NAME:M.KAVINA

REG NO:18BIT0453

LAB CYCLESHEET-3

1) AngularJS

i) Create a shipping address form using html in the format given below.

ii) Write an angular script that will be invoked when form is validated successfully.

iii) Display the status messages using angular directives and angular class values for all form elements.

Note: If the required fields are not filled with data, an alert should be thrown.

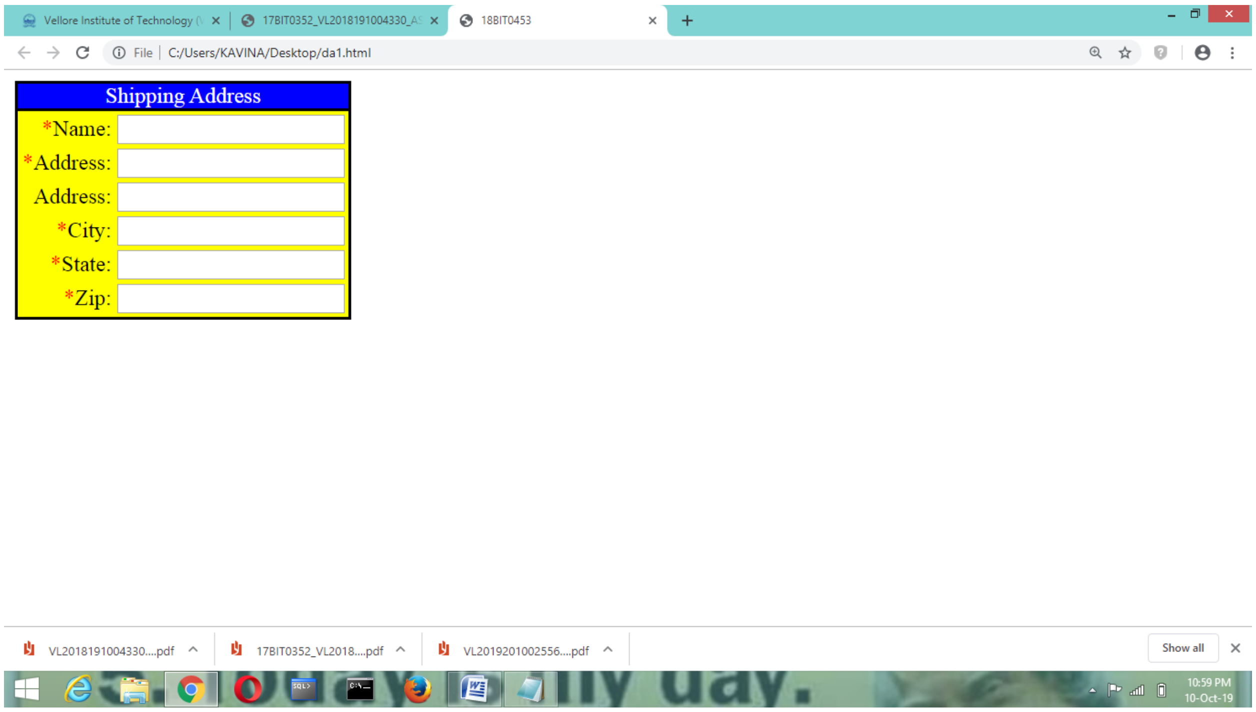
CODE:

```
<html>

<head>

<title>18BIT0453</title>

<script src="angular.min.js"></script>

<style>

table{

border: solid;

background-color: yellow;

}

caption{

border: solid;

border-bottom: none;

background-color: blue;

}

</style>

</head>
```

```
<body>

<form name="f1" ng-app="myApp" ng-controller="ctrl" novalidate>

<table>

<caption><font color='white' align="right">Shipping Address</font></caption>

<tr><td align='right'><font color='red'>*</font>Name:</td><td>

<input type="text" name="fname" ng-model="fname" required>

<span style="color:red" ng-show="f1.fname.$dirty&&f1.fname.$invalid">

<span ng-show="f1.fname.$error.required"></span></span></td></tr>

<tr><td align='right'><font color='red'>*</font>Address:</td><td>

<input type="text" name="addr" ng-model="addr" required>

<span style="color:red" ng-show="f1.addr.$dirty&&f1.addr.$invalid">

<span ng-show="f1.addr.$error.required">

</span></span></td></tr>

<tr><td align='right'>Address:</td><td><input type='text'></td></tr>

<tr><td align='right'><font color='red'>*</font>City:</td><td>

<input type="text" name="city" ng-model="city" required>

<span style="color:red" ng-show="f1.city.$dirty&&f1.city.$invalid">

<span ng-show="f1.city.$error.required"></span></span></td></tr>

<tr><td align='right'><font color='red'>*</font>State:</td><td>

<input type="text" name="state" ng-model="state" required>

<span style="color:red" ng-show="f1.state.$dirty&&f1.state.$invalid">

<span ng-show="f1.state.$error.required"></span></span>

<tr><td align='right'><font color='red'>*</font>Zip:</td><td>

<input type="text" name="zip" ng-model="zip" required>

<span style="color:red" ng-show="f1.zip.$dirty&&f1.zip.$invalid">
```

