Alex H, James M, Sophie H, Aakriti S

CMS 340 - Software Engineering

Project Document

voteSmart: Political Alignment System

Overview

This project, “voteSmart” is to help the users of this application (US citizens) to easily connect with political candidates that best fit their personal views. When the application first opens, users will be greeted with a short survey so that the application can compute how this user aligns with different parties and local candidates. This application will concisely state a party’s platform to the user, in comparison with other parties. If a user is not interested in taking the survey, they can skip it and continue to view the party’s platforms. This will be presented in a way that is easily understood by the common person so that the user can compare a candidate's views against others and make a sound decision based on their views. As an added feature, the application will include a database with all the voting locations where the user can input their location and search for which location, they want to vote in. We determined that one of the main reasons for the recent downturn in US voter turnout is a lack of proper education on current political platforms. The main purpose of this application is to altruistically help potential voters to make the voting decision that is the truest to their values; these educational resources could increase not only a voter’s confidence in their participation in democracy but also the number of citizens who vote.

Requirements Engineering Phase

Requirements Elicitation

During this phase, we went through the process of establishing the services required from this application and the constraints under which it will operate. From conducting interviews, we found that young citizens desired a more in-depth and objective education on current political platforms. We evaluated research about political trends and voter participation and found that there is a decreasing level of enthusiasm toward voting in elections. Observation of voter trends has also led us to find that the US has trailed behind other developed countries when it came to voter turnout. Through these techniques of elicitation, we determined the main reasons for this downtrend in voter turnout: lack of proper education on current political platforms, lack of time, and the feeling that their vote will not make a difference. While we cannot fix the two latter problems, we can try to educate citizens about politics, parties, and their platforms, and how and where to vote. Consequently, we have developed a list of requirements needed to implement the services our stakeholders desired from this application. The requirements include:

- The application must be compatible with iOS and Android phones
- The application will allow the user to make an account to keep track of their data
- The application will allow the user to save their username and password within the app
- The user should input their location
- Upon first use, the application will present the user with a political affiliation quiz
- Every new quiz question will adapt to the user's answer to the previous question
- The user should select one answer per quiz question
- The application will store results from the quiz and use them to show the user which political parties they most identify with

- The application will use the user’s location and party affiliation to recommend specific candidates from the user’s local elections
- The application should inform users of the dates and locations of upcoming elections
- The application will send a link to the user for a better description of the candidates
- The username and password will be securely stored within a database
- If the user is logged in and has already completed the quiz, the system will not require the user to retake it, but the user can opt to retake if they prefer
- The admin should update the quiz questions every week
- The admin will update candidates' descriptions every week

Along with the sources listed at the end of this document, included below are some images we found that helped us analyze the real-world problem of the lack of education of voters.

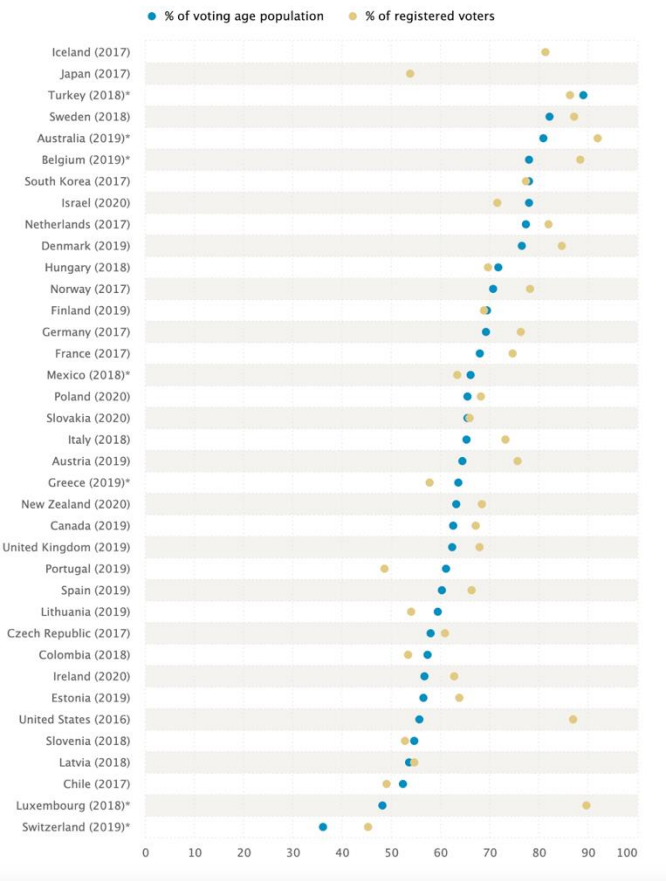


Image 1. Trends of the Percentage of Voting Age vs Registered Voting Population in Developed

Reference: Source 3

Countries Between 2017 and 2020

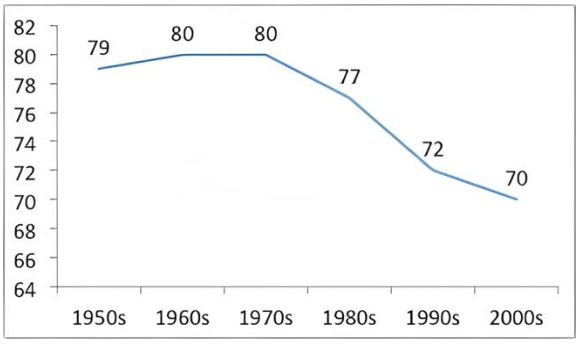


Image 2. Trends of Voter Turnout in the United States from the 1950s to the 2000s

