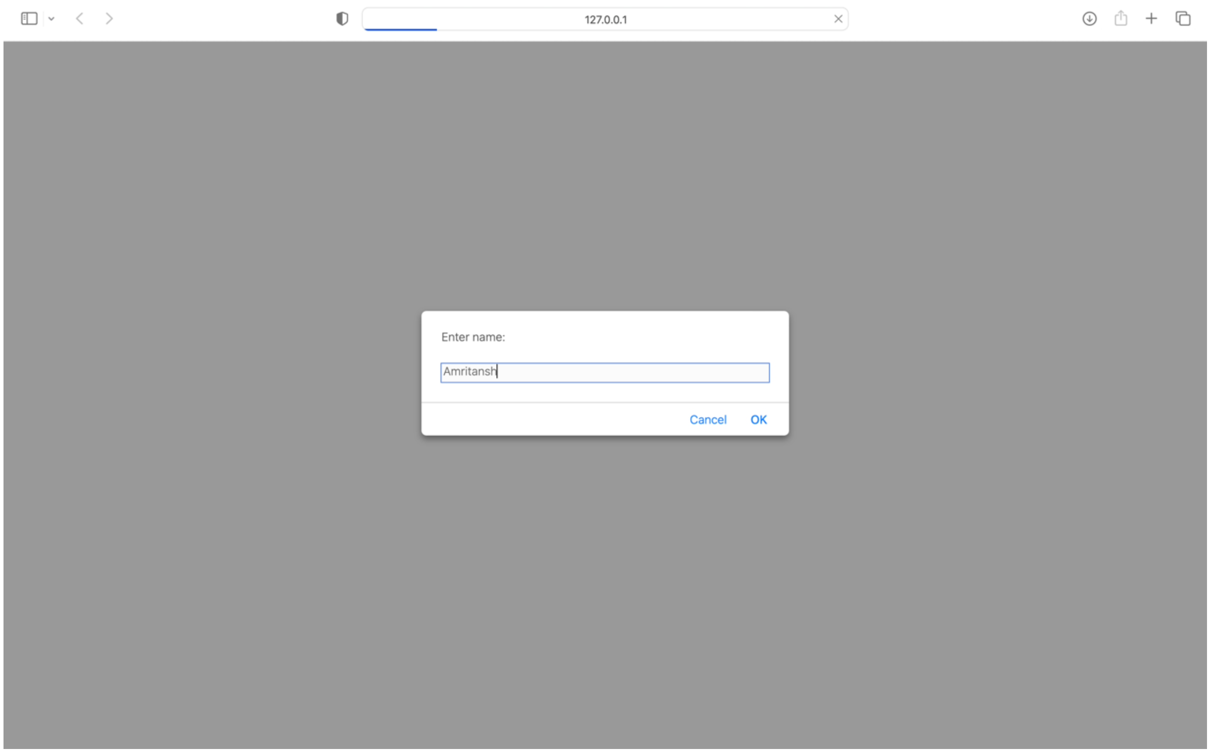
Name: AMRITANSH ANAND
Reg. No.: 20BCE1650

IWP-DA3

1. Selfhelp is an attendance management system at VIT. For a given employee get his name, entry date and time and exit date and time of a day as Javascript date object. If the hours in between is 8 hours mark as "present" else throw an exception stating "absent".

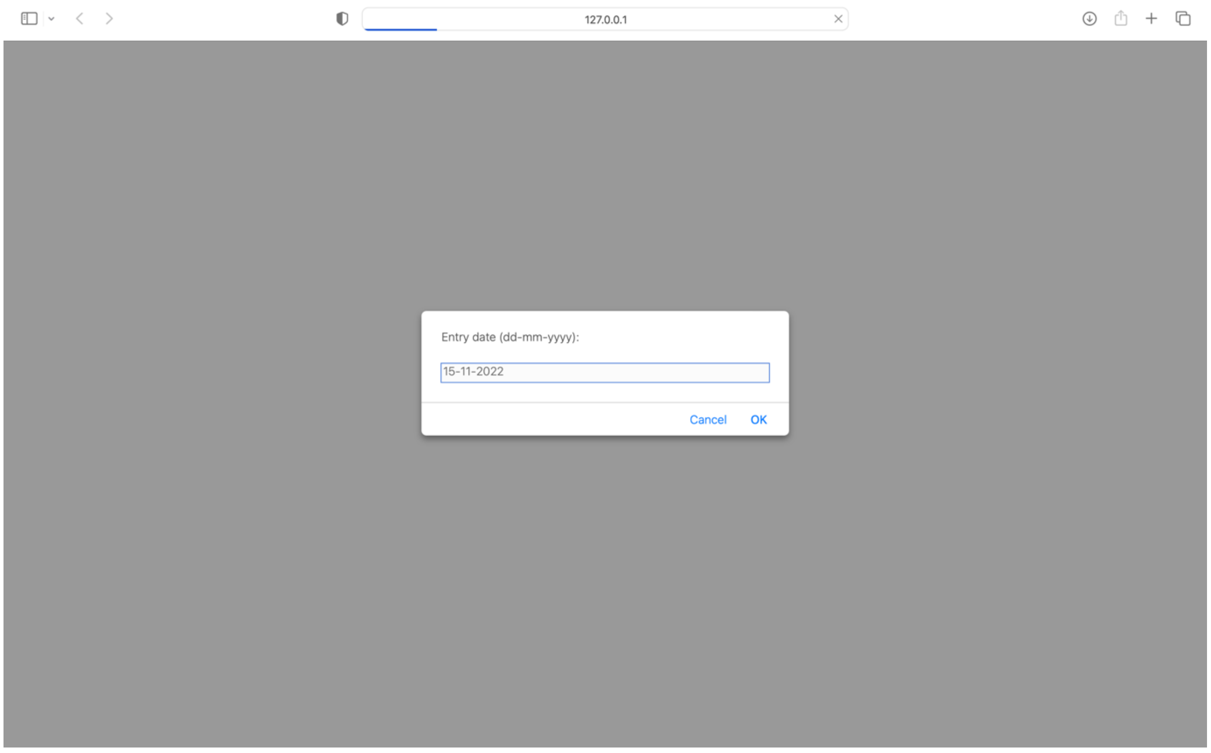
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        var x=prompt("Enter name:");
        var y=prompt("Entry date (dd-mm-yyyy):");
        var yt=prompt("Entry time (hh:mm):");
        var a=y.split("-");
        var at=yt.split(":");
        var z=prompt("Exit date(dd-mm-yyyy):");
        var zt=prompt("Exit time (hh:mm):");
        var b=z.split("-");
        var bt=zt.split(":");
        const d1 = new Date(a[2], a[1], a[0], at[0], at[1]);
        const d2 = new Date(b[2], b[1], b[0], bt[0], bt[1]);
        var diff = d2.getTime() - d1.getTime();
        var actualdiff=diff/(1000*60*60);
        var actualtime=8;
        try{
            if(diff>=actualtime){
                alert(x+" PRESENT");
            }
            else{
                alert(x+" ABSENT");
            }
        }
        catch(Error){
            document.write("Error found: "+e.message);
        }
    </script>
</head>
<body>
```