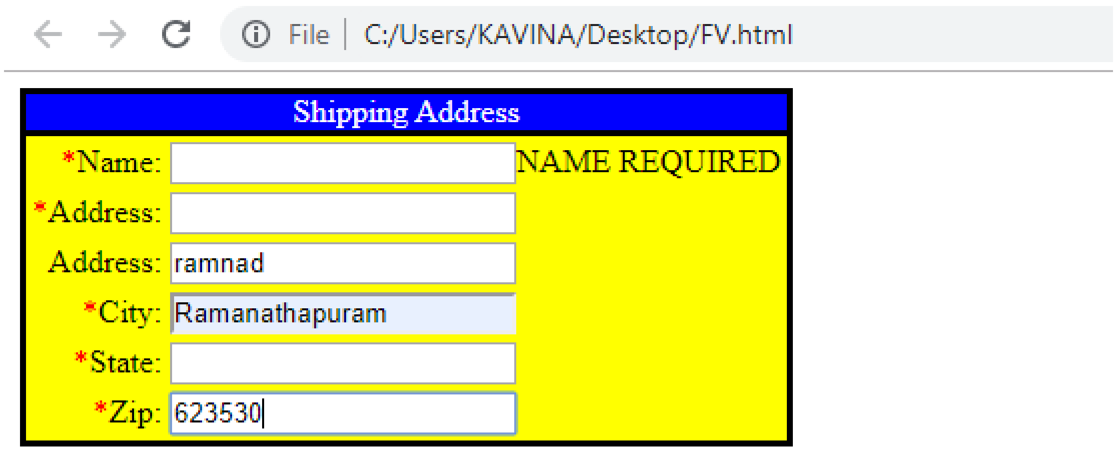
```
<span ng-show="f1.zip.$error.required"></span></span></td></tr>

</table>

</form>

<script>

var app=angular.module("myApp",[]);

app.controller("ctrl",function($scope){

$scope.fname="yasika";

$scope.addr="theni";

$scope.city="theni";

$scope.state="Tamil Nadu";

$scope.zip="629630";

});

</script>

</body>

</html>
```

OUTPUT:



Shipping Address

*Name:	
*Address:	
Address:	
*City:	
*State:	
*Zip:	



Shipping Address

*Name:		NAME REQUIRED
*Address:		
Address:	ramnad	
*City:	Ramanathapuram	
*State:		
*Zip:	623530	

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 add book 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.

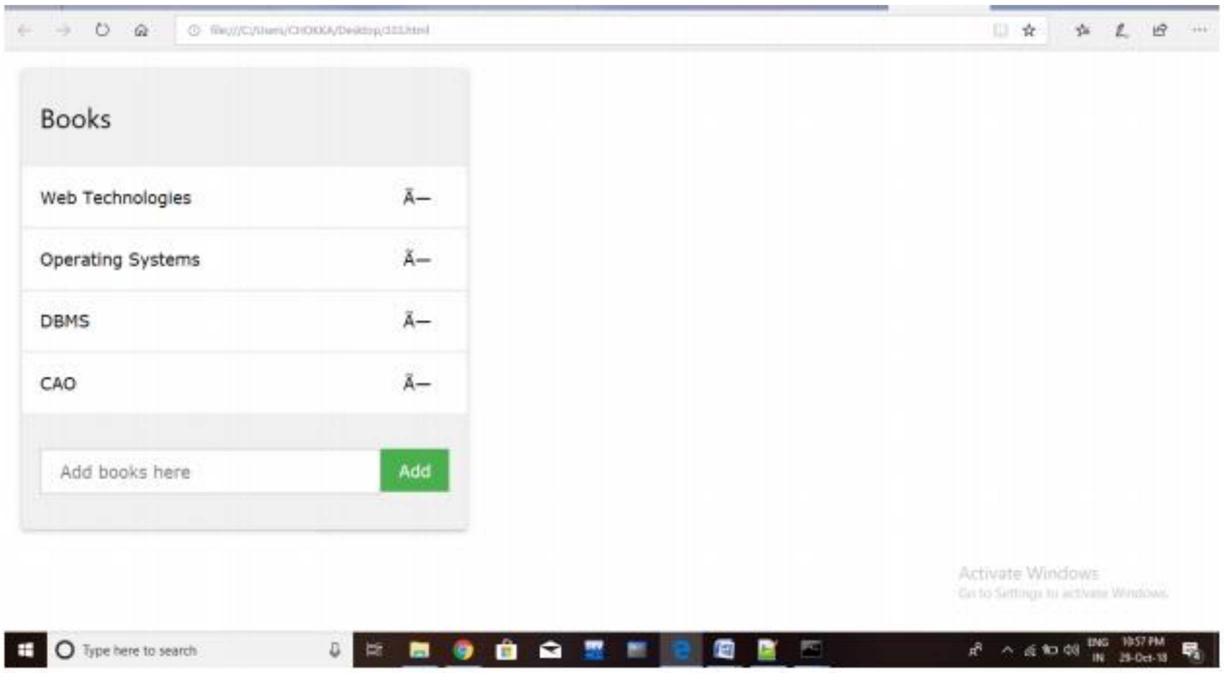
iii)If the book is selected twice, display an appropriate error message. Use angular animations to add and remove items

