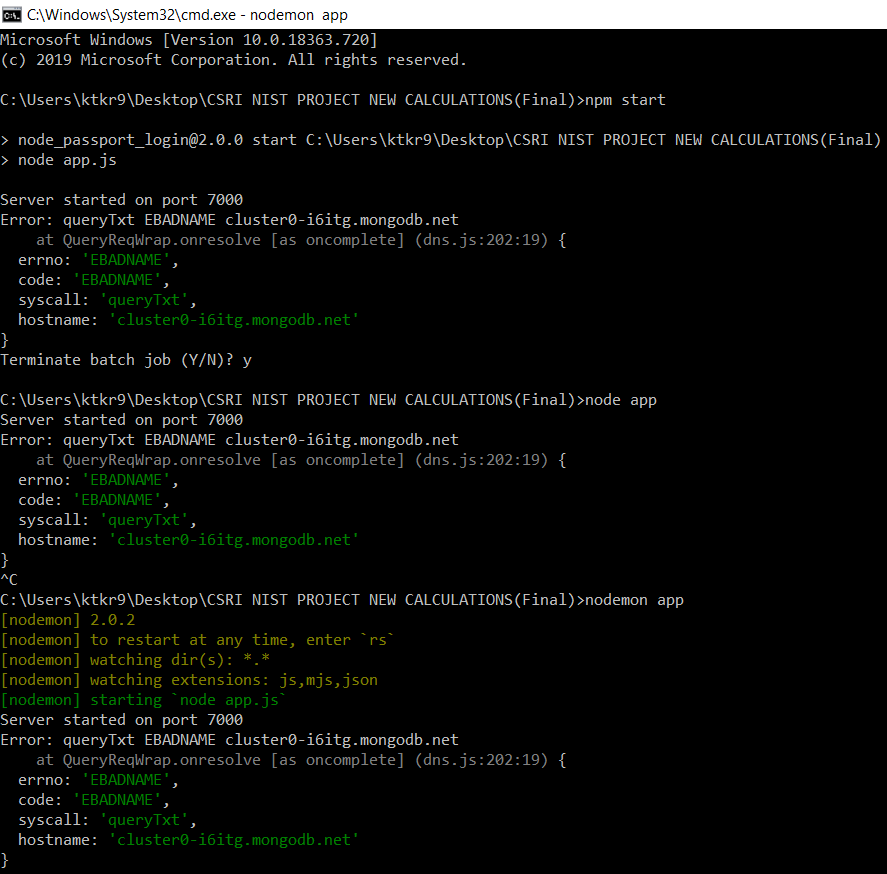
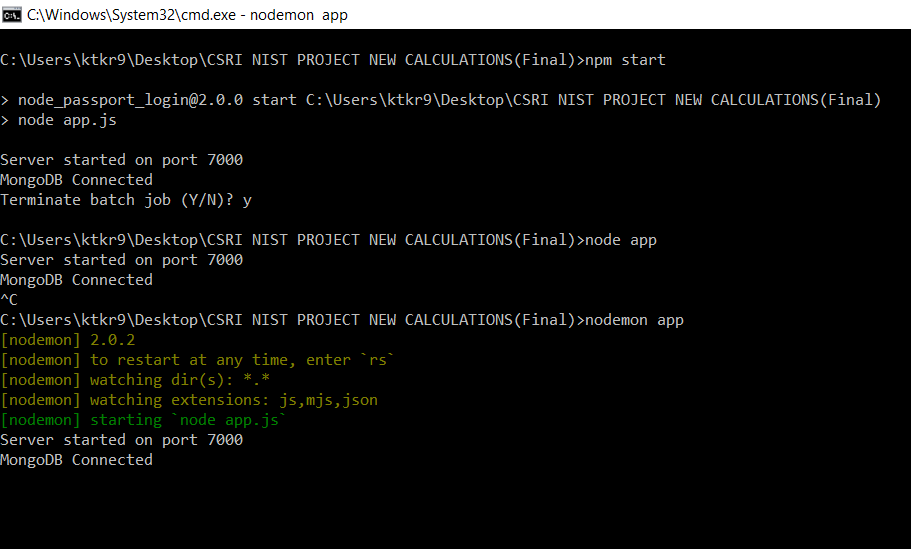
**PreRequisites:-**

You have to install Node.js to run the application.



We can run our web application using three different commands. It is by “npm start” or “node app” or “nodemon app”. Using one command is enough and the preferred command is “nodemon app”( by using this command you don’t have to re-start your web application if you make any changes in the backend part, it will automatically restart).

If you encounter the error highlighted in red portion. Try to connect your Wi-Fi to your mobile hotspot( this is the only way you can work on the web application). If you didn’t get any error means you are good to run the web application.