```
</body>
</html>
```


127.0.0.1

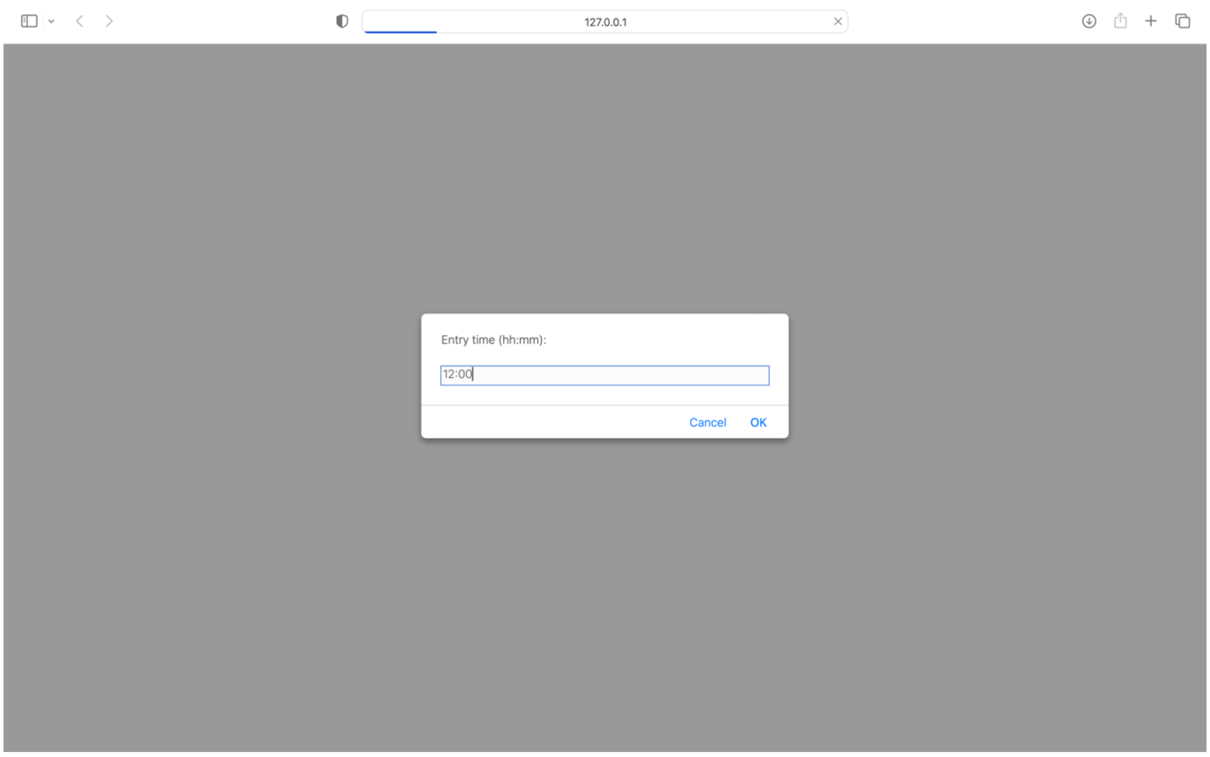
Enter name:

Amritansh


127.0.0.1

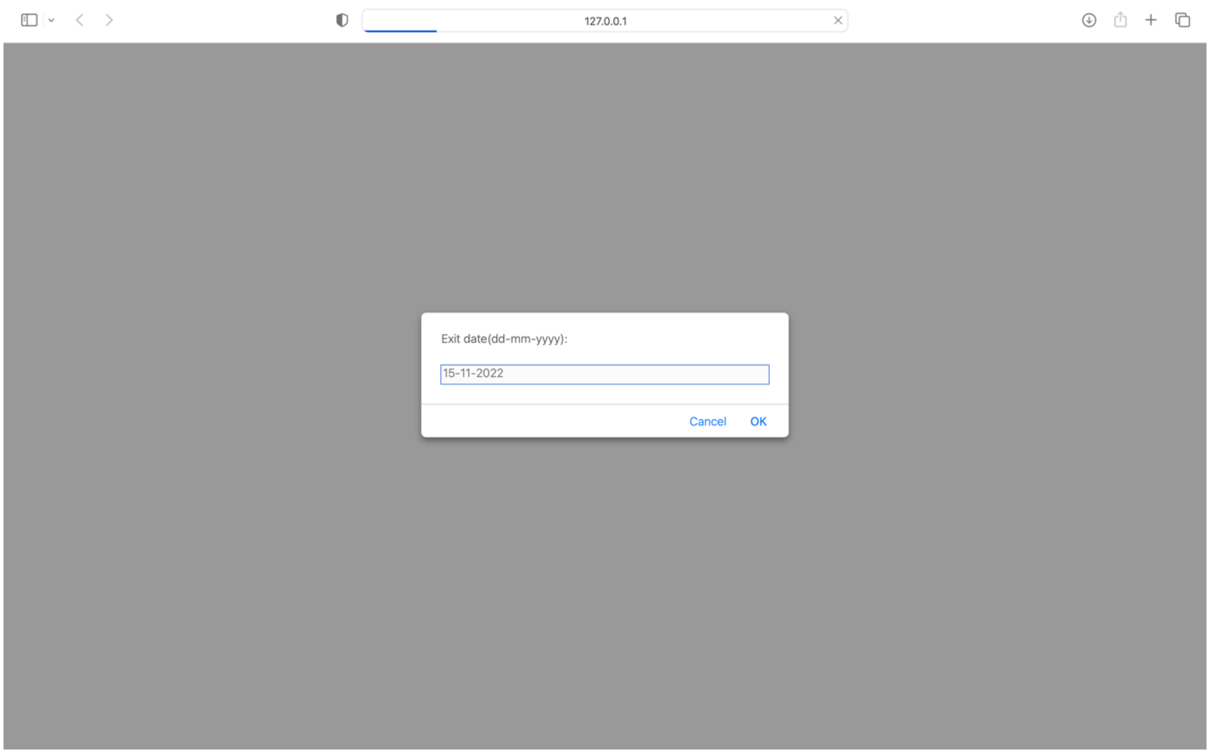
Entry date (dd-mm-yyyy):

15-11-2022



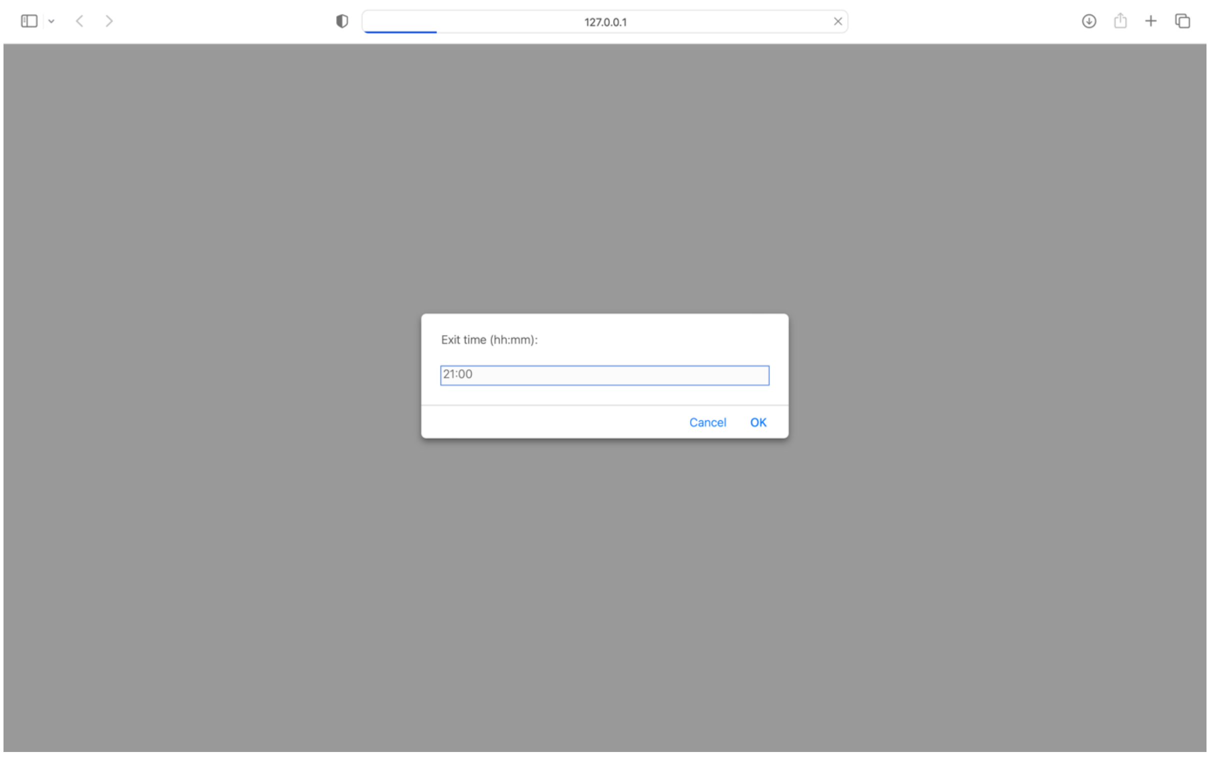
Entry time (hh:mm):

12:00



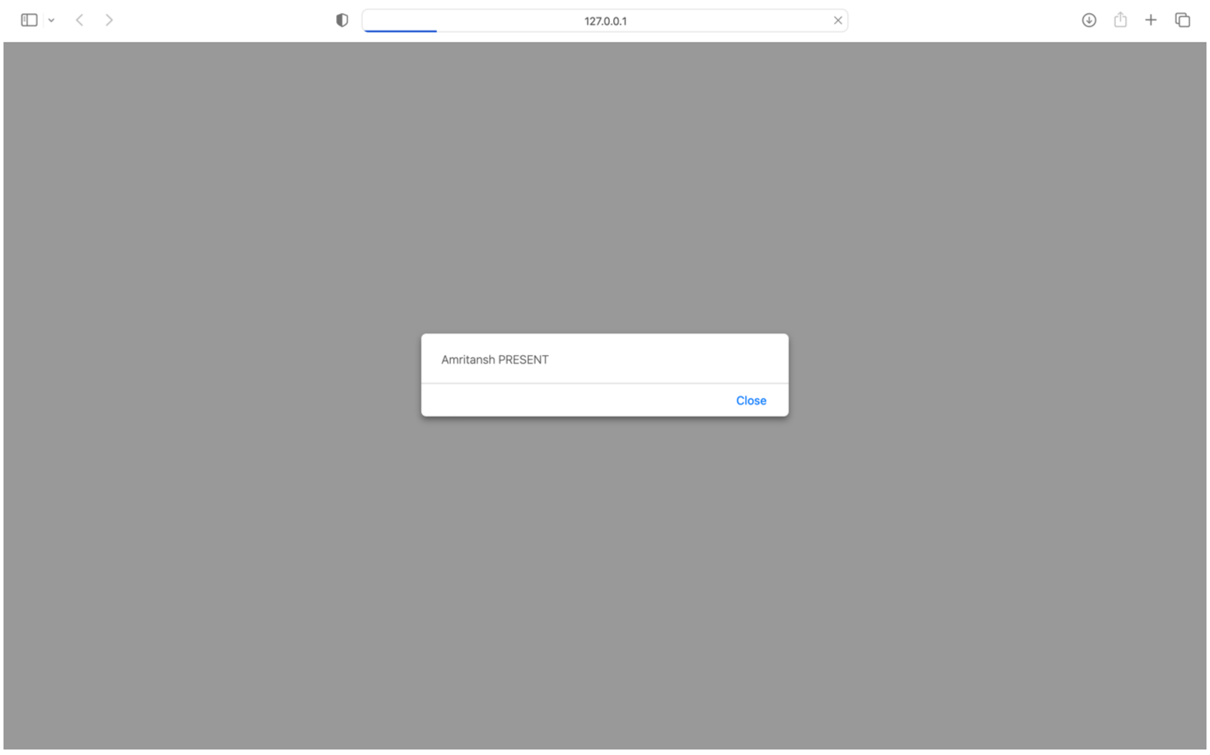
Exit date(dd-mm-yyyy):

15-11-2022


127.0.0.1

Exit time (hh:mm):

21:00


127.0.0.1

Amritansh PRESENT

2. Create a XML file for movies. Apply CSS for better presentation on the XML file. A movie has title, genere, release year, language. Use XSD to provide a choice of language can be tamil, English and hindi. Use XSD to provide check if release date is in date format using regular expression. Also ensure release year must be with in 2000-2022.

movies.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<!DOCTYPE movies SYSTEM "movies.dtd">
<movies>
    <moviename>
        <title>Uri</title>
        <genre>Thriller</genre>
        <releasedate>2019-01-10</releasedate>
        <releaseyear>2019</releaseyear>
        <language>Hindi</language>
    </moviename>
    <moviename>
        <title>Lootcase</title>
        <genre>Thriller</genre>
        <releasedate>2020-07-31</releasedate>
        <releaseyear>2020</releaseyear>
        <language>English</language>
    </moviename>
        <moviename>
        <title>Pushpa</title>
        <genre>Action</genre>
        <releasedate>2021-12-17</releasedate>
        <releaseyear>2021</releaseyear>
        <language>Tamil</language>
    </moviename>
</movies>
```

