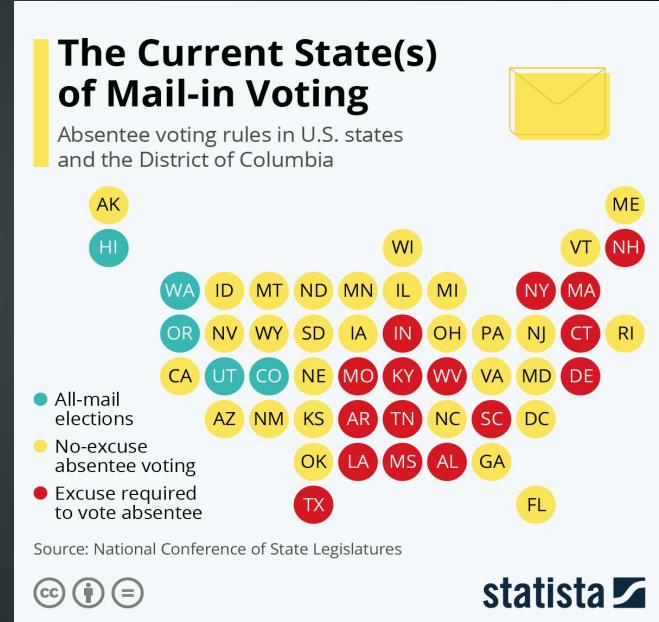
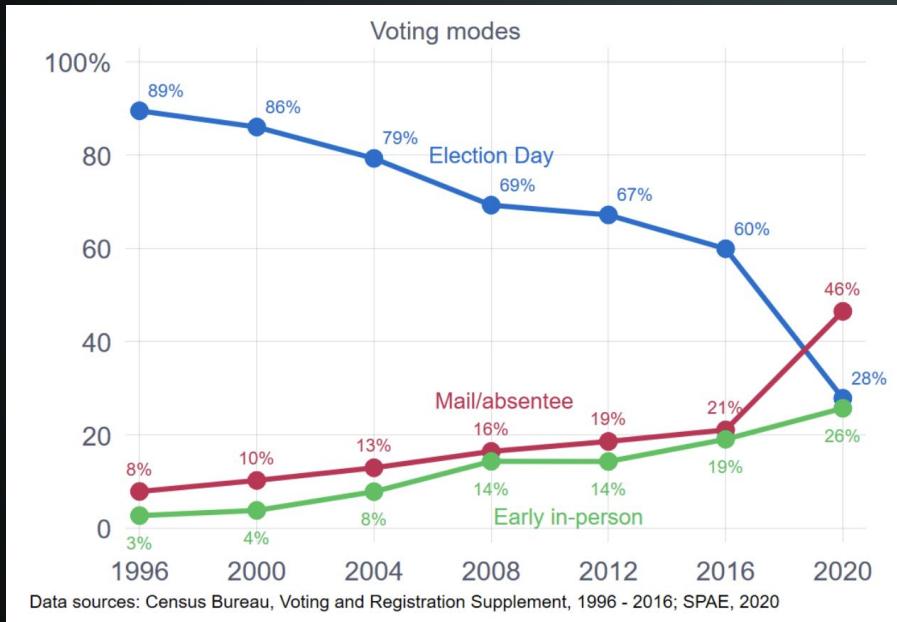


Ballot Curing Project

Nicholas Bowen, Isaac Frumkin, Brice Halder, Andrew Zhang

Background on Voting By Mail



Ballot Curing Efforts

Ballot curing is the process of correcting a ballot that was rejected due to certain issues that prevent it from being counted in its current form.

Common Issues That Can Be Cured

- Missing Signature
- Invalid Signature
- Additional ID Needed
- Submitted provisional ballot

**IMPORTANT INFORMATION
ABOUT THE JANUARY 5, 2021 SENATE RUNOFF ELECTION**

Dear _____,

Public records show that you must submit a copy of your ID so that your absentee ballot for the January 5, 2021 runoff election will count.

Step 1: Find an acceptable form of ID that shows your name and address, such as:

- Georgia Driver's License (can be expired)
- Valid Photo ID card issued by an entity of the state of Georgia, any other state, or the U.S.
- Valid U.S. Passport
- Student photo ID card issued by a Georgia public college, university, or technical school
- Valid employee photo ID issued by the U.S. government, Georgia government, or any county, municipality, board, authority, or other Georgia government entity
- Valid U.S. military ID card with photo
- Valid Tribal ID with photo
- Current utility bill showing name and address
- Current government check or paycheck showing name and address
- Current government document showing name and address
- Current bank statement showing name and address

Step 2: Submit a copy of your ID to your county registrar's office:

- By email - Send an email with a photo of your ID attached.
- In person - You or a third party can hand deliver a photocopy or printed photo of your ID.
- By fax - Fax a copy of your ID to your county elections office.
(Contact info:

Stakeholders

Tracking mail-in ballots

Most states allow voters to check online if their early votes are received by election officials. If a mail-in ballot is invalidated due to damage or a signature discrepancy, how it gets fixed depends on where you live.

States allowing voters to **track their ballots** after they are returned:



Sources: AP reports; National Conference of State Legislatures

AP

States **requiring a voter be notified** if a signature problem arises on a mail-in ballot:

AZ	CA	CO	FL	GA
HI	IL	IA	MA	MN
MT	NV	NJ	OH	OR
RI	UT	WA		

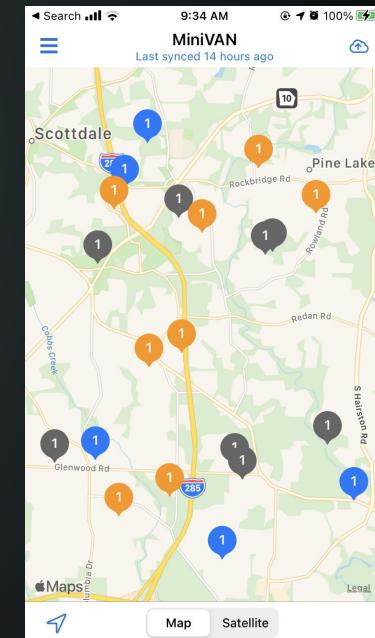


Motivating Case Examples - Recent Georgia Elections

- In the November 2020 Presidential Election, the pivotal swing state of Georgia was decided by 11,779 votes
 - Estimated **~20,000 cured ballots**

Motivating Case Examples - Recent Georgia Elections

- Spent several weeks cure canvassing around Atlanta and parts of northern Georgia for January 2021 Senate elections
- Relevant takeaways for this project:
 - Clear disparities in the communities in need of curing efforts
 - Structural inefficiencies
 - Potential for scalable impact



Interviews

Karin Ascenio - Colorado Democratic Party (Volunteer Coordinator)

Seth Morris - NC Democratic Party (Voter Protection Director)

Bruce Norikane - CO Democratic Party (Tech Director)

Colorado Secretary of State Office

Izzy Bronstein - Common Cause (National Campaigns Manager)