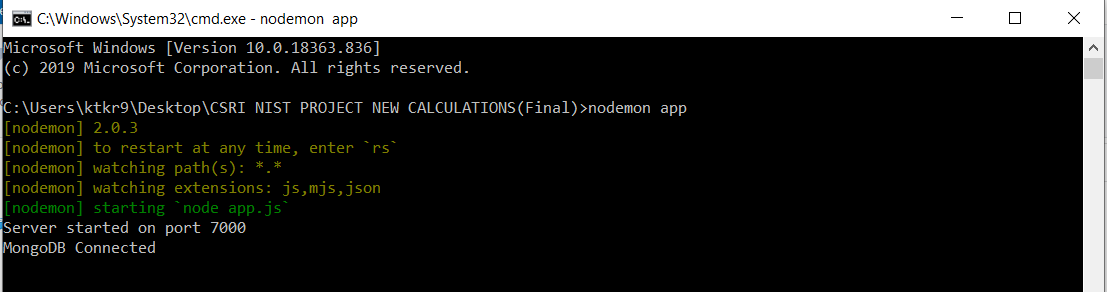


**Steps to run the application:-**

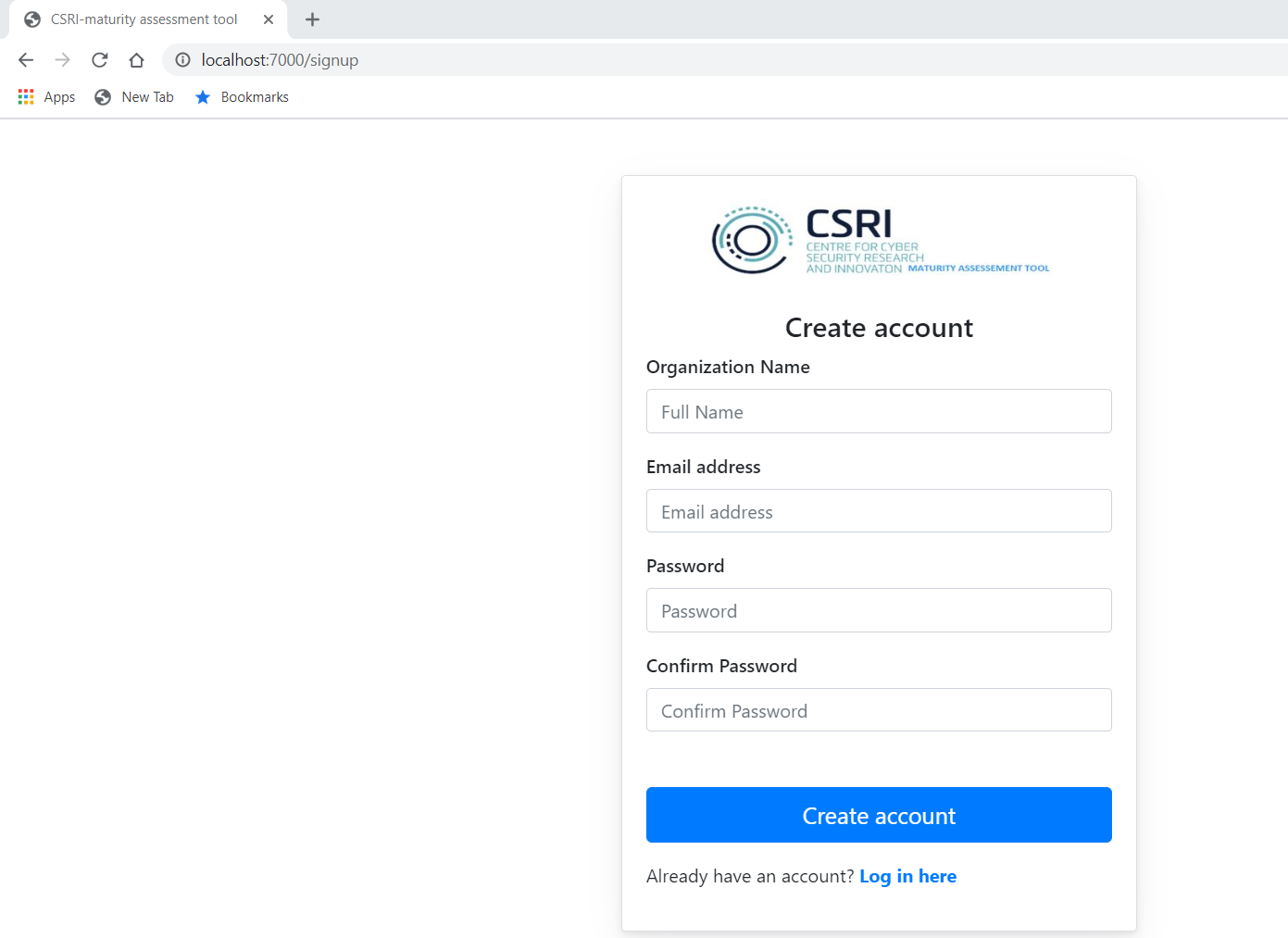
1. Run the app.js file in the command prompt by navigating to the application and start it as “**node app”**