CODE:

```
});

<html>
<head>
<title>18BIT0453</title>
<script src="angular.min.js"></script>
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
</head>
<body>
<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope) {
 $scope.products = ["Web Technologies", "Operating Systems","DBMS","CAO"];
 $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 ng-controller="myCtrl" class="w3-card-2 w3-

margin" style="max-width: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-padding-16">{{x}}<span ngclick="removeItem($index)"
style="cursor:pointer;" class="w3-right w3-marginright">×</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 books 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:

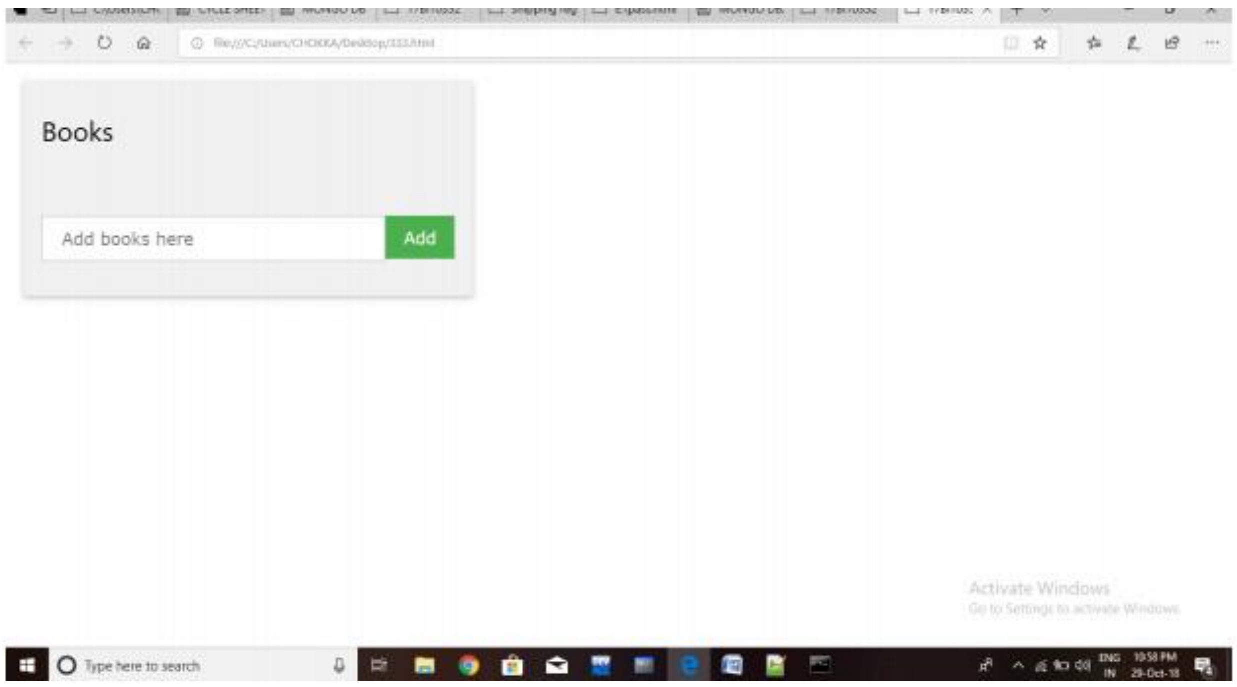

Books

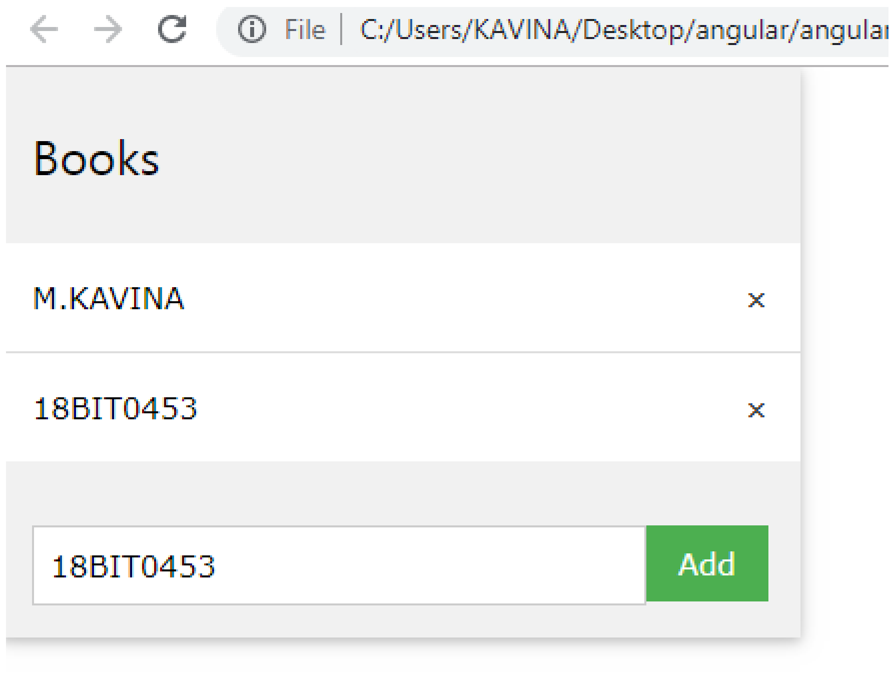
Web Technologies Ã—

Operating Systems Ã—

DBMS Ã—

CAO Ã—


Books


C:/Users/KAVINA/Desktop/angular/angula

Books

M.KAVINA ×

18BIT0453 ×

18BIT0453 Add

3)

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

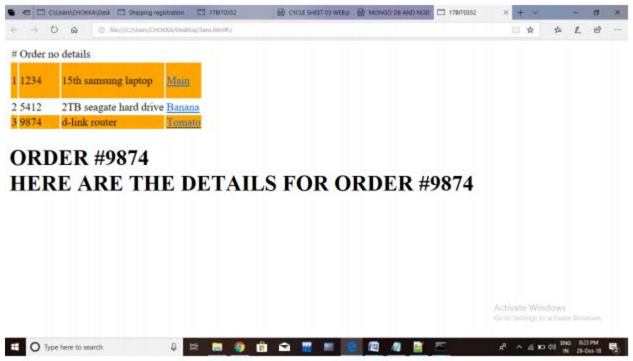
ii) When the “show details” is clicked for a particular order, the order details of particular order number must be displayed below.

PROGRAM:

```
<!DOCTYPE html><html>

<title>17BIT0352</title>

<script src="angular.min.js"> </script>

<script src="angular-route.js"></script>

<style>

table tr:nth-child(odd){background-color:white;border:none;}

table{border:#ff0000 #ooff01;}

table tr:nth-child(even){background-color:orange;
}

</style>

<body ng-app="myApp">

<form>

<table>

<tr><td>

#</td>

<td>Order no</td>

<td>details</td>

<td></td></tr>

<tr>

<td>1

</td>

<td>1234

</td>

<td>15th samsung laptop

</td>

<td><p><a href="#/!">show details</a></p>

</td>

</tr>

<tr>

<td>2

</td>

<td>5412

</td>
```

```
<td>2TB seagate hard drive

</td>

<td>

<a href="#!show details">show details</a>

</td>

</tr>

<tr>

<td>3

</td>

<td>9874

</td>

<td>d-link router

</td>

<td>

<a href="#!show details">show deatils</a>

</td>

</tr>

</table>

</form>

<div ng-view></div><script>

var app = angular.module("myApp", ["ngRoute"]);

app.config(function($routeProvider) {

$routeProvider

.when("/show details", {

template: "<h1>ORDER #1234<BR>HERE ARE THE DETAILS FOR ORDER # 1234</p>" })
.when("/show details", {
template : "<h1>ORDER #5412<BR>HERE ARE THE DETAILS FOR ORDER #5412</p>" })
.otherwise({
template : "<h1>ORDER #9874<BR>HERE ARE THE DETAILS FOR ORDER #9874</p>"});
});
</script>
</body>
</html>
```

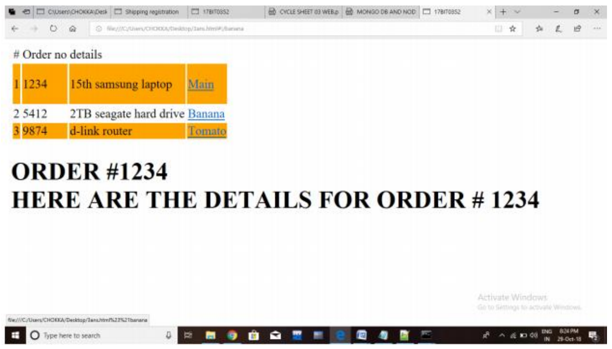
OUTPUT:



# Order no details

1	1234	15th samsung laptop	Main
2	5412	2TB seagate hard drive	Banana
3	9874	d-link router	Tomato

ORDER #9874
HERE ARE THE DETAILS FOR ORDER #9874



# Order no details

1	1234	15th samsung laptop	Main
2	5412	2TB seagate hard drive	Banana
3	9874	d-link router	Tomato

ORDER #1234
HERE ARE THE DETAILS FOR ORDER # 1234

4) Node.js andMongoDB

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 there quest.

iii)Display the different request types along with the username in the browser when the

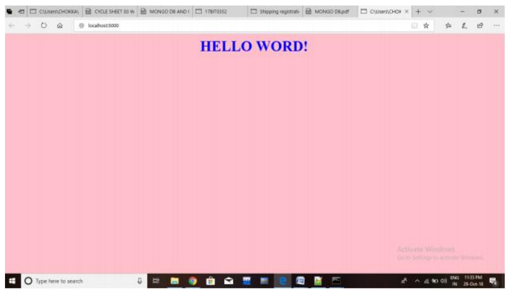
application is executed.

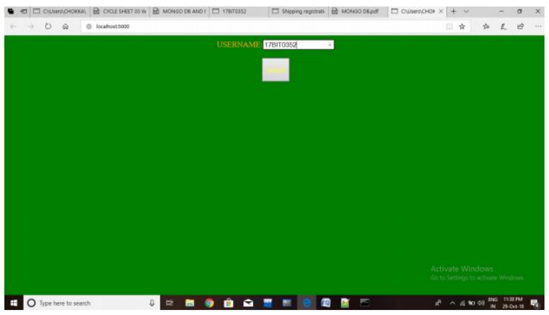
PROGRAM:

```
<!DOCTYPEhtml>
<title>18BIT0453</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/>