movies.dtd

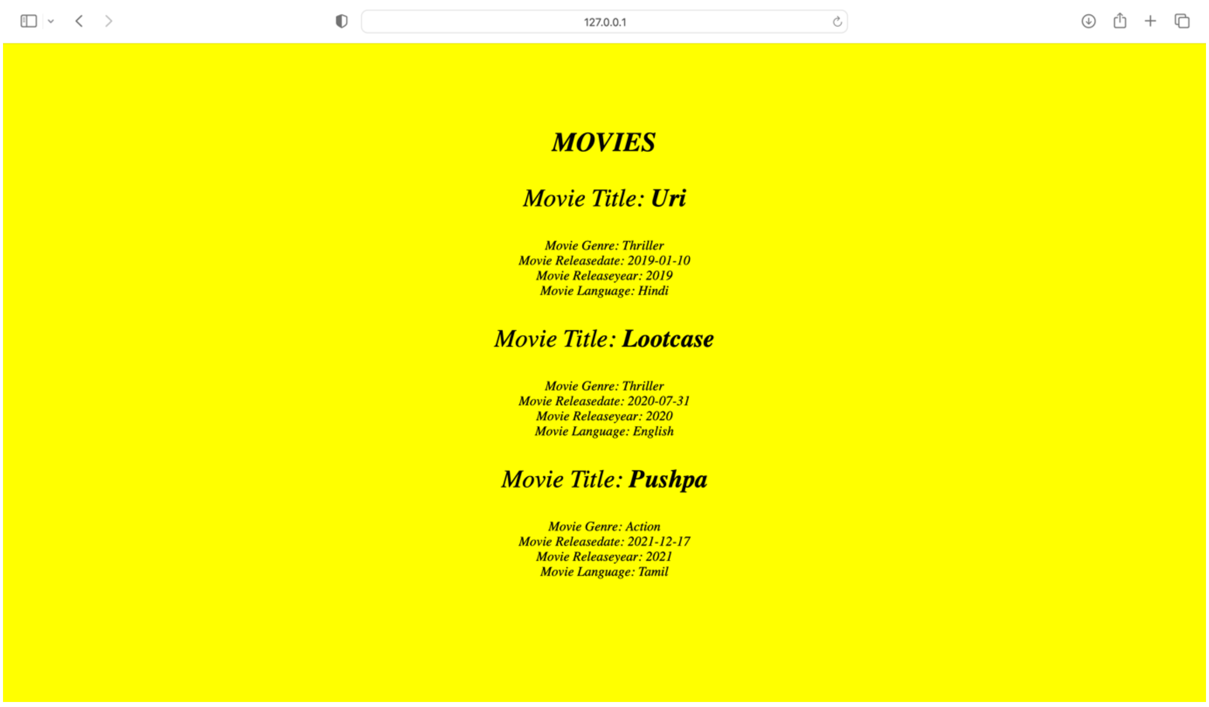
```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT movies ((moviename)+)>
<!ELEMENT moviename (title,genre,releasedate,releaseyear,language)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
<!ELEMENT releasedate (#PCDATA)>
<!ELEMENT releaseyear (#PCDATA)>
<!ELEMENT language (#PCDATA)>
```

style.xsl

```xml
<?xml version="1.0" encoding="UTF-8"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<body style="background-color: yellow;text-align:center;font-style:italic;margin-top:100px">
<h1>MOVIES</h1>
<xsl:for-each select="movies/moviename">
<p style="font-size:30px;">Movie Title: <B><xsl:value-of select="title"/></B></p>
<div>Movie Genre: <xsl:value-of select="genre"/></div>
<div>Movie Releasedate: <xsl:value-of select="releasedate"/></div>
<div>Movie Releaseyear: <xsl:value-of select="releaseyear"/></div>
<div>Movie Language: <xsl:value-of select="language"/></div>
</xsl:for-each>
</body>
</html>
```

movies.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="movies.xml"
elementFormDefault="qualified">
<xs:element name="movies">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="moviename" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="title" type="xs:string"/>
                        <xs:element name="genre" type="xs:string"/>
                        <xs:element name="releasedate">
                        <xs:simpleType>
                            <xs:restriction base="xs:date">
                                <xs:pattern value="[0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-
9]"/>
                            </xs:restriction>
                        </xs:simpleType>
                        </xs:element>
                        <xs:element name="releaseyear">
                        <xs:simpleType>
                            <xs:restriction base="xs:integer">
                                <xs:minInclusive value="2000"/>
                                <xs:maxInclusive value="2022"/>
                            </xs:restriction>
                        </xs:simpleType>
                        </xs:element>
                        <xs:element name="language">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="Hindi"/>
                                <xs:enumeration value="English"/>
                                <xs:enumeration value="Tamil"/>
                            </xs:restriction>
                        </xs:simpleType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```



127.0.0.1

MOVIES

*Movie Title: **Uri***

Movie Genre: Thriller
Movie Releasedate: 2019-01-10
Movie Releaseyear: 2019
Movie Language: Hindi

*Movie Title: **Lootcase***

Movie Genre: Thriller
Movie Releasedate: 2020-07-31
Movie Releaseyear: 2020
Movie Language: English

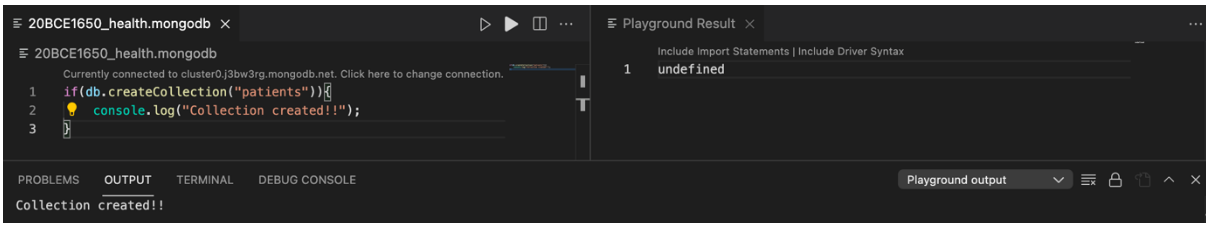
*Movie Title: **Pushpa***

Movie Genre: Action
Movie Releasedate: 2021-12-17
Movie Releaseyear: 2021
Movie Language: Tamil

3. Write a simple application using node js and mongo db for a health cube. Create a collection for patients. Insert patient details. Fetch a patient record based on patient id. Fetch all patient records whose name starts with ‘A’. Update patient age to 40 for a patient with patient id:123. Sort the patient ids based on age and display. Find the patient name who has paid maximum fee. Delete record of a patient based on a certain patient id.

