Political Tap

By

David Dougherty, Adam Holtmeier, Harshil Chaklashiya, & Kenil Chaklashiya

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

David Dougherty	April 2021
Adam Holtmeier	April 2021
Harshil Chaklashiya	April 2021
Kenil Chaklashiya	April 2021
Abdou Fall, Faculty Advisor	April 2021

University of Cincinnati
College of Education, Criminal Justice, and Human Services

April 2021

Table of Contents

List of Illustrations

Abstract

Political Tap is a web and mobile tool designed to help American voters make informed decisions about representatives and candidates during elections. According to a 2018 poll, approximately 29% of surveyed citizens claimed that they were not well-informed about their candidates, with only 14% being registered voters. In another 2018 survey, roughly 45% of participants reported having very little to no trust in television news. American voters do not feel as if they have the proper outlets of information on candidates' values while also not being aware of who is on the ballot. Our solution to these problems relies on developing an application which provides a list of candidates to users based on location and highlights their positions on major political issues through aggregated social media posts. This helps increase voter awareness and allows for easier comparison among candidates, making citizens feel more comfortable, confident, and knowledgeable at the polls.

Introduction

Problem

According to a Gallup poll conducted in 2018 among U.S. citizens, 45% of polled participants answered that they have very little to no trust in television news as an outlet, whereas 40% answered the same lack of trust towards newspapers (Saad, “Military, Small Business, Police Still Stir Most Confidence”). The modern consumption of media and information lends itself towards unique challenges, due to the saturation and politicization of media outlets. With this increase in media bias, there is no guarantee that potential U.S. voters can receive transparent and accurate information about the views of candidates.

According to research done by the Pew Research Center, 40% of people are not informed in their local districts when voting or do not feel they have enough information to make a final decision on a candidate (“Little Partisan Agreement on the Pressing Problems Facing the U.S.”). In addition, a study conducted by the Huffington Post and YouGov found that 29% of U.S. citizens rated their confidence in their degree of political candidate knowledge below both “somewhat well-informed” and “very well-informed”, with only 14% of that group being registered voters (Edwards-Levy, *Young Voters Don't Feel Well-Informed. Some Think That Means They Shouldn't Vote.*). This lack of information details a decrease in registered voters for local and presidential elections, due to the fact that some voters feel they are uniformed.

Solution

Political Tap is a mobile and web application which aggregates candidates’ tweets based on location data in order to give users an accurate portrayal of candidates’ views on major

political issues. The application categorizes candidates by running office, which allows for accurate comparisons to be made between all of the running candidates and their respective viewpoints. This helps reduce media bias since users can see the tweets of the running candidates without significant interference from media outlets or other third parties. It also highlights local politics, helping inform users about the candidates that are running in their home districts. The application spotlights the stances of candidates on major political issues, increasing awareness and allowing easy comparison. This helps users make well-informed decisions at the polls. Although it is impossible to completely eliminate media bias, this application helps reduce it significantly.

Project Goals

The goal of Political Tap is to leverage social media to allow potential voters easy access to political information through both a web and a mobile application. Political Tap uses location access to display the user's ballot and allow them to view a more comprehensive view of a candidate through a candidate profile page. The application also allows the user to see social media posts from the candidates on their ballot. Users can also view the same information as it relates to their current representatives. To complete these goals, Political Tap is developed to receive information about candidates from a web server while also being able to interact with social media APIs, such as Twitter, to gather social media posts by candidates and display it on the application. Other goals of the project are to make the applications more secure, easy to maintain with each election cycle, and cost efficient.

Overview

The following sections on this report illustrates how this solution was created. The remainder of this report contains: the project concept, design objectives, methodology, user profiles, use case diagram, technical architecture, budget, timeline, analysis of encountered problems, recommendations for improvement, lessons learned, and skills developed.

Discussion

Project Concept

Political Tap allows users to view information about their candidates and representatives through aggregated social media posts. It is available in both a mobile and web application. Political Tap uses a user's location to populate relevant political information about representatives for their district. This tool allows users to easily gain political insights towards key officials and their respective election offices to make an informed vote.

