8 marks question are

Describe the architecture of a MEAN stack application, including the role of each component.

Upload pdf

Explain the concept of request and response objects in Express.

ExpressJS Request & Response

Request and Response object both are the callback function parameters and are used for Express.js and Node.js. You can get the request query, params, body, headers, and cookies. It can overwrite any value or anything there. However, overwriting headers or cookies will not affect the output back to the browser.

Topics Covered

- Request object
- Request object properties
- Request object methods
- Response object
- Response object properties
- Response object methods

Request object

Express.js is a request & response objects parameters of the callback function and are used for the Express applications. The request object represents the HTTP request and contains properties for the request query string, parameters, body, HTTP headers, etc.

Syntax : app.get('/', function (req, res) { })

Request Object Properties

These properties are represented below:

S.No	**Properties**	**Description**
1	req.app	Used to hold a reference to the instance of the express application.

2	req.body	Contains key-value pairs of data submitted in the request body. By default, it is undefined and is populated when you use body-parsing middleware such as body-parser.
3	req.cookies	This property contains cookies sent by the request, used for the cookie-parser middleware.
4	req.ip	req.ip is remote IP address of the request.
5	req.path	req.path contains the path part of the request URL.
6	req.route	req.route is currently-matched route.

Request Object Methods

There are various types of request object method, these methods are represented below:

req.accepts (types)

It is used to the check content types are acceptable, based on the request accept HTTP header field.

Example :

```
req.accepts('html');
//=>?html?
req.accepts('text/html');
// => ?text/html?

Copy
```

req.get(field)

req.get is used to returns the specified HTTP request header field.

Example :

```
req.get('Content-Type');
// => "text/plain"
req.get('content-type');
// => "text/plain"
```

```
req.get('Something');
// => undefined

Copy
```

req.is(type)

If the incoming request is "CONTENT-TYPE", this method returns true. HTTP header field matches the MIME type by the type parameter.

Example :

```
// With Content-Type: text/html; charset=utf-8
req.is('html');
req.is('text/html');
req.is('text/*');
// => true

Copy
```

req.param(name [, defaultValue])

req.param method is used to fetch the value of param name when present.

Example :

```
// ?name=sonia
          req.param('name')
         // => "sonia"
         // POST name=sonia
          req.param('name')
         // => "sonia"
         // /user/soniafor /user/:name
         req.param('name')
        // => "sonia"

Copy
```

Response Object

The response object specifies the HTTP response when an Express app gets an HTTP request. The response is sent back to the client browser and allows you to set new cookies value that will write to the client browser.

Response Object Properties

S.No	properties	Description
1	res.app	res.app is hold a reference to the instance of the express application that is using the middlewa
2	res.locals	Specify an object that contains response local variables scoped to the request.

Response Object Method

There are various types of response object method, these methods are represented below :

Response Append Method

Syntax : res.append(field, [value])

Response append method appends the specified value to the HTTP response header field. That means if the specified value is not appropriate so this method redress that.

Example :

```
res.append('Link', ['<http://localhost/>', '<http://localhost:3000/>']);
res.append('Warning', '299 Miscellaneous warning');

Copy
```

Response Attachment Method

Syntax : res.attachment('path/to/js_pic.png');

Response attachment method allows you to send a file as an attachment in the HTTP response.

Example :

```
res.attachment('path/to/js_pic.png');   .
```

```
Copy
```

Response Cookie Method

Syntax: `res.cookie(name, value [, options])`

It is used to set a cookie name to value. The value can be a string or object converted to JSON.

Example :

```
res.cookie('name', 'alish', { domain: '.google.com', path: '/admin', secure: true });
res.cookie('Section', { Names: [sonica,riya,ronak] });
res.cookie('Cart', { items: [1,2,3] }, { maxAge: 900000 });

Copy
```

Response Download Method

Syntax: `res.download(path [, filename] [, fn])`

Example:

```
res.download('/report-12345.pdf');

Copy
```

res.download method is transfer file at path as an “attachment” and the browser to prompt user for download.

