

School of Information Technology and Engineering

ITE1002 – Web Technologies

Laboratory Assessment-5

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.

HTML code:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<style>
.card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  max-width: 300px;
  margin: auto;
  text-align: center;
  font-family: arial;
}
.title {
  color: grey;
  font-size: 18px;
}
button {
  border: none;
  outline: 0;
  display: inline-block;
  padding: 8px;
  color: white;
  background-color: #000;
  text-align: center;
```

```

    cursor: pointer;
    width: 100%;
    font-size: 18px;
}
a {
    text-decoration: none;
    font-size: 22px;
    color: black;
}
button:hover, a:hover {
    opacity: 0.7;
}
</style>
</head>
<body>
<h2 style="text-align:center">User Profile Card</h2>
<div class="card">
    
    <h1>AAYUSHI </h1>
    <p class="title">STUDENT </p>
    <p>VIT VELLORE</p>
    <div style="margin: 24px 0;">
    <a href="#"><i class="fa fa-dribbble"></i></a>
    <a href="#"><i class="fa fa-twitter"></i></a>
    <a href="#"><i class="fa fa-linkedin"></i></a>
    <a href="#"><i class="fa fa-facebook"></i></a>
    </div>
    <p><button>Contact</button></p>
</div>
</body>
</html>

```

Javascript code:

```

var http = require('http');
var fs = require('fs');
http.createServer(function (req, res)
{
    fs.readFile('abc.html', function(err, data) {

```

```
res.writeHead(200, {'Content-Type': 'text/html'});  
res.write(data);  
res.end();  
});}).listen(8080);
```

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.

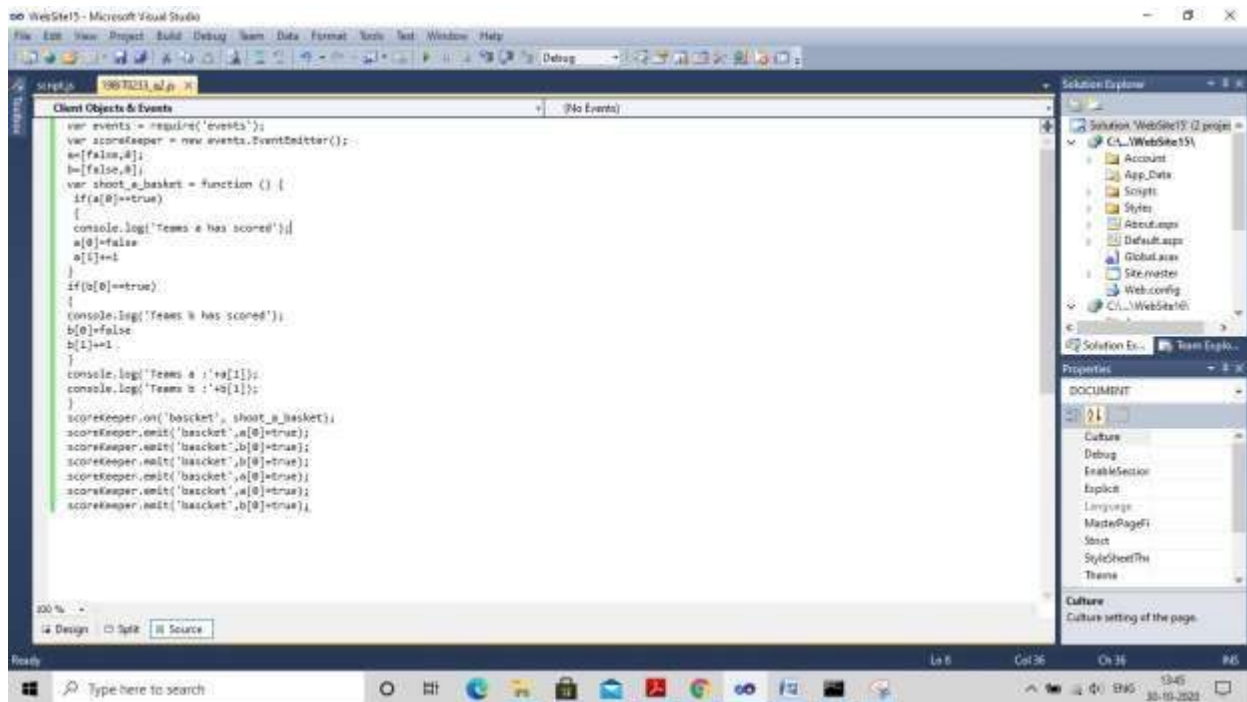
Code:

```
var events = require('events');
var scoreKeeper = new events.EventEmitter();
a=[false,0];
b=[false,0];
var shoot_a_basket = function () {
  if(a[0]==true)
  {
    console.log('Teams a has scored');
    a[0]=false
    a[1]+=1
  }
  if(b[0]==true)
  {
    console.log('Teams b has scored');
    b[0]=false
    b[1]+=1
  }
  console.log('Teams a :'+a[1]);
  console.log('Teams b :'+b[1]);
}
scoreKeeper.on('basket', shoot_a_basket);
scoreKeeper.emit('basket',a[0]=true);
scoreKeeper.emit('basket',b[0]=true);
scoreKeeper.emit('basket',b[0]=true);
scoreKeeper.emit('basket',a[0]=true);
```

