InnoVote - by Ujjwal Pasupulety, Saurodeep Bhattacharjee & Aditya Srinivasan

Summary

Considering the demographic of the population the solution aims to reach out to, we believe using Google Forms is a simple yet effective way to tackle the problem of physical absence of voters. InnoVote makes use of shareable Google Forms which can be used by voters to conveniently submit their vote response. The votes are stored on a Google Sheet which is designed to avoid duplicate responses. Using the Python Google Developer API, data from the sheet is assimilated in a central SQL database where the voter responses are verified in order to filter out unregistered voter responses. The simple user interface and scalable deployment enabled by Google's omniscient platform offers a unique and immediately implementable solution to tackle the millions of lost votes across the country every year while preserving the integrity of the voting process.

Description

Conventionally, votes cast during an election are tallied manually by volunteers situated across the country. This process is extremely time consuming and is susceptible to human error. InnoVote aims to eradicate these issues by utilizing Google's existing platforms and APIs while incorporating changes that cater to the requirement for a simple, secure and scalable voting system. The following technologies make up InnoVotes procedural pipeline:

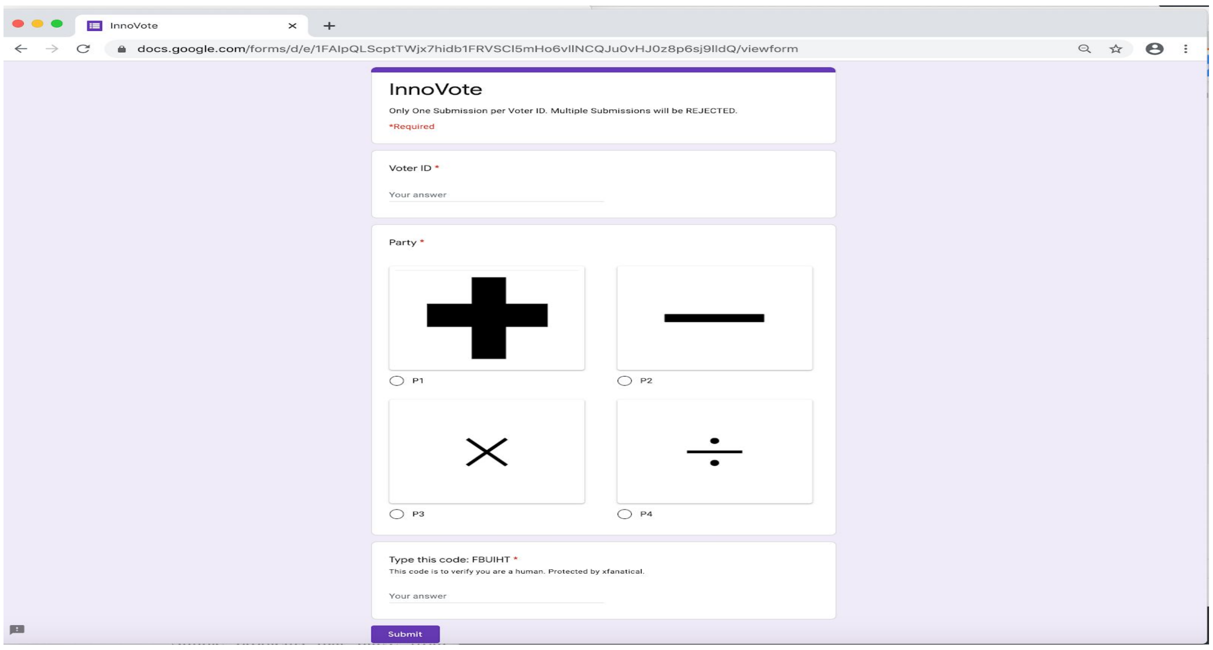
1. Google Forms - Single point of data collection from voters
2. Google Sheets - Consolidated view of voter responses
3. Google Apps Script - Implementation of form CAPTCHA and duplicate vote removal
4. Google Developer Console - Service account creation to enable API calls
5. Google Sheets API - Enable backend voter verification and vote tally

InnoVote's backend is programmed in Python and makes use of the following packages:

