Product Description

Introduction

With the 2016 presidential race well under way, the media has had a profound effect on the public's perception of each candidate. As the race continues to narrow down to only a few presidential candidates, it is becoming increasingly important that people understand the values and views each candidate would bring as president. However, major issues exist with the political system in the United States. One problem is that many sources for campaign news can be incredibly biased, especially when a candidate's campaign is being funded by the news organization. Another issue is the lack of participation in politics, especially among young voters, in the United States. Our app, named Check Your Bias (CYB), aims to help combat these problems.

CYB is a mobile application where users will respond to quotes, topics, and issues by using a sliding scale to indicate if they "Agree" or "Disagree". Some topics may include economy, gun control, immigration, etc. The novel feature of CYB is that there is no indication of which candidate supports or opposes the presented issue, removing the bias that the user may have had if they had known a candidate's position. After submitting a response, the user has the option to see where they stand in relation to other candidates' positions. Some users may find that their stance on an issue falls more closely towards a candidate they had no intention in voting for.

Target Audience

CYB's target audience is mainly anybody who is interested in American politics. It is aimed particularly at people who are staunch Democrats and Republicans who only consume media from the party of which they are aligned in. However, the app serves as a way for anybody to find where their interests align with American politicians. It is also possible that CYB can be used by political campaigns in order to get their candidates more publicity. Although not normally an issue for the front runner, getting information out for someone falling behind on the polls can be difficult and CYB can work with the campaigns because it aims to provide a nonbiased tool for every politician.

Problem

The two main political news sources, CNN and Fox, are heavily biased towards the Democratic and Republican parties, respectively. It is common for people to only read news from the source that supports their own political interests and sometimes it can be hard to realize that another candidate from the opposing party better aligns with their viewpoint because of the propaganda and bias that major news sources hold.

Many people are also interested in voting but find it hard to keep up with every single political candidate and their standpoints on every issue. Even those keeping up with debates and news can find it hard to remember the viewpoints of every politician, especially those that are not doing as well in the race because of reduced coverage. CYB aims to consolidate all politicians equally with their standings on major issues, and anonymizes the candidates in order to draw back on possible bias.

Related Work

There are relatively few competing political apps, and most of these are simply news apps with a political bent. Such apps generally would present political news from a certain perspective and users would not be exposed to information that challenges their previously-held beliefs unless they specifically sought it out. Additionally, these sources have a major weakness shared by other news sources: candidates rarely receive equal representation in coverage.

One political app that moves beyond simple news coverage is Brigade, which acts as a politics-focused social media platform. This allows users to share and debate views across a wide range of topics and find users and candidates who align closely with their stated views on issues. However, the aim is clearly more for social discourse than to allow users to gather information, and the lack of anonymity means it may reinforce biases rather than challenge them. In contrast, CYB eschews user-to-user interactivity for convenience, versatility, and variety of information presented. By anonymizing sources of opinions, CYB is able to provide users with views that they might not encounter from a news app or political-social app where they would likely only explore areas they consider relevant to their currently-held beliefs.

Major Features

Issue Rating

The app revolves around our major feature, the ability for users to make a decision about an anonymized political opinion free of media bias. This feature will show users quotes, tweets, and other text that represents a candidate's political stance (for example, a brief summary of a bill they have proposed). The user must then decide whether they agree with the statement on a gradient scale from "strongly disagree" to "strongly agree." Once the user submits their opinion, they are shown more information, including the candidate associated with the text and links to reliable sources to learn more about the topic. Users should never see the same text twice, and the stances presented should represent all candidates and topics evenly.

Candidate Analysis

Once a user has "voted" on some issues presented to them, they can view an analysis of their views and how it aligns with each of the candidates. This will be shown in the form of an interactive graphic, i.e., clicking on graphical elements will show different information. The user should also be able to filter by category/topic or group by political party.

Crowdsourcing

As one of the main uses (features 1 and 2) requires content, we will implement a crowdsourcing option that allows users to submit political content that they would like to see in CYB. Users may submit content in the form of quotes, issues, or topics that will then be sent to the backend server. For our version 1.0 release, this content will need to be approved by the moderators (the developers for the purpose of this class) before it will be shown to other users within the application.

Political Profile

The user will also be able to navigate to a profile section of the app that displays information and data regarding their time spent on CYB. A brief list will show a timeline of the user's voting history, displaying the same cards or issues that they were shown at an earlier time (with their vote). In addition to this, in order to motivate the usage of feature 3 (crowdsourcing), all of a user's submitted content will be displayed here along with the number of times another user has voted on the content. This enables people to see the direct impact that their submissions have in CYB.

Stretch Features

Automated Crowdsourcing

Not requiring a someone on the CYB team to manually approve crowdsourced content would create a far greater volume of content on the platform. To accomplish this, we could check if identical content has been submitted by multiple users before displaying it. Additionally, we could have an option to vote on whether a topic is correct/good, which could filter out any bad content that we accepted.