It shows **Server is running on port 7000**

**MongoDB Connected**

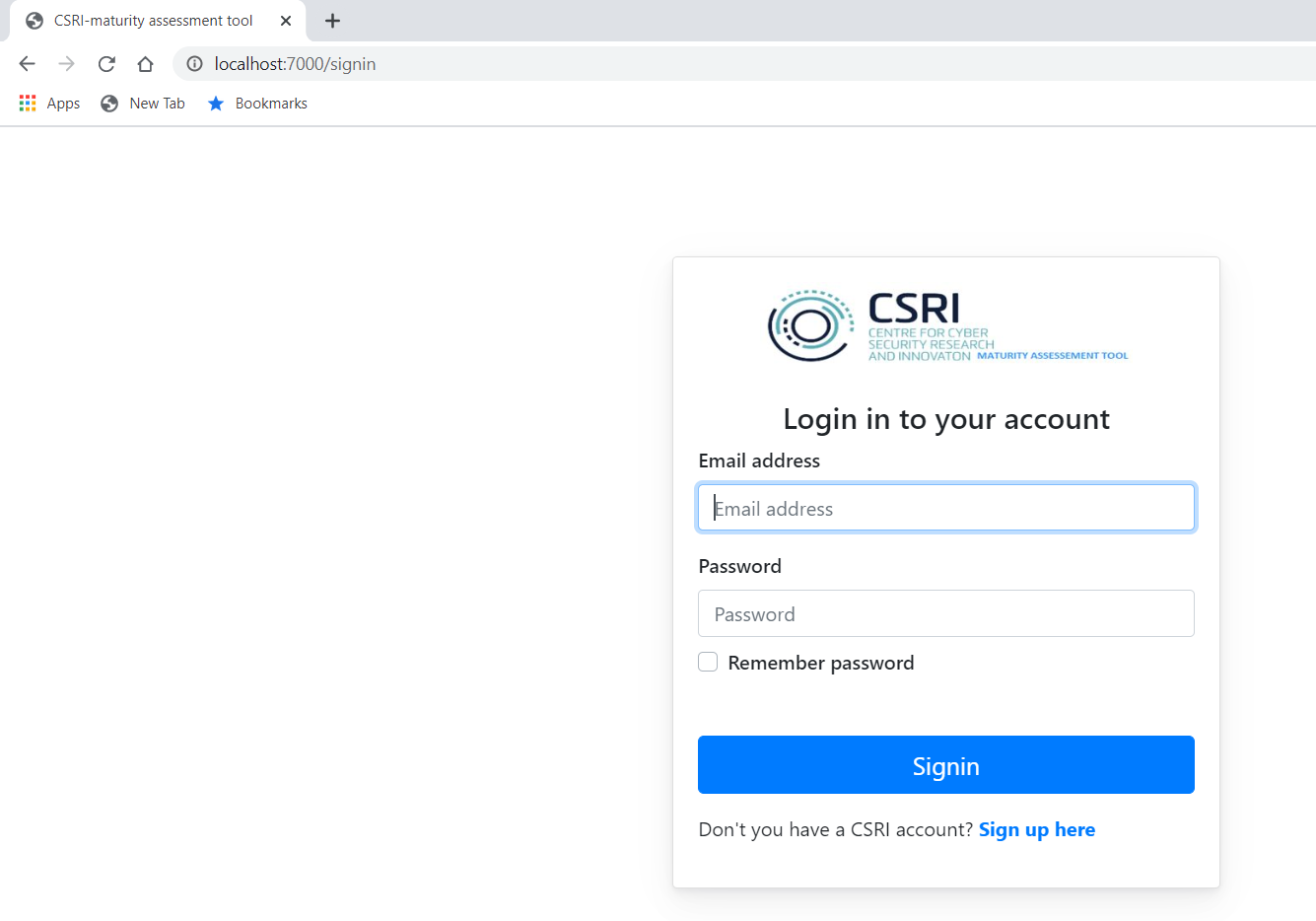


1. Run the application in the url of the browser( like chrome browser, fire fox, Internet Explorer etc) by typing as <http://localhost:7000/users/login>

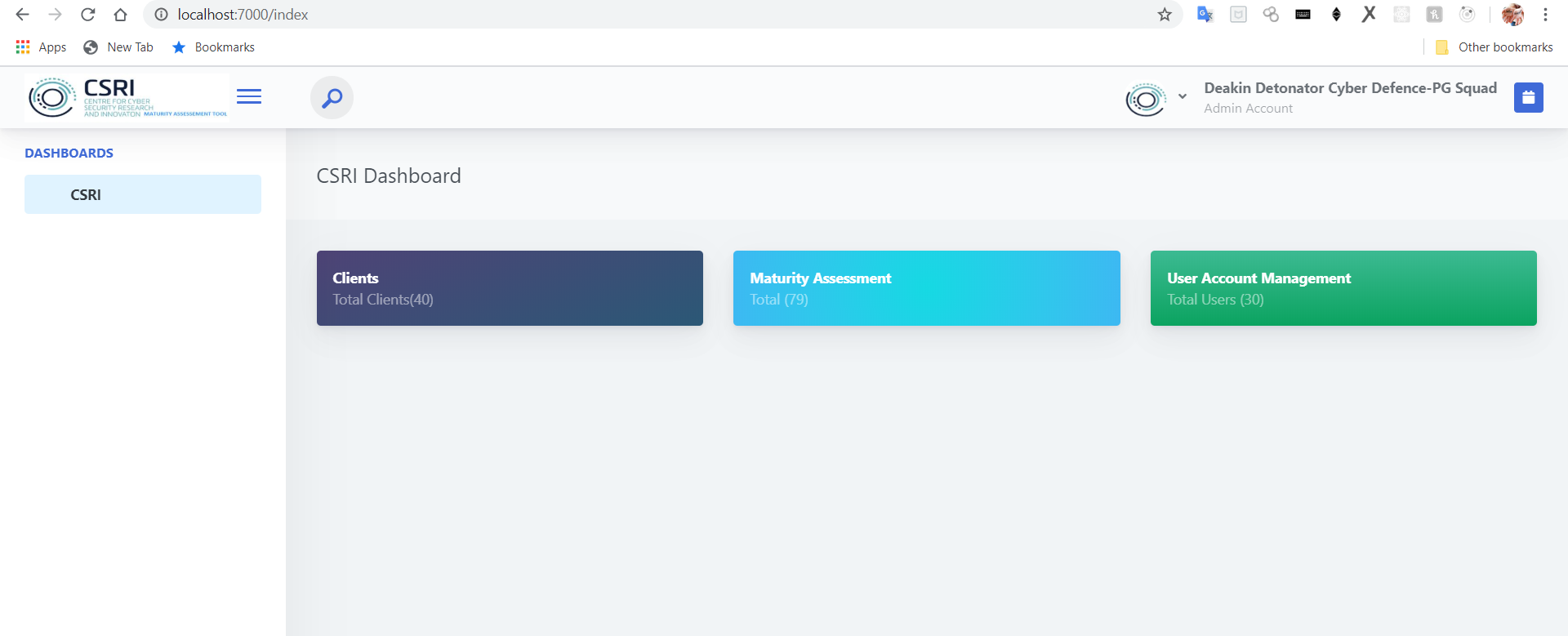


Provide email and password and click on signup. Then you will be navigated to signin page.

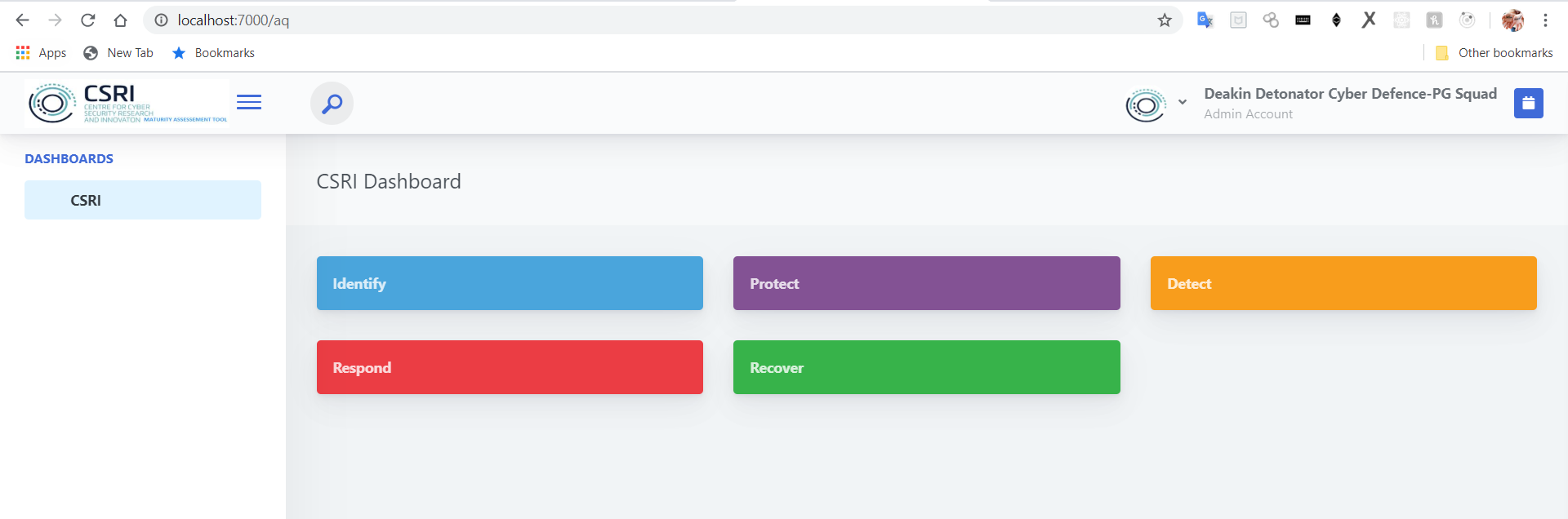
1. Provide the email and password you have used for signup.