1. gspread - Google Sheet data collection
2. sqlite3 - SQL database creation
3. oauth2client.service_account - Security

i) Google Forms

On election day, links to Google forms are mass distributed to all voters who can't make it to the voting booth in their respective constituencies via SMS. Figure 1 depicts a prototype of the form that will be received. Once the user opens the form, he/she can register to vote by entering their unique Voter ID or EPIC no. This Voter ID will be verified later in the pipeline using a centralized SQL database containing all the registered voter information. The user can then choose the party of choice they wish to vote for by marking the appropriate checkbox. Images of parties can be used in order to aid voters who are unable to read the options. Finally voters can submit their response after a simple CAPTCHA test at the end of the form. This mechanism prevents the system from being overwhelmed with entries submitted by spam bots.


docs.google.com/forms/d/e/1FAIpQLScptTWjx7hidb1FRVSCI5mHo6vlINCQJu0vHJ0z8p6sj9lldQ/viewform

InnoVote

Only One Submission per Voter ID. Multiple Submissions will be REJECTED.

*Required

Voter ID *

Your answer

Party *

P1

P2

P3

P4

Type this code: FBUIHT *

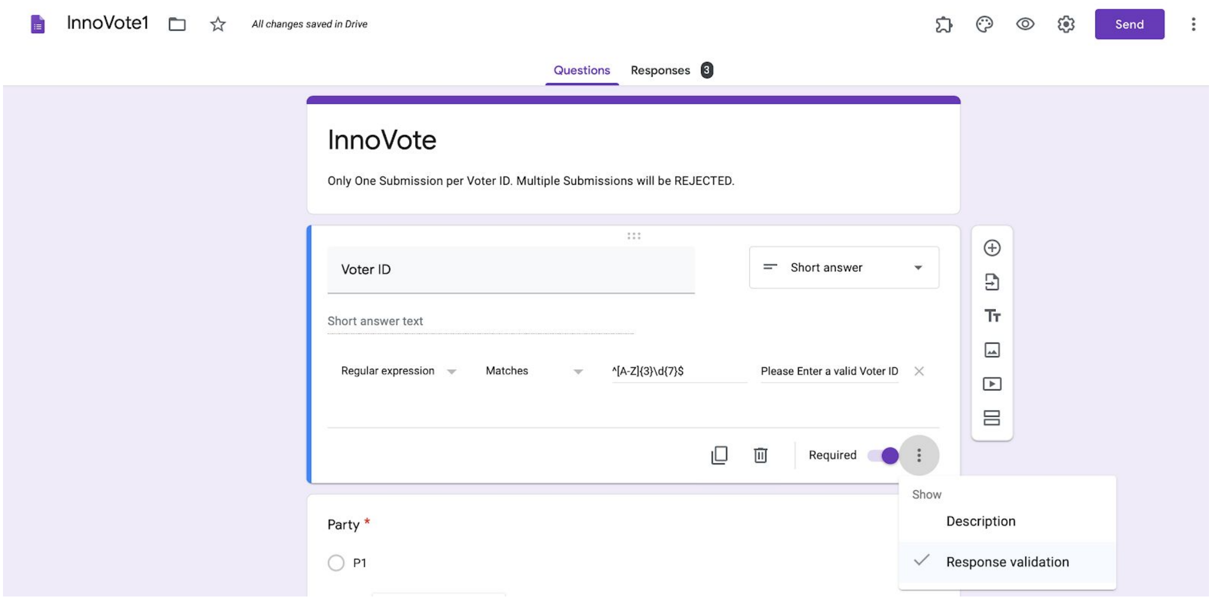
This code is to verify you are a human. Protected by xfanatical.

Your answer

Submit

Fig. 1 Sample Google Form received by voter. The symbols can be substituted for party logos

Since voter IDs begin with 3 letters followed by 7 digits, input validation can be performed on the spot so that voter responses only matching this particular sequence of characters will be accepted into the Google Sheet. This is achieved using regular expressions.


InnoVote1

All changes saved in Drive

Send

Questions

Responses 3

InnoVote

Only One Submission per Voter ID. Multiple Submissions will be REJECTED.