Generalize for Non-Presidential Elections

An issue CYB aims to combat is the lack of participation in non-presidential elections. An example is the 2014 midterm elections, which had a national turnout of 36.3% of registered voters - a 72-year low. For an election that determines all 435 seats in the House of Representatives and a third of the 100 seats in the Senate, electing candidates whose thoughts and opinions actually represent that of their respective majorities is crucial. However, the media rarely focuses on elections such as these, and many people are left in the dark. With this feature, users could discover who aligns most with their political views without taking the effort to read into every candidate. Whether an election determines the city council, governor, or congress, users would be just a few curated questions away from getting a good idea how to cast their vote. This is a stretch goal because it is dependent on having a high quality implementation of the automated crowdsourcing platform.

Non-functional Requirements

Trust

CYB needs to be a platform that people trust. Considering how rife mainstream media is with reports of lies spread by candidates, the public has become suspicious of not only the candidates themselves but also of the news sources reporting on the elections. In order to get our clients to give CYB a try and use the platform, the trust in the platform must be established. Every facet in the end product must be taken into account when considering the level of trust that the user will place in the app. Both the product's appearance and core functionality (must look appealing and avoid crashing, have a user-interface consistent with the experience the user expects) and the content in the product will be weighed by the end user.

Quality & Quantity of Information

Since there are many sources of information that are competing for the user's attention during election season, the quality and quantity of the content on our product must be satisfactory and of equal or higher quality than our competitor's. This would encourage users to come to CYB for information and help build trust.

This goes along with the rest of the requirements outlined above. Since it is crucial, in order for CYB to succeed, for content in the product to be plentiful, of high quality, and trustworthy, we must make sure that the product as a whole does not hold a sway to any particular side in the political debate. Existing and new content to the product must be screened carefully in order to maintain non-partisanship in the content that the product presents. This will in turn further assist in contributing to the requirements desired.

External Documentation

Very little external documentation will be required to enable users to understand and use CYB. To explain its purpose, we will use our web page, as well as the application download pages on the Google Play Store and the Apple Store. To help users to understand how to use the app, we will develop a UI that integrates help text to unambiguously display functionality.

Process Description

The system will primarily be composed of a database and a mobile client. We will use React to build the mobile client. React is a web component framework which allows us to define our client application as a interacting tree of reusable components. Once the client is built and tested in the browser we can deploy it to the mobile platform via Apache Cordova (formerly known as Phonegap). Cordova places the browser application inside a thin native platform wrapper for both iOS and Android. If time allows, we may explore React Native which bypasses the need for phonegap by producing an entirely native application.

For the database, we elect using Parse to store and interact with information regarding political candidates. If time allows and we expand our mobile client to gathering crowd sourced data, we can leverage the Parse event hook framework to live update the information for all clients. Using Parse removes the need to write a traditional server and authentication routines by giving us the ability to script such functionality into the database system itself.

Finally, for building and testing we will employ Gulp, Node Package Manager (NPM), and NodeUnit. Gulp is a streaming based build system which takes as input a stream of files to be processed by the build. In our project, we will be streaming TypeScript and JavaScript React classes into the build system which will then compile to vanilla JavaScript. The JavaScript will then be streamed to a compressor and minified into a single built file. For dependency management, we will use the popular package manager, Node Package Manager (NPM), which allows any team member to run a single command to obtain all the dependencies listed in this section. To test our TypeScript and

React modules in isolation we will use NodeUnit which provides a simple framework for writing asynchronous compatible unit tests.

Versioning System & Issue Tracking

We have a GitHub repository located at https://github.com/aaronnech/CheckYourBias. This is a public repository, and each of us is given contributor access to the repository allowing us to clone, branch, and merge directly into the repository. Because of our privileges as contributors, none of us will need to fork the repository. Additionally, we have integrated repository notifications into Slack, our application of choice for instant team communication. Whenever a commit is pushed upstream, or an issue is created, or a pull request is accepted, an automated bot will send a message to all users on Slack notifying them of the change. We will track all issues through GitHubs integrated issue tracker, located within the Github Repo. Issues will be organized by certain labels, such as "bugfix", "enhancement", or "feature".

Group Dynamics

We have chosen Sonja to act as the Project Manager (PM). Development will be driven by the SCRUM methodology with short one-week sprints, because this aligns with our customer meetings and the required weekly status reports. Tasks will be assigned based on progress made during the previous week and input from our customer. Todd, Riley, and Nick will work on the backend services, including database design and providing the appropriate data abstractions for the front end team. Geoffrey, Roee, and Aaron comprise the front end team, working on implementing the UI elements in our design and the interactions for each of those elements. Sonja will serve as a full stack engineer, helping out where needed and making sure the back end and front end are well integrated. This ensures that the PM is familiar with the progress of the whole system. Ryan will lead the crowdsourcing feature. Teams were chosen based on individual preference. Although specific roles have been assigned, we will allow a member to switch tasks – under reasonable conditions – if they are unsatisfied in their current position. Each team is responsible for writing tests for their code. Disagreements will be settled democratically during our weekly team meetings or on the #general Slack channel if urgent.