The idea for this project was conceived during a U.S. presidential election year at a team brainstorming session. During this session, a long list of potential project ideas was constructed and eventually narrowed down to two potential ideas which met key standards. Through more detailed conversations, this idea was eventually decided upon as it remained relevant to an election year and no other alternatives which provided similar capabilities to the proposed solution were available. The project's relevance, feasibility, practicality, and overall potential helped it stand out to all group members. After more research, existing APIs were discovered which could help with a lot of the data gathering and heavy lifting that this concept required. The feasibility of this project was further bolstered due to the sheer number of public APIs, documentation, and candidate information available to achieve the intended solution. As a result, the team unanimously decided to develop this concept into Political Tap.

Design Objectives

In the initial design stages of Political Tap, the group decided to utilize existing APIs to gather political information. The first stage revolved around determining the feasibility of gathering accurate data specific to candidates and their social media profiles. It was determined

by the team that gathering information is feasible through the use of existing APIs. The ultimate solution decided upon was to use VoteSmart, an external political API, as it stored all of the information required, and had relevant documentation which fit with the project's design goals.

Front-end design decisions have remained consistent from initial discussions. The web application is built using Angular, HTML, CSS and JavaScript while the mobile application implements the Flutter framework for developing a mobile front-end available on both iOS and Android. The back-end, which collects information from the various APIs, is built using Python. The back-end will utilize the VoteSmart API, as mentioned above, and the Twitter API to obtain the candidate information and social media aspects. Using all of these design decisions, a ballot and profile view is created with accurate information, while also having a social media feed which shows relevant politicians' tweets.

From initial goals, the team had to abandon filtering capabilities on the feed view as well as account functionality for users throughout the application. Since multiple requests to two different APIs were required for gathering Twitter feeds for a user's location, the API responses were too complex to filter tweets on the client side of the project. Similarly, filtering the responses in the back-end added extra time to data retrieval operations. Due to this inefficiency for filtering, this feature was abandoned to prioritize faster loading times for clients. In regards to account functionality, the team found, from user testing and polling, that adding account functionality at the current stage of the project did not enhance the project in a meaningful way. Therefore, account functionality had to be abandoned since it was not essential to the core product and users did not find the feature meaningful.

Methodology

Web Application

Political Tap's web application needed to be easy to maintain, navigate and understand while also being simple and efficient in implementation. To satisfy this approach, the team decided on using the Angular Framework as it allows for efficient development and design in regards to the project's endpoints. As Angular is a web framework, it utilizes common web technologies such as HTML, CSS and Javascript. The web application includes the following pages:

- **Home:** This page is internally referred to as the 'feed' view. Contains feed of social media posts for relevant candidates and representatives in a given location
- **Ballot:** Contains a user's ballot and all of the relevant candidates organized by key offices
- **Officials:** Contains a list of the user's representatives organized by key offices
- **Profile:** Contains a comprehensive view of a politician including information such as a personal bio, twitter feed, and bill voting history

Mobile Application

The goal of Political Tap's mobile application is to have the same functionality as the web version in a compatible mobile format. The mobile application utilizes the Flutter Framework which allows for concurrent development towards an iOS and Android application with one code-set. Since the design objectives are identical to the web application, the mobile application contains the same pages and information associated with the respective pages. Please refer to the web application methodology for further information on mobile pages.

Security

Political Tap requires gathering sensitive user data such as location. The application allows the user to enable location, and the service stops when the app is closed. Initial security discussions revolved around following secure development standards while being cognizant of storing sensitive personal information.

User Profiles

There are three types of users who use the application. First are the developers and administrators who are responsible for creating and maintaining the project. The end-users of the application are American citizens, categorized by their mode of accessing the application through either the mobile application or the website.

Table 1 Developer and Administrator User Profile

User Profile Form 1
Application: Flask. Python, Angular, Flutter, Dart, Vote Smart API, Twitter API, Git, Github Desktop
Potential Users: Developers and Administrators
Software and Interface Experience: Users should be familiar with the user interfaces of both the web and mobile applications. In addition, they should be proficient with the software languages and frameworks used to create or modify both applications. The user should also be comfortable interacting with REST APIs.

Experience with Similar Application: Users need to be familiar with languages used to create the applications, as well as GitHub. They should also be able to read API documentation.
Task Experience: Users should have experience using all applicable applications listed above.
Frequency of Use: Once the applications are created, bugs and maintenance will be done when brought up by the end user as they report them.
Key Interface Design Requirements that the Profile Suggests: The user should be able to work with any of the programming languages or tools that will be required to create and maintain the applications properly.