Voter ID

Short answer

Short answer text

Regular expression — Matches — ^[A-Z]{3}\d{7}$ — Please Enter a valid Voter ID

Required

Show

Description

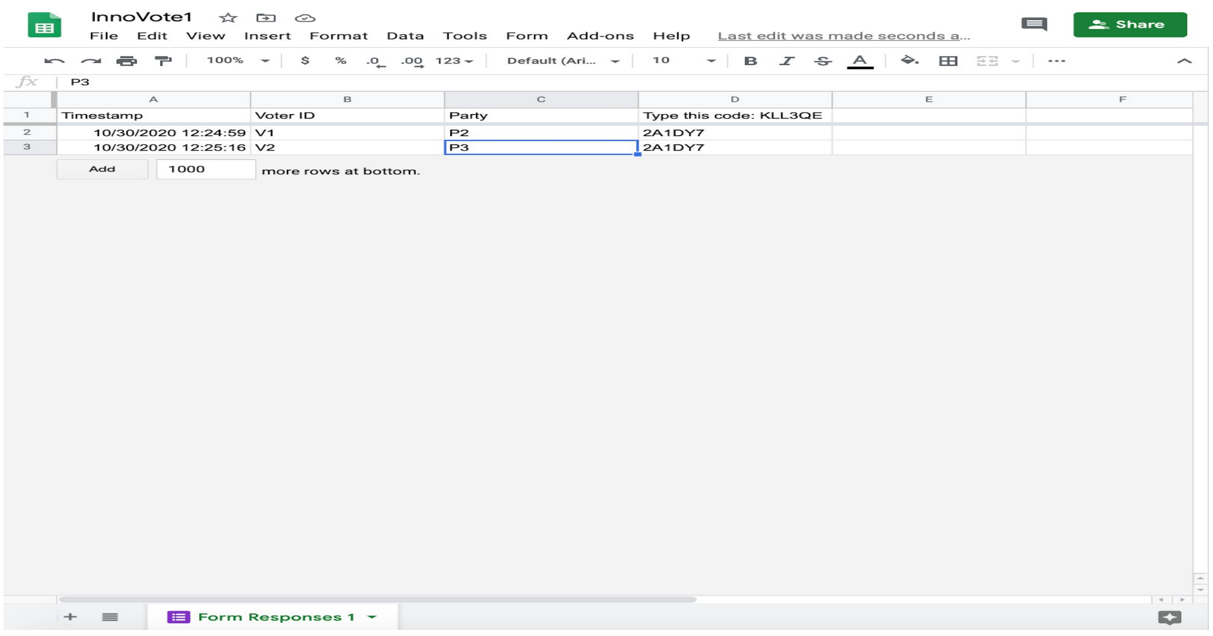
✓ Response validation

Party *

P1

ii) Google Sheets

Upon clicking the “Submit” button, the user’s vote is appended to a Google sheet, along with all the responses submitted by other voters(shown in Figure 2). This sheet is viewable only by the creator of the corresponding Google Form. While the votes will be visible in plain text, it is assumed that a single Election Commission officer will be responsible for the creation and management of the sheet. This is the only element of InnoVote which will require the aid of a human participant in order to ensure the integrity of the voting process.


InnoVote1
File Edit View Insert Format Data Tools Form Add-ons Help
Last edit was made seconds a...
Share
100%
Default (Ari...
10
Add
1000
more rows at bottom.
Form Responses 1

	A	B	C	D
1	Timestamp	Voter ID	Party	Type this code: KLL3QE
2	10/30/2020 12:24:59	V1	P2	2A1DY7
3	10/30/2020 12:25:16	V2	P3	2A1DY7

Fig. 2 Sample Google Sheet containing Form responses from voters

For a single sheet, Google has set an upper limit of 5 million cells which can be filled. With each response requiring 4 cells, this means that every sheet can accommodate roughly 1 million (10 lakh) votes. Keeping this restriction in mind, we propose that InnoVote can be deployed in the form of multiple Google form links. Voters from a particular constituency will receive an identical form with a unique URL. The form associated with this URL will not be shared to voters of other constituencies within the state. In this manner, the problem of overloaded Google sheets is circumvented. The responses stored in each sheet will be collected and verified against the central database.

iii) Google Apps Script

Publicly accessible Google Forms present the issue of duplicate entries, submitted either erroneously by the actual voter or by the actions of malicious entities. They also provide an easily accessible gateway to input a large number of invalid entries in the Sheet, resulting in a Denial of Service type attack. Fortunately, Google has the provision of making custom changes to its tools using custom or third-party scripts via Apps Script.