Timeline

After the design specification is completed as a group, the front end team, consisting of Geoffrey, Roee, and Aaron, will develop the user-facing aspects of the issue selection and candidate analysis features. Rudimentary versions of these UIs should be completed by the zero-feature release, while complete versions should be available by the beta release. Meanwhile, the back end team, consisting of Todd, Riley, and Nick, will work on the back end of these same features, including a database of issues and a database of user history, as well as the connections between these databases and the front end elements, which should as well be completed for the beta release. Both teams will then move to the issues of crowdsourcing and user profiles, the corresponding aspects of which should be completed by the feature-complete release. From this point, teams may work on an advanced crowdsourcing system and potentially the ability to expand to general elections if time permits.

January 29th — Software Design Specification Complete

February 2nd — Zero-Feature Release Check-in

- Basic UI framework complete for issue selection/rating
- Basic UI framework complete for candidate analysis

February 16th — Beta Check-in and Start of Integration Testing

- Backend for issue selection/rating complete
  - Pre-built database of issues and links to sources and info
  - User-based history of responses by issue
- Integration of candidate analysis UI with user history

February 23rd — Feature-Complete Release Check-in and Integration Testing

- Crowdsourcing of 'issues' complete
  - UI to allow users to submit issues
  - Backend to process and filter submitted issues
- User profiles complete
  - Integration with voting history and issue submission databases
  - UI to view user profiles

March 1st — Final Release Check-in and Start of Final Testing Phase

- Stretch features implemented if time permits

March 8th — Final Release

Risk Summary

1. **Content**
   Likelihood: Medium
   Impact: High
   Summary / Evidence: Content is a major component of Check Your Bias and it is incredibly important that this information be both accurate and complete. The content will be directly related to the overall quality of our application.
   Overall Plan: We plan to review all content that is put into our app as a team in the beginning. That means that before content can show up in the feed for users, it has to be approved by admins who will check the accuracy and completeness.
   Detection: This was discussed about, but all content will be moderated before it can show up in a user's feed.
   Mitigation Plan: Should this occur, users can report something as incorrect with a few simple taps when voting on a topic or issue.
2. **Crowdsourcing**
   Likelihood: Medium

Impact: Medium
Summary / Evidence: Crowdsourcing will allow the product to grow significantly and provide a platform for many people to share their knowledge and expertise in the topic, but it also produces significant challenges. Content that users supply to the app must be vetted and screened thoroughly and pass quality guidelines. It will be difficult balancing the effort and efficiency of undergoing this screening process.
Overall Plan: All content must be initially reviewed by moderators, but in order for this to scale there must exist a limit to how much one can spend on a single topic before making a decision.
Detection: Moderators will need to be careful that the time spent approving or disapproving content is of high quality so that a short amount of time can be spent on each submission while also insuring that only high quality content is approved.
Mitigation Plan: Should this fail to happen, then users will be able to flag content for review (as discussed in the previous risk).

3. **Content Creation**
Likelihood: Medium
Impact: Medium
Summary / Evidence: A major feature of our end product is providing users the option to answer a set of questions that will help the product guide them towards a candidate whose opinions match their own. Since this process and criteria will at first be created manually by us, the creators of the products, there is a risk that our bias or incomplete understanding of the political situation will impact the quality of this initial survey and will thus degrade the overall quality of the product.
Overall Plan: In addition to all content being approved by a set of moderators, each moderator should carefully consider whether the submission is inherently bias towards a particular candidate or viewpoint.
Detection: Detection can happen in 2 places: another moderator will catch it before it is approved or a user will flag the content.
Mitigation Plan: If a moderator finds content to be inherently biased, then they may edit the content to be better suited for the content feed.

4. **Lack of Engagement**
Likelihood: Low
Impact: High
Summary / Evidence: A user may come into our app and only answer a few questions and then stopping. This does not give us enough evidence to make any conclusions about who the user most closely relates to in terms of the candidates.
Overall Plan: We will display a message saying that there is not enough information yet to make any conclusions about the user's political views if the user tries to access the analysis page.
Detection: This can be detected if the user tries to access his or her political profile page before they have ranked enough topics or issues.
Mitigation Plan: The main way of mitigating this risk is to tell the user they must answer X questions or topics before we have enough information to make a conclusion about their

political profile. If a user has already taken the time and effort to download our application, then it is likely that asking them to spend a few more minutes rating topics will not deter them away from CYB. Unfortunately, it would be incredibly difficult to make any conclusions based on such little data.

