Cloud Opt

Intel-Granulate home assignment

Author: Adva Apelboim

Date: 29/8/2023

Table of contents

Introduction

This project was assigned to me as a home assignment by Granulate, an Intel company, for a student intern position on the cloud optimization team. In this document, I will showcase the project's main design and architecture, evaluate the time it took to fully implement the requirements, and suggest some improvements that I believe are necessary for the project's growth. Additionally, I will provide the project's instructions, outlining how to operate it, the limits of its use, and key points to consider while using the app.

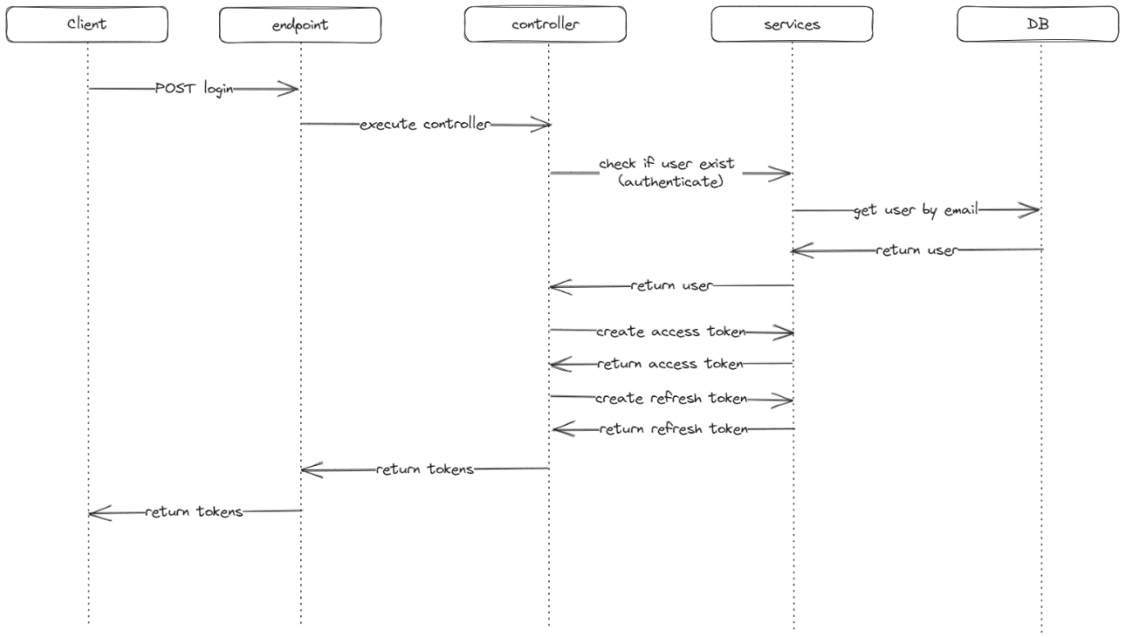
Technology in use

For the client side of the project, I used **React.js** in conjunction with **ChakraUI** for streamlined UI creation. On the server side, I employed Python's **FastAPI** to create the REST API portion of the project and used **MongoDB** as the primary database. For testing, I utilized the **pytest** and **unittest** Python libraries. One important note: I did not use the actual Intel API; instead, I mocked it as a service for this project.

