Queue Division Automation

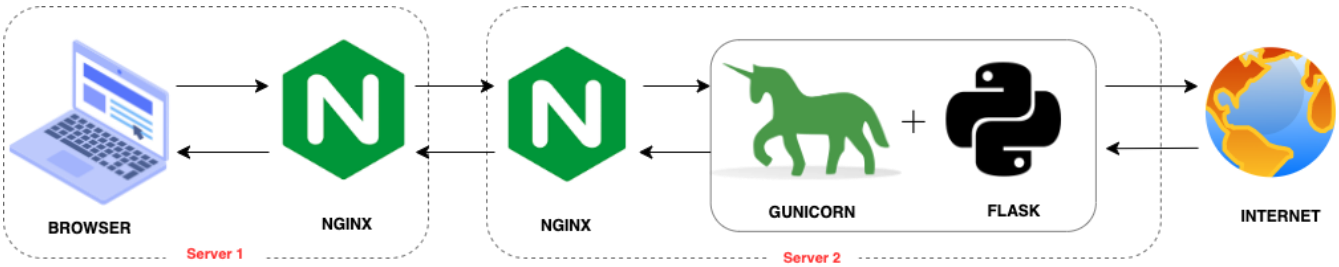
Summary:

Our Support team operates on a shift basis, with the number of engineers working each day varying. The workday extends for 13 hours, and engineers have the option to choose their shift, whether it be in the morning, afternoon, or evening. Previously, the division of the General Queue, which receives all customer tickets, among the engineers was done manually using an excel file. This process was time-consuming and prone to errors. To address this issue, I developed a web application that automates the process of generating the queue division based on the schedules and number of the engineers. The application presents the queue division in a clear and organized manner, and also allows for easy adjustment if an engineer is unable to work. Additionally, the queue division can be easily sent to Slack with the press of a button, eliminating the need for any manual work.

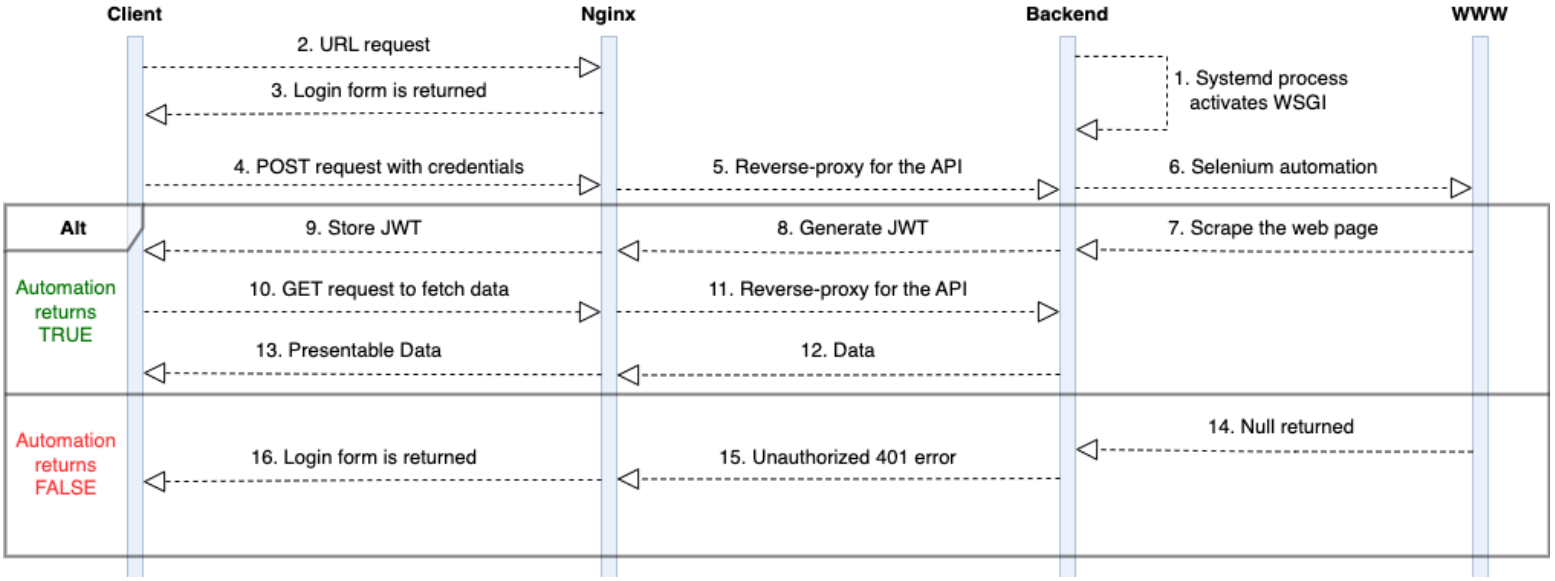
Technologies:

1. **Frontend:** React
2. **Backend:** Python (Flask, Selenium, bs4, etc.) + Gunicorn (WSGI)
3. **Intermediate Servers:** Nginx

Architecture:



User Flow:



1. Since this app runs on a production server systemd ensures the WSGI server is always running.
2. Client requests the URL of the web application.
3. Login form is returned to the client (static file that is served by Nginx).
4. The client enters his/her credentials and submit the form. A POST request is sent to authenticate the client.
5. Nginx that serves as a reverse-proxy for the API sends this request to the server that holds the BE endpoints. The Nginx server points to the address the WSGI server is listening to.
6. Once the request reaches the */login* API endpoint, the Selenium webdriver performs certain automation process.

Automation returns TRUE:

7. Selenium webdriver scrapes the content of the web page and stores it.
8. JWT token is generated (token expiries after 15 min).
9. JWT token is sent back to the client and stored in Local Storage.
10. As soon as the JWT token is stored a GET request is sent to perform the calculation and fetch the data.
11. Nginx serves as a reverse-proxy.
12. The requested data is returned.
13. The requested data is presented to the client.

Automation returns FALSE:

14. Selenium webdriver returns “null”.
15. The backend returns 401 error – unauthorized.
16. The login page is presented to the client with the error title.