5. **Rated Content**
   Likelihood: Medium
   Impact: Medium
   Summary / Evidence: As a user rates content in his or her feed, it is likely that they will encounter a category that is of not much importance to them. For example, if I have an extreme bias against Trump and disagree on every single one of his viewpoints – then it is highly unlikely for me to have my views changed (the main idea of CYB is to help educate voters about their views and how they relate to the candidates' views). If I rate a quote of Trump's that happens to be one of his main campaign points as "Highly Disagree", then I probably won't agree with many of his other points.
   Overall Plan: This can be mitigated by using a user's vote on a topic as an indicator whether he or she agrees with a candidate before the user's political profile is made. The server can then send topics to the user that help complete the profile rather than sending more topics on someone or something that they do not care for.
   Detection: Detection of this risk is a difficult task. We can look at the likelihood of a user voting "Highly Disagree" on a topic given that they also voted "Highly Disagree" on this other topic and use this as a metric in determining whether or not to show them.
   Mitigation Plan: Largely discussed above, we want to show users topics that help complete our political profile of them. This means that we should have a diverse set of content that can be utilized to help find the right topics to present to the user.

Use Cases

Knowing Who To Support In The Election

Goal	A user wants to know what is being said by politicians, and how much he or she agrees with them
Primary Actor	A college student who barely follows politics
Scope	Check Your Bias (CYB) app
Level	User
Precondition	User knows about the CYB app from primary or secondary sources.
Success end condition	User has decided how much they agree or disagree with at least one statement
Failure end condition	User has not decided how much they agree or disagree with at least one statement
Trigger	User opens the application, possibly for the first time

Main success scenario	1. User logs into their account or registers for a new account 2. Application displays a quote said by a politician. This quote does not include attribution 3. User indicates to the application his or her level of agreement with the quote 4. The application reveals the attribution of the quote to the user 5. Steps 2-4 can be repeated as desired
Extensions	1. User logs out or quites the app before indicating their level of agreement with the quote (a) On the user's next login, the application displays the same quote to the user, without attribution (b) Repeat step 1a until Step 3 in *Main success scenario* is satisfied
Variations	1. User can choose to skip the quote (a) The application reveals the attribution of the current quote (b) The application reveals a new quote to the user. See Step 2 (c) Steps 1a-b are repeated until the user does not skip the quote 2. User can choose to indicate "undecided" (a) The application continues as usual

Knowing Where You Stand, And With Whom You Stand

Goal	A user wants to know where they stand on various political issues, and the set of candidates who have similar views
Primary Actor	An adult who is passively interested in politics
Scope	Check Your Bias (CYB) app
Level	User
Precondition	User has repeated the steps in *Main success scenario* of the first use case enough times to be familiar with that use case of the application
Success end condition	User knows where they stand on various political issues, and/or User knows the names of the candidates who have similar views
Failure end condition	User is unable to know where they stand on various political issues, or User does not know the names of the candidates who have similar views
Trigger	Various external; e.g. election date approaching, watched televised debate

Continued on next page...

Main success scenario	1. User opens the application 2. User goes to the Analysis view of the application 3. User filters results by topic. Example: immigration 4. Application displays a graph of how similar the user's immigration views are to each candidate 5. (Optional) User clicks on a candidate to retrieve a quick summary of that candidate's views on immigration 6. User can repeat Steps 2-6 as desired
Extensions	1. No candidate shares similar views to the user on a given topic (a) The application still displays applicable candidates for that topic (b) (Optional) User clicks on a candidate to retrieve a quick summary of that candidate's views on the given topic (c) Steps 1a-b are repeated until the user does not skip the quote 2. No candidates have any quotes registered in the application that pertain to the selected topic (a) Application displays to the user a friendly message indicating that no candidates have been found for the selected topic (b) (Optional) User taps a Back button to go back to the Analysis view of the application 3. The user has not indicated enough their level of agreement with quotes relating to a particular topic. As a result, the application cannot display candidates whose views on such topic match those of the user (a) Such topic will not be made available to the user
Variations	None

