

Name: Manya Smriti  
Registration No. : 18BIT0127

## OOAD PROJECT

### Algorithm

#### Self made Rank Predictor Algorithm in the language of OOAD

//objects of classes involved in the interaction are highlighted

#### How it works??

Firstly the **rank\_predictor** asks for the analysed difficulty level of the paper from **admin**. It is analysed by various experts after the exam is conducted. The paper is scaled from 1 to 10 depending on the difficulty level. Evidences from past suggests that Resonance and Fiitjee Jee Advance paper analyser has been very successful in rating the difficulty level of the paper. This home made predictor needs to be fed in with the expert rating.

The **admin** enters no. of participants appeared for the exam in **rank\_predictor**. While entering the number of participant, one should be assure of entering the no. of participant who filled form for the exam and not the number of JEE main qualified student for greater accuracy. In year 2018, 65000 of qualified student did not appear for JEE advance. This number can highly affect the expected rank.

Then **rank\_predictor** asks for expected marks from **Student**. He may check his answer with some reliable answer key like that of Resonance or Fiitjee. The keys are released by evening or so by these institute or any other reliable source can be considered( just check the past analysis of the source)

**Rank predictor** takes marks as input from the **Student**. It gives 7 choices:

- 1) Common Rank Prediction - To get All India Rank corresponding to the entered mark.
- 2) OBC Rank Prediction - To get OBC Rank corresponding to the entered mark.
- 3) SC Rank Prediction - To get SC Rank corresponding to the entered mark.
- 4) ST Rank Prediction - To get ST Rank corresponding to the entered mark.
- 5) PWD Rank Prediction - To get PWD Rank corresponding to the entered mark.

6) Search College - the user can search for IITs one may get corresponding to the mark.

if the **Student** presses 1, he/she gets details about the rank secured on that mark in last 2 years. Alongwith this a rank range under which his rank may most probably fall is shown on **rank\_predictor**.

The next line displays the most expected rank in the range.

The same output pattern is displayed for each category ranking.

On pressing 6, the **Student** needs to enter the category corresponding to which they would like to know about the colleges they may get in the **rank\_predictor**. Under this choice the student is given detail about their ranks and college one may get.

### **Now comes the path that walks you to land of answering your dilemma-The mechanism involved behind**

The difficulty level of past 2 years is used in **Algorithm**. It is bi-parametric namely:

- i) The level of past year question on a scale 1-10.
- ii) The no. of candidates appearing for the exam that year

Difficulty level calculation in **Algorithm**(class) also uses the minimum of participants appearing amongst the data of previous two years. Let's call it min\_participants.

**Difficulty level is calculated by the formula:**

**$(\text{paper\_level}) * (\text{no\_of\_candidates\_that\_year} / \text{min\_participants})$**

### **Now the question comes..why so??**

In case of paper\_level, for say in 2017 the level is 5.3 and in 2018 it being 7.23. The difficulty level is easily visible on a scale of 1-10.

But in case of 245000 participants in 2019 and 165000 in 2018 the comparison needs to be in some ratio to estimate the overall difficulty level.

### **Now why the minimum and not maximum or average??**

The following are factors:

- i) With increasing years the craze of the exam is increasing. Students nowadays are enrolled in 7th class itself to prepare themselves for jee.

ii) Even though the parameters of rating a question can be same, but increase in participants with years includes ease in access to resources like youtube, online visual classes and other technology which makes the exam even more competitive.

So the above 2 points suggests that even if the paper\_level and no\_of\_candidates appearing in consecutive years be the same but still there is something more to deciding difficulty\_level. Taking minimum of the both the years increments current difficulty\_level.

Difficulty level of previous 2 years is calculated.

There can be **2 cases** here:

- i) Difficulty level being poles apart for example varying by 2-3
- ii) The difficulty level can be very close for example varying by 0.5-1

The code has 2 difficulty level :  
difficulty\_level1 and difficulty\_level2

**In the 1st case difficulty\_level1** is assigned the value of higher difficulty\_level amongst the 2 calculated ones while difficulty\_level2 is assigned the value of lower difficulty\_level. (Because these are 2 variety of database available for prediction which can be considered for prediction).

**In the 2nd case** difficulty\_level1 is assigned the value of lower difficulty\_level. While the difficulty\_level2 is assigned the value of difficulty\_level varying by 1.5-2 in last 5 years or in case of unavailability lowest difficulty\_level calculated in last 5 years can be taken.

Now if the current\_difficulty\_level is greater than or equal to difficulty\_level1, the rank prediction is done considering the rank vs marks database of the year whose difficulty level value is assigned to difficulty\_level1.

If not, the rank vs marks database of the year whose value is assigned to difficulty\_level2 is considered for the rank prediction.

The code written ensures that difficulty level allotted is in such a manner that difficulty\_level1 is higher than difficulty\_level2.

On marks entry, the rank corresponding to it is fetched. Rank predictor displays the predicted rank range. It first of all counts the no. of digits the rank contains.

If the marks entered in the **rank\_predictor** is higher than or equal to rank 1 of the **RankDb** (database) considered then the rank 1 is predicted.

#### Code Snippet

```
if(rank18==0 | rank18==1)
{
    min=1;
    max=1;
}
```

for example: if the rank fetched is less than 10. It would display a rank range 8-12.

#### Code Snippet

```
else if(rank18>10 && rank18<20)
{
    min=min-7;
    max=max+7;
}
```

Now if the rank ranges between 20-30. Rank is obtained by multiplying the rank fetched with current\_difficultylevel which is further multiplied by .1 and subtracted by 10 to get a minimum range and added by 10 to get the maximum range.

#### Logic behind:

If the difficulty level calculated is more than 10 (for example paper\_level being 8.3 and candidate ratio be like 1.3 would give difficulty level greater than 10) then the range should shift rightward. If the rank last year be 25 and let's say thus the range be something 14-30 then this year the predicted rank range would be something like 20-35.

In a similar way if the difficulty level calculated is less than 10, the predicted range would shift left.

#### Code Snippet

```
if(rank18>19 && rank18<31)
{
    min=min*(difficulty_level*.1)-10;
```

```
        max=max*(difficulty_level*.1)+10;
    }
```

In a similar way for range 31-200

```
    else if(rank18<201)
    {
        min=min*(difficulty_level*.1)-2*pow(10,1);
        max=max*(difficulty_level*.1)+2*pow(10,1);
    }
```

For range 201-500

```
    else if(rank18<501)
    {
        min=min*(difficulty_level*.1)-100;
        max=max*(difficulty_level*.1)+100;
    }
```

For range 501-1000

```
    else if(rank18<1001)
    {
        min=min*(difficulty_level*.1)-300;
        max=max*(difficulty_level*.1)+300;
    }
```

For range 1000-5000

k is used here for no. of digits-1. (for example 1006 have 4 digits, k=3)

```
    else if(rank18<5000)
    {
        min=min*(difficulty_level*.1)-3*pow(10,k);
        max=max*(difficulty_level*.1)+3*pow(10,k);
    }
```

For other lower rank:

```
    else
    {
        min=min*(difficulty_level*.1)-6*pow(10,k);
        max=max*(difficulty_level*.1)+6*pow(10,k);
    }
```

```
}
```

Similarly code for some higher ranks also follow norms like:

```
else if(rank18==2)
{
    min=1;
    max=3;
}
else if(rank18>3 && rank18<10)
{
    min=min-2;
    max=max+2;
}
```

The range for the marks for which rank is to be predicted varies from the value assigned to min to that of max is provided to the object of Student class.

If the, **Student enters choice 6**, he opts for searching college corresponding to entered marks. The **Rank\_Predictor** asks for the category from the student. The rank\_predictor fetches rank from **Rankdb** and apply the same **Algorithm** to calculate rank range. It displays rank range for marks. The most expected rank is taken to be average of min and max marks from the range and sent to Algorithm for fetching IITs considering the same **CollegeList** (college db) whose difficulty\_level is chosen for deciding the rank. The rank\_predictor displays IITs the Student may get.

To ensure that the Student need not always log out and log in to check multiple marks, rank\_predictor can be used for multiple times i.e it asks Student if he choose to reset the marks by entering 1. If reset is chosen, the entire mechanism mentioned above is repeated.

If Student finds that rank\_predictor is not working, Student can report the malfunction by entering 100 to report. The **admin** is notified with error message in case of report.  
Student can exit by pressing 0.

## CODE

### CODE 1:

### Interaction with Rank\_Predictor & Student

```
#include<iostream>
using namespace std;
class Student
{
public:
int marks;
int choice;
int callStudent()
{
    cin>>marks;
    return marks;
}
int choiceStudent()
{
    cout<<"Enter
1-Common_rank,2-OBC_rank,3-SC_rank,4-ST_rank,5-PWD_rank,6-Reset
marks,7-Search_college\n";
    cin>>choice;
    return choice;
}
};

class Rank_Predictor
{
public:
    void Predictor_output(string a)
    {
        cout<<"You choose to view "<< a <<" rank   for entered marks ";
    }
    void Reset()
    {
        cout<<"You choose to reset marks ";}
    void search()
    {
        cout<<"You wish to search colleges";
    }
    void callPredictor(int mark,int choice)
    {
        if(mark<0 || mark>360)
        cout<<"Invalid marks\n";
```

```

switch(choice)
{
case 1:Predictor_output("Common");
        break;
case 2:Predictor_output("OBC");
        break;
case 3:Predictor_output("SC");
        break;
case 4:Predictor_output("ST");
        break;
case 5:Predictor_output("PWD");
        break;
case 6:Reset();
        break;
case 7:search();
        break;
default:
        cout<<"Invalid Choice";
}
}
};

int main()
{
cout<<"Enter marks\n";
Student s;
Rank_Predictor p;
int mark=s.callStudent();
int choice=s.choiceStudent();
p.callPredictor(mark,choice);
}

```

## **CODE 2:**

### **Interaction with Rank\_Predictor & Algorithm for choice**

```

#include<iostream>
#include<cstdlib>
#include<time.h>
using namespace std;
class Rank_Predictor

```



```

{
    public:
    int marks,max,choice;
    void predict(string a,int c)
    {
        cout<<"Predicting "<<a <<" for marks "<<c<<"\n";
        cout<<"Marks sent to algorithm\n";
    }
    public:
    string callPredictor()
    {
        //fetched data
        int mark=298;
        srand(time(0));
        choice=rand()%7;
        switch(choice)
        {
            case 1:{
                predict("common_rank",mark);
                return "common rank";
                break;
            }
            case 2: {
                predict("OBC_rank",mark);
                return "OBC rank";
                break;
            }
            case 3: {
                predict("SC_rank",mark);
                return "SC rank";
                break;
            }
            case 4: {
                predict("ST_rank",mark);
                return "ST rank";
                break;
            }
            case 5: {
                predict("PWD_rank",mark);
                return "PWD rank";
                break;
            }
            case 6:{

```

```

        cout<<"Rank_Predictor sends marks to Algorithm\n";
        cout<<"Algorithm fetches last year collegeDb and sends to
rankPredictor\n";
        return "college list for rank";
        break;
    }
    default:cout<<"Rank_Predictor found some internal Error..needs
maintainance";
    }
    }
    int mark()
    {
        return marks;
    }
};
class Algorithm
{
    public:
    void callAlgorithm(string predictor_choice,int predictor_mark)
    {
        string a=predictor_choice;
        srand(time(0));
        int c=predictor_mark,e;
        e=rand()%50;
        cout<<"Algorithm sends "<<a<<" "<<e<<" to rank_predictor\n";
    }
};
int main()
{
    Rank_Predictor r;
    Algorithm a;
    string predictor_choice=r.callPredictor();
    int predictor_marks=r.mark();
    a.callAlgorithm(predictor_choice,predictor_marks);
}

```

### **CODE 3:**

#### **Interaction with Rank\_Db & Algorithm for fetching rank**

```

#include<iostream>
#include<string.h>

```

```

#include<math.h>
using namespace std;
class Algorithm
{
    public:
    int marks;
    string choice;
    int call1Algorithm()
    {
        marks=135;//fetched marks from rank Predictor
        cout<<"The Algorithm is fetching rank for "<<marks<<" ";
        return marks;
    }
    string call2Algorithm()
    {
        choice="Common";//fetched choice from rank Predictor
        cout<<"for "<<choice<<" category... \n";
        return choice;
    }
    void callMethod(int rank18,int rank17)
    {
        int i,j,difficulty_level,min,max,no_digit18,no_digit17,temp,t;
        cout<<"Please enter the difficulty level according to paper analyser for
this year\n";
        cin>>difficulty_level;// if difficulty level increases min range shifts to
left side
        int difficulty_level1=6;//difficulty level for year 2018
        int difficulty_level2=5;//difficulty level for year 2017
        if(difficulty_level1<difficulty_level2)
        {
            t=difficulty_level1;
            difficulty_level1=difficulty_level2;
            difficulty_level2=t;
            t=max;
            max=min;
            min=t;
        }
        temp=rank18,no_digit18=0,no_digit17=0;
        while(temp>0)
        {
            temp=temp/10;
            no_digit18+=1;
        }
    }
}

```

```

temp=rank17;
while(temp>0)
{
    temp=temp/10;
    no_digit17+=1;
}
if(difficulty_level>=difficulty_level1)
{
    min=rank18;
    max=rank18;
    int k=no_digit18-2;
    if(k<0)
    k=0;
    if(rank18>3 && rank18<10)
    {
        min=min-2;
        max=max+2;
    }
    else if(rank18>10 && rank18<20)
    {
        min=min-7;
        max=max+7;
    }
    else if(rank18>19 && rank18<31)
    {
        min=min-10;
        max=max+10;
    }
    else if(rank18<100)
    {
        min=min-2*pow(10,k+1);
        max=max+2*pow(10,k+1);
    }
    else
    {
        min=min-3*pow(10,k);
        max=max+3*pow(10,k);
    }
}
else
//if(difficulty_level<difficulty_level2)
{
    min=rank17;

```

```

        max=rank17;
        int k=no_digit17-2;
        if(k<0)
            k=0;
        if(rank17>3 && rank17<10)
        {
            min=min-2;
            max=max+2;
        }
        else if(rank17>10 && rank17<20)
        {
            min=min-7;
            max=max+7;
        }
        else if(rank17>19 && rank17<31)
        {
            min=min-10;
            max=max+10;
        }
        else if(rank17<100)
        {
            min=min-2*pow(10,k+1);
            max=max+2*pow(10,k+1);
        }
        else
        {
            min=min-3*pow(10,k);
            max=max+3*pow(10,k);
        }
    }
    cout<<"Possible rank range for your marks is "<<min<<"-"<<max;
}
};
class RankDb
{
    public:
        void Rank(int mark,int year,string c)
        {
            cout<<"Rank on mark "<<mark<<" for year "<<year<<" for "<<c<<"
category is ";
        }
        int commonrank18(int mark)
        {