<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('/submit-student-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>
```

OUTPUT:

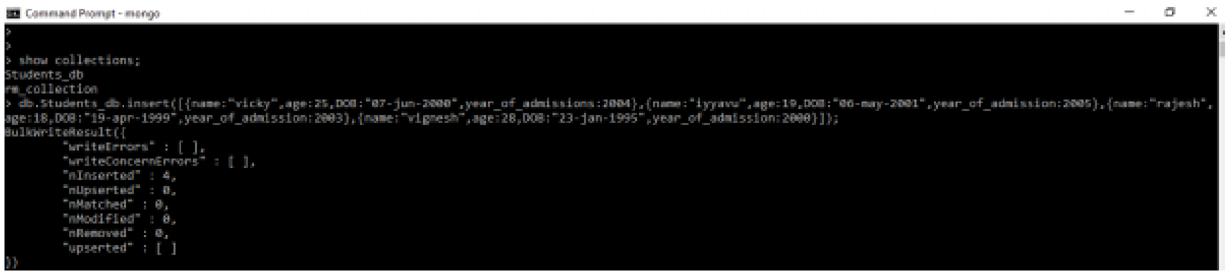




6)

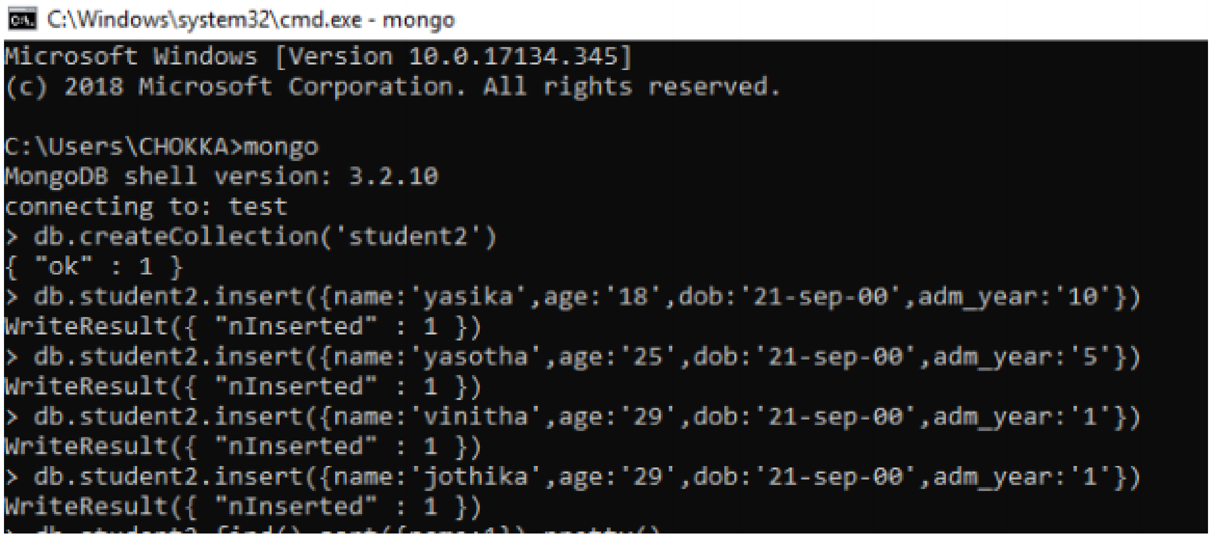
i) Design an application using node.js and mongodb to perform the following

operations in a student collection with name, age, DOB and year of admission.



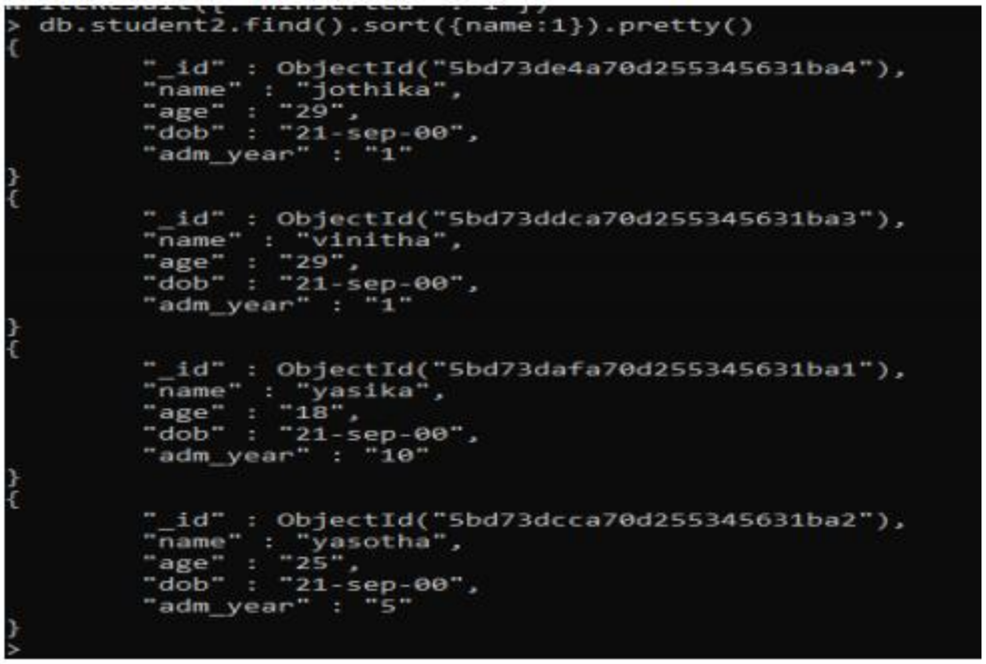
```
>
>
> show collections;
Students_db
rm_collection
> db.Students_db.insert([{name:"vicky",age:25,DOB:"07-jun-2000",year_of_admissions:2004},{name:"iyyavu",age:19,DOB:"06-may-2001",year_of_admission:2005},{name:"rajesh",
age:18,DOB:"19-apr-1999",year_of_admission:2003},{name:"vignesh",age:28,DOB:"23-jan-1995",year_of_admission:2000}]);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

ii)Insert multiple records into the collection.