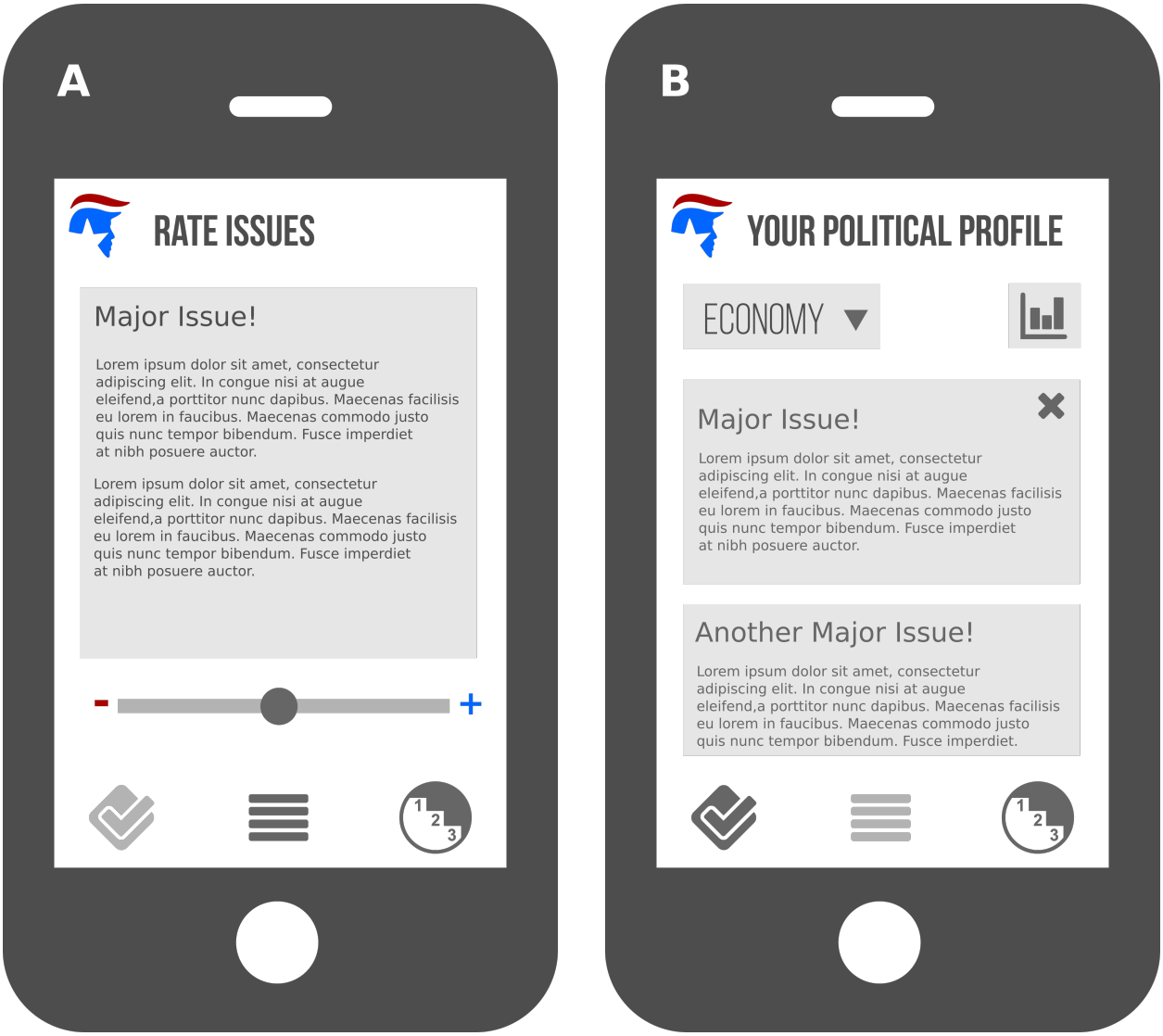
Adding Content To The Application

Goal	Introduce statements made by a candidate not previously registered in the application database
Primary Actor	A campaign volunteer
Scope	Check Your Bias (CYB) app
Level	Contributor; user
Precondition	The campaign volunteer has created their account
Success end condition	The content is created and saved in the application database
Failure end condition	The content is not created, or is not saved in the application database
Trigger	User wants to let other CYB users know more about a candidate.

Continued on next page...

