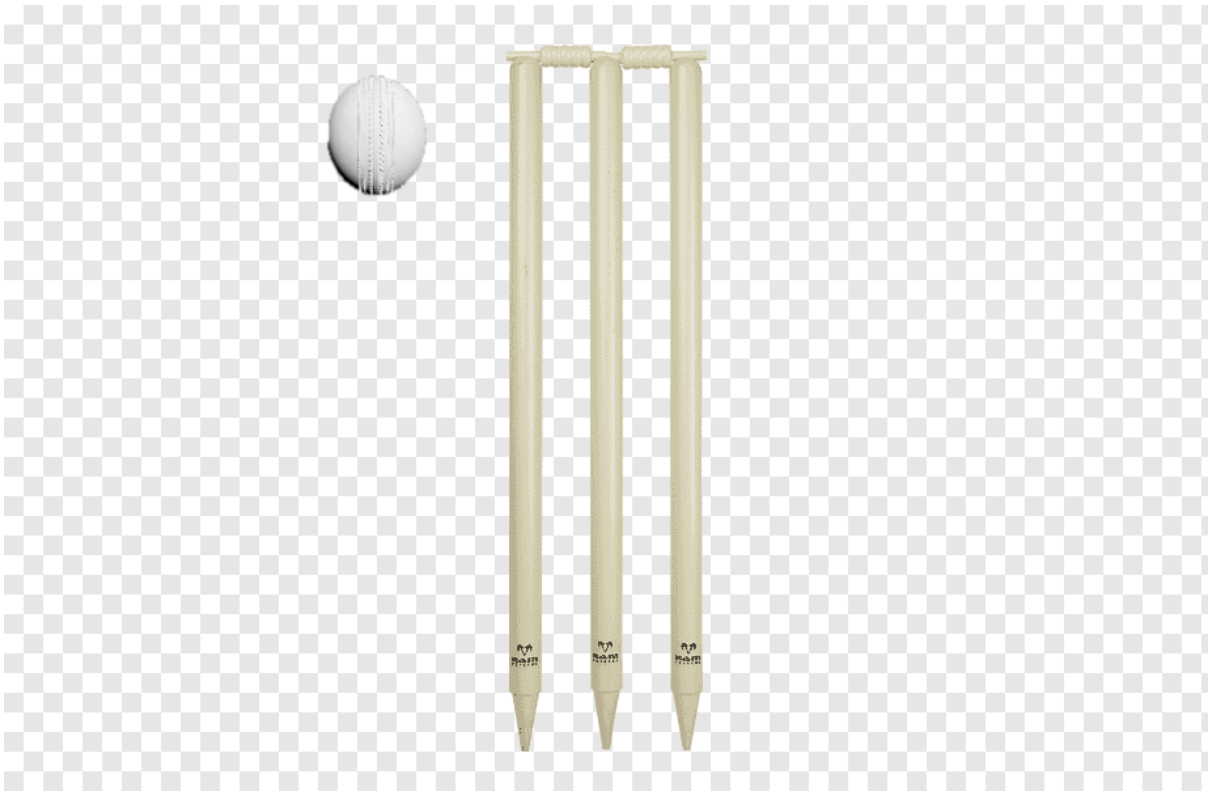
1. LEG STUMP

2.MIDDLE STUMP

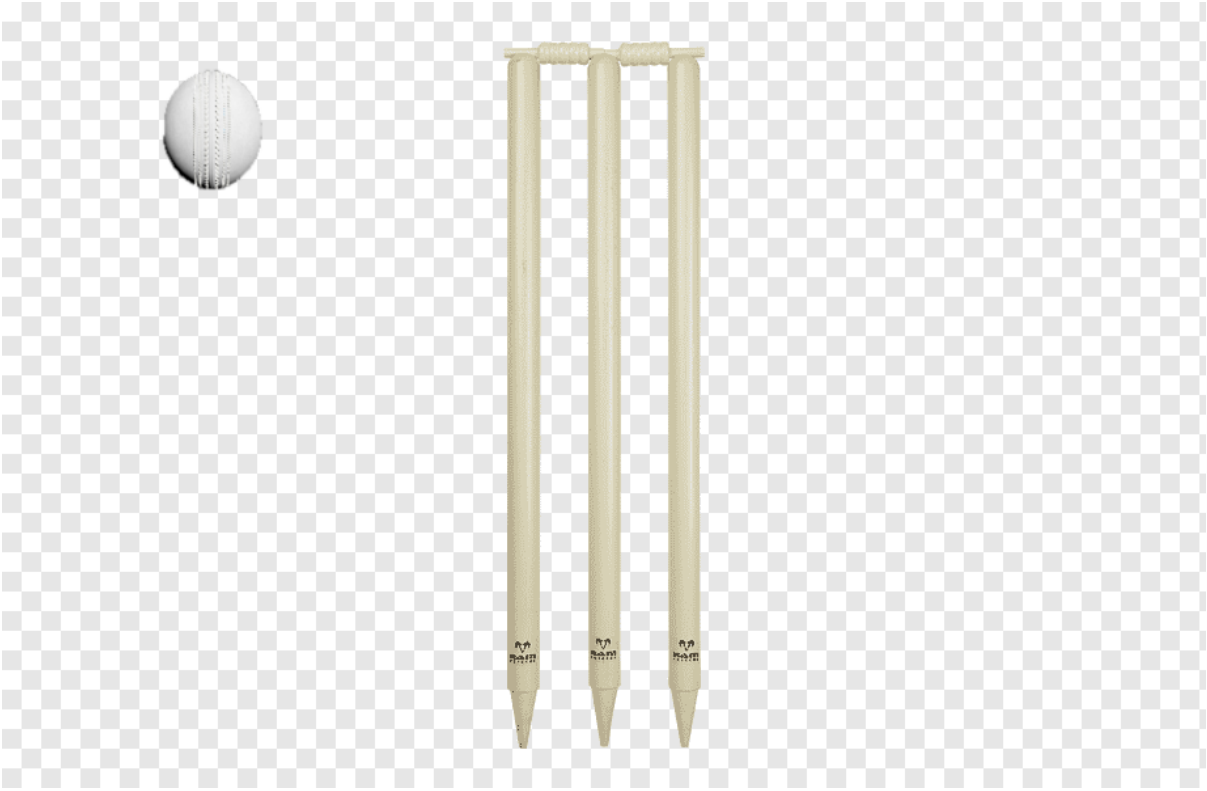
3.OFF STUMP



4.OFF STUMP CHANNEL

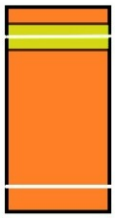


5.WIDE OUTSIDE OFF

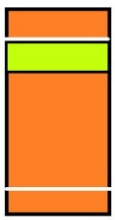


6.LENGTH VARIATION IN THE MATCH

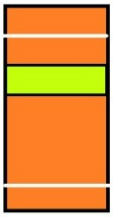
