PolitiPulse

By: Adam Botens and Austin Edwards

Project Name

The project name will be PoltiPulse, which is short for Political Pulse. The project itself will help its users stay in the know for elections by tracking candidates voting trends, campaign promises, financial data, and the users' political leanings. In short, PoltiPulse gives users a quick pulse on the state of the government and the current election.

Team Members Names

Team Member Name	Role
Adam Botens	Developer, Architect
Austin Edwards	Developer, Architect

Revised Requirements List

1. **(Onboarding) Login Form**:
    1.1. The login form will be linked to from the homepage and will consist of:
        1.1.1. A header that says “Login Form”
        1.1.2. A label to identify the email input box
        1.1.3. An input box for the user’s email
            1.1.3.1. There will be a placeholder “johnDoe@email.com”
        1.1.4. A label to identify the password input box
        1.1.5. An input box for the password
            1.1.5.1. There will be a placeholder “Password”
    1.2. A button labeled “Login”
            1.2.1.1. Will be aligned center under the password input box
            1.2.1.2. On click, it will query the account database for matching user credentials
                1.2.1.2.1. If a match is found, the user returns to the homepage
                1.2.1.2.2. If no match is found or the password is incorrect, a display message will appear above the login button
                1.2.1.2.3. Suspicious login attempts (e.g., repeated failed login attempts from the same IP address) will be handled through

CAPTCHA challenges provided by Firebase Authentication

1.2.2. A button under the Login button labeled “Register”

1.2.2.1. On click, it will open the registration page in the same tab

1.3. A link to Google sign-in

1.3.1. On click, it will open a popup to Google Auth

2. (Onboarding) Registration Form:

2.1. The registration form will be linked to from the login page and will consist of

2.1.1. A header that says “Registration Form”

2.1.2. A label to identify the email input box

2.1.3. An input box for the email

2.1.3.1. The placeholder in the email box will be “johnDoe@email.com

2.1.3.2. The input box will accept only valid emails

2.1.3.2.1. Emails must contain an @ symbol

2.1.3.2.2. Validation will be done using HTML input attributes

2.1.3.2.3. On invalid entries, a display message will appear above email label

2.1.4. A label to identify the username input box

2.1.5. An input box for the username

2.1.5.1. The placeholder in the username box will be JohnDoe

2.1.5.2. The input box will only accept valid usernames

2.1.5.2.1. Must contain at least one capital letter

2.1.5.2.2. Must contain at least one undercase letter

2.1.5.2.3. Must contain at least two numbers (0-9)

2.1.5.2.4. Must contain at least one special character (!, @, #, $, %, ^, &, *)

2.1.5.2.5. Must be at least eight characters long

2.1.6. A label to identify the password input box

2.1.7. An input box for the password

2.1.7.1. Passwords must be at least eight characters long

2.1.7.2. Passwords must contain a special character (!, @, #, $, %, ^, &, *)

2.1.7.3. Passwords must contain one capital letter

2.1.7.4. On invalid entries, a display message will appear above email label

2.1.8. A label to identify the confirm password input box

2.1.9. An input box to confirm the password

2.1.9.1. The input box will have “Password” as a placeholder

2.1.9.2. The input box will accept only valid passwords

2.1.9.2.1. The input must match the input from the first password insertion

2.1.9.2.2. On invalid entries, a display message will appear above email label

2.1.10. A label to identify the State dropdown menu

2.1.11. A dropdown menu for the State Dropdown menu

2.1.11.1. Lists all 50 states in their aberrated forms

2.1.11.2. Once the User chooses a state, opens Congressional District Menu

2.1.12. A label to identify the Congressional District Menu

2.1.13. A dropdown menu for the Congressional District Menu

2.1.13.1. Lists all Congressional Districts based on State Choice.

2.1.14. A label to identify the Birthday Year box

2.1.15. An input box to confirm the Birthday Year

2.1.15.1. The input box will have “XXXX” as a placeholder

2.1.15.2. The input box will accept only valid years (Current Year - 100 to Current Year - 14)

2.1.16. A label to identify the Birthday Month box

2.1.16.1. The input box will have “XX” as a placeholder

2.1.16.2. The input box will accept only valid months (01-12)

2.1.16.3. Once filled with a valid label, it unlocks the Birthday box.

2.1.17. A label to identify the Birthday box

2.1.18. An input box to confirm the Birthday

2.1.18.1. The input box will have “XX” as a placeholder

2.1.18.2. The input box will accept only valid days

2.1.18.2.1. Conditional on Month Set

2.1.18.2.1.1. Jan, Mar, May, July August, Oct, Dec = 31

2.1.18.2.1.2. Apr, June, Sep, Nov = 30

2.1.18.2.1.3. Feb = 28 (+1 if leap year)

2.1.19. An input box to confirm the Birthday

2.1.20. A header text that says “Finish”

2.1.21. A button labeled “Register”

2.1.21.1. It will be aligned bottom-right from the confirm password input box

2.1.21.2. On click, it will query Firebase Authentication

2.1.21.2.1. If an email match is found, it will display a message stating that the email is already in use

2.1.21.2.2. If a username match is found, will display a message stating that the username is already in use

2.1.21.3. If no match is found

2.1.21.3.1. The two passwords are compared

2.1.21.3.1.1. If they are equal

2.1.21.3.1.1.1. The user account is created, logged in, and returned to the homepage

2.1.21.3.1.1.2. Email, Username, Password, State, Congressional District, Birthday Year, Month, and Day are logged in the Account Info Database.