Reference: Source 4

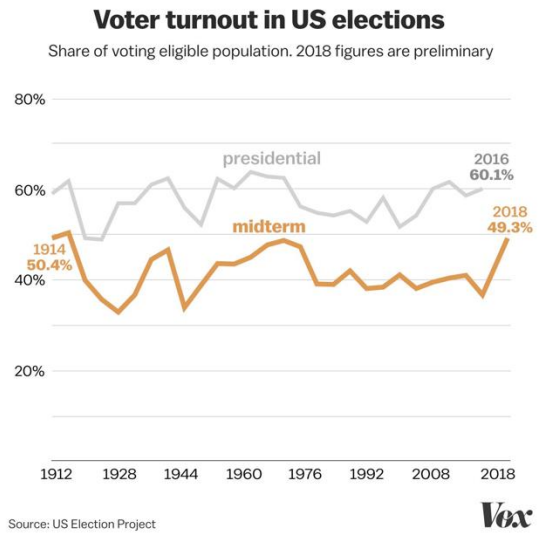


Image 3. Trends of Voter Turnout Percentages in the Presidential vs Midterm Elections from 1912 to 2018

Reference: Source 12

Requirements Analysis

In the requirements analysis phase, our group began by picking apart each elicited requirement from the phase prior and analyzing each one by breaking it into parts. We also classified each requirement by type (non-functional, functional, user, system, and business), clearness (ambiguous, clear, contradictory), and completeness (incomplete, complete). The first requirement, “Application must be compatible with iOS and Android phones”, is classified as a non-functional and system requirement that is clear and complete, as it states the constraint clearly and falls under the product requirements (usability requirements specifically). After unpacking this requirement, we decided to adjust it to not only be compatible with phones, but also with computers on a website. Therefore, this requirement was changed to “Application must be compatible with iOS, Android, macOS, Microsoft Windows, and Linux operating systems”.

The next requirement, “Application will allow the user to make an account to keep track of their data”, is a functional requirement, but also a system requirement. It is also clear, complete, and encapsulates the idea of the previous requirement, “Application will allow the user to save their username and password within the app”. Instead, this requirement can be combined and simply stated as “Application will allow the user to make an account to keep track of and save their data”. Moving on to the next requirement, “Users should input their location”. This requirement is classified as a functional and user requirement. We analyzed this requirement and determined that it was complete, clear, and sufficient to keep as is. The requirement, “Upon first use, the application will present the user with a political affiliation quiz”, is a non-functional requirement because it contributes to the usability of the system. After evaluation, we determined it can be simplified to “Application will present the user with a political affiliation quiz” so that it gets rid of the unnecessary sentence lead, making this requirement less ambiguous.

The “Every new quiz question will adapt to the user’s answer to the previous question” requirement is a non-functional and system requirement because it contributes to the performance of the system. It is clear, complete, and it states exactly what service the system will be providing. The next requirement, “Upon first use, the application will present the user with a political affiliation quiz”, is a non-functional requirement. It is clear, complete, and states exactly what occurs when a user first uses the application. “Every new quiz question will adapt to the user’s answer to the previous question” is a non-functional and system requirement that is also clear and complete, so it will be left as is. “User should select one answer per quiz question” is a user requirement but seemed to be unnecessary and not relevant to the requirements of the system, so we excluded it from the requirements. The requirement, “application will store results from the quiz and use them to show the user which political parties they most identify with”, is a

functional and system requirement, which is clear, but overlapping. We determined that this single requirement could be split into two: “application will store results from the quiz” and “application will use results from quiz to display the user’s top political party matches”. This way, both requirements are more precise, complete, and no longer overlap with one another.

The “application will use the user’s location and party affiliation to recommend specific candidates from the user’s local elections” requirement is a functional and system requirement. It is clear and complete, stating how the user's information will be used to recommend the best-fit candidate. The “application should inform users of the dates and locations of upcoming elections” requirement is functional. It is clear in describing how we are going to inform the users about when elections will take place and suggest the most convenient locations for them to vote. This will also be in the system saved depending on the location input by the user. The “application will send a link to the user for a better description of the candidates” requirement is a system requirement. We decided to exclude it from the requirements due to accessibility inconsistencies when it came to the idea of later implementation.

The “username and password will be securely stored within a database” requirement is functional and non-functional. The functional aspect is how the data is being stored within the application. The non-functional aspect of it is the security, assuring the data inside the database is protected. The “If the user is logged in and has already completed the quiz, the system will not require the user to retake it, but the user can opt to retake if they prefer” requirement is a system requirement. The user will not be required to complete the quiz upon logging in if the quiz has been completed already. We will supply a message saying the questions will have been updated and results may change due to the update. The “administration will update candidates' descriptions every week” is a system requirement that describes how the administration will

interact with the system. We found that combining this requirement with the “Administration should update the quiz questions every week” requirement makes for a more complete and proper description of the entire service being provided. This requirement consolidated to “Administration should update the quiz questions and candidate descriptions every week”. This requirement is classified as a non-functional and system requirement because it contributes to system maintainability.

