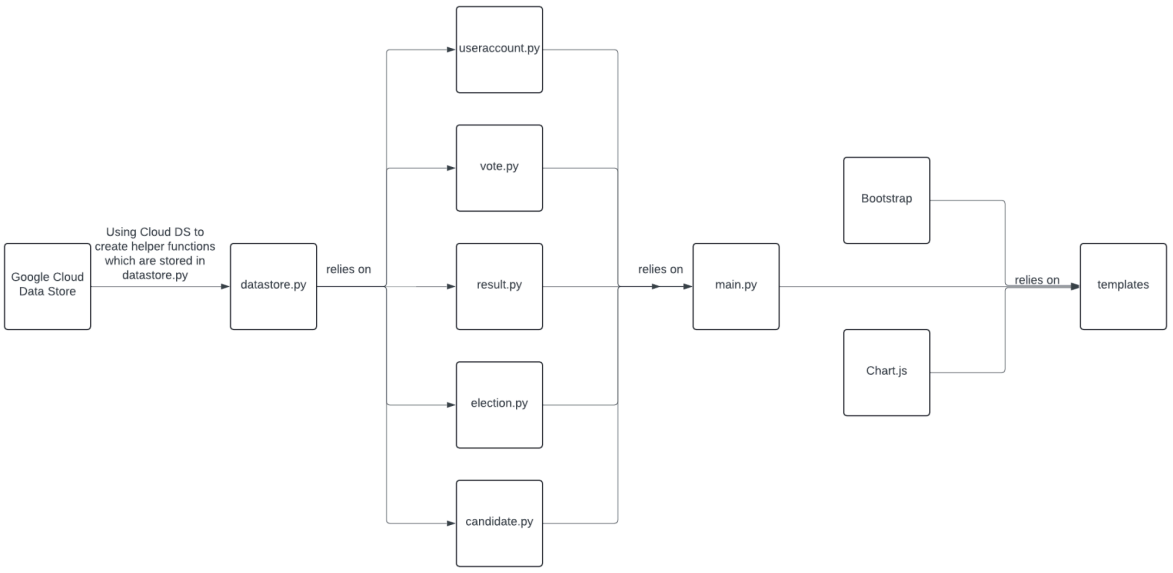
Student Voting System Report

Clearly state the project's objectives/meaningfulness/novelty:

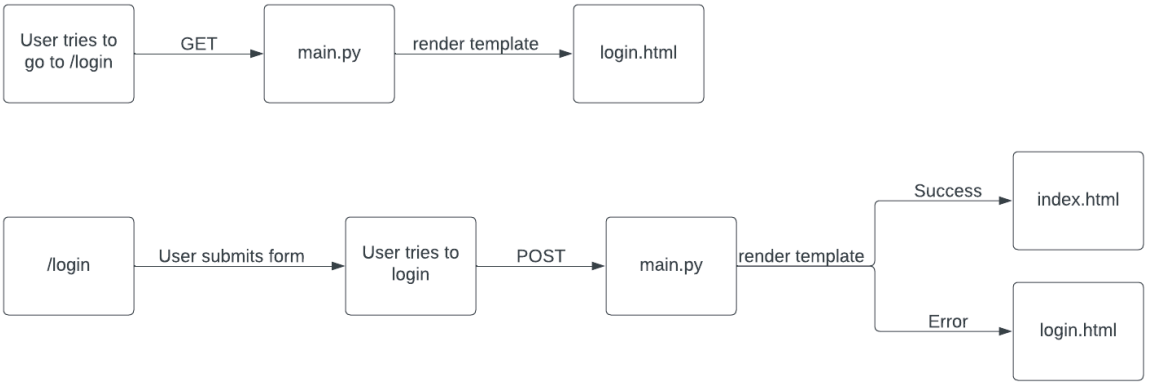
The project’s goal is to create a web platform where an admin can host Student Council President elections, the admin can create a new election if an election is not already ongoing and end an election if there is one ongoing. The objective is to streamline the election process for students and create one location for the admin to manage elections and users to run for election or vote. Students are able to sign up as voters or as Student Council members, every student that has signed up can vote (only once!) however council members have the ability to run for Student Council President (assuming that there is an election currently running). Every registered user, along with the admin, is able to also see past election results and current election statistics (if there is an election ongoing). This project is meaningful because it eases the election process for schools by providing a web platform for elections.

Code Structure + Explanation:



datastore.py uses the google cloud data store api to create database helper functions like getAll, etc. candidate.py, election.py, result.py, vote.py, and useraccount.py then use these helper functions in their respective logic in order to update the database according to changes that the user may make. Then the main.py uses these different classes (vote, election, etc.) to create the flask web app and dynamically renders each of the HTML templates based on what the user requests (GET vs POST) and updates the database accordingly. The HTML templates then show the user the content according to changes in the database and sends back user input if needed back to the server, the templates use Bootstrap for the CSS styling and Chart.js to show the graphs in both the results page and the statistics page.

Here is a more detailed view of how the template process works:



In the first diagram, the user tries to go to /login (https://student-voting-system-svs.appspot.com/login) which sends a GET request to the flask server, the server then recognizes that it is a GET request and renders the login.html template.

In the second diagram, the user is already in the login page and now tries to hit submit on the login form and tries to login to the website. As such a POST request is sent to the flask server, the server then tries to authenticate the user's entered credentials by checking it with the database, if the credentials are correct, it is a success and the user is logged in and sent

to the index page, if not, the user is not logged in and an error message is shown on the login page.

Provide clear installation and usage instructions:

Student-voting-system-svs.appspot.com

You can simply use the website link above.

The admin user id is admin@voting.com and the password is 123Vote

For local use (not recommended):

git clone https://github.com/Yash1907/Student-Voting-System.git

cd into the project directory

pip install -r ./requirements.txt

.\env_dir\Scripts\activate

python -m venv env_dir YOUR PATH TO PROJECT

gcloud beta emulators datastore start --data-dir=YOUR PATH TO PROJECT

(in a different terminal)

python main.py

Usage:

You can sign up as a Student Council member or as a regular student and vote if there is an ongoing election or run for President if there is an ongoing election (if you signed up as a council member). You can also log in as admin and start a new election/end a current one and see past stats for previous elections.

Video demo please watch:

Student Voting System DEMO (youtube.com)

Discuss project issues, limitations, and the application of course learnings:

The project can be expanded upon in many ways. In its current state it is a working election app that can only host one election at a time, which is a limitation, with minimal security as I only hash passwords currently. Another limitation that currently exists is that the Student Council Member Code that is used during sign up has no real function, it was supposed to be a way to check if the student is truly in the council, by giving each member a unique code however I did not implement it because I thought it was a bit convoluted and due to time constraints, I think that in the future I would like to have one code used by all council members that the server can use to check if the student is truly in the council. I think that in the future improvements to the security can be made, maybe a secure portal for voters, automatic sign out after x minutes of inactivity, etc. I also think that in the future I can allow for multiple elections to take place at the same time so you could create an election for Student President and an election for Student Treasurer, Secretary, Senate, etc. The UI for this already exists actually, it's just not implemented in the backend due to time constraints so it is something that I plan on making happen eventually. Additionally, I think having an email functionality for the admin where the admin could send emails to voters to get them to vote without registering.

From a course learning perspective I think that this project leverages several things that we have learned throughout this course. The project uses dicts and arrays to store its data and uses a vast number of loops and conditions to dynamically render the website based on real time user input and changes to the database.