2.1.21.3.1.1.3. Auto transfer back to the login page.

2.1.21.3.1.2. If they are not equal

2.1.21.3.1.2.1. A message will display stating that the passwords do not match

2.1.22. A bordered label with a link back to the login page

2.1.22.1. On click, it will open the login page in the same tab

3. (Navigation) Header

3.1. A button with the label “Home”

3.1.1. Takes the user back to the homepage

3.2. A drop-down menu with the label “House of Representatives”

3.2.1. A menu item with the label “Currently Elected”

3.2.1.1. Takes them to “Current House of Representatives” page

3.2.2. A menu item with the label “House Election”

3.2.2.1. Takes them to the “House Election” page

3.2.3. A menu item with the label “Upcoming Business”

3.2.3.1. Takes them to the “House of Representatives Upcoming Business” page

3.2.4. A menu item with the label “Recent Business”

3.2.4.1. Takes them to the “House of Representatives Recent Business” page

3.3. A dropdown menu with the label “Senate”

3.3.1. A menu item with the label “Currently Elected”

3.3.1.1. Takes them to “Current Senate” page

3.3.2. A menu item with the label “Senate Election”

3.3.2.1. Takes them to the “Senate Election” page

3.3.3. A menu item with the label “Senate Business”

3.3.3.1. Takes them to the “Senate Upcoming Business” page

3.3.4. A menu item with the label “Recent Business”

3.3.4.1. Takes them to the “Senate Recent Business” page

3.4. A dropdown menu with the label “User”

3.4.1. A menu item with the label “User Profile”

3.4.1.1. Takes them to the “User Profile” page

3.4.2. A menu item with the label “Political Spectrum”

3.4.2.1. Checks if they have taken the Political Spectrum

3.4.2.1.1. If they have not

3.4.2.1.1.1. Takes them to the “Political Spectrum Questionnaire” page

3.4.2.1.2. If they have

3.4.2.1.2.1. Dialog box Asks if they want to retake it

3.4.2.1.2.1.1. If the user clicks yes

3.4.2.1.2.1.1.1. Takes them to the “Political Spectrum Questionnaire” page

3.4.2.1.2.1.2. If the user clicks no

3.4.2.1.2.1.2.1. Takes them to “User Info” page

3.5. A button with the label “About”

3.5.1. Takes them to the “About” page

3.6. A button with the label “Contact”

3.6.1. Takes them to the “Contact” page

4. (WP) House of Representatives - Current Elected

4.1. Drop down list to sort them by state

4.1.1. Each representative will have a sub-page with a list of the most recent actions and most important bills.

4.1.1.1. Focus on sponsored bills as a highlight.

4.1.1.2. General Political Compass Gauge from Voting Records

5. (WP) House of Representatives - House Election

5.1.1. Drop down list to sort them by state

5.1.1.1. Drop down list to sort them by district

5.1.1.1.1. Election Prospects for the next election cycle

5.1.1.1.2. Speeches and election promises

5.1.1.1.3. Financial Info from their donors.

5.1.1.1.4. General Political Compass Gauge from info provided.

6. (WP) House of Representatives - Upcoming Business

6.1. Show in an image list format of all current bills up for vote next session

6.1.1. Sub Area for those in committees

6.1.2. Sub Area for those not brought to the committee yet.

6.1.3. General Political Compass Gauge from info provided.

7. (WP) House of Representatives - Recent Business

7.1. Show an image list of all bills voted upon for the last three sessions.

7.1.1. A section for passed bills

7.1.2. A section for rejected bills

8. (WP) Senate - Current Elected

8.1. Drop down list to sort them by state

8.1.1. Each Senator will have a sub-page with the list of the most recent actions and most important bills.
8.1.1.1. Focus on sponsored bills as a highlight.
8.1.1.2. General Political Compass Gauge from voting records

9. (WP) Senate - Senate Election

9.1.1. Drop down list to sort them by state
9.1.1.1. Drop down list to sort them by district
9.1.1.1.1. Election Prospects for the next election cycle
9.1.1.1.2. Speeches and election promises
9.1.1.1.3. Financial Info from their donors.
9.1.1.1.4. General Political Compass Gauge from info provided.

10. (WP) Senate - Upcoming Business

10.1. Show in an image list format of all current bills up for vote next session
10.1.1. Sub Area for those in committees
10.1.2. Sub Area for those not brought to the committee yet.
10.1.3. General Political Compass Gauge from info provided.

11. (WP) Senate - Recent Business

11.1. Show an image list of all bills voted upon for the last 3 sessions.
11.1.1. A section for passed bills
11.1.2. A section for rejected bills

12. (UI) Bills

12.1. Bills will be displayed in image + title button
12.1.1. Once clicked, two versions of the bill will come up on screen
12.1.1.1. One will be the original bill
12.1.1.2. Second will be the summary bill generated in plain English by the LLM
12.1.2. Each bill will be generally charted onto the political compass based upon its contents.

13. (PS) Political Spectrum Questionnaire

13.1. A paragraph explaining that upon clicking a link, the user will be redirected to https://www.politicalcompass.org/ to take a political spectrum questionnaire. The user will be requested to input their results below.
13.1.1. A button labeled “Political Compass”
13.1.1.1. On click will redirect the user to https://www.politicalcompass.org/
13.1.2. A label that says “Economic Left/Right”
13.1.2.1. An input box next to the label with placeholder “0.00”
13.1.3. A label that says “Social Libertarian/Authoritarian
13.1.3.1. An input box next to the label with placeholder “0.00”
13.1.4. A button that says “Submit”

13.1.4.1. Upon clicking will send information to the user's profile information in the database

14. (WP) Contact Page

14.1. Will be linked to from the questionnaire and from the website navigation panel

14.1.1. A label that states "Contact Us"

14.1.2.

