PolitiPulse

By: Adam Botens and Austin Edwards

PolitiPulse - Project Update

Frontend Changes

During development so far, the project has seen several changes to the overall structure and technologies planned to be used in the project's planning phase. To start, the group's understanding of how a website's backend structure and layout was off by a large margin as the app.jsx file has been an ongoing learning process as the team learns how and when to wrap their components with the main pages. Another key change to the website's overall structure is the changes to the navigation bars functionality, along with the inclusion of the react-router-dom library. Another massive change to the website's original plan was transitioning from a CSS file to React-Bootstrap for quick implementation of intuitive UI.

To go into depth about the overall structure changes of the main app file and its current state, we need to first talk about what it imports. The majority of its imports are simply page components that are connected in the default function of the app file to the navBar import and functionality. The app file also uses the components Route and Routes import from React Router Dom and AuthProvider from the authentication context. All these imports are what provide the default function with its functionality and, with it, how the entire website functions.

Within the function AuthProvider, it wraps the entire website, which allows all children webpages to use the authentication context and check to see if the user is logged in and get the userID to pass as a key to the user information database in Firestore along with restricted access to Firebase Authentication Database. The first child component within AuthProvider is the Navbar component outside the routes container. Regardless of the webpage the user selects, the navigation bar will always be present. The other children of AuthProvider are the Route setup, which looks for a change in the URL and changes the rendered component to the target page if the URL change matches one of the Routes.

The navigation changes happen with the creation of the navigation jsx file along with overall functionality. Implementation of the original plan led to the re-rendering of the entire web page each time the user changed components. To limit the burden on the server and database, the team redesigned the functionality of the navigation to implement react router dom. The file uses Bootstrap to set up the navbars UI along with the dropdown menus UI.

The Clink function is the backbone of the navigation component as it wraps the Link component from react router dom, takes the to prop to specify the navigation path, and children to specify what's inside the link. Before it does anything, the Clink function turns the path into an absolute path and checks to make sure it's the proper path for the route. It is the use of the Link component from the react-router-dom library that ensures the page does not reload, with the Clink cleaning up the navigation default function for ease of adding and taking away webpages along with handling route errors.

The only other frontend component that has deviated from the original plan was the fact we needed to create a separate database section in Firestore to log user details along with the authentication database firebase sets websites up with. This was primarily done so we can pull the user data and

customize their experience based on the region they live in along with the if what party they are currently registered as.

Backend Changes

The backend has also seen significant changes from the original design plan. The project did stick with Firestore as a database, but there have been additions and modifications to the schema. Some of these changes have occurred due to the API endpoints' structure, and others just due to the team gaining insight and project requirements becoming more apparent. The backend will likely see changes as PolitiPulse continues to evolve. Figure 1 in the appendix depicts the current state of the database.

The team has omitted the ‘xAxis,’ ‘yAxis,’ and ‘alignment’ fields from the user collection. Initially, the project was designed to include a feature that would link users with congress members based on a political alignment quiz. However, this concept has since been set aside, rendering the fields associated with the alignment quiz unnecessary. Additionally, the birthdate field has been eliminated due to its lack of relevance to our current objectives. The user collection now primarily includes fields such as district, email, state, and user_id, which align more with the project's current direction.

Originally, the plan was to include all members of Congress in a single collection. However, the team later determined it was more practical to divide Congress into two distinct collections: one for the House and another for the Senate. This decision was influenced by the structure of the ProPublica Congress API, which provides separate endpoints for each branch. Consequently, instead of consolidating all members under a general 'Candidate' collection, there are now dedicated 'house' and 'senate' collections, aligning better with the API's structure.

Each collection shares identical fields, with the sole exception being the addition of a ‘district’ field in the ‘house’ collection. The fields currently maintained across these collections include contact, date of birth (dob), facebook_account, first_name, gender, imageUrl, last_name, party, state, title, twitter_account, website, and youtube_account. Veering from the original schema, where the name was a singular field, it has now been separated into ‘last_name’ and ‘first_name.’ The description field has been discarded. Newly incorporated fields are ‘dob,’ varying social media details, and ‘gender.’ These modifications largely stem from the availability of data at the most relevant API endpoints.

