WebTech Lab DA-5

Binayak Bishnu

20BIT0155

Qs1

1. Consider a client requests the server to view his profile, which is available in "19BIT0--
-info.html". Implement it using the Node.js file system module.

```
<!DOCTYPE html>
<html>

<head>
    <style>
        body {
            background-color: #c4c4c4;
        }

        .card {
            box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2);
            max-width: 20%;
            margin: auto;
            text-align: center;

            padding: 10% 0%;

            background-image: linear-gradient(to bottom, blue, rgba(0, 0, 0, 0));
        }
    </style>
</head>

<body>
    <h1 style="text-align:center">My Profile</h1>
    <div class="card">
        <h1>Binayak Bishnu</h1>
        <p>2nd Year BTech</p>
        <p>VIT, Vellore</p>
        <a>
            <input type="button" value="Contact" style="width:80%; height:30px;
background-color:blue;" />
        </a>
    </div>
```

```
</body>

</html>
```

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
    fs.readFile('Qs-1.html', function (err, data) {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write(data);
        res.end();
    });
}).listen(8080);
```

Output

Qs2

2. Develop a small application for the Basketball scoreboard using Node.js event handling. Here, we have a scoreKeeper.on() event-listener listening for an event that would indicate a basket being made. Each scoring attempt is a call of the shoot_a_basket function. If the ball goes in the net (indicated by a true value for the shot), scoreKeeper.emit() is called, which alerts all event-listeners listening for the make_basket event announcement to run their callback function, and passes the value of the basket made. Display the scores of each team.

```
var events = require('events');
var scoreKeeper = new events.EventEmitter();
VIT_V = [false, 0];
VIT_C = [false, 0];
var shoot_a_basket = function () {
    if (VIT_V[0] == true) {
        console.log('VIT-V has scored');
        VIT_V[0] = false
        VIT_V[1] += 1
    }
    if (VIT_C[0] == true) {
        console.log('VIT-C has scored');
        VIT_C[0] = false
        VIT_C[1] += 1
    }
    console.log('VIT-V :' + VIT_V[1]);
```

```
    console.log('VIT-C :' + VIT_C[1]);
}
scoreKeeper.on('basket', shoot_a_basket);
scoreKeeper.emit('basket', VIT_V[0] = true);
scoreKeeper.emit('basket', VIT_C[0] = true);
scoreKeeper.emit('basket', VIT_V[0] = false);
scoreKeeper.emit('basket', VIT_V[0] = true);
scoreKeeper.emit('basket', VIT_C[0] = true);
scoreKeeper.emit('basket', VIT_V[0] = true);
```

Output

```
PS D:\VIT\Semester_III\Web_Tech_A2;L17,18\WebTech_L17_18\Assessment-5> node Qs-2.js
VIT-V has scored
VIT-V :1
VIT-C :0
VIT-C has scored
VIT-V :1
VIT-C :1
VIT-V :1
VIT-C :1
VIT-V has scored
VIT-V :2
VIT-C :1
VIT-C has scored
VIT-V :2
VIT-C :2
VIT-V has scored
VIT-V :3
VIT-C :2
PS D:\VIT\Semester_III\Web_Tech_A2;L17,18\WebTech_L17_18\Assessment-5>
```

Qs3

3. Create a MongoDB "Student" database and do the following operations:
    1. Insert a new student with a name: Dora
    2. Insert a new student with a name: Sinchan and id=2 (integer)
    3. Insert a new student with a name: Angush, the midterm score of 80, and a final score of 100. Scores should be embedded in a sub-document like this: scores:{midterm:0,final:0}.
    4. Finding a document by a single attribute as name as "your name".
    5. Display the list of students who scored between greater than 50 in the midterm.
    6. Search for students that have scored between [50,80] in midterm AND [80,100] in the final exam.
    7. Update the student Sinchan that you created back in exercise 2 to have a midterm score of 50 and a final score of 100 respectively.
    8. Sort according to the final score in descending order and display name and score without objectID.
    9. Delete user Angush that you created back in exercise 3
    10. Delete all users with a midterm score of less than 80.

