

LAB SLOT: L23+L24

DIGITAL ASSIGNMENT 3: Angular JS, Node JS, Express JS and MongoDB

```
<html>  
<head>  
<title>18BIT0048 q1</title>  
<script src="angular.min.js"></script>  
<style>  
div{  
border: 1px solid blue;  
}  
input.ng-invalid  
{  
background-color:orange;  
}  
input.ng-valid  
{  
background-color:green;  
}  
</style>  
</head>  
<body ng-app="">  
<form name="myform">  
<div style="width:30.55%;color:blue;height:4%"><b>Shipping Address</b></div>  
<div style="width:30%;background-color:#FFE5B4;padding:4px">  
<p>  
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<font color="red">*</font><b>Name</b>:  
<input type="text" name="myname" ng-model="myname"  
ng-pattern="/^[a-zA-Z]{3,30}$/" required>  
<span style="color:red" ng-show="myform.myname.$touched  
&&myform.myname.$error.required">The name is required.</span>  
</p>
```



```
<h5>Zip:{{myform.myzip.$valid}}</h5>
</body>
</html>
```

OUTPUT SCREENSHOT: form with wrong data:

Shipping Address

*Name: The name is required.

*Address: The address is required.

Address: (optional)

*City: The city name is required.

*State: The state name is required.

*Zip: The zip is required.

Submit

Name:false

Address:false

Address:true

City:false

State:false

Zip:false

OUTPUT SCREENSHOT: form with correct data:

Shipping Address

*Name: The name is required.

*Address: The address is required.

Address: (optional)

*City: The city name is required.

*State: The state name is required.

*Zip: The zip is required.

Submit

Name:true

Address:true

Address:true

City:true

State:true

Zip:true

QUESTION 2: i) Create an angular shopping application to buy books and remove if not required.

ii) If the user enters a book in the text box and clicks addbook button, the book is added and if the user clicks on cross symbol next to each book given in the picture below, the book is removed from the list.

iii) If the book is selected twice, display an appropriate error message. Use angular animations to add and remove items.

HTML CODE:

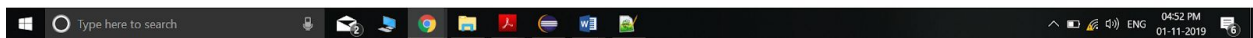
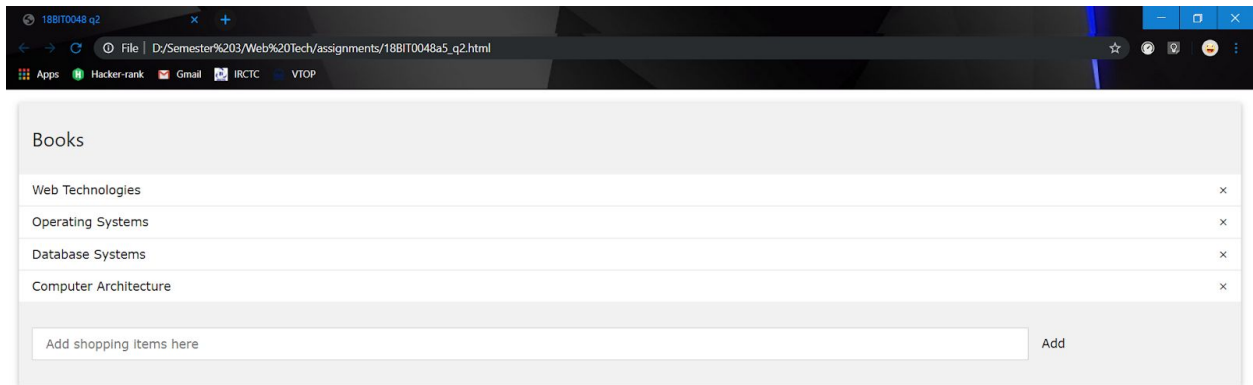
```
<!doctype html>
<html>
<head>
<title>18BIT0048 q2</title>
<script src="angular.min.js"></script>
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
</head>
<body ng-app="myShoppingList" ng-controller="myCtrl">
<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope)
{
$scope.products = ["Web Technologies","Operating Systems","Database
Systems","Computer Architecture"];
$scope.addItem = function()
{
$scope.errortext = "";
if(!$scope.addMe)
{
return;
}
if($scope.products.indexOf($scope.addMe) == -1)
{
$scope.products.push($scope.addMe);
}
else {
$scope.errortext = "The item is already in your shopping list.";
}
}
$scope.removeItem = function(x){
$scope.errortext = "";
$scope.products.splice(x, 1);
}
});
</script>
```

```

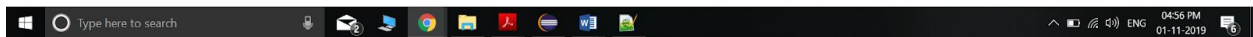
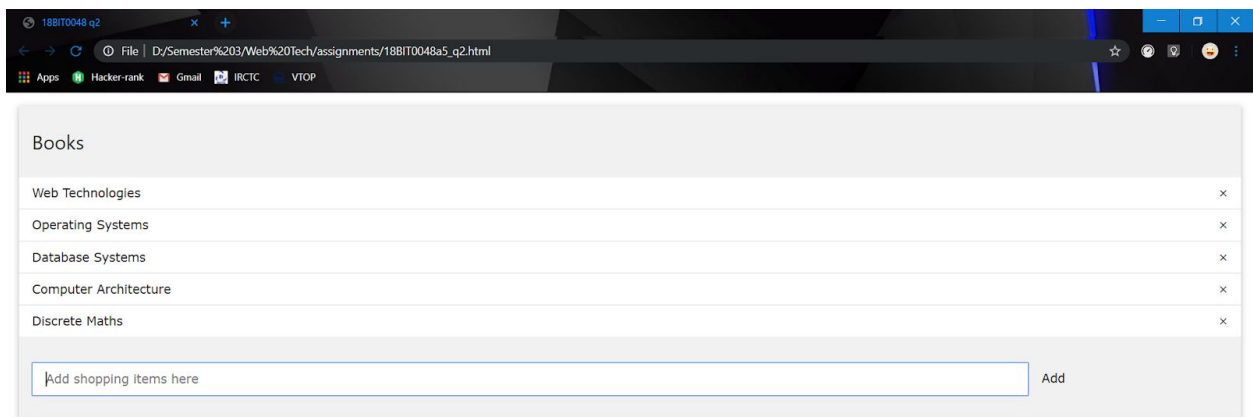
<div ng-app="myShoppingList" ng-cloak ngcontroller="myCtrl" class="w3-card-2
w3-margin" style="maxwidth:400px;">
<header class="w3-container w3-light-grey w3-padding-16">
<h3>Books</h3>
</header>
<ul class="w3-ul">
<li ng-repeat="x in products" class="w3-
padding16">{{x}}
<span ng-click="removeItem($index)"
style="cursor:pointer;" class="w3-right w3-marginright">x</span>
</li>
</ul>
<div class="w3-container w3-light-grey w3-padding-16">
<div class="w3-row w3-margin-top">
  <div class="w3-col s10">
<input placeholder="Add shopping items here" ng-model="addMe" class="w3-input
w3-border w3-padding">
  </div>
<div class="w3-col s2">
<button ng-click="addItem()" class="w3-btn w3-padding w3- green">Add</button>
</div>
</div>
<p class="w3-text-red">{{errortext}}</p>
</div>
</div>
</body>
</html>