Nikki Charlson - MD State Board of Elections (Deputy Admin)

John Schultz - LTN Global (VP of Software Development)

Major Pain Points Identified

1. Widespread mistrust of vote-by-mail/ballot curing
 - a. Lack of transparency and accurate information from certain state and local election offices
 - b. Falsehoods spread by leaders have sowed unfounded fears in large swaths of voters
2. Inefficient data collection/handling processes
 - a. VoteBuilder (VAN) has monopoly, lack of functioning alternatives
 - i. Costly, glitchy, and not automated
 - b. Varying processes across states

FACT CHECK

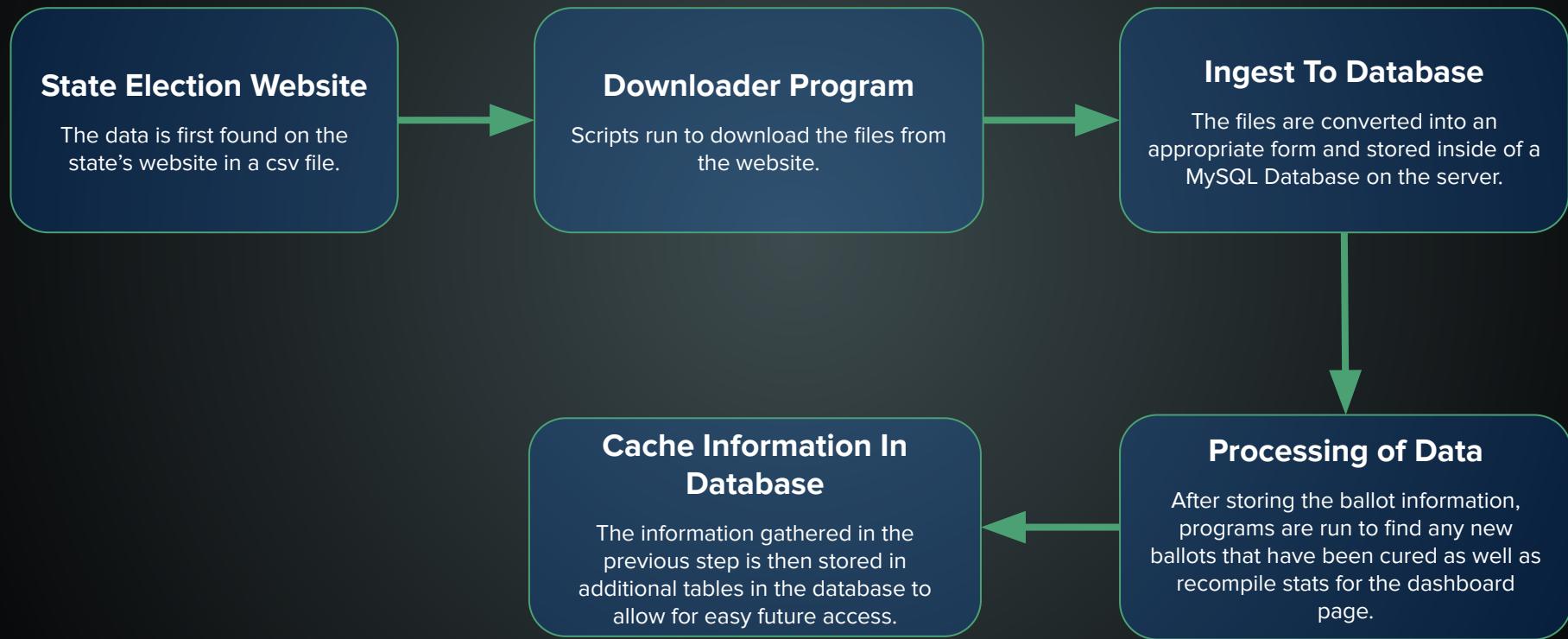
Fact check: Georgia ballot curing is not election fraud

Viral tweet spreads false information about mail ballots in DeKalb County, Ga.

Our Project

Design and implement a software system that will improve the
efficiency and **transparency** of the ballot curing process across
multiple states

Flow of Data Into System

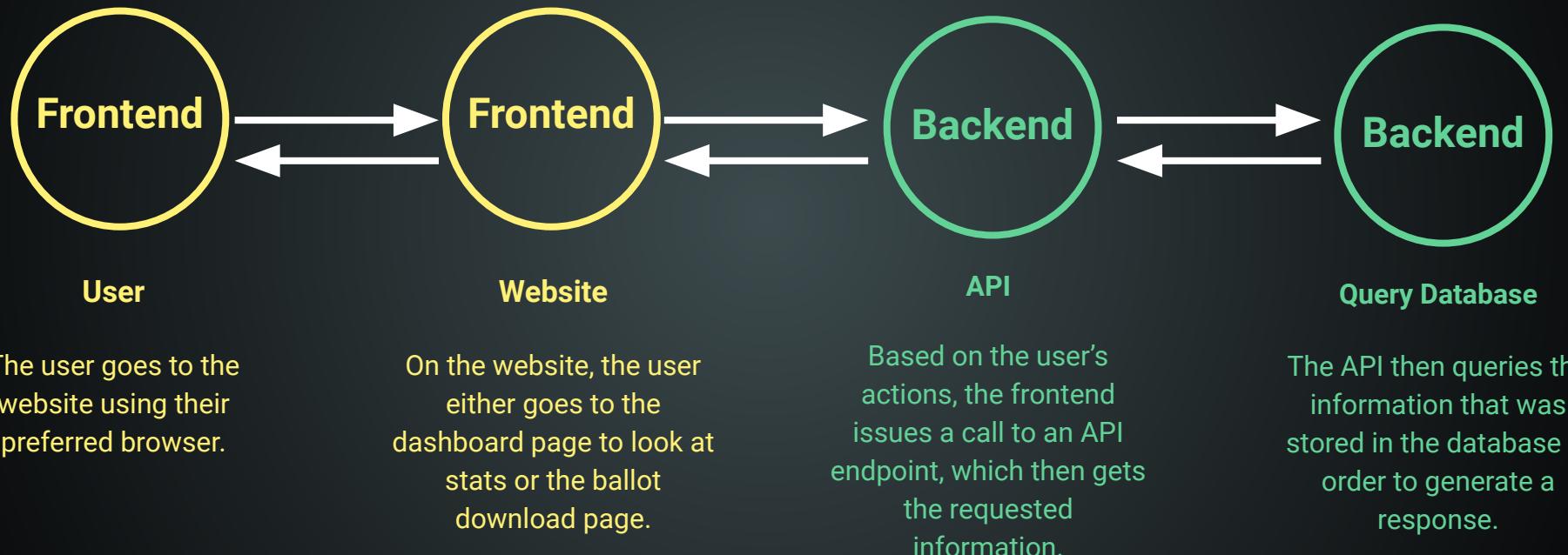


Ingest Demo

This short video shows a demonstration on how the downloading process works.