```

```

int q=0,i,marks=mark,r,rank,p=0;
int
Commonrank_db18[10000]={337,272,258,247,240,233,227,223,219,216,213,
210,207,204,202,200,198,196,195,193,191,190,188,

187,185,184,182,181,180,179,177,176,175,174,173,172,171,170,169,168,
167,165,164,164,163,162,161,161,160,159,158,158,

156,155,154,154,153,152,152,151,150,150,149,148,148,147,147,146,145,
145,144,144,143,143,142,142,141,140,140,139,139,

138,138,137,137,136,136,135,135,135,134,134,133,133,132,132,131,131,
131,130,130,129,129,129,128,128,127,127,127,126,

126,126,125,125,124,124,124,123,123,123,122,122,122,121,121,121,120,120,
120,119,119,119,118,118,118,117,117,117,116,

116,116,115,115,115,115,114,114,114,113,113,113,113,112,112,112,111,111,
111,110,110,110,110,109,109,109,108,108,108,

107,107,107,107,106,106,106,105,105,105,105,104,104,104,104,103,103,103,
102,102,102,102,101,101,101,101,101,100,100,

100,99,99,99,99,98,98,98,98,98,97,97,97,97,96,96,96,96,95,95,95,95,94,94,94,
94,94,93,93,93,93,92,92,92,92,91,91,91,

91,90,90,90,90,90,89,89,89,88,88,88,88,87,87,87,87,86,86,86,85,85,85,85,
84,84,84,84,84,83,83,83,82,82,82,82,81,80,80,

79,79,79,78,78,78,78,77,76,75,74,73,72,71,70,69,68,67,66,65,64,63,62,61,
60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,

45,44,43,42,41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,21,
20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,
3,2,1,0,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,-11};//Rank 1-337 marks,Rank
101-272 marks,Rank 201-258 marks,Rank 301-247 marks
/* In real the first index will give the 1st ranker's mark but due to
lack of DB here the interval taken btw 2 indices is 100*/
for(i=0;i<10000;i++)
{
    if(Commonrank_db18[i]==marks)
    {r=i;
    rank=r*100+1;

```

```

        return rank;
        break;}
    if(Commonrank_db18[i]<marks)
    {
        if(i==0)
            rank=i;
        else
        {
            r=i;
            rank=(r*100+1);
            p=Commonrank_db18[i-1]-Commonrank_db18[i];
            int q=marks-Commonrank_db18[i];
            rank=rank-(rank*q)/p;
        }
        return rank;
        break;
    }
}
}
int obcrank18(int mark)
{
    int q=0,i,marks=mark,rank,r;
    int
OBCrank_db18[10000]={319,213,198,188,179,173,167,163,159,155,152,149,1
46,143,141,138,136,134,132,130,129,127,126,124,

    123,121,120,118,117,116,115,113,112,111,110,109,108,107,106,105,104,
103,102,101,100,99,99,98,97,96,95,94,93,92,91,90,

    89,88,88,87,86,85,85,84,83,83,82,81,81,81,80,79,79,78,77,76,75,74,74,73,
72,72,72,71,70,69,68,67,66,65,64,63,62,61,60,

    59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,
35,34,33,32,31,30,29,28,27,26,25,24,23,21,20,

    19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-5,-6,-7,-8};
    for(i=0;i<10000;i++)
    {
        if(OBCrank_db18[i]==marks)
        {
            r=i;
            rank=r*100+1;
            return rank;

```

```

        break;}
    if(OBCrank_db18[i]<marks)
    {
        if(i==0)
            rank=i;
        else
        {
            r=i;
            rank=(r*100+1);
            int p=OBCrank_db18[i-1]-OBCrank_db18[i];
            int q=marks-OBCrank_db18[i];
            rank=rank-(rank*q)/p;
        }
        return rank;
        break;
    }
}

int scrank18(int mark)
{
    int
    SCrank_db18[10000]={278,148,130,117,109,104,98,93,89,86,83,80,78,75,73,7
    1,69,66,64,63,61,59,57,56,54,52,51,49,48,46,45,45,

    44,43,42,41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,21,20,
    19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,
    -1,-2,-3,-4,-5,-6,-7,-8};
    int q=0,i,marks=mark,rank,r;
    for(i=0;i<10000;i++)
    {
        if(SCrank_db18[i]==marks)
        {
            r=i;
            rank=r*100+1;
            return rank;
            break;}
        if(SCrank_db18[i]<marks)
        {
            if(i==0)
                rank=i;
            else
            {
                r=i;
                rank=(r*100+1);

```



```

        int p=SCrank_db18[i-1]-SCrank_db18[i];
        int q=marks-SCrank_db18[i];
        rank=rank-(rank*q)/p;
    }
    return rank;
    break;
}
}
}
int strank18(int mark)
{
    int q=0,i,marks=mark,rank,r;
    int
STrank_db18[10000]={266,124,101,85,76,68,61,55,50,46,45,44,42,40,35,33,3
1,28,25,22,21,19,17,15,13,11,7,3,1,-1,-5};
    for(i=0;i<10000;i++)
    {
        if(STrank_db18[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(STrank_db18[i]<marks)
        {
            if(i==0)
            rank=i;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=STrank_db18[i-1]-STrank_db18[i];
                int q=marks-STrank_db18[i];
                rank=rank-(rank*q)/p;
                return rank;
            }
            break;
        }
    }
}
int commonrank17(int mark)
{

```

int

Commonrank\_db17[10000]={339,305,295,289,284,279,276,272,269,266,264,  
261,258,257,254,253,251,249,247,246,244,243,241,

240,238,237,233,232,231,229,228,227,226,225,224,223,221,220,219,218,  
217,217,216,215,214,213,213,212,211,210,209,208,207,206,

206,205,205,204,204,203,202,202,201,200,200,199,198,198,197,197,196,  
195,195,194,193,193,192,191,190,190,189,189,188,188,187,

187,186,186,185,184,184,184,184,183,182,182,181,181,181,180,180,179,  
179,178,178,177,177,176,175,174,173,173,172,172,171,171,

171,170,170,170,169,169,169,169,168,168,167,167,167,166,166,166,165,  
165,164,164,164,163,163,163,162,162,162,161,161,161,160,

160,160,159,159,159,159,158,158,157,157,157,157,156,156,156,156,155,  
155,155,155,154,154,154,153,153,152,152,152,151,151,151,

151,150,150,150,149,149,149,149,149,148,148,148,148,148,148,147,147,  
147,147,146,146,145,145,145,145,144,144,144,144,144,144,

144,143,142,142,142,142,142,141,141,141,141,141,140,140,140,140,140,  
139,139,139,139,138,138,138,138,138,137,137,137,137,137,

137,136,136,136,136,136,135,135,135,135,135,135,134,134,134,134,133,  
133,133,133,133,133,133,132,132,132,132,132,131,131,

131,131,130,130,130,130,130,130,129,129,129,129,129,129,129,129,  
128,128,128,128,128,128,128,127,127,127,126,126,126,125,

125,124,124,124,123,123,123,122,122,122,121,121,121,120,120,120,119,  
119,119,118,118,118,117,117,117,116,116,116,115,115,115,

115,114,114,114,113,113,113,113,112,112,112,111,111,111,110,110,110,  
110,109,109,109,108,108,108,107,107,107,107,106,106,106,

105,105,105,105,104,104,104,104,103,103,103,102,102,102,102,101,101,  
101,101,101,100,100,100,99,99,99,99,98,98,98,98,98,97,97,

97,97,96,96,96,96,95,95,95,95,94,94,94,94,94,93,93,93,93,92,92,92,92,91,  
91,91,91,90,90,90,90,90,89,89,89,88,88,88,88,87,87,87,

87,86,86,86,85,85,85,85,84,84,84,84,84,83,83,83,82,82,82,82,81,80,80,79,  
79,79,78,78,78,78,77,76,75,74,73,72,71,70,69,68,67,66,

65,64,63,62,61,60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,  
41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,

23,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-5,-6,  
-7,-8};

```
int q=0,i,marks=mark,rank,r,p;
for(i=0;i<100000;i++){
    if(Commonrank_db17[i]==marks)
    {r=i;
    rank=r*100+1;
    return rank;
    break;}
if(Commonrank_db17[i]<marks)
{
    if(i==0)
    rank=i;
    else
    {
        r=i;
        rank=(r*100+1);
        p=Commonrank_db17[i-1]-Commonrank_db17[i];
        int q=marks-Commonrank_db17[i];
        rank=rank-(rank*q)/p;
    }
    return rank;
    break;
}
}
```

```
}
int obcrank17(int mark)
{
    int
```

OBCrank\_db17[10000]={340,209,193,183,170,168,165,156,152,149,146,144,1  
42,140,138,136,134,132,131,129,128,127,126,125,125,

123,122,121,119,116,114,113,112,112,111,110,109,108,107,106,105,104,  
103,102,101,100,99,98,97,96,96,95,93,92,91,90,89,88,88,

```
87,86,85,85,84,83,83,82,81,81,81,80,79,79,78,77,76,75,74,74,73,72,72,72,
71,70,69,68,67,66,65,64,63,62,61,60,59,58,57,56,55,
```

```
54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32,31,
30,29,28,27,26,25,24,23,21,20,19,18,17,16,15,14,13,
12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-8};
```

```
int q=0,i,marks=mark,rank,r,p;
```

```
for(i=0;i<10000;i++)
```

```
{
```

```
if(OBCrank_db17[i]==marks)
```

```
{r=i;
```

```
rank=r*100+1;
```

```
return rank;
```

```
break;}
```

```
if(OBCrank_db17[i]<marks)
```

```
{
```

```
if(i==0)
```

```
rank=i;
```

```
else
```

```
{
```

```
r=i;
```

```
rank=(r*100+1);
```

```
p=OBCrank_db17[i-1]-OBCrank_db17[i];
```

```
int q=marks-OBCrank_db17[i];
```

```
rank=rank-(rank*q)/p;
```

```
}
```

```
return rank;
```

```
break;
```

```
}
```

```
}
```

```
}
```

```
int scrank17(int mark)
```

```
{
```

```
int
```

```
SCrank_db17[10000]={273,188,130,117,107,104,98,93,87,86,83,80,78,75,73,7
1,69,65,64,63,61,59,57,56,54,52,51,49,48,46,45,45,
```

```
44,43,42,41,40,39,38,37,36,35,34,34,32,31,30,29,28,27,27,25,24,23,21,20,
19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,
-1,-2,-3,-4,-5,-6,-7,-8};
```

```
int q=0,i,marks=mark,rank,r;
```

```
for(i=0;i<10000;i++)
```

```

        {
            if(SCrank_db17[i]==marks)
            {r=i;
            rank=r*100+1;
            return rank;
            break;}
        if(SCrank_db17[i]<marks)
        {
            if(i==0)
            rank=i;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=SCrank_db17[i-1]-SCrank_db17[i];
                int q=marks-SCrank_db17[i];
                rank=rank-(rank*q)/p;
            }
            return rank;
            break;
        }
    }
}

int strank17(int mark)
{
    int
STrank_db17[10000]={256,134,111,76,76,67,61,56,50,46,45,44,40,40,35,33,3
1,29,27,22,20,19,17,15,12,11,7,3,1,-1,-5};
    int q=0,i,marks=mark,rank,r;
    for(i=0;i<10000;i++)
    {
        if(STrank_db17[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(STrank_db17[i]<marks)
        {
            if(i==0)
            rank=i;
            else
            {
                r=i;

```

```

        rank=(r*100+1);
        int p=STrank_db17[i-1]-STrank_db17[i];
        int q=marks-STrank_db17[i];
        rank=rank-(rank*q)/p;
    }
    return rank;
    break;
}
}
}
int callRankDb18(int mark,string choice)
{
    Rank(mark,2018,choice);
    int y;
    if(choice=="Common")
        y=commonrank18(mark);
    else if(choice=="OBC")
        y=obcrank18(mark);
    else if(choice=="SC")
        y=scrank18(mark);
    else if(choice=="ST")
        y=strank18(mark);
    cout<<y<<"\n";
    return y;
}
int callRankDb17(int mark,string choice)
{
    Rank(mark,2017,choice);
    int y;
    if(choice=="Common")
        y=commonrank17(mark);
    else if(choice=="OBC")
        y=obcrank17(mark);
    else if(choice=="SC")
        y=scrank17(mark);
    else if(choice=="ST")
        y=strank17(mark);
    cout<<y<<"\n";
    return y;
}
};
int main()
{

```

```

Algorithm a;
RankDb r;
int rank18,rank17,getMarks;
string getChoice;
getMarks=a.call1Algorithm();
getChoice=a.call2Algorithm();
rank18=r.callRankDb18(getMarks,getChoice);
rank17=r.callRankDb17(getMarks,getChoice);
a.callMethod(rank18,rank17);
}

```

#### **CODE 4:**

##### **Interaction with College\_Db & Algorithm for fetching college**

```

#include<iostream>
using namespace std;
class Algorithm
{
    int rank;
    public:
    int min_rank=8401;
    int max_rank=9001;
    int Rank=(min_rank+max_rank)/2;
    string category="Common";
    int callAlgo_rank()
    {
        cout<<"Algorithm is fetching college from collegeList for";
        return Rank;
    }
    string choice()
    {
        cout<<" category "<<category<<"...\n";
        return category; //fetched
    }
    void dispCollege(string college)
    {
        cout<<"The list of college you may get on rank "<<Rank<<" for
category "<<category<<" is: \n"<<college;
    }
}

```

```

};
class CollegeList
{
    public:
    string college(int rank,string choice)
    {
        string r="";
        cout<<"CollegeDb has rank vs college list\n";
        if(choice=="Common")
        {
            if(rank>=7 && rank<=4669)
                r+="IIT DELHI ";
            if(rank>=1 && rank<=4018)
                r+="IIT BOMBAY ";
            if(rank>=2121 && rank<=6299)
                r+="IIT GANDHINAGAR ";
            if(rank>=366 && rank<=6578)
                r+="IIT GUWAHATI ";
            if(rank>=696 && rank<=7376)
                r+="IIT VARANASI ";
            if(rank>=632 && rank<=5342)
                r+="IIT HYDERABAD ";
            if(rank>=40 && rank<=5722)
                r+="IIT KANPUR ";
            if(rank>=1366 && rank<=6895)
                r+="IIT JODHPUR ";
            if(rank>=1388 && rank<=4446)
                r+="IIT INDORE ";
            if(rank>=2819 && rank<=6190)
                r+="IIT PATNA ";
            if(rank>=180 && rank<=8430)
                r+="IIT KHARAGPUR ";
            if(rank>=2486 && rank<=5146)
                r+="IIT MANDI ";
            if(rank>=32 && rank<=5819)
                r+="IIT CHENNAI ";
            if(rank>=2845 && rank<=7222)
                r+="IIT DHANBAD ";
            if(rank>=614 && rank<=9290)