```
> use Student
switched to db Student
> db.createCollection("Details")
{ "ok" : 1 }
> db.Details.insert({name"Dora"})
uncaught exception: SyntaxError: missing : after property id :
@(shell):1:23
> db.Details.insert({name:"Dora"})
WriteResult({ "nInserted" : 1 })
> db.Details.insert({name:"Sinchan", id:2})
WriteResult({ "nInserted" : 1 })
> db.Details.insert({name:"Angush", scores:{midterm:80, final:100}})
WriteResult({ "nInserted" : 1 })
> db.Details.find()
{ "_id" : ObjectId("61aa0fc3878d3d7b4629c0bb"), "name" : "Dora" }
{ "_id" : ObjectId("61aa0fd7878d3d7b4629c0bc"), "name" : "Sinchan", "id" : 2 }
{ "_id" : ObjectId("61aa10bf878d3d7b4629c0bd"), "name" : "Angush", "scores" : { "midterm" : 80, "final" : 100 } }
```

```
> db.Details.find({name:"Binayak"}).pretty()
>
```

```
> db.Details.find({name:"Angush"}).pretty()
{
        "_id" : ObjectId("61aa10bf878d3d7b4629c0bd"),
        "name" : "Angush",
        "scores" : {
                "midterm" : 80,
                "final" : 100
        }
}
>
```

```
> db.Details.find({"scores.midterm":{$gt:50}}).pretty()
{
        "_id" : ObjectId("61aa10bf878d3d7b4629c0bd"),
        "name" : "Angush",
        "scores" : {
                "midterm" : 80,
                "final" : 100
        }
}
```

```
> db.Details.find({$and:[{"scores.midterm":{$gte:50, $lte:80}},{"scores.final":{$gte:80, $lte:100}}]}).pretty()
{
        "_id" : ObjectId("61aa10bf878d3d7b4629c0bd"),
        "name" : "Angush",
        "scores" : {
                "midterm" : 80,
                "final" : 100
        }
}
```

```
> db.Details.update({name:"Sinchan"}, {$set:{scores:{midterm:50, final:100}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Details.find({name:"Sinchan"}).pretty
function() {
    this._prettyShell = true;
    return this;
}
> db.Details.update({name:"Sinchan"}, {$set:{scores:{midterm:50, final:100}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.Details.find({name:"Sinchan"}).pretty()
{
        "_id" : ObjectId("61aa0fd7878d3d7b4629c0bc"),
        "name" : "Sinchan",
        "id" : 2,
        "scores" : {
                "midterm" : 50,
                "final" : 100
        }
}
>
```

```
> db.Details.find({},{_id:0}).sort({"scores.final":-1})
{ "name" : "Sinchan", "id" : 2, "scores" : { "midterm" : 50, "final" : 100 } }
{ "name" : "Angush", "scores" : { "midterm" : 80, "final" : 100 } }
{ "name" : "Dora" }
```

```
> db.Details.remove({name:"Angush"})
WriteResult({ "nRemoved" : 1 })
> db.Details.find()
{ "_id" : ObjectId("61aa0fc3878d3d7b4629c0bb"), "name" : "Dora" }
{ "_id" : ObjectId("61aa0fd7878d3d7b4629c0bc"), "name" : "Sinchan", "id" : 2, "scores" : { "midterm" : 50, "final" : 100 } }
>
```

```
> db.Details.remove({"scores.midterm":{$lte:80}})
WriteResult({ "nRemoved" : 1 })
> db.Details.find()
{ "_id" : ObjectId("61aa0fc3878d3d7b4629c0bb"), "name" : "Dora" }
>
```

Qs4

4. i) Design an application using node.js and MongoDB to perform the following operations in a student collection with a name, age, DOB, and year of admission.
   ii) Insert multiple records into the collection.
   iii) Sort all documents in the student collection

---

   iv) Update the document of a student with name='Kevin' to age=25
   v) Limit the result to 4 documents
   vi) Query the document based on age>25

```
const mongoose = require("mongoose")

mongoose.connect("mongodb://localhost:27017/Qs_4corrected", {
    useNewUrlParser: true
});

const detailsSchema = new mongoose.Schema({
    name: String,
    age: Number,
    dob: Date,
    yearofadm: Number
});

const Student = mongoose.model("Student", detailsSchema)

const Binayak = new Student({
    name: "Binayak",
    age: 19,
    dob: "2002-10-11",
    yearofadm: 2020,
});

const Bishnu = new Student({
    name: "Bishnu",
    age: 18,
    dob: "2001-09-11",
    yearofadm: 2020,
});
```

```
const Neo = new Student({
    name: "Neo",
    age: 20,
    dob: "2000-10-11",
    yearofadm: 2021,
});

const Neil = new Student({
    name: "Neil",
    age: 30,
    dob: "1992-10-15",
    yearofadm: 2009,
});

const ABC = new Student({
    name: "ABC",
    age: 26,
    dob: "1995-09-11",
    yearofadm: 2012,
});

Student.insertMany([Binayak, Bishnu, Neo, Neil, ABC], function (err) {
    if (err) {
        console.log(err);
    } else {
        console.log("All records added");
    }
});

Student.find(function (err, student) {
    if (err) {
        console.log(err)
    } else {
        student.forEach((student) => {
            console.log(student);
        });
    }
})

var sortage = { age: 1 };
Student.find({}, function (err, result) {
    if (err) {
        console.log("error query")
    } else {
        console.log(result)
    }
}).sort(sortage)
```

