Sudo rm -rf /
Politifind

Fall 2017

Overview: **NOTE**: With the implementation of our team choice searching functionality, our server depends on the `algoliasearch_django` module that can be installed by running `$ pip install algoliasearch-django`

Politifind aims to make data about all the happening in Congress accessible and intuitive to the average user. We have designed a web app to nicely compile all such data into pages that can easily be navigated too and connected with each other. Most government websites have this data in difficult to read formats, or in formats that are not intuitive/connected. Politifind aims eliminates this problem, and allows people to stay informed about all that's going on.

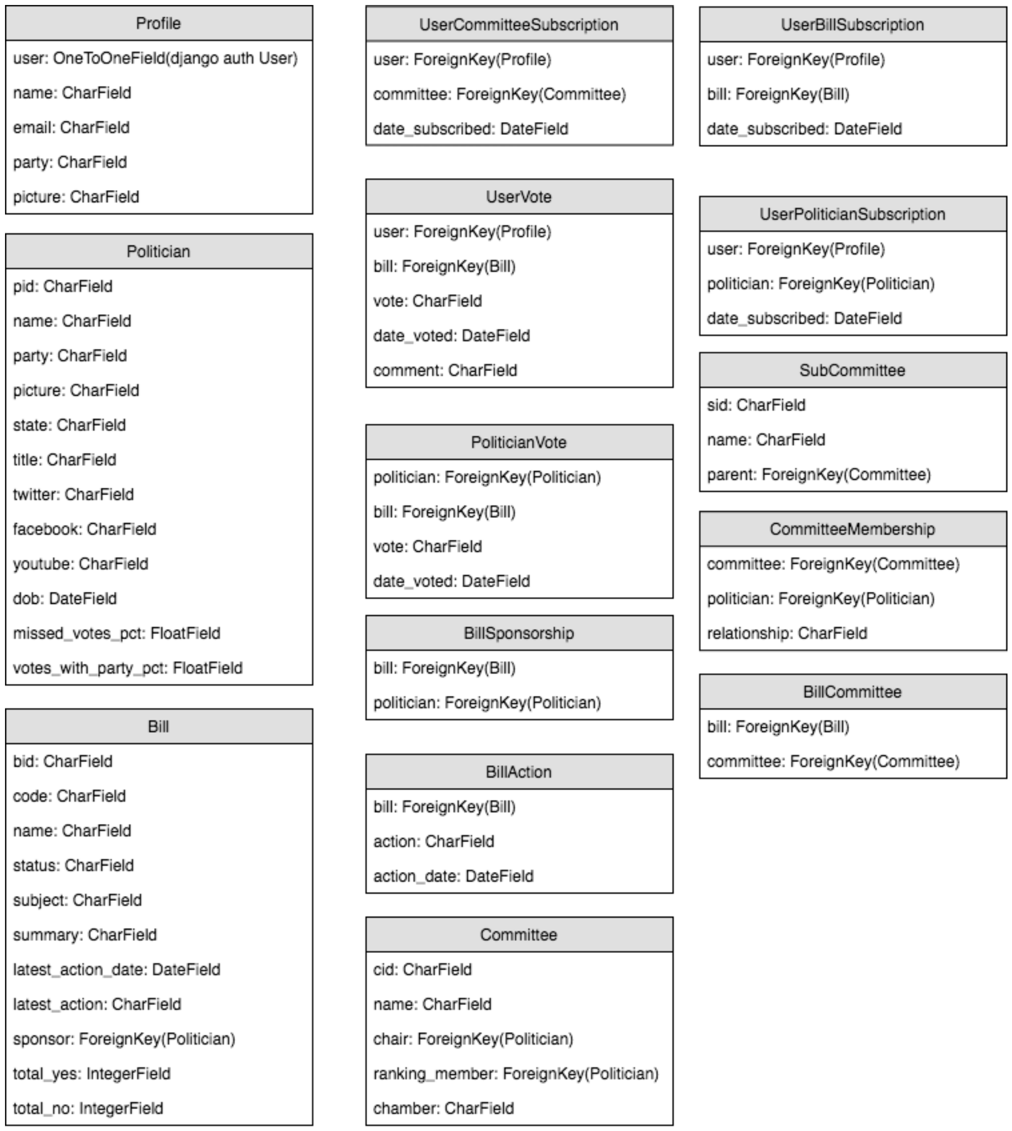
Members: Andrew Bass, Matthew Bissaillon, Matthew Gramigna, Justin Kennedy