```



```
r+="IIT GANDHINAGAR  ";
if(rank>=1276 && rank<=4152)
r+="IIT ROPAR  ";
if(rank>9290)
r+="It seems like you won't get IIT this year...";
}
else if(choice=="OBC")
{
if(rank>=7 && rank<=1500)
r+="IIT DELHI  ";
if(rank>=1 && rank<=3000)
r+="IIT BOMBAY  ";
if(rank>=521 && rank<=4200)
r+="IIT GANDHINAGAR  ";
if(rank>=366 && rank<=4100)
r+="IIT GUWAHATI  ";
if(rank>=196 && rank<=4566)
r+="IIT VARANASI  ";
if(rank>=632 && rank<=4342)
r+="IIT HYDERABAD ";
if(rank>=40 && rank<=3722)
r+="IIT KANPUR  ";
if(rank>=366 && rank<=4895)
r+="IIT JODHPUR  ";
if(rank>=188 && rank<=2446)
r+="IIT INDORE  ";
if(rank>=819 && rank<=3190)
r+="IIT PATNA ";
if(rank>=180 && rank<=4430)
r+="IIT KHARAGPUR ";
if(rank>=586 && rank<=3146)
r+="IIT MANDI  ";
if(rank>=12 && rank<=3819)
r+="IIT CHENNAI  ";
if(rank>=845 && rank<=3222)
r+="IIT DHANBAD  ";
if(rank>=314 && rank<=3290)
r+="IIT GANDHINAGAR  ";
if(rank>=376 && rank<=2152)
r+="IIT ROPAR  ";
```

```
if(rank>4895)
r+="It seems like you won't get IIT this year...";
}
else if(choice=="SC")
{
if(rank>=1 && rank<=500)
r+="IIT DELHI  ";
if(rank>=1 && rank<=700)
r+="IIT BOMBAY  ";
if(rank>=121 && rank<=900)
r+="IIT GANDHINAGAR  ";
if(rank>=166 && rank<=1100)
r+="IIT GUWAHATI  ";
if(rank>=56 && rank<=1166)
r+="IIT VARANASI  ";
if(rank>=12 && rank<=1342)
r+="IIT HYDERABAD ";
if(rank>=4 && rank<=922)
r+="IIT KANPUR  ";
if(rank>=166 && rank<=995)
r+="IIT JODHPUR  ";
if(rank>=88 && rank<=1446)
r+="IIT INDORE  ";
if(rank>=219 && rank<=1190)
r+="IIT PATNA ";
if(rank>=80 && rank<=1430)
r+="IIT KHARAGPUR ";
if(rank>=186 && rank<=1146)
r+="IIT MANDI  ";
if(rank>=12 && rank<=1319)
r+="IIT CHENNAI  ";
if(rank>=245 && rank<=1222)
r+="IIT DHANBAD  ";
if(rank>=114 && rank<=1290)
r+="IIT GANDHINAGAR  ";
if(rank>=276 && rank<=1152)
r+="IIT ROPAR  ";
if(rank>1446)
r+="It seems like you won't get IIT this year...";
}
```

```
else if(choice=="ST")
{
if(rank>=1 && rank<=500)
r+="IIT DELHI ";
if(rank>=1 && rank<=700)
r+="IIT BOMBAY ";
if(rank>=121 && rank<=900)
r+="IIT GANDHINAGAR ";
if(rank>=166 && rank<=1100)
r+="IIT GUWAHATI ";
if(rank>=56 && rank<=1166)
r+="IIT VARANASI ";
if(rank>=12 && rank<=1342)
r+="IIT HYDERABAD ";
if(rank>=4 && rank<=922)
r+="IIT KANPUR ";
if(rank>=166 && rank<=995)
r+="IIT JODHPUR ";
if(rank>=88 && rank<=1446)
r+="IIT INDORE ";
if(rank>=219 && rank<=1190)
r+="IIT PATNA ";
if(rank>=80 && rank<=1430)
r+="IIT KHARAGPUR ";
if(rank>=186 && rank<=1146)
r+="IIT MANDI ";
if(rank>=12 && rank<=1319)
r+="IIT CHENNAI ";
if(rank>=245 && rank<=1222)
r+="IIT DHANBAD ";
if(rank>=114 && rank<=1290)
r+="IIT GANDHINAGAR ";
if(rank>=276 && rank<=1152)
r+="IIT ROPAR ";
if(rank>1446)
r+="It seems like you won't get IIT this year...";
}
return r;
```

```
}
```

```

};
int main()
{
    Algorithm a;
    CollegeList c;
    int rank=a.callAlgo_rank();
    string choice=a.choice();
    string College_name=c.college(rank,choice);
    a.dispCollege(College_name);
}

```

## CODE 5:

### Interaction with Rank\_Predictor & Adminstrator for fetching college

```

#include<iostream>
using namespace std;
class Rank_Predictor
{
    public:
        string predictor_response()
        {
            string viewers_report="O.K.";//fetched from interface if viewer reports
            cout<<"Rank Predictors reports its status as "<<viewers_report<<".\n";
            return viewers_report;
        }
};
class Admin
{
    public:
        void admin_action(string resp)
        {
            if(resp=="O.K.")
                cout<<"Rank Predictor is working fine...";
            else
                cout<<"Rank Predictor needs maintainance..Work in Progress";
        }
};
int main()
{

```

```

Rank_Predictor r;
Admin d;
string response=r.predictor_response();
d.admin_action(response);
}

```

## FLOW OF OUTPUT BETWEEN OBJECTS OF DIFFERENT CLASSES:

### Interaction with Rank\_Predictor & Student



```

C:\Users\MANYA SMRITI\Documents\oodadpro1.exe
Enter marks
135
Enter 1-Common_rank,2-OBC_rank,3-SC_rank,4-ST_rank,5-PWD_rank,6-Reset marks,7-Search_college
1
You choose to view Common rank for entered marks
-----
Process exited after 3.724 seconds with return value 0
Press any key to continue . . .

```

### Interaction with Rank\_Predictor & Algorithm for choice



```

C:\Users\MANYA SMRITI\Documents\oodadpro2.exe
Predicting common_rank for marks 135
Marks sent to algorithm
Algorithm sends common rank 8750 to rank_predictor
-----
Process exited after 0.04769 seconds with return value 0
Press any key to continue . . .

```

(Rank predictor asks for rank to algorithm and after all internal processing ,algorithm sends back the rank)

### Interaction with Rank\_Db & Algorithm for fetching rank

 C:\Users\MANYA SMRITI\Documents\oodadpro3.exe

```
The Algorithm is fetching rank for 135 for Common category...
Rank on mark 135 for year 2018 for Common category is 8701
Rank on mark 135 for year 2017 for Common category is 24601
Possible rank range for your marks is 8401-9001
-----
Process exited after 0.06706 seconds with return value 0
Press any key to continue . . .
```

## Interaction with College\_Db & Algorithm for fetching college

 C:\Users\MANYA SMRITI\Documents\oodadpro4.exe

```
Algorithm is fetching college from collegeList for category Common...
CollegeDb has rank vs college list
The list of college you may get on rank 8701 for category Common is:
IIT GANDHINAGAR
-----
Process exited after 0.1032 seconds with return value 0
Press any key to continue . . .
```

## Interaction with Rank\_Predictor & Administrator

 C:\Users\MANYA SMRITI\Documents\oodadpro5.exe

```
Rank Predictors reports its status as O.K..
Rank Predictor is working fine...
-----
Process exited after 0.07522 seconds with return value 0
Press any key to continue . . .
```

## Execution

//Interaction with Rank\_Predictor & Student

```
#include<iostream>
#include<cstdlib>
#include<time.h>
#include<string.h>
#include<math.h>
using namespace std;
class Admin
```

```

{
    public:
        int askadmin()
        {
            cout<<"Please enter the difficulty level according to paper
analyser for this year\n";//on scale of 3
            float paper_level;
            cin>>paper_level;
            cout<<"Please enter the no. of participants appeared for the exam
this year\n";
            float participant;
            cin>>participant;
            float level;
            int no_of_participant2018=165000;
            level=(paper_level)*(participant/(no_of_participant2018));
            return level;
        }
        void admin_action(string resp)
        {
            if(resp=="O.K.")
            {}
            else
            cout<<"Rank Predictor sends message to admin that it needs
maintainance..Work in Progress\n";
        }
};
class Student
{
    public:
    int marks;
    int choice;
    int callStudent()
    {
        if(choice==6)
        cout<<"Enter marks\n";
        cin>>marks;
        return marks;
    }
    int choiceStudent()
    {
        cout<<"Enter
1-Common_rank,2-OBC_rank,3-SC_rank,4-ST_rank,5-PWD_rank,6-Search_coll
ege\n";
    }
}

```

```

cin>>choice;
    if(choice!=6)
        cout<<"Enter marks\n";
return choice;
}
int askcate()
{
    int cate;
    cout<<"enter category\n";
    cin>>cate;
    return cate;
}
int askChoice()
{
    cout<<"Press 1-reset marks 0- exit 100-report \n";
    cin>>choice;
    return choice;
}
};
class Algorithm
{
    public:
    int marks,rank,z;
    string choice;
    public:
    int call1Algorithm(int marks)
    {
        //fetched marks from rank Predictor
        cout<<"The Algorithm is fetching rank for "<<marks<<"\n";
        return marks;
    }
    int callMethod(float difficulty_level,int rank18,int rank17)
    {
        int i,j,no_digit18,no_digit17,temp,z;
        float min,max,t;
        cout<<"According to analyzed difficulty level this year\n";
        float difficulty_level1=5;
        float difficulty_level2=3;
        if(difficulty_level1<difficulty_level2)
        {
            t=difficulty_level1;
            difficulty_level1=difficulty_level2;
            difficulty_level2=t;
        }
    }
}

```



```

t=max;
max=min;
min=t;
} //to ensure difficulty_level1 gets right value.
temp=rank18,no_digit18=0,no_digit17=0;
while(temp>0)
{
    temp=temp/10;
    no_digit18+=1;
}
temp=rank17;
while(temp>0)
{
    temp=temp/10;
    no_digit17+=1;
}
if(difficulty_level>=difficulty_level1)
{
    min=rank18;
    max=rank18;
    if(rank18==0 || rank18==1)
    {
        min=1;
        max=1;
    }
    else if(rank18==2)
    {
        min=1;
        max=3;
    }
    int k=no_digit18-2;
    if(k<0)
    k=0;
    else if(rank18>3 && rank18<10)
    {
        min=min-2;
        max=max+2;
    }
    else if(rank18>10 && rank18<20)
    {
        min=min-7;
        max=max+7;
    }
}

```

```

        else if(rank18>19 && rank18<31)
        {
            min=min*(difficulty_level*.1)-10;
            max=max*(difficulty_level*.1)+10;
        }
        else if(rank18<201)
        {
            min=min*(difficulty_level*.1)-2*pow(10,1);
            max=max*(difficulty_level*.1)+2*pow(10,1);
        }
        else if(rank18<501)
        {
            min=min*(difficulty_level*.1)-100;
            max=max*(difficulty_level*.1)+100;
        }
        else if(rank18<1001)
        {
            min=min*(difficulty_level*.1)-300;
            max=max*(difficulty_level*.1)+300;
        }
        else
        if(rank18<5000)
        {
            min=min*(difficulty_level*.1)-3*pow(10,k);
            max=max*(difficulty_level*.1)+3*pow(10,k);
        }
        else
        {
            min=min*(difficulty_level*.1)-6*pow(10,k);
            max=max*(difficulty_level*.1)+6*pow(10,k);
        }
    }
    else
    //if(difficulty_level<difficulty_level2)
    {
        min=rank17;
        max=rank17;
        int k=no_digit17-2;
        if(k<0)
        k=0;
        if(rank17==0 || rank17==1)
        {

```

```

        min=1;
        max=1;
    }
    else if(rank17==2)
    {
        min=1;
        max=3;
    }
    else if(rank17>3 && rank17<10)
    {
        min=min-2;
        max=max+2;
    }
    else if(rank17>10 && rank17<20)
    {
        min=min-7;
        max=max+7;
    }
    else if(rank17>19 && rank17<51)
    {
        min=min-10;
        max=max+10;
    }
    else if(rank17<201)
    {
        min=min*(difficulty_level*.1)-3*pow(10,1);
        max=max*(difficulty_level*.1)+3*pow(10,1);
    }
    else if(rank17<501)
    {
        min=min*(difficulty_level*.1)-100;
        max=max*(difficulty_level*.1)+100;
    }
    else if(rank17<1001)
    {
        min=min*(difficulty_level*.1)-300;
        max=max*(difficulty_level*.1)+300;
    }
    else
    {
        if(rank17<5000){
            min=min*(difficulty_level*.1)-3*pow(10,k);
            max=max*(difficulty_level*.1)+3*pow(10,k);
        }
    }

```

```

    }
    else
    {
        min=min*(difficulty_level*.1)-6*pow(10,k);
        max=max*(difficulty_level*.1)+6*pow(10,k);
    }
}

    cout<<"Possible rank range for your marks is
"<<(int)min<<"-"<<(int)max<<"\n";
    z=(min+max);
    z=z/2;
    return z;
}
int callAlgo_rank()
{
    cout<<"Algorithm is fetching college from collegeList for";
}
void choice1()
{
    cout<<" this category...\n";

}
    void callAlgorithm(int avg_rank)
    {
        cout<<"Algorithm sends an average estimated rank "<<avg_rank<<" to
rank_predictor\n";
    }
    void dispCollege(int rank,string college)
    {
        cout<<"The list of college you may get on rank "<<rank<<" for this
category is: \n"<<college<<"\n";
    }
};
class CollegeList
{
    public:
    string college(int rank,int choice)
    {
        string r="";
        cout<<"CollegeDb has rank vs college list\n";
        if(choice==1)
        {

```

```
if(rank>=7 && rank<=4669)
r+="IIT DELHI  ";
if(rank>=1 && rank<=4018)
r+="IIT BOMBAY  ";
if(rank>=2121 && rank<=6299)
r+="IIT GANDHINAGAR  ";
if(rank>=366 && rank<=6578)
r+="IIT GUWAHATI  ";
if(rank>=696 && rank<=7376)
r+="IIT VARANASI  ";
if(rank>=632 && rank<=5342)
r+="IIT HYDERABAD ";
if(rank>=40 && rank<=5722)
r+="IIT KANPUR  ";
if(rank>=1366 && rank<=6895)
r+="IIT JODHPUR  ";
if(rank>=1388 && rank<=4446)
r+="IIT INDORE  ";
if(rank>=2819 && rank<=6190)
r+="IIT PATNA ";
if(rank>=180 && rank<=8430)
r+="IIT KHARAGPUR ";
if(rank>=2486 && rank<=5146)
r+="IIT MANDI  ";
if(rank>=32 && rank<=5819)
r+="IIT CHENNAI  ";
if(rank>=2845 && rank<=7222)
r+="IIT DHANBAD  ";
if(rank>=614 && rank<=9290)
r+="IIT GANDHINAGAR  ";
if(rank>=1276 && rank<=4152)
r+="IIT ROPAR  ";
if(rank>9290)
r+="It seems like you won't get IIT this year...";
}
else if(choice==2)
{
if(rank>=7 && rank<=1500)
r+="IIT DELHI  ";
if(rank>=1 && rank<=3000)
r+="IIT BOMBAY  ";
if(rank>=521 && rank<=4200)
r+="IIT GANDHINAGAR  ";
```

```
if(rank>=366 && rank<=4100)
r+="IIT GUWAHATI ";
if(rank>=196 && rank<=4566)
r+="IIT VARANASI ";
if(rank>=632 && rank<=4342)
r+="IIT HYDERABAD ";
if(rank>=40 && rank<=3722)
r+="IIT KANPUR ";
if(rank>=366 && rank<=4895)
r+="IIT JODHPUR ";
if(rank>=188 && rank<=2446)
r+="IIT INDORE ";
if(rank>=819 && rank<=3190)
r+="IIT PATNA ";
if(rank>=180 && rank<=4430)
r+="IIT KHARAGPUR ";
if(rank>=586 && rank<=3146)
r+="IIT MANDI ";
if(rank>=12 && rank<=3819)
r+="IIT CHENNAI ";
if(rank>=845 && rank<=3222)
r+="IIT DHANBAD ";
if(rank>=314 && rank<=3290)
r+="IIT GANDHINAGAR ";
if(rank>=376 && rank<=2152)
r+="IIT ROPAR ";
if(rank>4895)
r+="It seems like you won't get IIT this year...";
}
else if(choice==3)
{
if(rank>=1 && rank<=500)
r+="IIT DELHI ";
if(rank>=1 && rank<=700)
r+="IIT BOMBAY ";
if(rank>=121 && rank<=900)
r+="IIT GANDHINAGAR ";
if(rank>=166 && rank<=1100)
r+="IIT GUWAHATI ";
if(rank>=56 && rank<=1166)
r+="IIT VARANASI ";
if(rank>=12 && rank<=1342)
r+="IIT HYDERABAD ";
```

```
if(rank>=4 && rank<=922)
r+="IIT KANPUR  ";
if(rank>=166 && rank<=995)
r+="IIT JODHPUR  ";
if(rank>=88 && rank<=1446)
r+="IIT INDORE  ";
if(rank>=219 && rank<=1190)
r+="IIT PATNA ";
if(rank>=80 && rank<=1430)
r+="IIT KHARAGPUR ";
if(rank>=186 && rank<=1146)
r+="IIT MANDI  ";
if(rank>=12 && rank<=1319)
r+="IIT CHENNAI  ";
if(rank>=245 && rank<=1222)
r+="IIT DHANBAD  ";
if(rank>=114 && rank<=1290)
r+="IIT GANDHINAGAR  ";
if(rank>=276 && rank<=1152)
r+="IIT ROPAR  ";
if(rank>1446)
r+="It seems like you won't get IIT this year...";
}
else if(choice==4 || choice==5)
{
if(rank>=1 && rank<=500)
r+="IIT DELHI  ";
if(rank>=1 && rank<=700)
r+="IIT BOMBAY  ";
if(rank>=121 && rank<=900)
r+="IIT GANDHINAGAR  ";
if(rank>=166 && rank<=1100)
r+="IIT GUWAHATI  ";
if(rank>=56 && rank<=1166)
r+="IIT VARANASI  ";
if(rank>=12 && rank<=1342)
r+="IIT HYDERABAD ";
if(rank>=4 && rank<=922)
r+="IIT KANPUR  ";
if(rank>=166 && rank<=995)
r+="IIT JODHPUR  ";
if(rank>=88 && rank<=1446)
r+="IIT INDORE  ";
```

```

        if(rank>=219 && rank<=1190)
            r+="IIT PATNA ";
        if(rank>=80 && rank<=1430)
            r+="IIT KHARAGPUR ";
        if(rank>=186 && rank<=1146)
            r+="IIT MANDI ";
        if(rank>=12 && rank<=1319)
            r+="IIT CHENNAI ";
        if(rank>=245 && rank<=1222)
            r+="IIT DHANBAD ";
        if(rank>=114 && rank<=1290)
            r+="IIT GANDHINAGAR ";
        if(rank>=276 && rank<=1152)
            r+="IIT ROPAR ";
        if(rank>1446)
            r+="It seems like you won't get IIT this year...";
        }
        return r;
    }
};

class RankDb
{
    public:
        void Rank(int mark,int year,int c)
        {
            string cat;
            if(c==1)
                cat="Common";
            if(c==2)
                cat="OBC";
            if(c==3)
                cat="SC";
            if(c==4)
                cat="ST";
            if(c==5)
                cat="PWD";
            cout<<"Rank on mark "<<mark<<" for year "<<year<<" for
"<<cat<<" category is \n";
        }
        int commonrank18(int mark)
        {
            int q=0,i,marks=mark,r,rank,p=0;