Main architecture

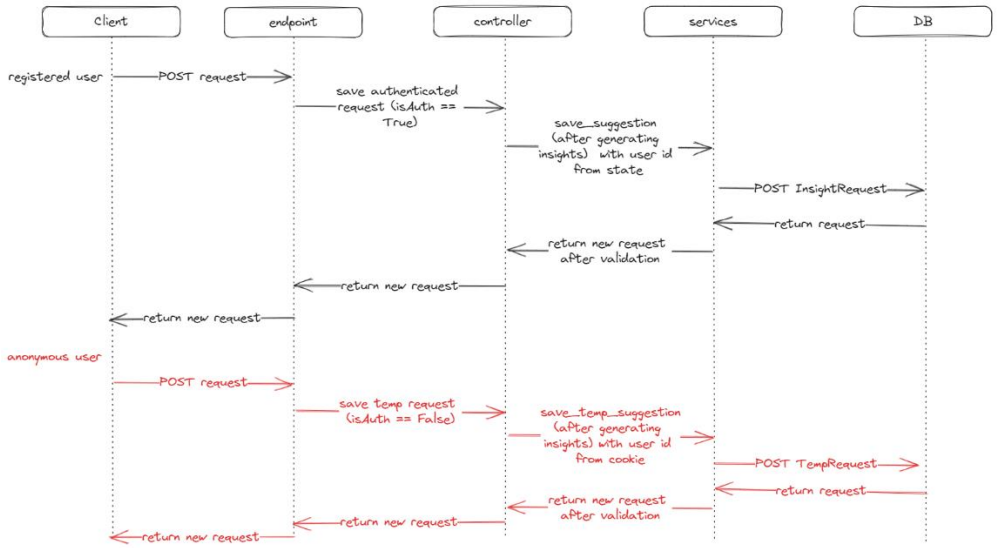
1. Services:
    a. Login service

        Using refresh and access tokens for improved authentication



b. Insights Analysis and storage

There are two types of requests – InsightRequest, made by authenticated user, and TempRequest, made by anonymous user. I made two different database collections for easier future restriction usage on the two databases.



Analyzing the inserted data (mocked, it this applications case), is done by a dedicated function, that calculates a few key attributes that can be important for the user:

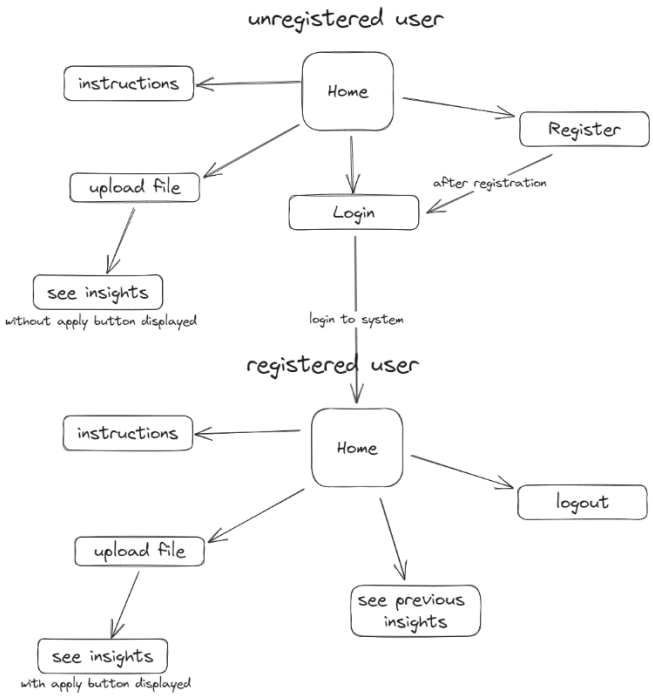
1) Pricing – overall/per vm per month
2) Instance Sizing – changes to be made if any.
3) Max Instance size - changes to be made if any.

c. User management

I have identified two types of users – anonymous user, and a registered one. The anonymous user can upload a JSON file and see the results (after it is saved to the database), but is unable to see his previous uploads insights, or have an option to apply the optimization suggestions. The registered user is also able to upload files and see the results, and apply the changes and see previous uploads insights by upload date. The anonymous user is identified by a cookie attached when opening the app for the first time, while the registered user is identified from the authentication state (JWT context)

