Day 18 — Asynchronous Programming in Node.js

Project Report

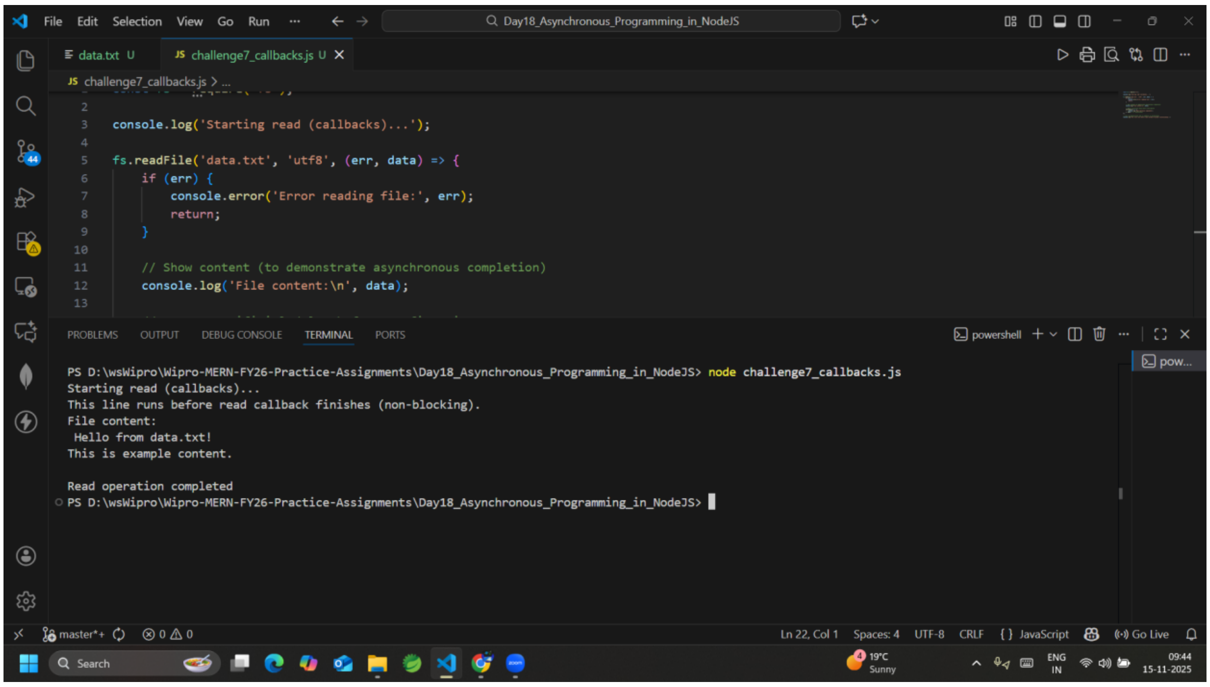
Challenge 7 — Callbacks

The user must read the content of a file (data.txt) asynchronously using callbacks and, after the read is complete, print a confirmation message. The challenge also requires demonstrating non-blocking behavior and adding an intentional delay before confirming completion.

Key Code Snippets:

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

console.log('Starting read (callbacks)...');

fs.readFile('data.txt', 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading file:', err);
        return;
    }

    // Show content (to demonstrate asynchronous completion)
    console.log('File content:\n', data);

    // Bonus: artificial delay before confirmation
    setTimeout(() => {
        console.log('Read operation completed');
    }, 1000); // 1 second delay
});

// This log demonstrates that fs.readFile is non-blocking
console.log('This line runs before read callback finishes (non-blocking).');
```

Output:


File Edit Selection View Go Run ···
Day18_Asynchronous_Programming_in_NodeJS
data.txt U
challenge7_callbacks.js U
challenge7_callbacks.js > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
master*+ ⊗ 0 ⚠ 0
Ln 22, Col 1 Spaces: 4 UTF-8 CRLF { } JavaScript Go Live
19°C Sunny
ENG IN
09:44
15-11-2025

```

console.log('Starting read (callbacks)...');

fs.readFile('data.txt', 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading file:', err);
        return;
    }

    // Show content (to demonstrate asynchronous completion)
    console.log('File content:\n', data);

```

```
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS> node challenge7_callbacks.js
Starting read (callbacks)...
This line runs before read callback finishes (non-blocking).
File content:
 Hello from data.txt!
This is example content.