InnoVote makes use of CAPTCHA in order to verify whether a vote entry has been sent by a genuine user by requesting the user to enter a sequence of alphanumeric characters. The implementation has been borrowed from a blog post by an entity called xFanatical. The script can be copied into the Google form's Script Editor wherein the appropriate permissions must be granted to it. Once enabled, the sequence of alphanumeric characters changes every minute. Even if an automated bot is able to figure out the sequence of characters, the window of time to insert erroneous entries is very limited.

To deal with the problem of duplicate entries, InnoVote employs a macro in the form's corresponding Google Sheet which calls a custom script that scans the form and deletes all duplicate entries and the blank rows left after they have been removed, retaining only the first instance. This macro is called every minute with the help of an automated trigger which can be configured once within the Google Apps Script service.

However, if a bot is programmed to send unique but invalid entries, these entries persist in the form and can only be filtered out during the final verification phase using the central SQL database. But from our experiments using an automated bot programmed using Selenium WebDriver, the process of accessing and filling the form is quite cumbersome. Only a few invalid entries can be inserted before the CAPTCHA subprocess shuffles the alphanumeric sequence required to submit a vote.

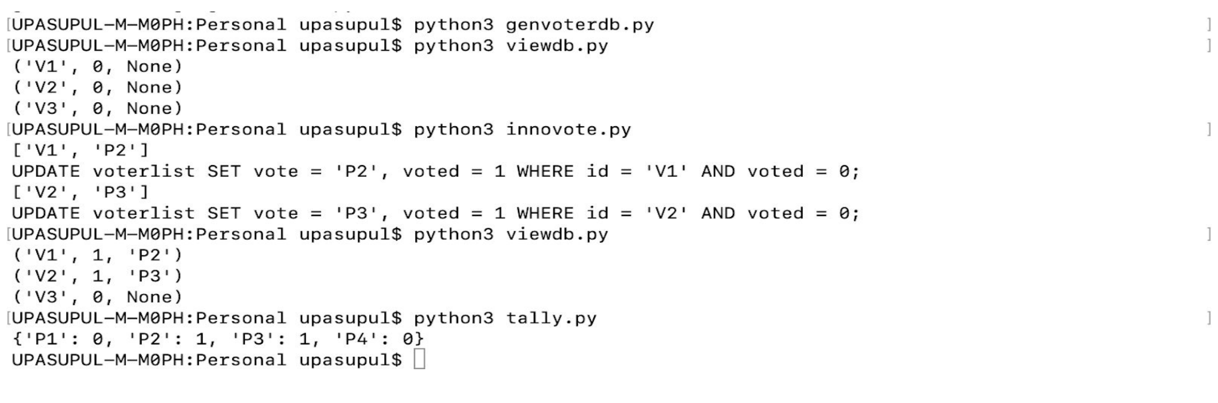
iv) Google App Developer Console and Sheets API

Once the system starts accepting votes (Google Forms) and recording voter responses (Google Sheets), a mechanism is required for programmatically accessing the voter data recorded in the Sheet and performing operations on a database. For this, we leverage the Google App Developer Console which allows the creator of the Google Sheet to share the Sheet with a unique service account. This service account provides JSON credentials which can then be applied in a Python script along with the gspread and oauth2client.service_account packages to securely access and pull data from the sheet.

The backend SQL database is implemented using sqlite3, a lightweight, SQL-esque Python package which can create a database to be stored locally on the disk. The schema of the database is as follows:

Voter ID (20 bytes)	Voted (4 byte)	Party (20 bytes)

The *Voted* data item is a boolean type flag which ensures that only one unique vote is registered in the central database. Figure 3. depicts a sample flow of operations required to pull the voter data from a Sheet (see Fig. 2), perform the corresponding operations on the SQL database and finally tally all the votes. For the sake of simplicity for our experiments, a toy database was generated with three unique voter IDs. In the real world, this would translate to three genuine registered voters. This database needs to be created by the ECI using official voter lists during an actual election.



```
UPASUPUL-M-M0PH:Personal upasupul$ python3 genvoterdb.py
UPASUPUL-M-M0PH:Personal upasupul$ python3 viewdb.py
('V1', 0, None)
('V2', 0, None)
('V3', 0, None)
UPASUPUL-M-M0PH:Personal upasupul$ python3 innovote.py
['V1', 'P2']
UPDATE voterlist SET vote = 'P2', voted = 1 WHERE id = 'V1' AND voted = 0;
['V2', 'P3']
UPDATE voterlist SET vote = 'P3', voted = 1 WHERE id = 'V2' AND voted = 0;
UPASUPUL-M-M0PH:Personal upasupul$ python3 viewdb.py
('V1', 1, 'P2')
('V2', 1, 'P3')
('V3', 0, None)
UPASUPUL-M-M0PH:Personal upasupul$ python3 tally.py
{'P1': 0, 'P2': 1, 'P3': 1, 'P4': 0}
UPASUPUL-M-M0PH:Personal upasupul$
```