```

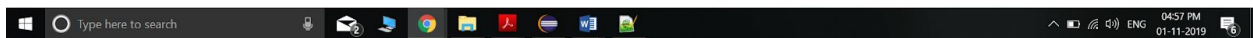
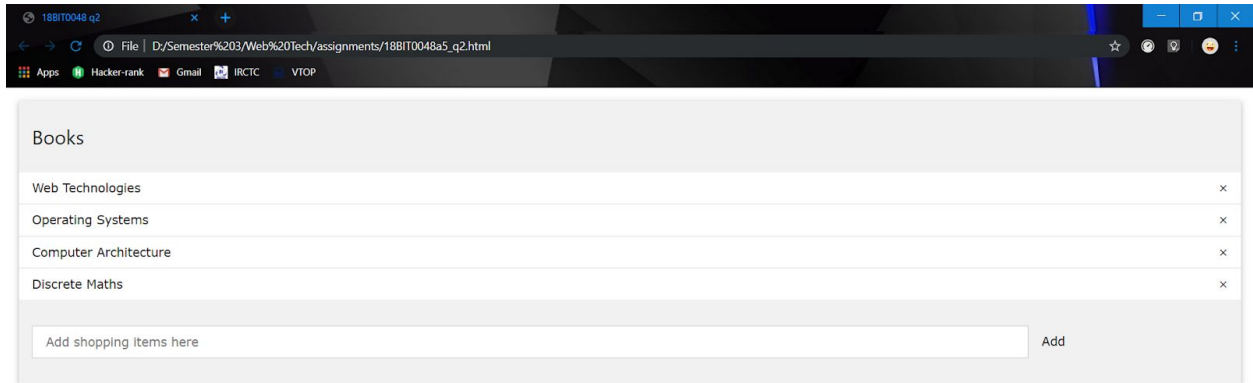
OUTPUT SCREENSHOT: initial webpage:



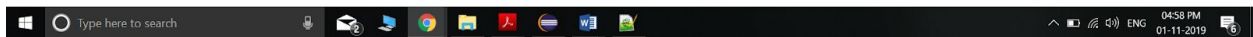
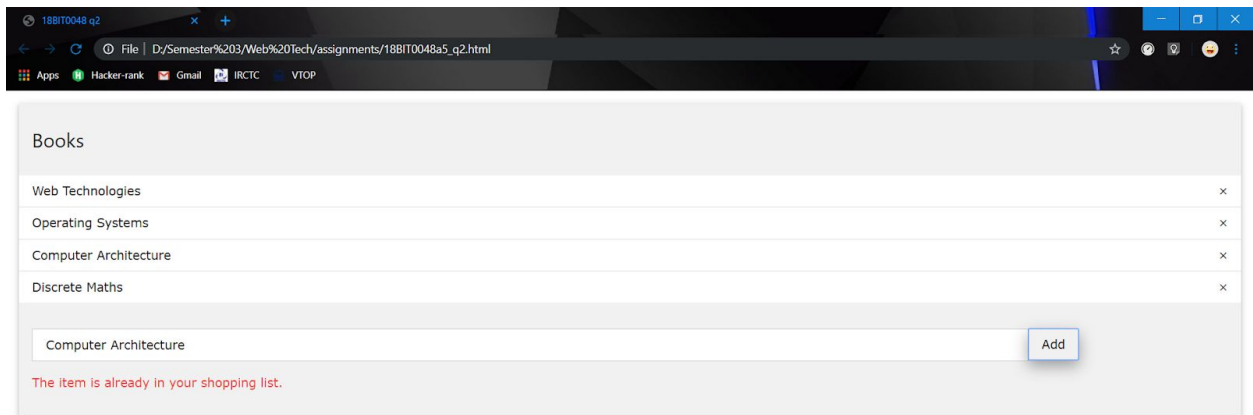
OUTPUT SCREENSHOT: after adding “discrete maths” book



OUTPUT SCREENSHOT: after deleting Database systems using cross button



OUTPUT SCREENSHOT: after entering Computer Architecture again for adding



QUESTION 3: i) Create an angular routing application that displays the default page as the order details page containing three orders as shown below.

ii) When the “show details” is clicked for a particular order, the order details of particular order number must be displayed below.