Response End Method

Syntax: `res.end([data] [, encoding])`

Response end method is used to end the response process.

Example :

```
res.end();
res.status(404).end();

Copy
```

Response Get Method

Syntax : res.get(field)

res.get method provides HTTP response header specified by field.

Example :

```
res.get('Content-Type');

Copy
```

Response JSON Method

Syntax : res.json([body])

Response JSON method returns the response in JSON format.

Example :

```
res.json(null)
res.json({ name: 'alish' })

Copy
```

Response Render Method

Syntax : res.render(view [, locals] [, callback])

Response render method renders a view and sends the rendered HTML string to the client.

Example :

```
// send the rendered view to the client


res.render('index');


// pass a local variable to the view


res.render('user', { name: 'monika' }, function(err, html) {
```

```
// ...

});

Copy
```

Response Status Method

Syntax : res.status(code)

res.status method sets an HTTP status for the response.

Example :

```
res.status(403).end();
res.status(400).send('Bad Request');

Copy
```

Response Type Method

Syntax : res.type(type)

res.type method sets the content-type HTTP header to the MIME type.

Example :

```
res.type('.html');              // => 'text/html'
res.type('html');               // => 'text/html'
res.type('json');              // => 'application/json'
res.type('application/json');   // => 'application/json'
res.type('png');               // => image/png:
```

Describe the process of building a web application using Express and Node.js.

Pdf under mean project setup

How do you create a simple server in Node.js that returns Hello World?

A Server is a piece of computer hardware or software that provides functionality for other programs or devices, called clients. This architecture is called the client-server model. Node is an open-source, cross-platform runtime environment that allows developers to create all kinds of server-side tools and applications in JavaScript.

In the following example, we will create a simple server in Node.js that returns *Hello World* using an express server.
Create NodeJS Application: Initialize the NodeJS application using the following command:
npm init

Module Installation: Install the *express* module which is a web framework for NodeJS using the following command.
npm install express

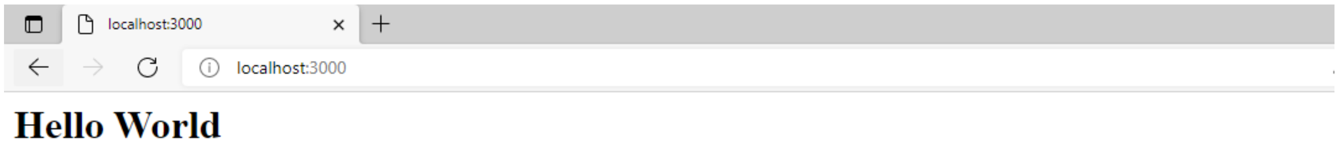
Implementation: Create an *app.js* file and write down the following code in it.

- app.js

```
// Require would make available the
// express package to be used in
// our code
const express = require("express");

// Creates an express object
const app = express();

// It listens to HTTP get request.
// Here it listens to the root i.e '/'
app.get("/", (req, res) => {

  // Using send function we send
  // response to the client
  // Here we are sending html
  res.send("<h1> Hello World </h1>");
});

// It configures the system to listen
// to port 3000. Any number can be
// given instead of 3000, the only
// condition is that no other server
// should be running at that port
app.listen(3000, () => {

  // Print in the console when the
  // servers starts to listen on 3000
  console.log("Listening to port 3000");
});
```

Step to run the application: Run the *app.js* file using the following command.
node app.js

Output: Now open your browser and go to *http://localhost:3000/*, you will see the following output:



localhost:3000

Hello World

output

So this is how you can set up the server and achieve the task. If you want to return anything else then pass that argument in *res.send()* of the *app.get()* function instead of “Hello World”.

What are CRUD operations in MongoDB? Enlist the various CRUD Operations in MongoDB Along with syntax for each.

Upload pdf under mongodb