During the Requirements Analysis Phase, we used strategies such as problem analysis and technology analysis to decompose our entire system into smaller components so that we could determine the inputs and outputs from each system functionality and define the relationships between each component. In the end, this helped us piece together the final list of requirements for our system.

Requirements Specification

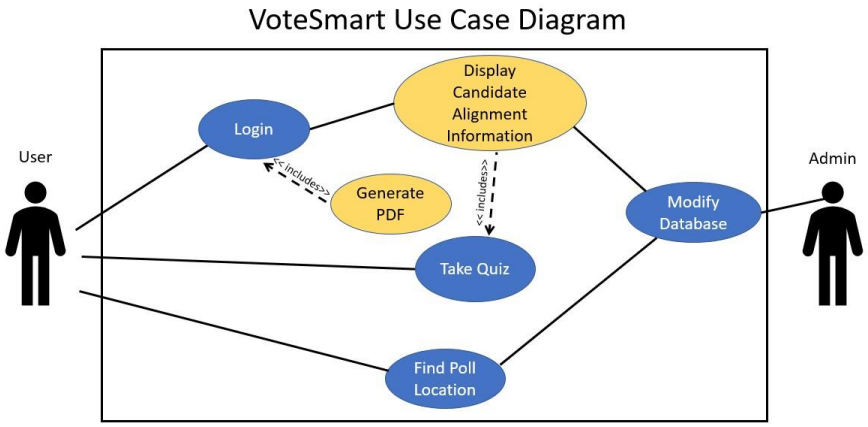


Diagram 1. Use Case Diagram

Here are our Use Case Models for each use case presented in the diagram above:

Element	Description
Use Case	Take Quiz
Brief Description	Use interface to get user zip code to be able to provide results based on the the candidates in their county/ state that they are able to vote for during the nearrest future election
Participating Actors	Communicates with **Voter**
Flow of Events	When users enter the system, they have the choice to make an account so that their data and information will be saved. Then the user will choose their county from a drop down menu. From there they will answer questions based on their own political values that are considered major issues in current events that candidates have a clear opinion on (ie. gun laws, public education, wage gap). Each question's answer will be represented by a value and added to a count total for the given topic. The system will update each new question to adapt to the user's answer to the previous question.
Entry	Location must be chosen before quiz can be taken.
Exit Condition	Once quiz is taken, application will proceed to calculating and displaying results.
Quality Requirements	This use case needs to be done in order for the user to have their personalized political candidate matches presented to them.

Model 1. Use Case Model for "Take Quiz"

Element	Description
Name	Display Results
Brief Description	Analyzing and displaying the results of the political values quiz.
Participating Actors	Initiated by **Voter**
Flow of Events	Use resulting total counts from Take Quiz to compare the results to political candidates running for office in the designated county. System will then display the candidates that match most closely with the user one the most topics. Display of candidates includes the percent matches on each defined topic and a statement summarizing the candidates platform. Any political affiliations should also be included.
Entry Condition	Must complete quiz.
Exit Condition	Results are displayed throught the interface.
Quality Requirements	At any point during the flow of events, this use case can **include** the Take Quiz use case. The Take Quiz use case is initiated when the Voter first uses the application.

Model 2. Use Case Model for “Display Results”

Element	Description
Name	Find Poll Location
Brief Description	Use list of locations categorized by county to provide of list of locations in the county of the voter without using the voters exact location.
Participating Actors	Communicates with **Voter**
Flow of Events	User chooses county from a list of options. Voting center in the county will be displayed with their address.
Entry Condition	User must choose one of the avaiable county options.
Quality Requirements	Must be an official poll location that can be found in the county selected by the voter.

Model 3. Use Case Model for “Find Poll Location”

Element	Description
Name	Update Database
Brief Description	Admin accesses quiz and changed questions and/or format to provide users with updated information on available candidates.
Participating Actors	Communicates with **Administration**
Flow of Events	Admin gains access to system. Admin uses data gathered to change quiz questions on a weekly basis; the value of the answers to the question and how they effect the resulting score, or the values corresponding to the candidates based on changes to thier platform. Admins can also add new candidates or remove previous candidates depending on the candidate's personal status
Entry Condition	Base quiz must already be created. Admin must have system access.
Exit Condition	Users no longer have access to previous quiz results and must retake the quiz for their new compatablity scores. However, results of previous quizzes and political candidate matches will be available in the history of their
Quality Requirements	For admin use only. Voters without permissions cannot access this function.

Model 4. Use Case Model for “Update Database”

Added Functionality

For contextual purposes, we have added this section to explain the slight changes in our system, since the Requirements Engineering phase. We have added some requirements in addition to this. The feature allows the user (if they have an account and are logged in) to generate a PDF version of their political party alignment. This PDF will be in the format of a sample ballot, so that the user now has their results in the format of how they will see it when they are voting. The requirements that were added include:

- If the user is logged in and has already completed the quiz, the system will generate a PDF version of their political party alignment.
- The user will be able to save and/or print their generated PDF.