Table 2 Web User Profile

User Profile Form 2
Application: Political Tap Web Application
Potential Users: American Citizens
Software and Interface Experience: The user should have familiarity with navigating websites using web browsers. The user should also understand, or be able to, understand how social media sites are organized and how to navigate them.
Experience with Similar Application: Twitter, Facebook, Ballot Sites
Task Experience: Basic interactions such as using the mouse to click on buttons and the

navigation. Using the keyboard to create their account and search for candidates and use the mouse to click their profile and learn more about their candidates.
Frequency of Use: Whenever the user needs to interact with the application which could be daily, monthly or yearly. Typically, the app will be used during election cycles, but it can be used at any time.
Key Interface Design Requirements that the Profile Suggests: The website needs to be secured, easily accessible by the user and the user needs to be able to navigate and filter easily.
User Stories: As a registered voter I want to see the statements of the politicians I am voting for so that I can determine which politician's values align with my beliefs the most. As a registered voter I want to see which politicians will appear on my ballot so that I know who to research and who I will be expected to vote for.

Table 3 Mobile User Profile

User Profile Form 3
Application: Political Tap Mobile Application

Potential Users: American Citizens
Software and Interface Experience: The user should have familiarity with navigating mobile apps. The user should also understand, or be able to, understand how social media apps are organized and how to navigate them.
Experience with Similar Application: Twitter, Facebook, Ballot Sites
Task Experience: Basic interactions such as using their finger to click on buttons and the navigation. Using the phone keyboard to create their account and search for candidates and use their finger to click their profile and learn more about their candidates.
Frequency of Use: Whenever the user needs to interact with the application which could be daily, monthly or yearly. Typically, the app will be used during election cycles, but it can be used at any time.
Key Interface Design Requirements that the Profile Suggests: The mobile application needs to be secured, easy for them to click the buttons and easy accessibility. The user should be able to feel comfortable navigating while using their phone.
User Stories: As a registered voter I want to see the statements of the politicians I am voting for so that I can determine which politician's values align with my beliefs the most.

As a registered voter

I want to see which politicians will appear on my ballot

so that I know who to research and who I will be expected to vote for.

Use Case Diagram

Figure 1 is a use case diagram that displays how the users interact with the mobile and web applications. Additionally, the diagram displays how developers and administrators develop and maintain the application.

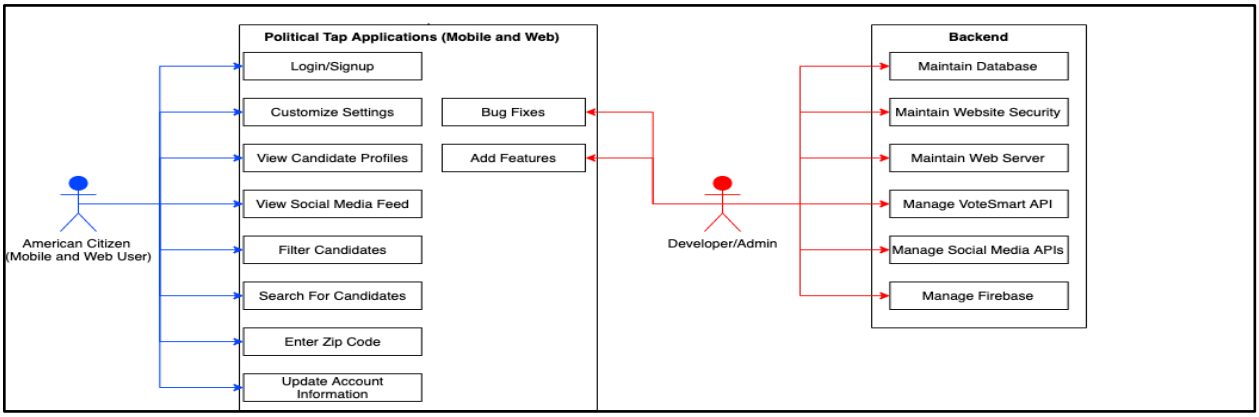

Figure 1 Use Case Diagram

Technical Architecture

Figure 2 represents the technical architecture of Political Tap. Since Political Tap is a web and mobile project, two front-end frameworks were used to develop client-side interactions. The Angular framework was used to develop the web front-end using the traditional web languages such as HTML, CSS, and TypeScript, whereas Flutter was used to develop the mobile front-end with Dart as the primary programming language. The back-end, which is built upon the Python framework of Flask, aggregates responses from two external APIs: Twitter and VoteSmart. The VoteSmart API is used to gather candidate and official information based on key

parameters such as user location. The Twitter API is used to gather tweets by the candidates available on the application. Political Tap's web front-end and Flask back-end is hosted on the cloud platform Heroku which makes the application publicly available to users.