```



```

int
Commonrank_db18[10000]={337,272,258,247,240,233,227,223,219,216,213,
210,207,204,202,200,198,196,195,193,191,190,188,

187,185,184,182,181,180,179,177,176,175,174,173,172,171,170,169,168,
167,165,164,164,163,162,161,161,160,159,158,158,

156,155,154,154,153,152,152,151,150,150,149,148,148,147,147,146,145,
145,144,144,143,143,142,142,141,140,140,139,139,

138,138,137,137,136,136,135,135,135,134,134,133,133,132,132,131,131,
131,130,130,129,129,129,128,128,127,127,127,126,

126,126,125,125,124,124,124,123,123,123,122,122,122,121,121,121,120,120,
120,119,119,119,118,118,118,117,117,117,116,

116,116,115,115,115,115,114,114,114,113,113,113,113,112,112,112,111,111,
111,110,110,110,110,109,109,109,108,108,108,

107,107,107,107,106,106,106,105,105,105,105,104,104,104,104,103,103,103,
102,102,102,102,101,101,101,101,101,100,100,

100,99,99,99,99,98,98,98,98,98,97,97,97,97,96,96,96,96,95,95,95,95,94,94,94,
94,94,93,93,93,93,92,92,92,92,91,91,91,

91,90,90,90,90,90,89,89,89,88,88,88,88,87,87,87,87,86,86,86,85,85,85,85,
84,84,84,84,84,83,83,83,82,82,82,82,81,80,80,

79,79,79,78,78,78,78,77,76,75,74,73,72,71,70,69,68,67,66,65,64,63,62,61,
60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,

45,44,43,42,41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,21,
20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,
3,2,1,0,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,-11};//Rank 1-337 marks,Rank
101-272 marks,Rank 201-258 marks,Rank 301-247 marks
/* In real the first index will give the 1st ranker's mark but due to
lack of DB here the interval taken btw 2 indices is 100*/
for(i=0;i<10000;i++)
{
    if(Commonrank_db18[i]==marks)
    {r=i;
    rank=r*100+1;
    return rank;
}

```

```

        break;}
        if(Commonrank_db18[i]<marks)
        {
            if(i==0)
            rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                p=Commonrank_db18[i-1]-Commonrank_db18[i];
                int q=marks-Commonrank_db18[i];
                rank=rank-(rank*q)/p;
            }
            return rank;
            break;
        }
    }
}

int obcrank18(int mark)
{
    int q=0,i,marks=mark,rank,r;
    int
OBCrank_db18[10000]={319,213,198,188,179,173,167,163,159,155,152,149,1
46,143,141,138,136,134,132,130,129,127,126,124,

    123,121,120,118,117,116,115,113,112,111,110,109,108,107,106,105,104,
103,102,101,100,99,99,98,97,96,95,94,93,92,91,90,

    89,88,88,87,86,85,85,84,83,83,82,81,81,81,80,79,79,78,77,76,75,74,74,73,
72,72,72,71,70,69,68,67,66,65,64,63,62,61,60,

    59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,
35,34,33,32,31,30,29,28,27,26,25,24,23,21,20,

    19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-5,-6,-7,-8};
    for(i=0;i<10000;i++)
    {
        if(OBCrank_db18[i]==marks)
        {
            r=i;
            rank=r*100+1;
            return rank;
            break;}
    }
}

```

```

        if(OBCrank_db18[i]<marks)
        {
            if(i==0)
                rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=OBCrank_db18[i-1]-OBCrank_db18[i];
                int q=marks-OBCrank_db18[i];
                rank=rank-(rank*q)/p;
            }
            return rank;
            break;
        }
    }
}

int scrank18(int mark)
{
    int
SCrank_db18[10000]={278,148,130,117,109,104,98,93,89,86,83,80,78,75,73,7
1,69,66,64,63,61,59,57,56,54,52,51,49,48,46,45,45,

    44,43,42,41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,21,20,
19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,
    -1,-2,-3,-4,-5,-6,-7,-8};
    int q=0,i,marks=mark,rank,r;
    for(i=0;i<10000;i++)
    {
        if(SCrank_db18[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(SCrank_db18[i]<marks)
        {
            if(i==0)
                rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=SCrank_db18[i-1]-SCrank_db18[i];

```

```

        int q=marks-SCrank_db18[i];
        rank=rank-(rank*q)/p;
    }
    return rank;
    break;
}
}
}
int strank18(int mark)
{
    int q=0,i,marks=mark,rank,r;
    int
STrank_db18[10000]={266,124,101,85,76,68,61,55,50,46,45,44,42,40,35,33,3
1,28,25,22,21,19,17,15,13,11,7,3,1,-1,-5};
    for(i=0;i<10000;i++)
    {
        if(STrank_db18[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(STrank_db18[i]<marks)
        {
            if(i==0)
            rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=STrank_db18[i-1]-STrank_db18[i];
                int q=marks-STrank_db18[i];
                rank=rank-(rank*q)/p;
                return rank;
            }
            break;
        }
    }
}
int Pwdrank18(int mark)
{
    int q=0,i,marks=mark,rank,r;

```

```

        int
Pwdrank_db18[10000]={200,124,101,78,61,55,50,28,25,22,21,19,17,15,13,11,
7,3,1,-1,-5};
        for(i=0;i<10000;i++)
        {
            if(Pwdrank_db18[i]==marks)
            {r=i;
rank=r*100+1;
            return rank;
            break;}
        if(Pwdrank_db18[i]<marks)
        {
            if(i==0)
            rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=Pwdrank_db18[i-1]-Pwdrank_db18[i];
                int q=marks-Pwdrank_db18[i];
                rank=rank-(rank*q)/p;
                return rank;
            }
            break;
        }
    }
}
int commonrank17(int mark)
{
    int
Commonrank_db17[10000]={339,305,295,289,284,279,276,272,269,266,264,
261,258,257,254,253,251,249,247,246,244,243,241,

    240,238,237,233,232,231,229,228,227,226,225,224,223,221,220,219,218,
217,217,216,215,214,213,213,212,211,210,209,208,207,206,

    206,205,205,204,204,203,202,202,201,200,200,199,198,198,197,197,196,
195,195,194,193,193,192,191,190,190,189,189,188,188,187,

    187,186,186,185,184,184,184,184,183,182,182,181,181,181,180,180,179,
179,178,178,177,177,176,175,174,173,173,172,172,171,171,

```

171,170,170,170,169,169,169,169,168,168,167,167,167,166,166,166,165,  
165,164,164,164,163,163,163,162,162,162,161,161,161,160,

160,160,159,159,159,159,158,158,157,157,157,157,156,156,156,155,  
155,155,155,154,154,154,153,153,152,152,152,151,151,151,

151,150,150,150,149,149,149,149,149,148,148,148,148,148,147,147,  
147,147,146,146,145,145,145,145,144,144,144,144,144,144,

144,143,142,142,142,142,142,141,141,141,141,141,140,140,140,140,  
139,139,139,139,138,138,138,138,138,137,137,137,137,137,

137,136,136,136,136,136,135,135,135,135,135,135,134,134,134,133,  
133,133,133,133,133,133,133,132,132,132,132,132,131,131,

131,131,130,130,130,130,130,130,129,129,129,129,129,129,129,129,  
128,128,128,128,128,128,128,127,127,127,126,126,126,125,

125,124,124,124,123,123,123,122,122,122,121,121,121,120,120,120,119,  
119,119,118,118,118,117,117,117,116,116,116,115,115,115,

115,114,114,114,113,113,113,113,112,112,112,111,111,111,110,110,110,  
110,109,109,109,108,108,108,107,107,107,107,106,106,106,

105,105,105,105,104,104,104,104,103,103,103,102,102,102,102,101,101,  
101,101,101,100,100,100,99,99,99,99,98,98,98,98,98,97,97,

97,97,96,96,96,96,95,95,95,95,94,94,94,94,94,93,93,93,93,92,92,92,92,91,  
91,91,91,90,90,90,90,90,89,89,89,88,88,88,88,87,87,87,

87,86,86,86,85,85,85,85,84,84,84,84,84,83,83,83,82,82,82,82,81,80,80,79,  
79,79,78,78,78,78,77,76,75,74,73,72,71,70,69,68,67,66,

65,64,63,62,61,60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,  
41,40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,

23,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-5,-6,  
-7,-8};

```
int q=0,i,marks=mark,rank,r,p;  
for(i=0;i<100000;i++){  
    if(Commonrank_db17[i]==marks)  
        {r=i;
```

```

        rank=r*100+1;
        return rank;
        break;}
    if(Commonrank_db17[i]<marks)
    {
        if(i==0)
            rank=i+1;
        else
        {
            r=i;
            rank=(r*100+1);
            p=Commonrank_db17[i-1]-Commonrank_db17[i];
            int q=marks-Commonrank_db17[i];
            rank=rank-(rank*q)/p;
        }
        return rank;
        break;
    }
}

int obcrank17(int mark)
{
    int
OBCrank_db17[10000]={340,209,193,183,170,168,165,156,152,149,146,144,1
42,140,138,136,134,132,131,129,128,127,126,125,125,

    123,122,121,119,116,114,113,112,112,111,110,109,108,107,106,105,104,
103,102,101,100,99,98,97,96,96,95,93,92,91,90,89,88,88,

    87,86,85,85,84,83,83,82,81,81,81,80,79,79,78,77,76,75,74,74,73,72,72,72,
71,70,69,68,67,66,65,64,63,62,61,60,59,58,57,56,55,

    54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32,31,
30,29,28,27,26,25,24,23,21,20,19,18,17,16,15,14,13,
    12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-8};
    int q=0,i,marks=mark,rank,r,p;
    for(i=0;i<10000;i++)
    {
        if(OBCrank_db17[i]==marks)
        {r=i;
            rank=r*100+1;
            return rank;
            break;}
    }
}

```

```

        if(OBCrank_db17[i]<marks)
        {
            if(i==0)
                rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                p=OBCrank_db17[i-1]-OBCrank_db17[i];
                int q=marks-OBCrank_db17[i];
                rank=rank-(rank*q)/p;
            }
            return rank;
            break;
        }
    }
}

int scrank17(int mark)
{
    int
SCrank_db17[10000]={273,188,130,117,107,104,98,93,87,86,83,80,78,75,73,7
1,69,65,64,63,61,59,57,56,54,52,51,49,48,46,45,45,

    44,43,42,41,40,39,38,37,36,35,34,34,32,31,30,29,28,27,27,25,24,23,21,20,
19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,
    -1,-2,-3,-4,-5,-6,-7,-8};
    int q=0,i,marks=mark,rank,r;
    for(i=0;i<10000;i++)
    {
        if(SCrank_db17[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(SCrank_db17[i]<marks)
        {
            if(i==0)
                rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=SCrank_db17[i-1]-SCrank_db17[i];

```



```

        int q=marks-SCrank_db17[i];
        rank=rank-(rank*q)/p;
    }
    return rank;
    break;
}
}
}
int strank17(int mark)
{
    int
STrank_db17[10000]={256,134,111,76,76,67,61,56,50,46,45,44,40,40,35,33,3
1,29,27,22,20,19,17,15,12,11,7,3,1,-1,-5};
    int q=0,i,marks=mark,rank,r;
    for(i=0;i<10000;i++)
    {
        if(STrank_db17[i]==marks)
        {r=i;
        rank=r*100+1;
        return rank;
        break;}
        if(STrank_db17[i]<marks)
        {
            if(i==0)
            rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=STrank_db17[i-1]-STrank_db17[i];
                int q=marks-STrank_db17[i];
                rank=rank-(rank*q)/p;
            }
            return rank;
            break;
        }
    }
}

int Pwdrank17(int mark)
{
    int q=0,i,marks=mark,rank,r;

```

```

        int
        Pwdrank_db17[10000]={200,124,101,78,61,55,50,28,25,22,21,19,17,15,13,11,
        7,3,1,-1,-5};
        for(i=0;i<10000;i++)
        {
            if(Pwdrank_db17[i]==marks)
            {r=i;
            rank=r*100+1;
            return rank;
            break;}
        if(Pwdrank_db17[i]<marks)
        {
            if(i==0)
            rank=i+1;
            else
            {
                r=i;
                rank=(r*100+1);
                int p=Pwdrank_db17[i-1]-Pwdrank_db17[i];
                int q=marks-Pwdrank_db17[i];
                rank=rank-(rank*q)/p;
                return rank;
            }
            break;
        }
    }
}

int callRankDb18(int mark,int choice)
{
    Rank(mark,2018,choice);
    int y;
    if(choice==1)
    y=commonrank18(mark);
    else if(choice==2)
    y=obcrank18(mark);
    else if(choice==3)
    y=scrank18(mark);
    else if(choice==4)
    y=strank18(mark);
    else if(choice==5)
    y=Pwdrank18(mark);
    cout<<y<<"\n";
    return y;
}