Logical Design

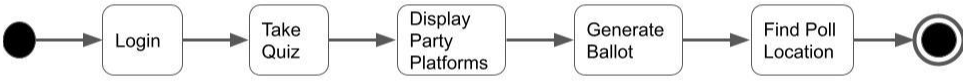
Sequence Diagram

VoteSmart Sequence Diagram

We decided to make a sequence diagram so that we can present the interactions between the User, Web Application, and Database in an easily interpretable manner. This not only displays the order of the steps throughout the operation of the system, but it also showcases the requirements necessary for the user to be able to access certain functionalities of the system as well. When the user enters the system, they will be presented with options on how they would like to proceed. Depending on their choice, the program will proceed in different ways, and the interactions between the User, Web Page, and Database will also be different.

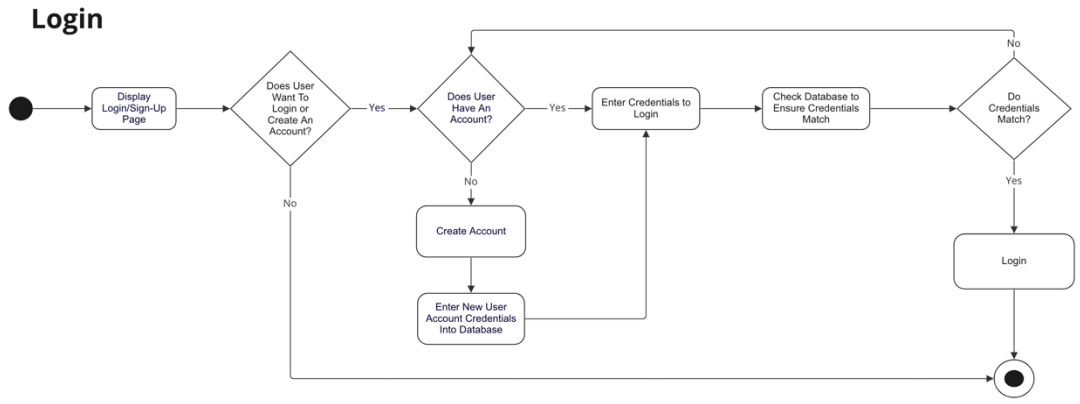
Activity Diagrams

Activity Diagram for Overview of System:



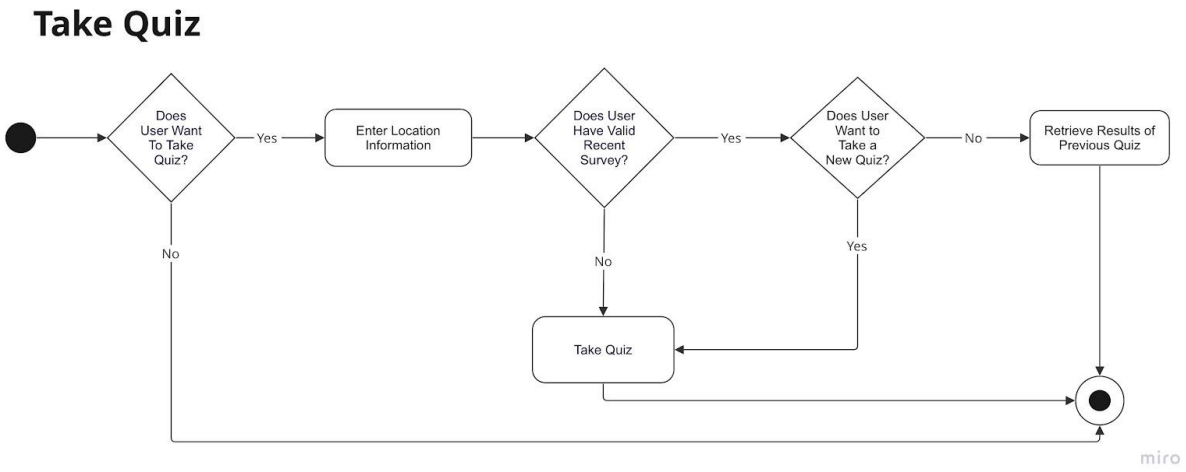
This diagram displays the overall sequence of events of the system. The diagrams described below will walk through each of these activities more intricately and with specificity.

Activity Diagram for Login Use Case:



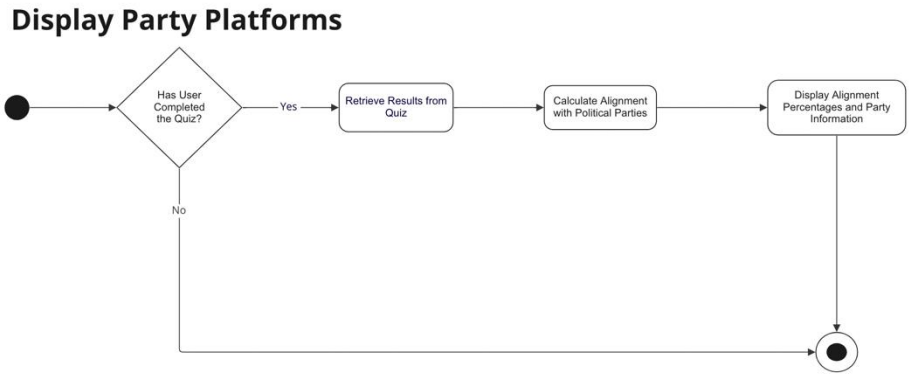
The system will display the login page and give the user the option of logging in or creating an account. If the user does not wish to create an account, the activity ends. If the user wants to log in and does not have an account, they will create an account and that new account information will be stored in a database. They will then enter their login credentials. The credentials will be checked with the database, and they will then be logged in. If the user already has an account, they will simply enter their login credentials which will be checked using the database. If the credentials are wrong, they will be instructed to enter their credentials again. If the credentials match the database, they will be logged in.

