Web Engineering-SE-IT-F21M
Assignment 1

Voting Machine Interface with Serialization and Database Integration

Important Instructions:

- Proper indentation,clean and tidy code ,make sure classes are in distributed files.
- Ensure all your functions are in running state.
- There should not be any syntax error in your code.
- Handle all the necessary exceptions and validation checks carefully in each function,otherwise will grade zero in that function.
- Creating logic instead of built-in functions are highly recommended.
- In case of any cheating and pledge,student will grade zero.

Submission :

Create a zip file containing all necessary classes and program.cs , named it as e.g : BITF21M001
Assignment-01 as your roll number.

You must have to submit your assignment till Friday 11:59pm pkt.

Contents and Objectives:

- Design and implement a simple voting machine interface.
- Demonstrate the use of classes, data members, functions, object serialization, ADO.NET for database operations, byte-oriented stream, character-oriented stream, object initializer syntax, params keyword, ToString override, and base object class.

Tools and Technologies:

- Visual Studio (or any preferred IDE for C# development) .Net
- SQL Server (or any preferred relational database by your instructor)
- C# programming language

<u>Interface:</u>

Class Candidate

- **Private Int CandidateID**:Generate this id using **rand function** for each new candid
- **Private String Name**:Take input Name from Candidate.
- **Private String Party**:Take input (party name)from candidate.Party-Name must be unique,
  as castVote() cast the vote on the basis of selected party name.
- **Private Int votes**:Just an incrementer, increased by one each time voter caste the vote.
- **Private int GenerateCandidateID()**:This function will generate ID using rand function
  and returns id.
- **Public Candidate(string name, string party)**:Assign properties in parameters , assign id using GenerateCandidateID(),Initialize votes with 0.
- **Public int CandidateID**:Create just getter of CandidateID
- **Public string Name**: Create getterand setter for Name.
- **Public string Party**: Create getter and setter for Party
- **Public int Votes**: Create just getter for Votes.
- **Public void IncrementVotes()**:Increment number of votes by 1.
- **ToString Override:**Override the ToString method in the Candidate class to provide a custom string representation of a candidate.

Class Voter

- **Private string VoterName**
- **Private String cnic**
- **Private String selectedPartyName**
- **Public Voter**(string VoterName,string cnic,string selectedPartyName)
- **Public String selectedPartyName**:Create only getter for selectedPartyName.
- **Public bool hasVoted(String cnic):**Check that if this voter already caste the vote, then return true else false

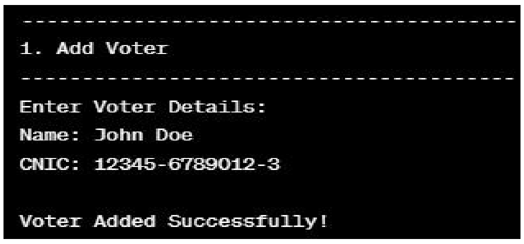
Class VotingMachine

private List<Candidate> candidates

Public VotingMachine()

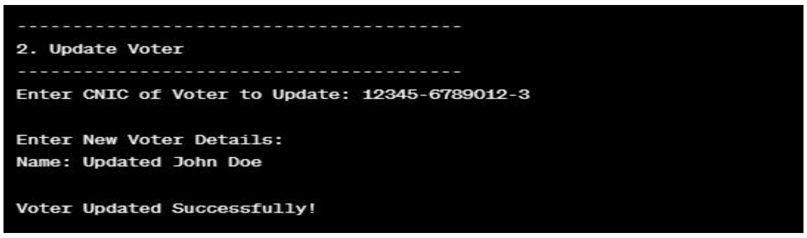
castVote(Candidate c,Voter v): check if this voter has not already casted the vote, if not ,then cast his vote.

- **Public addVoter() :** take voter details as input,store it in your Database as well as in File system.



