Ballot Curing Project

Nicholas Bowen, Isaac Frumkin, Brice Halder, Andrew Zhang
Background on Voting By Mail

![Graph showing voting modes over time](image1)

**The Current State(s) of Mail-in Voting**

Absentee voting rules in U.S. states and the District of Columbia

Source: National Conference of State Legislatures
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
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:

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

Sources: AP reports; National Conference of State Legislatures
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
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

State Election Website
The data is first found on the state’s website in a csv file.

Downloader Program
Scripts run to download the files from the website.

Ingest To Database
The files are converted into an appropriate form and stored inside of a MySQL Database on the server.

Cache Information In Database
The information gathered in the previous step is then stored in additional tables in the database to allow for easy future access.

Processing of Data
After storing the ballot information, programs are run to find any new ballots that have been cured as well as recompute stats for the dashboard page.
Ingest Demo

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

In actual usage, run without UI (headless)
The user goes to the website using their preferred browser.

On the website, the user either goes to the dashboard page to look at stats or the ballot download page.

Based on the user’s actions, the frontend issues a call to an API endpoint, which then gets the requested information.

The API then queries the information that was stored in the database in order to generate a response.
Dashboard Demo

http://rain16.cnds.jhu.edu/ballot-curing/dashboard/index.html
GA Rejection %

- Nearly half of rejected ballots can still be cured
# Disparities in Mail-Ballot Rejections (NC - 11/03/20)

<table>
<thead>
<tr>
<th>(Race)</th>
<th>White</th>
<th>Black</th>
<th>Asian</th>
<th>Native American</th>
<th>Undesignated</th>
<th>Other</th>
<th>Two+ Races</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of All Ballots</td>
<td>65.93%</td>
<td>19.52%</td>
<td>1.47%</td>
<td>0.56%</td>
<td>9.87%</td>
<td>2.13%</td>
<td>0.52%</td>
</tr>
<tr>
<td>% of All Rejected Ballots</td>
<td>50.42%</td>
<td>29.45%</td>
<td>3.41%</td>
<td>0.91%</td>
<td>11.05%</td>
<td>3.95%</td>
<td>0.82%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Age)</th>
<th>18-29</th>
<th>30-44</th>
<th>45-64</th>
<th>65+</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of All Ballots</td>
<td>15.28%</td>
<td>21.07%</td>
<td>36.10%</td>
<td>27.56%</td>
</tr>
<tr>
<td>% of All Rejected Ballots</td>
<td>23.71%</td>
<td>15.06%</td>
<td>28.43%</td>
<td>32.81%</td>
</tr>
</tbody>
</table>
# Disparities in Cure Rates (NC - 11/03/20)

## Race

<table>
<thead>
<tr>
<th>(Race)</th>
<th>White</th>
<th>Black</th>
<th>Asian</th>
<th>Native American</th>
<th>Undesignated</th>
<th>Other</th>
<th>Two+ Races</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Rejected</td>
<td>0.18%</td>
<td>0.36%</td>
<td>0.55%</td>
<td>0.39%</td>
<td>0.26%</td>
<td>0.44%</td>
<td>0.38%</td>
</tr>
<tr>
<td>% Cured</td>
<td>47.59%</td>
<td>33.52%</td>
<td>25.96%</td>
<td>25.36%</td>
<td>34.91%</td>
<td>30.85%</td>
<td>35.86%</td>
</tr>
</tbody>
</table>

## Age

<table>
<thead>
<tr>
<th>(Age)</th>
<th>18-29</th>
<th>30-44</th>
<th>45-64</th>
<th>65+</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Rejected</td>
<td>0.37%</td>
<td>0.17%</td>
<td>0.19%</td>
<td>0.28%</td>
</tr>
<tr>
<td>% Cured</td>
<td>31.03%</td>
<td>34.29%</td>
<td>38.82%</td>
<td>50.70%</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>county</th>
<th>voter_reg_id</th>
<th>city</th>
<th>state</th>
<th>zip</th>
<th>ballot status</th>
<th>ballot issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARROLL</td>
<td>03299948</td>
<td>ROOPVILLE</td>
<td>GA</td>
<td>30170-2537</td>
<td>R</td>
<td>Ballot Received after Deadline</td>
</tr>
<tr>
<td>CARROLL</td>
<td>12780481</td>
<td>CARROLLTON</td>
<td>GA</td>
<td>30117</td>
<td>R</td>
<td>Ballot Received after Deadline</td>
</tr>
<tr>
<td>CARROLL</td>
<td>04219098</td>
<td>CARROLLTON</td>
<td>GA</td>
<td>30116</td>
<td>R</td>
<td>Ballot Received after Deadline</td>
</tr>
<tr>
<td>CARROLL</td>
<td>02163110</td>
<td>ROOPVILLE</td>
<td>GA</td>
<td>30170-2343</td>
<td>R</td>
<td>Missing Signature</td>
</tr>
<tr>
<td>CARROLL</td>
<td>11688968</td>
<td>VILLA RICA</td>
<td>GA</td>
<td>30180</td>
<td>R</td>
<td>Invalid Signature</td>
</tr>
</tbody>
</table>
Implementation Details
The user goes to the website using their preferred browser.

