ATLAN Back-End Challenge

A. Design Specifications:

“dependencies”: {

“gcp”,

“python”: “^3.0.0”,

“pandas”: “^1.3.4”,

“mysql”: “^8.0.0”,

“fast-csv”: “^4.3.6”

}

B. Approaches/Ideas:

Task 1- One of our clients wanted to search for slangs (in local language) for an answer to a text question on the basis of cities (which was the answer to a different MCQ (question).

1. **Database Approach:**
   We can create a database using SQL or Firebase or use JSON file as a database as well to store and map the slangs of various languages.
   Later-on, we can compare the user entry from the already curated database of slangs and use that as a basis for our query to retrieve data from the main database or form.

```
{
    "slang":
    [
        {
            "language": "English",
            "slng": "Snog || snog"
        },
        {
            "language": "English",
            "slng": "Pissed || pissed"
        },
        {
            "language": "English",
            "slng": "Cheerio || cheerio"
        },
        {
            "language": "English",
            "slng": "Dishy || dishy"
        },
        {
            "language": "Hindi",
            "slng": "Yaar || yaar"
        },
```

Fig 1.1

2. **Translation API Approach:**
   We can also use translator APIs like Google Translate, Microsoft Text Translate, Linguatools Translate, IBM Watson Language Translator, etc.
   Among the plethora of choices out there, Google Translate provides the most accurate and most consistent result.
   We can use this API to translate the user input and check for the presence of slang in it and use it a as the parameter for the queries.

Task 2- A market research agency wanted to validate responses coming in against a set of business rules (example, monthly savings cannot be more than monthly income) and send the response back to the data collector to fix it when the rules generate a flag.

1. **Middleware Approach:**
   In this, the use of a simple relational database is prominent as the middle agent application stores data to the database and then checks for any inconsistency in the data based on the field values.

2. **Direct Validation Approach:**
   The data that is being collected through the HTML form can be verified immediately when the user hits the submit button.
   The script checks for all potential errors and if it finds any, it immediately pops an alert notifying the user of the error.

```
jQuery('#dataform').on('submit', function check(){
        var c_id=document.getElementById('cid').value;
        var c_email=document.getElementById('cem').value;
        var c_name=document.getElementById('cnme').value;
        var c_income=document.getElementById('cinc').value;
        var c_saving=document.getElementById('csav').value;
        var c_contact=document.getElementById('ccont').value;


        if(c_income<c_saving){
            alert('income can not be less than savings');
        }
        if((c_contact.length())<10){
            alert('contact number can not be less than 10 digits');
        }
    });
```

Fig 2.2

Task 3- A very common need for organizations is wanting all their data onto Google Sheets, wherein they could connect their CRM, and also generate graphs and charts offered by Sheets out of the box. In such cases, each response to the form becomes a row in the sheet, and questions in the form become columns.

1. **Middleware Approach:**
   Creation of a middle agent that fetches data from form and the database and then stores it on a google sheet is one method to implement this task.

```sql
CREATE DATABASE atlan;

CREATE TABLE client_income_data(
    client_id SERIAL PRIMARY KEY,
    client_email VARCHAR(255),
    client_name VARCHAR(255),
    income_per_annum FLOAT,
    savings_per_annum FLOAT,
    mobile_number VARCHAR(15)
);
```

Fig 3.1

2. **Sheets Integration with Form:**
   Another method to implement a Google-Sheets solution is to directly take the form data entered by the user and send it to google-sheet in a more efficient and fast manner.

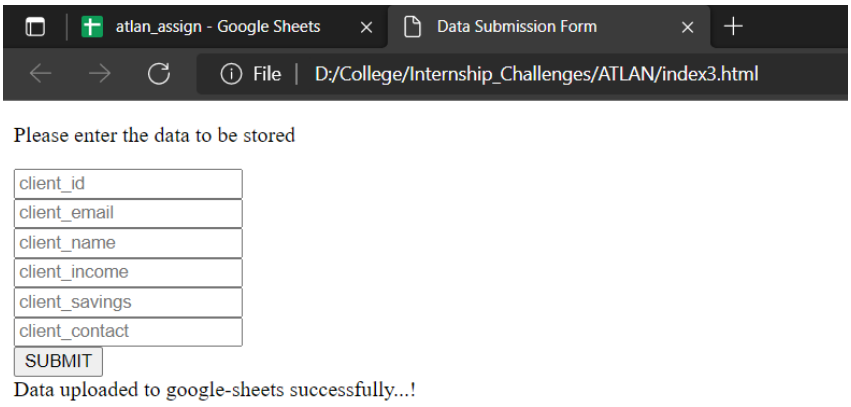

Fig 3.2

.

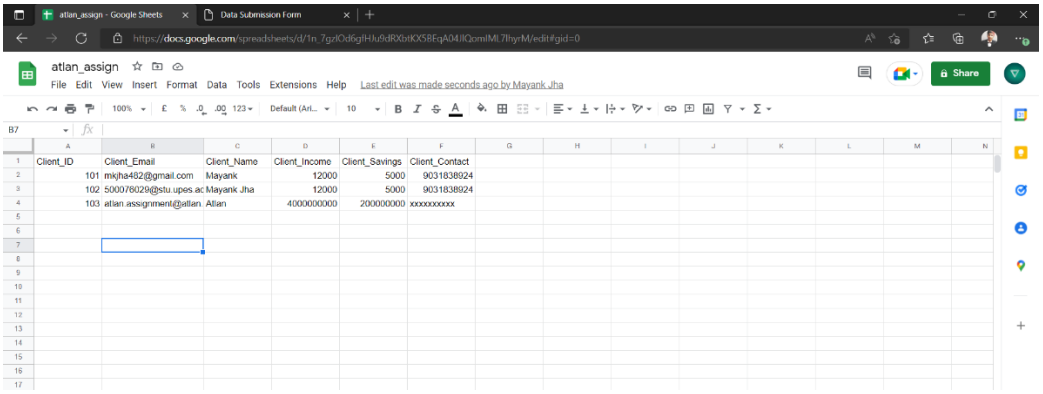

Fig 3.3

The integration of sheets to the form enabled a faster data transaction between the html form and the designated google sheet.

I used python to fetch the data stored all at once as a data frame and stored as a ‘.csv’ file.

The selection of the approach to a task was based purely on the efficiency and ease of an approach. One that can be implemented with minimal changes and does not require an overhaul of the entire framework.