14.1.3. A label that identifies the "Name" input

14.1.4. An input box for user's name

14.1.4.1. The placeholder text will be "john"

14.1.5. A label that identifies the "Email Address" input

14.1.6. An input box for the user's email

14.1.6.1. The placeholder in the email box will be "johnDoe@email.com

14.1.6.2. The input box will accept only valid emails

14.1.6.2.1. Emails must contain an @ symbol

14.1.6.2.2. Validation will be done using HTML input attributes

14.1.6.2.3. On invalid entries, display message will appear above email label

14.1.7. A label that identifies the "Subject" input

14.1.8. An input box for the subject

14.1.8.1. Must not be longer than 25 characters

14.1.9. A label that identifies the "Message input

14.1.10. An input box for the message

14.1.10.1. Must not be longer than 500 characters

14.1.11. A CAPTCHA/ReCAPTCHA will be under the final input box

14.1.11.1. Google Recaptcha provided by Google

14.1.12. A submit button in the bottom right

14.1.12.1. Upon clicking:

14.1.12.1.1. Will send to a shared email designated for use of the website

15. (WP) User Profile

15.1. Linked to from the website's navigation panel

15.1.1. A header that says "User Profile"

15.1.2. A form

15.1.2.1. A header that says "Political Spectrum"

15.1.2.1.1. A label that states "General: "

15.1.2.1.2. A label that displays the user's general political "title"

15.1.2.1.3. A label that states "Social"

15.1.2.1.4. A label that displays the user's political score on social policies

15.1.2.1.5. A label that states "Economic"

15.1.2.1.6. A label that displays the user's political score on economic policies

15.1.2.1.7.

15.1.2.2. A header that says "User Details"

15.1.2.2.1. A label identifying the user's username

15.1.2.2.2. A label identifying the user's email address

15.1.2.2.3. A label identifying

15.1.2.3. A header that says "Personal Information:"

15.1.2.3.1. A label identifying the user's name

15.1.2.3.2. A label identifying the user's location

15.1.2.3.3. A label identifying the user's congressional district

15.1.2.3.4. A label identifying the user's birthday

15.1.2.4. A header that says "Password Change"

15.1.2.4.1. A label to identify the current password input

15.1.2.4.2. An input box for the current password

15.1.2.4.2.1. There will be a placeholder "Current Password"

15.1.2.4.3. A label to identify the new password input box

15.1.2.4.4. An input box for the new password

15.1.2.4.4.1. Passwords must be at least 8 characters long

15.1.2.4.4.2. Passwords must contain a special character (!, @, #, $, %, ^, &, *)

15.1.2.4.4.3. Passwords must contain one capital letter

15.1.2.4.5. A label to identify the confirm password input box

15.1.2.4.6. An input box to confirm the new password

15.1.2.4.6.1. Passwords must be at least 8 characters long

15.1.2.4.6.2. Passwords must contain a special character (!, @, #, $, %, ^, &, *)

15.1.2.4.6.3. Passwords must contain one capital letter

15.1.2.4.6.4. Password must match the first password that was input

15.1.2.4.7. A submit button labeled "Submit"

15.1.2.4.7.1. On click:

15.1.2.4.7.1.1. If current password is correct, and both new passwords match

15.1.2.4.7.1.1.1. Change user's password and reload profile page

15.1.2.4.7.1.2. On invalid entries

15.1.2.4.7.1.2.1. Display message will appear below submit button

15.1.2.5. A header that says "Actions"

15.1.2.5.1. A button that says “Delete Profile”
15.1.2.5.2. Upon clicking brings up a modal
15.1.2.5.2.1. Modal label reads “Delete Account?”
15.1.2.5.2.2. An OK button in bottom right
15.1.2.5.2.2.1. Removes user’s account
15.1.2.5.2.3. A Cancel button on left of OK button
15.1.2.5.2.3.1. Closes the modal

16. (WP) About Page

16.1. Linked to from the website navigation panel
16.1.1. A header that says “About Us”
16.1.2. A paragraph that explains the website
16.1.3. A paragraph that talks about Adam
16.1.4. A paragraph that talks about Austin

17. (DB) Database - User Account

17.1. Stores user information
17.1.1. Fields
17.1.1.1. User ID (Primary Key)
17.1.1.2. Username (Unique)
17.1.1.3. Password
17.1.1.3.1. Salting the password before passing to hash
17.1.1.3.2. SHA-256 Hash for password security
17.1.1.3.3. Store hashed password.
17.1.1.4. Email
17.1.1.5. Birthday
17.1.1.6. State
17.1.1.7. Congressional District
17.1.1.8. xAxis score
17.1.1.9. yAxis score
17.1.1.10. General alignment
17.1.1.11. Last Successful Login Timestamp

18. (DB) Database - Candidates

18.1. Contains information about each political candidate
18.1.1. Fields
18.1.1.1. Candidate ID (Primary Key)
18.1.1.2. Name
18.1.1.3. Profile Picture URL
18.1.1.4. Brief Description
18.1.1.5. Party Affiliation
18.1.1.6. Overall political Alignment (to match with user results)
18.1.1.7. District