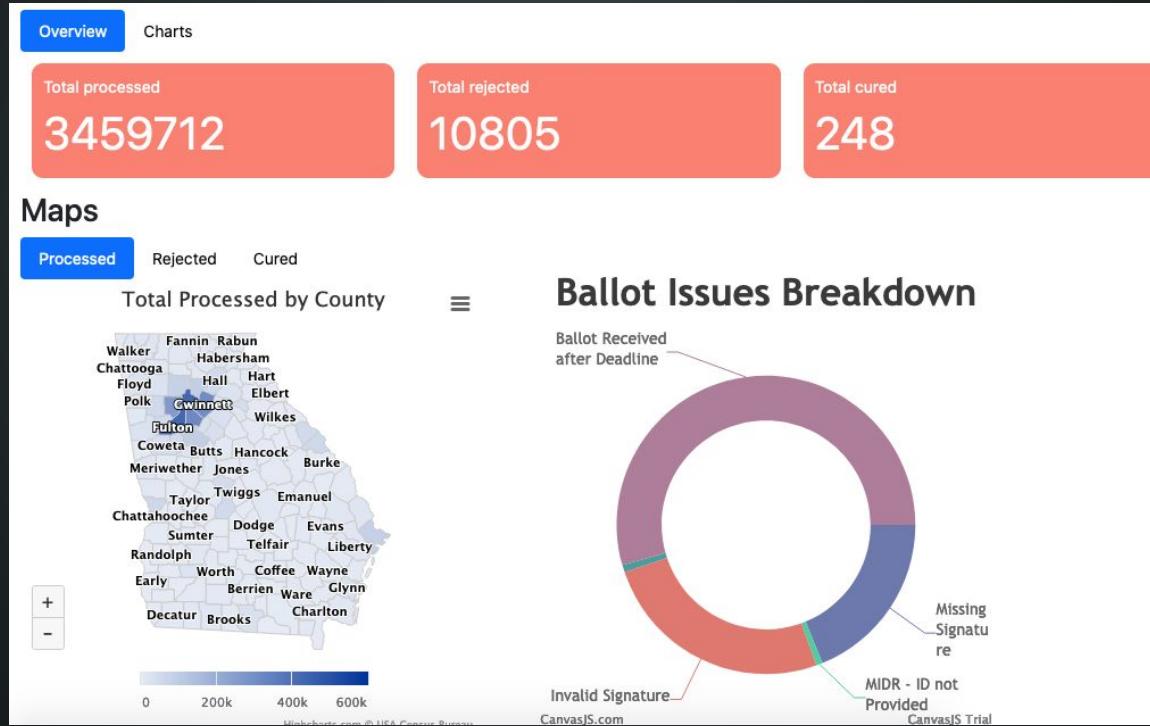
In actual usage, run without UI (headless)

How User Interacts With The System



Dashboard Demo

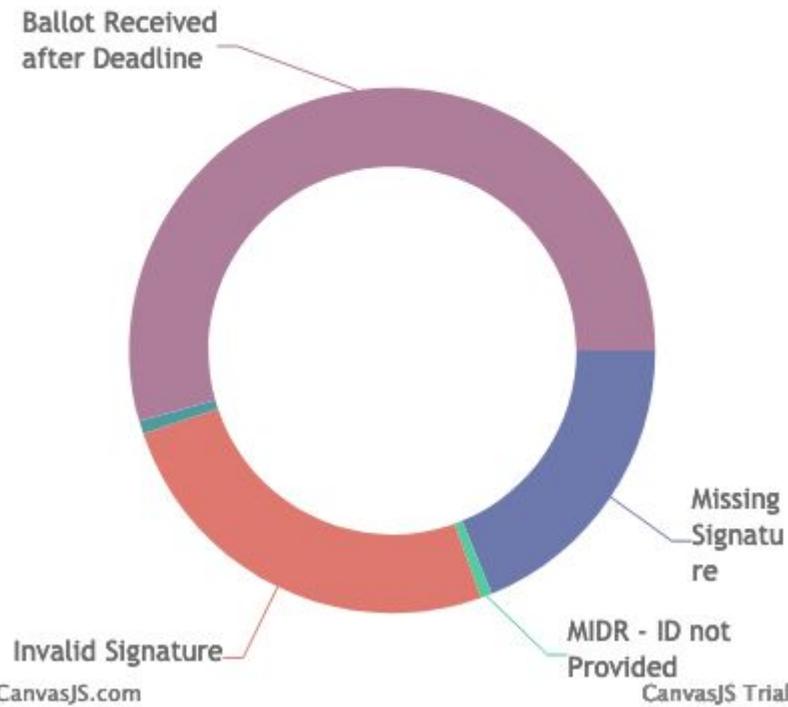
<http://rain16.cnds.jhu.edu/ballot-curing/dashboard/index.html>



GA Rejection %

- Nearly half of rejected ballots can still be cured

Ballot Issues Breakdown



Disparities in Mail-Ballot Rejections (NC - 11/03/20)

(Race)	White	Black	Asian	Native American	Undesignated	Other	Two+ Races
% of All Ballots	65.93%	19.52%	1.47%	0.56%	9.87%	2.13%	0.52%
% of All Rejected Ballots	50.42%	29.45%	3.41%	0.91%	11.05%	3.95%	0.82%

(Age)	18-29	30-44	45-64	65+
% of All Ballots	15.28%	21.07%	36.10%	27.56%
% of All Rejected Ballots	23.71%	15.06%	28.43%	32.81%

Disparities in Cure Rates (NC - 11/03/20)

(Race)	White	Black	Asian	Native American	Undesignated	Other	Two+ Races
% Rejected	0.18%	0.36%	0.55%	0.39%	0.26%	0.44%	0.38%
% Cured	47.59%	33.52%	25.96%	25.36%	34.91%	30.85%	35.86%

(Age)	18-29	30-44	45-64	65+
% Rejected	0.37%	0.17%	0.19%	0.28%
% Cured	31.03%	34.29%	38.82%	50.70%

Download Demo

<http://rain16.cnds.jhu.edu/ballot-curing/ballotFiles/index.html>

Download Absentee Ballot File

Follow the 3 steps below to view (and download) absentee ballot statuses of voters in the selected election

*Note: Downloaded files include additional attributes that are absent from the displayed table