The Candidate Bill Action collection underwent some minor revisions. This collection has been renamed to 'votes,' and its fields have been accordingly renamed for clarity and consistency. Within this 'votes' collection, documents are now organized by their roll call number, and the primary fields include ‘bill_id’ and ‘question.’ Additionally, a subcollection titled ‘positions’ has been introduced. This subcollection contains documents identified by the member IDs from Congress and includes a field named ‘vote_position,’ providing a more structured and detailed representation of voting data.

The ‘bills’ collection has been updated with only a few modifications. The present structure of this collection includes the following fields: bill_date, bill_description, bill_number, bill_short_summary, bill_status, bill_summary, bill_simplified, bill_title, bill_url, introduced, latest_major_action, and latest_major_action_date. Several new fields have been added to enrich the collection: bill_url, bill_title,

bill_short_summary, introduced, latest_major_action, and latest_major_action_date. These additions aim to provide a more comprehensive and detailed overview of each bill.

There were other minor changes to the backend. Initially, we believed we might need to use Cloud Functions to use scheduled updates between the database and the external APIs. We have since decided to keep the updates manual. With so many changes still happening to the project's requirements, filling out the database and preparing continuous autonomous updates is not reasonable. This change can be seen in Figure 2 of the appendix. The last changes worth mentioning are the addition of the Congress.gov API, which is being used for the Congress member image URLs, and the addition of the OpenAI API, which is being used for transforming bills.

Progress

Despite all the changes made from the original design, significant progress has been made since the inception of PolitiPulse. Not only is there a working version of each component of this project, but we have also achieved successful communication between the frontend, backend, and authentication platform. This section of the paper will further elaborate on some of the team's success.

The team has successfully extracted the necessary data from the external API and used it to build a Firestore database. The bills, house, senate, users, and votes collections are all working, and PolitiPulse is displaying the data on the website. Most of the data was pulled from ProPublica API. As mentioned, we used the Congress.gov API to pull the images for Congress members. The 'users' collection information is collected during user registration and can be modified from the user's profile page.

The team has additionally set up registration and login functionality for users who come onto the site along with creating a context to be used to retrieve user data from anywhere on the site to easily customize web pages to suit the user based on their location within the United States. As mentioned above, this primarily shows the user their representatives and senators first so they don't have to search through a massive list to find them.

These achievements would have little impact without being able to successfully query the database or tailor user experiences from the website, which PolitiPulse has managed to achieve. Once logged in, a user's home page will display the congress members for the user's respective state or district. At this point, user authentication has been confirmed, and their information is used to submit queries to the Firestore database in search of data related to the user. This single feature is an example of the functionality and integration between all the components involved in the design of PolitiPulse. Full functionality of all features is only available for a portion of the website.

Finally, the team has accomplished seamless navigation for those without fast internet and to limit calls to the database on webpage reload. This feature, while not one of the requirements in the project document, is crucial to maintaining a low cost with Firebase due to being charged for reads and writes to the database.

Unresolved Issues

Much progress has been made, but much work still needs to be done. The bill transforming component must still be fine-tuned and modified to process larger input. The inability to process bills larger than 4096 tokens would render the bill-transforming component unserviceable. In the worst-case scenario, the team can remove the feature from the website. This would have a minor impact on the project's success.

An API for recent political news has been identified, and adding this functionality to the home page is likely one of the next features to be worked on. There are also plans to add a statements collection, which can be utilized to display the recent statements made by Congress members. These features are necessary for the website to offer much, and so these features are seen as a priority along with finishing user functionality.