Activity Diagram for Take Quiz Use Case:



The user is presented with the option to take the quiz. If they want to, then they must enter their location information (in the form of a zip code). If the user has a recent quiz, they can use their old results but also take a new quiz. If they do not have a recent quiz, they will take the quiz. If the user does not want to take the quiz or use quiz results, they can skip to the end.

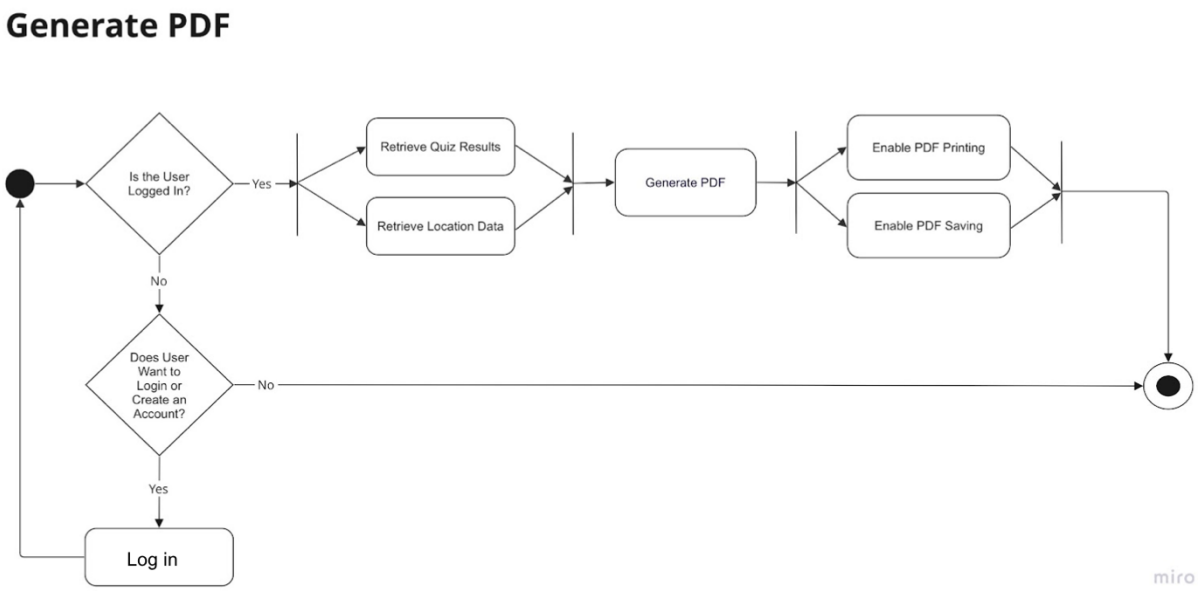
Activity Diagram for Display Party Platforms Use Case:



The system first checks if the user has completed the quiz. If they have, then the system retrieves the quiz results and calculates the user's alignment with the political parties and

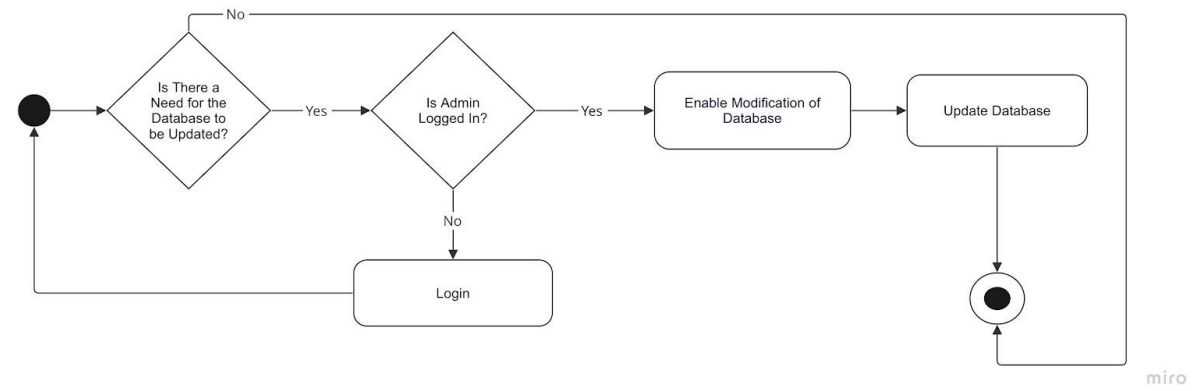
candidates. The system then displays the party/candidate information along with the user's compatibility percentage. If the user has not taken the quiz yet, this activity will be exited.

