# FINAL PROJECT REPORT

# HEALTHCARE INDUSTRY MANAGEMENT SYSTEM


**Pradyoth Singenahalli Prabhu**
*Roll No:* 42
*Student ID:* 02071847
Data Science
University Massachusetts Dartmouth, Dartmouth

**Bharath Anand**
*Roll No:* 02
*Student ID:* 02044023
Data Science
University Massachusetts Dartmouth, Dartmouth

**Varun W R**
*Roll No:* 49
*Student ID:* 02070206
Data Science
University Massachusetts Dartmouth, Dartmouth

# Abstract:

The healthcare industry is rapidly adopting new technologies to improve patient outcomes, increase efficiency, and reduce costs. One of the key challenges faced by healthcare organizations is managing vast amounts of data generated by patients, healthcare providers, and medical devices. Traditional relational databases have limitations in handling the volume and variety of data generated in healthcare. NoSQL databases have emerged as a solution to these challenges, offering scalability, high availability, and data security.

This project aimed to develop healthcare management system using NoSQL databases - DynamoDB, MongoDB, and CouchDB. The system is designed to efficiently manage patient data, medical records, and related information, providing a single platform for healthcare providers, patients, and other stakeholders to access and manage healthcare data. The system incorporates features such as data creation, loading, updating, and querying, and ensures data security and privacy.

The project involved the use of three different categories of NoSQL systems, each with a specific use case. DynamoDB was used to store data related to drugs, hospitals, insurance, patients, and pharmacy. MongoDB was used to store patient visit details, and CouchDB was used to store patient lab reports and diagnosis details. The system was designed to be scalable and highly available, ensuring that healthcare providers could access patient data in real-time, from any location.

The project contributes to the growing use of NoSQL databases in healthcare and demonstrates their potential to provide efficient management of large volumes of unstructured data. The use of NoSQL databases can help healthcare organizations overcome the limitations of traditional relational databases, enabling them to manage large volumes of data, improve patient outcomes, and reduce costs. Overall, this project demonstrates the potential of NoSQL databases to transform the healthcare industry and improve patient care.

# Introductions:

The healthcare industry generates vast amounts of data daily, including patient records, medical histories, prescribed medications, and demographic information. Efficient management of this data is critical to providing quality patient care, reducing costs, and ensuring compliance with regulatory requirements. Traditional relational databases have been the preferred choice for healthcare industry management systems. However, these databases have limitations in handling large volumes of unstructured data, leading to reduced efficiency and increased costs.

In recent years, NoSQL databases have emerged as a viable alternative to relational databases for managing large volumes of unstructured data. NoSQL databases offer several advantages, including scalability, flexibility, and the ability to handle complex data structures.

This project aims to enhance the effectiveness of healthcare industry management systems by incorporating NoSQL databases. The system will facilitate the efficient management of treatment records, prescribed medications, confidential patient demographics, and other relevant data. By utilizing NoSQL databases, the project aims to enhance scalability and flexibility in managing substantial amounts of unstructured data in a more economical and effective manner.

The system's primary goal is to efficiently organize treatment records, prescribed medications, and confidential patient demographics. Healthcare professionals will be able to access patient information quickly and accurately, leading to better patient care. This project will contribute to the growing use of NoSQL databases in healthcare and provide valuable insights into the benefits of incorporating NoSQL databases in healthcare industry management systems.

# Part 1: Incorporating Multiple Data Store Types in the Database-driven Application.

The planned database-driven application will make use of three different data store paradigms: DynamoDB, MongoDB, and CouchDB. Each data store will be used to manage different types of healthcare data, as follows:

DynamoDB will be used to manage structured data, including DRUGS, HOSPITAL, INSURANCE, PATIENTS, PHARMACY, and PATIENTS demographic data.

MongoDB will be used to manage semi-structured data, specifically patient visits to the hospital. MongoDB provides flexibility in managing complex data structures, making it ideal for storing and managing the medical records associated with patient visits.

CouchDB will be used to manage unstructured data, including lab reports that may include PDFs. CouchDB provides offline access and synchronization capabilities, making it ideal for managing

data that needs to be accessed in locations where internet connectivity may be limited or inconsistent.

Incorporating these three data store paradigms into the database-driven application will enhance its capabilities to manage various types of healthcare data more efficiently. The application will also be able to scale horizontally and vertically, thereby accommodating the increasing amount of data generated in the healthcare industry.

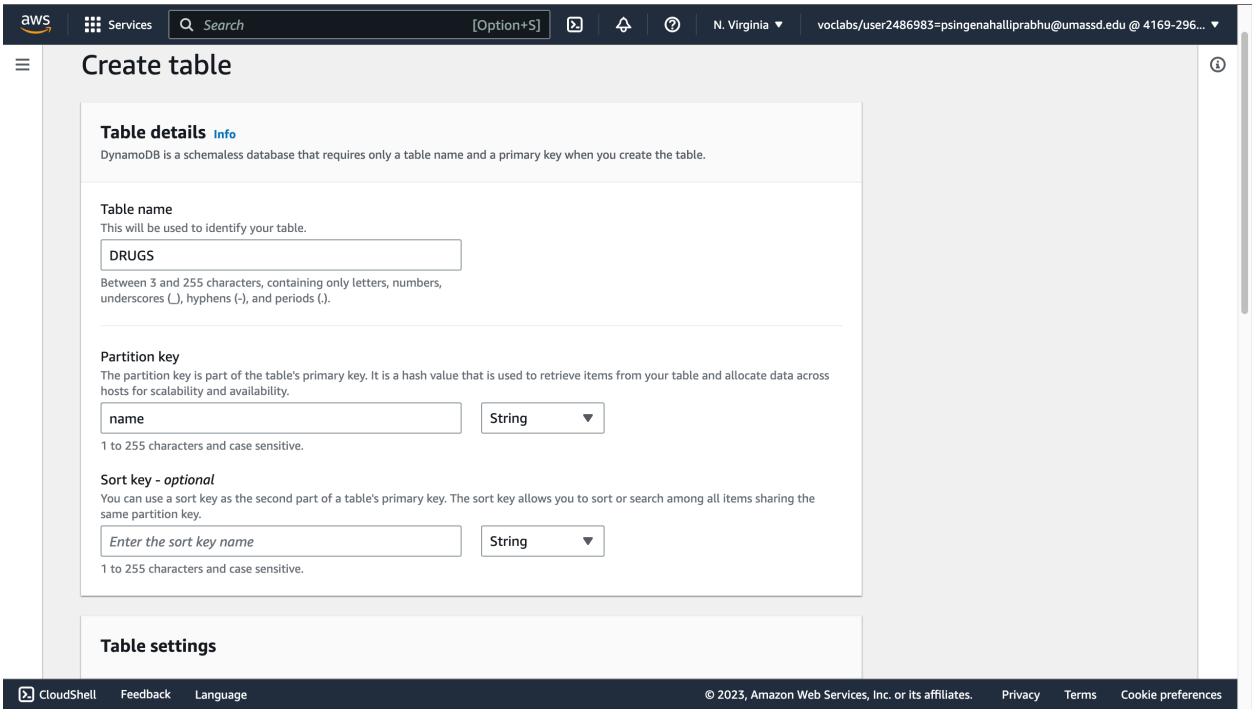
# Part - 2: Working Incorporation of 1st Data Store Paradigm – DynamoDB

To demonstrate the working incorporation of the first data store paradigm, DynamoDB, we have implemented the database-driven application with the following features:

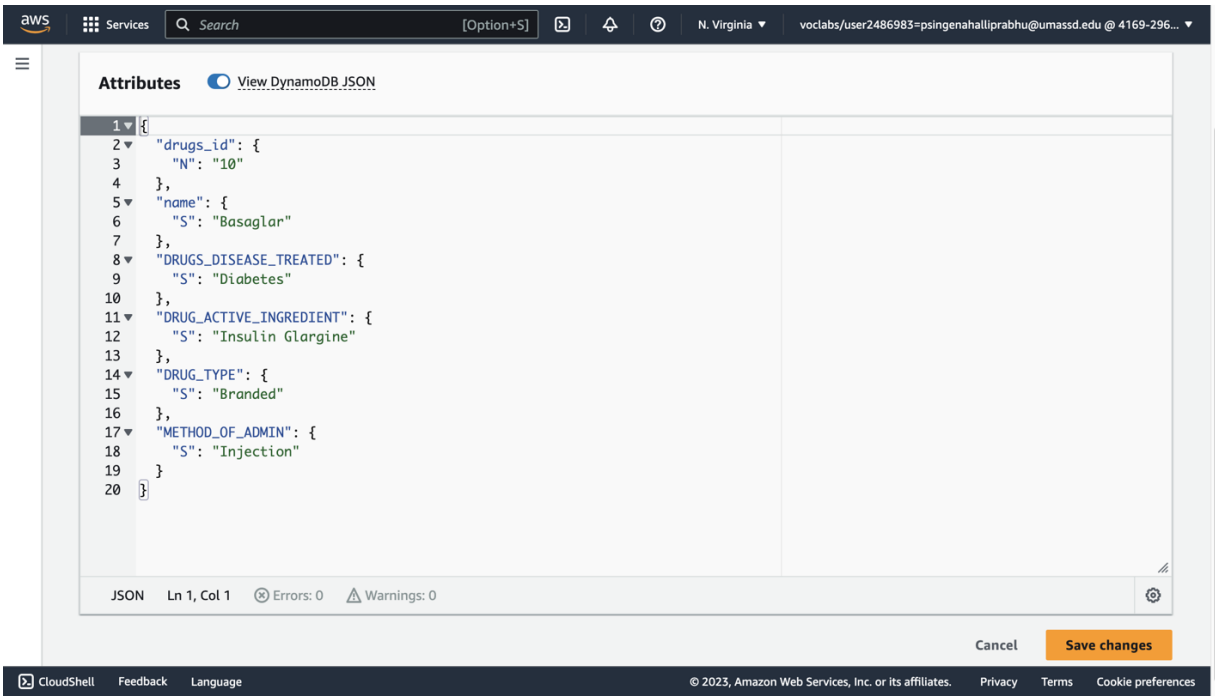
- Creation, loading, updating, and querying of DRUGS, HOSPITAL, INSURANCE, PATIENTS, PHARMACY, and PATIENTS demographic data in DynamoDB.

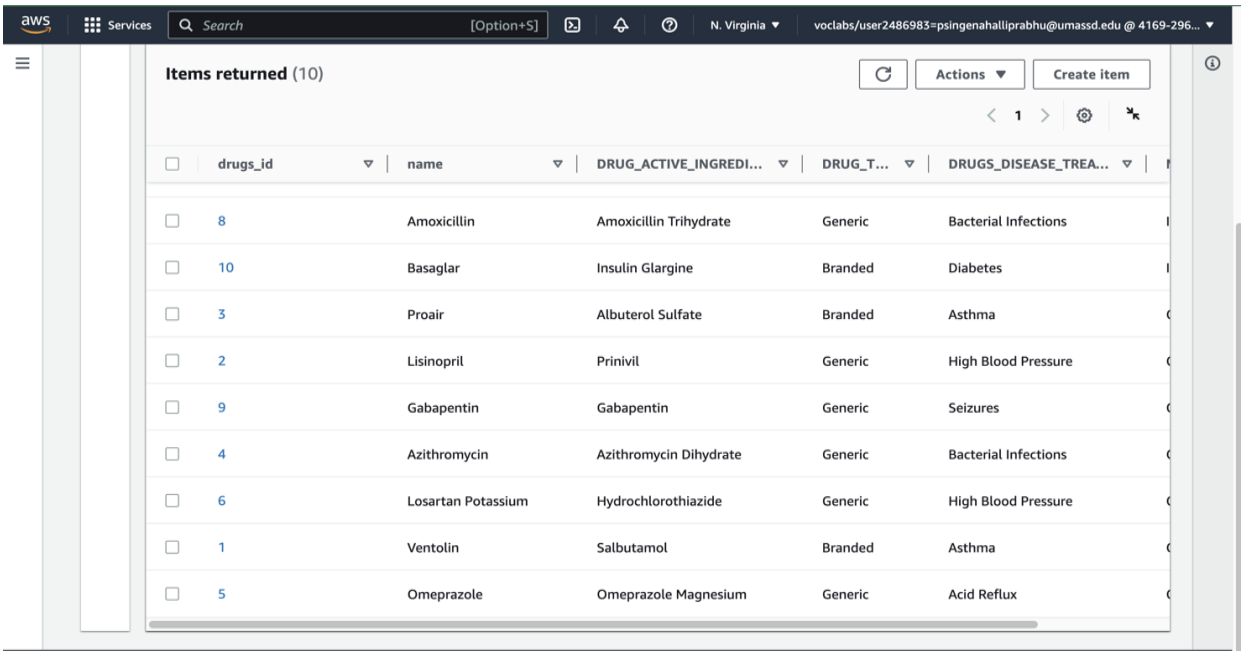
The screenshots below demonstrate the working implementation of the DynamoDB database-driven application.

## Create Table:

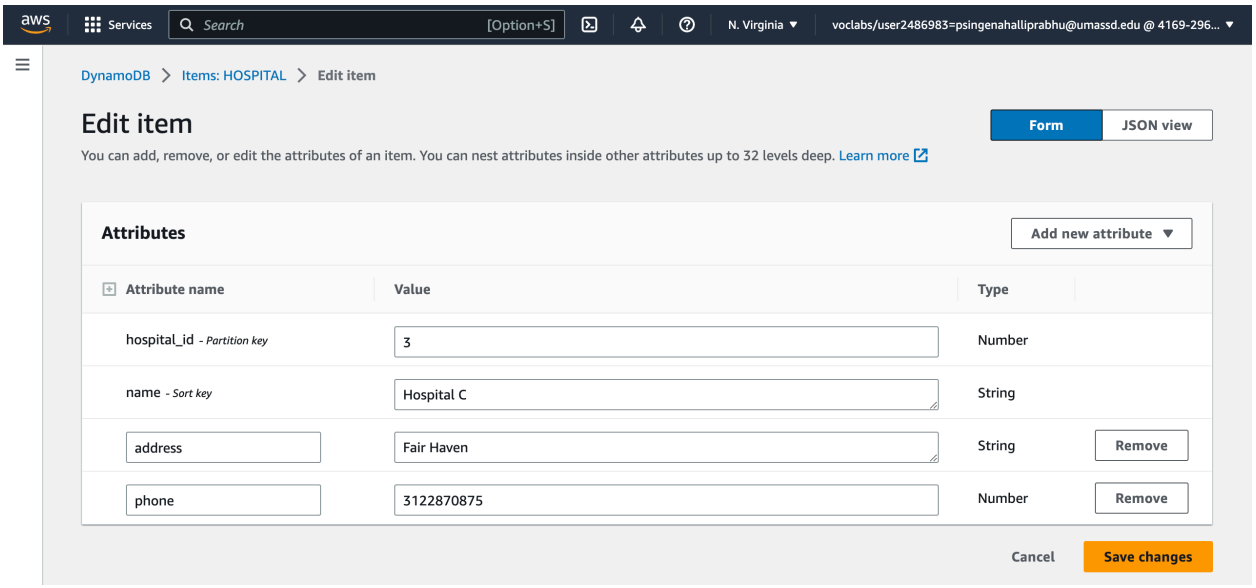


## Loading item:



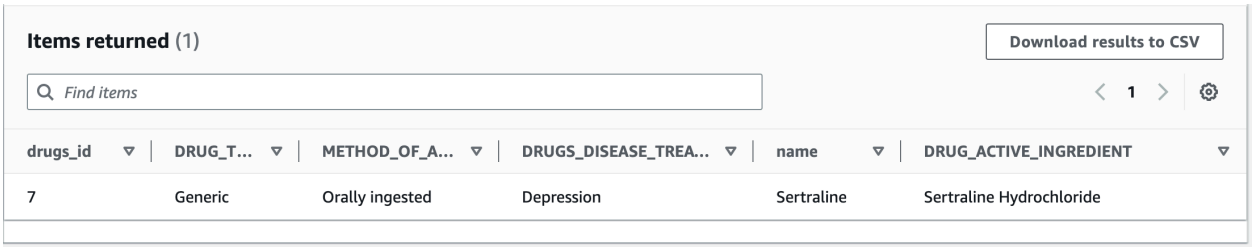
## Update Item:




## Query Item:

### Query 1:

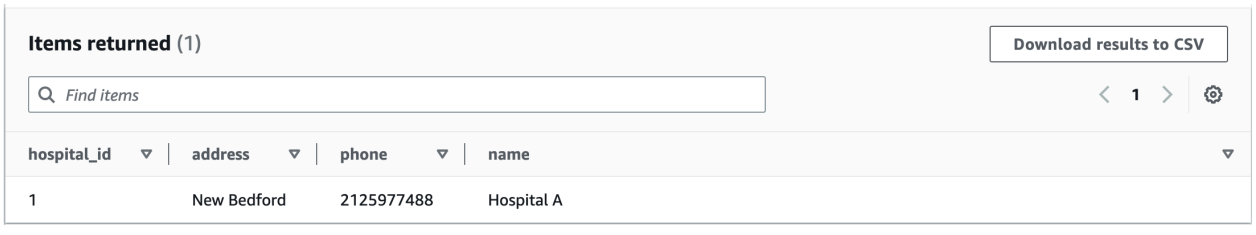
```
SELECT * FROM DRUGS WHERE name='Sertraline'
```




| drugs_id | DRUG_T... | METHOD_OF_A... | DRUGS_DISEASE_TREA... | name | DRUG_ACTIVE_INGREDIENT |
|---|---|---|---|---|---|
| 7 | Generic | Orally ingested | Depression | Sertraline | Sertraline Hydrochloride |

### Query 2:

```
SELECT * FROM HOSPITAL WHERE name='Hospital A'
```




| hospital_id | address | phone | name |
|---|---|---|---|
| 1 | New Bedford | 2125977488 | Hospital A |

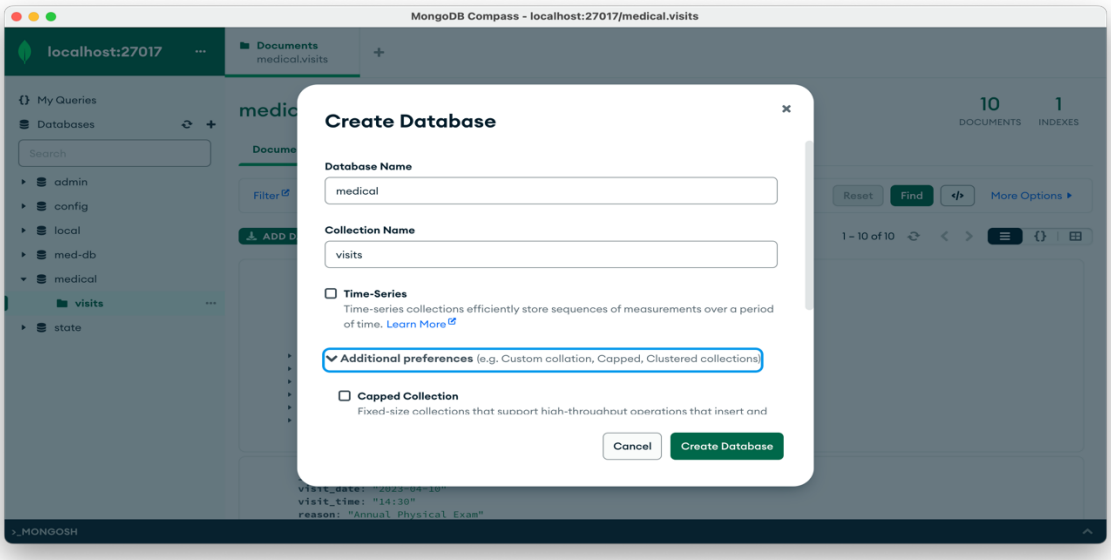
# Part - 3 (Working Incorporation of 2nd Data Store Paradigm) – MongoDB

For the second data store paradigm, we have used MongoDB to store patients' visits to the hospital. The structure of the data stored in MongoDB is as follows:

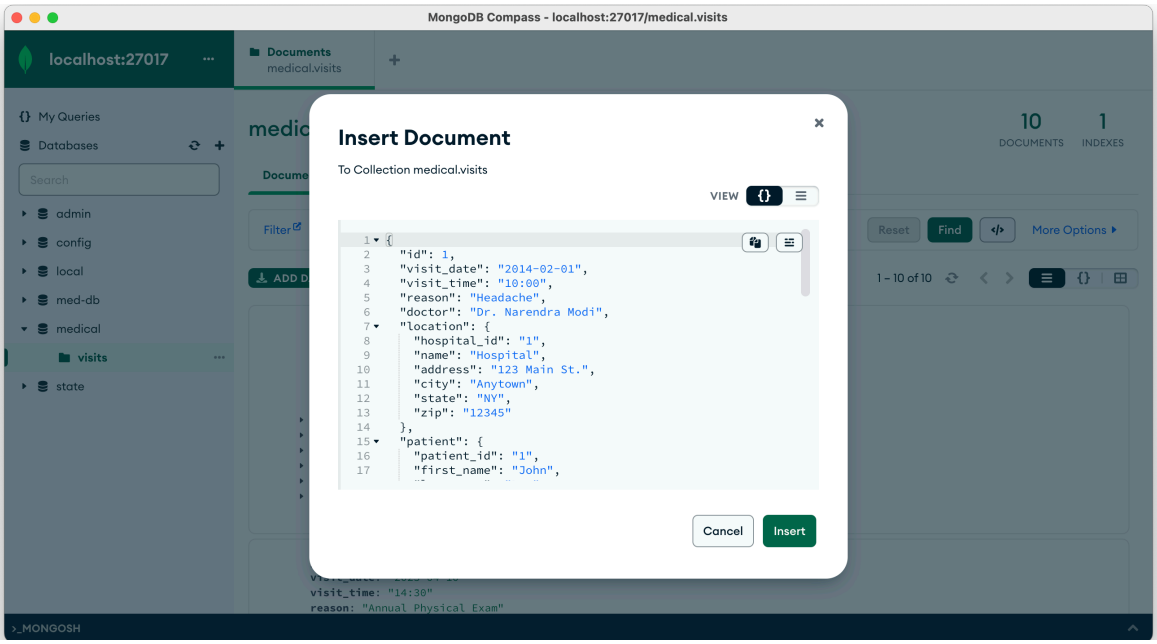
- `_id`: The unique ID of the document.
- `id`: A unique ID for each visit.
- `visit_date`: Date of the visit.
- `visit_time`: Time of the visit.
- `reason`: The reason for the visit.
- `doctor`: The doctor who attended to the patient during the visit.
- `location`: Information about the hospital where the visit took place.
    - `hospital_id`: The unique ID of the hospital.
    - `name`: The name of the hospital.
    - `address`: The address of the hospital.
    - `city`: The city where the hospital is located.
    - `state`: The state where the hospital is located.
    - `zip`: The ZIP code of the hospital.
- `patient`: Information about the patient.
    - `patient_id`: The unique ID of the patient.
    - `first_name`: The first name of the patient.
    - `last_name`: The last name of the patient.
    - `address`: The address of the patient.
    - `city`: The city where the patient lives.
    - `state`: The state where the patient lives.
    - `zip`: The ZIP code of the patient.
    - `phone`: The phone number of the patient.
- `insurance`: Information about the insurance policy held by the patient.
    - `insurance_id`: The ID of the insurance policy.
- `diagnosis`: Diagnosis details related to the patient.
    - `diagnosis_id`: The unique ID of the diagnosis.
    - `name`: The name of the diagnosis.
    - `description`: A description of the diagnosis.
- `medication`: Medication details related to the patient.
    - `medication_id`: The unique ID of the medication.
    - `name`: The name of the medication.
    - `description`: A description of the medication.
- `procedure`: Procedure details related to the patient.
    - `procedure_id`: The unique ID of the procedure.
    - `name`: The name of the procedure.
    - `description`: A description of the procedure.
- `notes`: Any notes about the patient's condition or treatment.