When it comes to the user, the user profile needs basic functionality, the ability to delete the user account and change personal data. Additionally, we seek to update the user registration that takes the user's zip code and ties it to a congressional district, as many users will not know their congressional district off the top of their heads. These features are seen as having a major impact if not implemented.

As noted in the "Backend Changes" section, the decision was made to exclude the Political Compass Quiz from the project's scope. This component was initially envisioned in the original concept. However, the team, consisting of two Computer Science students, faced challenges in sourcing a reliable method for grading the political questions. Recognizing this, the feature was discontinued early in the development process. Importantly, this decision is not expected to detract from the project's overall success, as the team continues to focus on other core functionalities and features that align more closely with their expertise and the project's objectives.

Conclusion

In conclusion, since its inception, the PolitiPulse project has undergone significant and necessary evolutions in both frontend and backend development. The frontend has seen substantial changes, notably in its structure and the adoption of React-Bootstrap and React Router Dom for improved UI and navigation functionality. These adaptations have enhanced user experience and streamlined the site's efficiency and responsiveness.

On the backend, strategic modifications to the Firestore database and the integration of additional APIs have been key to the project's progress. The decision to divide Congress into separate collections for the House and Senate and the refinement of data fields to align with the most relevant API endpoints have streamlined data management and retrieval.

Key achievements include the successful extraction and integration of data from external APIs into Firestore, effective user authentication and personalized user experiences, and the implementation of efficient navigation and database querying mechanisms. However, challenges remain, such as refining the bill-transforming component and enhancing user profile functionalities.