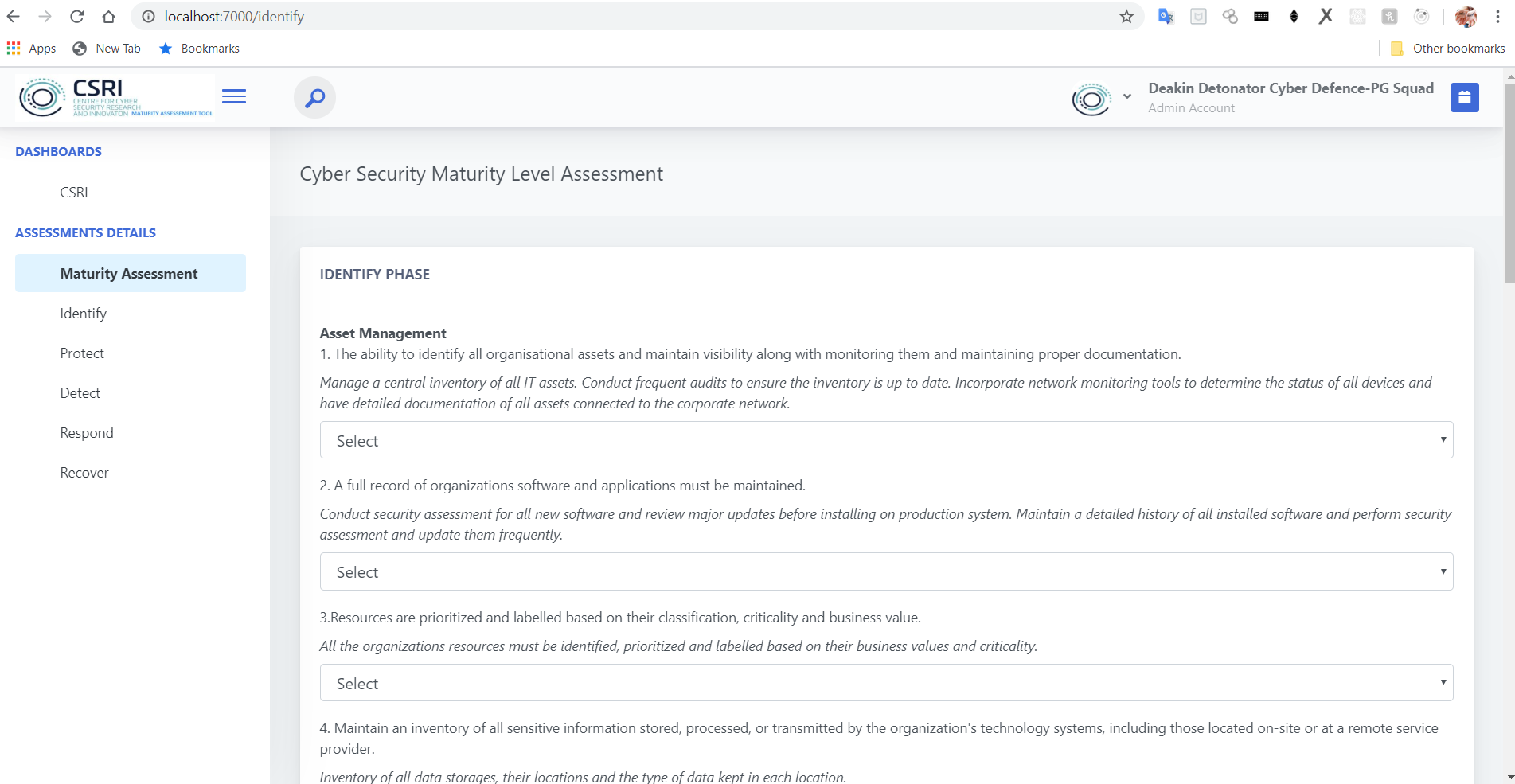
1. Then you will be taken to index page.