HTML CODE: main page:

```
<!doctype html>
<html >
<head>
  <title>18BIT0048 q3</title>
  <script
src="angular.min.js"></script>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular-route.js"></
script>
  <script src="script.js"></script>

  <style>
  div{
  background-color:#D1D0CE;
  color:#000;
  borderwidth:3px;
  border-color:lightgreen;
  border-style:solid;
  height:100px
  }
</style>
</head>
<body ng-app="myApp">
<table style="width:100%;" border="1" bordercolor="transparent"
cellpadding="4px">
  <tr bgcolor="#D1D0CE">
    <th align="left">#</th>
    <th align="left"><b>Order No.</b></th>
    <th align="left"><b>Details</b></th>
    <th align="left"><b></b></th>
  </tr>
  <tr bgcolor="#E5E4E2">
    <td align="left">1</td>
    <td align="left">1234</td>
    <td align="left">15<sup>n</sup>Samsung Laptop</td>
    <td><b><a href="#!first">show details</a></b></td>
  </tr>
  <tr bgcolor="#D1D0CE">
    <td>2</td>
    <td>5412</td>
    <td>2TB Segate Hard drive</td>
    <td><b><a href="#!second">show details</a></b></td>
  </tr>
  <tr bgcolor="#E5E4E2">
    <td>3</td>
    <td>9874</td>
```



```

        <td>D-link router</td>
        <td><b><a href="#!third">show details</a></b></td>
    </tr>
</table> <br>
<div ng-view></div>

</body>
</html>

```

JAVASCRIPT CODE: script.js

```

var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
        .when("/", { template: "Order Details" })
        .when("/first", { templateUrl: "first.html" })
        .when("/second", { templateUrl: "second.html" })
        .when("/third", { templateUrl: "third.html" })
        .otherwise({ redirectTo: "index.html" });
}).config(function($locationProvider) {
    $locationProvider.hashPrefix('!');
});

```

HTML CODE: first.html

```

<html>
<div>
    <b>ORDER #1234<br>HERE ARE THE DETAILS FOR ORDER
    #1234</b>
</div>
</html>

```

HTML CODE: second.html

```

<html>
<div>
    <b>ORDER #5124<br>HERE ARE THE DETAILS FOR ORDER
    #5124</b>
</div>
</html>

```

HTML CODE: third.html

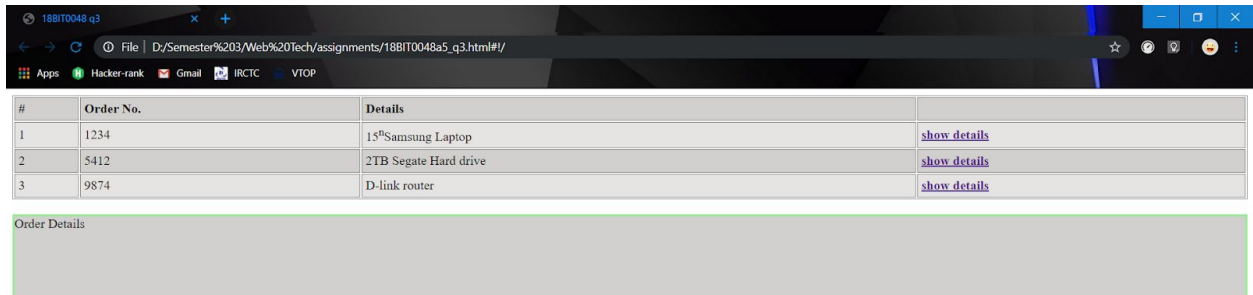
```

<html>
<div>
    <b>ORDER #9874<br>HERE ARE THE DETAILS FOR ORDER
    #9874</b>
</div>

```

</html>

OUTPUT SCREENSHOT: initial website:



OUTPUT SCREENSHOT: after clicking “show details” for order 9874:

#	Order No.	Details	
1	1234	15 th Samsung Laptop	show details
2	5412	2TB Segate Hard drive	show details
3	9874	D-link router	show details

ORDER #9874

HERE ARE THE DETAILS FOR ORDER #9874

QUESTION 4: i) Create a web application with username in HTML and node.js using express framework to handle different types of HTTP request namely get, post, put, options and delete.

ii) Provide different route paths for each of the request.

iii) Display the different request types along with the username in the browser when the application is executed.