```
scoreKeeper.emit('basket',a[0]=true);  
scoreKeeper.emit('basket',b[0]=true);
```

Command Prompt

```
Microsoft Windows [Version 10.0.18362.1082]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\DELL>cd Desktop  
  
C:\Users\DELL\Desktop>cd web tech da  
  
C:\Users\DELL\Desktop\web tech da>node 19BIT0233_q2.js  
Teams a has scored  
Teams a :1  
Teams b :0  
Teams b has scored  
Teams a :1  
Teams b :1  
Teams b has scored  
Teams a :1  
Teams b :2  
Teams a has scored  
Teams a :2  
Teams b :2  
Teams a has scored  
Teams a :3  
Teams b :2  
Teams b has scored  
Teams a :3  
Teams b :3  
  
C:\Users\DELL\Desktop\web tech da>
```



QUESTION 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.

1. Db.information.insert({'Name':'Dora'-});

```
> use Student
switched to db Student
> show collections
information
> db.information.insert({'Name':'Dora'});
WriteResult({ "nInserted" : 1 })
>
```

1. db.information.insert({'Name':'Sinchan','Id':2-});

```
> db.information.insert({'Name':'Sinchan','Id':2});
WriteResult({ "nInserted" : 1 })
>
```

2. db.information.insert({'Name':'Angush','scores':*, 'Midterm':70, 'Final':90-+-});

```
> db.information.insert({'Name':'Angush','Scores':[{ 'Midterm':70, 'Final':90 }]});
WriteResult({ "nInserted" : 1 })
>
```

3. db.information.find().pretty();

```
> db.information.find().pretty();
{ "_id" : ObjectId("5f94e9b92791cdf22ac5868"), "Name" : "Dora" }
{
  "_id" : ObjectId("5f94ea152791cdf22ac5869"),
  "Name" : "Sinchan",
  "Id" : 2
}
{
  "_id" : ObjectId("5f94eaa92791cdf22ac586a"),
  "Name" : "Angush",
  "Scores" : [
    {
      "Midterm" : 70,
      "Final" : 90
    }
  ]
}
```

5. db.information.find(Scores:\$elemMatch:,'Midterm':,\$gt:50}}}).pretty();

```
> db.information.find({Scores:$elemMatch:{'Midterm':{$gt:50}}}).pretty();
{
  "_id" : ObjectId("5f94e9b92791cdf22ac5868"),
  "Name" : "Dora",
  "Scores" : [
    {
      "Midterm" : 66,
      "Final" : 80
    }
  ]
}
{
  "_id" : ObjectId("5f94f2e92791cdf22ac586b"),
  "Name" : "Angush",
  "Scores" : [
    {
      "Midterm" : 70,
      "Final" : 90
    }
  ]
}
```

6. db.information.find({'Scores':{\$elemMatch:{'Midterm':{\$gte:50,\$lte:80}}}},{'Scores':
{ \$elemMatch:{'Final':{\$gte:80,\$lte:100}}}}).pretty();

```
> db.information.find({'Scores':{$elemMatch:{'Midterm':{$gte:50,$lte:80}}},{ 'Scores':{$elemMatch:{'Final':{$gte:80,$lte:100}}}}).pretty();
{
  "_id" : ObjectId("5f94e9b92791cdf22ac5868"),
  "Scores" : [
    {
      "Midterm" : 66,
      "Final" : 80
    }
  ]
}
{
  "_id" : ObjectId("5f94e9b92791cdf22ac5869"),
  "Scores" : [
    {
      "Midterm" : 50,
      "Final" : 100
    }
  ]
}
{
  "_id" : ObjectId("5f94f2e92791cdf22ac586b"),
  "Scores" : [
    {
      "Midterm" : 70,
      "Final" : 90
    }
  ]
}
```

7. db.information.find({}, {_id:0}).sort({'Scores.Final':-1}).pretty();

```
> db.information.find({}, {_id:0}).sort({'Scores.Final':-1}).pretty();
{
  "_id" : ObjectId("5f94e9b92791cdf22ac5868"), "Name" : "Dora"
}
{
  "_id" : ObjectId("5f94e9b92791cdf22ac5869"),
  "Name" : "Sinchon",
  "Id" : 2
}
{
  "_id" : ObjectId("5f94e9b92791cdf22ac586a"),
  "Name" : "Angush",
  "Scores" : [
    {
      "Midterm" : 70,
      "Final" : 90
    }
  ]
}
```


8.

```
> db.information.find({}, {_id:0}).sort({'Scores.Final':-1}).pretty();
{
  "Name" : "Sinchan",
  "Id" : 2,
  "Scores" : [
    {
      "Midterm" : 50,
      "Final" : 100
    }
  ]
}
{
  "Name" : "Angush",
  "Scores" : [
    {
      "Midterm" : 70,
      "Final" : 90
    }
  ]
}
{
  "Name" : "Dora",
  "Scores" : [
    {
      "Midterm" : 60,
      "Final" : 80
    }
  ]
}
```

9.

