```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract patientRecord
{
    address Owner;

    constructor()
    {
        Owner = msg.sender;
    }

    struct doctor
    {
        uint   docId;
        string name;
        string qualification;
        string workPlace;
    }

    struct patient
    {
        uint   patientId;
        string name;
        uint   age;

    }

    struct disease
    {   uint   patientId;
        string diseaseName;
    }

    struct medicine
    {
        uint   medicineId;
        string medicineName;
        string expiryDate;
        string dose;
        uint   medicinePrice;
    }

    struct prescribedMedicine
    {
        uint    patientId;
        uint    medicineId;
    }


    mapping(uint=>doctor) doctorMap;
    mapping(uint=>patient) patientMap;
    mapping(uint=>disease[]) diseaseMap;
    mapping(uint=>medicine) medicineMap;     //Add medicine details
    mapping(uint => prescribedMedicine[]) prescribedMap; //Add prescribed medicineId to Patient..
```

```
modifier onlyOwner()
  {
     require(msg.sender == Owner, "!!! Not the Patient !!!");
     _;
  }

  //No.1>>>!.....Function to register a new doctor.....!

  function registerDoctor(uint _docId, string memory _name,
                          string memory _qualification,
                          string memory _workPlace
                          )public
  {
     doctorMap[_docId] = doctor(_docId,_name,_qualification,_workPlace);
  }

  //No.2>>>!.....Function to register a new patient.....!

  function registerPatient(uint _patientId,string memory _name, uint _age) public
  {
    patientMap[_patientId] = patient(_patientId,_name,_age);
  }

  //No.3>>>!.....Function to add new disease.....
  function addNewDisease(uint _patientId,string memory _diseaseName) public
  {
     disease memory newDisease = disease(_patientId,_diseaseName);
      diseaseMap[_patientId].push(newDisease);
  }

  //No.4>>>!.....Function to add new medicine.....!

  function addMedicine( uint _medicineId, string memory _name,
                        string memory _expiryDate, string memory _dose,
                        uint _price
                       ) public
  {
     medicineMap[_medicineId] = medicine(_medicineId,_name,_expiryDate,_dose,_price);

  }

  //No.5>>>!.....Function to add prescribed medicine.....!

  function prescribeMedicine(uint _patientId, uint _medicineId) public
  {
     prescribedMedicine memory newprescribedMedicine = prescribedMedicine(_patientId,_medicineId);
     prescribedMap[_patientId].push(newprescribedMedicine);
  }

  //No.6>>>!.....Function to updateAge.....!

  function updateAge(uint _pId, uint _age) public onlyOwner
  {
     patientMap[_pId].age = _age;
  }
```

```
    //No.7>>>!....Funciton to view patient data from blockchain......!
 function viewPatientData(uint _patientId)public view onlyOwner returns(patient memory, disease[] memor
y)
 {
  return(patientMap[_patientId], diseaseMap[_patientId]);
 }

    //No.8>>>!....Funciton to view Medicine Details......!
    function viewMedicine(uint _id) public view returns(medicine memory)
    {
        return(medicineMap[_id]);
    }

    //No.9>>>!....Funciton to view patient data by doctor......!
 function patientDataByDoc(uint _patientId)public view returns(patient memory, disease[] memory)
 {
  return(patientMap[_patientId], diseaseMap[_patientId]);
 }

    //No.10>>>!....Funciton to view prescribed medicine to patient by Doctor......!

    function viewPrescribedMedicine(uint _patientId) public view returns(prescribedMedicine[] memory)
    {
        return(prescribedMap[_patientId]);
    }


    //No.11>>>!....Funciton to view doctor details......!
    function viewDoctorDetails(uint _docId) public view returns(doctor memory)
    {
        return(doctorMap[_docId]);
    }


}
```