Read operation completed
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS>
```

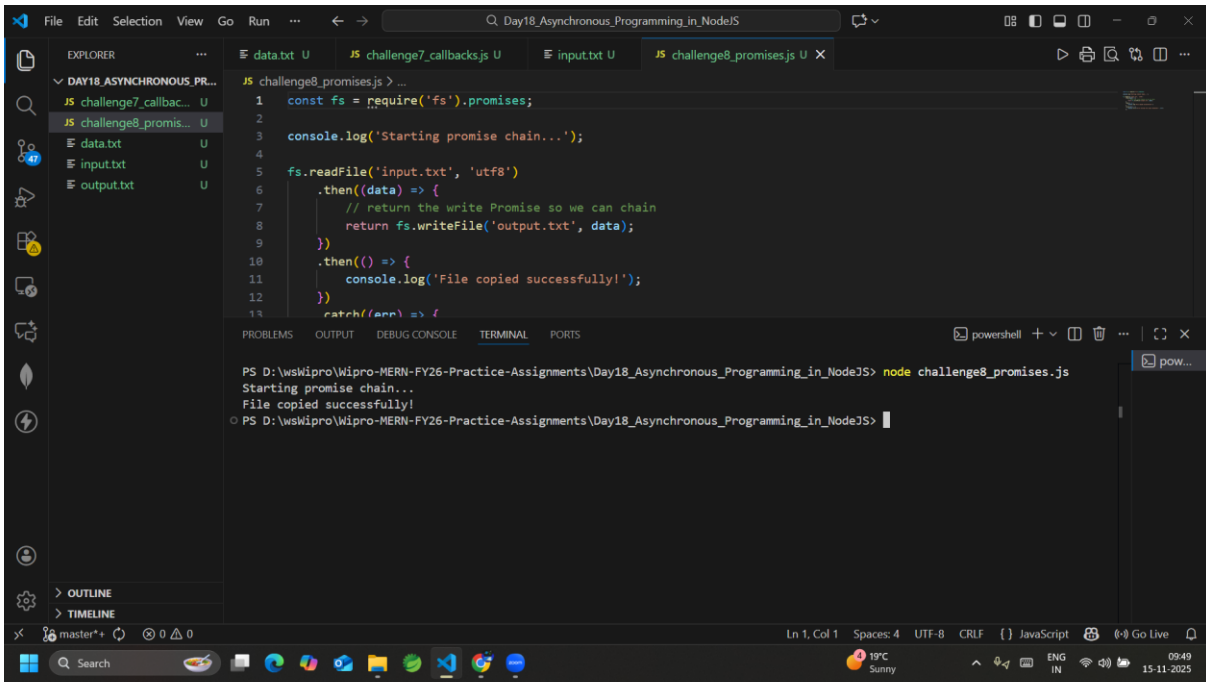
Challenge 8 — Promises

The user must perform a file-copy operation using Promises. File data is read from input.txt and then written into output.txt by chaining Promise operations. Errors must be handled gracefully using .catch().

Key Code Snippets:

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

console.log('Starting promise chain...');

fs.readFile('input.txt', 'utf8')
    .then((data) => {
        // return the write Promise so we can chain
        return fs.writeFile('output.txt', data);
    })
    .then(() => {
        console.log('File copied successfully!');
    })
    .catch((err) => {
        console.error('Error during file copy (promises):', err);
    });
```

Output:



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

console.log('Starting promise chain...');

fs.readFile('input.txt', 'utf8')
    .then((data) => {
        // return the write Promise so we can chain
        return fs.writeFile('output.txt', data);
    })
    .then(() => {
        console.log('File copied successfully!');
    })
    .catch((err) => {
```