18.1.1.8. State

19. (DB) Database - Bills

19.1. Contains information about various bills

19.1.1. Fields

19.1.1.1. Bill ID (Primary Key)

19.1.1.2. Bill Name

19.1.1.3. Brief Description

19.1.1.4. Full Bill Text

19.1.1.5. Date Introduced

19.1.1.6. Status (Passed, Vetoed, KIC (Killed in Committee)

19.1.1.7. Generated Summary

20. (DB) Database - Candidate Bill Actions

20.1. Links candidates to their actions related to specific bills

20.1.1. Fields

20.1.1.1. Action ID (Primary Key)

20.1.1.2. Candidate ID (Foreign Key Linking to Candidates Database)

20.1.1.3. Bill ID (Foreign Key linking to bills database)

20.1.1.4. Action Type (Sponsored, Co-Sponsored, Voted Yes, Voted No)

21. (BE) Bill Summarization Algorithm

21.1. Takes the original bill text

21.1.1. Runs it through an LLM and shortens it to its main points

21.1.2. Runs it through again and translates the legal speak into plain English

21.1.3. Stores summarized bill into Bills Database - Generated Summary

Design Description

Frontend Design

PolitiPulse's frontend user interface is designed to present a wealth of information in a concise and clear format, ensuring users can quickly and easily absorb key details at a glance. This layout allows for the efficient display of various data types, effectively breaking down complex information into manageable, easily digestible pieces, enhancing the user experience and facilitating seamless navigation and comprehension. The navigation bar stands as the backbone of the website's frontend, consistently and prominently displayed on the header of each webpage to ensure effortless accessibility and navigation. This design choice allows users to smoothly traverse through various website sections, eliminating the hassle of manually navigating through intricate linking trees and enhancing the overall browsing experience.

Homepage

The homepage serves as the initial landing page to PolitiPulse, being the first page users encounter and having the role of retaining their interest. It prominently features an updateable slideshow spotlighting key upcoming votes in both the House of Representatives and the Senate. Additionally, the navigation bar gains even more significance on this page, offering the sole means of navigating elsewhere on the site unless a user opts to delve deeper into the details of a bill presented in the slideshow.

Navigation Bar

From the navigation bar you are able to quickly traverse the multiple webpages that make up PolitiPulse. The first button on the navigation bar is the “Home” button which takes you back to the website's landing page. The second button is a drop down menu for the House of Representatives which has links to “Currently Elected House of Representatives”, “Election House of Representatives”, “Upcoming Business House of Representatives”, and “Recent Business House of Representatives”. Due to it being a dropdown for the house of representatives the user will not see the “House of Representatives” part of the title as that is only defined in the code. The next button is a similar dropdown but for the Senate which has links to “Current;y Elected Senate”, “Election Senate”, “Upcoming Business Senate”, and “Recent Business Senate”. The next option is “User” which is another dropdown menu where the user can navigate to their profile page or take the political compass quiz. The next two options are simple buttons “About”, and “Contact”. The user button takes the user to their profile page or login page if they are not logged in. About is a brief summary of the creators and upcoming goals of the project. Contact is how to reach the team for any questions or suggestions. On the navigation bar if the user is not logged in there are two buttons for “Register” and “Login”.

House Of Representatives and Senate Design

Emphasizing the principle of code reusability, the series of pages within PolitiPulse are primarily constructed to optimize the utilization of CSS and formatting functions. Given that both dropdown menus for the House of Representatives and the Senate address identical topics—Elected, Election, Upcoming Business, and Recent Business—the pages are differentiated solely by the distinct information they display. This efficient design approach ensures that subpages uniformly inherit the CSS sheet and format from their parent, guaranteeing visual and functional consistency across the platform while populating data from the database for each specific section. This streamlined structure not only enhances the user experience but also contributes to efficient and maintainable code management for the website.

User Profile/Political Compass

The user profile page allows the user to update any of their personal information such as age, place of residency, voting district, political compass values, username, and password along with other customization options. From the political compass button the user is transported to PoliticalCompass.org where they are to take the test and take the coordinate values of their political compass and feed them into their profile. This decision was made due to lack of political theory education among the developers of the website and instead of reinventing the wheel it is more accurate and time efficient to direct our users to a non-profit organization that specializes in the political compass quiz.

Login/Register and About/Contact

The Login/Register pages are form entry pages that allow the user to customize their experience with the website. If the user already has an account they can sign in using the login form or make an account with the register page. Both of these forms will be talked about in depth within the backend design. The About/Contact pages are simple information display pages that tell the users a little about the creators and how to contact us with any concerns, questions, or suggestions.

Politician Profiles

Politician Profiles can be accessed among the various Senate and House of Representatives pages and display a picture of the politician, an about section, important bills in the past election cycle, and all bills they have done work on. If the politician is up for election the user can easily see who they are running against and can quickly navigate to their competitions pages. From the user profile the user can see who are their currently elected officials based on their voting district.

Bills

Bills are displayed all over the website but are most prominently found within the upcoming and recent business pages along with the politician profiles. They are displayed like a big button that once clicked pops up a modal that gives the option to view the full or summary text of the bill. Once the user decides which to view a new tab opens displaying the desired text.

Backend Design

Database Structure

The foundation of PolitiPulse's backend design lies in the creation of a dynamic NoSQL database hosted on Cloud Firestore, a versatile database service offered by Firebase. Unlike the rigidity of traditional SQL databases, Firestore's appeal stems from its adaptable nature and rapid development capabilities. It allows for the construction of collections that resemble tables in SQL databases but operates with a more fluid data structure, reminiscent of JSON key-value pairs within documents. This agility, unencumbered by a fixed schema, empowers PolitiPulse to evolve its document structures effortlessly. Firestore collections serve as containers for these semi-structured documents, facilitating an efficient storage mechanism.

For a visual representation, we employ an ER diagram, offering familiarity and accommodating potential relationships between collections. The primary collections include:

1. **User Collection**: Each entry here comprises fields such as userId (number), xAxis (number), yAxis (number), alignment (string), district (number), state (string), and birthdate (string).
2. **Candidate Collection**: Documents in this collection feature fields including candidateId (number), name (string), pictureUrl (string), description (string), party (string), alignment (string), district (string), and state (string).
3. **Candidate Bill Action Collection**: Entries are anticipated to possess fields such as actionId (number), candidateId (number), bill_id (number), and actionType (string).
4. **Bill Collection**: Documents in this collection should encompass fields like bill_id (number), name (string), description (string), fullBill (string), date (string), status (string), and sumBill (string).