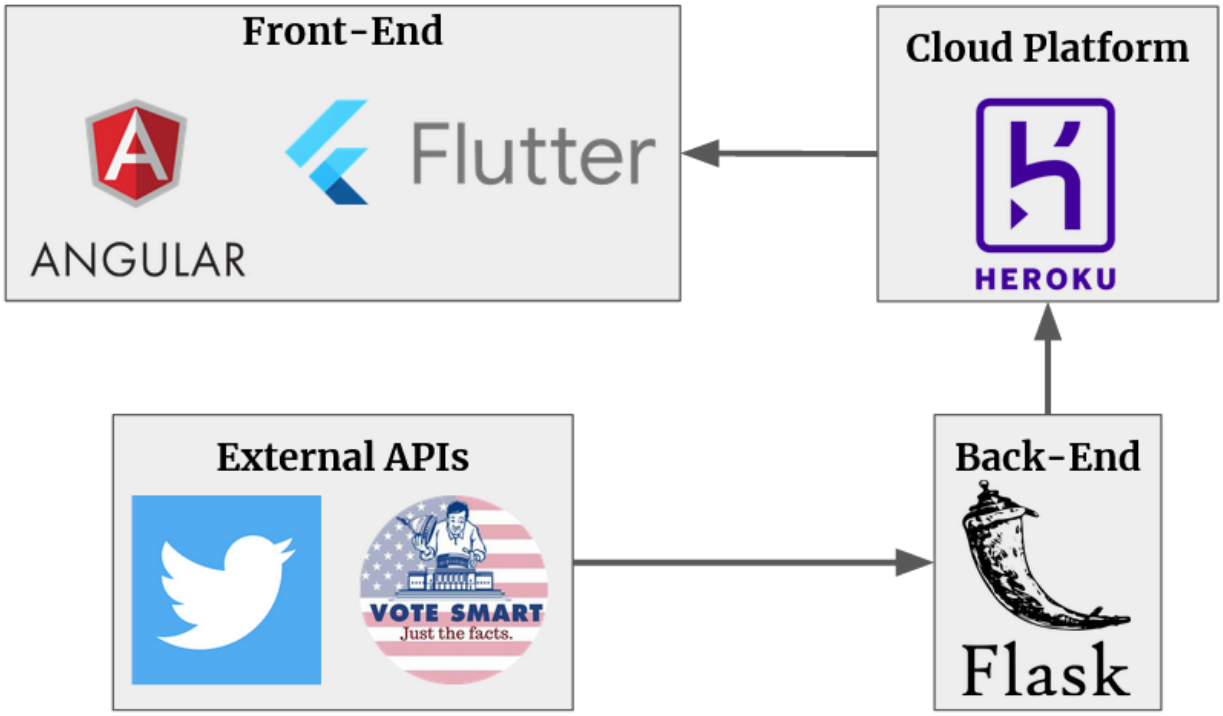

Figure 2 Technical Architecture

Testing Overview

The below sections provide insight into the methodology, scope, and objectives of the testing phase. A testing log is included, which displays various inputs related to the testing results. This report concludes with a discussion on the lessons learned from the overall process and how the team can apply it to future improvements, developments, and testing.

Testing Methodology

The team created issue-specific manual test cases to maintain the integrity of the code base and ensure a positive user experience. Prior to merging changes to the code base, the group individually went through these test cases and ensured proper functionality in addition to issue-specific code reviews. The test cases held requirements for maintaining stable connections to internal and external APIs while also including feature testing against the current project as a whole. This approach allowed the group to maintain the stability of the application while minimizing errors.

In addition to functionality testing, the group incrementally introduced user acceptance testing. The goal was to gathered participants who would be able to test the project and give the development team appropriate feedback relevant to user experience, design, and functionality. These participants ran through tasks and provided their thoughts and feedback. This was done to gain an understanding of user preferences and the ease of use of the application.