```
> db.information.deleteOne({'Name':'Angush'});
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

10. `db.information.deleteMany({'Scores.Final':{$lt:80}});`

```
> db.information.deleteMany({'Scores.Final':{$lt:80}});
{ "acknowledged" : true, "deletedCount" : 0 }
```

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

Code:

```
const mongoose = require("mongoose")
mongoose.connect("mongodb://localhost:27017/Class",{useNewUrlParser: true});
```

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

```
const Student = mongoose.model("Student",studentSchema)
```

```
const Aayushi = new Student({
  name: "Aayushi Srivastava",
  age: 19,
  dob: "2001-05-04",
  yearofadm: 2019
});
```

```
const khushboo = new Student({
  name: "khushboo rajawat",
  age: 18,
```

```
    dob: "2001-12-05",  
    yearofadm: 2019  
});
```

```
const aditi = new Student({  
    name: "Aditi Tripathi",  
    age: 20,  
    dob: "2000-10-04",  
    yearofadm: 2019  
});
```

```
const aakash = new Student({  
    name: "Aayushi Srivastava",  
    age: 20,  
    dob: "2000-10-15",  
    yearofadm: 2019  
});
```

```
const phoebe = new Student({  
    name: "Kevin",  
    age: 21,  
    dob: "1999-05-04",  
    yearofadm: 2016  
});
```

```
Student.insertMany([Aayushi,khushboo,aditi,aakash,kevin],function(err){  
    if(err){  
        console.log(err);  
    }else{  
        console.log("successfully saved all the fruits")  
    }  
});
```

```
Student.find(function(err, student){  
    if(err){  
        console.log(err)  
    }else{
```

```

        // console.log(fruits)
        student.forEach((student) => {
            console.log(student.name)
        });
        // mongoose.connection.close();

    }
})

```

III)

```
var sortage = {age: 1};
```

```

Student.find({},function(err,result){
    if(err){
        console.log("error query")
    }else{
        console.log(result)
    }
}).sort(sortage)

```

IV)

```

Student.updateOne({name: "Kevin"},{age: 25},function(err){
    if(err){
        console.log(err)
    }else{
        console.log('updation completed')
    }
})

```

v)

```

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

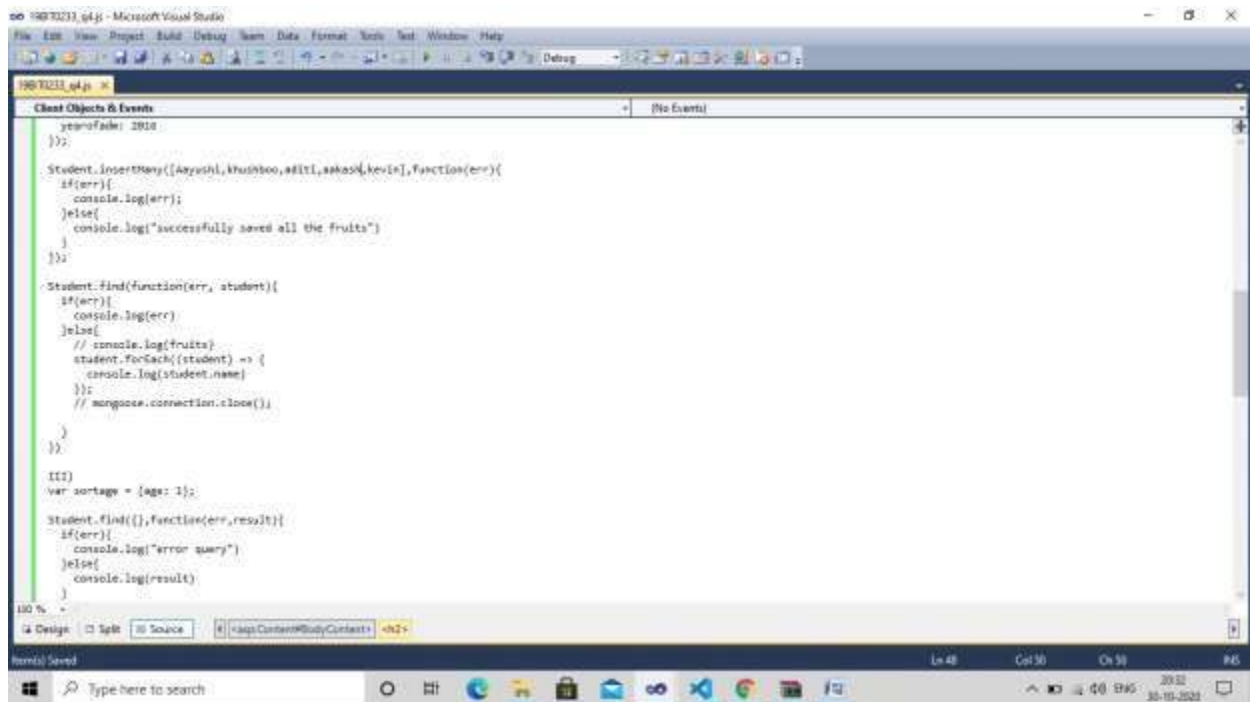
```

```

        console.log(student);
    });
    // mongoose.connection.close();

}
}).limit(4)

```



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.