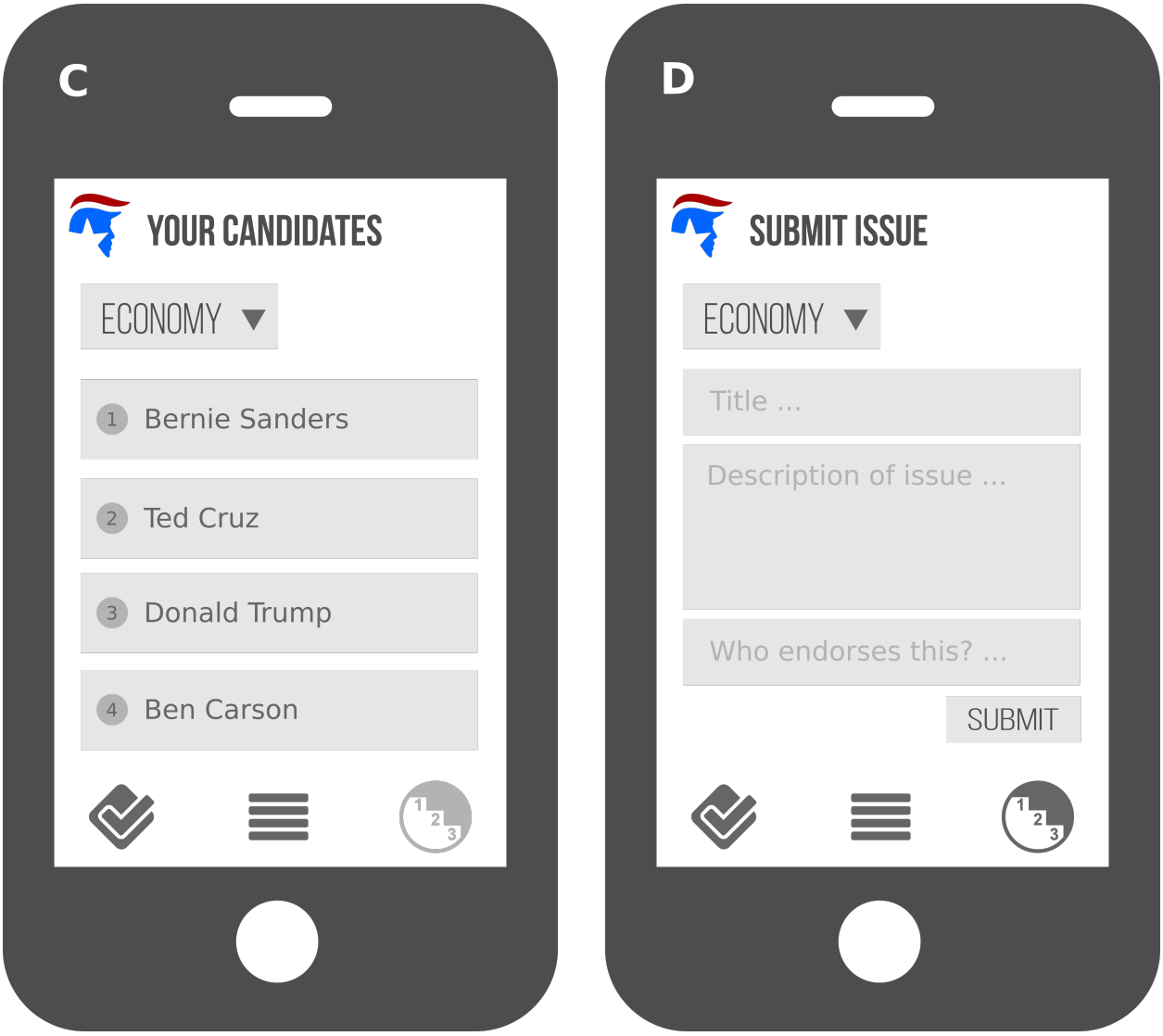
Main success scenario	1. User logs into the app 2. User goes to the Content Creation page 3. User enters content related to the candidate they support; e.g. a summary of a bill that said candidate has passed 4. User chooses a category and enters a link to a reliable source elaborating on their topic 5. User taps the Submit Content control on application 6. The application approves the user's submission. Note that significant time may pass between Steps 4 and 5
Extensions	1. User has not yet created an account (a) User goes to registration page (b) User completes registration (c) User continues with Step 2 2. Application rejects the user's submission, for any reason (a) Application notifies user of the rejection with the reason (b) Application presents user with two choices: i. Edit and Resubmit A. User chooses this option B. User is presented with screen to edit their content C. User taps the Resubmit Content control on application D. Return to 2 ii. Delete A. User chooses this option B. User is presented with confirmation C. If user taps yes, application deletes the content D. If user taps no, proceed to 2a 3. User fails to enter the required information (a) Application prevents user from completing Step 5 until user has successfully completed 3a. and 4a.
Variations	1. User does not want to submit content, but save it as "Draft" status (a) Application informs user that their content has been saved as draft (b) On subsequent logins, user is able to retrieve their saved drafts 2. User wants to start over with their submission (a) User presses a "Reset" button (b) Application presents the user with a confirmation dialog, asking if the user really wants to start over (c) If user selects yes, all content the user has entered is deleted (d) If user selects no, application closes confirmation dialog box and user can resume creating their submission.

UI Diagrams



A (left) Rate Viewpoints. The user is presented with various viewpoints on issues within context of topics such as economics or gun rights. The user then rates each their alignment with the opinion on a continuous scale from negative to positive. The user can then swipe to the next viewpoint.

B (right) The user can see the culmination of all their ratings from the Rate Viewpoints flow in a topical feed, along with various visualizations for how they align politically. They can also explore the origin of each viewpoint they have aligned to.



C (left) Your Candidates. The user can view a list of candidates which most closely align with their political profile on various topics. Candidates are ranked in a scrollable list.

D (right) Submit Issue: The user can submit a quote or issue that may be added as a prompt in Rate Viewpoints (A). They submit a title, description of the issue along with the candidate who has previously endorsed the viewpoint. It will be added either through positive user ratings based on previous submissions or by a member of the CYB team manually approving it.

Design Changes and Rationale (2/19)

Major Features

We did not make any changes to our major feature requirements and have most of the functionality described about them in this document implemented for the beta release.