```
Student.updateOne({ name: "Neil" }, { age: 25 }, function (err) {
    if (err) {
        console.log(err)
    } else {
        console.log('Updated');
    }
})

Student.find(function (err, student) {
    if (err) {
        console.log(err)
    } else {
        student.forEach((student) => {
            console.log(student);
        });
    }
}).limit(4)

var sortage = { age: 1 };
Student.find({age:{$gte:25}}, function (err, result) {
    if (err) {
        console.log("error query")
    } else {
        console.log(result)
    }
}).sort(sortage)
```

Output

```
PS D:\VIT\Semester_III\Web_Tech_A2;L17,18\WebTech_L17_18\Assessment-5> node Qs-4.js
[
  {
    _id: new ObjectId("61aa24853757394f60d539e9"),
    name: 'Bishnu',
    _id: new ObjectId("61aa24853757394f60d539e8"),
    name: 'Binayak',
    age: 19,
    dob: 2002-10-11T00:00:00.000Z,
    yearofadm: 2020,
    __v: 0
  },
  {
    _id: new ObjectId("61aa24853757394f60d539ea"),
    name: 'Neo',
    age: 20,
    dob: 2000-10-11T00:00:00.000Z,
    yearofadm: 2021,
    __v: 0
  },
  {
    _id: new ObjectId("61aa24853757394f60d539ec"),
    name: 'ABC',
    age: 26,
    dob: 1995-09-11T00:00:00.000Z,
    yearofadm: 2012,
    __v: 0
  },
  {
    _id: new ObjectId("61aa24853757394f60d539eb"),
    name: 'Neil',
    age: 30,
    dob: 1992-10-15T00:00:00.000Z,
    yearofadm: 2009,
    __v: 0
  }
]
```



```
> db.students.find({name:"Neil"})
{ "_id" : ObjectId("61aa24853757394f60d539eb"), "name" : "Neil", "age" : 25, "dob" : ISODate("1992-10-15T00:00:00Z"), "yearofadm" : 2009, "__v" : 0 }
```

```
PS D:\VIT\Semester_III\Web_Tech_A2;L17,18\WebTech_L17_18\Assessment-5> node Qs-4.js
{
  _id: new ObjectId("61aa24853757394f60d539e8"),
  name: 'Binayak',
  age: 19,
  dob: 2002-10-11T00:00:00.000Z,
  yearofadm: 2020,
  __v: 0
}
{
  _id: new ObjectId("61aa24853757394f60d539e9"),
  name: 'Bishnu',
  age: 18,
  dob: 2001-09-11T00:00:00.000Z,
  yearofadm: 2020,
  __v: 0
}
{
  _id: new ObjectId("61aa24853757394f60d539ea"),
  name: 'Neo',
  age: 20,
  dob: 2000-10-11T00:00:00.000Z,
  yearofadm: 2021,
  __v: 0
}
{
  _id: new ObjectId("61aa24853757394f60d539eb"),
  name: 'Neil',
  age: 25,
  dob: 1992-10-15T00:00:00.000Z,
  yearofadm: 2009,
  __v: 0
}
```

```
PS D:\VIT\Semester_III\Web_Tech_A2;L17,18\WebTech_L17_18\Assessment-5> node Qs-4.js
[
  {
    _id: new ObjectId("61aa24853757394f60d539eb"),
    name: 'Neil',
    age: 25,
    dob: 1992-10-15T00:00:00.000Z,
    yearofadm: 2009,
    __v: 0
  },
  {
    _id: new ObjectId("61aa24853757394f60d539ec"),
    name: 'ABC',
    age: 26,
    dob: 1995-09-11T00:00:00.000Z,
    yearofadm: 2012,
    __v: 0
  }
]
```

Qs5

5. i) Create a web application with username in HTML and node.js using the express framework to handle different types of HTTP requests namely get, post, put, options, and delete.
   ii) Provide different route paths for each of the requests.
   iii) Display the different request types with the username in the browser when the application is executed.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <title>Exercise 5</title>
</head>

<body>
    <form action="/submitform" method="POST">
        First Name: <input name="firstName" type="text" /><br />
        Last Name: <input name="lastName" type="text" /><br />
        <input type="submit" />
    </form>
    <form action="/deletedata" method="POST">
        First Name: <input name="firstName" type="text" /><br />
        Last Name: <input name="lastName" type="text" /><br />
        <input type="submit" value="Delete" />
    </form>
</body>

</html>
```

```
var express = require('express');
const path = require('path');

var app = express();

var bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: false }));

app.get('/', function (req, res) {
    res.sendFile(path.join(__dirname, 'Qs-5.html'));
});

app.post('/submitform', function (req, res) {
```

```
    var name = req.body.firstName + ' ' + req.body.lastName;
    res.send(name + ' POSTed!!!');
});

app.put('/updatedata', function (req, res) {
    res.send('PUT Request');
});

app.delete('/deletedata', function (req, res) {
    res.send('DELETE Request');
});

app.listen(4200, () => {
    console.log('Running...');
});
```

Output


Exercise 5
localhost:4200
First Name: Binayak
Last Name: Bishnu
Submit
First Name:
Last Name:
Delete


localhost:4200/submitform
localhost:4200/submitform
Binayak Bishnu POSTed!!!