These collections necessitate interconnections through relationships. It is essential to define these relationships clearly to efficiently query and retrieve data. An ER diagram will help in identifying relationships, such as one-to-one, one-to-many, or many-to-many, between entities. This clarity is useful when you need to retrieve related data. This necessitates the creation of a relationship model.

In this model:

- **Users** maintain an optional many-to-many relationship with **Candidates**. A user's absence of a political alignment could preclude matching with like-minded candidates. The Candidate-to-User relationship mirrors the User-to-Candidate association.
- **Candidates** exhibit a zero-to-many relationship with **Candidate Bill Actions**. Candidates may abstain from taking action on specific bills but can participate in multiple bill actions. A **Candidate Bill Action** strictly associates with a single Candidate, an obligatory requirement since any action must involve a Candidate.

- A **many-to-one** relationship exists between **Candidate Bill Actions** and **Bills**. Each Candidate Bill Action must align with precisely one Bill. Conversely, one Bill may relate to zero actions due to pending voting or numerous actions reflecting diverse candidates' votes.

Data Aggregation

Data Aggregation commences with Axios, a Javascript library, for retrieving data from the ProPublica Congress API. Rather than relying on a backend server that requires 24/7 up-time, PolitiPulse will be taking advantage of Google Cloud Functions, which is a “serverless framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests”(“Cloud Functions for Firebase”). This will all be facilitated by importing the 'firebase-functions,' 'firebase-admin,' and 'axios' modules into a Node.js environment with Node Package manager (npm) installed.

The 'updateDatabaseFromAPI' function assumes a dual role: initially importing essential data from the external API and periodically updating it. The ProPublica Congress API, a free API for political bulk data, furnishes the necessary dataset for the PolitiPulse database. Upon data retrieval, the 'updateDatabaseFromAPI' function further undertakes the responsibility of writing it to PolitiPulse’s corresponding Firestore collection.

To implement this data aggregation process, it is necessary to import key modules, including 'firebase-functions,' 'firebase-admin,' and 'axios,' into a Node.js environment equipped with Node Package Manager (npm). Central to this process is the 'updateDatabaseFromAPI' function, which assumes a dual role. Initially, it imports essential data from the external ProPublica Congress API, ensuring that PolitiPulse has access to the requisite political dataset. Once this data is retrieved, the 'updateDatabaseFromAPI' function also takes on the responsibility of writing it to the corresponding Firestore collections, maintaining data synchronization.

The 'updateDatabaseFromAPI' function is defined as an HTTP Cloud Function, accessible via an HTTP request. Within the function, Axios performs GET requests to the ProPublica API, fetching comprehensive political data across various endpoints, including Members, Votes, Bills, Statements, Personal Explanations, Nominations, Floor Actions, and Other Responses. Subsequently, the function utilizes Firestore's functions like 'collection(),' 'doc(),' 'set,' 'commit,' and 'batch()' to update the Firestore collections with the newly acquired political data.

For the purpose of modularity and error isolation, “updateDatabaseFromAPI” will utilize several other functions when it is called. Functions such as ‘updateMembersFromAPI()’, ‘updateVotesFromAPI()’, ‘ updateBillsFromAPI()’, ‘updateStatementsFromAPI()’, ‘updatePeFromAPI()’, ‘updateNomFromAPI()’, ‘updateFaFromAPI()’, and ‘updateOtherFromAPI()’ will be used to break ‘updateDatabaseFromAPI’ into smaller components. This will also help with testing code.

PolitiPulse further enhances data synchronization by defining a 'scheduledUpdate' function as a Scheduled Cloud Function. This function is triggered at regular 24-hour intervals

using 'pubsub.schedule()'. Through the 'pubsub.onRun()' function, any required updates are efficiently executed. These pubsub functions, integral to Google Cloud Functions, play a crucial role in ensuring that PolitiPulse's political data remains up-to-date and synchronized with the ProPublica Congress API.

Data Retrieval

To interact with Firestore, PolitiPulse relies on two essential functions: ‘collection()’ and ‘doc()’. The ‘collection()’ function is our gateway to specific collections, providing access to datasets like 'User,' 'Candidate,' 'Candidate Bill Action,' and 'Bill.' For example, when retrieving user profiles, we employ the ‘collection()’ function to reference the 'User' collection. Once we have a reference to a collection, we utilize the ‘doc()’ function to pinpoint individual documents, enabling us to access or modify specific records. For instance, to access a particular user's profile, we would use doc('user_id_here'). These functions give us granular control over our data, allowing us to work with individual records or collections as needed.

Retrieving data is a core operation in PolitiPulse, as it underpins the user experience by providing valuable political information. Firestore's ‘get()’ function is instrumental in this endeavor. With ‘get()’, we can fetch data from a collection or document. For instance, to retrieve user profiles, we use ‘userCollection.get()’. This function returns a ‘querySnapshot’ that we can iterate through to process the data. The ability to fetch data from Firestore is pivotal in offering users access to up-to-date political information, whether it's candidate details, bill summaries, or historical actions taken by politicians.

Filtering data is another key aspect of data retrieval in PolitiPulse. Firestore equips us with the ‘where()’ function, enabling us to query data based on specific conditions. This functionality is invaluable when users want to filter candidates from a particular state or find bills with a specific status. For instance, we can filter candidates from 'California' by using candidatesCollection.where('state', '==', 'California'). The ‘where()’ function empowers us to tailor the user experience, providing them with precisely the information they seek.

