Google Sheets API Limitation: Sheets API has per-minute quotas, and they're refilled every minute. For example, there's a read request limit of 300 per minute per project. If your app sends 350 requests in one minute, the additional 50 requests exceed the quota and generates a 429: Too many requests HTTP status code response.

Solutions: The solution is very simple, while exporting the data to google sheets, this limitation can be handled by giving the endpoint a wait time of 10 seconds when reaching 300 hits. We can define a function such that if a 429 (Too Many Requests) error occurs, the function waits for the specified duration (extracted from the retry-after header) and then retries the API call. This should help you stay within the rate-limiting constraints of the Google Sheets API.

NOTE: The code has been written to be failsafe

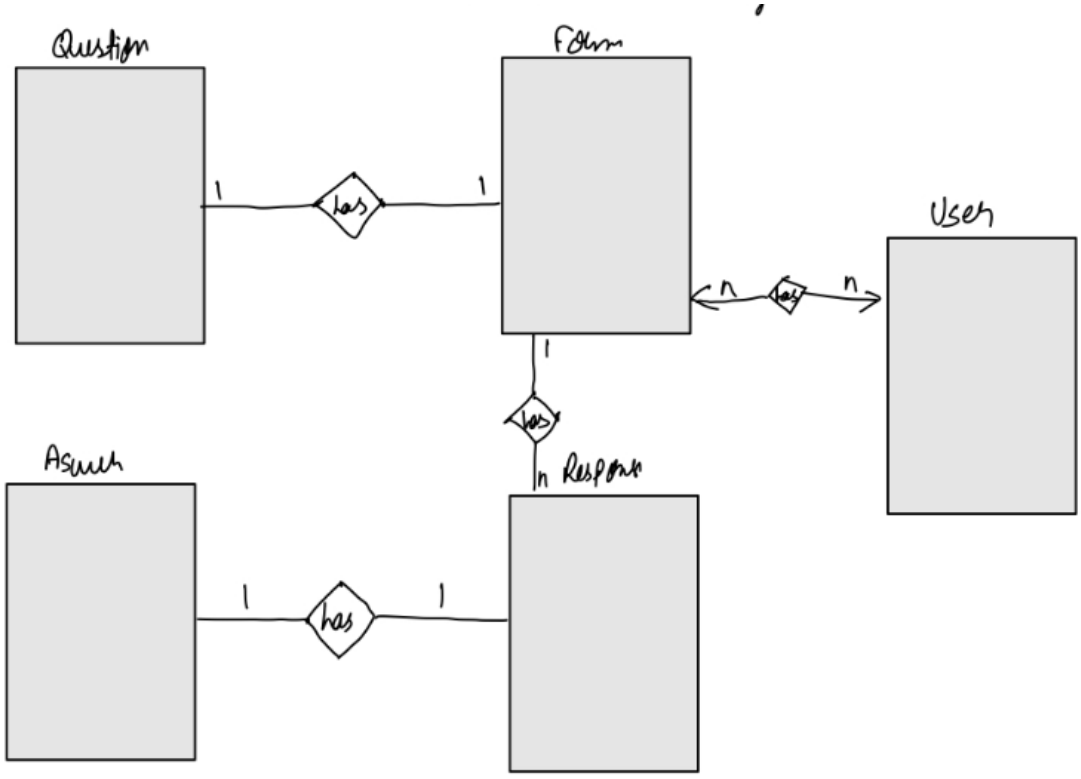
Database Choice: The database that I chose to go with was MongoDB since NoSQL is much faster than traditional SQL databases in terms of read and write speed, especially in key-value storage like Berkeley DB, which means less waiting time in scenarios such as online transactions.

- ➢ Introduction
  - The goal of this project is to create a data collection platform which can be used by customers to power their most critical activities. There are various use cases available which should be implemented in a plug and play fashion.
- ➢ System Architecture
  - The technology stack that I have used is MERN.
  - The architecture that I am following is Microservice architecture which allows me to achieve the plug and play goal, for more details refer to Project Approach pdf.
  - The architecture consists of three servers, main server, Google Sheets server and Market Research Agency Server.
  - The main server communicates with the database and can register users. Only registered users can create forms, add questions and various other things. Non users can only submit responses to a form.
- ➢ Available Route
  - Main Server
    - Create a User
    - Authenticate a User
    - Get logged-in User Details
    - Create a Form
    - Get Form ID by Name
    - Get Form Details by ID
    - Add Questions to a Form
    - Display Form
    - Get Submit Form
    - Get all Responses for a Form

        - Get Answers for an Answer
        - Submit Feedback
    - Google Sheets Server
        - Route to fetch questions and responses for a form and export data to Google Sheets
    - Market Research Agency Server
        - Fetch Data
        - Route to validate responses and send back flagged responses

➢ Database Design

- Right now for the database I have a schema with four models: Form, Questions, Responses and Answers where the relationship between Form-Question would be one-to-one as each form would correspond to one particular table of questions. The relationship between Form-Responses would still remain one-to-many as each form can have multiple responses. The relationship between responses-answer would be one-to-one as each response would be associated with a particular answer table given by a user.

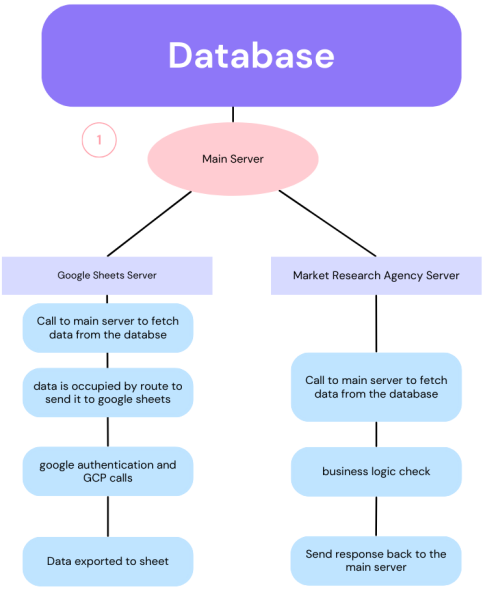


➢ Module/Component Design

- The modules/component design is such that each server has an app.js, routes and controllers. The main server has an additional database to it since its primary task is to communicate with the database.

- Once again pertaining to the limited time that I had I would have preferred to have a middleware to the main server to handle authentication and authorization which yes, also means that I wanted to have different users on the data platform as well.

➢ Data Flow Diagram



➢ Backend Design