- Make a point about syncing the db fie with a NAS or Microsoft product or something of the like
- Principle of least security with user accounts only accessing certain functions
- Flow in a logical sequence

Overview

Hi, my name is Jack Hagen, I'm from Oconomowoc high school, and this is my submission

For designing the platform I considered the needs of an organization which would use this tool. As a school district, the tool needed to be

- Easy to use and intuitive to non-technical users
- Lightweight to cut hardware costs
- And be able to run with little technical oversight and maintenance

In order to meet these needs I decided to make a web-based platform, as it's easiest for users and cross platform.

I decided to use Python for the back-end because of it's use in data science and it's simplicity.

Out of the server options available for python I chose Flask out of all the options I've tried it's the easiest to use, and has great support for creating dynamic web pages. For the database I used sqlite because it's widely used in industry, is easy to use, and is a reliable solution

My submission is a website with three main functions. A scanner page for attendants at school events to scan student IDs and mark them as present, a report page of charts analyzing student attendance, and an admin page for managing user accounts and troubleshooting problems with the website.

Because it needed to run with little oversight, there needed to be security within the basic architecture. In order to achieve this, I separated privileges into three types of user accounts with the principle of least security in mind.

- The scanner accounts' only privilege is marking students as present for the active event. It cannot read student data or mark student present for any event which isn't specified by other accounts. It's designed to be used by an attendant at a school event.
- The viewer accounts' only privilege is reading student involvement in the form of graphs. It cannot mark students as present. It's designed to be used by school administrators such as the principle or counselors.
- The admin accounts have access to every page in order to troubleshoot problems with the website. In addition to the scanner and report pages, the admin account also has access to the admin page, where they can create new users, delete users, change account passwords, and view a log of the site.

Technicals

For the back-end I used Flask with jinja as a templating engine. Jinja allows me to pass python variables to the html cleanly, allowing me to load parts of the database, data for charts, and to send success or error messages.

The server is hosted in the main.py file. Flask allows you to define responses for requests. For each path there is a return of an html file, along with defining variables for the request and what the page requires. For example, in the '/events' page, if the request is GET, it returns the html with data defining events in the database as well as the active event. If the request was POST, it would return the data included in a GET request as well as a single variable telling jinja to throw a success message. In order to use the POST/Redirect/GET pattern, there are some global variables for each path defining the success messages. The POST/Redirect/GET pattern is important because all of the website's controls rely on the POST request. But when a client sends a POST request, the browser throws a message asking if the user want to resubmit data, which makes it more complicated for non technical users, and less graceful overall.

The database.db file hosts all authentication, details, and attendance data. Because it's stored in a file, it's easy to backup the database at any time with a simple cron job on Linux, a syncing service, or a custom programmed implementation for sending to an outside device. SQL also has options to use differential or incremental backups. The authentication data is stored in the auth table. The passwords are in plaintext because I didn't have time to implement hashing properly. If I had more time I would have the client hash the inputted password and have it stored on the database in with sha256. The school_details table hosts details about the school. Right now it just hosts how many students are in the school for calculating attendance in the /report page, but in future releases it could include information such as how many students are in each grade. The event_list table contains information about school events which are tracked using involve, The IDs table is a table with every student ID, name, grade, and points. For each event created a table is created based on the ID of the event. The unique ID is the date that it was created plus a number which increments by one for every event on that date. Each table created for an event is just for listing the IDs tracked by a scanner at an event.

Flask uses a utility called 'session' which allows a server to manage cookies on a client. It is used to keep a user logged in as they browse. The cookies stored on the client are encrypted with the server's secret key, which prevents a user from editing their privileges. There are three cookies stored on the client. Username, password, and type. The type cookie is used to easily verify a user's privileges and is used to automatically redirect user to their assigned page if they somehow navigated to a page they're not allowed access to.

Demonstration

You each have one computer in front of you. Not that while these are stock Chromebooks, each are capable of running the website with little to no difference in performance. One computer is logged in with a viewer account, one is logged in as a scanner account, and one is logged in as an admin account. They are all connected to my laptop over my phone’s hotspot. I’m running the server on my laptop, and notice how fast it runs despite being on a very old laptop.

1. If we start at the viewer computer we can navigate to the events page to create a new event and mark it as the active event.
2. Then if we go to the scanner computer we can scan a few of these model ID cards I made. The scanner account is only limited by the equipment that it uses for scanning, so theoretically you could use an RFID reader or a QR code reader and achieve the exact same effect throughout the entire website, as long as the students are registered with the same data.
3. If we go to the viewer computer, you can see the statistics for all of the registered events and we can sort to the one we just created. And we can see how it compares to other events.
4. Now if we go to the winners page we can see a list of all of the students and their points. We can search for this student and see how many points they have. We can also pick random winners.

Any questions?