Testing Scope

Software Feature Testing Scope

- Endpoint Reliability
    - As new endpoints were developed, the team ensured that the endpoints were tested against numerous types of input to avoid any possible bugs. If bugs were detected, the back-end was modified to remove the respective bugs and the endpoints were re-tested accordingly. Similarly, integration testing was performed to appropriately test the reliability of the APIs as whole.

- Front-end Reliability
    - Front-end specific testing was performed through ensuring that back-end endpoints interacted properly with the web and mobile applications through relevant scenarios. If a problem occurred between these connections, the front-end displays an error message to alert the user of the problem.
    - User input forms were tested with a variety of test cases to ensure that required inputs were available for use throughout the application. Appropriate validation testing on user inputs was also conducted for verifying that the application received only valid inputs.

UAT Testing Scope

The testing scope for UAT included both the mobile (iOS/Android) and web versions. Additional testing was performed with the following features as they include the comprehensive functionality of the product:

- Location input
    - Testing to ensure that users can input their zip code and display the candidates in their area.
- Ballot navigation
    - Testing to ensure that users have a clear understanding of how to navigate the ballot page.
- Profile navigation:
    - Testing to ensure that users can navigate a candidate’s profile and find specific information with ease.

Testing Objectives

The objectives of testing are as follows:

- Test the functionality, features, and overall user experience of the product itself by having users from outside the team participate in testing for both the mobile and web versions of the application.
- Ensure that all features work as intended and fix bugs as they arise during testing by creating issues in the respective GitHub repositories.
- The final product must pass the testing standards outlined in this section.

Test Logs and Procedures

Table 4 outlines the user acceptance test results performed by four users which consisted of friends and colleagues of the development team. The table details the expected and actual outcome of three test cases for each user. The test cases include the navigation of the ballot page, feed page, and candidate profile pages.

Table 4 Test Logs and Procedures

Record	Test Case #	User role/ Input	Expected Outcome	Actual Outcome	Pass/ Fail	Reason for Failure	Date
User 1	1a	Ballot Navigation	User should see their candidates on Ballot	Screen continued loading wheel and blocked input	fail	Invalid zip codes used as input	2/28/21
	1b	Feed Navigation	User should see candidates tweets	Screen continued loading wheel and blocked input	fail	Invalid zip codes used as input	2/28/21

	1c	Profile Navigation	User should see the candidate's detail information	Screen continued loading wheel and blocked input	fail	Invalid zip codes used as input	2/28/21
User 2	2a	Ballot Navigation	User should see their candidates on Ballot	Ballot information did not load	fail	User attempted to use string in zip code	3/14/21
	2b	Feed Navigation	User should see candidates tweets	User saw their candidate's tweets	pass		3/14/21
	2c	Profile Navigation	User should see the candidate's detail information	User saw their candidate's detail information	pass		3/14/21
User 3	3a	Ballot Navigation	User should see their candidates on Ballot	User saw the candidates on Ballot	pass		3/20/21
	3b	Feed Navigation	User should see candidates tweets	User saw their candidate's tweets	pass		3/20/21
	3c	Profile Navigation	User should see the candidate's detail information	User saw their candidate's detail information	pass		3/20/21
User 4	4a	Ballot Navigation	User should see their candidates on Ballot	User saw the candidates on Ballot	pass		3/20/21
	4b	Feed Navigation	User should see candidates tweets	User saw their candidate's tweets	pass		3/20/21

	4c	Profile Navigation	User should see the candidate's detail information	ser saw their candidate's detail information	pass		3/20/21

Budget

Table 5 details the budget required for developing and maintaining Political Tap. The cost considerations revolve around the software and labor costs incurred to develop the solution. The expected future unit prices for the software section reflect the changes in these costs as the free tiers expire and the application grows over time.

Table 5 Budget

Political Tap: Budget Analysis				
ITEM	UNIT, HOURS	UNIT PRICE	EXPECTED FUTURE UNIT PRICE	TOTAL
SOFTWARE				
VoteSmart API Access	1	$45 (biannually)	$45 (biannually)	$90
Heroku Cloud Platform	1	$0	$300 (yearly)	$0
Twitter API access	1	$0	$0	$0
Subtotal				$90

LABOR				
Logo Design	1	$0 (flat fee)	$0	$0
Developer Hours	50	$0	$25	$1,250
Subtotal				$1,250
Total				**$1,340**

Project Timeline

To keep the project on track, the team prepared a project timeline to display the expected progress starting at the fall semester and ending at the IT Expo next spring. This timeline ensures that Political Tap meets the high personal and academic expectations for this academic year. Table 6 shows the project timeline until the final presentation at the IT Expo while also showing a timeline on work projects and assignments for each semester.

