Topic : Online Voting System for award Nomination Program

Group no : 3.02.03

Campus : Malabe

Submission Date : 19/05/2022

We declare that this is our own work, and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21244698	Thilakasiri P. K. V.C	0712906815
IT21245138	Gamaarachchi P.N	0766116892
IT21244766	Pathirana K.P. V	0711461016
IT21236228	Munaweera K.D.M.P	0710763213
IT21239366	Niwantha W.P.I	0773066724

List down the requirements you have identified, for the system you need to implement

1. As a guest user, he needs to register to become a voter/ nominee or Organizer by providing their details such as name, address, age, phone number.

2.If the guest user is not registered, they cannot access to the system.

3.As an Organizer, they can create voting programs by entering voting program name, voting program duration, and voting program category, and Nominee's Details.

4.System administrator can login to the system and edit/delete voting programs.

5.If any user violates rules, Admin can ban users from the system.

6.As a voter, they can search award nomination programs, view voting programs progress and can View nominee's details such as Name, Age etc.…

7.Also, Voter can select any specific nominee and cast one vote to selected nominee. (Each voter can cast only one vote)

8. Voter can share voting program details through any social media platforms.

9. once the voter cast their vote, they can give their feedbacks. As well as the users can view feedbacks and rate those feedbacks.

10. As a Nominee, they can search voting programs and view voting programs progress.

11.Election Officer can login to the system and generate reports (about voting programs, nominees, and voters).

12.Also, Election Officer Store details of the winner's and publish details of the winners.

Classes Identified

1. Voting program

   Assumption:

   if organizer creates voting programs in the system, we cannot get that as a system, because we cannot create any system, we can only implement the classes according to nouns. so, if organizer creates voting programs, voting program should not be a system.

2. Voter

3. Nominee

4. Organizer

5. Feedbacks

6. Reports

7. rule

CRC CARDS

Voting Program	
Responsibilities	**Collaboration**
Store Voting program Details	
Create voting programs	Nominee
Edit voting programs	
Delete voting programs	
Search voting programs	
View voting program progress	
Share voting program details	

Voter	
Responsibilities	**Collaboration**
Store Register details	
Ban user	

Nominee	
Responsibilities	**Collaboration**
Store Register details	
Ban user	
View nominee’s details	
Select specific nominee	
Cast vote	Voting program

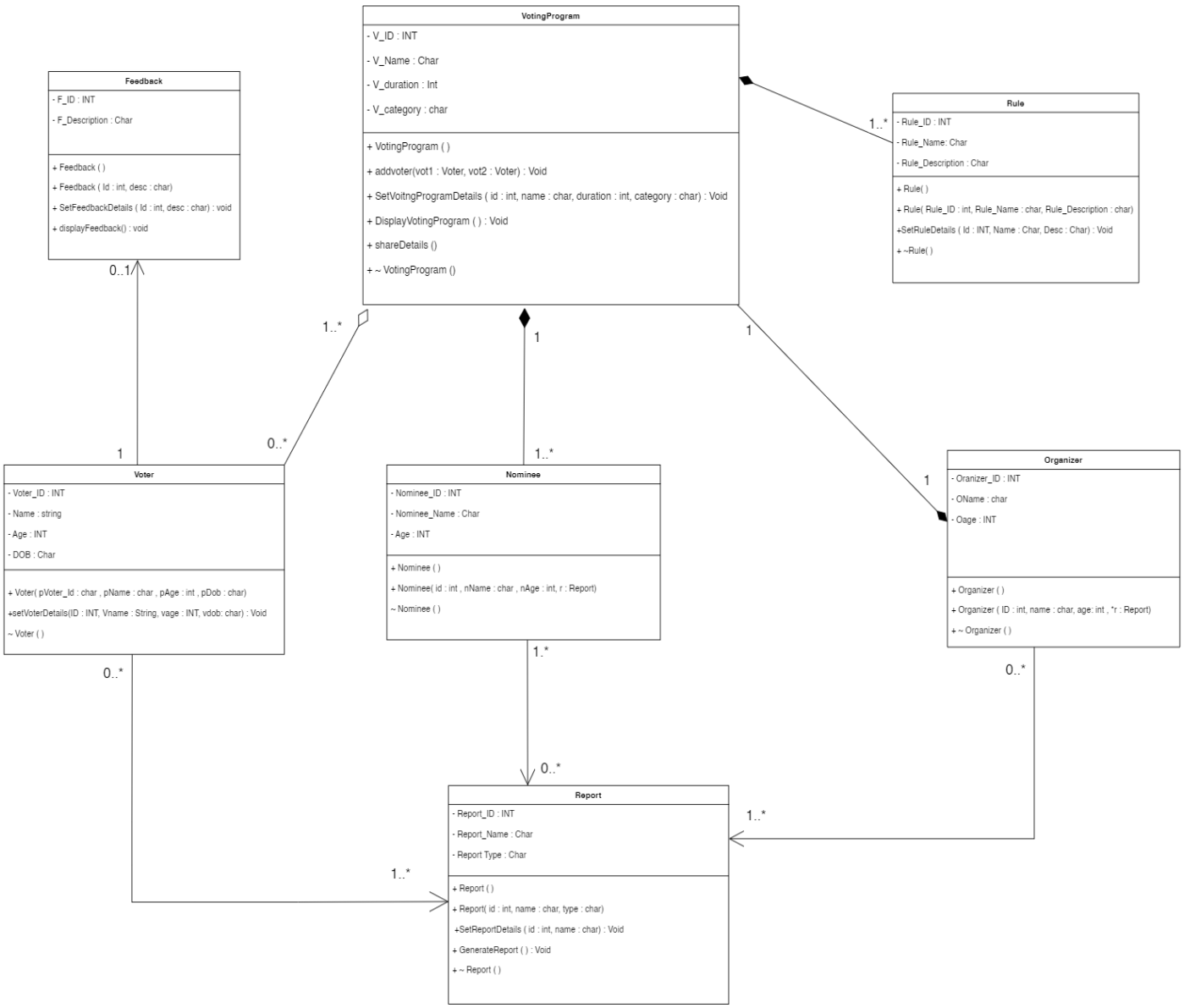
organizer	
Responsibilities	**Collaboration**
Store Register details	
Ban user	