1) GA ▾

2) 01-04-2021 ▾

3) (Optional) Select Additional Parameters Below

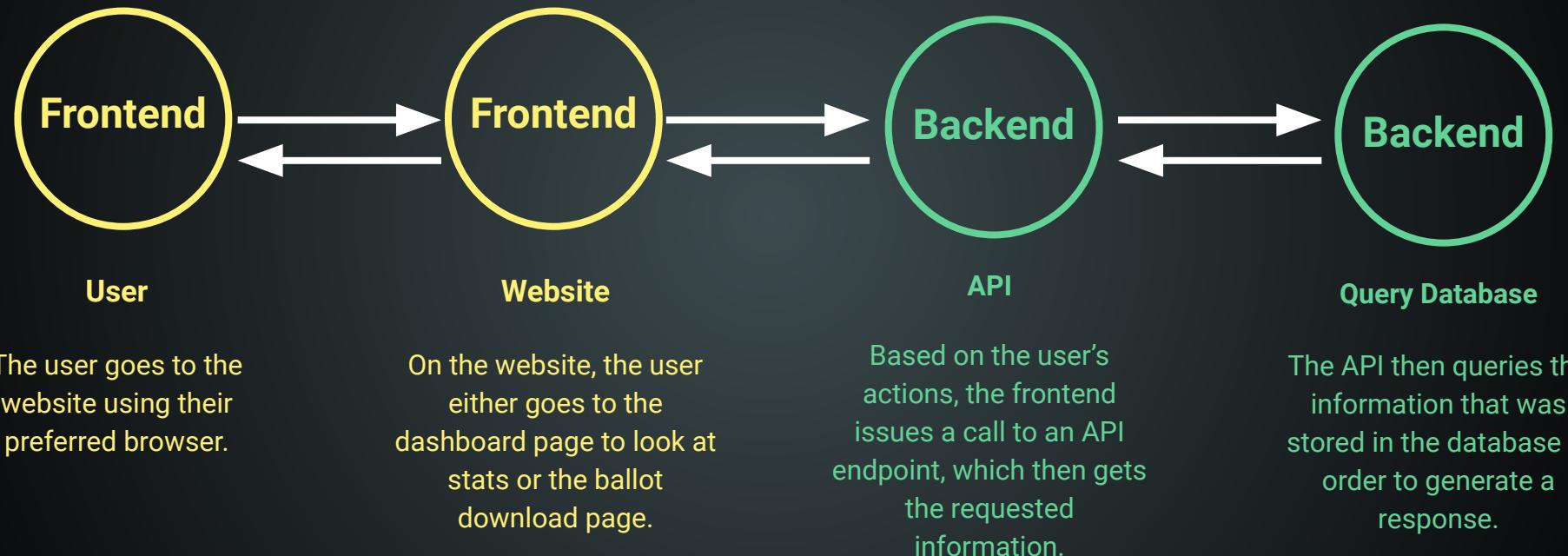
Enter Clear Filters

Download File (32 entries)

county	voter_reg_id	city	state	zip	ballot status	ballot issue
CARROLL	03299948	ROOPVILLE	GA	30170-2537	R	Ballot Received after Deadline
CARROLL	12780481	CARROLLTON	GA	30117	R	Ballot Received after Deadline
CARROLL	04219098	CARROLLTON	GA	30116	R	Ballot Received after Deadline
CARROLL	02163110	ROOPVILLE	GA	30170-2343	R	Missing Signature
CARROLL	11688968	VILLA RICA	GA	30180	R	Invalid Signature

Implementation Details

How User Interacts With The System



Tech Stack

Frontend:



Backend:



Database:

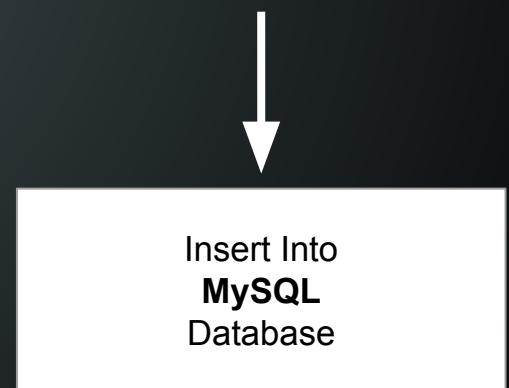
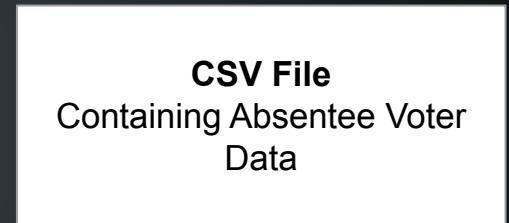
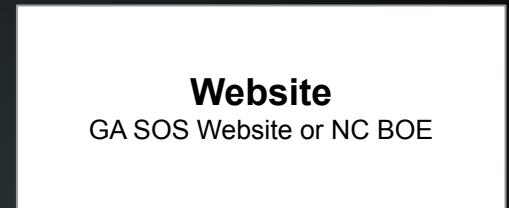


Services:



Data Ingestion

- Download absentee ballot file from state election site
 - Varying process for each state
- Insert into state database
 - Target database & table set in config
 - Process to fit schema
 - Update processed date
- Creates schema tables if not made yet
 - Easy to add elections and states



Database Design

- Database for each state
- Static state-wide tables: **elections**, **counties**
- For each election:

elections

counties

- Tables created: all ballots, rejected ballots, cured ballots

all_DDMMYYYY

rejected_DDMMYYYY

cured_DDMMYYYY

- Add entry to these tables: statewide stats, county-wide stats, statewide time-series info, county-wide time-series info are updated

state_stats

county_stats

state_time_series

county_time_series

Ballot Status Tables (Processed, Rejected, Cured)

For each ballot...

status_MMDDYYYY

Basics...

Demographics...

Political Info...

Ballot Info...