Qs6

6. i) Write an HTML and node.js program for creating and storing session information of the user. The HTML page contains username, password, remember me next time checkbox option and a login button.

ii) Assume, the user name and password are already registered in the node.js server.

iii) Perform the following:

a. If the user enters an invalid user username or password, display an appropriate error message.

b. After successful login, display a welcome message.

c. Allow the user to enter the username and password for 3 times.

If the user enters username and password more than 3 times, display the message "you are blocked".

```
<!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>Exercise 6</title>
</head>

<body>
    <form id="form1" name="myForm1" method="post" action="/signup">
        <h2>Sign Up</h2>
        <div>
            <label for="username">Username</label>
            <input type="text" name="username" name="username" />
        </div>
        <div>
            <label for="password">Password</label>
            <input type="password" name="password" name="password" />
        </div>
        <input type="checkbox" name="remember" />
        <input type="submit" name="submitbtn" value="Login" />
    </form>

    <form id="form2" name="myForm2" method="post" action="/login">
        <h2>Login</h2>
        <div>
            <label for="username">Username</label>
            <input type="text" name="username" name="username" />
        </div>
```

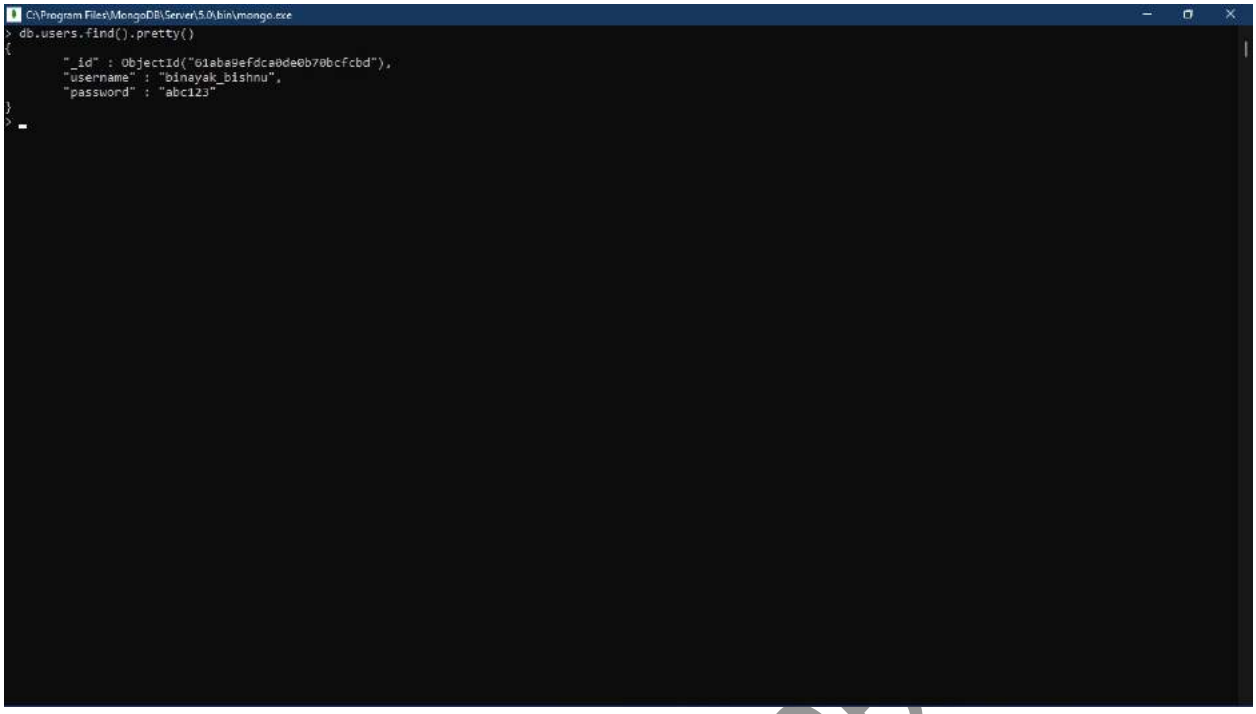
```
        <div>
            <label for="password">Password</label>
            <input type="password" name="password" name="password" />
        </div>
        <input type="checkbox" name="remember" />
        <input type="submit" name="submitbtn" value="Login" />
    </form>
</body>

</html>


var express = require('express');
const path = require('path');

var app = express();

var bodyParser = require('body-parser');

var u = bodyParser.urlencoded({
    extended: false
})
app.use(express.static('public'));
app.get('/', function (req, res) {
    res.sendFile(path.join(__dirname, 'Qs-6.html'));
})
app.post('/signup', u, function (req, res) {
    var response = {
        username: req.body.username,
        password: req.body.password,
    };
    var usernameobj = {
        username: req.body.username,
    }

    var MongoClient = require('mongodb').MongoClient, format =
require('util').format;
    MongoClient.connect('mongodb://127.0.0.1:27017/', function (err, db) {
        if (err) {
            throw err;
        }
        else {
            console.log("Connected");
        }
        var myDB = db.db("Qs-6");
        myDB.collection('users').findOne(usernameobj, function (err, result) {
            console.log(result);
            if (err) {
                throw err;
```

