

Backend Challenge By Atlan:



Solutions:

By: Vishant Shah

1. 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)

Approach 1:

- We can create API where all the words are marked to the slangs of the respective languages
- By calling the API during the search one can search in their local language and response can be get through the API

```
function matchSlang(req, next){
  console.log(req.query)
  const text = await translate(req.query.word, req.query.lang)
  //e.g req.query.lang = 'hello'(French)
  res.send(text);
  //it will send: bonjour
}
//if its invalid
catch(error){
  console.log("invalid");
}
next();
};
```

Approch 2:

- We can create a database where we can map the word entered by user to its respective slangs or languages.
- Here there will be huge no of records to store which can slow down the response process.

Which Approach Is **Efficient**?

- It can reduce the fetching time compared to database approach and also no more setup is required, just call the API, fetch the data and output it.

Task-2: A market research agency wanted to validate responses coming in against a set of business rules (eg. 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

Approach:

- In validation it should cover following things:

- Client ID
- Valid Client Email
- Valid Client Mobile No
- Valid annual income as mentioned

- Create a simple Database

```
CREATE DATABASE atlan;

CREATE TABLE client_income_data(
  client_id SERIAL PRIMARY KEY,
  client_email VARCHAR(100),
  client_name VARCHAR(100),
  income_per annum FLOAT,
  yearly_savings FLOAT,
  mobile_number VARCHAR(15)
);
```

- Validation Function while Inserting the data:

```
function validateData(req, res, next) {
  const { income_per annum, mobile_number } = req.body;
  if (income_per annum < yearly_savings) {
    res.send("Invalid Data");
  }
  else if (isNaN(mobile_number)) {
    res.send("Enter Mobile Number Properly");
  }
  else if (mobile_number.length !== 10) {
    res.send("Invalid mobile number, should be of 10 digits");
  }
  next();
};
```

- To validate all responses

```
app.get('/validate', async(req, res) => {
  try{
    let invalidData = await pool.query("SELECT * FROM client_income_data WHERE yearly_savings > income_per_annum")
    invalidData = invalidData.rows;
    if(invalidData.length === 0)
    {
      res.send("Records are Valid");
    }
    else {
      res.send(invalidData);
    }
  } catch (err) {
    console.log(err.message);
  }
});
```

Task 3: 1. 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.

Approach:

- We can use Google sheets API to fulfill the mentioned task.
- Google Sheets offers a really straightforward means to access data held in a Google Sheet via a special script call appended to a Sheet's *published* URL. That is, the Google Sheet must be made public to be able to call it.
- The Google Sheets Reader allows you to simply call a function, pass in some options (including the Sheet id and an API key) and run a callback function with the returned, processed, formatted results.

GitHub - bpk68/g-sheets-api: Utility package that will help you fetch, read and process data from a Google Sheet

Hello and welcome to this tiny (but hopefully mighty) utility package that will help you fetch, read and process data from a Google Sheet without the faff of having to deal with the Google Sheets API. Note: v2.0.0 of this package introduced breaking changes and a significant rewrite of the internals of the package.

 <https://github.com/bpk68/g-sheets-api#readme>

bpk68/g-sheets-api

Utility package that will help you fetch, read and process data from a Google Sheet

 7  231  114  26

- Procedure(broadly mentioned in github repo mentioned in above link)
 - Install the package
 - Set up a Google Sheet
 - Generate a Google Sheets API key
 - Call the reader function


Task 4: A recent client partner wanted us to send an SMS to the customer whose details are collected in the response as soon as the ingestion was complete reliably. The content of the SMS consists of details of the customer, which were a part of the answers in the response. This customer was supposed to use this as a “receipt” for them having participated in the exercise

Approach:

- We can use Twilio API with Node.js to send SMS to the clients.

Communication APIs for SMS, Voice, Video & Authentication | Twilio

Twilio Customer Engagement Platform Twilio powers personalized interactions and trusted global communications to connect you with customers. Did you attend SIGNAL? Share your thoughts on SIGNAL 2022-it only takes a minute! Or re-watch your favorite sessions on-demand in your account today!

 <https://www.twilio.com/>



Steps:

- Install Twilio

```
npm install twilio
```

- Open up a Node session by typing `node` in your terminal and require the `twilio` library.

```
var client = new twilio('TWILIO_ACCOUNT_SID', 'TWILIO_AUTH_TOKEN');
```

- With these you can now send a text message by calling `client.sendMessage()`

```
client.messages.create({
  to: 'mobile_number',
  from: 'YOUR_TWILIO_NUMBER',
  body: 'Your Details :\n
      Email ID :${client_email}\n
      Name : ${client_name}\n
      Income Per Annum: ${income_per_annum}\n
      Savings Per Annum: ${yearly_savings}\n
      Contact : ${mobile_number}\n
      Thankyou For Connecting With Us'
})
catch (err) {
  res.send("Failed to send SMS to the Client");
}
```

Thank You

Thank you for taking the time to review my project. I appreciate your feedback and will use it to improve my work in the future. Thank you again for your time and consideration.

Sincerely,
Vishant Shah