1. Click on maturity Assessment to navigate all phases for assessment.
2. Click on identify to start the assessment.



1. Then you will be taken identify phase assessment.

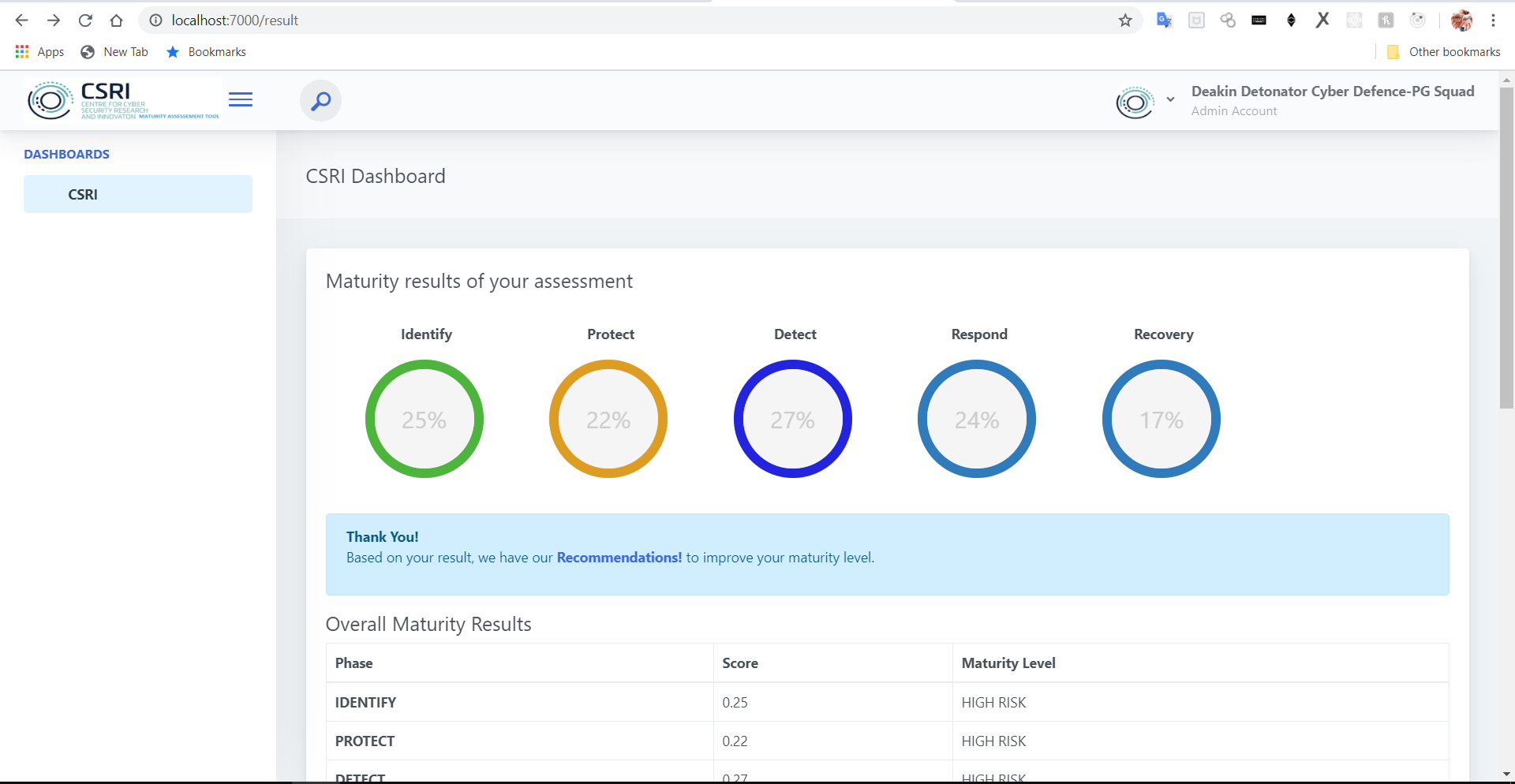


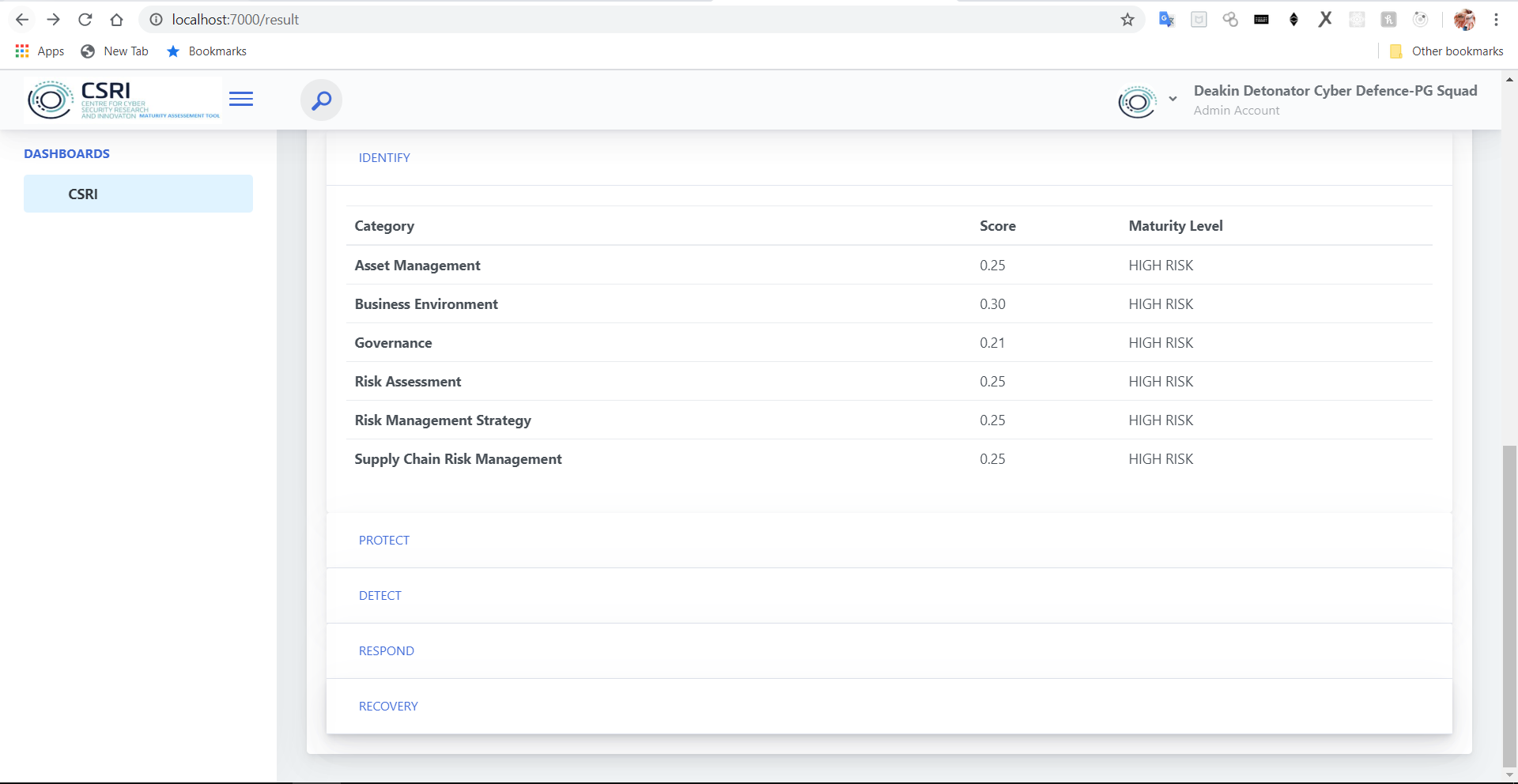
1. Select the option for each question and difficulty also(by default it highlights Low).

