Distributed Database Systems (CSE 512)

Group project

Task 5

Introduction

Our objective is to showcase how a NoSQL database, particularly MongoDB, can effectively meet the dynamic and diverse data management needs of modern healthcare systems. This task serves as a resource for database administrators, developers, and stakeholders involved in the deployment and maintenance of the database system.

Healthcare data can be diverse and change over time. MongoDB's schema-less nature allows for storing different types of data in the same collection without a rigid structure, accommodating evolving data requirements without the need for extensive database redesign. As healthcare data grows in volume and complexity, MongoDB scales horizontally by adding more servers. This scalability is crucial for handling large datasets, like patient records and medical images, efficiently.

MongoDB's flexibility, scalability, and powerful data processing capabilities make it an excellent choice for healthcare systems, which deal with diverse, complex, and sensitive data. Its ability to adapt to changing data models and scale to meet growing data demands ensures that healthcare applications built on MongoDB can continue to serve the needs of patients and healthcare providers effectively, now and in the future.

Scope

- System Design: Detailed overview of the database schema, including the structure of collections for patients, doctors, and appointments.
- Data Modeling: Insights into the process of modeling data for a healthcare system in a NoSQL environment, emphasizing the flexibility and scalability of MongoDB.

- CRUD Operations: Comprehensive guide on creating, reading, updating, and deleting data, which forms the core functionality of the database system.
- Implementation Examples: Code snippets and explanations for various operations performed using Python and the pymongo driver.
- Best Practices: Guidelines for effective database management, performance tuning, and ensuring data integrity and security within the MongoDB environment.

Code:

```
from pymongo import MongoClient
from faker import Faker
import random


def connect_to_mongo():
    client = MongoClient('mongodb://localhost:27017/')
    db = client.healthcare
    return db

def insert_random_data(db):
    faker = Faker()

    # Insert into patients collection
    patients = db.patients
    for _ in range(20):
        patient = {
            "name": faker.name(),
            "date_of_birth": faker.date_of_birth().isoformat(),
            "gender": random.choice(["M", "F"]),
            "address": faker.address(),
            "contact_number": faker.phone_number(),
            "email": faker.email()
        }
        patients.insert_one(patient)

    # Insert into doctors collection
    doctors = db.doctors
    for _ in range(20):
```

```
        doctor = {
            "name": faker.name(),
            "specialization": faker.job(),
            "contact_number": faker.phone_number(),
            "email": faker.email(),
            "availability": faker.day_of_week()
        }
        doctors.insert_one(doctor)

    # Insert into appointments collection
    appointments = db.appointments
    patient_ids = [p["_id"] for p in patients.find()]
    doctor_ids = [d["_id"] for d in doctors.find()]
    for _ in range(20):
        appointment = {
            "patient_id": random.choice(patient_ids),
            "doctor_id": random.choice(doctor_ids),
            "appointment_date": faker.date_time_this_year().isoformat(),
            "purpose": faker.sentence()
        }
        appointments.insert_one(appointment)

def display_data(db):
    print("Patients:")
    for patient in db.patients.find().limit(5):
        print(patient)

    print("\nDoctors:")
    for doctor in db.doctors.find().limit(5):
        print(doctor)

    print("\nAppointments:")
    for appointment in db.appointments.find().limit(5):
        print(appointment)

def update_query(db):
    # Fetching and randomly selecting a patient
    patient_ids = [p["_id"] for p in db.patients.find()]
    patient_id = random.choice(patient_ids)
    new_contact_number = "123-456-7890"
```

```
    print(f"Updating this ID's contact number with
123-456-7890:",patient_id)

    # Updating the patient's contact number
    result1 = db.patients.update_one({"_id": patient_id}, {"$set":
{"contact_number": new_contact_number}})

    # Fetching and randomly selecting a doctor
    doctor_ids = [d["_id"] for d in db.doctors.find()]
    doctor_id = random.choice(doctor_ids)
    print(f"Updating this ID's availability with Saturday:",doctor_id)

    new_availability = "Saturday"

    # Updating the doctor's availability
    result2 = db.doctors.update_one({"_id": doctor_id}, {"$set":
{"availability": new_availability}})

    return result1.modified_count, result2.modified_count


def delete_query(db):
    # Fetching and randomly selecting a patient
    patient_ids = [p["_id"] for p in db.patients.find()]
    if patient_ids:
        some_patient_id = random.choice(patient_ids)
        print(f"Deleting patient_id :",some_patient_id)
        deleted_count = db.patients.delete_one({"_id":
some_patient_id}).deleted_count
        print(f"Deleted {deleted_count} patient(s).")

    # Fetching and randomly selecting an appointment
    appointment_ids = [a["_id"] for a in db.appointments.find()]
    if appointment_ids:
        some_appointment_id = random.choice(appointment_ids)
        print(f"Deleting some_appointment_id :",some_appointment_id)
        deleted_count = db.appointments.delete_one({"_id":
some_appointment_id}).deleted_count
        print(f"Deleted {deleted_count} appointment(s).")
```

