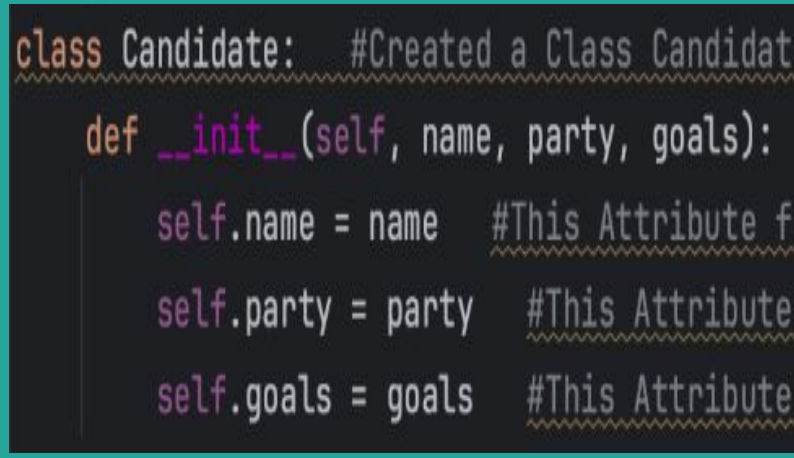
Election Management System

By: Wilson Quilli and Jason O'Neill

—

Description:

Our Project is designed to help run Elections with an easy-to-use system. We utilized several Classes throughout the code to store the attributes in.Some characteristics of our system is that it allows Admins to perform actions like adding a new Candidate or Voters to vote for that Candidate.

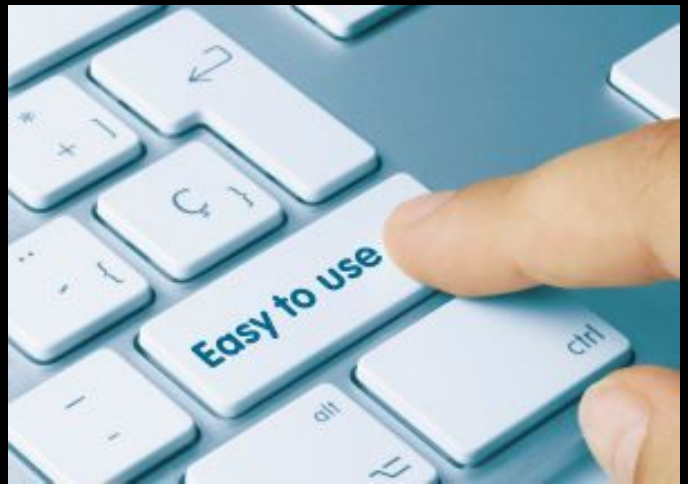


Project Goal:

Our goal with our Election Management System, was to create an easy-to-use system for admins and voters to use during Elections. We wanted it to be easy to sign up and login as an admin (with the correct key we set), and sign up and login as a voter (meeting the age requirements).

```
Welcome to Election Login Page
1. Admin Log In
2. Admin Sign Up
3. Voter Log In
4. Voter Sign Up
5. Exit System
Enter Your Choice: 2
Enter New Admin Username: admin
Enter New Admin Password: admin123
Enter Admin Key To Sign Up: wjes
Admin Signup Was Successful!

Welcome to Election Login Page
1. Admin Log In
2. Admin Sign Up
3. Voter Log In
4. Voter Sign Up
5. Exit System
Enter Your Choice:
```




```
Welcome to Election Login Page
1. Admin Log In
2. Admin Sign Up
3. Voter Log In
4. Voter Sign Up
5. Exit System
Enter Your Choice: 1
Enter Admin Username: admin
Enter Admin Password: admin123
Admin Login Successfully!

Admin Menu
1. Remove a Voter
2. Search for a Voter
3. Add a Candidate
4. Remove a Candidate
```

Instructions On How To Use

1. Start the Program
2. Sign Up as Either an Admin or Voter
3. After Creating an Account and Inputting the Correct Information, Log In
4. Once Logged In, Use the Menu, of the Option You Chose
5. Input the Number of the Desired Action in the Menu
6. Save Any Changes You Wish
7. Sign Out of the Menu
8. Exit/Close the Program with the Exit Feature in the Main Menu