```

```

    }
    int callRankDb17(int mark,int choice)
    {
        Rank(mark,2017,choice);
        int y;
        if(choice==1)
            y=commonrank17(mark);
        else if(choice==2)
            y=obcrank17(mark);
        else if(choice==3)
            y=scrank17(mark);
        else if(choice==4)
            y=strank17(mark);
        else if(choice==5)
            y=Pwdrank17(mark);
        cout<<y<<"\n";
        return y;
    }
};
class Rank_Predictor
{
    public:
    int marks,max,choice;
    public:
    string predictor_response()
    {
        string viewers_report="O.K.";//fetched from interface if viewer does not
report
        return viewers_report;
    }

    void Predictor_output(string a)
    {
        cout<<"You choose to view "<<a<<" rank   for entered marks \n";
    }
    int Reset()
    {
        cout<<"You choose to reset marks \n";
        cout<<"enter marks\n";
        int m,cat;
        cin>>m;

        return m;
    }
};

```

```

}
void search()
{
cout<<"You wish to search colleges \n";
}
int callPredictor(int mark,int choice)
{
if(mark<0 || mark>360)
cout<<"Invalid marks\n";
switch(choice)
{
case 1:{
    Predictor_output("Common");
    return 1;
        break;
    }
case 2:{
    Predictor_output("OBC");
    return 1;
        break;
    }
case 3:{
    Predictor_output("SC");
    return 1;
        break;
    }
case 4:{
    Predictor_output("ST");
    return 1;
        break;
    }
case 5:{
    Predictor_output("PWD");
    return 1;
        break;
    }
case 6:{
    search();
    return 0;
        break;
    }
default:

```

```

        {
            cout<<"Invalid Choice";
            return 1;
        }
    }
}

void predict(string a,int c)
{
    cout<<"Predicting "<<a <<" for entered marks\n";
    cout<<"Marks sent to algorithm\n";
}

public:
string callPredictor1(int choice,int mark)
{
    //fetched data

    srand(time(0));
    switch(choice)
    {
    case 1:{
        predict("common_rank",mark);
        return "common rank";
        break;
    }
    case 2: {
        predict("OBC_rank",mark);
        return "OBC rank";
        break;
    }
    case 3: {
        predict("SC_rank",mark);
        return "SC rank";
        break;
    }
    case 4: {
        predict("ST_rank",mark);
        return "ST rank";
        break;
    }
    case 5: {
        predict("PWD_rank",mark);
        return "PWD rank";
        break;
    }
    }
}

```

```

        }
        case 6:{
search();

            break;
        }
        default:cout<<"Rank_Predictor found some internal Error..needs
maintainance";
        }
        }
        int mark()
        {
            return marks;
        }
    };

int main()
{
    Admin d;
    int level=d.askadmin();
    Student s;
    Rank_Predictor p;
    int mark=100;
    int choice=10;
    while(choice!=0 && choice!=100)
    {
        int pi;
        choice=s.choiceStudent();
        if(choice==6)
            pi=1;
        mark=s.callStudent();
        int u=p.callPredictor(mark,choice);
        if(u!=1)
        {
            if(u!=0)
                mark=u;
            choice=s.askcate();
            u=p.callPredictor(mark,choice);
        }
        Algorithm a;
        string predictor_choice=p.callPredictor1(choice,mark);
        int predictor_marks=p.mark();
        RankDb r;
    }
}

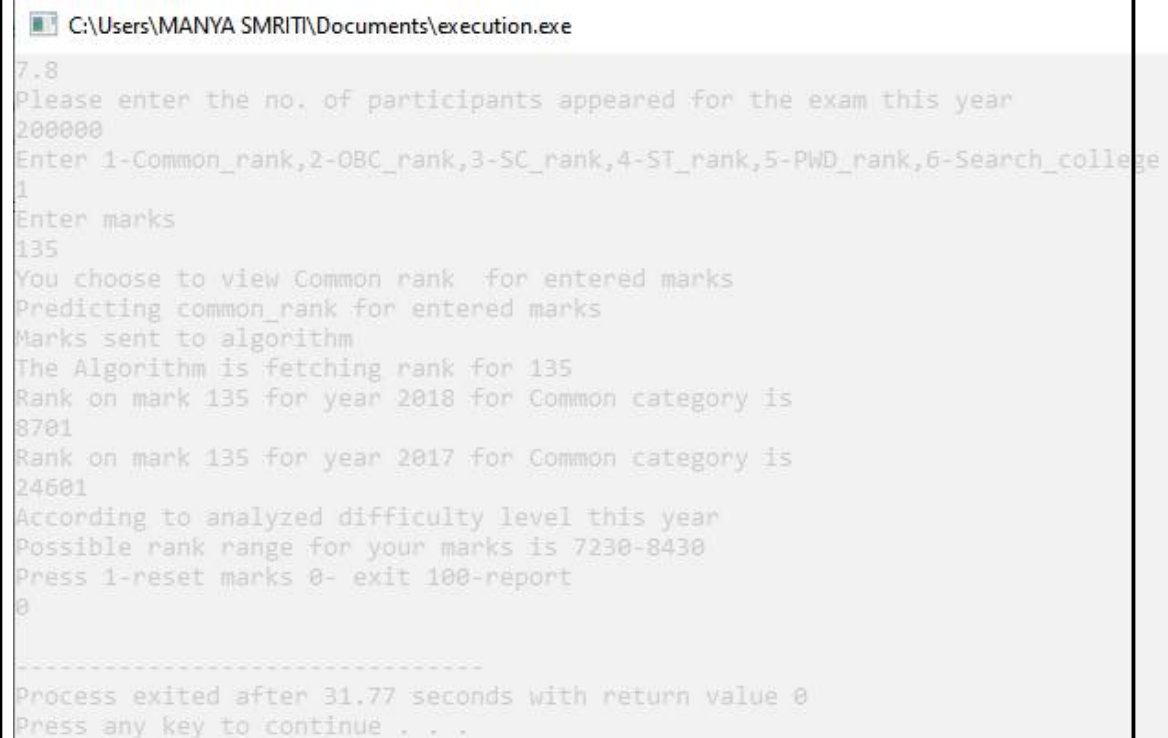
```

```

int rank18,rank17,getMarks;
string getChoice;
getMarks=a.call1Algorithm(mark);
rank18=r.callRankDb18(getMarks,choice);
rank17=r.callRankDb17(getMarks,choice);
int avg_rank=a.callMethod(level,rank18,rank17);
if(pi==1)
{
CollegeList c;
a.callAlgo_rank();
a.choice1();
a.callAlgorithm(avg_rank);
string College_name=c.college(avg_rank,choice);
a.dispCollege(avg_rank,College_name);
}
string response=p.predictor_response();
choice=s.askChoice();
if(choice==100)
d.admin_action("");
}
}

```

## OUTPUT



```

C:\Users\MANYA SMRITI\Documents\execution.exe
7.8
Please enter the no. of participants appeared for the exam this year
200000
Enter 1-Common_rank,2-OBC_rank,3-SC_rank,4-ST_rank,5-PWD_rank,6-Search_college
1
Enter marks
135
You choose to view Common rank for entered marks
Predicting common_rank for entered marks
Marks sent to algorithm
The Algorithm is fetching rank for 135
Rank on mark 135 for year 2018 for Common category is
8701
Rank on mark 135 for year 2017 for Common category is
24601
According to analyzed difficulty level this year
Possible rank range for your marks is 7230-8430
Press 1-reset marks 0- exit 100-report
0
-----
Process exited after 31.77 seconds with return value 0
Press any key to continue . . .

```

## Report

This paper deals with the JEE Advanced Rank Predictor modelling ,working and discusses how it can help student predict their rank. It predicts both Common and Category (OBC,SC,ST,PWD) rank. In addition,it also predicts the college one may get through the exam either through common rank or category rank. In what other ways rank predictor helps student is also discussed in the paper.Rank predictor uses what type of algorithm is also dealt and how the difficulty level factor changes the entire scenario. The limitation of the rank predictor is stated as well. The rank predictor also helps in searching college to help student know about the opening and closing rank of the college.

The objective is twofolded:

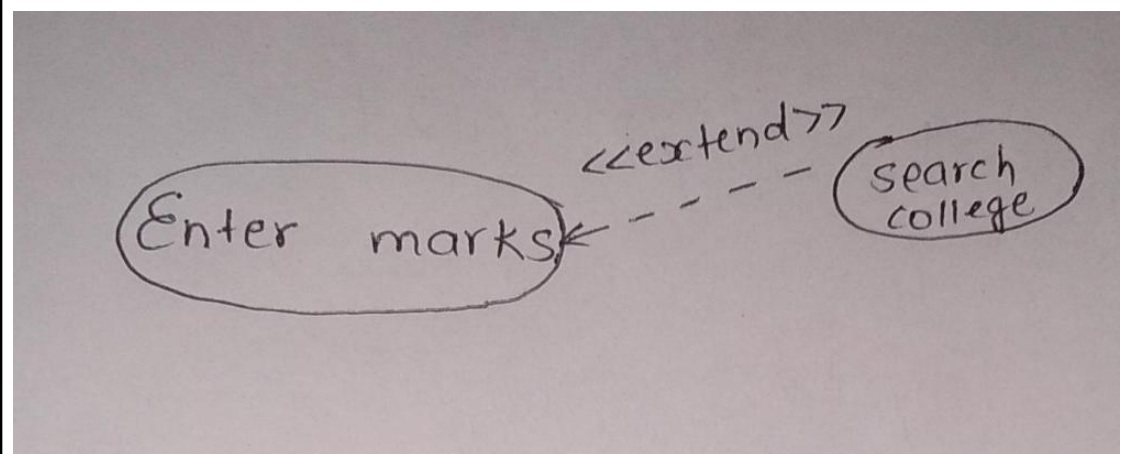
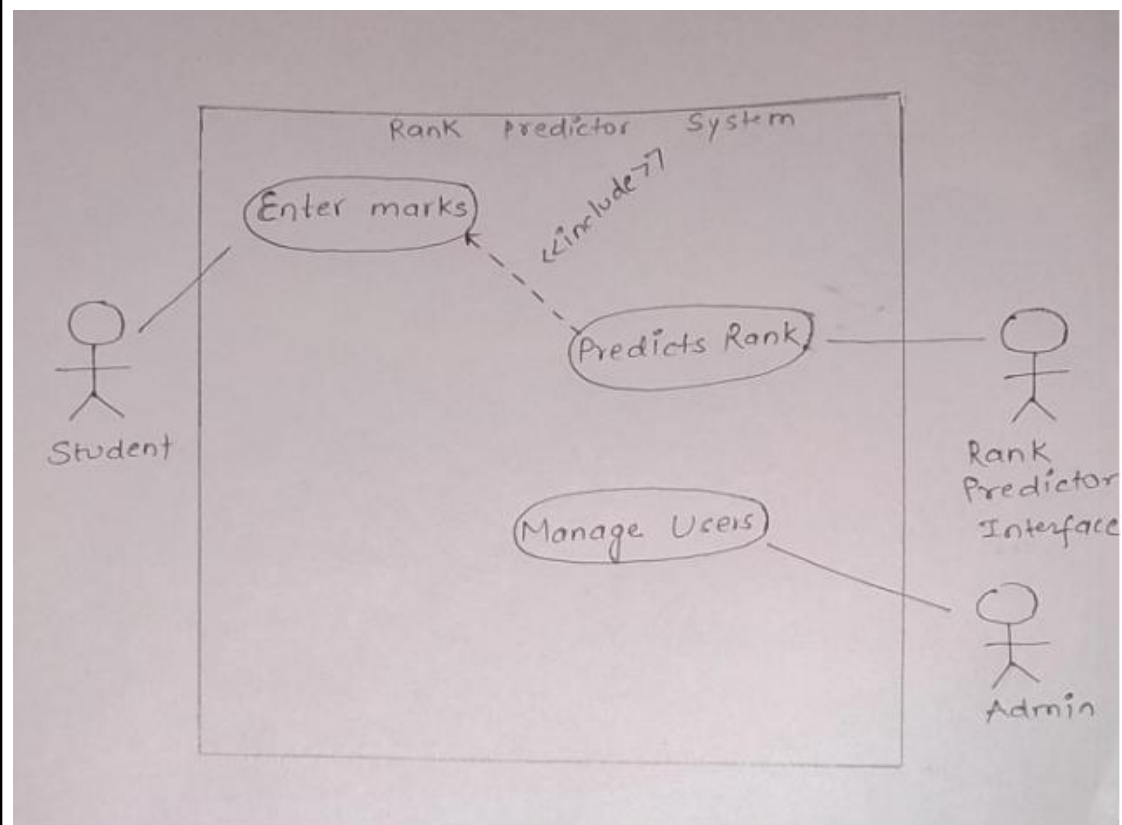
- i) Helps student predict the rank & college and
- ii) Let them know if they need to prepare themselves for other examination or attend counselling for other good colleges. The algorithm which predicts rank and how it ensures the accuracy and precision. Finally, how rank predictor notifies if it encounters any malfunctioning is also mentioned.