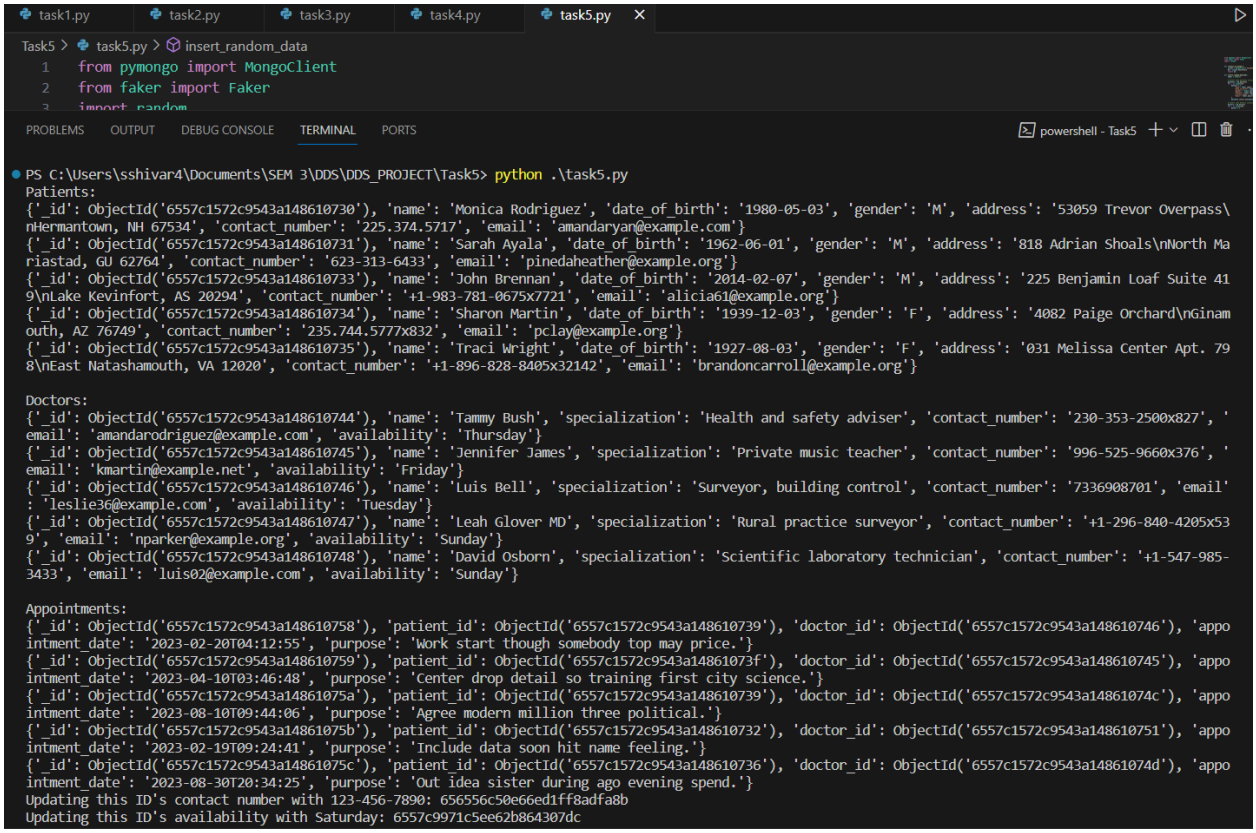
```
def main():
    db = connect_to_mongo()
    insert_random_data(db)
    display_data(db)
    update_result = update_query(db)
    print(f"Updated {update_result[0]} patient(s) and {update_result[1]}
doctor(s).")
    delete_query(db)
    display_data(db)

if __name__ == "__main__":
    main()
```