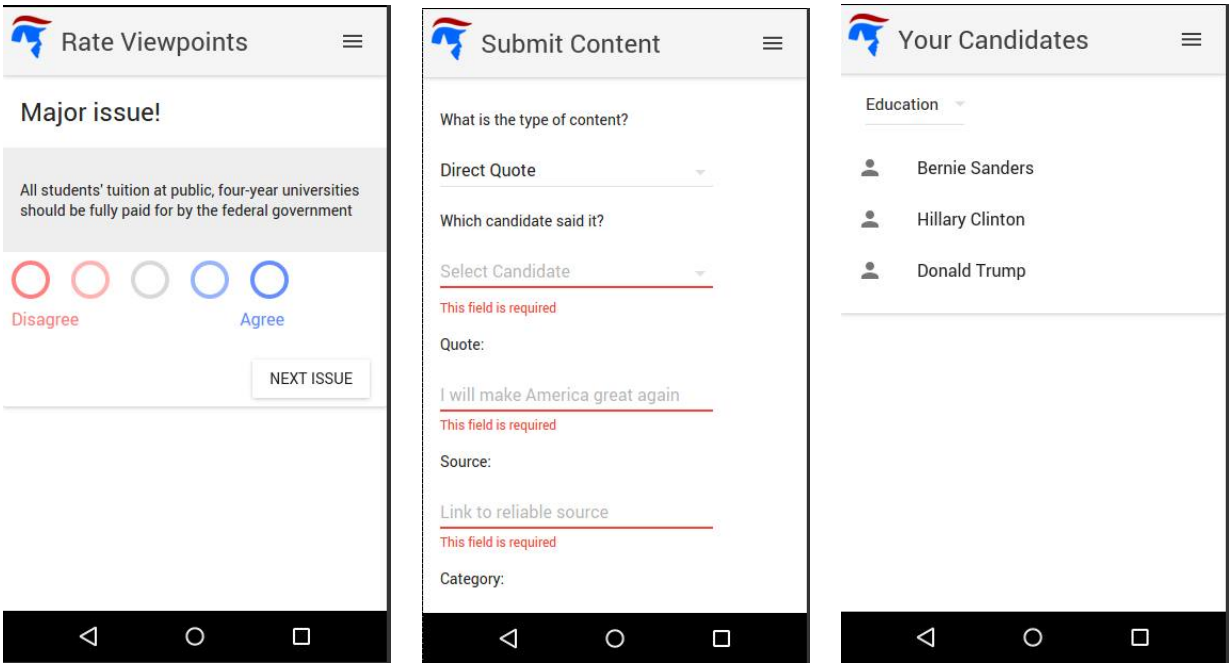
Parse

Fairly on in the project, it was announced that Facebook was discontinuing support for Parse at the end of 2016. It was decided as a group that we should move away from Parse in order to develop skills that are relevant well after this class ends. As a team, we decided that we should move our implementation over to Firebase, which acts very similar to Parse but will (hopefully) not be discontinued in the near future.

Timeline

While our timeline has mostly stayed the same as we have met all of our deadlines that we set as a team, we have fallen slightly behind on the integration of different systems. Specifically, the candidate analysis integration, which should of been completed by February 16th, will not be completed until our Beta release. This will postpone some of the testing that needs to take place in the application by a few days to a week depending on the severity of the delay. This should not have any effect on our beta release or our feature complete release during the following week.

User Interface



We made some major changes to the overall design of the UI. Notably, we changed the layout and number of fields in the crowdsourcing view.

Design Changes and Rationale (2/26)

iOS Support

After much effort, we have decided to drop iOS support on this project. In the beginning we chose to use a library called PhoneGap in order to port our web application to both Android and iOS. Unfortunately, we have had to drop support for iOS because some of the main functionality of the app does not work on iOS. Instead of spend engineering hours on debugging and fixing the issue, we have decided to focus our efforts on making the application better / implementing stretch features.

Updated Product Schedule:

(days to complete in parenthesis next to each task)

Class Due Dates (Week of)	Front End Team	Back End Team	Crowdsourcing Team
1/26 ● Submit Software Design Specification (2/1) ● Submit slides for SDS (2/2)	Familiarity with React and Gulp build system. Learn Material-UI for building React components. Aaron: Fill team in on development stack and architect major modules. (2)	Explore Firebase UML Diagram Nick, Todd: Learned about Parse and Firebase to determine the pros and cons of each(4)	Sequence Diagrams Sonja: Sequence Diagram for user voting (2) Ryan: Sequence Diagram for crowdsourcing. Wrote up docs in Latex and made final changes (2)
2/2 ● Submit zero-feature release (2/8)	Team: A React component for each major feature Aaron: Set up major application scaffolding and build infrastructure to support React front end. Continuous testing bot (3) Geoffrey: Implement “Rate Viewpoints” component screen (1). Implement “Your Candidates” screen (1). Various UI touch-ups (3). Work on a slider for voting (later deprecated) (2). Roee: implement the Political Profile page. Design the cards that will hold the information	Write typescript classes Riley: Preliminary implementation of User class (1) Nick: Preliminary implementation of User and Candidate class (1) Todd: Preliminary implementation of Issue, Category classes (1)	Sonja: Built crowdsourcing content submission form (3) Ryan: Designed and implemented the product website to showcase our work to users and developers. Updated docs to reflect the changes that we made up until this time (3)