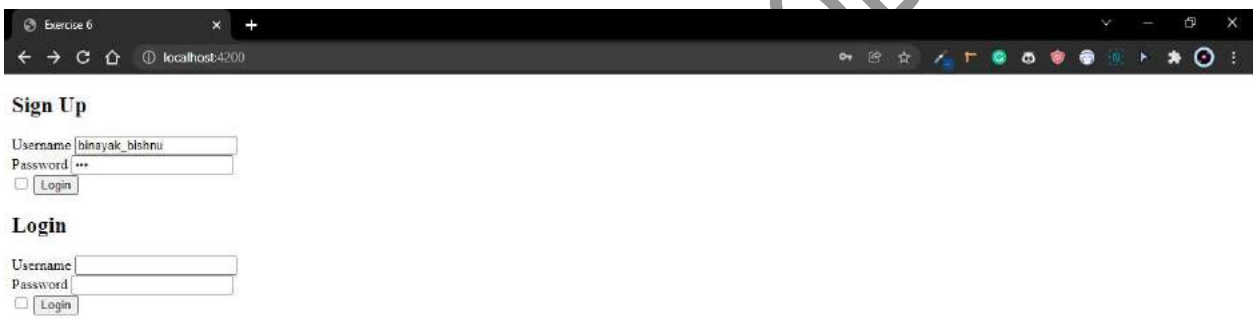
```
            }
            else if (result) {
                console.log(usernameobj.username + " already exists");
                res.sendFile(path.join(__dirname, 'Qs-6.html'));
            }
            else if (!result) {
                myDB.collection('users').insertOne(response, function (err, result)
{
                    if (err) {
                        throw err;
                    }
                    else {
                        console.log("New account created");
                        console.log(response);
                        res.sendFile(path.join(__dirname, 'Qs-6b.html'));
                    }
                });
            }
        });
    });
})

app.post('/login', u, function (req, res) {
    var response = {
        username: req.body.username,
        password: req.body.password
    };
    var usernameobj = {
        username: req.body.username
    }
    var passwordobj = {
        password: req.body.password
    }
    var MongoClient = require('mongodb').MongoClient, format =
require('util').format;
    MongoClient.connect('mongodb://127.0.0.1:27017/Qs-6', function (err, db) {
        if (err) {
            throw err;
        }
        console.log("Connected");
        var myDB = db.db("Qs-6");
        myDB.collection('users').findOne(usernameobj, function (err, result) {
            console.log(usernameobj);
            console.log(result);
            if (err) throw err;
            else if (!result) {
                console.log("No user found");
                res.sendFile(path.join(__dirname, 'Qs-6.html'));
```

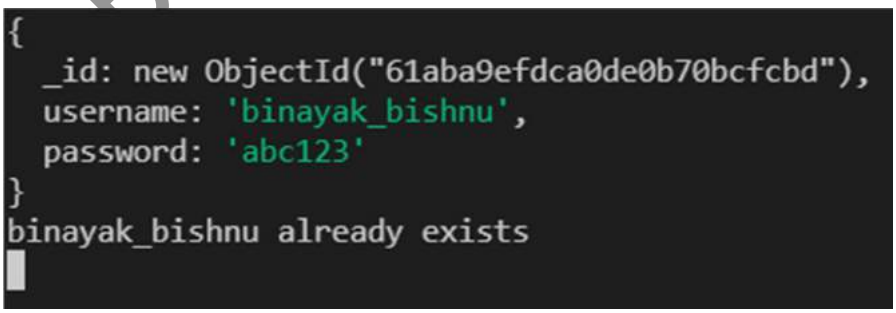
```
            }
            else if (result) {
                myDB.collection('users').findOne(passwordobj, function (err,
result2) {
                    if (err) {
                        throw err;
                    }
                    else if (!result2) {
                        console.log("Password is incorrect for " +
response.username);
                        res.sendFile(path.join(__dirname, 'Qs-6.html'));
                    }
                    else if (result2) {
                        console.log(response.username + " logged in");
                        res.sendFile(path.join(__dirname, 'Qs-6b.html'));
                    }
                });
            }
        });
    });
    console.log(response);
})

app.listen(4200, () => {
    console.log('Running...');
});
```