Reports	
Responsibilities	**Collaboration**
Generate reports	Voter, Nominee, Voting Programs

Rule	
Responsibilities	**Collaboration**
Store rules details	
Update Rules	
Violate Rules	

Feedbacks	
Responsibilities	**Collaboration**
Store Feedbacks	
Give feedbacks	
View feedbacks	
Rate feedbacks	

Class Diagram



Voter Feedback Association

```
#include <iostream>
#include <cstring>
using namespace std;



class Feedback
{
        private:
                int F_ID;
                char F_Description[20];
        public:
                Feedback();

                 Feedback(int id, const char desc[])

                {
                        F_ID = id ;
                   strcpy(F_Description,desc);
                }

                void setFeedbackdetails(int id, const char desc[]);

                void displayFeedback()
                {
                        cout << " Feedback ID  = " << F_ID << endl;
                        cout << "Feedback  = " << F_Description << endl;
```

```
                }
};



class Voter
{
  private:
        int Voter_ID;
        string Name;
        int Age;
        char DOB[20];
        Feedback *fd;

  public:
   Voter (int pVoter_id, string pName, int page, const char pDob[], Feedback *f)
        {
                Voter_ID = pVoter_id;
                Name = pName;
                Age = page;
                strcpy(DOB,pDob);
                fd = f;
        }

        void setVoterdetails(int id, string vname, int vage, const char vdob[]);

        void displayVoter()
        {
                cout << "voterID = " << Voter_ID << endl;
```

```
                cout << "voter Name = " << Name << endl;
                cout << "voter Age = " << Age << endl;
                cout << " Date Of Birth = " << DOB << endl;
                fd->displayFeedback();
        }

  ~Voter(){
    cout<<"Voter deleted"<<"endl";
  };

};

int main()
{
        char ch;
        Feedback *f = new Feedback(001, "great experience");
        f->displayFeedback();
        cout << " *************************" << endl;
        Voter *v = new Voter(29, "Kamal", 21, "2001.02.02", f);
        v->displayVoter();
        delete v;
        delete f;
        cin >> ch;
        return 0;
}
```

Voter Report Association

```
#include <iostream>
#include <cstring>
using namespace std;


class Report
{
	private:
		int Report_ID;
		char Report_Name[20];
		char Report_Type[20];
	public:
		Report();

		Report(int id, const char name[], const char type[])
		{
			Report_ID = id ;
		   strcpy(Report_Name,name);
		   strcpy(Report_Type,type);
		}

		 void setReportdetails(int id, char name, char type);

		void generateReport()
		{
			cout << " Report ID  = " << Report_ID << endl;
```

```
                        cout << "Report Name  = " << Report_Name << endl;
                        cout << "Report type =          " << Report_Type << endl;
                                }

   ~Report(){
     cout << "Report deleted" << endl;
   }
};

class Voter
{
  private:

int Voter_ID;
string Name;
int Age;
char DOB[20];
  Report *re;

  public:
   Voter (int pVoter_id, string pName, int pAge, const char pDob[], Report *r)
          {
                  Voter_ID = pVoter_id;
                  Name = pName;
                  Age = pAge;
                  strcpy(DOB,pDob);
                  re = r;
          }
```

```
    void setVoterdetails( int ID, string vname, int vage, const char vdob[]);

        void displayVoter()
        {
            cout << "voterID = " << Voter_ID << endl;
            cout << "voter Name = " << Name << endl;
            cout << "voter Age = " << Age << endl;
            cout << " Date Of Birth = " << DOB << endl;
            re->generateReport();
        }
};

int main()
{
        char ch;
        Report *r = new Report(005, "Voter Registration", "Voter details");
        r->generateReport();
        cout << " **************************" << endl;
        Voter *v = new Voter(52, "Devin", 21, "2001.05.07", r);
        v->displayVoter();
        delete v;
        delete r;
        cin >> ch;
        return 0;
}
```