Basics

id
proc_date
county
voter_reg_id
first_name
middle_name
last_name

Demographics

race
ethnicity
gender
age
street_address
city
state
zip

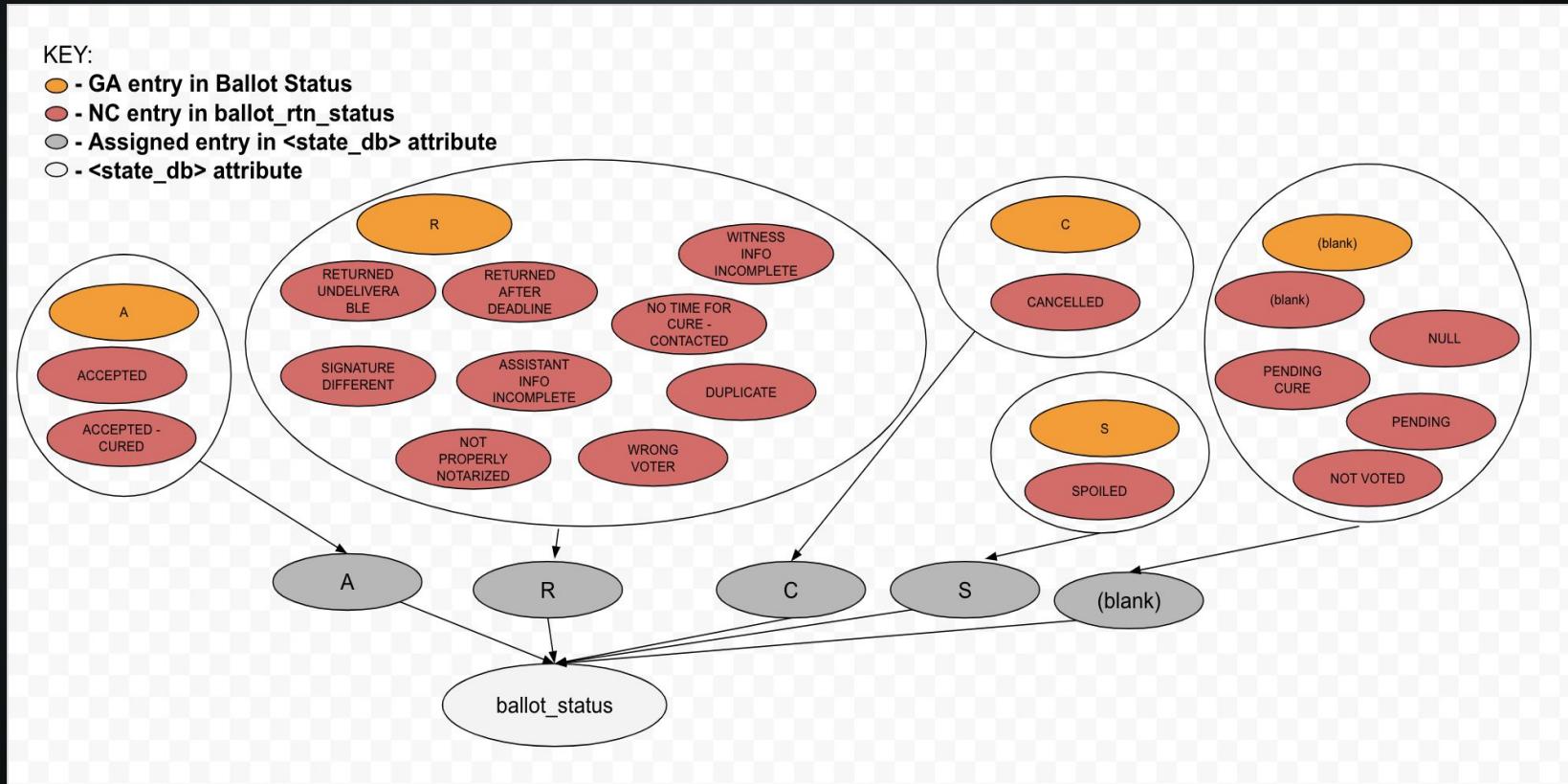
Political Info

election_dt
party_code
precinct
cong_dist
st_house
st_senate

Ballot Info

ballot_style
ballot_req_dt
ballot_send_dt
ballot_ret_dt
ballot_issue
ballot rtn status

Standardization Across States



Handling Different States

		GA	NC
1	Race, ethnicity, age, political party data	✗	✓
2	Separate ballot issue and ballot status	✓	✗
3	Daily data dump contains cumulative ballot info	★	✗
4	Distinguishes between cured and regular accepted	✗	✓

Georgia Ballot Roadblock



Georgia daily data dump did not actually contain cumulative data

- Would have meant only needed last day's data to find out information about what ballots were cured
- Turned out not to be the case so had to rework our methodologies for Georgia
- Needed to reformat how cured ballots were discovered

Algorithm 1: FIND_CURED

1. **for** each day in the election **do**
 2. Let **accepted** contain all of the ballots accepted up to that day
 3. Let **rejected** contain the ballot that were rejected on the previous day
 4. **newly_cured** = MERGE **accepted** and **rejected** on the voter registration number
 5. Add the ballots contained in **newly_cured** to the cured table in the database
 6. **end for**
-

Implementation Differences

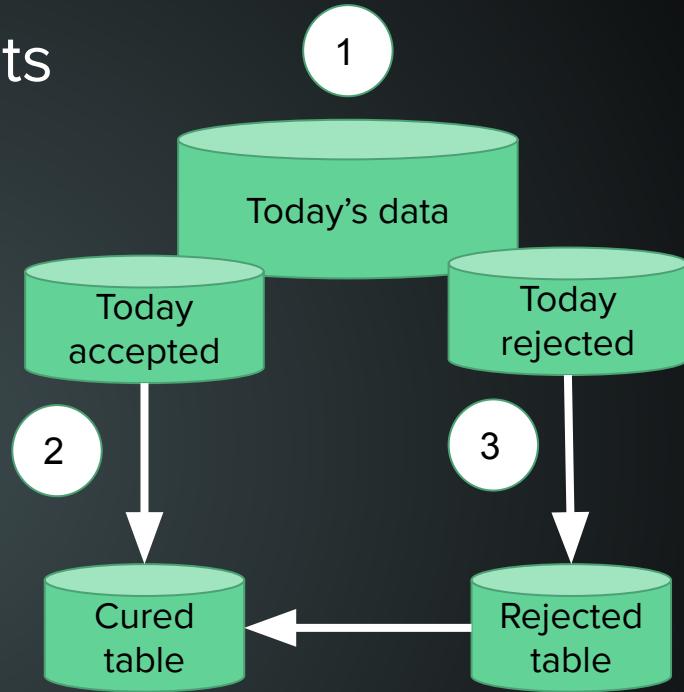
NC specifies cured vs. accepted, meaning our general algorithm was overkill.

Algorithm 1: FIND_CURED

1. **for** each day in the election
 2. Let **accepted** contain all the ballots accepted up to that day
 3. Let **rejected** contain all the ballot that were rejected on the previous day
 4. **newly_cured** = M_{Day}(**accepted** and **rejected** on the voter registration number)
 5. Add the ballots contained in **newly_cured** to the cured table in the database
 6. **end for**
-

Finding Cured and Rejected Ballots

- 1 **find_cured** script runs on new day's downloaded election data
- 2 Script looks for newly accepted ballots in rejected table, adding them to cured table
- 3 Adds any newly rejected ballots from today to the rejected table



Statistic Compilation - Overview

for each “active” election
compute today’s processed, cured, rejected
ballots

compute today’s demographic data

store in database

- Stored once per day → time series data
- State- and election-agnostic

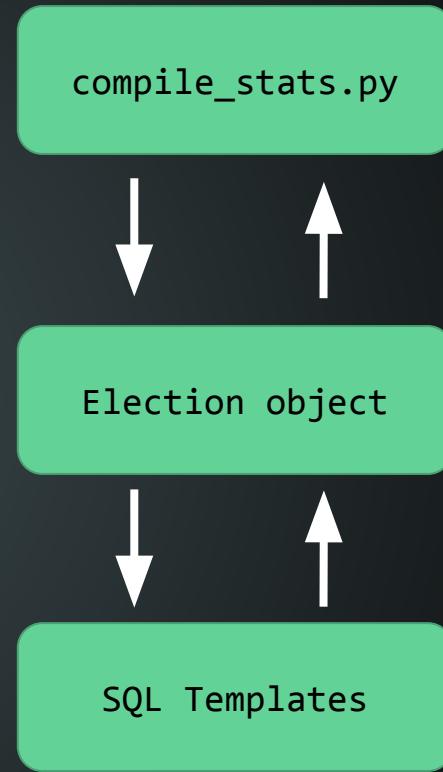
Compute statistics for
state- and county-level data



Insert into database for
quick retrieval

Statistic Compilation - Details

- Election Class representation:
 - SQL Cursor
 - State
 - County (default = None)
- Querying methods for:
 - Aggregate data
 - Demographic data
 - Daily unique data
- Prevent SQL Injection, cleaner design
- Creates statistic tables if not already present
 - Extensibility



API

How it Works?