Below is the brief overview of each function:

1. connect_to_mongo():Establishes a connection to a MongoDB instance running on localhost.
2. insert_random_data(db):Inserts randomly generated data for patients and doctors into their respective collections. Creates appointments by randomly pairing patients and doctors.
3. display_data(db): Fetches and prints the first five documents from the patients, doctors, and appointments collections.
4. update_query(db): Randomly selects a patient and a doctor from the database. Updates the contact number of the selected patient and the availability of the selected doctor.
5. delete_query(db): Randomly selects and deletes a patient and an appointment from their respective collections.

Results:


task1.py
task2.py
task3.py
task4.py
task5.py
Task5 > task5.py > insert_random_data
from pymongo import MongoClient
from faker import Faker
import random
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell - Task5
PS C:\Users\sshivar4\Documents\SEM 3\DDS\DDS_PROJECT\Task5> python .\task5.py
Patients:
{'_id': ObjectId('6557c1572c9543a148610730'), 'name': 'Monica Rodriguez', 'date_of_birth': '1980-05-03', 'gender': 'M', 'address': '53059 Trevor Overpass\nHermantown, NH 67534', 'contact_number': '225.374.5717', 'email': 'amandaryan@example.com'}
{'_id': ObjectId('6557c1572c9543a148610731'), 'name': 'Sarah Ayala', 'date_of_birth': '1962-06-01', 'gender': 'M', 'address': '818 Adrian Shoals\nNorth Mariastad, GU 62764', 'contact_number': '623-313-6433', 'email': 'pinedaheather@example.org'}
{'_id': ObjectId('6557c1572c9543a148610733'), 'name': 'John Brennan', 'date_of_birth': '2014-02-07', 'gender': 'M', 'address': '225 Benjamin Loaf Suite 419\nLake Kevinfort, AS 20294', 'contact_number': '+1-983-781-0675x7721', 'email': 'alicia61@example.org'}
{'_id': ObjectId('6557c1572c9543a148610734'), 'name': 'Sharon Martin', 'date_of_birth': '1939-12-03', 'gender': 'F', 'address': '4082 Paige Orchard\nGinamouth, AZ 76749', 'contact_number': '235.744.5777x832', 'email': 'pclay@example.org'}
{'_id': ObjectId('6557c1572c9543a148610735'), 'name': 'Traci Wright', 'date_of_birth': '1927-08-03', 'gender': 'F', 'address': '031 Melissa Center Apt. 798\nEast Natashamouth, VA 12020', 'contact_number': '+1-896-828-8405x32142', 'email': 'brandoncarroll@example.org'}
Doctors:
{'_id': ObjectId('6557c1572c9543a148610744'), 'name': 'Tammy Bush', 'specialization': 'Health and safety adviser', 'contact_number': '230-353-2500x827', 'email': 'amandarodriguez@example.com', 'availability': 'Thursday'}
{'_id': ObjectId('6557c1572c9543a148610745'), 'name': 'Jennifer James', 'specialization': 'Private music teacher', 'contact_number': '996-525-9660x376', 'email': 'kmartin@example.net', 'availability': 'Friday'}
{'_id': ObjectId('6557c1572c9543a148610746'), 'name': 'Luis Bell', 'specialization': 'Surveyor, building control', 'contact_number': '7336908701', 'email': 'leslie36@example.com', 'availability': 'Tuesday'}
{'_id': ObjectId('6557c1572c9543a148610747'), 'name': 'Leah Glover MD', 'specialization': 'Rural practice surveyor', 'contact_number': '+1-296-840-4205x539', 'email': 'nparker@example.org', 'availability': 'Sunday'}
{'_id': ObjectId('6557c1572c9543a148610748'), 'name': 'David Osborn', 'specialization': 'Scientific laboratory technician', 'contact_number': '+1-547-985-3433', 'email': 'luis02@example.com', 'availability': 'Sunday'}
Appointments:
{'_id': ObjectId('6557c1572c9543a148610758'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a148610746'), 'appointment_date': '2023-02-20T04:12:55', 'purpose': 'Work start though somebody top may price.'}
{'_id': ObjectId('6557c1572c9543a148610759'), 'patient_id': ObjectId('6557c1572c9543a14861073f'), 'doctor_id': ObjectId('6557c1572c9543a148610745'), 'appointment_date': '2023-04-10T03:46:48', 'purpose': 'Center drop detail so training first city science.'}
{'_id': ObjectId('6557c1572c9543a14861075a'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a14861074c'), 'appointment_date': '2023-08-10T09:44:06', 'purpose': 'Agree modern million three political.'}
{'_id': ObjectId('6557c1572c9543a14861075b'), 'patient_id': ObjectId('6557c1572c9543a148610732'), 'doctor_id': ObjectId('6557c1572c9543a148610751'), 'appointment_date': '2023-02-19T09:24:41', 'purpose': 'Include data soon hit name feeling.'}
{'_id': ObjectId('6557c1572c9543a14861075c'), 'patient_id': ObjectId('6557c1572c9543a148610736'), 'doctor_id': ObjectId('6557c1572c9543a14861074d'), 'appointment_date': '2023-08-30T20:34:25', 'purpose': 'Out idea sister during ago evening spend.'}
Updating this ID's contact number with 123-456-7890: 656556c50e66ed1ff8adfa8b
Updating this ID's availability with Saturday: 6557c9971c5ee62b864307dc