Appendix

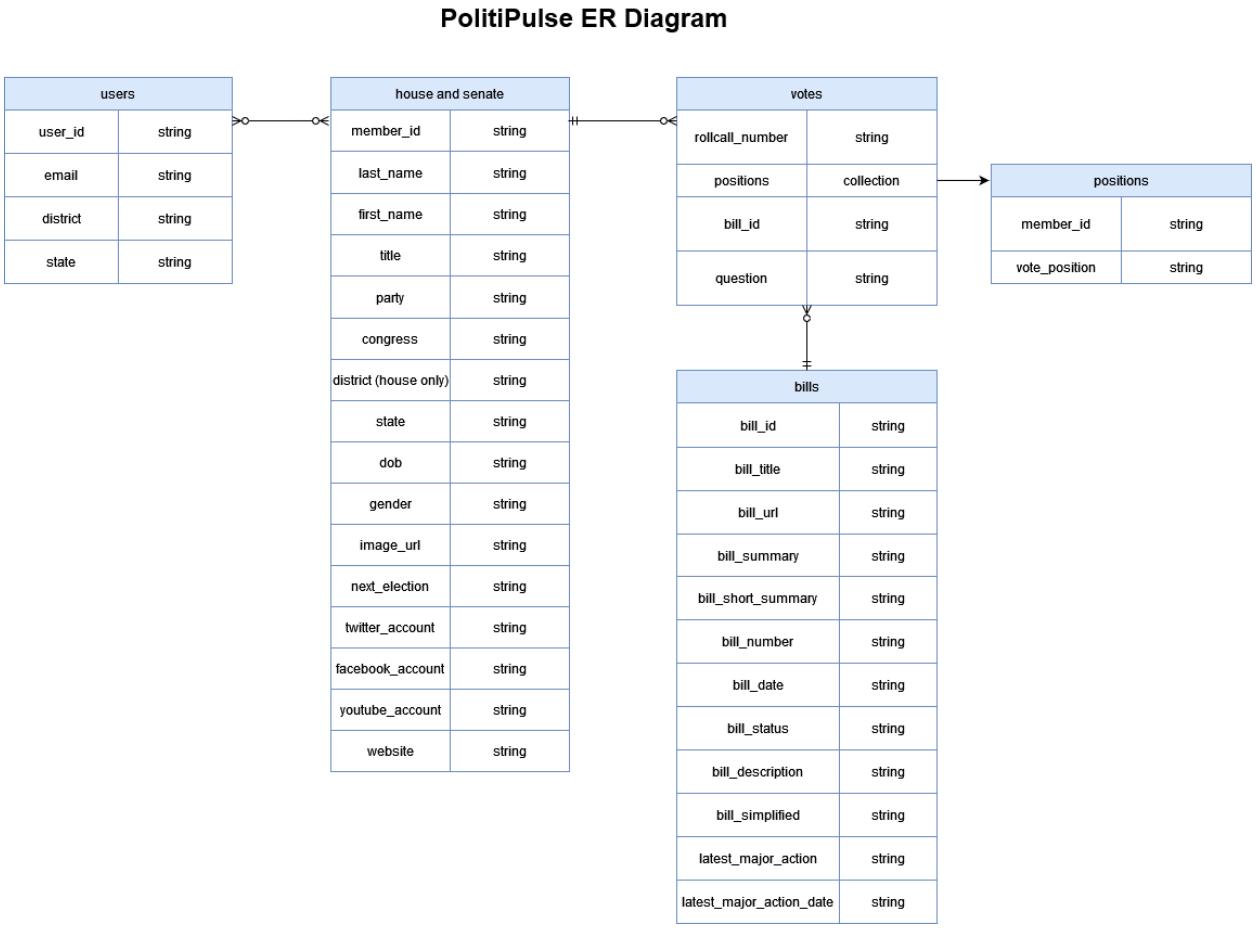

Figure 1: An updated schema of the PolitiPulse database.