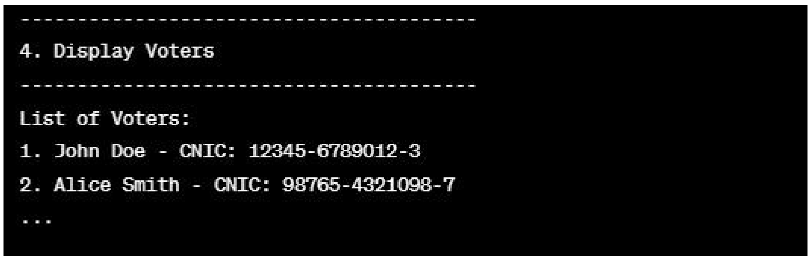
```
----------------------------------------
1. Add Voter
----------------------------------------
Enter Voter Details:
Name: John Doe
CNIC: 12345-6789012-3

Voter Added Successfully!
```

- **Public updateVoter(string cnic):** update details on the basis of cnic in your database as well as in your file system.



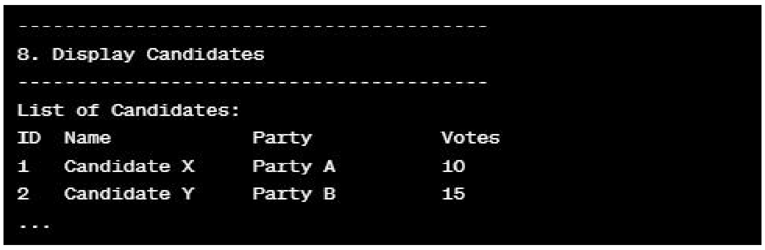
```
----------------------------------------
2. Update Voter
----------------------------------------
Enter CNIC of Voter to Update: 12345-6789012-3

Enter New Voter Details:
Name: Updated John Doe

Voter Updated Successfully!
```

- **Public displayVoters():** display details of all voters on the screen.



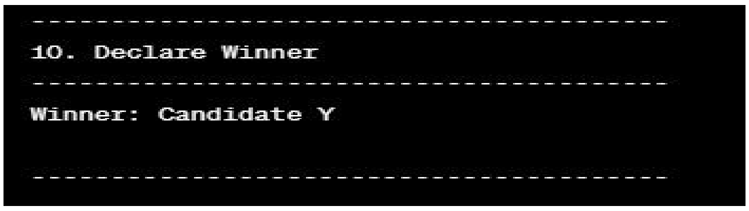
```
----------------------------------------
4. Display Voters
----------------------------------------
List of Voters:
1. John Doe - CNIC: 12345-6789012-3
2. Alice Smith - CNIC: 98765-4321098-7
...
```

- **Void displayCandidates()**: display all candidates with their names,partyname,votes



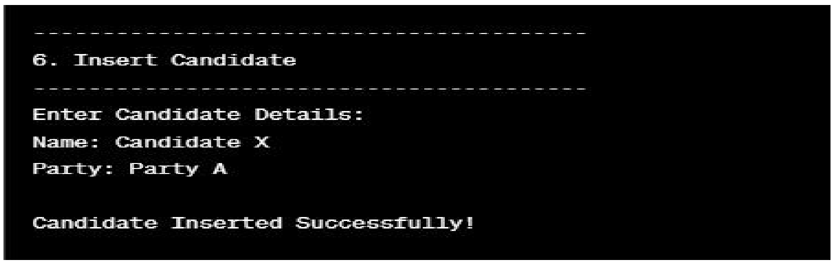
```
----------------------------------------
8. Display Candidates
----------------------------------------
List of Candidates:
ID  Name          Party          Votes
1   Candidate X   Party A        10
2   Candidate Y   Party B        15
...
```

- **Declarewinner():** display name and details of winner candidate.