Voter Voting Program Aggregation

```
#include<iostream>
#include<cstring>

#define SIZE 2

using namespace std;



//part class
class Voter
{
	private:

		int voter_Id;
		char name[20];
		int age;
		char dob[10];

	public :
		Voter(int pVoter_Id,const char pName[],int pAge,const char pDob[]){
			voter_Id = pVoter_Id;
			strcpy(name,pName);
			age = pAge;
			strcpy(dob,pDob);
		};
		void setVoterDetails(int id , string vName , int vAge,char vDob);
```

```
        void displayVoter(){
            cout<< "Voter ID : "<< voter_Id <<endl;
            cout<< "Name : " << name <<endl;
            cout<< "Age : " << age <<endl;
            cout<<"DOB : "          << dob << endl;
            cout << "************************" << endl;
        }
        int getVoterid();
        ~Voter(){
            cout<<"Deleting voter"<<endl;
        }

};

//whole class
class VotingProgram {

    private:
        int V_ID;
        char V_Name[20];
        int V_duration;
        char V_category[50];
        Voter *vot[SIZE];

    public:
        votingProgram();
```

```
        void addvoter(Voter*vot1, Voter*vot2){
            vot[0] = vot1;
            vot[1] = vot2;
        }

        void SetVotingProgramDetails(int id, const char name[], int duration, const
char category[]);

        void DisplayVotingProgram(){
            for(int i=0; i<SIZE; i++)
            vot[i]->displayVoter();
        }

        void shareDetails();

        ~VotingProgram(){
            cout << "voting program delete"<<endl;

        }

};

int main()
{
    Voter *vot1 = new Voter(021 , "Vidura" , 21,"2001.03.30");
    Voter *vot2 = new Voter(022 , "Madushi" ,21,"2001.11.03");

    VotingProgram *vp1 = new VotingProgram();
```

```
    vp1->addvoter(vot1, vot2);
    vp1->DisplayVotingProgram();

    vot2->displayVoter();
    vot1->displayVoter();

    delete vot1;
    delete vot2;
   return 0;

}
```

Nominee Report Association

```
#include<iostream>
#include<cstring>

using namespace std;

class Report{
	private:
		int Report_ID;
		char Report_Name[20];
		char Report_Type[10];

	public:
		Report();

		Report(int id, const char name[], const char type[]){
			Report_ID = id;
			strcpy(Report_Name, name);
			strcpy(Report_Type, type);
		};

		void SetReportDetails(int vid, int nid, const char type[] );

		void GenerateReport();

		~Report(){
			cout<<"Report delete"<<endl;
```

```
                    };
};
class Nominee
{
        private :
                    int nominee_Id;
                    char nominee_Name[50];
                    int age;
   Report *rpt;

        public :
                    Nominee(int id ,const char nName[] , int nAge, Report *r)

                    {
                                nominee_Id = id;
                                strcpy(nominee_Name,nName);
                                age = nAge;
         rpt = r;
                    };
                    ~Nominee()
                    {
                                cout<<"Nominee Deleted : "<<nominee_Id<<endl;
                    }
};
nt main(void){

         Report *r = new Report(1001, "Nominee Information", "nominee");
```

```
        Nominee *n = new Nominee(101, "Jananja osham", 21, r);

        delete r;

        delete n;

        return 0;

}
```

Organizer Report Association

```
#include<iostream>
#include<cstring>

using namespace std;

class Report{
	private:
		int Report_ID;
		char Report_Name[20];
		char Report_Type[10];

	public:
		Report();

		Report(int id, const char name[], const char type[]){
			Report_ID = id;
			strcpy(Report_Name, name);
			strcpy(Report_Type, type);
		};

		void SetReportDetails(int vid, int nid, const char type[] );

		void GenerateReport();

		~Report(){
			cout<<"Report delete"<<endl;
```

```
        };
};

class Organizer{
    private:
        int Organizer_ID;
        char Oname[20];
        int Oage;
        Report *rpt;

    public:
  Organizer();

        Organizer(int ID, const char name[], int age, Report *r){
            Organizer_ID = ID;
            strcpy(Oname, name);
            Oage = age;
            rpt = r;

        };

        ~Organizer(){
            cout<<"Organizer delete"<<endl;
        };
};

int main(void){
```

```
        Report *r = new Report(1001, "Voter Information", "organizer");

        Organizer *o = new Organizer(101, "Beesara Sahan", 21, r);

        delete r;

        delete o;

        return 0;

}
```

