Lab – 4

AIM: Perform MONGODB operations on Project Database.

Project Title: i-Medicare – The Hospital Management System

- Collection Schemas:

```
const userSchema = new Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['Admin', 'Doctor', 'Nurse', 'Patient', 'Staff'], required: true },
  email: { type: String, required: true, unique: true },
  created_at: { type: Date, default: Date.now },
  last_login: { type: Date },
  is_active: { type: Boolean, default: true },
});

const patientSchema = new Schema({
  first_name: String,
  last_name: String,
  date_of_birth: Date,
  gender: String,
  address: {
    street: String,
    city: String,
    state: String,
    zip_code: String,
  },
  contact_number: String,
  email: String,
  blood_group: String,
  medical_history: [{
    condition: String,
    date_diagnosed: Date,
    treatment: String,
  }],
  last_reports: {
    diabetes: String,
    blood_pressure: String,
  },
});
```

```
const doctorSchema = new Schema({
  first_name: String,
  last_name: String,
  specialization: String,
  contact_number: String,
  email: String,
  date_of_birth: Date,
  gender: String,
  years_of_experience: Number,
  address: {
    street: String,
    city: String,
    state: String,
    zip_code: String,
  },
  blood_group: String,
  patients: [{ type: Schema.Types.ObjectId, ref: 'Patient' }],
  appointments: [{ type: Schema.Types.ObjectId, ref: 'Appointment' }],
});

const appointmentSchema = new Schema({
  patient_id: { type: Schema.Types.ObjectId, ref: 'Patient' },
  doctor_id: { type: Schema.Types.ObjectId, ref: 'Doctor' },
  appointment_date: Date,
  reason_for_visit: String,
  prescription: String,
  status: String,
});

const billingSchema = new Schema({
  patient_id: { type: Schema.Types.ObjectId, ref: 'Patient' },
  doctor_id: { type: Schema.Types.ObjectId, ref: 'Doctor' },
  appointment_id: { type: Schema.Types.ObjectId, ref: 'Appointment' },
  total_amount: Number,
  date_issued: Date,
  description: String,
  paid: Boolean,
});

const staffSchema = new Schema({
  first_name: String,
  last_name: String,
  role: String,
  contact_number: String,
  email: String,
  shift_timings: String,
```

```
});

const roomSchema = new Schema({
  room_number: String,
  room_type: String,
  availability_status: Boolean,
  assigned_patient_id: { type: Schema.Types.ObjectId, ref: 'Patient' },
});

const medicalRecordSchema = new Schema({
  patient_id: { type: Schema.Types.ObjectId, ref: 'Patient' },
  record_type: String,
  record_date: Date,
  description: String,
});

const organDonationSchema = new Schema({
  donor_id: { type: Schema.Types.ObjectId, ref: 'Patient' },
  organ_type: String,
  date_of_donation: Date,
  status: { type: String, enum: ['Pending', 'Donated', 'Matched'], default: 'Pending' },
  recipient_id: { type: Schema.Types.ObjectId, ref: 'Patient', default: null },
  notes: String,
});

const financeSchema = new Schema({
  transaction_type: { type: String, enum: ['Income', 'Expense'], required: true },
  amount: { type: Number, required: true },
  description: { type: String, required: true },
  date: { type: Date, default: Date.now },
  category: { type: String, required: true },
});

const User = mongoose.model('User', userSchema);
const Patient = mongoose.model('Patient', patientSchema);
const Doctor = mongoose.model('Doctor', doctorSchema);
const Appointment = mongoose.model('Appointment', appointmentSchema);
const Billing = mongoose.model('Billing', billingSchema);
const Staff = mongoose.model('Staff', staffSchema);
const Room = mongoose.model('Room', roomSchema);
const MedicalRecord = mongoose.model('MedicalRecord', medicalRecordSchema);
const OrganDonation = mongoose.model('OrganDonation', organDonationSchema);
const Finance = mongoose.model('Finance', financeSchema);
```