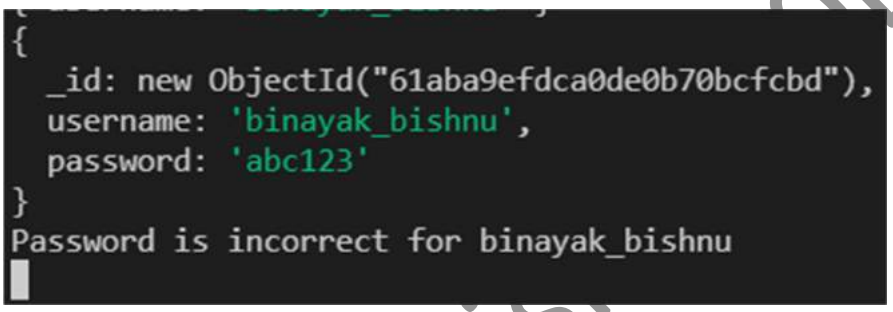
Output


```
{
  username: 'binayak_bishnu',
  password: 'abc123',
  _id: new ObjectId("61aba9efdca0de0b70bcfcbd")
}
```


C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.users.find().pretty()
{
"_id" : ObjectId("61aba9efdca0de0b70bcfcbd"),
"username" : "binayak_bishnu",
"password" : "abc123"
}
>


Exercise 6
localhost:4200
Sign Up
Username binayak_bishnu
Password
Login
Login
Username
Password
Login


{
_id: new ObjectId("61aba9efdca0de0b70bcfcbd"),
username: 'binayak_bishnu',
password: 'abc123'
}
binayak_bishnu already exists



Qs7

7. A company database maintains the vehicle information of their employees. It stores the information empid(int), vehicle number(string/int), owner name(string), brand name(string) and year of purchase(int).
    a) Create a Collection in MongoDB with the above fields and insert 10 documents at least.
    b) Create an HTML form using NodeJS and express for the employee to change the details when he/she changes his/her vehicle by submitting his/her empid and the new vehicle details. The form creation should use CSS for making it interactive and Use ExpressJS at the server-side.

```
<!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>UpdateInfo Form</title>
    <style>
        body {
            padding: 200px;
            background: linear-gradient(to bottom, #33ccff 0%, #ff99cc 100%);
        }

        div {
            background-color: rgba(255, 155, 25, 0.5);
            text-align: center;
            border-radius: 30px;
            padding: 10px;
            box-shadow: 5px 10px #888888;
        }

        table {
            width: 300px;
        }

        table td {
            border-radius: 20px;
        }
```

```
        button {
            background-color: white;
            color: black;
            border: 2px solid #e7e7e7;
            padding: 5px 15px;
            border-radius: 20px;
        }

        button:hover {
            background-color: #e7e7e7;
        }
    </style>
</head>

<body>
    <h1>Update your vehicle Info</h1>
    <form action="/" method="post">
        <table>
            <tr>
                <td>Employee ID:</td>
                <td><input type="number" name="eid"><br></td>
            </tr>

            <tr>
                <td>Vehicle Number:</td>
                <td><input type="number" name="vnum"><br></td>
            </tr>
```

```
            <tr>
                <td>Brand:</td>
                <td><input type="text" name="brnd"><br></td>
            </tr>

            <tr>
                <td>Year of Purchasing:</td>
                <td><input type="number" name="yop"><br></td>
            </tr>

            <tr>
                <td rowspan="2"><button type="submit">Update</button></td>
            </tr>
        </table>
    </form>
</body>

</html>

<!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>20BIT0195</title>
    <style>
```

```
        body {
            padding: 200px;
