SRN | | | | | | | | | | | | | |

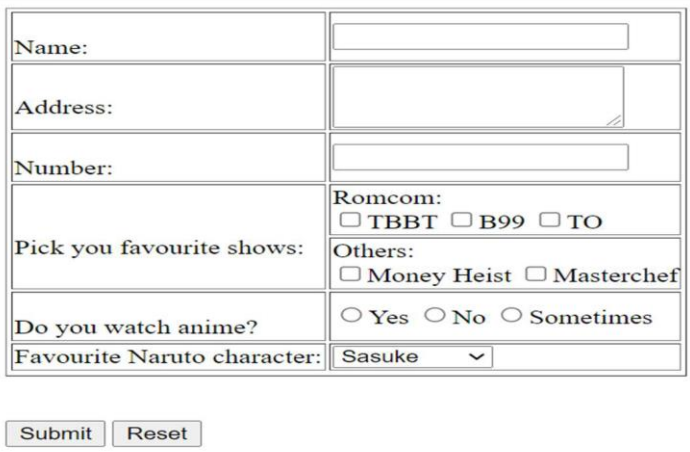
**PES University, Bangalore**
(Established under Karnataka Act No. 16 of 2013)

UE20CS204

**August –December 2021: END SEMESTER ASSESSMENT (ESA) B.Tech. III SEMESTER**

**UE20CS204: WEB TECHNOLOGIES**

Time: 3 Hrs. Answer All Questions Max Marks: 100

| | | | |
|---|---|---|---|
| 1. | a) | What is HTTP? Explain the structure of HTTP request message.<br>**Scheme and Solution**<br>**//HTTP:1 Mark**<br>HTTP stands for Hyper Text Transfer Protocol. This is a basis for data communication in the internet. The data communication starts with a request sent from a client and ends with the response received from a web server.<br>**//Structure of HTTP and explanation 3M**<br>• HTTP request is a *request line*, followed by zero or more *request headers*<br>• Request line: <method> <uri> <version><br>  • <version> is HTTP version of request (HTTP/1.0 or HTTP/1.1)<br>  • <uri> is typically URL for proxies, URL suffix for servers.<br>  • <method> is either GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE.<br>• Request Header<br>• Blank line (CRLF)<br>Message Body | 4M |
| | b) | Write a HTML code that outputs the following:<br>  <br><br>NOTE: The 'Number' field is a mandatory field, and takes in only a 10-digit number. Also, Submit and Reset buttons do their conventional tasks.<br>**Scheme and Solution**<br><html><br><head><br><title> My Netflix list </title><br></head> | 10M |

```
 <body>
//form 1 mark with attributes
 <form method='get' action='save.php' target='_blank'>
//Table 2 mark with proper rows and columns
 <table border="black">
 <tr>
 <td><br/>Name:</td>
//Name input field 0.5M
 <td><input type='text' name='uname'/></td>
 </tr>
 <tr>
 <td><br/>Address:</td>
//textarea 0.5M
 <td><textarea name='uadd' rows='3' cols='20'></textarea></td>
 </tr>
 <tr>
//Number input field 1 Mark with the required pattern
 <td><br/>Number:</td>
 <td><input type="tel" required pattern="[0-9]{10}" name='tele'></td>
 </tr>
 <tr>
//rowspan 1Mark and check boxes 1mark
 <td rowspan=2><br/>Pick you favourite shows:</td>
 <td>Romcom:<br/>
 <input type='checkbox' name='rom' value='TBBT'>TBBT
 <input type='checkbox' name='rom' value='B99'>B99
 <input type='checkbox' name='rom' value='TO'>TO</td>
 </tr>
 <tr>
 <td>Others:<br/>
 <input type='checkbox' name='oth' value='Money Heist'>Money Heist
 <input type='checkbox' name='oth' value='Masterchef'>Masterchef</td>
 </tr>
 <tr>
 <td><br/>Do you watch anime?</td>
//radio buttons 1M
 <td><input type='radio' name='anime' value='Yes'>Yes
 <input type='radio' name='anime' value='No'>No
 <input type='radio' name='anime' value='Sometimes'>Sometimes</td>
 </tr>
 <tr>
 <td>Favourite Naruto character: </td>
 <td>
//Menu Item 1M
 <select name='naruto'>
 <option value='Sasuke'>Sasuke
 <option value='Mighty Guy'>Mighty Guy
 <option value='Kakashi'>Kakashi
 <option value='Naruto'>Naruto
 <option value='Itachi'>Itachi
 </select>
 </td>
 <tr/>
 </table>
 <br/><br/>
//Buttons 1Mark
```