Activity Diagram for Generate PDF Use Case:



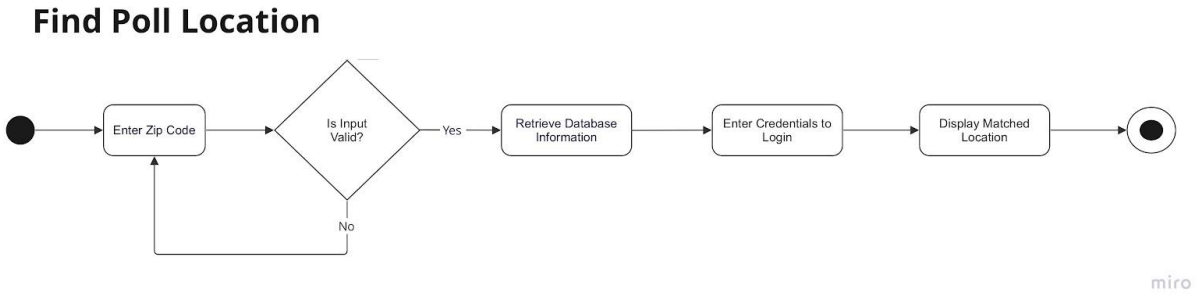
The system checks if the user is logged in. If not, the user is given the option of logging in or creating an account. If they opt not to, the activity ends. If the user is logged in, the system will simultaneously retrieve the quiz results and the location data. The system will then allow the user to generate a PDF displaying the local sample ballot. The system will allow the user to print or save the PDF.

Activity Diagram for Modify Database Use Case:



If there is no need for a database update, the activity ends. If so, the system checks if the admin is logged in. If not, the admin is directed to login. Once logged in, the system enables the modification of the database before the admin updates the database appropriately

Activity Diagram for Find Poll Location:



The user enters a zip code. If it is not a valid U.S. zip code, they are prompted to enter another one. Once the input is valid, the system retrieves the relevant information from the database. The user logs in and the system will then display the poll locations it found from the database.

Physical Design

Platforms

The voteSmart system will be able to run as a Web Application enabling it to be compatible with any device that can access the internet with any major web browser. It can be used in Chrome, Safari, and Internet Explorer. This system will also work on different operating systems: Mac, Windows, and mobile.

Programming Languages

The programming languages and frameworks used during the implementation and maintenance phases of our application's development will be Python, HTML, CSS, SQL, and the framework Flask. Each will be used for different purposes, as described below.

Python:

Python will allow us to build our application with the idea of data storage and analysis in mind. It will let us develop both front-end (with the help of CSS and HTML), to program how the user will interact with our system, and back-end so that we can program the interactions that our system will have with our data, which will be stored in a database. Using Python will also be advantageous because of the wide variety of frameworks at our disposal, including Flask.

Flask:

We have decided to use Flask in addition to Python, as this framework will help us with the foundation for our web application. Since a requirement for our application is for it to be compatible with Mac OS and Microsoft Windows operating systems, using Flask will enable us

to create a universal web application, which can be accessed on any device with a way of accessing the internet.

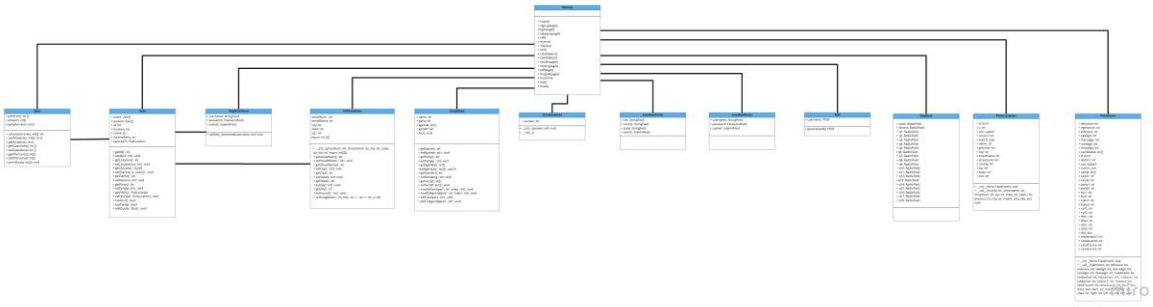
HTML and CSS:

The pairing of these languages and Python will enable us to structure the webpage, as well as provide a layout of how the user will view it. Using HTML and CSS will allow for more efficient maintenance and updating overall, and more formatting and design options so that the page appears visually inviting to the user. These languages, along with Python and Flask, will help with the front and back-end development of the webpage.

SQL:

Since we decided to store the data needed for this system in a database, we chose SQL to be able to store, easily access, create, retrieve, update, and delete data, while still being linked to our system. Database storage will be used for four reasons in our program, for storage of user information, quiz questions (this will be maintained by the Admin Actor), poll locations, candidate information. We used Sqlite as our Database Management System.