HTML CODE:

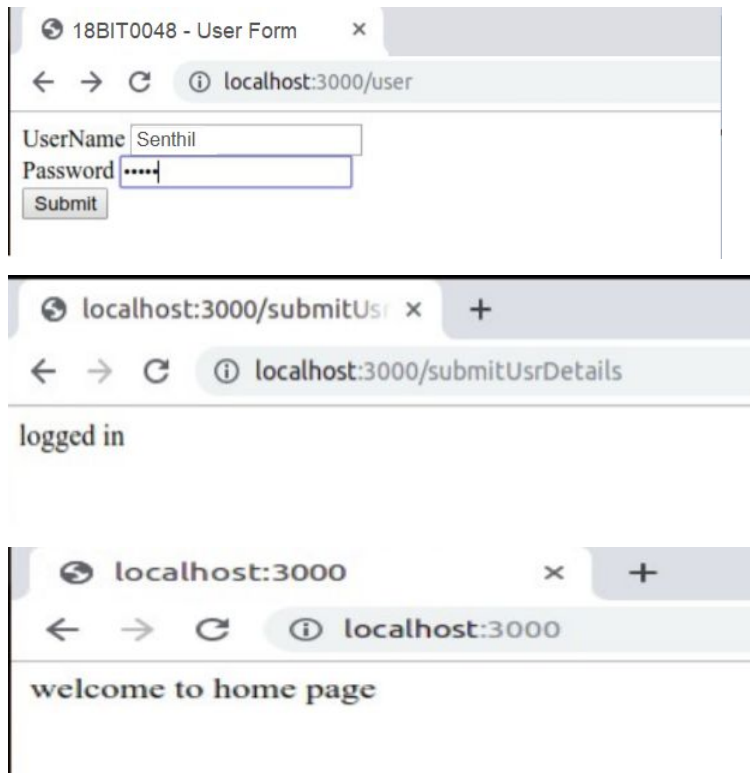
```
<html>
  <head>
<title>18BIT0048-User Form</title>
</head>
  <body>
    <form action="/submitUsrDetails" method="POST">
      <label name="userId">UserName</label>
      <input type="text" name="userId" value="">
      <br>
      <label name="password">Password</label>
      <input type="password" name="password" ><br>
      <button type="submit" name="submit">Submit</button>
    </form>
  </body>
</html>
```

NODE JS CODE:

```
const express=require('express');
let ejs=require('ejs');
const app=express();
app.get('/',(req,res)=>{
  res.send('welcome to home page');
});
app.put('/put', function (req, res) {
  res.send('Got a PUT request');
});
app.delete('/delete', function (req, res) {
  res.send('Got a DELETE request ');
})

app.get('/user',(req,res)=>{
  res.render('form.ejs');
});
app.post('/submitUsrDetails',(req,res)=>{
  console.log(res.body);
  res.send('logged in');
});
app.listen(3000,(err)=>{
  if (err) console.log(err);
  else{
    console.log("listening on port 3000");
  }
});
```

OUTPUT SCREENSHOTS:



- QUESTION 5:**
- i) Design an application using node.js and mongodb to perform the following operations in a student collection with 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 student with name='Kevin' to age=25
 - v) Limit the result to 4 documents
 - vi) Query the document on the basis of age>25

Ans i) NODE JS CODE:

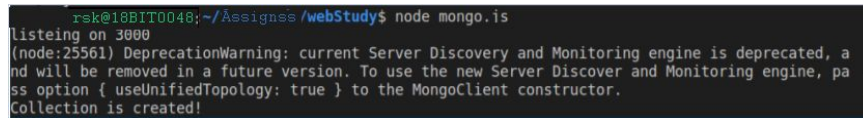
```
const express=require('express');
let ejs=require('ejs');
const app=express();
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/MongoDatabase";
app.get('/createDb', (req,res)=>{
  // create database and collection on mongo db using node js
  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    dbo.createCollection("studentRecords", function(err, res) {
```

```

if (err) throw err;
console.log("Collection is created!");
res.send("collection is created");
db.close();
});
});
});

```

OUTPUT SCREENSHOT:



```

rsk@18BIT0048:~/Assigns/webStudy$ node mongo.js
listing on 3000
(node:25561) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, please option { useUnifiedTopology: true } to the MongoClient constructor.
Collection is created!