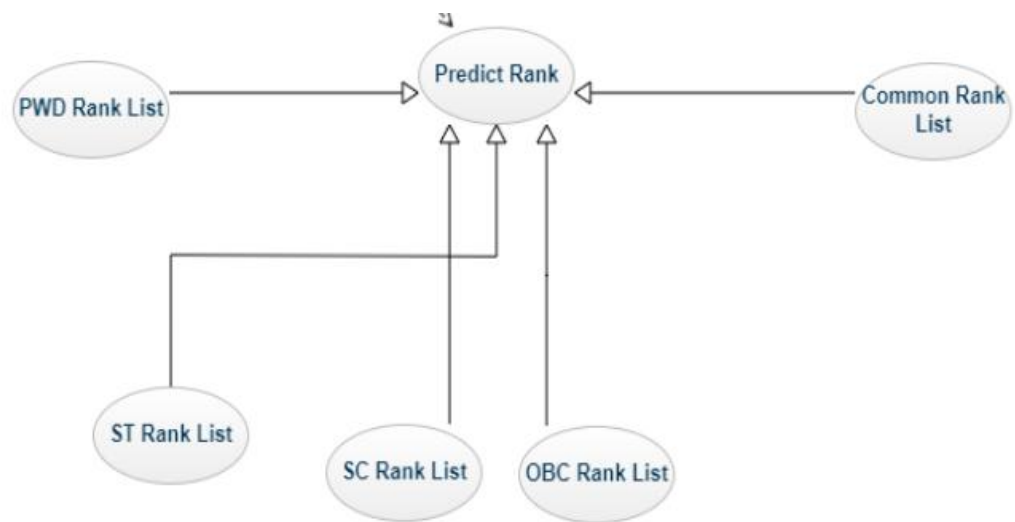
## Diagrams

**(1)**

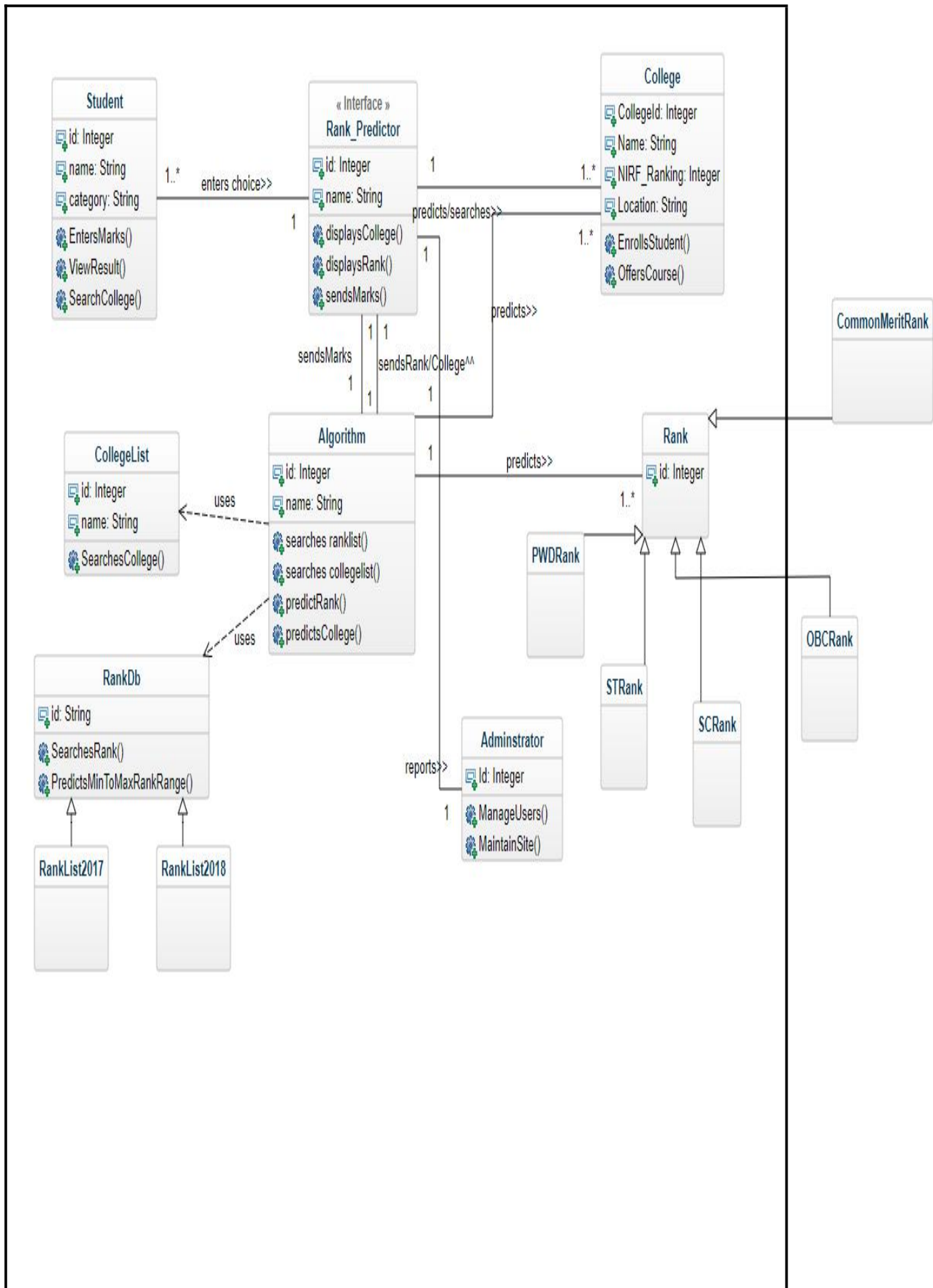
## Use Case Diagram



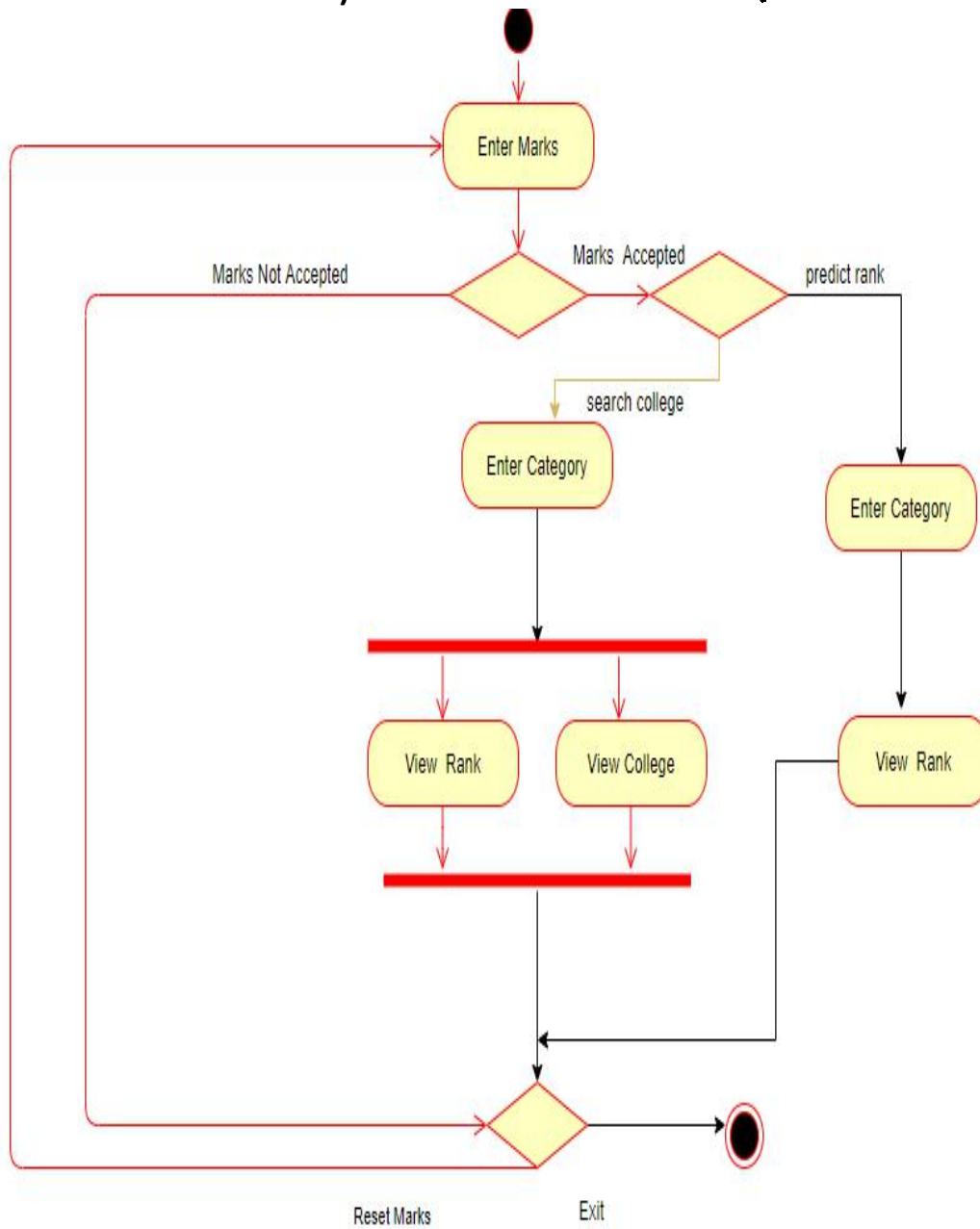




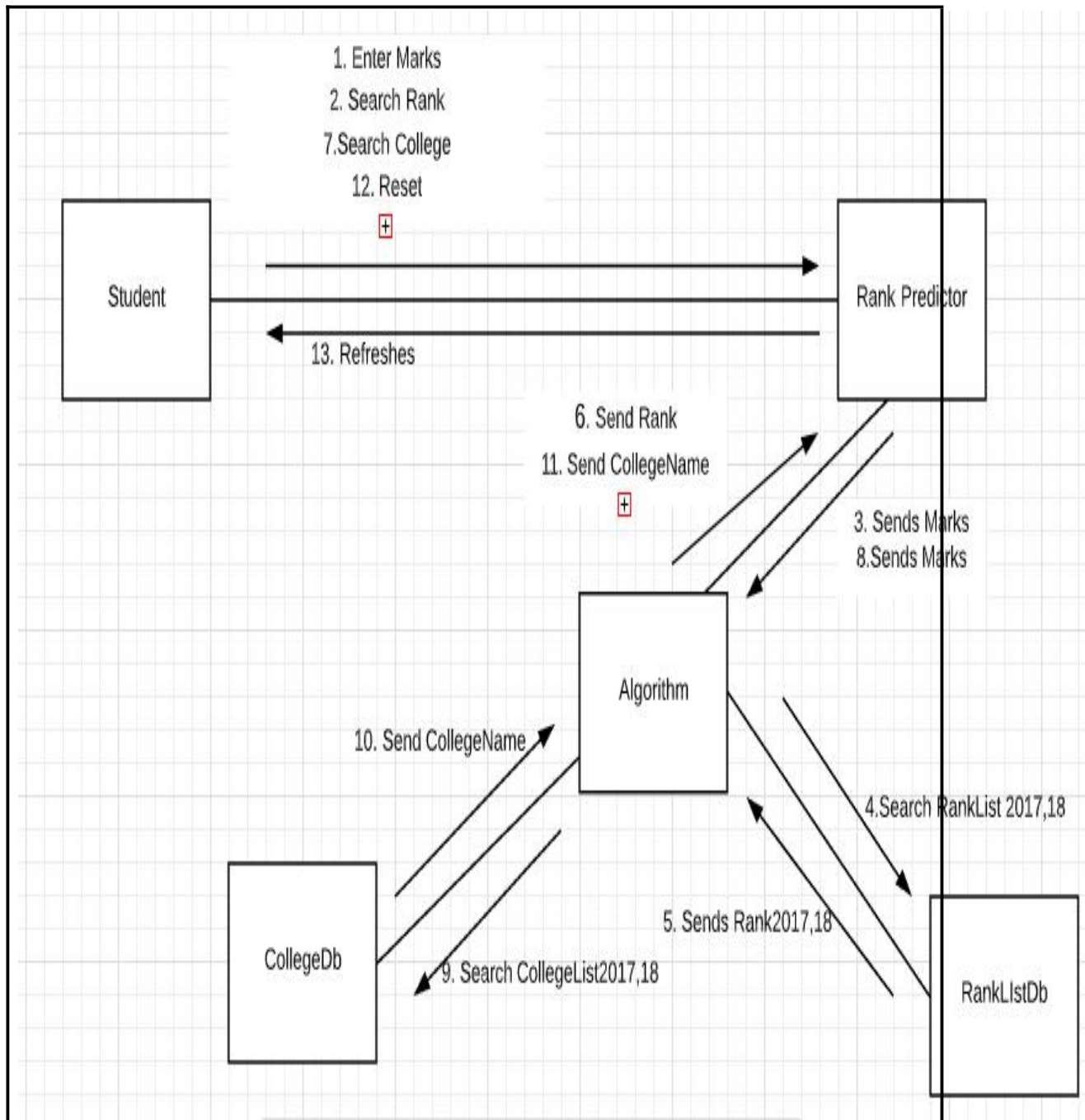
(2) Class Diagram



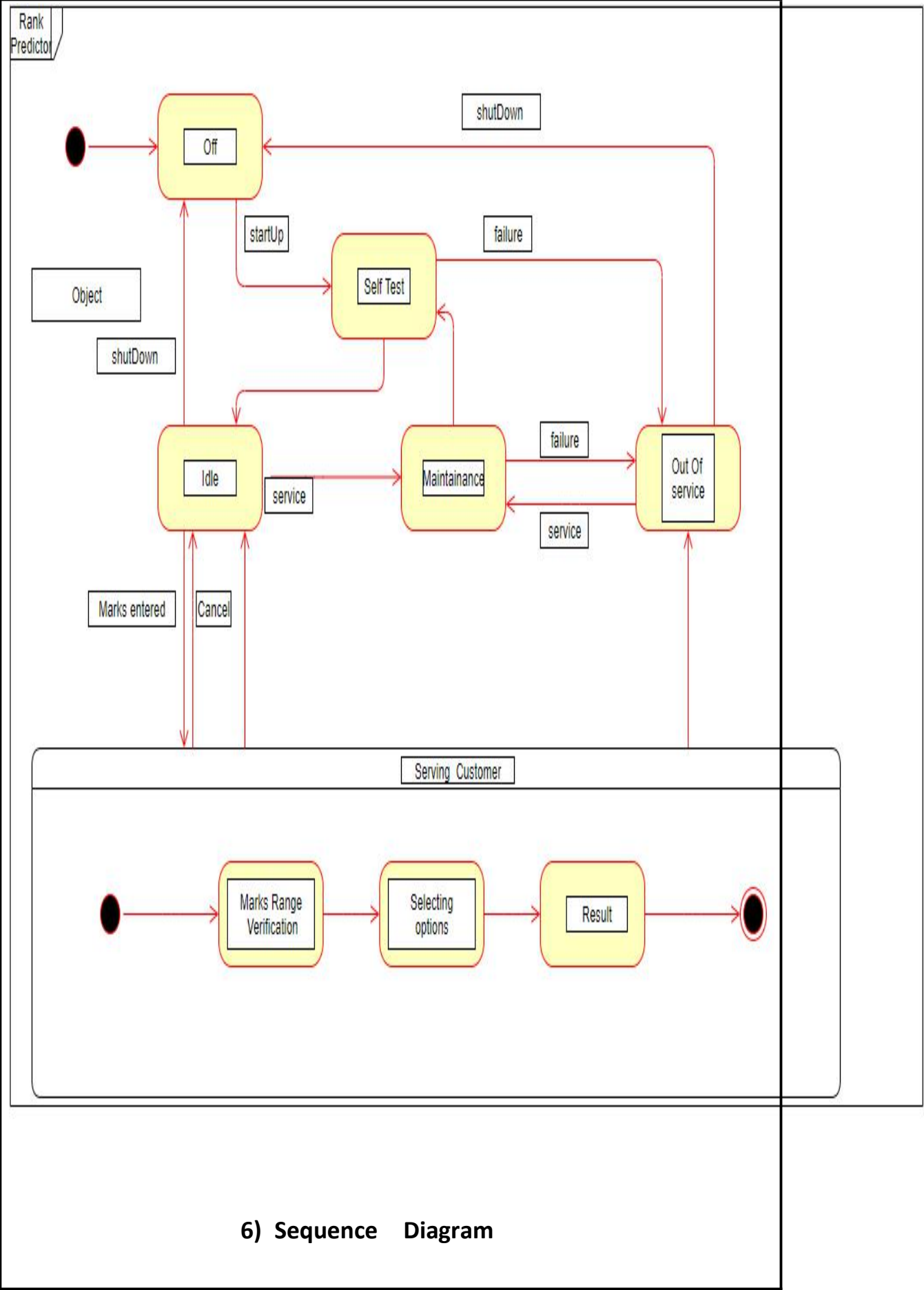
### 3) ACTIVITY DIAGRAM,



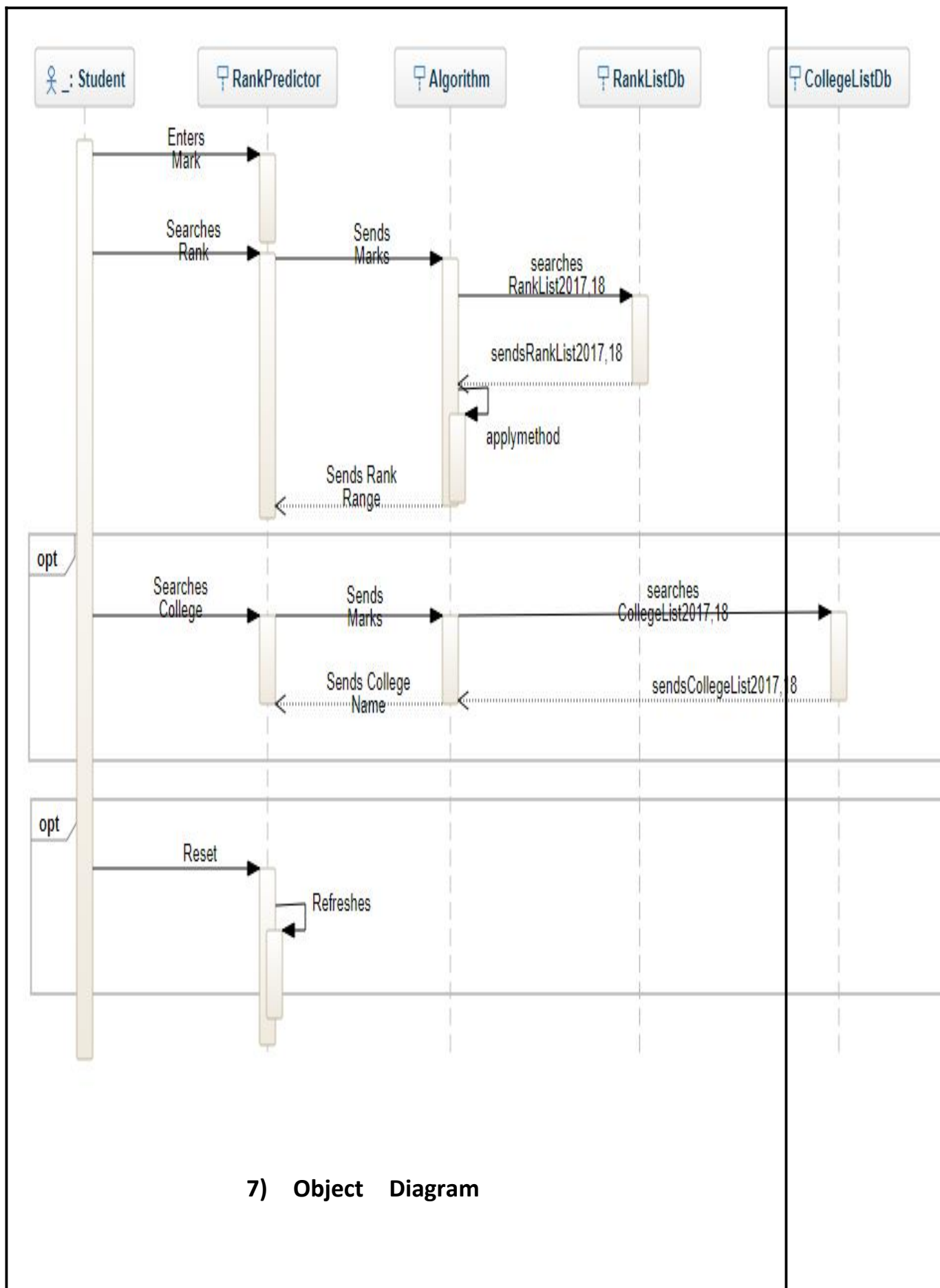
### 4) Collaboration Diagram



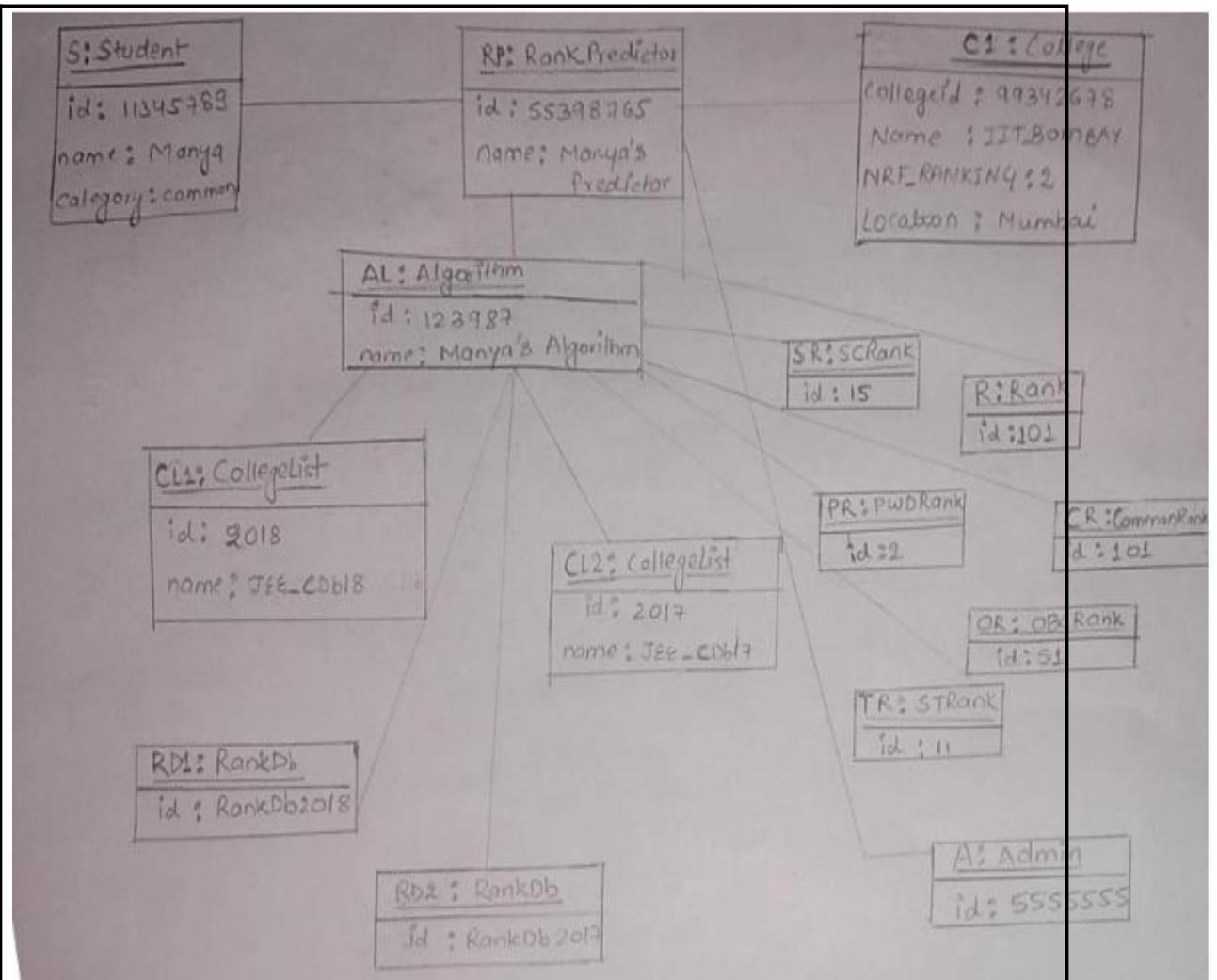
## 5) State Diagram



6) Sequence Diagram

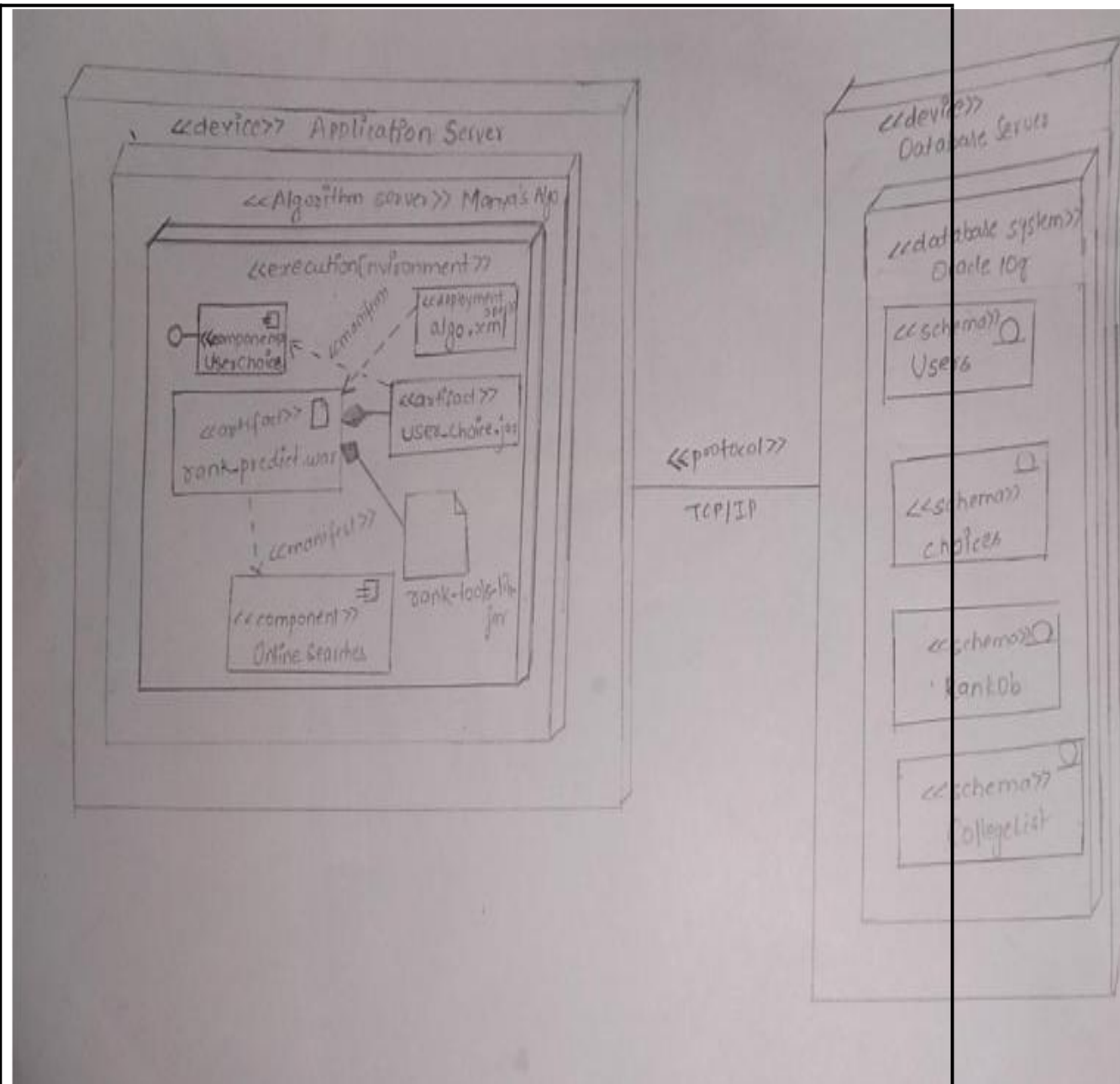


7) Object Diagram