Organizer Voting Program Composition

```
#include<iostream>
#include<cstring>

using namespace std;
//part class
class votingProgram{

    private:
        int V_ID;
        char V_Name[50];
        int duration;
        char V_category[50];

    public:
        votingProgram();

        votingProgram(int id, const char name[], int Vduration, const char vcat[]){

            V_ID = id;
            strcpy(V_Name, name);
            duration = Vduration;
            strcpy(V_category, vcat);
        };

        void SetVotingProgramDetails(int id, const char name[], int duration, const
char category[]);
```

```
        void DisplayVotingProgram();

        void shareDetails();

        ~votingProgram(){
            cout<<"Delete Voting Program no: " << V_ID << endl;
        };

};

//whole class
class Organizer{
    private:
        int Organizer_ID;
        char Oname[20];
        int Oage;
        Report *rpt;
        votingProgram *vot[3];

    public:
        Organizer(){
            vot[0] = new votingProgram(2001, "Best Singer award nomination",
25, "Music");
            vot[1] = new votingProgram(2002, "Best Actor award nomination",
24, "Drama");
```

```
                vot[2] = new votingProgram(2003, "Best inovator in year", 15,
"science");

        };

    Organizer(int ID, const char name[], int age);

        ~Organizer(){
            cout<<"Organizer deleted: " <<endl;
            for(int i = 0; i < 3; i++){
                delete vot[i];
            }
        };
};

int main(void){

    Organizer *org1 = new Organizer();

    delete org1;
}
```

Voting Program Rule Composition

```
#include<iostream>
#include<cstring>

using namespace std;

class Rule{
	private:
		int Rule_ID;
		char Rule_Name[50];
		char Rule_Description[60];

	public:
		Rule();

		Rule(int id, const char rname[], const char rdes[]){
			Rule_ID = id;
			strcpy(Rule_Name, rname);
			strcpy(Rule_Description, rdes);
		};

		void SetRuleDetails(int rid,  const char rname[], const char rdes[]);

		~Rule(){
			cout<<"Rule Deleted : "<< Rule_ID <<endl;
		};
};
```

```
class votingProgram{

    private:
        int V_ID;
        char V_Name[20];
        int duration;
        char V_category[50];
        Rule *rule[3];

    public:
        votingProgram(){
            rule[0] = new Rule(1000, "voting Rule", "Voter should cast a vote at a
time");
            rule[1] = new Rule(1001, "Organizer Rule","Organize can make only
one voting program");
            rule[2] = new Rule(1002, "User Ban Rule", "If any user violate those
rules system admin can ban users anytime");
        };

        void SetVotingProgramDetails(int id, const char name[], int duration, const
char category[]);

        void DisplayVotingProgram();

        void shareDetails();

        ~votingProgram(){
            cout << "Shutting down voting program"<<endl;
```

```
                for(int i = 0; i < 3; i++){
                        delete rule[i];
                }
        };

};

int main(void){

        votingProgram *v1 = new votingProgram();

        delete v1;

        return 0;
}
```

Voting Program Nominee Composition

```
#include<iostream>
#include<cstring>

using namespace std;

//Nominee

class Nominee
{
        private :
                int nominee_Id;
                char nominee_Name[50];
                int age;

        public :

   Nominee();

                Nominee(int id ,const char nName[] , int nAge)

                {
                        nominee_Id = id;
                        strcpy(nominee_Name,nName);
                        age = nAge;
                }
                ~Nominee()
```

```
		{
			cout<<"Nominee Deleted : "<<nominee_Id<<endl;
		}

};
class votingProgram{

	private:
		int V_ID;
		char V_Name[20];
		int duration;
		char V_category[50];
		Nominee *nominee[3];

	public:
		votingProgram(){
			nominee[0] = new Nominee(001 , "Kumara Sangakkara" , 55);
			nominee[1] = new Nominee(002 , "Mahela Jayawardhana", 48);
			nominee[2] = new Nominee(003 , "Lasith Malinga" , 40);
		};

		void SetVotingProgramDetails(int id, const char name[], int duration, const
char category[]);

		void DisplayVotingProgram();

		void shareDetails();

		~votingProgram(){
```

```
                cout << "Shutting down voting program"<<endl;

                for(int x = 0; x < 3; x++){
                        delete nominee[x];
                }
        }

};

int main(void){

        votingProgram *voting1 = new votingProgram();

        delete voting1;

        return 0;
}
```