- The API is written using Flask (a Python microframework). Each endpoint exists as a separate file that is then compiled together. Whenever a user (either by directly sending a request to the API or through interacting with our frontend) sends a request to a specific endpoint, the API then takes the request parameters and then queries the database based off those. It then sends the response in the form of an HTTP response.

Endpoints:

- Ballots, Stats (as well as county stats and time series), Last Processed

Example of flow for
Ballots endpoint:

User hits enter on the
Download Ballots Page



GET request sent to the **API**
with any specified
parameters



The API then does some
processing of the query
result before sending back
an **HTTP response**.

Purpose of Each Endpoint

Ballots

- This endpoint is used to query the general table in order to get a list of ballots based off specified parameters.

Download

- This endpoint has the same purpose as the ballots endpoint except it returns the information in the form of a CSV file.

Last Processed

- Returns information on the last time that the information for a certain state and election combination was updated.

Stats

- The function of this endpoint is to provide aggregate stats about the entire state. Examples of stats included are total rejected/cured, breakdown of cured/rejected by race, and more.

County Stats

- Provides similar stats/information as the main stats endpoint but instead provides it at the county level instead of state level.

Time Series

- Returns information on certain statistics on a day by day level instead of an aggregate level.

In-Depth Dive into the Download Endpoint



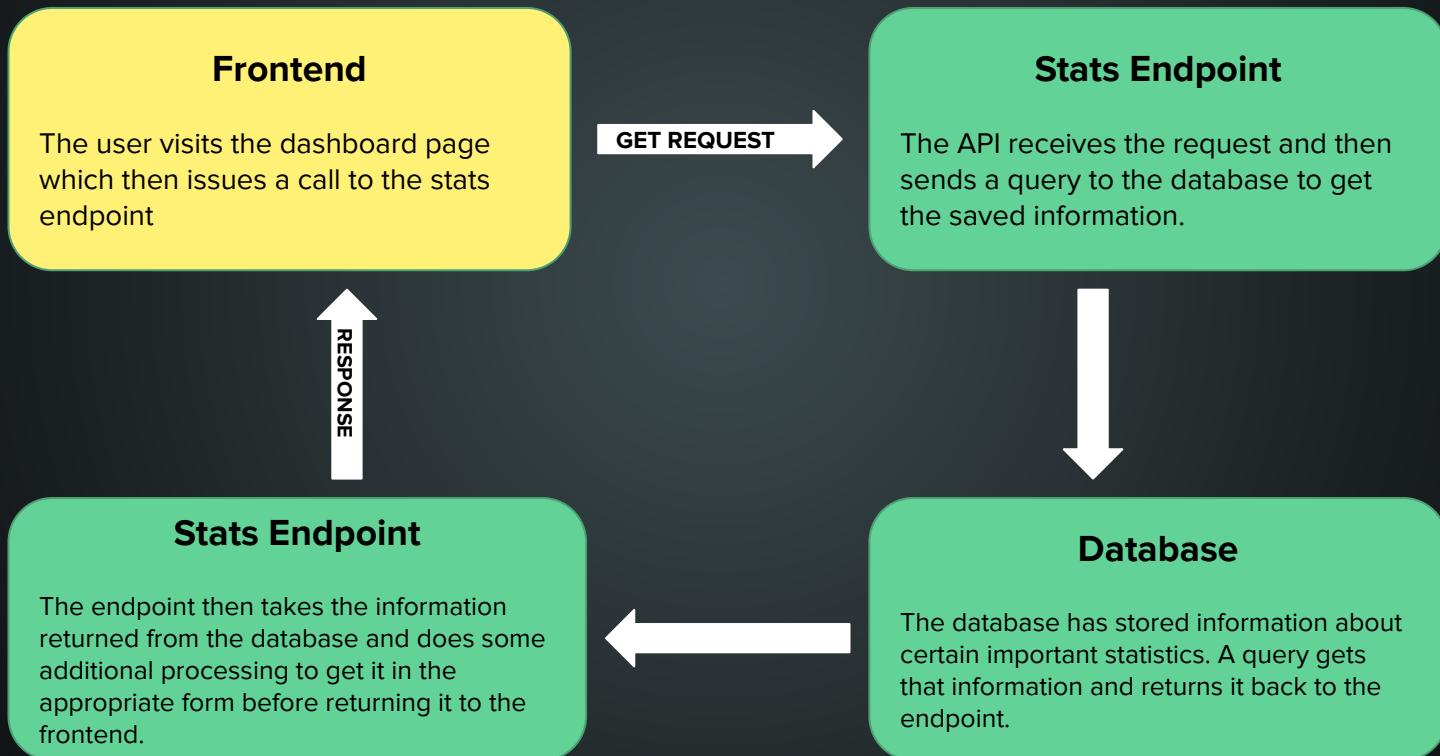
The user presses the download button on the website which causes the frontend to send a request to the API

The endpoint then writes the result of the query to a csv file and sends it as an attachment to a HTTP response.

Through the use of an after request tag, the endpoint then deletes the temporary csv file after it sends it as an attachment.

The download endpoint places a call to a function that gets the result from a database query based on the specified parameters

In-Depth Dive into the Stats Endpoint



Dashboard Page Architecture

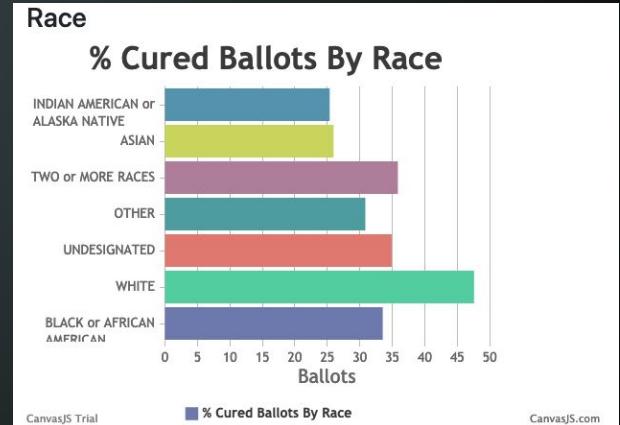
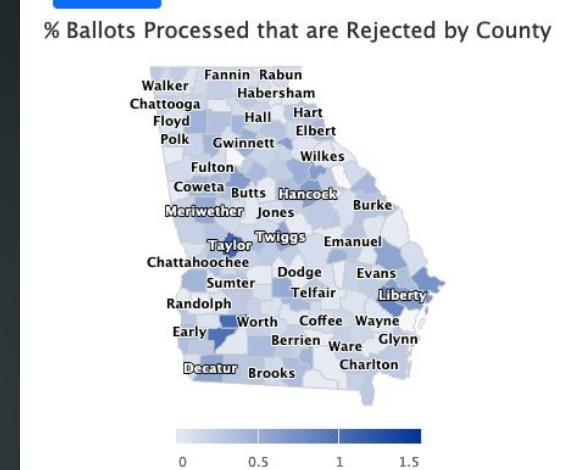
- jQuery AJAX calls
 - GET Request: state data
 - GET Request: county data
 - GET Request: time series data
 - GET Request: last updated date
- Highcharts and ChartJS API