```

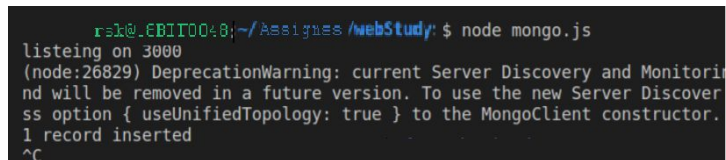
Ans ii)CODE for one record:

```

app.get('/insertOne', (req, res) => {
  // insert one record into database
  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    var myobj = { name: "Pankaj Shukla", age: "18", dob: "19-JUL-2000", yoa: "2018" };
    dbo.collection("studentRecords").insertOne(myobj, function(err, res) {
      if (err) throw err;
      res.send("1 record inserted");
      console.log("1 record inserted");
      dbo.close();
    });
  });
});

```

OUTPUT SCREENSHOT:



```

rsk@18BIT0048:~/Assigns/webStudy$ node mongo.js
listing on 3000
(node:26829) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, please option { useUnifiedTopology: true } to the MongoClient constructor.
1 record inserted
^C

```

Ans ii)CODE for multiple records:

```

app.get('/insertMany', (req, res) => {
  // insert many records at once
  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    var myobj = [{ name: "Pankaj Shukla", age: "18", dob: "19-jul-2000", yoa: "2018" },
      { name: "Somya Sethi", age: "25", dob: "18-aug-2000", yoa: "2018" },
      { name: "Kevin", age: "20", dob: "18-jan-2000", yoa: "2018" },
    ];
    dbo.collection("studentRecords").insertMany(myobj, function(err, res) {
      if (err) throw err;
      res.send("Multiple records inserted");
      console.log("Multiple records inserted");
      dbo.close();
    });
  });
});

```

```

{ name:"david",age:"25",dob:"18-feb-2000",yoa:"2018"},
{ name:"Gunjan ",age:"20",dob:"18-sept-2000",yoa:"2017"},
{ name:"Senthil ",age:"20",dob:"17-oct-2000",yoa:"2018"},
];
dbo.collection("studentRecords").insertMany(myobj, function(err, res) {
if (err) throw err;

console.log("multiple record inserted");
db.close();
});
res.send("multiple record inserted");
});
});

```

OUTPUT SCREENSHOT:



```

rsk@18BIT0048:~/Assigns/webStudy$ node mongo.js
listening on 3000
(node:27168) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, please option { useUnifiedTopology: true } to the MongoClient constructor.
multiple record inserted

```

Ans iii)CODE for sorting records:

```

app.get('/sort',(req,res)=>{
// sort and then show in console
MongoClient.connect(url, function(err, db) {
if (err) throw err;
var dbo = db.db("mydb");
var mysort = { name: 1 };
dbo.collection("studentRecords").find().sort(mysort).toArray(function(err,
result) {
if (err) throw err;
console.log(result);
db.close();

});
res.send('sorted');
});

```

OUTPUT SCREENSHOT:

```

listing on 3000
(node:29601) DeprecationWarning: current Server
nd will be removed in a future version. To use
ss option { useUnifiedTopology: true } to the
[ { _id: 5dbc424c71914770db5482ab,
  name: 'Gunjan ',
  age: '20',
  dob: '18-sept-2000',
  yoa: '2017' },
  { _id: 5dbc424c71914770db5482a9,
  name: 'Kevin',
  age: '25',
  dob: '18-jan-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482a7,
  name: 'Pankaj Shukla',
  age: '18',
  dob: '19-JUL-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482ac,
  name: 'Senthil ',
  age: '20',
  dob: '17-oct-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482a8,
  name: 'Somya Sethi',
  age: '25',
  dob: '18-aug-2000',
  yoa: '2018' } ],

```

Ans iv)CODE for update:

```

app.get('/updateOne', (req, res) => {
  // updating the record
  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    var myquery = { name: "Kevin" };
    var newvalues = { $set: {age:"25"} };
    dbo.collection("studentRecords").updateOne(myquery, newvalues, function(err,
res) {
      if (err) throw err;
      console.log("1 document updated");
      db.close();
    });
  });
});