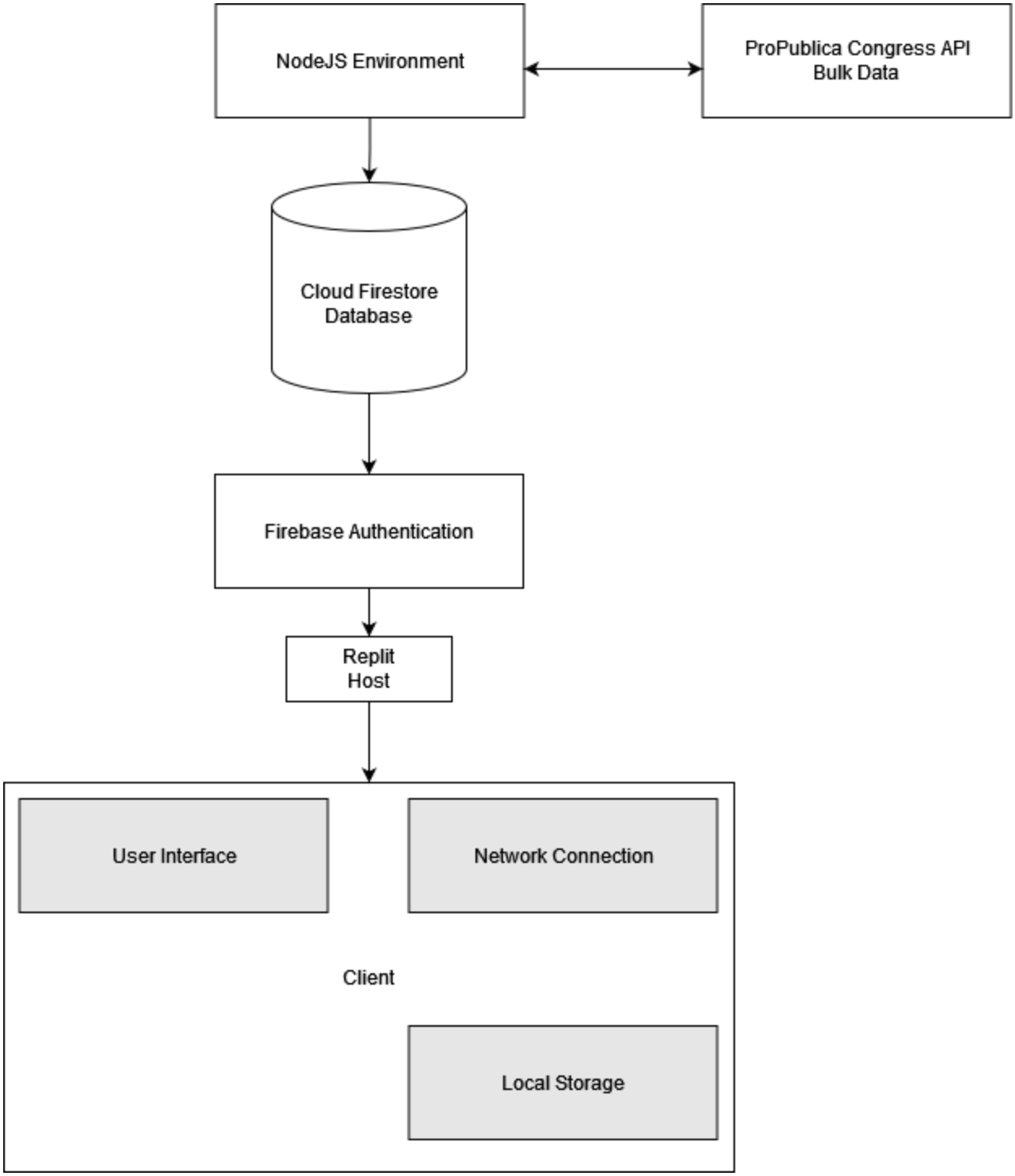

Figure 2: Updated block diagram.

