Server Side Scalable Application

Unit-1

1.1 File system module in node.js

The **File System (fs) module** in Node.js allows you to interact with the file system, such as creating, reading, updating, and deleting files.

1.1.1 Inputs from Users

Notes:

- In Node.js, you can take **user input** from the command line using the process.argv array.
- process.argv stores all command-line arguments passed when running a Node.js script.

process.argv is an array:

- process.argv[0] → Node.js executable path.
- process.argv[1] → File path of the script.
- process.argv[2] and beyond → User inputs.

Example 1:

```
// file: userInput.js
const name = process.argv[2]; // First input after filename
console.log(`Hello, ${name}! Welcome to Node.js.`);
```

Run:

```
node userInput.js Chirag
```

Output:

```
Hello, Chirag! Welcome to Node.js.
```

Example 2: Sum of Two Numbers

```
// file: sumInput.js
const num1 = parseInt(process.argv[2]);
const num2 = parseInt(process.argv[3]);
const sum = num1 + num2;

console.log(`Sum of ${num1} and ${num2} is: ${sum}`);
```

Run:

```
// Reading from the file
const data = fs.readFileSync('example.txt', 'utf8');
console.log('File Content:', data);

// Appending data
fs.appendFileSync('example.txt', '\nThis is the appended content.');

// Reading again after appending
const updatedData = fs.readFileSync('example.txt', 'utf8');
console.log('Updated File Content:', updatedData);
```

Output:

```
File Content: This is the initial content.
Updated File Content: This is the initial content.
This is the appended content.
```

1.1.4 Operations Associated with File System Module

Notes:

Here are the key operations:

Operation	**Method**	**Description**
Create	writeFileSync	Creates a new file and writes data
Read	readFileSync	Reads content from a file
Update	appendFileSync	Adds data to an existing file
Delete	unlinkSync	Removes a file from the system
Exists	existsSync	Check if file exists.
Create folder	mkdirSync	Create a folder.

Example: Full CRUD Operations

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

// 1. Create File
fs.writeFileSync('data.txt', 'Initial file content');
```

```
// 2. Read File
let fileData = fs.readFileSync('data.txt', 'utf8');
console.log('File Content:', fileData);

// 3. Update File
fs.appendFileSync('data.txt', '\nAdded new content');

// Read Updated File
fileData = fs.readFileSync('data.txt', 'utf8');
console.log('Updated File Content:', fileData);

// 4. Delete File
fs.unlinkSync('data.txt');
console.log('File deleted successfully.');

// Create folder
if (!fs.existsSync('myFolder')) {
  fs.mkdirSync('myFolder');
  console.log('Folder created successfully.');
}

// Create file inside folder
fs.writeFileSync('myFolder/info.txt', 'Folder file content');

// Read file from folder
const content = fs.readFileSync('myFolder/info.txt', 'utf8');
console.log('File content:', content);
```

1.2 JSON Data, HTTP Server and Client

JSON Data

- **JSON (JavaScript Object Notation)** is a lightweight data-interchange format.
- It is easy for humans to read and write and easy for machines to parse and generate.

Example JSON:

```
{
  "name": "John",
  "age": 30,
  "city": "New York"
}
```

Reading JSON in Node.js:

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

const data = fs.readFileSync('data.json');
const jsonData = JSON.parse(data);
console.log(jsonData);
```

HTTP Server and Client

HTTP Server Example (Node.js Built-in Module)

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

const server = http.createServer((req, res) => {
  if (req.url === '/home') {
    res.writeHead(200, {'Content-Type': 'application/json'});
    res.end(JSON.stringify({message: "Welcome to Home Page"}));
  } else {
    res.writeHead(404);
    res.end('Page Not Found');
  }
});

server.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

1.2.1 Sending and Receiving Events with EventEmitters

What is EventEmitter?

- Node.js has a built-in module called events.
- The EventEmitter class allows us to **create and handle custom events**.

Example:

```
const EventEmitter = require('events');
const eventEmitter = new EventEmitter();
```

```
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```
3. Restart Nginx:

```
sudo systemctl restart nginx
```
4. Test Deployment:

Visit your server IP or domain in the browser → App should be live!