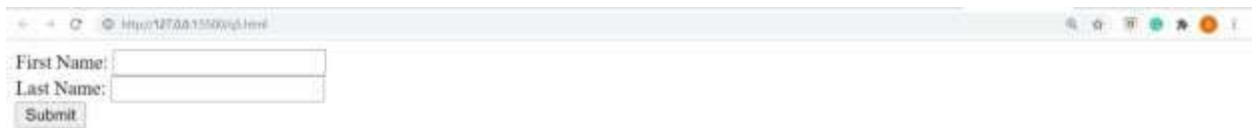
Code:

```

<!DOCTYPEhtml>
<title>19BIT0233</title>
<htmlxmlns="http://www.w3.org/1999/xhtml">

```

```
<head>
<metacharset="utf-8"/>
<title></title>
</head>
<body> <formaction="/submit-student-data"method="post">
First Name: <inputname="firstName"type="text"/><br/>
Last Name: <inputname="lastName"type="text"/><br/>
19BIT0233 AAYUSHI SRIVASTAVA
<inputtype="submit"/>
</form>
</body>
var express = require('express');
var app = express();
var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: false }));
app.get('/', function (req, res) {
res.sendFile('index.html'); });
app.post('/submitstudent-data', function (req, res)
{ var name =
req.body.firstName + ' ' + req.body.lastName;
res.send(name + ' Submitted Successfully!'); });
var server = app.listen(5000, function () {
console.log('Node server is running..');
});
</html>
```



A screenshot of a web browser window. The address bar shows 'http://127.0.0.1:5500/index.html'. The page content includes a form with two text input fields labeled 'First Name:' and 'Last Name:', and a 'Submit' button below them.

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”.**

```
var express = require('express');  
var app = express();
```

```

var bodyParser = require('body-parser');
// Create application/x-www-form-urlencoded parser
var urlencodedParser = bodyParser.urlencoded({ extended:
false })
app.use(express.static('public'));
app.get('/login.html', function (req, res) {
res.sendFile( dirname + "/" + "/public/login.html" );
})
//Sign up form details
app.post('/process1', urlencodedParser, function (req, res)
{ response = {
user_name:req.body.uname,
password:req.body.password, };
response2={user_name:req.body.uname}; //Store sign up details
in mongodb--db(Project1)
var MongoClient = require('mongodb').MongoClient, format =
require('util').format;
MongoClient.connect('mongodb://127.0.0.1:27017/
MyProject', function(err, db) {
if (err) throw err;
else{
console.log("Connected to Database");
}
//insert record
db.collection('newcustomer').findOne(response2,
function(err,result){
if(err) throw err;
if(result)
{
console.log("Username already Exists..Try another");
res.sendFile( dirname + "/" + "/
public/y.html" );
}
if(!result)
{
db.collection('newcustomer').insert(response, function(err,
result) {
if (err) throw err;

```



```

else
{
console.log("Record added & Account created" );
console.log("Your
username:"+req.body.username+ ",Your
password:"+req.body.pwd1);
res.sendFile( dirname + "/" + "/public/y.html" ); }
});
} db.close(); }); });
console.log(response);
//res.send(JSON.stringify(response));
//res.redirect("/y.html")
})
//Sign in form details
app.post('/login.html', urlencodedParser, function (req,
res) {
response = {
user_name:req.body.uname
}; response2={password:req.body.password}; //Check the user
is valid or not
var MongoClient = require('mongodb').MongoClient, format =
require('util').format;
MongoClient.connect('mongodb://127.0.0.1:27017/MyProject',
function(err, db) {
if (err) throw err;
console.log("connected to database");
//var collection=db.collection('login');
db.collection('newuser').findOne(response,
function(err,result){ if(err) throw err;
if(!result){
console.log("No user found..First Register in our Magic
Shopping");
res.sendFile( dirname + "/" + "/public/y.html" ); } if(
result) {
db.collection('newcustomer').findOne(response2,
function(err,result2){
if(err) throw err; if(!result2){