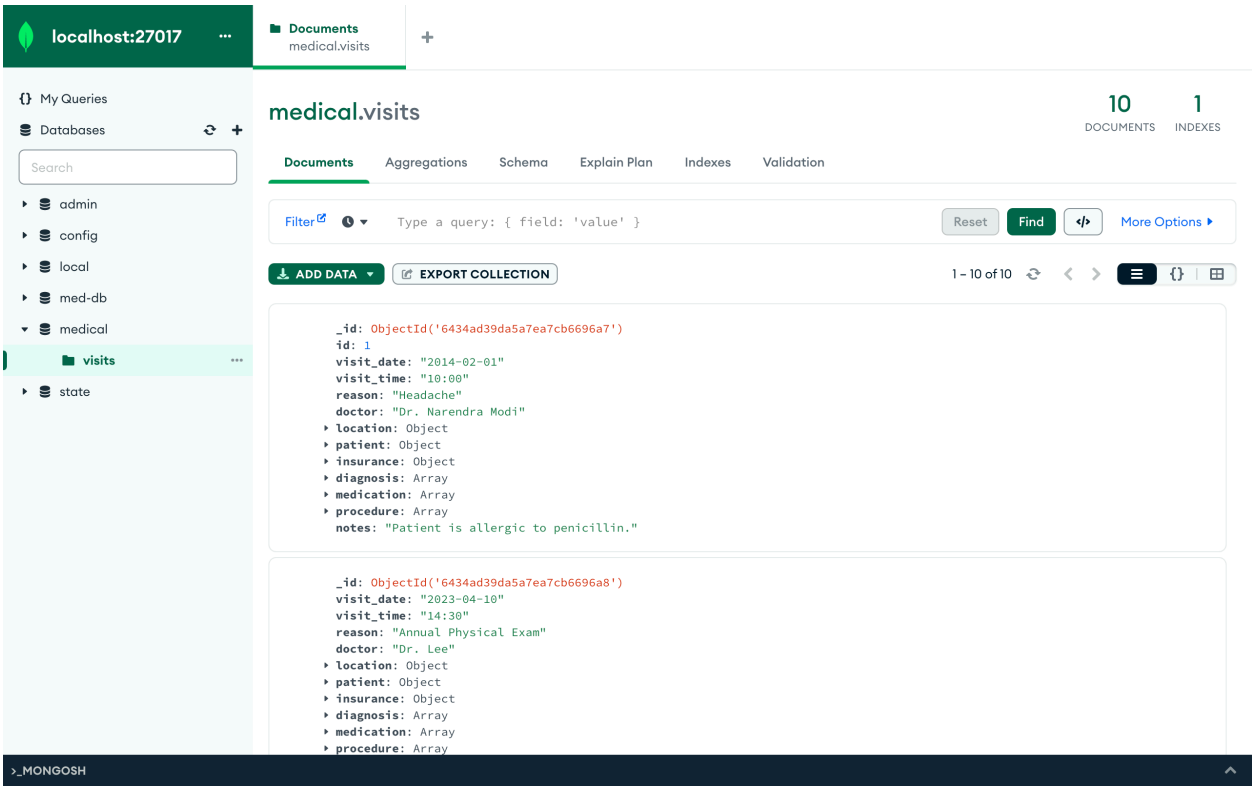
Screenshots of the working database-driven application with MongoDB integration have been provided below. These screenshots demonstrate that the application is able to create, load, update, and query data in accordance with our requirements.

