Spiel

In order to create a platform for tracking student involvement for school activities, I had to consider the needs of a school district. Namely being cost effective, easy to use, intuitive for non technical users, and able to run with little technical oversight.

To meet those goals I decided to build a web-based platform with Python and Flask for the back-end, Sqlite for the database, and Jinja for the templating engine.

Using Sqlite was the best option because of its built-in support in Python and it's capability for differential and incremental backups.

Jinja is a module included with Flask which allows me to pass variables to the client. Combined with Python's advantage in Data Science, Flask allows me to dynamically show insights from the database on the client.

My submission has four main functions operated by three main account types. Because the site needed to run with little technical oversight, I incorporated the principle of least privilege into the account design. The three types of accounts are ***show three accounts*** the admin accounts, viewer accounts, and scanner accounts. Each account type only has access to certain functions within the site.

- The scanner account is used by the attendants at school events, and is used to scan students' ID cards as they enter the event. It can only scan students into an event and has no other control.
- The viewer account is used by school staff such as the principle to analyze student attendance data. It has access to the viewer and winner pages and nothing more.
- The admin account is used by IT staff, or whoever is managing the site. It can add accounts, change passwords, set backup preferences, and view system logs.

Show main.py data analytics section These are the methods which call up data from the database, sort it, and return it to functions which show it on the site.

Show different paths on main.py These are the defined paths for each page. Every allowed request method (IE POST and GET) are defined in the header. Only paths which are defined here are allowed on the site. If a user tries to navigate to database.db for example, they are returned a 404 error. This, as well as other Flask security features such as SQL injection filtering and cross site scripting sanitation mitigates common security flaws.

Flask uses a utility called sessions, which allows me to easily and securely manipulate cookies on the client device. Every authenticated user has three cookies stored on their device. The username, password, and account type. The cookies are encrypted with the server's secret key, which prevents the user from manipulating their data and attackers from reading passwords stored on client devices.

Demonstration

- Viewer – Create event, mark it as active
- Admin – Create scanner account (show success message)
- Scanner – Scan IDs
- Viewer – View data
- Viewer – Get winners
- Viewer – Help page