PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE19CS204

NOVEMBER 2020: IN SEMESTER ASSESSMENT B Tech 3rd SEMESTER
ISA – 2

UE19CS204 (4 credits) – WEB TECHNOLOGIES

Time: 80 Minutes	Answer All Questions	Max Marks: 40

1. a) Describe the different types of Modules in NodeJS. Give examples of each. **3**

Answer:

- inbuilt global modules Ex: Console module, Timer Module
- Core modules that need to be explicitly included in our application but need not be installed Ex: fs, http
- Modules created by third parties that need to be installed before including ex. node-fetch, cookie-parser

b) Write a code snippet that reads from a file “data.txt” asynchronously (non-blocking). It then outputs the first 200 characters to the file “data200.txt” (Note: use string methods). It should handle error appropriately. **3**

Answer:

```
var fs = require('fs');
fs.readFile (‘data.txt', function(err,data) {
   if(err) console.log(err);
   else {
      console.log(data);
      fs.writeFile(“data200.txt”, data.substr(0,200), function(err){
           if (err) console.err(err);
      });
   }
});
```

ReadFile, WriteFile, Calling writeFile within Readfile callback function – 3 marks

c) Write a program that receives a POST request which is a JSON string and writes the message body into a file whose name is sent as the pathname in the URL. Note: write the server code only. Client code not required. **4**

Answer:

```
var http = require('http');
var fs = require('fs');
http.createServer(function(request,response){
         if(request.method=='POST'){
                    var myurl = url.parse(request.url)
                    var pathname = myurl.pathname;
                    let body = [];
                    request.on('data',(chunk)=>{
                              body.push(chunk);
                              console.log(chunk.toString())
                    })
                    .on('end',()=>{
```

		body = Buffer.concat(body).toString()
		fs.writeFile(pathname.substr(1),body,(err,res)=>{
		    response.writeHead(200,{'Content-type':'text/plain'});
		    response.end("Message Saved");
		})
		})
		}
		}).listen(8080);
		CreateServer + listen – 1 mark
		data and end events – 1 mark
		Buffer handling – 1 mark
		writeFile – 1 mark	

2.	a)	In a MongoDB database, there is a "course" collection. From the collection, write queries to do the following: - List all documents - List documents with code = "UE19CS204" - List the first document with credits = "4" Answer: - db.collection("course").find({}) - db.collection("course").find({code : "UE19CS204"}) - db.collection("course").findOne({credits = "4"}) (1 mark each)	3

	b)	Given a URL, http://localhost:8080/pes.htm?city=Bangalore&year=2020
		Write a code snippet to parse the URL and store the hostname, pathname, and the request parameters to a file named "requestlog.txt".
		Answer:
		var myurl = url.parse(request.url)
		var pathname = myurl.pathname;
		var query = request.query;
		var host = myurl.host;
		fs.writeFile("requestlog.txt", host + pathname + query, function(err){
		});
		(2 + 1 marks)
			3

	c)	Write code snippet to connect to a db named "ipl" on a Mongodb instance running on localhost:27017 and write the following details into the "player" collection: name="Virat Kohli", team="RCB", stats="{runs:'10000', srate:'123.4'}". It should handle error appropriately.
		Note: write only the code required to write into the database.
		Answer:
		MongoClient.connect('mongodb://localhost:27017',{ //OR localhost:27017/ipl - 1 mark
		    useUnifiedTopology:true
		}, function(err,client){
		    if(err) throw err; - 1 mark
		    const db = client.db('ipl');
		    db.collection('player).insertOne(data, - 2 marks
		    function(err,res){
		        if(err) throw err;
		        response.write('document inserted..')
		        client.close();
			4

```
                    response.end()
          })
})
```

MongoClient.connect – 1mark

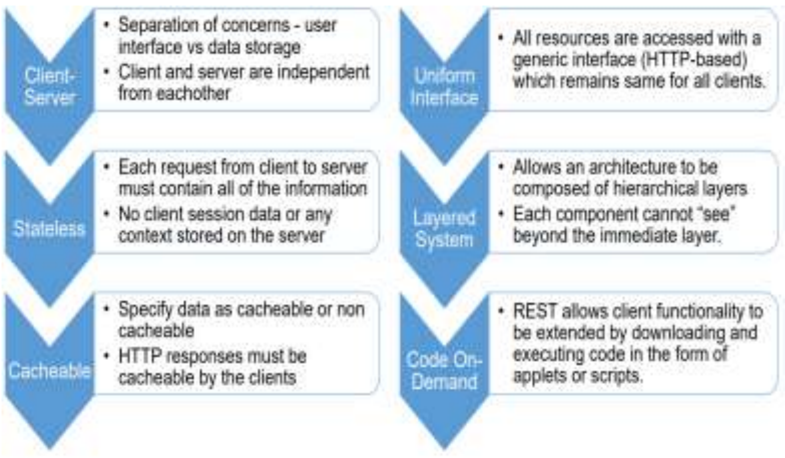
db.db or /db – 1 mark

insertOne – 1 mark

error handling – 1 mark

3. a) Explain any four design specifications (restrictions) of the REST APIs. List the mapping of HTTP methods with CRUD operations typically used. **5**

Answer:



Client-Server
- Separation of concerns - user interface vs data storage
- Client and server are independent from eachother

Stateless
- Each request from client to server must contain all of the information
- No client session data or any context stored on the server

Cacheable
- Specify data as cacheable or non cacheable
- HTTP responses must be cacheable by the clients

Uniform Interface
- All resources are accessed with a generic interface (HTTP-based) which remains same for all clients.

Layered System
- Allows an architecture to be composed of hierarchical layers
- Each component cannot "see" beyond the immediate layer.

Code On-Demand
- REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts.

GET – READ

POST – CREATE

PUT – UPDATE

DELETE – DELETE

(4 + 1 marks)

b) Write route for the following: **5**

GET /restaurant?rating=4

This should return all restaurants from the database "planner" and collection "restaurant" that have a rating given in the querystring.

Answer:

```
app.get("/restaurant", function(req,res){ - 1 mark
          MongoClient.connect('mongodb://localhost:27017',{ - 1 mark
                    useUnifiedTopology:true
          }, function(err,client){
                    if(err) throw err;
                    const db = client.db('planner'); mark
                    db.collection('student').find({rating:req.query.rating}).toArray(function(err,objs)
//OR find(req.query) - 2 mark
                    {
                              res.send(objs) - 1 mark
                    });
          });
});
```

4. a) Given the code snippet that defines the route and middleware functions. What will be the output when a request for http://localhost:3000/weather is made? **4**

```
app.get("/weather", function(req, res, next){
        console.log("Route Function")
        res.send("Welcome to Bangalore Weather page!!!")
})
app.use("/weather", function(req, res, next){
        console.log("This is executed after the route")
        next();
})
```

Answer:

It will only print Route Function in console and send the response Welcome to Bangalore Weather page!!!. The middleware function is never called as it is defined after the route function and the response cycle has completed.

(2 marks for output and 2 marks for explanation)

- 4 marks output

b) Write a middleware function that reads a “count” value from the cookie and increments it by 1 for every request received. The function should send a response like “You have visited the site: X times”. 6

Answer:

```
var cookieParser = require("cookie-parser"); - 1 mark
app.use(cookieParser()); - 1 mark
app.get(function(req,res){
    var count = parseInt(req.cookies.count); - 2 marks
    res.cookie("count",count+1).send("You have visited the site: "+count+1+”times”) - 2 marks
})
```