| | | | |
|---|---|---|---|
| | | <input type='submit' value='Submit'/><br><input type='reset' value='Reset'/><br></form><br></body><br></html> | |
| | c) | What is the output of the following function? Give an explanation regarding the same.<br>var x = 100;<br>function num() {<br>if (false) {<br>var x = 200;<br>}<br>console.log(x);<br>}<br>num();<br>**Scheme and Solution**<br>**Output: //1M**<br>undefined<br>**//explanation 2M**<br>In this example, we declared x to be 100 globally. Depending on an if statement, x could change to 200, but since the condition was false it should not have affected the value of x. Instead, x was hoisted to the top of the num() function, and the value became undefined. | 3M |
| | d) | Write a code to print "hi" at repeated intervals, where the length of the interval is the square of the previous interval.<br>**Scheme and Solution**<br>**//0.5 Marks**<br>let x = 100<br>**//2M for function**<br>function log()<br>{<br>console.log("hi")<br>if(true){<br>x = x*x;<br>t = setTimeout(log,x)<br>}<br>}<br>**//0.5 M**<br>var t = setTimeout(log,x); | 3M |
| | | | |
| 2. | a) | Consider the following HTML and JavaScript program:<br><html><br><head><br></head><br><body><br> <br><button id="button">Click me!</button><br> <br></body> | 4M |

```
<script>
function printWindow(event) {
console.log('Window says hello!');
}
function printTarget(event) {
console.log('Target says hello!');
}
function printContainer(event) {
console.log('Container says hello!');
}
var capture = true;
var button = document.getElementById('button');
var container = document.getElementById('container');
window.addEventListener('click', printWindow);
button.addEventListener('click', printTarget, capture);
container.addEventListener('click', printContainer,capture);
</script>
</html>
```

i. The page loads and the user clicks on the window object. What's the console output?

ii. The user clicks on the button that says "Click me!" What's the console output?

**Scheme and Solution**

**//1Mark**

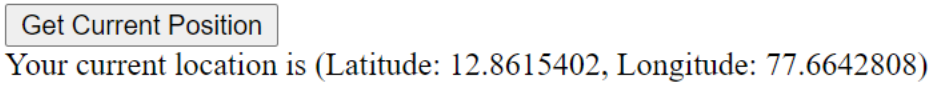
i. Window says hello!

**//3 Marks**

ii. Container says hello!
Target says hello!
Window says hello!

b) What is Geo Location? Explain the two methods of navigator.geolocation object?

Write a program to display your current location on the browser when the button "Get Current Position" is clicked as shown below.




**Scheme and Solution** — 1M+2M+4M

**//1Mark Explanation**

The HTML5 geo location feature lets you find out the geographic coordinates (latitude and longitude numbers) of the current location of your website's visitor.

**//1mark each with one line explanation**

getCurrentPosition() and watchPosition()

**// function getPos, showPosition, error 1Mark each, body 1Mark**

```
<head>
```

```
<script type="text/javascript">
        <!--
                var x= document.getElementById("location");
                function getPos(){
                        navigator.geolocation.watchPosition(showPosition,
error);
                }
function showPosition(position){
 var x = "Your current location is (" + "Latitude: " + position.coords.latitude +
", " + "Longitude: " +    position.coords.longitude + ")";
            document.getElementById("location").innerHTML = x;
    }
                        function error(error){
                                e = error;
                        }
                -->
</script>
</head>
<body>
        <button onclick="getPos()">Get Current Position</button>
        <div id="location"></div>
</body>
```

c) Write jQuery code to perform the following: 5M

1. For the last paragraph within the div with id “colortext”, set the color to green.
2. On moving the mouse over an h1 element, the text font size should be increased by 5 times
3. When the first li element with class name “liclass” on the page is clicked, it fades out in 2 seconds and then fades in 3 seconds

**Scheme and Solution**

```
//1Mark
1. $(“div#colortext p:last”).css(“color”,”green”)
//2 Marks
2. $(“h1”).mouseover(function(){
$(this).animate({
fontSize: 5em
}
}
// 2 Marks
3. $(“li.liclass”).click(function(event){
$(this).fadeOut(2000,function(){
$(this).fadeIn(5000, function(){});
});
});
```

d) What is AJAX? Explain in detail any 3 XHR object properties. 4M

**Scheme and Solution**

**//1Mark Explanation**

AJAX stands for Asynchronous JavaScript And XML uses XMLHttpRequest object to communicate with servers. It can send and receive information in various formats, including JSON, XML, HTML, and text files.