On the website, the user either goes to the dashboard page to look at stats or the ballot download page.

Based on the user’s actions, the frontend issues a call to an API endpoint, which then gets the requested information.

The API then queries the information that was stored in the database in order to generate a response.
Tech Stack

Frontend:
- jQuery
- canvasJS
- Bootstrap
- HIGHCHARTS

Backend:
- Flask
- Python

Database:
- MySQL

Services:
- POSTMAN
- GitHub
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

Website
GA SOS Website or NC BOE

CSV File
Containing Absentee Voter Data

Insert Into
MySQL Database
Database Design

- Database for each state
- Static state-wide tables: elections, counties
- For each election:
  - Tables created: all ballots, rejected ballots, cured ballots
  - Add entry to these tables: statewide stats, county-wide stats, statewide time-series info, county-wide time-series info are updated
Ballot Status Tables (Processed, Rejected, Cured)

For each ballot...

**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

status MMDDYYYY
Basics...
Demographics...
Political Info...
Ballot Info...
Standardization Across States
Handling Different States

<table>
<thead>
<tr>
<th></th>
<th>Race, ethnicity, age, political party data</th>
<th>GA</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>Separate ballot issue and ballot status</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>3</td>
<td>Daily data dump contains cumulative ballot info</td>
<td>🌟</td>
<td>✗</td>
</tr>
<tr>
<td>4</td>
<td>Distinguishes between cured and regular accepted</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
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 days
2. Let \textit{accepted} contain all ballots accepted up to that day
3. Let \textit{rejected} contain all ballots that were rejected on the previous day
4. \textit{newly cured} = merge \textit{accepted} and \textit{rejected} on the voter registration number
5. Add the ballots contained in \textit{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
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

```
compile_stats.py
```

```
Election object
```

```
SQL Templates
```
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
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

Frontend
The user visits the dashboard page which then issues a call to the stats endpoint

GET REQUEST

Stats Endpoint
The API receives the request and then sends a query to the database to get the saved information.

Database
The database has stored information about certain important statistics. A query gets that information and returns it back to the endpoint.

RESPONSE

Stats Endpoint
The endpoint then takes the information returned from the database and does some additional processing to get it in the appropriate form before returning it to the frontend.
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
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
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]
homedir: /home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py
download_dir: /home/cs310_prj3/Ballot-Curing-Project
file: /home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py

[DATABASE]
host: [REDACTED]
user: [REDACTED]
pwd: [REDACTED]
year: 2021
name: 01/05/2021 – JANUARY 5, 2021 FEDERAL RUNOFF ELECTION
filename: 35211.zip
keyboardInterrupt
timeout: 600
storage_dir: /home/cs310_prj3/storage/GA
csv_name: STATEWIDE.csv
db: vote_ga
query: 552:03:11 - dev - INFO - Computing GA state-level statistics for 01_05_2021
```

Example

```
[SYSTEM]
homedir: /home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py
download_dir: /home/cs310_prj3/Ballot-Curing-Project
file: /home/cs310_prj3/Ballot-Curing-Project/db/scripts/compile_stats.py

[DATABASE]
host: [REDACTED]
user: [REDACTED]
pwd: [REDACTED]
year: 2021
name: 01/05/2021 – JANUARY 5, 2021 FEDERAL RUNOFF ELECTION
filename: 35211.zip
keyboardInterrupt
timeout: 600
storage_dir: /home/cs310_prj3/storage/GA
```
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
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/1RnHn42gtodQffIiAMX15rEl24evBRC8LPwePILjp6Ok/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

<table>
<thead>
<tr>
<th>param_name</th>
<th>type</th>
<th>example</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>string</td>
<td>&quot;MD&quot;</td>
<td>The state for the election</td>
</tr>
<tr>
<td>election_dt</td>
<td>datetime</td>
<td>11-5-2020</td>
<td>The date of the election, format %m-%d-%Y</td>
</tr>
</tbody>
</table>
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 our MySQL Database. The program is written in Python and takes advantage of Selenium and the MySQL connector module among others.
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, 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
- 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