            background: linear-gradient(to bottom, #33ccff 0%, #ff99cc 100%);
        }

        div {
            background-color: rgba(255, 155, 25, 0.5);
            text-align: center;
            border-radius: 30px;
            padding: 10px;
            box-shadow: 5px 10px #888888;
        }

        table {
            width: 300px;
        }

        table td {
            border: 2px Solid black;
            border-radius: 20px;
        }
    </style>
</head>

<body>

    <h1>Updated vehicle Info</h1>
```

```
<table>
    <tr>
        <td>Employee ID:</td>
        <td>
            <%= empid %>
        </td>
    </tr>

    <tr>
        <td>Vehicle Number:</td>
        <td>
            <%= vehnum %>
        </td>
    </tr>

    <tr>
        <td>Brand:</td>
        <td>
            <%= band %>
        </td>
    </tr>

    <tr>
        <td>Year of Purchasing:</td>
        <td>
            <%= poy %>
        </td>
    </tr>
```

```
    </table>
</body>

</html>

const express = require("express");
const bodyParser = require("body-Parser");
const app = express();
app.set("view engine", "ejs");
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static("public"));

var MongoClient = require("mongodb").MongoClient;
var url = "mongodb://localhost:27017/mydb";

app.listen(3000, function () {
  console.log("Server is running on port 3000");
});

app.get("/", function (req, res) {
  res.sendFile(__dirname + "/form7.html");
});

var myobj = [
  {
    _id: 1,
    veh_num: 1324,
    name: "Parikshit",
```

```
  brand: "Nexa",
  YOP: 2020,
},
{
  _id: 2,
  veh_num: 2124,
  name: "Meghna",
  brand: "Hyundai",
  YOP: 2018,
},
{
  _id: 3,
  veh_num: 5814,
  name: "Umang",
  brand: "Ford",
  YOP: 2017,
},
{
  _id: 4,
  veh_num: 6474,
  name: "Mithila",
  brand: "Tata",
  YOP: 2021,
},
{
  _id: 5,
  veh_num: 7451,
  name: "Dhruv",
```

```
  brand: "Nexa",
  YOP: 2020,
},
{
  _id: 6,
  veh_num: 3521,
  name: "Phil",
  brand: "Toyota",
  YOP: 2015,
},
{
  _id: 7,
  veh_num: 4175,
  name: "Claire",
  brand: "Honda",
  YOP: 2014,
},
{
  _id: 8,
  veh_num: 6684,
  name: "Luke",
  brand: "Volkswagen",
  YOP: 2012,
},
{
  _id: 9,
  veh_num: 9145,
  name: "Manny",
```

```
    brand: "Kia",
    YOP: 2019,
  },
  {
    _id: 10,
    veh_num: 8419,
    name: "Gloria",
    brand: "Ferrari",
    YOP: 2021,
  },
];

var n, v, b, y;
app.post("/", function (req, res) {
  var n = Number(req.body.eid);
  var v = req.body.vnum;
  var b = req.body.brnd;
  var y = req.body.yop;
  res.render("updated", { empid: n, vehnum: v, band: b, poy: y });

  MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    dbo = db.db("company");
    var myquery = { _id: n };
    console.log(n);
    var newvalues = { $set: { veh_num: v, brand: b, YOP: y } };
    dbo
      .collection("vehicle")
```

```
      .updateMany(myquery, newvalues, function (err, res) {
        if (err) throw err;
        console.log(res);
        console.log("Document Updated");
        db.close();
      });
  });
});
```