**//3marks any 3 properties of XHR object**

Onreadystatechange: Specifies an event-handling function to be called whenever the readyState property of an object changes.

readyState:An integer property that reports on the status of a request. It can have any of these values: 0 = Uninitialized, 1 = Loading, 2 = Loaded, 3 = Interactive, and 4 = Completed.

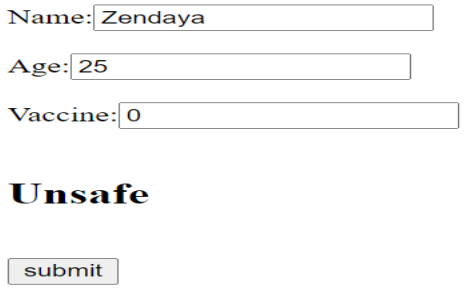
responseText :The data returned by the server in text format.

responseXML :The data returned by the server in XML format.

Status: The HTTP status code returned by the server.

statusText: The HTTP status text returned by the server.

| | | | |
|---|---|---|---|
| 3. | a) | Create a form (as shown below) using uncontrolled components (references) to take 3 inputs: name, age, and number of vaccine doses taken. Also have a <h2> tag inside the form which displays “Unsafe”, “Almost safe”, “Safe” and “Invalid” for the number of doses being 0, 1, 2 and ‘any other number’ respectively, on submission of the form. | 10M |




**Scheme and Solution**

```
<html>
 <head>
 <title>Question 3a</title>
 <script crossdomain src="https://unpkg.com/react@16/umd/react.development.js"> </script>
 <script crossdomain src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"> </script>
 <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js">
</script>
 </head>
 <body>
 <div id="container"></div>
 <script type="text/babel">
 var txt,ev;
```

**//class component with bind 3M**

```
 class NameAgeVacc extends React.Component{
 constructor(props){
```

```
 super(props);
 this.statusoutput = null;
 this.setName=(el)=>{this.nameinput=el}
 this.setAge=(el)=>{this.ageinput=el}
 this.setVacc=(el)=>{this.vaccinput=el}
 this.setStatRef=(el)=>{this.statusoutput=el}
 this.handleSubmit=this.handleSubmit.bind(this);
 }
//handleSubmit function call 2M
 handleSubmit=function(event){
        if(this.statusoutput){
 if (this.vaccinput.value == 0)
 this.statusoutput.innerHTML="Unsafe"
 else if (this.vaccinput.value == 1)
 this.statusoutput.innerHTML="Almost Safe"
 else if (this.vaccinput.value == 2)
 this.statusoutput.innerHTML="Safe"
 else
 this.statusoutput.innerHTML="Invalid"
 event.preventDefault();
 }
 }
 render(){
 return(
 <div>
// form elements with onsubmit function call 4M
 <form onSubmit={this.handleSubmit}>
 <label>
 Name:
 </label>
 <input name="name" defaultValue='Zendaya' type="text"
ref={this.setName}/>
 <br/><br/>
 <label>
 Age:
 </label>
 <input name="age" defaultValue='25' type="text" ref={this.setAge}/>
 <br/><br/>
 <label>
 Vaccine:
 </label>
 <input name="vaccine" defaultValue='0' type="text" pattern="[0-9]"
ref={this.setVacc}/>
 <br/><br/>
 <h2 ref={this.setStatRef}/>
```

```
<br/>
<input type="submit" value="submit"/>
</form>
</div>
)
}
}
```

**//render the component 1M**

```
ReactDOM.render(
<NameAgeVacc/>,
document.querySelector("#container")
)
</script>
</body>
</html>
```