Today's Snapshot

Last updated: 04/13/2021

Overview

Charts



Download Page Architecture

- jQuery AJAX calls:
 - GET Request: Ballots endpoint
 - GET Request: Download endpoint
- Pull data for dropdowns from JSON file
 - Based on user's selected state & election

```
{
  "GA": {
    "elections": [
      "01-04-2021", "11-03-2020",
      "01-04-2021": {
        "date": "01-04-2021",
        "status": "P",
        "issues": "Ballot Received after Deadline,Ineligible Elector,Invalid Signature,MI"
      },
      "11-03-2020": {
        "date": "11-03-2020",
        "status": "R",
        "issues": "Ballot Received after Deadline,Ineligible Elector ,Invalid Signature,M"
      }
    ],
    "counties": "Appling,Atkinson,Bacon,Baker,Baldwin,Banks,Barrow,Bartow,Ben Hill,Berrien,Bi",
    "cities": "Abbeville,Acworth,Adairsville,Adel,Adrian,Ailey,Alamo,Alapaha,Albany,Aldora,Alt"
  },
  "NC": {
    "elections": [
      "11-03-2020", "11-03-2020",
      "11-03-2020": {
        "date": "11-03-2020",
        "status": "P",
        "issues": "Assistant Info Incomplete,Conflict,Duplicate,E-Transmission Failure,N"
      }
    ],
    "counties": "Alamance,Alexander,Alleghany,Anson,Ashy,Avery,Baufort,Bertie,Bladen,Brunswi",
    "cities": "Aberdeen,Advance,Ahoskie,Alamance,Albemarle,Alliance,Altamahaw,Andrews,Angier,A"
  }
}
```

Download Absentee

Follow the 3 steps below to view (and download) absentee ballots.

*Note: Downloaded files include additional attributes that are not displayed here.

- 1) GA ▾
- 2) 11-03-2020 ▾
- 3) (Optional) Select Additional Parameters Below

County	City	Ballot Status	Ballo
Search..	g_id	city	
Appling			
Atkinson			
Bacon			
Baker			
Baldwin			
Banks			
Barrow			

Looking Ahead

Process of Onboarding New States

- Write a downloader and insertion script for that state in order to download the data. This is also where any needed standardization would occur (similarly to North Carolina).
- Run the other scripts in order to generate the stats, as well as the cured and rejected table for that state.
- Finally, update the website in order to support the state on the downloader page and on the dashboard page.

Configuration

- Configurations unique to each state
- Referenced in download scripts
- Keys, passwords, machine-specific paths

Example

```
[SYSTEM] /home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py", line 235, in
download_dir: /home/cs310_prj3/Ballot-Curing-Project
  File "/home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py", line 229, in
    compile_county_stats('GA', proc_date, election)
[DATABASE]
host: [REDACTED]
user: [REDACTED]
passwd: [REDACTED]
  File "/home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py", line 21, in
    create_state_time_series_table()
[GA]
res = self._query(query)
year: 2020
name: 01/05/2021 - JANUARY 5, 2021 FEDERAL RUNOFF ELECTION
filename: 35211.zip
  File "/home/cs310_prj3/.conda/envs/proj3/lib/python3.9/site-packages/MySQLdb/connection
      _mysql.connection.query(self, query)
KeyboardInterrupt
timeout: 600
storage_dir: /home/cs310_prj3/storage/GA
csv_name: STATEWIDE.csv
[GA] [REDACTED]@[REDACTED]:~$ py scripts/compile_stats.py
db: vote_ga 2021-05-03 17:52:08,316 - dev - INFO - Computing GA state-level statistics for 01_05_2021
table: 01_05_2021 13,849 - dev - DEBUG - Time series data entry finished @ 0.14s
table-test:jan5runoff 50 - dev - DEBUG - Total time for GA state-level statistics: 5.53s
ga_files: /storage/ga_files 2021-05-03 17:52:08,316 - dev - INFO - Computing GA county-level statistics for 01_05_2021
rain_ga_storage: /home/cs310_prj3/storage/ga-files
test_path: /home/cs310_prj3/Ballot-Curing-Project/test_GA_data
[NC]
[NC] 2021-05-03 17:52:22,977 - dev - DEBUG - Time series data entry finished @ 0.00s
[NC] 2021-05-03 17:52:22,977 - dev - DEBUG - GA county BACON, #3 / 159
zip_filename: nCDATA.zip 2021-05-03 17:52:22,977 - dev - DEBUG - proc_date: 2020-11-29, elec_dt: 2021-01-05 00:00:
url: https://s3.amazonaws.com/dl.ncsbe.gov/ENRS/2020_11_03/absentee_county_20201103.zip
db: vote_nc 2021-05-03 17:54:00,768 - dev - DEBUG - Time series data entry finished @ 97.49s
table: statewide 00,768 - dev - DEBUG - GA county BAKER, #4 / 159
csv_name: absentee_20201103.csv 2021-05-03 17:54:00,768 - dev - DEBUG - proc_date: 2020-11-29, elec_dt: 2021-01-05 00:00:
storage_dir: /home/cs310_prj3/storage/NC 2021-05-03 17:54:00,768 - dev - DEBUG - Time series queries retrieved from db @ 100.43s
[NC] 2021-05-03 17:54:00,768 - dev - DEBUG - Time series data entry finished @ 101.47s
```

Schema Changes

- (ADD STUFF HERE)

Organizational Features for Volunteering Efforts

- Ability for organizations to manage their ballot curing efforts
 - Queried lists divided into clusters based on location

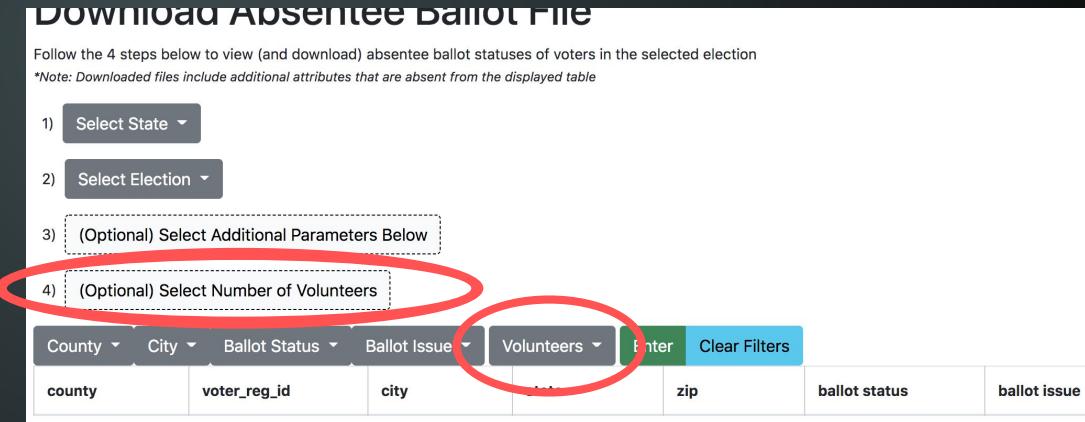
Download Absentee Ballot File

Follow the 4 steps below to view (and download) absentee ballot statuses of voters in the selected election
*Note: Downloaded files include additional attributes that are absent from the displayed table