Creating a collection patients

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
    if(err) throw err;
    var dbo = db.db("health");
    dbo.createCollection("patients",function(err,res){
        if(err) throw err;
        console.log("Collection Created!!");
        db.close();
    })
})
```



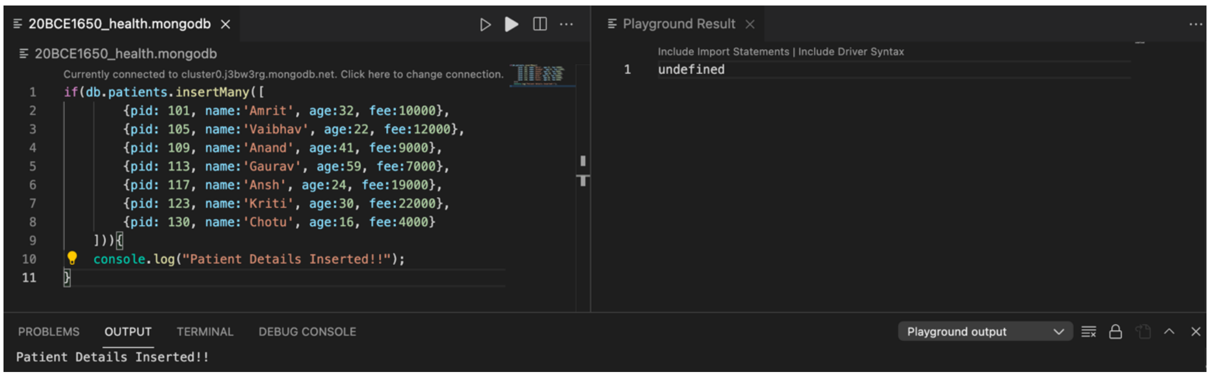
```
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change connection.
if(db.createCollection("patients")){
    console.log("Collection created!!");
}
```

```
Include Import Statements | Include Driver Syntax
undefined
```

```
Collection created!!
```

Inserting patient details

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
    if(err) throw err;
    var dbo = db.db("health");
    var details=[
        {pid: 101, name:'Amrit', age:32, fee:10000},
        {pid: 105, name:'Vaibhav', age:22, fee:12000},
        {pid: 109, name:'Anand', age:41, fee:9000},
        {pid: 113, name:'Gaurav', age:59, fee:7000},
        {pid: 117, name:'Ansh', age:24, fee:19000},
        {pid: 123, name:'Kriti', age:30, fee:22000},
        {pid: 130, name:'Chotu', age:16, fee:4000}
    ];
    dbo.collection("patients").insertMany(details,function(err,res){
        if(err) throw err;
        console.log("Number of patients inserted: "+res.insertedCount);
        db.close();
    })
})
```



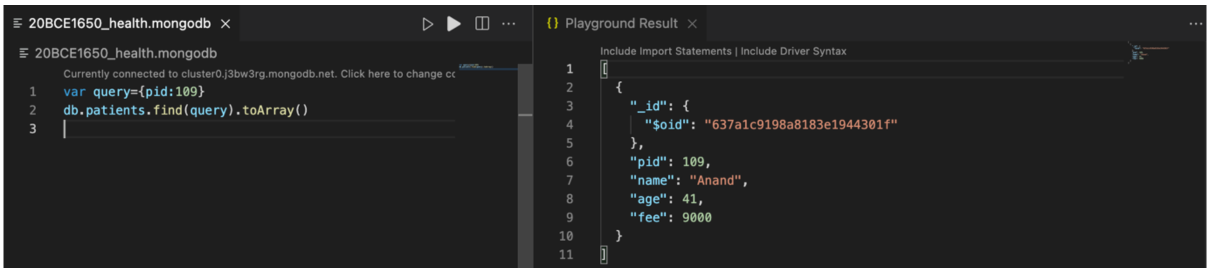
```
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change connection.
if(db.patients.insertMany([
        {pid: 101, name:'Amrit', age:32, fee:10000},
        {pid: 105, name:'Vaibhav', age:22, fee:12000},
        {pid: 109, name:'Anand', age:41, fee:9000},
        {pid: 113, name:'Gaurav', age:59, fee:7000},
        {pid: 117, name:'Ansh', age:24, fee:19000},
        {pid: 123, name:'Kriti', age:30, fee:22000},
        {pid: 130, name:'Chotu', age:16, fee:4000}
    ])){
    console.log("Patient Details Inserted!!");
}
```

```
Include Import Statements | Include Driver Syntax
undefined
```

```
Patient Details Inserted!!
```

Fetching a patient based on patient id

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
   if(err) throw err;
   var dbo = db.db("health");
   var query={pid:109};
   dbo.collection("patients").find(query).toArray(function(err,result){
      if(err) throw err;
      console.log(result);
      db.close();
   })
})
```



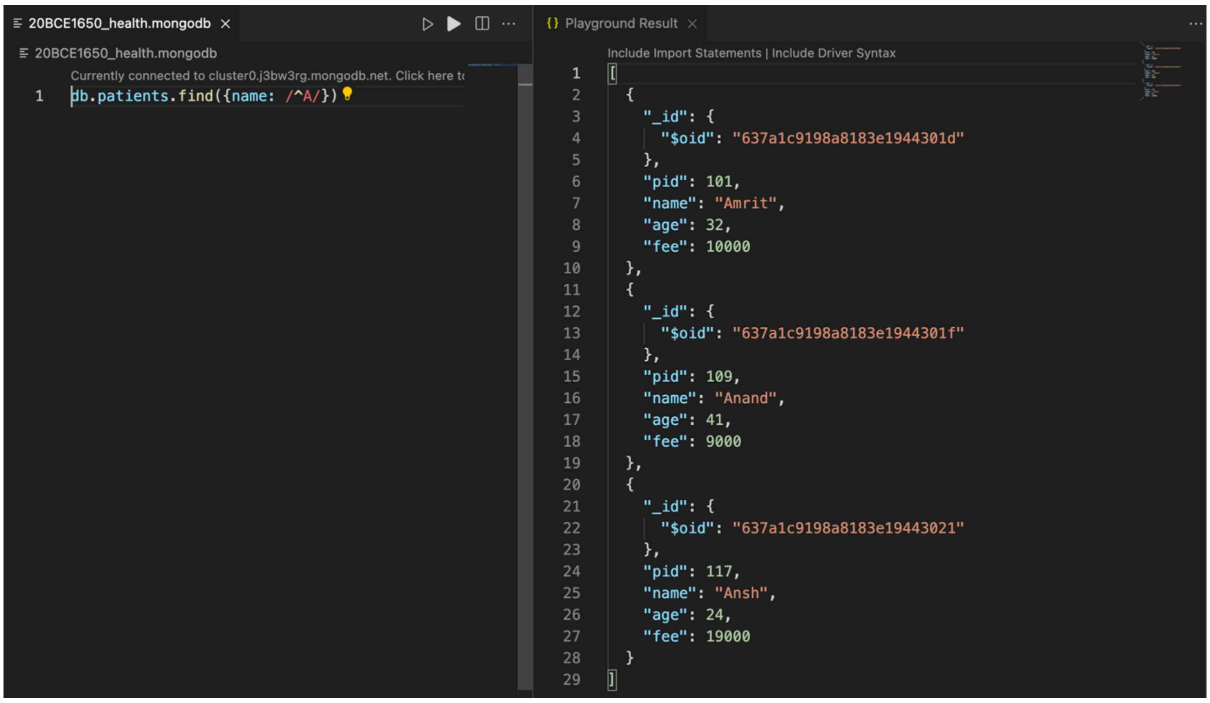
20BCE1650_health.mongodb
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc

```
var query={pid:109}
db.patients.find(query).toArray()
```

```
[
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301f"
    },
    "pid": 109,
    "name": "Anand",
    "age": 41,
    "fee": 9000
  }
]
```

Fetching all patients whose name starts with ‘A’

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
   if(err) throw err;
   var dbo = db.db("health");
   dbo.collection("patients").find({name:
/^S/}).toArray(function(err,result){
      if(err) throw err;
      console.log(result);
      db.close();
   })
})
```



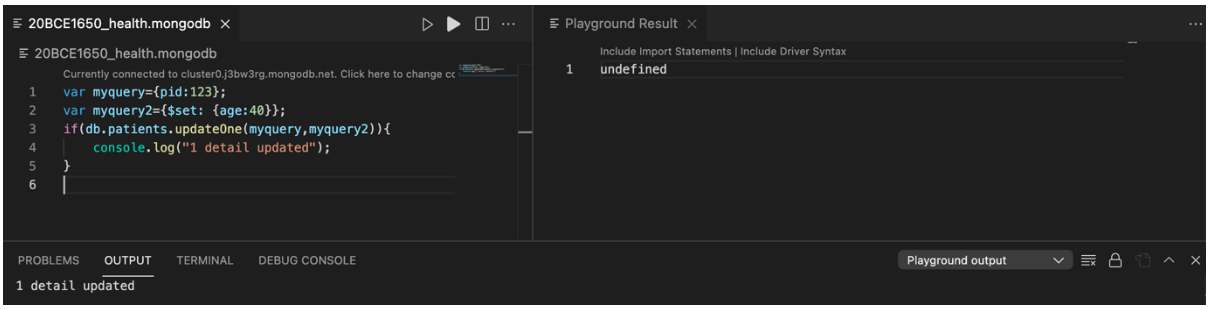
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here t

```
db.patients.find({name: /^A/})
```

```
[
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301d"
    },
    "pid": 101,
    "name": "Amrit",
    "age": 32,
    "fee": 10000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301f"
    },
    "pid": 109,
    "name": "Anand",
    "age": 41,
    "fee": 9000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443021"
    },
    "pid": 117,
    "name": "Ansh",
    "age": 24,
    "fee": 19000
  }
]
```

Update patient age to 40 for patient id 123

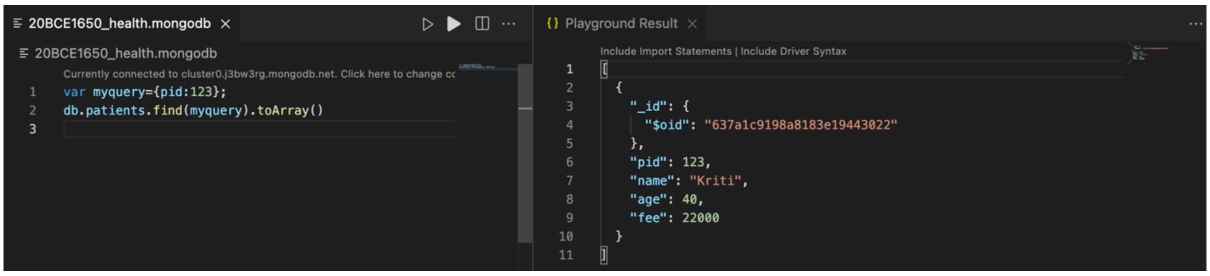
```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
    if(err) throw err;
    var dbo = db.db("health");
    var myquery={pid:123};
    var myquery2={$set: {age:40}};

dbo.collection("patients").updateOne(myquery,myquery2,function(err
,res){
        if(err) throw err;
        console.log("1 detail updated");
        db.close();
    })
})
```



```
20BCE1650_health.mongodb
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc
var myquery={pid:123};
var myquery2={$set: {age:40}};
if(db.patients.updateOne(myquery,myquery2)){
    console.log("1 detail updated");
}
```

```
Playground Result
Include Import Statements | Include Driver Syntax
undefined
```

```
1 detail updated
```

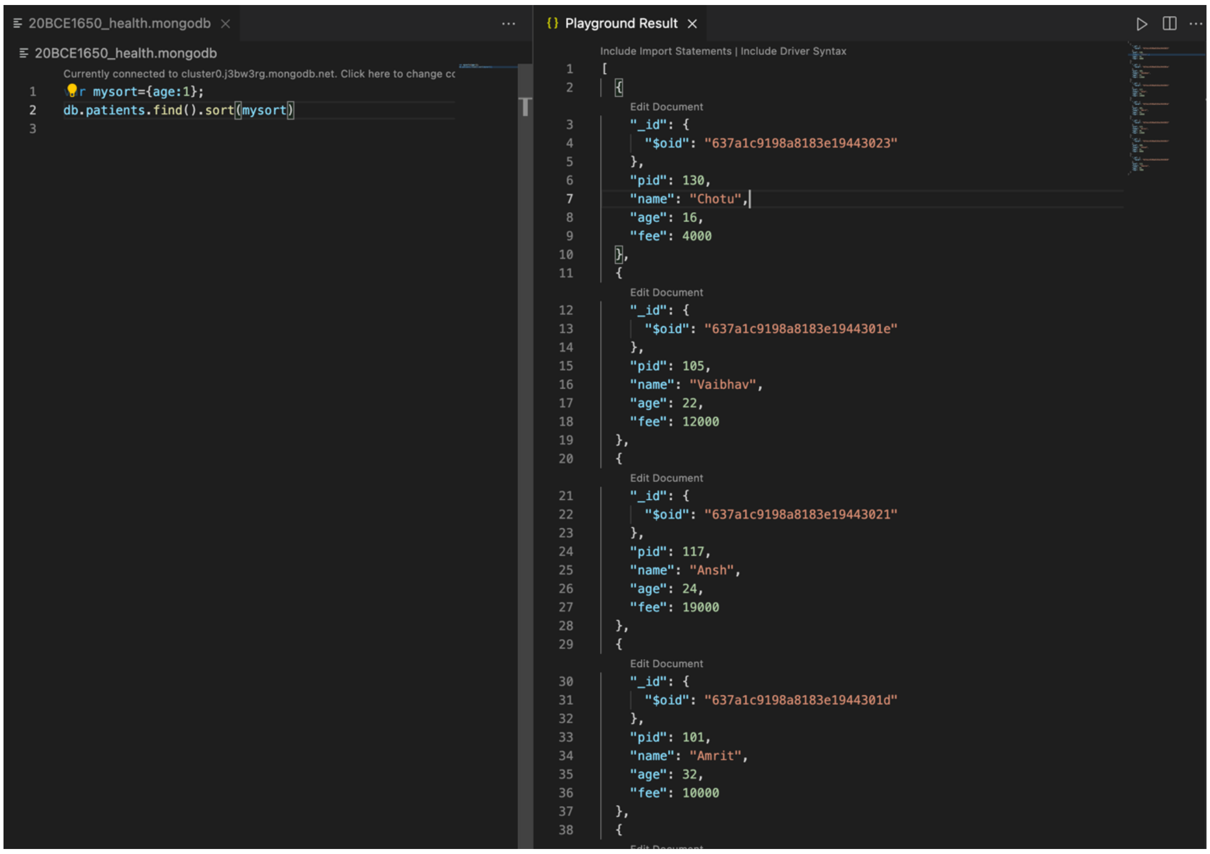


```
20BCE1650_health.mongodb
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc
var myquery={pid:123};
db.patients.find(myquery).toArray()
```

```
Playground Result
Include Import Statements | Include Driver Syntax
[
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443022"
    },
    "pid": 123,
    "name": "Kriti",
    "age": 40,
    "fee": 22000
  }
]
```