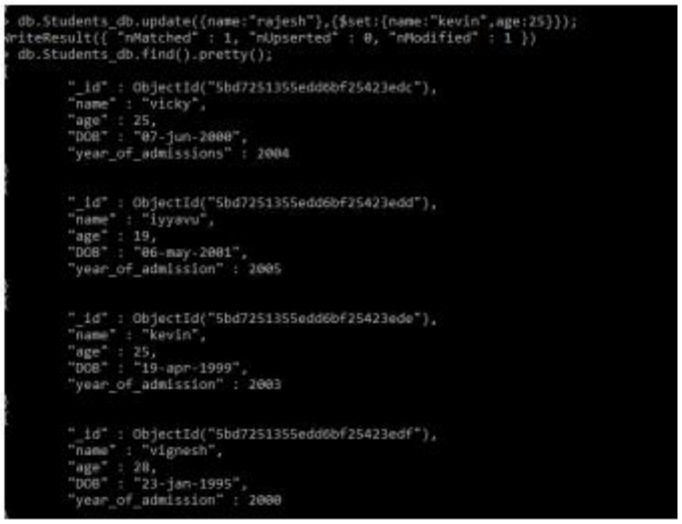
```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\CHOKKA>mongo
MongoDB shell version: 3.2.10
connecting to: test
> db.createCollection('student2')
{ "ok" : 1 }
> db.student2.insert({name:'yasika',age:'18',dob:'21-sep-00',adm_year:'10'})
WriteResult({ "nInserted" : 1 })
> db.student2.insert({name:'yasotha',age:'25',dob:'21-sep-00',adm_year:'5'})
WriteResult({ "nInserted" : 1 })
> db.student2.insert({name:'vinitha',age:'29',dob:'21-sep-00',adm_year:'1'})
WriteResult({ "nInserted" : 1 })
> db.student2.insert({name:'jothika',age:'29',dob:'21-sep-00',adm_year:'1'})
WriteResult({ "nInserted" : 1 })
> db.student2.find().sort({name:1}).pretty()
```

iii)Sort all documents in the student collection



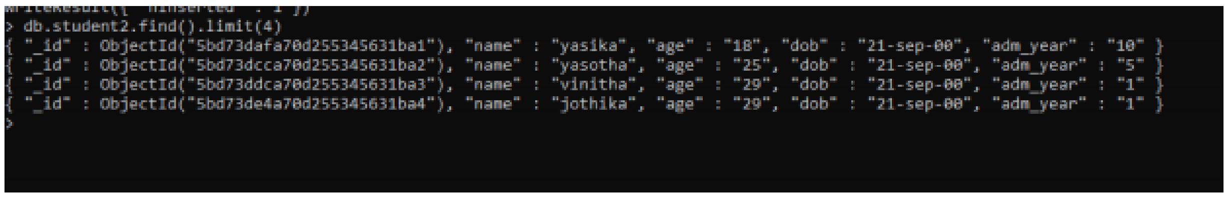
```
> db.student2.find().sort({name:1}).pretty()
{
        "_id" : ObjectId("5bd73de4a70d255345631ba4"),
        "name" : "jothika",
        "age" : "29",
        "dob" : "21-sep-00",
        "adm_year" : "1"
}
{
        "_id" : ObjectId("5bd73ddca70d255345631ba3"),
        "name" : "vinitha",
        "age" : "29",
        "dob" : "21-sep-00",
        "adm_year" : "1"
}
{
        "_id" : ObjectId("5bd73dafa70d255345631ba1"),
        "name" : "yasika",
        "age" : "18",
        "dob" : "21-sep-00",
        "adm_year" : "10"
}
{
        "_id" : ObjectId("5bd73dcca70d255345631ba2"),
        "name" : "yasotha",
        "age" : "25",
        "dob" : "21-sep-00",
        "adm_year" : "5"
}
>
```

iv)Update the document of student with name='Kevin' to age=25



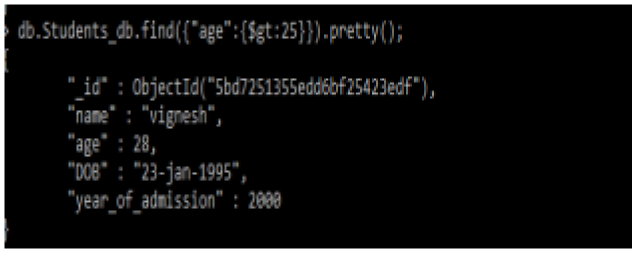
```
> db.Students_db.update({name:"rajesh"},{$set:{name:"kevin",age:25}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Students_db.find().pretty();
{
        "_id" : ObjectId("5bd7251355edd6bf25423edc"),
        "name" : "vicky",
        "age" : 25,
        "DOB" : "07-jun-2000",
        "year_of_admissions" : 2004
}
{
        "_id" : ObjectId("5bd7251355edd6bf25423edd"),
        "name" : "iyyavu",
        "age" : 19,
        "DOB" : "06-may-2001",
        "year_of_admission" : 2005
}
{
        "_id" : ObjectId("5bd7251355edd6bf25423ede"),
        "name" : "kevin",
        "age" : 25,
        "DOB" : "19-apr-1999",
        "year_of_admission" : 2003
}
{
        "_id" : ObjectId("5bd7251355edd6bf25423edf"),
        "name" : "vignesh",
        "age" : 28,
        "DOB" : "23-jan-1995",
        "year_of_admission" : 2000
```

v)Limit the result to 4 documents



```
> db.student2.find().limit(4)
{ "_id" : ObjectId("5bd73dafa70d255345631ba1"), "name" : "yasika", "age" : "18", "dob" : "21-sep-00", "adm_year" : "10" }
{ "_id" : ObjectId("5bd73dcca70d255345631ba2"), "name" : "yasotha", "age" : "25", "dob" : "21-sep-00", "adm_year" : "5" }
{ "_id" : ObjectId("5bd73ddca70d255345631ba3"), "name" : "vinitha", "age" : "29", "dob" : "21-sep-00", "adm_year" : "1" }
{ "_id" : ObjectId("5bd73de4a70d255345631ba4"), "name" : "jothika", "age" : "29", "dob" : "21-sep-00", "adm_year" : "1" }
>
```

vi)Query the document on the basis of age>25



```
> db.Students_db.find({"age":{$gt:25}}).pretty();
{
        "_id" : ObjectId("5bd7251355edd6bf25423edf"),
        "name" : "vignesh",
        "age" : 28,
        "DOB" : "23-jan-1995",
        "year_of_admission" : 2000
}
```

7)

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.jsserver.

iii) Perform thefollowing:

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 3times.

