St. Francis Institute of Technology Borivli (West), Mumbai-400103

(Autonomous Institute)

Department of Information Technology

Sub: Internet Programming

Experiment -7: File-operations using Nodejs Web Server

1. Aim: To build a simple Web Server using Node.js to perform file operations.

2. Objective: To build the web server using node.js and use the server for performing file operations.

3. Lab Outcome: Students will be able to create back-end applications using Node.js/ Express (PO3, PO5, PSO3, PSO4)

4. Prerequisite: Node.js

5. Requirements: The following are the requirements –

- PC/Laptop
- Visual Studio Code
- Browser

6. Pre-Experiment Theory:

Node.js is an open-source server environment. Node.js allows you to run JavaScript on the server. A common task for a web server can be to open a file on the server and return the content to the client. Node.js handles a file request in the following manner:

- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request. Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

Node.js can generate dynamic page content. It can create, open, read, write, delete, and close files on the server. It can collect form data, add, delete and modify data in your database,

Node.js files contain tasks that will be executed on certain events. A typical event is someone trying to access a port on the server. Node.js files must be initiated on the server before having any effect

Node.js has a set of built-in modules which you can use without any further installation. To include a module, use the *require()* function with the name of the module: You can create your own modules, and easily include them in your applications.

A. Node.js as a Web Server

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client. Use the *createServer()* method to create an HTTP server:

```
var http = require('http');
//create a server object:
```

```
http.createServer(function (req, res) {
res.write('Hello World!'); //write a response to the client
res.end(); //end the response
 }).listen(8080); //the server object listens on port 8080
```

The function passed into the *http.createServer()* method, will be executed when someone tries to access the computer on port 8080.

B.File handling using Node.js

The Node.js file system module allows you to work with the file system on your computer.
To include the File System module, use the *require()* method.

The ***fs*** module provides an API for interacting with the file system of your operating system.
We can use this module, in code like this:

```
const fs = require('fs');
```

Using this node module, we can perform various tasks such as opening files, writing data into a file, reading data from files, etc.

You can use ***fs.readFile()*** method to read files in Node. For example:

```
const fs = require('fs');
fs.readFile('./lorem.txt', (err, data) => {
  if(err) {
    return console.log('Error occurred while reading file');
   }
  console.log(data.toString());
});
```

If you just want to check the file existence in the system, use the ***access()*** function.

```
const fs = require('fs');
const path = './config.js';
fs.access(path, fs.F_OK, (err) => {
  if (err) {
    console.error(err);
    return;
  }
});
```

The file system module provides three methods to create files:

1. fs.open()
2. fs.writeFile()
3. fs.appendFile()

fs.open() method opens a new file or creates a new empty file if it does not exist in the specified path. It takes the second parameter which acts as a flag such as w for writing, w+ for reading and writing, etc.
fs.writeFile() method allows you to create or replace files with the content.
If a file exists, it will replace the content with the provided content and if the file does not exist, it will create it.

```
const fs = require('fs');
fs.open('file.txt', 'w', (err, file) => {
  if (err) {
```

```
    throw err;
  }
  console.log('Saved!');
});
```

fs.appendFile() method appends the provided content at the end of the file.

```
const fs = require('fs');
fs.appendFile('file.txt', ' Hello World', (err) => {
  if (err) {
    throw err;
  }
  console.log('Updated!');
});
```

To delete a file, we can use ***fs.unlink()*** method.

```
const fs = require('fs');
fs.unlink('file.txt', (err) => {
  if (err) {
    throw err;
  }
  console.log('File deleted!');
});
```

To rename a file, we can use the ***fs.rename()*** method.

```
const fs = require('fs');
fs.rename('newfile.txt', 'oldfile.txt', (err) => {
  if (err) {
    throw err;
  }
  console.log('File Renamed!');
});
```

You can also copy files using the ***fs.copy()*** method.

```
const fs = require('fs');
fs.copyFile(file.txt', 'copyfile.txt', (err) => {
  if (err) {
    throw err;
  }
  console.log('File is copied!');
});
```

You can write a switch case in node.js to choose to perform any one operation using the code

```
const prompt = require('prompt-sync')({sigint: true});
var ch=parseInt(prompt("enter"));
switch(ch)
{
case 1:access1(); break;
case 2:open();  break;
case 3:appendfile();   break;
case 4: read();  break;
}
```

7. Laboratory Exercise:

A. Procedure

1. **Create a Web Server**
   - Install node.js (npm installs automatically)
   - Open terminal window (use cmd command)
   - Check version of node.js (node -v) and version of npm (npm -v)
   - Open vs-code and open the folder.
   - Write the node.js code. Save the file with .js extension.
   - Initiate the file using the command node <filename>

View the Output on browser window by accessing the local host on the port number used in the code. (eg. http://localhost:8080)

2. **File Operations using Node.js**
   - Install node.js (npm installs automatically)
   - Open terminal window (use cmd command)
   - Check version of node.js (node -v)
   - Check version of npm (npm -v)
   - Install prompt-sync from terminal using following commands -
     - npm install prompt-sync
   - Open vs-code and open the folder.
   - Create a text file.
   - Write the node.js code

View the Output on terminal window by running the command *node <filename>*

B. Program Code

1. Write a code to build a simple web server using Node.js that returns a message when the user requests the server. Use HTTP module to achieve web server functionality.
2. Write a code to perform the following File Handling operations using Node.js as per the chosen operation (use switch case for choosing the operation)

- Read
- Access
- Open
- Write
- Append

8. Post Experimental Exercise-

1. Differentiate between net and http module?
2. What is a web server? Draw and explain Web Application Architecture.

9. Results/Observations/Program output:

Present the program code and output

10. Conclusion:

Write what was performed in the experiment.

11. References:

- https://nodejs.org/en/
- https://www.w3schools.com/nodejs/default.asp
- https://www.tutorialsteacher.com/nodejs/create-nodejs-web-server
- https://www.youtube.com/watch?v=lm86czWdrk0
- https://codeforgeek.com/node-js-tutorial-step-by-step/