Qs8

8. The IPL website has a MongoDB database of players. The following information about the players are stored – name, IPL franchise, country, bid_amount

   a) Create an HTML form with appropriate CSS containing text field. Name the text field as **find** and radio button(IPL, country, bid). Name the radio button as **find_details**. On submitting an Express JS in Node server-side code is called that displays information about

   b) The player if the name of the player is entered in **find** and no radio button is checked.

   c) The players of a particular country representing IPL. If the radio button IPL is clicked and country name is entered in **find**

   d) The player name, IPL franchise, and country for the player whose bid amount is greater than or equal or bid amount given in **find**. if the bid radio button is checked and the bid amount is entered in **find**.

   e) Store the data in MongoDB database

Name	IPL Franchise	Country	Bid_Amount
M.S.Dhoni	Rising Pune Super Gaints	India	500000
Raina	Gujarat Lions	India	50000
Bravo	Gujarat Lions	West Indies	200000
Chris Gayle	Royal Challengers Banglore	West Indies	100000
du Plessis	Rising Pune Super Gaints	South Africa	150000
Virat Kohli	Royal Challengers Banglore	india	200000
David Warner	Sunrisers hyderabad	Australia	100000
Sunil Narine	Kolkota Knight Riders	SriLanka	160000

```
<!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>Search Player Info Form</title>
    <style>
        body {
            padding: 200px;
            background: linear-gradient(to bottom, #33ccff 0%, #ff99cc 100%);
        }

        div {
            background-color: rgba(255, 255, 255, 0.5);
            text-align: left;
            border-radius: 30px;
            padding: 10px;
            box-shadow: 5px 10px #888888;
            width: 300px;
        }

        button {
            background-color: white;
            color: black;
            border: 2px solid #e7e7e7;
            padding: 5px 15px;
            border-radius: 20px;
```

```
        }

        button:hover {
            background-color: #e7e7e7;
        }
    </style>
</head>

<body>
    <h1>Search Players Info</h1>
    <form action="/" method="post">
        <table>
            <tr>
                <td>
                    <input name="finddetails" type="radio" value="1">
                    <label for="rname"><input type="text" name="name"
placeholder="Player Name"></label>
                </td>
            </tr>

            <tr>
                <td>
                    <input name="finddetails" type="radio" value="2">
                    <label for="rfranchise">
                        <select name="Franchise">
                            <option name="none" value="">Select
Franchise</option>
                            <option name="RPS" value="Rising Pune Super
Giants">Rising Pune Super Giants</option>
```

```
                            <option name="GL" value="Gujrat Lions">Gujrat
Lions</option>

                            <option name="RCB" value="Royal Challengers
Banglore">Royal Challengers Banglore</option>

                            <option name="SRH" value="Sunrisers
Hyderabad">Sunrisers Hyderabad</option>

                            <option name="KKR" value="Kolkata Night
Riders">Kolkata Night Riders</option>

                            <option name="RR" value="Rajasthan
Royals">Rajasthan Royals</option>

                            <option name="PKBS" value="Punjab Kings">Punjab
Kings</option>

                            <option name="DC" value="Delhi Capitals">Delhi
Capitals</option>

                        </select>

                    </label>

                </td>

            </tr>


            <tr>

                <td>

                    <input name="finddetails" type="radio" value="3">

                    <label for="rcountry">

                        <select name="country">

                            <option value="nope">Select Country</option>

                            <option value="India">India</option>

                            <option value="West Indies">West Indies</option>

                            <option value="South Africa">South Africa</option>

                            <option value="Australia">Australia</option>

                            <option value="Sri Lanka">Sri Lanka</option>

                        </select>
```

```
                    </label>
                </td>
            </tr>

            <tr>
                <td>
                    <input name="finddetails" type="radio" value="4">
                    <label for="rbid">
                        <input type="number" name="bida"
placeholder="Bid_Amount">
                    </label>
                </td>
            </tr>

            <tr>
                <td rowspan="2"><button type="submit">Search</button></td>
            </tr>
        </table>
    </form>
</body>

</html>
```



```
const express = require("express");
const bodyParser = require("body-Parser");
const app = express();
app.set("view engine", "ejs");
app.use(bodyParser.urlencoded({ extended: true }));
```

```
app.use(express.static("public"));

app.listen(3000, function () {
  console.log("Server is running on port 3000");
});

var MongoClient = require("mongodb").MongoClient;
var url = "mongodb://localhost:27017/";

app.get("/", function (req, res) {
  res.sendFile(__dirname + "/form8.html");
});

var myobj = [
  {
    _id: 1,
    name: "MS Dhoni",
    IPL_Franchise: "Rising Pune Super Giants",
    Country: "India",
    Bid_Amount: 500000,
  },
  {
    _id: 2,
    name: "Raina",
    IPL_Franchise: "Gujrat Lions",
    Country: "India",
    Bid_Amount: 50000,
  },
```

```
{
  _id: 3,
  name: "Bravo",
  IPL_Franchise: "Gujrat Lions",
  Country: "West Indies",
  Bid_Amount: 200000,
},
{
  _id: 4,
  name: "Chris Gayle",
  IPL_Franchise: "Royal Challengers Banglore",
  Country: "West Indies",
  Bid_Amount: 100000,
},
{
  _id: 5,
  name: "Du Plessis",
  IPL_Franchise: "Rising Pune Super Giants",
  Country: "South Africa",
  Bid_Amount: 150000,
},
{
  _id: 6,
  name: "Virat Kohli",
  IPL_Franchise: "Royal Challengers Banglore",
  Country: "India",
  Bid_Amount: 200000,
},
```

```
  {
    _id: 7,
    name: "David Warner",
    IPL_Franchise: "Sunrisers Hyderabad",
    Country: "Australia",
    Bid_Amount: 100000,
  },
  {
    _id: 8,
    name: "Sunil Narine",
    IPL_Franchise: "Kolkata Night Riders",
    Country: "SriLanka",
    Bid_Amount: 160000,
  },
];

app.post("/", function (req, res) {
  var n = req.body.name;
  var f = req.body.Franchise;
  var c = req.body.country;
  var b = Number(req.body.bida);
  var r = Number(req.body.finddetails);

  MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    dbo = db.db("IPL");
    var myquery;
    function mymongo(myquery) {
```

```
    dbo.collection("iplinfo").find(myquery).toArray(function (err, res) {
        if (err) throw err;
        console.log(res);
        db.close();
      });
  }
  switch (r) {
    case 1: //Find by name
      myquery = { name: n };
      mymongo(myquery);
      break;

    case 2: //Find by franchise
      myquery = { IPL_Franchise: f };
      mymongo(myquery);
      break;

    case 3: //Find by Country
      myquery = { Country: c };
      mymongo(myquery);
      break;

    case 4: //Find by Bid_Amount
      myquery = { Bid_Amount: { $gte: b } };
      mymongo(myquery);
      break;
  }
});
```

```
});
```