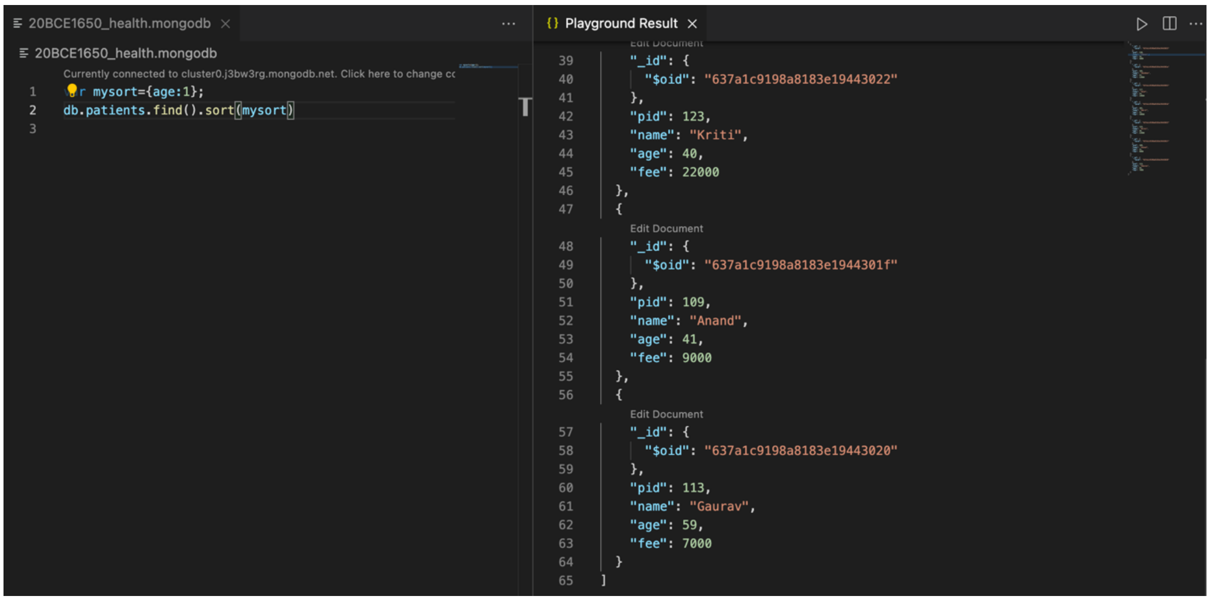
Sorting the patient ids based on age

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
   if(err) throw err;
   var dbo = db.db("health");
   var mysort={age:1};

dbo.collection("patients").find().sort(mysort).toArray(function(err,res
){
       if(err) throw err;
       console.log(res);
       db.close();
   })
})
```



```
20BCE1650_health.mongodb
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc
r mysort={age:1};
db.patients.find().sort(mysort)
```

```
[
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443023"
    },
    "pid": 130,
    "name": "Chotu",
    "age": 16,
    "fee": 4000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301e"
    },
    "pid": 105,
    "name": "Vaibhav",
    "age": 22,
    "fee": 12000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443021"
    },
    "pid": 117,
    "name": "Ansh",
    "age": 24,
    "fee": 19000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301d"
    },
    "pid": 101,
    "name": "Amrit",
    "age": 32,
    "fee": 10000
  },
  {
```



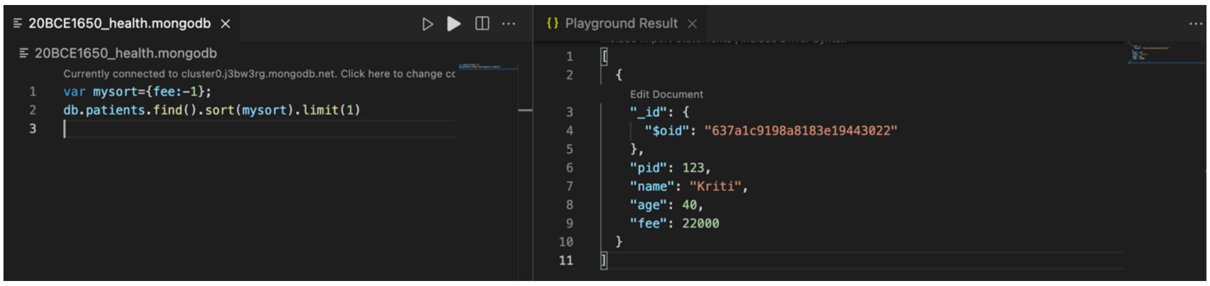
```
20BCE1650_health.mongodb
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc
r mysort={age:1};
db.patients.find().sort(mysort)
```

```
    "_id": {
      "$oid": "637a1c9198a8183e19443022"
    },
    "pid": 123,
    "name": "Kriti",
    "age": 40,
    "fee": 22000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e1944301f"
    },
    "pid": 109,
    "name": "Anand",
    "age": 41,
    "fee": 9000
  },
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443020"
    },
    "pid": 113,
    "name": "Gaurav",
    "age": 59,
    "fee": 7000
  }
]
```

Finding the patient name who has paid maximum fee

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
   if(err) throw err;
```

```
    var dbo = db.db("health");
    var mysort={fee:-1};

dbo.collection("patients").find().sort(mysort).limit(1).toArray(functio
n(err,res){
        if(err) throw err;
        console.log(res);
        db.close();
    })
})
```


20BCE1650_health.mongodb

Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc

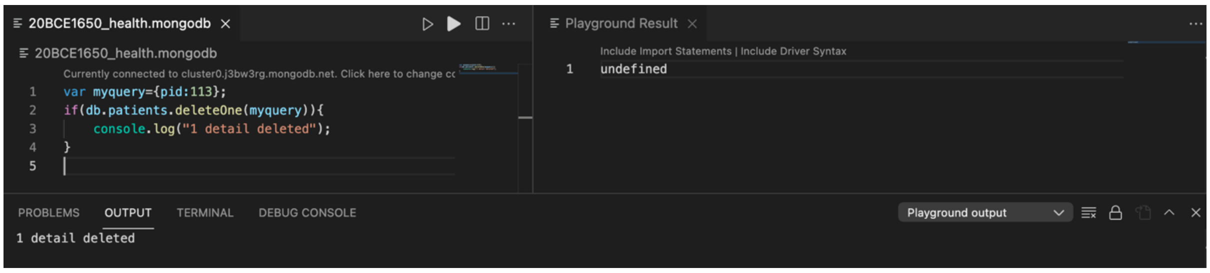
```
var mysort={fee:-1};
db.patients.find().sort(mysort).limit(1)
```

