Name: Ridham Bhagat
Mobile: +919999708076
Email: rb605@snu.edu.in
ridhambhagat@gmail.com

Design Specification
Tech Stack Used:
Backend: NodeJS
Frontend: Angular
Choice of database: MongoDB

Actions in the application:

- Create forms
- Enter data in forms
- Generate sheets from the form responses
- Send sms based on successful entry
- List slangs in cities

Reason of choosing MongoDB over other choices like SQL:

- There won't be much updations happening
- There is a finite set number of queries that will be performed
- The data has no definite structure, implementing such unstructured data wil SQL would cause an additional overhead trying to implement normalization alongside.

Each use case must be just plugged in
Solution: Implement a plugin system
There are a few components that will make up the core feature of our application i.e. form creation and deletion. We need a way to build upon that. For that a simple plugin system has been devised.
Implementation:
All the code for the plugin process goes in a separate folder and all the plugins are listed in a json file. Then a middleware is called which calls the necessary plugins that should be executed when a certain action is performed.

Setting up logs
Solution: For logging, sentry has been used. It provides metrics for individual processes like time taken for it to run and error logs.

Latency

For that, sentry will provide us with metrics where the process takes more than a threshold time so we can pinpoint where the application is failing.
As for solving it from my end, I have tried to use async coding without the await, for instance the process of generating google sheets so that other tasks can be carried out.

Mitigating limitations of third party Services
Google sheets have limits on the number of reads and writes that can be solved by batching. Batching can also contribute to making the system **failsafe**.
For instance, the requests can be batched in a cache which can be backed up in a postgreSQL database. So in case of failure, we have the data , say form responses in a persistent storage.

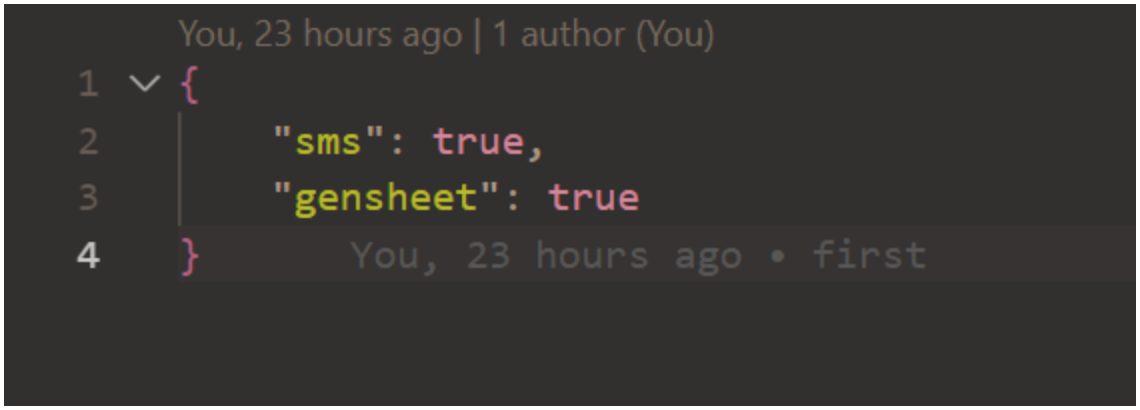
Task concerning market research agency
In this particular case, I have assumed that the market research agency would have a webhook for validating response, a placeholder plugin for the same can be found in /backend/plugin/webhook.js. It can be integrated with the plugin system implemented as per requirement.

Structure of the program
key/ : key files for google sheets.
middleware/ : middlewares required for the program
plugins/ : plugin files
routes/ : routes for the application
utils/ : schema files for the mongoose odm

Reasons for using Sentry
Sentry provides information as to how much time a process takes, logs the errors and other necessary information as well.
This provides for a convenient interface to understand the possible scopes of improvement in an application.

Implementation of plugins
Step 1: Make a file with your plugin code in the js file
Step 2: add your entry in the **plugin.json** file something like this :

Step 3: Update **plugins.json** to specify if the plugin requires an input something like this.



Step 4:Import the file in **middleware/loadplugins.js** to get a dictionary having a list of all plugins which can be called using something like **module_dict[“plugin_name”]()**.

An example of it can be seen in **addFormData** function **middleware/addformstruct.js.**

In my program, I have implemented 2 plugins for demonstration purposes namely:

- Sms: to send sms on successful completion
- Gensheet: to generate google sheet based on user response.

In the context of the submitted application , these plugins are called when the user inputs data in a form