## Create Collection:

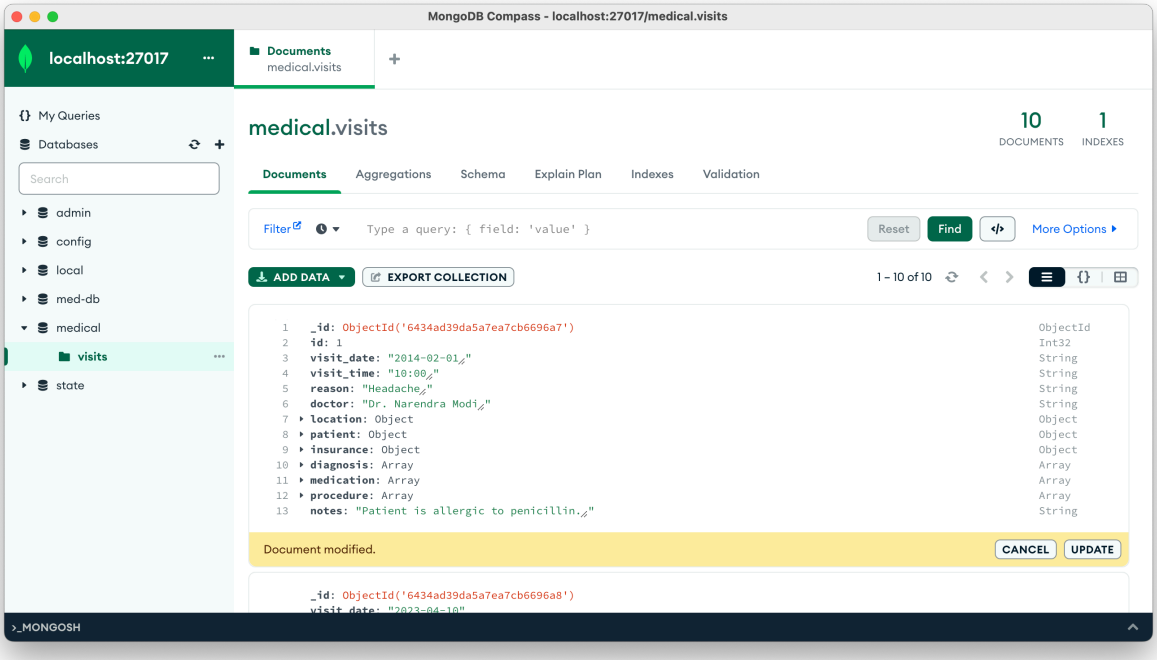



## Load Document:





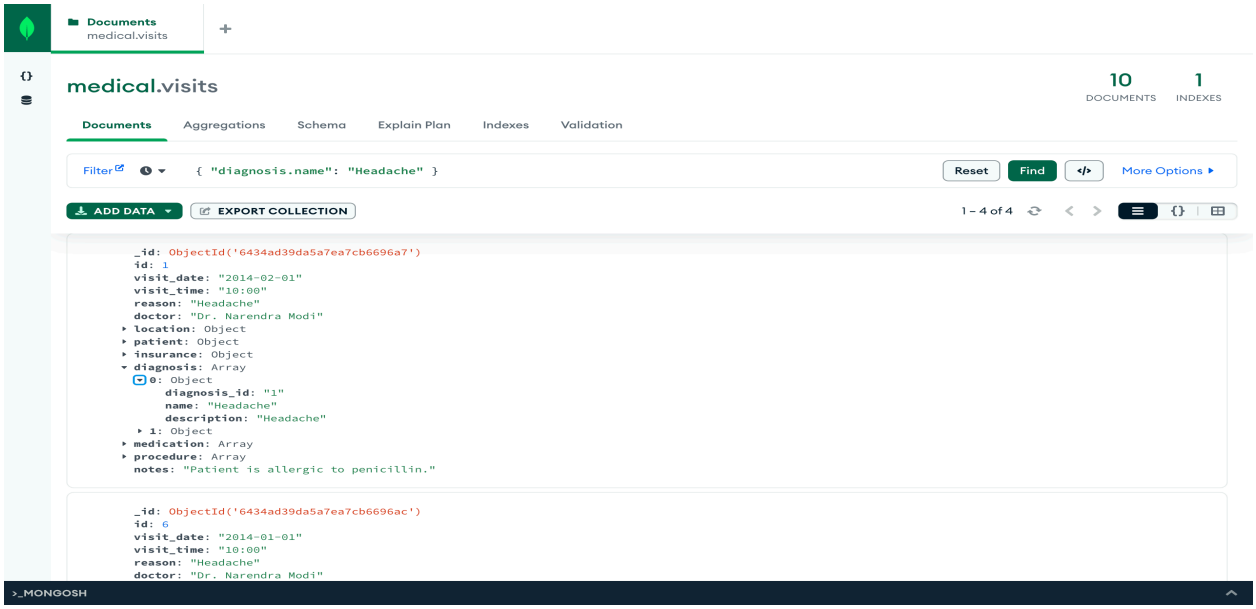

# Update Document:



## Query Document:

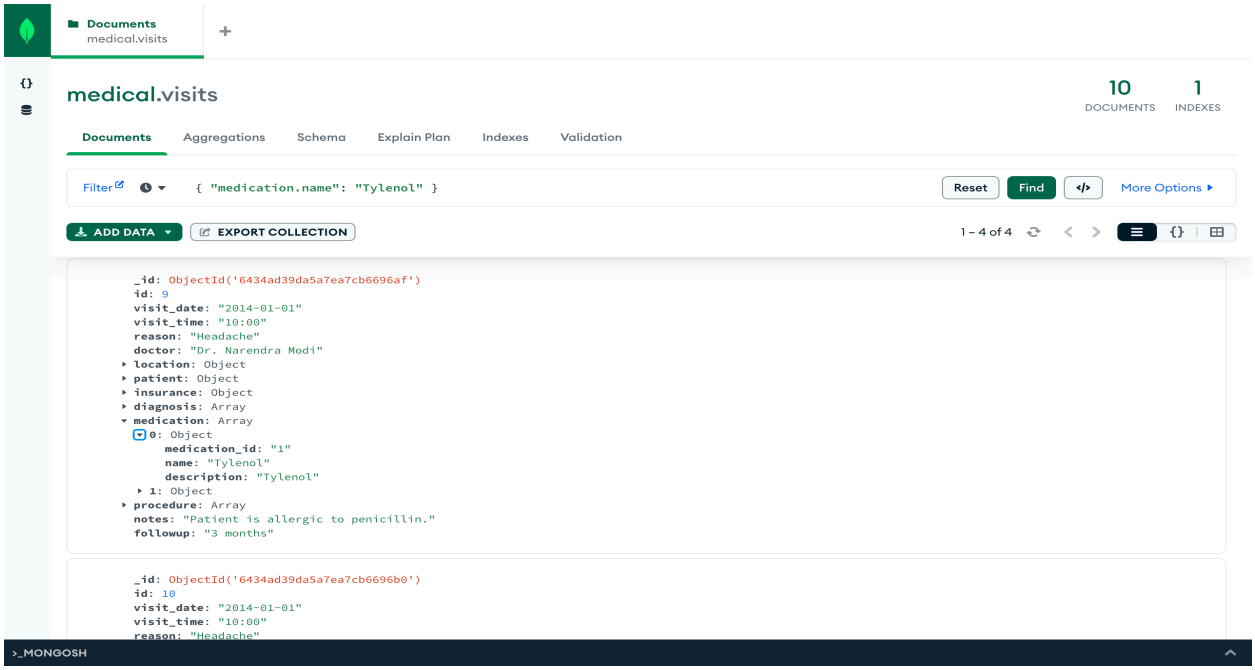
1. Find all patient visits where a particular diagnosis (Headache) was made:

   Query: { "diagnosis.name": "Headache" }




2. Find all patient visits where a particular medication (Tylenol) was prescribed:

   Query: { "medication.name": "Tylenol" }



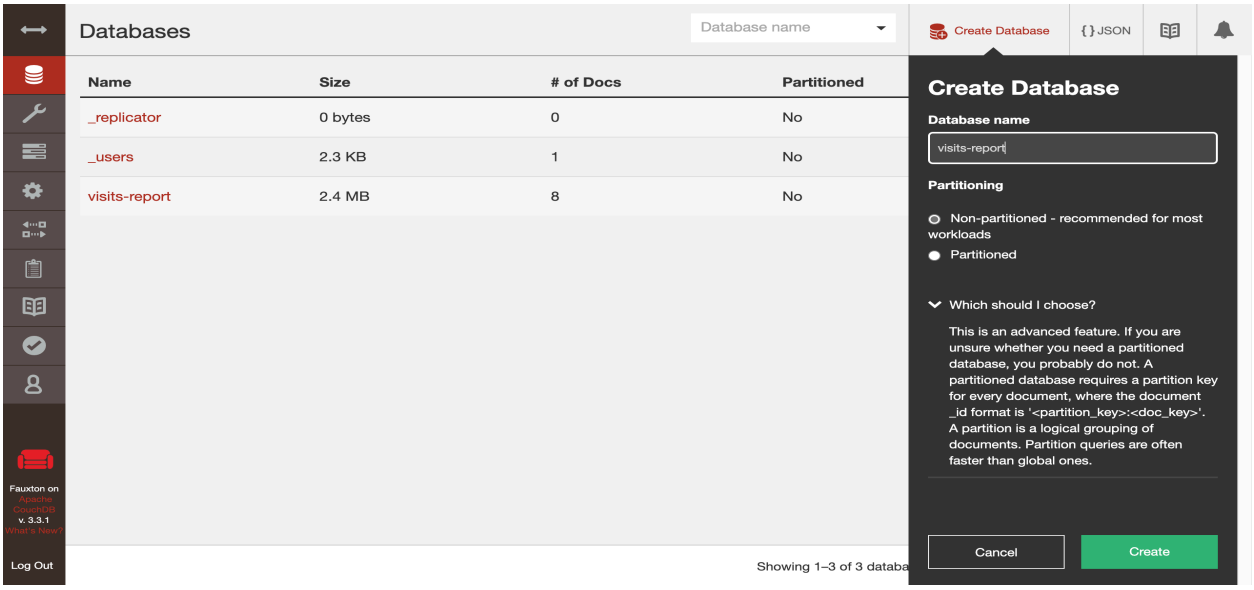
# Part - 4 (Working Incorporation of 3rd Data Store Paradigm) – CouchDB

For the third data store paradigm, we have used CouchDB to store patients' lab reports including PDFs.