**NOTE:-**

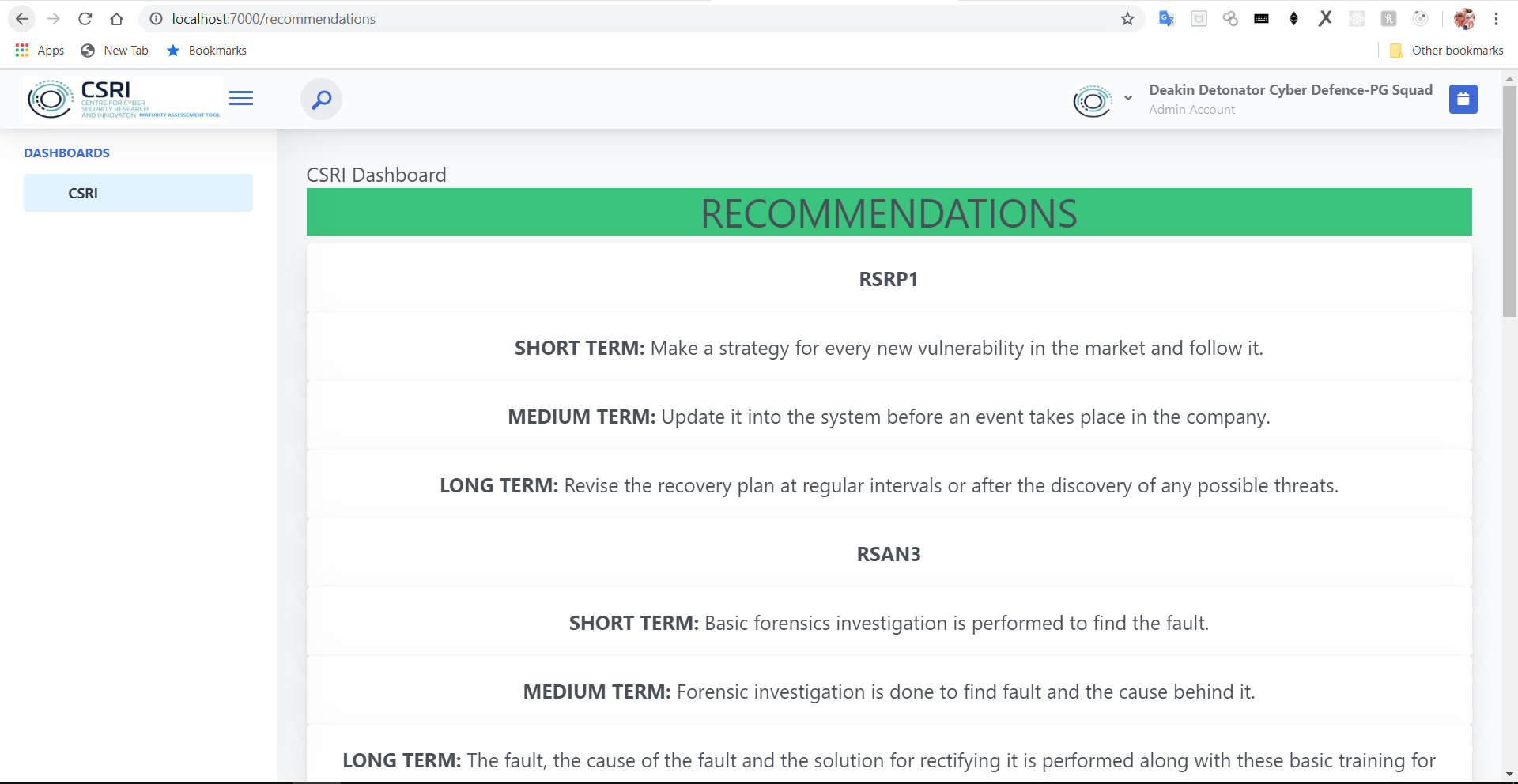
Difficulty appears for Not Implemented and Partially Implemented cases. It does not appear for Fully Implemented.

1. Click next to go to the next phase
2. Continue the process for all the phases
3. Finally you will see the **results** and **recommendations** based on your answers after answering all the questions in each phase.





1. Click on each phase to see the results in each subcategory.
2. Click on recommendations to go to next page and see recommendations



**NOTE:-**

As recommendations are stored in local mongodb you won’t be able to see results in your system.

1. Click on arrow pointing downwards(left of Deakin Detonator Cyber Defence-PG Squad) on the top navigation bar and click on sign out to sign out of the application.
2. Terminate the node application by pressing **ctrl+c** to stop the server in the command prompt.

**What have we used?**

In this application we have implemented the backend **(app.js)** and other supportive .js files **(index.js, user.js(in routes), keys.js, auth.js, passport.js(in config), User.js, identifyschema.js, protectschema.js, detectshema.js, respondschema.js, recoverschema.js and recommendations.js(in models))** .

We used **Node.js** as the server.

We used **express.js** as middleware.

We used **cloud mongodb atlas** as the database.

**Credentials for MongoDb atlas:**

email id: [csrimaturitycalculation@gmail.com](mailto:csrimaturitycalculation@gmail.com)

Password: CSRI@deakin

**How have we implemented this?**

For each **question** in each phase, we have **select** tag with **options** and **input** tags.

We used **name** and **id** attributes for every **select** and **input** tags. For every **option** and **input** tags we have taken **value** as attribute.