```

```

console.log(req.body.uname+" is exists.But password is
incorrect..");
res.sendFile( dirname + "/" + "/public/y.html" );
} if(result2){
console.log("user available.."+req.body.uname+" is now
logged in..");
res.sendFile( dirname + "/" + "/public/login.html" ); } });
} }); });
console.log(response); })
//create a server
var server = app.listen(8081, function () {
var host = server.address().address
var port = server.address().port
console.log("Example app listening at http://%s:%s", host,
port) })

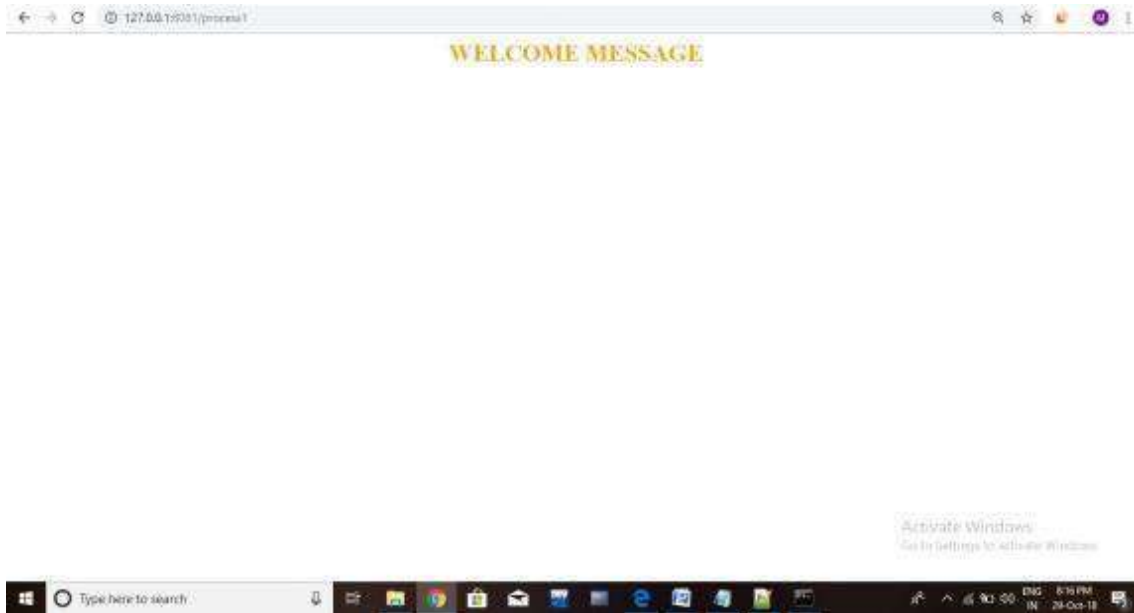
```

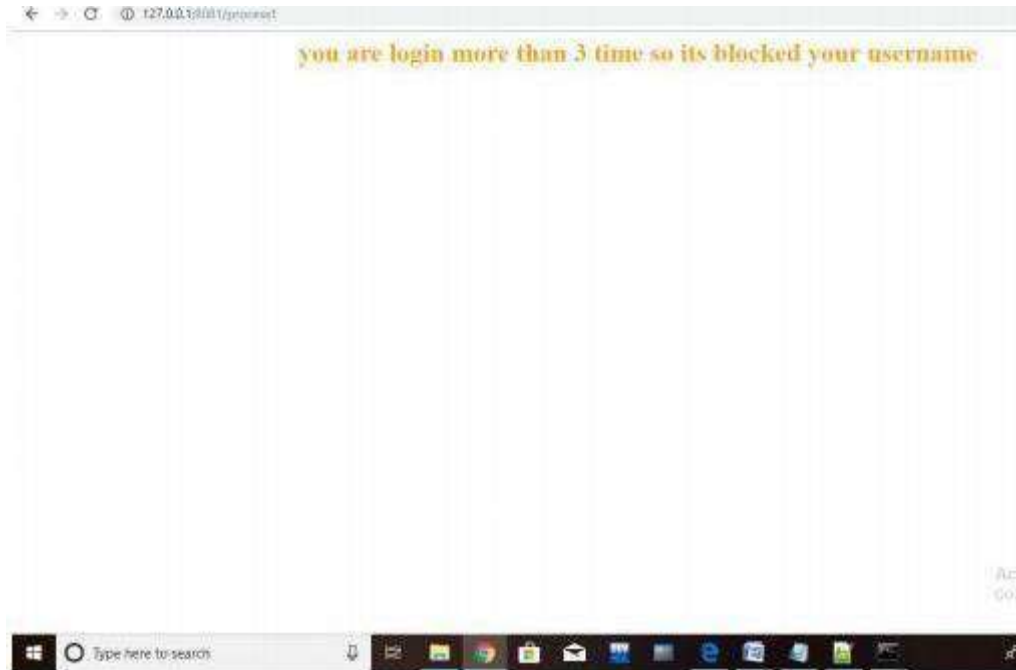


```

C:\Windows\system32\cmd.exe - mongo
> db.student.find()
> db.yasika.find()
> db.PROJECT.find()
> db.myProject.find()
> db.myProject.find().pretty()
> db.yasika.find().pretty()
> db.student.insert({'username':'yasotha',password:'Yasotha123*'})
WriteResult({ "nInserted" : 1 })
> db.student.find()
{ "_id" : ObjectId("5bd72f00078bb8238f3bef5c"), "username" : "yasotha", "password" : "Yasotha123*" }
> db.student.find().pretty()
{
  "_id" : ObjectId("5bd72f00078bb8238f3bef5c"),
  "username" : "yasotha",
  "password" : "Yasotha123*"
}
> db.student.insert({'username':'yasika',password:'Yasika123*'})
WriteResult({ "nInserted" : 1 })
> db.student.find().pretty()
{
  "_id" : ObjectId("5bd72f00078bb8238f3bef5c"),
  "username" : "yasotha",
  "password" : "Yasotha123*"
}
{
  "_id" : ObjectId("5bd72f3a078bb8238f3bef5d"),
  "username" : "yasika",
  "password" : "Yasika123*"
}