	displayed on the page and pick the appropriate Material UI elements for the task (3)		
2/9	Implement interaction for most components. Geoffrey: Document components (1). Mock up voting and getting new issues (2-3). Roee: Adding filtering to the political profile page via constants hard coded into the app (4)	Test the backend thoroughly and ensure smooth integration with frontend Riley: Implement User methods, most significantly the method to get a new issue for a user such that candidates are equally represented (3) Nick: Implement User methods, tweaking the method to get a new issue for a method previously written by Riley and writing half of the ranking method. (3) Todd: Implement Candidate, Category, Issue methods (2)	Sonja: Worked on the crowdsourcing approval component, including a 5 point selector that was used in the rate viewpoints component. (2) Ryan: Make crowdsourcing submission write to Firebase. Wrote code to surface data from subcomponents to the parent component. (2)
2/16 ● Submit beta release (2/19)	Implement Political Profile Add candidate avatars Improve Your Candidates Geoff: Sync Rate Viewpoints with	Improve utility and integration of backend classes Riley: Add features to account for approval of issues (2), create utility methods for categories and candidates (getAll and conversion	Sonja: Normalized styling with rest of app, migrated crowdsourcing to work with Categories and Candidates pulled from Firebase instead of hardcoded data. (2) Ryan: Added a check

	backend, allowing displaying of issues and voting (2). Fixes and improvements for UI issues on Rate Viewpoints (2). Work with backend on issues with User class (3-4). Moving from mock data to real data pulled from database in “Your Candidates” screen (2-3). Work with Aaron on getting webdriver tests set up (1). Aaron: Set up webdriver testing framework, integrate with gulp build system. Setup stand alone selenium runtime. Multiple bug fixes with integration of Model classes with React frontend such as category fetching for issue submission. (2) Roee: adding candidate avatars to Political Profile page. Connecting the political profile page to DB category and candidate data (2)	between ids and names) (2) Nick: Updated tests for User methods and worked on utility methods for categories and candidates. (1 each) Todd: Changed return structure of getRankings to make it not depend on array indices for candidate ids (2). Wrote corresponding tests.	for valid URLs in the crowdsourcing form, updated the UI for required fields to be less intrusive by using red underlines rather than text. (2) Aaron: Categories and candidates pull from firebase instead of being hardcoded constants. (1)
2/23 ● Submit feature complete release (2/26)	Option to skip issues when voting Show how similar a candidate is to the user	Improve ranking algorithm, continue supporting frontend team with additional features Riley: Implement	Sonja: Added ability to add multiple sources and candidates to the crowdsourcing form. Made the news source URLs

	Geoffrey: Implement showing candidate information after voting on an issue (5). Worked with backend team to retrieve issues independent of category (2). Implement skipping issues (and bugfixes for that), with collaboration on backend team (6). Reworking backend to retrieve candidates' avatars (3). Begin work on showing additional candidate ranking data in "Your Candidates" (3). Begin attempting to display categories in sorted order in "Your Candidates" (2-3). Roee: providing Candidate avatars to all app pages. Currently attributing quotes on Political Profile page (2) Aaron: Extensions to webdriver testing framework and authentication infrastructure to allow testuser login to access and vote on production data, while cleaning up temporary data. (2)	improved ranking algorithm (1), improved tests (2), fix errors in User method edge cases (2), normalize ranking results to more grokkable values (1), add ability to get next issue from any category (1) Nick: Continued to support the front-end team in integrating the back-end and front-end code. Also worked on streamlining some back-end code that wasn't necessary for use with Firebase. (5) Todd: Added more utilities (getAllCandidatesSorted, getAllCategoriesSorted) (2). Changed the structure of data returned to resolve front-end issues (1).	clickable in RateViewpoints (2) Ryan: Added module to determine if the user has an internet connection in order to show the correct error in the crowdsourcing form. Updated SRS and SDS docs (2).
3/1 ● Submit release	Webdriver Tests	Fix inconsistencies in data representation	Sonja: Conducted a user study. Fleshed

candidate (3/4)	Geoffrey: Finish work on showing additional candidate ranking data in “Your Candidates” (2). Show profile pictures of candidates in “Your Candidates” (2). Finish work on displaying categories in sorted order in “Your Candidates” (3). Complete code review for Roee (1). Various infrastructure changes (2). Roee: Fix bugs in your Political Profile page that arose as a result of switching to a purely DB-pulling for info and getting rid of the hard-coded issues (3) Aaron: Wrote another webdriver test featuring issue voting. (1)	Riley: Add author field to issues to make handling of direct quotes more convenient (1) Nick: Added back-end functionality to skip issues and filtered content for approving issues so that submitters are unable to approve their own submissions. (3)	out the Group Retrospective document. (1) Ryan: Conducted a user study. Wrote the user study doc and analysis. Added features to improve the crowdsourcing experience. (3) Aaron: Conducted 2 user studies (1)
3/8 ● Submit final release (3/8)	Individual Retrospective	Individual Retrospective	Individual Retrospective