- Insert Operation on Collections

```
const mongoose = require('mongoose');
const { User, Patient, Doctor, Appointment, Billing, Staff, Room, MedicalRecord,
OrganDonation, Finance } = require('./models');

const newUser = new User({
   username: 'john_doe',
   password: "Password@123",
   role: 'Doctor',
   email: 'john.doe@example.com',
});

newUser.save();

const newPatient = new Patient({
   first_name: 'John',
   last_name: 'Doe',
   date_of_birth: new Date('1990-01-01'),
   gender: 'Male',
   address: {
     street: '123 Main St',
     city: 'Somewhere',
     state: 'CA',
     zip_code: '90210',
   },
   contact_number: '123-456-7890',
   email: 'john.doe@example.com',
   medical_history: [
     { condition: 'Diabetes', date_diagnosed: new Date('2010-05-15'), treatment: 'Insulin'
},
   ],
   last_report: {
      diabetes: "normal",
      blood_pressure: "normal"
   }
  });

newPatient.save();

const newDoctor = new Doctor({
   first_name: 'Jane',
   last_name: 'Smith',
   specialization: 'Cardiology',
   contact_number: '987-654-3210',
   email: 'jane.smith@example.com',
```

```
    years_of_experience: 10,
  });

newDoctor.save();

const patient = Patient.findOne({ first_name: 'John', last_name: 'Doe' });
console.log(patient);

const doctor = Doctor.findOne({ last_name: 'Smith' });
console.log(doctor);

const newAppointment = new Appointment({
    patient_id: patient._id,
    doctor_id: doctor._id,
    appointment_date: new Date('2024-08-10'),
    reason_for_visit: 'Regular Checkup',
  });

newAppointment.save();

const appointment = Appointment.findOne({ patient_id: patient._id });
console.log(appointment);

const newStaff = new Staff({
    first_name: 'Emily',
    last_name: 'Davis',
    role: 'Nurse',
    contact_number: '555-555-5555',
    email: 'emily.davis@example.com',
    shift_timings: 'Night Shift',
  });

newStaff.save();

const newRoom = new Room({
    room_number: '101',
    room_type: 'Private Room',
    availability_status: true,
  });

newRoom.save();

const newOrganDonation = new OrganDonation({
    donor_id: patient._id,
    organ_type: 'Kidney',
    date_of_donation: new Date('2024-08-01'),
```

```
    status: 'Pending',
    notes: 'Potential match for transplant.',
  });

newOrganDonation.save();

const newExpense = new Finance({
    transaction_type: 'Expense',
    amount: 5000,
    description: 'Purchase of medical supplies',
    category: 'Medical Supplies',
  });

newExpense.save();

const newIncome = new Finance({
    transaction_type: 'Income',
    amount: 10000,
    description: 'Payment received from patient John Doe',
    category: 'Patient Payment',
});

newIncome.save();

const newBilling = new Billing({
  patient_id: patient._id,
  doctor_id: doctor._id,
  appointment_id: appointment._id,
  total_amount: 200.00,
  date_issued: new Date(),
  paid: false,
});

newBilling.save();

const newMedicalRecord = new MedicalRecord({
  patient_id: patient._id,
  record_type: 'X-ray',
  record_date: new Date('2024-07-20'),
  description: 'Chest X-ray',
  file_url: 'https://example.com/xray/chest-xray.jpg',
});

newMedicalRecord.save();
```

➢ Queries on Collection

```
> db.finances.updateOne({_id: ObjectId('66b5070bde13414147c5dc80')},{$set: {amount: 12000}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.finances.find({_id: ObjectId('66b5070bde13414147c5dc80')})
< {
    _id: ObjectId('66b5070bde13414147c5dc80'),
    transaction_type: 'Income',
    amount: 12000,
    description: 'Payment received from patient John Doe',
    category: 'Patient Payment',
    date: 2024-08-08T17:57:31.888Z,
    __v: 0
  }
i_medicare>
```

```
> db.appointments.updateOne({_id: ObjectId('66b60d4cf31b5f419bfbe9c8')}, {$set: {status: 'pending'}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.appointments.find({_id: ObjectId('66b60d4cf31b5f419bfbe9c8')})
< {
    _id: ObjectId('66b60d4cf31b5f419bfbe9c8'),
    appointment_date: 2024-08-10T00:00:00.000Z,
    reason_for_visit: 'Regular Checkup',
    __v: 0,
    status: 'pending'
  }
i_medicare>
```

```
> db.medicalrecords.deleteOne({_id: ObjectId('66b610377564f1952124c74a')})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.medicalrecords.find({_id: ObjectId('66b610377564f1952124c74a')})
<
i_medicare>
```

```
> db.staffs.deleteOne({name:'Emily'})
< {
    acknowledged: true,
    deletedCount: 0
  }
> db.staffs.find({name:'Emily'})
<
i_medicare>
```

```
> db.finances.aggregate({$group: { _id: '$transaction_type', total_amount: {$sum: '$amount'} }})
< {
    _id: 'Income',
    total_amount: 12000
  }
  {
    _id: 'Expense',
    total_amount: 5000
  }
i_medicare>
```

```
>_MONGOSH
> db.patients.aggregate([ { $group: { _id: "$gender", totalPatients: { $sum: 1 } } } ])
< {
    _id: 'Male',
    totalPatients: 1
  }
i_medicare>
```

```
> db.doctors.findOneAndDelete({first_name: 'Emily'})
< {
    _id: ObjectId('66b5070bde13414147c5dc7c'),
    first_name: 'Emily',
    last_name: 'Smith',
    specialization: 'Cardiology',
    contact_number: '987-654-3210',
    email: 'jane.smith@example.com',
    years_of_experience: 10,
    patients: [
      '66b5070bde13414147c5dc79'
    ],
    appointments: [
      '66b60d4cf31b5f419bfbe9c8'
    ],
    __v: 0
  }
> db.doctors.find({first_name: 'Emily'})
<
i_medicare>
```

```
>_MONGOSH
> db.patients.aggregate([
    { $unwind: "$medical_history" },
    {
      $group: {
        _id: "$medical_history.condition",
        count: { $sum: 1 }  // Corrected syntax
      }
    },
    { $sort: { count: -1 } },
    { $limit: 5 }
  ])
< {
    _id: 'Diabetes',
    count: 1
  }
i_medicare>
```