Table 6 Project Timeline

Task Number	Task Name	Duration (Days)	Start Date	End Date
1.0	**Project Management and Deliverables**	250	8/24/20	5/3/21

1.1	Fall Semester Assignment 0: Team Members & Project Name	1	8/24/20	8/24/20
1.1.1	Project Name	7	8/24/20	8/31/20
1.2	Project Logo and Branding	7	8/24/20	8/31/20
1.3	Fall Semester Assignment 1: Team Contract	7	8/24/20	8/31/20
1.4	Project Approval	7	8/31/20	9/7/20
1.5	Work Breakdown Structure	7	8/24/20	8/31/20
1.6	Fall Semester Assignment 2: Project Abstract for Tech Expo	10	10/3/20	10/12/20
1.7	Fall Semester Assignment 3: Team Contract Resubmission	10	10/3/20	10/12/20
1.8	Fall Semester Assignment 4: User Profile	7	10/12/20	10/19/20
1.9	Fall Semester Assignment 5: Use Case Diagram	7	10/12/20	10/19/20
1.10	Fall Semester Assignment 6: Draft Report	7	11/2/20	11/9/20

1.11	Fall Semester Assignment 7: Final Fall Semester Report	21	11/9/20	11/30/20
1.12	Presentation Practice	7	11/26/20	12/2/20
1.13	Fall Semester Oral Presentation	1	12/2/20	12/2/20
1.14	Spring Semester Assignment 1: Testing Plan/Report	7	1/11/21	1/18/21
1.15	Spring Semester Assignment 2: Abstract	7	1/18/21	1/25/21
1.16	Spring Semester Assignment 3: Draft Tech Expo Poster	7	1/25/21	2/1/21
1.17	Spring Semester Assignment 4: Final Poster	7	2/1/21	2/8/21
1.18	Presentation Practice	7	2/8/21	2/15/21
1.19	Spring Semester Oral Presentation	1	2/15/21	2/15/21
1.20	Spring Semester Assignment 5: Final Report	7	2/15/21	2/22/21
1.21	Spring Semester Assignment 6: Safe Assign Final Report	7	2/22/21	3/1/21

1.22	Spring Semester Assignment 7: Final Library Copy	7	3/1/21	3/8/21
2.0	**Design**			
2.1	Design Prototype for Web and Mobile	7	8/31/20	9/7/20
2.2	Wireframe Diagrams	7	8/31/20	9/7/20
2.3	Determine APIs to Use	7	8/31/20	9/7/20
2.4	Design API Functionality	7	8/31/20	9/7/20
3.0	**Environment Set-Up**			
3.1	Setup IDE	1	9/4/20	9/4/20
3.2	Import Libraries for Development	1	9/4/20	9/4/20
3.3	Setup GitHub Repositories	1	9/4/20	9/4/20
3.4	Setup Database	21	9/11/20	10/2/20
3.5	Setup Internal APIs	7	9/11/20	9/18/20
4	**Development (Back End and Front End)**			
4.1	Create Front End			
4.1.1	Create Front End Screens for Web	204	9/11/20	4/3/21

4.1.2	Create Front End Screens for Mobile	204	9/11/20	4/3/21
4.2	Create Back End Logic			
4.2.1	Hook up Social Media APIs	14	9/18/21	10/2/21
4.2.2	Create Option for Filtering Candidates	21	10/2/21	10/23/21
4.2.3	Add Account Functionality	49	1/11/21	3/1/21
4.2.4	Create Database for Accounts	21	1/11/21	2/1/21
5	**Testing**			
5.1	Functionality Test for Alpha Version	7	12/1/20	12/8/20
5.2	Functionality Test for Final Version	7	1/4/21	1/11/21
5.3	User Pilot Test	14	1/4/21	1/18/21
5.4	Usability Tests	14	1/4/21	1/18/21
6	**IT Expo**			
6.1	IT Expo Exhibit and Preparation	14	3/30/21	4/13/21
6.2	IT Expo Presentation	1	4/13/21	4/13/21

Analysis of Encountered Problems

One thing the team encountered during the project was trying to gain access to the API required for political information. Since the political information API is the backbone of the

project, ensuring it met all of the requirements was vital before choosing a specific API provider. In this process, countless clarifying emails were required for ensuring the API met all of the requirements for the project. A lot of APIs did not have all the components needed or did not update their information on election data in a timely manner which impacted the team's decisions. Similarly, the representatives that were contacted took a few business days to respond to all of the emails which delayed the progress on the backend. Choosing an API took longer than expected, however, the team is now satisfied with VoteSmart as the initial political API provider.