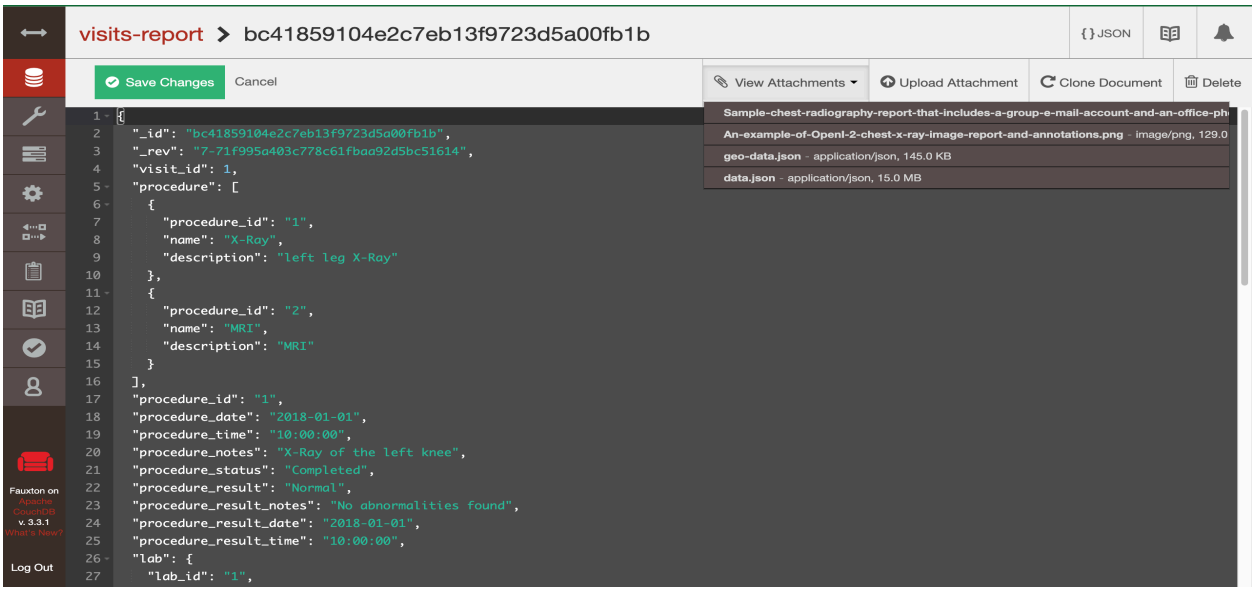
Here are the details of the fields for a report in the CouchDB database:

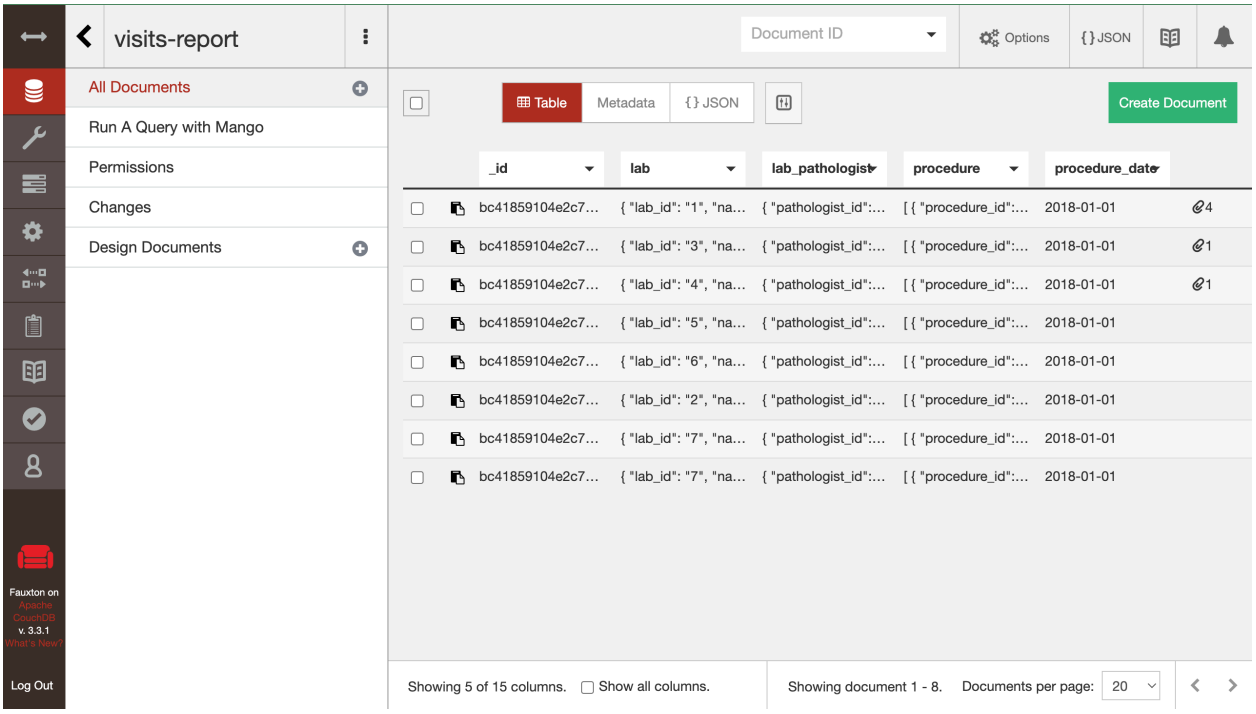
- `_id`: The unique identifier for the document.
- `_rev`: The revision number of the document.
- `visit_id`: The unique identifier of the visit associated with the report.
- `procedure`: An array of procedures performed during the visit, with details about each procedure.
- `procedure_date`: The date when the procedure was performed.
- `procedure_time`: The time when the procedure was performed.
- `procedure_notes`: Any notes related to the procedure.
- `procedure_status`: The status of the procedure, such as "completed" or "pending".
- `procedure_result`: The result of the procedure, such as "normal" or "abnormal".
- `procedure_result_notes`: Any notes related to the result of the procedure.
- `procedure_result_date`: The date when the result of the procedure was determined.
- `procedure_result_time`: The time when the result of the procedure was determined.
- `lab`: Details about the laboratory where the procedure was performed, including its name, address, and contact information.
- `lab_pathologist`: Details about the pathologist who interpreted the results of the procedure, including their name and contact information.

## Create Database:



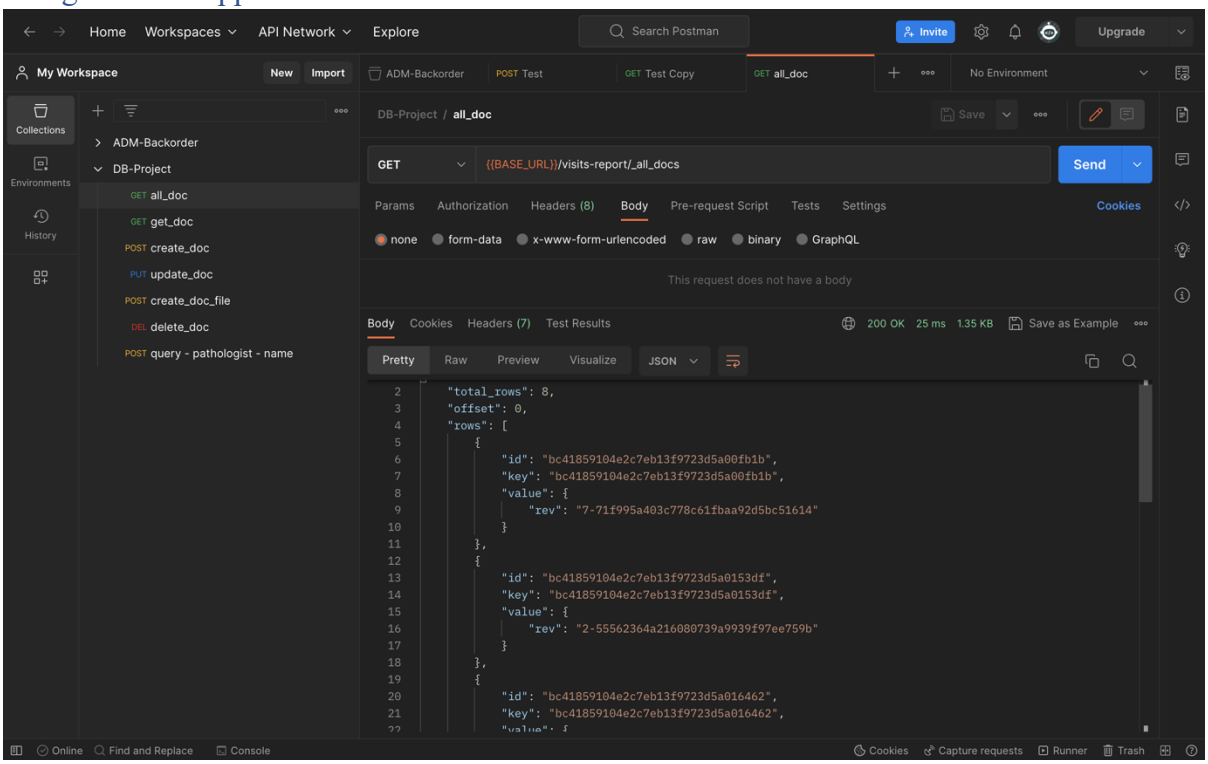
## Insert Document:



Using HTTP Rest API to get all documents:

Using Postman application:



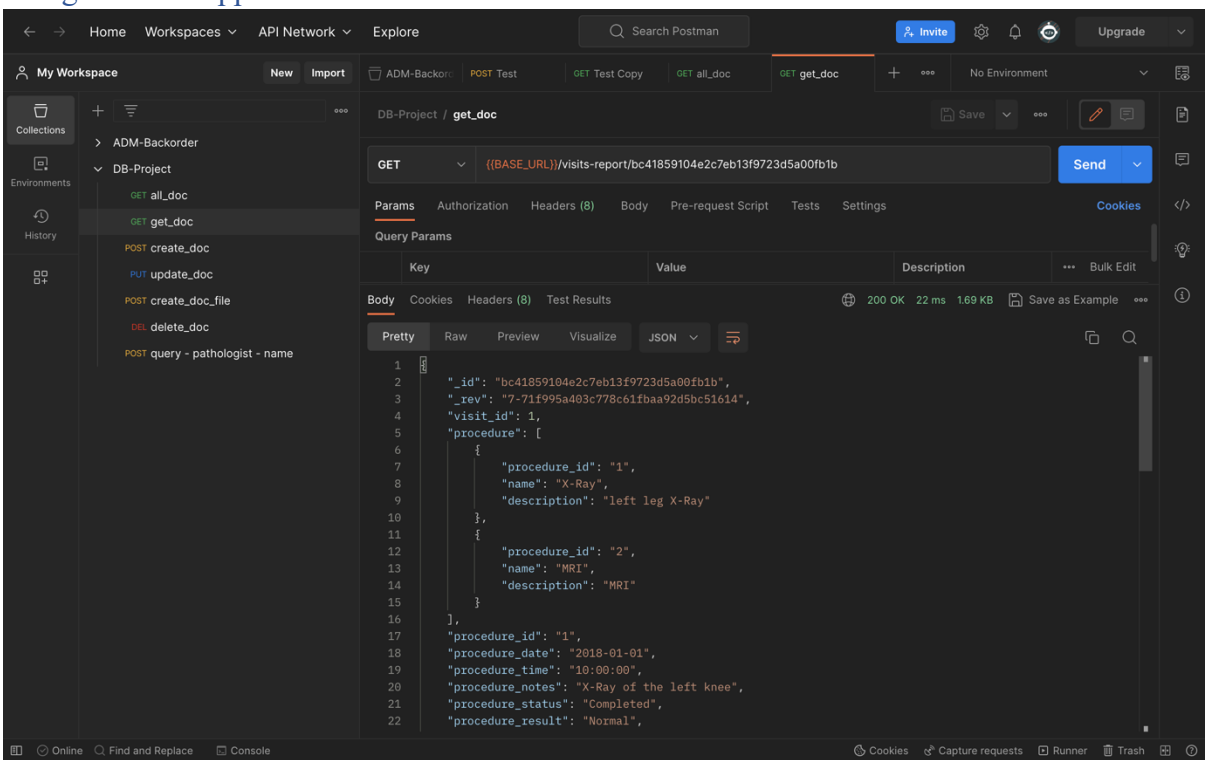

Using curl :
URL: http://root:password@localhost:5984/visits-report/_all_docs

```
➜  ~ curl -X GET http://root:password@localhost:5984/visits-report/_all_docs

{"total_rows":8,"offset":0,"rows":[
{"id":"bc41859104e2c7eb13f9723d5a00fb1b","key":"bc41859104e2c7eb13f9723d5a00fb1b","value":{"rev":"7-71f995a403c778c61fbaa92d5bc51614"}},
{"id":"bc41859104e2c7eb13f9723d5a0153df","key":"bc41859104e2c7eb13f9723d5a0153df","value":{"rev":"2-55562364a216080739a9939f97ee759b"}},
{"id":"bc41859104e2c7eb13f9723d5a016462","key":"bc41859104e2c7eb13f9723d5a016462","value":{"rev":"2-5cfb91aec142b3e9ec81d0fbaee3fc51"}},
{"id":"bc41859104e2c7eb13f9723d5a017adf","key":"bc41859104e2c7eb13f9723d5a017adf","value":{"rev":"1-b30bcd2307bd705d83bc0008ca690e20"}},
{"id":"bc41859104e2c7eb13f9723d5a018ca2","key":"bc41859104e2c7eb13f9723d5a018ca2","value":{"rev":"1-34acf97a9d4e71b1d262e948a588d735"}},
{"id":"bc41859104e2c7eb13f9723d5a020b07","key":"bc41859104e2c7eb13f9723d5a020b07","value":{"rev":"1-5f410911faf0187ea6cf190c1fade31c"}},
{"id":"bc41859104e2c7eb13f9723d5a03117d","key":"bc41859104e2c7eb13f9723d5a03117d","value":{"rev":"3-89a4ab576d30e74d69fd8af8166f6411"}},
{"id":"bc41859104e2c7eb13f9723d5a035774","key":"bc41859104e2c7eb13f9723d5a035774","value":{"rev":"2-f92e421d923fa185a0fedc6d027a3a35"}}
]}
➜  ~
```

Using HTTP Rest API to get single document using _id :

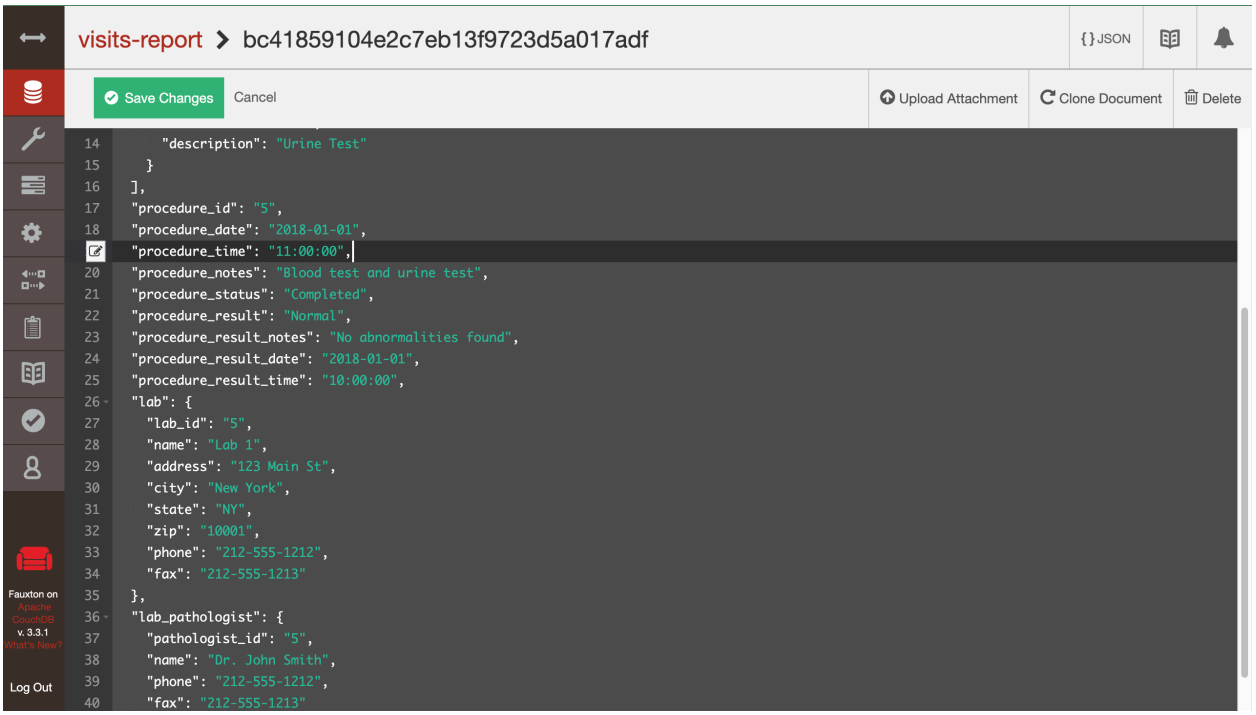
Using Postman application:




Using CURL:

URL: curl -X GET http://root:password@localhost:5984/visits-report/bc41859104e2c7eb13f9723d5a00fb1b

## Update Document:




## Query:

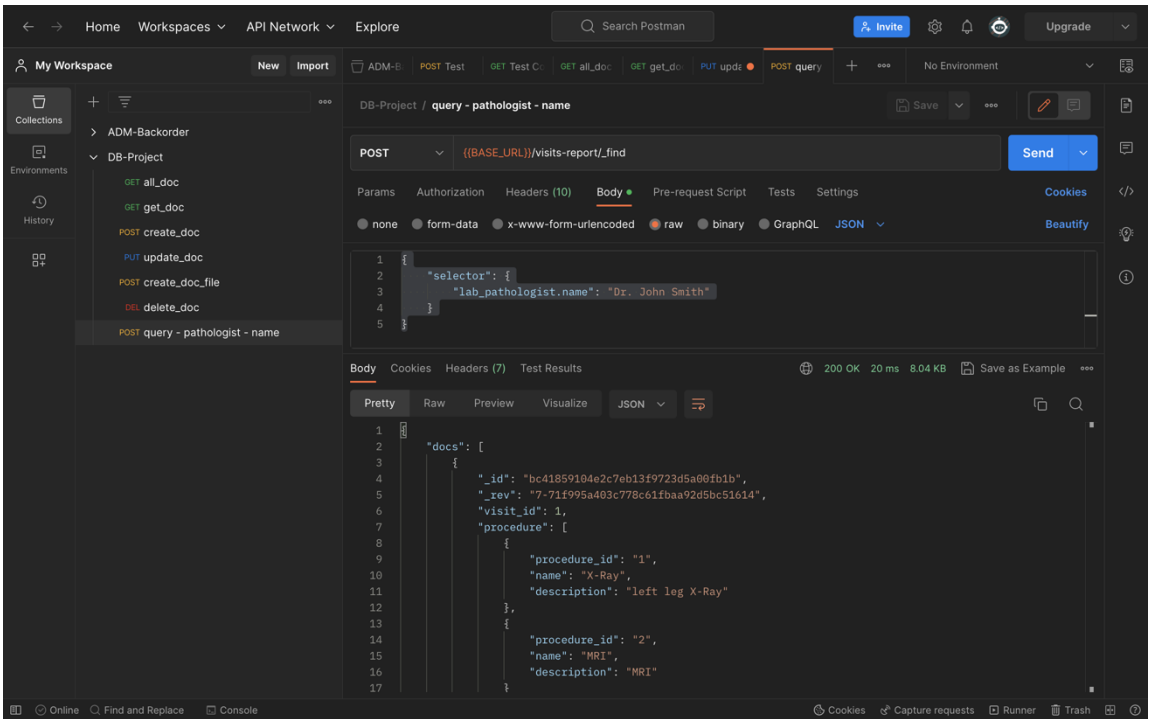
1. **To find lab reports assigned to a specific pathologist (**Dr. John Smith)**:**

The provided query is searching for lab reports where the name of the pathologist is "Dr. John Smith". It is using the *selector* syntax of CouchDB, which is a way of specifying search criteria for documents in the database. The query specifies the key *lab_pathologist.name* to search for the name of the pathologist. The value for this key is set to "Dr. John Smith". Therefore, the query will return all the lab reports that have a pathologist with the name "Dr. John Smith".

### Using Postman:

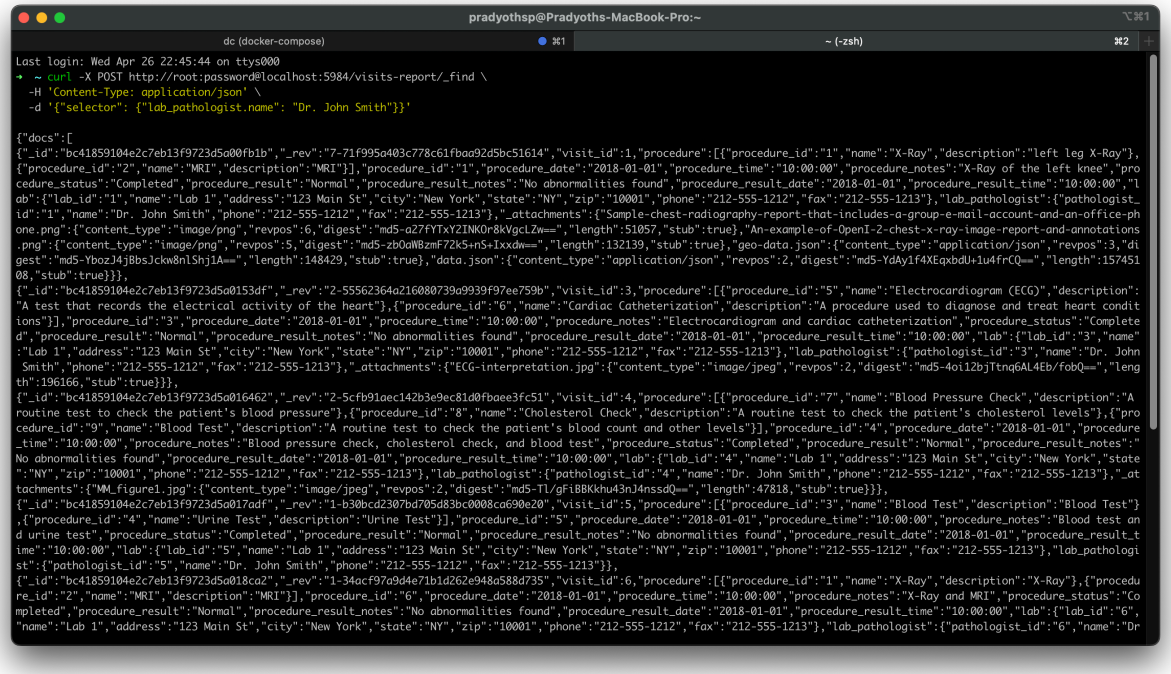
Query:

```
{
"selector": {
       "lab_pathologist.name": "Dr. John Smith"
       }
}
```




## Using CURL:

```
curl -X POST http://root:password@localhost:5984/visits-report/_find \
  -H 'Content-Type: application/json' \
  -d '{"selector": {"lab_pathologist.name": "Dr. John Smith"}}'
```



# Conclusion:

In conclusion, this project involved the development of a healthcare application that allows doctors to manage patient information, diagnoses, medications, and procedures. The application was designed with scalability and reliability in mind, utilizing three different NoSQL databases: DynamoDB, MongoDB, and CouchDB.

DynamoDB was used to store patient and doctor information, as well as diagnosis and medication details. MongoDB was used to store visit and procedure information, providing fast and reliable access to data through its highly scalable and fully managed NoSQL database service. Finally, CouchDB was used to store patient lab reports, including unstructured data like PDFs and images.

The use of NoSQL databases allowed for flexible and dynamic data modeling, enabling quick and efficient retrieval of data. The implementation of multiple data stores ensures high availability, fault tolerance, and disaster recovery capabilities.

Overall, this project demonstrates the power and flexibility of NoSQL databases in developing highly scalable and reliable applications for the healthcare industry.