

Terminal :-

The screenshot shows a Visual Studio Code window with a terminal running a Node.js script. The Explorer sidebar on the left shows the file structure of the project, including 'index.js', 'data.json', and 'package.json'. The terminal output shows the execution of the script, which implements a task manager with the following features:

- Task Manager:**
 - 1. Add a new task
 - 2. View tasks
 - 3. Mark task as complete
 - 4. Remove a task
 - 5. Exit

The terminal output shows the following sequence of events:

```
at markTaskComplete (C:\Users\Dell\Desktop\FS-19\Node.js\FS and path Module and readline Implement\index.js:54:19)
PS C:\Users\Dell\Desktop\FS-19\Node.js\FS and path Module and readline Implement> node .\index.js

Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 1
Enter new task: Shyam
task added

Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 2
1. Bittu
2. Ram
3. Shyam

Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 3
Enter task number to mark as complete: 2
Task "Ram" marked as complete.

Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 4
Enter task number to remove: 1
Task removed: "Bittu"

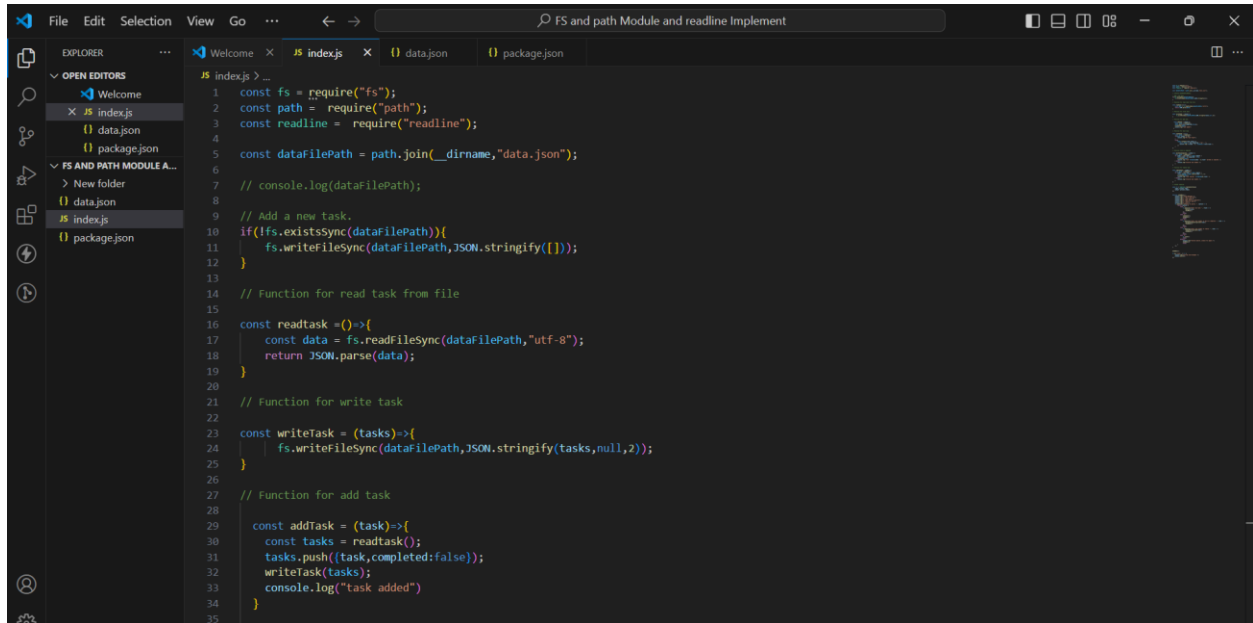
Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 4
Enter task number to remove: 1
Task removed: "Bittu"

Task Manager:
1. Add a new task
2. View tasks
3. Mark task as complete
4. Remove a task
5. Exit

Choose an option: 5
Exiting task manager.
PS C:\Users\Dell\Desktop\FS-19\Node.js\FS and path Module and readline Implement>
```