- 1) Select State ▾
- 2) Select Election ▾
- 3) (Optional) Select Additional Parameters Below
- 4) (Optional) Select Number of Volunteers

County ▾ City ▾ Ballot Status ▾ Ballot Issue ▾ Volunteers ▾ Filter Clear Filters

county	voter_reg_id	city	state	zip	ballot status	ballot issue
--------	--------------	------	-------	-----	---------------	--------------



Authentication

- Different states have different levels of access for ballot data
 - GA, NC: publicly available on website
 - CO, MD: pay money to access
- Develop method to stagger access for organizations via API keys

Handoff

Documentation

API Documentation:
<https://docs.google.com/document/d/1RnHn42gtodQffliAMX15rEl24evBRC8LPwePILjp6Ok/edit>

API Docs

Design Philosophy: We want a minimal number of simple API endpoints with optional parameters to account for various data requests.

Ballots

GET api/v1/ballots/rejected

Returns information on rejected ballots of a state. By default, returns the most up-to-date information for every voter. Optional parameters are to filter ballot query by ballot attributes or to get historical data.

If someone is accessing protected state information, they must include a auth token in the header of the request

Required Parameters

param_name	type	example	description
state	string	"MD"	The state for the election
election_dt	datetime	11-5-2020	The date of the election, format %m-%d-%Y

Installation

GitHub repos and code on website:

- Backend: <https://github.com/Ballot-Curing/ballot-curing-backend>
- Frontend: <https://github.com/Ballot-Curing/ballot-curing-dashboard>
- Website: <http://www.cnds.jhu.edu/courses/cs310/ballot-curing/>

Instructions to run are in READMEs

Acknowledgments

Special thanks to all those we interviewed, Sahiti Bommareddy, Daniel Qian, Jerry Chen, and especially Professor Yair Amir for running the class and continually pushing us and challenging us to do better.

Questions?

Intro

(intro vote by mail, what is ballot curing, and our value proposition)

- Problem motivation
 - Slim margins in recent elections, rejected ballots, etc
- Isaac talks about his experience curing in Georgia
- The people we interviewed and what we learned from each

Downloader/Ingest Program

How it works?

- The program downloads the file from the election site. This file contains the information of the voters who voted via absentee ballots. An additional script then takes the file and inserts into to our MySQL Database. The program is written in Python and takes advantage of Selenium and the MySQL connector module among others.

Stage 1

Website

GA SOS Website or NC BOE

Stage 2

CSV File

Containing Absentee Voter Data

Stage 3

Inserted Into MySQL Database

High level demo

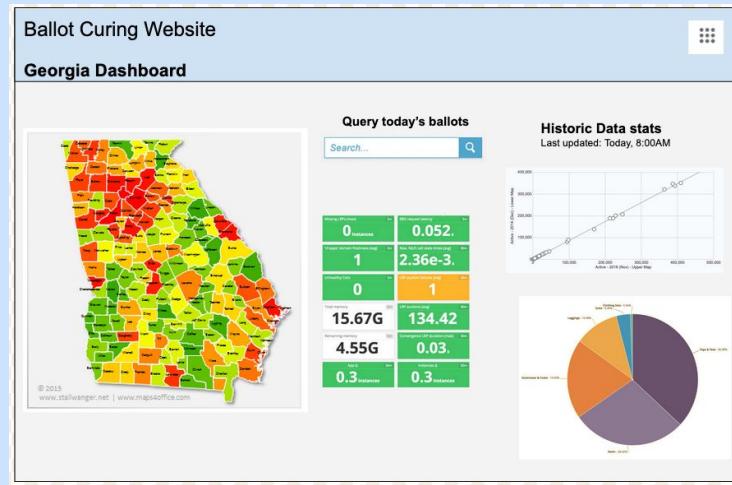
Go thru a basic demo

- Video of Selenium
 - Download from SOS
- Ingest process
 - Finding cures, rejected
 - Stat compilation
- Interactions with the dashboard
- Interactions with download page

Frontend Demo

<http://rain16.cnds.jhu.edu/ballot-curing/dashboard/index.html>

Original wireframe:



Implementation Details

- Components and how they interact at a lower level
- What data do we have
- Schema
- API
- Frontend

Database

- Current data
 - GA
 - NC
- Unified schema formation
 - Differences
- Standardization

Basics

id
proc_date
county
voter_reg_id
first_name
middle_name
last_name

Demographics

race
ethnicity
gender
age
street_address
city
state
zip

Political Region

election_dt
party_code
precinct
cong_dist
st_house
st_senate

Ballot Info

ballot_style
ballot_req_dt
ballot_send_dt
ballot_ret_dt
ballot_issue
ballot rtn status

Improvements for the future

- Changing the schema for performance reasons
- Ability to group queried voters (i.e. 5 groups of 40 people in Cobb county) for organization purposes
- Authentication

Outline for 11/29 (20 min)

High level demo: (5-10 min)

- Frontend side:
 - Playing around with dashboard (looking at different states)
 - Going to downloads page and selecting different possibilities

Low level explanation of the software architecture → how it's running, its components, etc

- How components interact with each other
- Data -> how much data, what states we have
- Go in depth about architecture and schema → how the unified schema came to be, differences
- How our API works - explain many endpoints
- PIPELINE VISUALIZATION FOR DATABASE

Demo lower level things in the system → most of the details in the backend

Low level frontend (AJAX, API calls, downloader)

Outline for Final (50 min)

Intro: 10 min (intro vote by mail, what is ballot curing, and our value proposition)

- Motivate the problem
- Isaac can talk about his experience curing in Georgia
- The people we interviewed and what we learned from each

High level demo: (5-10 min)

- Frontend side:
 - Playing around with dashboard (looking at different states)
 - Going to downloads page and selecting different possibilities

Outline for Final (50 min)

Low level explanation of the software architecture → how it's running, it's components, etc

- How components interact with each other
- Data -> how much data, what states we have
- Go in depth about architecture and schema → how the unified schema came to be, differences
- How our API works - explain many endpoints

Demo lower level things in the system → most of the details in the backend

Low level frontend

Improvements for the future

- Talk about schema improvements (Sahiti recommended a different schema to begin with)
- Authentication
- Onboarding more states

Handoff

- How to continue this project (things to install, steps to go over, etc)
- How it would work in real life, during an election