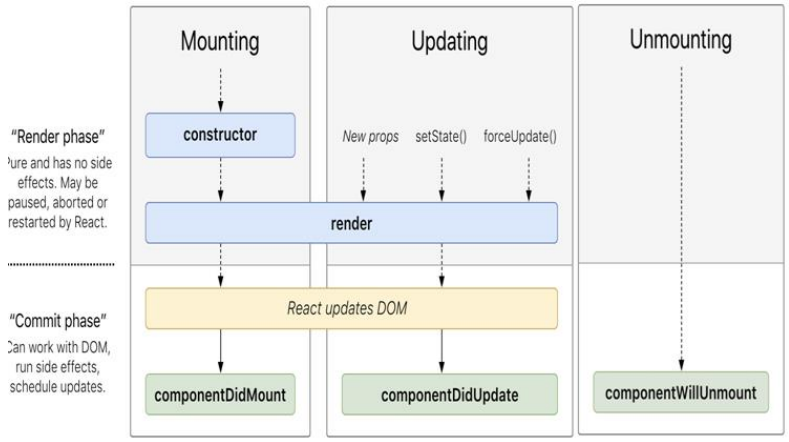
b) Create a class component which will render a H1 and paragraph element apply different inline style for H1 and P element created using java script objects (key value pair). — 4M

**Scheme and Solution**

**//2 Marks for creating component and object for style**

**//1 mark each for creating elements and rendering the component**

```
<body>
<div id="root"></div>
<script type="text/babel">
class StyleInline extends React.Component
{
render(){
var StyleH1={
                    backgroundColor:"#FF55EE",
                    fontFamily:"monospace",
                    textAlign:"center"
                    };
var StyleP={
                    color:"#FFEE00",
                    fontFamily:"sans-serif",
                    textAlign:"left"
          };
return (
<div>
      <h1 style={StyleH1}>Welcome to PES World!</h1>
                    <p style={StyleP}> Batch of 2020-2021</p>
                    </div>
                    )}
                    }
ReactDOM.render(  <StyleInline />,    document.getElementById("root")
```

| | | | |
|---|---|---|---|
| | | );<br></script><br></body> | |
| | c) | With a neat diagram explain component life cycle.<br>**Scheme and Solution**<br>**//Diagram :2 Marks**<br>  <br><br>**//Explanation 4 Marks:**<br>Mounting - Birth of the Component<br>Updating- Growing of component<br>Unmounting- End of the component<br>In the entire lifecycle of a React component, the following methods are used to accomplish the functions:<br>•componentWillMount() – On client and server side this function gets executed just before the rendering<br>•componentDidMount() – After first render it gets executed on the client side<br>•componentWillReceiveProps() – This function is invoked when the props are received from the parent class and another render is not being called.<br>•shouldComponentUpdate() – This Boolean function returns true or false as per situation like if the component needs to be updated then true is returned else false is returned<br>•thecomponentWillUpdate() – It is called when rendering is not being called<br>•componentDidUpdate() – It is called just after when render function is called<br>•componentwillUnmount() – When a component gets un-mounted from DOM then this function is called. | 6M |
| | | | |
| 4. | a) | What are buffers in Node.js? With help of syntax explain any 2 buffer operations.<br>**Scheme and Solution**<br>**//1 Mark**<br>Node provides Buffer class which provides instances to store raw data similar to an array of integers.<br>**//Any 2 with explanation: 2 marks each**<br>Creating buffer<br>Syntax: Buffer.alloc(size, fill, encoding)<br>Writing to Buffer<br>Syntax: buf.write(string[, offset][, length][, encoding]) | 5M |

Reading from Buffer

Syntax: buf.toString([encoding][, start][, end])

Compare Buffer

Syntax: buf.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])

Copy Buffer

Syntax: buf.copy(target[, targetStart[, sourceStart[, sourceEnd]]])

b) Given a url of the format http://localhost:8080/sample.txt?gender=xxxx&halloween_heist=yyyy. Write a server code to get all objects from a MongoDB collection that satisfies the query. Also include code to "post" one object given in a post request body to this collection. Handle errors as well.

**Scheme and Solution**

```
//2 Marks for importing all modules
var http = require('http');
 var url = require('url');
 var fs = require('fs');
 var qs = require('querystring');
 var MongoClient = require('mongodb').MongoClient;

//2 marks for create server and parse URL
 http.createServer(function(request,response){
 if(request.method=='GET'){
 response.writeHead(200,{'Content-type':'text/html'});
 var myurl = url.parse(request.url)
 var query = myurl.query;
 var qobj = qs.parse(query);

//3 Marks for database connectivity and GET/find the data
 MongoClient.connect('mongodb://localhost:27017',
 {useUnifiedTopology: true},
 function(err,client){
 if(err) throw err;
 const db = client.db('newdb');
 db.collection('any_collection').find({qobj}).toArray(
 function(err,docs){
 if (err) throw err;
 response.writeHead(200,{'Content-type': 'application/json'})
 response.write(JSON.stringify(docs))
 client.close();
 response.end()
 }
 )
 })
 response.end();
 }

// 4 marks for POST method with error handling
 if(request.method == 'POST'){
 var myurl = url.parse(request.url)
 var pathname = myurl.pathname;
 let body = [];
 request.on('data',(chunk)=>{
 body.push(chunk);
```