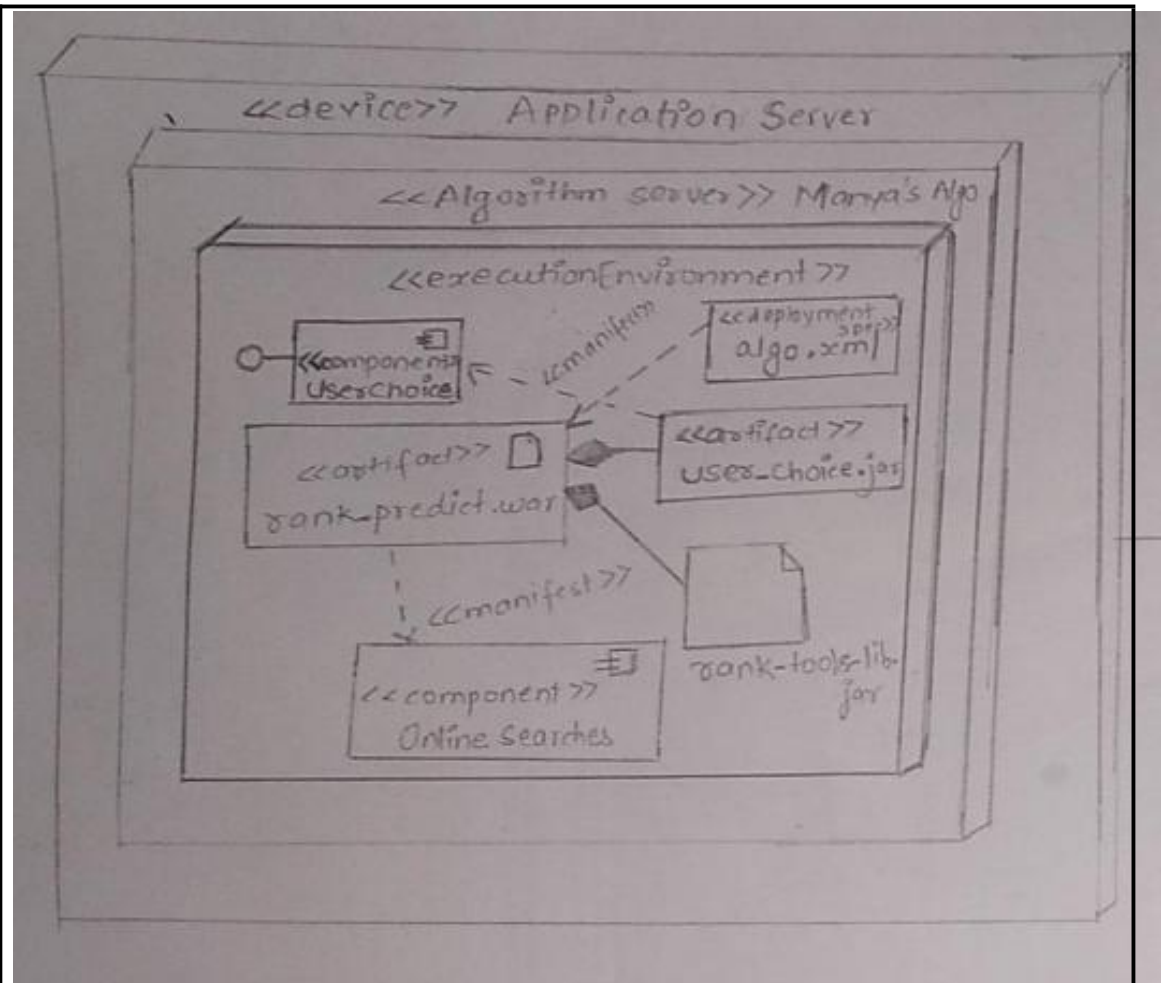


## 8) Deployment diagram

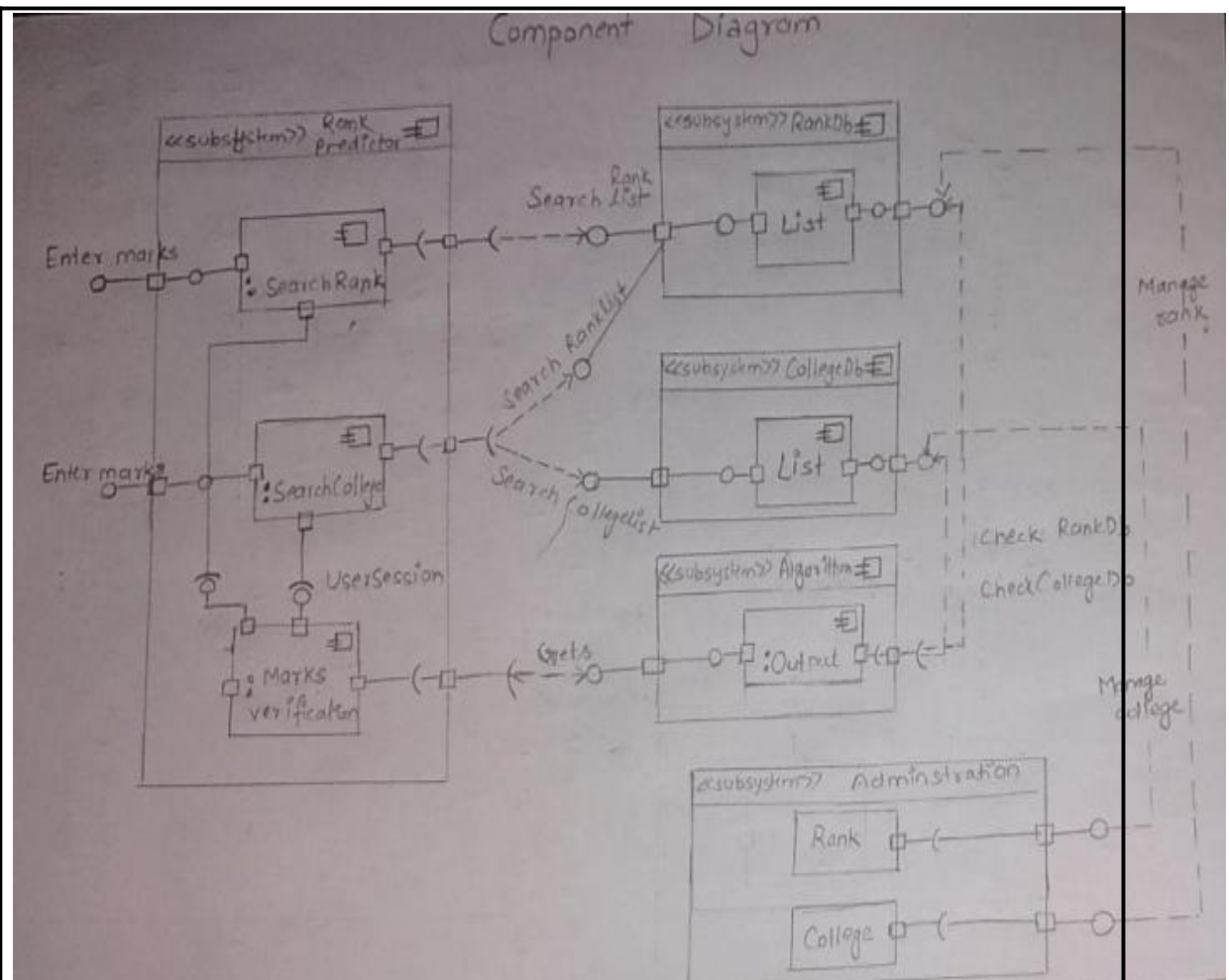




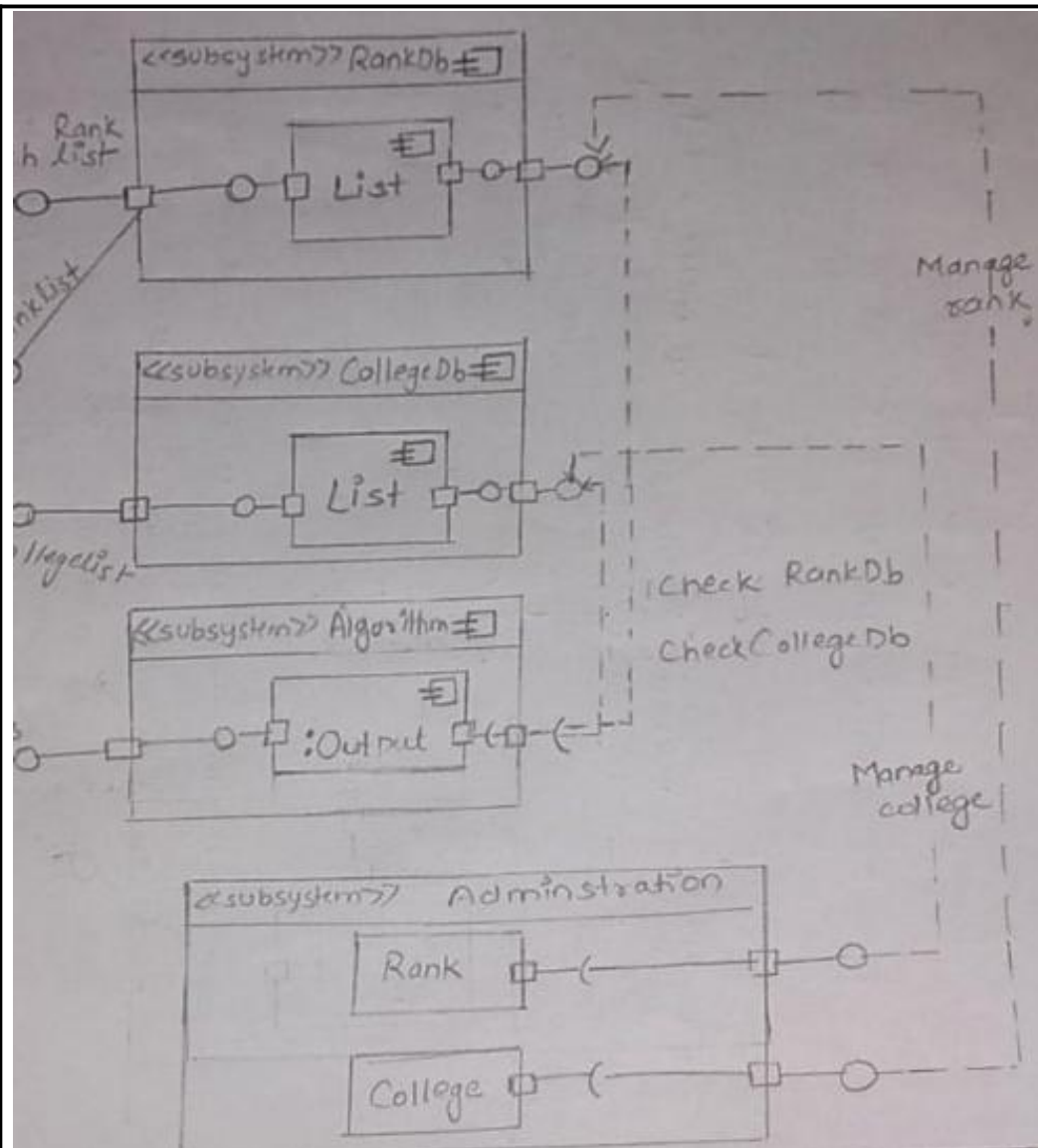
(for clarity)

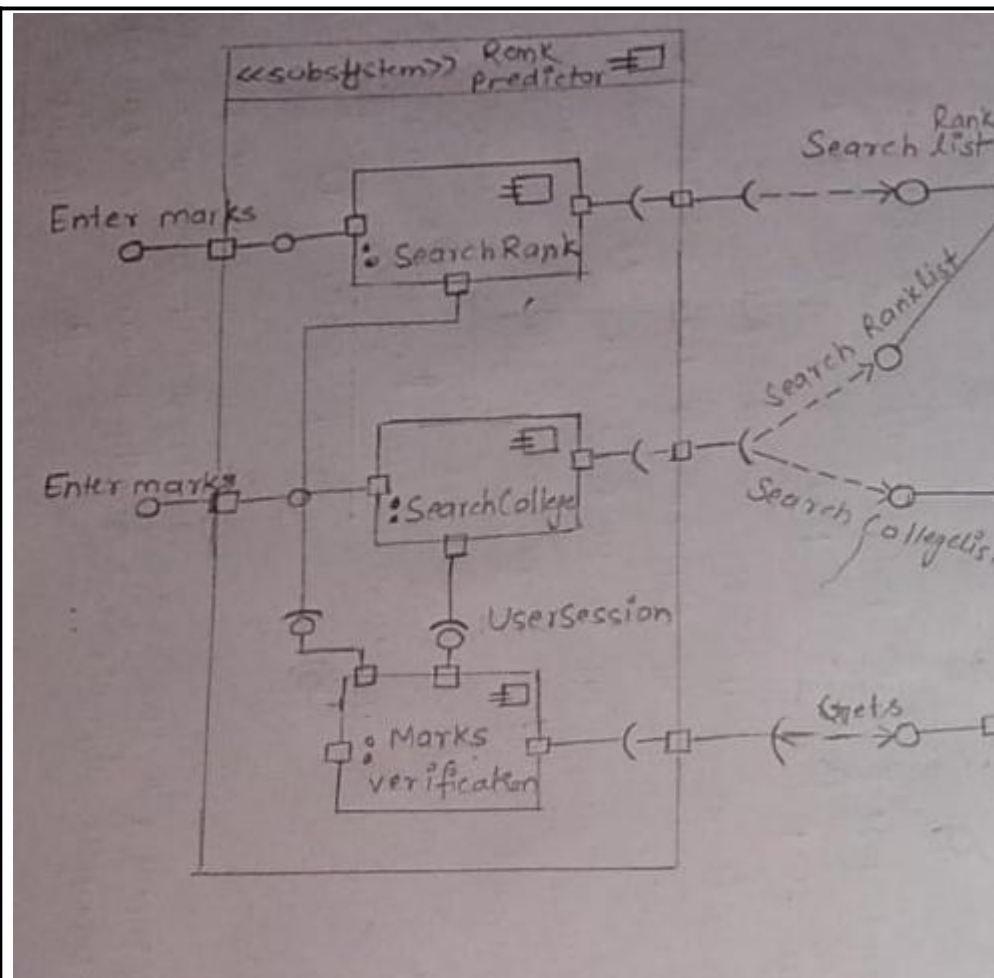


## 9) Component Diagram



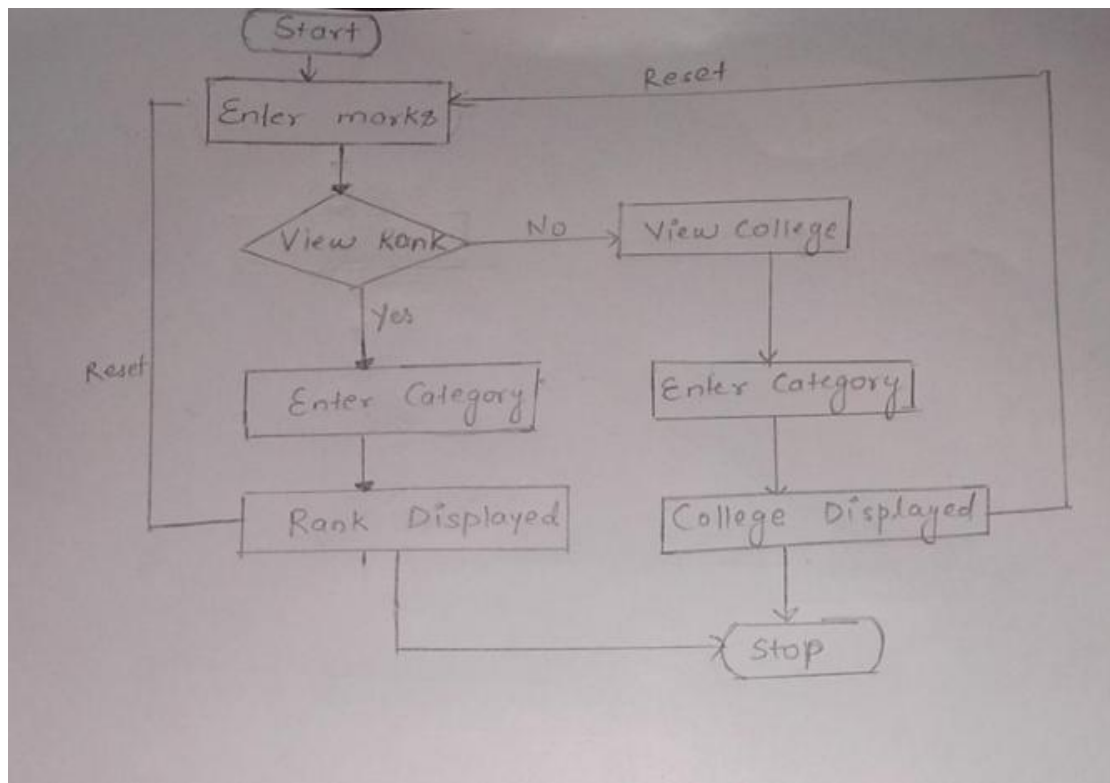
For Clarity



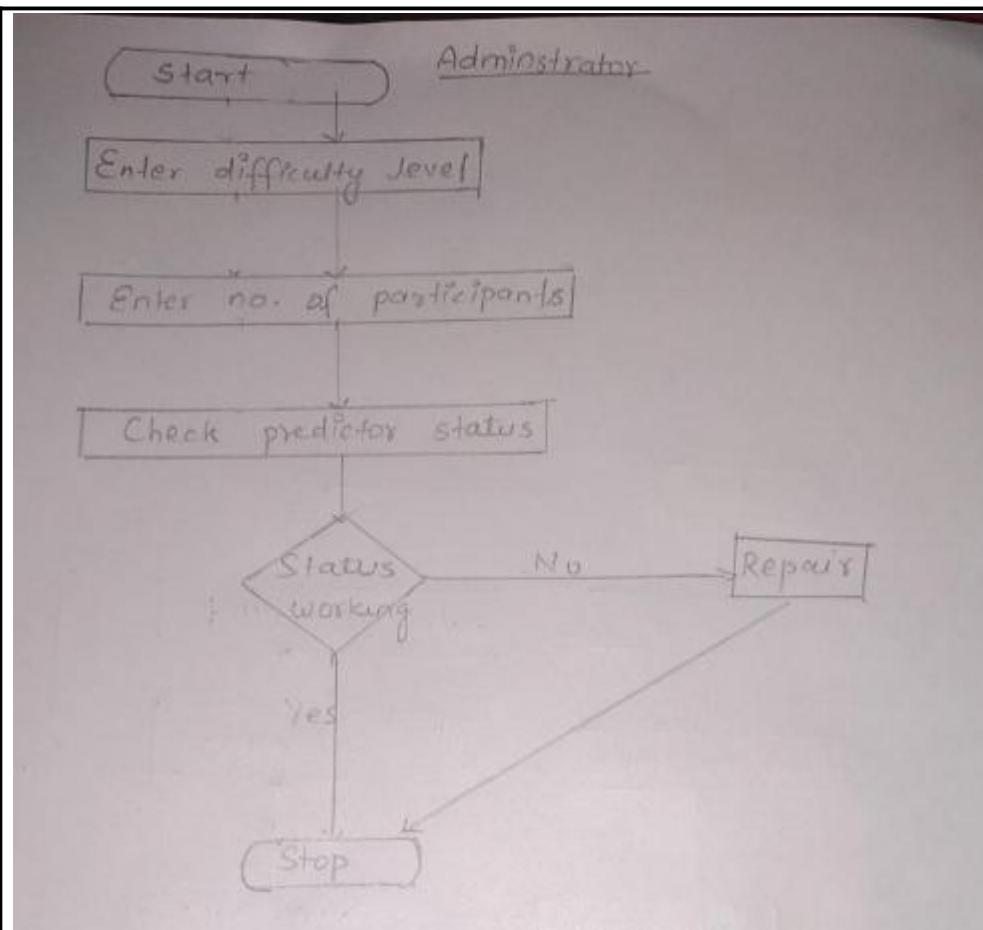


FlowChart

User



**Adminstrator**



Result:

## Results and Comparisons

I used the predictor to calculate ranking of JEE Advance Common Rank. Despite the fact the database is fed manually so obviously marks corresponding to every rank couldn't be fed, an interval of 100 rank was taken, but still it was precise in most of the cases about the rank range. Below are some of the results:

### General Category:

Marks	Predictor's Rank Range	Actual Rank	Shiksha's
Predictor			
251	193-233	217	1149- <b>1319</b> - 1489
153	5637-6837	5514	<b>17285</b>

201

536-1120

1220

5049-**5493**-5937

Even Quora has many cases of rank being predicted wrongly in range of 10000s. By different predictors.