```
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS> node challenge8_promises.js
Starting promise chain...
File copied successfully!
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS>
```

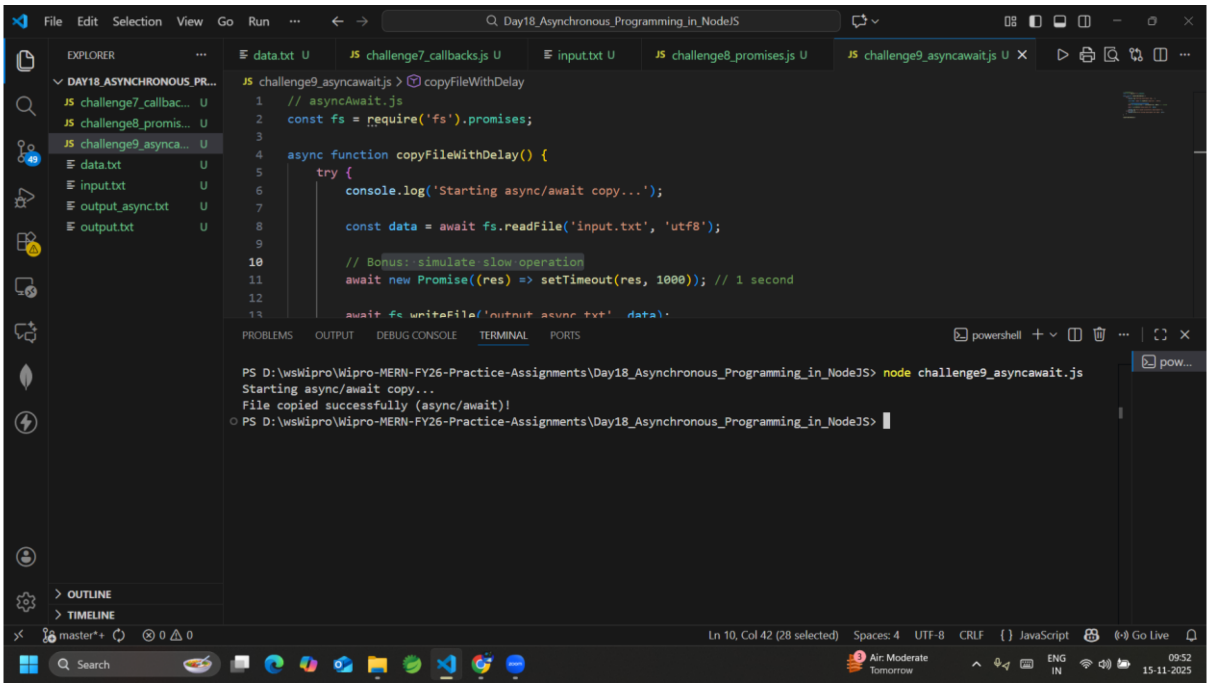
Challenge 9 — Async/Await

The user must implement the same file-copy operation using async/await for cleaner, synchronous-looking asynchronous code. A custom delay must be added to simulate slow operations, and error handling must be done using try/catch.

Key Code Snippets:

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

async function copyFileWithDelay() {
    try {
        console.log('Starting async/await copy...');

        const data = await fs.readFile('input.txt', 'utf8');

        // Bonus: simulate slow operation
        await new Promise((res) => setTimeout(res, 1000)); // 1 second

        await fs.writeFile('output_async.txt', data);

        console.log('File copied successfully (async/await)!');
    } catch (err) {
        console.error('Error during async/await file copy:', err);
    }
}

copyFileWithDelay();
```

Output:



File Edit Selection View Go Run
Day18_Asynchronous_Programming_in_NodeJS
EXPLORER
DAY18_ASYNCHRONOUS_PR...
challenge7_callbac... U
challenge8_promis... U
challenge9_asynca... U
data.txt U
input.txt U
output_async.txt U
output.txt U
data.txt U
challenge7_callbacks.js U
input.txt U
challenge8_promises.js U
challenge9_asyncawait.js U
challenge9_asyncawait.js > copyFileWithDelay
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
OUTLINE
TIMELINE
master*+
Ln 10, Col 42 (28 selected) Spaces: 4 UTF-8 CRLF { } JavaScript Go Live

```javascript
// asyncAwait.js
const fs = require('fs').promises;

async function copyFileWithDelay() {
    try {
        console.log('Starting async/await copy...');

        const data = await fs.readFile('input.txt', 'utf8');

        // Bonus: simulate slow operation
        await new Promise((res) => setTimeout(res, 1000)); // 1 second

        await fs.writeFile('output_async.txt', data);
```

```
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS> node challenge9_asyncawait.js
Starting async/await copy...
File copied successfully (async/await)!
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day18_Asynchronous_Programming_in_NodeJS>
```