12M

```
 })
 .on('end',()=>{
 body = Buffer.concat(body).toString()
 MongoClient.connect('mongodb://localhost:27017',
 {useUnifiedTopology: true},
 function(err,client){
 if(err) throw err;
 const db = client.db('newdb');
 db.collection('any_collection').insertOne(JSON.parse(body),
 function(err, res){
 if (err) throw err;
 console.log('document inserted');
 client.close();
 response.end();
 }
 )
 })
 });
 }
 }).listen(8080); //1 Mark
console.log('Server is up and running on http://localhost:8080');
```

c) The following Node.js program uses the Node fs module to read a large file twice using two different API calls. When run, the programs print the numbers 1 through 5 to the console. List the order in which the numbers are printed and justify your answer.

```
var fs = require("fs");
fs.readFile("./sample.txt", function () {
                console.log("1");
              });
          console.log("2");
function Fileread(fileName, readcallback) {
              var f = fs.readFileSync(fileName);
              console.log("3");
             readcallback();
          }
Fileread("./sample.txt", function () {
           console.log("4");
          });
console.log("5");
```

**Scheme and Solution**

**Output**: **//1 Mark**

2
3
4
5
1

**//2 Marks for explanation**

The fs.readFile call will call it's done callback later so the first log we get is '2'. Afterwards, Fileread will be called and print '3' and then call its readcallback

3M

| | | | |
|---|---|---|---|
| | | which prints '4'. Execution then continues with '5' printed and later the fs.readFile callback will fire, printing '1'; | |
| | | | |
| 5. | a) | What is RESTful API? Explain any 4 design specification/constraints of REST API.<br>**Scheme and Solution**<br>**//1 M Explanation on RESTAPI**<br>A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer<br>**//1 Mark each for any 4 design considerations**<br>Client Server, Scalable, Cacheable, Uniform Interface, Layered System, Code on Demand | 5M |
| | b) | When doing routing of URLs in ExpressJS, we used routes that contained a colon character (e.g. "/hello/:id" ) yet we never included a colon in the hierarchical part of a URL we used.<br>(i) Explain the purpose of this colon character?<br>(ii) Describe what would happen if we just deleted the colon from the routes.<br>**Scheme and Solution**<br>**//2 Marks each**<br>i. The colon character indicates that 'id' is a route parameter, i.e., whatever the user places in that position in the URL will be a route parameter named ‘id’. Router will attach this parameter to our component's 'props' object so we can display the data associated with this parameter in the component.<br>ii. If we delete the colon, Router will only route /hello/id to our component, instead of /hello/anything to our component. Since our component also probably<br>has some dependency on the ‘id’ component on the ‘props’ object, our app may not render what is required. | 4M |
| | c) | Explain the role of express middleware function. What are the different types of middleware an express application can use?<br>**Scheme and Solution**<br>**//2 reduce 1 mark if next is not mentioned +1 Mark(for any 2 types)**<br>Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle. These functions are used to modify req and res objects for tasks like parsing request bodies, adding response headers, etc.<br>• Application-level middleware<br>• Router-level middleware<br>• Error-handling middleware<br>• Built-in middleware<br>Third-party middleware | 3M |
| | d) | Write a program to upload a file to Node.js server using express file upload library. | 8M |

**Scheme and Solution**

```
// 1 mark for importing modules express and express fileupload
var express=require("express")
var app= express()
var fileupload= require("express-fileupload")
app.use(fileupload());
//3 marks for post method
app.post('/upload',function(req,res){
   if(!req.files||req.files.length==0)
   return res.status(400).send("No file to upload")
   var sampleFile= req.files.sampleFile;
   sampleFile.mv("./files/"+sampleFile.name,function(err){
      if(err)
      return req.statusCode(500).send(err)
      res.send("File"+sampleFile.name+"Uploaded")
   })
})
//3 Marks for get method and form creation
app.get("/form",function(req,res){
   var retform="<form action='http://localhost:3000/upload' method='post' encType='multipart/form-data'><input type='file' name='sampleFile'/> <input type='submit' value='upload'/></form>";
   res.send(retform)
   })
//1 Mark
app.listen(3000,function(){
   console.log("Server is up and running")
})
```