```





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.

```
const mongoose = require('mongoose');  
mongoose.connect('mongodb://localhost:27017/company',{useNew  
UrlParser: true});
```

```
const compSchema = new mongoose.Schema({  
  empid: Number,  
  vehicalno: String,  
  owner: String,  
  brand: String,  
  year: Number  
});
```

```
const Employee = mongoose.model("Employee",compSchema)
```

```
const aayushi =new employee({  
  empid: 1,  
  vehicalno: "A101",  
  owner: "Aayushi",  
  brand: "xyz",  
  year: 2020  
});
```

```
const aditi = new Employee({  
  empid: 2,  
  vehicalno: "A145",  
  owner: "aditi",  
  brand: "abc",  
  year: 2019  
});
```

```
const abhishek = new Employee({  
  empid: 3,  
  vehicalno: "D456",  
  owner: "Abhishek",  
  brand: "pod",  
  year: 2018  
});
```

```
const dhiraj = new Employee({  
  empid: 4,  
  vehicalno: "K908",  
  owner: "Dhiraj",  
  brand: "xyz",  
  year: 2020  
});
```

```
const aakash = new Employee({  
  empid: 5,  
  vehicalno: "R666",
```

```
    owner: "Aakash",
    brand: "wer",
    year: 2017
  });

const ankush = new Employee({
  empid: 6,
  vehicalno: "A101",
  owner: "ankush",
  brand: "iot",
  year: 2014
});

const ishita = new Employee({
  empid: 7,
  vehicalno: "B546",
  owner: "Ishita",
  brand: "pqr",
  year: 2019
});

const raunak = new Employee({
  empid: 8,
  vehicalno: "A101",
  owner: "Raunak",
  brand: "xyz",
  year: 2020
});

const isha = new Employee({
  empid: 9,
  vehicalno: "WW32",
  owner: "isha",
  brand: "iot",
  year: 2000
});

const soma = new Employee({
```

```
    empid: 10,  
    vehicalno: "A101",  
    owner: "Soma",  
    brand: "xyz",  
    year: 2020  
  });
```

```
Employee.insertMany([aayushi,aditi,abhishek,dhiraj,aakash,an  
kush,ishita],function(err){  
  if(err){  
    console.log(err);  
  }else{  
    console.log("successfully saved all the employees")  
  }  
});
```

```
<!DOCTYPE html>  
<html lang="en" dir="ltr">  
  <head>  
    <meta charset="utf-8">  
    <title>19BIT0233</title>  
    <style media="screen">  
      body{  
        text-align: center;  
        align-items: center;  
        justify-content: center;  
      }  
  
      form{  
        text-align: center;  
        align-items: center;  
        justify-content: center;  
      }  
  
      table{
```

```

        margin: auto;
    }

    td{
        padding: 10px 20px;
    }

    #submit-button{
        padding: 5px;
        background-color: lightblue;
        margin: 20px;
    }
</style>
</head>
<body>
    <h1>Employee Vehicle detail form</h1>
    <form action="/confirm" method="post">
        <table>
            <tr>
                <td>Employee Id:</td>
                <td><input type="number" name="empid"
value=""></td>
            </tr>
            <tr>
                <td>Vehical No.:</td>
                <td><input type="text" name="vehno" value=""></td>
            </tr>
            <tr>
                <td>Owner:</td>
                <td><input type="text" name="owner" value=""></td>
            </tr>
            <tr>
                <td>Brand:</td>
                <td><input type="text" name="brand" value=""></td>
            </tr>
            <tr>
                <td>Year:</td>

```



```
        <td><input type="number" name="year"
value=""></td>
    </tr>
</table>
    <input type="submit" name="" value="submit"
id="submit-button">
</form>
</body>
</html>
```

```
<script>
    var express=require("express");
    var bodyParser=require("body-parser");

    const mongoose = require('mongoose');
    mongoose.connect('mongodb://localhost:27017/company',{useNew
UrlParser: true});

    var app=express()

    app.use(bodyParser.json());
    app.use(express.static('public'));
    app.use(bodyParser.urlencoded({
        extended: true
    }));

    const compSchema = new mongoose.Schema({
        empid: Number,
        vehicalno: String,
        owner: String,
        brand: String,
        year: Number
    });

    const Employee = mongoose.model("Employee",compSchema)
```