Playground Result

```
[
  {
    "_id": {
      "$oid": "637a1c9198a8183e19443022"
    },
    "pid": 123,
    "name": "Kriti",
    "age": 40,
    "fee": 22000
  }
]
```

Deleting record of a patient based on a certain patient id

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url,function(err,res){
    if(err) throw err;
    var dbo = db.db("health");
    var myquery={pid:113};
    dbo.collection("patients").deleteOne(myquery,function(err,res){
        if(err) throw err;
        console.log("1 detail deleted");
        db.close();
    })
})
```


20BCE1650_health.mongodb
Playground Result
Include Import Statements | Include Driver Syntax
Currently connected to cluster0.j3bw3rg.mongodb.net. Click here to change cc
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Playground output

```
var myquery={pid:113};
if(db.patients.deleteOne(myquery)){
    console.log("1 detail deleted");
}
```

```
1   undefined
```

```
1 detail deleted
```

---

4. Assume you are developing an online solution for a school. Discuss all the doimain names and which domain is most suitable for online student management system. Explain important internet protocols. What are the security vulnerabilities in this online solution.

A domain name is a string of text that maps to a numeric IP address, used to access a website from client software. In plain English, a domain name is the text that a user types into a browser window to reach a particular website. For instance, the domain name for Google is 'google.com'. The actual address of a website is a complex numerical IP address (e.g. 103.21.244.0), but due to DNS, users are able to enter human-friendly domain names and be routed to the websites they are looking for. This process is known as a DNS lookup.

Domain names are all managed by domain registries, which delegate the reservation of domain names to registrars. Anyone who wants to create a website can register a domain name with a registrar, and there are currently over 300 million registered domain names.

Domain names are typically broken up into two or three parts, each separated by a dot. When read right-to-left, the identifiers in domain names go from most general to most specific. The section to the right of the last dot in a domain name is the top-level domain (TLD). These include the 'generic' TLDs such as '.com', '.net', and '.org', as well as country-specific TLDs like '.uk' and '.jp'. To the left of the TLD is the second-level domain (2LD) and if there is anything to the left of the 2LD, it is called the third-level domain (3LD).

For the online student management system as well as other schools and universities, .edu is a distinguishing domain name that clearly identifies the institution and website as being in the educational category. The verification process, which occurs during the domain registration process, is to ensure that any group using this domain extension is truly an educational institution.

On the other hand, .college is an unrestricted generic domain extension designed for academic institutions, businesses and individuals. While this domain extension quickly identifies your organization as an institution for higher education, it does not require a verification process and is therefore not as exclusive as .edu. However, we may consider registering domain names with .com, .org, .edu and .college to ensure that people find our school during online searches. This also helps to prevent schools from claiming a domain name similar to ours which may cause unnecessary confusion.

Protocols are set of rules that help in governing the way a particular technology will function for communication. In other words, it can be said that the protocols are digital languages implemented in the form of networking algorithms. There are different networks and network protocols, user's use while surfing.

1. Transmission Control Protocol (TCP): TCP is a popular communication protocol which is used for communicating over a network. It divides any message into series of packets that are sent from source to destination and there it gets reassembled at the destination.
2. Internet Protocol (IP): IP is designed explicitly as addressing protocol. It is mostly used with TCP. The IP addresses in packets help in routing them through different nodes in a network until it reaches the destination system. TCP/IP is the most popular protocol connecting the networks.
3. User Datagram Protocol (UDP): UDP is a substitute communication protocol to Transmission Control Protocol implemented primarily for creating loss-tolerating and low-latency linking between different applications.
4. Post office Protocol (POP): POP3 is designed for receiving incoming E-mails.
5. Simple mail transport Protocol (SMTP): SMTP is designed to send and distribute outgoing E-Mail.
6. File Transfer Protocol (FTP): FTP allows users to transfer files from one machine to another. Types of files may include program files, multimedia files, text files, and documents, etc.
7. Hyper Text Transfer Protocol (HTTP): HTTP is designed for transferring a hypertext among two or more systems. HTML tags are used for creating links. These links may be in any form like text or images. HTTP is designed on Client-server principles which allow a client system for establishing a connection with the server machine for making a request. The server acknowledges the request initiated by the client and responds accordingly.
8. Hyper Text Transfer Protocol Secure (HTTPS): HTTPS is abbreviated as Hyper Text Transfer Protocol Secure is a standard protocol to secure the communication among two computers one using the browser and other fetching data from web server. HTTP is used for transferring data between the client browser (request) and the web server (response) in the hypertext format, same in case of HTTPS except that the transferring of data is done

in an encrypted format. So it can be said that https thwart hackers from interpretation or modification of data throughout the transfer of packets.

9. Telnet: Telnet is a set of rules designed for connecting one system with another. The connecting process here is termed as remote login. The system which requests for connection is the local computer, and the system which accepts the connection is the remote computer.
10. Gopher: Gopher is a collection of rules implemented for searching, retrieving as well as displaying documents from isolated sites. Gopher also works on the client/server principle.

The huge volume of student and staff data built into the system, along with high turnover and a collaborative multi-platform environment results in rampant data loss at institutions. Some of the possible top data risks & security vulnerability are:

1. Malware

Digital hackers are watching our every move and trick you to download malware and take control of our computer remotely. They use malware to attack computer networks to perpetrate crimes. Fraudsters use virus, malware, spyware, spams, and phishing to gain access to your sensitive personal information and commit financial crimes. Defend our data against malware through secure servers, whether physical or in cloud, and shield against vulnerabilities.

2. Theft & Loss

Unauthorized users without permissions who have access to sensitive data can cause harm to educational institutions as a result of theft. There is a risk of the sensitive academic data will be leaked by staff. It becomes easy to lose our storage media with backup data due to misplacement or theft. When we suffer data loss due to various incidents such as mechanical damage, power failure, software crash, disasters or loss of our laptops and mobile devices, it is another way of inadvertent data exposure. Keep all our data safe and secure using role-based access control to ensure confidentiality and privacy.

3. Unsafe data

If adequate safety precautions are not taken when files and documents are shared in website, smartphones and tablets via internet networks, the information contained on them might gain access to the devices and get exposed to risks. We can make use of cloud deployments to manage the education system better and better.

4. Negligence

When data is stored in computers or laptops, it has become so natural that people lose the information when files are accidentally deleted or even it could fall into the wrong hands. Ensure a proper backup strategy to keep your data on important devices and run them smoothly without hassles.

5. Third party APIs
Quite often, institutions tend to use their credentials for accessing 3rd party APIs. When we integrate third party application into the existing system, we could be exposing the institution and its data at risk. Institutions should integrate the right third party API with their system and enjoy the best quality of service critical to the operations of the institution.