task1.py task2.py task3.py task4.py task5.py

Task5 > task5.py > insert_random_data

```
1  from pymongo import MongoClient
2  from faker import Faker
3  import random
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - Task5

```
Appointments:
{'_id': ObjectId('6557c1572c9543a148610758'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a148610746'), 'appo
intment_date': '2023-02-20T04:12:55', 'purpose': 'Work start though somebody top may price.'}
{'_id': ObjectId('6557c1572c9543a148610759'), 'patient_id': ObjectId('6557c1572c9543a14861073f'), 'doctor_id': ObjectId('6557c1572c9543a148610745'), 'appo
intment_date': '2023-04-10T03:46:48', 'purpose': 'Center drop detail so training first city science.'}
{'_id': ObjectId('6557c1572c9543a14861075a'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a14861074c'), 'appo
intment_date': '2023-08-10T09:44:06', 'purpose': 'Agree modern million three political.'}
{'_id': ObjectId('6557c1572c9543a14861075b'), 'patient_id': ObjectId('6557c1572c9543a148610732'), 'doctor_id': ObjectId('6557c1572c9543a148610751'), 'appo
intment_date': '2023-02-19T09:24:41', 'purpose': 'Include data soon hit name feeling.'}
{'_id': ObjectId('6557c1572c9543a14861075c'), 'patient_id': ObjectId('6557c1572c9543a148610736'), 'doctor_id': ObjectId('6557c1572c9543a14861074d'), 'appo
intment_date': '2023-08-30T20:34:25', 'purpose': 'Out idea sister during ago evening spend.'}
Updating this ID's contact number with 123-456-7890: 656556c50e66ed1ff8adfa8b
Updating this ID's availability with Saturday: 6557c9971c5ee62b864307dc
Updated 1 patient(s) and 1 doctor(s).
Deleting patient_id : 6557c9971c5ee62b864307c2
Deleted 1 patient(s).
Deleting some_appointment_id : 6557c9971c5ee62b864307eb
Deleted 1 appointment(s).
Patients:
{'_id': ObjectId('6557c1572c9543a148610730'), 'name': 'Monica Rodriguez', 'date_of_birth': '1980-05-03', 'gender': 'M', 'address': '53059 Trevor Overpass\
nHermantown, NH 67534', 'contact_number': '225.374.5717', 'email': 'amandaryan@example.com'}
{'_id': ObjectId('6557c1572c9543a148610731'), 'name': 'Sarah Ayala', 'date_of_birth': '1962-06-01', 'gender': 'M', 'address': '818 Adrian Shoals\nNorth Ma
riastad, GU 62764', 'contact_number': '623-313-6433', 'email': 'pinedaheather@example.org'}
{'_id': ObjectId('6557c1572c9543a148610733'), 'name': 'John Brennan', 'date_of_birth': '2014-02-07', 'gender': 'M', 'address': '225 Benjamin Loaf Suite 41
9\nLake Kevinfort, AS 20294', 'contact_number': '+1-983-781-0675x7721', 'email': 'alicia61@example.org'}
{'_id': ObjectId('6557c1572c9543a148610734'), 'name': 'Sharon Martin', 'date_of_birth': '1939-12-03', 'gender': 'F', 'address': '4082 Paige Orchard\nGinam
outh, AZ 76749', 'contact_number': '235.744.5777x832', 'email': 'pclay@example.org'}
{'_id': ObjectId('6557c1572c9543a148610735'), 'name': 'Traci Wright', 'date_of_birth': '1927-08-03', 'gender': 'F', 'address': '031 Melissa Center Apt. 79
8\nEast Natashamouth, VA 12020', 'contact_number': '+1-896-828-8405x32142', 'email': 'brandoncarroll@example.org'}

Doctors:
{'_id': ObjectId('6557c1572c9543a148610744'), 'name': 'Tammy Bush', 'specialization': 'Health and safety adviser', 'contact_number': '230-353-2500x827', '
email': 'amandarodriguez@example.com', 'availability': 'Thursday'}
{'_id': ObjectId('6557c1572c9543a148610745'), 'name': 'Jennifer James', 'specialization': 'Private music teacher', 'contact_number': '996-525-9660x376', '
email': 'kmartin@example.net', 'availability': 'Friday'}
{'_id': ObjectId('6557c1572c9543a148610746'), 'name': 'Luis Bell', 'specialization': 'Surveyor, building control', 'contact_number': '7336908701', 'email'
: 'leslie36@example.com', 'availability': 'Tuesday'}
```

task1.py task2.py task3.py task4.py task5.py

Task5 > task5.py > insert_random_data

```
1  from pymongo import MongoClient
2  from faker import Faker
3  import random
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - Task5