Another problem encountered was related to client-side filtering on the feed view so that users could organize their feed by election. For example, if a user wanted to organize their feed view to only see candidates running for President, they would enable the filter on the feed view to narrow their feed. However, the two external APIs returned large payloads which took tremendous processing time to translate into organized responses for client-side filtering. In order to properly organize the data for client-side filtering, multiple requests to both APIs were required which was not efficient towards loading times. Similarly, the API responses needed to be combined to organize the data appropriately. Therefore, this feature was abandoned since implementing filtering capabilities on the feed view increased loading times significantly.

Recommendations for Improvements

If given the opportunity to do this project over again, more time would be spent on the planning stages of the project. In choosing the external API for candidate and official information, many alternatives were rejected due to cost constraints. The final decision came between two APIs: VoteSmart and Ballotpedia. Although the most affordable plan for

Ballotpedia was expensive, we would choose this plan over VoteSmart if we had the chance to work on this project again. Working with the VoteSmart API proved to be difficult due to inconsistencies in the data types which the API returned. For example, in some instances where the majority of candidates returned an array of objects, another candidate would return the same property with a single object. Since these two data types were different, one being an array and another an object, this required extra conditions to be added in the back-end to ensure consistent data was returned to the front-end. Due to Ballotpedia's documentation and data consistency, we would choose Ballotpedia over VoteSmart as the primary political API.

If given more time to complete the project, new enhancements would be implemented which could further improve the solution. The main area of improvement would be around the user experience. A possible feature that could be added would be a comparison feature between different candidate's profiles. By allowing a side-by-side view for two different candidates, a user could directly compare multiple candidates to better understand their political differences. This would improve the user experience since users would not have to visit each page separately. Similarly, incremental changes to the user interface based upon user feedback would continue to be added.

Another area in which we could make improvements would be the backend. The backend could be optimized by using multithreading to handle operations that involve multiple candidates. Currently, the backend processes data from one candidate at a time. The use of multithreading would allow multiple candidates to be processed at one time. These changes in the backend would mean a faster and more clean experience for the users.

Conclusion

Lessons Learned

All members of the team learned more about integrating and linking various technical aspects into fluid products. Since the project utilizes a wide variety of technologies, it was important to stay organized with relevant documentation and appropriate GitHub repository management. With all of these new technologies, the team also learned about the best standardization for coding and ensuring that proper styling guidelines were followed. Lastly, our team learned how to adapt to collaborating and working together in a remote environment.

Skills Developed

The team developed a broad array of project management skills through planning and discussions. Working together remotely and holding everyone accountable for action items was vital to success as a team. These skills were further developed through creating relevant project deliverables and setting up meetings each week to meet these deadlines.

Each team member learned technical skills regarding their responsibilities. We became familiar with new technologies that we interacted with throughout the semester as well as other team member's tasks. The team used services such as Heroku and GitHub to gather technical skills related to code management. We also learned new frameworks such as Flask, Flutter, and Angular in developing the functional requirements for Political Tap.

References

Edwards-Levy, A. (2018, October 18). *Young Voters Don't Feel Well-Informed. Some Think That Means They Shouldn't Vote.* Retrieved October 12, 2020, from http://www.huffpost.com/entry/young-voters-not-informed-midterm-elections_n_5bc90345e4b0d38b587661b4

Little Partisan Agreement on the Pressing Problems Facing the U.S. (2018, October 15). Retrieved October 12, 2020, from https://www.pewresearch.org/politics/2018/10/15/little-partisan-agreement-on-the-pressing-problems-facing-the-u-s/

Saad, L. (2018, June 28). *Military, Small Business, Police Still Stir Most Confidence* Retrieved October 12, 2020, from https://news.gallup.com/poll/236243/military-small-business-police-stir-confidence.aspx