👷 Day 12 Lab Sheet – File System, URL Module & Express.js Introduction

Objective

By the end of this lab, you will:

- Work with Node.js **File System (fs)** module
- Parse and handle **URLs** in Node
- Create a basic **Express.js** web application
- Build simple APIs using Express

Setup

1. Create a folder:
2. mkdir Day12_NodeExpress
3. cd Day12_NodeExpress
4. Open in VS Code or any IDE.
5. Make sure Node.js and npm are installed:
6. node -v
7. npm -v

Task 1: Read & Write Files using fs

1. Create fileOps.js:
2. const fs = require('fs');
3.
4. // Write to a file
5. fs.writeFile('message.txt', 'Hello, this is Node.js!', (err) => {
6. if (err) throw err;
7. console.log('File written successfully.');
8.
9. // Read the file after writing
10. fs.readFile('message.txt', 'utf8', (err, data) => {
11. if (err) throw err;
12. console.log('File Content:', data);
13. });

14. });
15. Run:
16. node fileOps.js

✅ A file message.txt should be created, and its contents printed in the console.

Task 2: Append & Delete Files

1. Append data to the existing file:
2. const fs = require('fs');
3.
4. fs.appendFile('message.txt', '\nThis text is appended.', (err) => {
5. if (err) throw err;
6. console.log('Text appended successfully.');
7. });
8. Delete a file:
9. fs.unlink('oldFile.txt', (err) => {
10. if (err) console.log('File not found.');
11. else console.log('File deleted successfully.');
12. });

✅ Observe the changes in your project folder.

Task 3: Working with JSON Files

1. Create data.json with this content:
2. {"name": "Alice", "course": "Web Development"}
3. Create jsonOps.js:
4. const fs = require('fs');
5.
6. fs.readFile('data.json', 'utf8', (err, data) => {
7. if (err) throw err;
8. const obj = JSON.parse(data);
9. console.log('Student Name:', obj.name);
10. });

✅ Reads JSON data and displays the name in console.

Task 4: Using the URL Module

1. Create urlDemo.js:
2. const url = require('url');
3.
4. const address = 'http://localhost:3000/home?name=Alice&age=22';
5. const parsed = url.parse(address, true);
6.
7. console.log('Host:', parsed.hostname);
8. console.log('Path:', parsed.pathname);
9. console.log('Query Object:', parsed.query);
10. console.log('Name from Query:', parsed.query.name);
11. Run:
12. node urlDemo.js

✅ Outputs parsed URL details in the console.

Task 5: Combine fs, http & url

1. Create serverFileRead.js:
2. const http = require('http');
3. const fs = require('fs');
4. const url = require('url');
5.
6. http.createServer((req, res) => {
7. const q = url.parse(req.url, true);
8.
9. if (q.pathname === '/read') {
10. fs.readFile('message.txt', 'utf8', (err, data) => {
11. if (err) {
12. res.writeHead(404);
13. res.end('File not found');

```
    } else {
      res.writeHead(200, { 'Content-Type': 'text/plain' });
      res.end(data);
    }
  });
} else {
  res.writeHead(200);
  res.end('Visit /read to view file content');
}
}).listen(3000);

console.log('Server running at http://localhost:3000');
```

✅ Visit http://localhost:3000/read → file content is displayed.

Task 6: Setup Express.js

```
1. Initialize npm project:
2. npm init -y
3. npm install express
4. Create app.js:
5. const express = require('express');
6. const app = express();
7.
8. app.get('/', (req, res) => {
9.   res.send('Welcome to Express.js!');
10. });
11.
12. app.listen(4000, () => {
13.   console.log('Server running at http://localhost:4000');
14. });
```

✅ Visit http://localhost:4000 → shows message from Express.

Task 7: Express Routes

Add more routes inside app.js:

```
app.get('/about', (req, res) => {
  res.send('About Page');
});


app.get('/contact', (req, res) => {
  res.send('Contact Page');
});
```

✅ Visit:

- /about → About Page
- /contact → Contact Page

Task 8: Sending JSON Responses

```
app.get('/api/student', (req, res) => {
  res.json({
    name: 'Alice',
    age: 22,
    course: 'Web Development'
  });
});
```

✅ Visit /api/student → JSON data appears in the browser.

Task 9: Handling Query Parameters

```
app.get('/greet', (req, res) => {
  const name = req.query.name || 'Guest';
  res.send(` Hello, ${name}! ` );
});
```

✅ Visit /greet?name=Bob → displays “Hello, Bob!”.

Task 10: Serving Static Files

1. Create a folder named public and add index.html inside:
2. <h1>Welcome to Static Express Page</h1>
3. Add this to app.js:
4. app.use(express.static('public'));

✅ Visit http://localhost:4000/index.html → static file loads.

✅ **Deliverables**

- fileOps.js, jsonOps.js, urlDemo.js, serverFileRead.js
- app.js (Express routes and API)
- public/index.html (static file demo)

All scripts should execute successfully and display correct output in browser or terminal.

💡 **Optional Challenge**

👉 Create an Express app that:

- Has routes /home, /about, /contact.
- Serves a JSON list of students at /api/students.
- Uses fs to read data from a file and send it as a response.