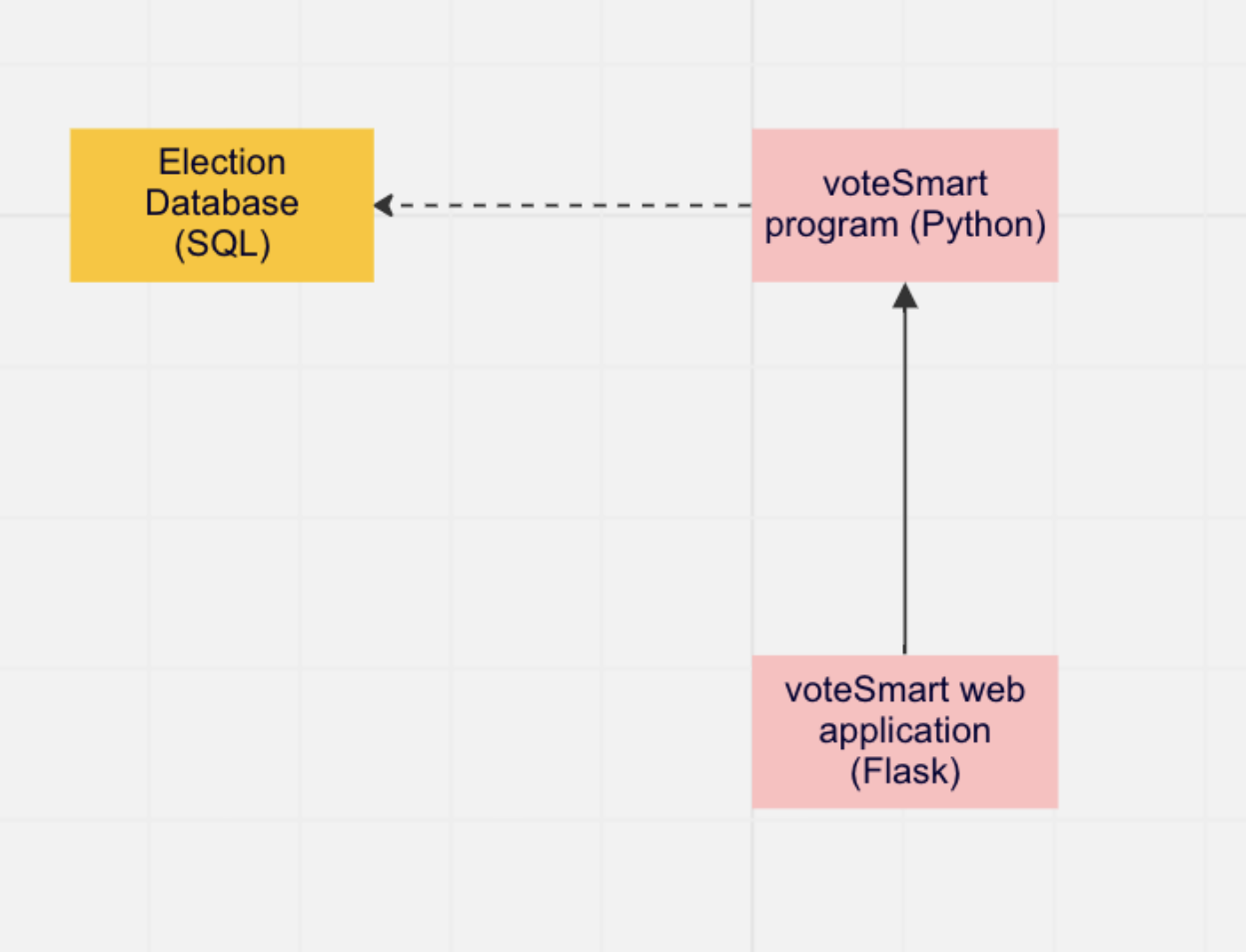
Class Diagram (UML)



Program Classes:

The specific classes for this program were chosen based on what needs to be manipulated and stored as objects. Having these thirteen classes will allow for easy and organized access to necessary information stored by the system. Routes will be used as the driver class that can access any class objects and their data members through their public methods and define routes to different web pages. The User class will be used to create User objects for each individual user and can store their personal information such as their name and general location (zip code). Each Quiz a user takes will have an array of Quiz objects stored within their specific User object. The User class can also store a Poll Location object so that the user can save the Poll Location that is most convenient for them and access its characteristics efficiently. User also contains a static array of all User objects in the system. Candidate is not used to interacting with User, Quiz, or Poll Location specifically, but is where information on specific candidates is stored for access based on quiz results or to be displayed for users without an account. So therefore, the Candidate class is only connected to the main so it can be accessed to provide information to the user but is not directly accessed or stored by any other class. The helper classes will provide easy access to different Flask forms that will be presented to the user: Register, Login, and Signup.

Component Diagram



This is a high-level overview of how our physical systems interact with each other. The voter uses the voteSmart web application to access the voteSmart program. The program will take data from the Election database to collect information on the user and local elections and candidates.