Github: `https://github.com/arbass22/politifind`

User Interface:

Data Model: The following diagram is our updated data model for Politifind with the following descriptions:

Profile: Includes authentication fields for the django User as well as information about this user's politifind profile.
Politician: Represents a member of congress in either the house or senate.
Bill: A congressional bill.
PoliticianVote: Indicates how a specific Politician voted on a specific bill.
UserVote: Indicates how a specific politifind user voted on a specific bill.
Committee: Represents a committee in congress.
SubCommittee: Represents a subcommittee of an existing politifind Committee object.
CommitteeMembership: Indicates that a specific Politician is a member of a specific Committee.
BillCommittee: Indicates that a specific Committee introduced a specific Bill.
BillSponsorship: Indicates which Politician sponsored a specific Bill.
BillAction: An action that has happened on a specific Bill.
UserPoliticianSubscription: Indicates that a politifind user subscribed to a specific Politician.
UserBillSubscription: Indicates that a politifind user subscribed to a specific Bill.
UserCommitteeSubscription: Indicates that a politifind user subscribed to a specific Committee.



URL Routes/Mappings:

route	name	description
r‘ˆ$’	index	route users to the home page
r‘ˆbill/(.*)/$’	bill	individual bill page
r‘ˆpolitician/(.*)/(bills|votes)*’	politician	individual politician page
r‘ˆpoliticians/$’	politicians	list of all politicians
r‘ˆbills/$’	bills	list of all bills
r‘ˆcommittee/(.*)/(bills|subcomittees)*$’	committee	individual committee page
r‘ˆcommittees/$’	committees	list of all committees
r‘ˆprofile/$’	profile	user’s profile page
r‘ˆsearch/$’	search	display’s search results
r‘ˆaccounts/’	accounts	adds login/logout/etc urls
r‘ˆvote/(.*)/(yay|nay)*/’	vote	view individual vote page
r‘ˆsubscribe/$’	subscribe	user can subscribe
r‘ˆunsubscribe/$’	unsubscribe	user can unsubscribe
r‘ˆprofile/update/’	update profile	have a user update their profile page

Authentication/Authorization: We added a one-to-one mapping between Django’s User model and our custom Profile model, allowing a user to specify additional information, such as political party, profile picture, etc.. Our users can browse Politifind without logging in, but when they do they have some additional functionality, such as subscribing to various politicians, bills, and committees, or voting on bills to see how they stack up against all politfind users and/or congress. We also enable to profile view to a logged in user where they can view all of their subscriptions and change their profile info. Lastly, the homepage is templated differently based on whether or not a user is logged in. We have one permissions group for a user that allows them to modify their info, or add votes and subscriptions, but it does not allow them to for example change the data of a bill, as it shouldn't.

Team Choice: For our team choice, we decided to integrate the Algolia Search API into our app to allow the user to very quickly search through all the data in politifind. To do this, we added two features. The first is an autocomplete search that reacts to what the user types in the search bar and displays results in a dropdown. this searching can happen from any of our views. The other feature is an actual search page. If a user does not want to look at the dropdown results, they can hit enter and it will bring them to a search page displaying the results from their query. We used the Algolia Javascript client for the autocompleting, and the algoliasearch-django pip module mentioned before for the search page.

Conclusion: