SHARE FE-DB Reqs

Friday, August 20, 2021 6:49 PM

info and general requirements/instructions for the 'votersrevenge-frontend-db'

INTRO

the results of this project will ultimately be used in another open source project called "votersrevenge", @ github.com/sldev2/votersrevenge. See votersrevenge.info for more information on the Voter's Revenge project.

A *posse* is groups of citizens; in the old US Wild West, posses used to hunt down criminals; the votersrevenge app, for which this project is being developed, has the purpose of empowering citizens to "fire" elected politicians who are corrupt, ineffective, etc. In the US, it is in general not possible for citizens to remove elected officials (though it should be...). Consequently, by "firing" a politician, I mean ensuring that the politician doesn't get re-elected. (In the case of the wrangler role, the idea is to exert pressure on elected officials by shaming the them, while educating fellow citizens about their misdeeds. This will indirectly damage their re-election prospects.)

A *local posse* is a subset of a posse. There will typically be many local posses for each posse (eventually). If a user of the system joins a local posse, directly, he/she is automatically made a member of the parent posse. However, joining a posse doesn't require joining a local posse. Local posse members will typically live near each other, while posse members, in general, will only live in the same state (potentially far from each other)

Every posse and local_posse is directed against 1 politician, the *target*, and has 1 specific *issue* they are focused on.

It is hoped that citizens across the world will find votersrevenge useful, including when dealing with parliamentary democracies, which the US is not.

INFORMATION

user roles	registration
super_sheriff	registered
sheriff	registered
deputy	registered
voteslinger	registered
wrangler	registered
follower	registered
visitor	not registered

- registered user can have different roles in different posses and any of their local posses, though a super_sheriff is a super_sheriff in all posses, and all local posses

REQUIREMENTS

- ☐ instructions on how to install and run whatever bits are delivered (feel free to ask if I already know how do something)
- ☐ the front end must access the database only through a GraphQL api, generated by the Hasura GraphQL engine, with MS SQL Server in the back end
  - NOTE: I am running SSMS 17.1, so please try to deliver a database compatible with that; if necessary, I can upgrade, on my end

MILESTONE #1

- ☐ create a schema for the database (defined in MILESTONE #2, below), with a database diagram
  // by "schema", I just mean tables and fields definitions; NOT any stored procs or triggers

MILESTONE #2

- ☐ create the database, with the indicated lookup tables preferably already populated

- Tables

//THIS IS A STARTING GUIDE you may need to add, say, some cross tables; plus fields, indexes, I think a stored proc or trigge r or two
// also, you may need to restructure the database, somewhat; you will generally add data in the next milestone
// though the lookup tables could be populated, in this milestone:
// generally, I think every table X will have an id field: X_id

- user
  - user_id
  - username : string (max length 20)
  - email // nullable, system email
  - is_validated :bool// got validation of the email during the registration process
  - public_email // nullable, visible to everybody
  - statement : string// nullable, 1200 char max

- user_location
  - user_id // user : user_location is 1 to 0 , since members don't have to provide their location unless they join a local posse
  - state : string // nullable, 5 chars max

// if a geolocation type object is sent over the wire, have db stored proc or trigger determine lat, lng, and write those
// also, if lat, lng is sent over the wire, have db stored proc determine the geolocation value, and write it, also
// my guess is that only lat, lng should be sent over the wire; but geolocation data type is useful for doing queries like
// "find all users within X miles of user Y", where user Y's location is already in the database

  - geolocation : geography // geography is a SQL Server data type
    // nullable (follower type users will not have to provide a location)
  - lat : float // null
  - lng : float // null
  - zipcode: string(10 char max) // nullable

- government_level // **a lookup table**
  - government_level_id
  - name: string (max 15 chars)
  - abbr2 : string*2
  - abbr3 : string*3

values in this table:

1	National	NT	NAT
2	State	ST	STA
3	County/Town	CT	CTY

- government_branch // **a lookup table**
  - government_branch_id
  - name: string (max 15 chars)
  - abbr2 : string*2
  - abbr3 : string*3

values in this table:

1	House	HR	HOR
2	Senate	SN	SEN
3	Executive	EX	EXE
4	Other	OT	OTH

- issue
  - issue_id
  - name : string (max 50 chars)
  - short_desc : string (max 320 chars)
  - long_desc: string (max 1600 chars)

values in this table

1	Election Steal 2020	Widespread fraud in the 2020 Federal Elections occurred, for both the Presidency and Congress, with over 3,000 sworn affidavits to this effect.	Election Steal 2020 lorem ipsum
2	Medical Tyranny	The regulatory bureaucracy in the US (including CDC and FDA) is non-transparent and out-of-control. Additionally, state governors and school boards are abusing their powers, all in the name of public health. Frontline medical workers who choose not to get vaccinated are being terminated.	Medical Tyranny lorem ipsum
3	Medicare for All	Americans vastly overpay for their healthcare, and a "Medicare for All" single payer model will greatly rectify this price gouging.	Medicare for All lorem ipsum

- state // lookup table
  - state_id

- name string: (max 12 chars) non-null
- abbr2 string*2 non-null

- values in this table; **one row for each of 50 states**

1	Alabama	AL	
2	Alaska	AK	
3	Arizona	AZ	
ETC.			
51	United States	US	// this is to handle the **special case** of the US President, who is associated with all states, not just one

- target // these are politicians, who typically are holding office
  - target_id
  - first_name :string (max 20 chars) // non-null
  - last_name :string (max 28 chars) // non-null
  - name : string // this is calculated ; the formula is first_name + ' ' + last_name
  - current_position : string (max 40 chars) // non-null
  - state_id // FK
  - government_level_id // FK
  - current_term_end_date // nullable

- posse
  - posse_id
  - name : string // this is calculated ; the formula is simply target.first_name + ' ' + target.last_name + ' - ' + issue.name
  - issue_id // FK
  - target_id // FK
  - government_level_id : integer // 1 is National, 2 is State, 3 is County/Town
  - government_branch_id // 1 is House, 2 is Senate, 3 is Executive, 4 is Other
- posse_members // posse : posse_members is 1 to Many
  //not 100% sure about 1 to Many … since the same user can be in multiple posses
  - posse_id // FK
  - user_id // FK
- local_posse
  - posse_id // posse : local_posse is 1 to Many
  - local_posse_id
  - nickname: string(25 chars max) ; non-null; at least 7 chars
  - name : string // this is calculated ; the formula is (parent) posse.name + ' - ' + nickname
  - zipcode : string *5 // I'm aware of the fact that users's zipcodes have a different definition
  - quick_find_code : string*9 // strangers can search by zip-###, where zip Is 5 chars
    - e.g., is 2 local posses are located in zip code 07052, their quick_find_code's could be: 07052 -001 and 07052-002

- local_posse_member // local_posse : local_posse_member is 1 to Many
  //not 100% sure about 1 to Many … since the same user can be in multiple local_posses
  - posse_id // FK
  - local_posse_id // FK
  - user_id // FK
- posse_roles // **lookup table**
  - posse_role_id
  - name : non-null; string (max 18)
  - is_registration_required : boolean

values in this table

1	super_sheriff	true
2	sheriff	true
3	deputy	true
4	voteslinger	true
5	wrangler	true
6	follower	true
7	visitor	false

- default_redline_deadline // these will only be defined by sheriff and deputies
  - // if pledge_type = 1 (i.e., voteslinger), then voteslinger_deadline_date and voteslinger_deadline must be non-null

- default_redline_deadline_id
- target_id // FK
- issue_id // FK
- posse_id // FK
- redline_date // non-null
- redline : string (max 450 chars) // non-null
- pledge_type : integer // 1 for Voteslinger 2 for Wrangler ONLY (**don't** allow: 3 for CreateShindig or 4 for AttendShindig)
- voteslinger_deadline_date // nullable
- voteslinger_deadline : string(max 450 chars) // nullable
- is_general_election : boolean // default = false
  - // only meaningful when pledge_type = 1, i.e., voteslinger

- default_deadline // default_redline_deadline : default_deadline is 1 to Many
  - default_deadline_id
  - default_redline_deadline_id // FK
  - deadline_date // non-null
  - deadline : string(max 450 chars) // non-null

- pledge
  // a user has a choice between pledging to accept a default redline/deadline pair, created by admins
  // or else pledging to accept their own redline/deadline pair, defined by themselves
  // the exception is voteslinger deadlines and their associated deadline_dates
  // these are set in stone by SYSTEM, but will be enforced primarily via business rules in the program
  // BUSINESS RULE
  // still, the database insertion routine should check that, for a given target_id and pledge_type==1, a
  // voteslinger pledge's deadline & deadline date must already exist in default_redline_deadline
  // BUSINESS RULE
  // and, secondly, that a user cannot have multiple voteslinger pledges with the same target_id and deadline_date
  - pledge_id
  - pledge_date // timestamp\

  - posse_id nullable
  - local_posse_id nullable
  - // BUSINESS RULE
  - // all pledges exist with respect to either a posse, or local posse; so, 1 and only 1 of
    // posse_id or local_posse_id should be null
  - user_id // user to pledge is 1 to Many
  - default_redline_deadline_id //nullable
  - default_deadline_id // nullable
  // BUSINESS RULE
  // if default_redline_deadline_id and default_deadline_id are null then:
    // personal_redline_date,
    // personal_redline,
    // personal_deadline_date, AND
    // personal_deadline
  // must all be non-null and vice versa
  - personal_redline_date // nullable
  - personal_redline : string(max 450 chars) // nullable
  - personal_deadline_date // nullable
  - personal_deadline : string(max 450 chars) // nullable
  - fullfilled // boolean, default = false
  - fulfillment_date // nullable, default = null
  - pledge_type // 1 for Voteslinger 2 for Wrangler or 3 for CreateShindig or 4 for AttendShindig
  - shindig_id // nullable; only relevant for pledge_type = 3 or pledge_type = 4

  (NOTE only users who are sheriffs or deputies in a local posse can create shindigs)

- shindig
  - shindig_id
  - pledge_id FK ; non-null
  - event_date
  - title : string (max 50 chars)
  - short_desc : string (max 320 chars)
  - long_desc: string (max 1600 chars
  - is_cancelled : boolean

MILESTONE #3

- create the graphql api, and populate database with sufficient data to test, and also create tests for each of the following items (**probably any test technology would do, but please check with me, first**)
- exception to "populate database with sufficient data to test" - just deliver with data corresponding to government_level = 1,2 (not 3); in other words, I am not looking to test data for local and county level target politicians, and posses directed at such; only federal and state level
  - determine how many local posses exist for any given posse
  - determine how many *wrangler* members there are in any given posse via 2 separate methods
    - method #1 just sum the wranglers found in the posse, ignoring local posses membership, completely
    - method #2 count wranglers in the posse who have no presence in any of the posses' local posse
      - add to this the wranglers who are in at least 1 local posse, also, but take care not to double count such wranglers who have that role in more than 1 such local posse
      - I expect this to be so computationally expensive that it will serve more as a validation check for back end maintainers and testers
  - determine how many *voteslinger* members there are in any given posse via 2 separate methods
    - method #1 just sum the voteslingers found in the posse, ignoring local posses membership, completely
    - method #2 count voteslingers in the posse who have no presence in any of the posses' local posse
      - add to this the voteslingers who are in at least 1 local posse, also, but take care not to double count such votesllingers who have that role in more than 1 such local posse
      - I expect this to be so computationally expensive that it will serve more as a validation check for back end maintainers and testers
  - determine how many *follower* members there are in any given posse via 2 separate methods
    - method #1 just sum the followers found in the posse, ignoring local posses membership, completely
    - method #2 count followers in the posse who have no presence in any of the posses' local posse
      - add to this the followers who are in at least 1 local posse, also, but take care not to double count such followers who have that role in more than 1 such local posse
      - I expect this to be so computationally expensive that it will serve more as a validation check for back end maintainers and testers
  - determine how many CreateShindig events (from TODAY, onwards, comparing with the deadline date) there are in any given posse (given posse_id); **do not consider CreateShindig events in its local posses**
  - determine how many CreateShindig events (from TODAY, onwards, comparing with the deadline date) there are in any given posse (given posse_id); **do not consider CreateShindig events in its local posses**
  - determine how many days until the next redline_date occurs, for any given user_id and posse_id
  - determine how many days until the next deadline_date occurs, for any given user_id and posse_id
  - determine how many days until the next redline_date occurs, for any given user_id and local_posse_id
  - determine how many days until the next deadline_date occurs, for any given user_id and local_posse_id
  - determine how many days until the next default *voteslinger* redline_date occurs, for any posse_id
    - use the default_redline_deadline table
  - determine how many days until the next default *general election voteslinger* deadline_date occurs, for any posse_id
    - use the default_redline_deadline table (pledge_type = 1)
  - determine how many days until the next default *non-general election voteslinger* deadline_date occurs, for any posse_id
    - use the default_redline_deadline table (pledge_type = 1)
  - determine how many days until the next default *voteslinger* redline_date occurs, for any local_posse_id
    - use the default_redline_deadline table (pledge_type = 1)
  - determine how many days until the next default *general election voteslinger* deadline_date occurs, for any local_posse_id
    - use the default_redline_deadline table (pledge_type = 1)
  - determine how many days until the next default *non-general election voteslinger* deadline_date occurs, for any local_posse_id
    - use the default_redline_deadline table (pledge_type = 1)
  - determine posses that a user is a member of, given their user_id
    - return a collection of { posse_id, posse.name }
  - determine local posses that a user is a member of, given their user_id
    - return a collection of { local_posse_id, local_posse.name, posse_id, posse.name }

  - determine how many voteslinger members there are in an given local posse (given local_posse_id)
  - determine how many wrangler members there are in any given local posse (given local_posse_id)
  - determine how many follower members there are in any given local posse (given local_posse_id)
  - determine how many CreateShindig events (from TODAY, onwards) there are in any given local posse (given local_posse_id), for which is_cancelled = false
  - determine how many CreateShindig events (from TODAY, onwards) there are in any given posse (given posse_id), for which is_cancelled = false
  - return the collection of CreateShindig events (from TODAY, onwards) there are in any given local posse (given local_posse_id), for which is_cancelled = false; return { shindig.event_date, shindig.title, shindig.short_desc, shindig.long_desc, user.user_name (of the user who created the shindig) }
  - return the collection of CreateShindig events (from TODAY, onwards) there are in any given posse (given posse_id), for which is_cancelled = false; return { shindig.event_date, shindig.title, shindig.short_desc, shindig.long_desc, user.user_name (of the user who created the shindig) }

  <u>pledge_type = 1</u>
  - for a given user_id and posse_id, and an { input_date } return the collection of voteslinger pledges with X_redline_date >= { input_date }, namely:

- { pledge_id, X_redline, X_redline_date, X_deadline, X_deadline_date, fulfilled, fullfillment_date}
  by "X_" I mean 1) include both default and non-default pledge items 2) include both personal and non-personal pledge items
- { input_date } is a parameter specified by the graphQL consumer

- for a given user_id and **local_**posse_id, and an { input_date } return the collection of voteslinger pledges with X_redline_date >= { input_date }, namely:
  - { pledge_id, X_redline, X_redline_date, X_deadline, X_deadline_date, fulfilled, fullfillment_date}
    by "X_" I mean 1) include both default and non-default pledge items 2) include both personal and non-personal pledge items

pledge_type = 2

- for a given user_id and posse_id, and an { input_date } return the collection of **wrangler** pledges with X_redline_date >= { input_date }, namely:
  - { pledge_id, X_redline, X_redline_date, X_deadline, X_deadline_date, fulfilled, fullfillment_date}
    by "X_" I mean 1) include both default and non-default pledge items 2) include both personal and non-personal pledge items
- for a given user_id and **local_**posse_id, and an { input_date } return the collection of **wrangler** pledges with X_redline_date >= { input_date }, namely:
  - { pledge_id, X_redline, X_redline_date, X_deadline, X_deadline_date, fulfilled, fullfillment_date}
    by "X_" I mean 1) include both default and non-default pledge items 2) include both personal and non-personal pledge items

pledge_type = 3

similar

pledge_type = 4

similar

- given a posse_id, return { posse.name, target.name }
- given a local_posse_id, return { local_posse.name, target.name, posse.posse_id}
- given a quick_find_code, return { local_posse.local_posse_id, local_posse.name, target.first_name + target.last_name, issue.issue_id, issue.name, issue.short_desc }

- given a posse_id, return the number of CreateShindig pledges

- create sample data, sufficient for testing
  - there is an attached data file is for **users** in posses that operate at government_level = 2 (state), viz. **data for votersrevenge-messaging.xlsx**
  - there are 2 attached data file for (a minimum) of the **targets** that I want to see used for tests, **State target's.xlsx** and **National target's.xlsx**
  - **don't add data or tests for any posses that operate at the government_level = 3 (county/town) // that would be Day 2**
  - do add data and tests for posses that operate at government_level = 1 (national)
  - do add data and tests for posses that operate at government_level = 2 (state)
  - please add test data corresponding to all 3 issues defined in the issue table, viz
    - Election Steal 2020
    - Medical Tyranny
    - Medicare for All

    feel free to re-use the same targets (i.e., politicians), and users that are already in the test database
    you can also re-use rows in the default_redline_deadline, but change the verbiage in default_redline_deadline.redline and default_redline_deadline.deadline, slightly, to match the issue_id

  - for each posse, please add 2 local posses
    - // you can use boring names for the local_posse.nickname field, such as "nickname A", "nickname B", etc.
  - please add at least 4 users from each role to each posse and at least 2 users from each posse to each of its local posses
    - // this is close to what I did in the attached data file, for government_level = 2, viz. **votersrevenge-messaging.xlsx**
  - **for wrangler type pledges,** for the default_redline_deadline table, please add data following these string pattern for the redline and deadline fields, :
    - for issue_id = 1
      - redline: " { target.name } must support full forensic audit of all state elections that occurred in 2020"
      - deadlines: // in default_deadline
        1. "pass out, or arrange to have passed out, 300 flyers to school children, from a public street next to a public or private sch ool, informing potential voters about { issue.name }, and the violation of our redline, { default_redline_deadline.redline }"

        2. "pass out, or arrange to have passed out, 300 flyers to the general public, from a sidewalk in a high foot -traffic area, informing potential voters about { issue.name }, and the violation of our redline, { default_redline_deadline.redline }"

        3. "demonstrate for 4 hours, holding signs informing potential voters about { issue.name }, and the violation of our redline, { default_redline_deadline.redline }"

        4. "demonstrate while marching for 4 hours, holding signs informing potential voters about { issue.name }, and the violation of our redline, { default_redline_deadline.redline }, and banging pots with wooden spoons"

        5. "take out a half page ad in a newspaper of a nearby university, informing potential voters about { issue.name }, and the viol ation of our redline, { default_redline_deadline.redline }"

        6. "take out a quarter page ad in a newspaper, informing potential voters about { issue.name }, and the violation of our redline , { default_redline_deadline.redline }"

7. "any deadline action OK'd by your sheriff"

- for issue_id = 2
  - redline: "{ target.name } must declare mask and vaccine mandates illegal, and support leglislation banning such mandates"
  - deadlines: // in default_deadline
    - same pattern as for issue_id = 1

- for issue_id = 3
  - redline: "{ target.name } must support Medicare for All, verbally as well as voting for legislation"
  - deadlines: // in default_deadline
    - same pattern as for issue_id = 1

- **for voteslinger type pledges,** for the default_redline_deadline table, please add data following these string pattern for the redline and voteslinger_deadline fields, :
  - for issue_id = 1
    - redline: "{ target.name } must support full forensic audit of all state elections that occurred in 2020"
    - voteslinger_deadline:
      - when is_general_election = true
        "vote against { target.name } in his/her next general election, on { default_redline_deadline.voteslinger_deadline_date }"
      - when is_general_election = false
        "vote against { target.name } in his/her next primary election, or caucus, on { default_redline_deadline.voteslinger_deadline _date }"

  - for issue_id = 2
    - redline: "{ target.name } must declare mask and vaccine mandates illegal, and support leglislation banning such mandates"
    - voteslinger_deadline:
      - same pattern as for issue_id = 1

  - for issue_id = 3
    - redline: "{ target.name } must support Medicare for All, verbally as well as voting for such legislation"
    - voteslinger_deadline:
      - same pattern as for issue_id = 1

- implement the following business rules on the server side, **if this makes sense** (will also be implemented in the GUI)
  // return an error indicating the nature of the problem, when this occurs
  a new wrangler pledges' fulfillment_date must be within 2 months of the redline_date
  a new shindig pledges' fullfillment_date must be within the next 3 months (starting TODAY)