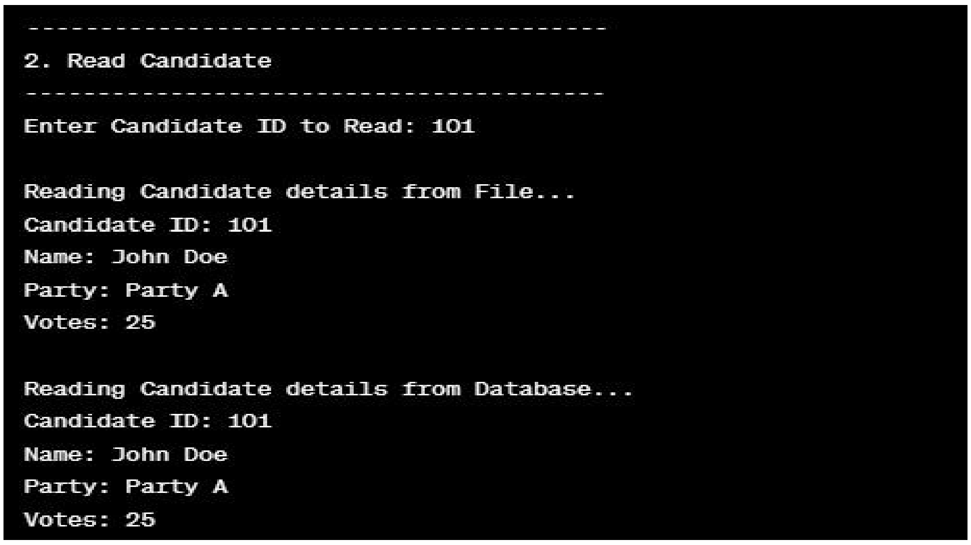
```
----------------------------------------
10. Declare Winner
----------------------------------------
Winner: Candidate Y

----------------------------------------
```

- **InsertCandidate(Candidate c)**:insert candidate both in file and in database.Implement a function to write candidate data to a text file using StreamWriter.Serialize an instance of the Candidate class to a binary file.



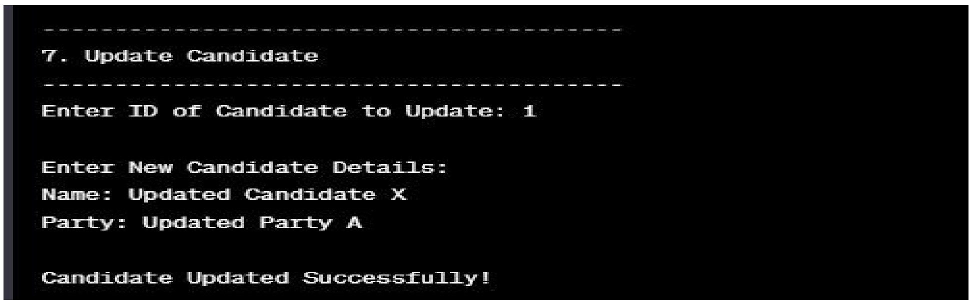
```
----------------------------------------
6. Insert Candidate
----------------------------------------
Enter Candidate Details:
Name: Candidate X
Party: Party A

Candidate Inserted Successfully!
```

- **readCandidate(int id)**:read candidate both from file and db.Implement this function to read and deserialize candidate data from a binary file using FileStream and BinaryFormatter.



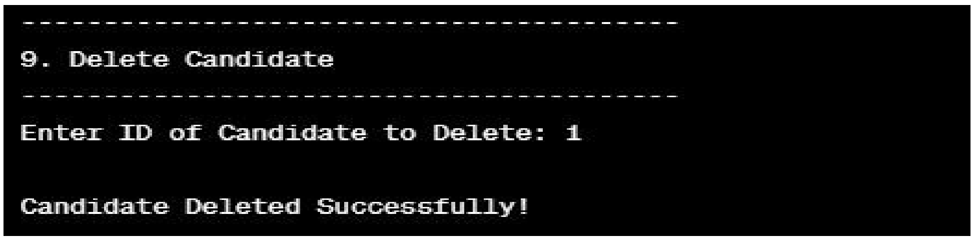
```
----------------------------------------
2. Read Candidate
----------------------------------------
Enter Candidate ID to Read: 101

Reading Candidate details from File...
Candidate ID: 101
Name: John Doe
Party: Party A
Votes: 25

Reading Candidate details from Database...
Candidate ID: 101
Name: John Doe
Party: Party A
Votes: 25
```