**Example:-**

<div class="form-group">

<label for="formGroupExampleInput">1. The ability to identify all organisational assets and maintain visibility to them</label>

<label for="formGroupExampleInput"><i>Manage a central inventory of all IT assets. Conduct frequent audits to ensure the inventory is up to date. Incorporate network monitoring tools to determine the status of all devices</i></label>

<select class="form-control" id="i1" name="i1">

<option value="-1"> Select </option>

<option value="0">Not implemented</option>

<option value="1">Partially implemented </option>

<option value="2">Fully Implemented</option>

</select>

<div class="selectOption">

<label>Difficulty Level: </label>

<input type="radio" value="1" checked name="id1" /> Low

<input type="radio" value="2" name="id1" />

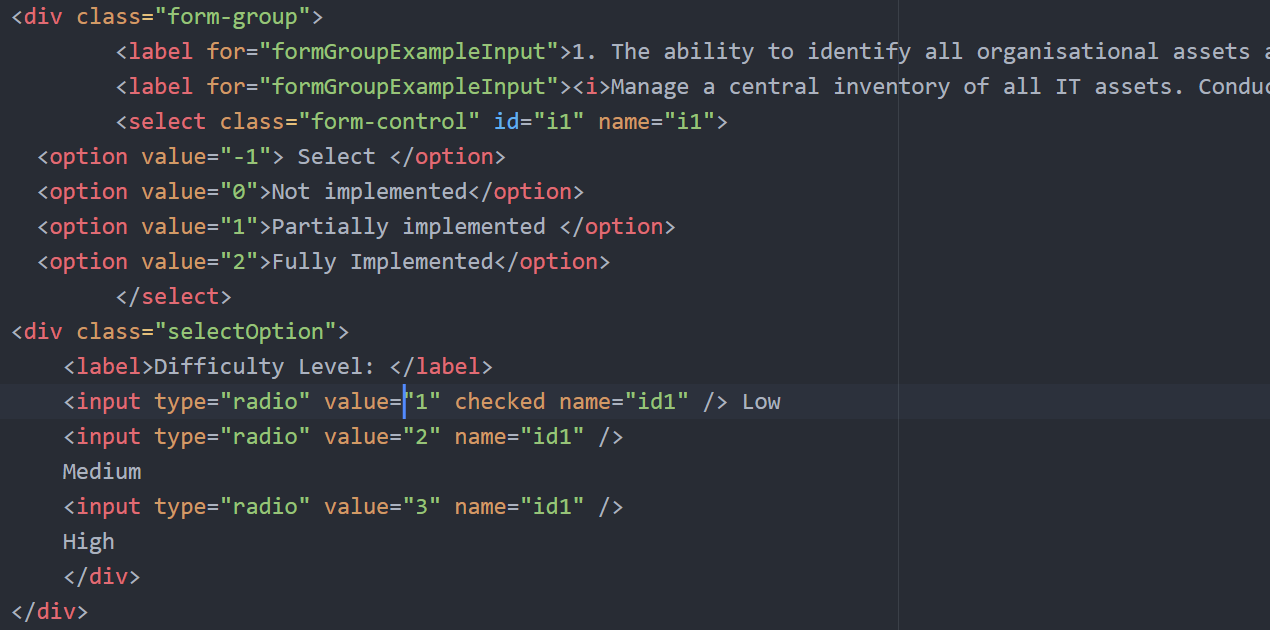
Medium

<input type="radio" value="3" name="id1" />

High

</div>

</div>



**How have we collected data?**

By using the id of each select and input tag and value of the option selected in the front end we are collecting the data.

This process of collection of data occurs when the user presses the button(i.e., Next for identify, protect, detect, and respond and Submit for recovery phase) at the bottom of the page. We provided Back to go to previous phase also.

We have implemented a script at the bottom of the html page of each phase( you can see the script by opening the html page of each phase using the notepad or notepad++ or any other editor).

In the script we implemented a function. In the function we used post as the method to take the data to backend(app.js). Post is a secure way to transfer the data from front end to backend.

**Example:-**

<script>

$("#Next").click(function() {

var validationCheck = false;

var selectBoxes = document.querySelectorAll("select");

var objResult = {};

for(var i =0; i<selectBoxes.length; i++){

var element = selectBoxes[i];

if (element.value == -1) {

element.className = "form-control selectorError";

validationCheck = true;

}

else {

element.className = "form-control";

objResult["i"+(i+1)] = element.value;

}

}

if(validationCheck == true){

alert("please update the required fields");

}

else {

alert("Proceed to the next page");

var formData = {

'i1': $('#i1').val(),

'i2': $('#i2').val(),

'i3': $('#i3').val(),

'i4': $('#i4').val(),

'i5': $('#i5').val(),

'i6': $('#i6').val(),

'i7': $('#i7').val(),

'i8': $('#i8').val(),

'i9': $('#i9').val(),

'i10': $('#i10').val(),

'i11': $('#i11').val(),

'i12': $('#i12').val(),

'i13': $('#i13').val(),

'i14': $('#i14').val(),

'i15': $('#i15').val(),

'i16': $('#i16').val(),

'i17': $('#i17').val(),

'i18': $('#i18').val(),

'i19': $('#i19').val(),

'id1':$("input[name=id1]:checked").val(),

'id2':$("input[name=id2]:checked").val(),

'id3':$("input[name=id3]:checked").val(),

'id4':$("input[name=id4]:checked").val(),

'id5':$("input[name=id5]:checked").val(),

'id6':$("input[name=id6]:checked").val(),

'id7':$("input[name=id7]:checked").val(),

'id8':$("input[name=id8]:checked").val(),

'id9':$("input[name=id9]:checked").val(),

'id10':$("input[name=id10]:checked").val(),

'id11':$("input[name=id11]:checked").val(),

'id12':$("input[name=id12]:checked").val(),

'id13':$("input[name=id13]:checked").val(),

'id14':$("input[name=id14]:checked").val(),

'id15':$("input[name=id15]:checked").val(),

'id16':$("input[name=id16]:checked").val(),

'id17':$("input[name=id17]:checked").val(),

'id18':$("input[name=id18]:checked").val(),

'id19':$("input[name=id19]:checked").val(),

};

console.log(formData);

$.post("http://localhost:7000/api/identify", formData, function(response) {

console.log("hai")

if (response.error == undefined) {

console.log(response.response);

window.location.href = "protect";

} else {

alert(response.error)

