Rohan Kinraid [rek0261]

[Company name]  [Company address]

Iteration 2

# Introduction:

The goal of this iteration of the project Is to set up the flask backend of the application, and also to ensure that the Get\_candidates() object is working properly. The result of this should be displayed on the voting page, with each candidate for a specific electorate displaying accordingly.

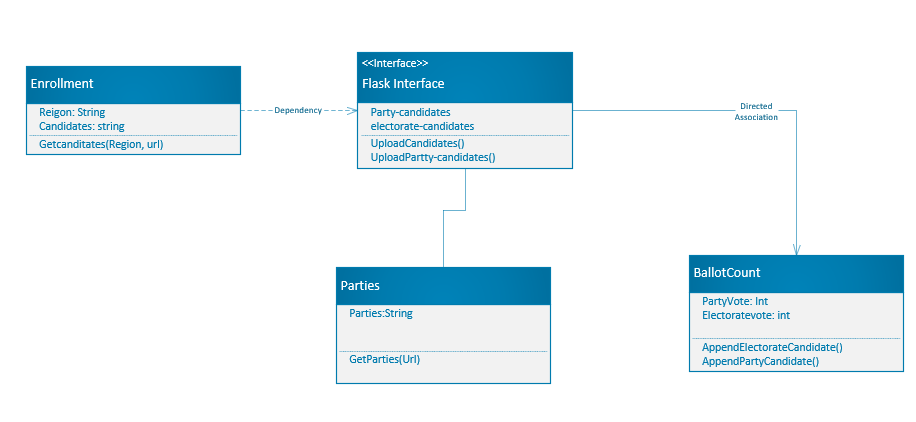
# Sequence of operations and product of each task:

The sequence of events for this iteration will be to create the already made html site into a flask app, create a function which downloads the csv file from the election website, have another function loop through the result of that, cleans the data and displays the desired candidates of that electorate. And finally display those candidates through the flask app.

# Timed Planning and actual time:

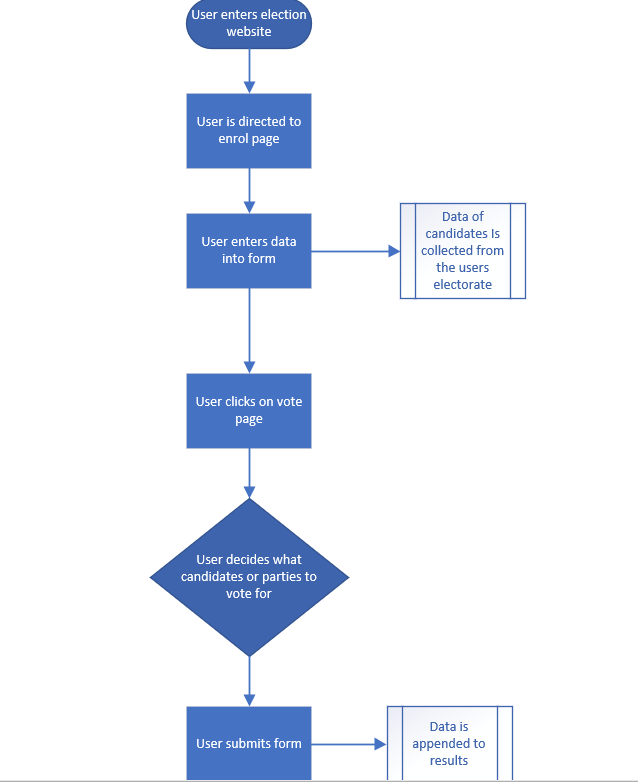
|  |  |  |
| --- | --- | --- |
| Task | Estimated time | Time it took |
| Create Flask App | 30mins | 2 hours |
| Get\_csv function | 1 hour | 1.5 hours |
| Get\_candidate function | 2 hours | 1 hour |
| Apply class to app | 1 hour | 30mins |

