



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

AY: 2025-26

<b>Class:</b>	<b>TE</b>	<b>Semester:</b>	<b>V</b>
<b>Course Code:</b>	<b>CSC502</b>	<b>Course Name:</b>	<b>WC</b>

<b>Name of Student:</b>	<b>Shreya Sandeep Yadav</b>
<b>Roll No. :</b>	<b>80</b>
<b>Experiment No.:</b>	<b>06</b>
<b>Title of the Experiment:</b>	<b>Node.Js: Installation and Configuration, Callbacks, Event loops, Creating express</b>
<b>Date of Performance:</b>	<b>03/09/25</b>
<b>Date of Submission:</b>	<b>10/09/25</b>

### Evaluation

<b>Performance Indicator</b>	<b>Max. Marks</b>	<b>Marks Obtained</b>
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

<b>Performance Indicator</b>	<b>Exceed Expectations (EE)</b>	<b>Meet Expectations (ME)</b>	<b>Below Expectations (BE)</b>
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

**Name of Faculty : Ms. Kshitija Gharat**

**Signature :**

**Date:**



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

**Aim:** To implement file system operations and path handling in Node.js, and to design a simple REST API using Express framework with middleware functions.

### Objective:

To understand the usage of File System (fs) module for reading, writing, and updating files

To learn how to work with path module for handling file and directory paths

To develop RESTful API endpoints using Express framework

To apply middleware in Express for logging and request handling

### Requirement:

Node.js installed (version 18 or above recommended)

Express framework (installed via npm)

Text editor such as VS Code

Web browser or Postman for API testing

Theory:

Node.js provides inbuilt modules and frameworks for backend development.

File System Module (fs):

This module allows reading, writing, updating, and deleting files. Example functions include `fs.readFileSync` and `fs.writeFileSync`.

Path Module (path):

This module helps in working with file and directory paths. Example usage is `path.join(__dirname, 'file.txt')`.

Express Framework:

Express is a lightweight Node.js framework to build web servers and REST APIs. It provides routing, middleware, and response handling.

Middleware in Express:

Middleware functions are executed between request and response. They are used for logging, authentication, validation, and other purposes.

### Procedure:

Step 1: Install Node.js and initialize the project using the following commands

```
mkdir node-experiment
```

```
cd node-experiment
```

```
npm init -y
```

```
npm install express
```

Step 2: Create `server.js` file and include required modules

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

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

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

```
const app = express();
```

```
const PORT = 3000;
```

Step 3: Perform File System operations

```
fs.writeFileSync('sample.txt', 'Hello, Node.js FS Module!');
```

```
const data = fs.readFileSync('sample.txt', 'utf8');
```



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

```
console.log("File Content:", data);
```

Step 4: Use Path Module

```
const filePath = path.join(__dirname, 'sample.txt');
```

```
console.log("Absolute File Path:", filePath);
```

Step 5: Create Express REST API with Middleware

```
app.use((req, res, next) => {
```

```
  console.log(`${req.method} ${req.url}`);
```

```
  next();
```

```
});
```

```
app.get('/', (req, res) => res.send('Welcome to Node.js REST API'));
```

```
app.get('/data', (req, res) => res.json({ message: "Hello World" }));
```

```
app.listen(PORT, () => {
```

```
  console.log(`Server running at http://localhost:${PORT}`);
```

```
});
```

Step 6: Run the server using the command

```
node server.js
```

Step 7: Test endpoints using browser or Postman

<http://localhost:3000/> → Displays welcome message

<http://localhost:3000/data> → Returns JSON response

### Code-

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

```
const app = express();
```

```
const port = 3000;
```

```
app.use(express.json());
```

```
let items = [];
```

```
app.post('/items', (req, res) => { const newItem = { id: items.length + 1, ...req.body };
```

```
  items.push(newItem);
```

```
  res.status(201).json(newItem);
```

```
});
```

```
app.get('/items', (req, res) => { res.json(items);
```

```
});
```

```
app.get('/items/:id', (req, res) => { const item = items.
```

```
  find(i => i.id === parseInt(req.params.id));
```

```
  if (!item) return res.status(404).send('Item not found');
```

```
  res.json(item);
```

```
});
```

```
app.put('/items/:id',
```

```
(req, res) => { const item = items.find(i => i.id === parseInt(req.params.id));
```

```
  if (!item) return res.status(404).send('Item not found');
```

```
  Object.assign(item, req.body);
```

```
  res.json(item);
```

```
});
```

```
app.delete('/items/:id',
```



```
(req, res) => { const index = items.findIndex(i => i.id === parseInt(req.params.id));  
if (index === -1) return res.status(404).  
send('Item not found');  
const deletedItem = items.splice(index, 1);  
res.json(deletedItem[0]);  
});  
app.listen(port, () =>  
{ console.log(`Server running at http://localhost:\${port}`);  
});
```

### Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  POSTMAN CONSOLE  COMMENTS  
  
PS D:\WC_NODE\my-crud-app> node server.js  
>>  
Server running at http://localhost:3000  
█
```

### Conclusion:

Node.js provides powerful modules like fs and path for file handling. Using Express, developers can easily create REST APIs and enhance functionality with middleware. This experiment demonstrates integration of core Node.js modules with Express framework for backend development.