}

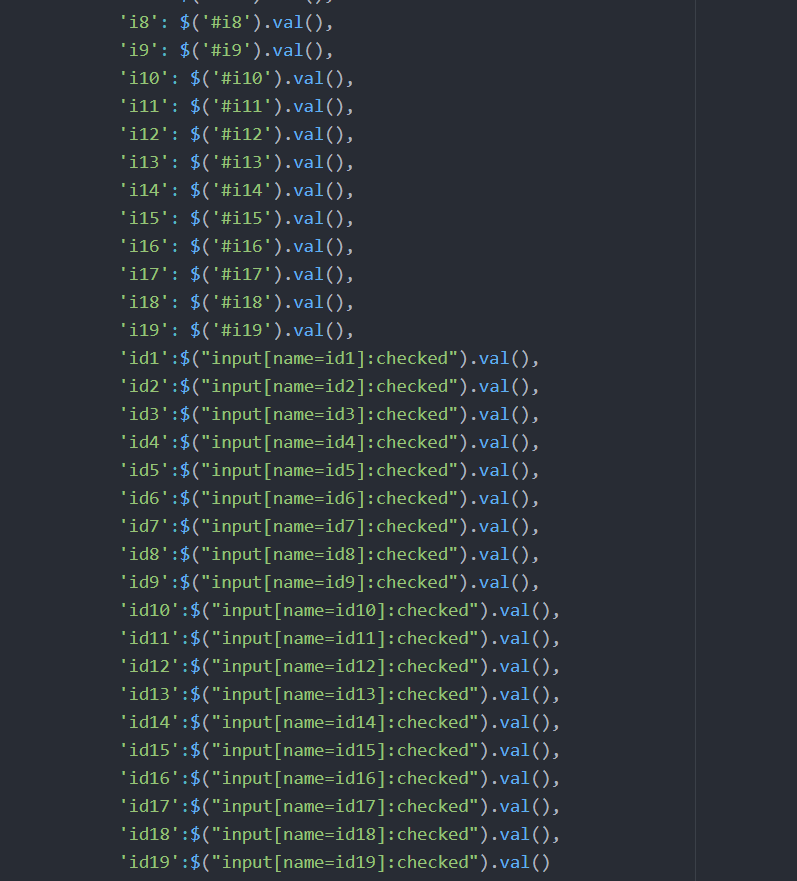
});

document.getElementById("Next").setAttribute("href","protect");

}

});</script>







We collect the data using the post method implemented in the app.js file(you can see the code in the app.js file using notepad or notepad++ or any other editor).

**Example:-**

app.post('/api/identify', function(req, res) {

console.log("IDENTIFY PHASE");

var i1=parseInt(req.body.i1);

console.log(i1);

var i2=parseInt(req.body.i2);

console.log(i2);

var i3=parseInt(req.body.i3);

console.log(i3);

var i4=parseInt(req.body.i4);

console.log(i4);

var i5=parseInt(req.body.i5);

console.log(i5);

var i6=parseInt(req.body.i6);

console.log(i6);

var i7=parseInt(req.body.i7);

console.log(i7);

var i8=parseInt(req.body.i8);

console.log(i8);

var i9=parseInt(req.body.i9);

console.log(i9);

var i10=parseInt(req.body.i10);

console.log(i10);

var i11=parseInt(req.body.i11);

console.log(i11);

var i12=parseInt(req.body.i12);

console.log(i12);

var i13=parseInt(req.body.i13);

console.log(i13);

var i14=parseInt(req.body.i14);

console.log(i14);

var i15=parseInt(req.body.i15);

console.log(i15);

var i16=parseInt(req.body.i16);

console.log(i16);

var i17=parseInt(req.body.i17);

console.log(i17);

var i18=parseInt(req.body.i18);

console.log(i18);

var i19=parseInt(req.body.i19);

console.log(i19);

var id1=parseInt(req.body.id1);

console.log(id1);

var id2=parseInt(req.body.id2);

console.log(id2);

var id3=parseInt(req.body.id3);

console.log(id3);

var id4=parseInt(req.body.id4);

console.log(id4);

var id5=parseInt(req.body.id5);

console.log(id5);

var id6=parseInt(req.body.id6);

console.log(id6);

var id7=parseInt(req.body.id7);

console.log(id7);

var id8=parseInt(req.body.id8);

console.log(id8);

var id9=parseInt(req.body.id9);

console.log(id9);

var id10=parseInt(req.body.id10);

console.log(id10);

var id11=parseInt(req.body.id11);

console.log(id11);

var id12=parseInt(req.body.id12);

console.log(id12);

var id13=parseInt(req.body.id13);

console.log(id13);

var id14=parseInt(req.body.id14);

console.log(id14);

var id15=parseInt(req.body.id15);

console.log(id15);

var id16=parseInt(req.body.id16);

console.log(id16);

var id17=parseInt(req.body.id17);

console.log(id17);

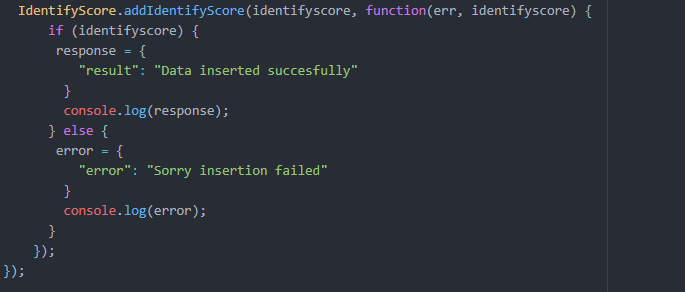
var id18=parseInt(req.body.id18);

console.log(id18);

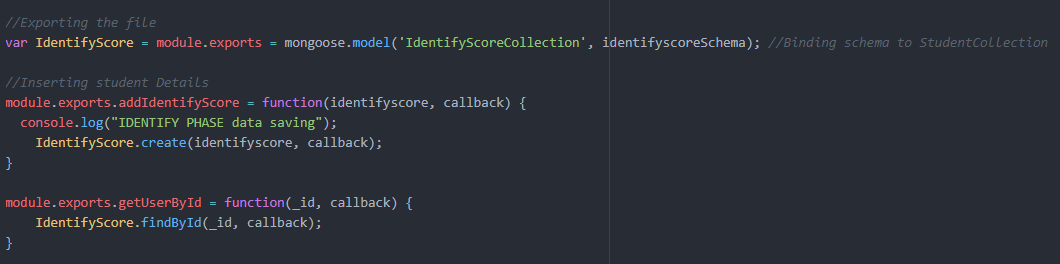
var id19=parseInt(req.body.id19);

console.log(id19);





Then we have calculated the total weights and clientscore for each phase. Then we have stored the data in database and sent the data to the front end through an object by collecting the required data which is calculated for all the phases.



Finally we have displayed the results at the result page and recommendations at recommendations page in the frontend.

**NOTE:-**

We can observe all the calculations, data, results and recommendations in the console for the back end parts.