Yorker



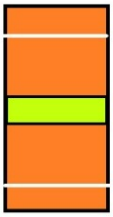
Full Length



Good Length

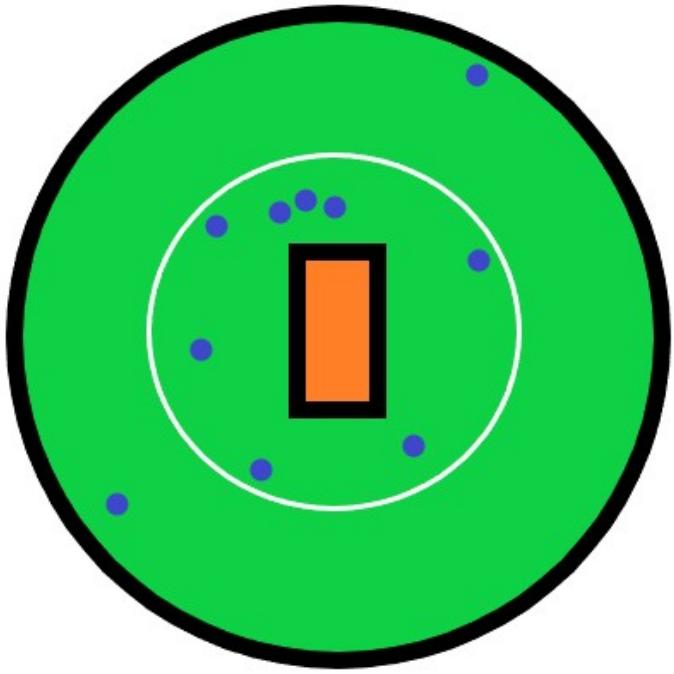


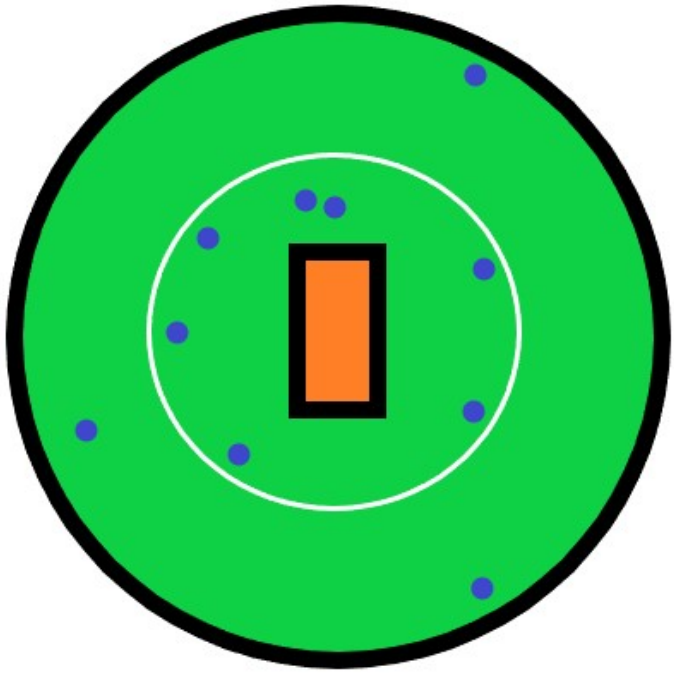
Short of Good Length

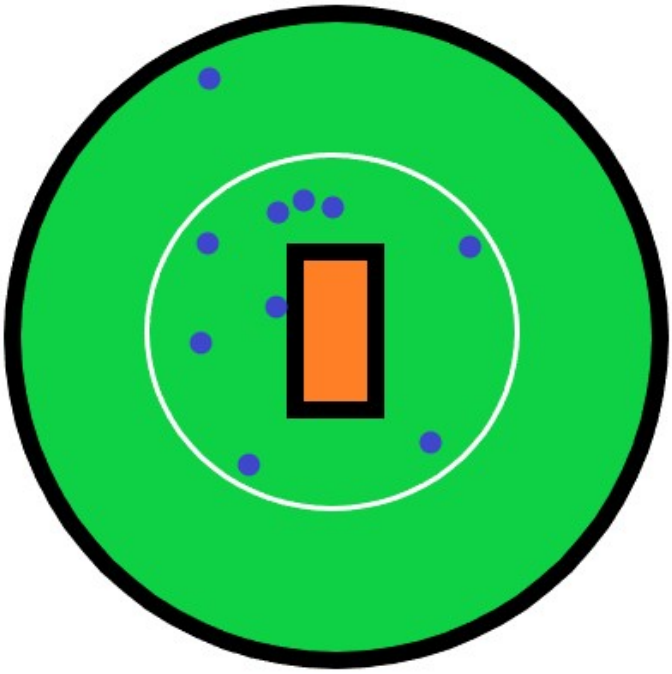


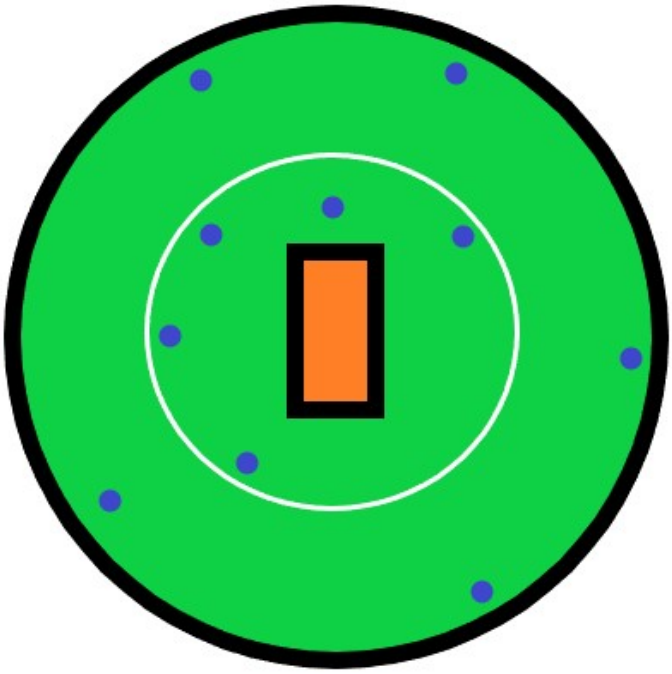
Bouncer

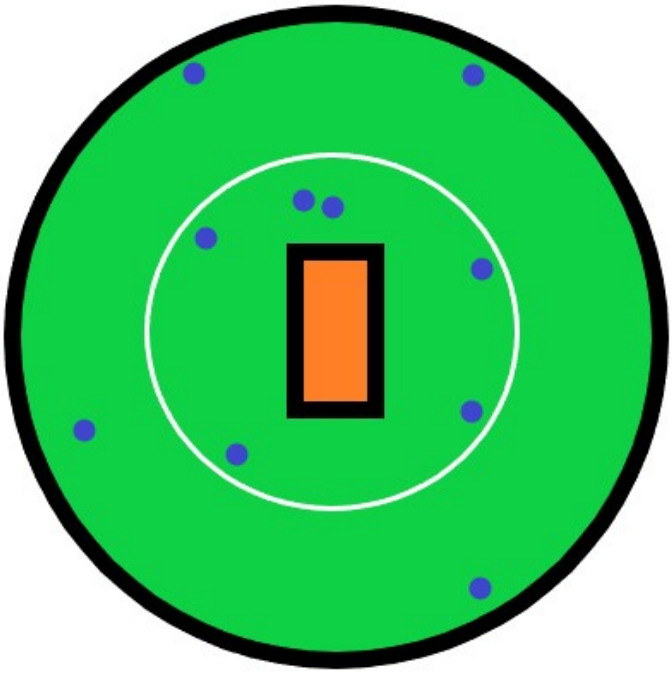
6.LENGTH VARIATION IN THE MATCH











Conclusion

In this project we shall create a cricket manager game which implements AI and makes uses intricate backend code to make it intricate.

Reference Links –

Kaggle Datasets -

https://www.kaggle.com/nowke9/ipldata

Football Manager -

https://www.footballmanager.com/