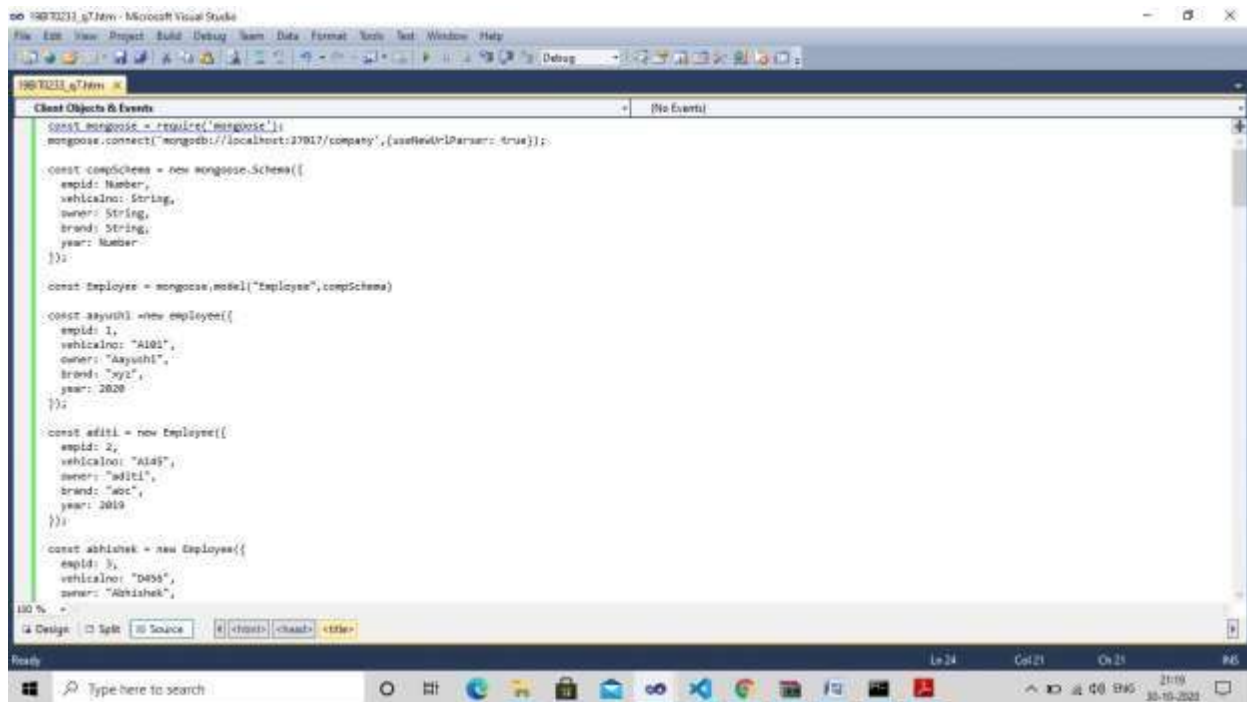
```

app.post('/confirm', function(req,res){
    var empid = req.body.empid;
    var vehno = req.body.vehno;
    var owner = req.body.owner;
    var brand = req.body.brand;
    var year = req.body.year;

    Employee.findOneAndUpdate({empid: empid},{vehicalno:
vehno, owner:owner,brand: brand, year: year },function(err){
        if(err){
            console.log(err);
        }else{
            res.send("Updation completed");
        }
    })
})

app.get("/",function(req,res){
    res.sendFile(_dirname + "/index.html")
})
app.listen(3000,function(){
    console.log("server has started on 3000");
})
</script>
</body>
</html>

```



```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/company', {useNewUrlParser: true});

const compSchema = new mongoose.Schema({
  empId: Number,
  vehicalno: String,
  owner: String,
  brand: String,
  year: Number
});

const Employee = mongoose.model('Employee', compSchema)

const aayush1 = new Employee({
  empId: 1,
  vehicalno: "A1001",
  owner: "Aayush1",
  brand: "xyz",
  year: 2020
});

const aditi1 = new Employee({
  empId: 2,
  vehicalno: "A1045",
  owner: "aditi",
  brand: "aoc",
  year: 2019
});

const abhishek = new Employee({
  empId: 3,
  vehicalno: "D056",
  owner: "Abhishek",
  year: 2018
});
```

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

- 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
- The player if the name of the player is entered in find and no radio button is checked.
- The players of a particular country representing IPL. If the radio button IPL is clicked and country name is entered in find
- 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.
- Store the data in MongoDB database

Code:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Index</title>
</head>
<body>
    <form action="/show" method="POST">
        <label for="find">FIND</label>
        <input type="text" name="find" id="find"
placeholder="find"><br><br>
        <input type="radio" name="find_details" id="ipl"
value="ipl">IPL<br>
        <input type="radio" name="find_details"
id="country" value="country">country<br>
        <input type="radio" name="find_details" id="bid"
value="bid">bid<br>
        <br>
        <button>submit</button>
    </form>
</body>
</html>
```

Show.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>IN SHOW</title>
```

```

    <style>
      table, th, td {
        border: 1px solid black;
      }
    </style>
</head>

<body>
<table>
  <tr>
    <th>NAME</th>
    <th>FRANCHISE</th>
    <th>COUNTRY</th>
    <th>BID AMOUNT</th>
  </tr>
  <% for(let p of players) { %>
    <tr>
      <td><%= p.name %></td>
      <td><%= p.franchise %></td>
      <td><%= p.country %></td>
      <td><%= p.bid_amount %></td>
    </tr>
    <% }%>
  </table>
</body>

</html>

```

Showp.html

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>IN SHOW</title>

```

```
<style>
  table, th, td {
    border: 1px solid black;
  }
</style>
</head>

<body>
<table>
  <tr>
    <th>NAME</th>
  </tr>
  <% for(let p of players) { %>
    <tr>
      <td><%= p.name %></td>
    </tr>
  <% }%>
</table>
</body>

</html>
```

```
Index.js
const express = require('express');
const app = express();
const path = require('path');
const mongoose = require('mongoose');
const methodOverride = require('method-override')
const player = require('./models/player');

mongoose.connect('mongodb://localhost:27017/ipl', {
  useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log("MONGO CONNECTION OPEN!!!")
  })
  .catch(err => {
    console.log("OH NO MONGO CONNECTION ERROR!!!!")
    console.log(err)
```