Deleted 1 patient(s).
Deleting some_appointment_id : 6557c9971c5ee62b864307eb
Deleted 1 appointment(s).
Patients:
{'_id': ObjectId('6557c1572c9543a148610730'), 'name': 'Monica Rodriguez', 'date_of_birth': '1980-05-03', 'gender': 'M', 'address': '53059 Trevor Overpass\
nHermantown, NH 67534', 'contact_number': '225.374.5717', 'email': 'amandaryan@example.com'}
{'_id': ObjectId('6557c1572c9543a148610731'), 'name': 'Sarah Ayala', 'date_of_birth': '1962-06-01', 'gender': 'M', 'address': '818 Adrian Shoals\nNorth Ma
riastad, GU 62764', 'contact_number': '623-313-6433', 'email': 'pinedaheather@example.org'}
{'_id': ObjectId('6557c1572c9543a148610733'), 'name': 'John Brennan', 'date_of_birth': '2014-02-07', 'gender': 'M', 'address': '225 Benjamin Loaf Suite 41
9\nLake Kevinfort, AS 20294', 'contact_number': '+1-983-781-0675x7721', 'email': 'alicia61@example.org'}
{'_id': ObjectId('6557c1572c9543a148610734'), 'name': 'Sharon Martin', 'date_of_birth': '1939-12-03', 'gender': 'F', 'address': '4082 Paige Orchard\nGinam
outh, AZ 76749', 'contact_number': '235.744.5777x832', 'email': 'pclay@example.org'}
{'_id': ObjectId('6557c1572c9543a148610735'), 'name': 'Traci Wright', 'date_of_birth': '1927-08-03', 'gender': 'F', 'address': '031 Melissa Center Apt. 79
8\nEast Natashamouth, VA 12020', 'contact_number': '+1-896-828-8405x32142', 'email': 'brandoncarroll@example.org'}

Doctors:
{'_id': ObjectId('6557c1572c9543a148610744'), 'name': 'Tammy Bush', 'specialization': 'Health and safety adviser', 'contact_number': '230-353-2500x827', '
email': 'amandarodriguez@example.com', 'availability': 'Thursday'}
{'_id': ObjectId('6557c1572c9543a148610745'), 'name': 'Jennifer James', 'specialization': 'Private music teacher', 'contact_number': '996-525-9660x376', '
email': 'kmartin@example.net', 'availability': 'Friday'}
{'_id': ObjectId('6557c1572c9543a148610746'), 'name': 'Luis Bell', 'specialization': 'Surveyor, building control', 'contact_number': '7336908701', 'email'
: 'leslie36@example.com', 'availability': 'Tuesday'}
{'_id': ObjectId('6557c1572c9543a148610747'), 'name': 'Leah Glover MD', 'specialization': 'Rural practice surveyor', 'contact_number': '+1-296-840-4205x53
9', 'email': 'nparker@example.org', 'availability': 'Sunday'}
{'_id': ObjectId('6557c1572c9543a148610748'), 'name': 'David Osborn', 'specialization': 'Scientific laboratory technician', 'contact_number': '+1-547-985-
3433', 'email': 'luis02@example.com', 'availability': 'Sunday'}

Appointments:
{'_id': ObjectId('6557c1572c9543a148610758'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a148610746'), 'appo
intment_date': '2023-02-20T04:12:55', 'purpose': 'Work start though somebody top may price.'}
{'_id': ObjectId('6557c1572c9543a148610759'), 'patient_id': ObjectId('6557c1572c9543a14861073f'), 'doctor_id': ObjectId('6557c1572c9543a148610745'), 'appointment_da
te': '2023-04-10T03:46:48', 'purpose': 'Center drop detail so training first city science.'}
{'_id': ObjectId('6557c1572c9543a14861075a'), 'patient_id': ObjectId('6557c1572c9543a148610739'), 'doctor_id': ObjectId('6557c1572c9543a14861074c'), 'appointment_da
te': '2023-08-10T09:44:06', 'purpose': 'Agree modern million three political.'}
{'_id': ObjectId('6557c1572c9543a14861075b'), 'patient_id': ObjectId('6557c1572c9543a148610732'), 'doctor_id': ObjectId('6557c1572c9543a148610751'), 'appointment_da
te': '2023-02-19T09:24:41', 'purpose': 'Include data soon hit name feeling.'}
{'_id': ObjectId('6557c1572c9543a14861075c'), 'patient_id': ObjectId('6557c1572c9543a148610736'), 'doctor_id': ObjectId('6557c1572c9543a14861074d'), 'appointment_da
te': '2023-08-30T20:34:25', 'purpose': 'Out idea sister during ago evening spend.'}