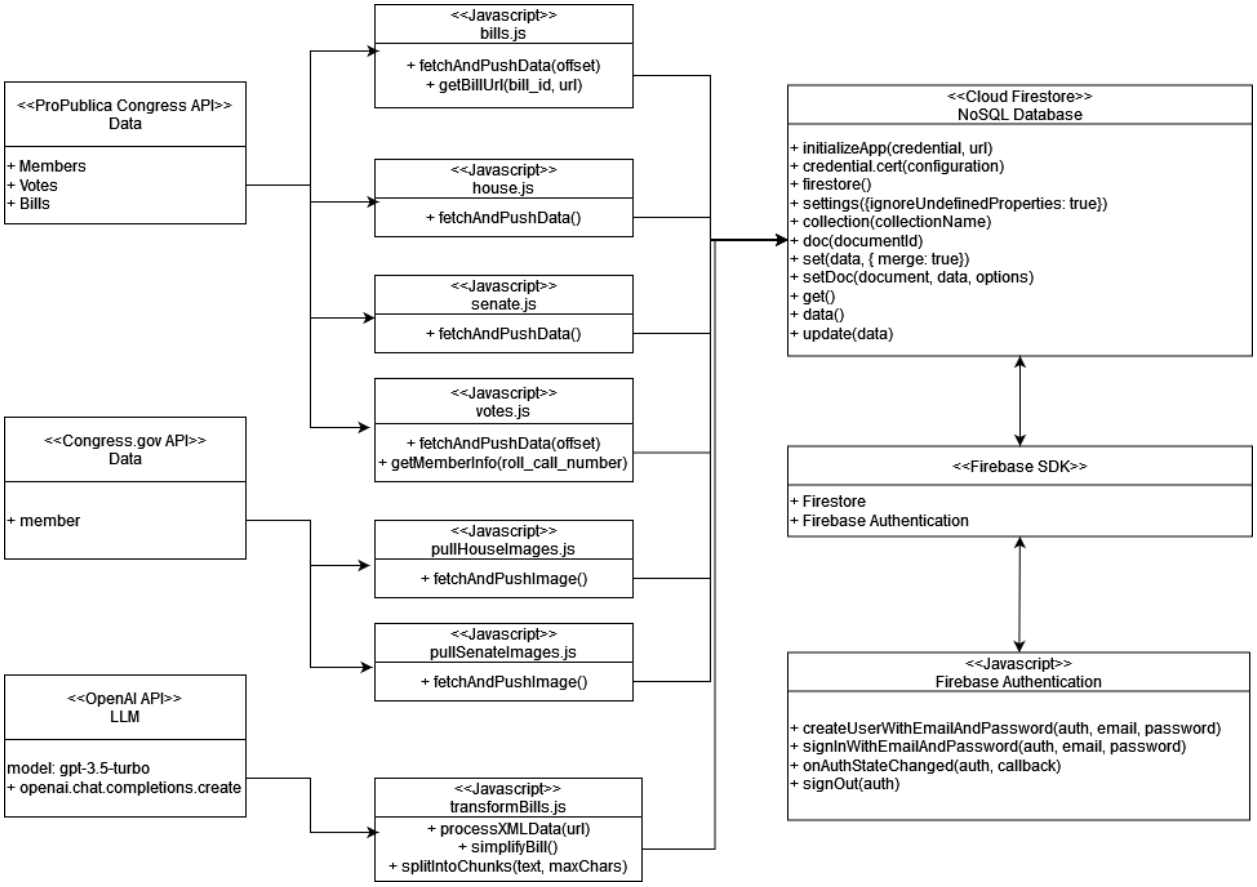

Figure 3: Updated UML diagram of the backend.

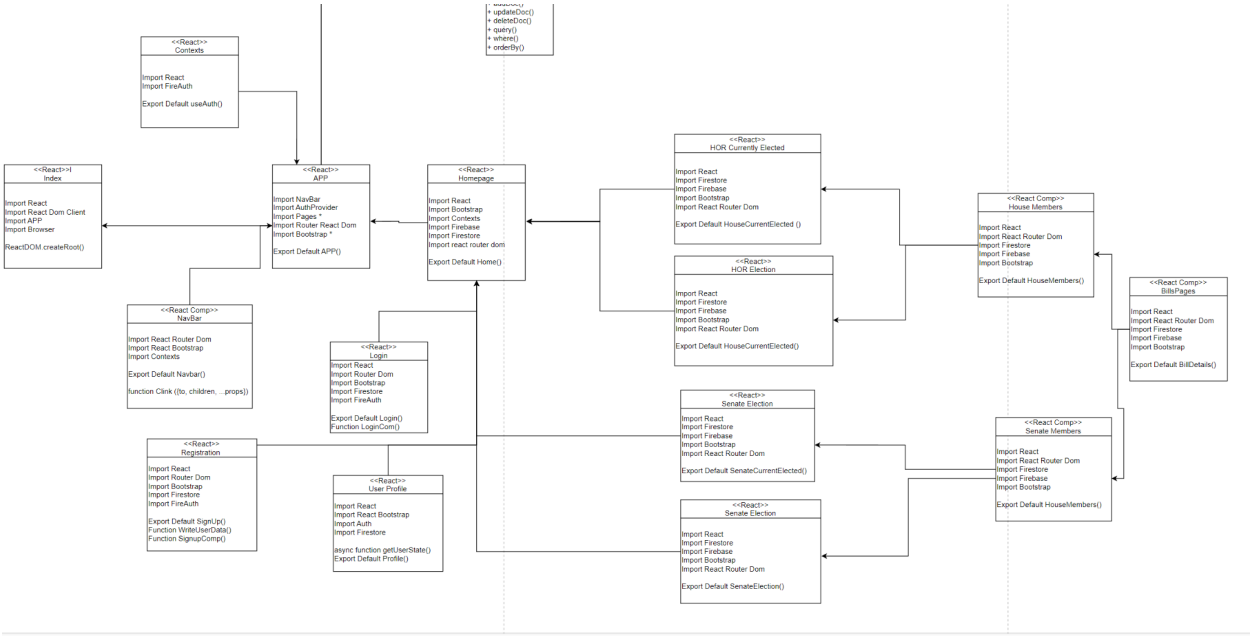

Figure 4: Updated UML diagram of the frontend.