```
    })

    app.set('views', path.join(_dirname, 'views'));
    app.set('view engine', 'ejs');

    app.use(express.urlencoded({ extended: true }));
    app.use(methodOverride('_method'))

    app.post('/show', async (req, res) => {
        const checkb = (req.body);
        console.log(checkb);
        if (checkb.find_details === undefined) {
            var chn = await player.find({ name: req.body.find
        });
            if (chn.length !== 0) {
                res.render('show', { players: chn });
            }
            else {
                res.send("no name given");
            }
        }
        else if (checkb.find_details === 'ipl') {
            var chn = await player.find({ country: req.body.find
        });
            if (chn.length !== 0) {
                res.render('showp', { players: chn });
            }
            else {
                res.send("invalid");
            }
        }
        else if (checkb.find_details === 'bid') {
            var chn = await player.find({ bid_amount: { $gte:
req.body.find } });
            console.log(chn);
            if (chn.length !== 0) {
                res.render('showp', { players: chn });
            }
        }
    })
}
```

```
        else {
            res.send("invalid");
        }
    }
    else {
        res.send("Oopsy!!You got it wrong");
    }
})

app.get('/', (req, res) => {
    res.render('index');
})

app.listen(5000, () => {
    console.log("APP IS LISTENING ON PORT 5000!")
})
```

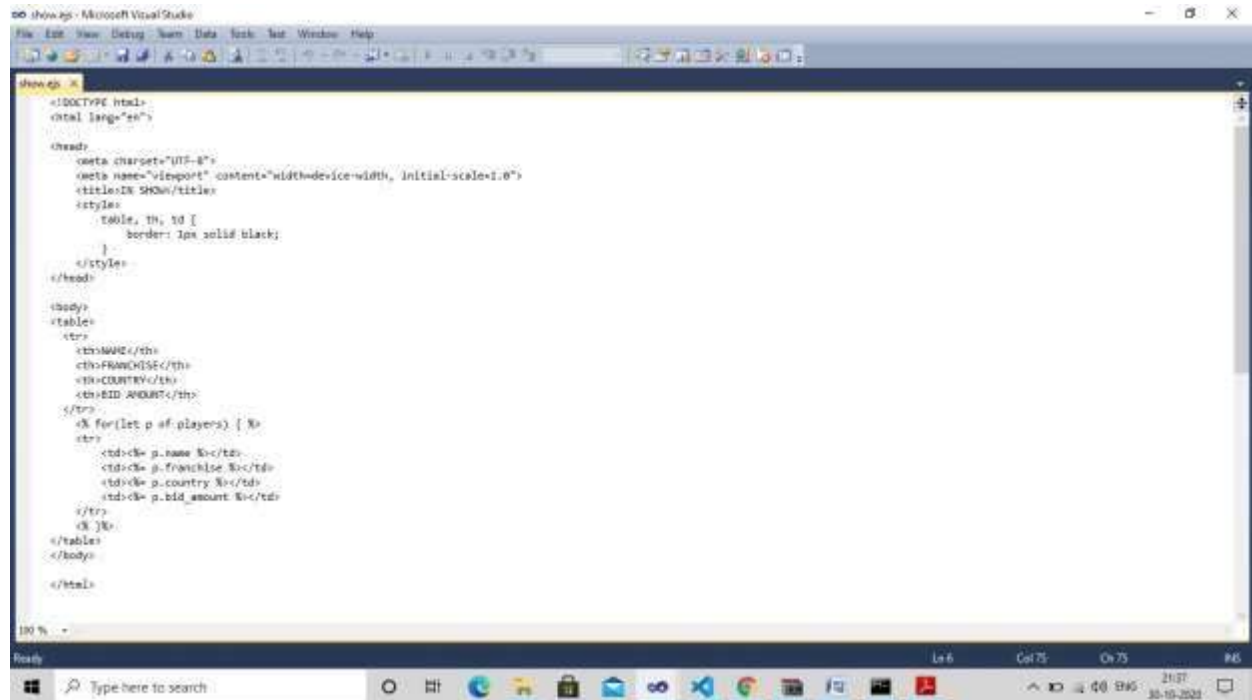
Player.js

```
const mongoose = require('mongoose');

const playerSchema = new mongoose.Schema({
    name: String,
    franchise: String,
    country: String,
    bid_amount: Number
})

const player = mongoose.model('player', playerSchema);

module.exports = player;
```

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>IN 2020</title>
  <style>
    table, th, td {
      border: 1px solid black;
    }
  </style>
</head>

<body>
  <table>
    <tr>
      <th>NAME</th>
      <th>FRANCHISE</th>
      <th>COUNTRY</th>
      <th>BID AMOUNT</th>
    </tr>
    <tr>
      <td>for(let p of players) { %>
      <tr>
        <td>${p.name}</td>
        <td>${p.franchise}</td>
        <td>${p.country}</td>
        <td>${p.bid_amount}</td>
      </tr>
    </tr>
  </table>
</body>
</html>
```