```

OUTPUT SCREENSHOT:

```

  name: 'Somya Sethi',
  age: '25',
  dob: '18-aug-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482aa,
  name: 'david',
  age: '25',
  dob: '18-feb-2000',
  yoa: '2018' } ]
1 document updated

```

Ans v)CODE for limiting:

```

app.get('/limit', (req, res) => {

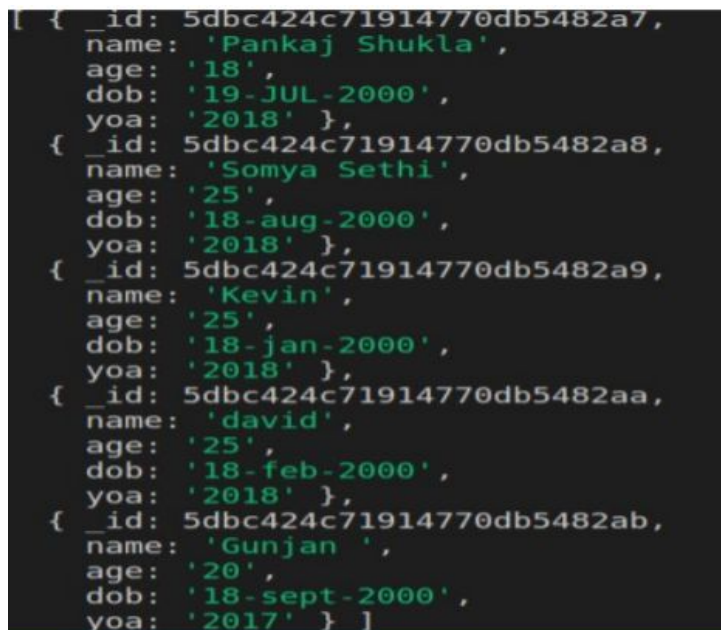
```

```

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("studentRecords").find().limit(5).toArray(function(err, result)
{
  if (err) throw err;
  console.log(result);
  db.close();
});
});
});

```

OUTPUT SCREENSHOT:



```

[ { _id: 5dbc424c71914770db5482a7,
  name: 'Pankaj Shukla',
  age: '18',
  dob: '19-JUL-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482a8,
  name: 'Somya Sethi',
  age: '25',
  dob: '18-aug-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482a9,
  name: 'Kevin',
  age: '25',
  dob: '18-jan-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482aa,
  name: 'david',
  age: '25',
  dob: '18-feb-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482ab,
  name: 'Gunjan ',
  age: '20',
  dob: '18-sept-2000',
  yoa: '2017' } ]

```

Ans vi)CODE for querying based on age:

```

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { age: "25" };
  dbo.collection("studentRecords").find(query).toArray(function(err, result) {
  if (err) throw err;
  console.log(result);
  db.close();
});
});
});

```

OUTPUT SCREENSHOT:


```

rsk@18BIT0048:~/Assignee/webStudy$ node mongo.js
listening on 3000
(node:29456) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, please add the --monitoring option to the MongoClient constructor.
[ { _id: 5dbc424c71914770db5482a8,
  name: 'Somya Sethi',
  age: '25',
  dob: '18-aug-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482a9,
  name: 'Kevin',
  age: '25',
  dob: '18-jan-2000',
  yoa: '2018' },
  { _id: 5dbc424c71914770db5482aa,
  name: 'david',
  age: '25',
  dob: '18-feb-2000',
  yoa: '2018' } ]

```

6)i) Write a HTML and node.js program for creating and storing session information of user. The HTML page contains username, password, remember me next time check box option and 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 invalid user username or password, display appropriate error message.

b. After successful login, display welcome message.

c. Allow the user to enter username and password for 3 times. If the user enters username and password more than 3 times, display the message “you are blocked”.