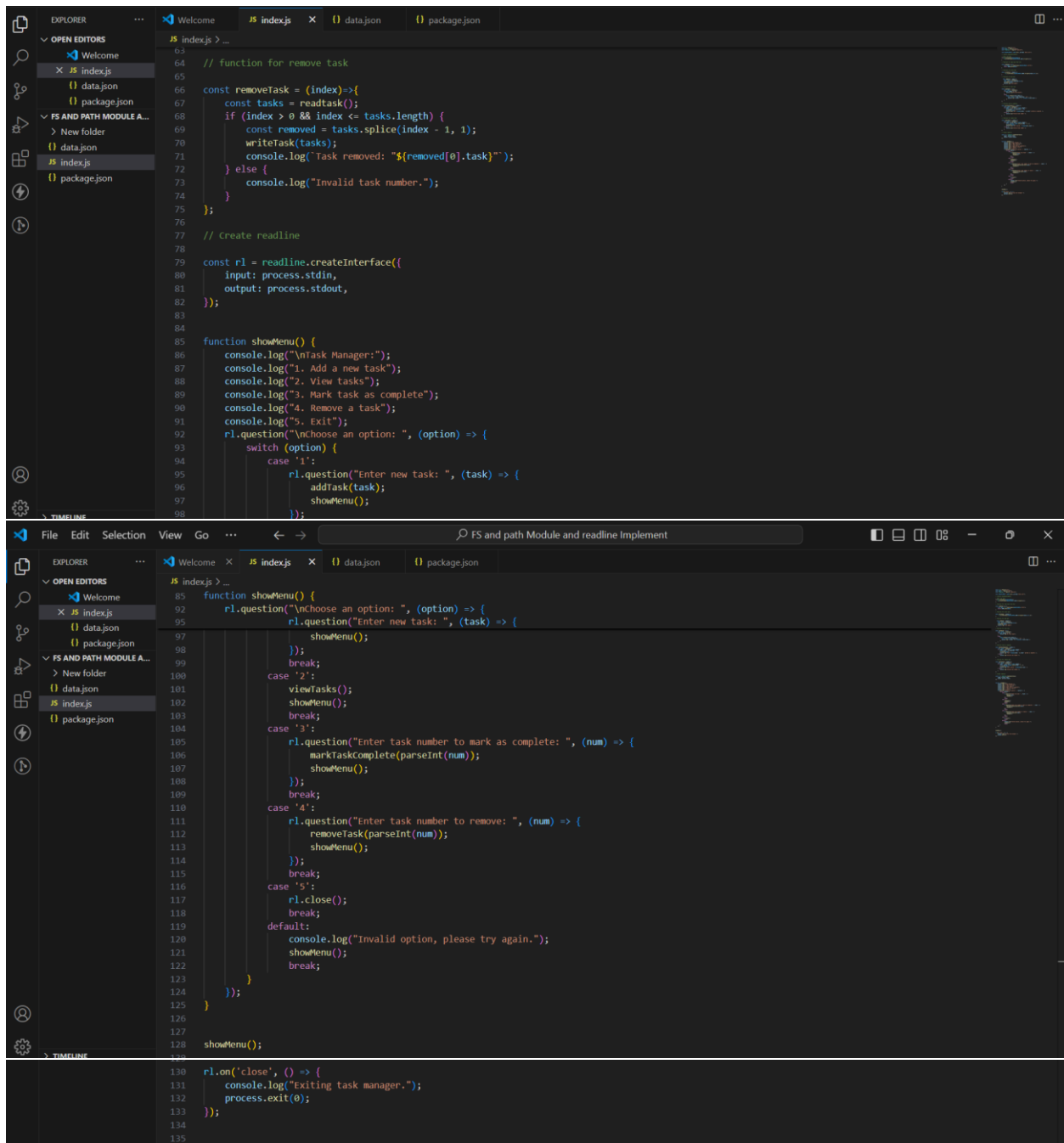
Code :-



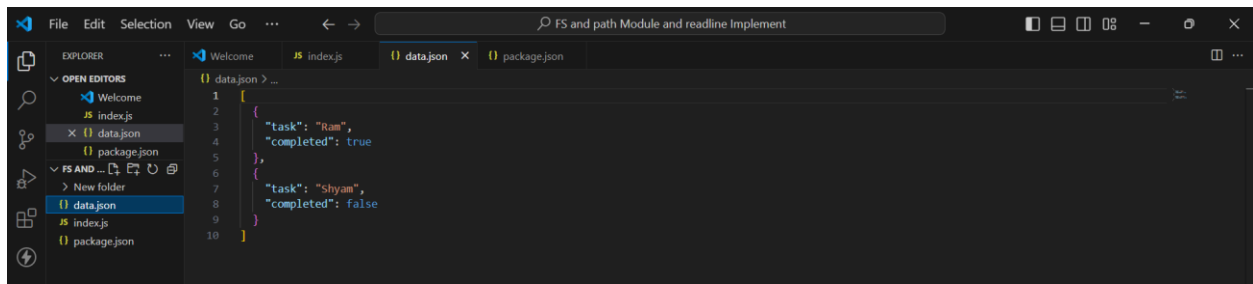
```
1  const fs = require("fs");
2  const path = require("path");
3  const readline = require("readline");
4
5  const dataFilePath = path.join(__dirname, "data.json");
6
7  // console.log(dataFilePath);
8
9  // Add a new task.
10 if(!fs.existsSync(dataFilePath)){
11   fs.writeFileSync(dataFilePath, JSON.stringify([]));
12 }
13
14 // Function for read task from file
15
16 const readtask = ()=>{
17   const data = fs.readFileSync(dataFilePath, "utf-8");
18   return JSON.parse(data);
19 }
20
21 // Function for write task
22
23 const writetask = (tasks)->{
24   fs.writeFileSync(dataFilePath, JSON.stringify(tasks, null, 2));
25 }
26
27 // Function for add task
28
29 const addTask = (task)->{
30   const tasks = readtask();
31   tasks.push({task, completed:false});
32   writetask(tasks);
33   console.log("task added")
34 }
35
```



```
36 // Function for task view
37
38 const viewTasks = ()=>{
39   const tasks = readtask();
40   if(tasks.length === 0){
41     console.log("No Task Found");
42   }
43   else{
44     tasks.forEach((task, index)->{
45       const status = task.completed ? "ok" : "";
46       console.log(`${index + 1}. ${status} ${task.task}`);
47     })
48   }
49 };
50
51 // Function task as complete
52
53 const markTaskComplete = (index)->{
54   const tasks = readtask();
55   if (index > 0 && index <= tasks.length) {
56     tasks[index - 1].completed = true;
57     writetask(tasks);
58     console.log(`Task "${tasks[index - 1].task}" marked as complete.`);
59   } else {
60     console.log("Invalid task number.");
61   }
62 }
63
```



Data.JSON



The screenshot shows a code editor with a dark theme. The Explorer sidebar on the left displays a file structure with 'data.json' selected. The main editor area shows the content of 'data.json', which is a JSON array with two objects. The first object has 'task' as 'Ram' and 'completed' as true. The second object has 'task' as 'Shyam' and 'completed' as false. The code is syntax-highlighted, and line numbers 1 through 10 are visible on the left side of the editor.

```
1  [
2  {
3    "task": "Ram",
4    "completed": true
5  },
6  {
7    "task": "Shyam",
8    "completed": false
9  }
10 ]
```