If the user enters username and password more than 3 times, display the

message “you are blocked”.

PROGRAM:

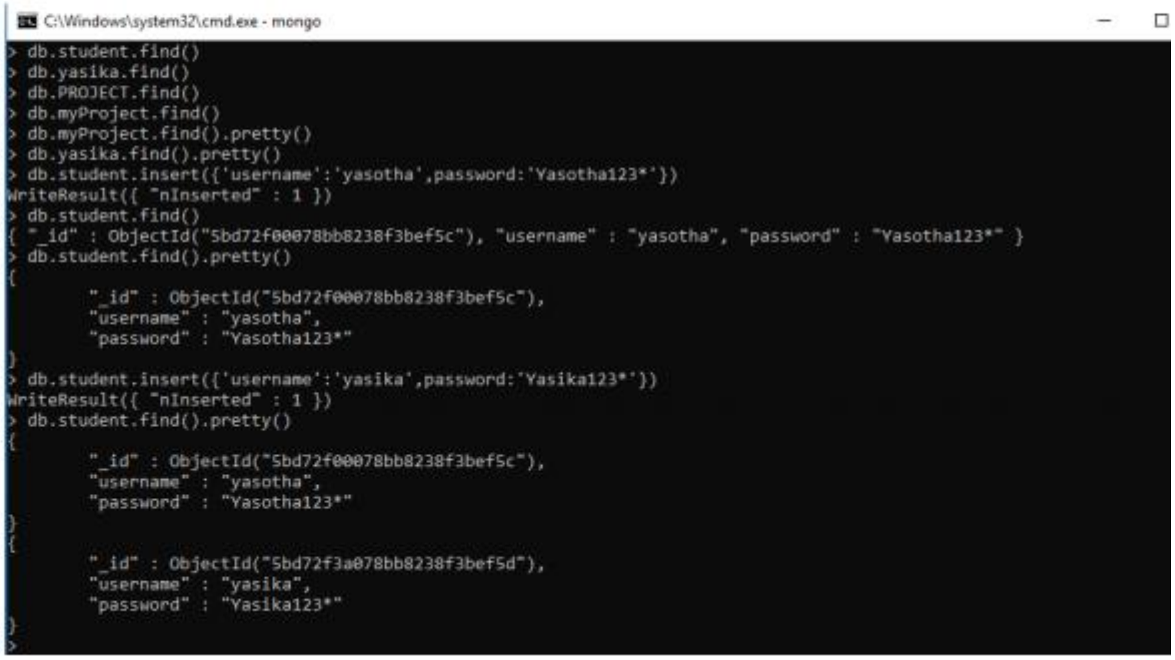
```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
// Create application/x-www-form-urlencoded parser
var urlencodedParser = bodyParser.urlencoded({ extended: false })
app.use(express.static('public'));
app.get('/login.html', function (req, res) {
```

```
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 detais 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)
})
```

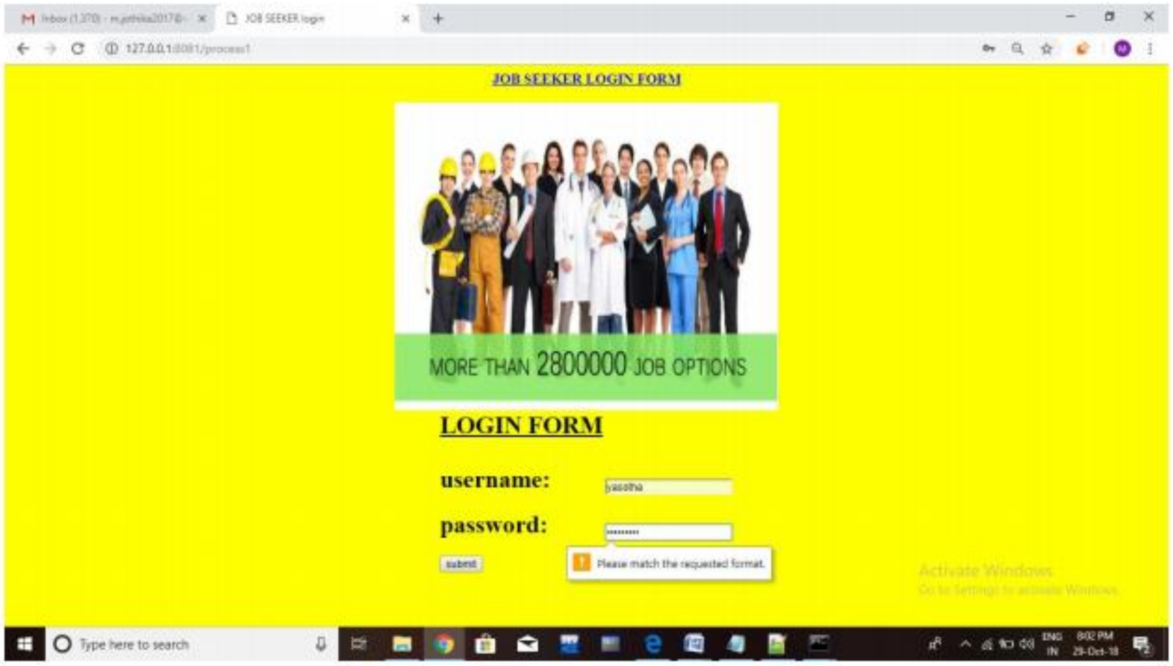
OUTPUT:



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*"
}
>
```

a)



JOB SEEKER LOGIN FORM

MORE THAN 2800000 JOB OPTIONS

LOGIN FORM

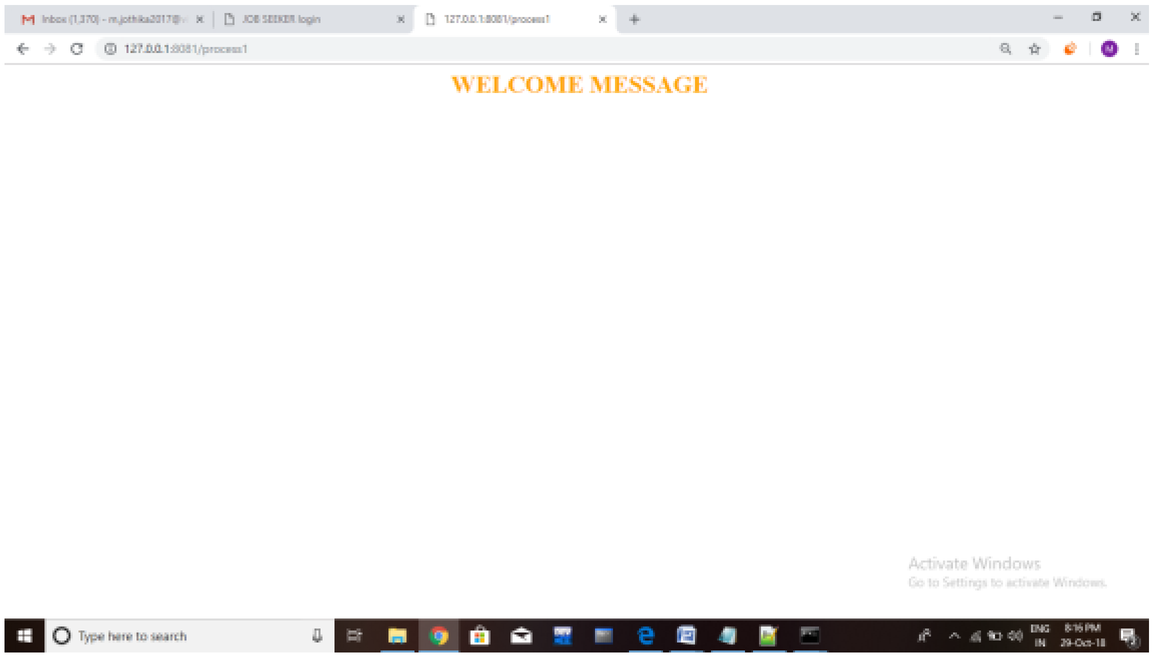
username: yasotha

password: ••••••••••

submit

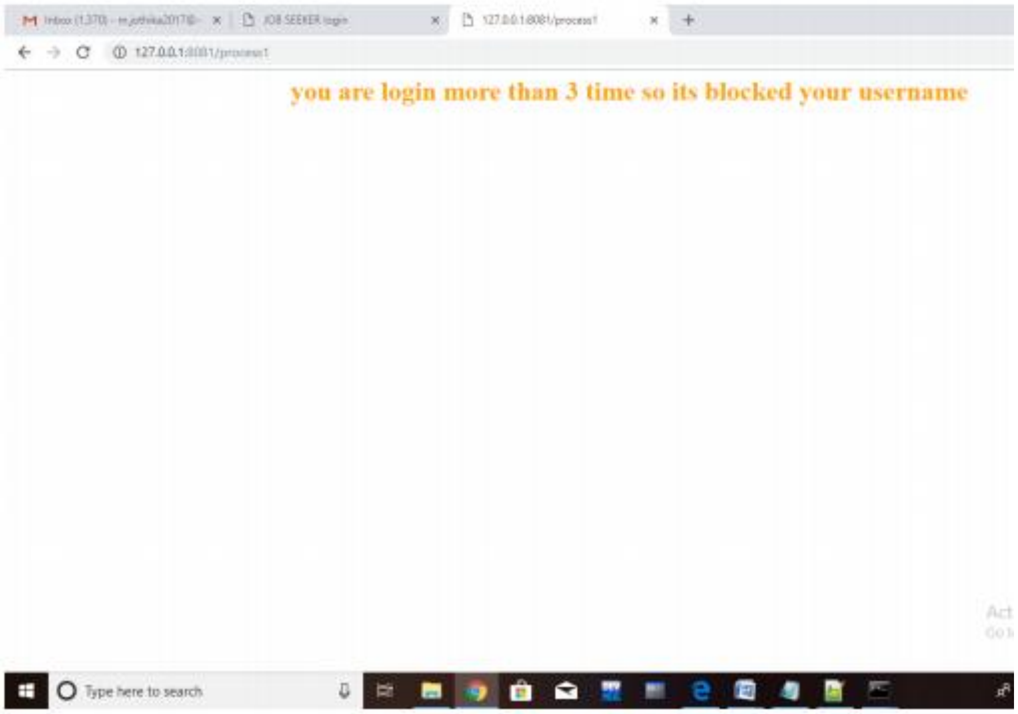
Please match the requested format.

b)



WELCOME MESSAGE

c)



you are login more than 3 time so its blocked your username