Lessons Learned

Overall, we loved working with each other on this project and learned so much about ourselves and in turn, also broadened our knowledge about the specifics of the SDLC as we were able to get firsthand experience with it. We had many obstacles we had to overcome to complete this

project, but as a team, we worked together to get through it all. One key thing we learned from this project was new languages and uses within the language. We had to learn more than just the basics of Python and Flask and how they interact with CSS, HTML, and SQL in order to create the application we were imagining in our heads from the planning stage. We overcame this by watching videos on the key concepts we did not understand, creating a basis to build on when we started creating our system. We then were able to tackle the implementation phase in a more effective and efficient way. Another obstacle that we faced was time management. Throughout this project, we had deadlines we had to meet for certain aspects of our software. Although these checkpoints helped us progress with our project, they were also exceedingly difficult to stick to as we basically developed our entire project in one month due to the hurricanes pushing everything back, thanksgiving, and other classes. We had to adjust ourselves to each other's schedules, communicate effectively, and logically plan in order to complete these tasks, which we did successfully. Choosing an appropriate scope for the project is also a lesson we learned throughout the process. At the beginning, we had so many ideas about how we wanted our system to look, the features we wanted it to have, and the easiest way for the user to interact with it, but throughout the process we had to narrow our scope to make our deadlines and our final product something of quality. An example of this is that, in the requirements engineering phase, one of our requirements was to include a PDF generator to make a PDF version of the user's quiz results in the form of a sample ballot. Towards the end of implementation, we modified this functionality to simply produce a sample ballot, without the information from the user's quiz. In the future, we would like to modify the PDF generator to generate a sample ballot with the user's results input, expand to all 50 states and not just Florida, save more than just republican, democrat, and independent quiz results when a user has an account, and improve the overall user

interface and experience. We have also discussed launching it to the public and using social media to target potential users.

Contributions from the Group

- Aakriti: synthetization and compilation of code segments over different languages.
- James: quiz creation and result calculation code.
- Sophie: overall organization, user, and candidate code.
- Alexander: database creation and connection code.

Sources

1. Blais, André. “What Affects Voter Turnout?” *Annual Review of Political Science*, vol. 9, no. 1, 2006, pp. 111–125., https://doi.org/10.1146/annurev.polisci.9.070204.105121.
2. Chandavale, Aditya. “Creating a Quiz Using Flask-WTF.” *Medium*, Medium, 27 Aug. 2021, https://medium.com/@adityachandavale/creating-a-quiz-using-flask-wtf-1875884941ae.
3. DeSilver, Drew. “Turnout in U.S. Has Soared in Recent Elections but by Some Measures Still Trails That of Many Other Countries.” *Pew Research Center*, Pew Research Center, 1 Nov. 2022, https://www.pewresearch.org/fact-tank/2022/11/01/turnout-in-u-s-has-soared-in-recent-elections-but-by-some-measures-still-trails-that-of-many-other-countries/.

4. Ferrini, Luca. “Why Is Turnout at Elections Declining across the Democratic World?” *E*, 28 Sept. 2012, https://www.e-ir.info/2012/09/27/why-is-turnout-at-elections-declining-across-the-democratic-world/.
5. Harder, Joshua, and Jon A. Krosnick. “Why Do People Vote? A Psychological Analysis of the Causes of Voter Turnout.” *Journal of Social Issues*, vol. 64, no. 3, 2008, pp. 525–549., https://doi.org/10.1111/j.1540-4560.2008.00576.x.
6. Haynes, George H. “The Education of Voters.” *Political Science Quarterly*, vol. 22, no. 3, 1907, pp. 484–497. *JSTOR*, https://doi.org/10.2307/2141059. Accessed 30 Nov. 2022.
7. Khorasani, M. “How to Generate Automated PDF Documents with Python.” *Medium*, Towards Data Science, 17 Sept. 2022, https://towardsdatascience.com/how-to-generate-automated-pdf-documents-with-python-55981f4d9e3.
8. “Learn Flask for Python - Full Tutorial.” *YouTube*, YouTube, 28 May 2019, https://www.youtube.com/watch?v=Z1RJmh_OqeA&t=1456s. Accessed 30 Nov. 2022.
9. Madeira, Tiago. “Why Use Python for Web Development?” *Blog | Imaginary Cloud*, Blog | Imaginary Cloud, 29 Apr. 2021, https://www.imaginarycloud.com/blog/why-use-python-for-web-development/.
10. “Python Website Full Tutorial - Flask, Authentication, Databases & More.” *YouTube*, YouTube, 1 Feb. 2021, https://www.youtube.com/watch?v=dam0GPOAvVI&t=5667s. Accessed 30 Nov. 2022.
11. Rosenstone, Steven J., and Raymond E. Wolfinger. “The Effect of Registration Laws on Voter Turnout.” *American Political Science Review*, vol. 72, no. 1, 1978, pp. 22–45., https://doi.org/10.2307/1953597.

12. Stewart, Emily. “2018's Record-Setting Voter Turnout, in One Chart.” *Vox*, Vox, 19 Nov. 2018, https://www.vox.com/policy-and-politics/2018/11/19/18103110/2018-midterm-elections-turnout.
13. “The Visual Collaboration Platform for Every Team: Miro.” *Https://Miro.com/*, https://miro.com/.
14. *W3C*, https://www.w3.org/standards/webdesign/htmlcss#:~:text=HTML%20.

Appendix

a. readme.txt

- Contains file imports and how to run the project

b. voteSmartFinal.zip

- Contains all code: .py files, .html files, .css files

c. UMLClassDiagram.jpg

- Contains the UML Class Diagram that is too hand to see in the final documentation