Fig. 3 Backend output

At the beginning of the election, every ID's *Voted* and *Party* entry is set to '0' and 'None' respectively. After the election, once the Google Sheet data is procured using the Sheets API, the unnecessary fields (timestamp and CAPTCHA entry) are stripped and an SQL query is constructed using the remaining pieces of information. If an invalid voter ID(ex. 'V4' voting for 'P4') is entered, the SQL query will be rendered ineffective since the corresponding entry in the database would be absent. Once the modifications are made to the database, a simple script(tally.py) will scan the entire database and tally all the votes and print them in the form of a dictionary. The "innovote.py" script can be modified to collect and operate on entries from multiple sheets by changing the Sheet name argument. Since the database is central, in case a voter(after casting their genuine vote) attempts to vote using another sheet that was not sent to them, the new vote would be rejected since the *Voted* flag associated with the corresponding voter's *VoterID* will be set to 1.

Flow

Figure 4 shows each step involved in InnoVote's pipeline of operations followed by a brief explanation of each procedural stage:

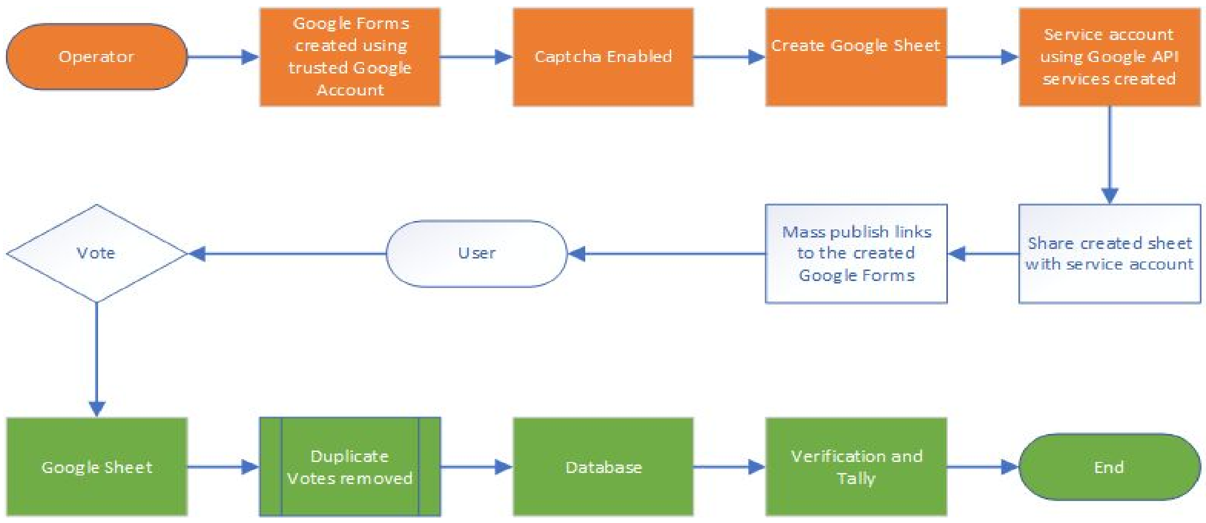

Operator
Google Forms created using trusted Google Account
Captcha Enabled
Create Google Sheet
Service account using Google API services created
Share created sheet with service account
Mass publish links to the created Google Forms
User
Vote
Google Sheet
Duplicate Votes removed
Database
Verification and Tally
End

Fig. 4 Flow Diagram of the process

1. Mass SMS containing links to created Google Forms sent to voters across the country based on regions.
2. An Operator in ECI using our solution logs into their Google Account to enable the responses to be recorded.
3. As the votes are cast, the voter ID, the party associated with the vote are extracted and stored onto the Google sheet. This data is only visible to the operator using our solution. The sheet is periodically refreshed to remove duplicate votes in real-time.
4. After all votes are cast a sqlite3 database is generated to save the pulled data from all sheets with help of a python script "innovote.py".