Data manipulation, including creating, updating, and deleting data, is vital for maintaining the integrity of the database. Firestore offers functions like ‘set()’ and ‘update()’ to facilitate these operations. The ‘set()’ function is used to create or update a document. For instance, when a new user registers on PolitiPulse, their profile data is written to Firestore using ‘set()’. Similarly, ‘update()’ is employed for incremental updates to specific fields within a document, ensuring that only relevant changes are applied without overwriting the entire record. These functions are pivotal in preserving data accuracy and keeping user-profiles and political records current.

User Authentication

PolitiPulse will rely on several key functions provided by the Firebase Authentication SDK to establish a robust and user-friendly authentication system. These functions play a crucial role in ensuring the security and usability of the platform for its users.
The createUserWithEmailAndPassword(email, password) function serves as the cornerstone during the user registration process. New users can securely create accounts by furnishing their email addresses and passwords. This function not only facilitates the registration process but also underscores PolitiPulse's commitment to safeguarding user data, ensuring that sensitive information is treated with the utmost care and protection.

Returning users, on the other hand, will find solace in the signInWithEmailAndPassword(email, password) function. By inputting their registered email and password, this function grants access to their accounts in a secure manner. It stands as a gatekeeper, ensuring that only authorized individuals can log in and access their profiles and data within the platform.

The significance of user security is paramount in PolitiPulse, and the signOut(auth) function plays a pivotal role in this regard. Users can confidently log out, knowing that this function will terminate their active sessions and prevent any unauthorized access to their accounts. This empowerment ensures that users maintain control over their sessions and data.

In the realm of session management, the onAuthStateChanged(auth, callback) function takes center stage. This dynamic function monitors a user's authentication status in real-time, allowing PolitiPulse to respond appropriately. Depending on whether a user is signed in or out, the platform can seamlessly direct them to the relevant interface, be it the dashboard or login page. It epitomizes PolitiPulse's commitment to providing users with the right experience based on their authentication status.

For user account verification, the sendEmailVerification(user) function steps in post-registration. It dispatches a verification email to the user's provided address, reinforcing the authenticity of their email. This not only bolsters account security but also nurtures trust within the platform, assuring users that their information is handled with care.

When users forget their passwords or require a reset for security reasons, the sendPasswordResetEmail(email) function assumes a vital role. It empowers users to initiate the password reset process through a simple email request, ensuring a secure method for regaining access to their accounts.

Account management also includes the ability to update email addresses, and the updateEmail(user, newEmail) function makes this process seamless. Users can modify their email addresses securely while preserving the integrity of their accounts.

Similarly, in the realm of password management, the updatePassword(user, newPassword) function takes charge. Users can bolster their account security by changing their passwords efficiently and securely.

These Firebase Authentication functions will seamlessly integrate into PolitiPulse's architecture, ensuring that user registration, login, security, and account management are handled efficiently. They collectively contribute to a smooth and secure user experience while prioritizing data security and confidentiality. PolitiPulse is committed to providing users with a user-friendly and secure platform, and these functions are essential tools in achieving that goal.