# UML Diagram:



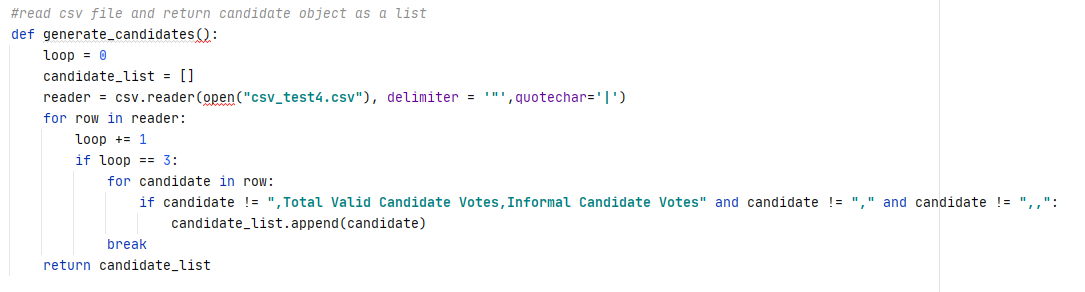
As you can see the UML diagram has not changed from the previous iteration because there are no real changes to the operation of this task. I am considering maybe removing the ballotcount object because it does seem a little irrelevant to the task, however, I will come to that decision in iteration 3 of this project.

# Activity diagram:

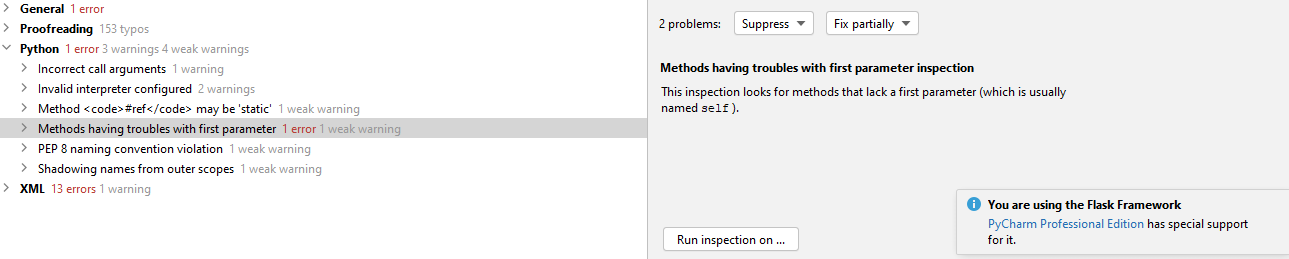


# Error testing data:

When downloading the csv file and reading it, a lot of data is create in dictionaries and lists. In order to view these we must iterate through each object and display the candidates, however the is a lot of unwanted data which could mean that the wrong information is displayed on the webpage. One such example is that on the end of line 3 in each csv file are unnecessary lines of text, like “Total valid candidate votes , and informal candidate votes. How we can get around this is stopping the loop on the line where we find the candidates, checking if the candidates are not of the erroneous data, and then appending those candidates to a list.



# Code Validation:



Ignoring the XML errors (which was because we are looping through the same element giving it multiple ids, which is the point anyway :/) the only other major error was the class lacking the self parameter.

# What happened vs what was planned:

There were no real changed from the original plan except for the time it took to perform those tasks. Some changes I did make over all was to swap around the position of the candidates and the parties in the html, just so I could confirm that the app was working.

# Performance review:

## What worked and didn’t work:

This time everything usually went to plan which was a bit of a shock. Iterating through the csv results was easier than expected and didn’t take much time as I had initially thought.

Downloading the csv file proved a little more difficult than expected because for some reason I was getting a 403 forbidden result. But it turned out, using the liburl libretreive request was blocked by that website. Instead I had to use a standard get request from the requests module.

What I would change next time is to properly construct the class so that I wont get any errors in the code validation.

## Performance:

My own personal performance was largely reliant on if I already knew how to do a certain solution or not. For example I’ve never webscraped from a flask application before so that was a little bit of a challenge to understand the specifics of downloading from a urllib compared to a standard get request.