5. “innovote.py” pulls all the data from all the sheets and store it in the newly generated database where first the legitimacy of the voter ID’s is checked by comparing it to database of existing voters provided by the ECI, after which another script, “tally.py” is run to tally the cast votes and produce a final count.

Conclusion

In this document, we have proposed InnoVote , a simple, scalable and secure voting system aimed at solving the problem of lost votes due to physical absence of voters. However, there is a lot of room for improvement and we have evaluated our own approach using the following rubrics, outlining its strengths and the challenges that lie ahead in realizing this vision. We are confident that with the right guidance and mentorship, InnoVote can have a significant impact on future elections by taking into account all Indian voters regardless of their physical location.

i) Completeness

While the examples outlined involve a working proof of concept, we believe that this prototype takes advantage of existing simple, scalable and secure platforms and is innately ready to be deployed on the field. The Google Forms and Google Sheets(coupled with Apps Script changes and API creation) are readily usable and require minimal effort to setup. An area to be resolved is the automation of the python backend scripts and integration with Google Sheets. Currently, the scripts are run manually from a terminal after collecting all the voter responses. Consolidation of voter responses from multiple sheets across multiple constituencies into one centralized database requires a distributed setup for further experimentation. Also, for the voter ID verification phase, a centralized database needs to be created with the permission of the ECI since this requires access to sensitive voter list information.

ii) Effectiveness and Efficiency

Through limited experimentation, InnoVote has been found to be capable of handling a steady flow of incoming voter responses. InnoVote’s rotating CAPTCHA system keeps the Sheets from becoming overwhelmed by bulk responses generated by spam bots. Theoretically, it would be impossible to bring down any of the Google Services employed using any sort of Distributed Denial of Service attack due to the inherent robustness of the platform. The efficacy of the system at scale using multiple Google Sheets for across multiple constituencies requires a distributed setup for further testing. Also, a periodic duplication removal macro keeps the Sheet free from multiple entries containing the same voter ID. The centralized SQL database filters unregistered voters.

iii) Design and Usability

Usability is InnoVote’s strongest point of contention. Almost everyone who has ever accessed the internet has encountered a Google Form. Its simple, no-frills user interface provides an extremely convenient platform for the collection of voter responses. Images of party logos aid those voters who may be unable to read the listed choices. CAPTCHA allows for human-only responses. The Google Developer Console and Sheets API (along with scripts/macros) can be set up in a few minutes with a vast number of YouTube tutorials on the same. Explaining and training the ECI staff to create and manage the Google Forms requires minimal effort, while issues are easy to troubleshoot.

iv) Creativity and Innovation

We believe that InnoVote is a one-of-a-kind offering that stands on the shoulders of a technology platform known for its ease of use, ability to scale with user demand and attention to privacy. It requires no app-installation procedure from the user end, thereby protecting sensitive information stored on devices. While it may be challenging to explain solutions based on technologies on AI and blockchain, setting up InnoVote only requires a basic working knowledge of computers. Google's rich documentation provides a source for configuring and troubleshooting.

Instructions

A repo containing all the source code can be found at -

https://github.com/ujdcodr/InnoVote

1. Create a Google Form using any trusted Google Account
2. Enable Captcha on the form by copying the script from https://xfanatical.com/blog/captcha-for-forms/ into the script editor of the form
3. Copy the code from the "duprem" file in the repo to the script editor of the Google Sheet containing form responses.
4. Configure a time driven trigger on the Google Sheet's Apps Script page to call the duplicate removal macro every minute.
5. Display the form responses on a Google Sheet
6. Create a service account using GoogleAPI services. Download the JSON key produced at the end. This key must be kept in a secure location.

   (Tutorial:https://www.youtube.com/watch?v=cnPlKLEGR7E)

7. Share the Google Sheet with the service account created
8. Publish the form and begin accepting votes
9. Create an sqlite3 database containing fields for voterID, party voted for, and a boolean flag that is set to 1 once a vote is cast
10. Once the voting period has elapsed, run "innovote.py" on all the sheets created, using the JSON key from step 6.
11. Run "tally.py" to sum all the votes collected from the central database.

For detailed instructions on enabling CAPTCHA, removing duplicate entries and setting up a Google service account, refer the InnoVote Support document.