Structure of the Code

We designed our Project utilizing Object Oriented Programming. This can be shown through all the Classes (Admin, Voter, Candidate) used throughout the code. All of these classes encapsulate their own different attributes that are relevant to their specific functions But the “Core” of the System is the Election Class, which coordinates all of the Election’s interactions and controls the overall management o system.

```python
class Voter:   #Created a Class Voter For All the Attributes for Voters
    def __init__(self, name, voter_ID, age):   #Attributes such as a Voter's Name, their ID, and their Age
        self.voter_ID = voter_ID   #This Attribute is for the ID of the Voters
        self.age = age      #This Attribute is for the age of the Voters
        self.vote = None    #This is for the Vote Function and the Count of the Votes
        self.feedback = None    #This Attribute is for the Feedback Function

    def display(self):   #This Function will print out/display the details of a voter
        print(f"Name: {self.name}")   #Will display the Voter's Name
        print(f"voter_ID: {self.voter_ID}")   #Will display the Voter's ID
        print(f"Age: {self.age}")   #Will display the Voter's Age
        if self.vote:   #This will check if the voter has voted yet
            print(f"Vote: {self.vote}")    #If they did it will print out their vote
        if self.feedback:   #This will check if the voter has left any feedback
            print(f"Feedback: {self.feedback}")   #If there is any feedback, this will print out that feedback
```

```python
class Admin:   #Created a Class Admin For All the Attributes for Admins
    def __init__(self, username, password):   #Attributes such as the Username of an Admin and a Password for the Admin Account
        self.username = username   #This Attribute for the Username of the Admin
        self.password = password   #This Attribute for the Password of the Admin
```

```python
class Candidate:   #Created a Class Candidate For All the Attributes for Candidates
    def __init__(self, name, party, goals):   #Attributes such as the Name, Their Political Party, and Their Goals as President for Candidates
        self.name = name   #This Attribute for the Name of the Candidate
        self.party = party   #This Attribute for the Party of the Candidate
        self.goals = goals   #This Attribute for the Goals of the Candidate

    def display(self):   #This function will print out/display the details of a Candidate
        print(f"Name: {self.name}")   #Will display the Name of a Candidate
        print(f"Party: {self.party}")   #Will display the Candidate's Political Party
        print(f"Goals: {self.goals}")   #Will display the Candidate's Goals as President
```

```python
class Election:   #Created a Big Class Election For All The Attributes for Election
    def __init__(self, voters_file="voters.csv", candidates_file="candidates.csv", admins_file="admins.csv"):   #Attributes for the Storage System and Election Candidates
        self.voters = []   #Voters in Election Class in a List
        self.candidates = []   #Candidates in Election Class in a List
        self.admins = []   #Admins in Election Class in a List
        self.voters_file = voters_file    #File of the Voters in Voters Storage CSV System
        self.candidates_file = candidates_file    #File of the Candidates Storage CSV System
        self.admins_file = admins_file   #File fo the Admins Storage CSV Storage
        self.admin_key = "WJES"   #Admin Key for New Admins Signing Up. #WJES - WilsonJasonElectionSystem
        self.feedback_list = []   #Feedback in a list for Admin's to view from their menu
        self.votes_info = {}   #This is to store votes in a dictionary to tell us who voted
```

Applications of Class Learnings

Throughout our Project, we used multiple different learning methods/experiences from both our CMPSCI 131 and 132 Class.

Topics:

1. CSV Files/Storage System
2. Object Oriented Programming (OOP)
3. Encapsulation
4. Exception/Error Handling
5. Data Structures

Shown in Project:

1. We used CSV Files to create a storage system in our project to allow users to save any data they wish to and clear data as well. We both learned this in our CMPSCI 131 Class.
2. We learned a lot about OOP in our CMPSCI 132 and in our code, and in our system we used it in places like storing all feedback in a dictionary, a voter might've left.
3. Relating to OOP, in our system, Encapsulation can be seen in storing attributes in classes such as the Voter Class encapsulating a Voter's Name and ID.
4. In our CMPSCI 131 class, we learned about Exception/Error Handling, by utilizing the Try and Except Method. This can be seen in our Admin Key Function for Sign Up.
5. In both, our CMPSCI 131 and 132 class, we learned about Data Structures and this can be seen in Lists, Dictionaries, and Tuples used throughout our system.

Conclusion

Our Election Management System leverages a variety of techniques learned in CMPSC 131 and CMPSC 132, including importing CSV files for data storage and implementing classes such as Candidate with attributes like name, political party, and presidential goals. Object-oriented programming principles can be seen throughout our system, such as encapsulation. We also utilized concepts from data structures and algorithms, the system performs tasks like candidate searching efficiently using data structures like Tuples and Lists. By incorporating file handling, exception handling, and user input, our system provides a user-friendly experience for both administrators and voters, and it can be seen as an example of how all the concepts we've learned this past semester can be applied to create real-world applications and