HTML CODE:main page

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>18BIT0048</title>
</head>
<body>
  <fieldset>
    <form method='POST' action='login'>
      <legend>LOGIN</legend>
      <label for='username'>Username</label><br>
      <input type='text' id='username' name='username'>
      <br>
      <label for='password'>Password</label><br>
      <input type='text' id='password' name='password'>
      <br>
      <input type='submit' value='LOGIN'>
    </form>
  </fieldset>

```

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

HTML CODE: invalidlogin html page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">

</head>
<body>
<p>invalid Userid or password</p>
</body>
</html>
```

HTML CODE: admin html page after successful login

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>profile</title>
</head>
<body>
    <h2>Hello rsk@vit</h2>
    <a href="">logout</a>
</body>
</html>
```

HTML CODE:excedMaxTimes html page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">

</head>
<body>
<p>You tried login more than three times you have been blocked</p>
</body>
</html>
```

NODE JS CODE:

```
const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');
const redis = require('redis');
const redisStore = require('connect-redis')(session);
const client = redis.createClient();
const router = express.Router();
const app = express();
var users=['rsk@vit','travis@vit','lee@vit'];
var passwd=['1234','tra#12','l@34'];
app.use(session({
  secret: 'vit',
  // create new redis store.
  store: new redisStore({ host: 'localhost', port: 6379, client: client,ttl :
260}),
  saveUninitialized: false,
  resave: false
}));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(express.static(__dirname + '/views'));
router.get('/',(req,res) => {
  let sess = req.session;
  if(sess.email) {
    return res.redirect('/admin');
  }
  res.sendFile('index.html');
});
router.post('/login',(req,res) => {
  req.session.email = req.body.email;
  if(req.session.email in users){

  }

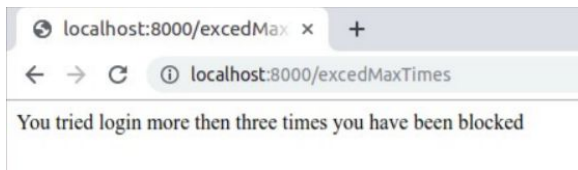
});
router.get('/admin',(req,res) => {
  if(req.session.email) {
    res.write(`<h1>Hello ${req.session.email} </h1><br>`);
    res.end('<a href='+'/logout'+>Logout</a>');
  }
  else {
    res.write('<h1>Please login first.</h1>');
    res.end('<a href='+'/'+>Login</a>');
  }
}
```

```

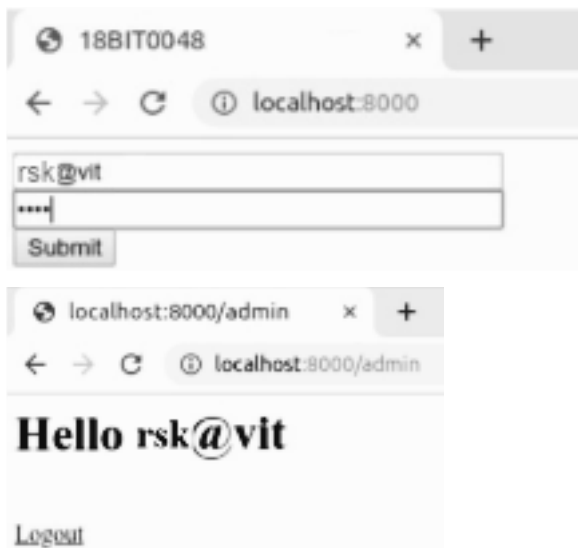
});
router.get('/logout', (req, res) => {
  req.session.destroy((err) => {
    if(err) {
      return console.log(err);
    }
    res.redirect('/');
  });
});
router.get('/excedMaxTimes', (req, res)=>{
  res.send('You tried login more then three times \n you have been blocked ');
});
router.get('/invalidLogin', (req, res)=>{
  res.send('Invalid Userid or password ');
});
app.use('/', router);
app.listen(8000, (err) => {
  if (err) throw err;
  console.log(`App Started on PORT 8000`);
});

```

OUTPUT SCREENSHOT: after trying 3 times



OUTPUT SCREENSHOT: correct user credentials



OUTPUT SCREENSHOT: wrong user credentials