Appendix

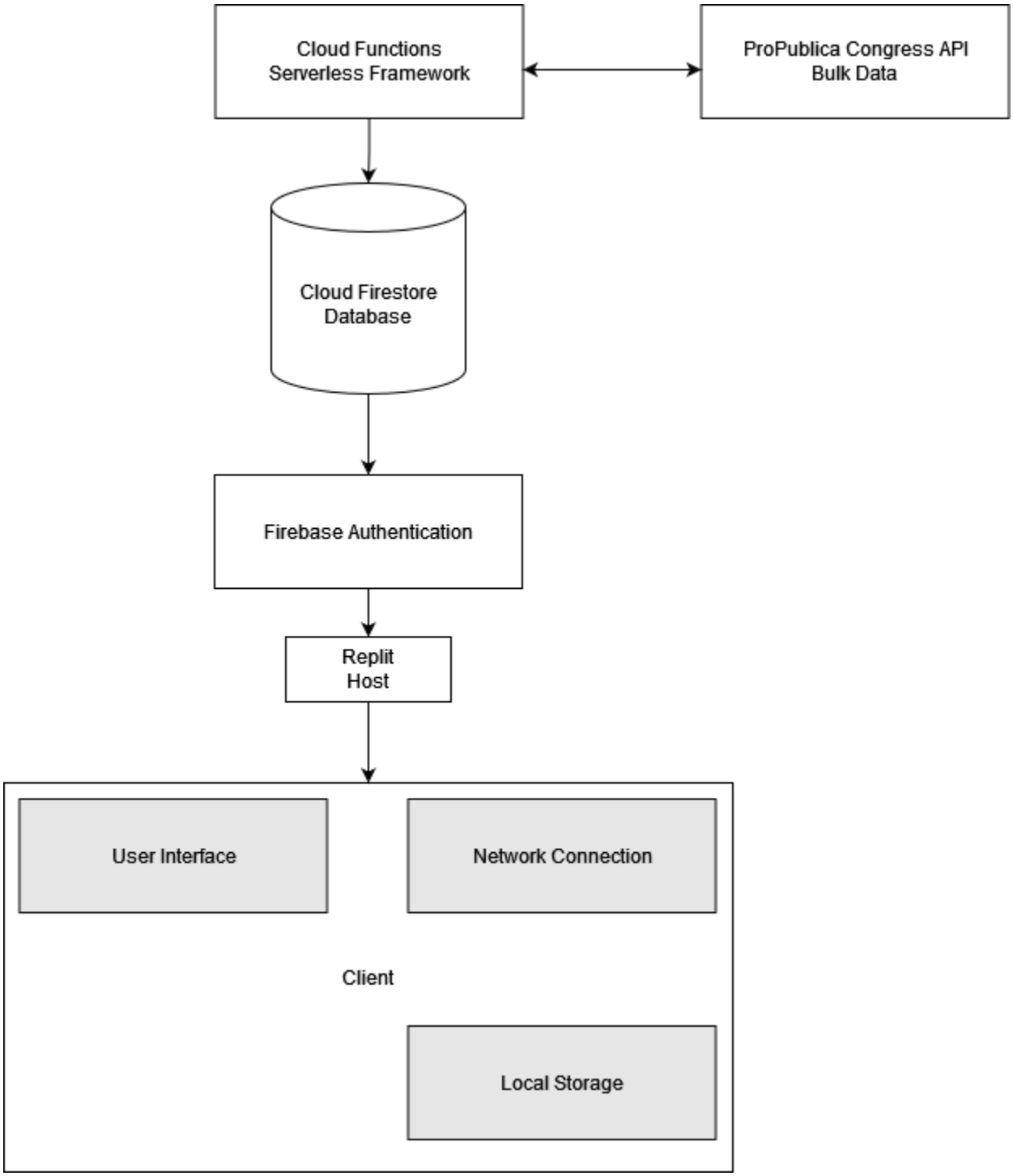

Figure 1. Block Diagram

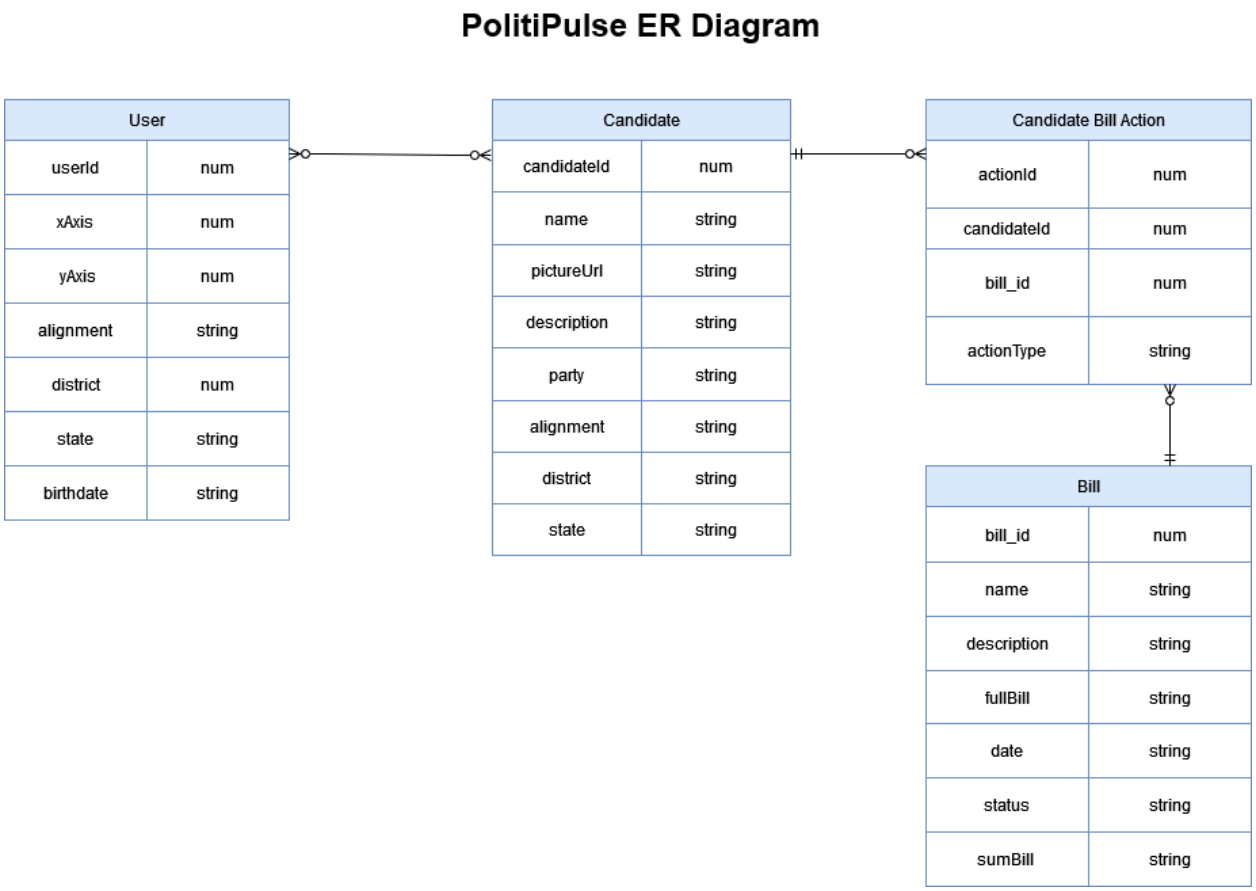

Figure 2. ER Diagram

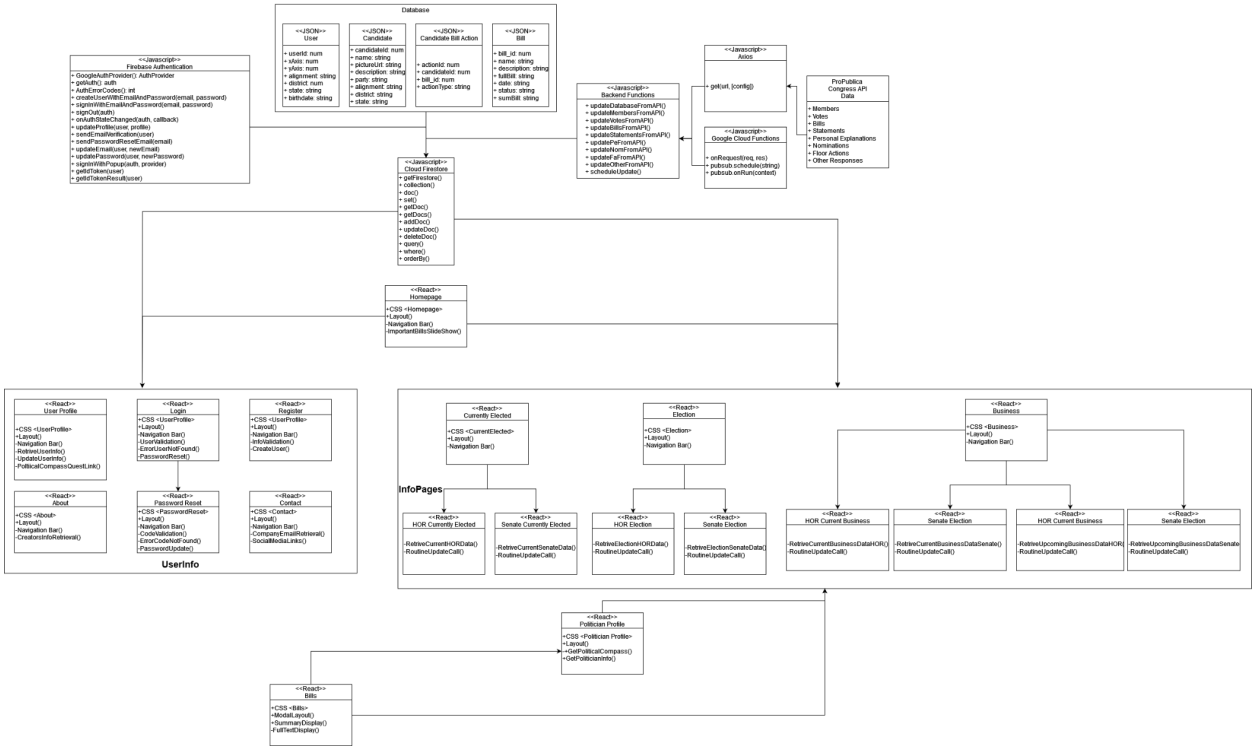

Figure 3. UML Diagram

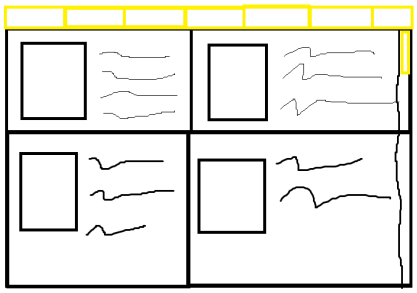

Figure 4. Current Elected UI Mockup

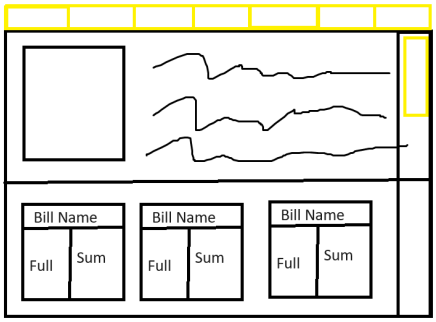

Figure 5. Politician Page UI Mockup

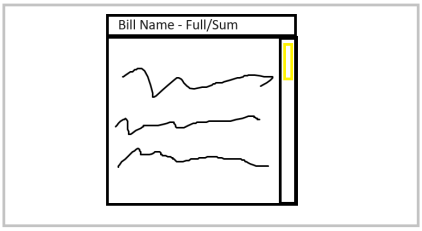

Figure 6. Bill Display UI Mockup

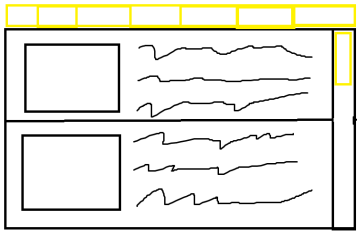

Figure 7. Up For Election UI Mockup

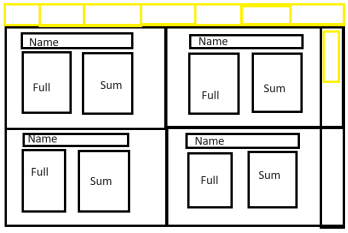

Figure 8. Upcoming/Recent Bill Mockup

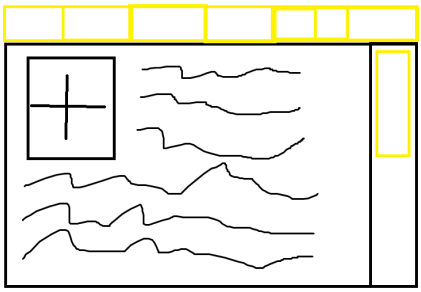

Figure 9. Profile UI Page

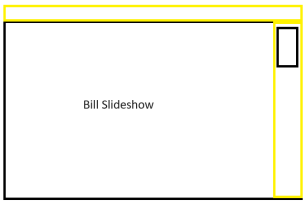

Figure 10. Homepage

Works Cited

"Cloud Functions for Firebase." *Google*, Google, firebase.google.com/docs/functions/. Accessed 30 Sept. 2023.

"Firebase Authentication." *Google*, Google, firebase.google.com/docs/auth. Accessed 30 Sept. 2023.

FirebaseExtended. "FirebaseExtended/Firebase-Video-Samples: This Repository Contains Sample Code for Some of the Videos on the Firebase YouTube Channel." *GitHub*, github.com/FirebaseExtended/firebase-video-samples. Accessed 30 Sept. 2023.

"Firestore | Firebase." *Google*, Google, firebase.google.com/docs/firestore/. Accessed 30 Sept. 2023.