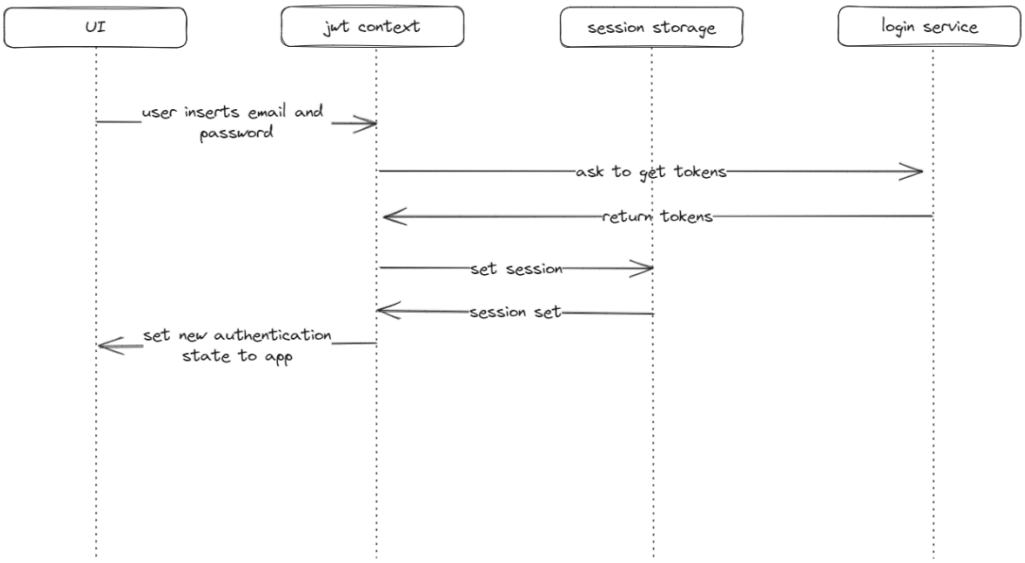
2. Client

a. User flow



b. Authorization tools

Login frontend flow



Time caption

Summary:

Portion	Estimation	Actual time
Python Learning curve	20h	25h
UI and Authentication	10h	12h
Server and API	10h	12h
Testing	3h	6h
Total	**43h**	**55h**

Extension:

1. **Frontend** – Considering I had never implemented a full authentication system on the frontend before, I believe I managed my time well. I fulfilled all the requirements outlined in the project document, including creating an instructions page and ensuring compatibility with both desktop and mobile devices. Opting for ChakraUI was a good choice; it saved me time that would otherwise have been spent crafting individual components manually. I also allocated time to make the interface as user-friendly as possible, incorporating features like loaders for an overall satisfying user experience.

2. **Server and API** – fastApi was a very good and easy tool to use, as it helped me make a well structures restfullAPI around the time I have estimated.

3. **Python Learning Curve** – I underestimated the time it would take to familiarize myself with Python. Initially, 20 hours seemed like a reasonable time to learn a new language, but I ended up spending a considerable portion of my project time understanding Python's handling of functions, classes, types, and more. Writing a web project in Python was a new experience, and although it took longer than expected, I was thrilled to learn it through the course of this project.

4. **Testing** – Understanding pytest presented a challenge and designing tests—something I've only done once or twice before—took up about half the time I allocated for testing. Nonetheless, I tested all the business logic of the application through the controllers and passed all tests.

5. **Design** – The design aspect isn't delineated separately for a reason. When I embark on a project, I prefer to plan as I go along. This approach allows me to see how my application evolves, making it easier for me to modify the design accordingly. Therefore, the design process was integrated into the overall project development.

Suggestions

As I was progressing with the project making, I did notice some areas where the project can improve. Some I have implemented, and some I thought were out of scope at that point. I'll go through them briefly:

1. Separating user types and request types – there was a requirement to do so, but I did implement it both in the authentication logic and the database structure. I thought it would be useful in the future, when we will want to monitor the user traffic in file uploading manner and have a better sense of knowing what kind of users use our app, and what kind of files (cloud infrastructure) they upload.

2. User types and dashboard– I believe that there is a need for a full admin dashboard for this app, to get a good understanding of real time traffic, insight appliances (made by registered users), and maybe even for subscriptions and grouping management. Different user types will be able to make different operations, have different appliance options for insights and more.

Instructions & important notes

1. **Login and Register**– you can log in the app using a test user:

   Email: test@user.com

   Password: 12345678910

   This user already has existing insights in the database to be displayed in the "my insights" page. You could Also register a new user.

2. **File uploading** – you can upload a valid JSON file to get the insights. I was using mock data for both the input (file generated from the script) and output (a JSON file with example suggestions). In the database, a user can upload a file only once per file id (generated in the script). So, to test the app with the mock data I have used, you'll have to change the id field, both for registered users and authenticated ones.

3. **Insights output display** – You can see all the previous insights generated for the user int "My insights" page. They are sorted and displayed by date. When you click on a button, you'll get a full display of this uploaded file insights. Every button shown below is a single vm instance. Clicking on it will show the suggested changes to be made to it.
   Something to note – I had only one output mock file. In real usage, each one will have a unique id. Since this is a mock, there is no unique id for each output, and I did not handle it for now, because it is not necessary for real app usage.

4. **Instructions page** – displays the script running instructions to the user. There is a link to the script itself (needs an intel user – I did not handle it), and to the upload page.