- **updateCandidate(Candidate c , int ID)**:update candidate both in file and db on the basis of ID.



```
----------------------------------------
7. Update Candidate
----------------------------------------
Enter ID of Candidate to Update: 1

Enter New Candidate Details:
Name: Updated Candidate X
Party: Updated Party A

Candidate Updated Successfully!
```

- **deleteCandidate(int ID)**:delete candidate both from file and db on the basis of ID.



```
----------------------------------------
9. Delete Candidate
----------------------------------------
Enter ID of Candidate to Delete: 1

Candidate Deleted Successfully!
```

Database Schemas

Candidate Table:
creates a table named **Candidates** with the following columns:

CandidateID: An integer column serving as the primary key.
Name: A string column for the candidate's name.
Party: A string column for the candidate's party affiliation.
Votes: An integer column to store the number of votes the candidate has received.

Voter Table:
Create a table name Voters with following column:

VoterID: An integer column serving as the primary key.
VoterName: A string column (nvarchar) to store the name of the voter.
SelectedPartyName: A string (nvarchar) to store the name of the selected Party Name.

Menu based interface

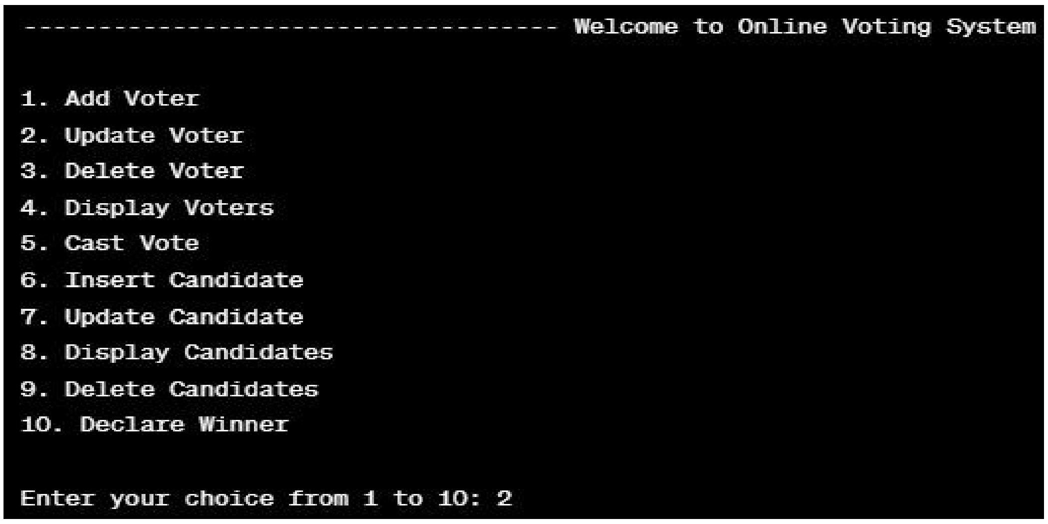
Create a menu based program from the admin point of view , who had access to perform all crud operations

----------------------------------Welcome to Online Voting System--------------------------------------

Create this menu system from admin point of view.

1. addVoter()
2 updateVoter(string cnic)
3 deleteVoter(string cnic)
4 displayVoters()
5 Castevote()
6 insertCandidate()
7 UpdateCandidate()
8 DisplayCandidates()
9 DeleteCandidates()
10 DeclareWinner()

Enter your choice from 1 to 10: ------------



```
------------------------------------ Welcome to Online Voting System

1. Add Voter
2. Update Voter
3. Delete Voter
4. Display Voters
5. Cast Vote
6. Insert Candidate
7. Update Candidate
8. Display Candidates
9. Delete Candidates
10. Declare Winner

Enter your choice from 1 to 10: 2
```