Analysis of various eligible approaches:-

After looking at the problem statement, I went for a **NoSQL** database. NoSQL databases are often more scalable and provide better performance and are best suited for cloud computing.
I designed the schema on **Mongoose** which is an object modelling tool for **MongoDB** and hosted the database on MongoDB Atlas. Built the backend on ExpressJS.

Pros:

1. Provides the facility of **embedding** in hierarchical fashion minimizing the need of joins as in case of regular RDBMS, making the process faster.
2. NoSQL databases are **flexible** and support widely used data formats.

I have used **embedding**, according to the requirements of the database (**assumption**: data will be used for reading only and has a relation of 1:squillion), so as to make the process faster.
We can host images/videos on any image and video cloud based management service like **cloudinary**. It works well with NodeJS and I have worked on it before.

To have an efficient system for instant query results and **plug and play system** to accommodate any novice requirement we can use any **columnar database** that stores the data in **consecutive locations** in column by column fashion rather than traditional row by row basis as in the case of any regular RDBMS. This approach makes the data retrieval faster. At the same time columnar database gives flexibility to store millions of attributes that is not possible in case of predefined schema based RDBMS.

All these features resonated with the problem statement as it is flexible, faster and more scalable and thus seem to be perfect for Collect.

Other alternative:
We can use cloud based databases and hosting services to scale up and make things efficient and faster. We can **transpose this database** that I have designed to other cloud based databases but MongoDB is faster.
We can then monitor the data using **real time analysis** and create **microservices** related to the product so that the application is failsafe. And since it's on cloud, data will recover during power or service outage.

I used **MongoDB Atlas** to host this schema which I designed on Mongoose because most of the cloud services are paid and the ones that are free, were asking for credit card information. I am not a credit card holder as of now. Moreover, this schema can be transposed to the cloud so learning the cloud and building this schema over there shouldn't be that hard.