Data Service Dashboard for Virtual Labs

AIM

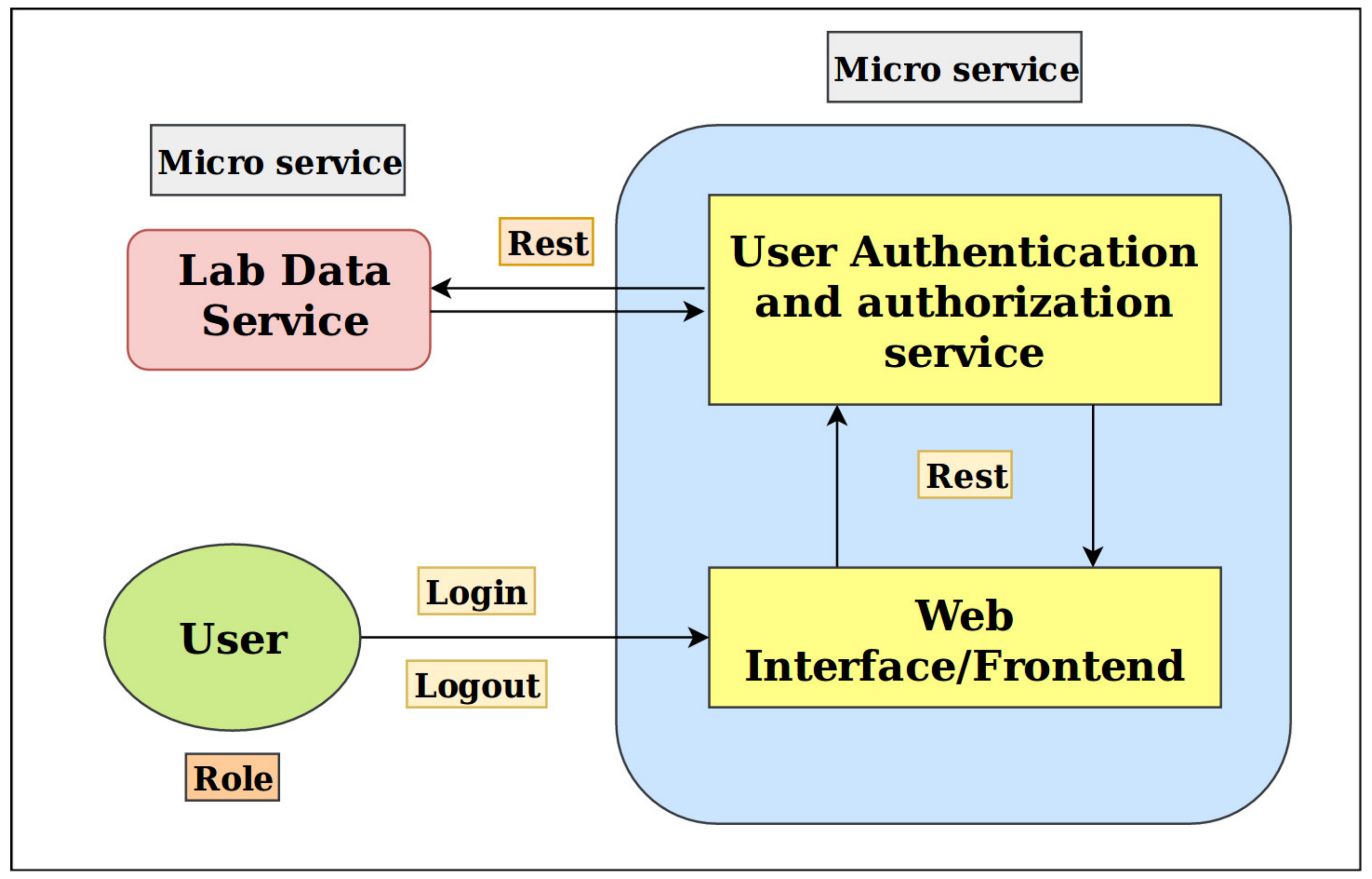
The project's aim is to design a user-friendly dashboard for LDS (Lab Data Service). Our motivation is to minimise the user's efforts in accessing and manipulating data of all the virtual labs. We are developing it as an SPA (Single Page Application) using plain JavaScript without any frameworks.

INTRODUCTION

- Lab data microservice helps to get the meta data of all the virtual labs. All the other micro services can view and manipulate this data. Our dashboard allows the user to perform CRUD (Create, Read, Update and Delete) operations on this data.
- The most notable difference between a regular website and an SPA, like our dashboard is user experience since SPAs have a heavier usage of AJAX - a way to communicate with back-end servers without doing a full page refresh.
- As a result, the process of rendering pages happens mostly on the client-side and navigation depends on a router which tweaks the content of address bar and notifies rest of the system of the URL changes without the page reload.

APPROACH

- For logging in to the dashboard, google credentials of the user are required since authentication is done using Google OAuth service.
- Roles and Session management has been implemented using Python Flask-Login.
- Authorization is managed using function decorators present in Python.
- We have dealt with JSON specification for rendering the view rather than template rendering.
- Hash route in the URL decides which section of the view is to be displayed on the browser screen at that moment.
- In the future ,if new roles are to be added with new access rights, automation of the JSON spec generation can be done.
- Interaction between front-end and back-end uses AJAX.
- The user makes a request using data service dashboard and then our back-end interacts with the LDS microservice to perform the said operation.



RESULTS

- We could develop an SPA for the dashboard using plain JavaScript.
- We were able to decode JSON specification efficiently allowing us to render the view based on role.

CONCLUSIONS

- We have developed data service dashboard using plain JavaScript without the help of any frameworks.
- Absence of frameworks allows manipulation of code to incorporate any features within the limit of the language.
- JSON has a great potential when it comes to automating procedures like generation of html code.
- Hash routing allows us to embed the exact state of the web app in the page URL.
- Attaching element to an id instead of doing the opposite can allow unit-testing of DOM elements.

Software Engg Research Center

Interns: Utkarsh, Yahnit
Mentor: Madhavi